GeNIe Modeler

User's Manual

Version 4.1.R2, Built on 4/27/2024
BayesFusion, LLC

This page is intentionally left blank.

This page is intentionally left blank.

Read me first

1 Read me first

Welcome to GeNIe manual, Version 4.1.R2, Built on 4/27/2024.

This manual is available in CHM, PDF and HTML formats, all available at https://support.bayesfusion.com/docs/.

CHM version of GeNIe manual is also distributed with GeNIe.

If you are new to GeNIe and would like to start with an informal, tutorial-like introduction, please start with the *Hello GeNIe!* section. If you are an advanced user, please browse through the *Table of Contents* or search for the topic of your interest. We recommend the chapter *Using GeNIe* for tutorial-like introductions to various functional modules of GeNIe.

Hello GeNIe!

2 Hello GeNIe!

This section offers an informal introduction to GeNIe, similar to the light introduction to the C programming language offered by Brian Kerninghan and Dennis Ritchie in their milestone book (see Kernighan & Ritchie, 1988). We will show you how to create a simple Bayesian network model, how to save and load it, and how to perform Bayesian inference with it. Bayesian networks are just one of several powerful modeling tools implemented in GeNIe. Once you have made yourself familiar with GeNIe in this informal way, you can proceed with the *Using GeNIe* chapter, which offers a thorough introduction to various functional modules of GeNIe.

We will sketch the basic functionality of GeNIe on a simple example. While this example contains only two variables, it illustrates all basic concepts, which once understood can be used in building more complex models. Please keep in mind that the functionality covered in this section merely touches one functionality of GeNIe, Bayesian networks. It just gives you a taste of Bayesian modeling.

Consider the following scenario:

Imagine a venture capitalist who considers a risky investment in a start-up company. A major source of uncertainty about her investment is the success of the company. She is aware of the fact that only around 20% of all start-up companies succeed. She can reduce this uncertainty somewhat by asking expert opinion. Her expert, however, is not perfect in his forecasts. Of all start-up companies that eventually succeed, he judges about 40% to be good prospects, 40% to be moderate prospects, and 20% to be poor prospects. Of all start-up companies that eventually fail, he judges about 10% to be good prospects, 30% to be moderate prospects, and 60% to be poor prospects.

How can our investor use the information from the expert? What is the chance for success if the expert judges the prospects for success to be good? What if he judges them to be poor?

We will create a Bayesian network that will allow us to determine the exact numerical implications of the expert's opinion on the investor's expectation of success of the venture. The Bayesian network will contain two nodes representing random variables: *Success of the venture* and *Expert forecast*.

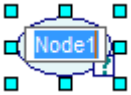
If you have not already started GeNIe, start it now.

The *Tools Menu* shows a list of different types of nodes that you can create. These are also displayed as buttons on the *Standard Toolbar*.

A. Let us create the node for the variable *Success of the venture*.

Click *Chance* button () *from the Standard Toolbar or Tools Menu*.

The *Chance* button will become recessed and the cursor will change to an arrow with an ellipse in bottom right corner. Move the mouse to a clear portion of the screen inside GeNIe window (the main model window is called the *Graph View*) and click the left mouse button. You will see a new node appearing on the screen as shown below:



The small squares around the node indicate that the node is selected. The most recently created node is automatically selected. You can also select any node by clicking on it. You can change the size of the selected node by dragging the small squares.

If you want to draw multiple nodes of the same type, then you can avoid having to select the node button again and again by double-clicking on the corresponding node button instead of single-clicking it. This will place you in "sticky mode," in which the tool button stays recessed and you can draw multiple nodes of that type. You can return to normal mode by clicking on the *Select Objects* button ( ) or clicking on the recessed button again.

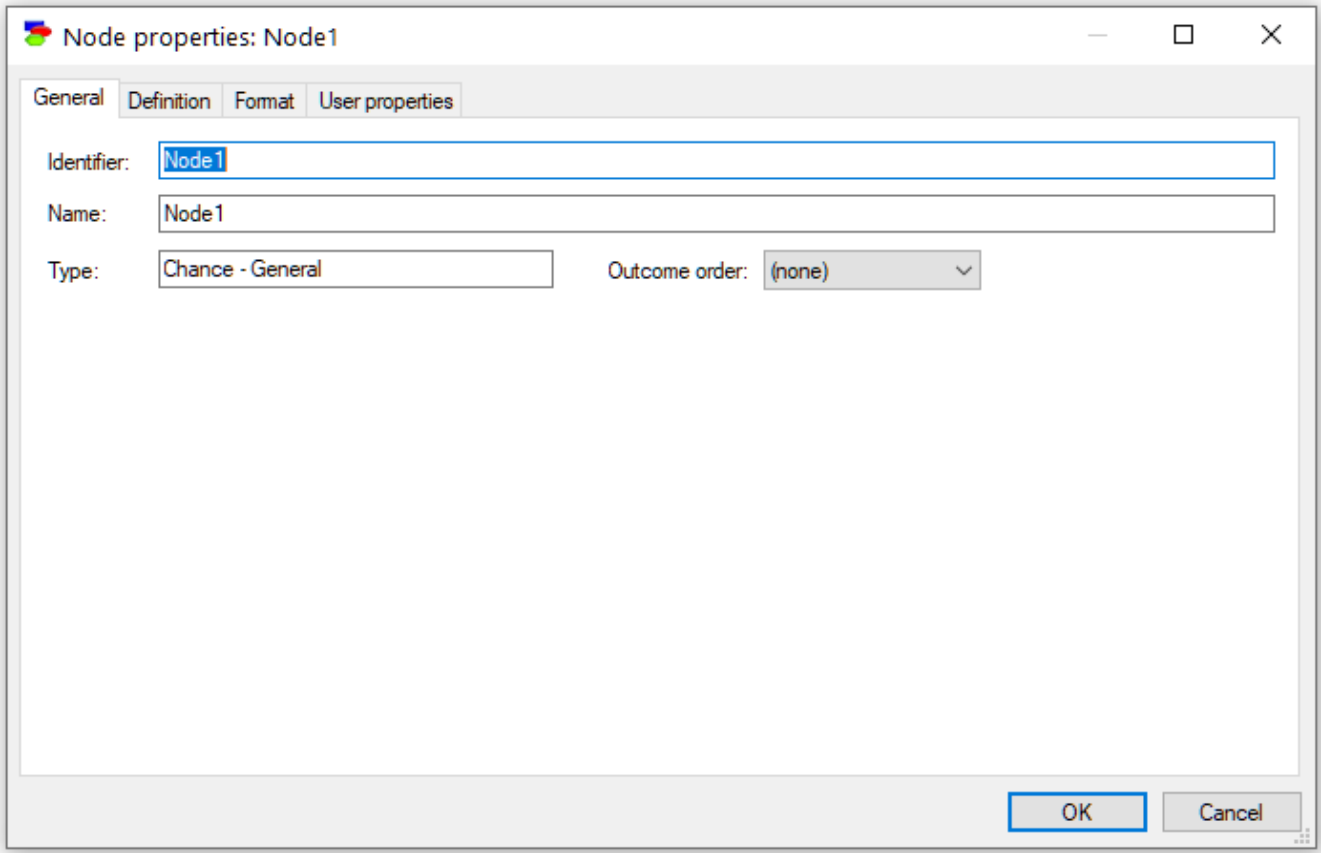
Once you have drawn the node on the *Graph View*, the *Chance* button on the toolbar will become normal again and the *Select Objects* button will become recessed.

GeNIe associates two labels with each node: an identifier and a name. Identifiers are similar to variable names in programming languages: they should start with a letter followed by any sequence of letters, digits or underscore characters. Letters are a-z and A-Z but also all Unicode characters above codepoint 127, which allows using characters from other alphabets than the Latin alphabet. Names are simply strings of characters with no limitations. GeNIe assigned the node that you have just created both the identifier and the name *Node1*. GeNIe also places a newly created node's name in *Edit* mode immediately, so you can enter a more descriptive name if you want.

B. Let us assign a meaningful identifier and a name for the newly created node.

Double click on the node *Node1*.

GeNIe will display the following dialog box:



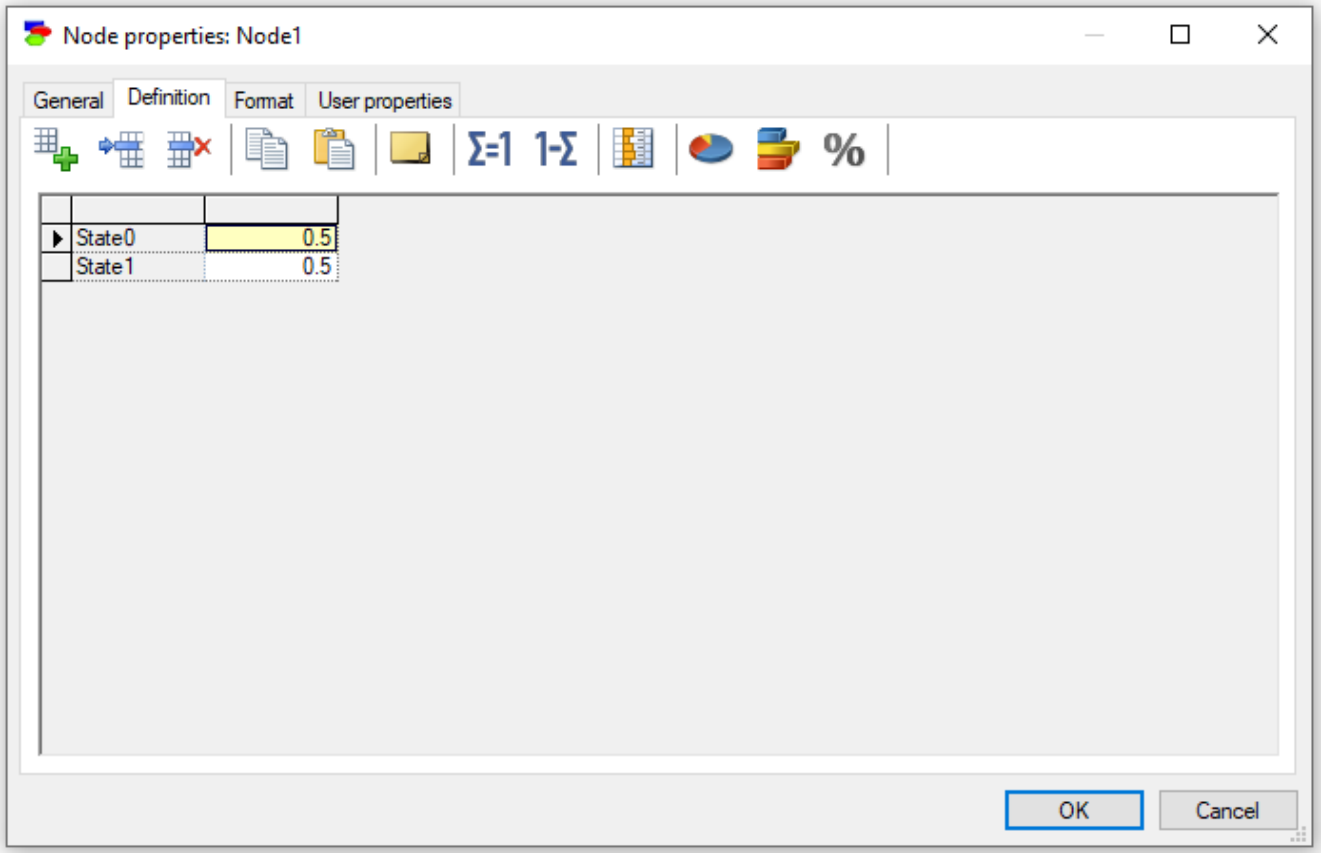
These are the *Node Property Sheets*, which are used to specify various properties of the node.

Now, change the identifier to *Success* and the name to *Success of the venture*.

C. We will define the outcomes of this variable (node) and their probabilities. We can do this from the *Definition* tab of the *Node Property* sheets.

1. Click on the *Definition* tab

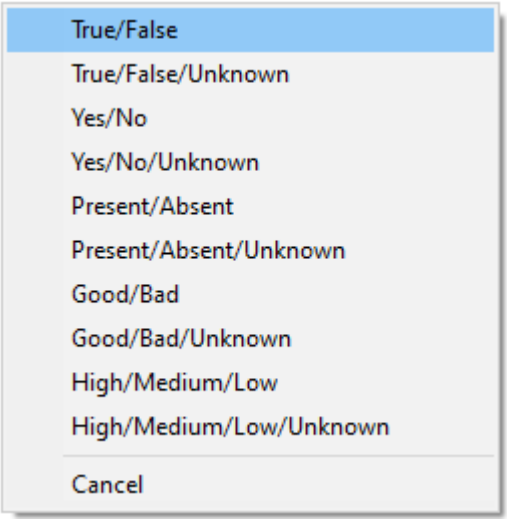
GeNIe will display the following dialog:



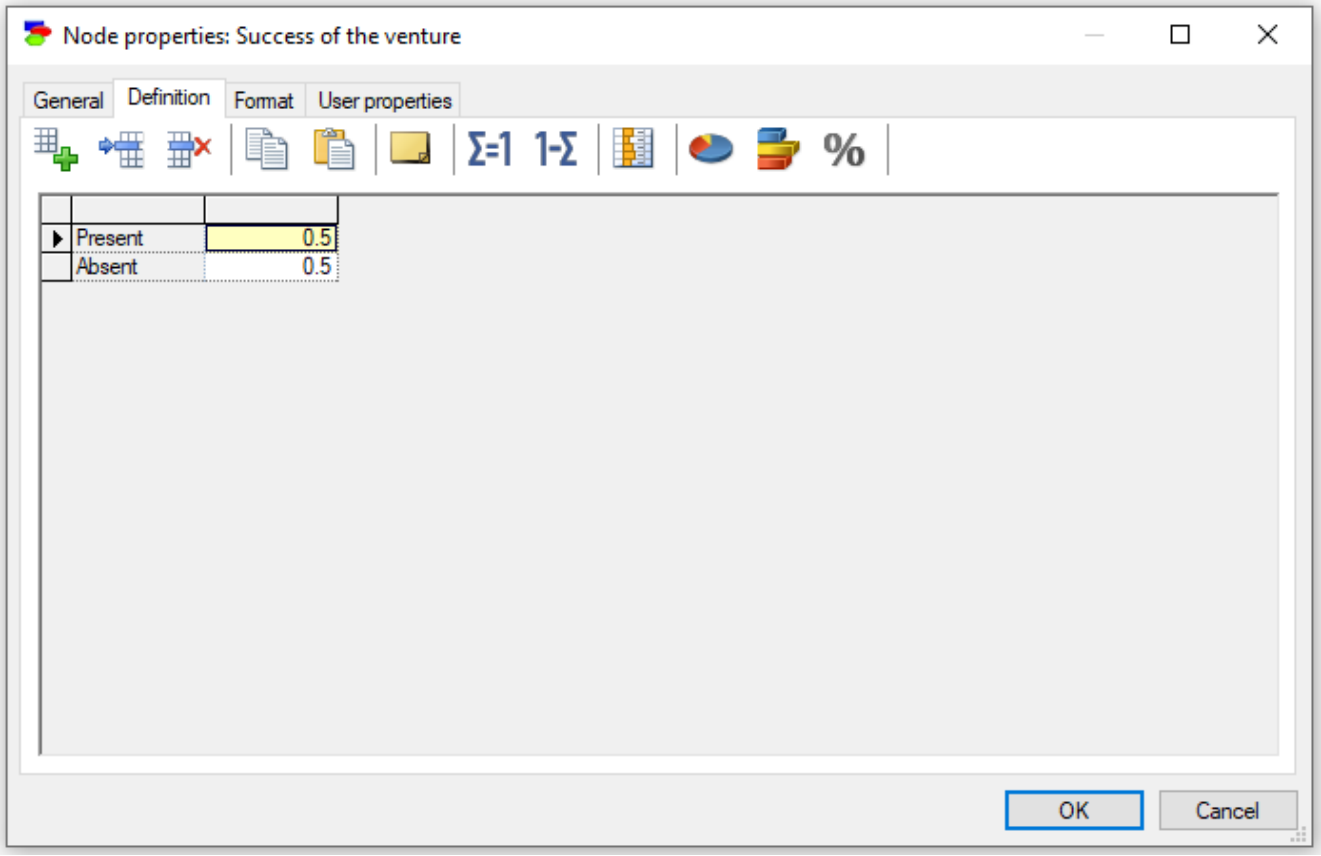
2. Double-click on the identifier of the first state, and change its name to *Success*.

3. Similarly rename *State1* into *Failure*.

You may also select the states of the node from a *Quick States* list at the time of creating a new chance node in the *Graph View*. To do so, select (*Chance* button) from the *Standard Toolbar* or *Tools Menu*. The *Chance* button will become recessed and the cursor will change to an arrow with an ellipse in bottom right corner. Move the mouse to a clear portion of the screen inside GeNIe window, hold the *SHIFT* key on the keyboard and click the left mouse button. The following menu with Quick states list for the nodes will pop up:



Select the states that are relevant to your node, for example if you want the chance node to have states *Present* & *Absent*, select this option from the menu and click the left mouse button. You will now see a new node on the screen. If you double click the node button and select the *Definition* tab. The following dialog box pops up:



If the required states of the node are not listed in the pop up menu, please select *Cancel* option from the pop-up menu and then just left-click anywhere in the *Graph View*. This will create a binary node with default states *State0* and *State1*. You can change the names of these states to *Success* and *Failure* interactively.

Now let us enter the probabilities of occurrence of each of the states. In building GeNIe, we followed the design rule that any model, even in the middle of its construction, is always syntactically correct. Whatever you create interactively in GeNIe, may be gibberish but it will always compose a mathematically correct model. In this spirit, GeNIe creates a new node with an explicitly defined probability distribution, which we chose to be the uniform distribution. For a node with two states, GeNIe sets the probabilities to 0.5 and 0.5, i.e., both states are equally likely.

We want to set P(*Success*) to 0.2 and P(*Failure*) to 0.8. This expresses our knowledge that other information absent, the chance that this business will succeed is 20% (on the average, 80% of start-up businesses fail). To change the current 0.5 and 0.5 probabilities, do the following:

4. Double click on the value field for the *Success* state and enter 0.2.

5. Select the probability for the Failure state (currently 0.5).

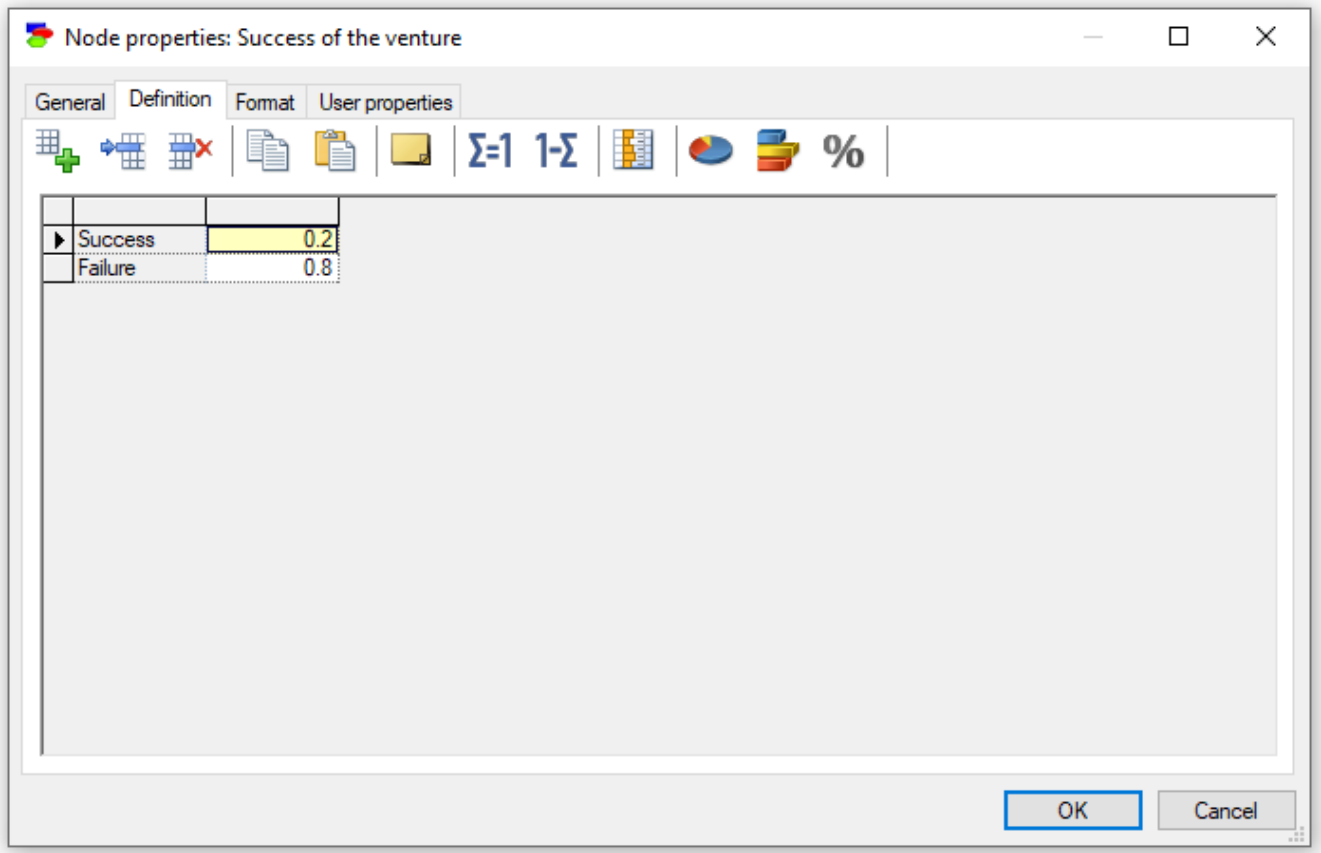
6. Click on the *Complement* (1-Σ) button. This will set the value field to 0.8, which is 1-0.2, so that the sum of probabilities is equal to 1.0.

The *Complement* button simply subtracts the sum of the probabilities in the same column from 1 and adds the remainder to the selected field.

If the sum of probabilities for all the states does not add up to 1.0, you can select the distribution in question and press the *Normalize* (Σ=1). This is a convenient function if you prefer to enter probabilities as percentages. Entering 20 and 80 for the states *Success* and *Failure* respectively and pressing the *Normalize* button will change them to 0.2 and 0.8.

Probabilities can also be entered graphically using a probability wheel or a bar chart. We will cover these later on but if you want, you can explore these now.

After you are done, the tab should look as follows:



We are done defining the properties for this node.

7. Press the *OK* button to return to the *Graph View*.

D. Now let us create the node for the variable *Expert forecast*.

Click on the *Chance* button and then place below the previously created node in the *Graph View*.

Your *Graph View* will look similar to this:

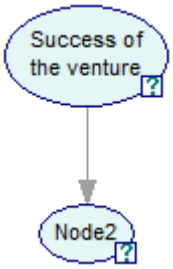
The label for the "Succ..." node is not completely displayed because we have changed the text inside the properties tab but not adjusted the node size. You can adjust the size of the node to fit the text by selecting it and dragging one of the small squares. You can also select multiple nodes and drag on one of them, which will result in all nodes being change at the same time. GeNIe offers also a semi-automatic way of adjusting node size. To do so, right click on the node and select *Resize to Fit Text* from the node pop-up menu. The node will become larger and will display the entire "*Success of the venture*" label.

The newly created node will represent the expert's prediction.

E. In order to represent the fact that the expert's prediction depends on the actual prospects for success, we will create an influence arc between the two nodes.

Click on the *Arc* ( ) tool (note that the cursor changes), then click on the *Success of the venture* node, hold the left mouse button and drag the mouse to the new node (*Node2*), and release the button anywhere within the new node.

GeNIe will draw an arc from *Success of the venture* node to *Node2*. The diagram will now look as follows:



The arc between the two nodes means that whether or not the venture is going to be successful makes a difference for the probability distribution over various statements made by the expert (this is going to be expressed by the conditional probability distributions over *Node2*). After we first draw an arc, it is typically dimmed. This is a feedback from GeNIe stating that the arc is superfluous and does not make any difference for the dependence of the two nodes. It is because we have not yet defined the numerical influence of the parent on the child. We will fix it in Step F.

F. Now, let us define the properties of the new node.

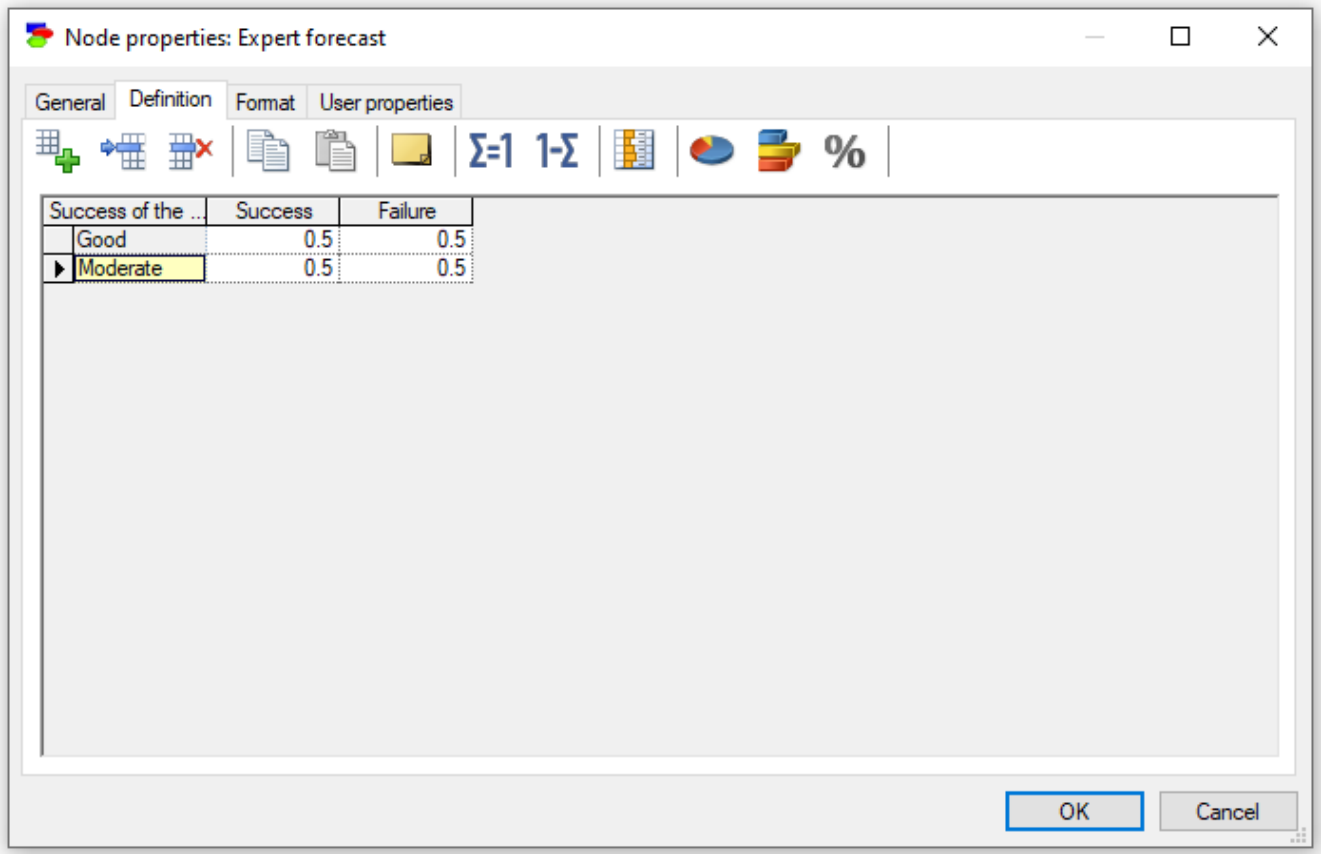
1. Double-click on *Node2*.

2. Change its *Identifier* and *Name* to *Forecast* and *Expert forecast*, respectively.

3. Click on the *Definition* tab.

4. Rename the two states (*State0* & *State1*) to *Good* and *Moderate*.

The screen should look as follows:

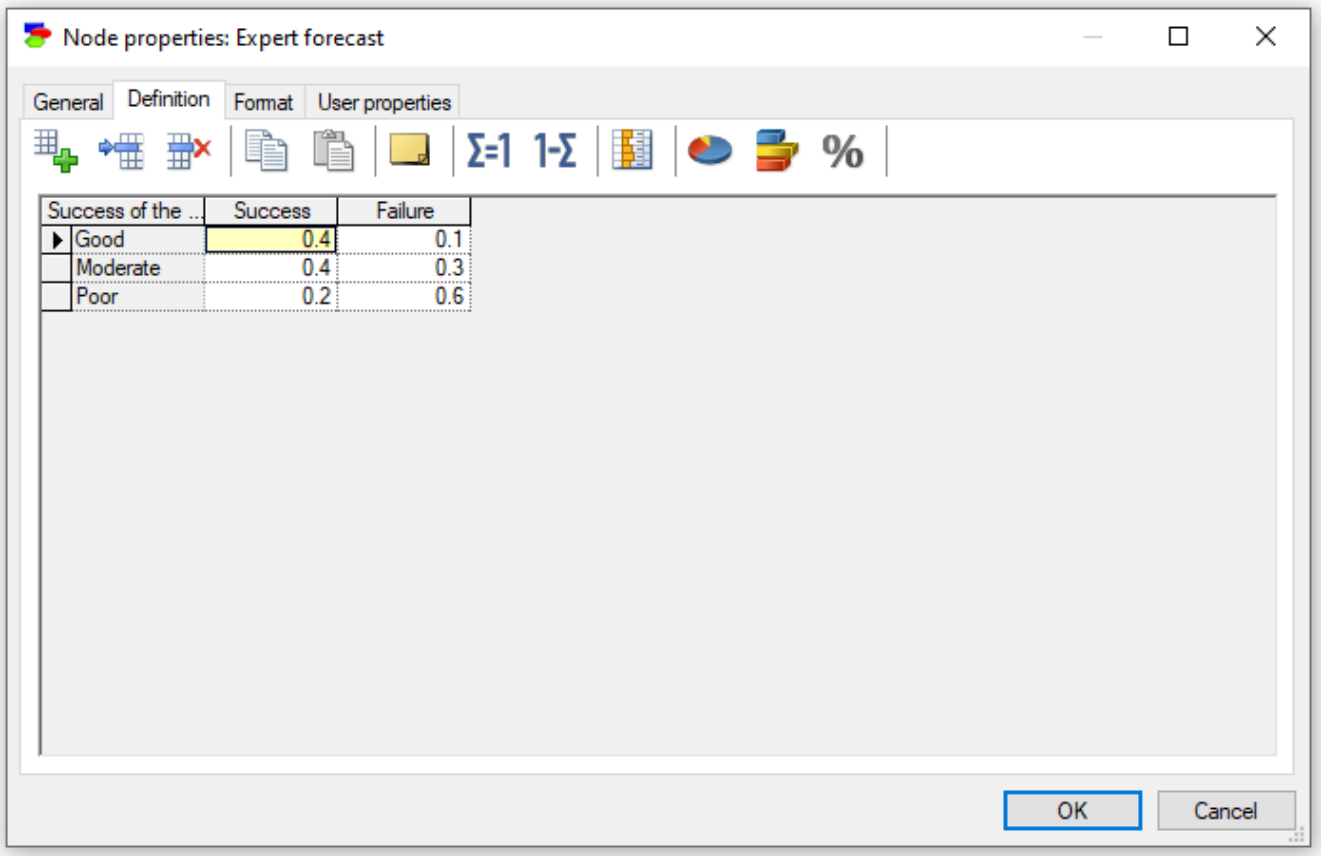


But the expert's forecast can have three possible values: *Good*, *Moderate*, and *Poor*. We have defined only two states, *Good* and *Moderate*. To define the state *Poor*, we need to add one more state.

5. Click on the *Add Outcome* ( ) button. This will add a new state named *State2* below the state *Moderate*.

6. Rename the newly added state to *Poor*.

7. Now you can enter the probabilities for each state combination. We suggest that you use the values shown below:



The probability table above encodes the conditional probabilities of different expert forecasts for all possible actual prospects of the investment. (In general, a node with parents will encode the conditional probability distributions over the node for all possible combinations of outcomes of these parents.) For example, the first column encodes our knowledge that if the prospects are good (the venture is going to succeed), the expert will designate it as *Good* with chance 0.4 (40%), as *Moderate* with chance 0.4 (40%), and as *Bad* with a chance 0.2 (20%). Similarly, the second column encodes our knowledge that the expert will designate an eventually failing venture as *Good*, *Moderate*, and *Poor* 10%, 30%, and 60% of the time.

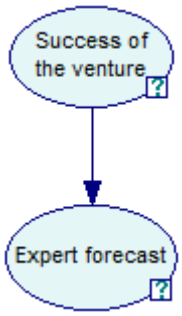
8. Click *OK* to accept all changes and return to the *Graph View*.

You may want to resize the "*Exper...*" node so that the entire name of the node is visible.

9. Right-click on the node and select *Resize to Fit Text* from the menu. The node will become larger and will display the *Expert Forecast* label.

If you want to align the two nodes to make the graph look neater, select both the nodes and click on the *Align Centers* ( ) button on the *Format Toolbar*.

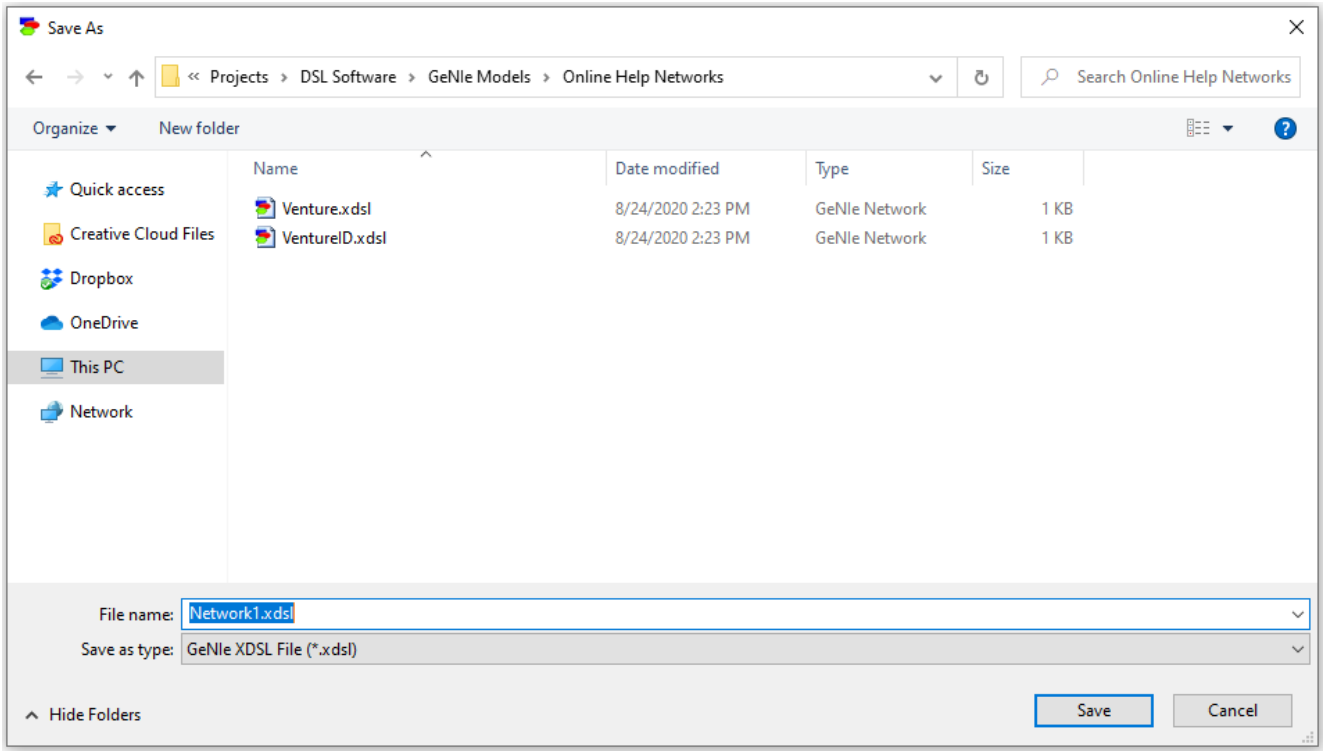
Your network should look like this:



G. At this point you should save your work.

1. Click on *Save* button ( ) on the *Standard Toolbar*.

GeNIe will display the *Save As..* dialog shown below:

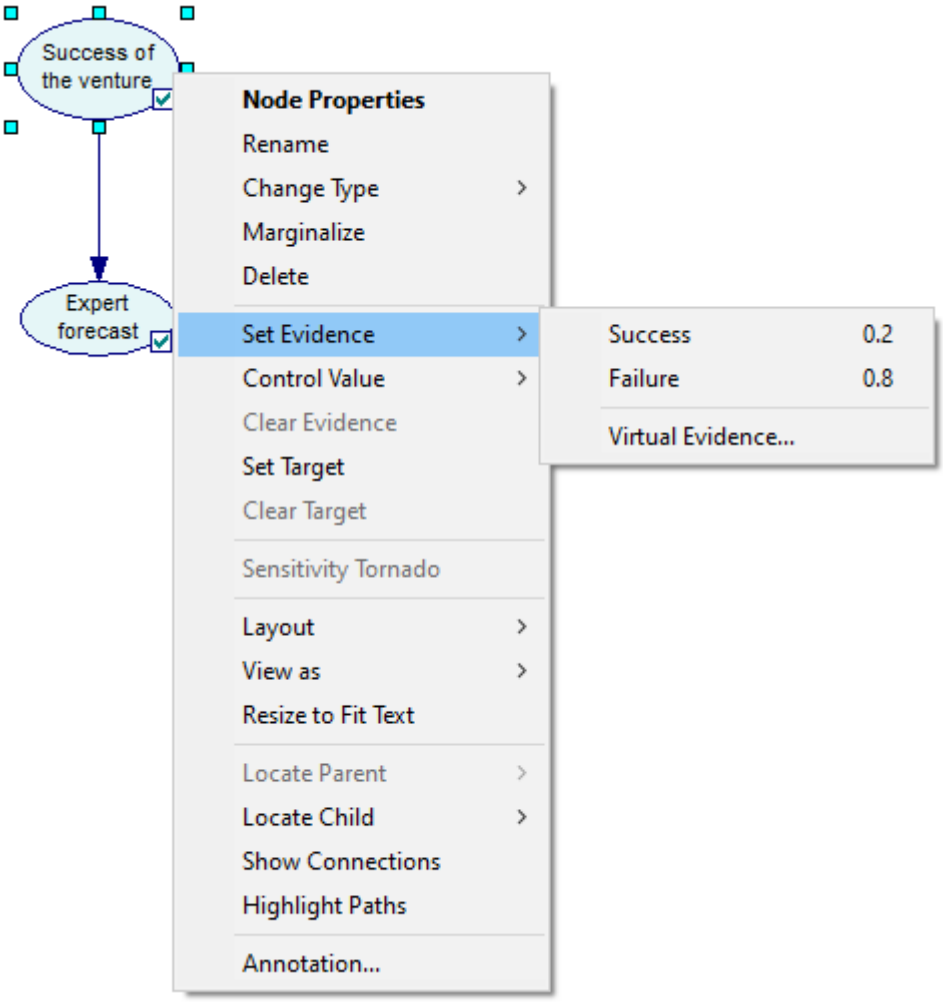


2. Enter *VentureBN* as the *File name* and click on *Save*.

H. Now let us put our model to work and answer the questions posed in the beginning of this tutorial.

To answer the question *What is the chance for success if the expert judges the prospects for success to be good?*, you will need to tell GeNIe that you have observed a value of the *Forecast* variable to be *Good* and ask it to update its probability distribution over the variable *Success*. There are several ways of doing this.

1. Right-click on the variable *Expert forecast* and choose *Set evidence* / *Good*.



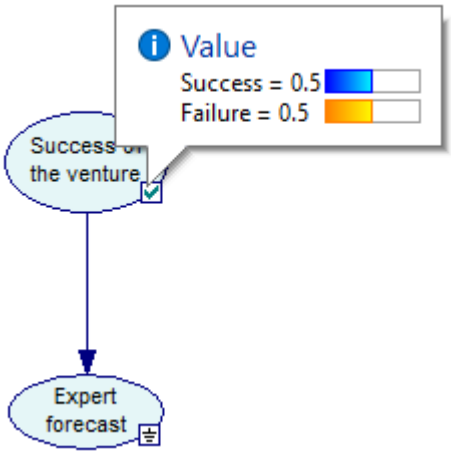
Note that the status icon on the bottom right of the node changes from ⍰ to ⊟. This indicates that the node has been observed.

2. Click on the *Update* tool ( ) on the *Standard Toolbar*.

This updates the probability distributions in light of the observed evidence. Notice that the status icon for the *Success of the venture* node changes to ☑ from ⍰.

3. Move the mouse cursor over the ☑ for the *Success of the venture* node.

This will display the posterior probability distribution over the *Success of the venture* node:



We see that expert's forecast *Good* has changed the probability of *Success* of the venture from 0.2 to 0.5.

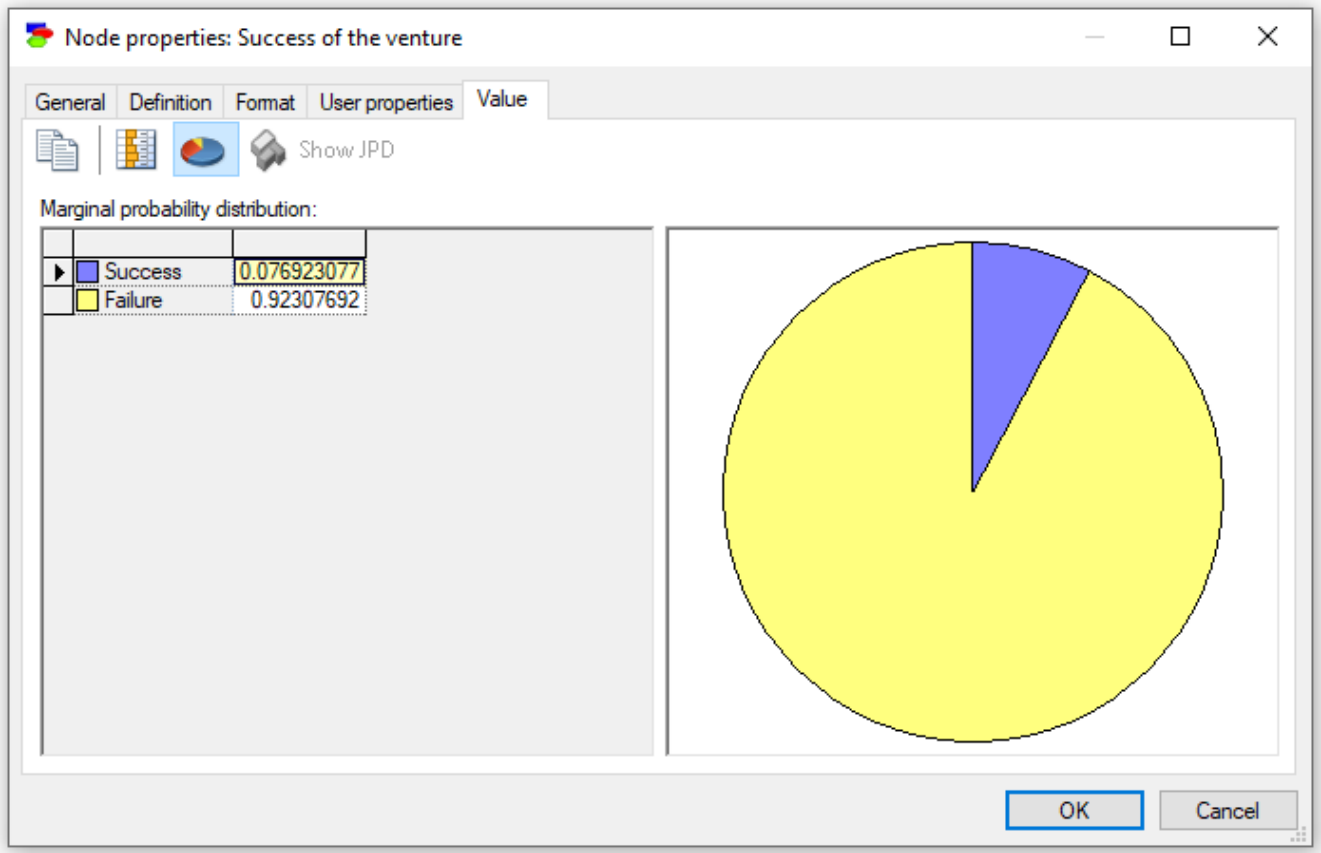
To answer the second question *What if he judges them to be poor?*, we will set the evidence in the node *Expert forecast* to *Poor*, update the model, and observe that the probability of success is now less than 0.08.

Results of Bayesian updating can be also viewed by double-clicking on a node and selecting the *Value* tab.

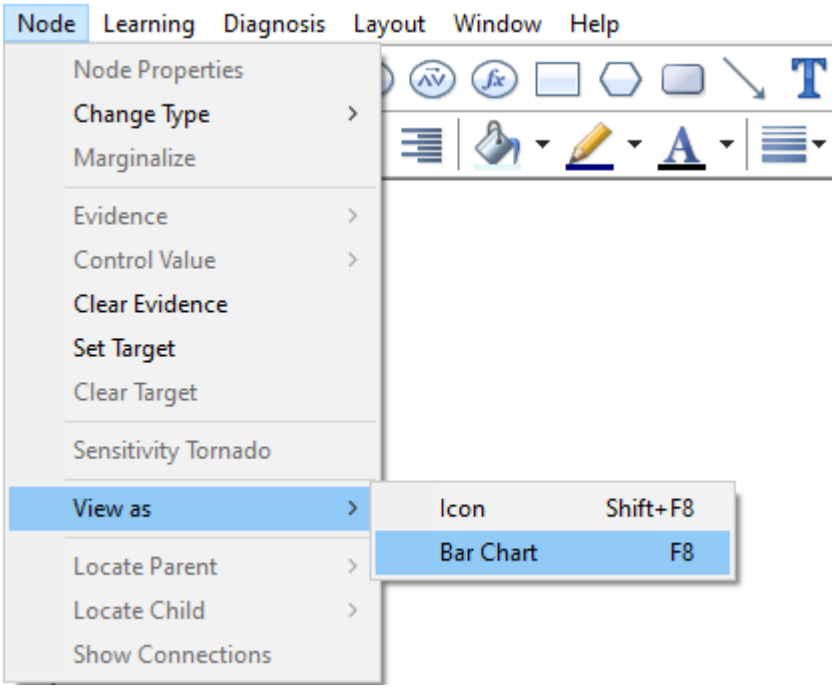
4. Double click on the *Success of venture* node.

5. Select *Value* tab from the *Node Property* sheets

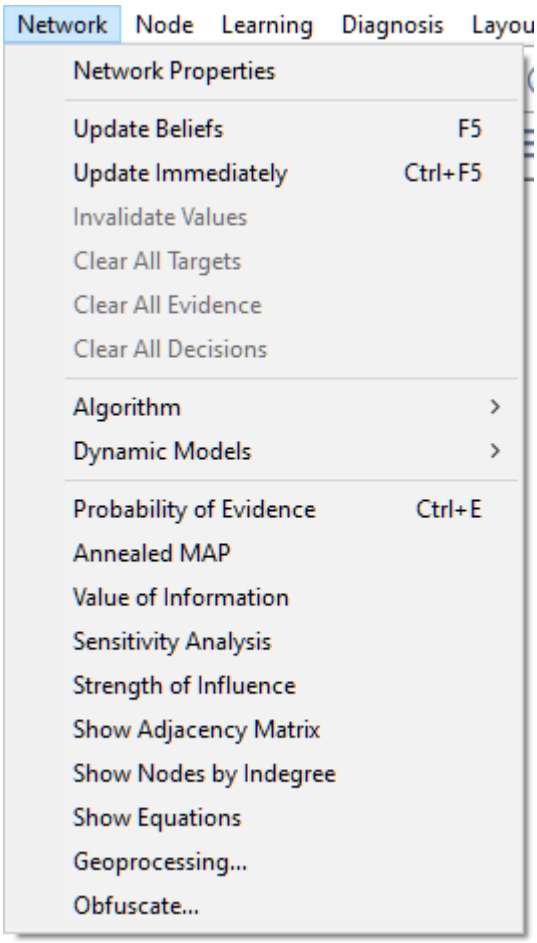
The result should look as follows:



Entering evidence and displaying results can be done much simpler by changing node appearance from *Icon* to *Bar Chart*. To achieve this, first select all nodes that you want to display as bar charts (you can select them with a mouse or, if you want to change all nodes, you can select all nodes by pressing *CTRL-A*).

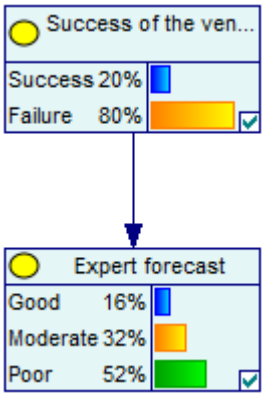


Changing this setting for all nodes can be also achieved by deselecting all nodes, in which case the *View as* / *Bar Chart* command will apply to all nodes. Also, set the *Update Immediately* flag:

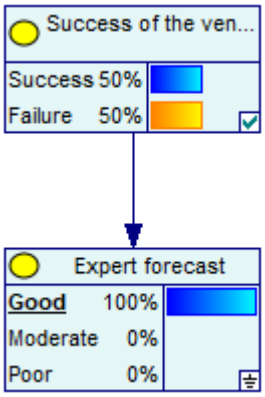


The *Update Immediately* flag, when set, invokes Bayesian inference as soon as any change happens to the model. It is convenient to have it turned off when we are still busy with building a model (to avoid computation when the model is incomplete and it does not make much sense yet) and turned on when the model is ready.

The network should look as follows:



To observe a value, double-click on the bar that corresponds to it. To answer the first question, we double click on the state *Good* in the *Expert forecast* node. We observe the following:



The evidence entered is marked by underlining the state and showing the bar to be 100%. The posterior marginal probability distribution and, hence, the answer to our question, is shown in the node *Success of the venture*.

What we created was a simple Bayesian network. You can create more complex models in a similar way.

You can find the above model named *VentureBN.xdsl* in the *Example Networks* directory among other example models that come with GeNIe.

Introduction

3 Introduction

3.1 Guide to GeNIe manual

The best user interface to a computer program is one for which there is no need for a manual. This has been from the very start our design motto and, from the compliments of our users, we learn that we have (almost) reached our objective, often at the expense of a considerable development effort. Most of the tasks in GeNIe are intuitive and conform to the general standards of well designed user interfaces. If you do not know how to do something, just try what you would find most intuitive and and what you would do in a well designed modern user interface. There is a good chance that this will work for you.

This manual may turn out to be superfluous for many users. Still, there are many ways in which people learn and some users will find it useful. Some may find it handy to use the search functionality in case they encounter a problem. Others, familiar with the methodology, may want to read through the section *Using GeNIe*, which describes various GeNIe modules. Yet others, may want to the treat this manual as a handbook for decision-theoretic methods.

To introduce the basics of decision-theoretic modeling is not as easy as it is to write a brief document, prepare a slide show, or draw a picture. While there are many good books available that cover decision-theoretic modeling, they typically use theoretical concepts and draw models symbolically on paper. Building them in software is different. Besides, some of GeNIe functionality is novel and not described or otherwise covered anywhere. This is the main reason why even GeNIe needs an easily accessible manual, tutorial-style introductions, and on-line help.

GeNIe manual, the introduction of which you are reading at the moment, is available for a variety of platforms: HTML, compiled HTML (CHM), PDF, etc., all available from https://support.bayesfusion.com/docs/.

3.2 GeNIe Modeler

GeNIe Modeler is a development environment for building graphical decision-theoretic models. It was created and developed at the Decision Systems Laboratory, University of Pittsburgh between 1995 and 2015. Its name and its uncommon capitalization originates from the name Graphical Network Interface, given to the original simple user interface to SMILE, our library of classes for graphical probabilistic and decision-theoretic models.

GeNIe and SMILE have been originally developed to be major teaching and research tools in academic environments and have been used at thousands of universities world-wide. Most research conducted at the Decision Systems Laboratory, University of Pittsburgh, found its way into both programs. Because of their versatility and reliability, GeNIe and SMILE have become very popular and became de-facto standards in academia, while being embraced by a number of government, military, and commercial users. In 2015, we created a company, BayesFusion, LLC, and acquired a license for GeNIe from the University of Pittsburgh. Since that time, all further development of GeNIe, as well as its extended family of products for decision-theoretic modeling, happens at BayesFusion, LLC. Continuing the tradition of the Decision Systems Laboratory, we are making it available free of charge to the academic community for research and teaching use in order to promote decision-theoretic methods in decision support systems. GeNIe has been tested extensively in many teaching, research, and commercial environments. We are continuously improving it and are interested in user comments. We encourage the users of GeNIe to let us know about encountered problems and possible suggestions.

GeNIe has been written for the Windows operating systems. While we cannot guarantee 100% compatibility, we are verifying with each build that it runs on macOS (formerly OS X) and Linux under Wine. Please see *GeNIe on a Mac* or *GeNIe on Linux* sections for more information. GeNIe allows for building models of any size and complexity, limited only by the capacity of the operating memory of your computer.

GeNIe is a modeling environment. Models developed using GeNIe can be embedded into any applications and run on any computing platform, through GeNIe's Application Programmer's Interface (API) SMILE, which is fully portable. Our customers have used SMILE on a variety of platforms, including cloud/serverless, embedded avionics, and computing clusters.

While GeNIe offers unprecedented functionality, its strongest element, one that distinguishes it from a large number of other graphical modeling tools, is its user interface. We have paid a lot of attention to it and it shows. While developing decision-theoretic models takes typically an enormous amount of time, GeNIe cuts the effort by at least an order of magnitude and it will lead to a fast return of the investment in its licensing fees. SMILE, GeNIe's API, is not far behind and belongs to the easiest to learn and use, most reliable, and fastest libraries for graphical models.

3.3 QGeNIe

QGeNIe is a qualitative abstraction of GeNIe and is an interactive development environment for rapid creation of qualitative causal models of uncertain domains. These models represent propositions by means of nodes in an acyclic directed graph. These nodes are always propositional and take two possible values: *True* and *False*. The colors of these nodes represent the degrees of truth of the propositions. Mathematically speaking, the colors represent the probability of the state *True* (or *False* - it is the users' choice). While QGeNIe users can define the color scale, the default is a range between red and green, representing undesirable and desirable states. QGeNIe allows for an interactive exploration of the models, examining the effects of observations and manipulations of individual variables.

There are two important applications of QGeNIe:

1. It is a standalone system for rapid creation of simplified causal models, useful in all kinds of strategic planning problems, where problems are complex enough to be a challenge for an unaided human mind and, at the same time too complex to model by means of fully specified, precise quantitative models. QGeNIe captures the knowledge and intuitions of decision makers and focuses group discussion on calculating the global effects of various decision options, which is for sufficiently complex problems a challenge for an unaided human mind. Working with QGeNIe helps with uncovering indirect pathways through which actions may propagate through the system, often with surprising effects. QGeNIe models applied in this way are typically projected in the meeting room of group meetings, where participants propose different actions and explore their consequences. QGeNIe offers what can be called an instant gratification interface in the sense of showing interactively the effects of observations and manipulations.

2. Models developed by means of QGeNIe are simple first-cut versions of quantitative probabilistic models. They can be exported to GeNIe for further refinement into more precise quantitative models.

3.4 SMILE Engine

SMILE (Structural Modeling, Inference, and Learning Engine) is a fully platform independent library of functions implementing graphical probabilistic and decision-theoretic models, such as Bayesian networks, influence diagrams, dynamic Bayesian networks, and structural equation models. Its individual functions, defined in SMILE Applications Programmer Interface (API), allow to create, edit, save, and load graphical models, and use them for probabilistic reasoning and decision making under uncertainty.

SMILE is implemented in C++ in a platform independent fashion. We also provide Java (jSMILE), .NET (SMILE.NET), .COM (SMILE.COM), Python (PySMILE) and R (rSMILE) wrappers for users who want to use SMILE with languages other than C++. Through the Java wrapper, SMILE can be used in programming environments such as Matlab or Ruby. Through the .NET wrapper, it can be used, among others, from C# and VB.NET. SMILE is equipped with an outer shell, a developer's environment for building graphical decision models, known as GeNIe. GeNIe is platform dependent and runs on Windows computers, although many of our users have successfully used it on MacOS and Linux operating systems. SMILE can be embedded in programs that use graphical probabilistic models as their reasoning engines. Such programs can be distributed to end users or placed on servers for cloud use. Models developed in SMILE can be equipped with a user interface that suits the user of the resulting application most.

GeNIe, QGeNIe and SMILE have been originally developed to be major teaching and research tools in academic environments and have been used at hundreds of universities world-wide. Most research conducted at the Decision Systems Laboratory, University of Pittsburgh, found its way into GeNIe and SMILE. Because of their versatility and reliability, GeNIe, QGeNIe and SMILE have become very popular and became *de facto* standards in academia, while being embraced by many government and commercial users.

The strongest elements of SMILE, those that distinguish it from a large number of other graphical modeling tools, is its ease of use from a programmer's perspective (it offers a modern object-based API), availability for multiple platforms, its reliability (it has been tested heavily in practical research and commercial applications since 1998), and speed (it has shown excellent performance in UAI speed competitions). Speed especially is crucial, as most calculations in probabilistic graphical models are exponential in nature.

3.5 Embedding decision-theoretic methodology into custom software

GeNIe allows for interactive model building and examination. Every action performed in GeNIe is translated to a call to SMILE, GeNIe's API. It is possible to automate these actions and embed decision-theoretic methodology into user software by accessing SMILE directly. We offer a collection of wrappers that allow for accessing SMILE from a variety of programming languages. Calls to SMILE are simple and intuitive and can be used by any programmer.

Here is a Python code sample that illustrates loading a previously saved model (`VentureBN.xdsl`), entering evidence, and obtaining results:

```
import pysmile

def hello_smile():
  net = pysmile.Network()
  net.read_file("VentureBN.xdsl");
  net.set_evidence("Forecast", "Moderate")
  net.update_beliefs()

  beliefs = net.get_node_value("Success")
  for i in range(0, len(beliefs)):
    print(net.get_outcome_id("Success", i) + "=" + str(beliefs[i]))

hello_smile()
```

Java code for the same problem:

```
import smile.*;

public class Hello {
  public static void main(String[] args) {
    Network net = new Network();
    net.readFile("VentureBN.xdsl");
    net.setEvidence("Forecast", "Moderate");
    net.updateBeliefs();
    double[] beliefs = net.getNodeValue("Success");
    for (int i = 0; i < beliefs.length; i ++) {
      System.out.println(
          net.getOutcomeId("Success", i) + " = " + beliefs[i]);
    }
  }
}
```

R code for the same problem:

```
library(rSMILE)

net <- Network()
net$readFile("VentureBN.xdsl")
net$setEvidence("Forecast", "Moderate")
net$updateBeliefs()
beliefs <- net$getNodeValue("Success")

for (i in 1:length(beliefs)) {
  cat(sprintf(
    "%s = %f\n", net$getOutcomeId("Success", i-1L), beliefs[i]))
}
```

C# code for the same problem:

```
using System;
using Smile;

namespace SmileNetTutorial
```

```
{
  class Hello
  {
    static void Main(string[] args)
    {
        Network net = new Network();
        net.ReadFile("VentureBN.xdsl");
        net.SetEvidence("Forecast", "Moderate");
        net.UpdateBeliefs();
        double[] beliefs = net.GetNodeValue("Success");
        for (int i = 0; i < beliefs.Length; i++)
        {
          Console.WriteLine("{0} = {1}",
              net.GetOutcomeId("Success", i),
               beliefs[i]);
          }
      }
  }
}
```

Finally, here is C++ (native to SMILE) code sample for the same problem:

```
#include <cstdio>
#include <smile.h>

int main()
{
    DSL_errorH().RedirectToFile(stdout);
    DSL_network net;
    int res = net.ReadFile("VentureBN.xdsl");
    if (DSL_OKAY != res)
    {
        return res;
    }

    net.GetNode("Forecast")->Val()->SetEvidence("Moderate");
    net.UpdateBeliefs();
    DSL_node* sn = net.GetNode("Success");
    const DSL_Dmatrix& beliefs = *sn->Val()->GetMatrix();
    const DSL_idArray& outcomes = *sn->Def()->GetOutcomeIds();
    for (int i = 0; i < outcomes.GetSize(); i++)
    {
        printf("%s=%g\n", outcomes[i], beliefs[i]);
    }

    return DSL_OKAY;
}
```

3.6 BayesBox

BayesBox is a member of BayesFusion's family of products developed for modeling, learning, and decision support. It facilitates unlimited online sharing of probabilistic graphical models (such as Bayesian networks, influence diagrams, dynamic Bayesian networks, and hybrid Bayesian networks) created for internal or public use. Customers can upload to the repository any number of models organized into any number of categories.

Server part of BayesBox runs on Linux or Windows and is available as a software installable and fully controlled by customers on-premises, or hosted as a service by BayesFusion. Users access BayesBox models through any standards-compliant web browser, can explore the model structure, enter evidence and examine results.

Bayesian inference in the BayesBox server is performed by SMILE Engine, our cross-platform Bayesian software library.

BayesBox can be fine-tuned to user needs and reflect customer brand, including a name, a logo, and a color scheme. Optionally, administrator interface allows for enabling access control through a login page.

GeNIe allows for opening models from an existing BayesBox instance, which makes BayesBox a useful tool for teams working on decision-theoretic methodologies.

BayesFusion's public model repository is powered by BayesBox (see https://repo.bayesfusion.com/). For demonstration purposes, we have also created a BayesBox-based web site of a fictitious company Evidentious, Inc., at https://demo.bayesfusion.com/.

3.7 Distribution information

Hardware and software requirements

Disk space

Full installation of GeNIe requires less than 30 MB of disk space.

Memory

GeNIe has practically no minimum memory requirements and can run under a minimum Windows configuration. The actual memory requirement will depend on the size and complexity of the models that you create. Too little memory may result in decreased performance. In general, conditional probability tables grow exponentially with the number of parents of a node. The maximum number of parents of a node will, therefore, determine memory requirements. In addition, memory requirements of the clustering algorithm grow with the connectivity of the network.

Operating system

GeNIe has been written for the Windows operating systems. Installation of GeNIe under Windows operating systems may require administrator privileges. While we cannot guarantee 100% compatibility, we are verifying with each build that it runs on macOS (formerly OS X) and Linux under Wine. Please see *GeNIe on a Mac* section for more information on running GeNIe on a macOS. Getting GeNIe to run under Linux is virtually identical.

Directory, file, and path naming

GeNIe is compatible with any language version of Windows and any Unicode-based naming of files and directories, so directory path names containing non-Latin characters, such as for example Arabic, Chinese, Cyrillic, Hebrew, Hindi, and special diacritical characters in languages otherwise using the Latin alphabet, are not a problem.

GeNIe version

To determine the version of GeNIe that you have installed, select *About GeNIe* from the *Help Menu*. The version number is listed in the small frame of the following window:

Examples

GeNIe installation creates a directory named *Examples*, containing a sizable library of example models, data sets, and networks used in this manual. The directory has several sub-directories, listed alphabetically below:

Clemen Models: models extracted from the textbook by Robert T. Clemen, *Making Hard Decisions: An Introduction to Decision Analysis* (Clemen, 1996).

Discrete Bayesian Networks: the most popular type of Bayesian network models, those consisting of discrete variables.

Dynamic Bayesian Networks: discrete-time models used for modeling dynamic systems.

Hybrid Bayesian Networks: models consisting of continuous variables specified by means of equations or mixtures of continuous and discrete variables.

Influence Diagrams: influence diagram models.

Learning: data sets and other auxiliary files useful in learning Bayesian networks and causal discovery.

Models from the Manual: a collection of models and other auxiliary files used in this manual.

Qualitative Models: qualitative models that can be opened only with QGeNIe.

3.8 GeNIe on a Mac

GeNIe can be used on a Mac with Boot Camp (available only on Intel-based Macs, boots the computer into Windows installed on a separate partition), a virtual machine (VM) such as VirtualBox or Parallels, or with Wine (a compatibility layer capable of running Windows applications). Boot Camp and VM-based solutions give the user access to a complete Windows operating system, where GeNIe can be installed and run natively. Wine is an open source project. It can be downloaded and used for free, and works on Apple Silicon (ARM-based) Macs.

Note that recent versions of macOS, starting with 10.15 Catalina, released in 2019, do not support 32-bit applications. If you need to use Wine to run 32-bit GeNIe (such as GeNIe Academic) follow the instructions for 32-bit mode below.

Another Wine-based option for both 32-bit and 64-bit GeNIe is Crossover by CodeWeavers. Crossover makes it easier to use Wine and CodeWeavers provides excellent technical support to its users. All purchases of Crossover are used to directly fund the developers working on Wine. Crossover is available at https://www.codeweavers.com/.

Running 64-bit GeNIe on macOS with Wine:

1. Install Wine on your Mac. Follow the instructions at https://wiki.winehq.org/MacOS.

2. Download 64-bit MSI GeNIe installer. Note that the EXE installer for 64-bit GeNIe will not work, because the installer stub is 32-bit.

3. Run Wine from Launchpad. This opens the Terminal window configured to run Windows programs. In Wine's terminal, use the following command to run the MSI installer. The example below assumes that the installer is located in user's *Desktop* folder, and its filename is `genie-setup-4.1.3315-x64.msi`.

   ```
   wine64 msiexec /i ~/Desktop/genie-setup-4.1.3315-x64.msi
   ```

4. In Wine's terminal, use the `cd` command to navigate to GeNIe's installation directory. Assuming that GeNIe was installed in its default location, the command is:

   ```
   cd ".wine/drive_c/Program Files/GeNIe 4.1"
   ```

   Note that quotes are required, because some of the directories' names contain spaces.

5. After changing the directory, run GeNIe:

   ```
   wine64 genie.exe
   ```

   This starts GeNIe using appropriate Wine configuration.

Running 32-bit GeNIe Academic on macOS with Wine

1. Install the wine-crossover package on your Mac using homebrew. The package contains code developed for the commercial Crossover product, but is free to download and use.

   ```
   brew install --cask --no-quarantine gcenx/wine/wine-crossover
   ```

2. Download 32-bit GeNIe installer, either EXE or MSI.

3. Run "Wine Crossover" from Launchpad. This opens the Terminal window configured to run Windows programs. In Wine Crossover's terminal, use the following command to run the MSI installer. The example

below assumes that the installer is located in user's *Desktop* folder, and its filename is `genie-academic-setup-4.1.3315-x64.msi`.

```
wine msiexec /i ~/Desktop/genie-academic-setup-4.1.3315-x64.msi
```

4. In Wine's terminal, use the `cd` command to navigate to GeNIe's installation directory. Assuming that GeNIe was installed in its default location, the command is:

   ```
   cd ".wine/drive_c/Program Files/GeNIe Academic 4.1"
   ```

   Note that quotes are required, because some of the directories' names contain spaces.

5. After changing the directory, run GeNIe:

   ```
   wine genie.exe
   ```

   This starts GeNIe using appropriate Wine configuration.

3.9 GeNIe on Linux

GeNIe can be used on a Linux system with Virtual Machine (VM)-based solution like VirtualBox, or with Wine (a compatibility layer capable of running Windows applications). Wine is an open source project. It can be downloaded and used for free.

Running GeNIe on Linux with Wine:

1. Install Wine on your Linux system. On some Linux systems, including Ubuntu, Wine can be installed through package manager. On Ubuntu 22, the command to install Wine is:

   ```
   sudo apt install wine
   ```

2. If Wine is not available in the package manager, it can be downloaded from https://wiki.winehq.org/Download.

3. Download GeNIe installer.

4. In Linux terminal, use the `wine` command to run GeNIe installer. The example below assumes that GeNIe installer is located in the current directory and its filename is `genie-setup-4.0.2304-x64.exe`.

   ```
   wine genie-setup-4.0.2304-x64.exe
   ```

5. GeNIe (and QGeNIe) icons should be added to Linux application launcher. If the icons are available, double-click on the icon to run the program.

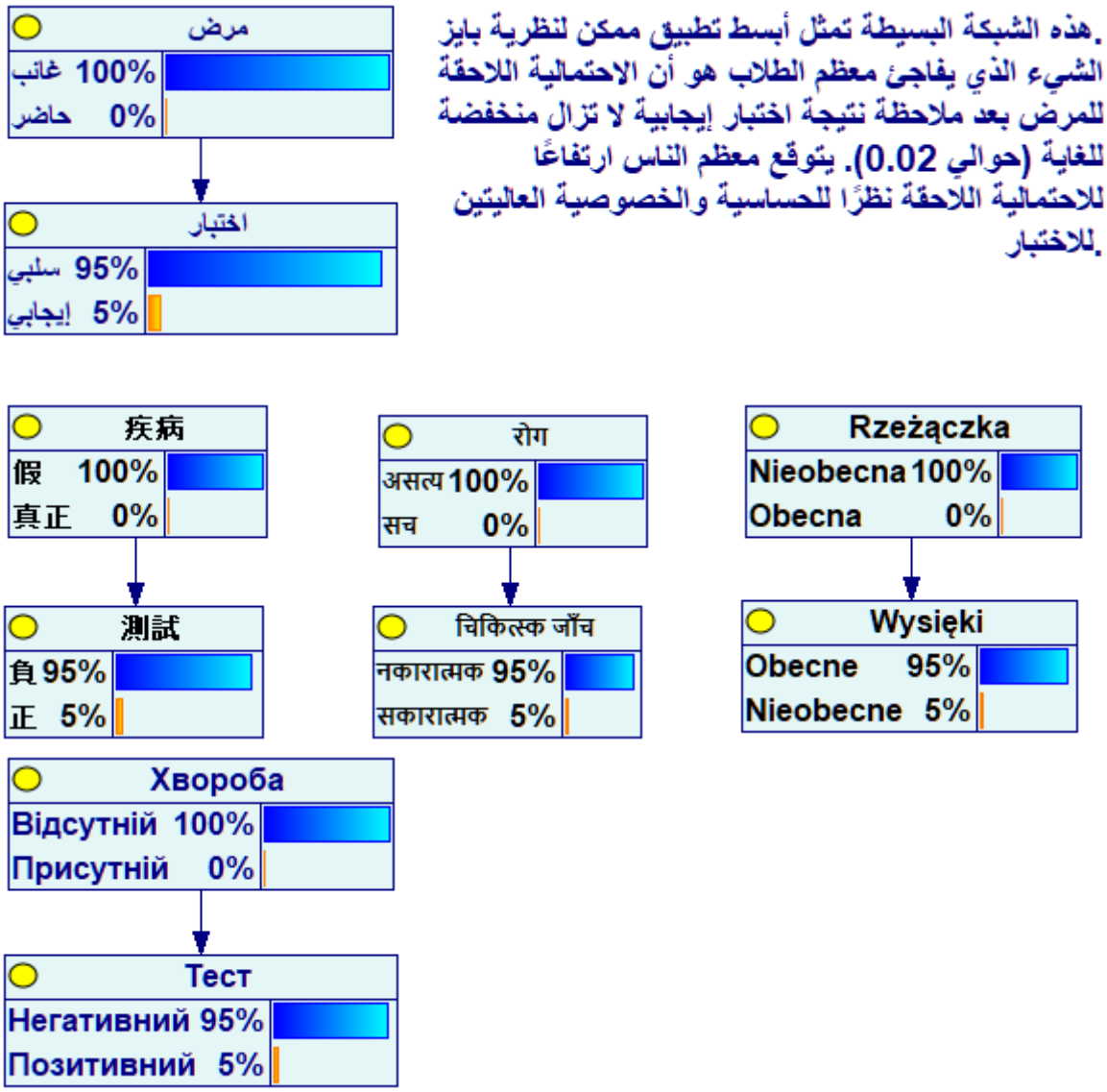
6. If the icons are not available in the launcher, GeNIe can be launched from the terminal. To run GeNIe from the terminal, first change the current directory to GeNIe's install directory. By default, the install directory is `"~/.wine/drive_c/Program Files/GeNIe 4.0"`. To execute GeNIe, run the following command:

   ```
   wine genie.exe
   ```

   Note that GeNIe Academic installs to `"GeNIe 4.0 Academic"` by default.

3.10 Non-Latin alphabets in GeNIe

GeNIe accommodates non-Latin alphabets - any text within a model can be written using letters outside of the Latin alphabet. This includes variables IDs, names, comments, etc. Here is the `Disease-Test` model in several alphabets:



3.11 Copyright notice

Academic use of GeNIe and SMILE

We support teachers using GeNIe in their classes and maintain shareware resources, such as network repositories, that are useful in teaching. (Please, visit BayesFusion, LLC web site for more information.) GeNIe, QGeNIe and SMILE are also useful in academic research projects. Our software is free for academic teaching and research use. In return for this, to get credit for our work, we ask that all publications of research or applications in which GeNIe, QGeNIe or SMILE were used contain an explicit acknowledgment to that effect. Examples of simple acknowledgments are listed below:

The models described in this paper were created using the GeNIe Modeler, available free of charge for academic research and teaching use from BayesFusion, LLC, https://www.bayesfusion.com/.

The core of our implementation is based on the SMILE reasoning engine for graphical probabilistic models, available free of charge for academic research and teaching use from BayesFusion, LLC, https://www.bayesfusion.com/.

3.12 Disclaimer

GeNIe, QGeNIe and SMILE are made available on an "as is" basis. We have performed extensive tests of the software, which has been used in thousands of research, teaching, and commercial projects, but we are not providing any guarantees as to its correct working and take no responsibility for effects of possible errors. We do appreciate suggestions and bug reports and will do the best within our capabilities to correct errors and accommodate users' needs in our future development plans. If you have suggestions or have discovered a bug in the program, please send us electronic mail at support@bayesfusion.com.

Similarly, while we have taken much care in writing this manual and making it as accurate as possible, we assume no responsibility for possible errors that it may contain. We encourage the readers to send us their corrections and suggestions at support@bayesfusion.com.

3.13 Acknowledgments

Support for the development GeNIe, QGeNIe and SMILE at the University of Pittsburgh was provided in part by the Air Force Office of Scientific Research under grants F49620-97-1-0225 and F49620-00-1-0122, by the National Science Foundation under Faculty Early Career Development (CAREER) Program, grant IRI-9624629, by Hughes Raytheon Laboratories, Malibu, California, by ARPA's Computer Aided Education and Training Initiative under grant N66001-95-C-8367, and by the University of Pittsburgh Central Development Fund.

While little of the original code has remained and most of the programs have been rewritten with time, the principal developers of GeNIe, QGeNIe and SMILE (listed alphabetically) included:

Saeed Amizadeh, Steve Birnie, Jeroen J.J. Bogers, Girish Chavan, Hanyang Chen, Jian Cheng, Denver H. Dash, Martijn de Jongh, Marek J. Druzdzel, Daniel Garcia Sanchez, Nancy Jackson, Randy Jagt, Joost Koiter, Marcin Kozniewski, Hans van Leijen, Yan Lin, Tsai-Ching Lu, Paul Maaskant, Agnieszka Onisko, Hans Ove Ringstad, Tomek Sowinski, Carl P.R. Thijssen, Miguel Tjon Kon Fat, Daniel Tomalesky, Mark Voortman, Changhe Yuan, Haiqin Wang and Adam Zagorecki.

We would like to acknowledge contributions of the following individuals (listed alphabetically) to coding, documentation, graphics, web site, and testing of SMILE and GeNIe: Kimberly Batch, Avneet S. Chatha, Cristina Conati, Roger Flynn, Abigail Gertner, Charles E. Grindle, Christopher Hall, Christopher A. Geary, William Hogan, Susan E. Holden, Margaret (Peggie) Hopkins, Jun Hu, Kent Ma, Robert (Chas) Murray, Zhendong Niu, Shih-Chueh (Sejo) Pan, Bharti Rai, Michael S. Rissman, Luiz E. Sant'Anna, Jeromy A. Smith, Jiwu Tao, Kurt VanLehn, Martin van Velsen, Anders Weinstein, David Weitz, Zaijiang Yuan, Jie Xu, and many others.

Students in the courses *Decision Analysis and Decision Support Systems* at the University of Pittsburgh, *Decision Support Systems for Public Managers* at Carnegie Mellon University, *Decision Support and Expert Systems* at the University of Alaska, Anchorage, and *Advanced Databases and Data Warehouses* at the Bialystok University of Technology, Poland, provided us with useful feedback and suggestions.

GeNIe, QGeNIe and SMILE embed a number of good ideas that we have gratefully assimilated over time from other software, whether decision-theoretic or not. The great user interface of Analytica was an inspiration and a role model for us. Analytica's user interface was developed by Max Henrion and Brian Arnold at Carnegie Mellon University in late 1980s and early 1990s. Our treatment of submodels is essentially the same as in Analytica. GeNIe Metalog Builder interface is strongly inspired by the interface provided by Tom Keelin at Metalog Distribution web site at http://metalogdistributions.com/. Knowledge Industries' (KI) WinDX software was a source of inspiration for our diagnostic functionality and interface. The ideas contained in WinDX, on information gathering modes, such as discriminating among groups of hypotheses and pursuing specific hypothesis, were developed over a span of time from the mid-1980s to the early 1990s by David Heckerman, Eric Horvitz, Mark Peot and Michael Shwe. The use of alternative abstractions of the differential diagnosis in value of information computations was pioneered in the Pathfinder project that was commercialized as Intellipath and then, refined later in the KI Bayesian network inference tool kit. Beyond functionality, the configuration of panes and bar charts for abnormalities, observations, and valuable tests in the KI software were an inspiration for the interface of GeNIe and SMILE.

We would like to thank the U.S. News and World Report for considerable data collection effort and generosity in making the collected retention related data available. These data (file `retention.txt`) have been used in describing the learning component of GeNIe.

Map files used as examples for explaining GeNIe geo-processing capability originate from National Center for Environmental Information (NCEI, https://www.ncei.noaa.gov/). We relied on the ETOPO1 Global Relief Model Maps (https://www.ngdc.noaa.gov/mgg/global/) and extracted them through their interactive Grid Extract interface https://www.ncei.noaa.gov/maps/grid-extract/.

This page is intentionally left blank.

Decision-theoretic modeling

4 Decision-theoretic modeling

4.1 Decision analysis

Decision analysis is the art and practice of decision theory, an axiomatic theory prescribing how decisions should be made. Decision analysis is based on the premise that humans are reasonably capable of framing a decision problem, listing possible decision options, determining relevant factors, and quantifying uncertainty and preferences, but are rather weak in combining this information into a rational decision.

Decision analysis comes with a set of empirically tested tools for framing decisions, structuring decision problems, quantifying uncertainty and preferences, discovering those factors in a decision model that are critical for the decision, and computing the value of information that reduces uncertainty. Probability theory and decision theory supply tools for combining observations and optimizing decisions. While GeNIe is under continuous development, it already implements a large set of these tools.

While decision analysis is based on two quantitative theories, probability theory and decision theory, its foundations are qualitative and based on axioms of rational choice. The purpose of decision analysis is to gain insight into a decision and not to obtain a recommendation. The users of GeNIe will notice that this important premise is reflected in its functionality and, most importantly, its user interface.

4.2 Discrete and continuous variables

One of the most fundamental properties of variables is their domain, i.e., the set of values that they can assume. While there is an infinite number of possible domains, they can be divided into two basic classes: discrete and continuous.

Discrete variables describe a finite set of conditions and take values from a finite, usually small, set of states. An example of a discrete variable is *Success of the venture*, defined in the tutorial on Bayesian networks. This variable can take two values: *Success* and *Failure*. Another example might be a variable *Hepatitis-B*, assuming values *True* and *False*. Discrete variables can be numerical. For example, variable *Total bilirubin* in model *HeparII* has four interval states: 0..2, 2..7, 7..20, and 20..88. Variable *Financial gain* may assume three numerical point values: *$10K*, *$20K*, and *$50K*.

Continuous variables can assume an infinite number of values. An example of a continuous variable is *Body temperature*, assuming any value between *30* and *45 degrees Celsius*. Another might be *Financial gain*, assuming any monetary value between *zero* and *$50K*.

Most exact algorithms for Bayesian networks and influence diagrams are designed for discrete variables. To take advantage of these algorithms, most Bayesian network and influence diagram models include discrete variables or conceptually continuous variables that have been discretized for the purpose of reasoning. GeNIe offers continuous and hybrid models, in which some or all variables are continuous and the interactions among variables are described by means of equations. In such cases, GeNIe uses stochastic sampling or on-demand discretization.

While the distinction between discrete and continuous variables is crisp, the distinction between discrete and continuous quantities is rather vague. Many quantities can be represented as both discrete and continuous. Discrete variables are usually convenient approximations of real world quantities, sufficient for the purpose of reasoning. And so, success of a venture might be represented by a continuous variable expressing the financial gain or stock price, but it can also be discretized to [*Good*, *Moderate*, *Bad*] or to [*$5*, *$20*, *$50*] *price per share*. *Body temperature* might be continuous but can be also discretized into intervals or categorized as *Low*, *Normal*, *Fever*, and *High fever*. Experience in decision analytic modeling has taught that representing continuous variables by their three to five point discrete approximations performs well in most cases.

4.3 Probability

Decision theoretic and decision analytic methods quantify uncertainty by probability. It is quite important for a decision modeler to understand the meaning of probability. There are three fundamental interpretations of probability:

- **Frequentist interpretation**

  Probability of an event in this view is defined as the limiting frequency of occurrence of this event in an infinite number of trials. For example, the probability of heads in a single coin toss is the proportion of heads in an infinite number of coin tosses.

- **Propensity interpretation**

  Probability of an event in this view is determined by physical, objective properties of the object or the process generating the event. For example, the probability of heads in a single coin toss is determined by the physical properties of the coin, such as its flat symmetric shape and its two sides.

- **Subjectivist interpretation**

  The frequentist and the propensity views of probability are known as objectivist as they assume that the probability is an objective property of the physical world. In the subjectivist, also known as Bayesian interpretation, probability of an event is subjective to personal measure of the belief in that event occurring.

While the above three interpretations of probability are theoretical and subject to discussions and controversies in the domain of philosophy, they have serious implications on the practice of decision analysis. The first two views, known collectively as objectivist, are impractical for most real world decision problems. In the frequentist view, in order for a probability to be a meaningful measure of uncertainty, it is necessary that we deal with a process that is or at least can be imagined as repetitive in nature. While coin tosses provide such a process, uncertainty related to nuclear war is a rather hard case - there have been no nuclear wars in the past and even their repetition is rather hard to imagine. Obviously, for a sufficiently complex process, such as circumstances leading to a nuclear war, it is not easy to make an argument based on physical considerations. The subjectivist view gives us a tool for dealing with such problems and is the view embraced by decision analysis. The process of obtaining a subjective probability from a decision maker is known as probability elicitation.

The subjectivist view interprets probability as a measure of personal belief. It is legitimate in this view to believe that the probability of heads in a single coin toss is 0.3, just as it is legitimate to believe that it is 0.5 as long as one does not violate the axioms of probability, such as one stating that the sum of probabilities of an event and its complement is equal to 1.0. It is also legitimate to put a measure of uncertainty on the event of nuclear war. Furthermore, this measure, a personal belief in the event, can vary among various individuals. While this sounds perhaps like a little too much freedom, this view comes with a rule for updating probability in light of new observations, known as Bayes theorem. There exist *limits theorems* that prove that if Bayes theorem is used for updating the degree of belief, this degree of belief will converge to the limiting frequency regardless of the actual value of the initial degree of belief (as long as it is not extreme in the sense of being exactly zero or exactly one). While these theorems give guarantees in the infinity, convergence is fast in practice, and a reasonable prior belief makes convergence even faster.

The subjectivist view makes it natural to combine frequency data with expert judgment. Numerical probabilities can be extracted from databases, can be based on expert judgment, or a combination of both. Obtaining numbers for probabilistic and decision-theoretic models is not really difficult. The process of measuring the degree of belief is referred to as a probability assessment. Various decision-analytic methods are available for probability assessment.

4.4 Utility

An integral element of all decision problems, one without which no decision can be made, is the notion of preference. Very often, preference can be based on an objective quantity, such as material usage, factory output, or financial gain. Typically, however, decision problems involve quantities that have no obvious numerical measure, such as state of health, customer satisfaction, or pain. Another complication is a possibly conflicting set of attributes, such as price and quality. Even if a numerical measure of goodness of an outcome is available, such as is the case with financial gains and losses, it may not reflect well decision maker's preferences in presence of risk.

Decision theory introduces a measure of preference, known as utility. Utility is a function mapping the attributes of the possible outcomes of a decision process on the set of real numbers. Utility is determined up to a linear transformation, i.e., a decision maker's preference over different decision alternatives is invariant to multiplying the utility by a non-negative number and adding a constant. This implies that utility has neither a meaningful zero point, nor a meaningful scale.

Utility is by assumption subjective: various decision makers facing the same choice and even sharing the same set of beliefs about the world may choose differently because of their different preference structure and different utility functions. A utility function for any decision problem needs to be obtained from a decision maker. The process of obtaining a utility function from a decision maker is known as utility elicitation.

It is worth pointing out that variables measuring utility are always continuous: they can assume any values from a continuous interval. Sometimes they are mistakenly taken for discrete variables, as in graphical models, such as influence diagrams, they usually have discrete parents and take a finite number of values. It is much clearer to see that *multi-attribute utility* (MAU) variables are continuous - they specify a function by which the values of their parents, *utility* nodes, are combined.

4.5 Bayesian networks

Bayesian networks (also called *belief networks*, *Bayesian belief networks*, *causal probabilistic networks*, or *causal networks*) (Pearl 1988) are acyclic directed graphs in which nodes represent random variables and arcs represent direct probabilistic dependences among them. The structure of a Bayesian network is a graphical, qualitative illustration of the interactions among the set of variables that it models. The structure of the directed graph can mimic the causal structure of the modeled domain, although this is not necessary. When the structure is causal, it gives a useful, modular insight into the interactions among the variables and allows for prediction of effects of external manipulation.

The name Bayesian originates from the fact that the joint probability distribution represented by a Bayesian network is subjective (please recall that they are sometimes called *belief networks*; *Bayesian approach* is often used as a synonym for subjective view on probability) and this subjective probability distribution can be updated in the light of new evidence using Bayes theorem.

Bayesian networks: The theoretical minimum

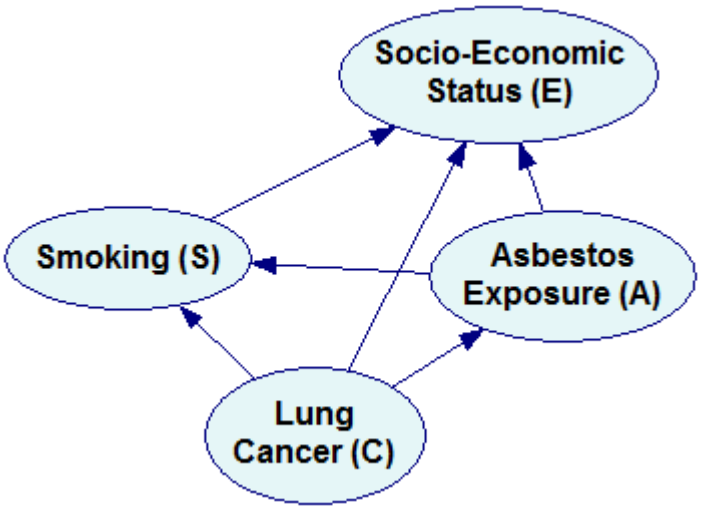
Bayesian networks are acyclic directed graphs that represent factorizations of joint probability distributions. Every joint probability distribution over n random variables can be factorized in *n*! ways and written as a product of probability distributions of each of the variables conditional on other variables. Consider a simple model consisting of four variables *Socio-Economic Status* (E), *Smoking* (S), *Asbestos Exposure* (A), and *Lung Cancer* (C). The joint probability distribution over these four variables can be factorized in 4!=24 ways, for example,

```
Pr(E,S,A,C)=Pr(E|S,A,C) Pr(S|A,C) Pr(A|C) Pr(C)
Pr(E,S,A,C)=Pr(E|S,A,C) Pr(S|A,C) Pr(C|A) Pr(A)
Pr(E,S,A,C)=Pr(E|S,A,C) Pr(S|A,C) Pr(A|C) Pr(C)
…
Pr(E,S,A,C)=Pr(C|E,S,A) Pr(A|E,S) Pr(S|E) Pr(E)
…
Pr(E,S,A,C)=Pr(C|E,S,A) Pr(A|E,S) Pr(S|E) Pr(E)
```

Each of these factorizations can be represented by a Bayesian network. We construct the directed graph of the network by creating a node for each of the factors in the distribution (we label each of the nodes with the name of a variable before the conditioning bar) and drawing directed arcs between them, always from the variables on the right hand side of the conditioning bar to the variable on the left hand side. For the first factorization above

```
Pr(E,S,A,C)=Pr(E|S,A,C) Pr(S|A,C) Pr(A|C) Pr(C)
```

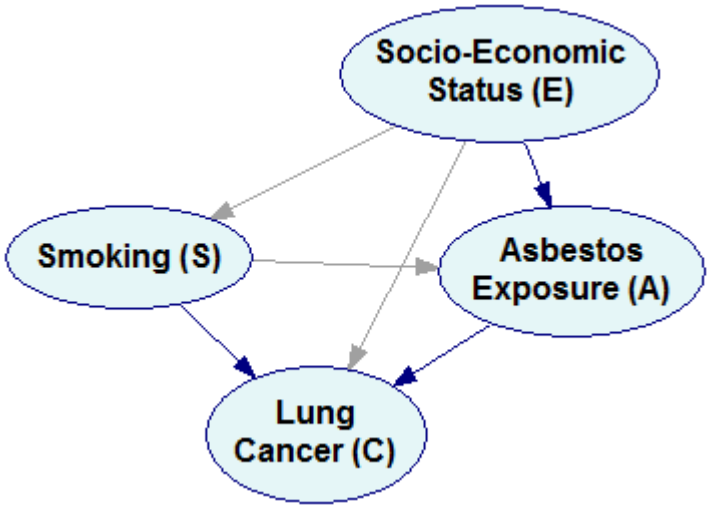
we will construct the following Bayesian network



The fourth factorization

```
Pr(E,S,A,C)=Pr(C|E,S,A) Pr(A|E,S) Pr(S|E) Pr(E)
```

will yield the following Bayesian network



Please note that we have an arc from E to S because there is a factor Pr(S|E) in the factorization. Arcs from S, E, and A to C correspond to the factor Pr(C|E,S,A).

A graph tells us much about the structure of a probabilistic domain but not much about its numerical properties. These are encoded in conditional probability distribution matrices (equivalent to the factors in the factorized form), called conditional probability tables (CPTs) that are associated with the nodes. It is worth noting that there will always be nodes in the network with no predecessors. These nodes are characterized by their prior marginal probability distribution. Any probability in the joint probability distribution can be determined from these explicitly represented prior and conditional probabilities.

For the network above, we will have the following CPTs for the nodes E, S, A and C respectively:

Educated	0.2
UnEducated	0.8

Socio-Economic Status (E)	Educated	UnEducated
S	0.1	0.1
NS	0.9	0.9

Socio-Economic Status (E)	Educated		UnEducated	
Smoking (S)	S	NS	S	NS
A	0.0001	0.0001	0.015	0.015
NA	0.9999	0.9999	0.985	0.985

Smoking (S)	S				NS			
Asbestos Exposure (A)	A		NA		A		NA	
Socio-Economic Status (E)	Educated	UnEducated	Educated	UnEducated	Educated	UnEducated	Educated	UnEducated
C	0.2	0.2	0.1	0.1	0.1	0.1	0.01	0.01
NC	0.8	0.8	0.9	0.9	0.9	0.9	0.99	0.99

A straightforward representation of the join probability distribution over n binary variables requires us to represent the probability of every combination of states of these variables. For *n* binary variables, for example, we have 2^n-1 such combinations. In the example network above, we need 2^4-1=15 numerical parameters. While there are 30 numbers in the four tables above, please note that half of them are implied by other parameters, as the sum of all probabilities in every distribution has to be 1.0, which makes the total number of independent parameters equal to 15.

Now, suppose that we know that *Socio-Economic Status* and *Smoking* are independent of each other. If so, then knowing whether somebody is educated or not tells us nothing about this person's smoking status. Formally, we have Pr(S|E)=Pr(S). This allows us to simplify the factorization to the following

```
Pr(E,S,A,C)=Pr(C|E,S,A) Pr(A|E,S) Pr(S) Pr(E)
```

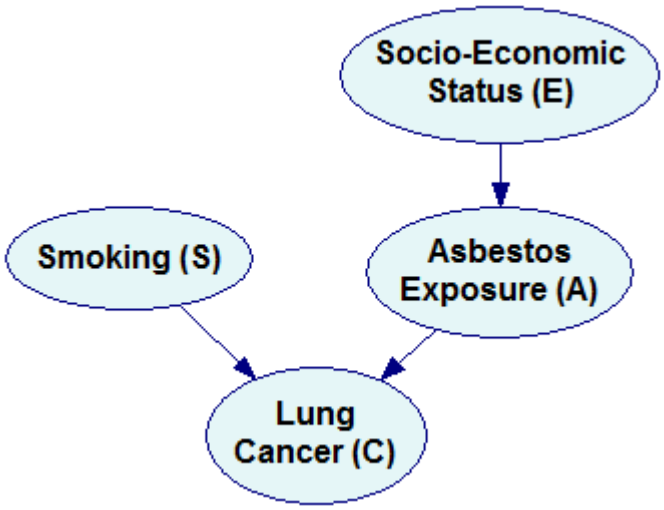
Similarly, suppose that we know that *Asbestos Exposure* and *Smoking* are independent of each other. If so, then knowing whether somebody smokes or not tells us nothing about this person's exposure to asbestos. Formally, we have Pr(A|S)=Pr(A). This allows us to further simplify the factorization to the following

```
Pr(E,S,A,C)=Pr(C|E,S,A) Pr(A|E) Pr(S) Pr(E)
```

Finally, supposed that we realize that knowledge of *Smoking* status and *Asbestos Exposure* makes *Socio-Economic Status* irrelevant to the probability of *Cancer*, i.e., Pr(C|E,S,A)= Pr(C|S,A). We can simplify the factorization further to

```
Pr(E,S,A,C)=Pr(C|S,A) Pr(A|E) Pr(S) Pr(E)
```

The Bayesian network representing the simplified factorization looks as follows



Please note that the new network is missing three arcs compared to the original network (marked by dimmed arcs in the previous pictures). These three arcs correspond to three independences that we encoded in the simplified factorization. It is a general rule that every independence between a pair of variables results in a missing arc. Conversely, if two nodes are not directly connected by an arc, then there exists such set of variables in the joint probability distribution that makes them conditionally independent. A possible interpretation of this is that the variables are not directly dependent. If we need to condition on other variables to make them independent, then they are indirectly dependent.

Let us look at the conditional probability tables encoded in the simplified network

Educated	0.2
UnEducated	0.8

S	0.1
NS	0.9

Socio-Economic Status (E)	Educated	UnEducated
A	0.0001	0.015
NA	0.9999	0.985

Smoking (S)	S		NS	
Asbestos Exposure (A)	A	NA	A	NA
C	0.2	0.1	0.1	0.01
NC	0.8	0.9	0.9	0.99

This simpler network contains fewer numerical parameters, as the CPTs for the nodes *Smoking*, *Lung Cancer* and *Asbestos Exposure* are smaller. The total number of parameters is 16 and the total number of independent

parameters is only 8. This reduction in the number of parameters necessary to represent a joint probability distribution through an explicit representation of independences is the key feature of Bayesian networks.

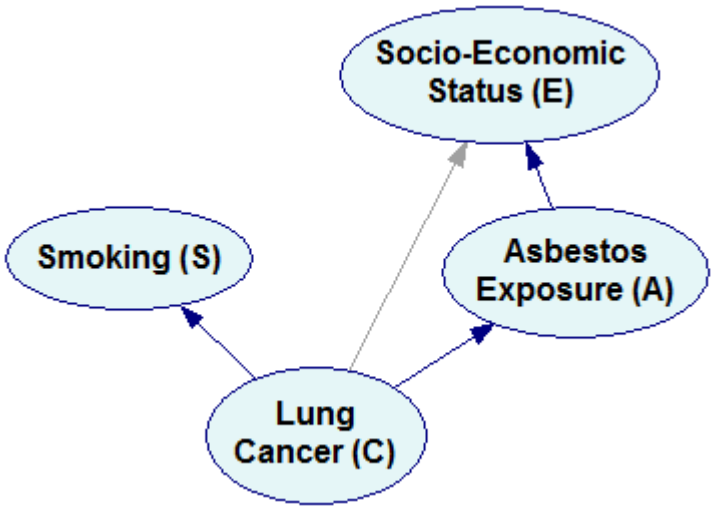
Of all the factorizations possible, those that are capable of representing all independences known in the domain are preferable. Please note that the three independences that we were aware of, Pr(S|E)=Pr(S), Pr(A|S)=Pr(A) and Pr(C|E,S,A)= Pr(C|S,A) could not be captured by each of the possible factorizations. For example, in case of the first factorization

```
Pr(E,S,A,C)=Pr(E|S,A,C) Pr(S|A,C) Pr(A|C) Pr(C)
```

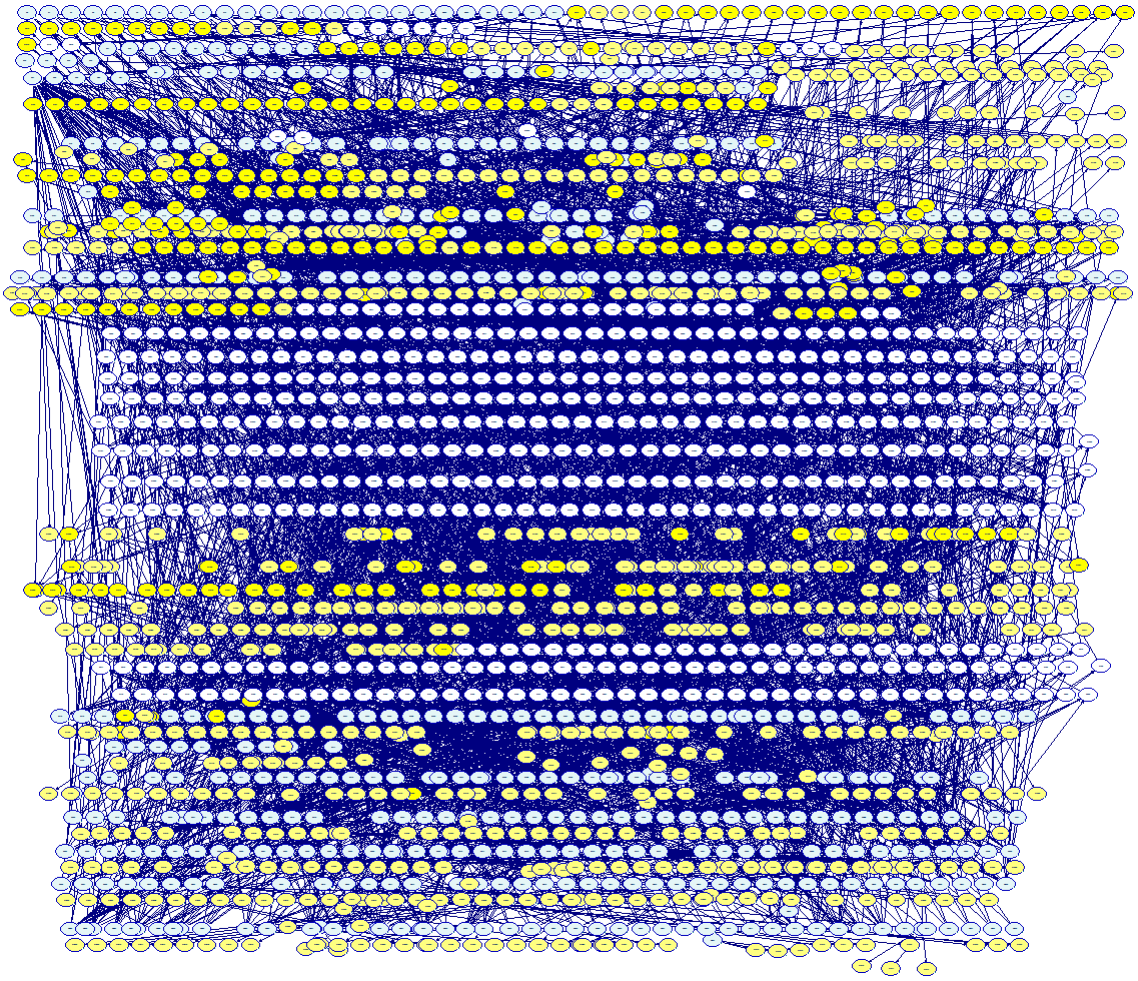
there is no way we could capture the independence Pr(C|E,S,A)= Pr(C|S,A) and the factorization can be simplified to only

```
Pr(E,S,A,C)=Pr(E|A,C) Pr(S|C) Pr(A|C) Pr(C)
```

which corresponds to the following Bayesian network



Using independences to simplify the graphical model is a general principle that leads to simple, efficient representations of joint probability distributions. As an example, consider the following Bayesian network, modeling various problems encountered in diagnosing Diesel locomotives, their possible causes, symptoms, and test results (included among the example networks as `MachineDiagnosis.xdsl`)



This network contains 2,127 nodes, i.e., it models a joint probability distribution over 2,127 variables. Given that each of the variables is binary, to represent this distribution in a brute force way, we would need 2^{2127} (which is around 10^{632}) numbers. To make the reader aware of the magnitude of this number, we would like to point out that the currently held estimate for the number of atoms in the universe is merely 10^{82}, which is 550 orders of magnitude (!) smaller than 10^{632}, i.e., it would require as many parameters as the number of atoms in 10^{550} universes like ours. The model pictured above is represented by only 6,433 independent parameters. As we can see, representing joint probability distributions becomes practical and models of the size of the example above are not uncommon.

Deterministic nodes

Deterministic nodes, usually drawn as double-circles or double-ovals, represent either constant values or values that are algebraically determined from the states of their parents. In other words, if the values of their parents are known, then the value of a deterministic node is also known with certainty. *Deterministic* nodes are quantified similarly to *Chance* nodes. The only difference is that their probability tables contain all zeros and ones (note that there is no uncertainty about the outcome of a deterministic node once all its parents are known).

Bayesian networks: A less formal (although also perfectly sound) view

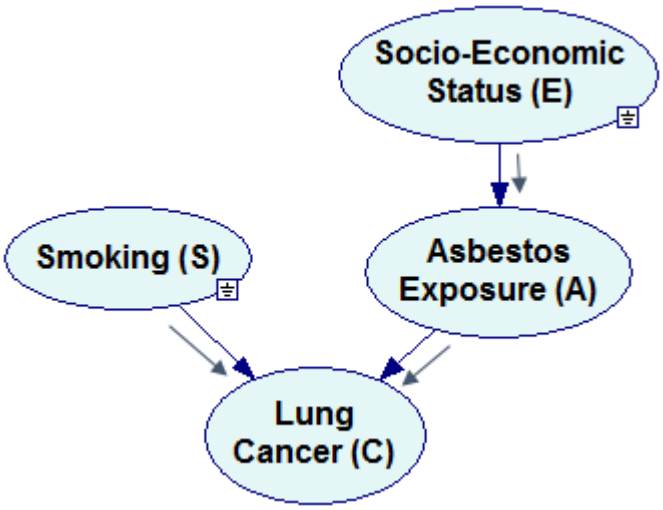
The purely theoretical view that Bayesian networks represent independences and that lack of an arc between any two variables X and Y represents a (possibly conditional) independence between them, is not intuitive and convenient in practice. A popular, slightly informal view of Bayesian networks is that they represent causal graphs in which every arc represents a direct causal influence between the variables that it connects. A directed arc from X to Y captures the knowledge that X is a causal factor for Y. While this view is informal and it is easy to construct mathematically correct counter-examples, it is convenient and widely used by almost everybody applying Bayesian networks in practice. There is a well-established assumption, with no convincing counter-examples, that causal graphs will automatically lead to correct patterns of independences. Please recall the Bayesian network constructed in the previous section. Its arcs correspond to the intuitive notions that *Smoking* and *Asbestos Exposure* can cause *Lung Cancer* and that *Socio-Economic Status* influences *Asbestos Exposure*. Lack of arcs between pairs of variables expresses simple facts about absence of causal influences between them. Independences between these pairs of variables follow from the structure of the graph. It is, thus, possible to construct Bayesian networks based purely on our understanding of causal relations between variables in our model.

Both, the structure and the numerical parameters of a Bayesian network, can be elicited from an expert. They can also be learned from data, as the structure of a Bayesian network is simply a representation of independences in the data and the numbers are a representation of the joint probability distributions that can be inferred from the data. Finally, both the structure and the numerical probabilities can be based on a mixture of expert knowledge, measurements, and objective frequency data.

Reasoning with Bayesian networks

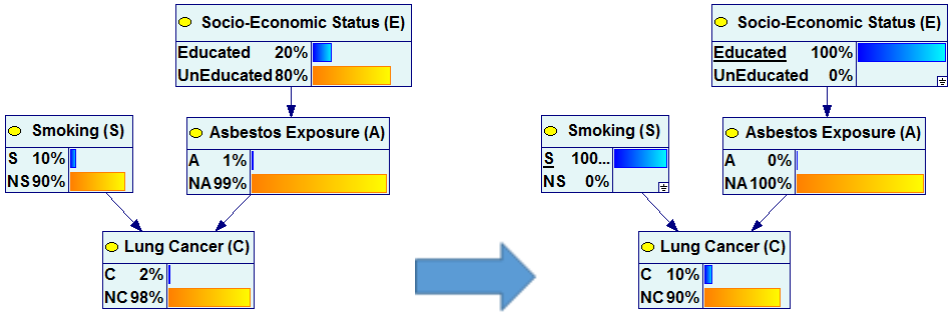
Structural properties of Bayesian networks, along with the conditional probability tables associated with their nodes allow for probabilistic reasoning within the model. Probabilistic reasoning within a BN is induced by observing evidence. A node that has been observed is called an evidence node. Observed nodes become instantiated, which means, in the simplest case, that their outcome is known with certainty. The impact of the evidence can be propagated through the network, modifying the probability distribution of other nodes that are probabilistically related to the evidence. For the example network, modeling various causes of *Lung Cancer*, we can answer questions like *What is the probability of lung cancer in an educated smoker?*, *What is the probability of asbestos exposure in a lung cancer patient?*, or *Which question should we ask next?*

In case of the first question, we instantiate the variables *Smoking* and *Socio-Economic Status* to *S* and *Educated* respectively and perform update of the network. The evidence entered can be visualized as spreading across the network.



This process amounts at the foundations to a repetitive application of Bayes theorem in order to update the probability distributions of all nodes in the network. Different ways of applying Bayes theorem and different order of updating lead to different algorithms. Essentially, the existing algorithms for reasoning in Bayesian networks can be divided into three groups: message passing, graph reduction, and stochastic simulation. Explicit representation of independences allows for an increased computational tractability of probabilistic reasoning. Probabilistic inference in singly connected BNs is very efficient. Unfortunately, exact algorithms for multiply connected networks are liable to exponential complexity in the number of nodes in the network. Cooper (1990) has shown that the problem is NP-hard in general. Still, efficient software, like SMILE, offers reasonable computing times even in networks consisting of thousands of nodes, like the network for the diagnosis of Diesel locomotives illustrated above.

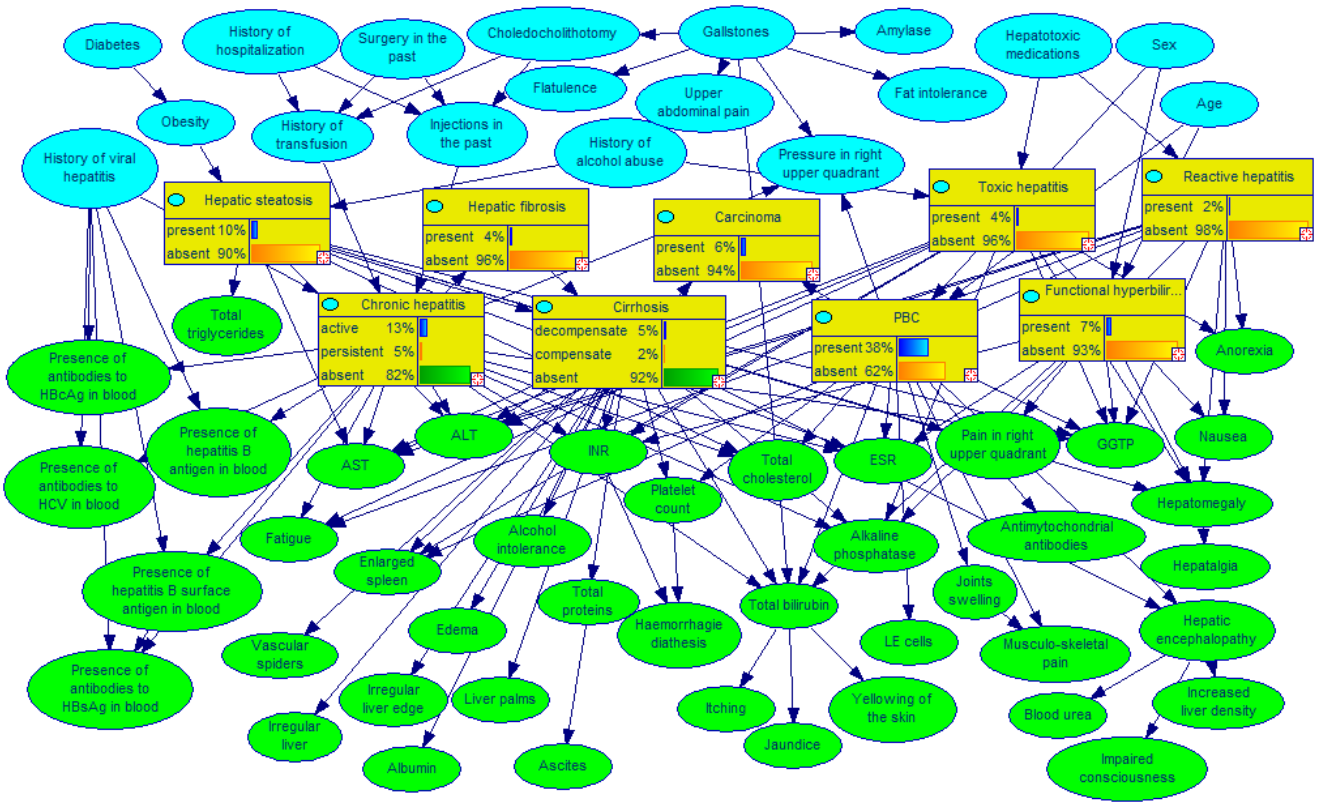
The following pair of graphs illustrates this process. It is our example network but all nodes in the graph are viewed as "bar charts."



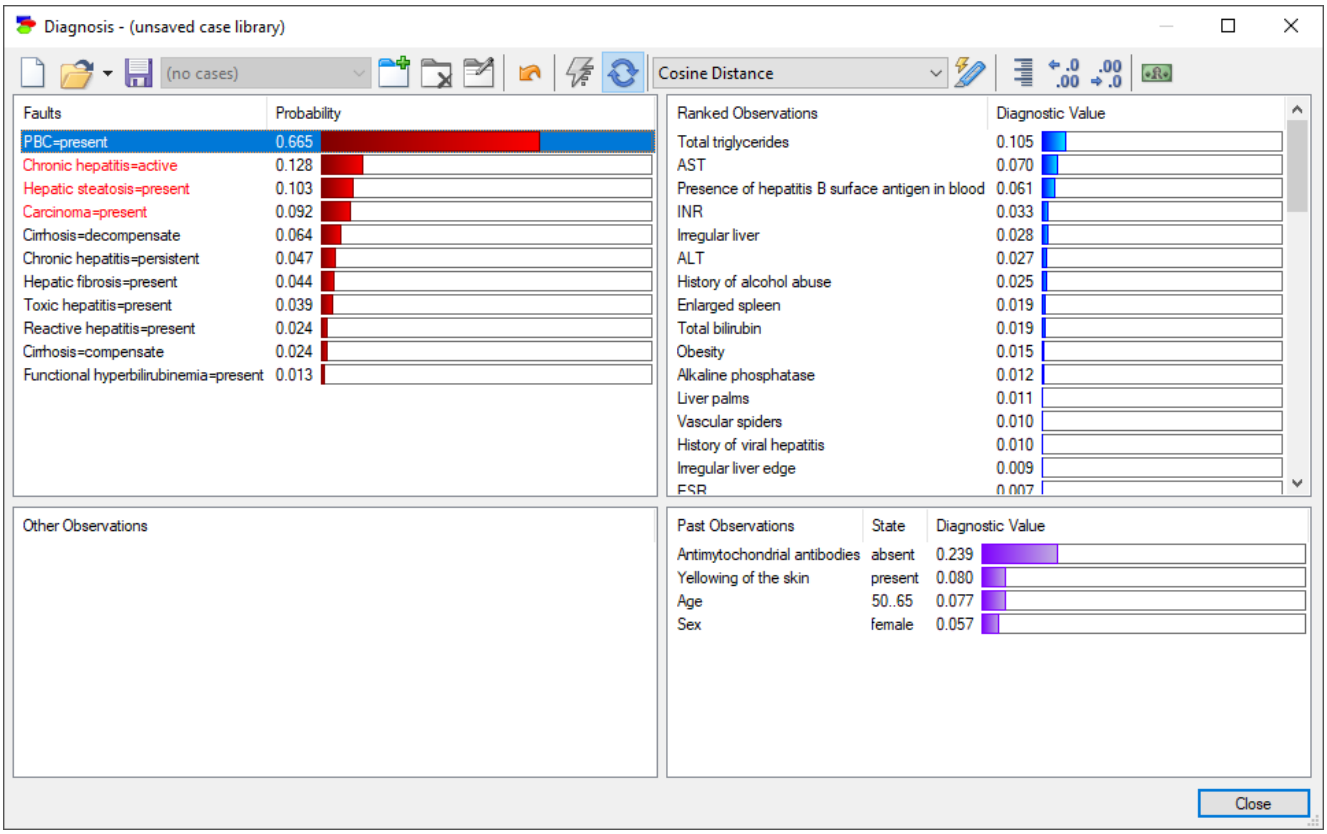
We can see that once the observation of smoking and education have been made, the marginal probability of C (*Lung Cancer* present) changes from 2% (*a-priori* probability) to 10% (*a-posteriori* probability).

Complex models allow for asking questions similar to the ones asked of the simple example network. And so, in case of the Diesel locomotives example, we can ask the probability of any fault given a set of symptoms or test results, we can ask for the most probable faults, and the most effective questions to be posed in order to reach a final diagnosis.

The following network (included among the examples as `HeparII.xdsl`) models common diseases of the liver.



Special diagnostic dialog allows for viewing the diseases (upper-left pane) from the most to least likely and the possible observations and medical tests rank-ordered from the most to least informative given the current patient case.



This is all straightforward to compute from the joint probability distribution represented by the Hepar II Bayesian network.

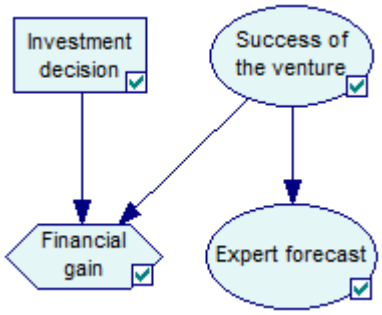
Embedding Bayesian networks technology into user programs

Bayesian networks can be embedded into custom programs and web interfaces, helping with calculating the relevance of observations and making decisions. SMILE Engine, our software library embedding Bayesian networks, has been deployed in a variety of environments, including custom programs, web servers, and on-board computers.

4.6 Influence diagrams

Influence diagrams (Howard & Matheson 1984), sometimes also called *relevance diagrams*, are acyclic directed graphs representing decision problems. The goal of influence diagrams is to choose a decision alternative that has the highest expected gain (expected utility).

Similarly to Bayesian networks, influence diagrams are useful in showing the structure of the domain, i.e., the structure of the decision problem. Influence diagrams contain four types of nodes (*Decision*, *Chance*, *Deterministic* and *Value*) and two types of arcs (influences and informational arcs). The following influence diagram (available among the example models as `VentureID.xdsl`), discussed in detail in the section on influence diagrams, models a decision related to an investment in a risky venture:



Nodes in an influence diagram represent various types of variables.

Decision nodes, usually drawn as rectangles (such as node *Investment decision* above), represent variables that are under control of the decision maker and model available decision alternatives, modeled explicitly as possible states of the decision node. Node *Investment decision* above has two alternatives *Invest* and *DoNotInvest*.

Invest
DoNotInvest

Chance nodes, usually drawn as circles or ovals (such as nodes *Expert forecast* and *Success of the venture* above) are random variables and they represent uncertain quantities that are relevant to the decision problem. They are quantified by conditional probability distributions, identical to those presented in the section on Bayesian networks. In fact, the subset of an influence diagram that consists of only chance nodes is a Bayesian network, i.e., an influence diagram can be also viewed as Bayesian network extended by decision and value nodes. When *Decision* nodes precede *Chance* nodes, they act exactly similar to those predecessors that are *Chance* nodes - they index the conditional probability table of the child node.

Value nodes, usually drawn as diamonds (such as node *Financial gain* above), represent utility, i.e., a measure of desirability of the outcomes of the decision process. They are quantified by the utility of each of the possible combinations of outcomes of the parent nodes. Node *Financial gain* above is quantified in the following way:

Investment decision	Invest		DoNotInvest	
Success of the venture	Success	Failure	Success	Failure
Value	10000	-5000	500	500

Please note that value nodes are always continuous and their domains are the domains of the utility function.

Normally, an arc in an influence diagram denotes an influence, i.e., the fact that the node at the tail of the arc influences the value (or the probability distribution over the possible values) of the node at the head of the arc. Some arcs in influence diagrams have clearly a causal meaning. In particular, a directed path from a decision node to a chance node means that the decision (i.e., a manipulation of the graph) will impact that chance node in the sense of changing the probability distribution over its outcomes.

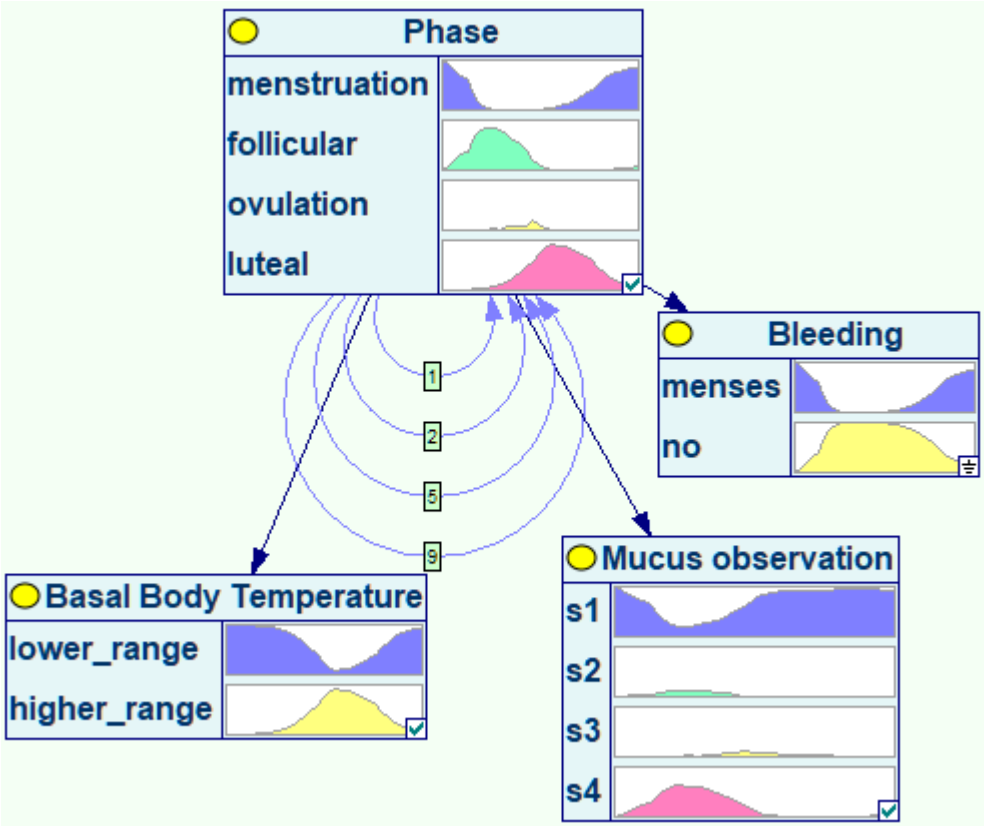
Arcs coming into decision nodes are called informational arcs and have a different meaning. As decision nodes are under the decision maker's control, these arcs do not denote influences but rather temporal precedence (in the sense of flow of information). The outcomes of all nodes at the tail of informational arcs will be known before the decision is made. In particular, if there are multiple decision nodes, they all need to be connected by informational arcs. This reflects the fact that the decisions are made in a sequence and the outcome of each decision is known before the next decision is made.

4.7 Dynamic Bayesian networks

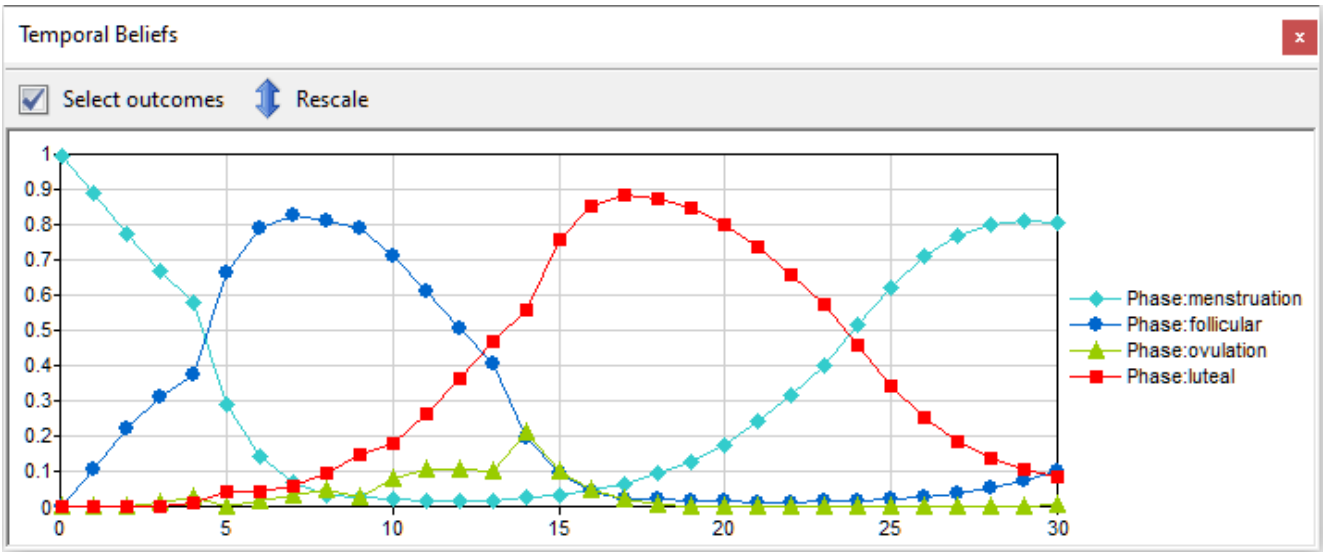
Most systems in the world change over time and sometimes we are interested in how these systems evolve over time more than we are interested in their equilibrium states, as modeled by Bayesian networks. Whenever the focus of our reasoning is change of a system over time, we need a tool that is capable of modeling dynamic systems.

A *dynamic Bayesian network* (*DBN*) is a Bayesian network extended with additional mechanisms that are capable of modeling influences over time (Murphy, 2002). The temporal extension of BNs does not mean that the network structure or parameters changes dynamically, but that a dynamic system is modeled. In other words, the underlying process, modeled by a DBN, is stationary. A DBN is a model of a stochastic process. While Bayesian networks could be thought of as close relatives of systems of simultaneous equations, DBNs can be thought as close relatives of difference equations (difference rather than differential equations because time, as modeled in DBNs, is discrete).

The following example of a DBN originates from the doctoral research conducted by Dr. Anna Lupinska-Dubicka (2014) and is available from BayesFusion's interactive on-line model repository. It models woman's monthly cycle and helps with predicting the day of ovulation, with applications to fertility awareness and difficulties with conception.



This DBN model allows for deriving probability distributions over variables of interest over time. The following plot shows the probabilities of various phases of the monthly cycle as a function of time.


Temporal Beliefs
Select outcomes
Rescale
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
Phase:menstruation
Phase:follicular
Phase:ovulation
Phase:luteal

4.8 Bayesian updating

Bayesian networks allow for performing Bayesian inference, i.e., computing the impact of observing values of a subset of the model variables on the probability distribution over the remaining variables. For example, observing a set of symptoms, captured as variables in a medical diagnostic model, allows for computing the probabilities of diseases captured by this model.

Bayesian updating, also referred to as belief updating, or somewhat less precisely as probabilistic inference, is based on the numerical parameters captured in the model. The structure of the model, i.e., an explicit statement of independences in the domain, helps in making the algorithms for Bayesian updating more efficient. All algorithms for Bayesian updating are based on a theorem proposed by Rev. Thomas Bayes (1702-1761) and known as *Bayes theorem*.

Belief updating in Bayesian networks is computationally complex. In the worst case, belief updating algorithms are NP-hard (Cooper 1990). There exist several efficient algorithms, however, that make belief updating in graphs consisting of tens or hundreds of variables tractable. Pearl (1986) developed a message-passing scheme that updates the probability distributions for each node in a Bayesian networks in response to observations of one or more variables. Lauritzen and Spiegelhalter (1988), Jensen et al.(1990), and Dawid (1992) proposed an efficient algorithm that first transforms a Bayesian network into a tree where each node in the tree corresponds to a subset of variables in the original graph. The algorithm then exploits several mathematical properties of this tree to perform probabilistic inference.

Several approximate algorithms based on stochastic sampling have been developed. Of these, best known are *probabilistic logic sampling* (Henrion 1998), *likelihood sampling* (Shachter & Peot 1990, Fung & Chang 1990), *backward sampling* (Fung & del Favero 1994), *Adaptive Importance Sampling* (AIS) (Cheng & Druzdzel 2000), and quite likely the best stochastic sampling algorithm available at the moment, *Evidence Pre-propagation Importance Sampling* (EPIS) (Yuan & Druzdzel 2003). Approximate belief updating in Bayesian networks has been also shown to be worst-case NP-hard (Dagum & Luby 1993).

In most practical networks of the size of tens or hundreds of nodes, Bayesian updating is rapid and takes a fraction of a second. Updating more complex networks may take a few seconds.

4.9 Solving decision models

In brief, solving decision-theoretic models amounts to computation of the expected utility of each of the possible decision alternatives (or strategies in case of multiple decision stages) and selecting the alternative or strategy with the highest expected utility. The first algorithm for inference in influence diagrams was proposed by Olmsted (1983) and later refined by Shachter (1988). This algorithm reverses arcs and removes nodes in the network structure until the answer to the given probabilistic query can be read directly from the graph. Cooper (1988) proposed an algorithm for inference in influence diagrams that transforms an influence diagrams into a Bayesian network and finds the expected utilities of each of the decision alternatives by performing repeated inference in this network.

Decision-theoretic models can be also studied with respect to the value of information, i.e., the value of observing a variable (reducing its uncertainty to zero) before making a decision. Another set of questions that can be asked of a decision model involve sensitivity analysis, i.e., the impact of imprecision in the model's numerical parameters on the solution.

It is important to realize that the insight into a decision problem, including the qualitative structure of the problem, available decision alternatives, expected utility of choosing any of them, importance of various sources of uncertainty, the value of reducing this uncertainty, are by far more important than the actual recommendation.

4.10 Changes in structure

Changes in structure are external manipulations that modify a system in question. An example of a change in structure is imposition of a new tax within the economic system of a country. It is of critical interest to decision makers to be able to make predictions of the effects of changes in structure. Since the model reflects the reality in its unmanipulated form and changes in structure of the kind contemplated by the decision maker have perhaps never been performed, predicting their effect is in general daunting.

In order to be able to predict the effect of arbitrary changes in structure, it is necessary that the model contain causal information. Directed graphs allow for representation of causality. One may adopt the convention that each arc in the graph denotes a direct causal relation between the parent and the child node. We recommend that all models built are causal in that sense. The operation of controlling a value is an example of a causal manipulation and may result in changes in structure.

GeNIe and SMILE are unique in supporting changes in structure in decision models. Please see *Controlling values of variables* section of this document for additional details.

4.11 Decision support systems

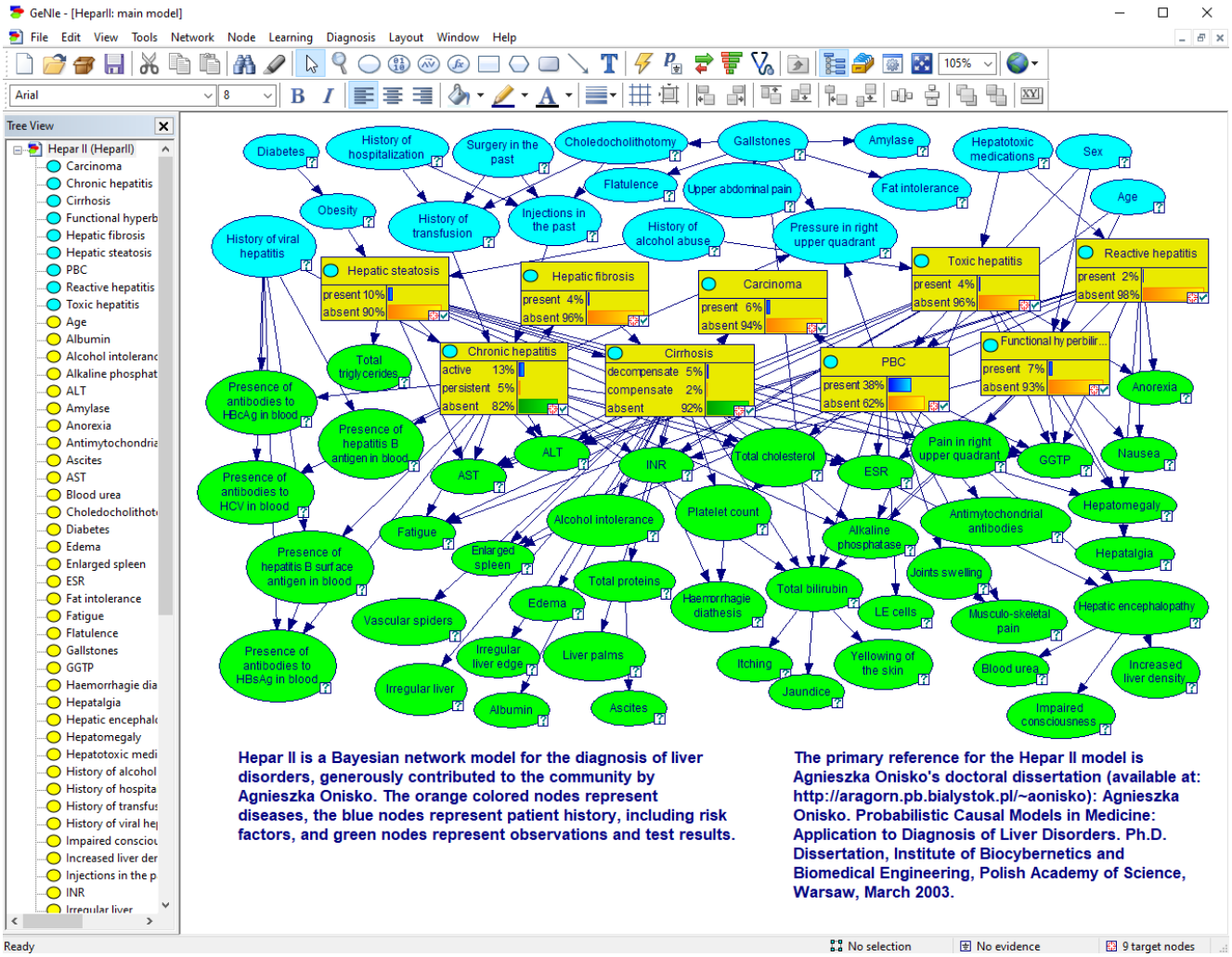
The principles of decision-analytic decision support, implemented in GeNIe and SMILE can be applied in practical decision support systems (DSSs). In fact, quite a number of probabilistic decision support systems have been developed, in which GeNIe plays the role of a developer's environment and SMILE plays the role of the reasoning engine. A decision support system based on SMILE can be equipped with a dedicated user interface, both standalone and web-based.

Probabilistic DSSs, applicable to problems involving classification, prediction, and diagnosis, are a new generation of systems that are capable of modeling any real-world decision problem using theoretically sound and practically invaluable methods of probability theory and decision theory. Based on graphical representation of the problem structure, these systems allow for combining expert opinions with frequency data, gather, manage, and process information to arrive at intelligent solutions.

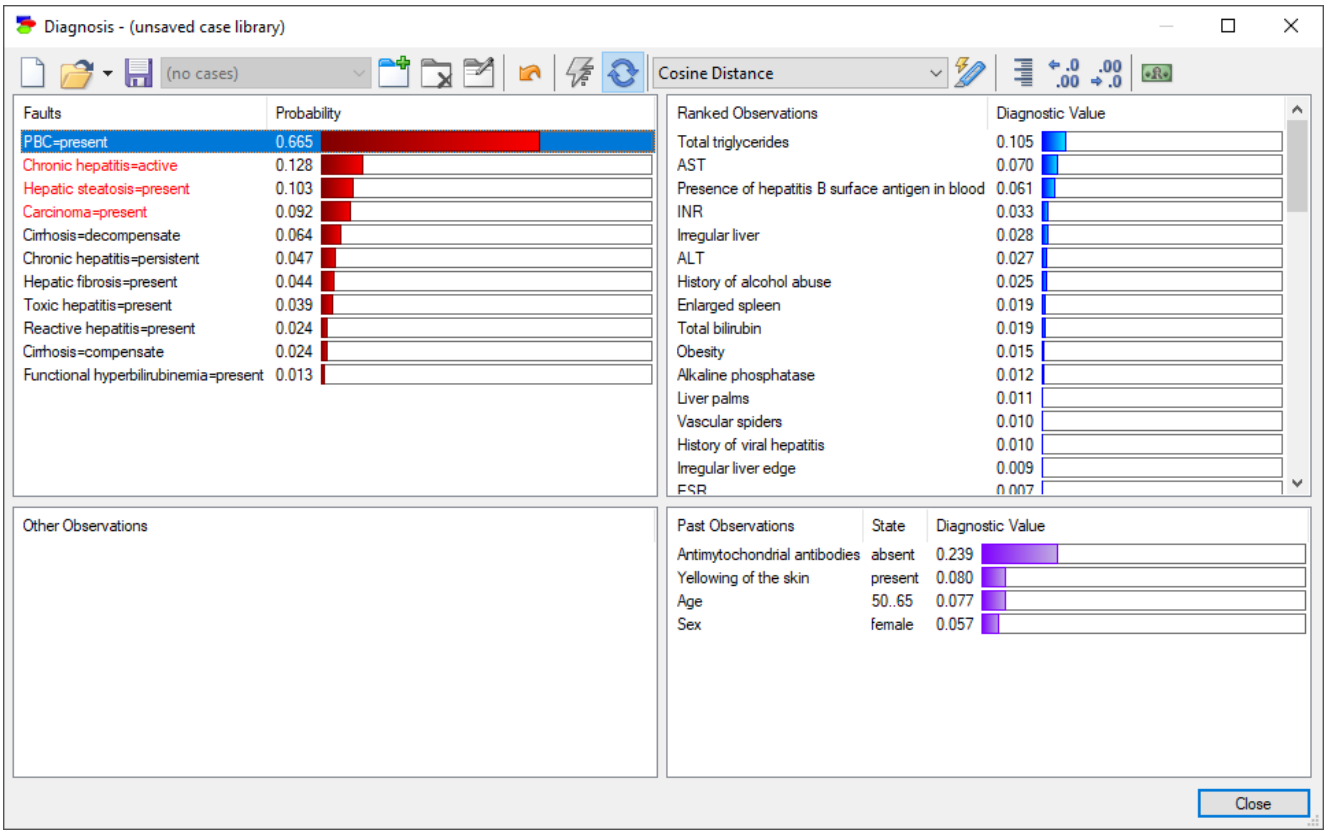
Probabilistic DSSs are based on a philosophically different principle than rule-based expert systems. While the latter attempt to model the reasoning of a human expert, the former use an axiomatic theory to perform computation. The soundness of probability theory provides a clear advantage over rule-based systems that usually represent uncertainty in an ad-hoc manner, such as using certainty factors, leading to under-responsiveness or over-responsiveness to evidence and possibly incorrect conclusions.

Probabilistic DSSs are applicable in many domains, among others in medicine (e.g., diagnosis, therapy planning), banking (e.g., credit authorization, fraud detection), insurance (e.g., risk analysis, fraud detection), military (e.g., target detection and prioritization, battle damage assessment, campaign planning), engineering (e.g., process control, machine and process diagnosis), and business (e.g., strategic planning, risk analysis, market analysis).

An example DSS developed using GeNIe and SMILE is the medical diagnostic system Hepar II (Onisko et al. 1999, 2000), available among the example models as `HeparII.xdsl`. The system aids physicians in diagnosis of liver disorders. The structure of the model, consisting of almost 100 variables, has been elicited from physician experts, while its numerical parameters have been learned from a database of patient cases.



The Hepar II system is equipped with a simple dedicated user interface that allows for entering various observations such as symptoms and results of medical tests and displays the probability distribution over various possible disorders in the order of most to least likely. It also rank-orders the possible observations according to their diagnostic value.



The system was used both as a diagnostic aid and a training tool.

4.12 Computational complexity

One has to remember that probabilistic inference is worst-case NP-hard, which means in practice that probabilistic models can easily reach size and complexity that is prohibitive. Complexity of probabilistic models stems from two sources: (1) exponential growth of conditional probability tables in the number of parents, and (2) connectivity of the directed graphs modeling the problem structure. While SMILE is very efficient and belongs to the fastest (if it is not the fastest!) probabilistic inference engines, it is not terribly difficult to create models that will exhaust available memory and processor time.

Our advice to users is to keep an eye on the number of parents of a node. Please note that the size of the conditional probability table in the node grows exponentially with the number of nodes and while 10 binary parents will lead to $2^{(10+1)}$=2,048 parameters, 11 binary parents will lead to $2^{(11+1)}$=4,096 parameters, and 20 binary parents will lead to $2^{(20+1)}$=2,097,152 parameters. Adding just a few more parents may exhaust your computer's memory.

There is no magic number that one should not exceed but we believe that one should slow down with adding parents to nodes when their number exceeds 15 or so. When the number of states of the nodes in question is large, this number becomes even smaller. GeNIe will issue warnings when more than 20 parents are added (this threshold can be changed through the Options dialog). GeNIe offers also a dialog that presents a list of all model nodes sorted from the highest to the lowest in-degree. We advise to make generous use of this function and tend to those nodes that have the highest in-degree. Section *Structural analysis* describes this functionality in detail.

There are several techniques for reducing the number of model parameters. One of them is the use of so-called canonical gates, such as the Noisy-MAX gates, implemented in GeNIe and SMILE. They not only reduce the number of parameters and, hence, elicitation effort but also significantly speed up inference. Another technique, described thoroughly in Bayesian network literature, is called *parent divorcing*, which amounts to introducing intermediate nodes that reduce the number of parents of a single node. We encourage users of decision-theoretic techniques to study these methods, as they will lead to increased productivity and will significantly move the boundaries of what can be modeled in practice.

Building blocks of GeNIe

5 Building blocks of GeNIe

5.1 Introduction

This section of GeNIe documentation reviews elements of GeNIe that form building blocks to its functional modules. We will often refer to the following terms, so we will define them briefly:

Property sheets

Property sheets are used to define the properties of the networks, nodes and submodels used. Each element has its own property sheet.

Menus

Menus are collections of possible actions and option switches, typically present on the top of the main program window.

Pop-up menus

Pop-up menus are menus that appear on the screen when the user right clicks on any element in GeNIe. Sometimes pop-up menus appear when clicked upon in a dialog window.

Toolbars

Toolbars are used for quick access to frequently used commands. All the commands found on the toolbar can be found in one of the menus.

Dialogs

Dialogs are usually small windows containing descriptions of features that can be interactively modified.

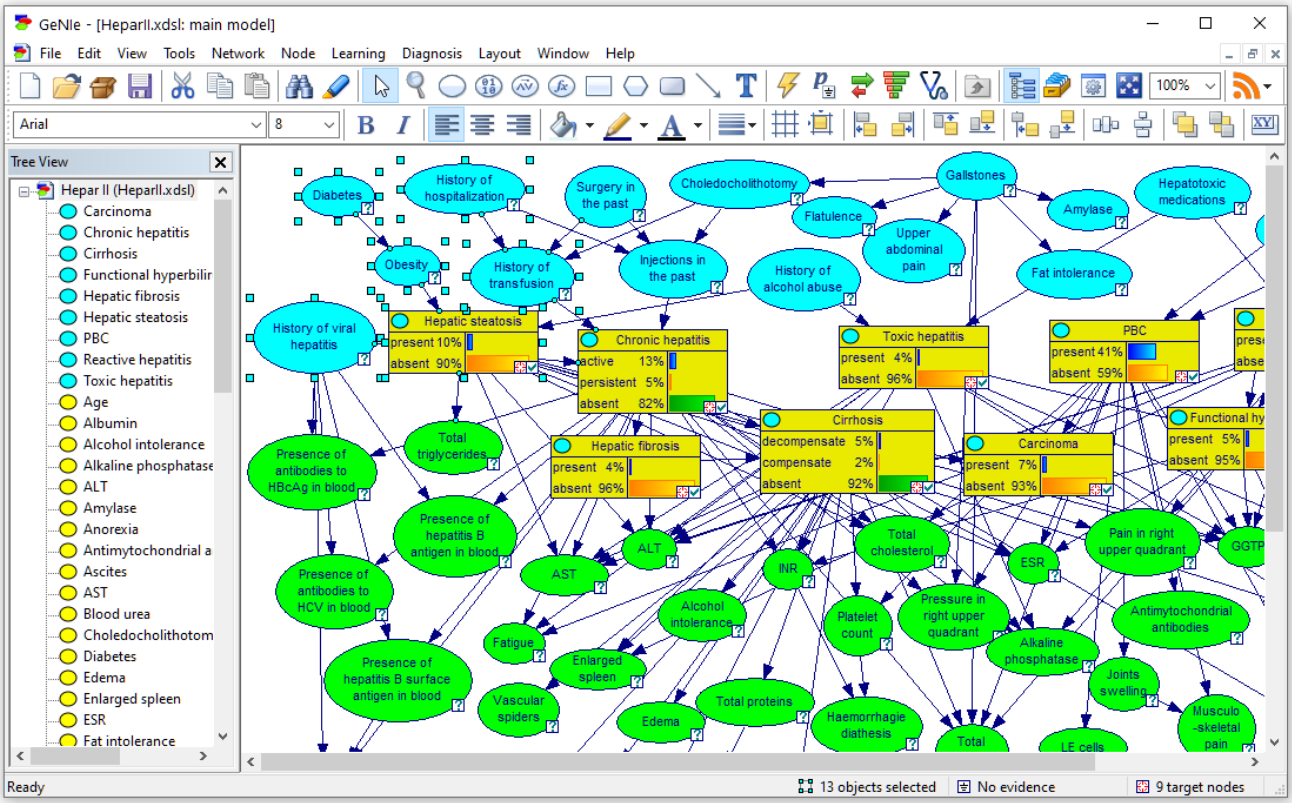
Keyboard shortcuts

Many commands in GeNIe can be executed by pressing a sequence of keys. We will report shortcuts for many of the commonly used operations but will also provide a section summarizing all GeNIe shortcuts at the end of this chapter.

5.2 GeNIe workspace

5.2.1 Introduction

GeNIe workspace is what you see and use when you work with GeNIe. Its main goal is to allow you to view the network under development in many alternative ways, which we call *views*. GeNIe window looks as follows:



The fundamental views that most users work with are the *Graph View* and the *Tree View*.

When diagnostic features of GeNIe are enabled, two additional views become available, these are the *Cost Graph View*, and the *Spreadsheet View*, both documented in the *Support for Diagnosis* section.

Status Bar displays the number of objects currently selected, the number of evidence and target nodes that are set for the active model.

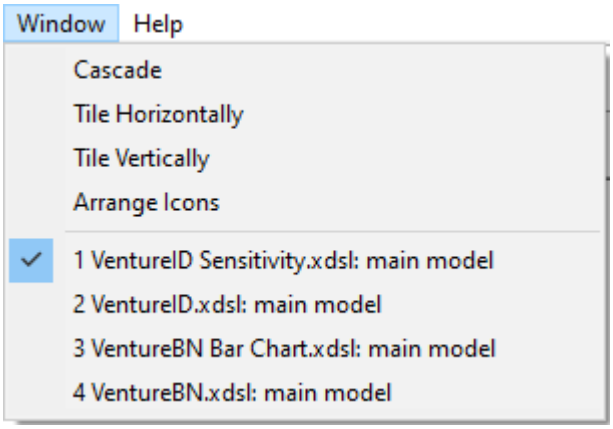
Case Manager allows users to create cases for different combinations of observed evidence.

The *Output Window* is where GeNIe will display important messages for you.

The tool bars and menus, along with the property sheets for nodes, submodels and networks are described in the *Building Blocks of GeNIe* section.

GeNIe allows for working with multiple models at the same time. Each model can have an unlimited number of arbitrarily nested submodels. Each model and submodel can be viewed simultaneously in the *Graph View* and the *Tree View*.

At any given moment, only one window in GeNIe workspace can be active. The active window can be easily recognized by both being completely in the foreground and by its distinct characteristic (such as a dark blue top bar) determined by your Windows settings. To make a window active just click on any of its elements. An alternative way of making a window active, useful when the workspace contains many windows, is by selecting its name on the *Window* menu.



The *Window* menu displays a list of all currently open windows. A check mark appears in front of the name of the active window. The user can select any of the windows in the bottom part of the Window menu to be active. To select a window, select its name and release the mouse button.

The *Window* menu offers commands that help in arranging multiple views of multiple documents in the application window:

Cascade command arranges all windows in an overlapping fashion with their *Title* bars clearly visible.

The *Tile Horizontally* and *Tile Vertically* commands arrange windows into non-overlapping tiles either horizontally or vertically (depending on the option selected), allocating equal space to each window.

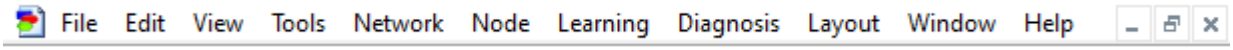
Arrange icons arranges icons of all minimized windows at the bottom of the main GeNIe window. Please note that if there is an open window that covers the bottom of the main window, then it may cover some or all of the icons and they may not be visible.

5.2.2 The menu bar

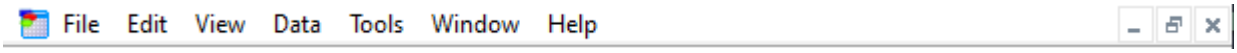
The *Menu Bar* is displayed at the top of the GeNIe window and displays menu headings. Clicking on a menu heading will open the menu and display a list of commands under that menu. You can click on a command name to choose that command. The most frequently used commands are also displayed as tool bars (collections of buttons under a common theme). The menus that are available depend on what is open in the workspace. GeNIe has four different menu bars:

(1) when no network and no data file is open,

(2) when the current window is a network (in *Graph View*),



and (3) when the current window is a data file spreadsheet.



The differences have to do with the fact that not all menus are relevant for each of the situations. Here is a brief introduction to each of the menus. Details for each of the menus are covered in various sections of this manual.

File Menu has commands for the interaction of GeNIe with external storage, such as creating a new model, opening, saving, closing, and printing a model, loading and saving data files, exporting annotations, and opening the directory in which the current model or the data file are located.

Edit Menu has commands for cutting, copying and pasting elements of the model, searching for an element, selecting, and highlighting multiple model elements.

View Menu has commands for viewing and hiding various toolbars and *Status Bar*, selecting format of labeling for nodes, and displaying the *Spreadsheet View*.

Data Menu has for viewing, cleaning up, and generally processing of data.

Tools Menu has commands for selecting the various drawing tools for drawing different type of nodes.

Network Menu has commands operating on the entire network, for example, displaying network properties, updating the model, setting and clearing evidence, decisions and target nodes, selecting the inference algorithm, commands for dynamic Bayesian networks, discretization of hybrid Bayesian networks, and several specialized algorithms and operations, such as probability of evidence, annealed MAP, value of information in influence diagrams, and sensitivity analysis. It also contains commands for calculating the strength of connections between nodes, displaying the model graph's adjacency matrix, and obfuscation of the network.

Node Menu has commands for displaying *Node properties*, changing node type, setting and clearing evidence, decisions, and manipulated node values, setting *Targets*, selecting view type of the node, locating parents and children of the node, and showing its overall connections in the model graph.

Learning Menu has commands for learning models and their parameters from data.

Diagnosis Menu has commands for using the diagnostic features of GeNIe.

Layout Menu has commands to adjust grid properties and layout options for the nodes.

Window Menu has commands to arrange the open windows in GeNIe and to switch between windows.

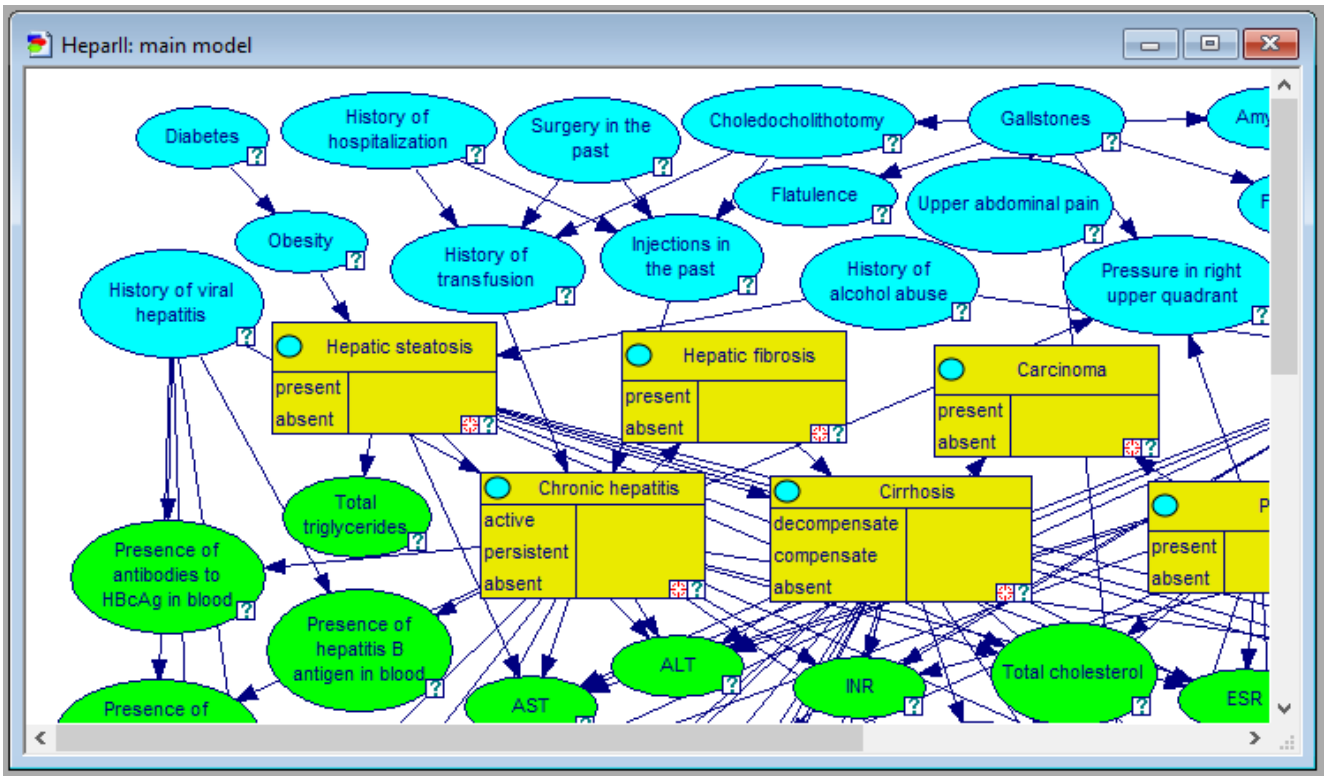
Help Menu has commands to display online help and modify help settings.

Note: *Menu items that are grayed out do not apply to the current selection or are unavailable in the current view.*

5.2.3 Graph view

The *Graph View* is the primary model view in GeNIe. It shows a directed graph in which each node represents a variable and each arc represents an influence between two nodes. It is an intuitive environment for creating and editing networks, useful in gaining insight into models by making the structure of their graphs explicit. A slightly modified version of the *Graph View* is the *Cost Graph View*, described in section *Cost of observation*.

An example of the *Graph View* is shown below:



Graph View can be enhanced dramatically by structuring the model hierarchically into submodels. Please see the section on GeNIe *submodels* to learn more about it.

The *Layout Menu* and buttons on the *Format Toolbar* can be used to change the aesthetic properties of the *Graph View*.

Commands for displaying or hiding the grid and aligning the elements in the graph can be found in the *Layout Menu*.

The *Format Toolbar* has buttons for changing the font, color and size of the labels of the nodes, and buttons for performing the aligning operations on text and on the elements of the graph.

Please see *Layout Menu* and *Format Toolbar* for more information.

Opening a Graph View window

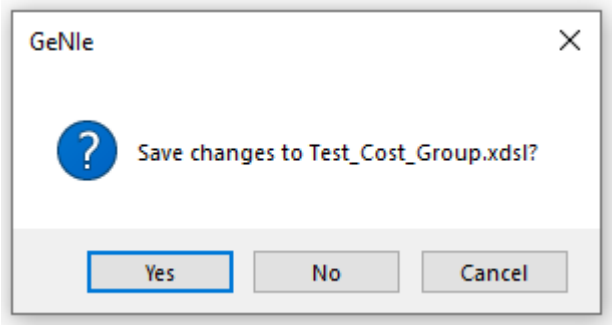
The graph view window is always open by default whenever a new model is opened or created. It is a large sheet with variables placed at user-designated locations. You can select the *Open* option from the *File Menu* to open a saved network file. You can create a new network by selecting the *New* option from the *File Menu*.

Closing a Graph View window

There are three ways in which you can close a *Graph View* window:

- By clicking on the *Close* ( )button at the top right of the *Graph View* window.
- By selecting the *Close* option in the *File Menu*.
- By selecting the *Close Network* option from the *Network Pop-up* menu in the *Tree View*.

If you close all the windows of an open network then it will result in closing the file, and if any changes have been made on the network, GeNIe will give you a warning with the dialog box shown below.



You can save the changes by clicking on the *Yes* button. Click on *Cancel* to continue working on the network.

Working with networks in the Graph View:

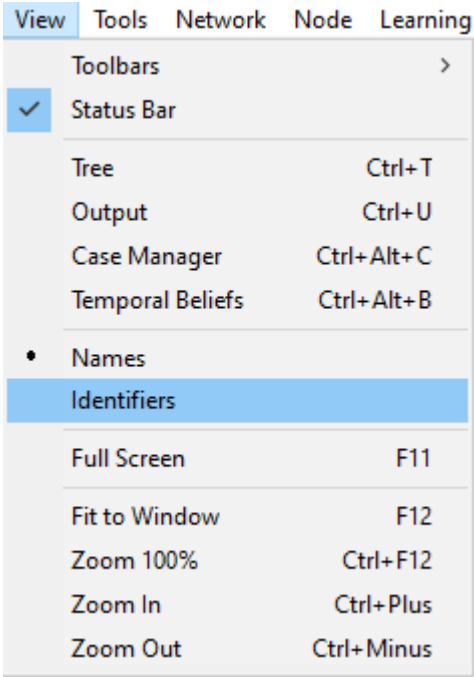
Each network is opened in a separate graph view sheet in the workspace. Double clicking on a clear area of the graph view sheet will open the *Network Property Sheet*. Right clicking on any clear area of the graph view sheet will display the *Network Pop-up* menu, which can be used to modify various properties of the network.

Working with nodes in the Graph View:

You can draw new nodes in the *Graph View* by selecting the appropriate tool from the *Tools Menu* or clicking on the appropriate button on the *Standard Toolbar*.

Double clicking on any node will open its *Node Properties Sheet*. Right clicking on the node will display the *Node Pop-up* menu. It can be used to modify the properties of the node.

By default, GeNIe displays the node names within the node icons in the *Graph View*. If you want GeNIe to display node identifiers instead, select *Identifiers* in the *View Menu*. to switch the display to identifiers.



Working with submodels in the Graph View:

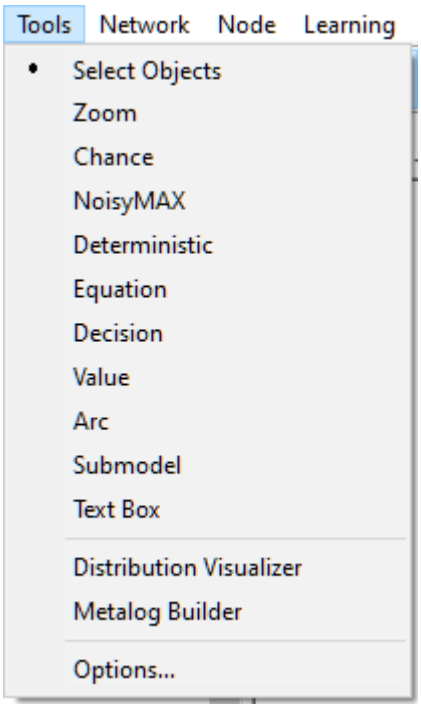
Double clicking on any submodel in the *Graph View* will open a *Graph View* for that submodel. You can go back to the main network by either minimizing or closing the submodel using the buttons on the top right of the submodel window. Right clicking on the submodel will display the *Submodel Pop-up Menu*. It can be used to modify the properties of the submodel.

Adding model elements

You can draw the following model elements in the *Graph View*:

- Nodes
- Submodels
- Arcs
- Text Boxes

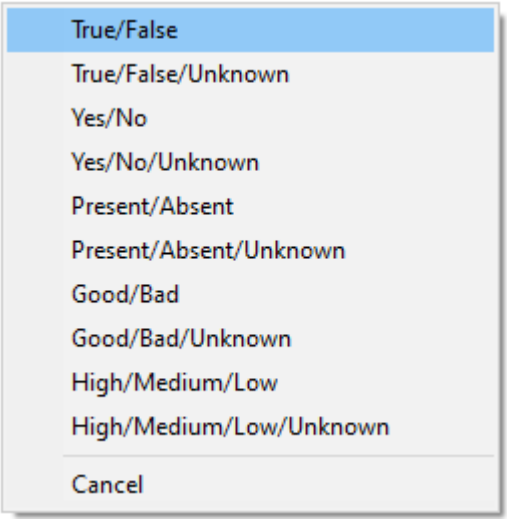
To add any model element to the *Graph View*, you have to first select a tool, either from the *Tools Menu* below



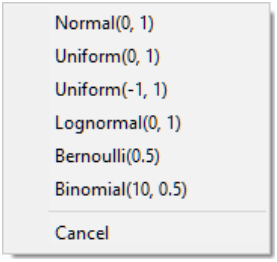
or a button from the *Standard Toolbar* below

The next step is to click on any clear area of the *Graph View*. For all elements except the arc, GeNIe will draw the icon of the element in the Graph View.

If you press and hold the *SHIFT* key on the keyboard and click the left mouse button, GeNIe will propose a small set of standard definitions. For discrete nodes, the following menu with Quick states list for the nodes will pop up:



For continuous (Equation) nodes, the following menu will pop up:



To create a node with a standard definition listed in these menus, please select the definition that is most relevant to your node and click the left mouse button. In both cases, the definition can be subsequently modified to the desired values.

To add an arc between two nodes,

1. Select the arc tool and click on the parent node.

2. Drag the mouse cursor to the child node and release the mouse button.

GeNIe will draw an arc from the parent node to the child node.

To learn more about creating nodes and arcs, See Building a Bayesian network and Building an influence diagram.

Selecting, re-sizing, and moving model elements

You need to select an element to perform an operation that is specific to it.

You can select a single element by clicking on it. The element will show tracker points (small squares around the perimeter of the selected element) that can be used to re-size it. You can re-size the element in any direction by dragging one of these points. Arcs cannot be re-sized, as GeNIe automatically draws them for you between pairs of nodes connected by arcs.

Sometimes, it may be convenient to select several nodes at a time. There are four ways of selecting nodes in groups in the *Graph View*:

- **Rectangular selection**

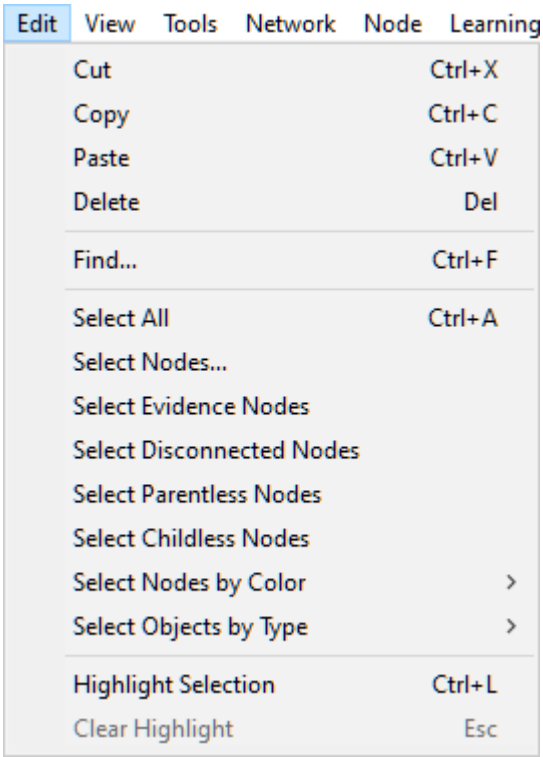
  You can select a group of nodes by clicking on an empty area of the *Graph View* and dragging a selection rectangle in any direction that you wish. Any node completely within the rectangle will be selected. When you draw a rectangular selection with *SHIFT* key pressed, the selection will be added to the existing selection. When you draw a rectangular selection with both *CTRL* and *SHIFT* pressed, the current selection status will be inverted, i.e., model elements selected will become unselected and model elements unselected will become selected.

- **Extended selection**

  Once you have an element, such as a node or a group of nodes or text boxes, selected, you can add or remove individual elements from the selection by holding the *SHIFT* key while clicking on them. This selection process acts as a toggle, i.e., nodes that are currently selected will be de-selected.

- **Group selection**

  You can select a specific group of nodes or all nodes in the current window by choosing *Select All* from the *Edit Menu*. The shortcut for this selection is *CTRL+A*.



- ***Select Nodes... dialog***

Select Nodes... dialog allows for selecting nodes based on their names or IDs. We will discuss this powerful selection tool in section *Selection of model elements*.

- ***Select Evidence Nodes***

  This command selects all nodes that have observed evidence in them. If there are no evidence nodes in the network, this choice is dimmed.

- ***Select Disconnected Nodes***

  This command selects all nodes that are disconnected from the graph, i.e., have neither parents no children. Typically, such nodes are left disconnected by mistake, so selecting and subsequently highlighting them is a good way of finding them in the model, which may be otherwise challenging in sizable models. If there are no disconnected nodes in the network, this choice is dimmed.

- ***Select Parentless Nodes***

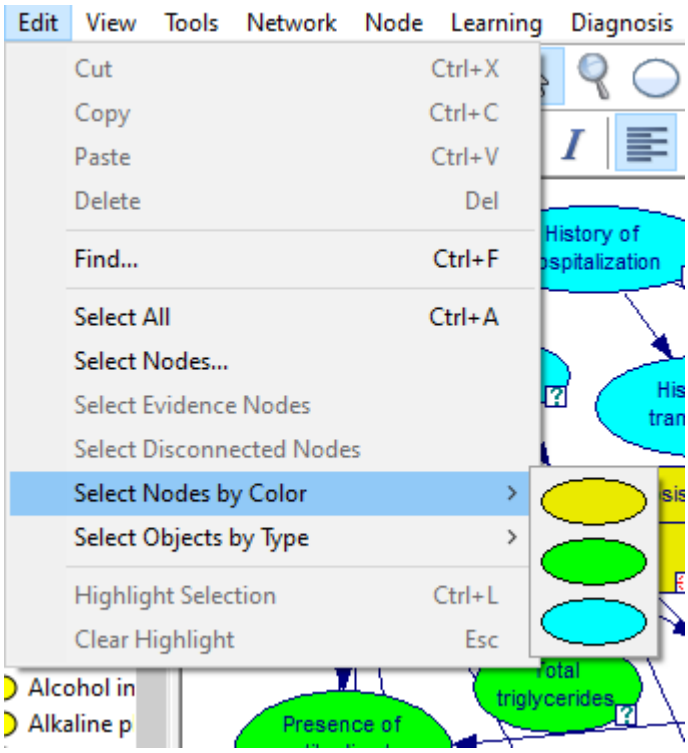
  This command selects all nodes that have no parents. This is a fast way of identifying those nodes that are exogenous to the model. This selection command works within the current sub-model window, so while a model will always have nodes without parents, there may be no such nodes in the current sub-model. If this happens, the command will be grayed-out.

- ***Select Childless Nodes***

  This command selects all nodes that have no children. This is a fast way of identifying possibly barren nodes. This selection command works within the current sub-model window, so while a model will always have nodes without children, there may be no such nodes in the current sub-model. If this happens, the command will be grayed-out.

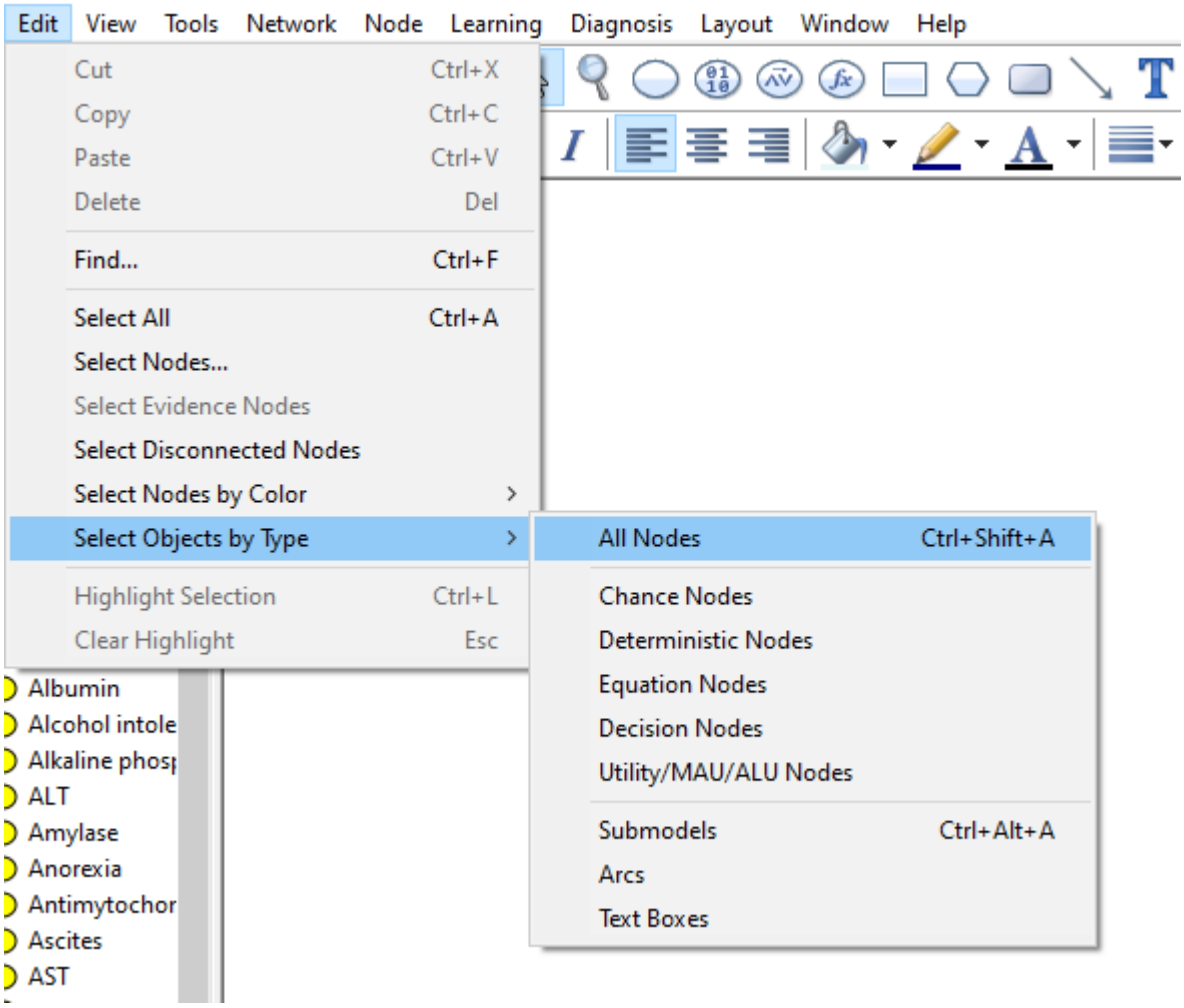
- ***Select Nodes by Color* sub-menu**

  Nodes can be selected based on their colors. This can be done by selecting a color listed in the *Select Nodes by Color* sub-menu shown below. Colors will be listed in the sub-menu only if the model contains nodes in these colors.



- ***Select Objects by Type* sub-menu**

  You can select nodes of each type (*Chance*, *Deterministic*, *Decision*, and *Utility* nodes), submodels, arcs, and text boxes in the entire model by choosing the appropriate option from the *Select Objects by Type* sub-menu shown below.

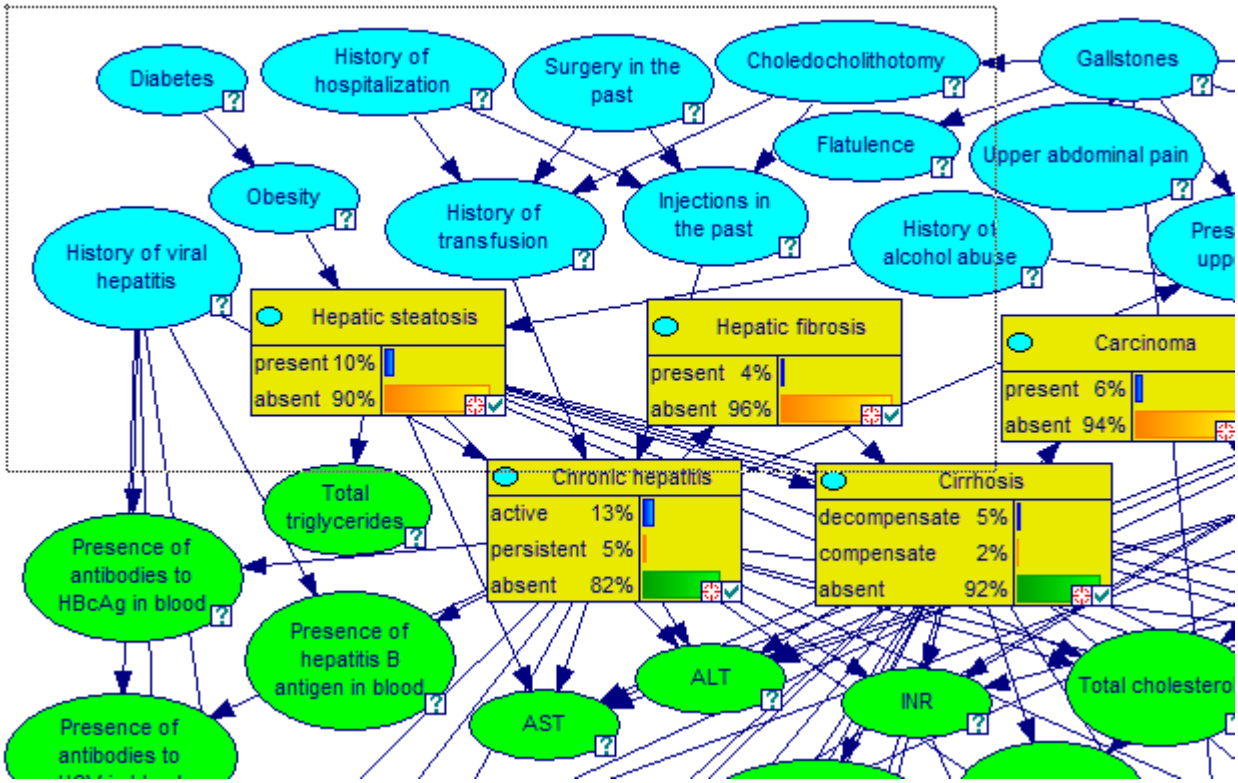


You can move a node or a group of nodes to another location by dragging it. When you press the *SHIFT* key (you need to do it after clicking on the node or the selection, as *SHIFT*-Click has the meaning of adding an object to the selection), the moving is limited to horizontal and vertical directions.

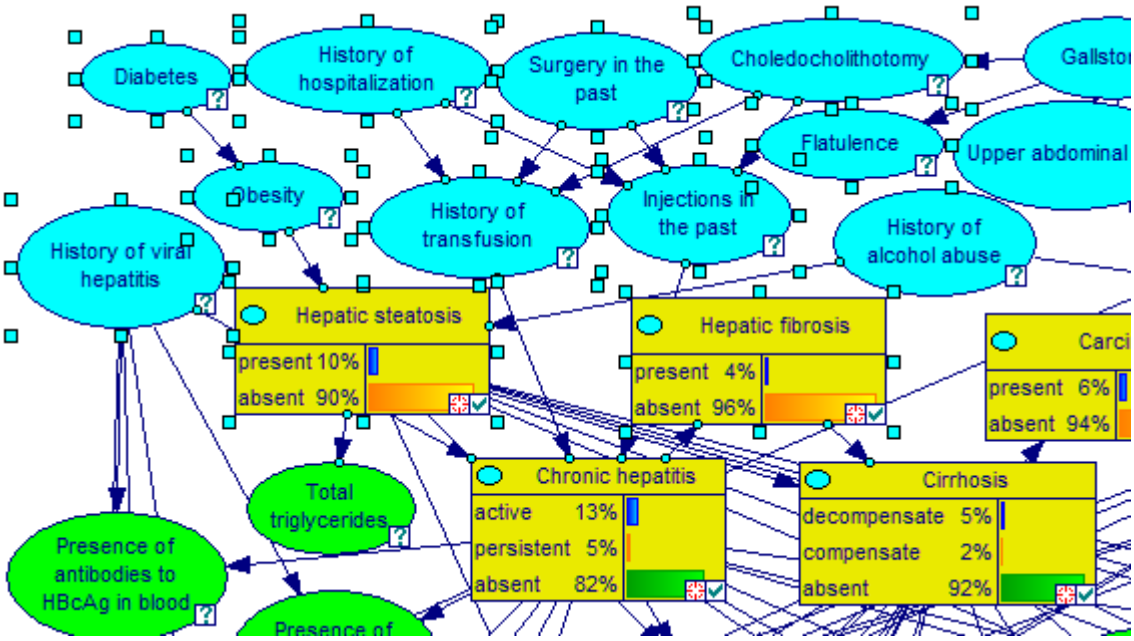
A node or a set of nodes can be also moved between submodels. To move a node to a different submodel, drag and drop it into a submodel icon or into a submodel window.

Highlighting selected model elements

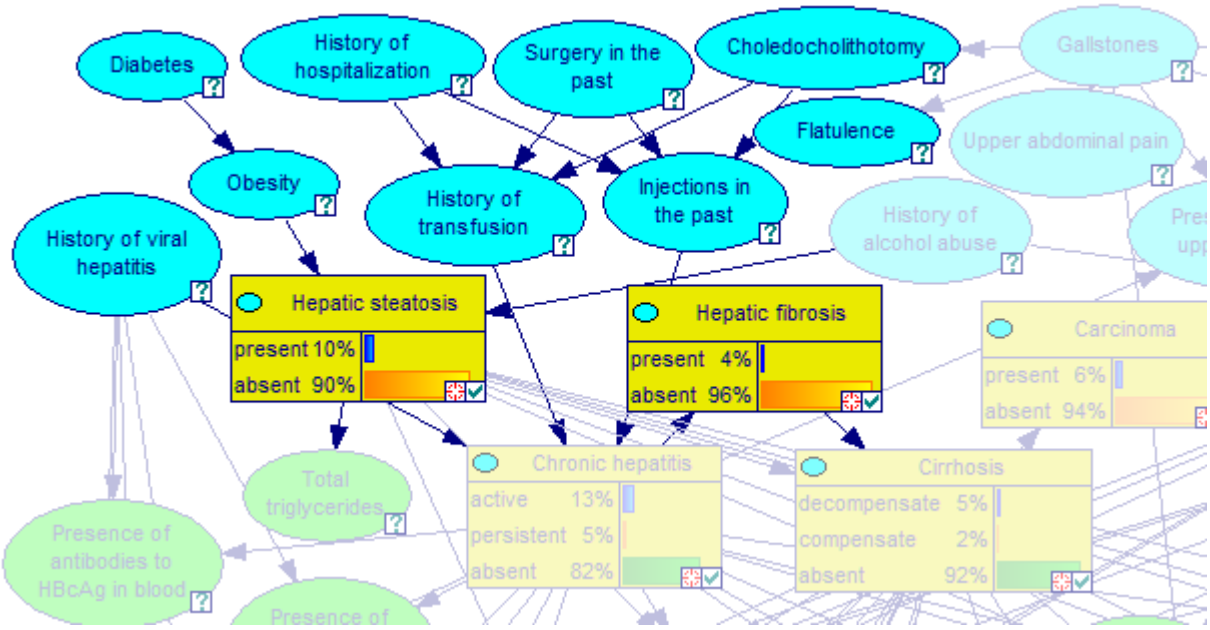
GeNIe allows for highlighting the selected model elements (nodes, arcs, submodels, etc.) through the command *Highlight Selection* in the *Edit Menu* (short-cut *CTRL-L*) or through the *Highlight* button (🖊) on the main toolbar. The following screen shot shows a rectangular selection of model elements in *Graph View*,



This results in selecting a group of nodes and arcs



When these elements are highlighted, the *Graph View* changes to the following



Pressing the *ESC* button or selecting *Clear Highlight* from the *Edit Menu* clears the selection.

Extending the workspace

GeNIe is somewhat restrictive in how far the *Graph View* space stretches in every direction and there is typically not too much space beyond the borders. If you want to move model elements in any direction, please bring them close to the border and this will force GeNIe to move the border.

Deleting model elements

To delete an element or a group of elements, select them and press the *Delete* key on the keyboard.

Deleting a node deletes all its incoming and outgoing arcs.

Deleting an arc has important implications for the nodes to which they point. They are no longer indexed by the nodes from which the arcs were coming. GeNIe will reduce the dimension of the conditional probability table and it makes a good faith attempt to preserve as much as possible, removing part of the table. You should always reexamine the conditional probability table to check whether the new table is what you intended it to be.

Copying, cutting, and pasting model elements

Model elements or group of elements can be copied or cut into the *Windows Clipboard*, and subsequently pasted into the same or a different *Graph View* window or into another application. To invoke any of these commands, please select them from the *Edit Menu* or from the *Standard Toolbar* (shown above, buttons , , and , respectively). Nodes pasted into the same window will preserve their incoming arcs but will lose their outgoing arcs. The reason for this is simple - the nodes need their parents in order to preserve their definition,

which is typically conditional on its parents. On the other hand, preserving the outgoing arcs would mean that their child nodes would have double set of parents, i.e., the original nodes and their copies.

Clipboard supports multiple formats. GeNIe is capable of storing data on the clipboard in four different formats:

1. Native GeNIe format for cut, copy, and paste operations between models

2. Standard unformatted text and unformatted Unicode text, for example names of selected nodes and comments

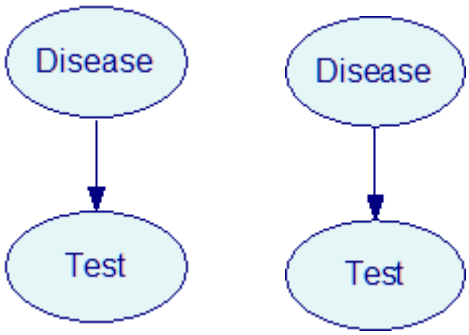
3. Standard bitmap and device independent bitmap, which are used for selected objects that have a graphical component

4. Picture (*metafile* or *enhanced metafile*), used only for objects selected from the *Graph View*.

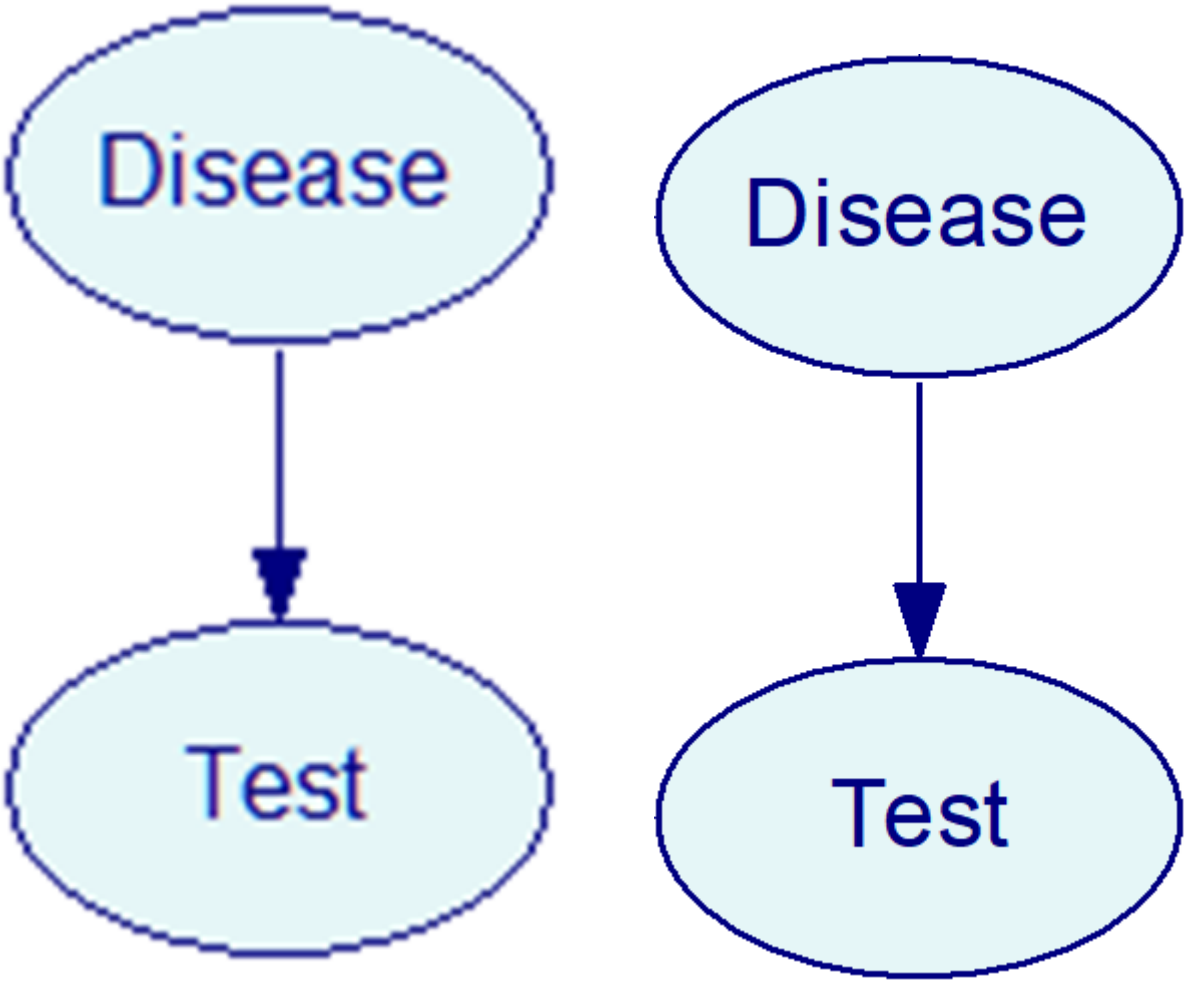
Each time you invoke *Copy* or *Cut* in the *Graph View*, data in all four formats are sent to the *Clipboard*. Each external application has its default paste format. Plain *Paste* command does thus different things in different programs - it pastes data as text into *Word* by default but may paste a bitmap image in *Paint*. Should you want to change this default format, all you need to do is select *Paste special* from that application's *Edit Menu* (e.g., in *Word* or *PowerPoint*). This should bring the dialog box with all format names.

To copy a complete model as a bitmap picture, first select all elements in the model using the *CTRL+A* shortcut. Then use the *CTRL+C* shortcut to copy the model to the clipboard. Subsequently, open the program in which you want to paste the model image (e.g., *Adobe Photoshop* or *MS Word*) and use *Paste special* and then *Bitmap* or *Picture* in *Word*. If you use *Photo Editor* or any graphics editing program, *Paste* will paste the model image by default.

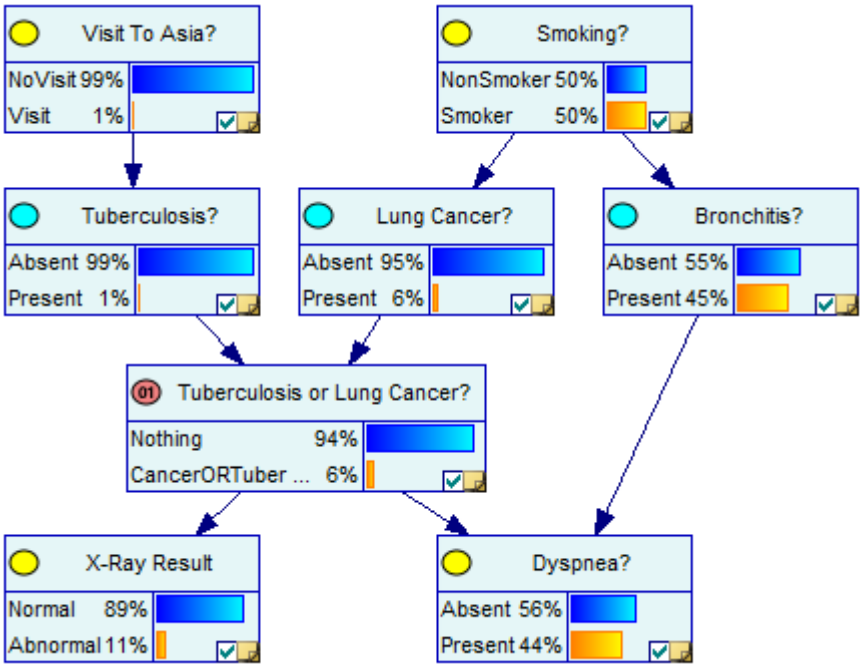
The *Picture* format contains the image content as vectors, which allows for better scaling and (in some programs) additional effects. Enlarging *Picture* objects leads to minimal loss of quality. Compare the following fragment of the tutorial network pasted as a *Bitmap* (left) and a *Picture* (right) objects:



There is no visible difference between the two. However, when we enlarge the two images by 300%, we will notice that the *Bitmap* object yields a much worse quality image than the *Picture* object (right):

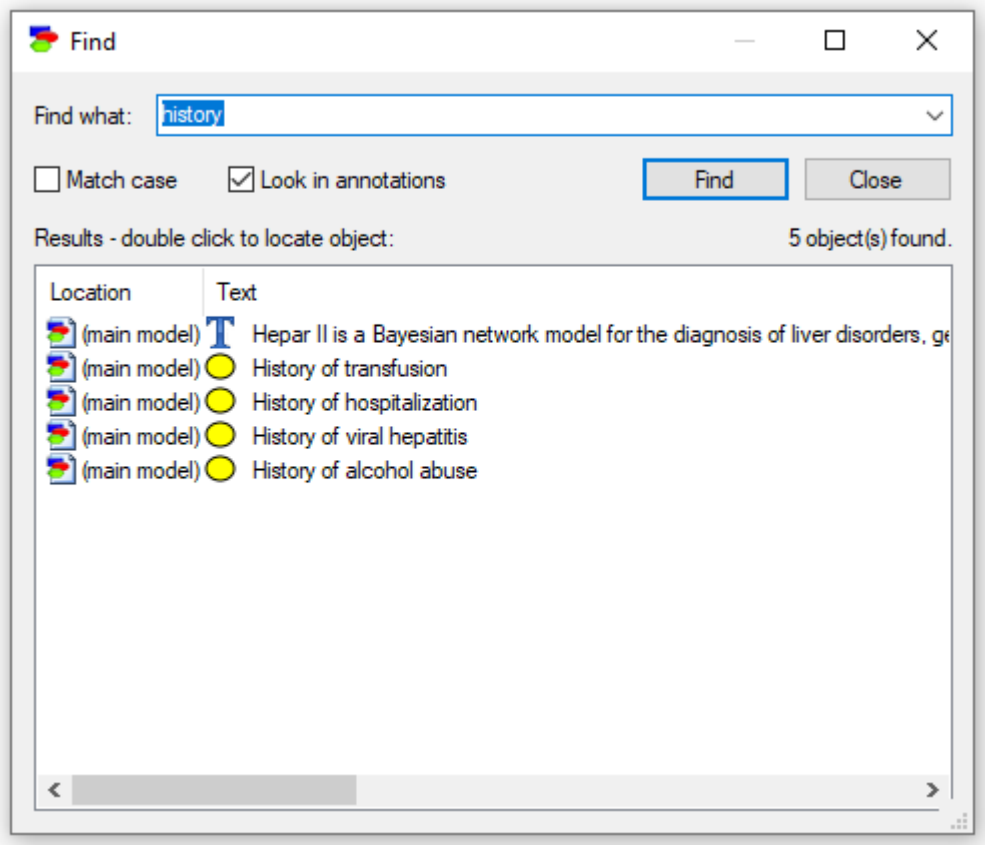


Furthermore, the Picture object can be processed further, for example enhanced by means of *Picture Tools* in *Word*:



Text-based search for model elements (*Find* command)

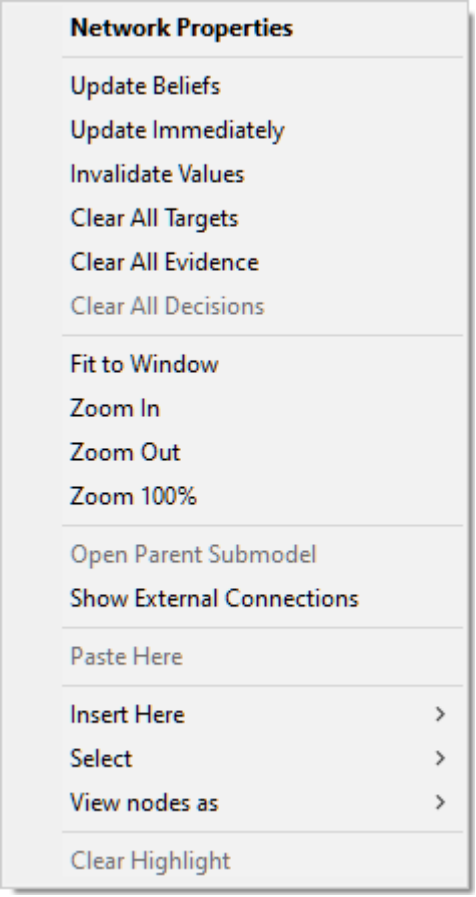
Find ( ) button from the toolbar or selecting *Find* from the *Edit Menu* (shortcut *CTRL-F*) invokes the following dialog, which allows for finding model elements, such as nodes, through text search



The *Find* dialog allows for finding the text string specified in the *Find what* box in the names and identifiers of all elements of the models and submodels. It will also search within annotations if the *Look in annotations* check box is checked. *Match case* flag allows for additional customization of the search. Pressing the *Find* button starts the search. If any matches are found, they are displayed in the dialog box and the *Find* button changes into the *Locate* button. Selecting one of the results and pressing the *Locate* button locates the selected node in the *Graph View*, centers it, and flashes three times. You can also locate a node by double clicking on one of the results.

Network Pop-up menu for *Graph View*

The *Network Popup Menu* for the *Graph View* can be accessed by right clicking on any clear area of the *Graph View*. Some of the options might be disabled depending on the properties of the network selected.



Most of the commands found here can be also invoked from the *Network Menu*.

Network Properties (the default operation) opens the *Network Properties* sheet for the network.

Update Beliefs runs the selected algorithm on the model and brings the values of each of the variables of interest up to date. The *Update Beliefs* command can be also executed by pressing the *Update* ( ) tool from the *Standard Toolbar*. The algorithm to be applied can be selected from the *Network Menu*. For more information on various inference algorithms supported by GeNIe, see *Inference algorithms* section.

Update Immediately is a flag, which when set invokes Bayesian inference as soon as any change happens to the model. It is convenient to have it turned off as we build a model (to avoid computation when the model is incomplete and it does not make much sense yet) and turned on when the model is ready.

Invalidate Values is the same as the *Invalidate Values* command in *Network Menu* - it removes all computation results from the network.

Clear All Targets/Evidence/Decisions are the same as the corresponding commands in the *Network Menu*.

Fit to Window makes the network as large or as small as it takes to fit entirely in the *Graph View*.

Zoom In zooms into the network. Every application of this command increases the zoom by 25%. The current zoom percentage is displayed on the top right of the *Standard Toolbar*. A similar effect can be obtained by using the zoom tool from the *Standard Toolbar* or the *Tools Menu*.

Zoom Out is the opposite of the *Zoom In* and it zooms out of the network. Every application of this command decreases the zoom by 25%. The current zoom percentage is displayed on the top right of the *Standard Toolbar*.

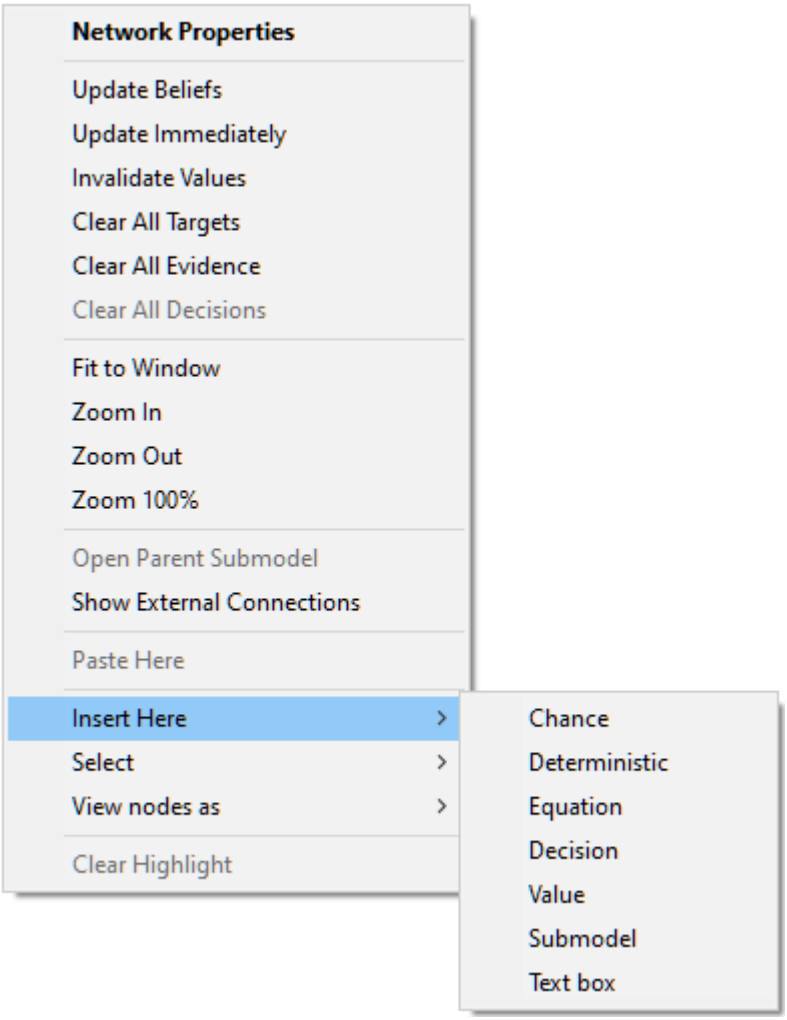
Zoom 100% brings the model displayed in the *Graph View* to its original size.

Open Parent Submodel is enabled only if the current network in the *Graph View* is a submodel of another network. The result of this command is opening the *Graph View* window displaying the parent submodel.

Show External Connections is a structure analysis tools described in the *Structural Analysis* section.

Paste Here pastes the contents of the clipboard onto the *Graph View* into the exact position of the mouse click that invoked the pop-up menu. This choice will be active only if the clipboard has data that have been entered using the *Cut* or *Copy* command within GeNIe. You cannot *Cut* or *Copy* items from other programs into GeNIe *Graph View*. You can *Cut* or *Copy* nodes between two running instances of GeNIe.

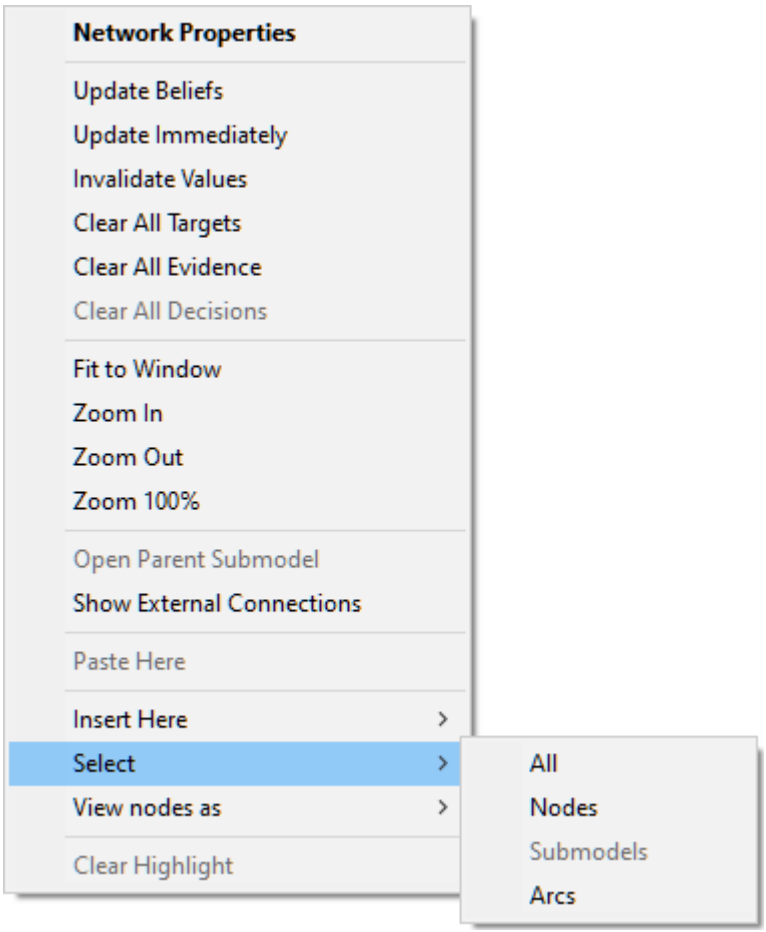
Insert Here submenu



The *Insert Here* submenu contains a list of all elements that can be drawn in the *Graph View*. Select any of the items on the list to place that item at the current cursor position. See *Components of GeNIe models* section for more information on each item.

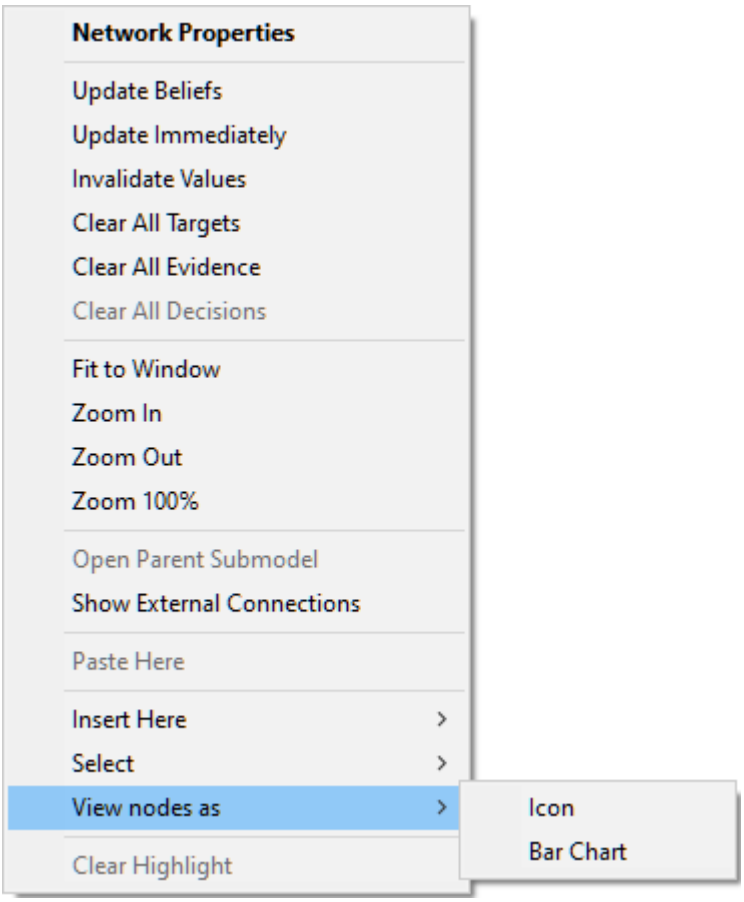
Select submenu

Selection of items enables certain operations to be performed on them without affecting other items that are not selected. Some options in GeNIe will not be enabled unless some item has been selected.



The *Select* submenu contains following options: *All* (select all items in the *Graph View*), *Nodes* (select all nodes in the *Graph View*), *Submodels* (select all submodels in the *Graph View*; dimmed if the current *Graph View* does not contain any submodels), and *Arcs* (select all arcs in the *Graph View*).

***View nodes as* submenu**

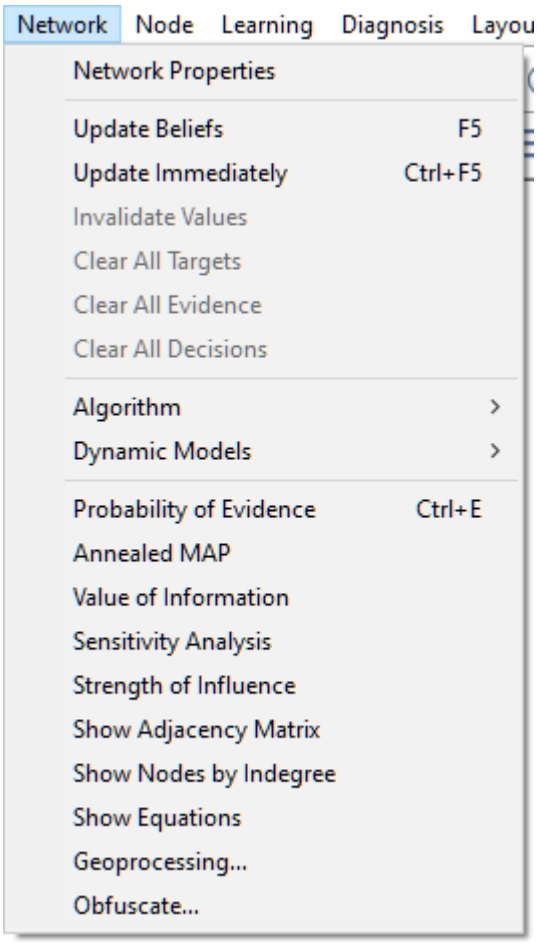


The *View nodes as* submenu is used to select how the nodes should be displayed in the *Graph View*. It is similar to the *View As* submenu in the *Node Menu* but because right-clicking on the background of the model deselects everything, it always applies to all nodes rather than to the possibly previously selected nodes.

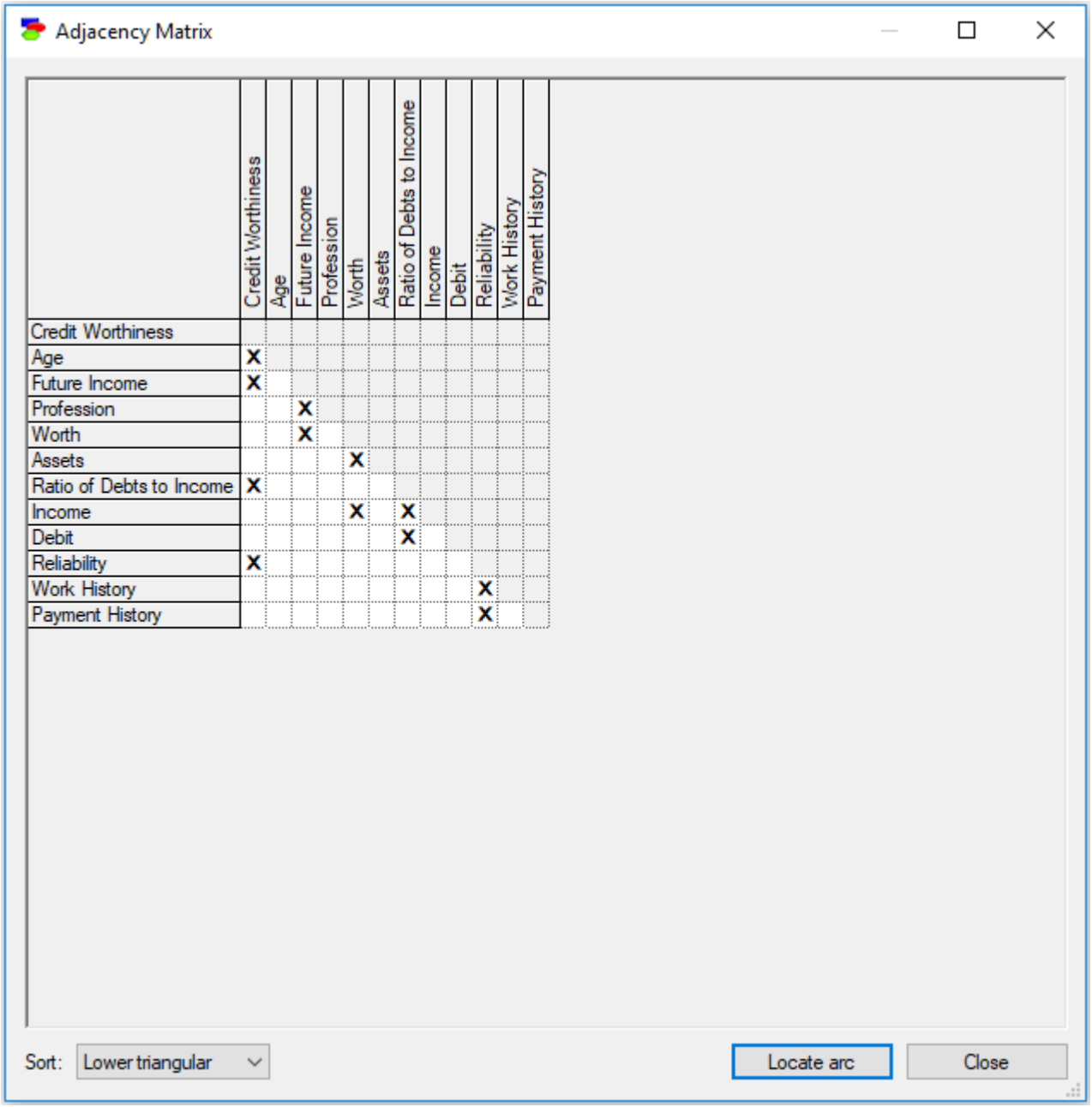
Clear Highlight clears highlighting of model elements (dimmed if no model elements are currently highlighted).

Adjacency matrix

For any directed graph of a Bayesian network, it is possible to view its adjacency matrix, which is a compact way of viewing the graph's dependences. To invoke the adjacency matrix, please select *Show Adjacency Matrix* from the *Network Menu*.

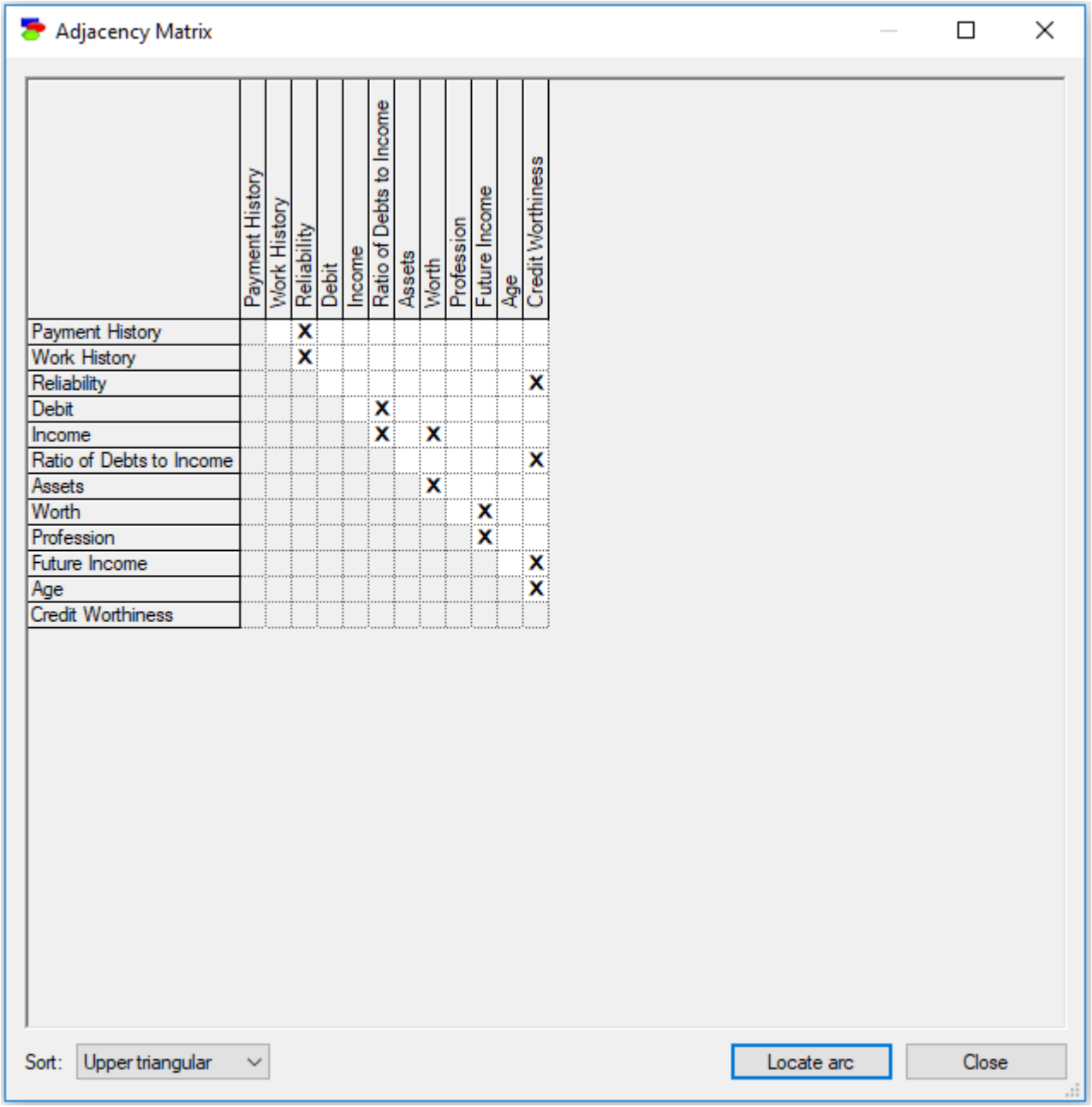


Adjacency matrix is a square matrix offering an alternative representation of a graph. The elements of the matrix indicate whether pairs of nodes are adjacent or not in the graph. for a directed graph, element (i,j) of the adjacency matrix represents a directed edge between nodes i and j. When the directed graph is acyclic (as Bayesian network graphs are), the adjacency matrix is triangular.



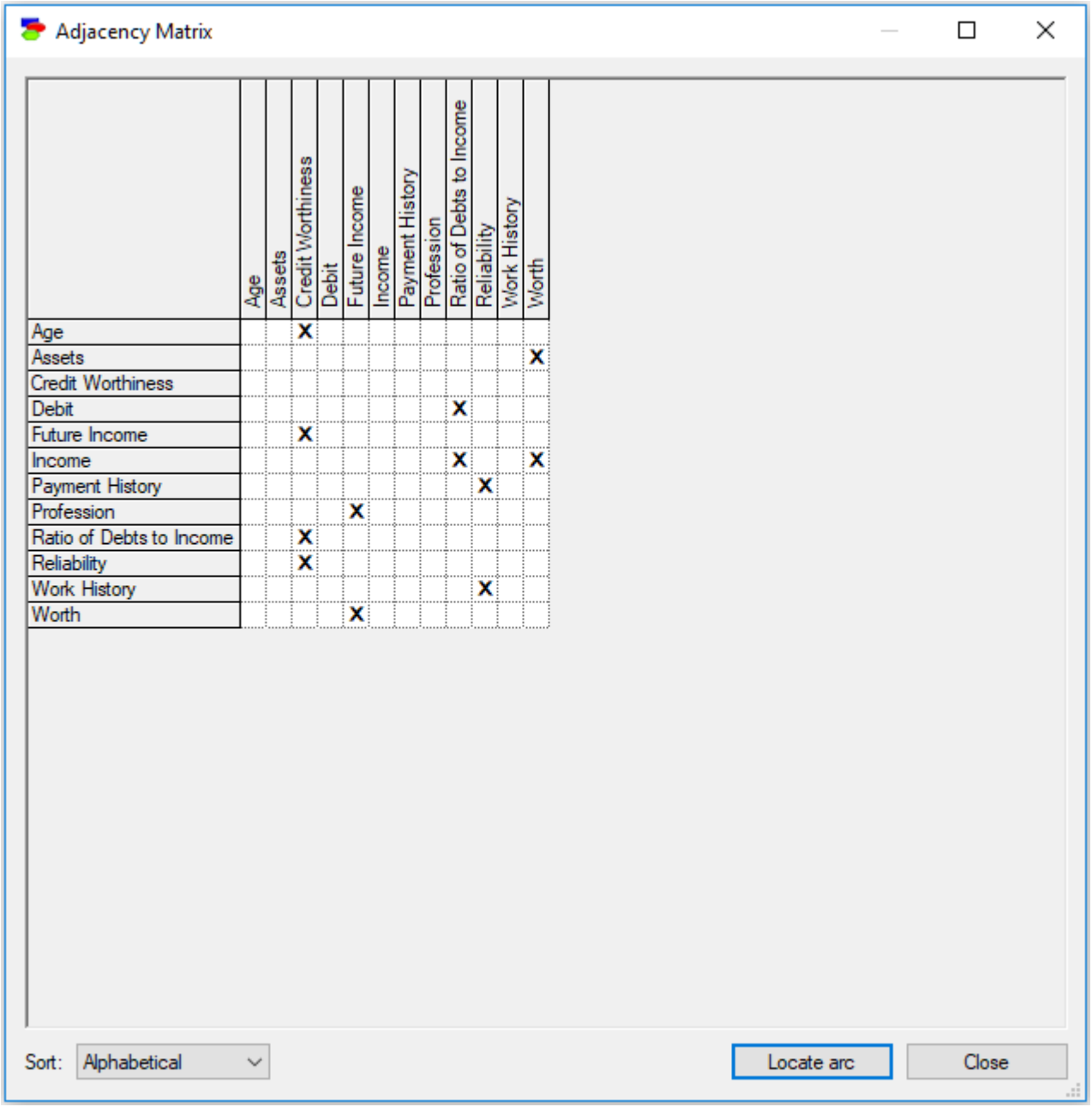
	Credit Worthiness	Age	Future Income	Profession	Worth	Assets	Ratio of Debts to Income	Income	Debit	Reliability	Work History	Payment History
Credit Worthiness												
Age	X											
Future Income	X											
Profession			X									
Worth			X									
Assets					X							
Ratio of Debts to Income	X											
Income					X		X					
Debit							X					
Reliability	X											
Work History										X		
Payment History										X		

The adjacency matrix can be viewed in three different modes. The default is the lower-triangular view (see above). The two alternative views are upper-triangular



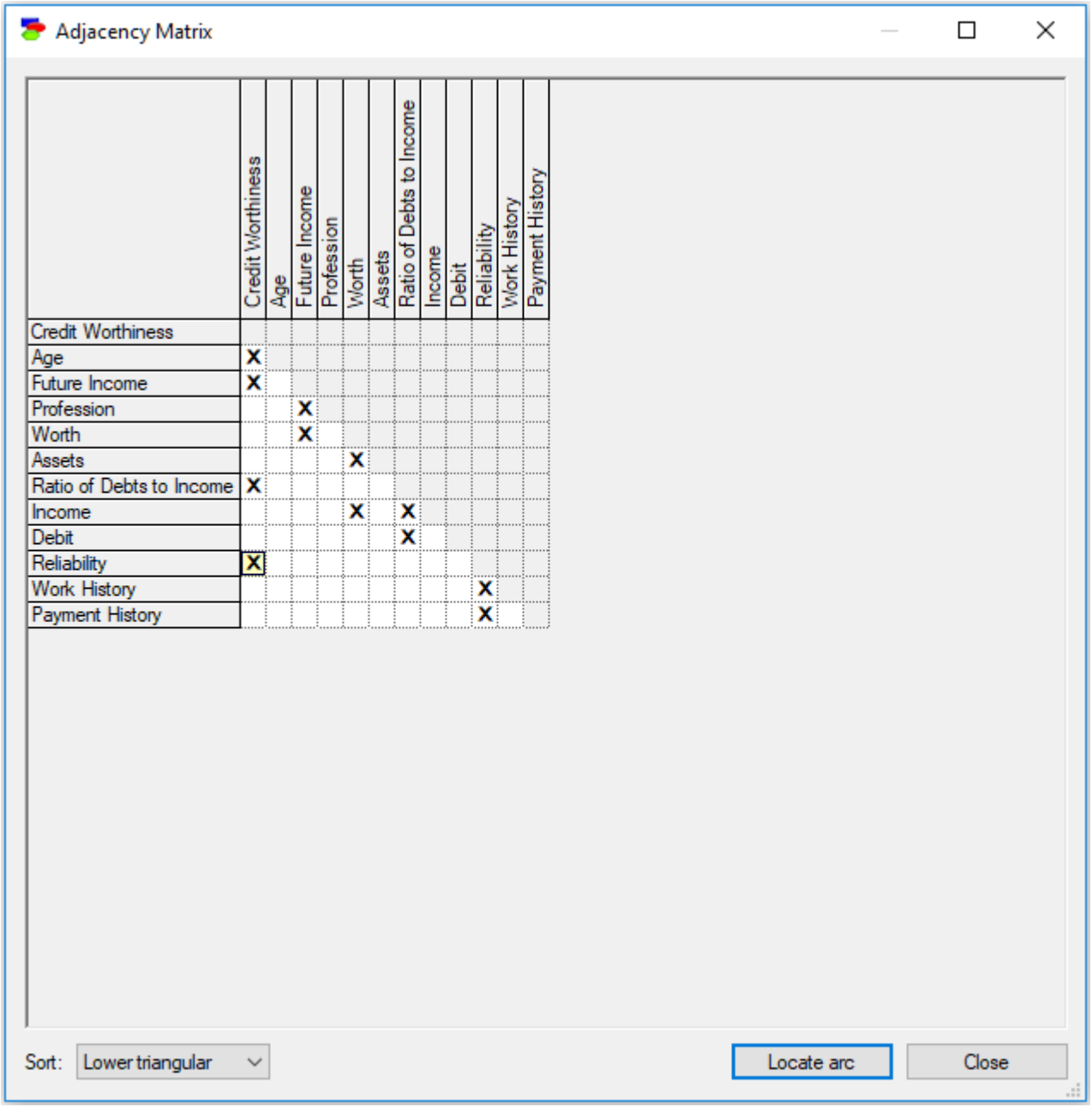
	Payment History	Work History	Reliability	Debit	Income	Ratio of Debts to Income	Assets	Worth	Profession	Future Income	Age	Credit Worthiness
Payment History			X									
Work History			X									
Reliability												X
Debit						X						
Income						X		X				
Ratio of Debts to Income												X
Assets								X				
Worth										X		
Profession										X		
Future Income												X
Age												X
Credit Worthiness												

and alphabetical

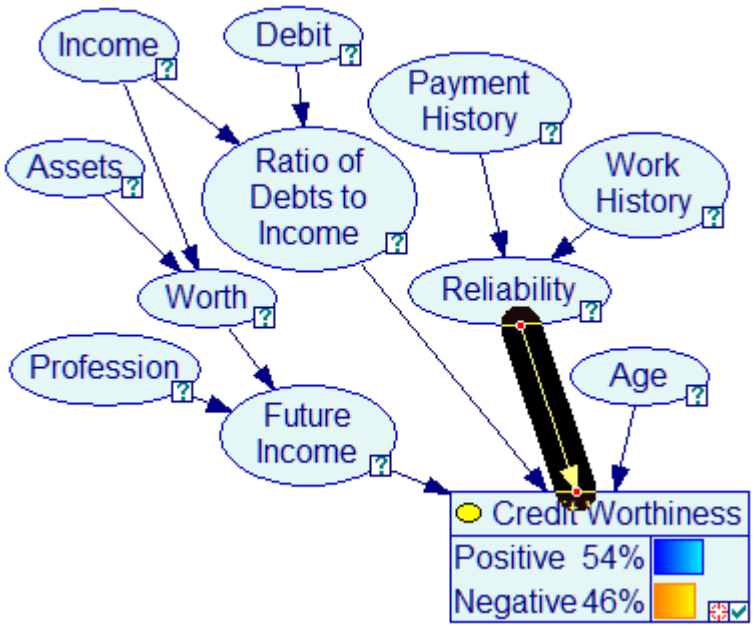


	Age	Assets	Credit Worthiness	Debit	Future Income	Income	Payment History	Profession	Ratio of Debts to Income	Reliability	Work History	Worth
Age			X									
Assets												X
Credit Worthiness												
Debit									X			
Future Income			X									
Income									X			X
Payment History										X		
Profession					X							
Ratio of Debts to Income			X									
Reliability			X									
Work History										X		
Worth					X							

Every X in the adjacency matrix represents an edge in the directed graph of a Bayesian network. Selecting an X and pressing on the *Locate arc* button shows the arc in the *Graph view* window.



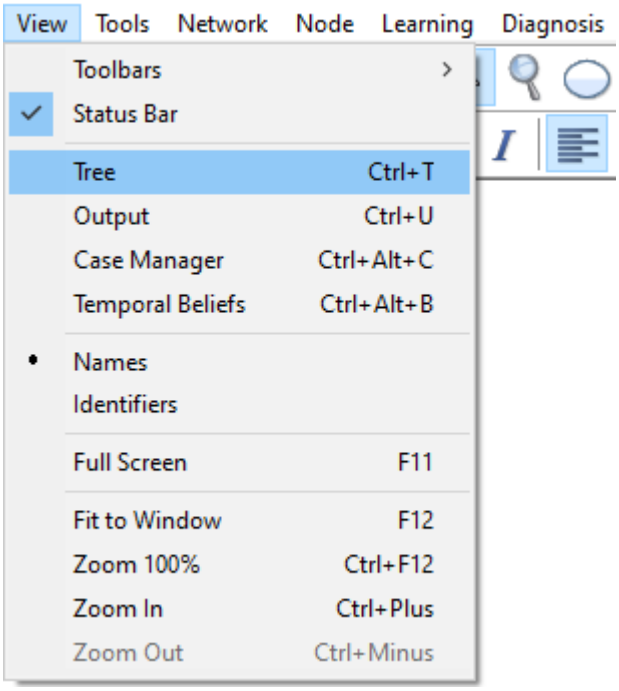
The X selected in the above picture corresponds to an arc from the node *Reliability* to the node *Credit Worthiness*



5.2.4 Tree view

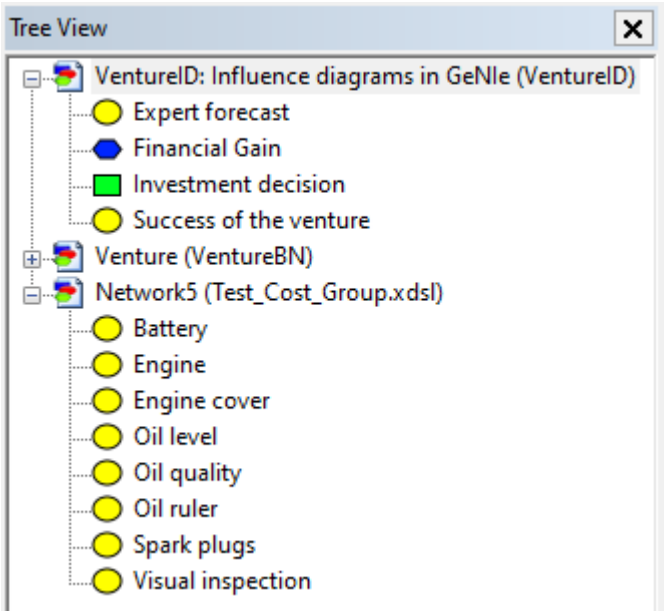
GeNIe provides an alternative method of model navigation known as *Tree View*. The *Tree View* in GeNIe is very similar to Windows tree view. It shows a hierarchical, alphabetically sorted list of all networks currently open, and all nodes in each of the networks. Most operations available in the *Graph View* can be also performed in the *Tree View*. Whatever changes are made in the *Tree View*, they are reflected immediately in the *Graph View*. The *Tree View* can be also used to navigate in the *Graph View*, for example to open submodel windows. Another important feature of the *Tree View* is that you can drag and drop nodes between different submodels and networks. We will illustrate the basic elements of *Tree View* functionality in this section.

Tree View can be displayed or hidden by checking or un-checking the *Tree* option in the *View Menu*. You can also use the keyboard shortcut *CTRL+T* to toggle *Tree View* display.



The *Tree View* panel can be detached from its position and placed anywhere on the screen by dragging it using its title bar. It snaps back into place if dragged close to the left or right border of the GeNIe window.

Shown below is a typical *Tree View* panel.



The *Tree View* above shows three networks, *TestCostGroup*, *VentureID* and *VentureBN*. A network or a submodel can be expanded or collapsed by clicking on ⊞ or ⊟ beside its name. *VentureBN* is not expanded. (hence, ⊞ is displayed beside it). *TestCostGroup* and *VentureID* are fully expanded (hence, ⊟ is displayed beside

them). Nodes that are part of the networks are listed within each of the trees. The shapes and colors of the icons indicate node types.

Note : *Double clicking on the name of a network or submodel will also expand or collapse it.*

Working with networks, submodels, and nodes in the Tree View

Right clicking on the network name, node name, or submodel name will open the corresponding *Network Pop-up* menu, *Node Pop-up* menu or *Submodel Pop-up* menu. You can use these menus to change properties of the network, node, or submodel. Follow the links to each of the menus for more information on how to performs these operations.

Moving nodes between networks and submodels

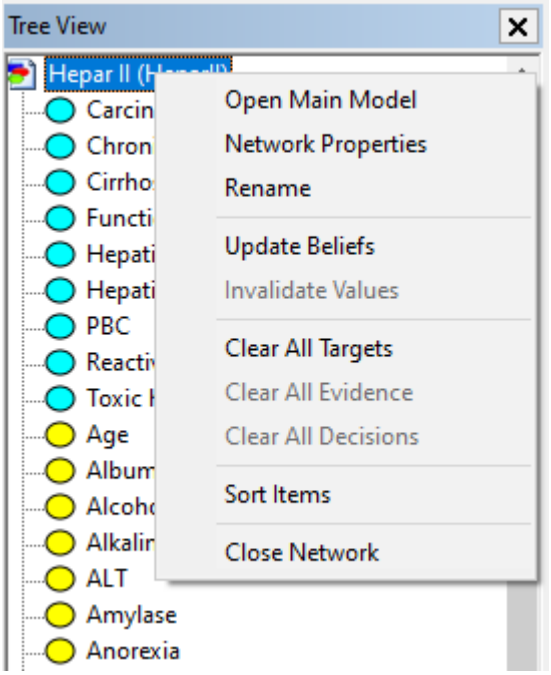
You can select any number of nodes and submodels in the *Tree View* and drag and drop them into any model or submodel in the *Tree View* or the *Graph View*. You can perform the drag and drop operations between different networks.

Note: *If the nodes are being dropped in a submodel which is part of the same network, then the nodes are* ***moved*** *to their new location.*

If the nodes are being dropped in a different network or a submodel in a different network, then the nodes are ***copied*** *to their new location.*

Network Pop-up menu in Tree View

The Network Pop-up menu in the *Tree View* can be invoked by right clicking on the network name in the *Tree View*.



Most of the choices are the same as in the *Network Pop-up* menu in the *Graph View*.

Open Main Model opens the main network in the *Graph View*.

Rename allows for changing the name of the network interactively.

Sort items causes the list of element names under the network to be sorted in alphabetical order.

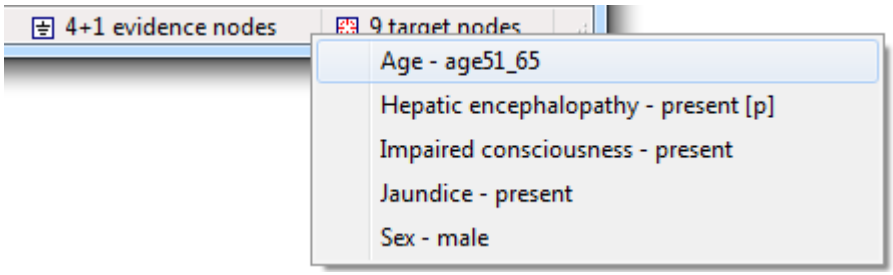
Close Network closes the network file. If any changes have been made to the network, then GeNIe will warn you that you may lose the changes. You can also close the network by clicking on the *Close* button at the top right of the *Graph View* window or selecting *Close* option from the *File Menu*.

5.2.5 Status bar

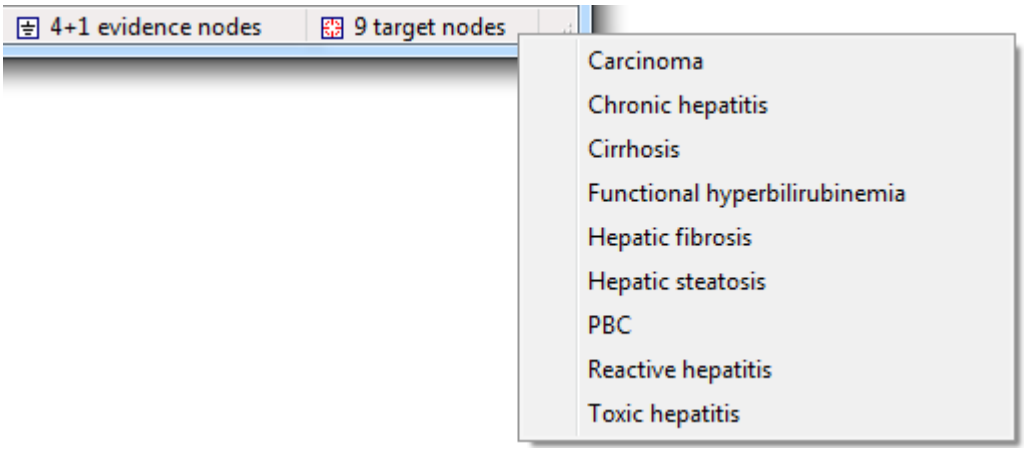
The *Status Bar* is a horizontal bar located at the very bottom of the main GeNIe window. The *Status Bar* shows a short description of the command to be executed by the selected menu item or a tool on a toolbar on the left side and lists the number of objects currently selected, the number of evidence and target nodes present in the network on the right side.

The number of selected objects is useful in some operations. For example, we might want to select the Markov blanket of a node through the *Show Connections* dialog and then subsequently verify its size by glancing at the *Status Bar*.

If there are any evidence or target nodes (see *Relevance reasoning* section for how target nodes are used) set in the network, and any of the observed evidence propagates to other nodes, it will be indicated in the *Status Bar* as shown in the figure above. The text on the Status Bar *4+1 evidence nodes* indicates that there are four observations and one propagated evidence (i.e., evidence implied by the observations). There are nine target nodes in the current network. The list of evidence nodes can be displayed by right-clicking on the text:

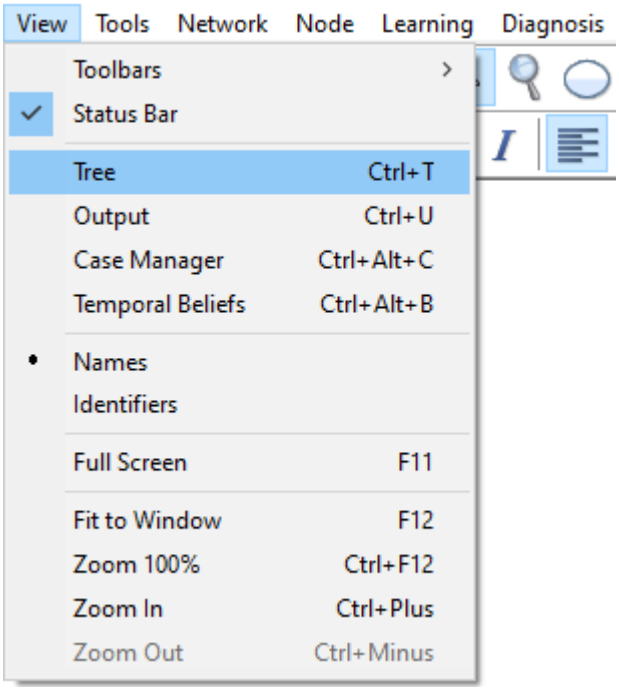


So can the list of target nodes:



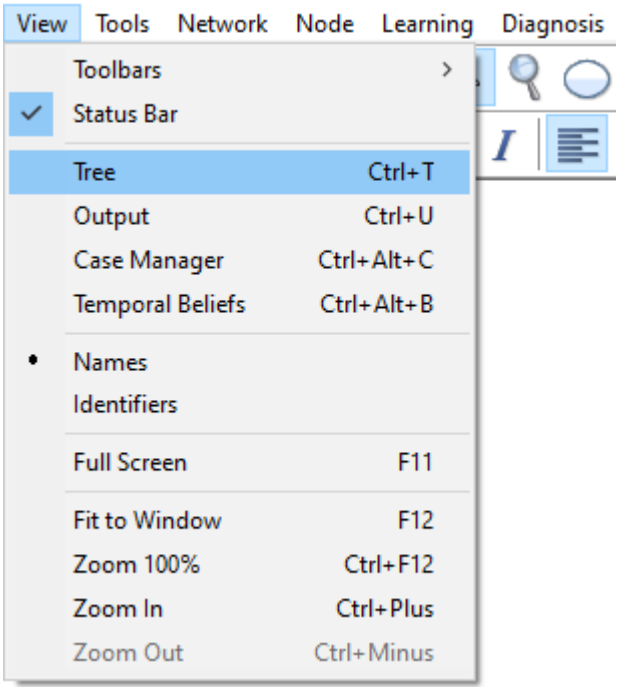
Any propagated evidence will be listed with a suffix *[p]*. To display only real evidence, right-click on the *Status Bar* when holding *CTRL* key. To display only propagated evidence, right-click on the *Status Bar* when holding the *SHIFT* key. Clicking a node name on the list of evidence or target nodes will locate the node in the *Graph View*.

Status Bar can be switched on and off by selecting or deselecting the *Status Bar* option from the *View Menu*. A check mark appears next to the menu item when the *Status Bar* is displayed.

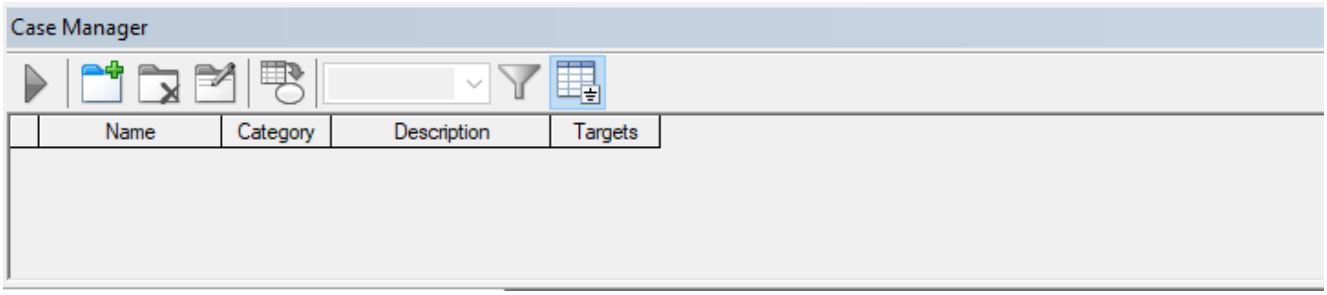


5.2.6 Case manager

GeNIe includes a *Case Manager* window that allows users to save a partial or a complete session as a case and retrieve this case at a later time. Cases are saved alongside the model, so when the model is loaded at a later time, all cases are going to be available. *Case Manager* window can be opened through the *View Menu* or by pressing the *Toggle Case Manager window* ( ) button:

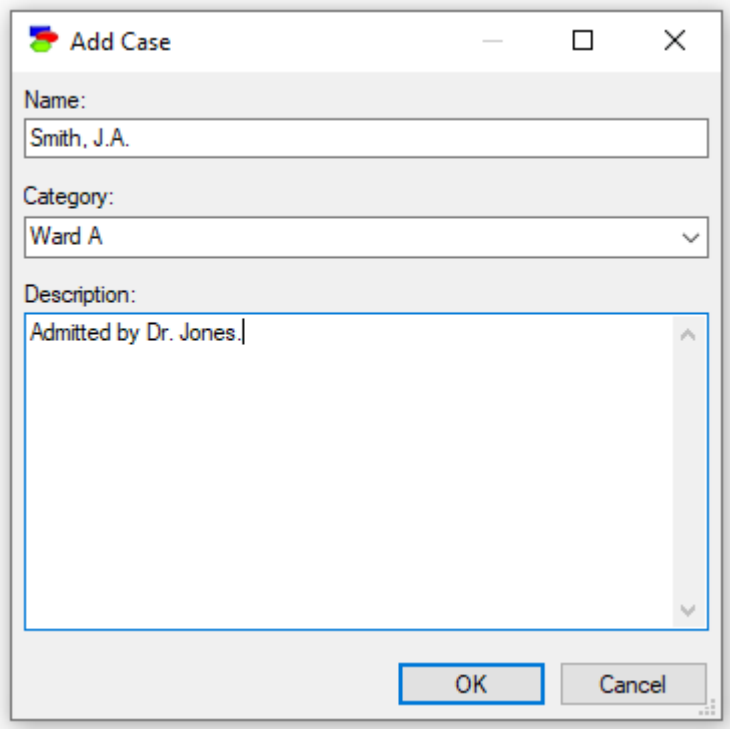


Case Manager window looks initially as follows:

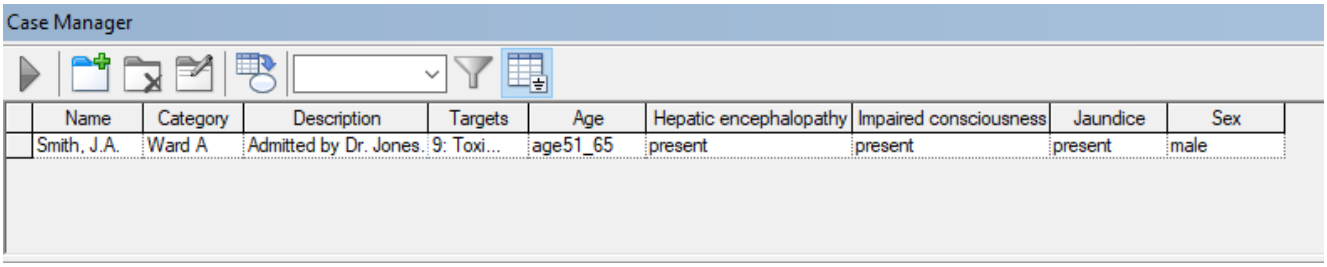


Adding cases to *Case Manager*

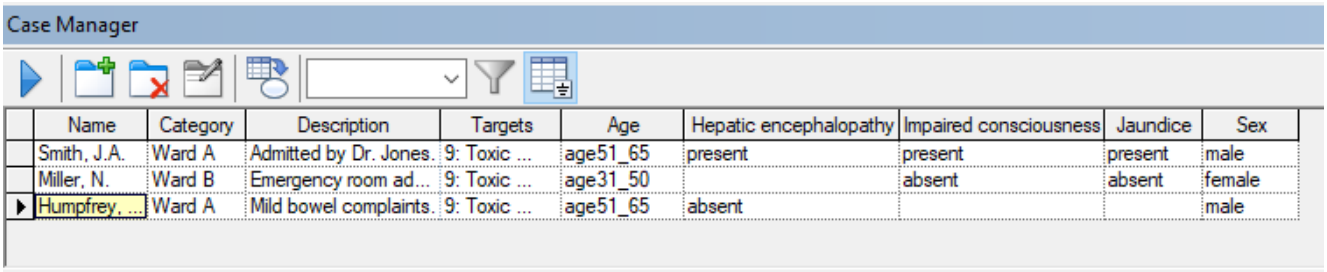
We can add the current case (this amounts to saving all evidence, as entered in the network) by clicking on the *Add new case* button ( ). This results in the following dialog, which allows for entering case details.



Once we click OK, the case is visible in the Case Manager:

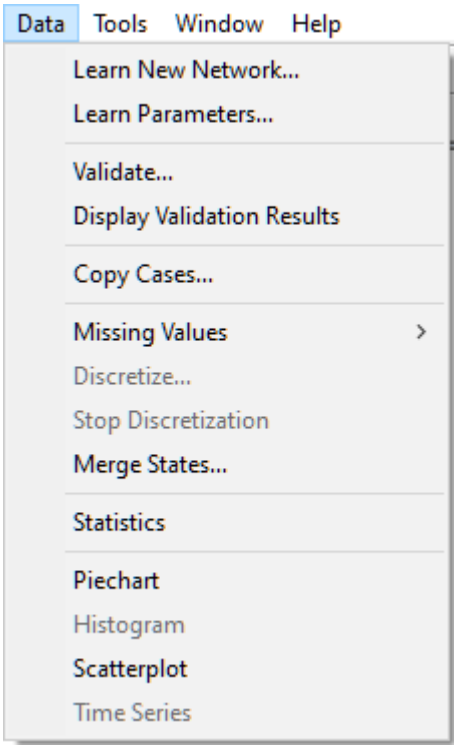


The *Show only evidence nodes* button ( ) reduces the number of columns displayed to those only that have any observations at all. The *Apply case* ( ) button allows for transferring a case to the *Graph View* window. The *Case Manager* window shows the currently applied case with a grayed background.

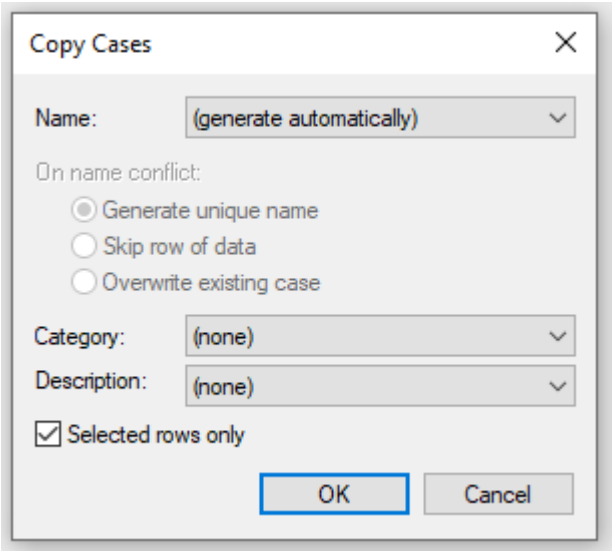


Importing case records from a data file

It is possible to import cases into the *Case Manager* automatically from a data file. To start the process, open a data file corresponding to the model (in the sense of having the same variables and their outcomes) and select *Copy Cases...* from the *Data Menu*.



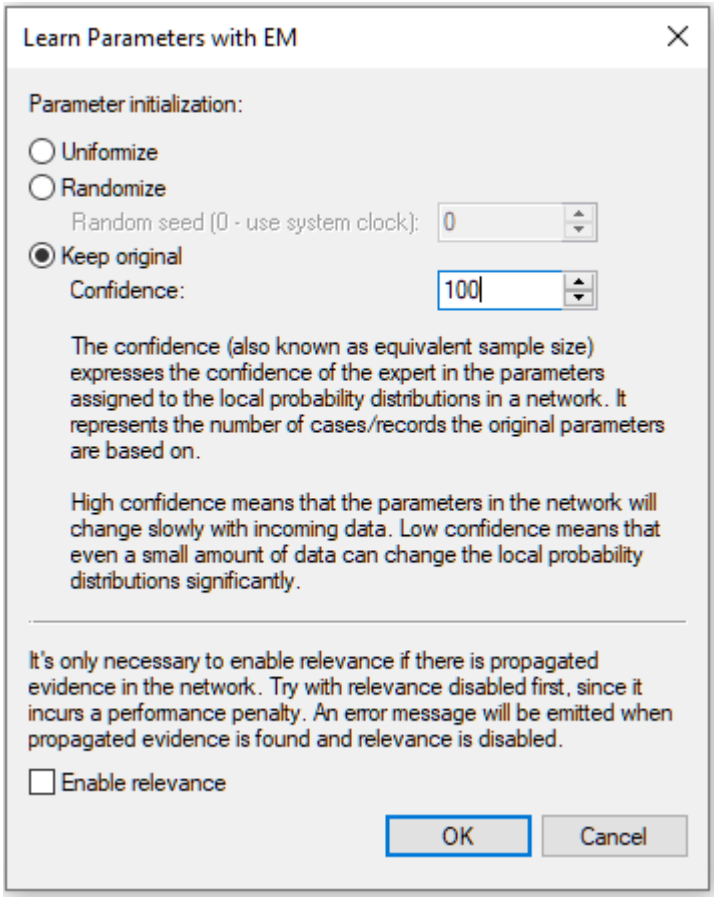
This opens the *Copy Cases* dialog



It is possible to import all records from the data or just selected records. Both the data and the network have to be open.

Refining model parameters from accumulated cases

When a case is applied, all its evidence is entered into the model and visible as evidence in the *Graph View*. We can work on the case, for example by entering new observations. The *Update applied case with the network evidence* ( ) button allows for bringing newly entered evidence from the network back to the case. Individual cases can be deleted using *Delete case* ( ) button. The Run EM algorithm ( ) button allows us to refine the parameters of the network with the accumulated cases.



The EM algorithm is discussed in the context of parameter learning in GeNIe. Roughly speaking, the *Confidence* parameter is known as *equivalent sample size* (ESS), which is the number of records that the current network parameters are based on. The interpretation of this parameter is obvious when the entire network or its parameters have been learned from data. When they are elicited from an expert, we can view them as the number of cases that the expert has seen before providing us with the current parameters. The larger the ESS, the less weight is assigned to the new cases, which gives a mechanism for gentle refinement of model numerical parameters.

Once we have entered a number of cases, we can navigate through them. To locate a case, enter the case name or specific characters of case name in the filter string box (Smith). Pressing the *Filter cases by text* (abc) button displays cases that match the string. *Show filtered cases* ( ) button toggles between all cases and cases matching the filtering text.

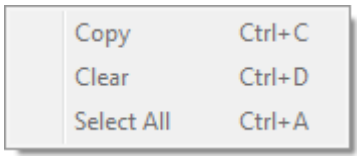
Exporting case records to a data file

Currently, GeNIe does not allow to export cases from a model to a data file. However, it is possible to retrieve cases using any text editor from the model file (.xdsl). To locate the case descriptions, please search for the tag `<cases>` in the XML model file.

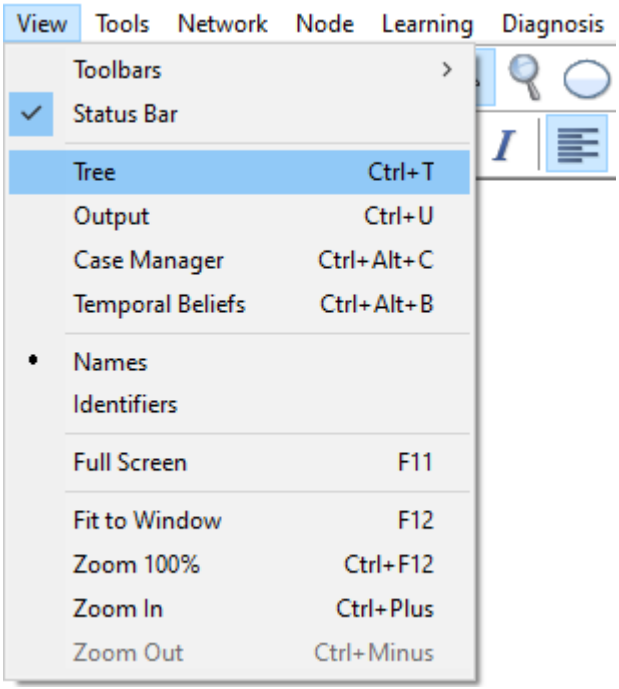
5.2.7 Output window

GeNIe includes a *Output Window* that is used for notifying the user about possible problems with the model or program errors. The *Output Window* is usually shown in the bottom part of the screen, but can be moved to any location by the user.

You can perform selections of the messages or clear the contents of the *Output Window* through a context menu, available by right-clicking within the area of the window.

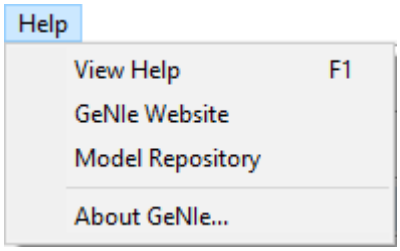


The *Output* window can be hidden or made visible by the user by changing the *Output* flag in the *View Menu* (shortcut *CTRL-U*).



5.2.8 Help menu

The *Help* menu offers commands providing assistance to the user:



It contains three commands:

View Help (or *F1* key) invokes GeNIe on-line help, which is the document that you are reading at the moment. GeNIe on-line help is composed in HTML format and is, in addition to being distributed with the program, also available on BayesFusion, LLC's support WWW pages. To exit the on-line help, simply close its window.

GeNIe Website will take you to the official GeNIe website at https://www.bayesfusion.com/

About GeNIe shows the following simple window:

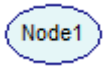
It displays a copyright notice, version number and the date of your build of GeNIe, information about availability of updates, license ID and expiration dates, license holder's name and institution. GeNIe informs its user about availability of updates through a non-obtrusive message in the *Output window*. Frequency of messages (default is once a week) can be set through program *Options*. GeNIe also reminds its users about upcoming license expiration date through a pop-up message that appears within a month of the license expiration.

5.3 Components of GeNIe models

5.3.1 Node types

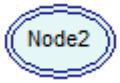
GeNIe supports the following node types:

Chance nodes, drawn as ovals, denote uncertain variables.



There are three basic types of discrete chance nodes: *General*, *Noisy Max* and *NoisyAdder*. There is no distinction between the three types in the graph view, as they differ only in the way their conditional probability distributions are specified. See *Canonical models* section for more information on the *Noisy Max* and *NoisyAdder* nodes. *Chance* nodes are discrete but are capable of representing continuous quantities discretized as *Identifiers*, *Intervals*, *Identifiers with intervals*, and *Identifiers with point values*. Defining a node as *Intervals* or *Point values* makes the node numerical, even though it is discrete. We will discuss these nodes in detail in the *Node properties* section.

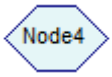
Deterministic nodes, usually drawn as double-circles or double-ovals, represent either constant values or values that are algebraically determined from the states of their parents. In other words, if the values of their parents are known, then the value of a deterministic node is also known with certainty. *Deterministic* nodes are quantified similarly to *Chance* nodes. The only difference is that their probability tables contain all zeros and ones (note that there is no uncertainty about the outcome of a deterministic node once all its parents are known).



There is a popular error, made by novices in the area of probabilistic graphical models, to equip models with deterministic parentless nodes. This is a bad practice that is perhaps not changing the numerical properties of the model but it serves no purpose, obscure the picture, and make the model larger, hence, harder to update. You will typically not notice the impact of these nodes on the speed of calculations in GeNIe because it is so efficient and fast but at some point, even GeNIe may choke - after all, calculations in Bayesian networks are worst-case NP-hard. A typical motivation of modelers is to add prior knowledge that is relevant to the model. If this is the motivation, then it is best to make this prior knowledge described in on-screen text boxes or annotations. For example, a text box that lists model assumptions may state that "the economy is struggling", etc. From the theoretical point of view, every probability in a model is conditional on the background knowledge, so for any probability p, one could write $p=\Pr(X|\zeta)$, where X is the event in question and ζ is the background knowledge. Because every model parameter would have to be conditioned on ζ, one typically omits it. If there is a non-zero chance that the economy will change during the lifetime of the model (for example, one might want to use the same model for a different region/country), then it is best to make it a chance node. Chance nodes allow for observing their states (e.g., economy is low now). It is possible to calculate the *Value of Information* (VOI) for such a chance node.

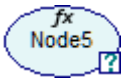
Decision nodes, drawn as rectangles, denote variables that are under decision maker's control and are used to model decision maker's options. Decision nodes in GeNIe are always discrete and specified by a list of possible states/actions.

Value nodes (also called *Utility* nodes), drawn as hexagons, denote variables that contain information about the decision maker's goals and objectives. They express the decision maker's preferences over the outcomes over their direct predecessors.

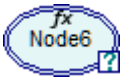


There are two fundamental types of value nodes: *Utility* and *Multi-Attribute Utility*. The latter include a special case of *Additive Linear Utility* (ALU) functions. There is no distinction between the two in the graph view, as they differ only in the way they specify the utility functions. *Utility* nodes specify the numerical valuation of utility and *Multi-Attribute Utility* nodes specify the way simple *Utility* nodes combine to form a *Multi-Attribute Utility* function. *Utility* nodes cannot have other utility nodes as parents. The *Multi-Attribute Utility* nodes can have only *Utility* nodes as parents. See Multiple Utility Nodes section for more information.

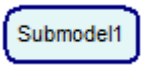
Equation nodes, which are relatives of chance nodes, drawn as ovals with a wave symbol, denoting that they can take continuous values. Instead of a conditional probability distribution table, which describes the interaction of a discrete node with its parents, an equation node contains an equation that describes the interaction of the equation node with its parents. The equation can contain noise, which typically enters the equation in form of a probability distribution.



Deterministic equation nodes, drawn as double ovals with a wave symbol, denote equation nodes without noise, i.e., they are either constants or equations that do not contain a noise element. Once we know the states of their parents, the state of the child is, thus, determined.



Submodel nodes, drawn as rounded rectangles, denote submodels, i.e., conceptually related groups of variables. Submodel nodes are essentially holders for groups of nodes, existing only for the purpose of the user interface, and helping with making models manageable.



To learn how to create nodes and arcs between them, see the introductory sections on *Building a Bayesian network* and *Building an influence diagram*.

There is one useful functionality that is best discussed in this section. When creating new nodes, one has to select one of the node creation tools (*Chance*, *Deterministic*, *NoisyMAX*, *Equation*, *Decision*, and *Value*). To facilitate fast construction of models, GeNIe allows for quick change of the selected tool to a related one. This is

useful especially when using the "sticky mode," i.e., when double-clicking on a tool. When the CTRL key is pressed when a node tool is active, GeNIe selects temporarily an alternate (closely related) node type. The list below shows the transition to alternate types when the CTRL key is pressed:

Chance -> *Equation*

Equation -> *Chance*

Deterministic -> *Chance*

Decision -> *Chance*

NoisyMAX -> *NoisyAdder* (the cursor remains the same)

Value -> *ALU* (the cursor remains the same)

This functionality is useful, for example, when working on hybrid Bayesian networks, as the user may remain in the sticky mode, creating *Chance* and *Equation* nodes alternatively, depending on the need.

5.3.2 Canonical models

Canonical probabilistic nodes, such as *Noisy-MAX/OR*, *Noisy-MIN/AND* and *Noisy-Adder* gates, implemented by GeNIe, are convenient knowledge engineering tools widely used in practical applications. In case of a general *Chance* binary node with n binary parents, the user has to specify 2^n parameters, a number that is exponential in the number of parents. This number can quickly become prohibitive - please note that when the number of parents n is equal to 10, we need 1,024 parameters, when it is equal to 20, the number of parameters is equal to 1,048,576, with each additional parent doubling it. A *Noisy-OR* model allow for specifying this interaction with only n+1 parameters, one for each parent plus one more number. This comes down to 11 and 21 for n equal to 10 and 20 respectively.

The savings stemming from the use of canonical models in terms of the number of probability elicitations may be dramatic, especially when the number of parents of a node becomes large. Canonical models are not only great tools for knowledge engineering - they also lead to significant reduction in computation through the independences that they model implicitly. Using canonical gates makes thus model construction easier but also leads to models that are easier to solve.

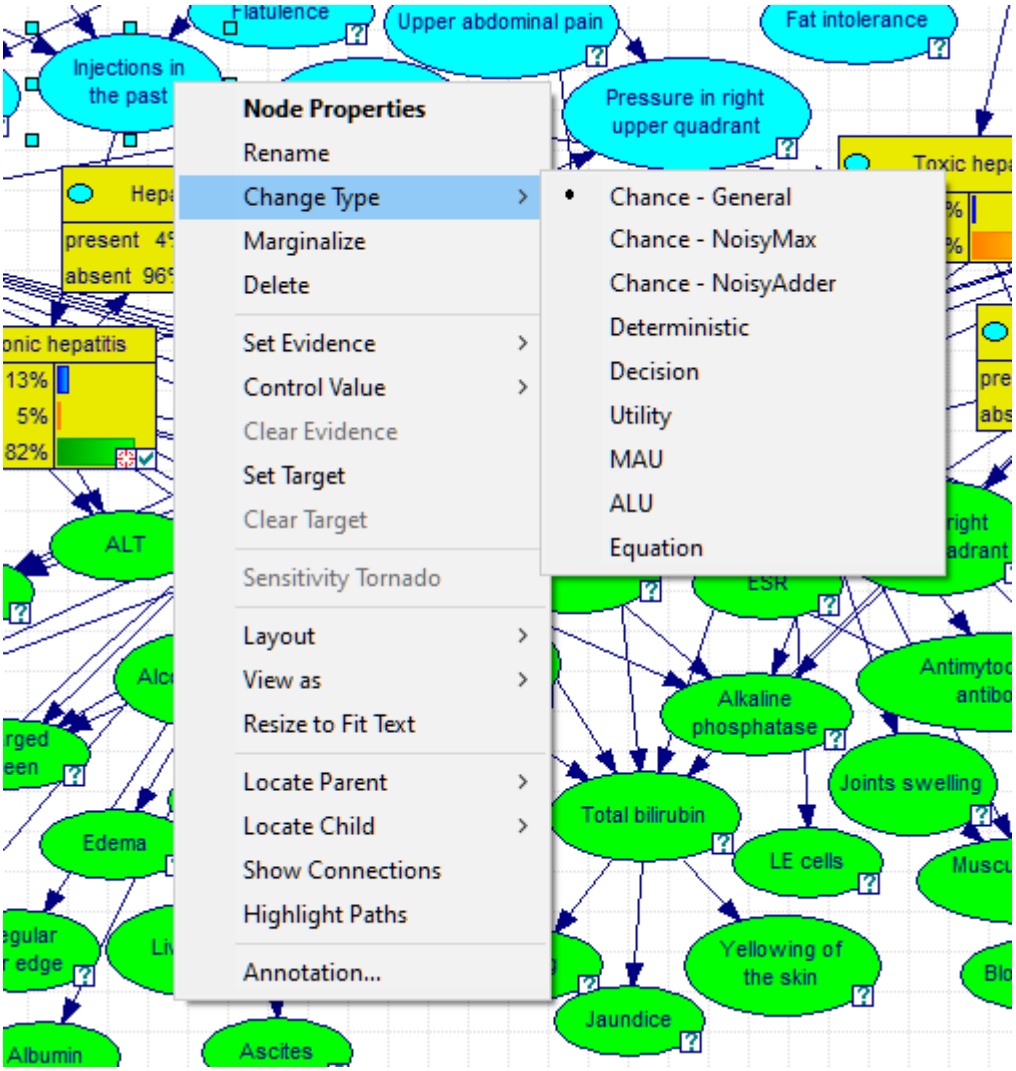
This section gives a brief introduction to how GeNIe implements *Noisy-OR/MAX*, *Noisy-AND/MIN* and *Noisy-Average* gates, assuming that the reader is familiar with the concepts and has a basic knowledge of the principles applied in these gates. To learn more about the *Noisy-OR/MAX* and *Noisy-AND/MIN* gates and their practical value, please refer to the excellent paper on the topic by Henrion (1989). Diez & Druzdzel (2006) summarize the theory behind the *Noisy-MAX* and other canonical gates.

5.3.2.1 Noisy-MAX model

The Noisy-OR model, a precursor of the Noisy-MAX model, was introduced for binary variables by Pearl (1988) and extended to binary leaky Noisy-OR gates by Henrion (1989). Generalizations to multi-valued Noisy-OR gates were proposed independently by Diez (1993) and Srinivas (1993). GeNIe implementation allows using two parametrizations, proposed by Diez (1993) and Henrion (1989) respectively. This section assumes a basic knowledge of the canonical gates. To those readers, who would like to understand them better, we recommend the original paper by Henrion (1989), an introductory paper of Heckerman & Breese (1996), and the review paper by Diez & Druzdzel (2006).

Both, *Noisy-OR* and *Noisy-AND* types of nodes can be modeled using *Noisy-MAX* nodes implemented in GeNIe. *Noisy-MAX* nodes can be created in two ways: (1) by CTRL-clicking in the *Graph View*, and (2) by changing the

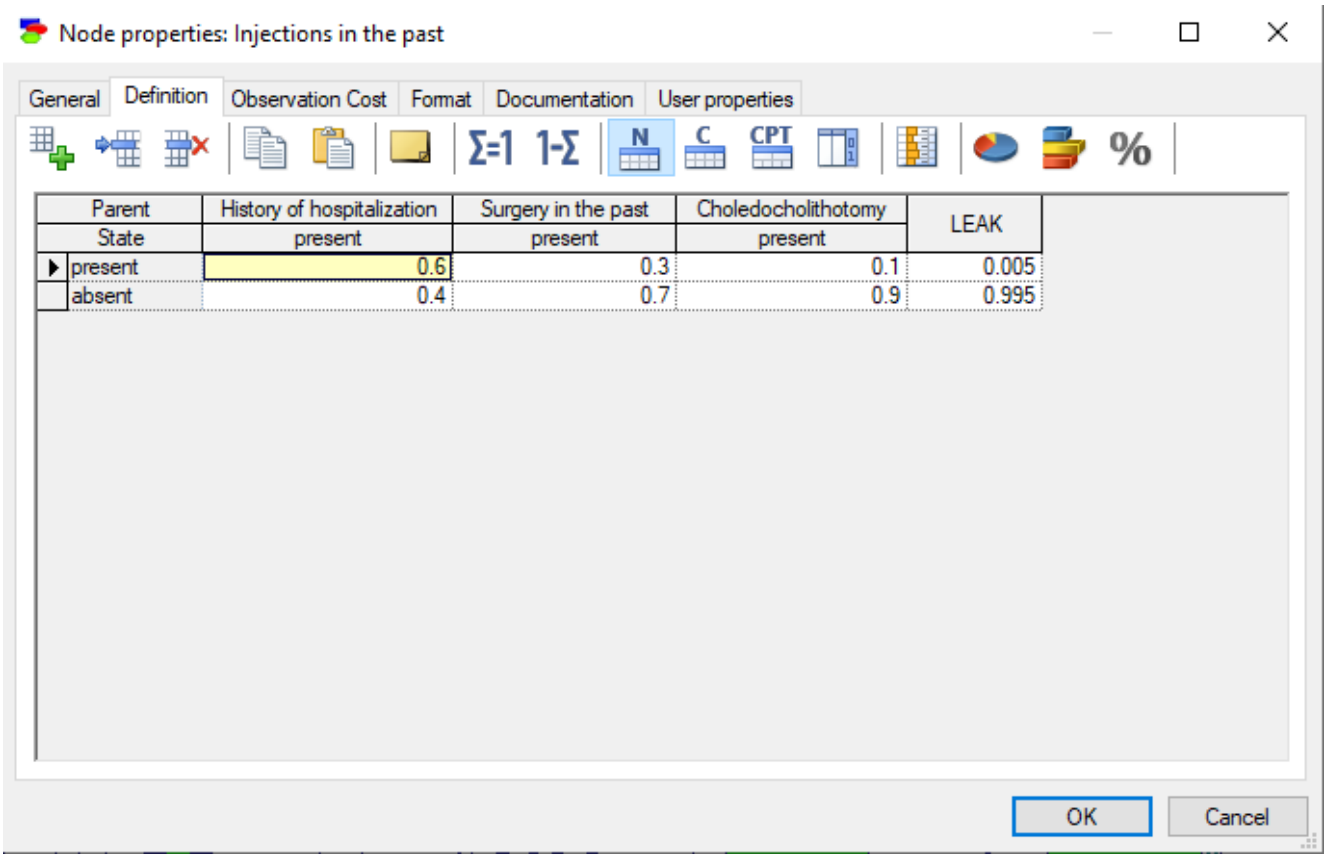
node type to *NoisyMAX*. To change the type of a node from general to *Noisy-MAX*, right-click on a node, select *Change Type* and then *NoisyMAX*.



Noisy-MAX is a generalization of a popular canonical gate *Noisy-OR* and is capable of modeling interactions among variables with multiple states. If all the nodes in question are binary, a *Noisy-MAX* node reduces to a *Noisy-OR* node. The *Noisy-MAX*, as implemented in GeNIe, includes an equivalent of negation. By DeMorgan's laws, the *OR* function (or its generalization, the *MAX* function) along with a negation, is capable of expressing any logical relationship, including the *AND* (and its generalization, *MIN*). This means that GeNIe's *Noisy-MAX* can be used to model the *Noisy-AND/MIN* functions, as well as other logical relationships.

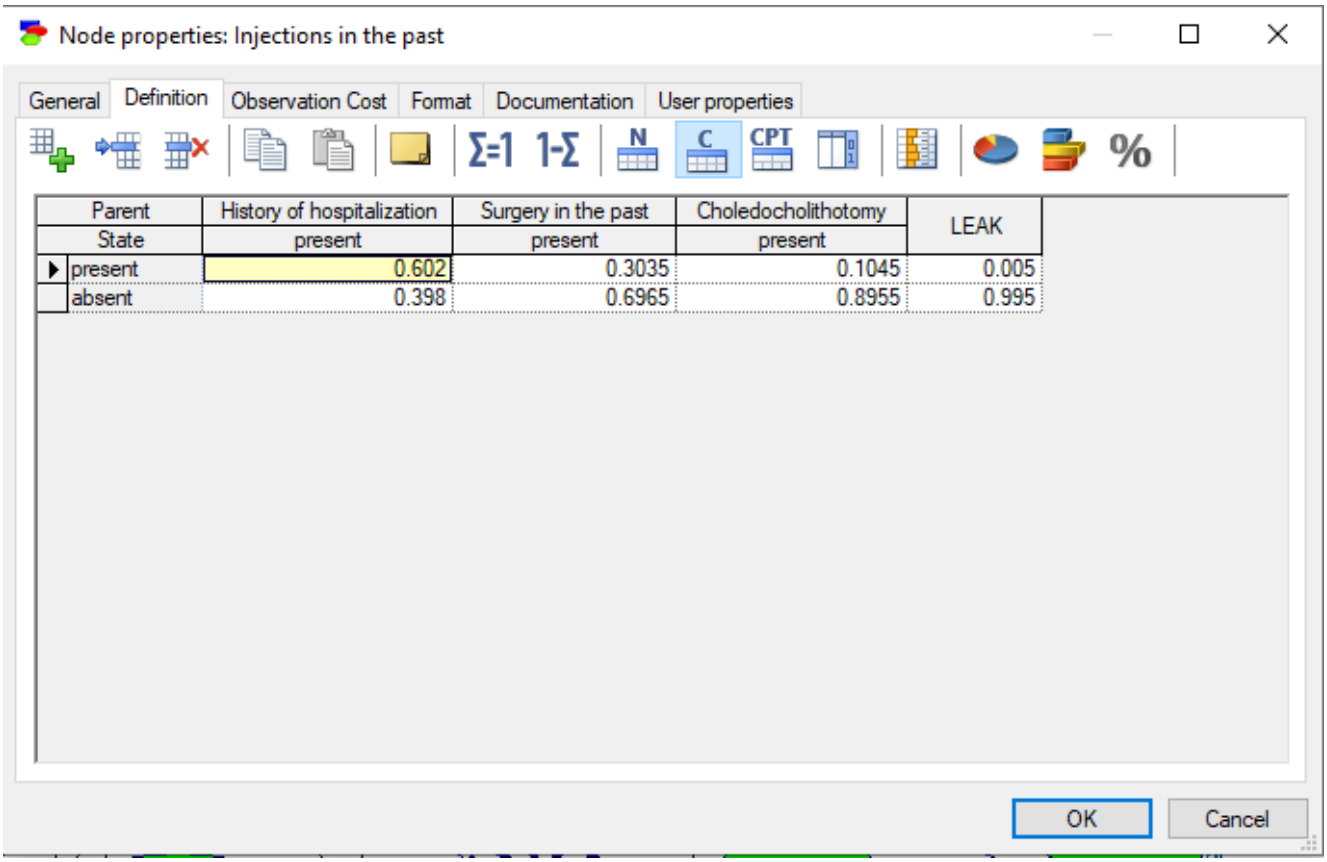
There are two different (but mathematically equivalent) parameterizations of the *Noisy-OR/MAX* gates. They are often referred to by names of the researchers who proposed them, Henrion (1989) and Diez (1993). The two parameterizations are in the forms of $P(Y|X_i)$ and $P(Y|\sim X_1,\ldots,X_i,\ldots,\sim X_n,X_L)$. The latter parametrization (Diez's) is called in GeNIe the *net* representation, while the former (Henrion's) is called *compound*. GeNIe gives the user the freedom to specify *Noisy-MAX* gates using any of the two representations.

To examine the node definition of a *Noisy-Max* gate, open the *Definition* tab among the node's *Node Properties*.



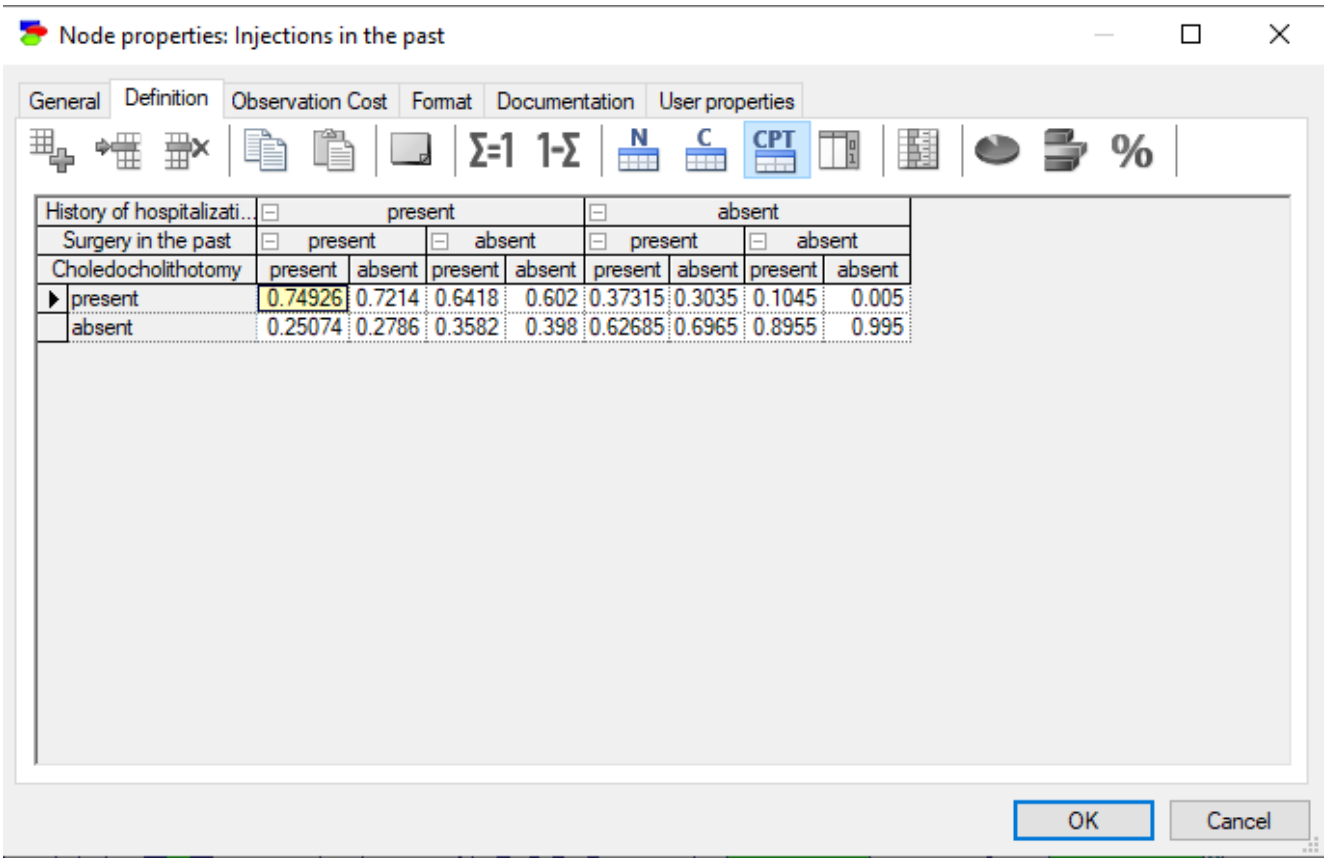
Parent	History of hospitalization	Surgery in the past	Choledocholithotomy	LEAK
State	present	present	present	
present	0.6	0.3	0.1	0.005
absent	0.4	0.7	0.9	0.995

The default definition format for *Noisy-MAX* nodes in GeNIe is *net*. *Net* probabilities in a *Noisy-MAX* node for a parent X_i express the probability of the effect happening when the cause X_i is present and none of the other causes of the effect, whether modeled or unmodeled is present. It is possible to examine the *compound* parameters that correspond to the *net* parameters show above. To see the compound parameters, click on the *Show compound parameters* ( ) button.



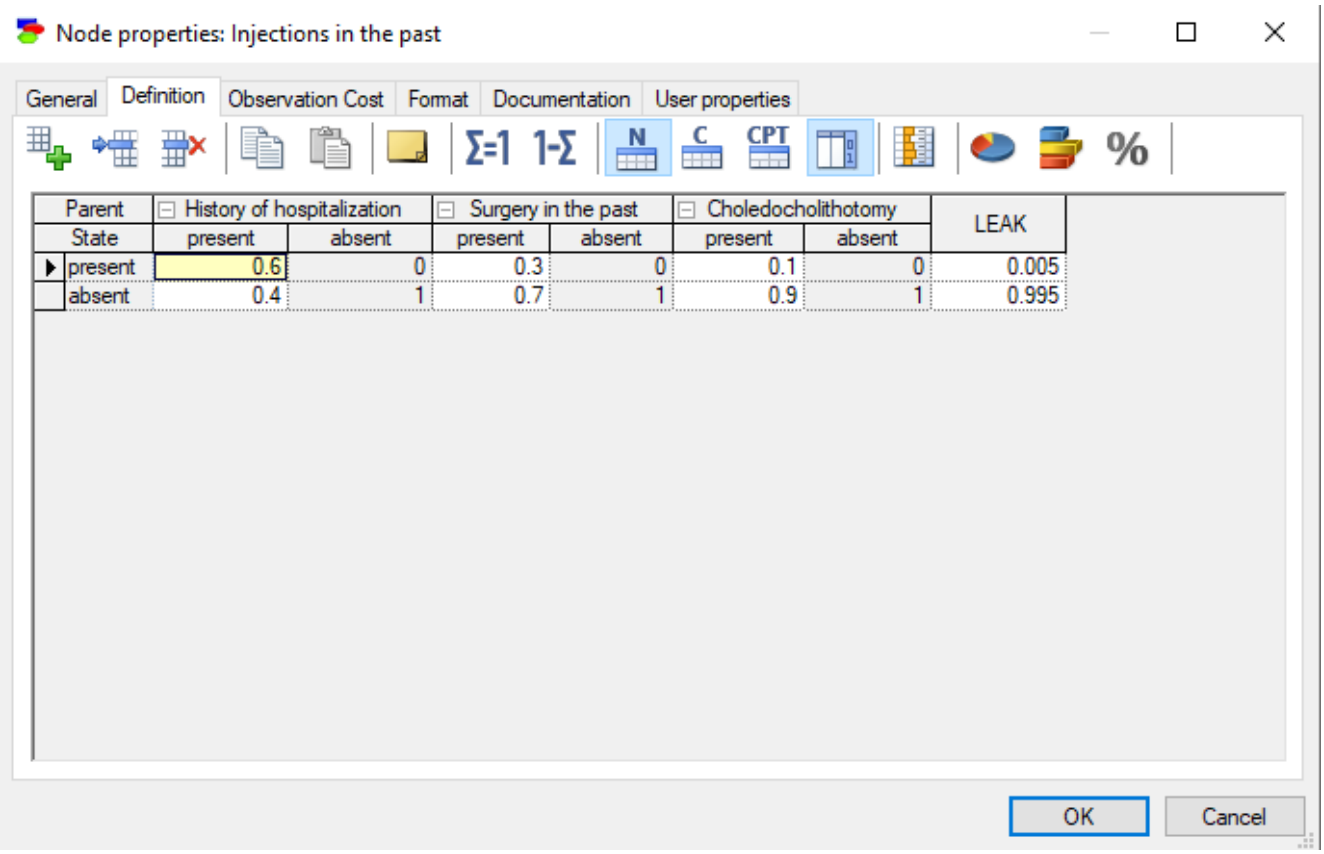
Parent	History of hospitalization	Surgery in the past	Choledocholithotomy	LEAK
State	present	present	present	
present	0.602	0.3035	0.1045	0.005
absent	0.398	0.6965	0.8955	0.995

You can examine the CPT that corresponds to the *Noisy-MAX* definition at any time by clicking on the *Show CPT* ( ) button.



History of hospitalizati...	present				absent			
Surgery in the past	present		absent		present		absent	
Choledocholithotomy	present	absent	present	absent	present	absent	present	absent
present	0.74926	0.7214	0.6418	0.602	0.37315	0.3035	0.1045	0.005
absent	0.25074	0.2786	0.3582	0.398	0.62685	0.6965	0.8955	0.995

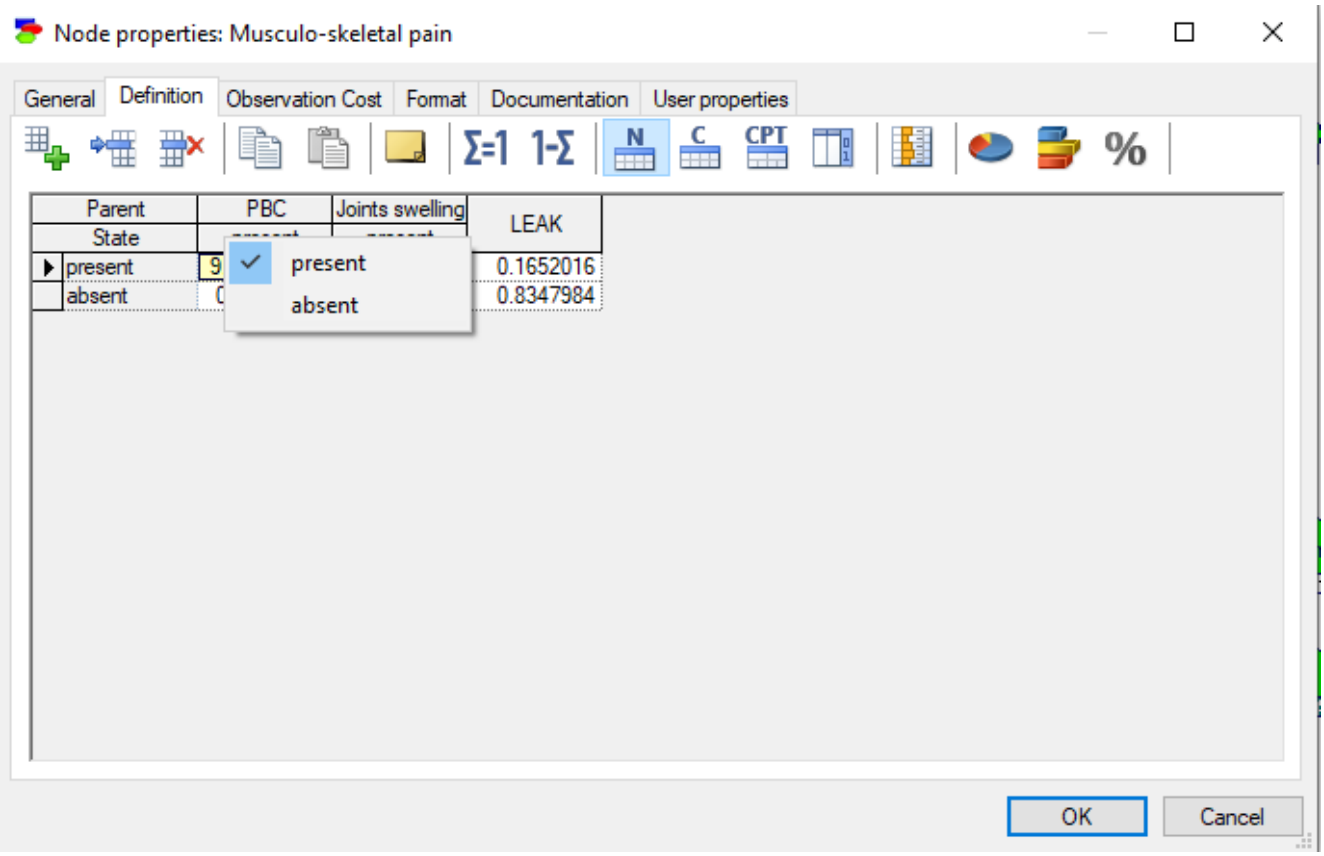
By default GeNIe hides those columns of the *Noisy-MAX* definition that correspond to *designated* parent states. You can view these columns by pressing the *Show constrained columns* ( ) button.



Parent	History of hospitalization		Surgery in the past		Choledocholithotomy		LEAK
State	present	absent	present	absent	present	absent	
present	0.6	0	0.3	0	0.1	0	0.005
absent	0.4	1	0.7	1	0.9	1	0.995

The regions shown with a gray background denote constraints on the *Noisy-MAX* parametrization - the last state of each parent is assumed to have distribution {0.0, 0.0, ..., 0.0,1.0}. States of the node and of its parents can be ordered, which amounts to a negation. *Noisy-MAX* implementation in GeNIe allows the user to control the order of states of the parent nodes, as they enter the relation with the child. *Noisy-MAX* table always follows the order of strengths - this means, that the first column for each parent is assigned to the outcome of the greatest strength, the second column to the outcome of the second strength end so on. The distinguished state is always assigned to the last column, which is by default hidden. To move states around and, by this, change the order of states within any parent or child node, drag and drop the states to their desired position.

Whenever there is at least one node among the parent nodes that has more than two states, the *Show constrained columns* ( ) button is pressed by default and grayed out. This is to allow for easy rearrangement of the order of their states. When all parent nodes are binary and the *Show constrained columns* button is depressed, it is still possible to change the order of states of the parents by right-clicking on the state and choosing the active state of the node.



In the screen shot above, the user right-clicked on the state *Present* of the node *Liver Cancer* and selected *Present*.

In order for the Noisy-MAX model to be applicable in practice, the following conditions have to be fulfilled (see also Diez & Druzdzel 2006).

There should be a causal mechanism for each parent such that the parent is able to impact the child variable in the absence of the other anomalies. If co-occurrence of two or more parents is necessary to impact the child, the model may not be suitable.

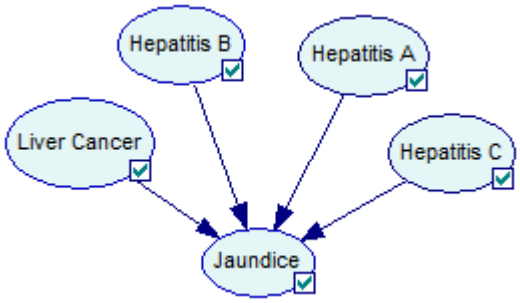
Are the causal mechanisms independent? This is quite likely the hardest condition to verify, because our knowledge of the domain may not be precise enough to verify that the causal mechanisms in question do not interact with one another. In practice, however, unless there are known interactions, it is not unreasonable to assume that independence of causal influences (also known as the ICI condition) holds and the model can be applied. Should this condition not hold, it is typically possible to restructure the model in such a way that the ICI condition is approximately satisfied.

In case of existence of other, unmodeled causes (a non-zero Leak parameter), are these unmodeled causes independent of all parents of the effect variable? Typically, we assume that this condition holds unless there is evidence against it. In statistical modeling, this condition is often referred to as "independence of error terms."

We should keep in mind that the above conditions have to be satisfied approximately, as the modeling effort involving GeNIe graphs is an attempt to approximate reality and the methodology is robust to minor violations of assumptions.

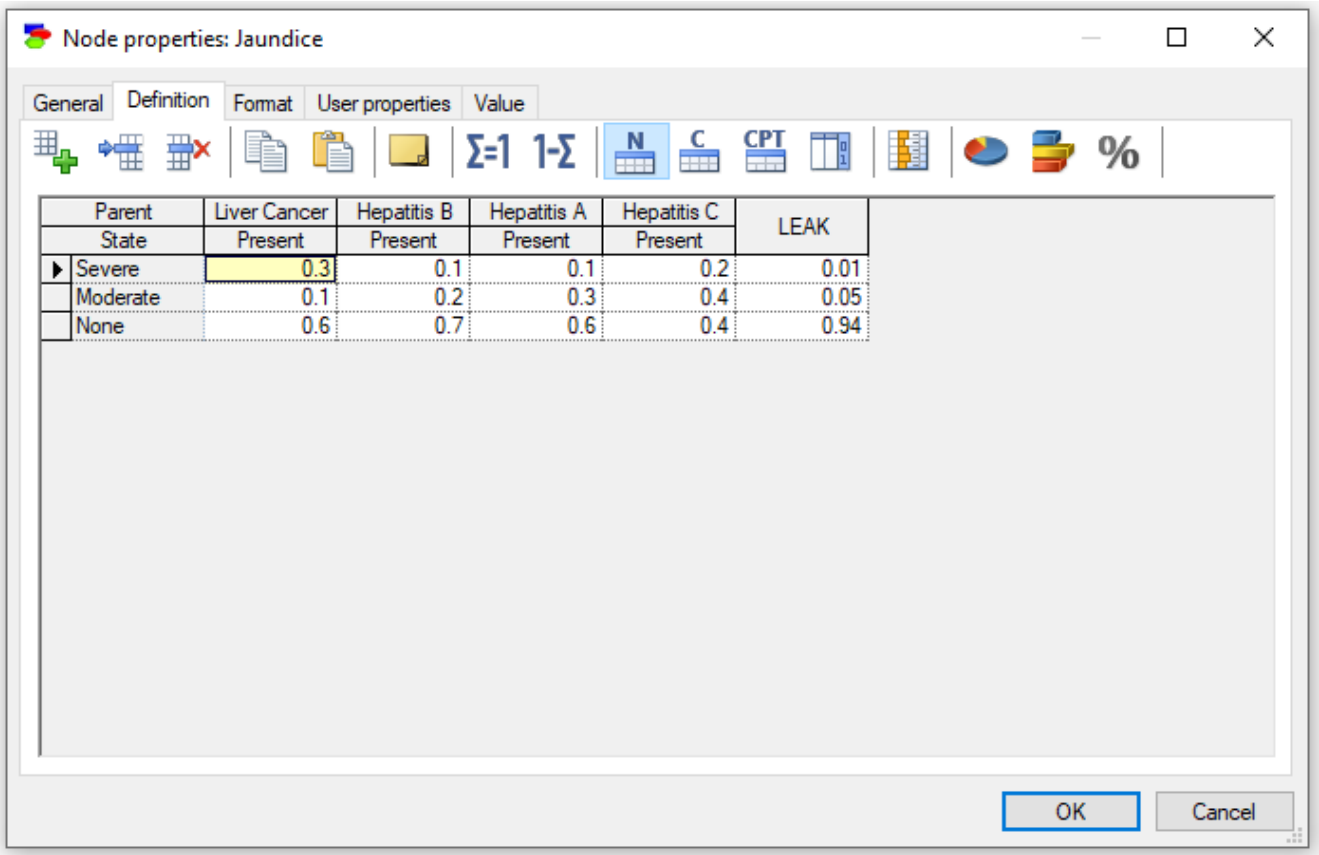
Noisy-MAX example

Consider the following Bayesian network modeling the interaction of various liver disorders (*Liver Cancer*, *Hepatitis A*, *Hepatitis B* and *Hepatitis C*) in in producing *Jaundice*. We start by creating the structure of the network as follows.



We create the node *Jaundice* as a general *Chance* node and subsequently change its type into a *Noisy-MAX* node. To this effect, we right click on the node, and choose *Change Type*, from the *Node Context Menu*, then *Chance - NoisyMax* and click *OK*. It can be reasonably assumed that each of the four causes of *Jaundice* works independently of the other causes. They are capable of causing *Jaundice* in separation and they do not interfere in each other's ability to cause *Jaundice*. *Jaundice* can also occur even if none of the four causes is active.

The definition tab of a *Noisy-MAX* node is similar to that of the definition of a conditional probability table.



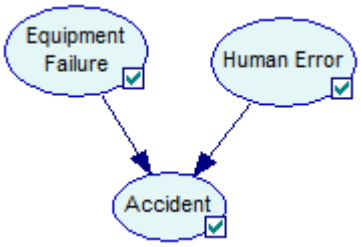
Parent	Liver Cancer	Hepatitis B	Hepatitis A	Hepatitis C	LEAK
State	Present	Present	Present	Present	
Severe	0.3	0.1	0.1	0.2	0.01
Moderate	0.1	0.2	0.3	0.4	0.05
None	0.6	0.7	0.6	0.4	0.94

The probability numbers in the table are specified for each non-distinguished state of the causes (here, for each of the causes, the distinguished state is *Absent* and for the *Jaundice* the distinguished state is None). Distinguished states of the causes do not have any influence on *Jaundice*. The numbers in the table are *net* parameters, which means that they express the probability that *Jaundice* takes *Severe* or *Moderate* state when of all possible causes of jaundice, only the current state of the current cause is present. For example, if the outcome of the node *Liver Cancer* is *Present* (i.e., cancer is present) and all other possible causes of *Jaundice* (including the dummy *LEAK* node) are absent, then the probability of *Severe Jaundice* is 0.3. When none of the four explicitly modeled causes of *Jaundice* are present, there is still 0.01 chance of *Severe* and 0.05 chance of *Moderate Jaundice* (because of all possible unmodeled causes of *Jaundice*), as specified in the last (*LEAK*) column.

It is important that the outcomes of the node *Jaundice* are ordered from the highest to the lowest. The outcomes of the parent nodes should also be ordered from the highest to the lowest (in terms of strength of the causal influence on the child node), i.e., from the one that influences the child most to the one that influences it least. To change the order of outcomes, drag and drop them at their desired places. To change the order of the states of any of the parents, click on the *Show constrained columns* ( ) button or, if all parent nodes are binary, right-click on the state of the node in question to choose a non-distinguished state.

Noisy-AND example (Modeling *Noisy-AND* nodes with *Noisy-MAX*)

Consider the following Bayesian network modeling the interaction of two causes of an *Accident*: *Equipment Failure* and *Human Error*. These two causes are both needed for an accident to happen and it seems that we are dealing with an AND-type interaction. We start by creating the structure of the model.

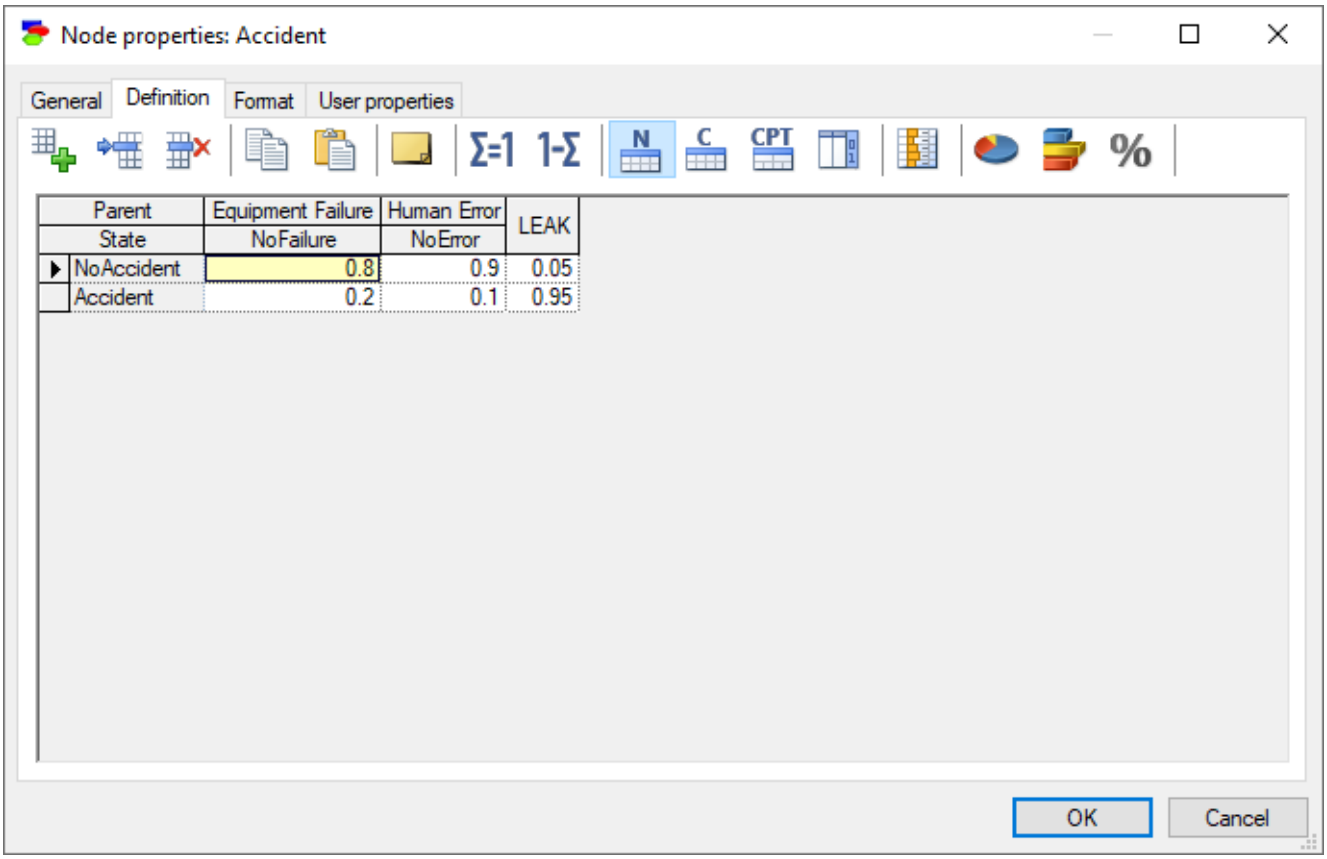


We create the node *Accident* as a general *Chance* node and subsequently change its type into a *Noisy-MAX* node. To this effect, we right click on the node, and choose *Change Type*, from the *Node Context Menu*, then *Chance - NoisyMax* and click *OK*. It can be reasonably assumed that each of the two causes of *Accident* works independently of the other causes. They are both needed for an *Accident* to happen but *Accident* can happen if just one of them or neither of them is present.

Noisy-MAX implementation in GeNIe does not require a separate implementation of *Noisy-AND/MIN* nodes. This is because a *Noisy-AND* node can be modeled by a *Noisy-OR* node and negation, based on DeMorgan's laws (we mean here, in particular $X \wedge Y = \neg(\neg X \vee \neg Y)$). Hence, having a Noisy-OR model and the ability to negate its inputs and its output, one can implement *Noisy-AND* gate. Negation corresponds to reversing the order of outcomes (or causal strengths), which can be easily done in GeNIe.

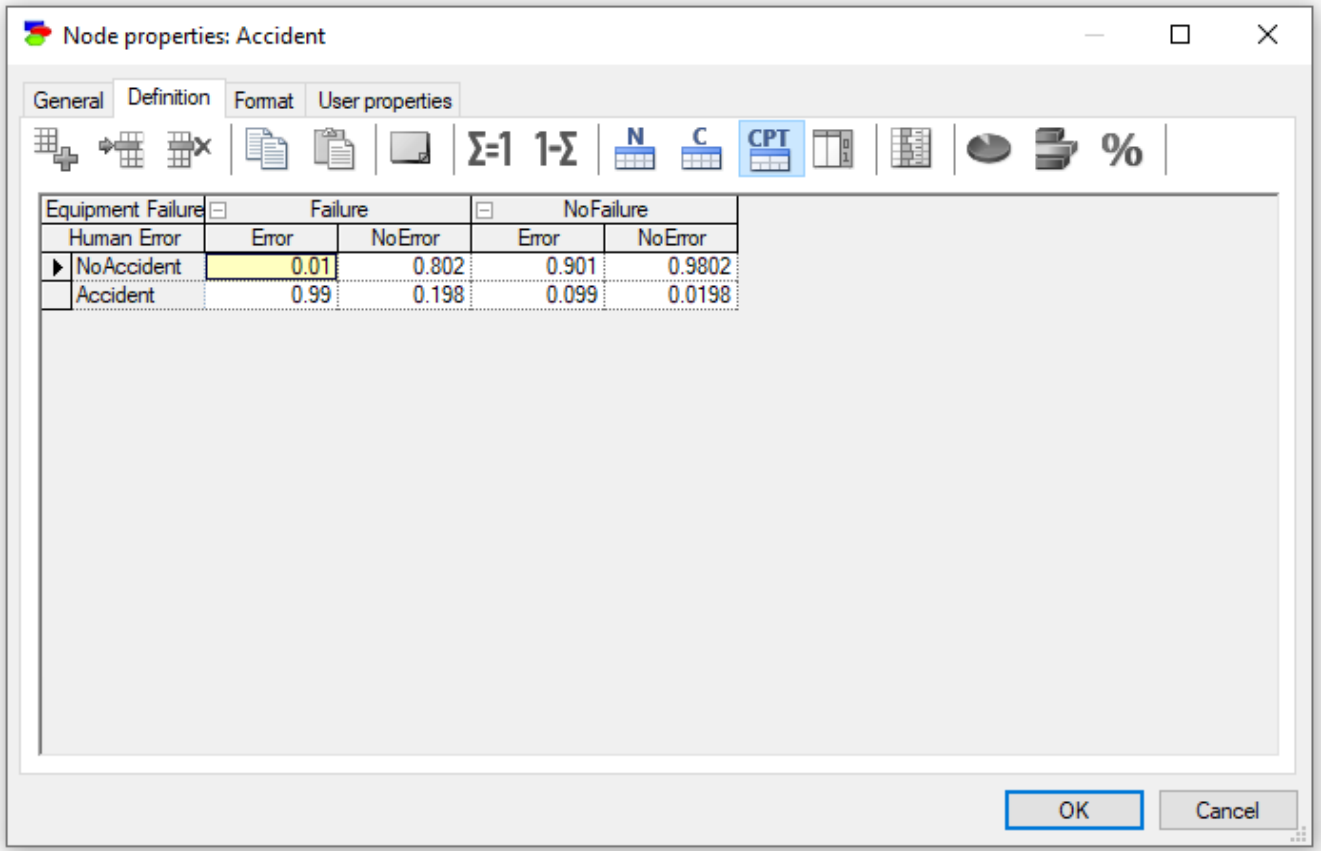
Noisy-AND nodes work in a similar way to *Noisy-OR* nodes, but the parameters correspond to the probability of the effect being active even if the cause in question is inactive, given that all other causes are active (please note that we are dealing with a negation of every parent!). Each parameter is defined as the probability that the *Noisy-AND* node is in the *designated* state given that the specified parent node is in a certain inactive state but all other parent nodes, causes of the effect, are in *designated* state, which usually corresponds to *Active* or *Present*. For example, when there is no *Equipment Failure* (i.e., the state of that variable is *NoFailure*) but the other cause of accident, *Human Error* is present (state *Error*), we have the probability of 0.9 that there will still

be no accident. When there is no *Human Error* (i.e., the state of that variable is *NoError*) but the other cause of accident, *Equipment Failure* is present (state *Failure*), we have the probability of 0.8 that there will still be no accident. When both of the causes are active (states *Failure* and *Error* respectively), the probability that the accident will not happen is 0.01, which is the leak probability here. The following table shows the parametrization of the node *Accident* for this case.



Parent	Equipment Failure	Human Error	LEAK
State	NoFailure	NoError	
NoAccident	0.8	0.9	0.05
Accident	0.2	0.1	0.95

The resulting CPT will take the following form:



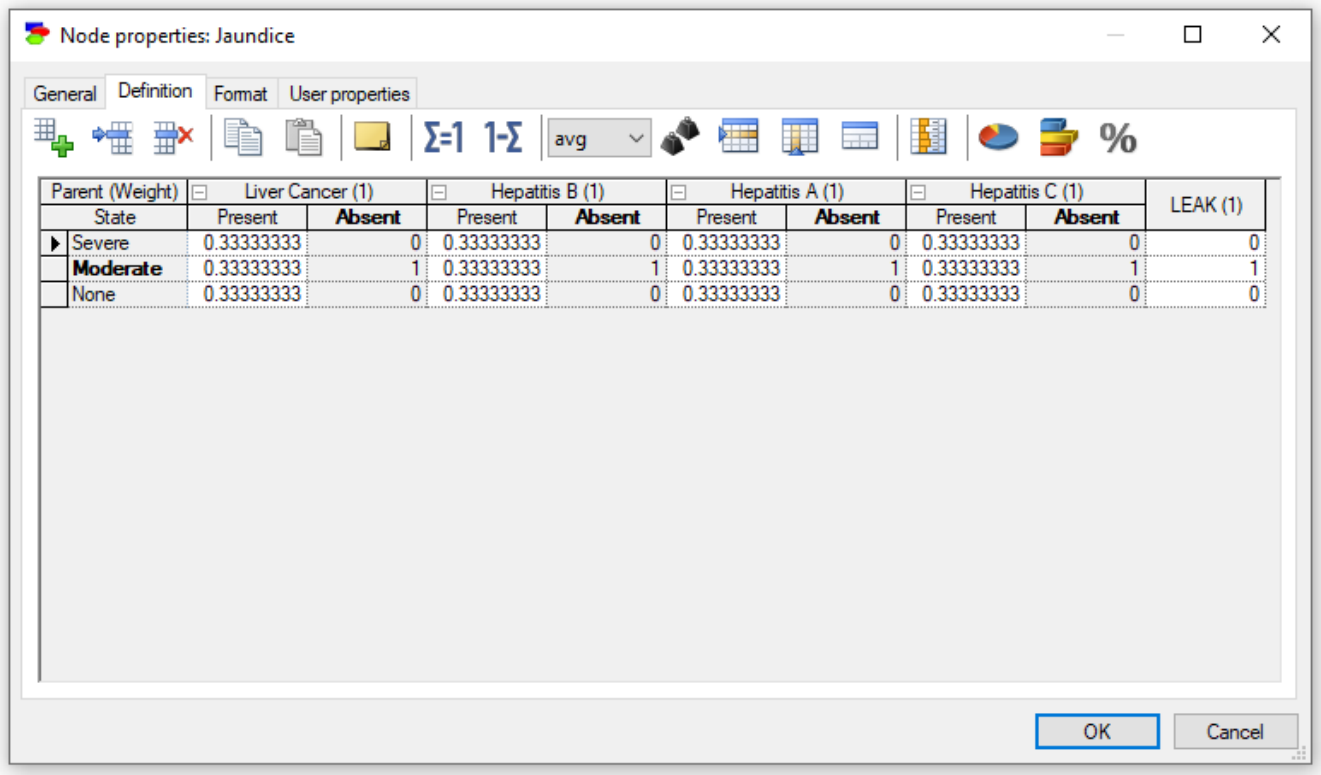
Equipment Failure	Failure		NoFailure	
Human Error	Error	NoError	Error	NoError
NoAccident	0.01	0.802	0.901	0.9802
Accident	0.99	0.198	0.099	0.0198

Please note that the CPT corresponds to our intuition: An accident is most likely to happen when both *Equipment Failure* and *Human Error* take place.

5.3.2.2 Noisy-Adder model

The *Noisy-Adder* model is described in the doctoral dissertation of Adam Zagorecki (2010) (*Section 5.3.1 Non-decomposable Noisy-average*). Essentially, it is a non-decomposable model that derives the probability of the effect by taking the average of probabilities of the effect given each of the causes in separation.

To examine the node definition of a Noisy-Adder gate, open the *Definition* tab among the node's *Node Properties*.



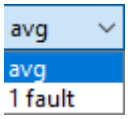
Parent (Weight)	Liver Cancer (1)		Hepatitis B (1)		Hepatitis A (1)		Hepatitis C (1)		LEAK (1)
State	Present	**Absent**	Present	**Absent**	Present	**Absent**	Present	**Absent**	
Severe	0.33333333	0	0.33333333	0	0.33333333	0	0.33333333	0	0
Moderate	0.33333333	1	0.33333333	1	0.33333333	1	0.33333333	1	1
None	0.33333333	0	0.33333333	0	0.33333333	0	0.33333333	0	0

There are two additional buttons on the *Definition* tab that allow for specifying which of the states of the current node and the parents are the distinguished states.

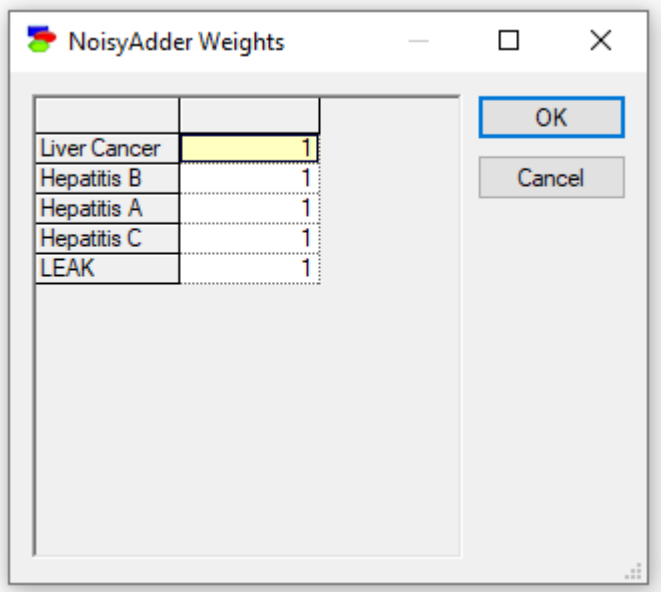
The *Set distinguished state of this node* ( ) button allows for choosing the distinguished state of the current node. The state with the cursor will become the distinguished state after clicking the button.

The *Set distinguished state of parent node* ( ) button allows for choosing the distinguished state of the selected parent node. The state with the cursor will become the distinguished state after clicking the button.

There are two types of *Noisy-Adder* nodes: *average* and *single fault*, settable through a pop-up menu

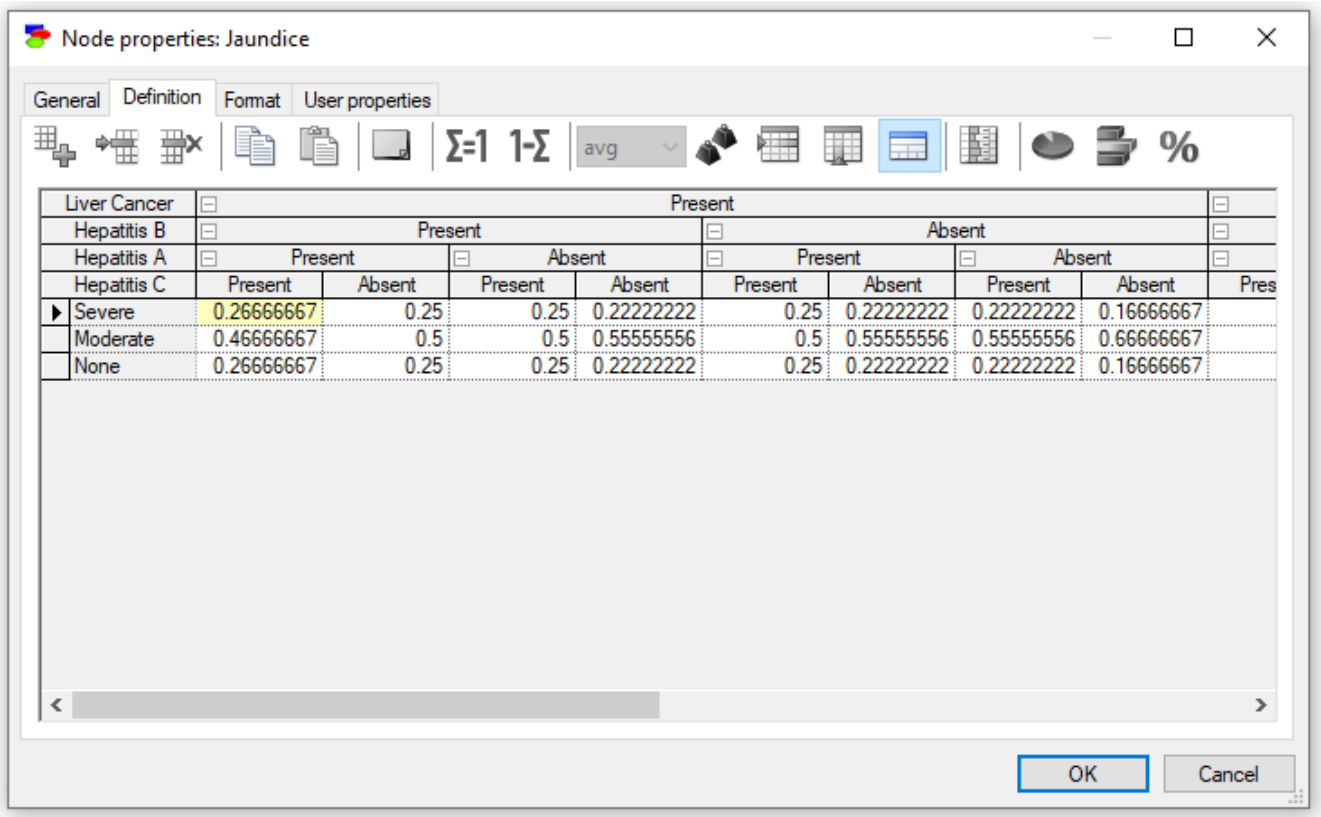


The *average*-type *Noisy-Adder* has weights associated with each of the parents. These weights can be edited by invoking the *NoisyAdder Weights* dialog through the *Set weights* ( ) button.

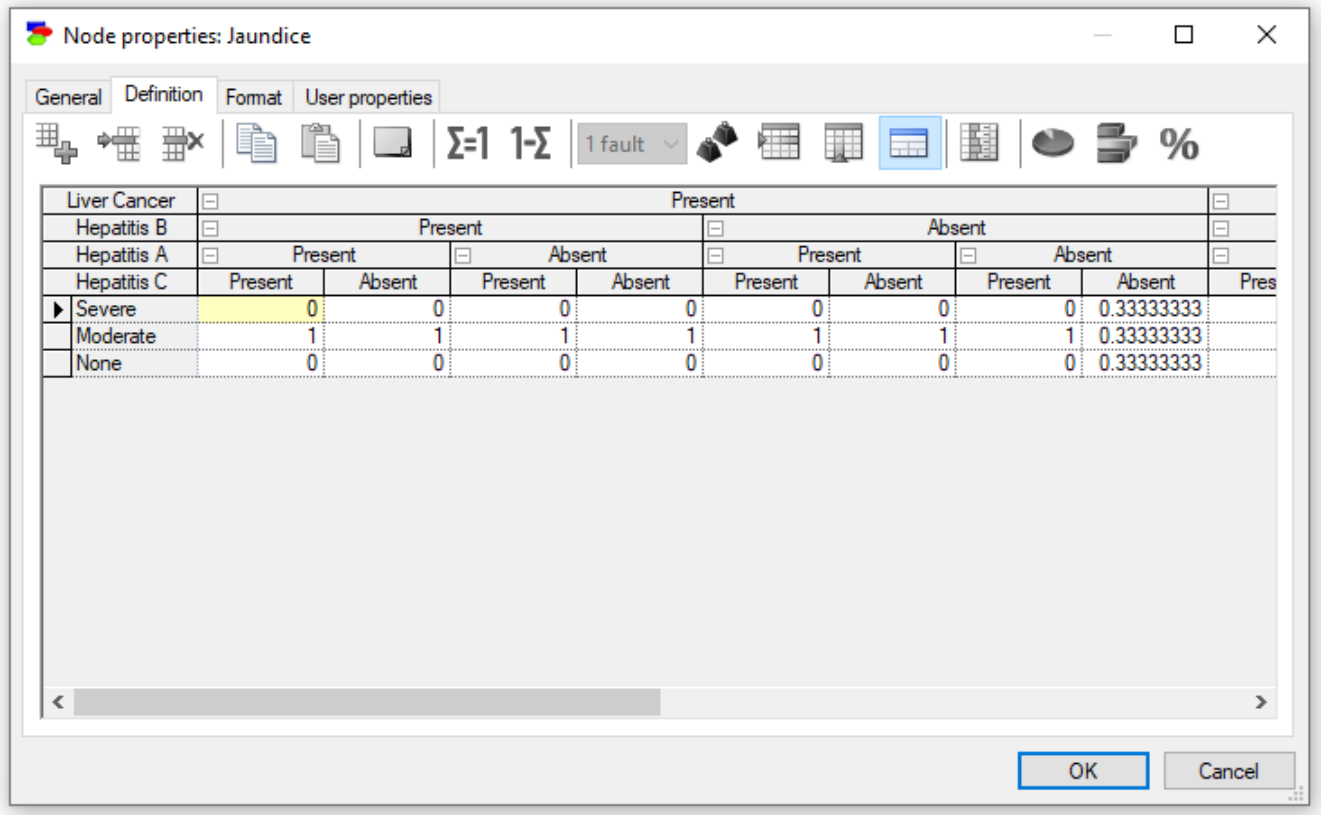


The weight are all equal to one by default and they are displayed in the header of the definition table.

The *average*-type *Noisy-Adder* node calculates the CPT from the *Noisy-Adder* parameters by taking the average of probabilities of the effect given each of the causes in separation. Please note that the leak node (and the leak probability) also take part in this calculation. Each of the nodes is taken with the weight specified in the *NoisyAdder Weights* dialog. You can examine the CPT that corresponds to the *Noisy-Adder* definition above by clicking on the *Show CPT* ( ) button. This is the CPT that corresponds to the *average* type



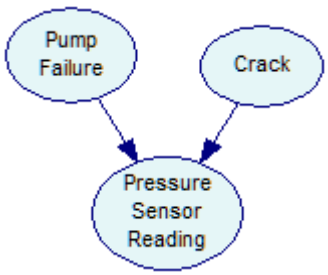
The *single fault* type *Noisy-Adder* node assumes that only one of the parent nodes will be in the non-distinguished state. If none of the modeled causes of the effect is active, the *Leak* node will be active. The CPT corresponding to the definition above looks as follows



Liver Cancer	Present								
Hepatitis B	Present				Absent				
Hepatitis A	Present		Absent		Present		Absent		
Hepatitis C	Present	Absent	Present	Absent	Present	Absent	Present	Absent	Pres
Severe	0	0	0	0	0	0	0	0.33333333	
Moderate	1	1	1	1	1	1	1	0.33333333	
None	0	0	0	0	0	0	0	0.33333333	

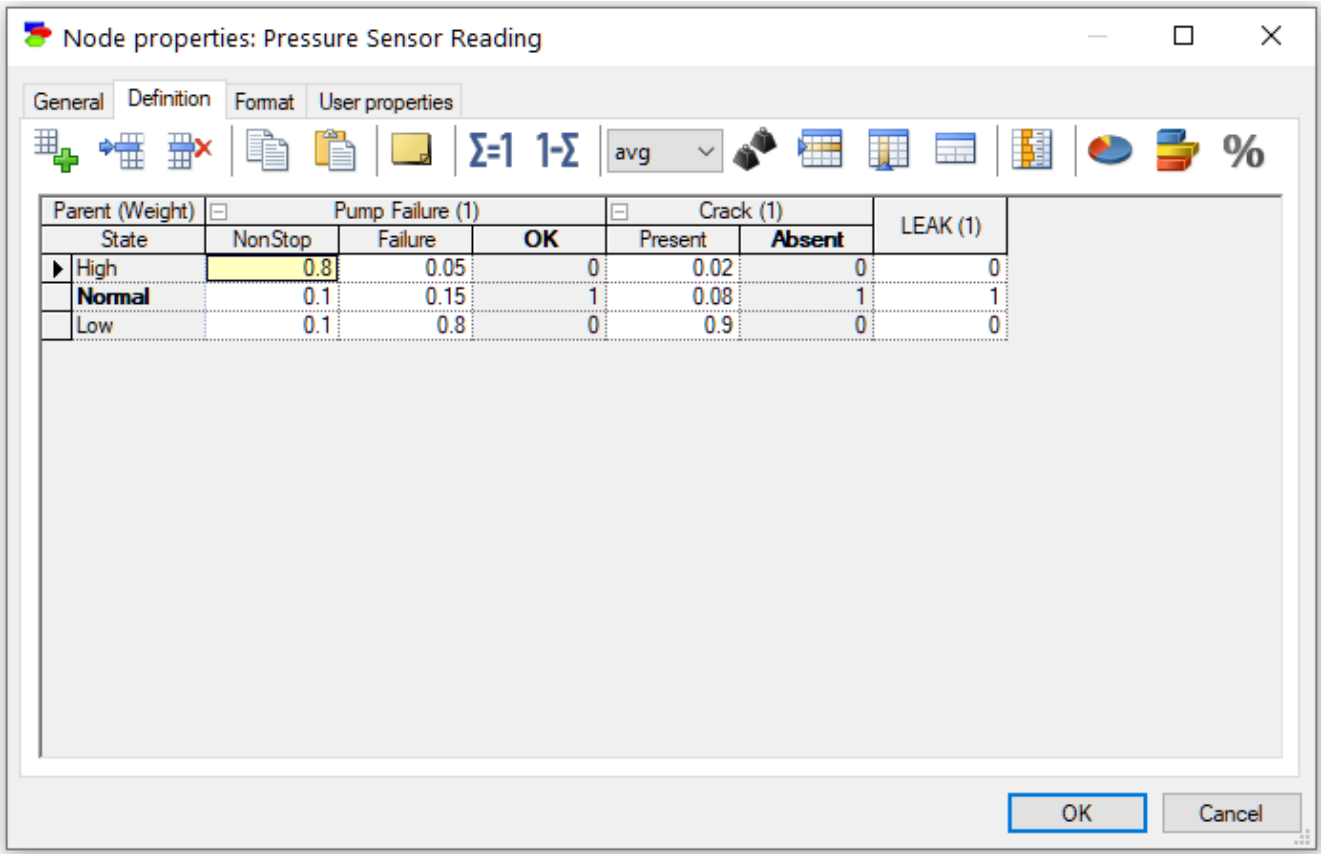
Noisy-Adder example

We present here a simple example offered by Adam Zagorecki (2010, pages 106-108) of a diagnostic model of an engine cooling system. Two possible problems, *Pump Failure* and a *Crack* in a cooling hose, may impact a *Pressure Sensor Reading* in the hose, which can be in three states: *High*, *Normal*, or *Low*. The pump can malfunction in two distinct ways: it may work *NonStop* or simply *Fail* and not work at all. It may also be fine, which is denoted by its state *OK*. A *Crack* in the hose can be *Present* or *Absent*. The structure of this problem is as follows:



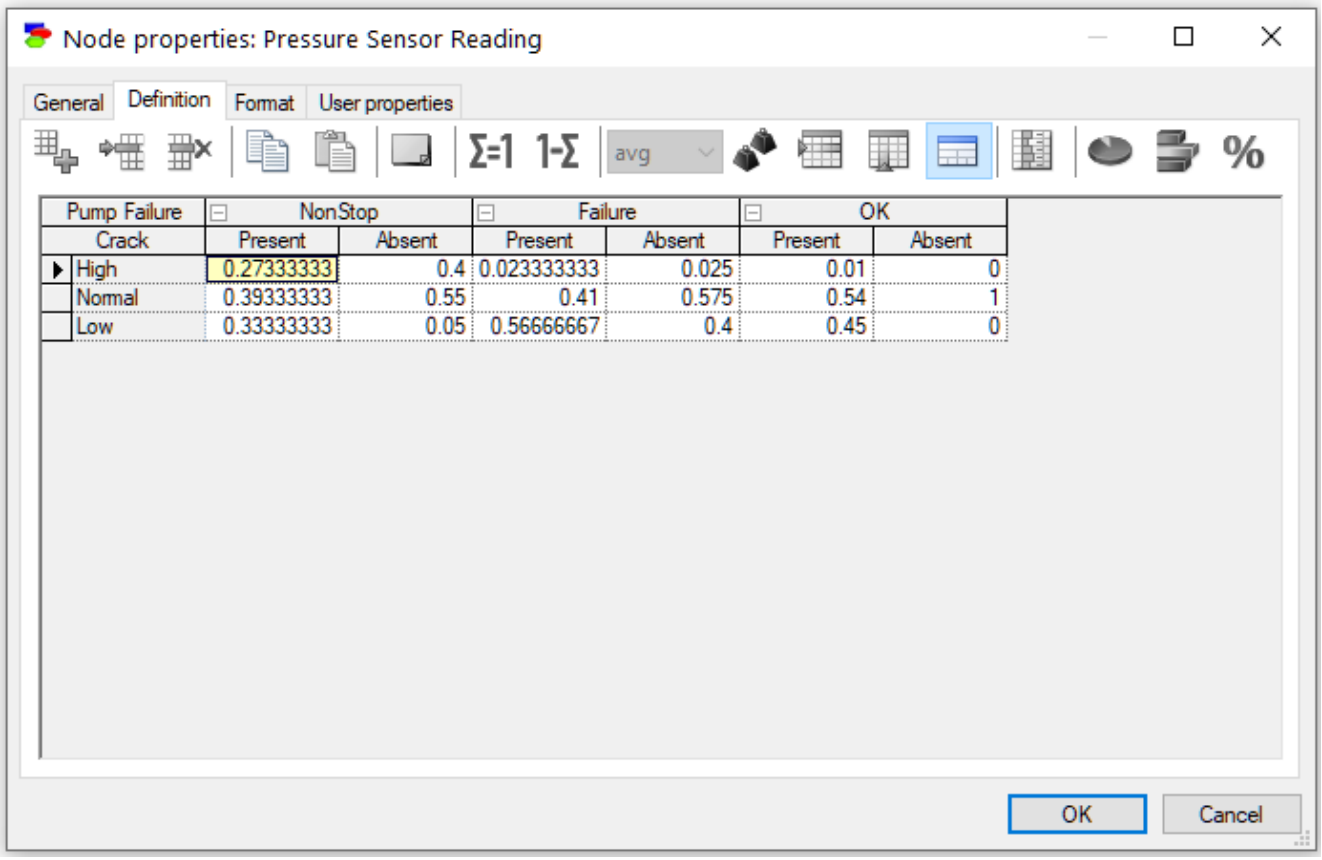
Please note that the *Noisy-MAX* model is not appropriate here, because the distinguished state of the *Pressure Sensor Reading* (*Normal*) does not correspond to the lowest value in the ordering relation. In other words, the neutral value is not one of the extremes, but instead lies in the middle. To apply the *Noisy-Adder* model, first we need to identify the distinguished states of the three variables, which are naturally *OK* for *Pump Failure*, *Absent*

for *Crack*, and *Normal* for *Pressure Sensor Reading*. The next step is deciding whether we should worry about a possible *Leak*, i.e., influence of unmodeled causes of the *Pressure Sensor Reading*. When there are no unmodeled causes of *Pressure Sensor Reading*, we have P(*Pressure Sensor Reading=Normal | Pump Failure=OK,Crack=Absent*)=1 and the other two states of Pressure Sensor Reading have probability zero. For simplicity, we will assume that there is no *Leak*. Let the definition of the node *Pressure Sensor Reading* be the following:



Parent (Weight)	Pump Failure (1)			Crack (1)		LEAK (1)
State	NonStop	Failure	**OK**	Present	**Absent**	
High	0.8	0.05	0	0.02	0	0
Normal	0.1	0.15	1	0.08	1	1
Low	0.1	0.8	0	0.9	0	0

We designate the distinguished states of the parents and the distinguished state of the child node (see Section *Chance-NoisyAdder nodes* in *Node properties*). The numbers in the table are self-explanatory and denote the probabilities of the three states of the *Pressure Sensor Reading* node given various states of each of the parents, assuming that the other parents are in their distinguished states. After all, Noisy-Adder is a canonical model. The CPT expressed by the above definition looks as follows:



Pump Failure	NonStop		Failure		OK	
Crack	Present	Absent	Present	Absent	Present	Absent
High	0.27333333	0.4	0.023333333	0.025	0.01	0
Normal	0.39333333	0.55	0.41	0.575	0.54	1
Low	0.33333333	0.05	0.56666667	0.4	0.45	0

Intuitively, the *Noisy-Adder* node combines the influences of its parents by averaging probabilities. In case where all active influences (i.e., non-distinguished states of the parents) imply high probability of one value, this value will have a high posterior probability and there will be synergy of the causes, similarly to the *Noisy-MAX* model. If the active parents indicate different states of the effect variable, the combined effect will be an average of the individual influences.

5.3.3 Multi-attribute utility nodes

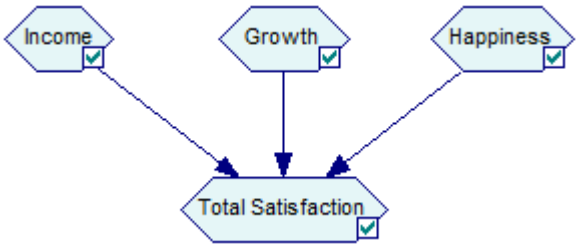
Some outcomes of decision problems involve several, possibly conflicting attributes. For example, outcome of a business decision may optimize productions costs, product quality, company image, and employee satisfaction. While such complex utility structures can be modeled directly by one utility function, it is usually easier for a decision maker to elicit utility functions over each of the attributes in separation and then combine them in a single multi-attribute utility function. MAU functions can be nested hierarchically, so it is possible to have any number of nodes structured hierarchically in such a way that nodes at the next level combine the utilities specified at the previous level.

GeNIe supports two types of multi-attribute utility nodes: (1) general *Multi-Attribute Utility* (MAU) nodes, which allow any explicitly framed multi-attribute utility function, and (2) a special case of MAU, the *Additive Linear Utility* (ALU) functions.

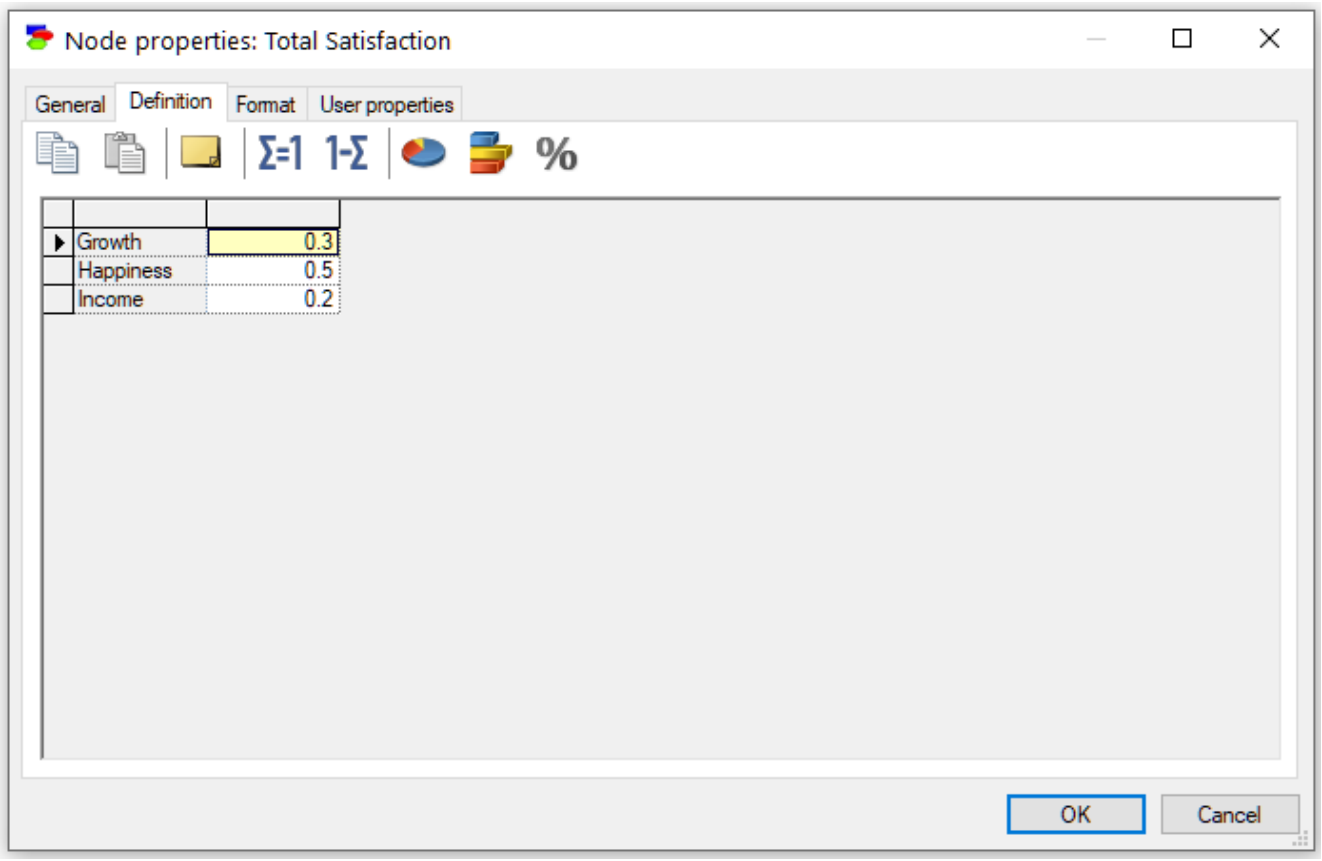
To create a MAU node, create a normal *Value node* first. There are two ways in which such a node can be turned into a MAU node: (1) by right-clicking on the node and choosing *Change Type* from the menu. Choose *MAU* or

ALU, and (2) when we draw an arc from a *Value* node to another *Value* node, GeNIe changes the type of the child node into an *ALU* node. If *MAU* is what you want, (1) is the only way you can achieve this transformation.

Consider the following simple model fragment containing four value nodes, three *Utility* nodes (*Income*, *Growth*, and *Happiness*) and one *ALU* node (*Total Satisfaction*), included among the example models as `ALU.xdsl`.



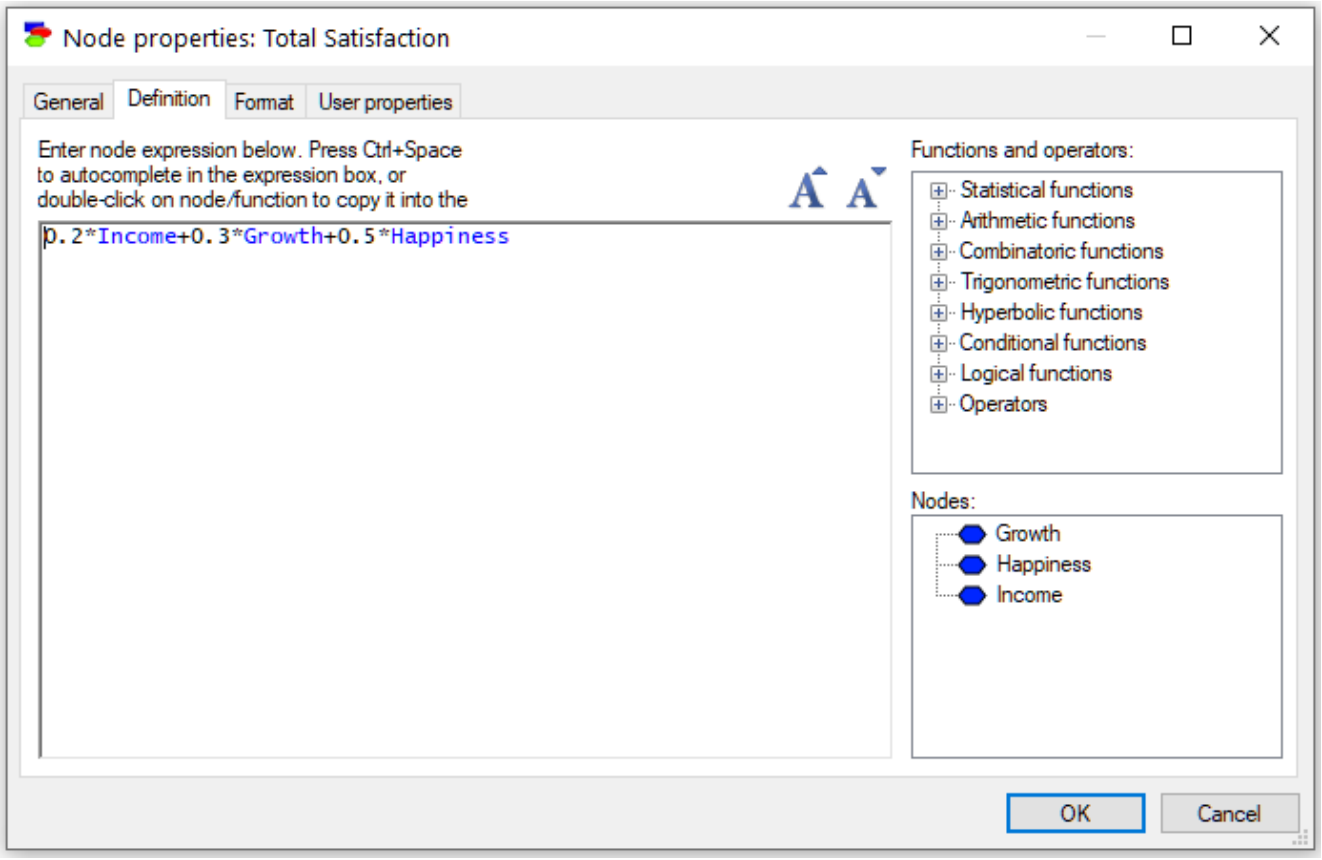
Let us define the coefficient (weights) of the linear multi-attribute additive linear utility function (*ALU*) in the following way:



Each of the parent nodes is an ordinary *Utility* node. The node *Total Satisfaction* is an *ALU* node that combines the parent utility nodes using the following linear function:

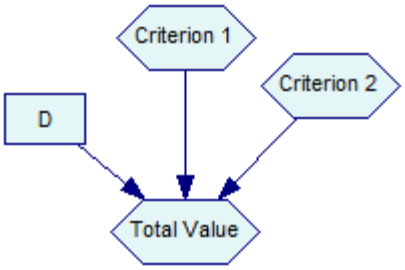
TotalSatisfaction = 0.5 * *Income* + 0.1 * *Growth* + 0.4 * *Happiness*

Let us change the type of this *ALU* node into *MAU*. Here is what the definition looks like now:

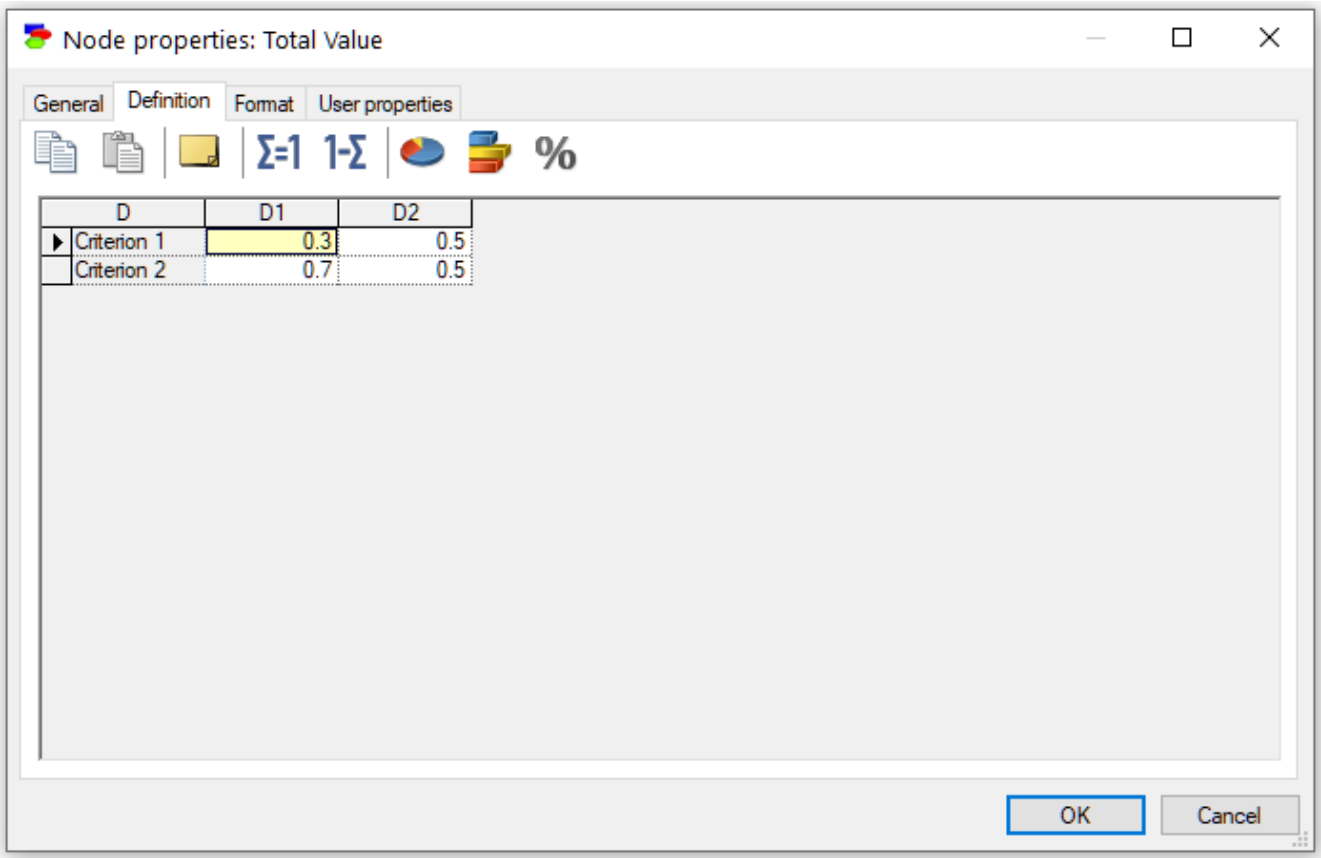


The definition is an equation involving the three parent utility nodes. There are no constraints on this equation, so any functional form can be used here, for example one that expresses the interaction between the three parent utility nodes as multiplicative. *MAU* nodes are thus as general as they get.

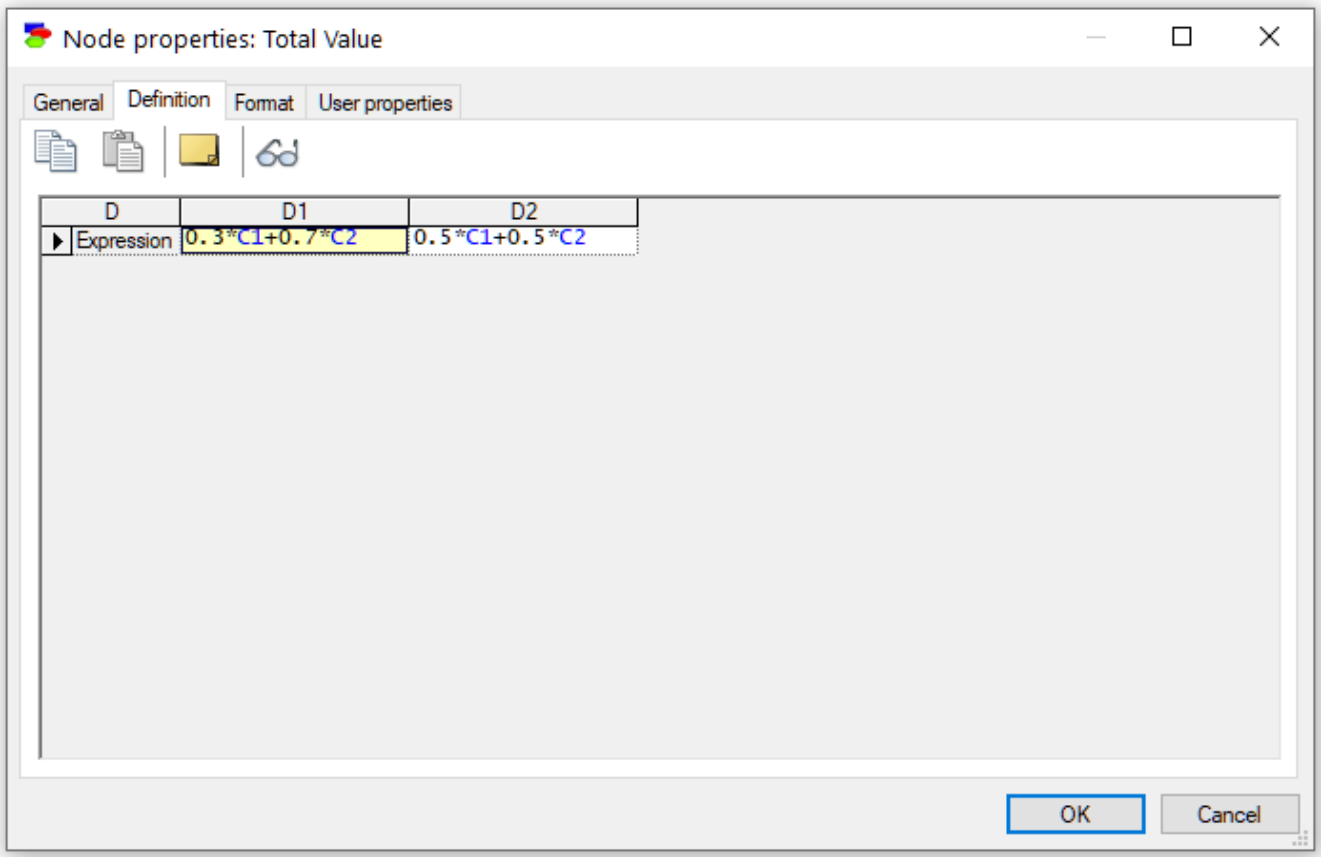
The only parents of ALU and MAU nodes can be utility or other ALU or MAU nodes. There is one exception that is allowed by GeNIe - decision nodes can also be parents of ALU and MAU nodes. Consider the following network fragment



The semantics of such a construct is that the decision node (*D*) changes the way that utility nodes (*Criterion 1* and *Criterion 2*) interact to produce *Total Value*. When *Total Value* is an ALU node, its definition looks as follows



We can see that the weights of the utility nodes (*Criterion 1* and *Criterion 2*) are different for the two decision options *D1* and *D2*. When *Total Value* is an MAU node, its definition looks as follows



While the two equations in the above model have the same form, just different coefficients, they can be completely different.

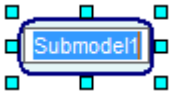
There is one more functionality added to GeNIe for user's convenience. GeNIe allows several childless value nodes, which is not really a proper model in decision analysis. This allows for separate optimization over several attributes. In case of such several childless nodes, GeNIe computes the expected utility for each of them, indexed by the states of all decision nodes (i.e., the decision alternatives) and the predecessors of the decision nodes (i.e., nodes that will be observed before the decision is made). The expected utilities can be examined in separation in each of these childless nodes. Expected utility displayed in the decision nodes, however, is a simple linearly-additive combination of the utilities of the childless nodes with the assumption that the weights are all equal to 1.0.

5.3.4 Submodels

Submodels are special types of nodes that host sub-graphs of the entire graph and make the *Graph View* structured hierarchically. Submodeling facilitates modularity in large models. The internals of a submodel, along with its structure can be examined in separation from the entire model.

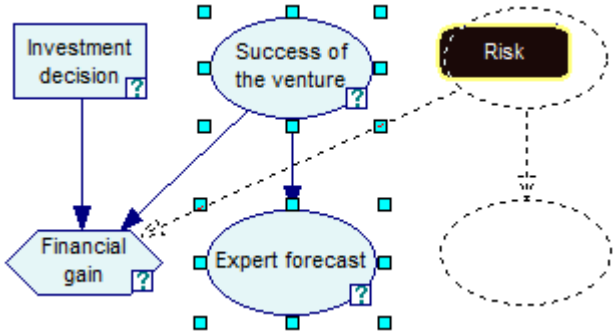
Creation of submodels, moving nodes, navigating through submodels

To create a submodel in GeNIe, select *Submodel* from the *Tools Menu* or the *Submodel* ( ) tool from the *Standard Toolbar* and click on the *Graph View*. You will see a new submodel.

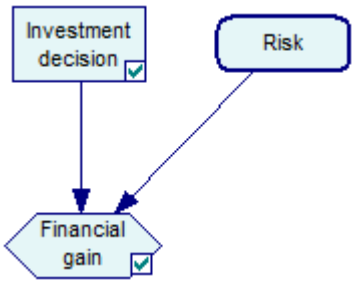


Submodel windows can be opened by double-clicking on the *Submodel* icon or right-clicking on the submodel icon and choosing *Open Submodel* from the *Submodel properties* menu.

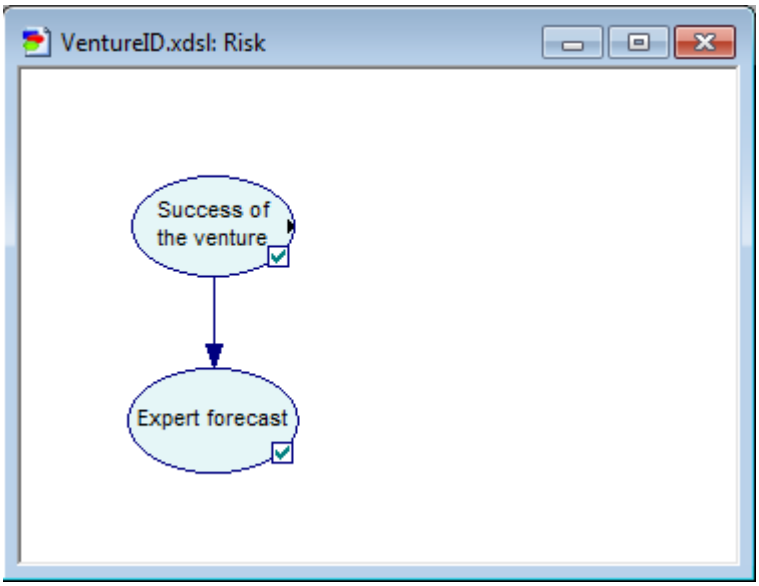
Nodes can be moved between submodels by selecting them in the source submodel, dragging, and dropping them in the destination submodel. For example, we might want to create a submodel for the variables *Success* and *Forecast* in the *influence diagram* model used in the *Creating an influence diagrams* section. We do this by creating a submodel node, renaming it to *Risks*, and then dragging and dropping the nodes *Success of the venture* and *Expert Forecast* to the new submodel.



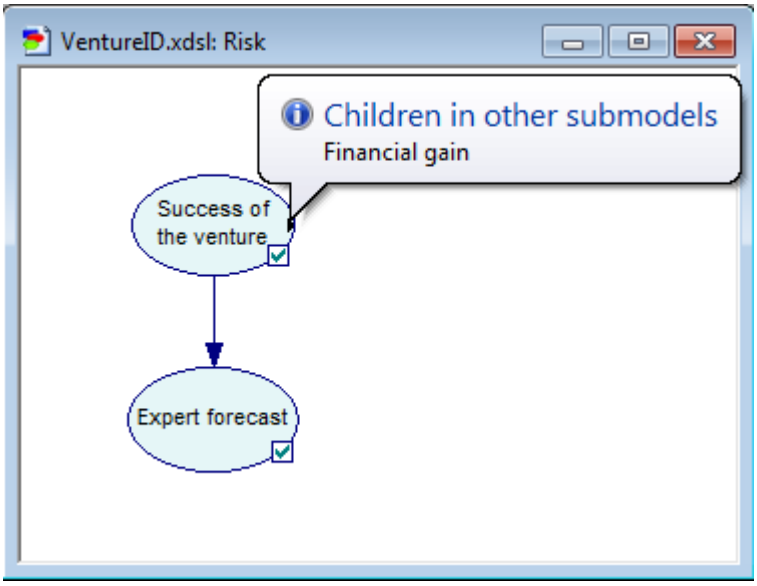
The resulting model will look as follows:



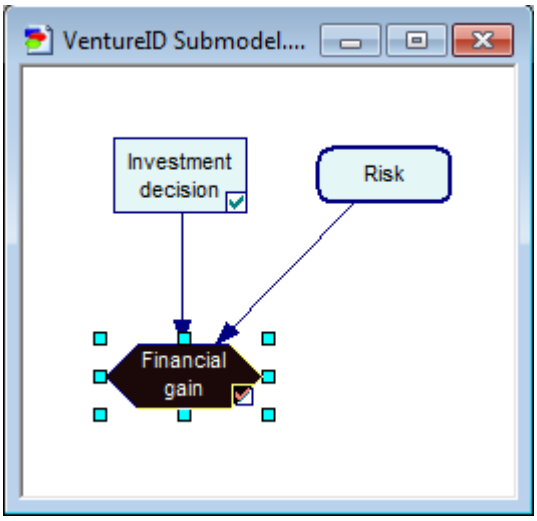
Submodels are opened by double-clicking on them. Double-clicking on the submodel *Risks* yields the following:



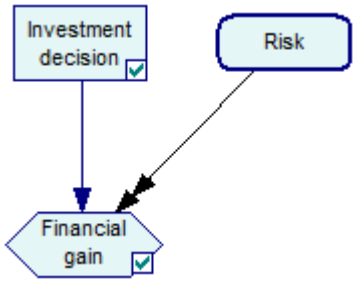
One thing that becomes less clear in submodels is the connections that the submodel has with the external world. GeNIe does not show arcs that are coming from outside or that go to the outside world. It does let the user know that there are such connections. First of all, by showing these connections as coming into the submodel node (note the arcs from the submodel node *Risks* coming into the nodes *Invest* and *Gain* at the main model level). It also adds small triangle-shaped marks on the left and right sides of the internal submodel nodes showing that there are incoming and outgoing arcs respectively. The user can examine these connections by placing the cursor over the small triangle. This will display the name of the child of the node in another submodel as follows:



You can locate the child of this node by right clicking and choosing *Locate Child* from the *Node Pop-up* menu. Select the appropriate name from the *Locate Child* sub-menu to flash the child on the screen as shown below:



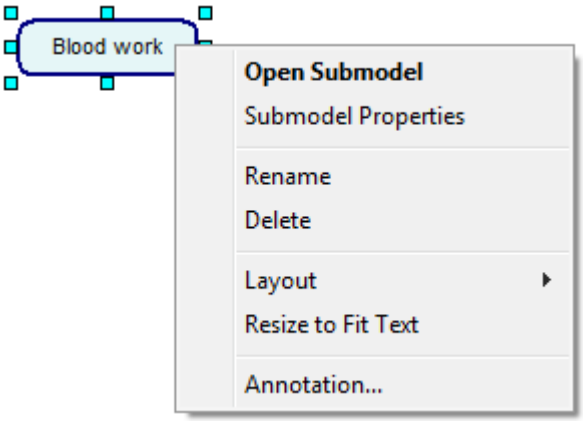
It is possible to add arcs between nodes that are located in different submodels in the very same way that arcs are added between nodes in the same submodel. When more than one arc is drawn between a submodel and a node, then GeNIe draws a double arrow arc from the submodel to the node as shown below:



All the above functions can be also performed through GeNIe *Tree View*. The above model is included among the example models as `VentureEVPISub`.

Submodel properties

Submodel properties sheet can be displayed by right clicking on the name of the submodel in the *Tree View* or right clicking on the submodel icon in the *Graph View*. This will display the *Submodel Pop-up* menu. Select *Submodel Properties* from the menu.

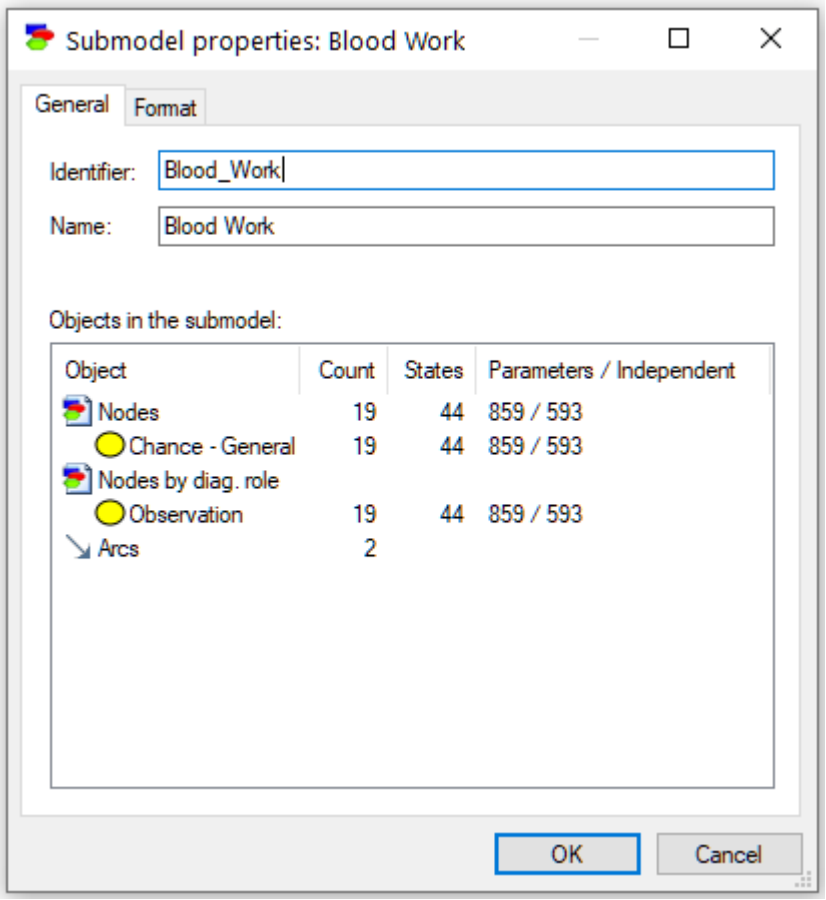


Note : *Double clicking on the submodel will open the graph view of the submodel, it will not open the Submodel properties sheet.*

The *Submodel properties* sheet consist of two tabs: *General* and *Format*. The *General* tab allows to change the identifier and the name of the submodel, the *Format* sheet allows to change the graphical properties of the submodel icon and is identical to the property sheet described in node property sheets.

General tab

The *General* tab displays the *Identifier* and the *Name* of the submodel, along with the submodel's basic statistics.



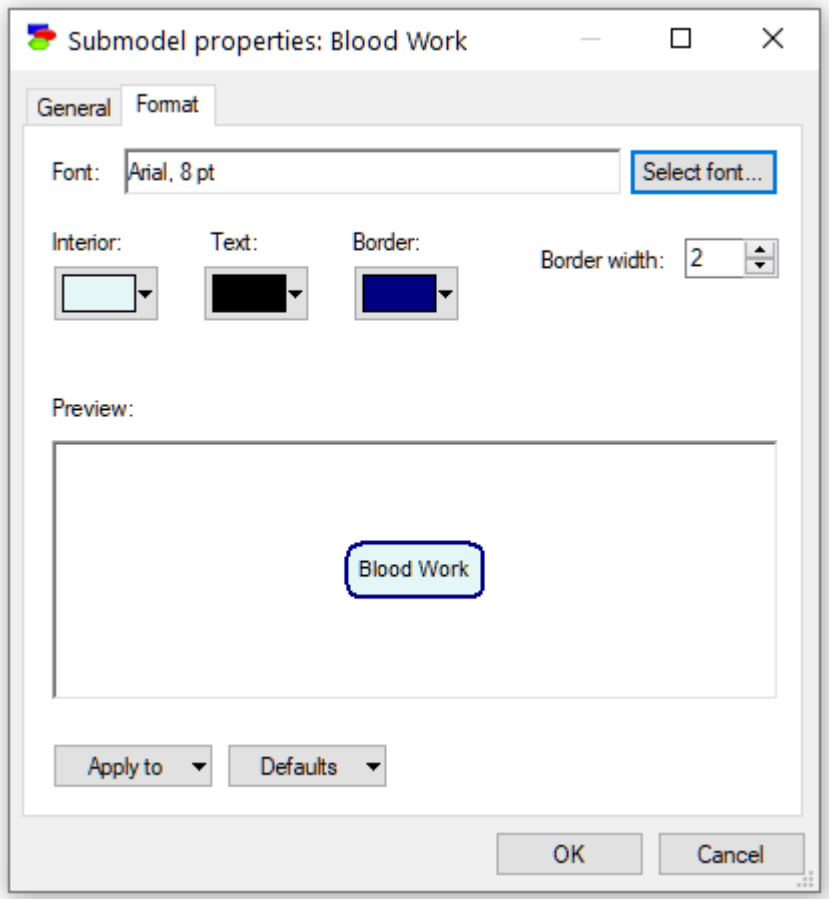
Identifier displays the identifier for the submodel, which is user-specified. Identifiers must start with a letter, and can contain letters, digits, and underscore (_) characters. The identifier for the network shown above is *Blood_work*.

Name displays the name for the submodel, which is user-specified. There are no limitations on the characters that can be part of the name. The name for the network shown above is *Blood work*.

The *Objects in the submodel* lists counts of various types of objects and numerical parameters in the submodel. They give an idea of the submodel's complexity.

Format tab

The *Format* tab allows to modify the visual properties of the submodel icon, i.e., how the submodel icon is displayed in the *Graph View*.



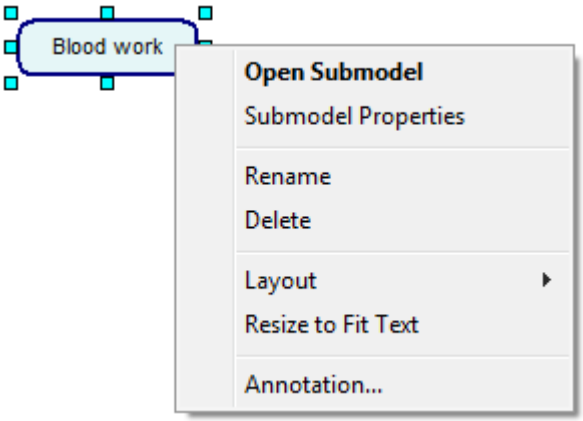
The *Format* tab is similar in function to the *Format* tab of the *Node properties* sheet.

Other submodel operations

Submodel Popup menu is slightly different for the *Graph View* and the *Tree View*.

Submodel Pop-up menu for the *Graph View*

The *Submodel Pop-up* menu for the *Graph View* can be displayed by right clicking on the submodel icon in the *Graph View*.



Open Submodel opens the submodel in a new *Graph View* window.

Submodel Properties opens the *Submodel Properties* sheet.

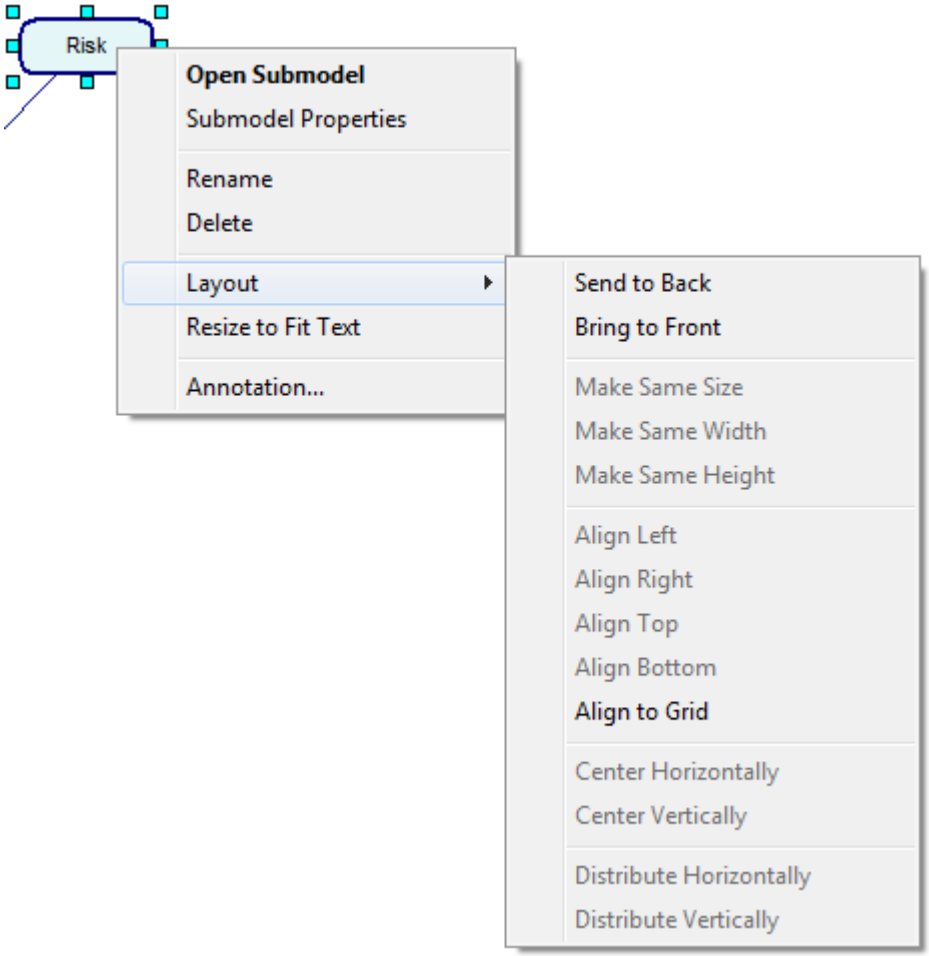
Rename allows you to rename the submodel by placing the submodel icon in edit mode. You can also rename a submodel by modifying the *Name* field in *Submodel Properties* sheet.

Delete deletes the selected submodel.

Resize to Fit Text resizes the submodel icon so that it fits the entire submodel name.

Annotation... opens up the annotation dialog so that you can add an annotation to the submodel (see *Annotations* section for more information).

Layout submenu



Most of the commands on the *Layout* submenu are the same as those in the *Layout* menu. The only commands here that are not found in the *Layout* menu are:

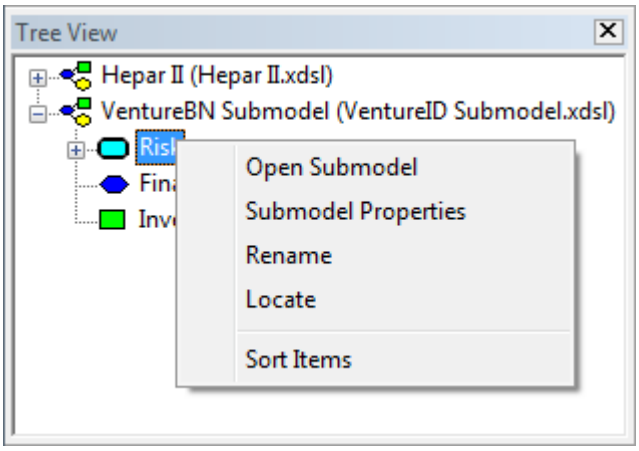
*Make Same Size (*enabled only if two or more items are selected in the *Graph View*) resizes the selected items so that they are the same size as the item that was right clicked.

*Make Same Width (*enabled only if two or more items are selected in the *Graph View*) resizes the selected items so that they are the same width as the item that was right clicked.

*Make Same Height (*enabled only if two or more items are selected in the *Graph View*) resizes the selected items so that they are the same height as the item that was right clicked.

Submodel Popup* menu for the *Tree View

The *Submodel* pop-up menu for the *Tree View* can be displayed by right clicking on the submodel name.



The two commands that are not available in the *Graph View* are:

Locate locates the submodel in the *Graph View* of its parent model or submodel. Once located, the submodel icon flashes several times on the screen to attract user attention.

Sort Items sorts the list of nodes or submodels of the current submodel listed in the tree view in alphabetical order.

5.3.5 Arcs

Arcs between nodes denote direct influences between them.

One remark about editing diagrams is that GeNIe does not allow moving arcs between nodes, i.e., it is not possible to select and drag the head or the tail of an arc from one node to another. If this is what you want, the way to accomplish this task is to first delete the original arc and then create a new arc. These operations have serious consequences on the definitions of the nodes pointed by the heads of the arcs - deleting an arc deletes a portion of the definition of the node, adding an arc leads to a default extension of that definition. GeNIe tries to minimize the impact of adding and deleting arcs in terms of changing the conditional probability distributions. Whenever you add an arc, which amounts to adding a dimension to the child variable's conditional probability table, GeNIe will duplicate the current table, preserving the numbers from the original table. Whenever you delete an arc, which amounts to reducing a dimension of the child variable's conditional probability table, GeNIe will remove only a part of the table, preserving the rest.

Normally, an arc in a Bayesian network or an influence diagram denotes an influence, i.e., the fact that the node at the tail of the arc influences the value (or the probability distribution over the possible values) of the node at the head of the arc. These arcs are drawn as solid lines. Some arcs in influence diagrams have clearly causal meaning. In particular, a directed path from a decision node to a chance node means that the decision (i.e., a manipulation of the graph) will impact that chance node in the sense of changing its probability distribution.

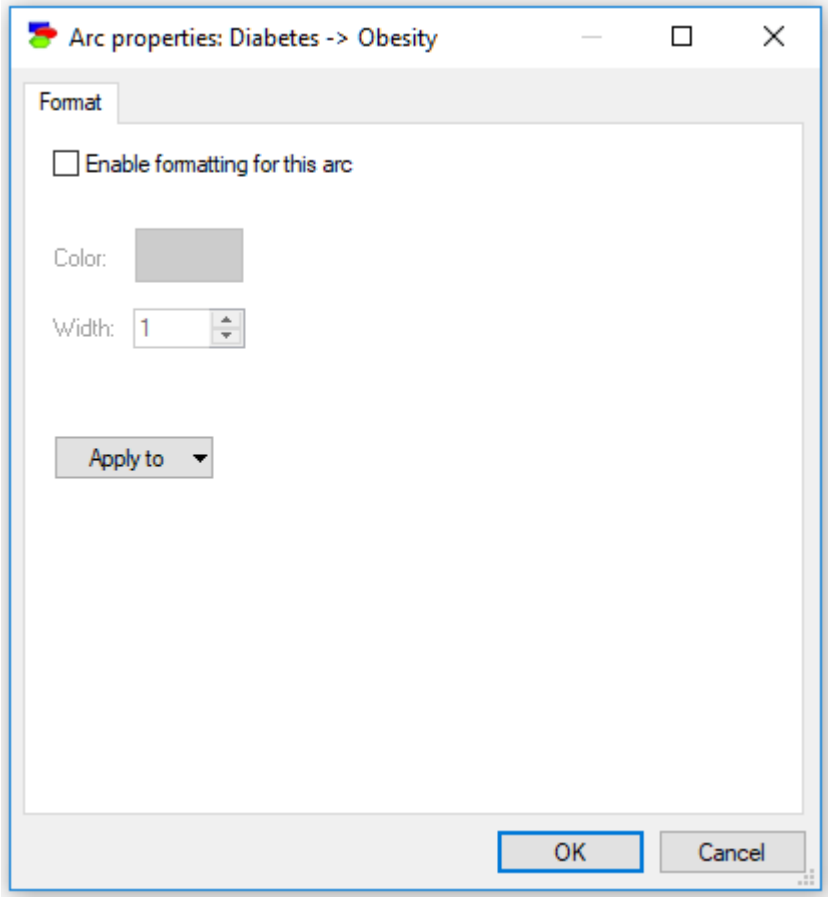
Arcs coming into decision nodes have a different meaning. Because decision nodes are under decision maker's control, these arcs do not denote influences but rather temporal precedence (in the sense of flow of information). The outcomes of all nodes at the tail of informational arcs will be known before the decision will need to be made. In particular, if there are multiple decision nodes, they need to be all connected by

informational arcs. This reflects the fact that the decisions are made in a sequence and the outcome of each decision is known before the next decision is made. Informational arcs are drawn as dashed lines.

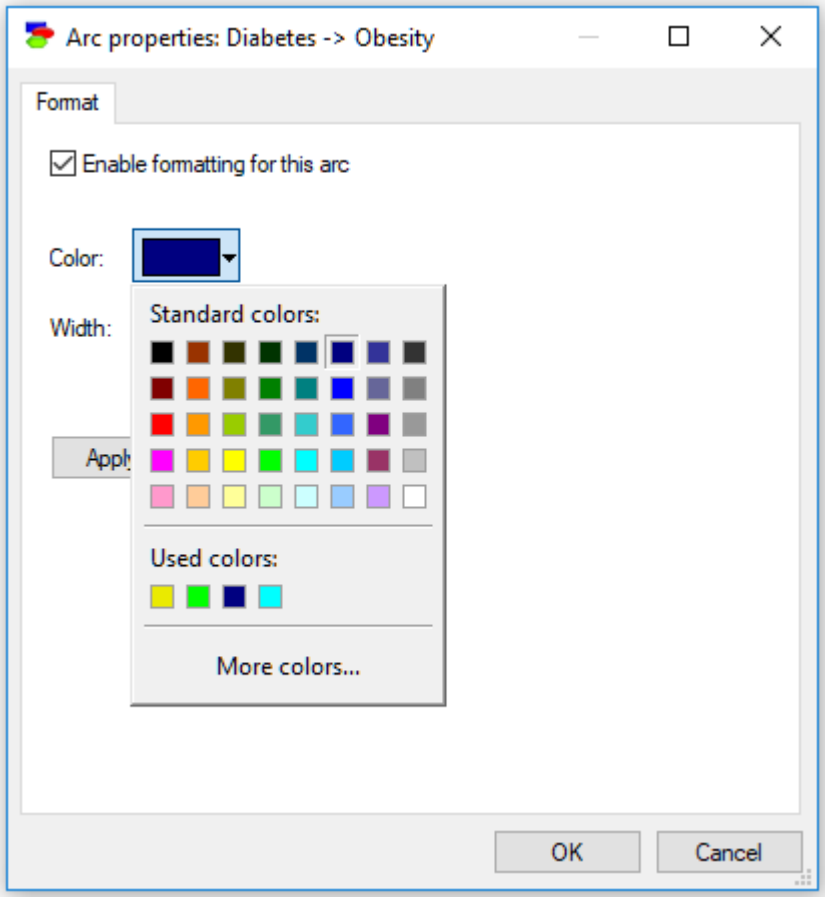
GeNIe displays also arcs between nodes and submodels. An arc from a node *N* to a sub-model *S* means that at least one node in *S* depends on *N*. An arc from a sub-model *S* to a node *N* means that *N* depends on at least one node in *S*. An arc from a sub-model *S1* to a sub-model *S2* means that there is at least one node in *S2* that depends on at least one node in *S1*. Arcs between sub-models can be double-headed, in which case the relations listed above is reciprocal. For example, a double-headed arrow between a node *N* and a sub-model *S* means that there is at least one node in *S* that depends on *N* and that there is at least one node in *S* that influences *N*. GeNIe does not show arcs that are coming from the outside of the current sub-model window. Existence of arcs coming from outside of the current sub-model and ending in a node in the current sub-model is marked by a small triangle on the left-hand side of the node. Existence of arcs originating in a node in the current sub-model and ending in a higher-level sub-models is marked by a small triangle on the right-hand side of the node. These links can be followed by right-clicking on the small triangles.

Whether arcs are influences or are informational, cycles in the graph, i.e., directed paths that start and end at the same point, are forbidden (unless the graph is dynamic, such as a Dynamic Bayesian Network, which is covered in a separate section). GeNIe will not allow you to draw cyclic graphs. Please note that even though GeNIe will enforce that the underlying graph is acyclic, you may still be able to observe cyclic graphs involving submodel nodes. This is due to the meaning assigned to arcs between submodels.

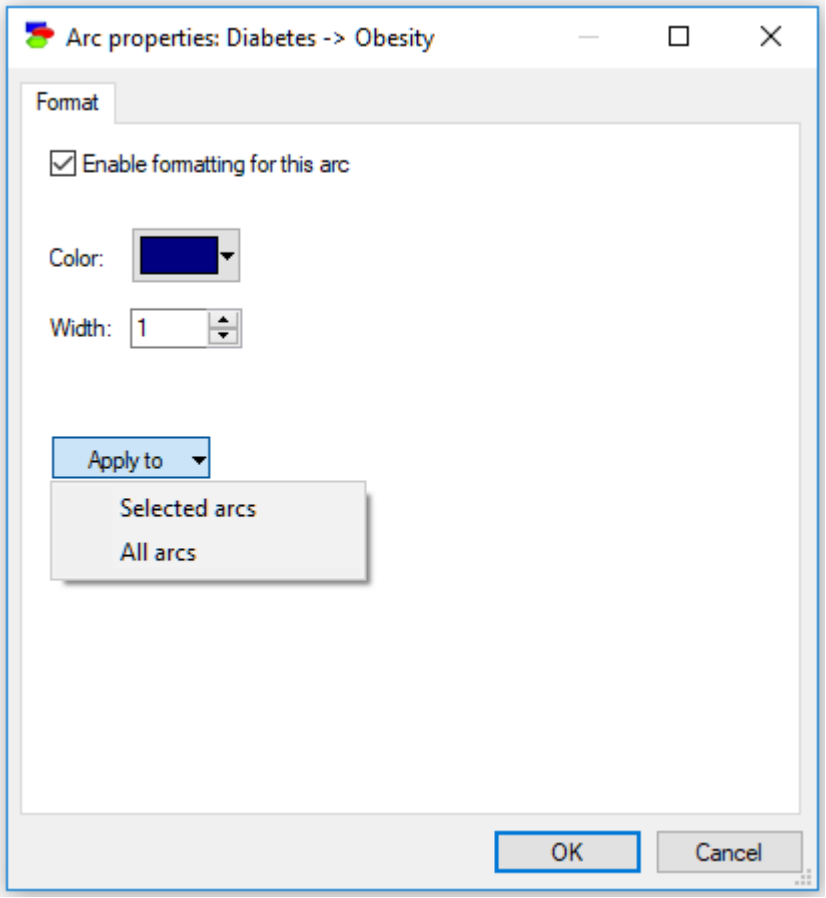
Arcs in GeNIe have dark navy color and are one pixel wide. It is possible to modify both the color and the width of selected arcs using the *Arc properties* dialog. The dialog can be invoked by right-clicking on the arc and selecting *Arc properties* or simply double-clicking on the arc (the former is a better choice in case we want to modify properties of several selected arcs).



The *Arc propeties* dialog is inactive by default and can be activated by checking the *Enable formatting of this arc* check box.



The dialog allows for changing the color and the width of the arc. Changes can be applied to *Selected arcs* or to *All arcs*

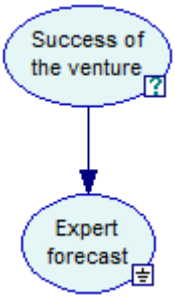


5.3.6 Node status icons

Each node in the *Graph View* is marked by one or more *node status icons*. These are tiny icons displayed in the lower right corner of the node icon. There are six different node status icons: *Observed*, *Implied*, *Controlled*, *Target*, *Valid*, and *Invalid*. We will explain their meaning on simple examples.

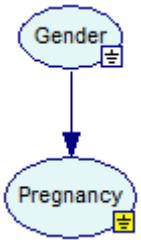
Observed Status Icon

The *Observed* status icon (⊜) is displayed when the user enters evidence into a node and it signifies that the node is an evidence node. Node *Expert forecast* in the following model is an evidence node and is marked with the *Observed* status icon:



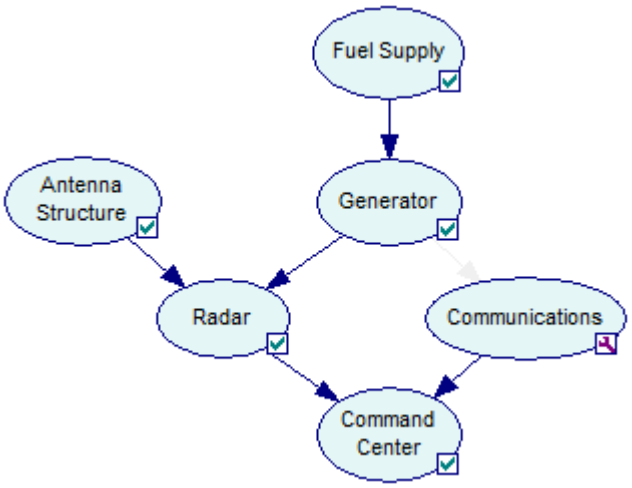
Implied Status Icon

Sometimes, observing a node implies the values of other nodes. For example, observing that a patient is biologically male in a medical decision support system will imply that he is not pregnant. This is possible because the program realizes that for a male patient pregnancy is impossible. GeNIe marks the nodes whose values are implied by observations of other nodes by the *Implied* status icon (). The *Implied* status icon is identical to the *Observed* status icon, but it is yellow in color. In the following example, the *Gender* of the patient has been observed to be *Male* and the value of the *Pregnancy* variable has been determined by GeNIe to be *False*.



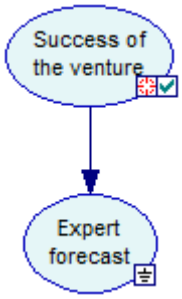
Controlled Status Icon

Controlling the value of a node is different from observing it (see *Changes in structure* section). When a node is controlled, GeNIe marks it with the *Controlled* status icon (). Node *Communications* in the model below has been manipulated and marked as *Controlled*.



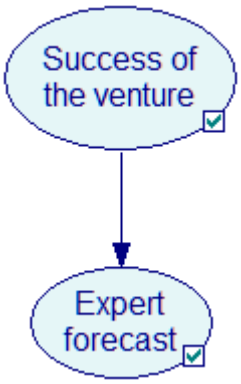
Target Status Icon

The *Target* status icon ( ) has to do with GeNIe's support for relevance reasoning. Very often in a decision support system, only a small number of variables are of interest to the user. When the model used by the system is large, the amount of computation to update all variables may be prohibitive while being in a large part useless. GeNIe allows to designate those variables that are of interest to the user as targets. Target nodes are always guaranteed to be updated by the program during its updating procedure. Other nodes, i.e., nodes that are not designated as targets, may be updated or not, depending on the internals of the algorithm used, but are not guaranteed to be updated. When no nodes are designated as targets, GeNIe assumes that all variables in the model are of interest to the user, i.e., all of them are targets. The node *Success of the venture* has been marked as *Target* in the following model:



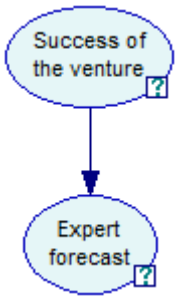
Valid Status Icon

The *Valid* status icon ( ) marks those nodes that have been updated and have their values available for examination. *Valid* status icon can appear together with the *Target* mark, but never with the *Observed* and *Implied* marks (note that when a node is observed, its value is known). To make the value valid, simply update the model. Both variables in the following model are marked by the *Valid* status icon, which means that their marginal probability distributions are available and ready for examination:



Invalid Status Icon

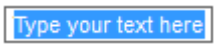
The *Invalid* status icon (?) marks those nodes that need updating, i.e., their values (or the probability distributions) are invalid and cannot be inquired. *Invalid* mark can appear together with the *Target* mark, but never with the *Observed* and *Implied* marks (note that when a node is observed, its value is known). To make the value valid, which will make the *Invalid* icon disappear, simply update the model. Both variables in the following model are marked by the *Invalid* status icon, which means that their marginal probability distributions need recomputing:



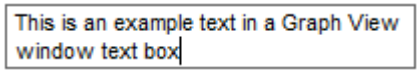
5.3.7 Text boxes

You can put an arbitrary text in the *Graph View* window. This text may be useful as a comment explaining the details of the model. To add a text you need to create a *Text box*.

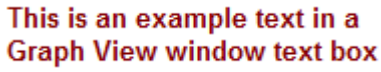
Select the *Text box* (T) button from the *Standard Toolbar* or *Tools Menu* (note that the cursor shape changes). The *Text box* button will become recessed. Move the mouse to a clear portion of the *Graph View* and click the left mouse button. You will see a rectangle appear on the screen:



You can type any text inside the box. You can use any of the regular editing tools, such as cursor movement, selection, *Cut*, *Copy*, and *Paste*. After you have typed your text, press *Enter* or click anywhere outside of the box. The box may look as follows:

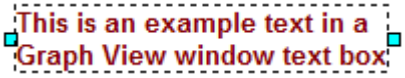


You can change the font type, style, size, and color using appropriate tools from the *Format Toolbar*. Shown below are some effects of using the *Format Toolbar*:



You can always come back to editing the text in the box by double-clicking on the box. Double-clicking makes the text visible for editing again.

You can select the box by single-clicking on it. A selected box shows its boundaries and two small squares at its left and right boundary.



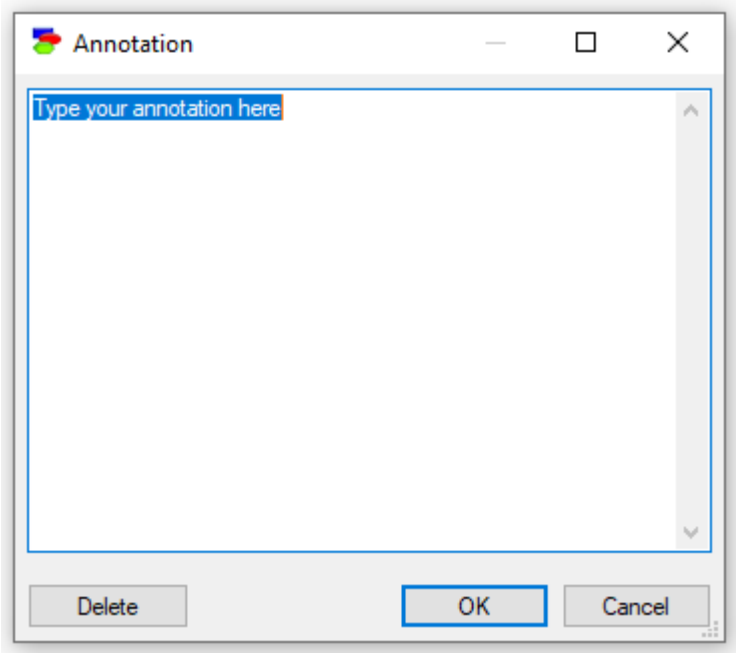
You can re-size a selected box by dragging on one of these small squares. You can delete it by pressing the *Delete* key. You can drag the box to a new location by clicking on it and holding the mouse button down while moving it to a new location.

5.3.8 Annotations

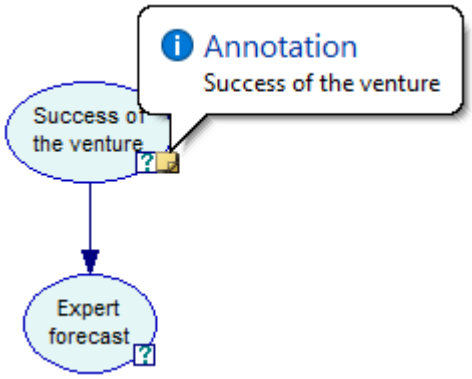
GeNIe supports annotations for nodes and states of nodes. Following the idea that one of the main goals of a model is documenting the decision making process, annotations are useful for explaining function of nodes and states, or to note down just about anything the user feels is important regarding the node or state.

Annotations for nodes

You can specify annotations for nodes by right clicking on the node and selecting *Annotation* from the *Node Pop-up* menu. This will display the annotation box as shown below:



Enter the annotation in the white blank space and click on *OK* to save it. Once an annotation has been saved against a node, GeNIe displays a small yellow note ( ) beside the status icon of the node. To view the annotation for a node, hover the cursor over the note. GeNIe will display the annotation as follows:

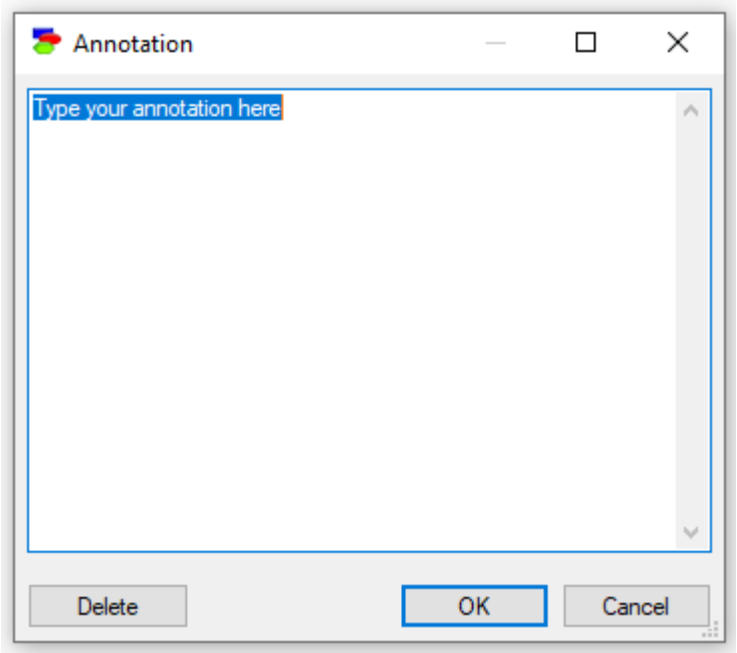


Double click on the note to display the annotation box, which will allow you to edit the annotation.

To delete an annotation, double click on the note and delete all contents of the annotation box.

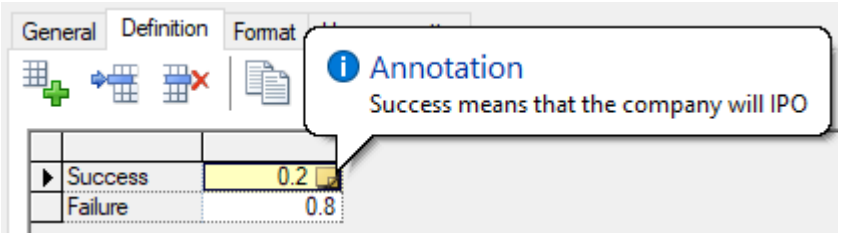
Annotations for states

You can annotate individual states of a node too. To annotate a selected state of a node open the *Node Properties* sheet for that node. Select the *Definition Tab*. Click on the *Annotation* ( ) button. It will open a annotation box as shown below:



You can enter the annotation in the blank white space and click *OK* to save it. Click on *Delete* to remove an existing annotation. After a state is annotated, a small yellow note ( ) will appear beside the value for the state as shown below:

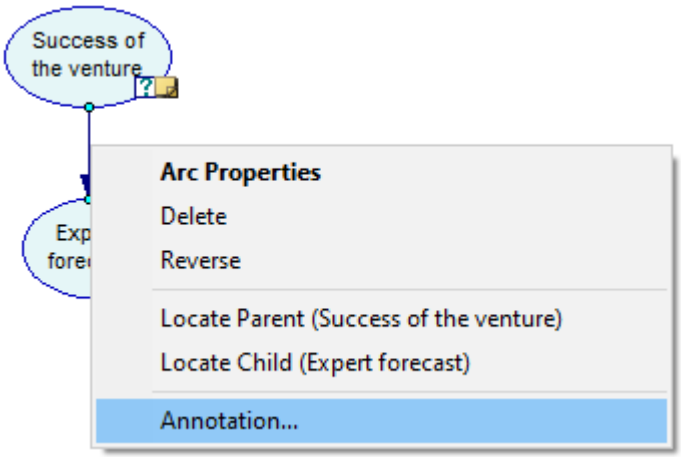
Hovering the mouse over the note icon will display the annotation text as shown below:



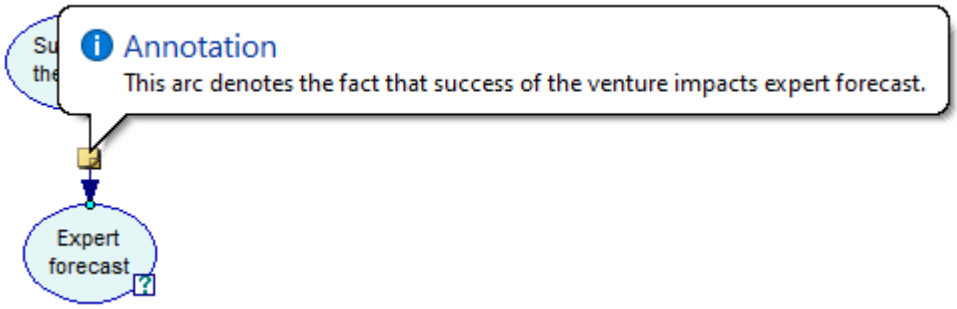
To edit the annotation click on the note.

Annotations for arcs

You can annotate arcs between nodes as well. To annotate an arc, right click on it and choose *Annotate* from the pop-up menu that shows:



Choosing *Annotate* brings up the annotation window. Hovering over the yellow stick-it-note shows the text of the annotation.



5.4 Model and component properties

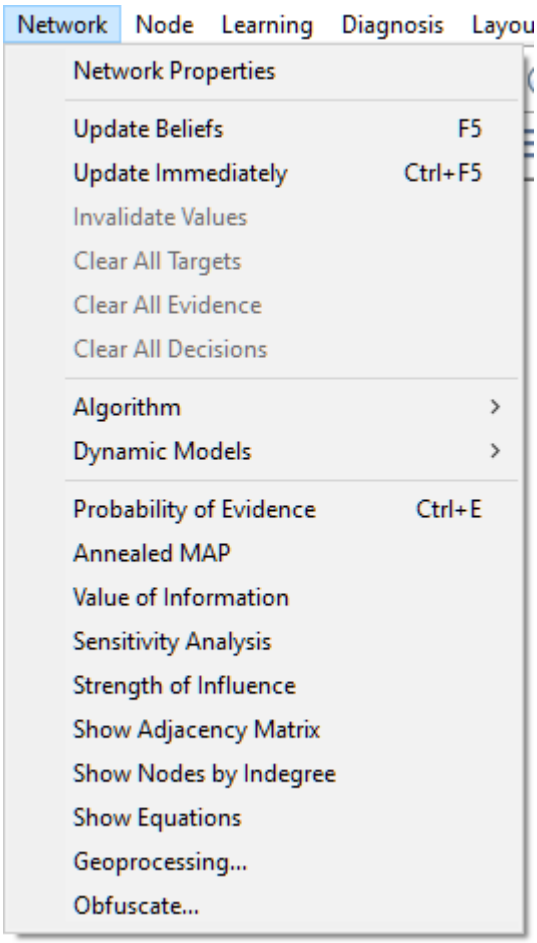
5.4.1 Network properties

Network properties sheet summarizes all properties that are specified at the network level. It can be invoked in three ways:

1. Double clicking on a clear area of the network in the graph view.

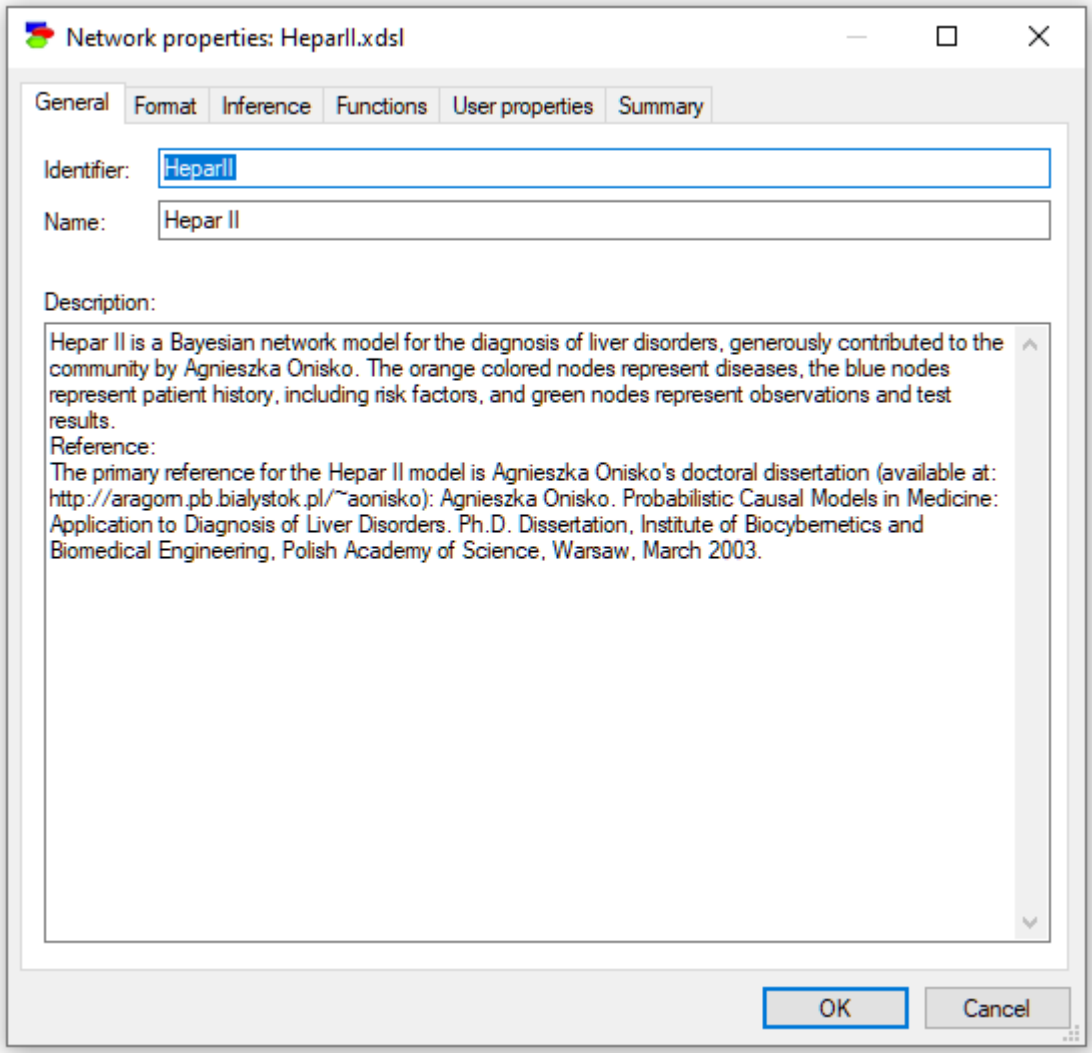
2. Right clicking on the name of the network in the *Tree View* or right clicking on a clear area of the network in the *Graph View*. This will display the *Network Pop-up* menu. Select *Network Properties* from the menu.

3. Select *Network Properties* from the *Network Menu* as shown below.



The *Network properties* sheet, once opened, consists of several tabs.

Shown below is a typical *Network Properties Sheet* with the *General* tab:



General tab

The *General* tab of *Network properties* (shown above) consists of the following fields:

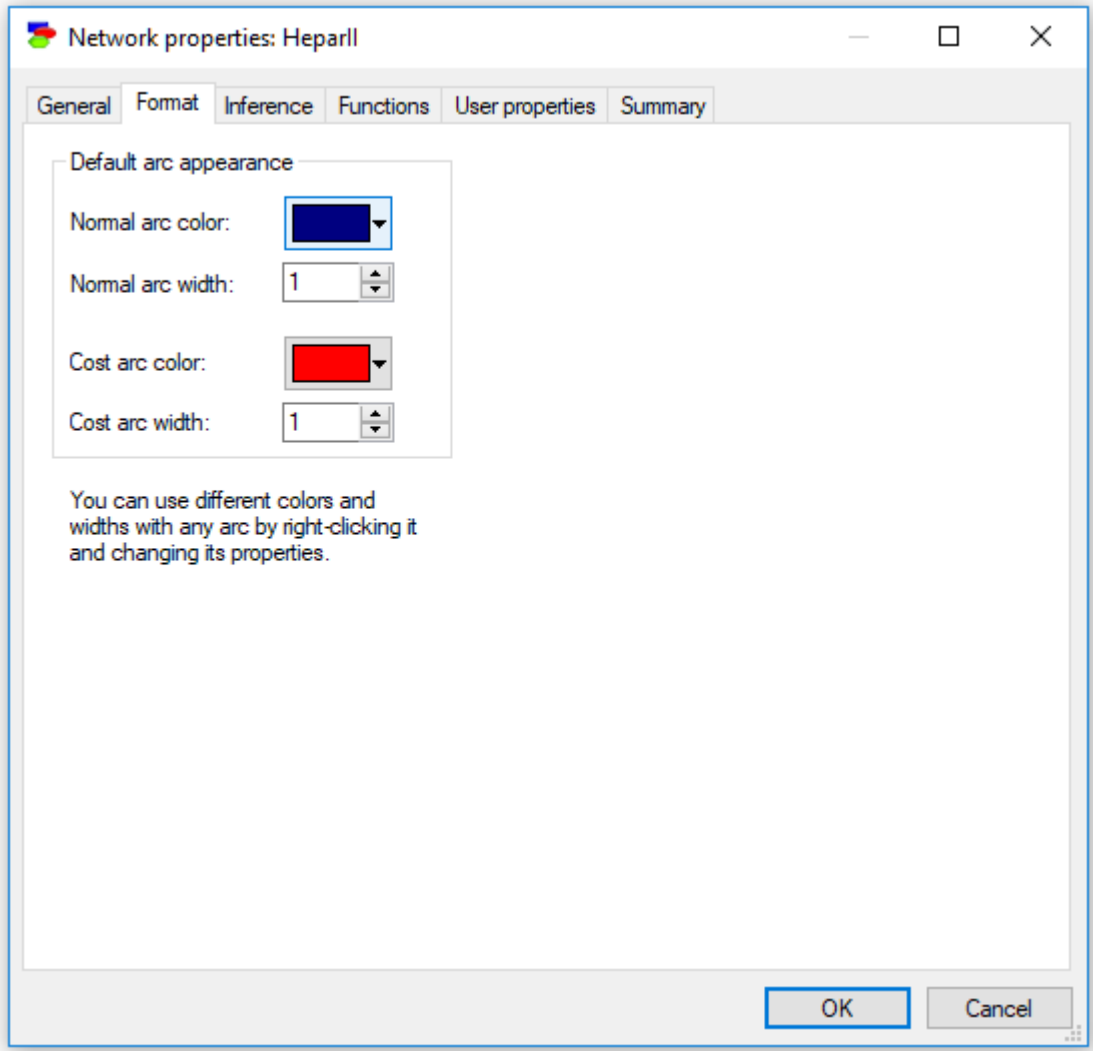
Identifier displays the identifier for the network, which is user-specified. Identifiers must start with a letter, and can contain letters, digits, and underscore (_) characters. The identifier for the network shown above is *HeparII*.

Name displays the name for the network, which is user-specified. There are no limitations on the characters that can be part of the name. The name for the network shown above is *Hepar II*.

Description is a free text describing the network. Please note that a model is a documentation of the problem and use descriptions generously.

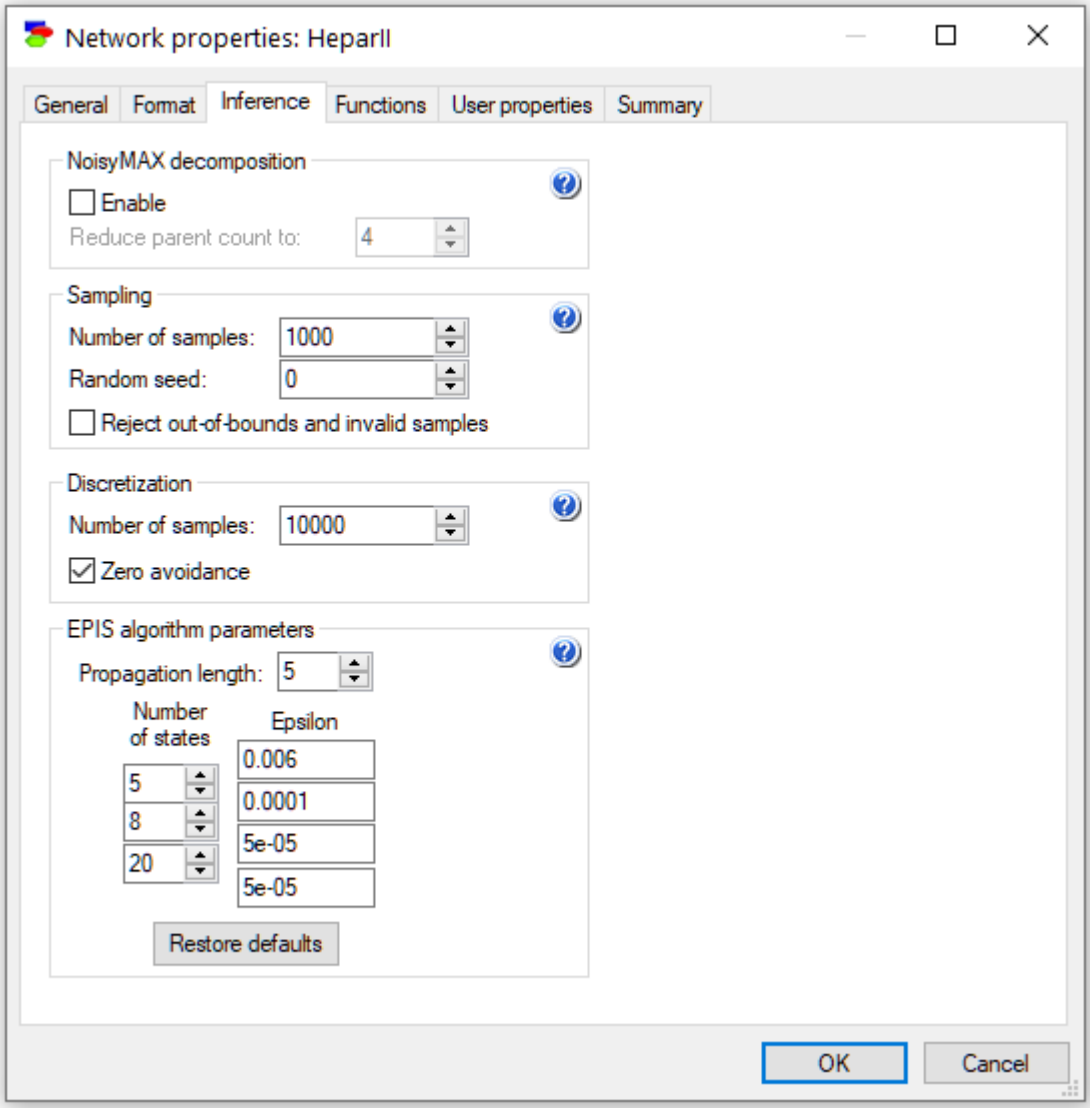
Format tab

The *Format* tab of *Network properties* (shown below) allows for choosing the default appearance of arcs in the network. These defaults can be overridden in each individual model arc.



Inference tab

The *Inference* tab allows the user to set the various parameters of GeNIe inference algorithms.



The *Inference* tab allows you to enable Noisy-MAX decomposition, which makes inference in models with Noisy-MAX nodes more efficient and, in very complex models, even possible at all. The *Reduce parent count to* parameter controls the decomposition of Noisy-MAX models. After the decomposition, no Noisy-MAX node will have more than that number of parents. The default value of 4 is reasonable but if inference is slow on your network, you may want to explore other values.

Number of samples sets the number of samples used in each execution of a sampling algorithm. All sampling algorithms are approximate. The number of samples determines the precision of the results (the more samples, the more precise the result, although no simple formula exists that translates the number of samples into precision) but at the same time it determines the computation time (the more samples, the longer the running time - running time is pretty much linear in the number of samples). Error in the posterior marginal probabilities calculated by the EPIS-BN algorithm reduces as a square root of the number of samples. Please keep in mind that if you would like to recompute the values with the new number of samples (larger number of samples give you a higher precision), you will need to invoke the updating algorithm again, either by pressing the

Update tool ( ) or, when *Update immediately* option is on, by invalidating all values (see the *Invalidate values* command) and by this forcing GeNIe to recompute them during the next run of the algorithm.

Random seed allows for making stochastic sampling algorithms (such as AIS-BN or EPIS-BN) to be replicable, i.e., to produce the same results each time they are run in a give situation (i.e., with the same model and the same set of observations). Any number that is different than zero makes the results replicable. A random seed of zero uses a truly random number seed and makes the results different each time the algorithm is run. This value is a property of the model and is saved in the .xdsl file.

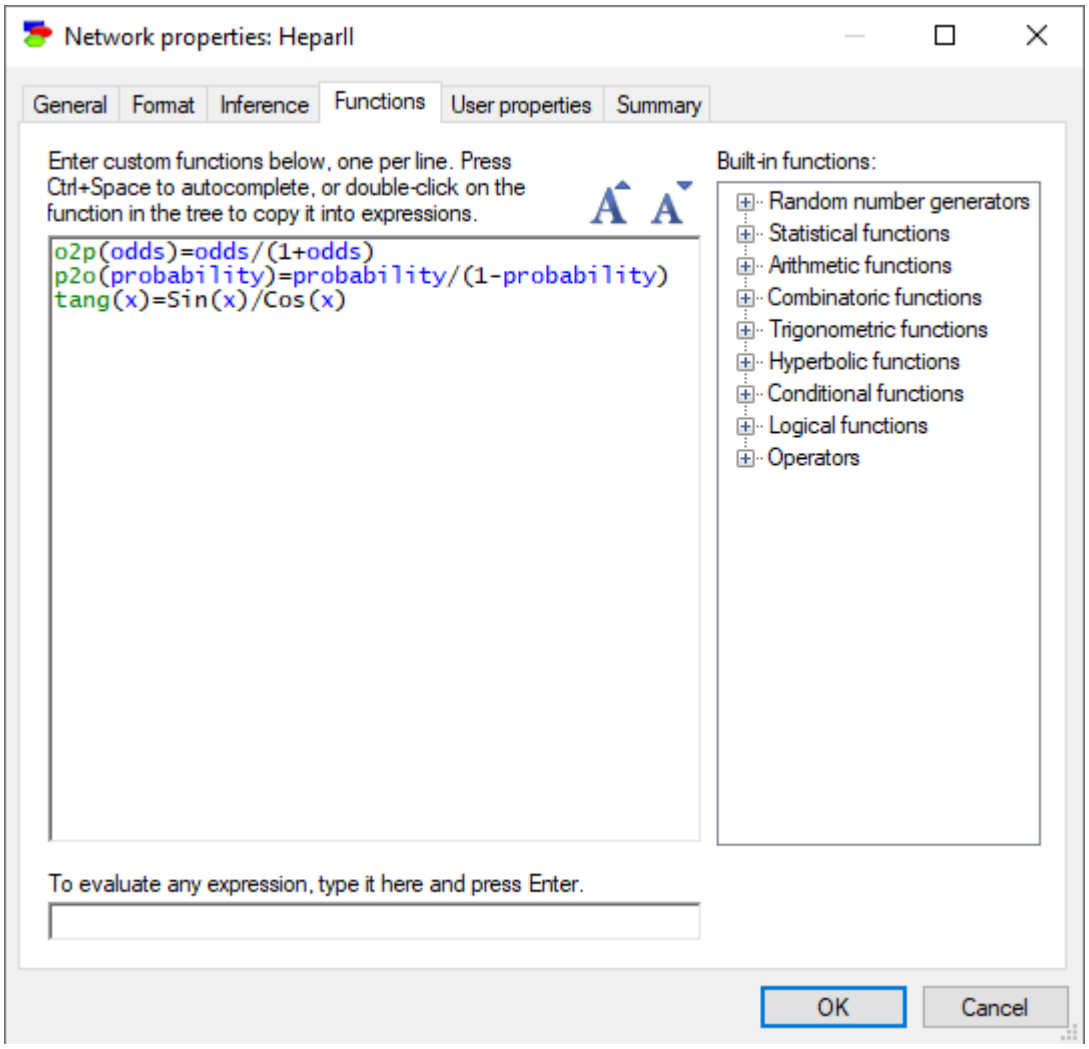
Reject out-of-bound and invalid samples option, when checked, causes every sample that is invalid (e.g., *not a number* when the algorithm encounters division by zero or square root of a negative number) or out of bounds (defined on the Definition tab of the *Node properties*) to be discarded. Having this option checked improves the convergence of the algorithm at the expense of computation time. Outlier samples make minimal contribution to the posterior probability distribution in importance sampling, while costing as much as all other samples in computation.

Discretization pane serves to set the number of samples in deriving the CPTs for discretized continuous nodes (see also the *Autodiscretization* algorithm). Higher number of samples takes more time but leads to a better precision of parameters. Zero avoidance, when checked, avoids placing zeros in the CPTs, even if there are no samples generated for a given parameter.

EPIS algorithm parameters are used to fine tune the EPIS algorithm execution. The EPIS algorithm uses *Loopy Belief Propagation* (LBP), an algorithm proposed originally by Judea Pearl (1988) for polytrees and later applied by others to multiply-connected Bayesian networks. EPIS uses LBP to pre-compute the sampling distribution for its importance sampling phase. *Propagation length* is the number of LBP iterations in this pre-computation. EPIS uses Epsilon-cutoff heuristic (Cheng & Druzdzel, 2000) to modify the sampling distribution, replacing probabilities smaller than epsilon by epsilon. The table in the *Sampling* tab allows for specifying different threshold values for nodes with different number of outcomes. For more details on the parameters and how they influence the convergence of the EPI-BN algorithm, please see (Yuan & Druzdzel 2003).

Functions tab

The *Functions* tab allows the user to define functions that will be used in the model. The following sheet contains three definitions of functions: `o2p()`, `p2o()`, and `tang()`. GeNIe functions may not be recursive but they may refer to functions defined earlier (in the same *Functions* tab).

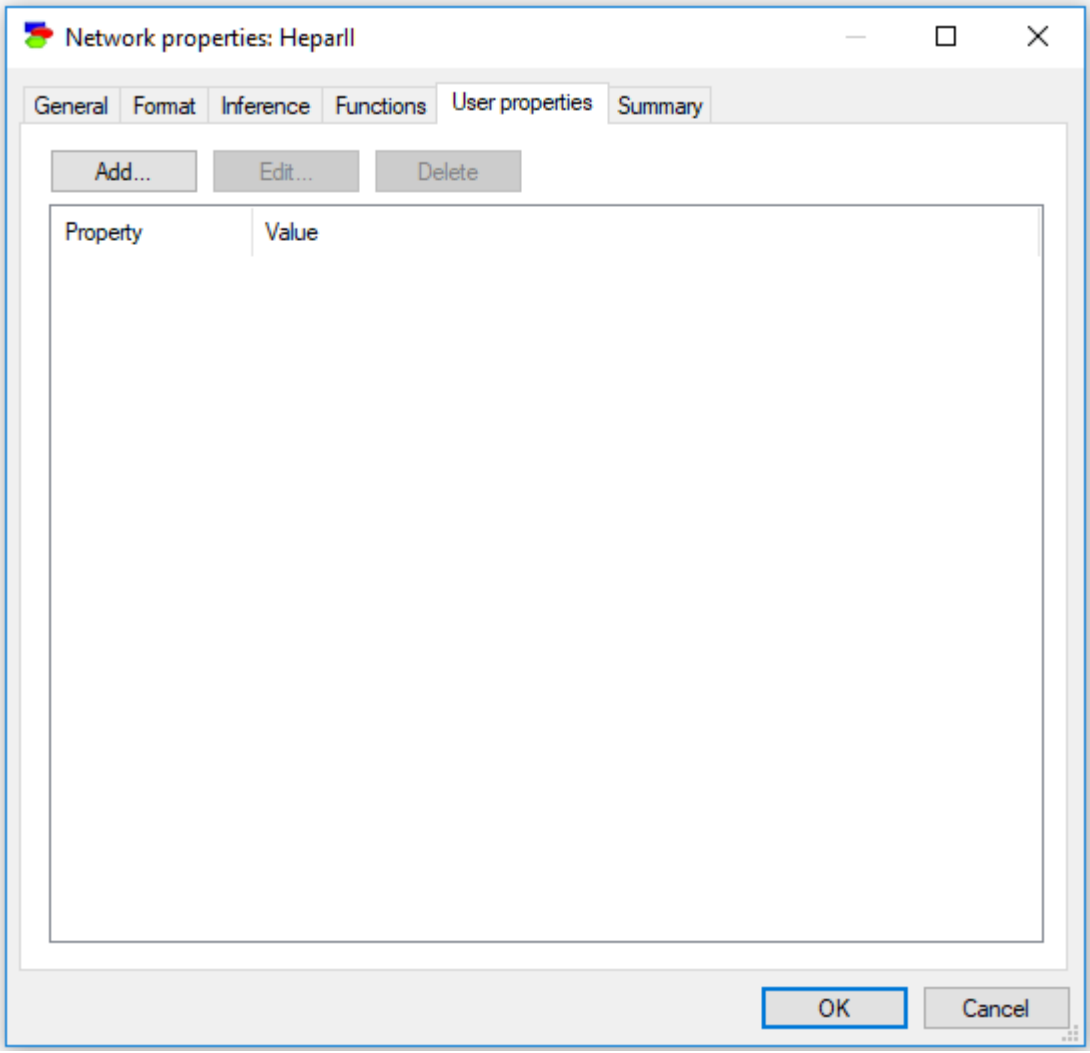


It is possible to test the newly defined functions by evaluating calls to them with concrete values of parameters at the bottom of the tab. Please note that arguments of the evaluated function have to be constants.

User-defined functions (also called *Custom functions*) are defined per network and stored along with the network. GeNIe copies user-defined functions between networks when nodes using them are copied from a source network and pasted in the destination network. In this case, conflicts may occur, for example a function with the same definition may exist already in the destination network. GeNIe performs a simple check of the name of the function, its number of parameters, and determinism status (i.e., whether the function makes calls to random number generators) and will use the definition in the destination network if these three agree. Otherwise, it will copy the definition from the source to the destination network. We recommend that meaningful names are used and different definitions of the same function in different models are avoided.

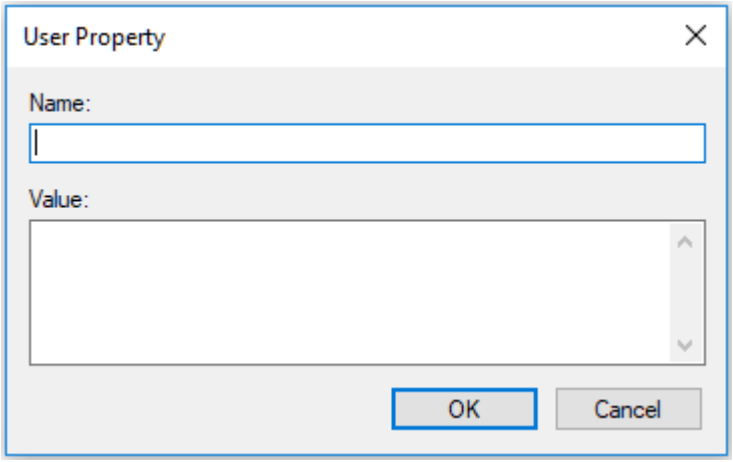
User properties tab

User properties tab allows the user to define properties of the model that can be later retrieved by an application program using SMILE.



For example, we can add a property *AUTHOR* with the value "*https://www.bayesfusion.com/*". Neither GeNIe nor SMILE use these properties and they provide only placeholders for them. They are under full control and responsibility of the user and/or the application program using the model. GeNIe and SMILE support only editing and storing/retrieving them.

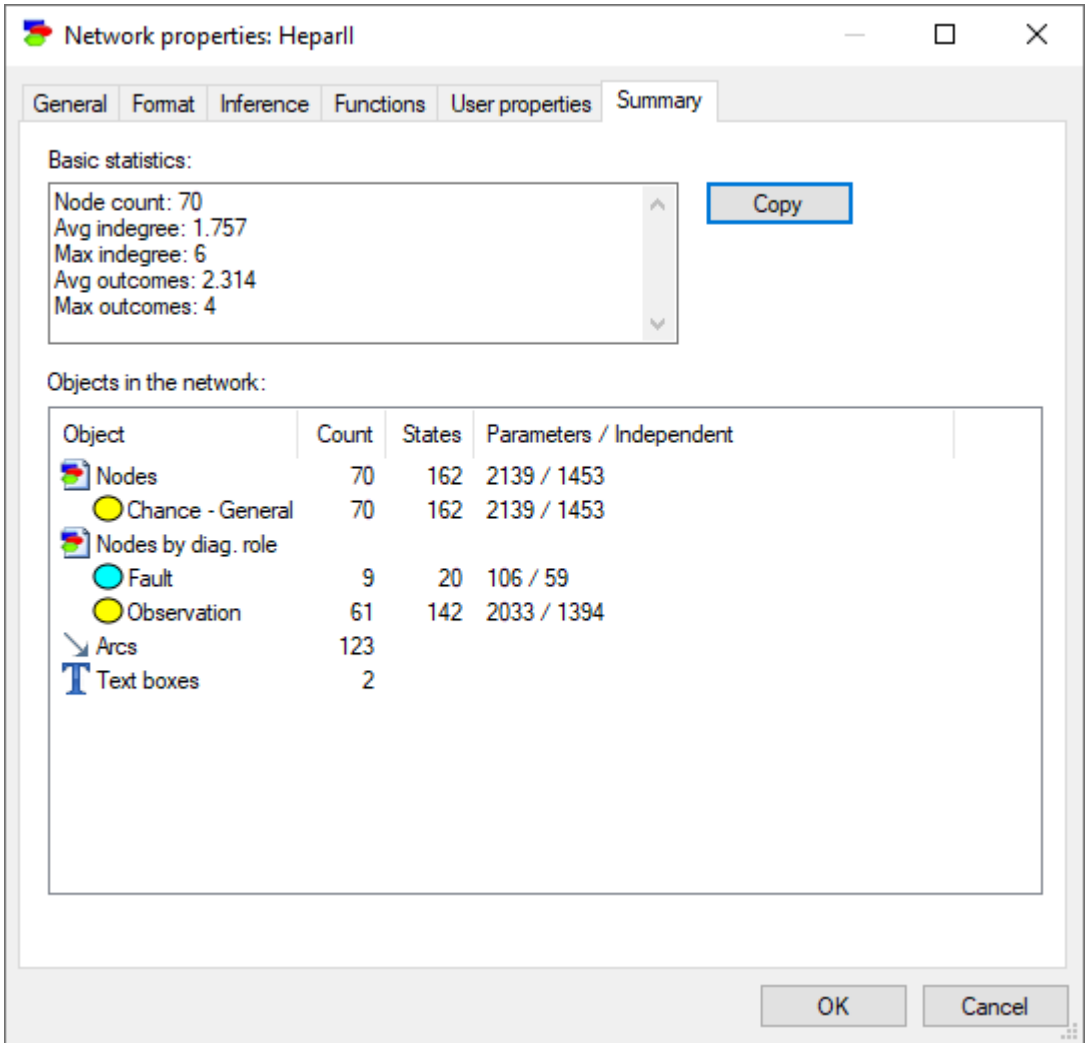
Button *Add* invokes the following dialog that allows for defining a new user property:



Buttons *Edit* and *Delete* allows for editing and removing a selected property, respectively.

Summary tab

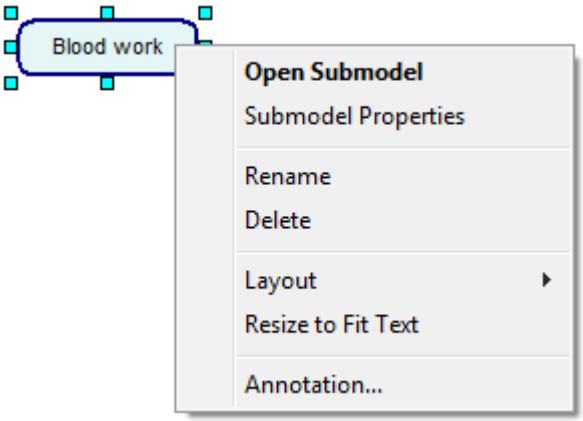
Summary tab contains summery statistics of the network, as illustrated below:



Statistics focus on the structural properties of the network, such as the number of nodes of each type in the network, the average and the maximum in-degree (the number of parents of a node), the average and the maximum number of outcomes of nodes, node counts by their diagnostic role, the number of arcs and the number of text boxes, and, finally, the number of states and parameters. In dependent parameters take into account that some parameters are just complements, making sure that probabilities have to add up to 1.0. Hepar II, shown in all illustrations in this section, contains 70 nodes, of which 9 are diagnostic fault nodes (diseases) and 61 are observation nodes. The number of arcs (123) and the average in-degree (1.757) give an idea of the structural complexity of the network. Elapsed time of the last inference call gives an idea of the difficulty that your computer system experienced with solving the model.

5.4.2 Submodel properties

The *Submodel properties* sheet can be displayed by right clicking on the name of the submodel in the *Tree View* or right clicking on the submodel icon in the *Graph View*. This will display the *Submodel Pop-up Menu*. Select *Submodel Properties* from the menu.

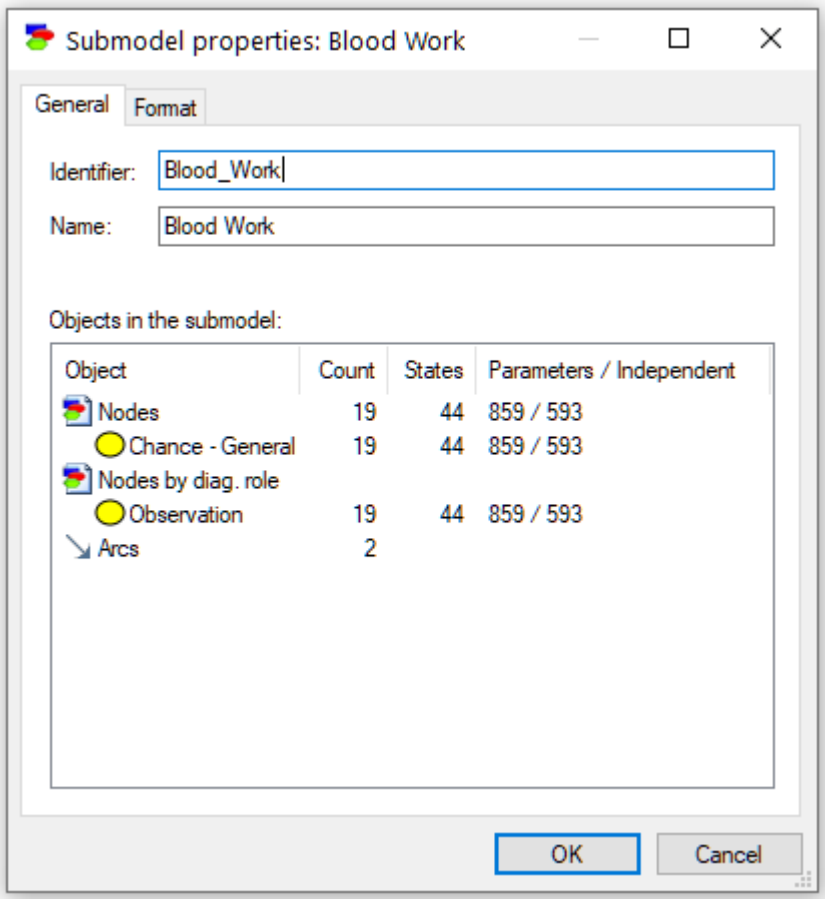


Note : *Double clicking on the submodel will open the graph view of the submodel, it will not open the Submodel properties sheet.*

The *Submodel properties* sheet consist of two tabs: *General* and *Format*.

General tab

The *General* tab displays the *Identifier* and the *Name* of the submodel, along with the submodel's basic statistics.



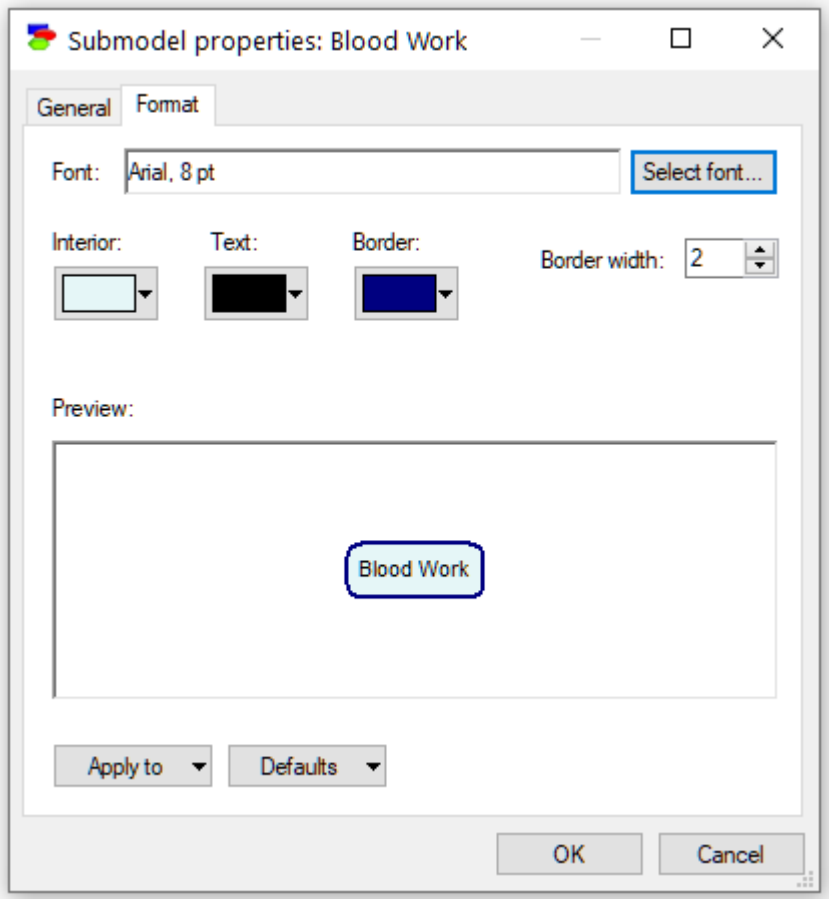
Identifier displays the identifier for the submodel, which is user-specified. Identifiers must start with a letter, and can contain letters, digits, and underscore (_) characters. The identifier for the network shown above is *Blood_work*.

Name displays the name for the submodel, which is user-specified. There are no limitations on the characters that can be part of the name. The name for the network shown above is *Blood work*.

The *Objects in the submodel* lists counts of various types of objects and numerical parameters in the submodel. They give an idea of the submodel's complexity.

Format tab

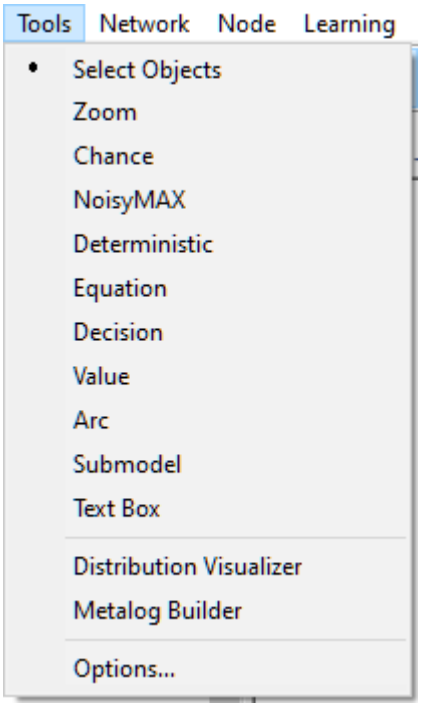
The *Format* tab allows to modify the visual properties of the submodel icon, i.e., how the submodel icon is displayed in the *Graph View*.



The *Format* tab is similar in function to the *Format* tab of the *Node properties* sheet.

5.4.3 Tools menu and Standard toolbar

Tools Menu is the main bag of tools for building models.



Most of these tools are replicated in the *Standard Toolbar*, which is a bar with buttons that offer quick mouse shortcuts for a number of menu commands.

It is also accessible in a floating form

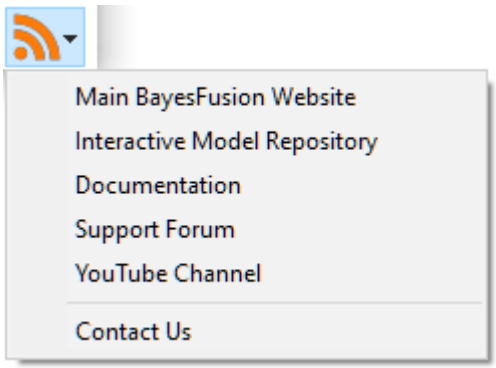
Standard Toolbar can be made invisible using the toggle command *Toolbar-Standard* on the *View Menu*. It can be also moved to any position within GeNIe application window. To move the toolbar from a locked position, click on the vertical bar at the left edge of the toolbar and drag it to its destination. Besides the standard buttons for opening, closing, and saving a file, this toolbar has buttons for selecting various tools found in the *Tools Menu*.

We review here *Standard Toolbar* tools that mimic the *Tools Menu* tools and allow for creating objects in the *Graph View* window. The drawing tool currently selected is marked by a dot on the left side of its name in the *Tools Menu*. Each of the *Tools Menu* tools can be also selected using a corresponding *Standard Toolbar* icon. While choosing a tool from the *Tools Menu*, selects the tool for a single editing action (with the exception of the *Select Objects* tool, which is the default tool), the tools on the *Standard Toolbar* work also in *sticky mode*. When a drawing tool on the *Standard Toolbar* is double-clicked, it remains selected until it is deselected (single-clicked upon) or another tool is explicitly selected. The tools *Chance* ( ), *Deterministic* ( ), *NoisyMAX* ( ), *Equation* ( ), *Decision* ( ) and *Value* ( ) draw a corresponding node type in the *Graph View*.

Submodel ( ) draws a submodel, *Text Box* ( ) allows for creating on-screen comments, and *Arc* ( ) allows for creating an arc between two nodes.

The two choices, *Distribution Visualizer* and *Metalog Builder* invoke dialogs that are invaluable in choosing continuous probability distributions for equation-based systems. They are described in detail in sections *Distribution visualizer* and *Metalog builder* respectively.

The BayesFusion internet resources button ( ) is a short-cut to the *Internet Resources* menu:



Clicking on any of the links leads to opening a corresponding web page in your default web browser:

Main BayesFusion Website:

The main page of BayesFusion's Internet web site.

Interactive Model Repository:

The site of BayesFusion's interactive model repository, powered by *BayesBox*.

Documentation:

A page with links to various documents, manuals, etc.

Support Forum:

BayesFusion's support forum, a place where you can ask questions and browse previous questions and our specialists' answers.

YouTube Channel:

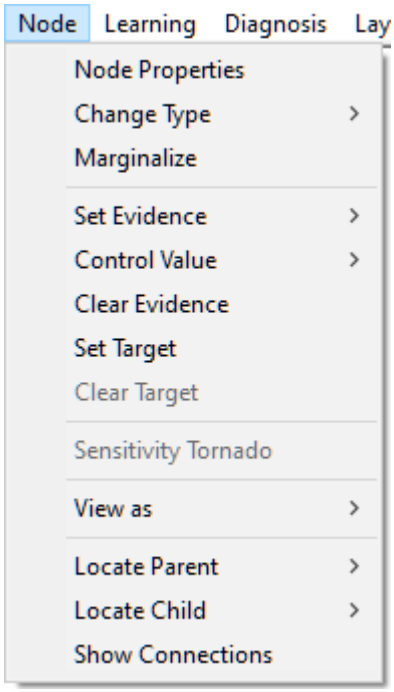
BayesFusion's YouTube channel with movies and instructional videos.

Contact Us:

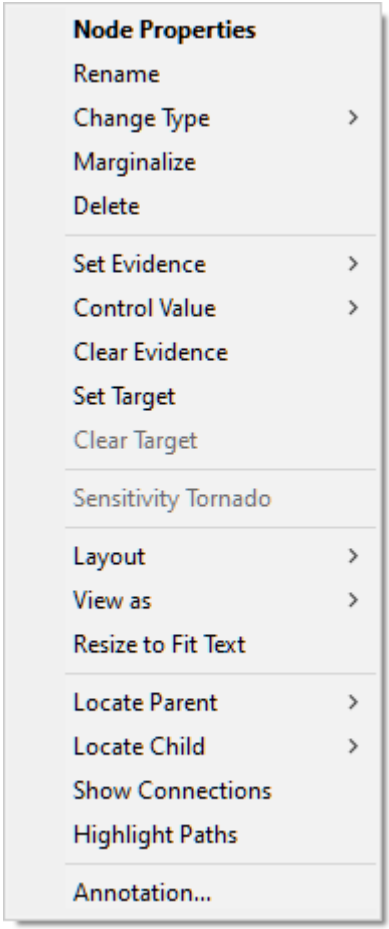
While we can be reached most reliably by EMail, this is a link to a web-based contact form that can be used to send us an Email message.

5.4.4 Node menu and node pop-up menu

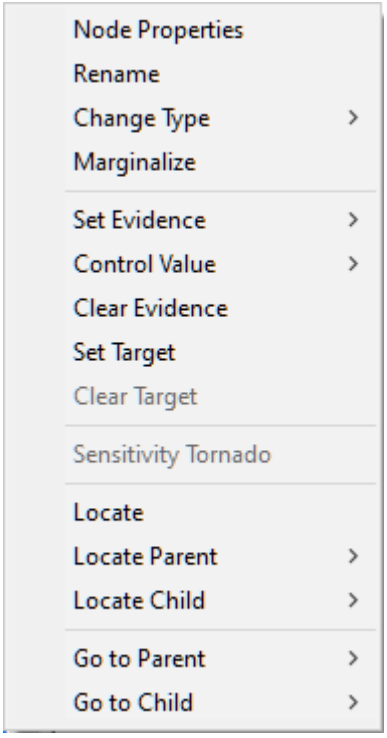
The *Node Menu* offers the following commands:



Node Pop-up menu, which is to a large degree duplicates the *Node Menu*, is different in the *Graph View* and the *Tree View*. The *Node Pop-up* menu in the *Graph View* can be displayed by right clicking on the node icon in the *Graph View*.



The *Node Pop-up* menu in the *Tree View* can be displayed by right clicking on the node name in the *Tree View*.

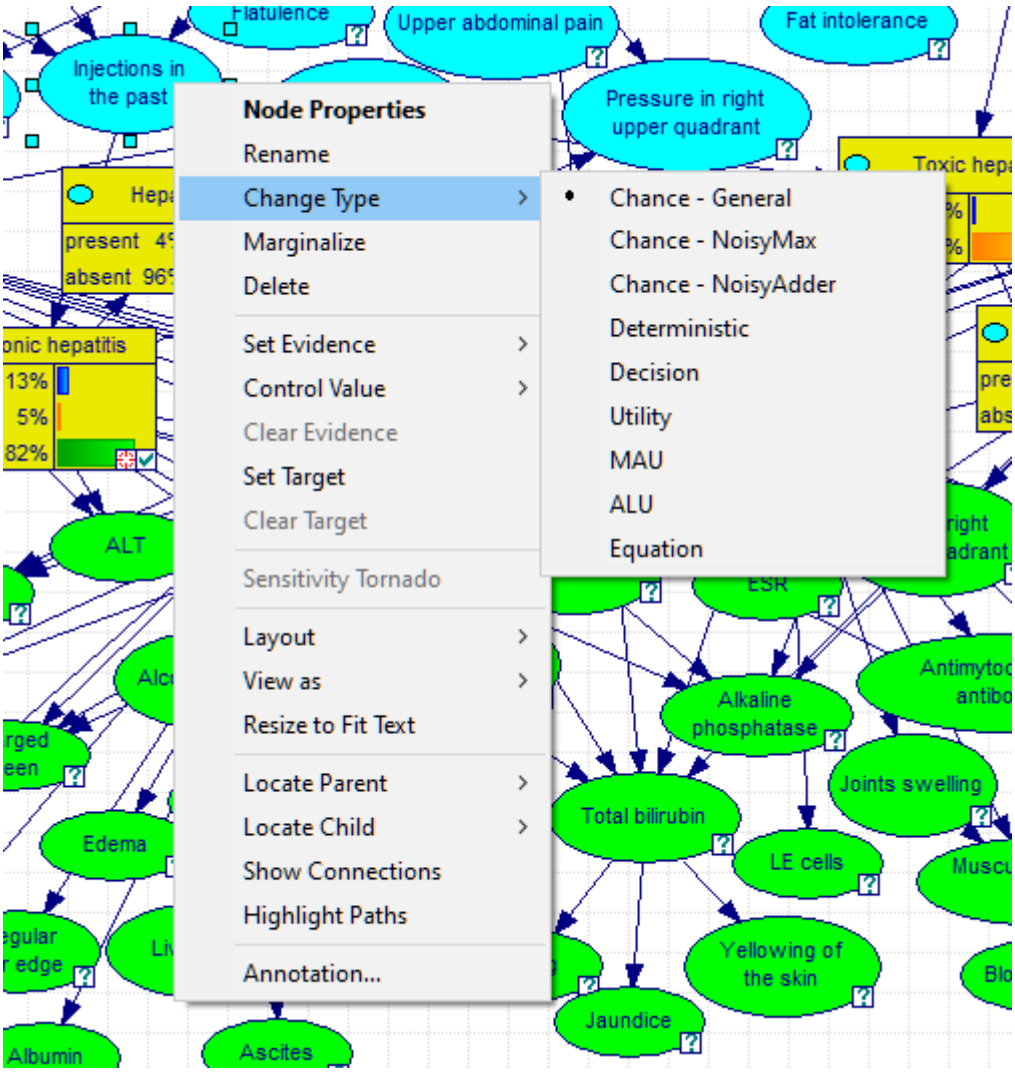


We will defer the command *Marginalize* to the *Useful structural transformations* section and commands *Set Evidence*, *Control Value* , *Clear Evidence*, *Set Target*, *Clear Target* and *Sensitivity Tornado* to the *Node menu* subsection in the *Inference algorithms* section. The remaining commands are described below.

Node Properties command in the *Node Menu* is active only if there is one (and only one) node selected in the *Graph View*. It invokes the *Node Properties* sheet for the selected node. In case of the *Node Pop-up* menu, more than one node may be selected but the properties are opened always of the node that the user right-clicked on.

Rename allows for editing the name of the node.

Change Type submenu allows for changing the type of the selected node.

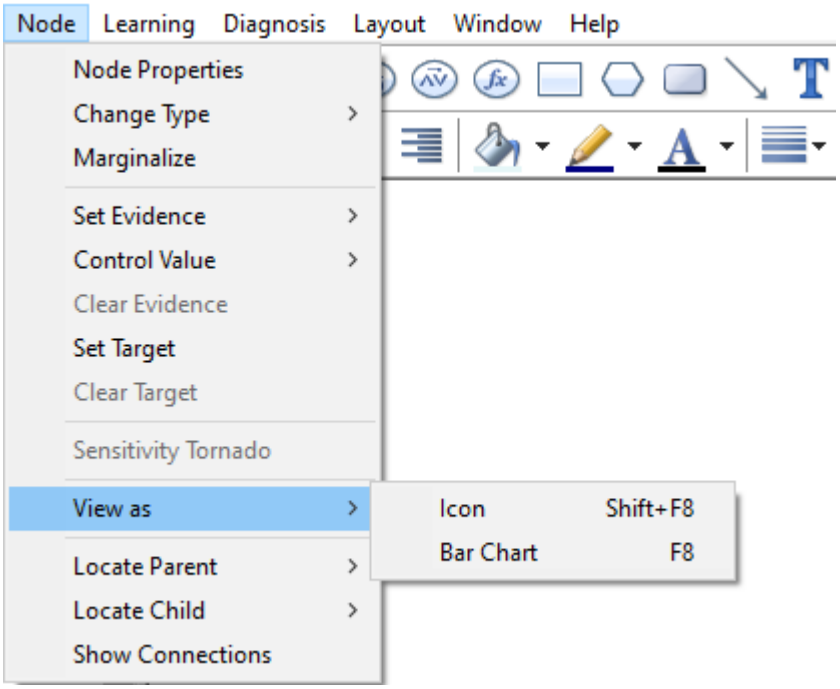


The current node can be changed to any of the listed types. Please note that one of the main effects is changing the node definition. This may lead to losing information. For example, changing the node type from *Chance* to *Decision* leads to losing the node definition, although it preserves the states and their names. State names will be lost as well when changing to node type to *Utility* or *Equation*, which are both continuous nodes.

Delete command, available only in the *Node Pop-up* menu in the *Graph View*, leads to deletion of the selected nodes.

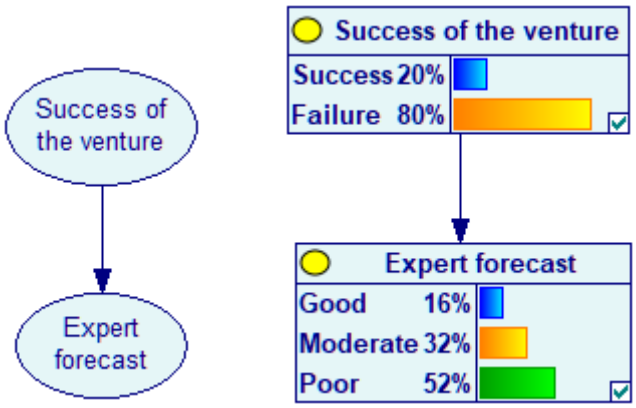
The *Layout* submenu duplicates all commands present in the *Layout Menu* and the *Format toolbar*.

The *View As* submenu controls how the nodes are displayed in the *Graph View*.



The *Icon* option displays the node as an elliptical or polygonal shape (depending on node type). The probability distribution over the node states can be viewed, for example, by updating the model and placing the cursor over the *Updated* (☑) status icon.

The *Bar Chart* option displays the node as a bar chart of the marginal probabilities of each state of the node. This view is useful when we want to see the marginal probabilities at a glance.



The *View As* command applies to the currently selected nodes. If no nodes are selected, then the command applies to all nodes.

Resize to Fit Text command, available only in the *Graph View*, changes the size of the node in the *Graph View* so that it fits its name (or identifier).

The following three commands are useful in navigating very large models that do not fit on the screen.

Locate command, available only in the *Tree View*, finds the selected node in the *Graph View* and flashes three times.

Locate Parent and *Locate Child* submenus help in navigating through the model. They display the list of all parents (or children) of the node. When one of them is selected, it is located it in the *Graph View* and is flashed three times.

The *Annotation...* command (*Node Pop-up* menu only) invokes the *Annotation* dialog, which allows for adding an annotation to the node. See *Annotations* section for more information.

The *Go to* commands, available only in the Tree View, aid navigation within the tree view. The *Go to Parent* command can be used to select a parent of the selected node. If there is more than one parent, then it will display a list of all parents of the selected node. Clicking on a parent selects it in the *Tree View*. The *Go to Child* command is analogous and can be used to select a child of the selected node.

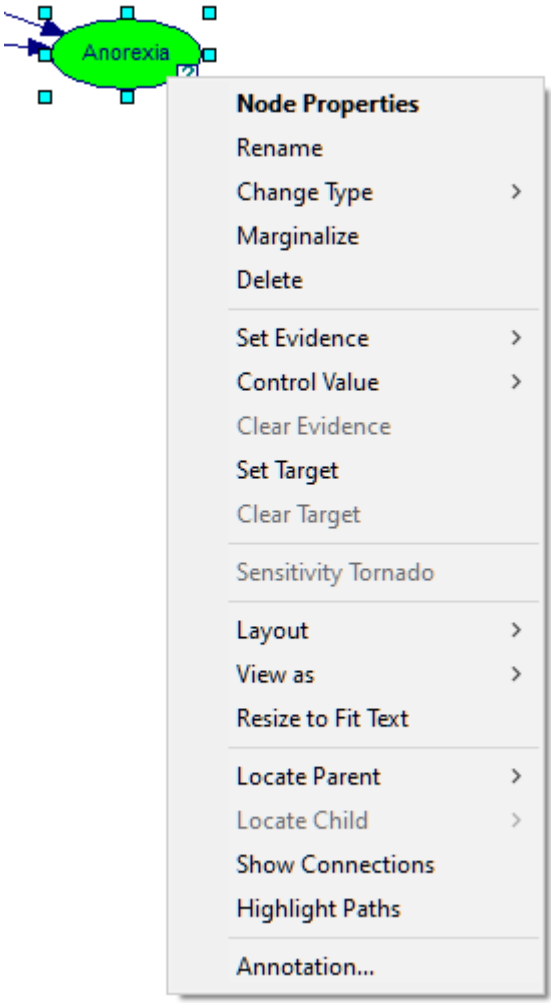
Finally, *Show Connections* and *Highlight Paths* are described in detail in section *Structural Analysis*.

5.4.5 Node properties

Node properties sheets allow for modifying properties of model nodes. They can be opened in the following two ways:

1. Double-click on a node in the *Graph View*.

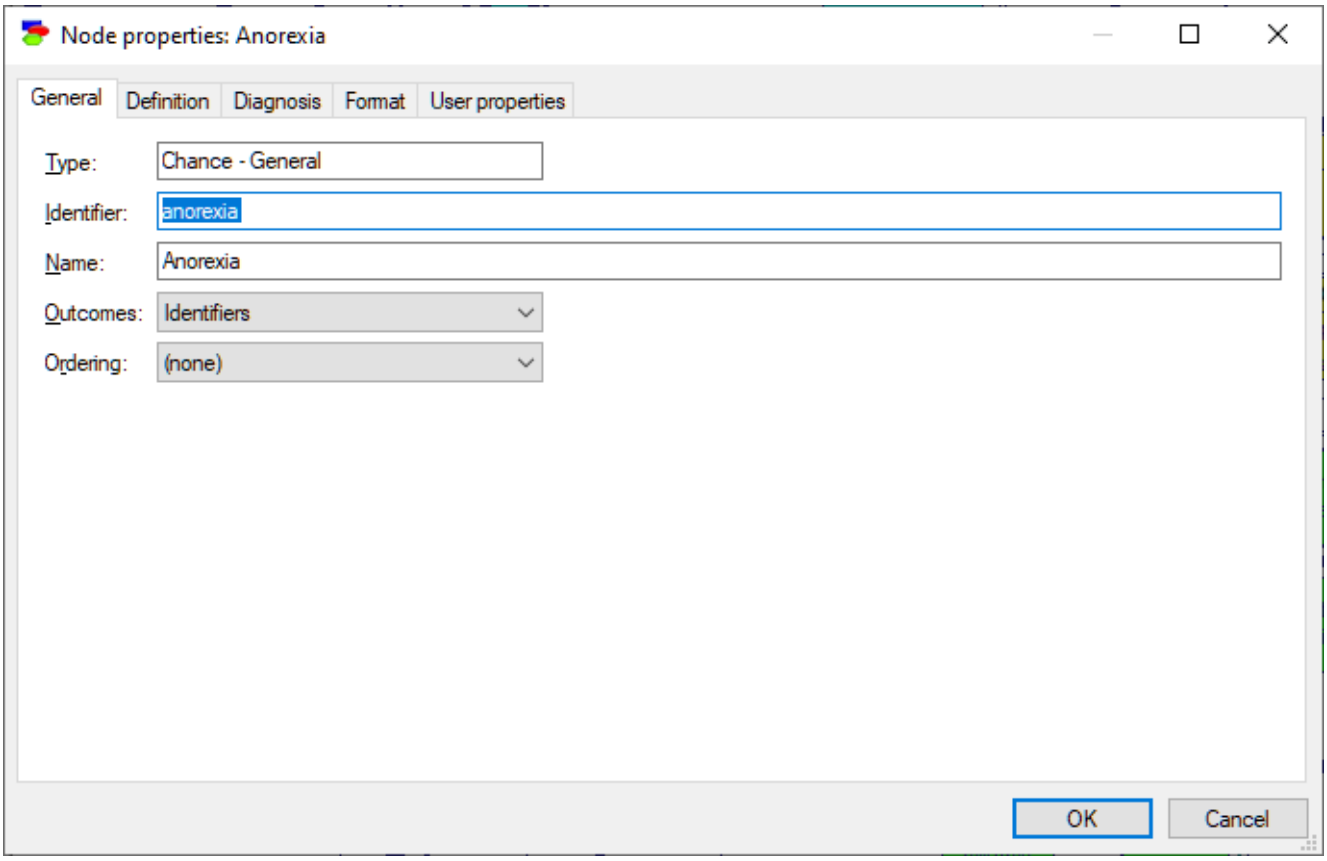
2. Right click on the name of the node in the *Tree View* or right click on the icon of the node in the *Graph View*. This will display the *Node Pop-up* menu. Select *Node Properties* from the menu.



The *Node properties* consist of several tabs. While we will discuss all of the tabs in this section, not all of the tabs appear among the *Node properties* at the same time. The *Value* tab appears only when the value of the node is available. Some tabs appear only when the diagnostic features of GeNIe are enables.

General tab

General tab is the first tab of the *Node properties*. Shown below is a snapshot of the *General* tab when the diagnostic options are disabled:



The *General* tab contains the following properties:

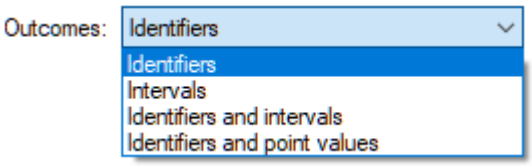
Node *Type* (in the picture, it is *Chance - General*) cannot be changed in the *General* tab and serves only informational purpose. Node type is selected during node creation (by selecting the appropriate icon from the *Standard Toolbar*). It can be changed after creation by either right clicking on the node in the *Graph View* and selecting *Change Type...* from the *Node Pop-up* menu or selecting *Change Type* from the *Node Menu* in the *Menu Bar*. This will display a sub-menu from which you can choose the new type for the node. See *Node Menu* section for more information.

Identifier displays the identifier for the node, which is user-specified. Identifiers must start with a letter, and can contain letters, digits, and underscore (_) characters. Letters are a-z and A-Z but also all Unicode characters above codepoint 127, which allows using characters from other alphabets than the Latin alphabet. The identifier for the network shown above is *anorexia*.

Name displays the name for the network, which is user-specified. There are no limitations on the characters that can be part of the name. The name for the network shown above is *Anorexia*.

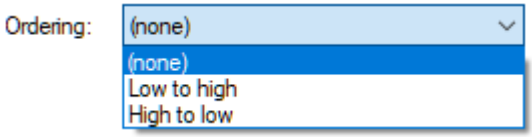
The reason for having both, the *Identifier* and the *Name* is that nodes may be referred to in equations. In that case, the reference should avoid problems with parsing, which could easily appear with spaces or special characters. On the other hand, *Identifiers*, which are meant to refer to nodes, may be too cryptic when working with a model, so we advise that *Names* be used for purposes such as displaying nodes.

Outcomes allows for specifying the type of outcomes that the node will have. There are four basic types of outcomes



Identifiers are those nodes that consist of categorical states, *Intervals* and *Point values* (possibly with *Identifiers*) allow for specifying discrete numerical nodes. We will discuss each of these sub-types in the *Definition tab* section below.

Ordering gives the modeler the opportunity to indicate whether the outcomes of the node are sorted from the highest to the lower or lowest to the highest value. This becomes important in some types of nodes, for example in *canonical models*.

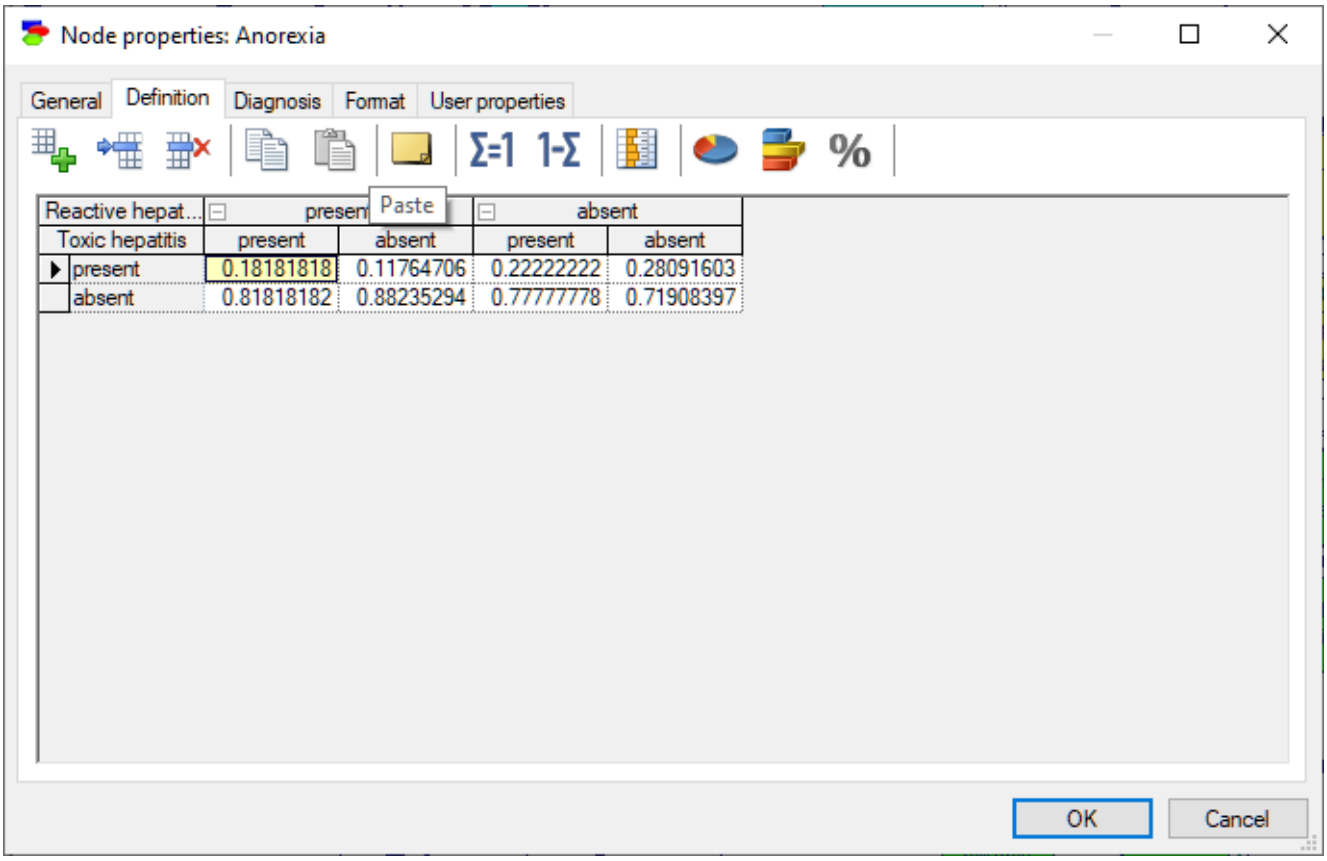


Definition tab

Definition tab allows for specifying node definition, i.e., how the node interacts with other nodes in the model. While there are common elements among various definition tabs, there are as many definition tabs as there are combinations of node kinds, domains, and probability types.

Chance nodes: Identifiers

Chance nodes specified as *Identifiers* allow for modeling discrete random variables that have categorical outcomes described by labels. Their definition consists of a set of conditional probability distributions, one for each combination of the parents' outcomes, and collected in a table names conditional probability table (CPT). (In case of Noisy-MAX nodes, the definition is slightly different but it is equivalent to a CPT.)



The rows of the table correspond to states of the random variable modeled by the node (in this case, *Anorexia*). The state names can be changed by clicking on them. The top rows, with gray background correspond each to a parent of *Anorexia*: *Reactive hepatitis* and *Toxic hepatitis*. Because each of the variables involved in this interaction is binary and there are two parents, we have 2x2=4 columns in the CPT. Each column corresponds to one combination of outcomes of the parents. For example, the first column from the left corresponds to *Reactive hepatitis* being *present* and *Toxic hepatitis* being *present*. The order of parents can be changed by dragging and dropping them in their new position, which may prove useful in probability elicitation, as some orders may turn out to be counter-intuitive. The order of states in the table can be changed by dragging and dropping as well. Individual probabilities can be edited by clicking on them. There are several convenient tools that help with filling the tables with probabilities:

Add ( ), *Insert* ( ), and *Delete* ( ) buttons are useful in adding and removing states. They add a new state after the selected state, add a new state before the selected state, and delete the selected state, respectively.

Copy ( ) and *Paste* ( ) buttons allow for copying and pasting fragments of the definition spreadsheet. Please note that GeNIe allows for copying and pasting to and from other programs on your computer, for example Microsoft Excel.

Tip: *You can select the entire spreadsheet quickly by clicking on the Node name, or an entire column by clicking on the state name. You can also select the spreadsheet by having the cursor in any of the cells and*

pressing CTRL-A. You can select all state names quickly by having the cursor in any of the names and pressing CTRL-A.

The *Quickbar* ( ) button turns on graphical display of the probabilities in the background. It is useful for visualizing the order of magnitude of numerical probabilities.

The *Annotation* ( ) button (shortcut *CTRL+T*) allows for adding annotations to state names and to individual probabilities. Please use it generously, as models are best viewed as documents of your decision problem.

The *Normalize* ( ) and *Complement* ( ) buttons are useful when entering probability distributions.

The *Normalize* button (shortcut *CTRL+N*) normalizes the contents of the selected column by dividing each number by the sum of all numbers in the column. The effect of this operation is that the sum of all numbers in the column becomes precisely 1.0, something that is expected of a probability distribution. This button makes it convenient to enter probabilities as percentages (e.g., 10, 30, and 70), then selecting the column and pressing the *Normalize* button, which changes the numbers to (0.1, 0.3, and 0.7) so that they are a correct probability distribution.

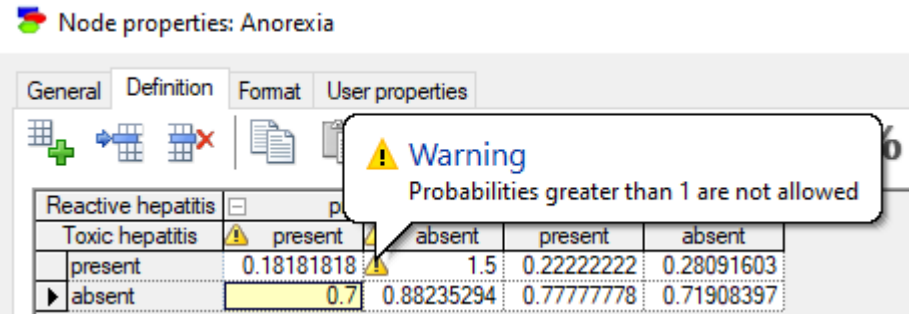
The *Complement* button (shortcut *CTRL+O*) can be pressed after selecting one or more cells (they do not have to be cells in the same column). When pressed, it fills in the selected cell the complement of the remaining cells in the column, i.e., a number that will make the sum equal precisely to 1.0. For example, in the simplest case, when two of the three cells contain 0.1 and 0.3 and the *Complement* button is pressed with the third cell selected, the cell will receive 0.6, which is 1.0-(0.1+0.3). When multiple cells within a column are selected, GeNIe will distribute the complement among the selected cells using the existing values as weights when distributing. Here is an example: Let the probabilities in a column be (0.4, 0.3, 0.4, 0.2). If the last two cells (with 0.4 and 0.2) are selected when the *Complement* button is pressed, the probabilities will change to (0.4, 0.3, 0.2, 0.1). The complement probability (1-0.4-0.3=0.3) gets distributed between the selected cells in the proportion 0.4:0.2 or 2:1, yielding 0.2 and 0.1. When cells in multiple columns are selected, the complement operation will be performed in each of the columns with selected cells in separation. Pressing the *Complement* button will lead to an error message if the sum of probabilities of the other fields in the column exceeds 1.0. The buttons saves typing and ensures that the probability distribution is correct.

It is possible to use the cell-oriented probability distribution interface as a simple calculator by typing in a cell a mathematical expressions preceded by the = character. Such expressions are evaluated immediately and the result of the evaluation is placed in the cell. For example, typing in the cell =1/3 will evaluate to a floating point number 0.33333333. It also works with functions and probability distributions (see the *Writing equations in GeNIe* section for a list of functions and distributions known by GeNIe). For example, typing =Uniform(0,1) will result in a single random number from the Uniform(0,1) distribution, e.g., 0.39330341.

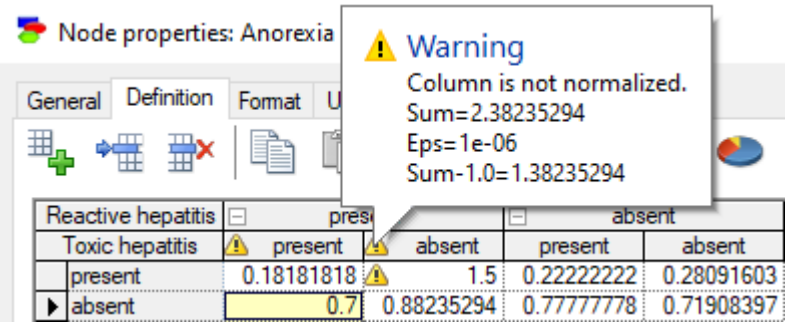
GeNIe warns the user of problems in the probability distribution tables whenever these contain incorrect distributions. Any number outside of the range [0..1] causes GeNIe to raise a flag in the cell. Also, when the sum of probabilities in a column is not 1.0, GeNIe places a flag on the column.

Reactive hepatitis	present		absent	
Toxic hepatitis	present	absent	present	absent
present	0.18181818	1.5	0.22222222	0.28091603
absent	0.7	0.88235294	0.77777778	0.71908397

Hovering the mouse over the flag displays a warning message with the reason for the flag. Probabilities greater than 1.0 result in the following warning:



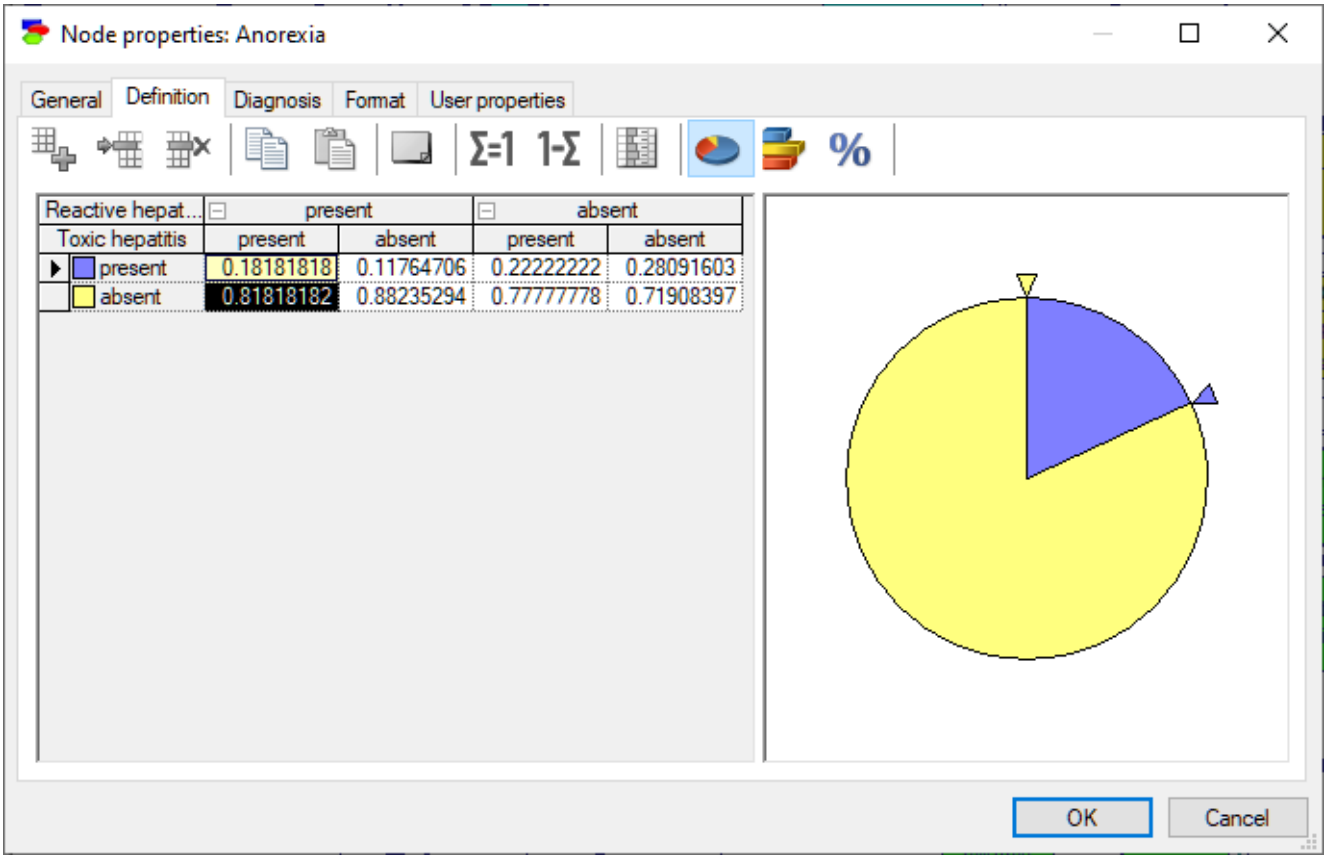
Sum of probabilities not equal to 1.0 results in the following warning:



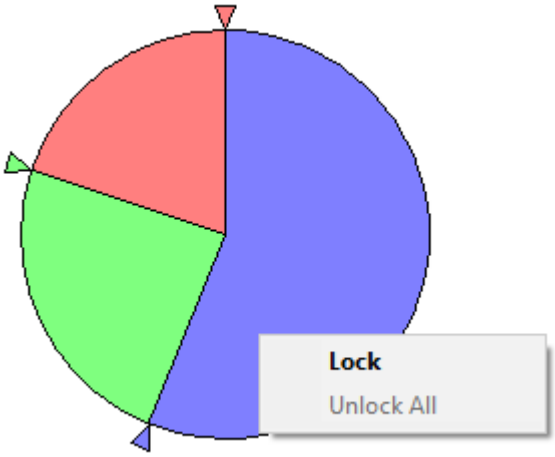
The warning displays the sum of probabilities, the tolerance threshold value (in the above example Eps=1e-06=0.000001), and the difference between the sum and the theoretically enforced sum of 1.0. The tolerance threshold is adaptive and is never larger than the smallest probability value in the CPT.

GeNIe will not allow to exit the *Node properties* dialog if the definition is incorrect.

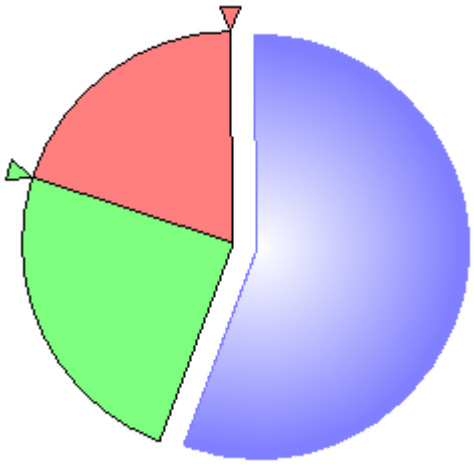
Finally, it is possible to enter probability distributions graphically, either through a probability wheel or a bar chart. Pressing on the *Elicitation piechart* ( ) button invokes a probability wheel dialog:



You can modify the probability distribution graphically using the small triangles. Drag a triangle around the circumference of the pie to increase or decrease the probability distribution for that state. As we adjust the size of one section of the piechart, the remaining section change proportionally. In case of variables with multiple states, it is sometimes useful to lock a selected part of the piechart to prevent it from changing. We can lock a part of the pie chart by right-clicking on the part and selecting *Lock* (or simply double-clicking on that part).

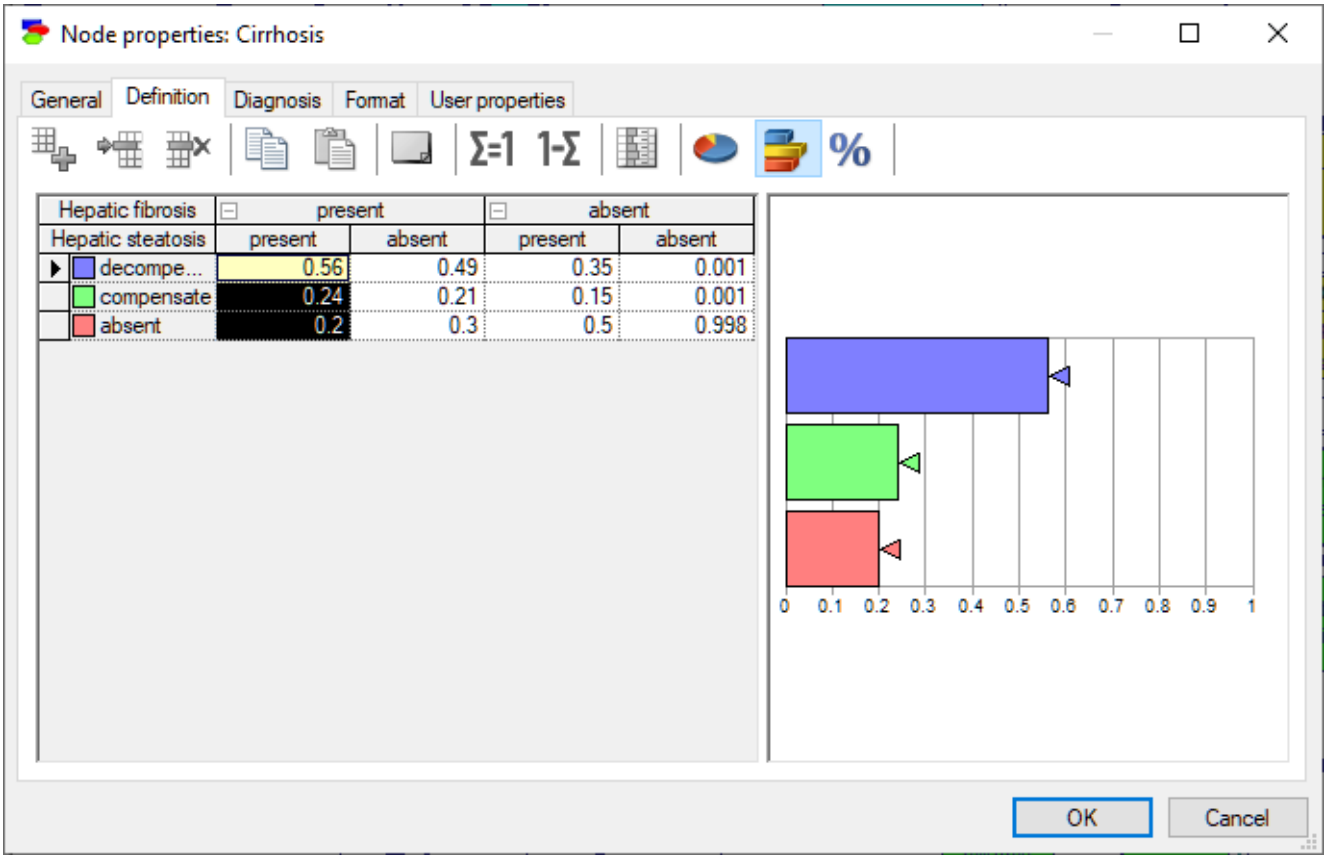


The effect of this is that the locked part shows as separate from the rest of the pie and no longer takes part in elicitation, remaining constant until released.



It is possible to lock multiple parts.

Elicitation barchart ( ) button invokes a similar graphical elicitation dialog but based on a bar chart:

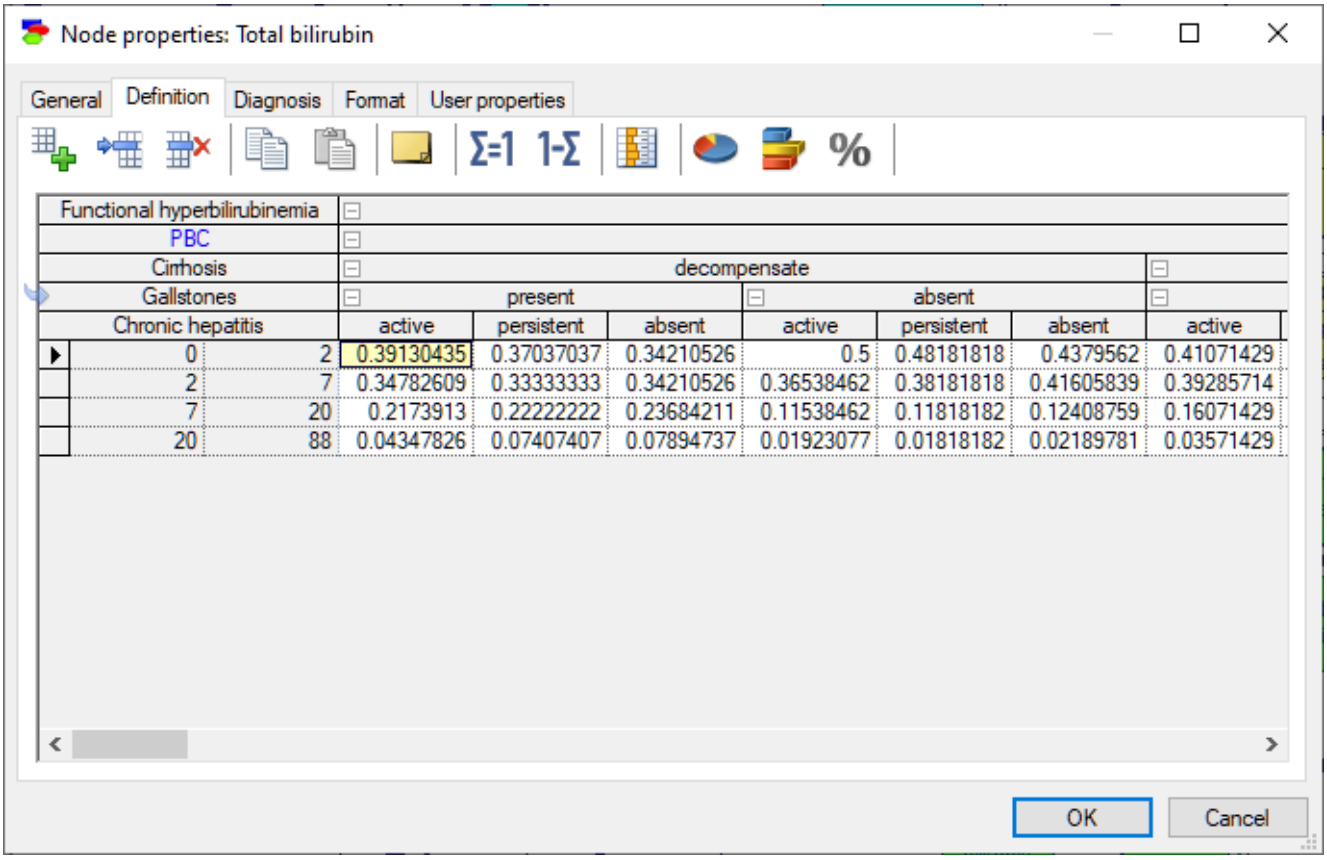


Here also you can change the size of the individual bars by dragging the small triangles on their right-hand side. You can lock those bars that we are happy with to prevent then from further changing.

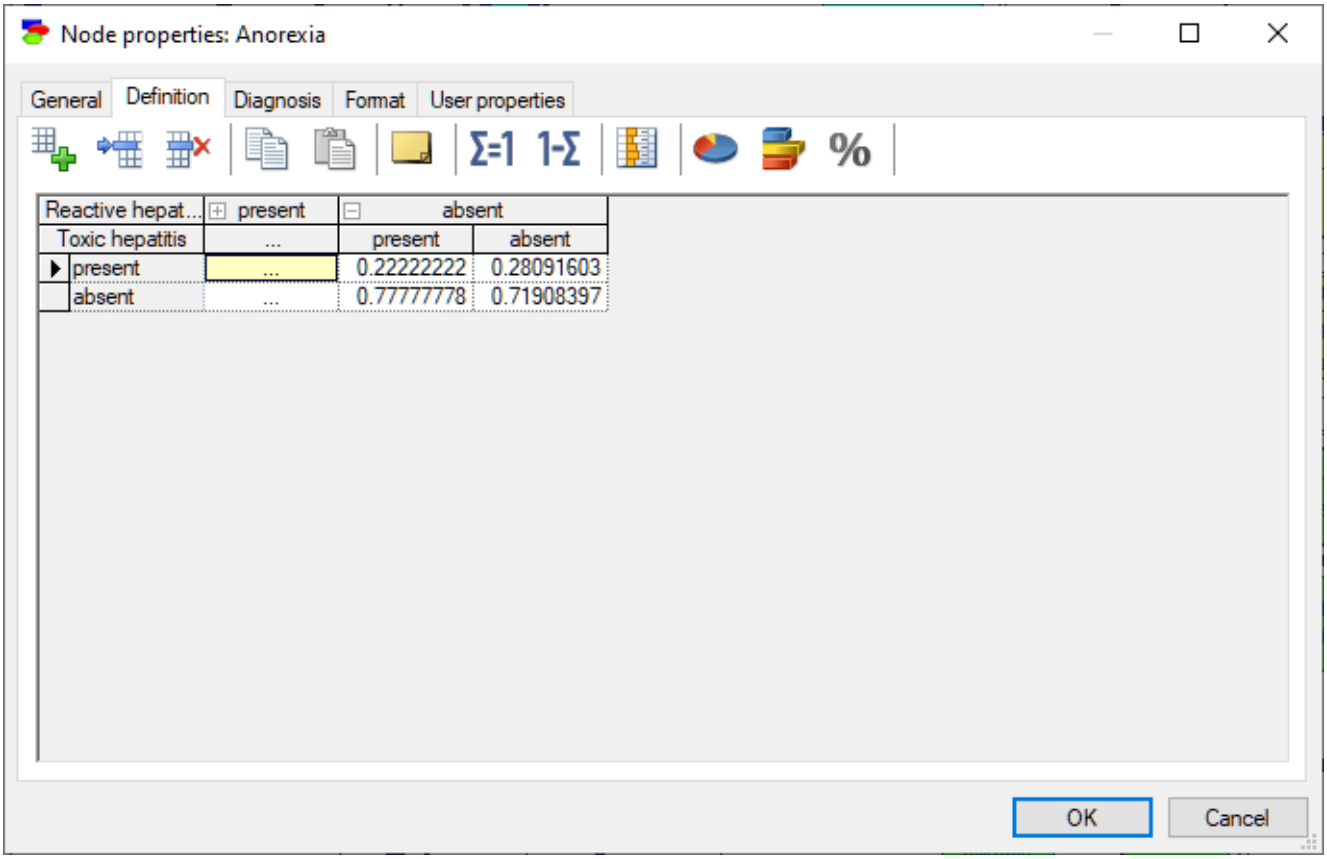
The *Show percentages in the elicitation chart* (%) button turns on a tool tip that shows the numerical value of the modified probability. There has been empirical research showing that viewing the probabilities during graphical elicitation may not be the best idea in terms of leading to a decreased accuracy of the elicitation.

In case of both, the piechart and barchart, it is possible to elicit multiple columns at the same time. Just select the columns that you desire before entering the elicitation dialog. The resulting probability distributions will be entered in all the selected columns. The elicitation piechart and barchart are equivalent formally but are convenient for different purposes at the user interface. The barchart is better at showing the absolute value of probability while the piechart may be better at relative comparisons.

Sometimes, the order of parents or the order of states in a node may be not intuitive. GeNIe allows for an interactive change of the order of parents and states. Simply click on the parent name or the state name and drag it to its destination position.

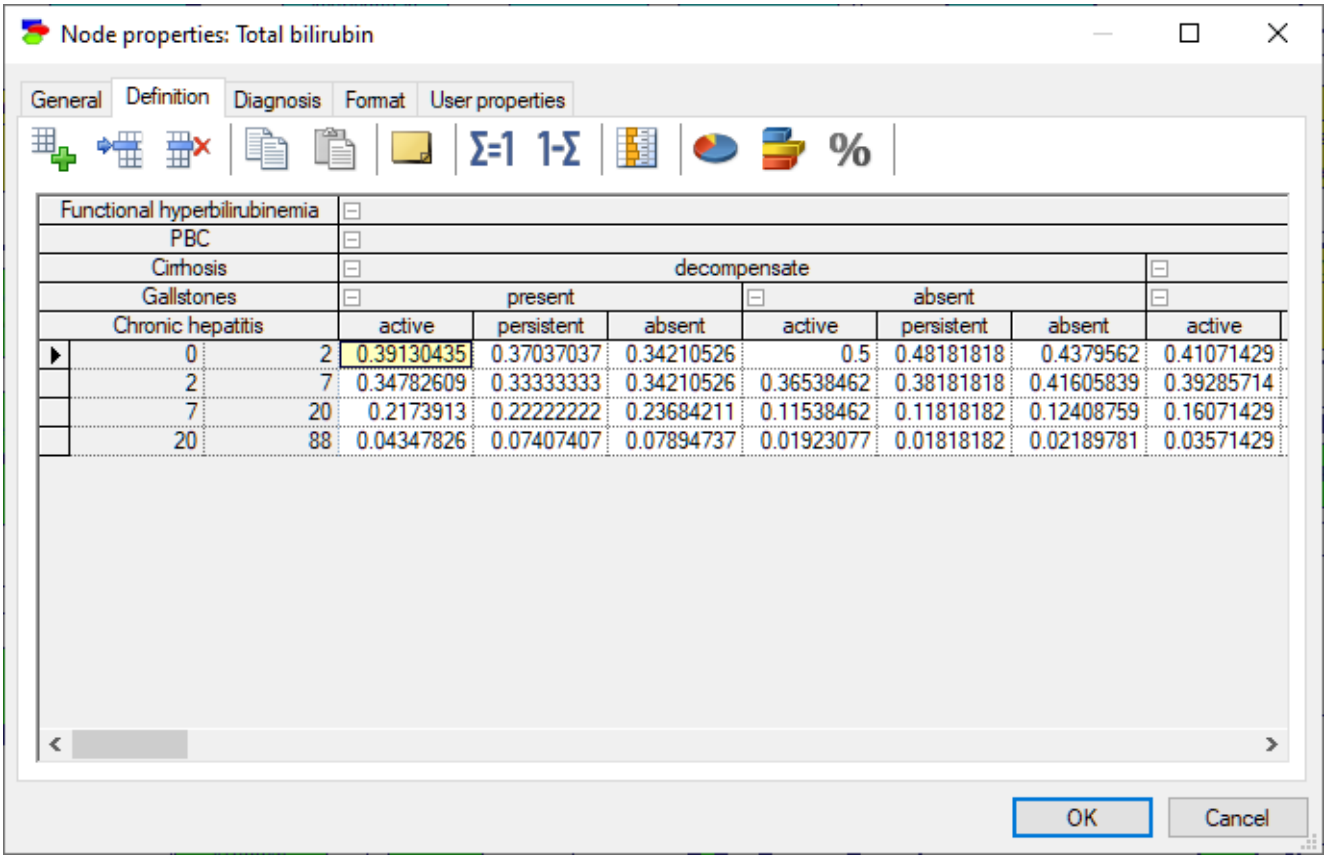


When a CPT is very large, it is useful to shrink parts of it. To that effect, please use the small buttons in the header lines (⊞ and ⊟).

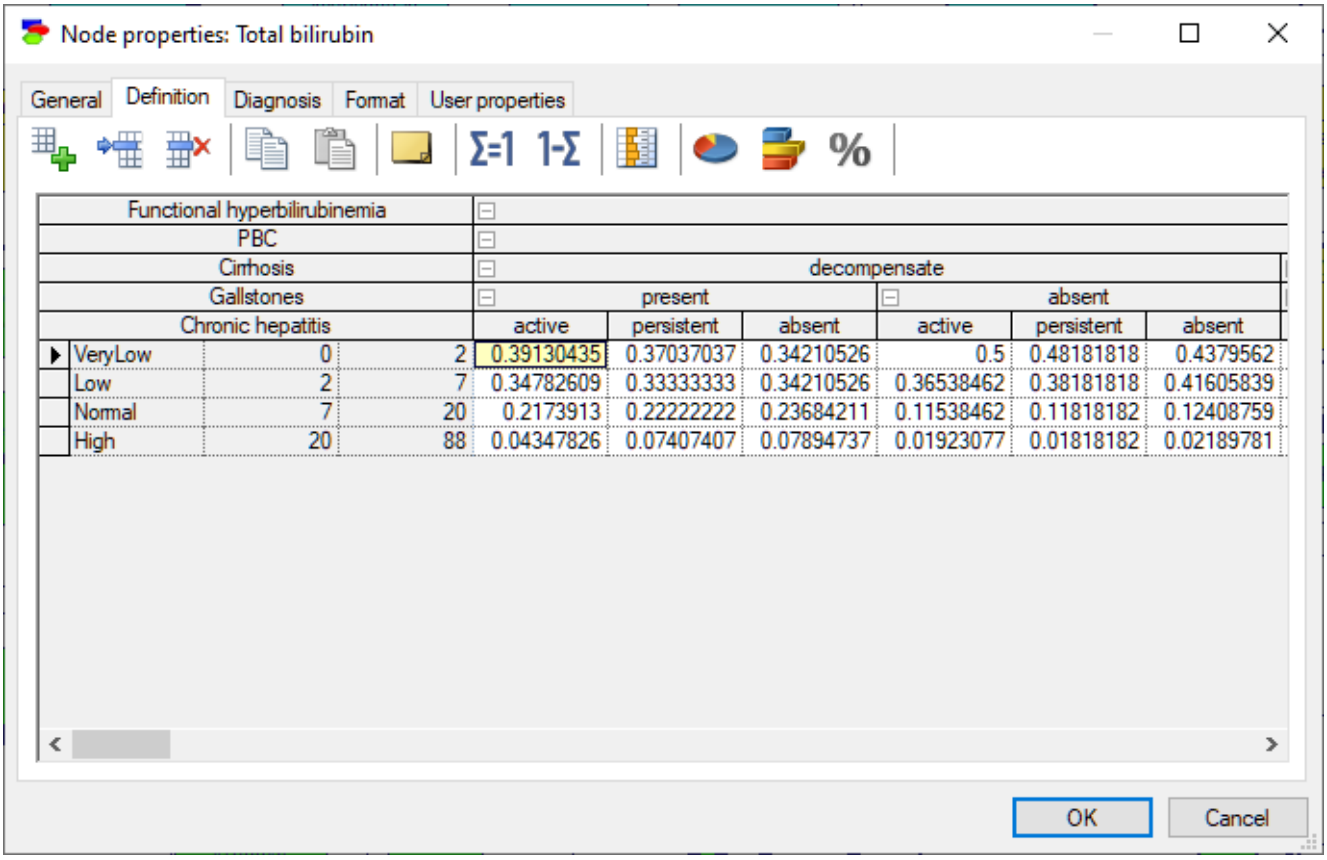


Chance nodes: Intervals

Chance nodes specified as *Intervals* or *Identifiers with intervals* allow for modeling discrete random variables that are numerical in nature. The domain of such variables is divided into adjacent intervals, with the borders specified explicitly, as in the variable below (the Definition tab is for a *Chance-General* node; definition tabs for other discrete chance nodes are similar):



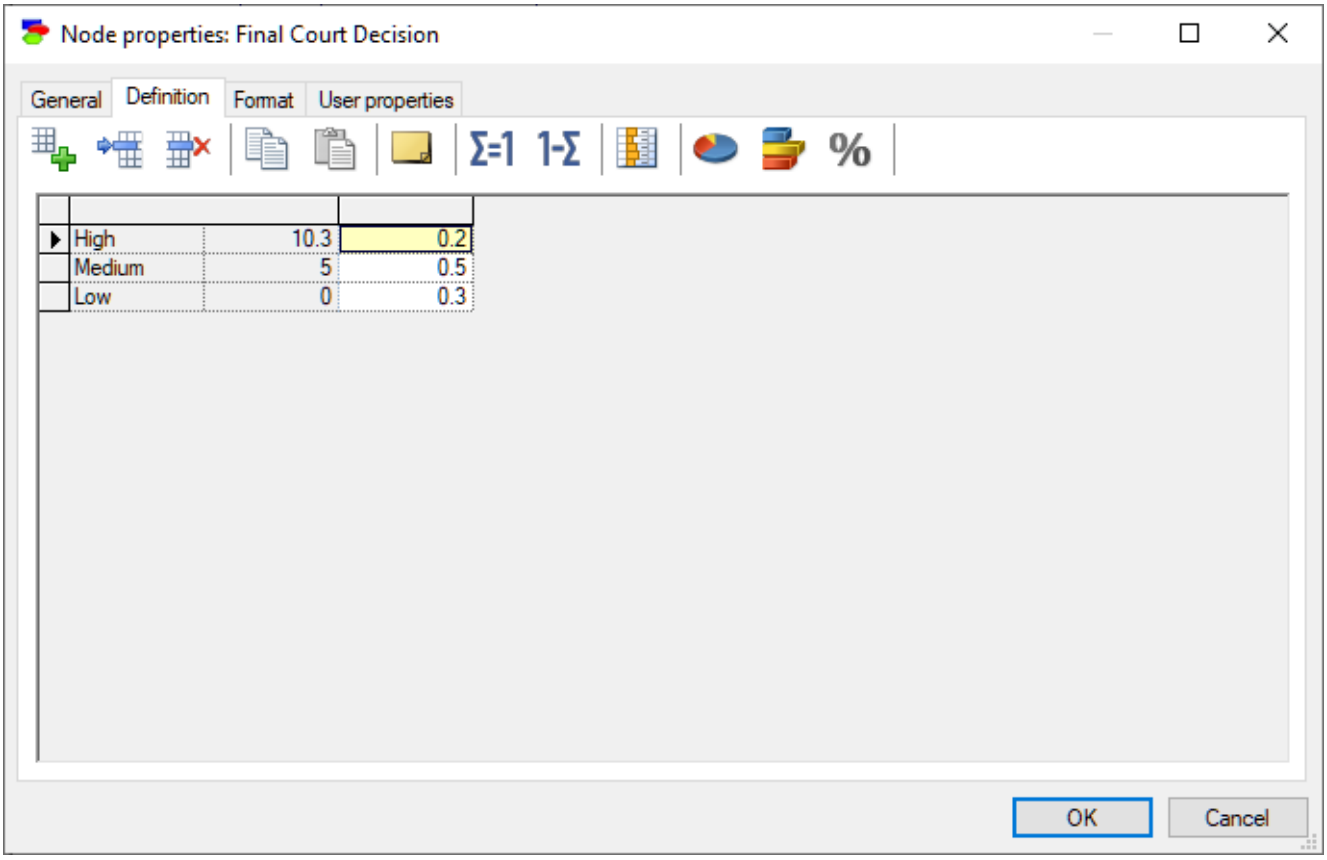
Nodes specified as *Identifiers with intervals* allow for naming each interval, in addition to specifying their boundaries.



The probability distribution over such variables is discrete and each interval is treated as a discrete state. For the purpose of numerical calculations, such as the calculation of the mean or standard deviation, the possible (numerical) values of the variable are assumed to be distributed uniformly within each of the intervals, except for open intervals, i.e., intervals with left boundary of `-inf` or right boundary of `inf` (empty cells act as `-inf` and `inf`, respectively). The definition of a variable that is specified as *Intervals* or *Identifiers with intervals* consists also of a set of conditional probability distributions, one for each combination of the parents' outcomes, and collected in a table names conditional probability table (CPT). All operations of the CPT are the same as in case of variables with *Identifiers*.

Chance nodes: Point values

Chance nodes specified as *Identifiers and point values* allow for modeling discrete random variables that are numerical in nature. The domain of such variables is a set of discrete numerical values, as in the variable below:



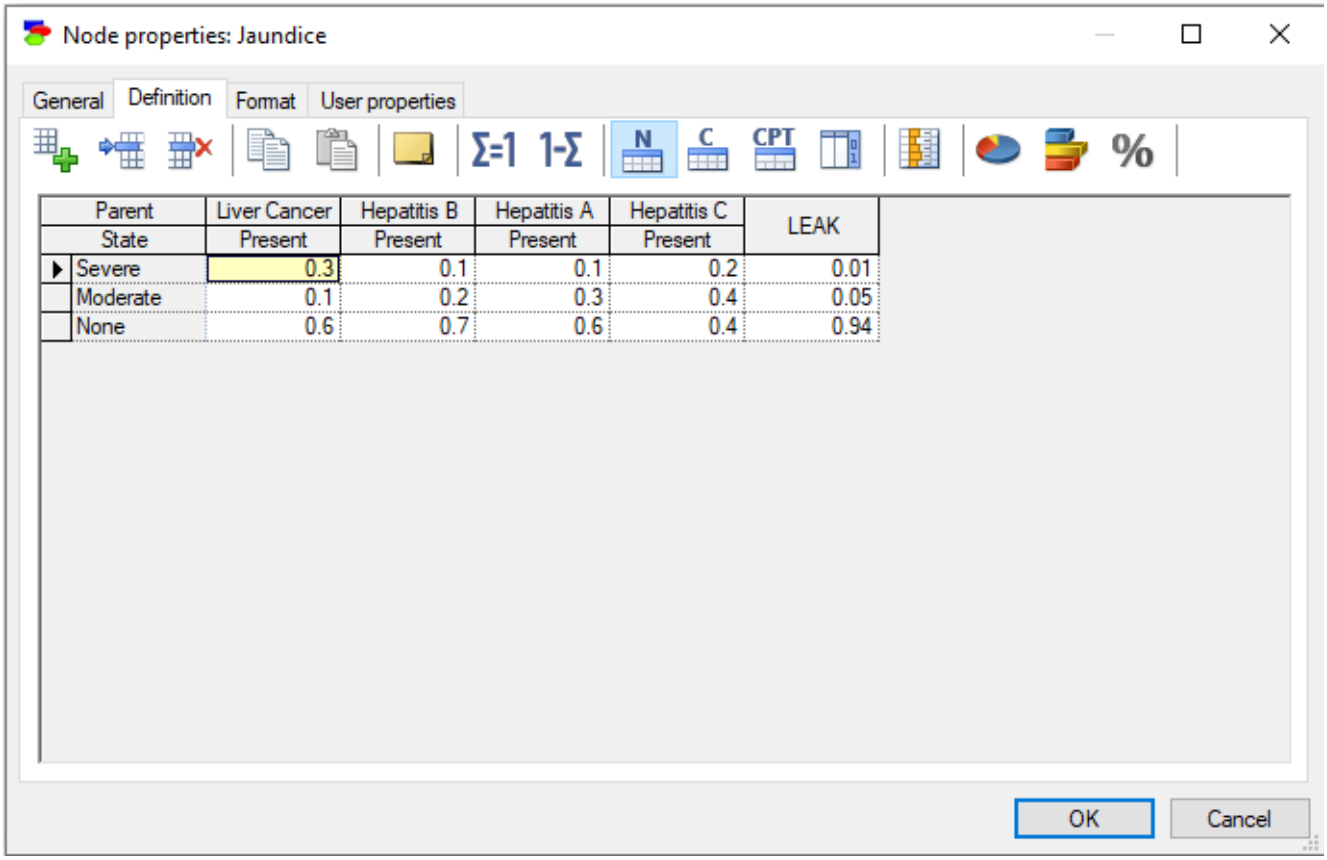
Nodes specified as *Identifiers and point values* allow for naming each numerical point value, in addition to specifying the value itself.

The probability distribution over such variables is discrete and each (labeled) numerical value is treated as a discrete state. For the purpose of numerical calculations, such as the calculation of the mean or standard deviation, the possible (numerical) values of the variable are assumed to be distributed uniformly within each of the intervals, except for open intervals, i.e., intervals with left boundary of `-inf` or right boundary of `inf` (empty cells act as `-inf` and `inf`, respectively). The definition of a variable that is specified as *Intervals* or *Identifiers with intervals* consists also of a set of conditional probability distributions, one for each combination of the parents' outcomes, and collected in a table names conditional probability table (CPT). All operations of the CPT are the same as in case of variables with *Identifiers*.

Chance-NoisyMax nodes

Chance-NoisyMax are a special case of chance nodes modeling discrete random variables using the Noisy-Max assumption (please see the section *Noisy-MAX model*). *Chance-Noisy-MAX* nodes, similarly to *Chance-General* nodes, can have domains specified by *Identifiers*, *Intervals*, *Identifiers and intervals*, and *Identifiers and point*

values. Their definition consists of a set of conditional probability distributions, one for each state of each parents' outcomes. The *NoisyMax* nodes allow for specification of the interaction with their parents in a simplified way, requiring fewer parameters. The specifications can be viewed as a conditional probability table (CPT).



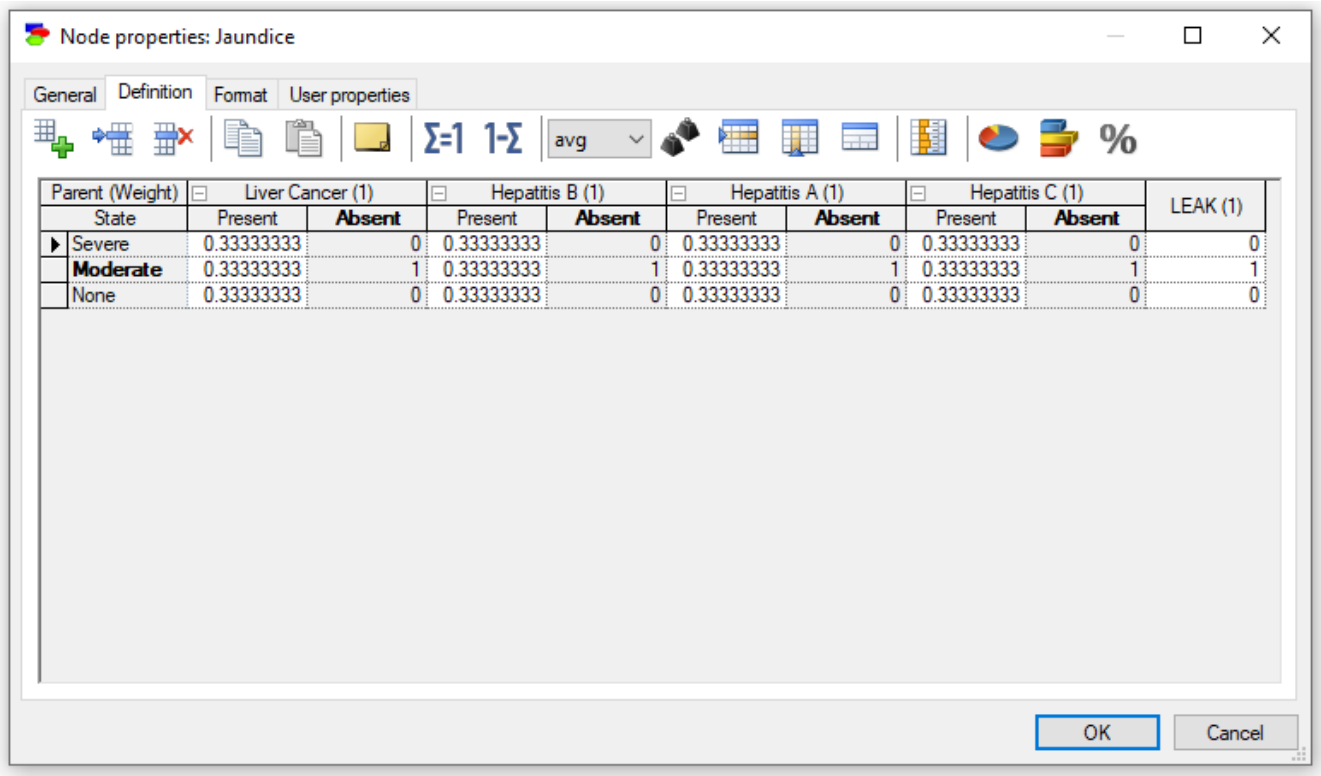
Parent	Liver Cancer	Hepatitis B	Hepatitis A	Hepatitis C	LEAK
State	Present	Present	Present	Present	
Severe	0.3	0.1	0.1	0.2	0.01
Moderate	0.1	0.2	0.3	0.4	0.05
None	0.6	0.7	0.6	0.4	0.94

Most of this dialog resembles the dialog for the *Chance-General* nodes. The additional functionality, which we discuss below, allows for *Noisy-MAX*-specific activities. The buttons *Show net parameters* (N) and Show compound parameters (C) switch between two representations of the *Noisy-MAX* parameters: net and loaded. Only one of the two buttons can be pressed at any given time. The *Show CPT* (CPT) button shows the CPT that corresponds to the *Noisy-MAX* specification. Finally, the *Show constrained columns* ( ) button brings forward trivially-filled columns of the *Noisy-MAX* specification that are normally not needed to parametrize a *Noisy-MAX* distribution but are handy in case one wants to change the order of states of any of the parent variables. The order of states of the parent variables is changed here only in the context of the current definition definition of interaction between the node and its parents and does not impact the definitions of the parent variables. Please see the *Noisy-MAX model* section for a tutorial-like introduction to the *Noisy-MAX* gate.

Chance-NoisyAdder nodes

The *Noisy-Adder* nodes are a special case of chance nodes modeling discrete random variables using the noisy-adder assumption. *Chance-NoisyAdder* nodes, similarly to *Chance-General* nodes, can have domains specified by *Identifiers*, *Intervals*, *Identifiers and intervals*, and *Identifiers and point values*. *Noisy-Adder* is a non-

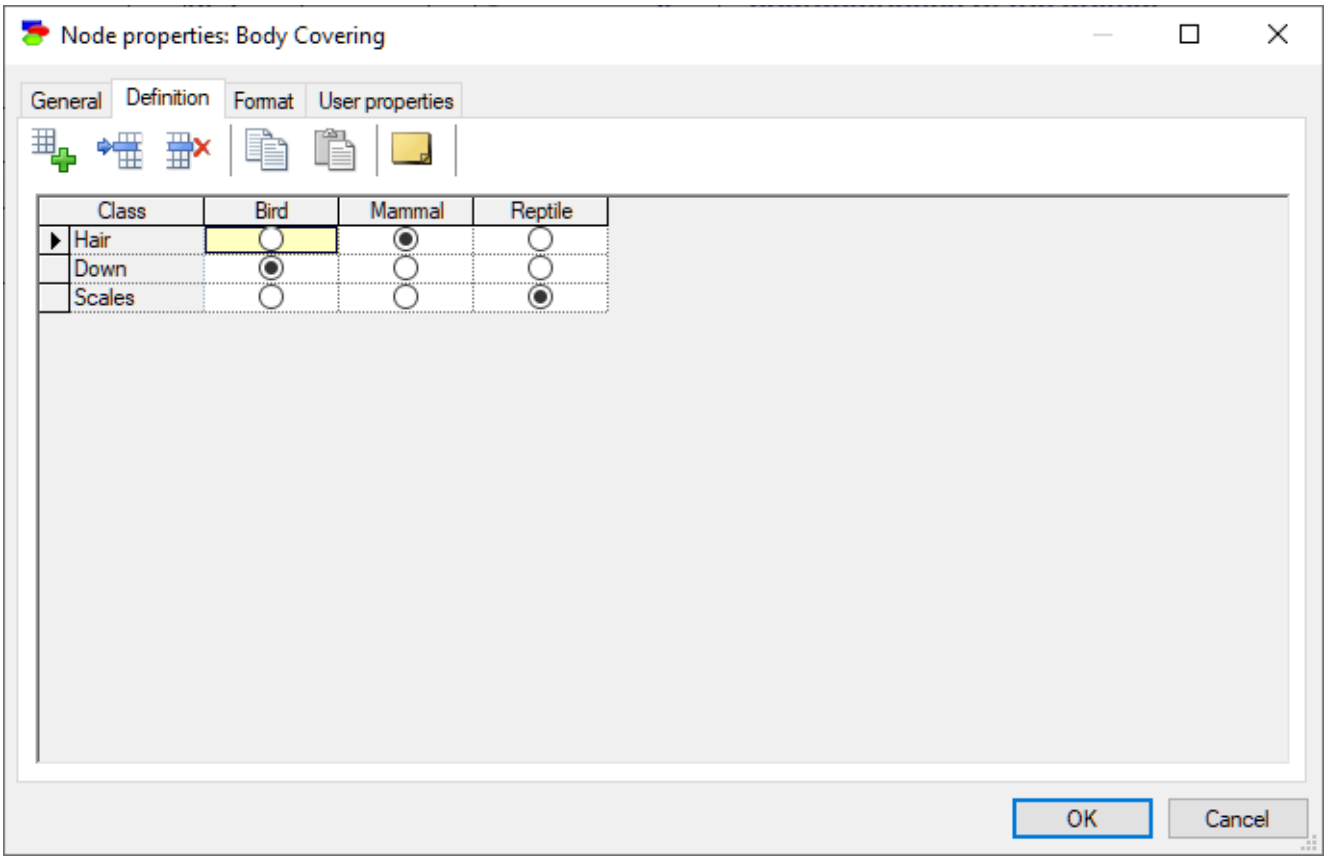
decomposable model that derives the probability of the effect by taking the average of probabilities of the effect given each of the causes in separation. Similarly to the *Noisy-MAX* nodes, noisy-adder nodes allow for specification of the interaction with their parents in a simplified way, requiring fewer parameters. The specifications can be viewed as a conditional probability table (CPT). Please see the *Noisy-Adder model* section for a tutorial-like introduction to the *Noisy-Adder* nodes.



Parent (Weight)	Liver Cancer (1)		Hepatitis B (1)		Hepatitis A (1)		Hepatitis C (1)		LEAK (1)
State	Present	Absent	Present	Absent	Present	Absent	Present	Absent	
Severe	0.33333333	0	0.33333333	0	0.33333333	0	0.33333333	0	0
Moderate	0.33333333	1	0.33333333	1	0.33333333	1	0.33333333	1	1
None	0.33333333	0	0.33333333	0	0.33333333	0	0.33333333	0	0

Deterministic nodes

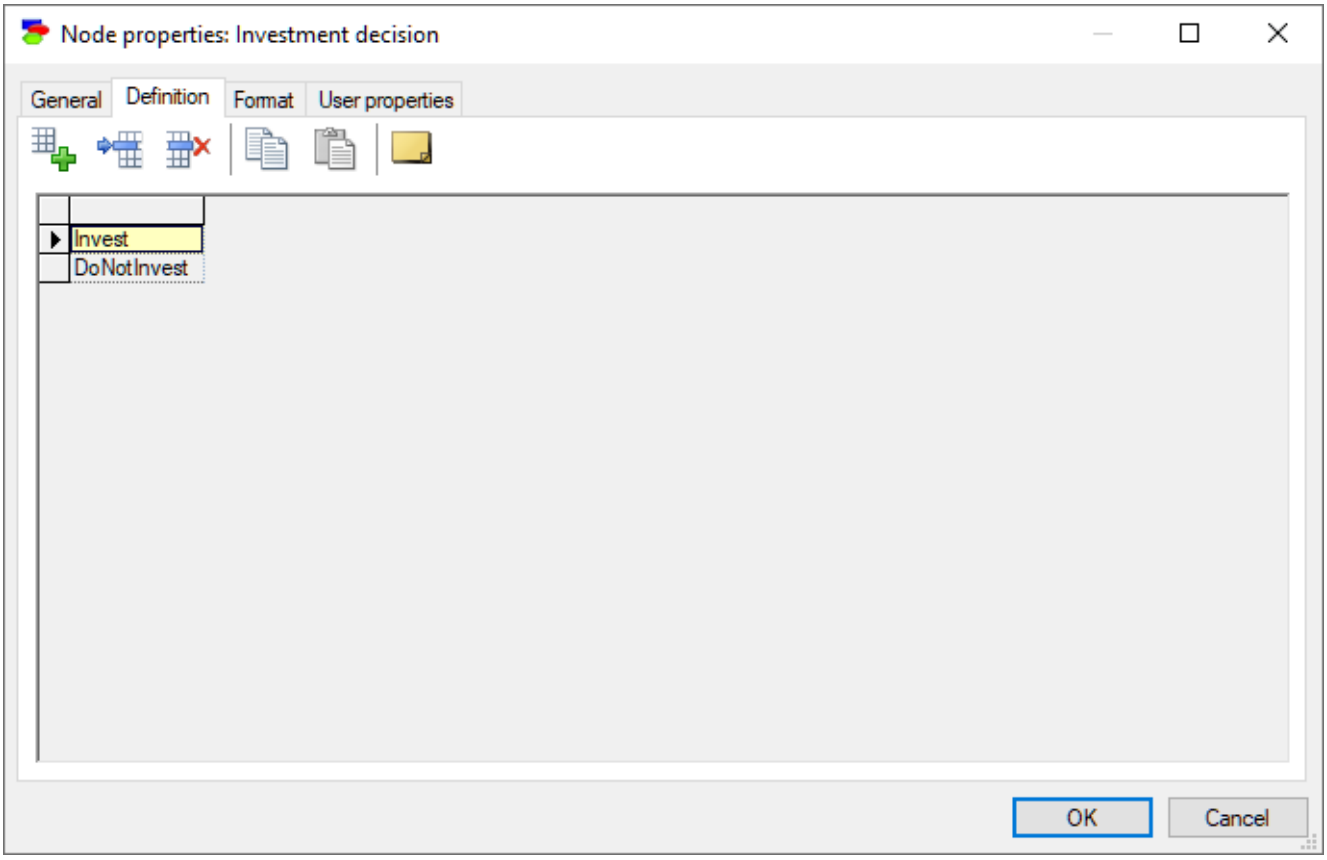
Deterministic nodes are discrete nodes that have no noise in them, i.e., we know their state with certainty if we know the states of their parents. *Deterministic* nodes, similarly to *Chance-General* nodes, can have domains specified by *Identifiers*, *Intervals*, *Identifiers and intervals*, and *Identifiers and point values*. The definition of a deterministic node is a truth table - knowing the values of the parents of a deterministic node defines its state. The only difference between a *Chance-general* node and a *Deterministic* node is that the values in the table of the latter are radio buttons rather than zeros and ones. Consider the following deterministic node expressing the relationship between class of an animal and its body covering. Once we know the class of the animal, we know the body covering: birds have down, mammals have hair, and reptiles have scales.



The definition corresponds to one in which we have 1.0 in exactly one of the cells of each column (the one with the radio button turned on) and zeros in all the other cells. All editing actions are the same as in *Chance-general* nodes.

Decision nodes

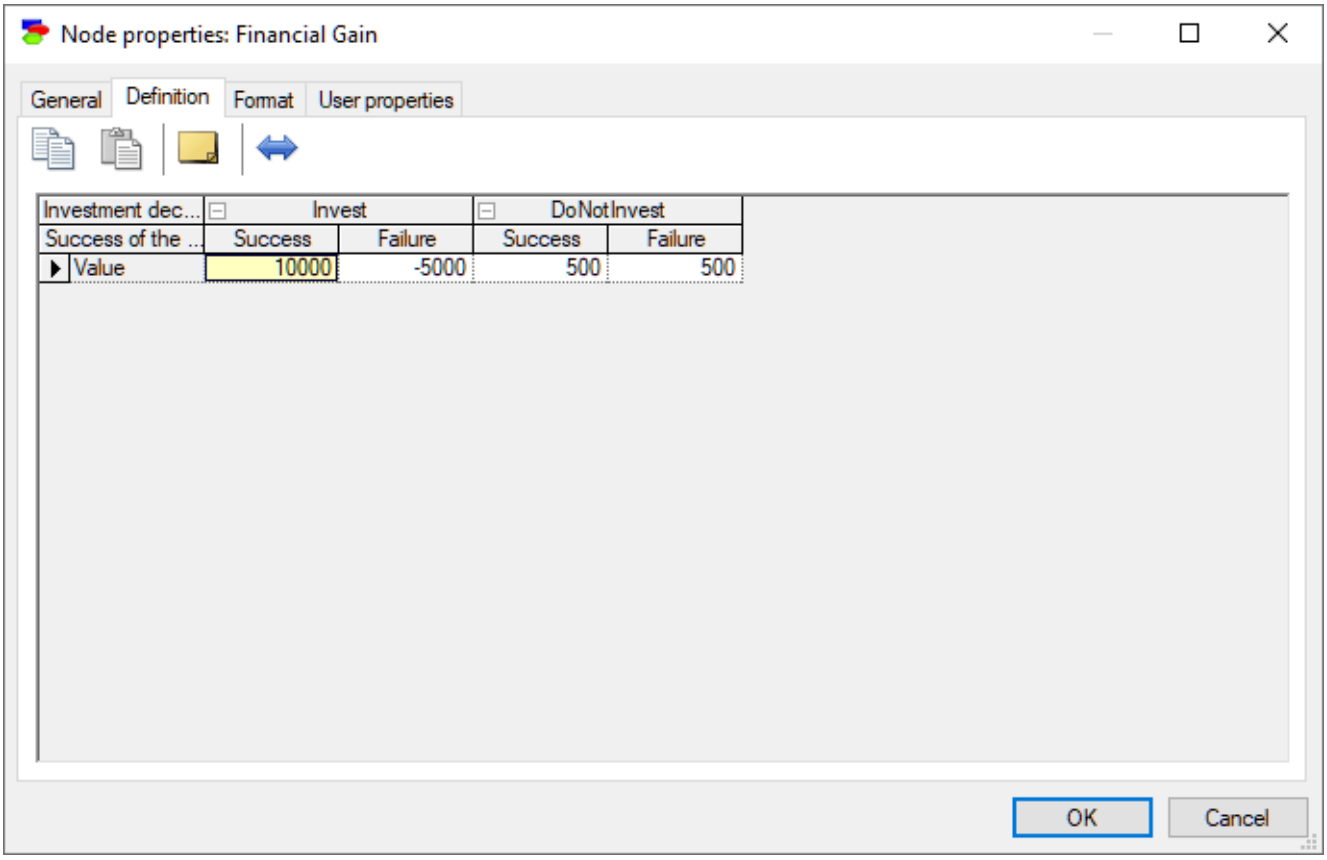
Decision nodes are discrete nodes that are under control of the decision maker and, hence, have no parents influencing them and no numerical specification. *Decision* nodes, similarly to *Chance-General* nodes, can have domains specified by *Identifiers*, *Intervals*, *Identifiers and intervals*, and *Identifiers and point values*.



All editing actions are the same as in *Chance-general* nodes.

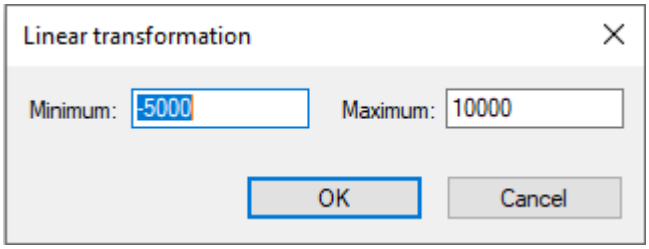
Utility nodes

Utility nodes model decision maker's preferences for various states of their parents. *Utility* nodes are continuous and can assume any real values. When their parents are discrete, each cell in the value node defines a measure of preference of the combination of states of these parent nodes.



In the example above, node *Financial Gain* expresses the monetary gain for all combinations of states of the nodes *Investment decision* and *Success of the venture*. In expected utility theory, decisions are optimal when they maximize the expected utility. It turns out that the maximization process is independent on the unit and the scale of the utilities but only on their relative values. Utility values are determined up to a linear transformation.

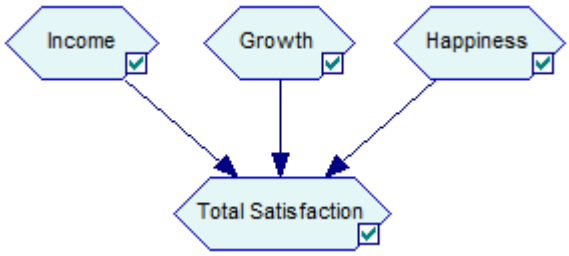
You can transform your utility function to a different scale by pressing the *Linear transformation* ( ) button, which will invoke the following dialog:



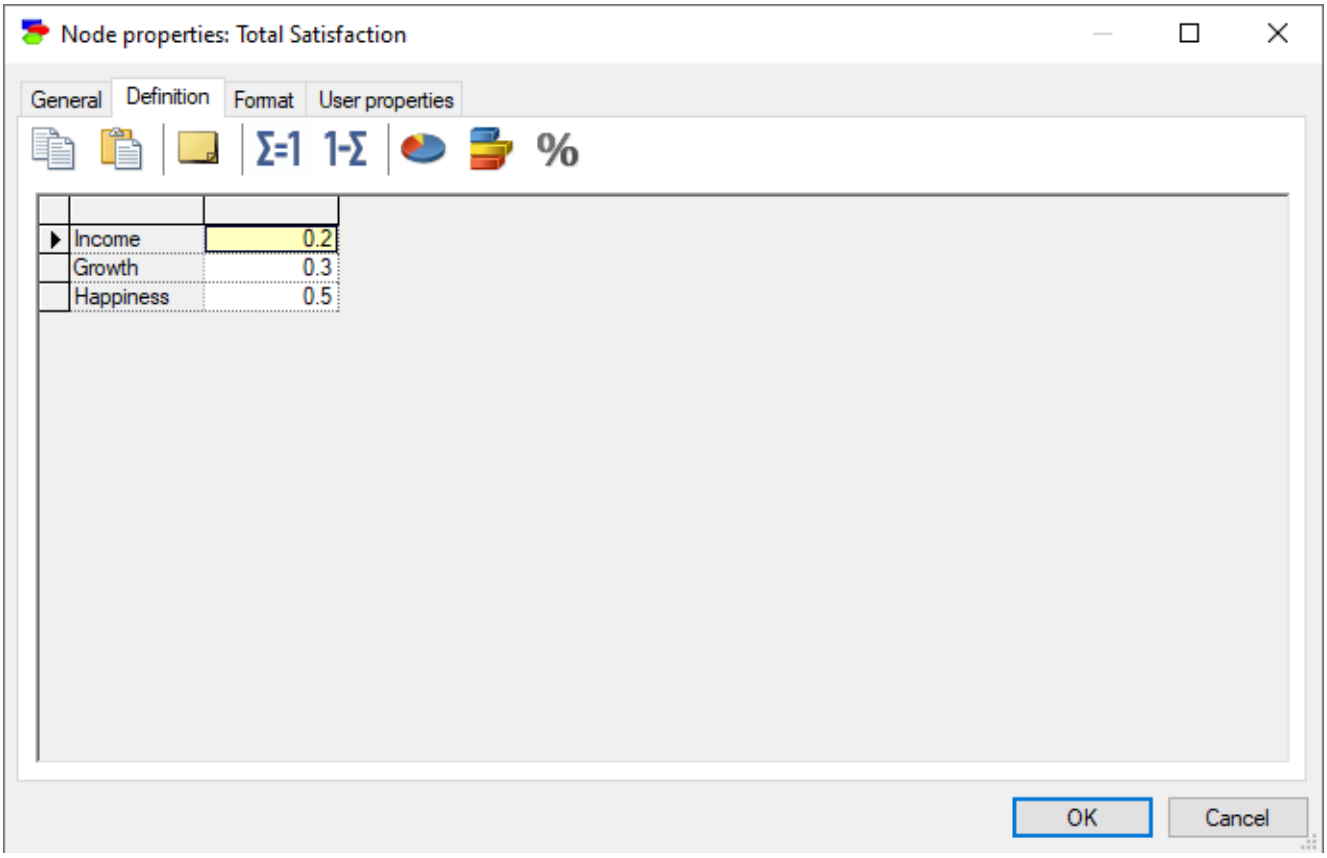
The fields *Minimum* and *Maximum* contain the lowest and the highest value in the table, respectively. When these values are edited and the *OK* button is pressed, the values in the table are linearly transformed to the new interval. This operation is useful in case the numbers in the table are utilities. Very often a decision modeler wants to have the utility function located in a given interval, usually [0..1] or [0..100].

ALU nodes

ALU or *Additive-Linear Utility* nodes are continuous nodes that model decomposable utility functions. An *ALU* node brings together utilities of several *Utility* nodes in an additively-linear function. Consider the following model, in which the node *Total Satisfaction* is a function of three utility nodes: *Income*, *Growth* and *Happiness*.



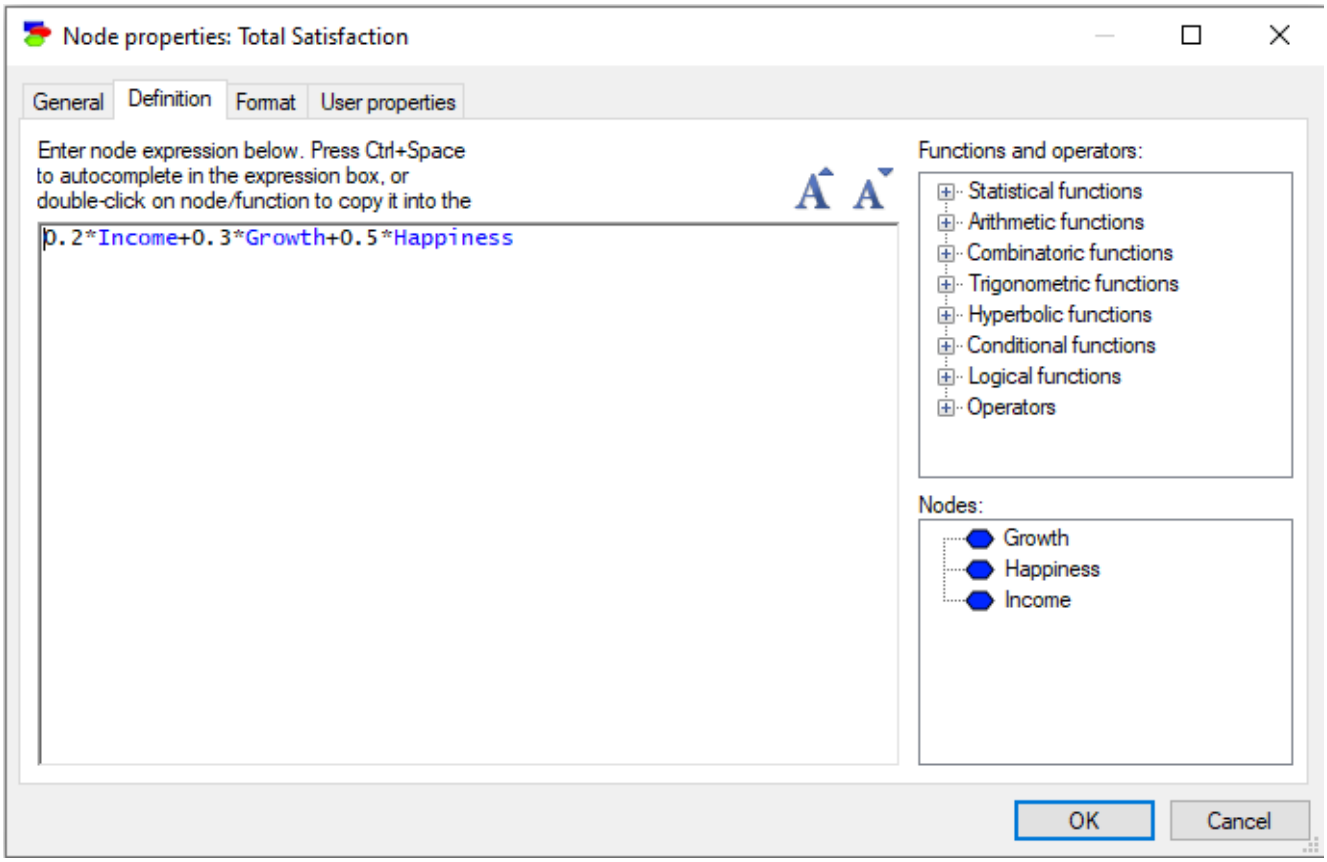
Let us assume that these combine linearly with weights 0.2, 0.3, and 0.5, respectively, i.e., we are dealing with the following function of the utilities of the three components: *Total Satisfaction* = 0.2 *Income* + 0.3 *Growth* + 0.5 *Happiness*. The following definition of an *ALU* node captures this interaction



All editing actions are the same as in *Chance-general* nodes. In particular, the *Normalize* (Σ=1) and *Complement* (1-Σ) buttons are useful because it is customary to make the weights in an ALU function add up to 1.0. Because of this, GeNIe offers also graphical utility elicitation.

MAU nodes

MAU or *Multi-Attribute Utility* nodes generalize the ALU nodes to any function of the parent utilities. The *Definition* tab of a MAU node looks as follows:

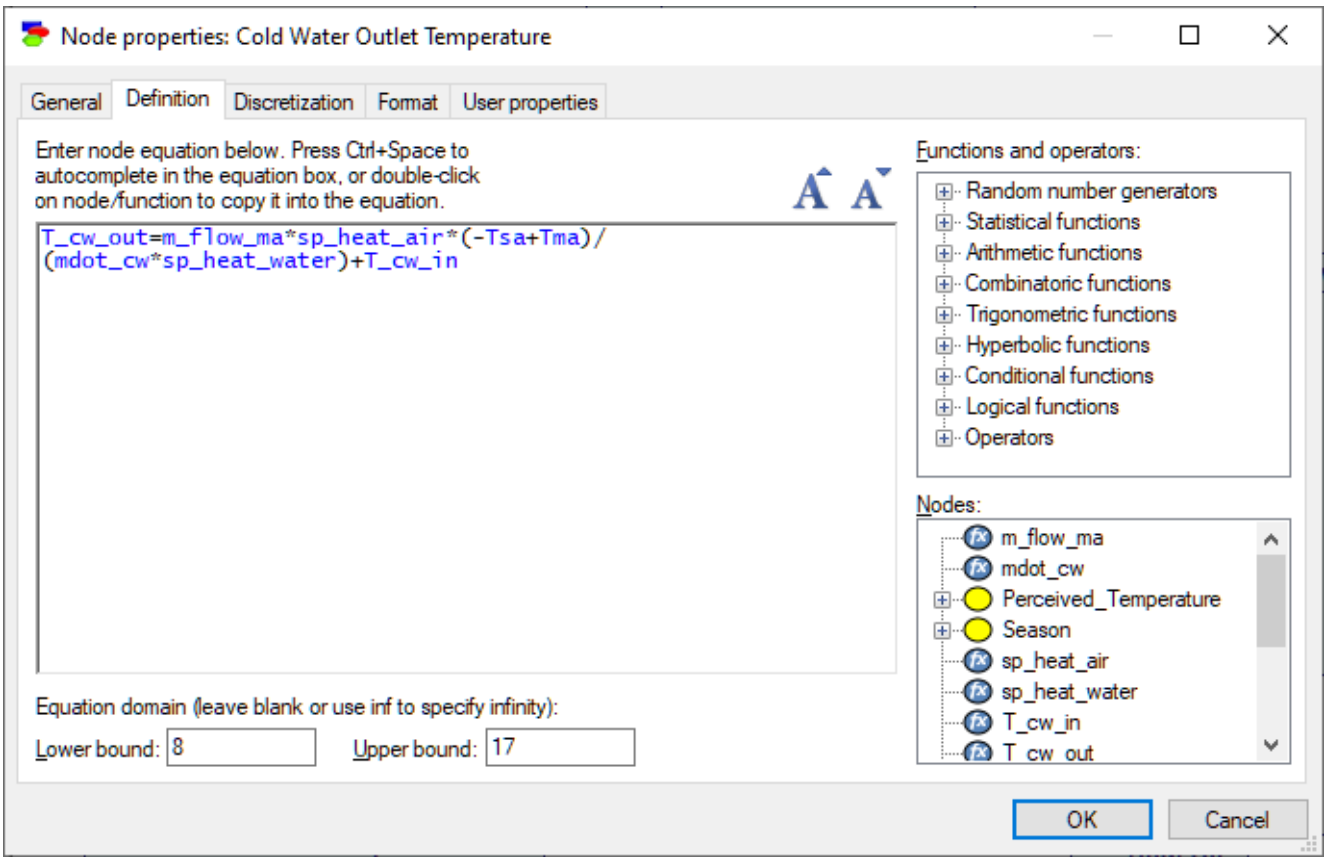


The dialog allows for entering any function of the parent utilities. The current definition shows the additive linear function corresponding to the example used for the *ALU* nodes. In addition to arithmetic operators, GeNIe offers a number of arithmetic, combinatoric, trigonometric, hyperbolic, and logical/conditional functions, which can be typed directly or selected from the lists in the right-hand side window pane. When editing an equation, press *Ctrl*+Space to auto complete in the *Expression* box or double-click on node/function to copy it into the equation. Pressing on the increase/decrease font size icons or using *Ctrl*+wheel allows to change the font size.

Equation nodes

Equation nodes are continuous chance nodes, whose interaction with their parents can be described by means of an equation. *Equation* nodes are a bridge between Bayesian networks and systems of simultaneous structural equations, popular in physics and engineering applications. The following dialog shows a variable describing the cold water outlet temperature of a building. It is described by the following equation:

$$Tcw_{out}=mflow_{ma}*sp\_heat_{air}*(-T_{sa}+T_{ma})/(mdot_{cw}*sp\_heat_{water})+Tcw_{in}$$

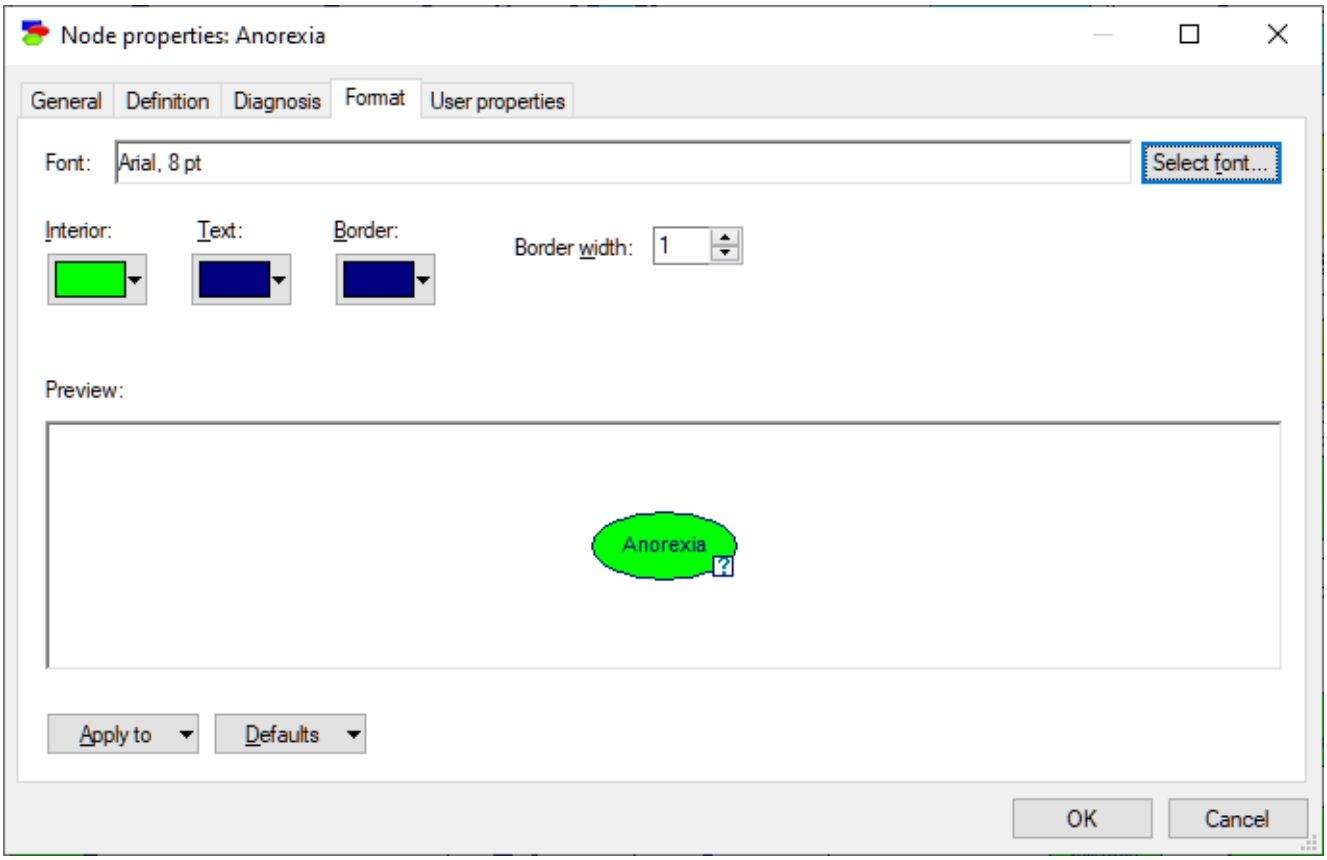


The window panes on the right-hand side of the tab contain a set of standard functions known by GeNIe and the set of nodes that participate in the equation. When editing an equation, press *Ctrl*+Space to auto complete in the *Expression* box or double-click on node/function to copy it into the equation. Pressing on the increase/decrease font size icons or using *Ctrl*+wheel allows to change the font size.

Another element of the dialog describes the domain of the variable, which is any real number between *Lower bound* and *Upper bound*. Because a complete freedom in the model specification prevents GeNIe from using an exact algorithm that will solve the model, the default algorithms are based on stochastic sampling. Specification of the domain of the variable helps in limiting the sampling domain and improves the algorithm efficiency, allowing for warnings in case values fall outside of the specified bounds. One or both domain bounds can be infinite. Just leave the corresponding box empty or enter the text `-inf` and `inf`, respectively.

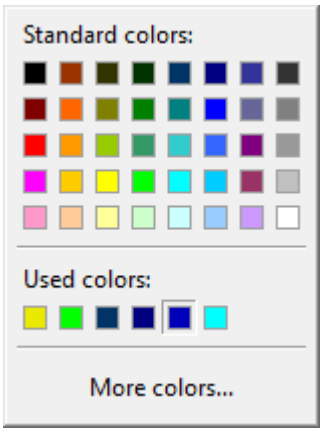
Format tab

The *Format* tab is used to specify how the node will be displayed in the *Graph View*. It has a preview window which displays the altered view of the node with the new settings.

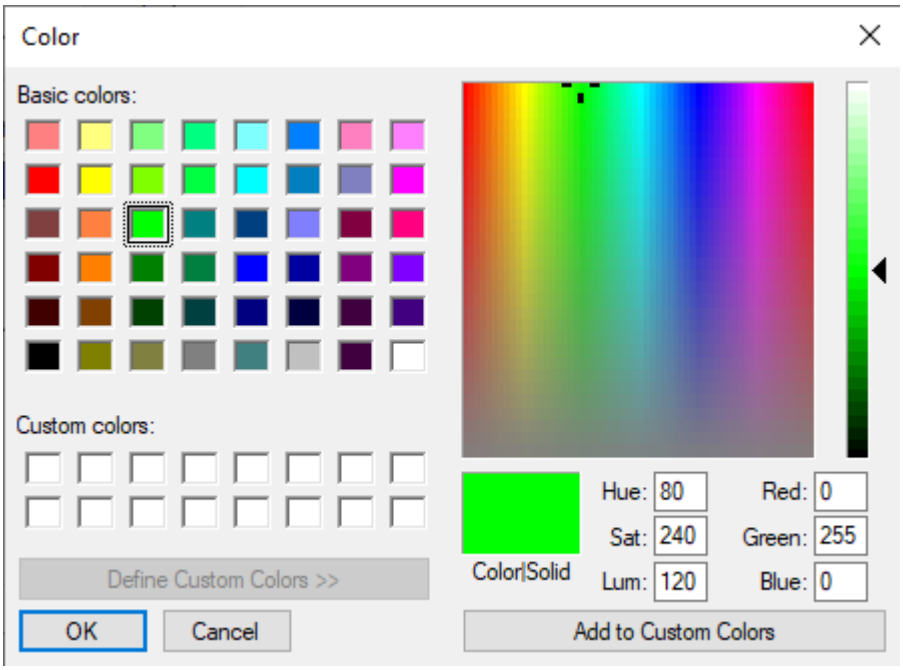


Select font button is used to select the font of the text displayed within the node. Clicking on this button will cause a standard *Font Selection* dialog box to be displayed. You can select font type, style, and size from this box.

Interior, *Text*, and *Border* colors allow for choosing the colors of the interior, font, and border colors, respectively. Clicking on any of these buttons brings up a color selection palette, example of which is shown below



You can select any of the 40 palette colors or can bring up the possibility to select from more colors by clicking on *More colors*.... This will display a selection dialog as follows:

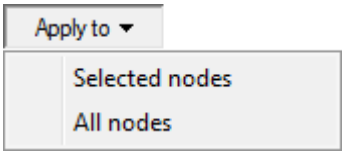


You can select any color on the rainbow-like part of the window and then add the selected color to one of the 16 custom colors (by clicking on the button *Add to Custom Colors*).

Border width allows to change the width (in pixels) of the border around the node. This parameter can be used to change the node visibility in the *Graph View*. The default width for most nodes is one pixel. Submodel nodes have borders of width two.

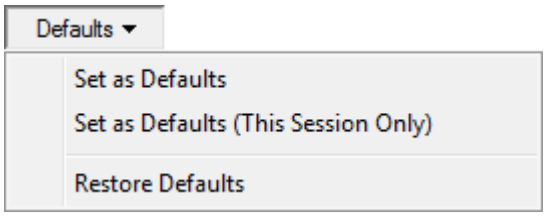
The *Node properties* dialog is powerful but unless you want the changes to apply only to one node that you have used to invoke the dialog, you need to make careful choices in two separate menus: the *Apply to* menu and the *Defaults* menu.

The *Apply to* menu gives two choices: *Selected nodes* and *All nodes*.



If you make no choice here, your changes will apply only to the current node (its name is displayed in the preview window). If you want to apply the changes to a group of nodes (selected before invoking this dialog), choose *Selected nodes*. If you want to apply the changes to all nodes in the network, choose *All nodes*. Generally, to change the formatting properties of a group of nodes, it is best to select them first (please consider using the corresponding selection commands from the *Edit Menu* or from the *Diagnosis menu* (nodes can be selected by type, diagnostic type, or color), once the diagnostic extensions are enabled) and then right-click on any of the selected nodes to invoke *Node properties*. (Invoking node properties by double-clicking on a node has a side-effect of selecting the node and clearing any existing selections.)

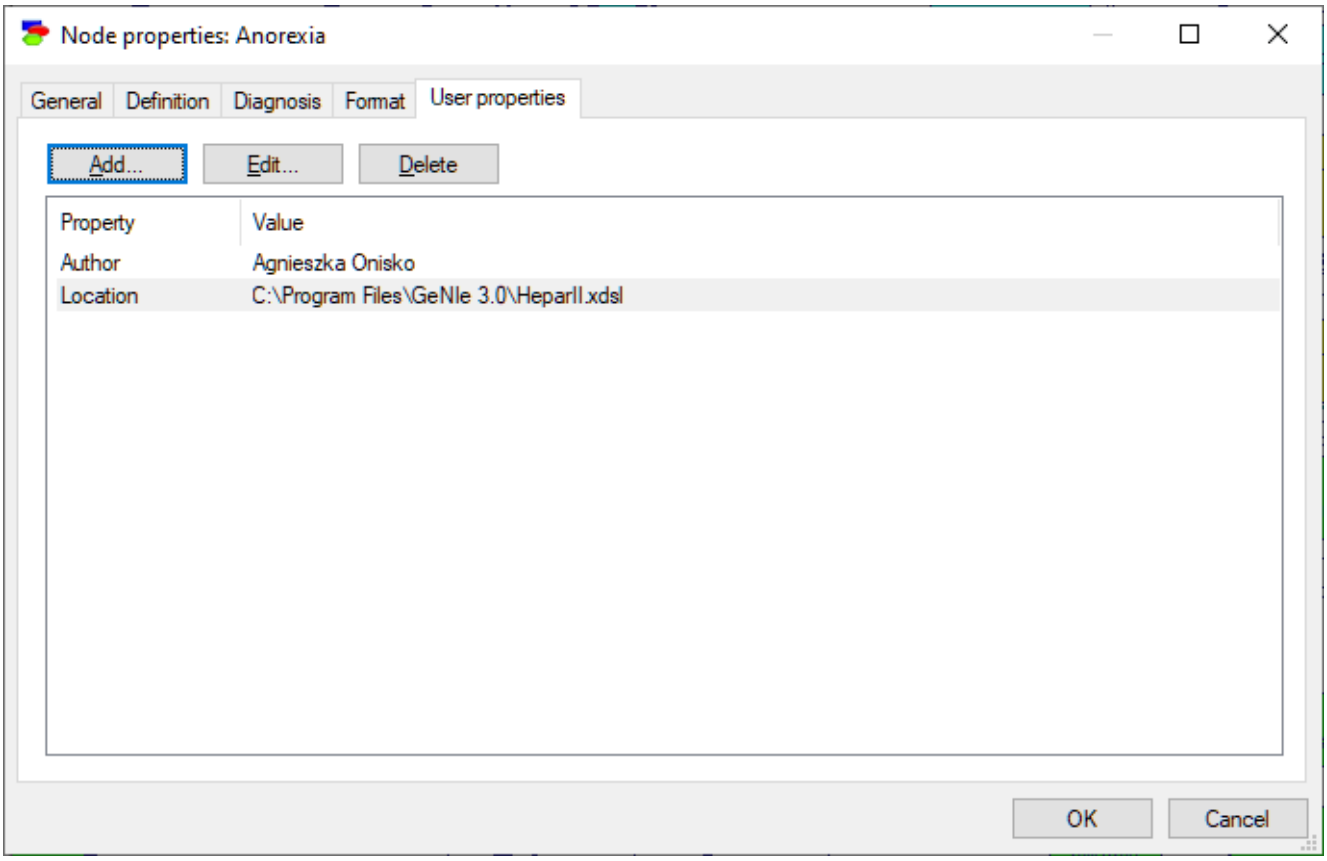
The *Defaults* pop-up menu allows you to modify the program defaults, which will make all new nodes appear the way you have specified. The menu gives you three choices: *Set as Defaults*, *Set as Defaults (This Session Only)*, and *Restore Defaults*.



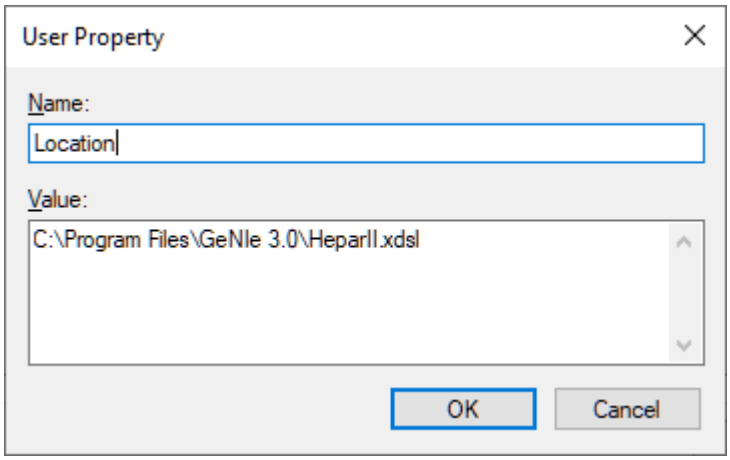
Set as Defaults set the new format as the default format for all new nodes. This default setting will be maintained between multiple sessions of GeNIe. *Set as Defaults (This Session Only)* makes the new setting a default but only for this session only. After you quit and start GeNIe again, the previous default settings will be restored. *Restore Defaults* restores the factory settings.

User properties tab

The *User Properties* tab allows the user to define properties of the node that can be later retrieved by an application program using SMILE. For example, the following tab for the node *Anorexia* contains two properties: *Author* with the value *Agnieszka Onisko* and *Location* with the value *C:\Program Files\GeNIe 2.1\HeparII.xdsl*. Neither GeNIe nor SMILE use these properties and they provide only placeholders for them. They are under full control and responsibility of the user and/or the application program using the model. GeNIe only allows for editing them.



Both *Add* and *Edit* invoke the following dialog:

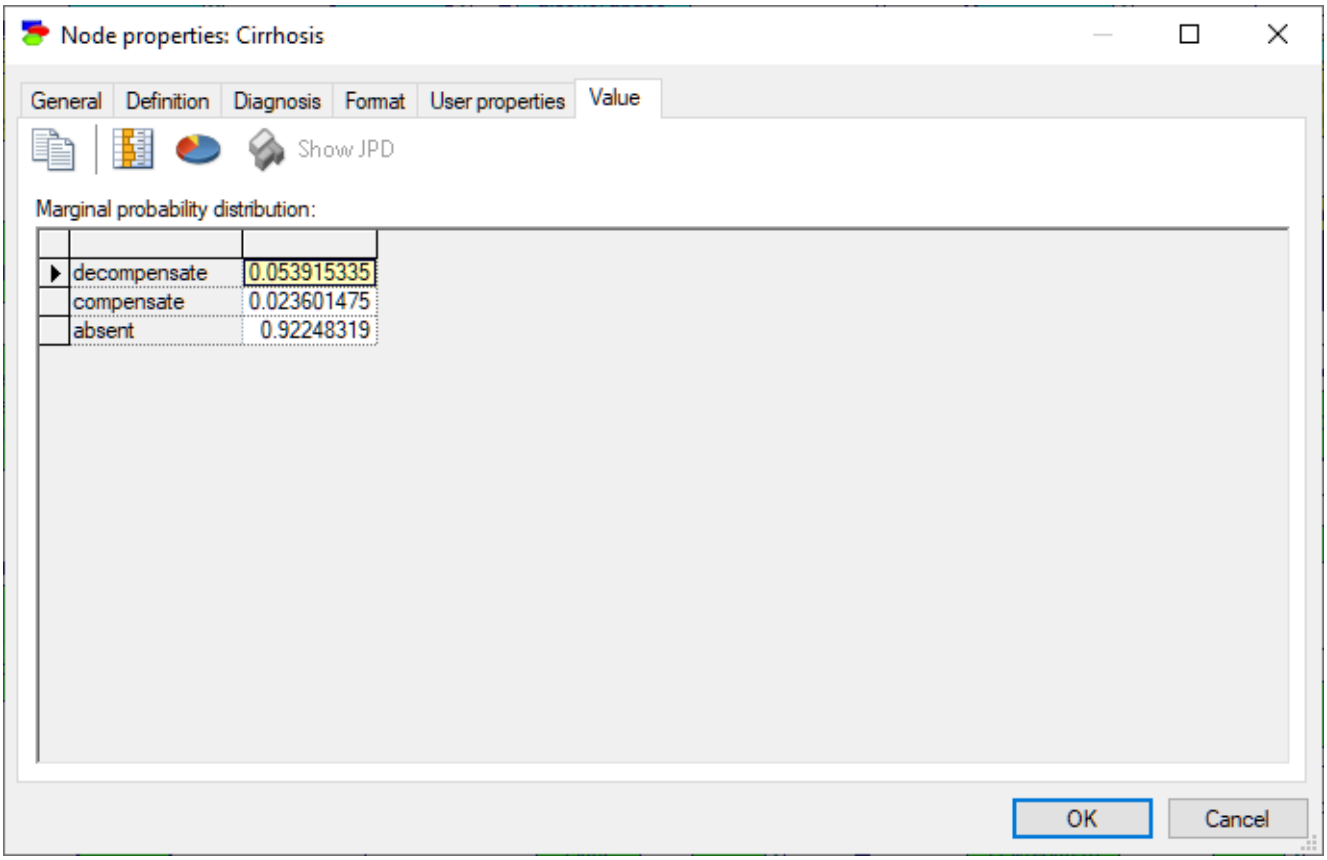


Delete removes the selected property.

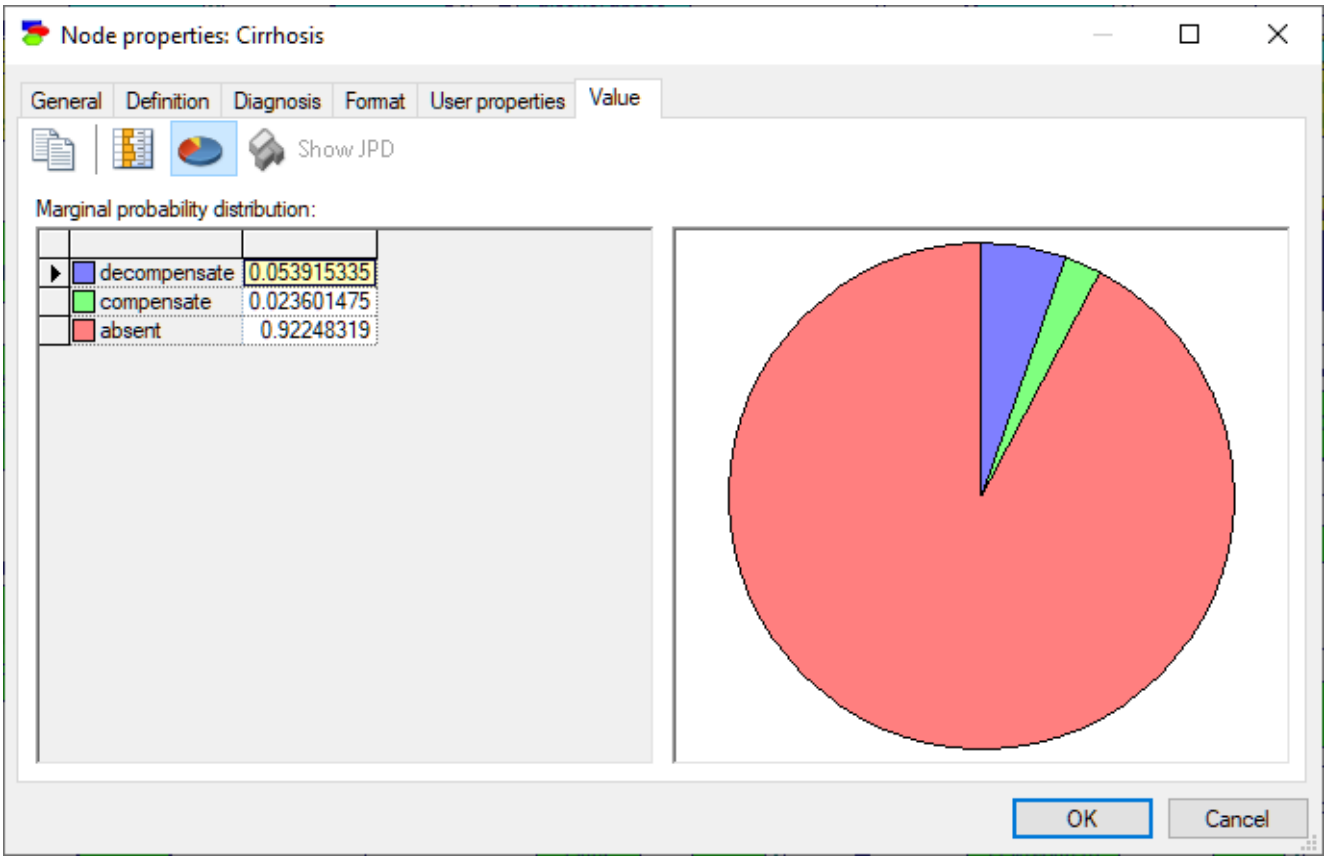
Value tab

The *Value* tab is visible only if an algorithm has been applied to the model and the node contains the calculated values (typically, the marginal probability distribution or the expected utilities). The precise format for the *Value* tab depends on the node type.

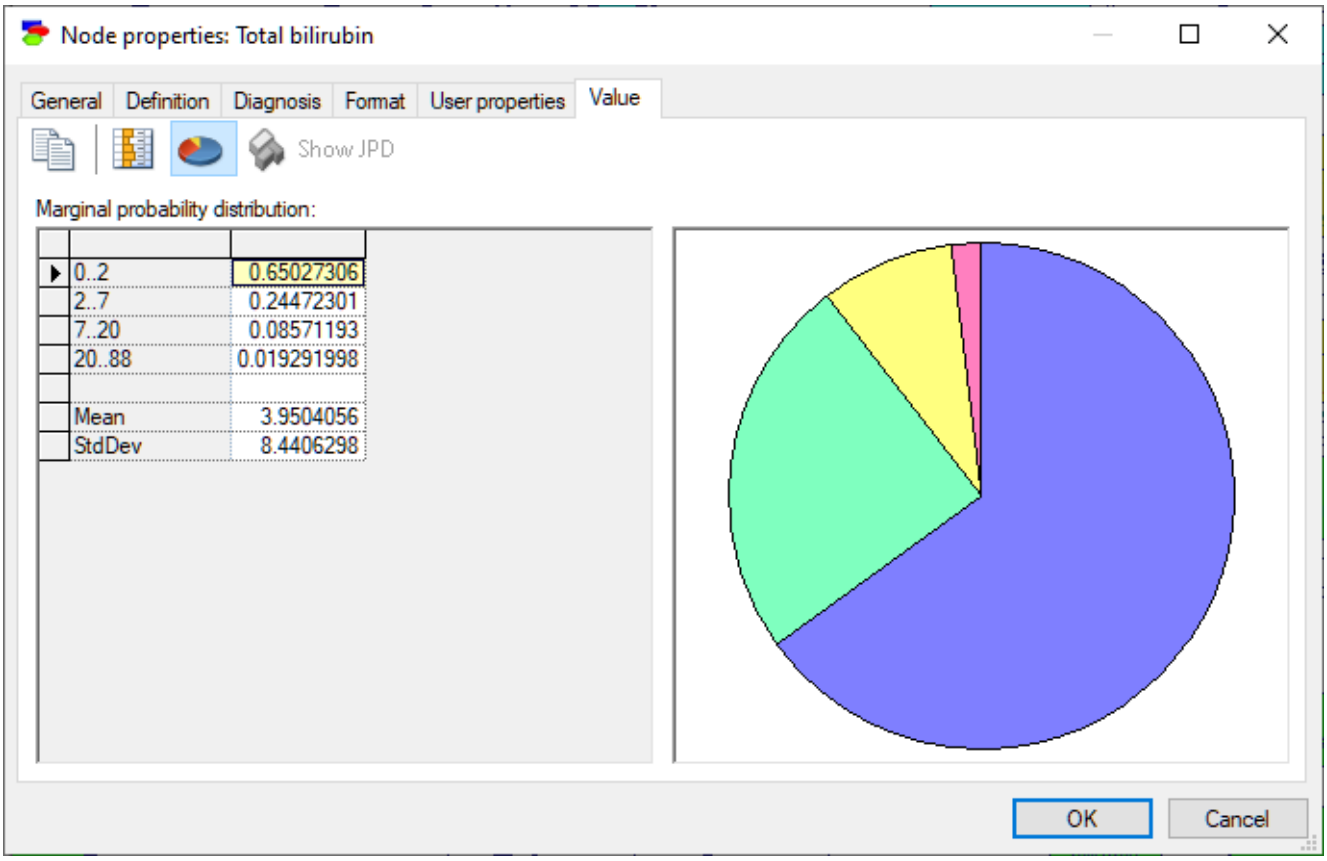
Discrete *Chance* and *Deterministic* nodes show their marginal probability distributions.



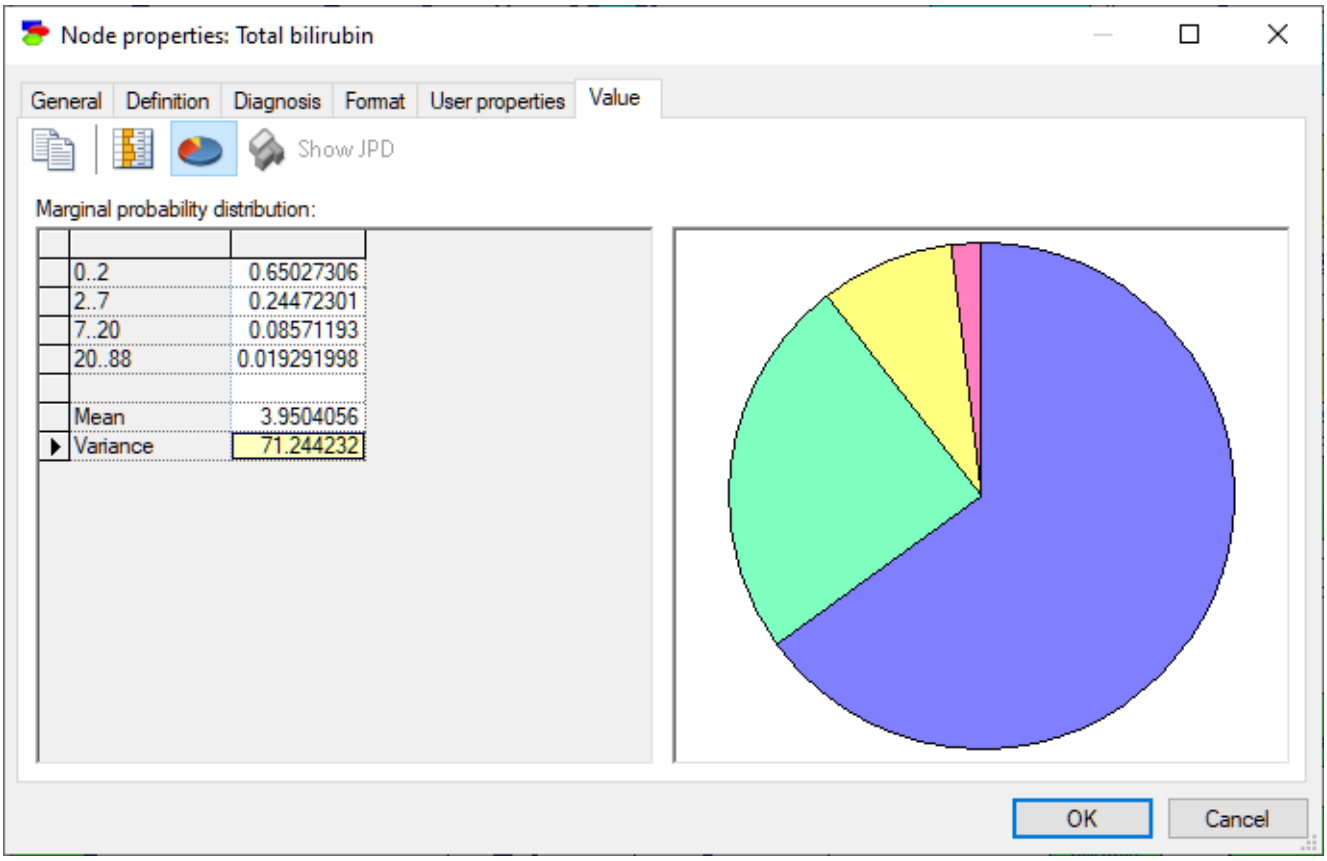
To make the display graphical, press the *Piechart* ( ) and/or the *Show QuickBars* ( ) buttons. They add a pie chart display of the posterior marginal distribution and bars in the result spreadsheet (a list of states with their posterior probabilities) respectively.



Nodes that are numerical but discrete show, in addition to probability distribution over its states, their mean and standard deviation.

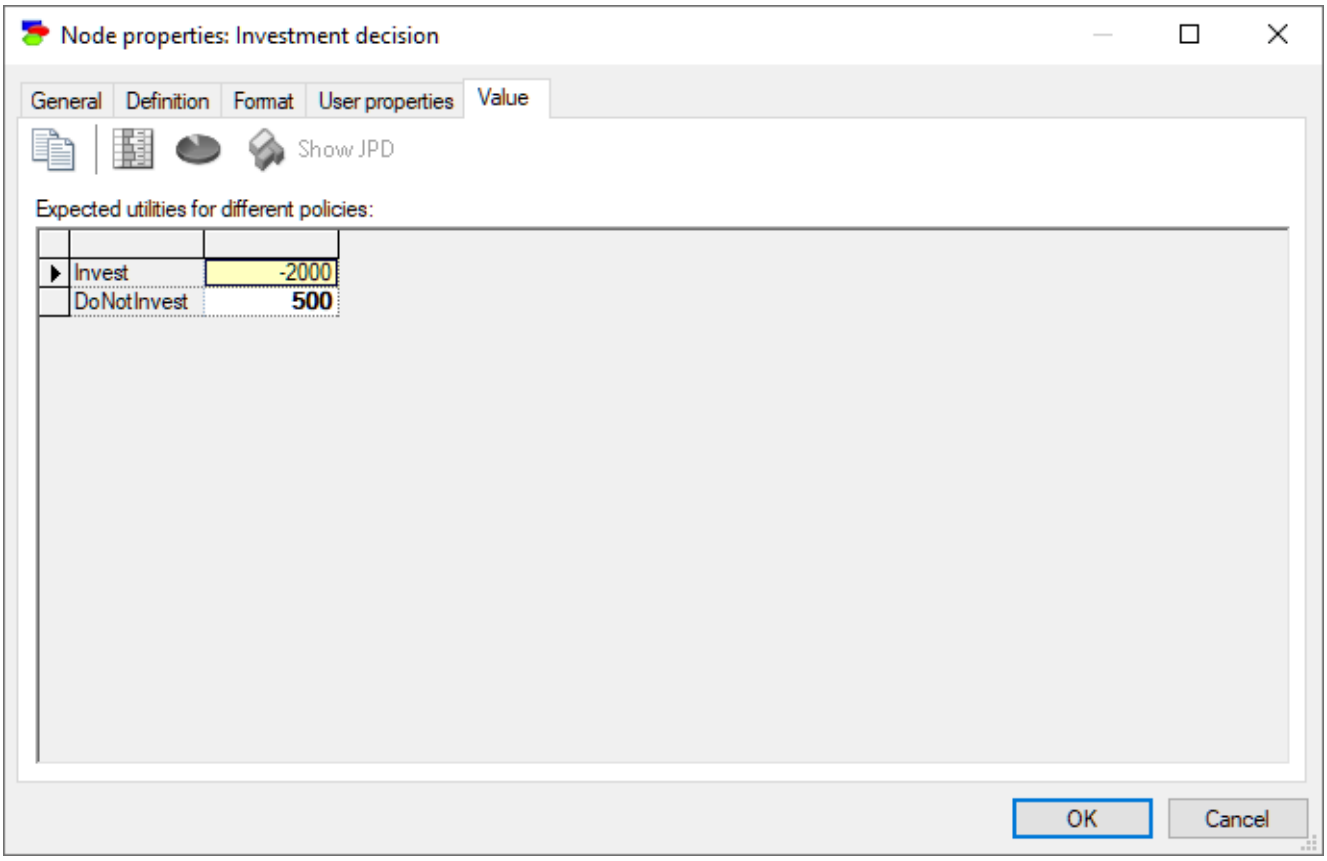


Standard deviation can be turned into variance by double-clicking on the row showing standard deviation:

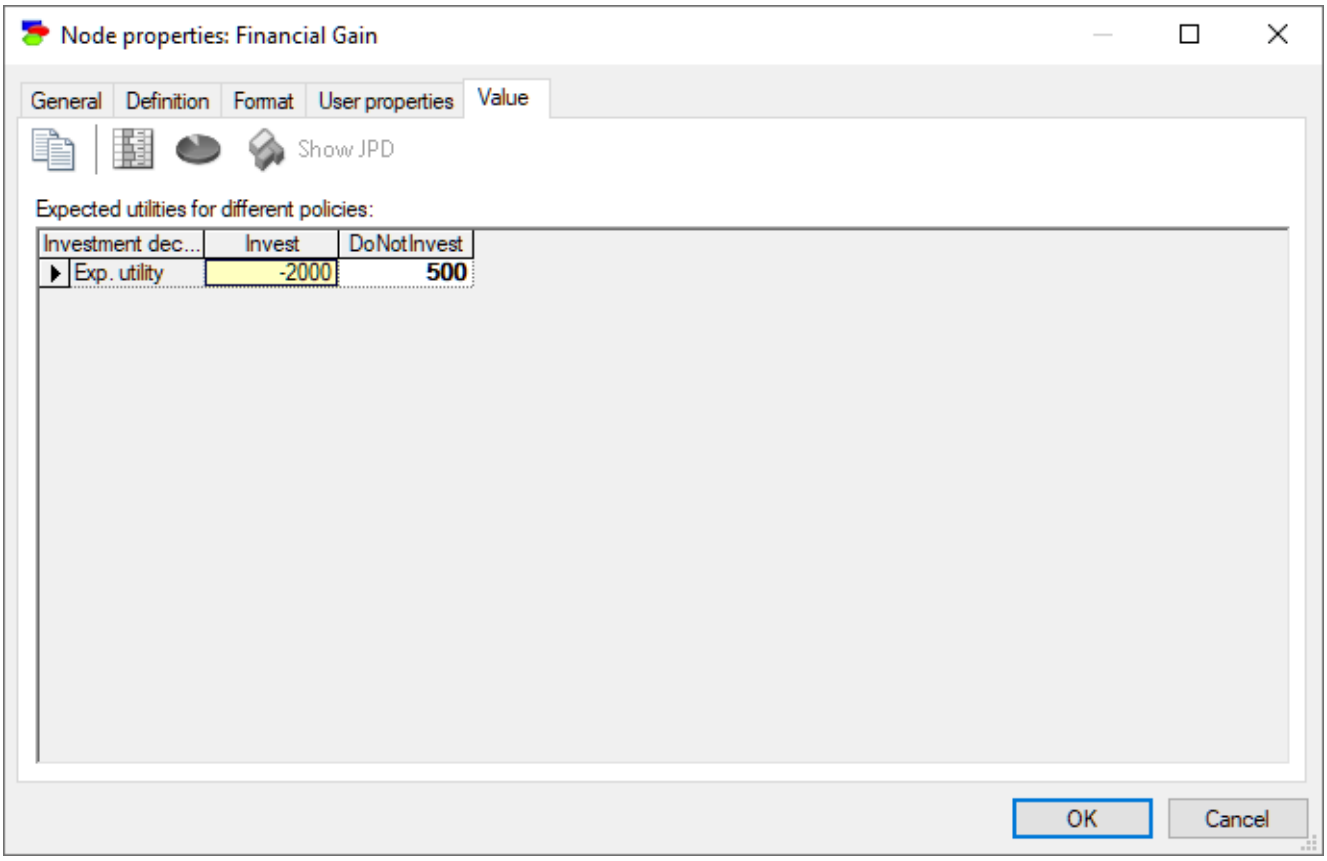


Calculation of the mean and standard deviation/variance is straightforward in nodes with numerical point values. In case of interval nodes, GeNIe assumes that the distribution is uniform within each of the intervals.

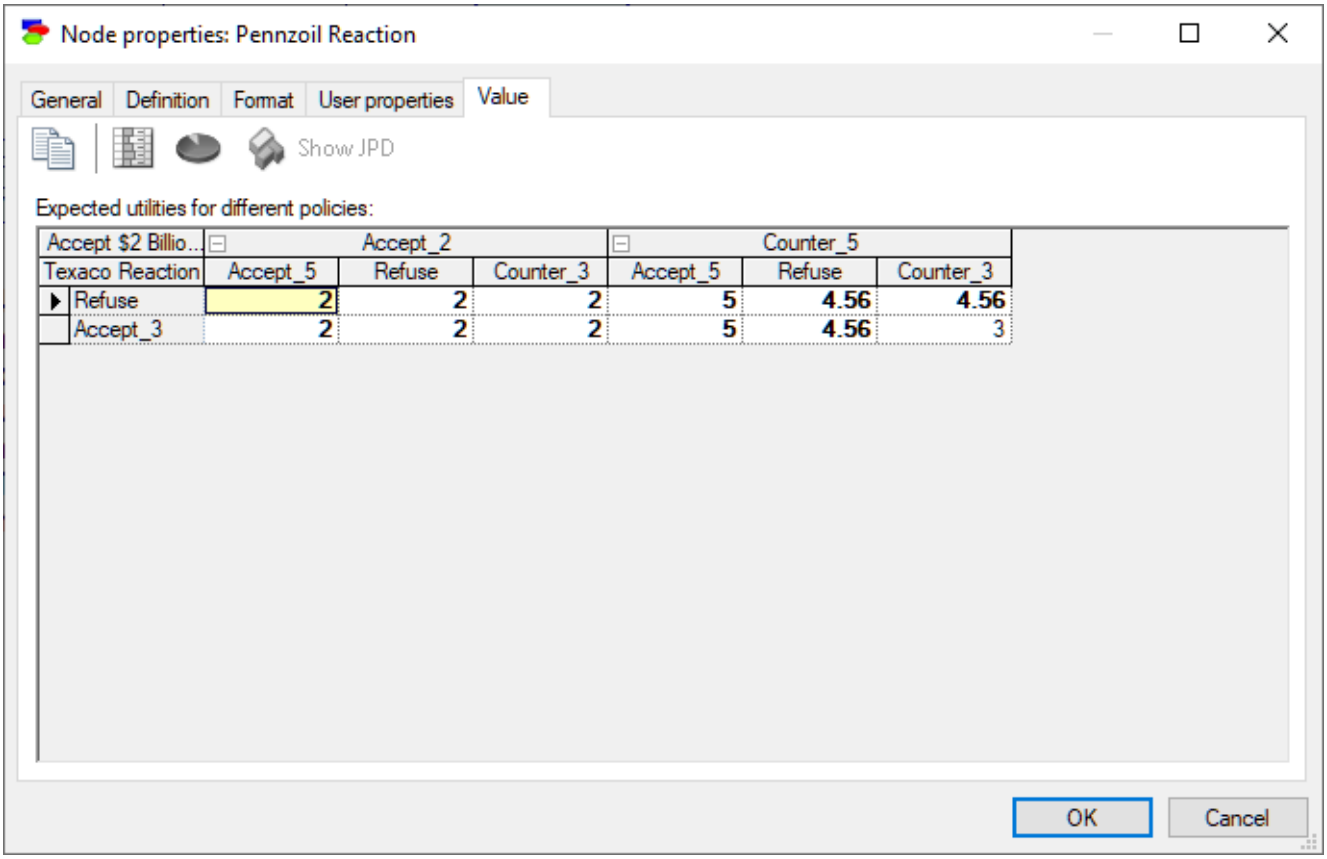
Decision, *Utility*, *ALU* and *MAU* nodes show expected utilities of each of the decision options. There is a subtle difference between the two displays in that *Decision* nodes show the result for each of the states of the decision node (rows of the result table) and in case of *Utility* nodes, states of the *Decision* node are indexing the result. Here is the value tab of a *Decision* node:



The same information shown by the *Utility* node of the same model:

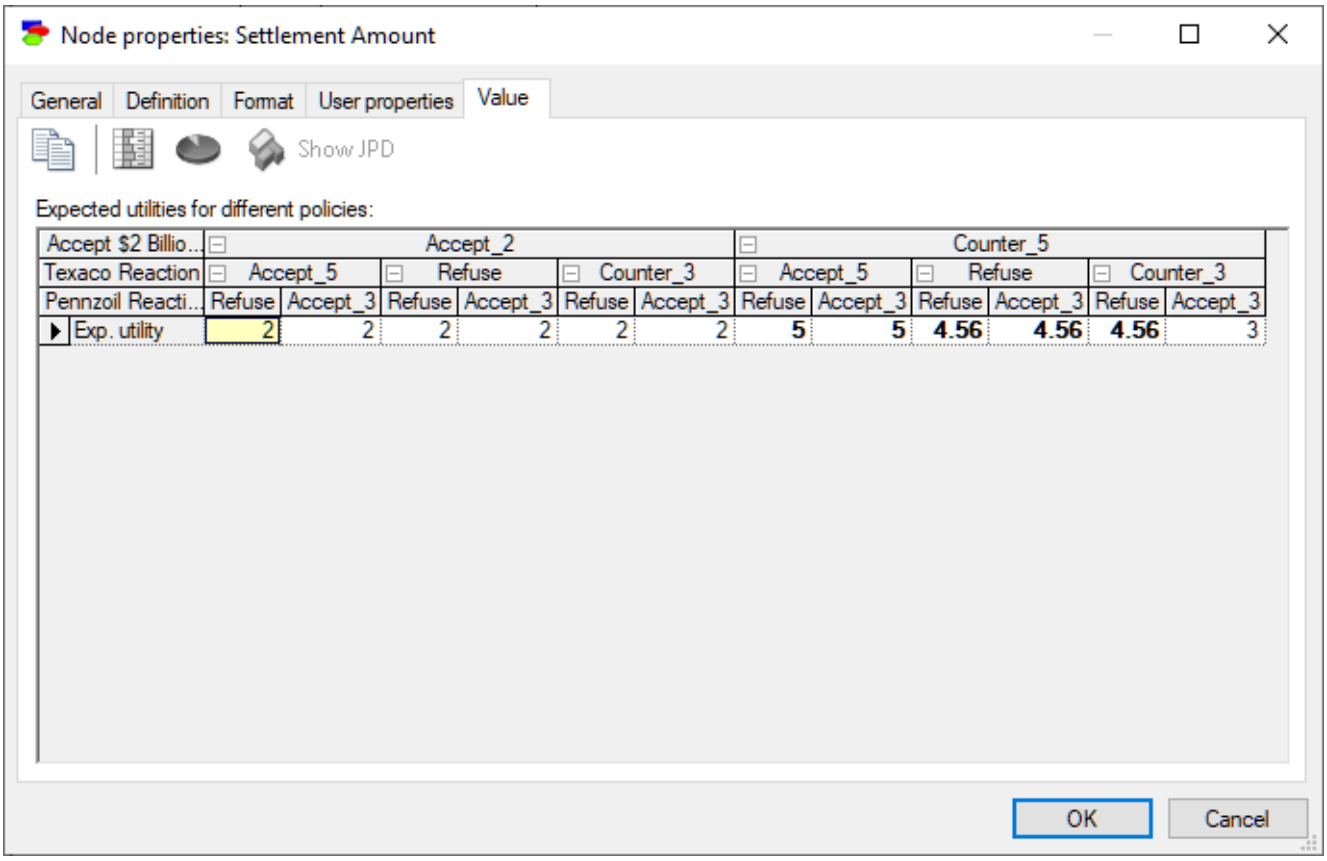


When there are unobserved decision nodes that precede the current node or when there are unobserved chance nodes that should have been observed, GeNIe shows the results as a table indexed by the unobserved nodes.

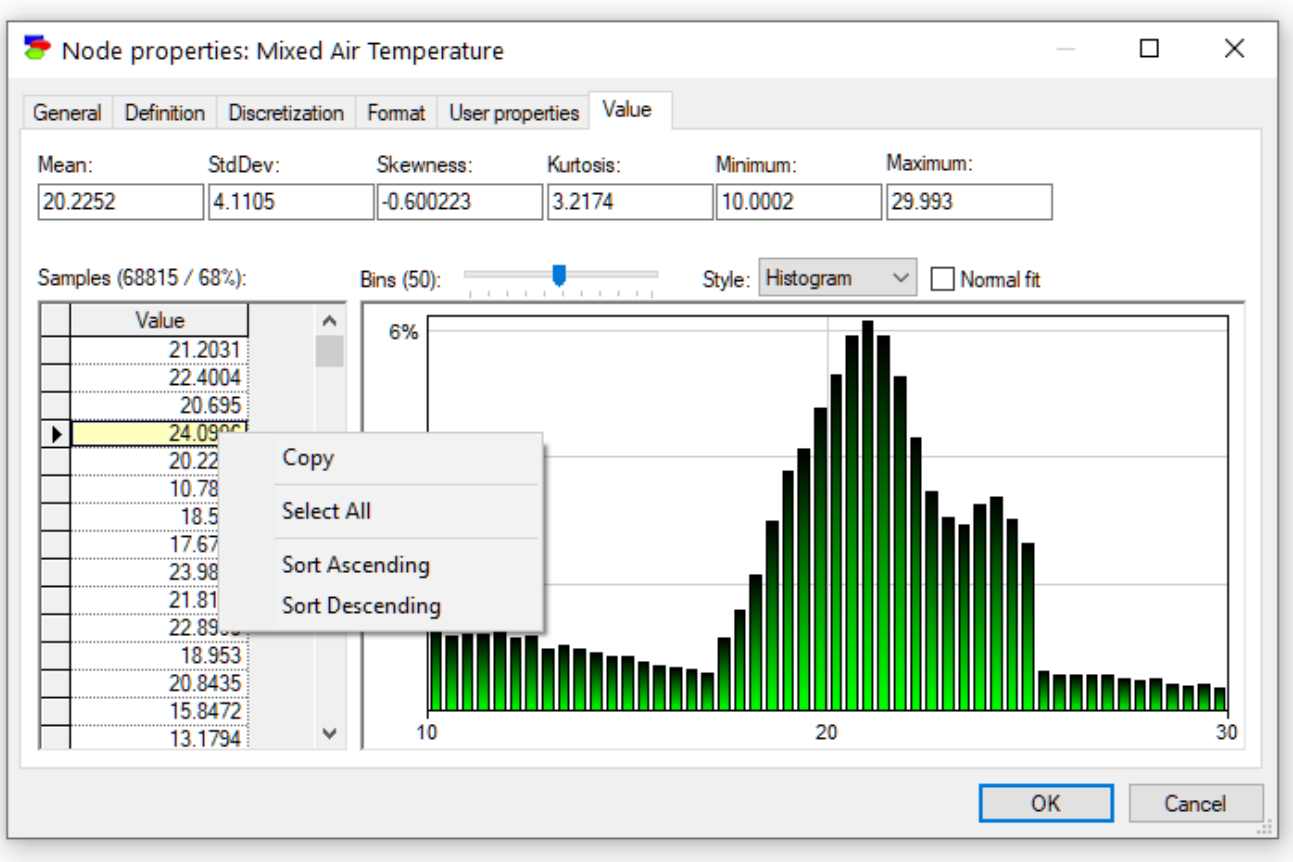


Accept $2 Billio...	Accept_2			Counter_5		
Texaco Reaction	Accept_5	Refuse	Counter_3	Accept_5	Refuse	Counter_3
Refuse	2	2	2	5	4.56	4.56
Accept_3	2	2	2	5	4.56	3

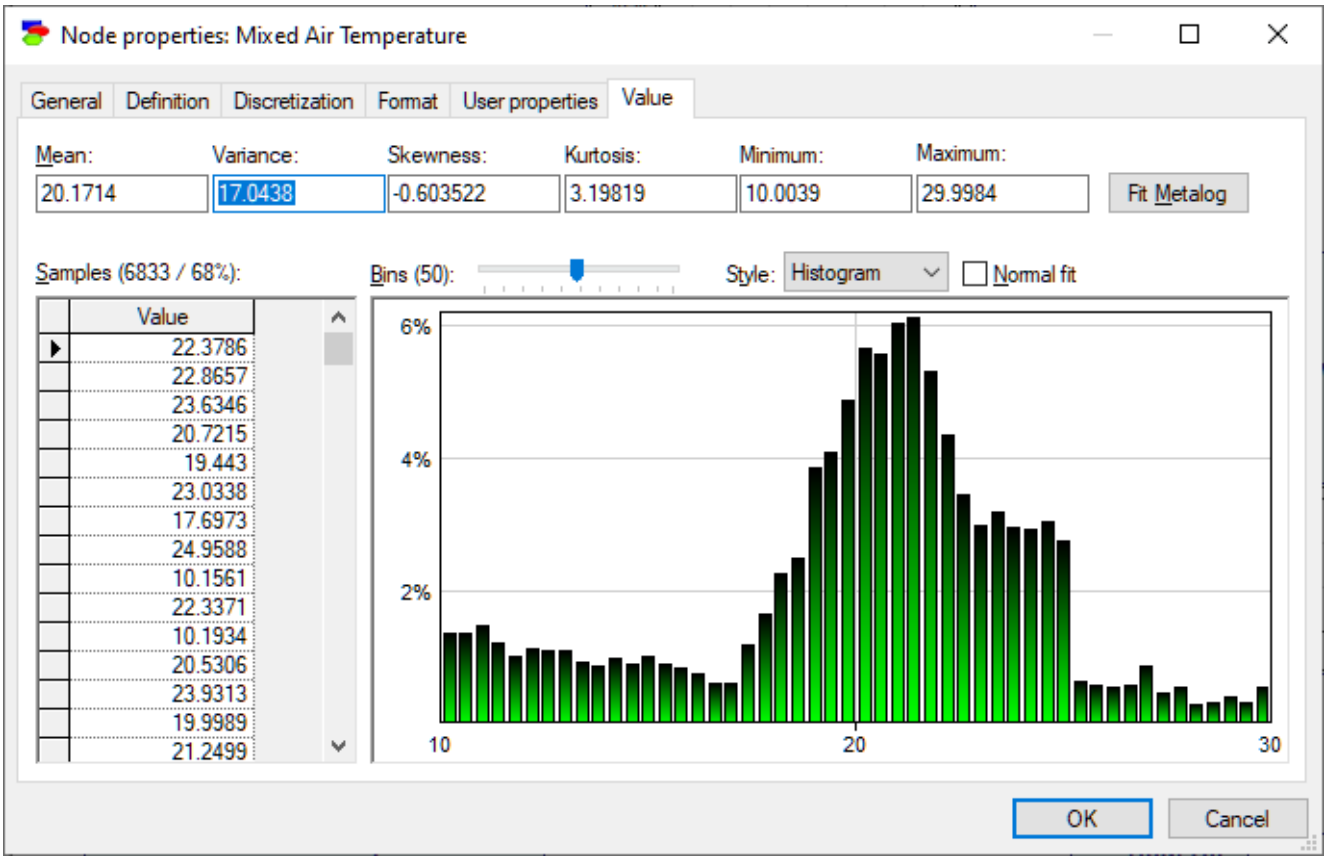
The same information shown in the *Utility* node of the same model:



Equation nodes are continuous and show the results in form of a plot of the samples obtained during the most recent run of the sampling algorithm. The tab shows the first four moments of the distribution: *Mean*, *StdDev*, *Skewness* and *Kurtosis* along with the *Minimum* and the *Maximum*. By default, the plot shows the histogram of the samples:



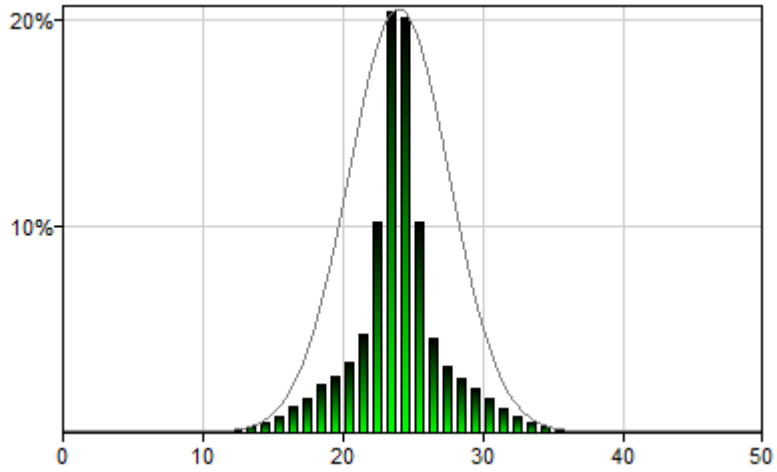
Standard deviation can be turned into variance by double-clicking on the cell showing standard deviation



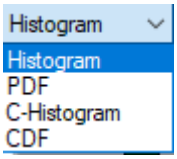
The samples themselves are preserved and displayed in the vector on the left-hand side. They can be selected, copied, and sorted using a right-click context pop-up menu (shown in the screen short). Similarly to the histogram interface in the data pre-processing module, the user can change the number of bins in the histogram.

Fit Metalog opens the *Fitting Metalog distribution to data* dialog that allows for fitting a metalog distribution to the samples (in other words, fitting a metalog distribution to the histogram shown in the *Histogram* pane).

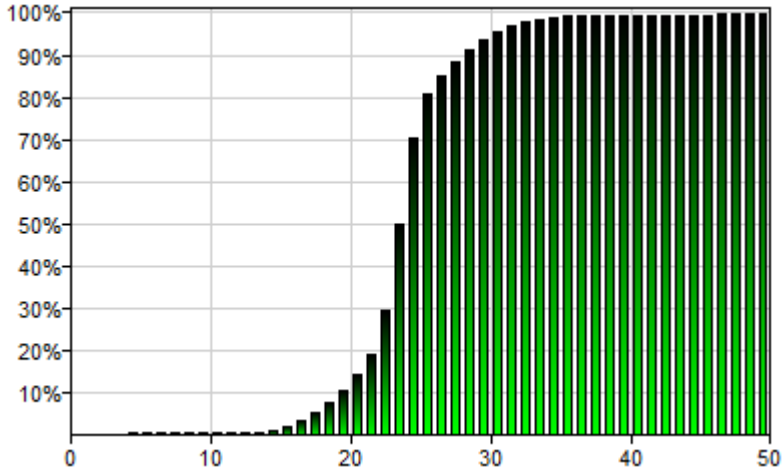
Normal fit check box draws a Normal distribution over the domain of the variable with the mean and standard deviation equal to those of the sample.



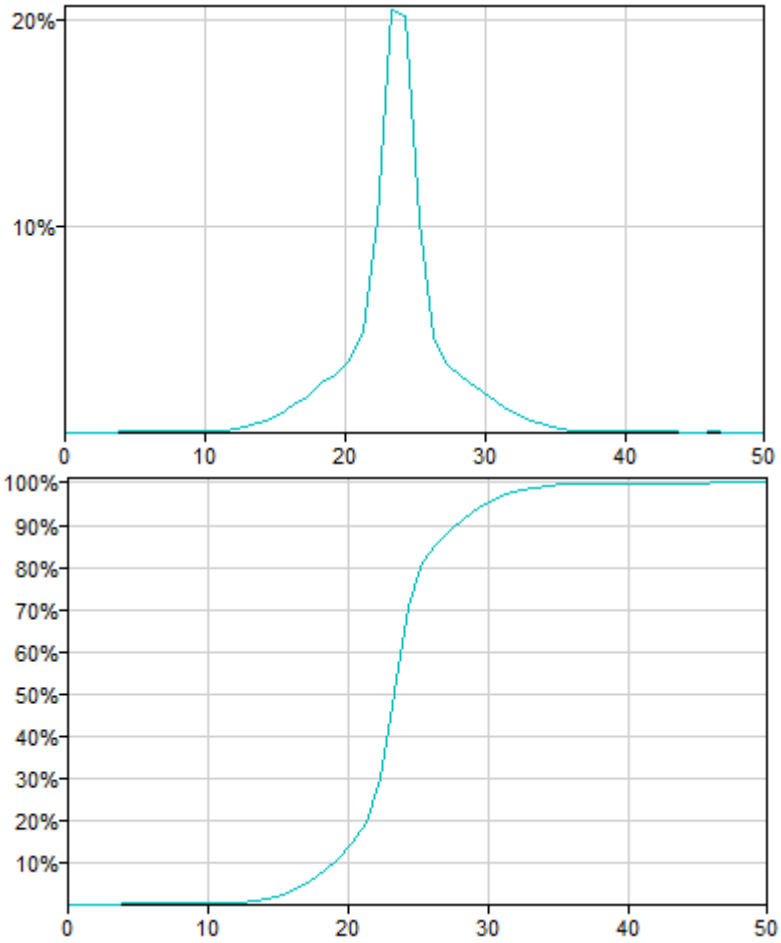
Style pop-up menu allows for choosing a different plot:



C-Histogram is a cumulative version of the *Histogram* plot.



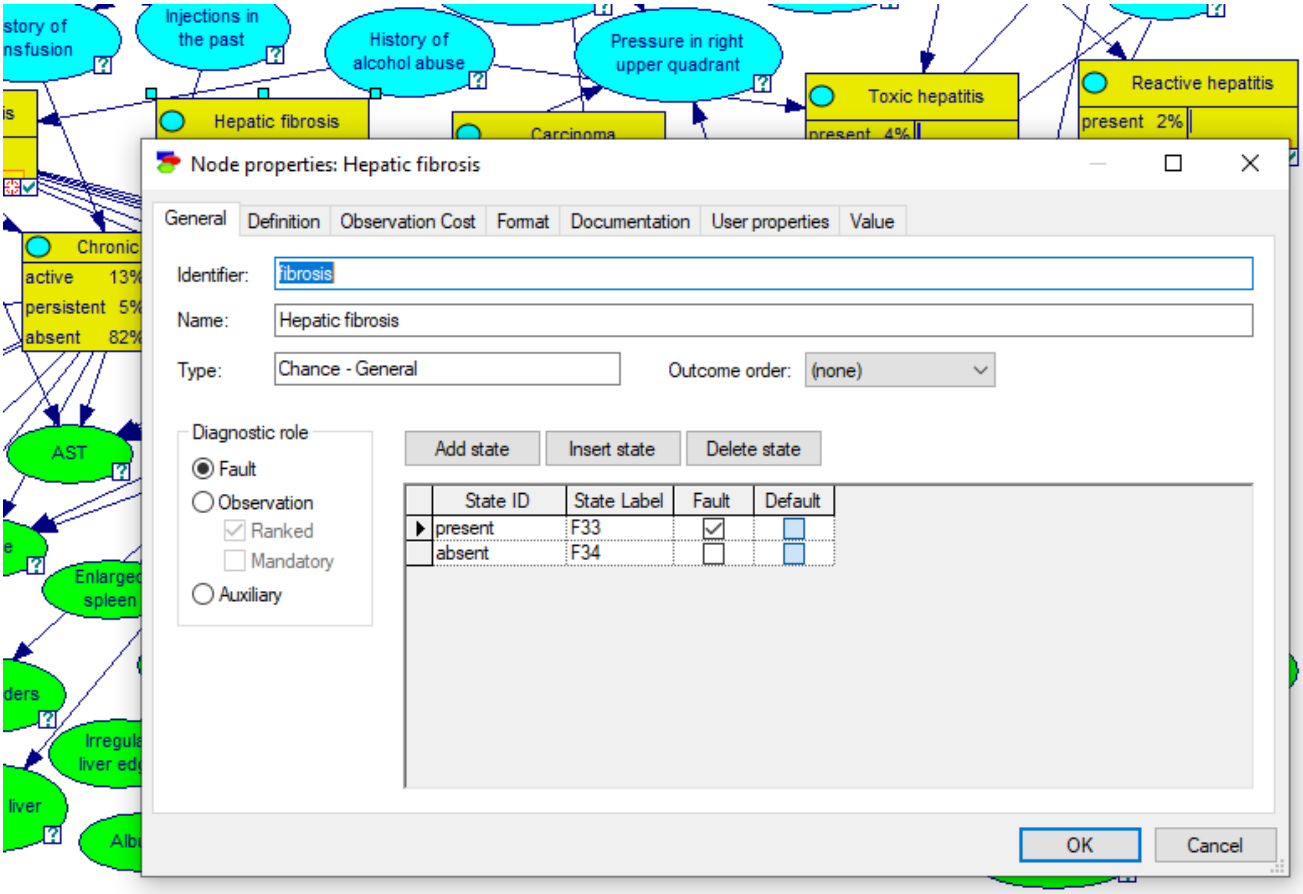
PDF and *CDF* are abstractions of the two histogram plots:



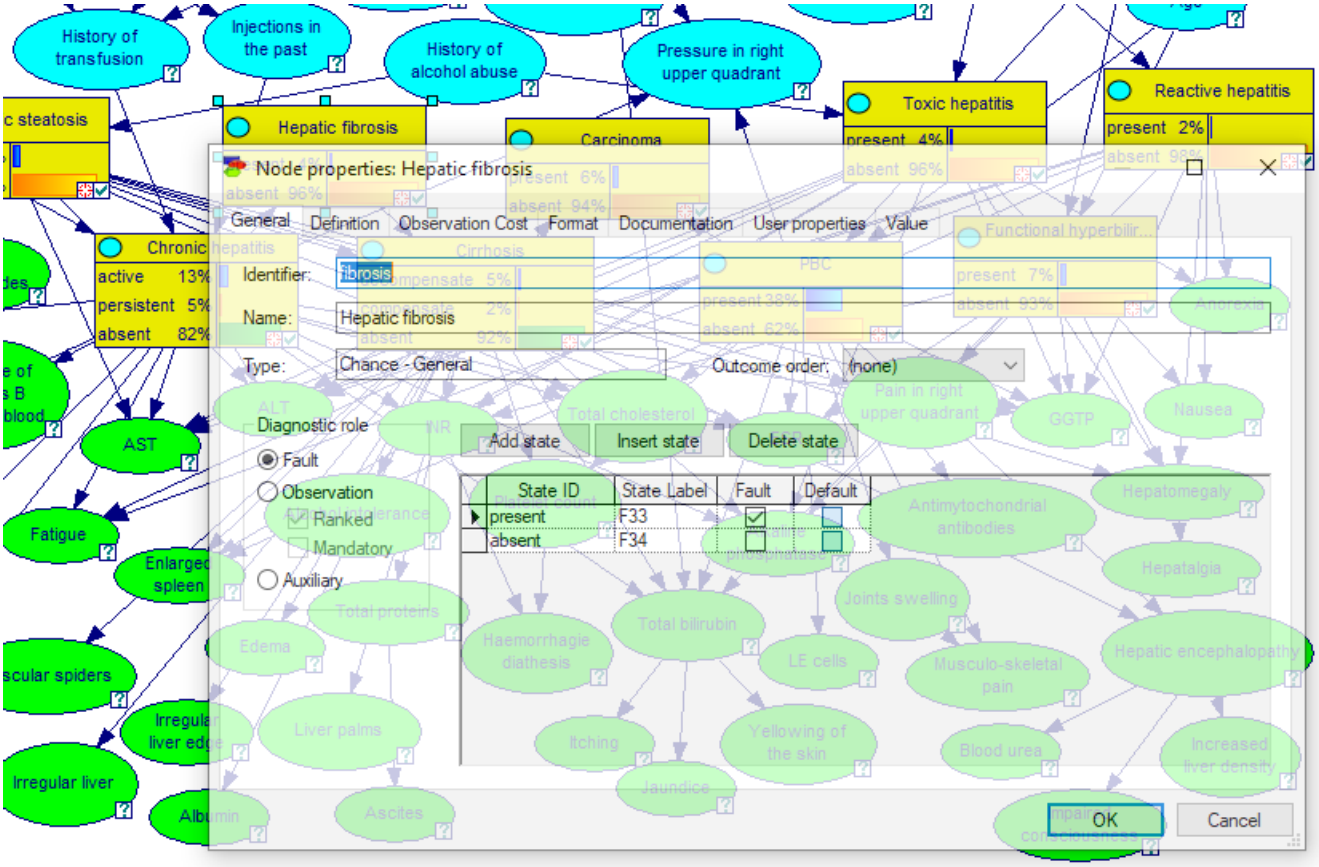
With the *AutoDiscretize* algorithm used for inference in continuous models, the *Value tab* of *Equation* nodes is identical to those of *Chance* nodes.

Transparent mode

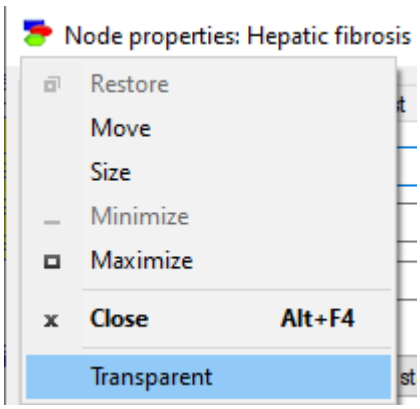
By default, the node property sheets are displayed in opaque mode, i.e., when open, they cover whatever is under them. Here is a screen shot from the Hepar II model



It is possible to make the property sheets transparent, which may be handy when navigating through a model. Transparent mode allows to see what is under the property sheets. The same model in transparent mode looks as follows



To toggle between the opaque and transparent mode, check the *Transparent* flag in the *System Menu*, available by clicking on the icon in the upper-left corner of the node property sheets



The setting holds only for the currently open property sheet and only as long as it is open.

The transparent mode may be especially useful when the property sheets are maximized, in which case it allows to see what else is on the screen and in the *Graph View* window.

5.5 Visual appearance, layout, and navigation

5.5.1 Introduction

One of the major strengths of GeNIe is its graphical user interface. GeNIe users have repeatedly and consistently praised it for being pleasant, easy to use, and intuitive. It literally cuts the model development time by orders of magnitude. Because this seems to be the bottleneck of applying decision-theoretic methods in practice, it translates directly to considerable savings. We have paid a lot of attention to detail and one of the reasons why it is so good is its physical appearance. While each of the sections of this manual contains elements of graphical user interface, which we hope the reader will experience as pleasant and intuitive, this section points out some ways in which your interaction with the program can be enhanced and made more efficient.

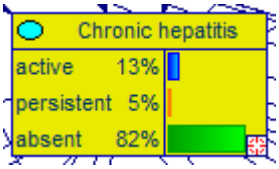
5.5.2 Viewing nodes in the Graph View

We review in this section two important aspects of viewing nodes in the *Graph View*.

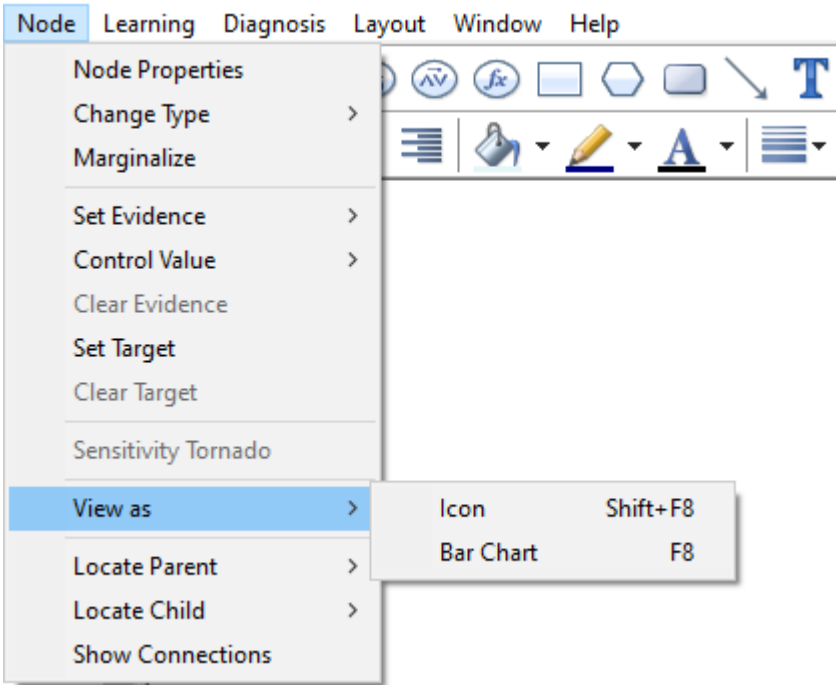
Icons and bar charts

There are two ways of viewing a node in the *Graph View*, as an icon

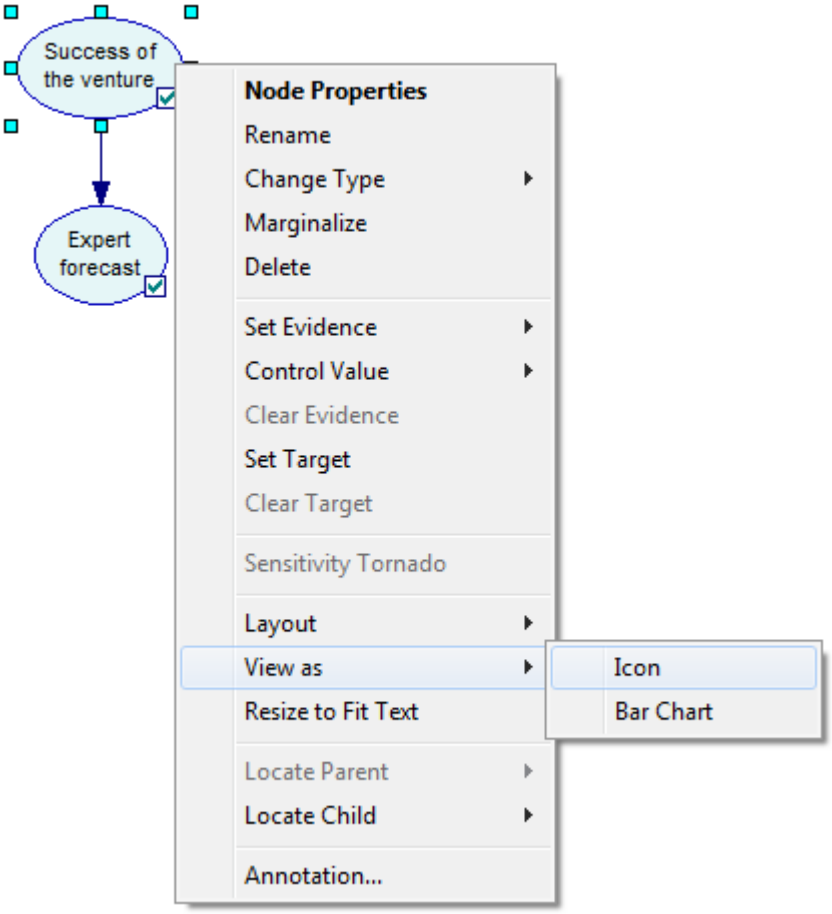
and as a bar chart, which displays graphically the node's marginal probability distribution



The advantage of seeing a node as a bar chart is that we can see at any point in time its marginal probability distribution. The disadvantage is that a bar chart takes more space on the screen and may unnecessarily draw user's attention. We advise that those nodes, whose marginal probability distributions are of interest, are viewed as bar charts. To switch between the two view, select the nodes in question and select the view on the *Node Menu*:



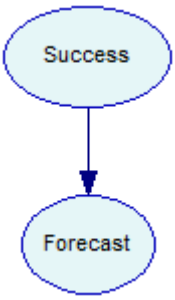
The same can be accomplished by means of the *Node Pop-up* menu:



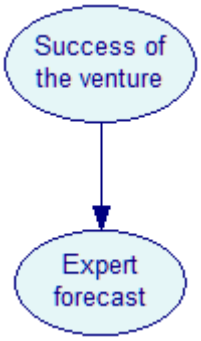
In order to apply the *Icon* or *Bar Chart* view to all nodes in the model, please invoke the *View as* choice from the *Network pop-up* menu when no nodes are selected.

Names and identifiers

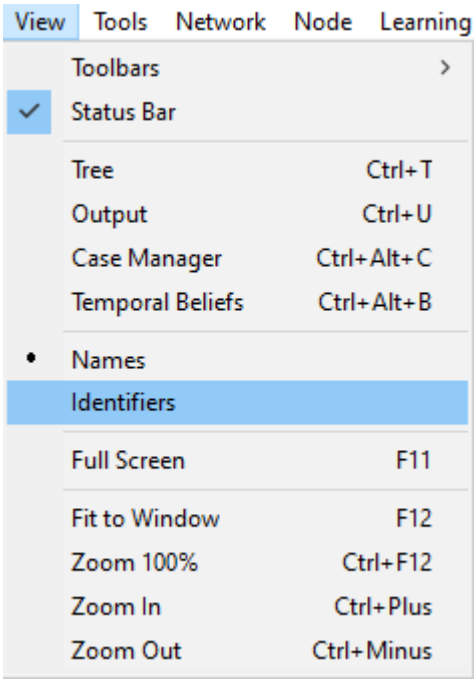
There is another important element of viewing nodes: Viewing their IDs or their names. IDs are short and play the role of variable names. GeNIe uses them for the purpose of compatibility with equation-based variables, where reference to other nodes in a node's definition has to be through a unique identifier. Identifiers have to start with a letter followed by any combination of letters, digits, and underscore characters. Letters are a-z and A-Z but also all Unicode characters above codepoint 127, which allows using characters from other alphabets than the Latin alphabet. Names are longer and have no limitations on the characters that they are composed of. A model viewed with identifiers looks cryptic.



A model viewed with names is more digestible to human users.



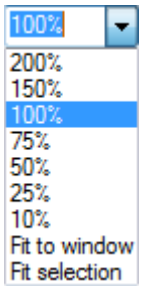
To switch between IDs and names, please use the *View Menu*



We advise that, unless there are important reasons for viewing them as identifiers, nodes be viewed by names. This is more readable for human users.

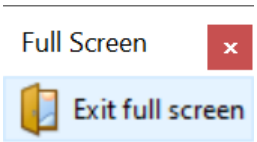
5.5.3 Zooming and full screen mode

Zoom ( ) is one of the navigational tools and allows for zooming in and out the *Graph View* by clicking with the left mouse button and the right mouse button respectively. An alternative way of zooming is through rolling the mouse wheel while holding down the *SHIFT* key. Yet another is pressing *CTRL*-+ and *CTRL*--. The effective zoom percentage is displayed on right side of the *Standard Toolbar*. Zooming can be also performed directly through the *Zoom* menu.



Zoom menu allows for choosing from a small set of predefined zoom percentage values. Two additional useful functions are *Fit to window* and *Fit selection* allow for further customization of the display. *Fit to window* selects the zoom value that makes the entire model visible in the *Graph View*. *Fit selection* will select an optimal zoom value to make the selected model elements centered and visible in the *Graph View*.

An additional functionality, full screen view is useful in case of limited screen space. To enter the full screen mode, press the *Toggle full screen* ( ) button or choose *Full Screen* from the *View Menu*. The *Graph View* will be expanded to cover the full screen (physically!). To exit the full screen mode, please press the *Close full screen* button in the following dialog that is present in the upper-right corner of the screen.



This will result in returning to the standard *Graph View*. It is also possible to remove this dialog without exiting the full screen mode, for example in case the dialog covers important parts of the screen. To close the dialog, click on the on the top-right corner of the window. Without the dialog available, you can still exit the full screen mode by pressing the *Esc* of *F11* keys.

5.5.4 Format toolbar and Layout menu

Note: All operations performed using the *Format* toolbar are applicable to the currently selected item in the *Graph View*. All new items created follow the current settings on the *Format* toolbar. *Tree View* cannot be changed using the *Format* toolbar.

The *Format* toolbar includes tools for refining the aesthetic aspects of the *Graph View*. It can be made invisible using the toggle command *Toolbar-Format* on the *View* menu. It can be also moved to any position on the screen in its free floating form. To move the toolbar from a locked position, click on the vertical bar at the left edge of the toolbar and drag it to its destination. The *Format* toolbar has buttons for changing font, color, and size of the text, and for alignment of text and various nodes in the model.

Here is the *Format* toolbar

In its free-floating form, the *Format* toolbar appears as follows

Font properties buttons

allow for selecting the font, its size, and appearance (Bold or Italic).

Text justification tools

allow for specification of the text alignment within text boxes, notes, and nodes.

Color tools

allow for setting the interior color of node and submodel icons, line color, and text color. This is similar to color selection in the *Format* tab of *Node Properties* sheet and in *Options*.

Line width pop-up tool is used to select the width of the boundary lines of the node/submodel icons.

Node/Submodel Align buttons

become active when at least two nodes are selected in the *Graph View*. They allow for aligning the drawing of nodes to each other or to the grid. Their functionality is self-explanatory and essentially similar to the functionality of most drawing software packages.

Show gridlines button () turns the background grid on and off.

Align to grid button () aligns the center points of the selected objects to the nearest intersection of the grid lines. At least one object must be selected for this option to be active.

Align left and *Align right* align the leftmost and rightmost points of the selected objects, respectively.

Align top and *Align bottom* have analogical functions.

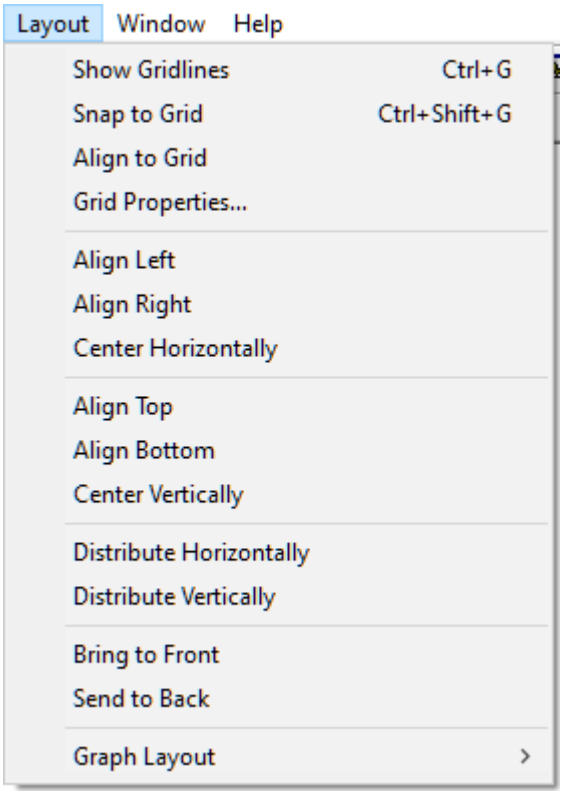
Center vertically and *Center horizontally* align the object centers.

Distribute horizontally and *Distribute vertically* distribute evenly the selected objects respectively horizontally and vertically between the position of the farthest nodes. Both tools will be active only if at least three objects are selected.

Bring to front brings the selected object to the front so that none of its parts is covered by other objects.

Send to back sends the selected object to the back so that none of its parts covers any other objects.

The functionality of *Node/Submodel Align* buttons is repeated in the *Layout* menu



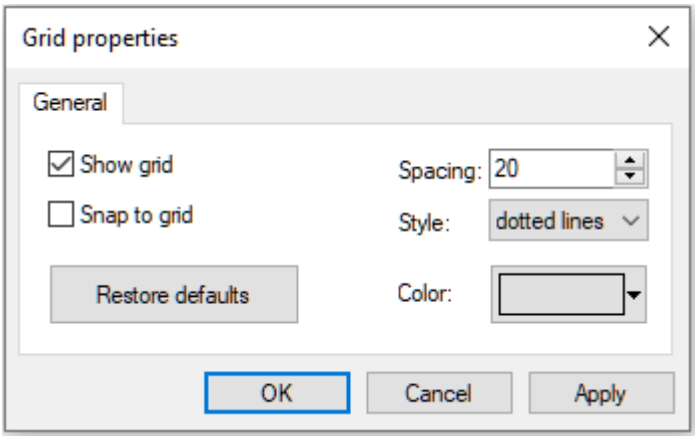
There are several additional functions offered by the *Layout* menu that are not offered by the *Format* toolbar:

Show Gridlines (shortcut *CTRL+G*) toggles display of the grid, i.e., a mesh of perpendicular horizontal and vertical lines in the background of the *Graph View*. The grid is useful in drawing and aligning nodes.

Snap to Grid (shortcut *CTRL+SHIFT+G*), when enabled, makes all new nodes that are created in the *Graph View* aligned to the grid. (If you want to align existing nodes to the grid, select them and use the *Align to Grid* tool, described above.) Dragging of nodes, with the *Snap to Grid* option on, will not be smooth, as nodes will jump from one grid line to another.

Grid Properties submenu

Grid Properties... opens up the *Grid Properties* dialog shown below



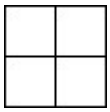
Show grid checkbox has a same function as the *Show Gridlines* option in the *Layout* menu. When checked, the grid lines are displayed in the *Graph View*.

Snap to grid checkbox has the same function as the *Snap to Grid* option in the *Layout* menu. When checked, all new nodes created will be automatically aligned to the grid.

Spacing defines spacing (in pixels) between the lines of the grid. A smaller value will result in a finer grid.

Style defines how the grid lines will be displayed:

Solid lines makes the grid lines solid.



Dotted lines makes the grid lines dotted.

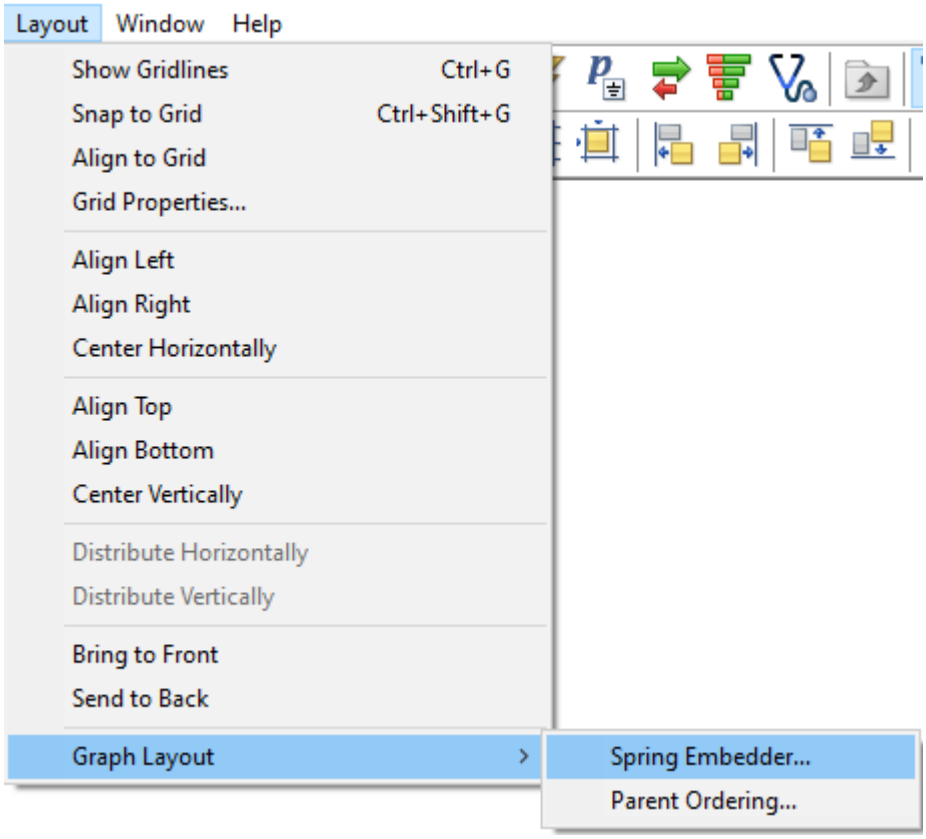
Dots makes only the intersection points of the grid lines displayed.

Color sets the color of the grid lines. You can select any color from the palette or define your own color. Grid color selection is similar to color selection for nodes. While darker colors are better when the grid style is dotted or dotted lines, we advise lighter colors for solid grid lines so that they do not clutter the *Graph View* unnecessarily.

Restore Defaults restores the grid display settings to the original factory settings.

5.5.5 Graph layout functions

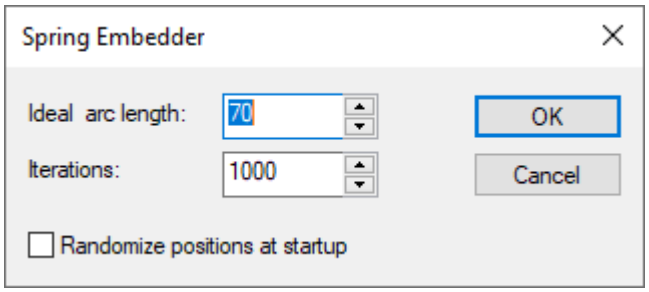
It is possible that a newly loaded network, for example learned from data or created by another program and translated by GeNIe, will have no layout information (i.e., positions of nodes on the screen). GeNIe supplies functions that arrange nodes within the *Graph View* automatically using two graph layout algorithms: Spring Embedder and Parent Ordering. Layout algorithms are computationally complex and their output may be far from perfect from the point of view of a human user. It is generally a good idea to treat their output as a starting point for manual arrangement of nodes.



Spring embedder

The *Spring Embedder* algorithm (Quinn & Breuer 1979; Eades 1984) is a more sophisticated algorithm of the two and it yields fewer arc crossings and a generally better layout of the graph. It rearranges the positions of the nodes in such a way that the nodes do not overlap each other, arc crossings are minimized, and the layout is readable for the user.

Clicking on the *Graph Layout/Spring Embedder* opens the following dialog:



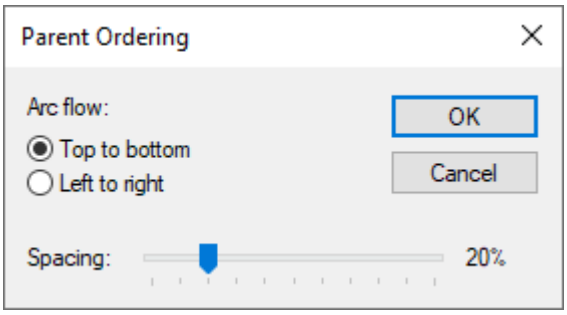
Ideal arc length specifies the ideal distance (in pixels) between two interconnected nodes. Please note that not all arcs will be of this length, the algorithm may modify this length so as to reduce overlaps.

Iterations specifies the number of iterations for the layout algorithm. Reducing this number, especially for very large networks, will translate directly to shorter execution time.

Randomize positions at startup, if checked, causes node positions to be randomized before running the layout algorithm. If it is not checked, the algorithm starts with the original node positions.

Parent ordering

Parent Ordering is a simple algorithm for graph layout that essentially places the elements of the model in top-down or left-to right order starting with the parent-less ancestors and ending with childless descendants. The *Graph Layout/Parent Ordering* option opens the following dialog:



Top to Bottom: This places the nodes in the model in the top to bottom on the graph view.

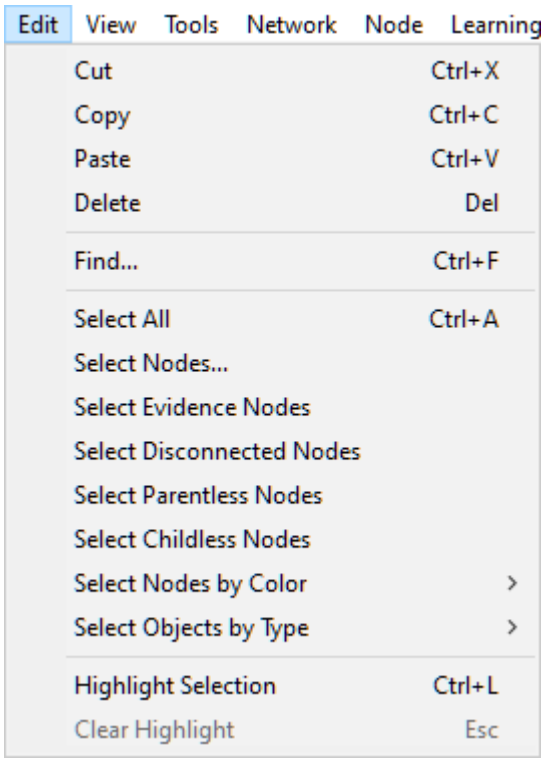
Left to Right: This places the nodes in the model from left to right on the graph view.

Spacing: This specifies the distance between nodes.

5.5.6 Selection of model elements

Often, when constructing models, we want to select their parts so as to change them as a group. The simplest way of selecting a model element is by left-clicking on them. Another common way is by selecting an area in the *Graph View* using the mouse. Everything in the area selected by the mouse will be selected. Adding new elements to the selected set can be accomplished by holding the SHIFT key when left-clicking on the mouse.

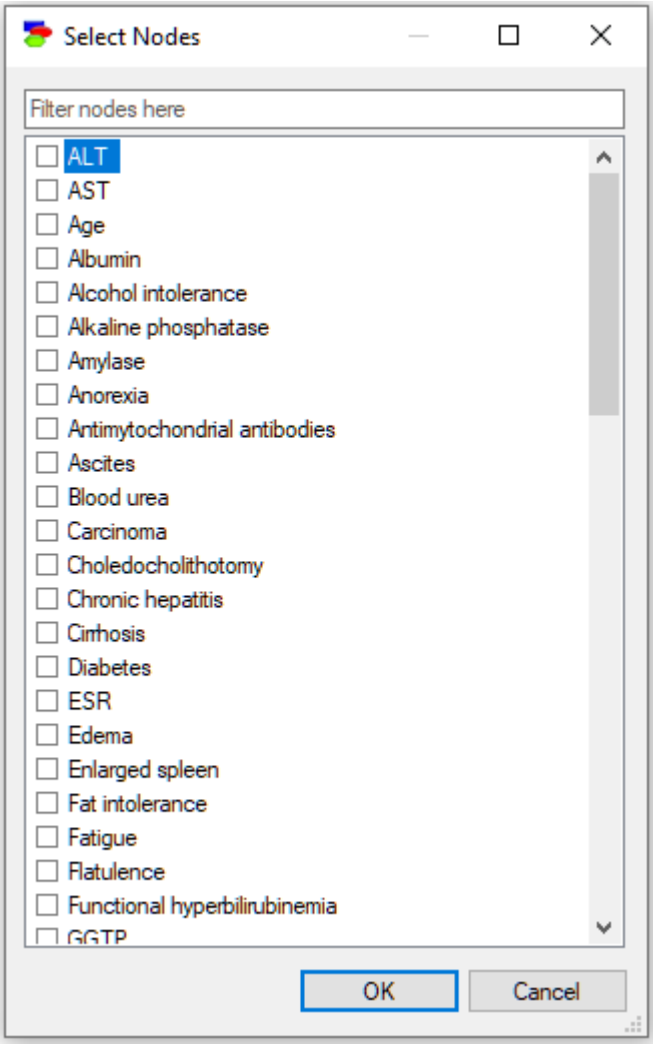
Edit Menu offers several other selection shortcuts that may make your life easier:



Select All, with a shortcut *CTRL+A* works in most views and selects all elements (for example, nodes, submodels, and all arcs between them) of the current *Graph View* window.

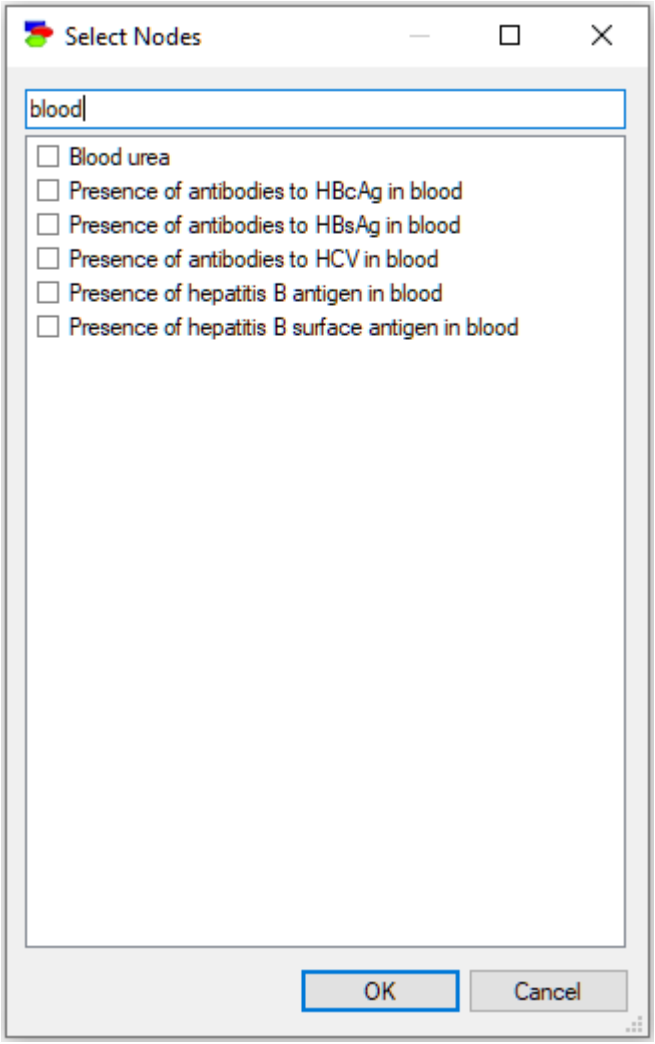
Select Evidence Nodes selects those nodes, for which the evidence has been set by the user. The function is dimmed out if no evidence is present in the model.

Select Nodes... dialog allows for sophisticated selection of nodes based on their names.

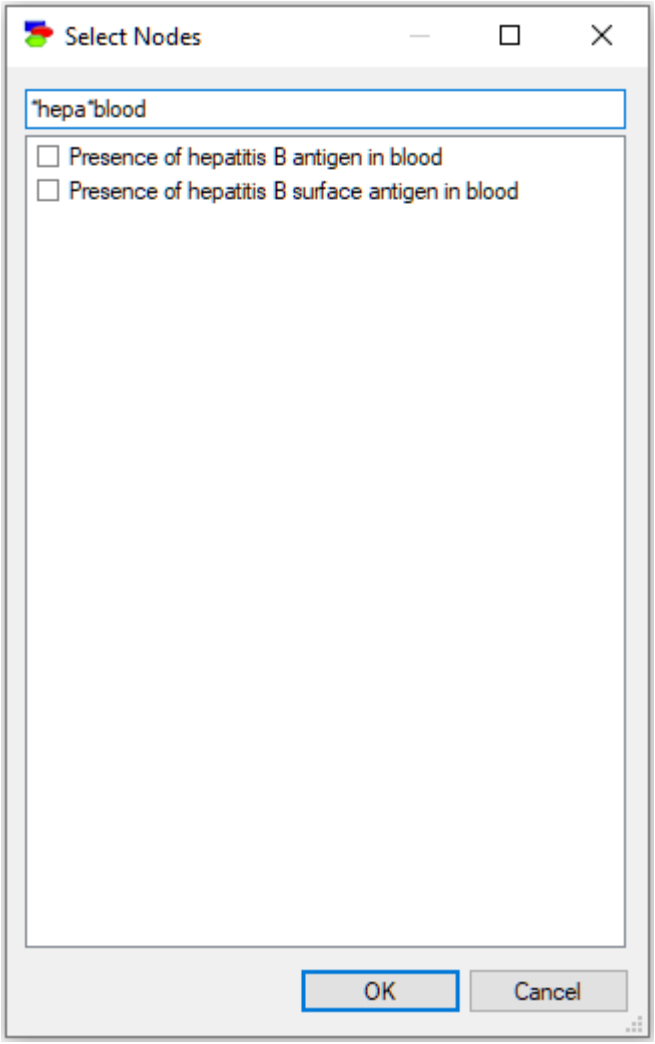


Check boxes next to variable names on the left-hand side allow us for selecting variables. All variables selected in the *Graph View* when invoking the dialog will have the check-box checked. Conversely, all variables with the check-box checked, will become selected in the *Graph View* when we exit the dialog.

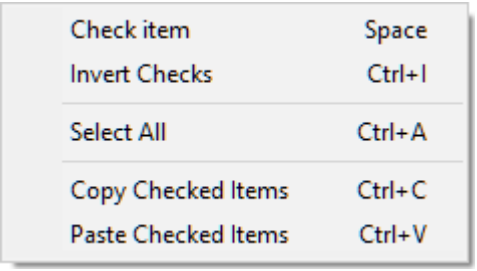
Given that a practical model may contains many variables (we have seen in our work with clients models as large as several thousand variables), locating and selecting them may be daunting. The filter field above the list of data columns serves for selecting columns. Typing anything in the filter field causes the dialog to limit the names of variables to those that match the filter. For example, typing "blood" into the above dialog will lead to the following selection:



Letter case is ignored, so a lower case letter (e.g., "a") is equivalent to an upper case letter (i.e., "A"). In addition to regular characters, the filter field interprets wildcard characters, such as an asterisk (*) or a question mark (?) similarly to Windows. Typing "*hepa*blood" will select only those variable names that contain "hepa" string and end with "blood" string:



Any of the variables visible in the list can be selected or de-selected. Selection can be made by checking the small check-box to the left of the variable name or by pressing the space key. All highlighted variables will be selected or deselected by this. Right-clicking anywhere in the window brings up the variable list context menu:



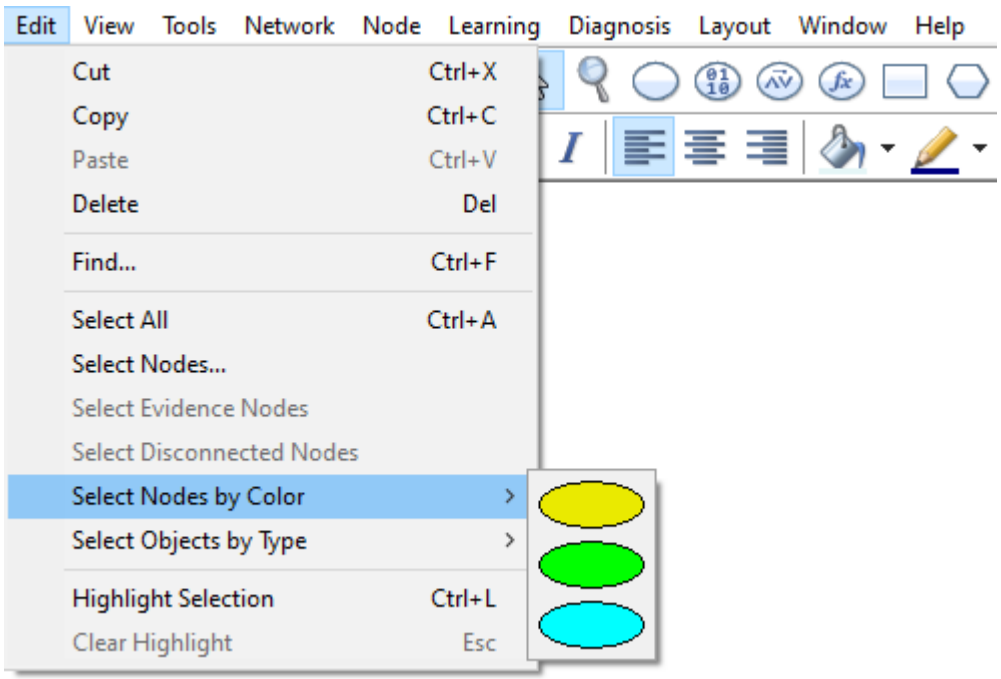
Check item (or pressing the space key) will change the value of the check-box: When it is check, it will be unchecked and when unchecked, it will be checked.

Invert Checks will invert all checks on the list.

Select All will highlight all nodes currently visible in the window. Subsequent pressing of the space key will select or deselect all of them but will have no effect on the remaining (currently invisible) model variables. There is no *Select None* choice in the menu but deselecting all nodes can be accomplished by selecting all nodes and pressing the space key once or twice (if the first pressing leads to selection).

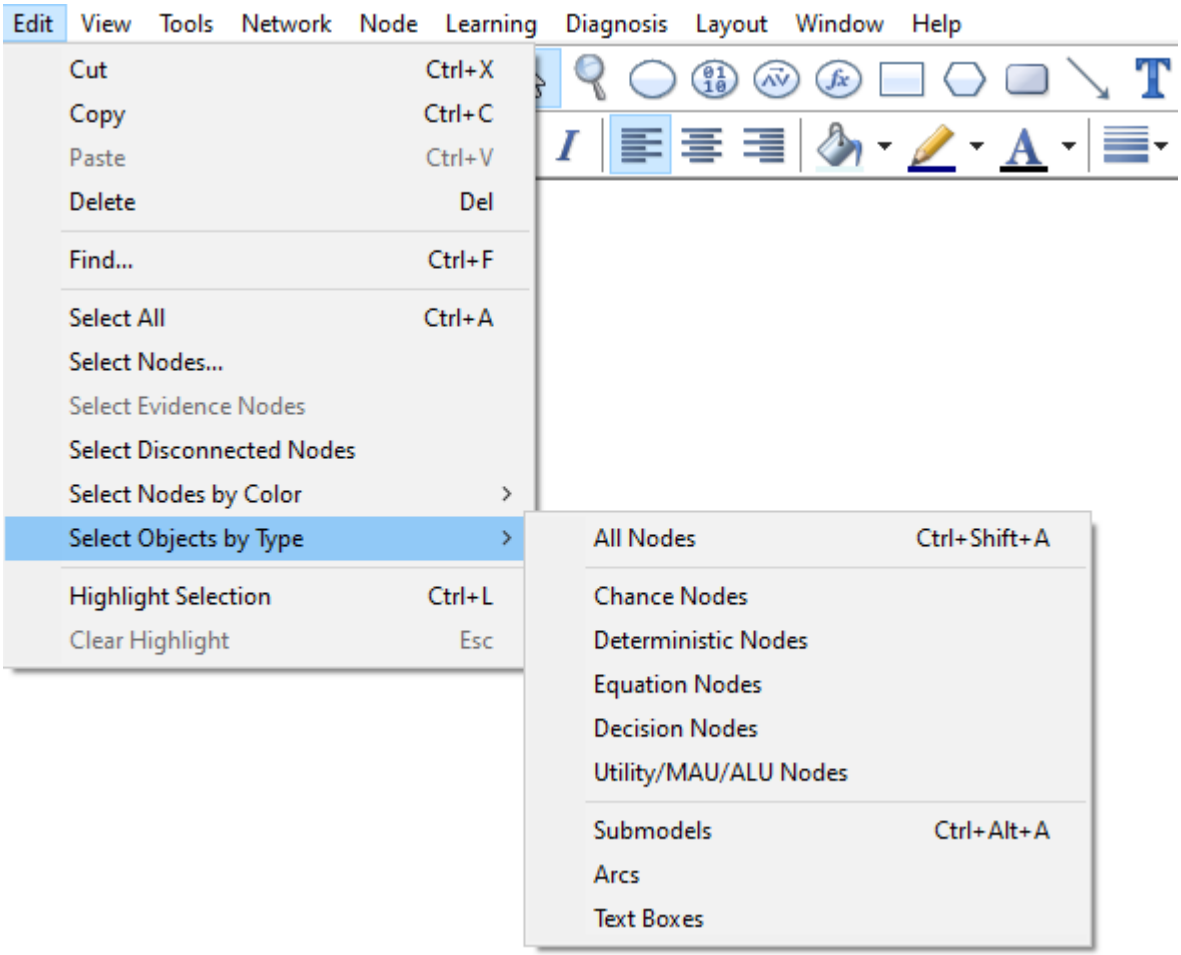
Finally, the interface allows for copying and pasting the list of selected variables. These operations disregard the filter and copy/paste nodes from/to the entire model. *Copy Checked Items* places the list of the selected nodes (text format) on the clipboard. The clipboard format is multi-line text containing the names of items in the list. If, at some later stage, we select and copy the text list outside of GeNIe and choose *Paste Checked Items*, GeNIe will select only the variables on the list. *Paste Checked Items* command will remove the current check marks, unless the *SHIFT* key is pressed when command is invoked. This functionality is very convenient in case the list of variables is large and repetitive variable selection process is laborious.

Select Nodes by Color submenu



shows all colors used in the current model and allows to select only the nodes of a given color.

Select Objects by Type submenu



offers additional ways of selecting nodes.

All Nodes, with a shortcut *CTRL-SHIFT-A*, selects all nodes in the current *Graph View*. Please note that a similar command, *Select All*, selects all objects and that includes all submodels, arcs, text boxes, etc., while *All Nodes* selects only nodes.

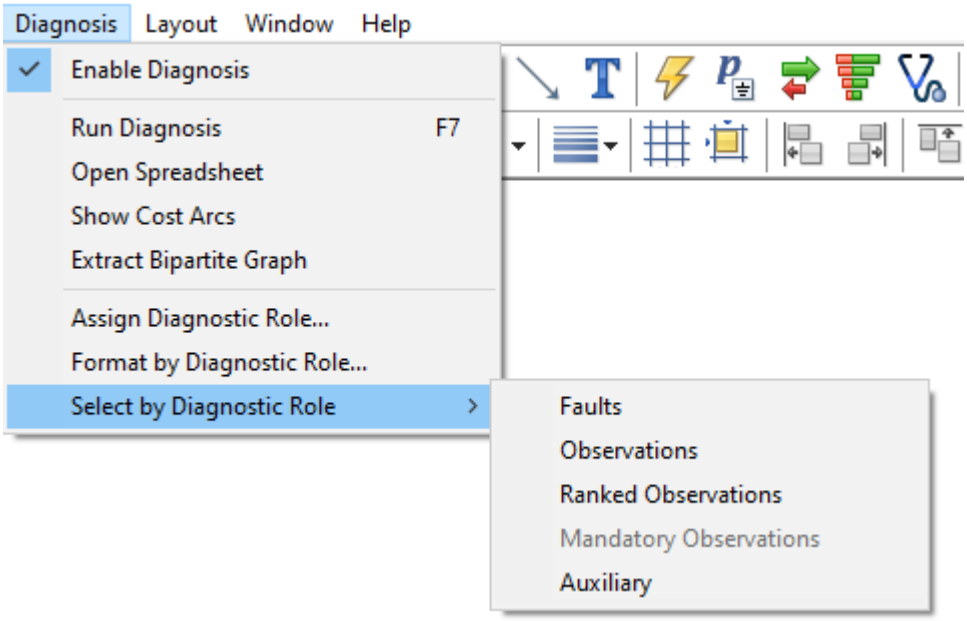
Chance Nodes, *Deterministic Nodes*, *Decision Nodes*, *Utility/MAU/ALU Nodes*, selects all nodes of the corresponding type in the current *Graph View*.

Submodels (shortcut *CTRL+ALT+A*) selects all submodels in the current *Graph View*.

Arcs and *Text Boxes* select all arcs and text boxes in the current *Graph View*, respectively.

Selection of nodes by their diagnostic type

In addition to the above ways of selecting nodes, with diagnostic extensions turned on, *Select Nodes* submenu in the *Diagnosis Menu* allows for selecting all nodes belonging to one of the three types of diagnostic nodes, *Faults*, *Observations* or *Auxiliary* nodes.



The typical application of this selection is joint editing of each type of nodes, for example coloring or displaying as bar charts.

Selection modifiers

Simple selection, such as rectangular selection and the selections by color, by type and by the diagnostic type can be modified by pressing the *SHIFT* key or *CTRL* and *SHIFT* keys simultaneously. Effectively, three selection modes are available:

- *Default selection*: When neither *SHIFT* nor *CTRL* keys are pressed, leads to selection of only objects selected. The remainder of the model elements becomes unselected.
- *Add-to election*: When the *SHIFT* key is pressed but *CTRL* key is not pressed, the objects selected by the selection operation are selected, the remaining model elements do not change their selection status.
- *Invert selection*: When both the SHIFT and the CTRL keys are pressed, the objects selected by the selection operation invert their selection status, the remaining model elements do not change their selection status.

For example, consider the *Hepar II* model. Let us select all model elements with Ctrl+A first, then go to the *Edit Menu* and *Select Nodes by Color* submenu. When keeping both *SHIFT* and *CTRL* pressed, we select yellow nodes. The selection status for yellow nodes will invert and everything except for yellow nodes will be selected.

5.5.7 Model navigation tools

GeNIe includes several simple tools that facilitate model navigation. Each of them is described in detail in other parts of this document. This section lists them in order to expose them and their purpose.

Submodels

Submodels is a construct introduced in GeNIe for the purpose of user interface. Very often, when a decision model is large, it becomes impossible to navigate through its graph - it may look like a spaghetti of nodes and arcs. Luckily, real world systems and their models tend to exhibit a hierarchical structure (Simon, 1996). There may be several variables that are strongly connected with each other and only weakly connected with the rest of the model. Such may be the case in a business model - purchasing, production, and sales may be three almost autonomous subsystems that can be connected with each other through a small number of links, their inputs and outputs. A decision maker may want to examine each of these subsystems in detail, but may also want to have a global view of the entire business without unnecessary detail. We advise to use submodels whenever a model becomes sufficiently complex.

One problem that a user will experience is navigation between submodels. To aid navigation, GeNIe allows to traverse the model by right-clicking on small triangles in those nodes that have parents or children in other submodels and locating these.

Tree View and *Graph View*

Models are typically developed, edited, and viewed in *Graph View*, which is the program's primary view. Some operations, however, may be more convenient in the *Tree View*, which offers an alternative to the *Graph View*. Nodes in the *Tree View* are listed alphabetically, so finding them may be sometimes easier than locating them on the screen. The *Tree View* shows the submodel hierarchy and allows for moving nodes between various submodels. The two views work side by side, similarly to a tree view and directory view in Windows.

Model as document

Following the idea that one of the main goals of a model is documenting the decision making process, GeNIe supports two constructs that aid documenting the model: text boxes and annotations.

Text boxes allow for adding an arbitrary text to the background of the *Graph View* window. This text may be useful as a comment explaining the details of the model.

Annotations, which are small yellow stick-it notes, which can be added to nodes, arcs, states of nodes, individual probabilities, etc., are useful for explaining function of nodes and states, or to note down just about anything the user feels is important regarding various model elements.

Text search

Find button and *Find* choice in the *Edit Menu* (described in the *Graph View* section) allows for searching through the model for a text. It searches through all text elements of the model, such as IDs, names, descriptions, annotations, text boxes, and displays a list of elements found. These elements can then be located within the model.

Visualization of strength of relationships

Intuitively, interactions between pairs of variables, denoted by directed arcs, may have different strength. It is often of interest to the modeler to visualize the strength of these interactions. GeNIe offers a functionality (described in the *Strength of influences* section) that pictures the strength of interactions by means of arc thickness. This is especially useful in the model building and testing phase. Model builders or experts can verify

whether the thickness of arrows corresponds to their intuition. If not, this offers an opportunity to modify the parameters accordingly.

Status Bar and *Output* windows

Status bar tells about problems: The *Status Bar* command displays and hides the *Status bar*. The *Status bar* is a horizontal bar located at the very bottom of the main GeNIe window. For more information on the *Status Bar*, see the *Status bar* section of GeNIe workspace.

The *Output* command displays and hides the *Output Window*. For more information, see the *Output Window* Section of GeNIe workspace.

5.6 Saving and loading models in GeNIe

5.6.1 Introduction

While GeNIe is a general purpose decision modeling environment, it has been originally written with research and teaching environments in mind. Historically, several academic and commercial environments have been developed with the purpose of decision modeling and each of these introduced its own file format. At some point, a group of researchers in the Uncertainty in Artificial Intelligence community realized that it would be beneficial for everybody if models developed using different systems could be shared. In order to facilitate sharing and exchanging models, an attempt was made to develop a standard file format, known as Bayesian Networks Interchange Format (BNIF). Unfortunately, while the core of the BNIF was developed and made available on the World Wide Web, there was not sufficient agreement as to what elements it should contain. Effectively, none of the commercial or academic software implement the standard, with a notable exception of MSBN, a Bayesian network package developed at Microsoft Corporation and made available free of charge to the scientific community. One of the reasons for the fact that a common file format has not taken off is that every software for Bayesian networks has specific model features that it wants to save and load.

An alternative approach, which we believe has been more successful, is to provide reading and writing functionality in each of the popular formats, attempting to convert as many as possible software specific model properties. This is the approach we have followed. We believe that the idea of exchanging models among various researchers is excellent and we support it by implementing several popular file formats in addition to the native GeNIe format. GeNIe is able to read and write files in the following formats: Ergo Version 1.0, most recent version of the BNIF format implemented in the Microsoft MSBN package, Hugin, Netica and KI. For backward compatibility reason, we are also supporting the format used by GeNIe in mind-1990s (DSL file format). If you are a software developer and want to add your file format to the formats supported by GeNIe, please contact us.

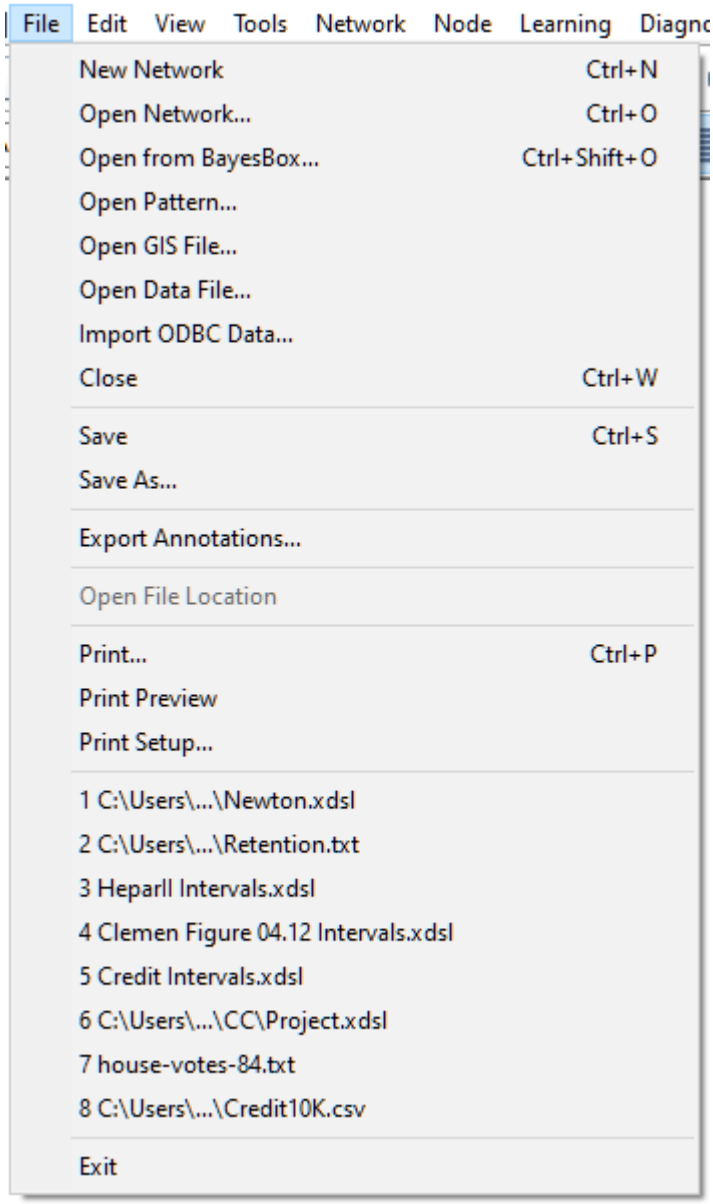
GeNIe can read and write models created by other programs. It can be thus used also as a conversion program between different formats. The user should keep in mind that various programs may have extensions and functionalities that are not supported by other programs. We would like to warn the users that conversions between various formats will, in general, lead to a loss of information, as various formats may lack elements such as submodels, node colors, node size, etc. To prevent loss of information, we recommend the users of GeNIe to use its native format, which is in almost all cases a super set of other formats.

Opening a previously saved model in GeNIe

There are three ways in which you can open a model in GeNIe,

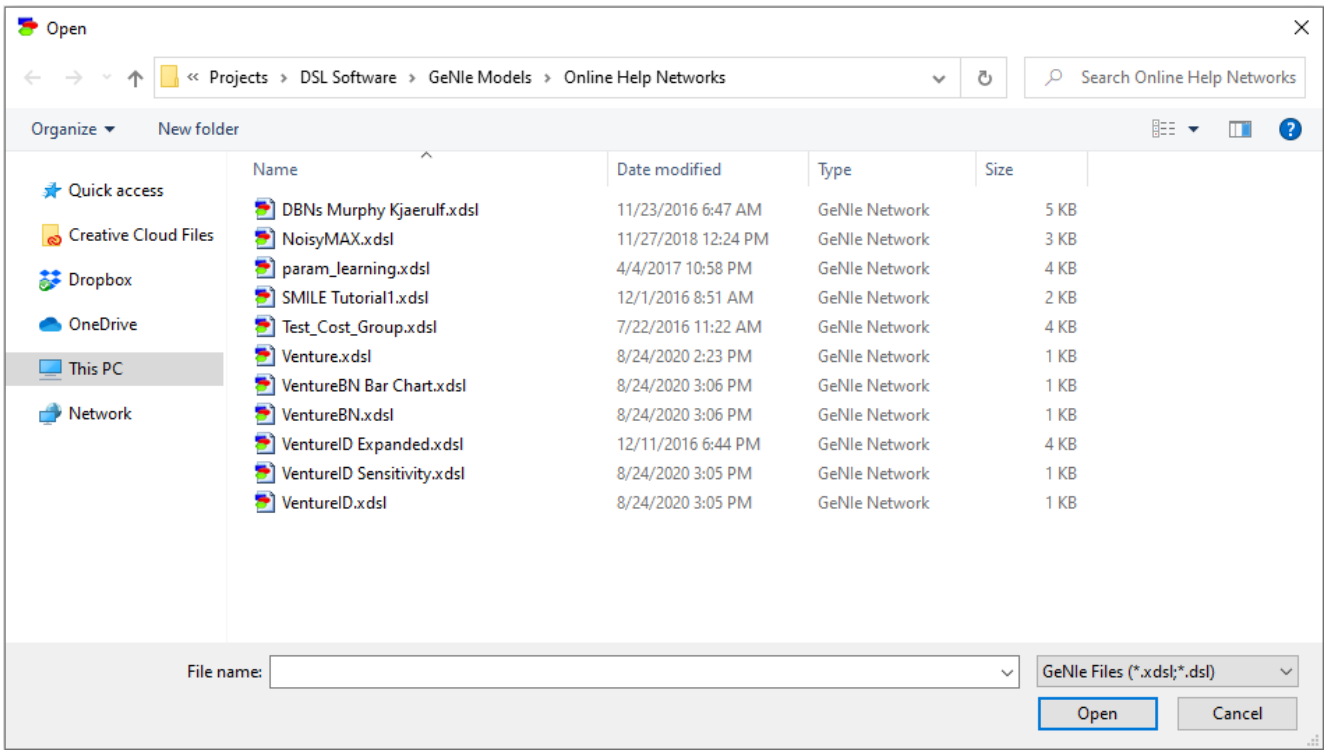
- Choose *Open Network...* from the *File Menu*
- Click on *Open* ( ) button from the *Standard Toolbar*
- Use the *CTRL+O* shortcut

Shown below is the *Open Network...* option in *File Menu*.



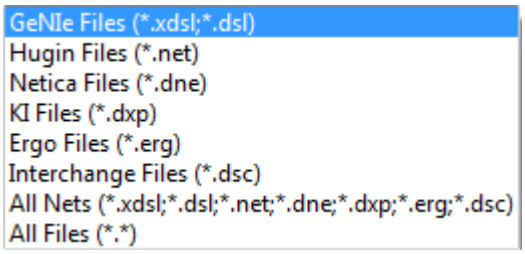
The numbered list of files at the end of the *File Menu* are the *Most Recently Used* (*MRU*) file list. You can click on any of those names to re-open the file.

Each of the three ways of invoking *File Open* dialog leads to the following:



The dialog box that appears allows you to choose a file to load.

GeNIe supports multiple file formats, and you can choose the format of your file by using the *Files types* drop down list.



GeNIe uses the XDSL file format, which is an XML-based format. In addition to the XDSL native format (*.xdsl), GeNIe is able to read a legacy file format used by GeNIe 1.0 (this was between 1995 and 2000) and also supports several external file formats.

In order to see all the network files recognized by GeNIe in the directory, choose *All Nets*. In order to see all files in the directory, choose *All Files*.

You can also possible to bypass the *File Open* dialog altogether and open a model in GeNIe by dragging and dropping a model icon into GeNIe's model window.

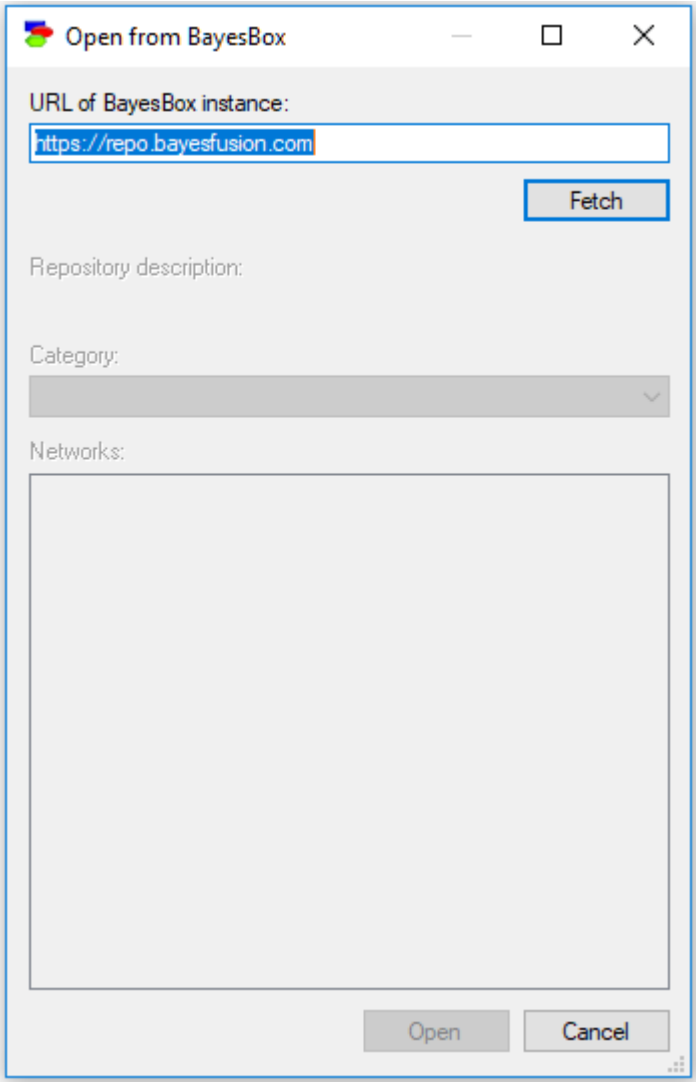
Note : You can have multiple files open at the same time.

Opening a GeNIe model from a BayesBox-powered model repository

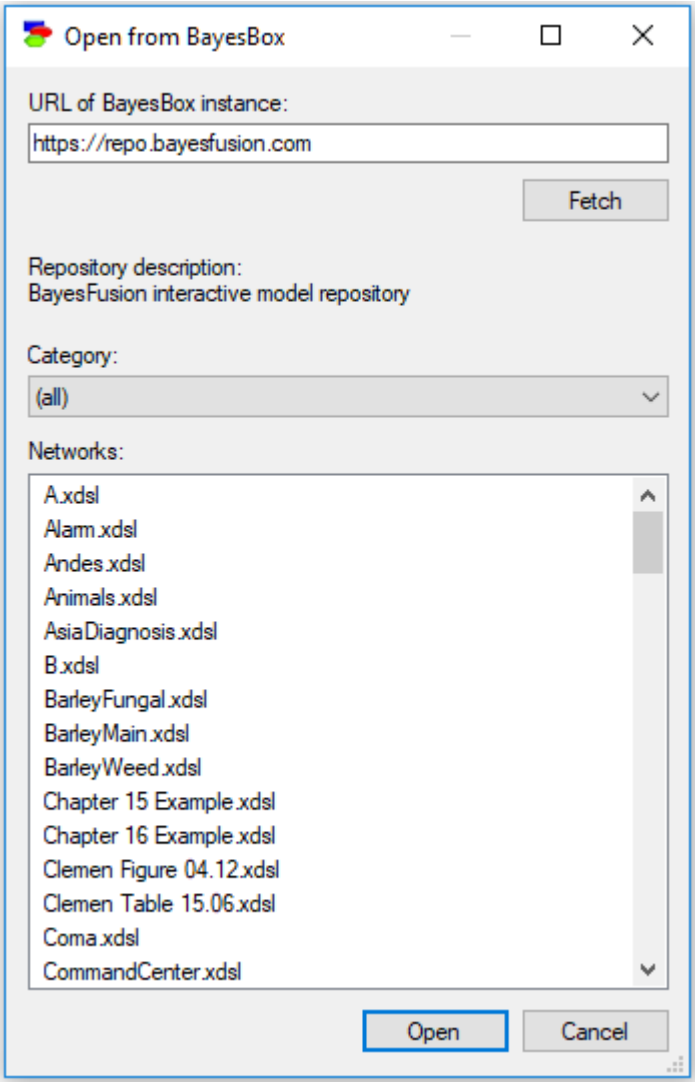
GeNIe supports loading models from BayesBox-powered repositories, which are web-based repositories, possibly local and internal to user's organization. There are three ways in which you can open a repository model in GeNIe,

- Choose *Open from BayesBox...* from the *File Menu*
- Use the *CTRL*+SHIFT+*O* shortcut
- Press the *Open network from BayesBox* ( ) button

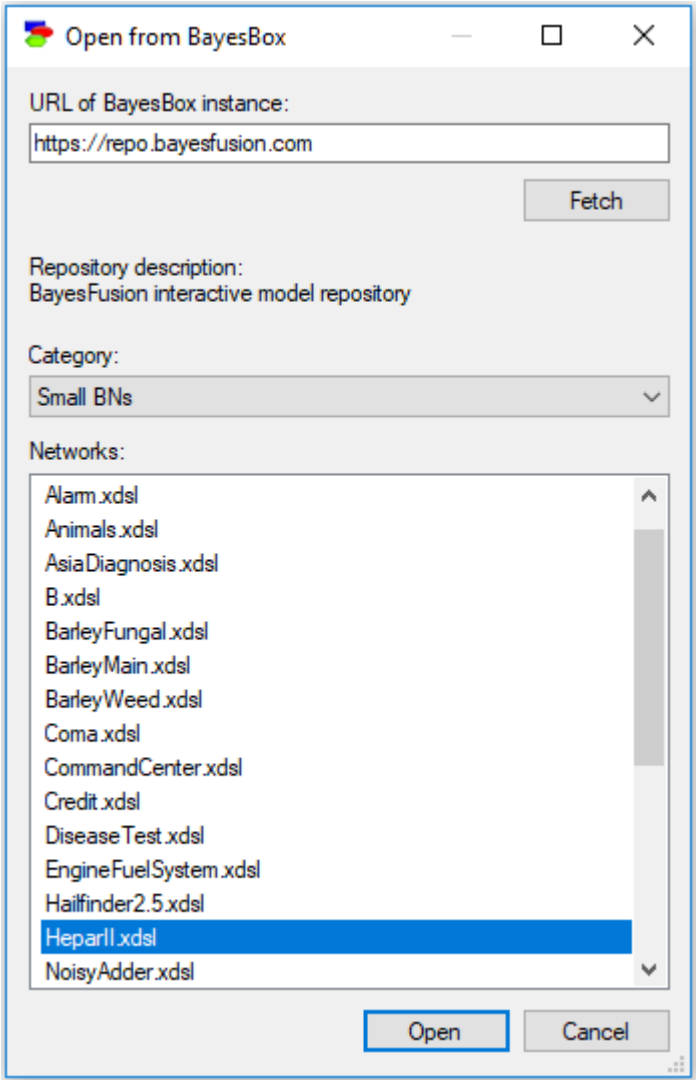
Each of the three actions opens the *Open from BayesBox* dialog:



The default BayesBox instance is BayesFusion's model repository but any web address can be entered here. GeNIe remembers the last successful connection to a BayesBox instance, which is convenient for those users who rely on their internal BayesBoxes. Fetch allows to see models and categories present in the chosen BayesBox instance:



It is possible to traverse BayesBox's directory structure by clicking on the *Category* pop-up menu:



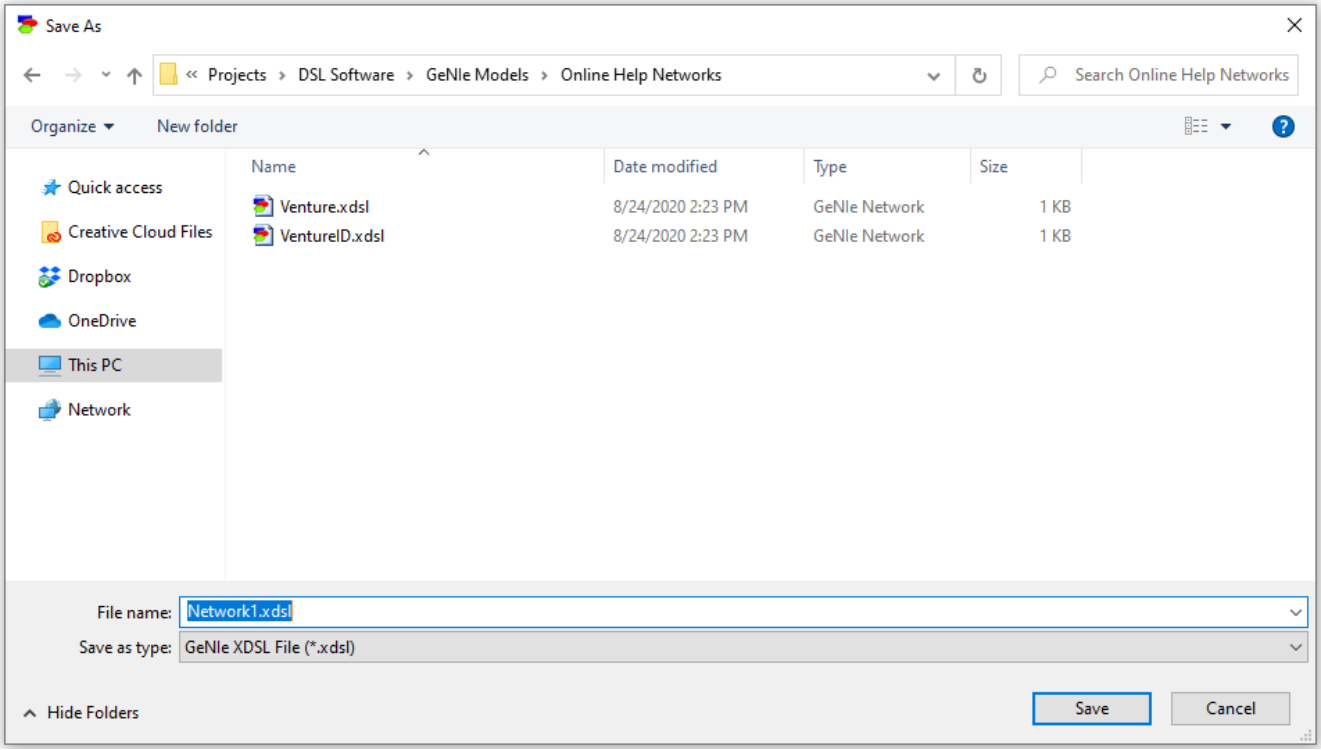
Selecting a model from the list (in this case, the *HeparII* model) fetches the model from the current BayesBox instance and opens it in GeNIe.

Saving a model in GeNIe

There are three ways in which you can save a file in GeNIe:

- Choose *Save or Save As* from the *File Menu*
- Click on Save ( ) button from the *Standard Toolbar*
- Use the *CTRL+S* (*Save*) shortcut

The difference between *Save* and *Save As* is that *Save As* lets you store the file under a different name, so that you can keep the original model file intact. *Save* will store the changes in the original file. However, if you are working on a new file, then Save As is the only option and *Save* converts to *Save As*.

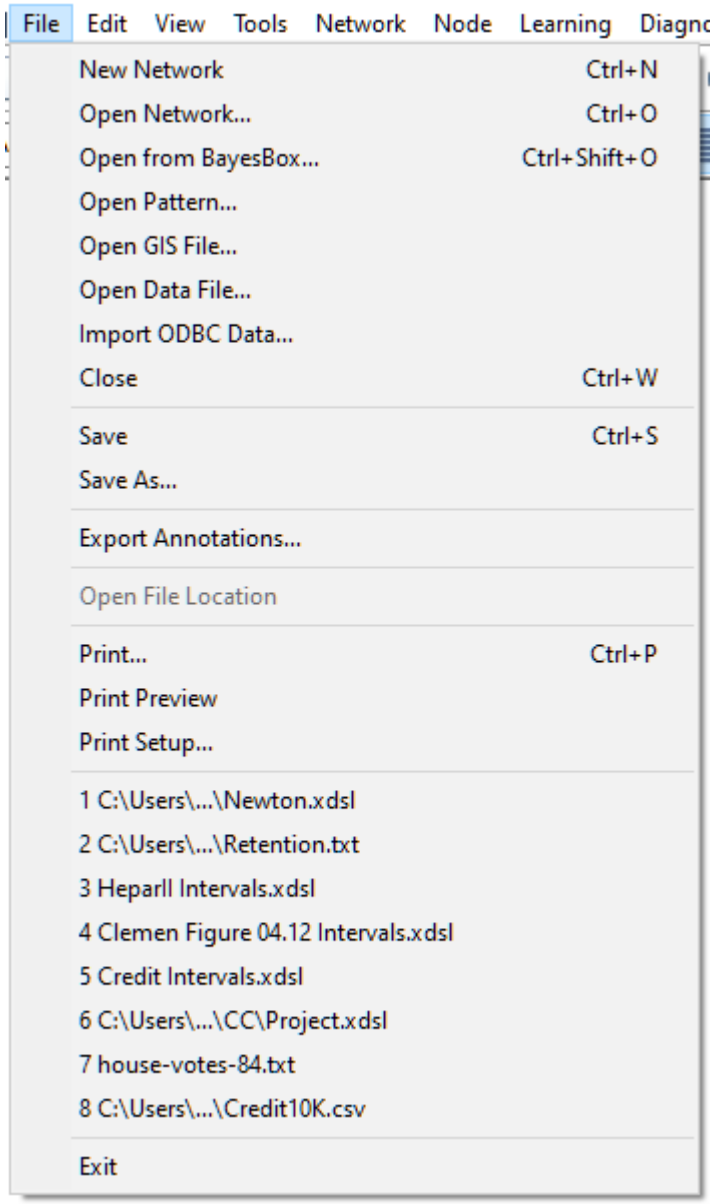


GeNIe can save in any of the file formats listed in the *Loading files in GeNIe* section above. These can be selected from the *Save as type* drop down list.

However only saving in the native GeNIe format (*.xdsl) will guarantee that all GeNIe-specific features are saved. There may be loss of information while saving in the other formats.

5.6.2 File menu

The *File* menu offers the following commands:



New Network starts a new GeNIe model and opens it in a new Graph View window. This command can be also invoked by pressing the New ( ) tool on the *Standard Toolbar* or using the *CTRL+N* shortcut.

Open Network opens an existing model that has been saved previously on the disk. This command can be also invoked by pressing the Open ( ) tool on the *Standard Toolbar* or using the *CTRL+O* shortcut.

Open from BayesBox opens an existing model from a BayesBox-powered model repository. This command can be also invoked by using the *CTRL*+SHIFT+*O* shortcut.

Save saves the currently opened model using the current file name and file format. This command can be also invoked by pressing the Save ( ) tool on the *Standard Toolbar* or using the *CTRL+S* shortcut.

Save As saves the currently opened document to a newly specified file name in a possibly newly specified format. If the document is new (i.e., if it has never before been saved), this command is equivalent to the *Save As* command.

The *Open Network*, *Open from BayesBox* and *Save/Save As* commands are discussed in detail in the *introduction* to this section.

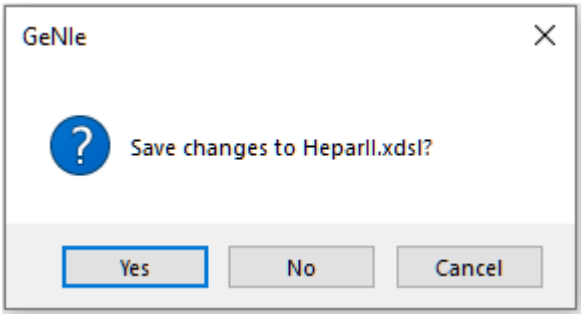
Open GIS File... opens an existing map file that has been stored in the disk for the purpose of geo-processing (described in the *Geo-processing* section).

Open Data File... opens an existing data file that has been stored in the disk.

Import ODBC Data... opens an existing database file that has been stored in the disk.

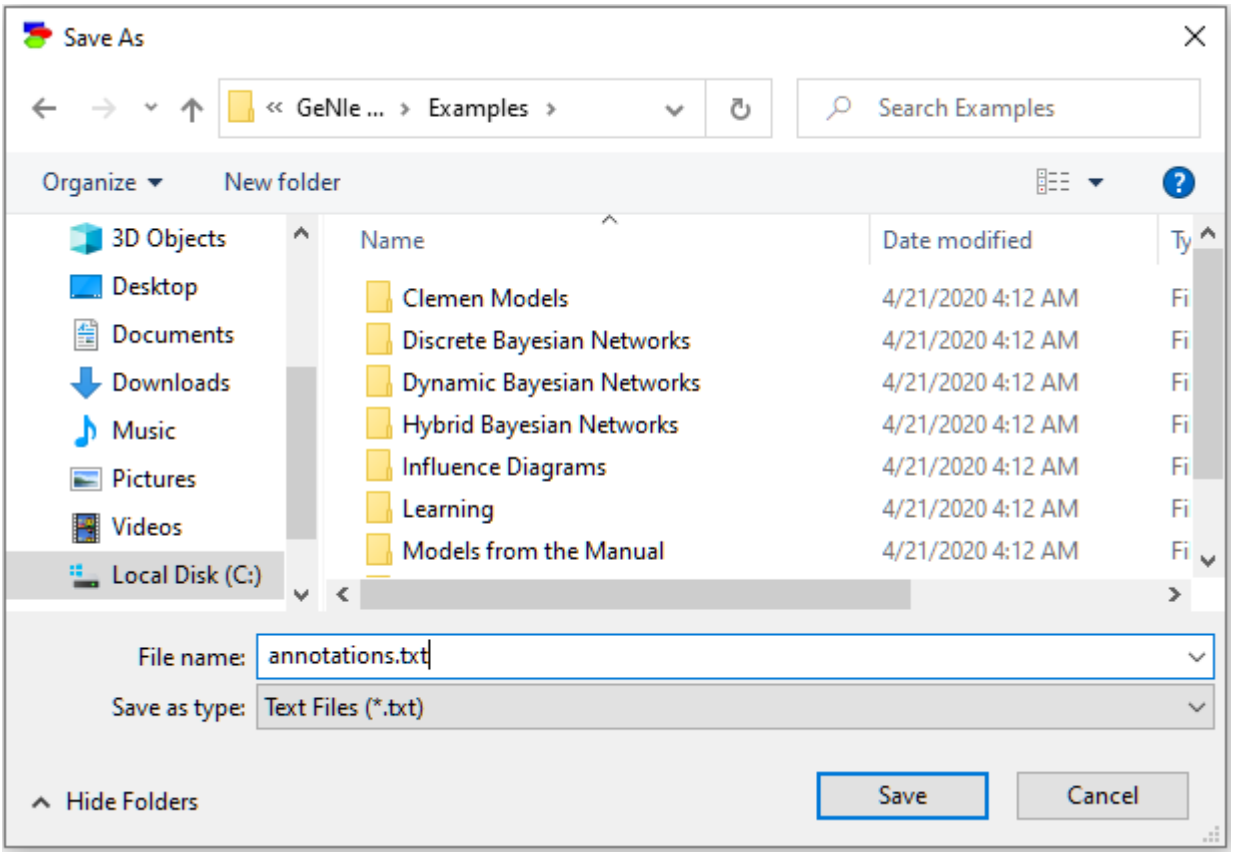
Open Data File... and *Import ODBC Data...* commands are discussed in detail in *Accessing data* section.

Close closes all windows of the current model. If there are any changes to the currently opened model that have not been saved, GeNIe will warn you using the following dialog box

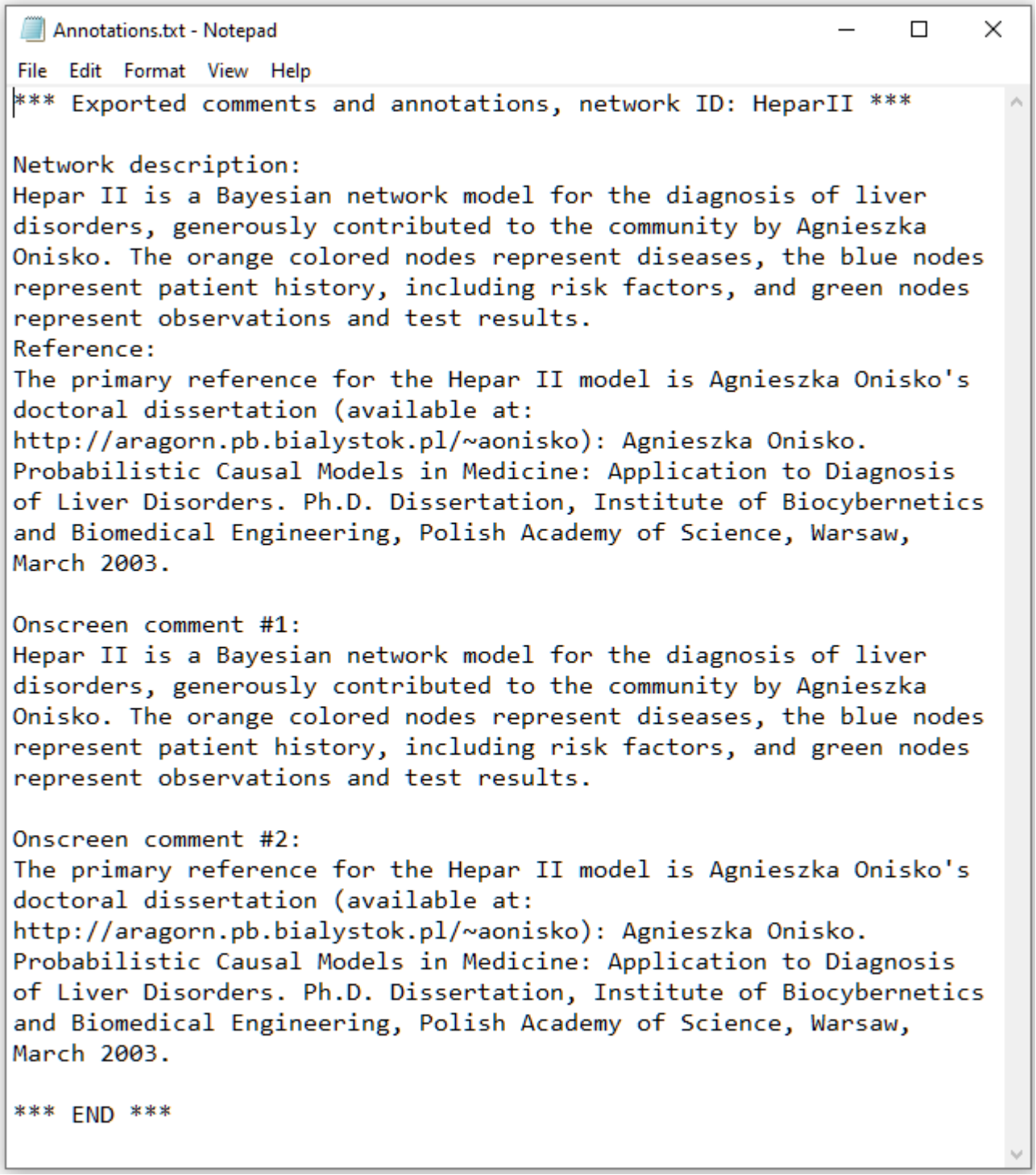


If you want to save the changes that you have made since you have last saved the model, click the *Yes* button or press *Enter*. If you want to discard them, click the *No* button. If you have second thoughts about exiting GeNIe, click *Cancel*.

Export Annotations... is a simple way of extracting all verbal descriptions and comments from the model and saving them in a single text file.



The annotations extracted from the Hepar II network will look as follows in Notepad:

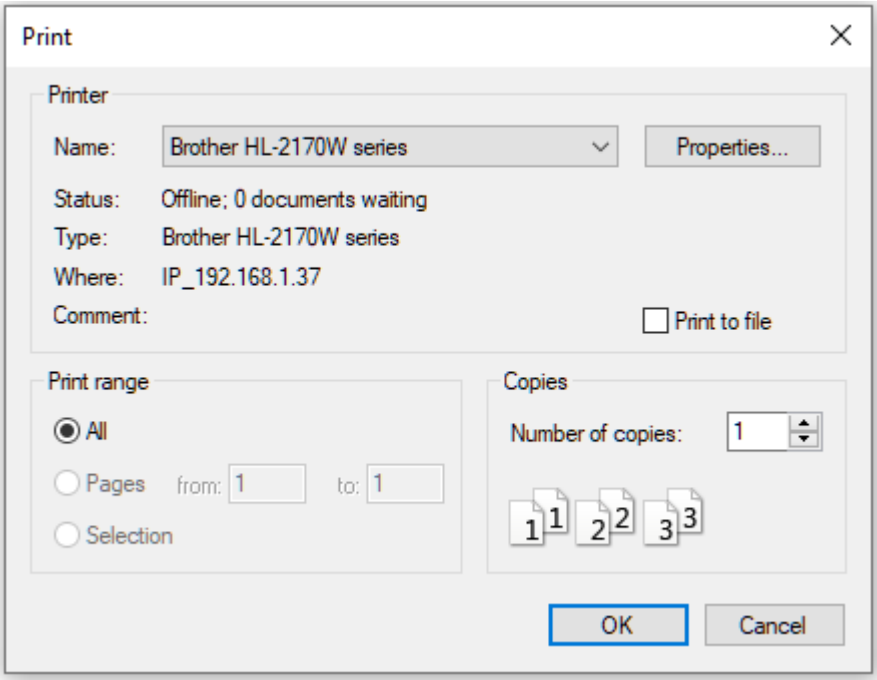


If there are any annotations or texts in the model, they will be included in the text file.

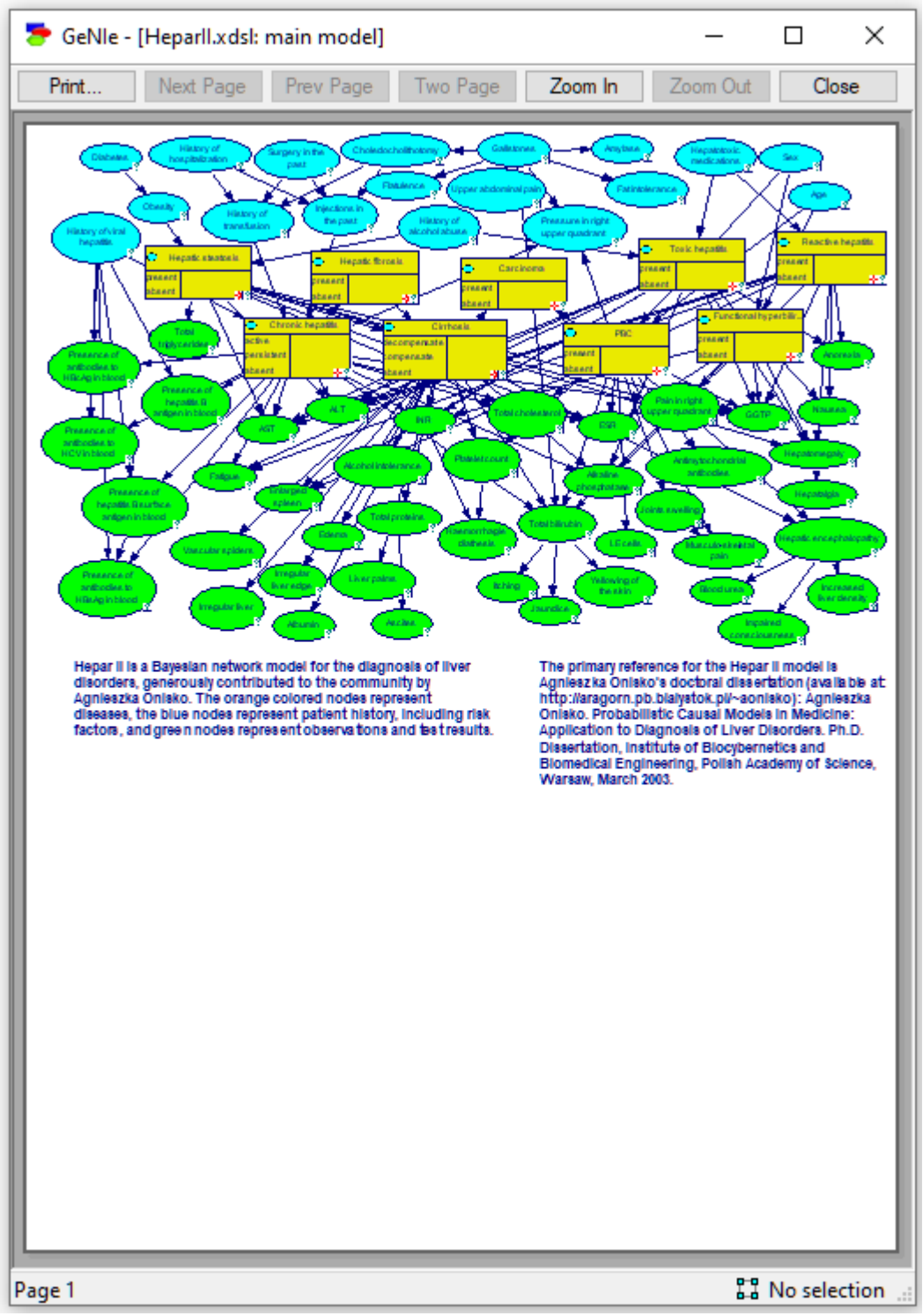
Open File Location opens the directory where the current model comes from.

Print... prints the current *Graph View* window. This command can be also invoked by pressing the *Print* ( ) tool from the *Standard Toolbar* or using the *CTRL+P* shortcut.

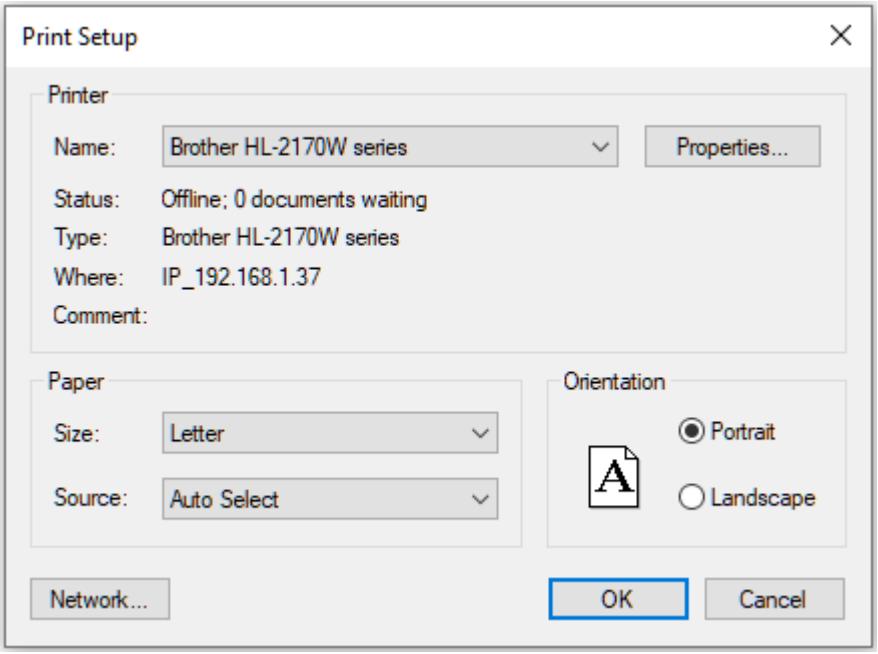
The following dialog box allows you to modify some of the printing options, such as choose the printer, the range of pages to be printed, the number of copies to be printed, and other printer properties (through *Properties*... button).



Print Preview displays the content of the current *Graph view* window on the screen as it would appear when printed. When you choose this command, the main window is replaced with a print preview window in which one or two pages will be displayed in their printing format. The print preview toolbar offers you options to view either one or two pages at a time; move back and forth through the document; zoom in and out of pages, and initiate the print job (bypassing the *Print* command).



Print Setup... opens a dialog that allows for selecting the printer and printer and its properties, including the paper size, source, and orientation. This command presents a *Print Setup* dialog box, where you specify the printer and its connection. The appearance of the dialog box below may vary depending upon your system configuration.



GeNIe displays the names and disk locations of the most recently used models. The number of these can be set in *Program options*. You can load them by choosing their names from the menu, bypassing the *Open* command.

Exit ends your GeNIe session and exits GeNIe.

5.6.3 XDSL file format

The XML Schema for GeNIe's native XDSL file format can be found at the following location: https://support.bayesfusion.com/docs/.

5.6.4 DSL file format

For backward compatibility reason, we are also supporting an old format used by GeNIe in mind-1990s (we call it the DSL file format).

Here is an abbreviated BNF specification of the DSL file format:

```
<file> ::= <net>
<net> ::= net <id> { [<netstatement>;]* };
<netstatement> ::= <netfield> | <node>
<netfield> ::= HEADER = { [<headerstatement>;]* } |
CREATION = { [<creationstatement>;]* } |
NUMSAMPLES = <integer>
<node> ::= node <id> { [<nodestatement>;]* }
<nodestatement> ::= <nodefield>
<nodefield> ::= TYPE = <id> |
HEADER = { [<headerstatement>;]* } |
```

```
PARENTS = <identifierlist> |
DEFINITION = { [<definitionstatement>;]* }
<headerstatement> ::= ID = <id> |
NAME = <string> |
COMMENT = <string>
<creationstatement> ::= CREATOR = <string> |
CREATED = <string> |
MODIFIED = <string>
<definitionstatement> ::= NAMESTATES = <identifierlist> |
PROBABILITIES = <doublelist> |
NAMECHOICES = <identifierlist> |
RESULTINGSTATES = <identifierlist> |
UTILITIES = <doublelist> |
WEIGHTS = <doublelist>
<identifierlist> ::= ( [<id>] [,<id>]* )
<doublelist> ::= ( [<real>] [,<real>]* )
<integerlist> ::= ( [<integer>] [,<integer>]* )
<boolean> ::= TRUE | FALSE
```

Identifiers (<id>), strings (<string>), numbers (<real> and <integer>), and comments follow the syntax of C++. Identifiers, in particular, have to start with a letter followed by any sequence of letters, numbers, and the underscore character (_). Letters are a-z and A-Z but also all Unicode characters above codepoint 127, which allows using characters from other alphabets than the Latin alphabet. Control characters inside strings are preceded by the backslash character (\). Comments are of three types: (1) two characters // start a comment, which terminates at the and of the line on which they occur, (2) the characters /* start a non-nesting comment terminated with the characters */, and (3) the characters /# start a nesting comment terminated with the characters #/.

The content of matrices is written as a flat list of doubles (<doublelist>), listed in the order of columns, i.e., the fastest changing index is that of the current variable, then the last parent, then the one before last, etc. The first parent supplies the slowest changing index.

5.6.5 Ergo file format

Ergo file format was originally implemented by Noetic, Inc. in their implementation of a Bayesian network development environment, known as Ergo 1.0. It was quickly and in a somewhat ad-hoc manner embraced by various researchers in the Uncertainty in Artificial Intelligence (UAI) community because of its simplicity. Several important models developed in the first years of existence of the field of UAI were developed using Ergo format. Noetic, Inc., the developers and marketers of Ergo, have since changed their file format (we have implemented the format defined in Ergo version 1.02; we approached Noetic, Inc., for a specification of the new format but have received no response) and they seem to no longer support their original format. This original, simple format has still survived in terms of useful Bayesian network models. The file extension, after Noetic, Inc., is *.erg.

The format supports only *Chance* nodes. Only node identifiers, state names, conditional probability tables, and locations of the node centers are saved. You will lose all other information. There is no description of the format available on-line but you should be able to figure it out by looking at some simple models. Here is the content of the Ergo file for the Venture BN example used throughout this document:

2

```
2       3
0
1       1

/*      Probabilities  */
2
0.2     0.8
6
0.4     0.4     0.2     0.1     0.3     0.6

/*      Names   */
Success Forecast

/*      Labels  */
Success Failure
Good    Moderate        Poor

/*      Centers */
98      159
98      253
```

The first line in the file states how many nodes the model contains (in this case, 2). This is followed by the number of states of each of the node (2 and 3). The next lines state the parents of each of the nodes (the first node has 0 parents and the second node has 1 parent, node 1).

The *Probabilities* section lists the contents of the conditional probability tables (CPTs). The number that precedes each table is the number of parameters in each table.

Finally, the *Names* and *Labels* contain the node IDs and state IDs. Centers are coordinates of the centers of each of the nodes.

5.6.6 Netica file format

This is an implementation of the format used by Norsys Inc. in their program Netica (we have implemented file format defined by Netica Version 1.06). The file extension, after Norsys, is *.dne. You can find detailed information about Netica file format at Norsys, Inc.'s WWW pages.

5.6.7 BN interchange format

This format is an attempt to design a common format for graphical probabilistic models. The format has not been established as a standard and our implementation is a good faith implementation of what has been agreed upon. Our implementation allows for reading and writing Bayesian networks files written by the package supplied by Microsoft Corporation and known as MSBN (we have implemented file format defined by MSBN Version 1.0.1.7). The file extension, after Microsoft Inc, is *.dsc. You can find information about the BNIF file format at the Microsoft Research MSBN WWW pages.

5.6.8 Hugin file format

This is an implementation of the format used by Hugin A.G. in their program Hugin (we have implemented the file format defined in Hugin Version 3.1.1). The file extension, after Hugin A.G., is *.net. You can find detailed information about Hugin file format at Hugin A.G.'s WWW pages.

5.6.9 KI file format

This is an implementation of the format used by Knowledge Industries, Inc., in their program DXpress (we have implemented the file format defined in DExpress 3.1). The file extension, after Knowledge Industries, Inc., is *.dxp. We wish we could point our readers to Knowledge Industries, Inc., website but it seems that it has gone off-line.

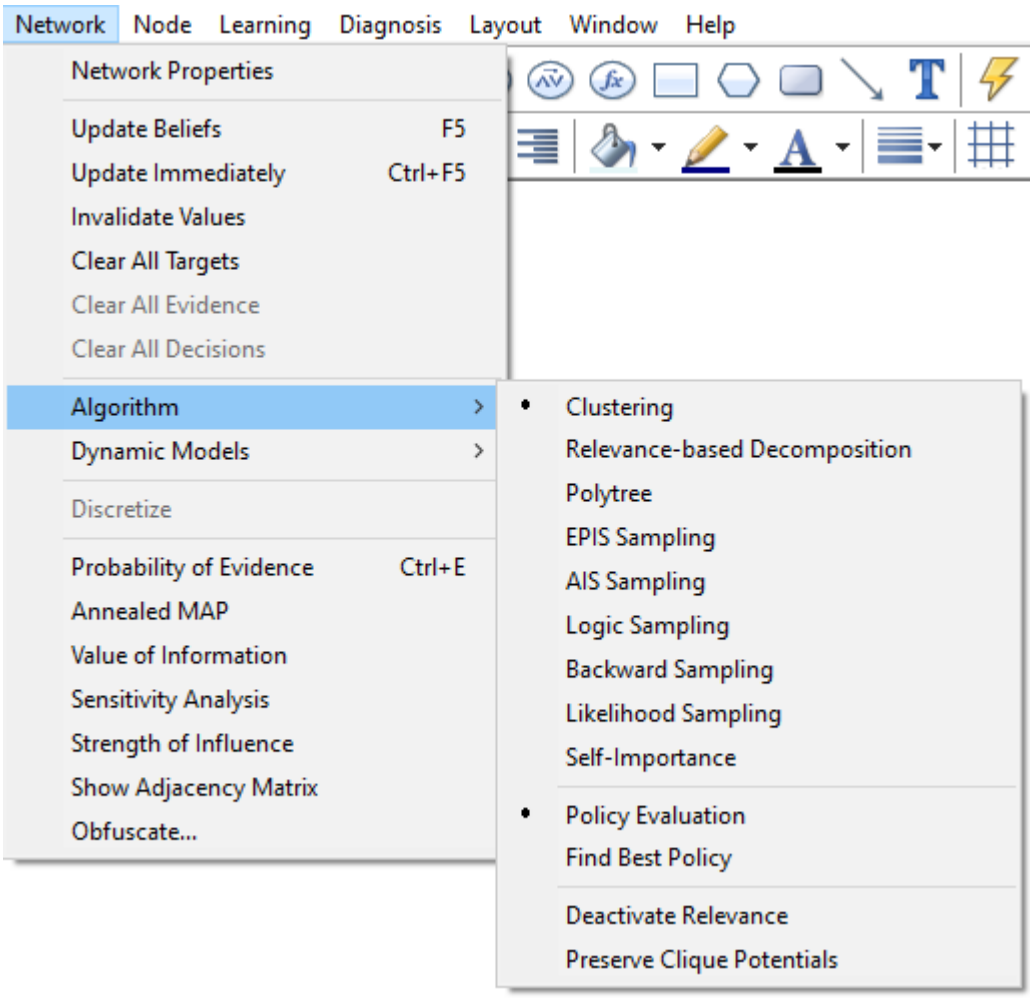
5.7 Inference algorithms

5.7.1 Introduction

GeNIe originates from a research and teaching environment and, as such, has seen the development and use of a variety of algorithms. The user can choose which algorithm should be used for updating by selecting an algorithm from the list displayed in the *Network Menu*. SMILE and GeNIe implement several popular Bayesian networks inference algorithms, including the clustering algorithm and several stochastic sampling algorithms. There are also two influence diagrams algorithms: policy evaluation and finding the best policy. The motivation for maintaining a pool of algorithms has been historically twofold. Firstly, both GeNIe and SMILE were originally used in research and teaching environments and the algorithms were used for benchmarking and comparative studies. Secondly, even though the clustering algorithm (default in GeNIe) is quite possibly the fastest implementation of this algorithm in existence and the fastest exact algorithm in GeNIe, there are networks for which the memory requirements or the updating time may be not acceptable. In these cases, the user may decide to sacrifice some precision and choose an approximate algorithm. Sampling algorithms are, roughly speaking, based on a statistical technique known as Monte Carlo simulation, in which the model is run through individual trials involving deterministic scenarios. The final result is based on the number of times that individual scenarios were selected in the simulation.

The algorithm pool is under continuous development and improvement. We will list references to literature describing individual algorithms when covering the algorithms. For readers interested in an overview paper on algorithms, we recommend (Huang & Darwiche 1996) and (Henrion 1990). An reasonable overview of stochastic sampling algorithms can be found in (Shachter & Peot 1990) or (Yuan & Druzdzel, 2005).

It is up to GeNIe user (or, in case of SMILE, up to the application programmer) to call the algorithm of his or her choice. Both GeNIe and SMILE use the concept of the default algorithm. In GeNIe, the default algorithm can be chosen using the *Network Menu*:



The menu allows for choice of the default algorithm (marked with a bullet). Whenever updating takes place, the current default algorithm will be executed. We would like to note that the influence diagram algorithms are based on the Bayesian network algorithms and the choice of a Bayesian network algorithm will have impact on the precision and performance of the influence diagram algorithm.

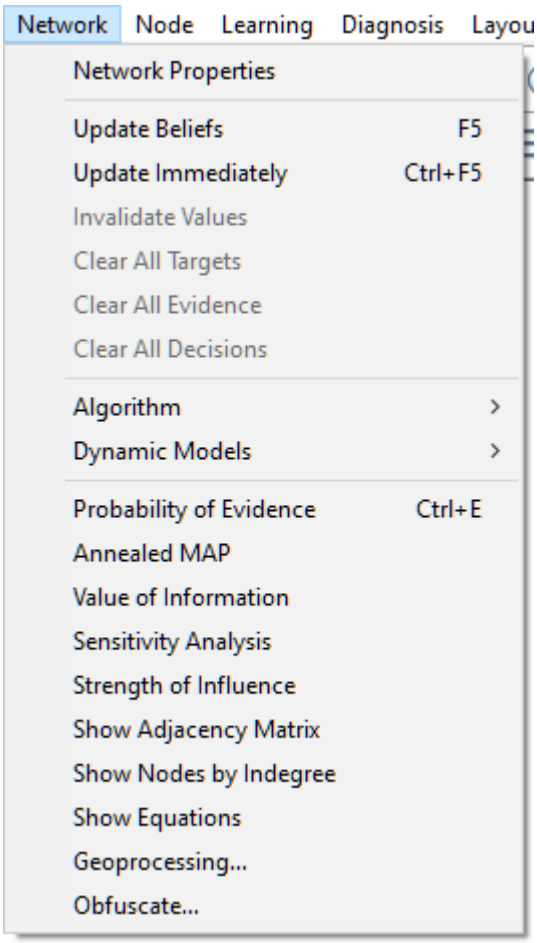
We would like to stress that the choice of algorithms has one important practical application: When a network cannot be updated using the fastest known exact algorithm, the *Clustering* algorithm (GeNIe's default), the user can still update the model using an approximate algorithm. Tradeoff between precision and computation time can be controlled by selecting the number of samples. Our recommendation for such cases is the *EPIS Sampling* algorithm, quite likely the most efficient stochastic sampling algorithm in existence.

With all functionality that needs randomness, GeNIe relies on the JKISS random number generator with a period of $2^{127} = 1.7 \times 10^{38}$. This is sufficient for all practical purposes. Non-zero seeds are hashed before they are used to initialize the generator. Zero seed causes the generator to be initialized with the value based on the system clock and, therefore, makes the output of the algorithm using the pseudo-random generator differ across runs.

5.7.2 Immediate and lazy evaluation

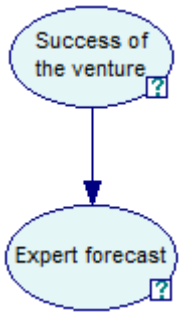
GeNIe operates in two modes: immediate and lazy updating. In lazy updating mode, every time we modify the model, enter evidence, or control the value of a node, we need to explicitly invoke an algorithm to update the values and to view the result of the changes. In immediate updating mode, GeNIe automatically updates the model as soon as any change, observation, or control is made to the model. Hence you do not need to update the model explicitly.

To switch between the two modes, choose *Update Immediately* from the *Network Menu*.



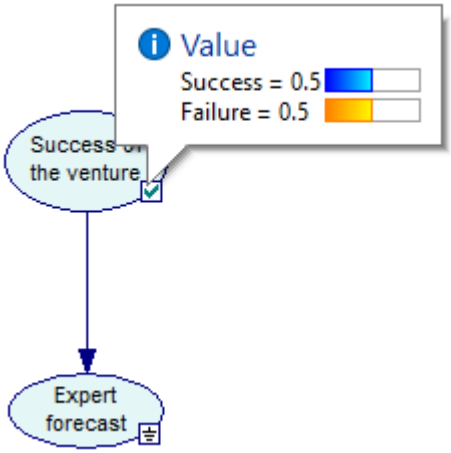
The lazy updating mode, is useful when the model is in its development stage or when it is so large that updating takes an annoyingly long time. In this case, you can run an evaluation algorithm to update the model either by choosing *Update* from the *Network Menu* or, alternatively, by pressing the *Update* ( ) button.

GeNIe uses status icons for the nodes that indicate whether a node has been updated or not. Nodes that are not updated have a small question mark ( ) icon on them, like in the picture below:



The values of such nodes cannot be examined, as they are not available, and GeNIe will not display the *Value* tab in the *Node Properties Sheet* for these nodes.

When the values are up to date, GeNIe displays a small check (☑) icon on the node, like in the picture below:



Values for such nodes can be displayed by hovering over them (like in the picture above) or by opening their *Value* tab.

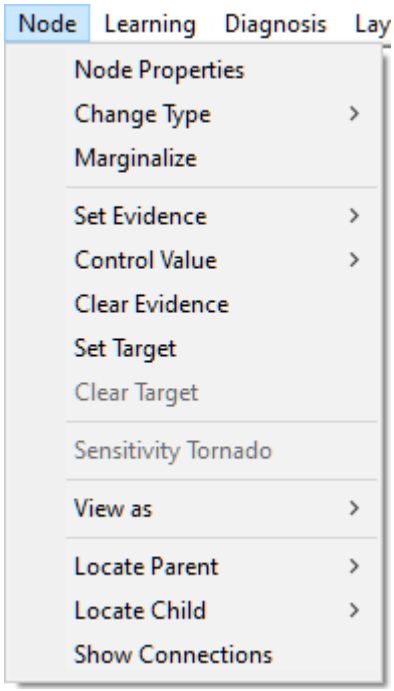
5.7.3 Relevance reasoning

A feature that is unique to GeNIe among all Bayesian network software that we are aware of is relevance reasoning. Very often in a decision support system, only a small number of variables need updating, either because they are up to date or because they are of no interest to the user. For example, we might want to know the posterior probability distributions over diseases captured in a model but not in the probability distributions over outcomes of unobserved risk factors, symptoms, or test results. When the model used by the system is large, the amount of computation to update all variables may be prohibitive, while potentially unnecessary. Focusing inference on those nodes that we are interested in, can save a lot of computation. Reasoning in GeNIe and the underlying SMILE is always preceded by a pre-processing step that explores structural and numerical properties of the model to determine what part of the network is needed to perform computation.

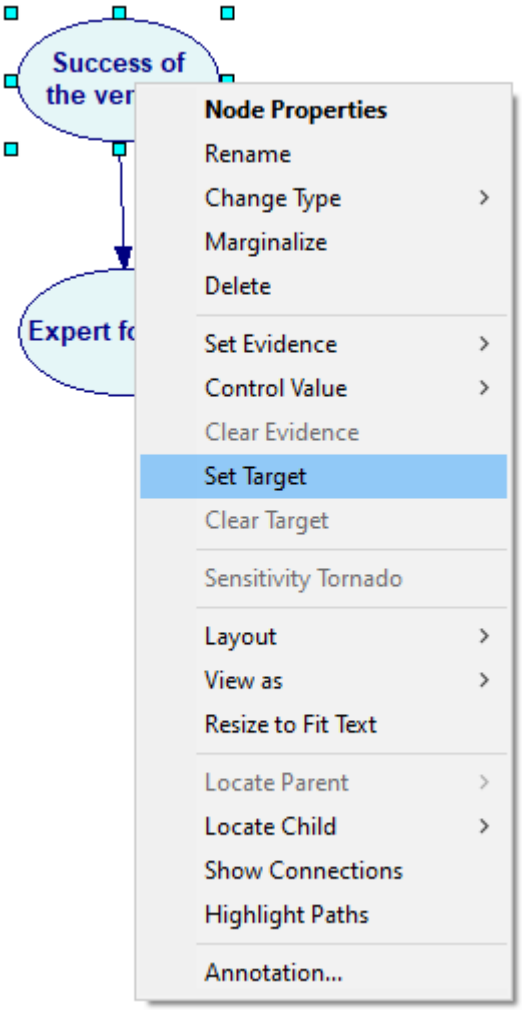
GeNIe keeps track which variables are up to date and which are not and marks them by the *Invalid* (?) *node status icon*. It also allows the model builder to designate those variables that are of interest to the user as targets.

Target nodes, marked by the *Target* ( ) *node status icon*) are always guaranteed to be updated by the program during its updating procedure. Other nodes, i.e., nodes that are not designated as targets, may be updated or not, depending on the internals of the algorithm used, but are not guaranteed to be updated. Relevance reasoning is triggered in GeNIe by marking some of the nodes as *Targets*. If there is at least one target in a model, GeNIe guarantees that all targets will be updated by its reasoning algorithms but does not give any guarantees with respect to any other nodes. When no nodes are designated as targets, GeNIe assumes that all variables in the model are of interest to the user, i.e., all of them are targets.

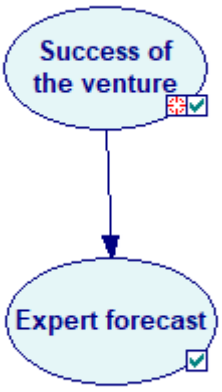
To set a node to be a target, select it and then choose *Set Target* from the *Node Menu*.



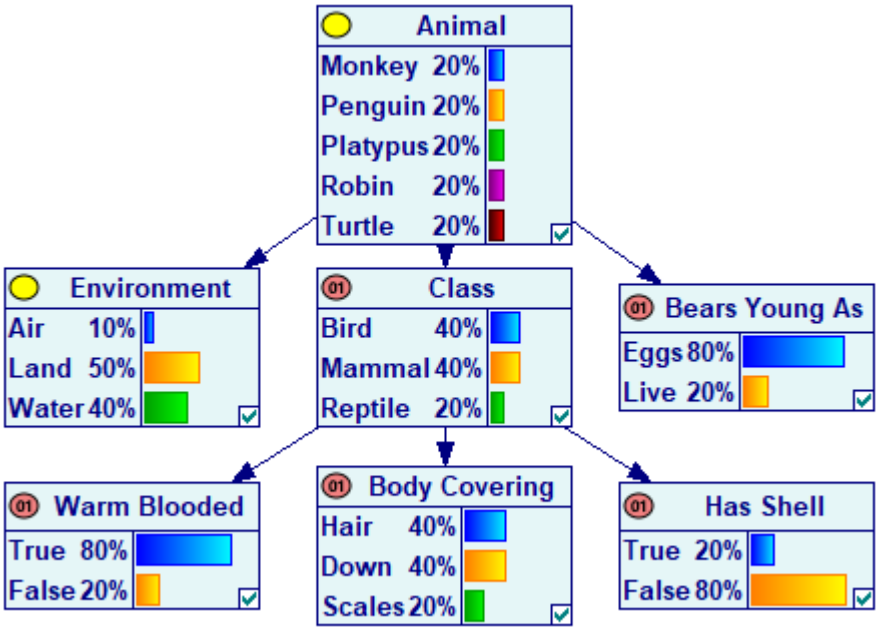
Alternatively, right-click on the chosen node and choose *Set target* from the its pop-up menu:



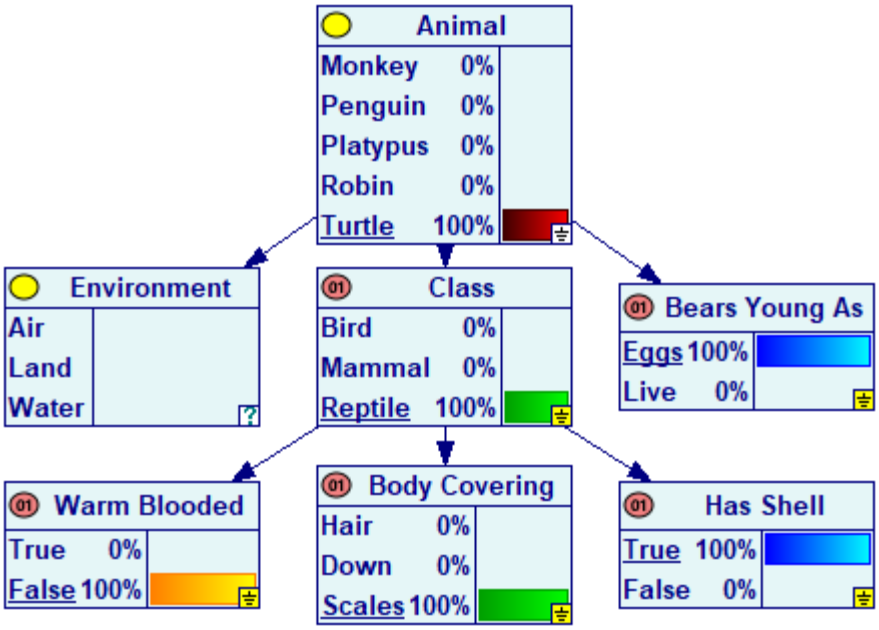
A target node will display a small target icon ( ).



One of the elements of relevance reasoning is propagation of evidence. When a node is observed, it often implies the states of its neighboring nodes with certainty. Consider the following simple network (`Animals.xdsl`, among the example networks in GeNIe) modeling a simple animal guessing game:



When we establish in the game that the animal has shell, we know for sure that the animal is a reptile, that it is a turtle, lays eggs and has scales for its body covering. These conclusions are just logical implications and can be drawn directly from the underlying probability tables without the need for performing any inference. GeNIe performs the operation of evidence propagation as one of the first steps of relevance reasoning and displays propagated evidence by a modified, yellow observation icon (see the screen shot below).

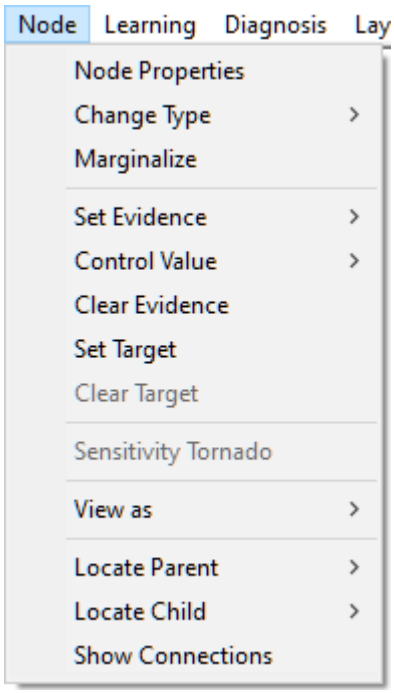


The foundations of relevance-based algorithms implemented in GeNIe are described in (Druzdzel & Suermondt 1994) and (Lin & Druzdzel 1997, 1998), although GeNIe's implementation goes a few steps further. Relevance algorithms usually lead to substantial savings in computation. We call this pre-processing step collectively *relevance reasoning*. Relevance reasoning is transparent to the user and the application programmer.

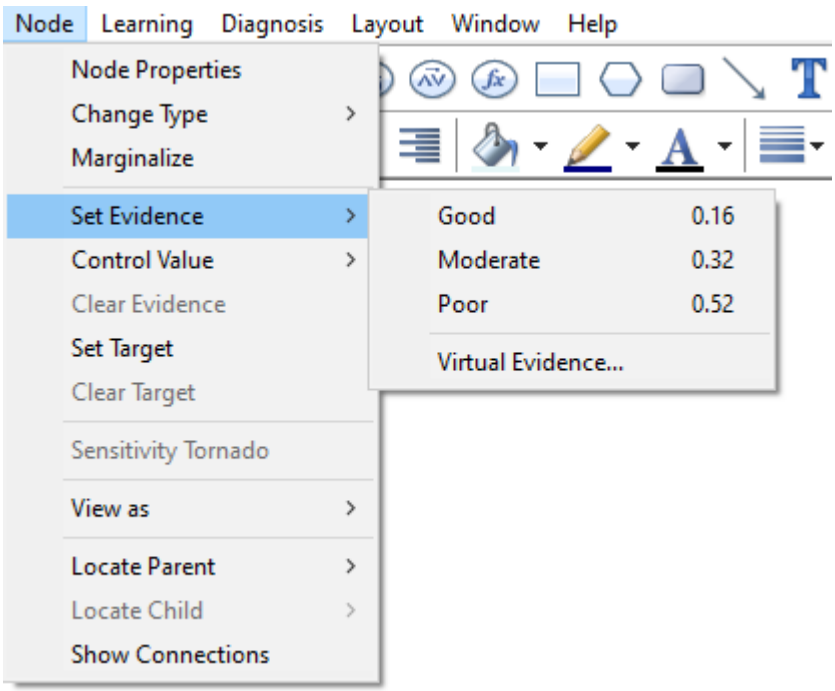
GeNIe has been originally written with teaching graphical models in mind. One of the fundamental concepts on which graphical models are built is conditional independence and relevance (Dawid 1979). In order to demonstrate how nodes are relevant to target variables and to evidence variables, GeNIe allows its user in a mode that does not immediately update nodes after the user has entered an observation or made a change in the model. This mode is also useful in case of an editing session. When editing a network, posterior probabilities may be of little interest. When the model is very large, consisting for example of hundreds of variables, this mode improves the reaction time.

5.7.4 Node menu

We have deferred the description of the commands *Set Evidence*, *Control Value*, *Clear Evidence*, *Set Target*, *Clear Target* and *Sensitivity Tornado* to the current section, which offers background information to Bayesian network algorithms. We reproduce the *Node Menu*, which is to a large degree duplicated by the node pop-up menu, below:



Set Evidence submenu (for Decision nodes, this submenu is called *Set Decision*) allows for setting the node state, which amounts to observation (or making a decision). The submenu is active only if there is one (and only one) node selected in the *Graph View*.



To select a state of the currently selected node, select this state on the submenu listing the states and release the mouse button. The state will have a check mark next to its name. The node will from that point on be equipped with the *Observed* ( ) status icon, indicating that one of the states of this node has been observed. If the node was equipped with the *Invalid* ( ) status icon, the icon will disappear (please note that the value of an observed node is known and it is, therefore, valid). To change the node back to the unobserved state, choose *Clear Evidence* from the *Node Menu*.

Control Value submenu works precisely like the *Set Evidence* submenu but it stands for controlling rather than observing the value. Controlling means that the value has been set from outside. GeNIe's implementation of controlling the value follows so called arc-cutting semantics, which means that the incoming arcs of the controlled node become inactive (nothing inside the model influences the node, as its value is set from outside). GeNIe shows these inactive arcs as inactive by dimming them. See *Controlling values* for more information about this functionality.

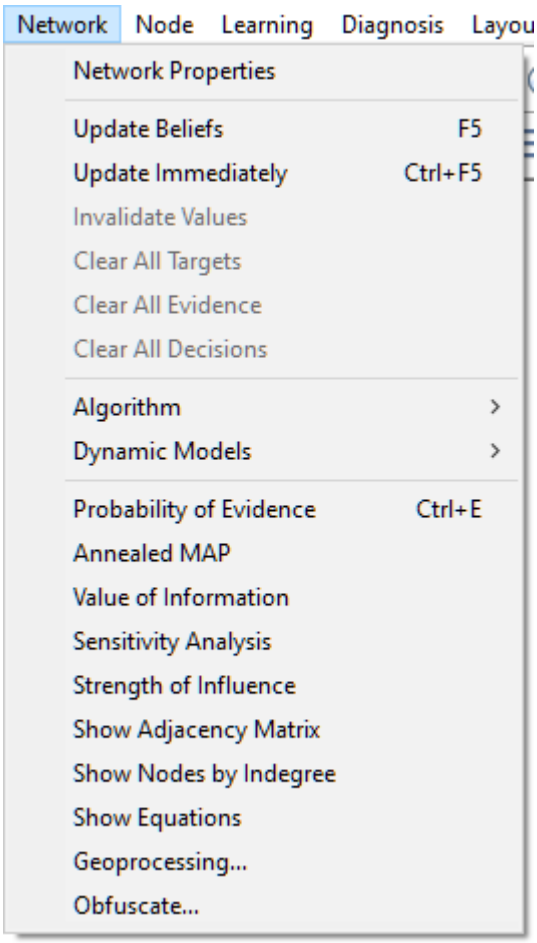
Clear Evidence command (for controlled nodes, this command is called *Release Value*) is active only if there are nodes selected in the *Graph View* and at least one of these node has been previously observed (or controlled). *Clear Evidence* un-observes a state, i.e., it reverses the effect of the *Set Evidence* command. *Release Value* un-controls a state, i.e., it reverses the effect of the *Control Value* command. The check mark next to the state name will disappear. The *Observed* ( ) status icon (*Controlled* status icon in case of controlled nodes) will disappear and possibly the Invalid ( ) status icon will appear.

The *Set Target* command is active only if there is at least one node selected in the *Graph View*. It allows you to set the status of the selected node(s) to be *Target*. Each of the selected nodes will from that point on be equipped with the Target ( ) status icon, indicating that the node is a target (i.e., its value or the probability distribution over its possible values are of interest to the user.

The *Clear Target* command is active only if there is at least one node selected in the *Graph View* and it is marked as a target. *Clear Target* reverses the effect of the *Set Target* command. The *Target* ( ) status icon will disappear.

5.7.5 Network menu

The *Network Menu* plays an important role in algorithms and allows for performing operations that relate to the entire network. It offers the following commands:



Network Properties invokes the *Network Properties* sheet for the current model. The network property sheet can also be invoked by double-clicking in any clear area on the main model *Graph* view window. See *Network properties* section for more information.

Update Beliefs (shortcut *F5*) command invokes the selected algorithm on the model. The *Update Beliefs* command can be also executed by pressing the *Update* ( ) tool from the *Standard Toolbar*.

Update Immediately (Shortcut *CTRL+F5*) command toggles between the immediate and lazy updating of models.

Invalidate values is a command that mimics a tool box in a Rolls Royce, which in theory should never be needed. Normally, GeNIe will take care that all values in the nodes that are not marked as *Invalid* are up to date. Occasionally, because of an (unlikely) error in the program, the values in nodes that are not *Invalid* may be wrong. Also, if you have run a stochastic sampling algorithm and would like to recompute the values with the new number of samples (larger number of samples give you a higher precision), you may need to invalidate all

values and force GeNIe to recompute them. The *Invalidate values* command is a manual escape for such situations. It will invalidate all values in the network and allow to update them, which in most cases should fix the problem.

Clear All Targets, *Clear All Evidence*, and *Clear All Decisions* allow for retracting all target markings, evidence, and decisions in one simple step rather than doing it for each individual node.

Algorithm submenu allows for choosing the default belief updating algorithm for Bayesian networks and the default evaluation algorithm for Influence diagrams. It is discussed in detail in the *Introduction* to the current section.

Dynamic Models submenu groups all operations on dynamic Bayesian networks. This submenu is discussed in detail in the section *Dynamic Bayesian networks*.

Probability of Evidence (shortcut *CTRL+E*) and *Annealed MAP* are special algorithms discussed in *Special algorithms* section.

Value of Information is a special algorithms for *Influence diagrams*, discussed in the *Influence diagrams* section.

Strength of Influence is a model exploration technique discussed in *Strength of influences* section.

Obfuscate... is a special algorithm for obfuscating the network for the purpose of protecting intellectual property, discussed in *Obfuscation* section.

Enable Diagnosis checkbox is used to enable/disable diagnostic features of GeNIe, discussed in *Support for Diagnosis* section.

5.7.6 Bayesian networks algorithms

5.7.6.1 Exact algorithms

5.7.6.1.1 Clustering algorithm

Clustering algorithm is the fastest known exact algorithm for belief updating in Bayesian networks. It was originally proposed by Lauritzen and Spiegelhalter (1988) and improved by several researchers, e.g., Jensen et al. (1990) or Dawid (1992).

The clustering algorithm works in two phases: (1) compilation of a directed graph into a junction tree, and (2) probability updating in the junction tree. It has been a common practice to compile a network and then perform all operations in the compiled version. Our research in relevance reasoning (Lin & Druzdzel 1997, 1998) has challenged this practice and has shown that it may be advantageous to pre-process the network before transferring it into a junction tree. GeNIe does not include the compilation phase in its user interface and the model under construction is always ready to perform inference.

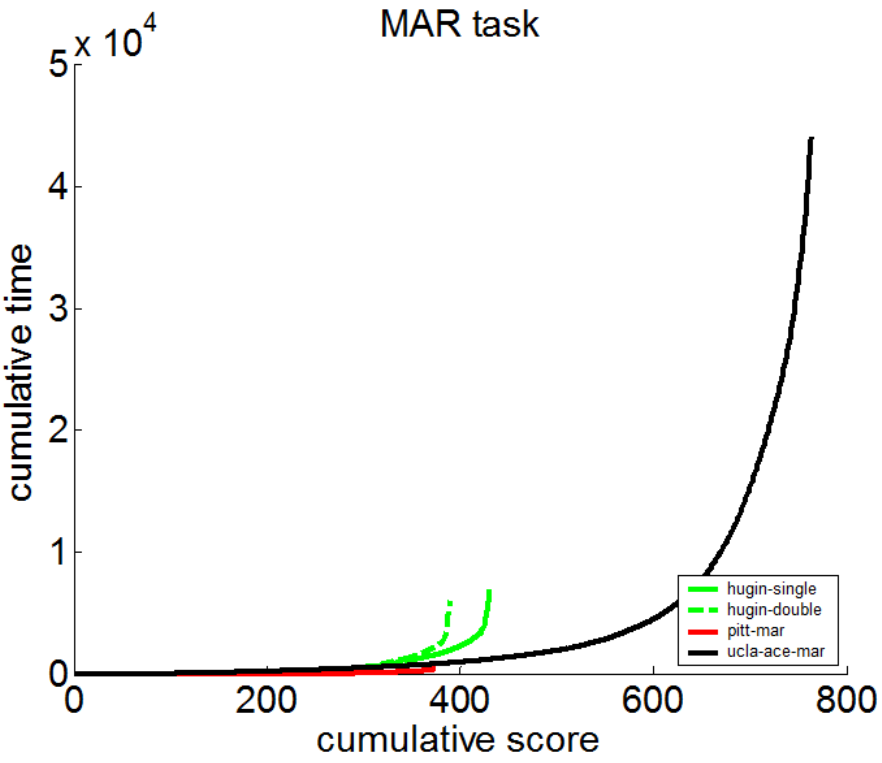
The clustering algorithm, like all of the algorithms for Bayesian networks, produces marginal probability distributions over all network nodes. In addition, it is possible to preserve clique potentials in the network, which allows for viewing joint probability distribution over those variables that are located within the same clique. Should you wish to derive the joint probability distribution over any variable set, just make sure that they are in the same clique

before running the clustering algorithm. One way of making sure that they are in the same clique is creating a dummy node that has all these variables as parents. In any case, when the *Preserve Clique Potentials* flag is on, there is an additional button in the *Value* tab of *Node Properties* dialog, *Show JPD* ( ), which will open a dialog for selecting a set of variables for viewing the joint probability distribution over them.

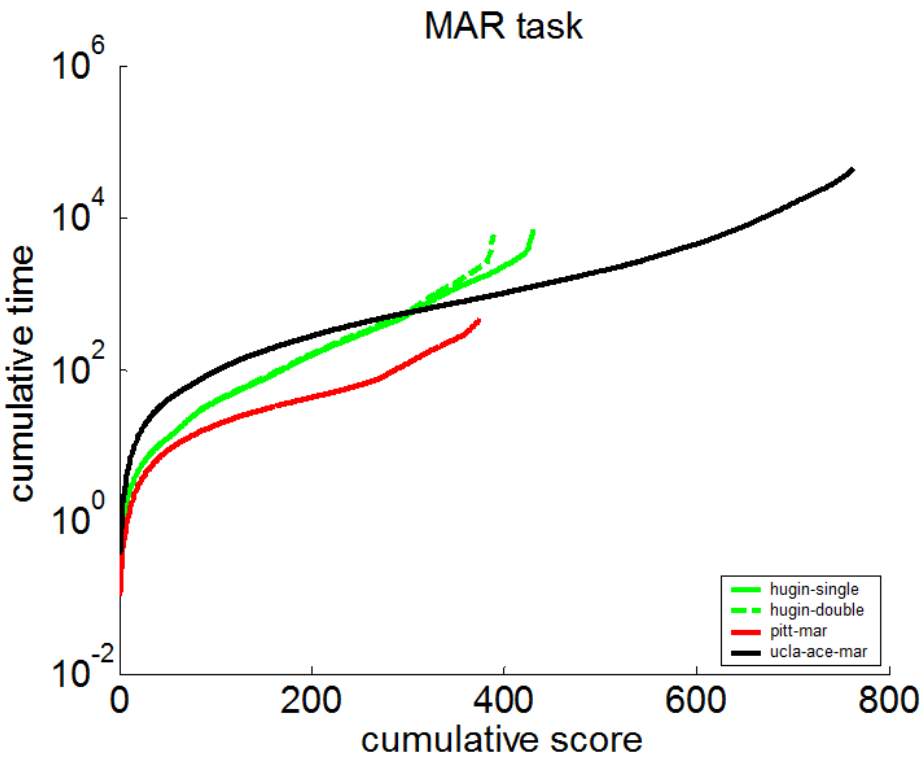
The clustering algorithm is GeNIe's default algorithm and should be sufficient for most applications. Only when networks become very large and complex, the clustering algorithm may not be fast enough. In that case, we suggest that the user choose an approximate algorithm, such as one of the stochastic sampling algorithms. The best stochastic sampling algorithm available for discrete Bayesian networks is EPIS-BN (Yuan & Druzdzel, 2003).

SMILE's clustering algorithm in UAI-2006 and UAI-2008 inference evaluation

SMILE's implementation of the clustering algorithm underlies the implementation of calculation of the marginal probability and the probability of evidence. SMILE did very well in the UAI-2006 and UAI-2008 inference evaluation. We report the results of the evaluation in the probability of evidence section. Here we show the results of the marginal probability competition in UAI-2008. The following plot show the cumulative time to solve the test instances. The lower the curve, the better. SMILE is shown by the red curve (see https://www.ics.uci.edu/~dechter/softwares/benchmarks/UAI08/uai08-evaluation-2008-09-15.pdf for the full report).



The same curve in logarithmic scale shows that SMILE was typically half an order of magnitude faster than the other software taking part in the competition.



See also the evaluation results for computing the probability of evidence and approximate marginals in the *Probability of evidence* and *EPIS Sampling* sections respectively.

5.7.6.1.2 Relevance-based decomposition

Relevance-based decomposition is an exact algorithm based on the clustering algorithm that performs a decomposition of the network when the network is very large. The algorithm was described in (Lin & Druzdzel, 1997). Relevance-based decomposition extends the boundary of what is computable, while gracefully changing into the clustering algorithm for small networks. Because there is some overhead related to decomposition, we suggest that this algorithm be used only when the clustering algorithm cannot handle your networks.

5.7.6.1.3 Polytree algorithm

The belief updating algorithm for singly connected networks (polytrees) was proposed by (Pearl 1986). It is the only belief updating algorithm that is of polynomial complexity, but unfortunately this result and the algorithm works only in singly connected networks (i.e., networks in which any two nodes are connected by at most one undirected path). GeNIe will not start the algorithm unless the model is singly connected.

5.7.6.2 Stochastic sampling algorithms

5.7.6.2.1 Probabilistic Logic Sampling

The probabilistic logic sampling algorithm is described in (Henrion 1988), who can be considered the father of stochastic sampling algorithms for Bayesian networks. The probabilistic logic sampling algorithm should be credited as the first algorithm applying stochastic sampling to belief updating in Bayesian networks.

Essentially, the algorithm is based on forward (i.e., according to the weak ordering implied by the directed graph) generation of instantiations of nodes guided by their prior probability. If a generated instantiation of an evidence node is different from its observed value, then the entire sample is discarded. This makes the algorithm inefficient if the prior probability of evidence is low. The algorithm is very efficient in cases when no evidence has been observed or the evidence is very likely.

5.7.6.2.2 Likelihood Sampling

This likelihood sampling algorithm is described in (Fung & Chang 1990) and in (Shachter & Peot 1990). Our implementation is based on (Fung & Chang 1990).

The likelihood sampling algorithm makes an attempt to improve the efficiency of the *Probabilistic Logic Sampling* algorithm by instantiating only non-evidence nodes. Each sample is weighted by the likelihood of evidence given the partial sample generated. It is a simple algorithm with little overhead that generally performs well and certainly better than *Probabilistic Logic Sampling* in cases with observed evidence.

5.7.6.2.3 Backward Sampling

The *Backward Sampling* algorithm is described in (Fung & del Favero 1994).

It attempts to defy the problems with unlikely evidence by sampling backward from the evidence nodes. As nodes can be sampled both backward and forward, depending on whether they have direct ancestors or descendants sampled, this algorithms is an ingenious extension to forward sampling algorithms.

5.7.6.2.4 AIS algorithm

The *Adaptive Importance Sampling* (*AIS*) algorithm is described in (Cheng & Druzdzel 2000). This algorithm offered a breakthrough in the field of stochastic sampling algorithms when first published in 2000. In really difficult cases, such as reasoning under very unlikely evidence in very large networks, the AIS algorithm produced two orders of magnitude smaller error in posterior probability distributions than other sampling algorithms available at that time. Improvement in speed given a desired precision were even more dramatic. The AIS algorithm is based on importance sampling. According to the theory of importance sampling, the closer the sampling distribution is to the (unknown) posterior distribution, the better the results will be. The AIS algorithm successfully approximates its sampling distribution to the posterior distribution by using two cleverly designed heuristic methods in its first stage, which leads to the big improvement in performance stated above.

The AIS algorithm was judged to be one of the most influential developments in the area of Artificial Intelligence in 2005, receiving Honorable Mention in the 2005 IJCAI–JAIR Best Paper Prize. The IJCAI-JAIR (*International Joint Conference on Artificial Intelligence* and *Journal of Artificial Intelligence Research*) Best Paper Prize is awarded to an outstanding paper published in JAIR in the preceding five calendar years. For the 2005 competition, papers published between 2000 and 2005 were eligible. The algorithm achieved up to two orders of magnitude better accuracy than any other sampling algorithm available at that time. This algorithm was surpassed in 2003 by another algorithm developed by Prof. Druzdzel's Decision Systems Laboratory, the EPIS algorithm (Yuan & Druzdzel 2003), which sometimes offered an order of magnitude improvement over the AIS algorithm.

5.7.6.2.5 EPIS Sampling

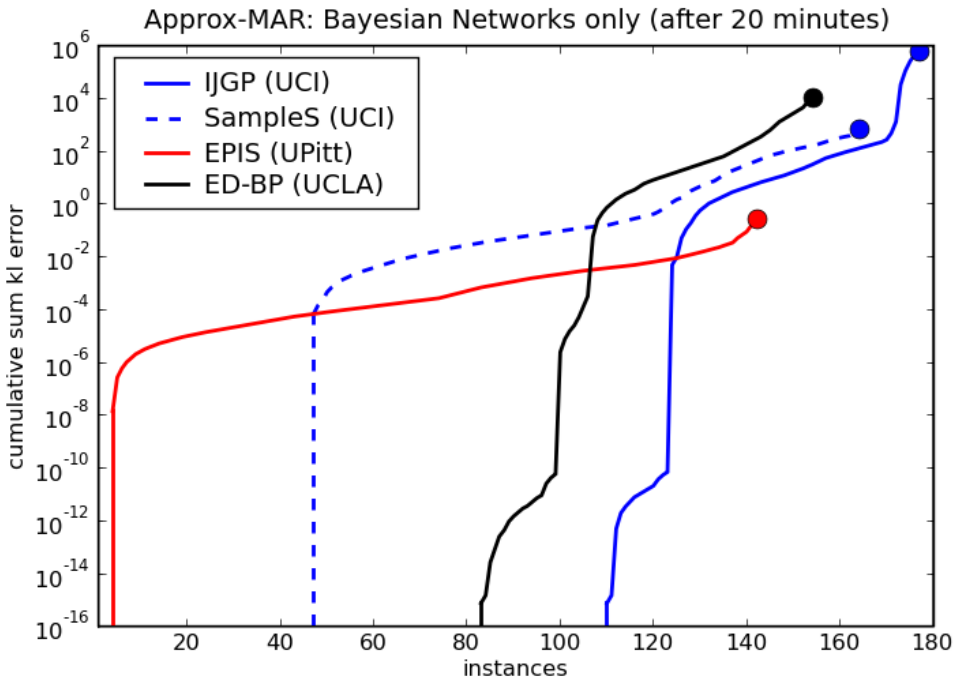
The *Estimated Posterior Importance Sampling* (*EPIS*) algorithm is described in (Yuan & Druzdzel 2003). This is quite likely the best stochastic sampling algorithm for discrete Bayesian networks available. It produces

results that are even more precise than those produced by the AIS-BN algorithm and in case of some networks produces results that are an order of magnitude more precise. The EPIS-BN algorithm uses loopy belief propagation to compute an estimate of the posterior probability over all nodes of the network and then uses importance sampling to refine this estimate. In addition to being more precise, it is also faster than the AIS-BN algorithm, as it avoids the costly learning stage of the latter.

EPIS algorithm can be tuned to improve its accuracy. Its default parameters are listed in the Inference tab of the *Network properties* dialog. The EPIS algorithm uses *Loopy Belief Propagation* (LBP), an algorithm proposed originally by Judea Pearl (1988) for polytrees and later applied by others to multiply-connected Bayesian networks. EPIS uses LBP to pre-compute the sampling distribution for its importance sampling phase. *Propagation length* is the number of LBP iterations in this pre-computation. EPIS uses Epsilon-cutoff heuristic (Cheng & Druzdzel, 2000) to modify the sampling distribution, replacing probabilities smaller than epsilon by epsilon. The table in the *Sampling* tab allows for specifying different threshold values for nodes with different number of outcomes. For more details on the parameters and how they influence the convergence of the EPI-BN algorithm, please see (Yuan & Druzdzel 2003).

EPIS Sampling algorithm in UAI-2008 inference evaluation

SMILE's implementation of the EPIS Sampling algorithm did well in the UAI-2008 inference evaluation. The following plot show the cumulative error when solving the test instances.



We misunderstood the rules of the competition at the time of submission and did not use exact inference whenever it was possible (for the easiest instances). EPIS Sampling algorithm ran on even the simplest networks and against algorithms that used exact inference on those networks that were computable. Hence, its error on the left-hand side of the graph (the simplest networks) is much larger than that of the other algorithms. Had we used SMILE's clustering algorithm for these simple cases, EPIS's curve (shown in red) would be quite likely close to the *x* axis. Still, as time progressed and the algorithms faced more difficult cases, EPIS ended up with a lower cumulative error than the other algorithms. Please note that the increase in error between the simplest and more

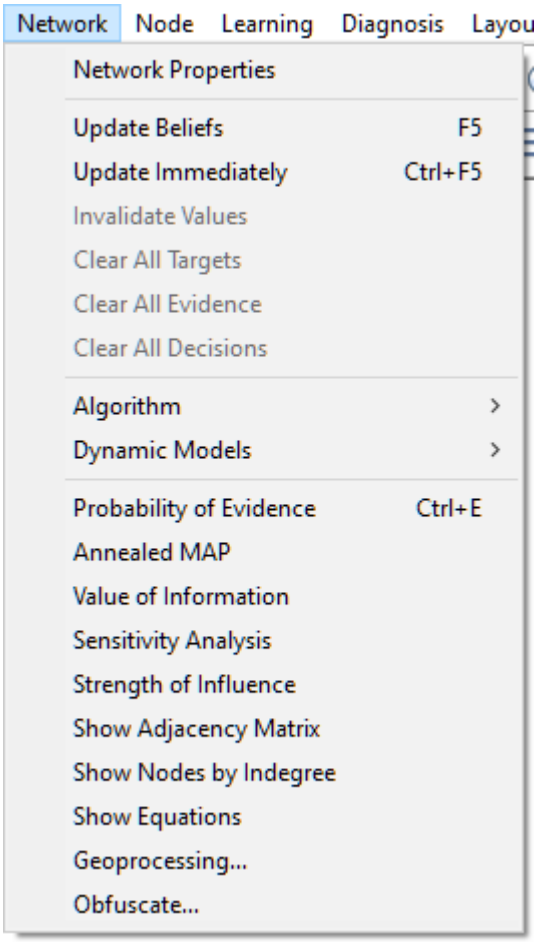
complex instances is not large. See https://www.ics.uci.edu/~dechter/softwares/benchmarks/UAI08/uai08-evaluation-2008-09-15.pdf for the full report.

See also the evaluation results for computing the probability of evidence and marginals in the *Probability of evidence* and *Clustering algorithm* sections respectively.

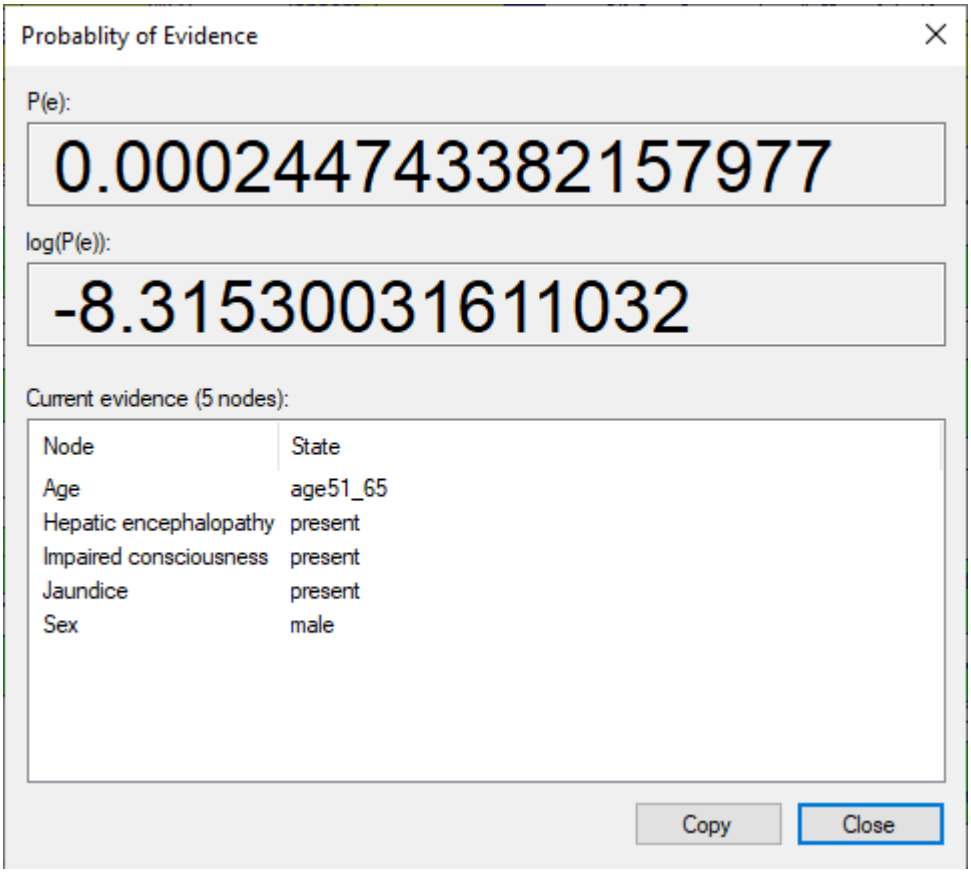
5.7.6.3 Special algorithms

5.7.6.3.1 Probability of evidence

One of the useful possible calculations in a probabilistic model is the (*a-priori*) probability of evidence. Given a number of observations entered in a network, we ask the question: *How likely is this set of observations within this model?* To invoke the probability of evidence calculation, choose *Probability of Evidence* (shortcut *CTRL+E*) from the *Network Menu*. or press the *Calculate probability of evidence* ( ) button from the main toolbar.



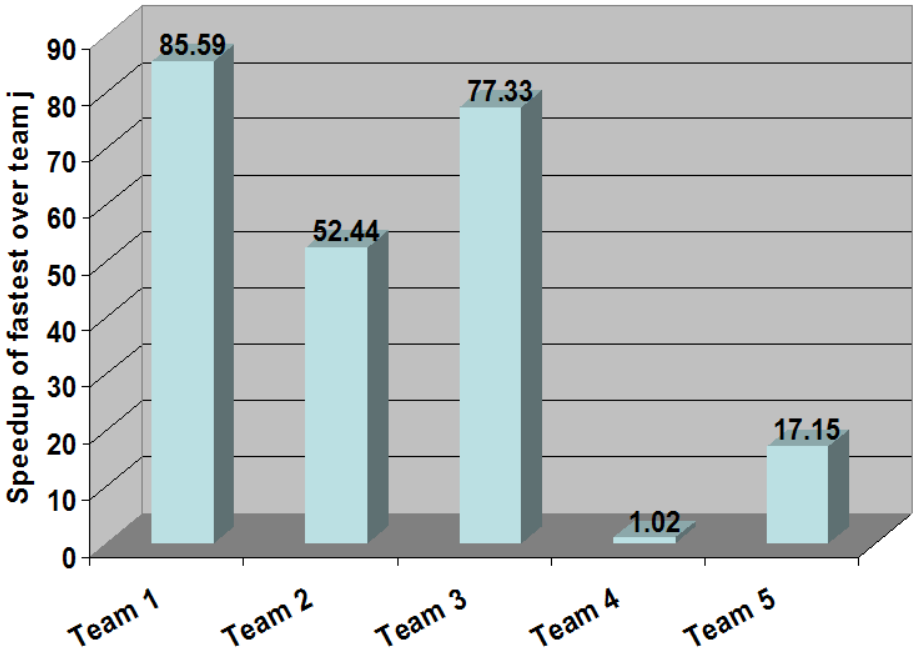
The following dialog displays the results of this calculation:



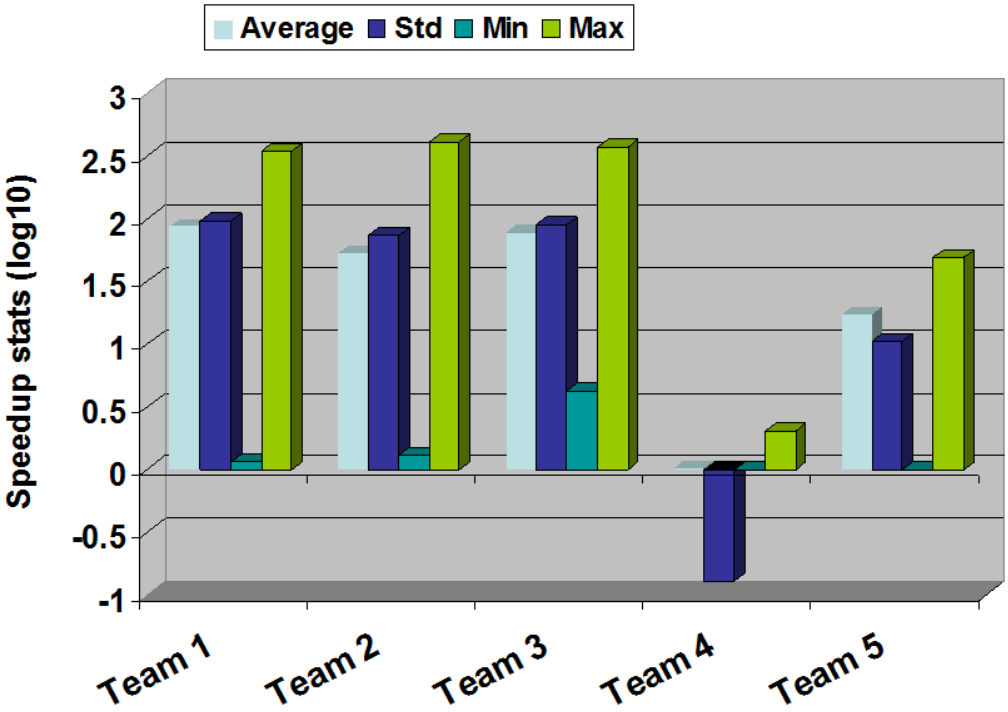
We can see that according to the model at hand, the *a-priori* probability of observing a *male* patient between the ages of *51* and *65* with *Hepatic encephalopathy*, *Impaired consciousness* and *Jaundice* is, according to the Hepar II model, roughly 0.000245.

SMILE's probability of evidence algorithm in UAI-2006 and UAI-2008 inference evaluation

SMILE's probability of evidence algorithm performed very well in the UAI-2006 inference evaluation competition (see https://melodi.ee.washington.edu/~bilmes/uai06InferenceEvaluation/). The following plot shows the average speedup of the fastest team over each of the teams (the lower the bar the better with 1.0 being perfect). SMILE (Team 4) scored 1.02 and was roughly two orders of magnitude faster than other teams. SMILE was slower than the best team's program in less than 2% of the test cases.

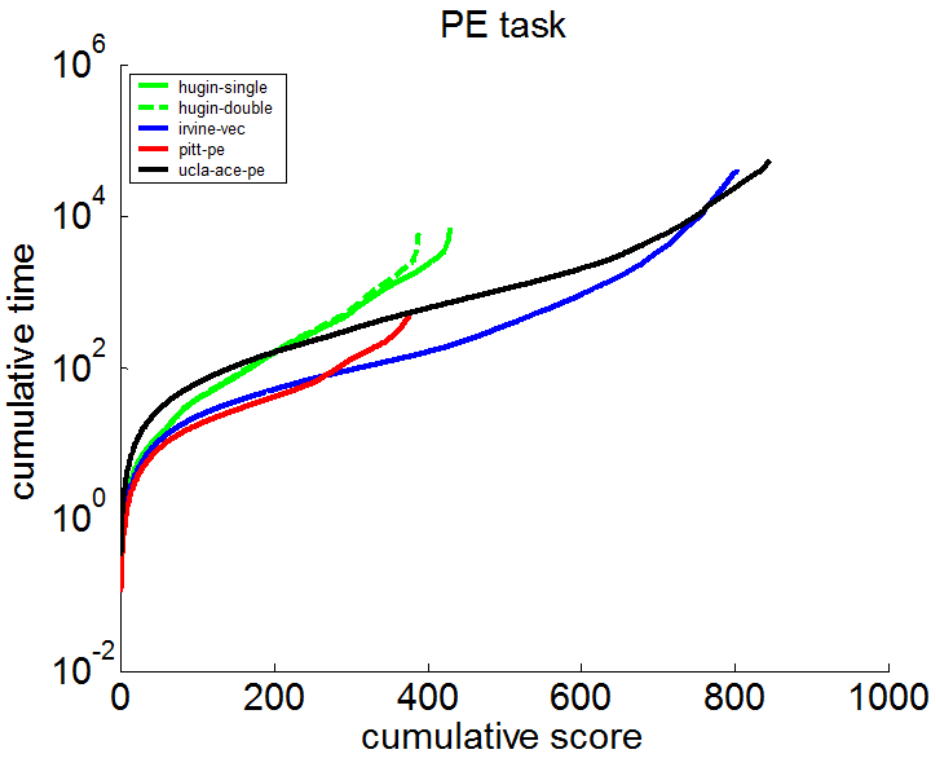


The following plot shows the spread parameters of the speedup (previous plot). SMILE (Team 4) was reliably the fastest with the average standard deviation of 0.01.



In the UAI-2008 inference evaluation competition (see https://www.ics.uci.edu/~dechter/softwares/benchmarks/UAI08/uai08-evaluation-2008-09-15.pdf for the full

report), the algorithm did very well again. SMILE typically computed very fast whatever was computable within a short time. The following plot shows the logarithm of the cumulative time taken over test instances. SMILE was up to roughly half an order of magnitude faster than the other software.

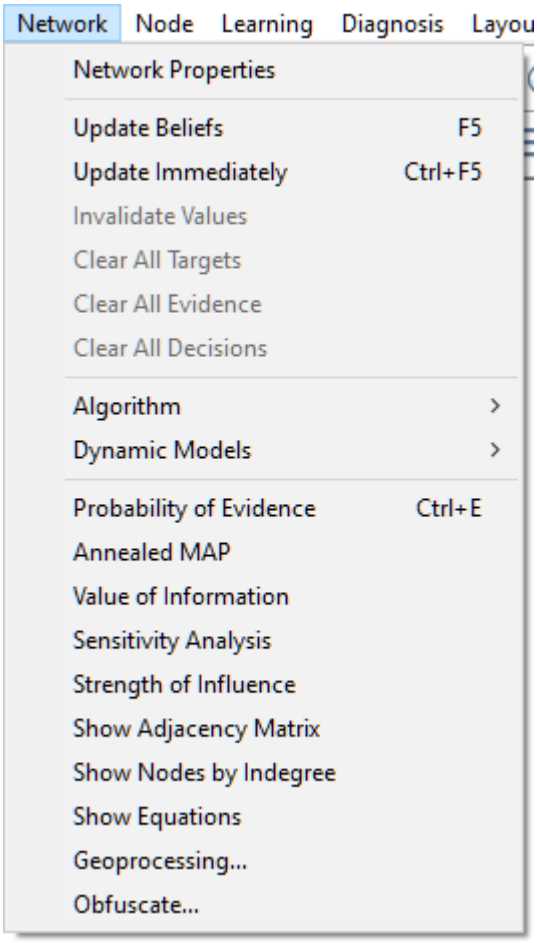


See also the results of computing the exact and approximate marginals in the *Clustering Algorithm* and *EPIS Sampling* sections respectively.

5.7.6.3.2 Annealed MAP

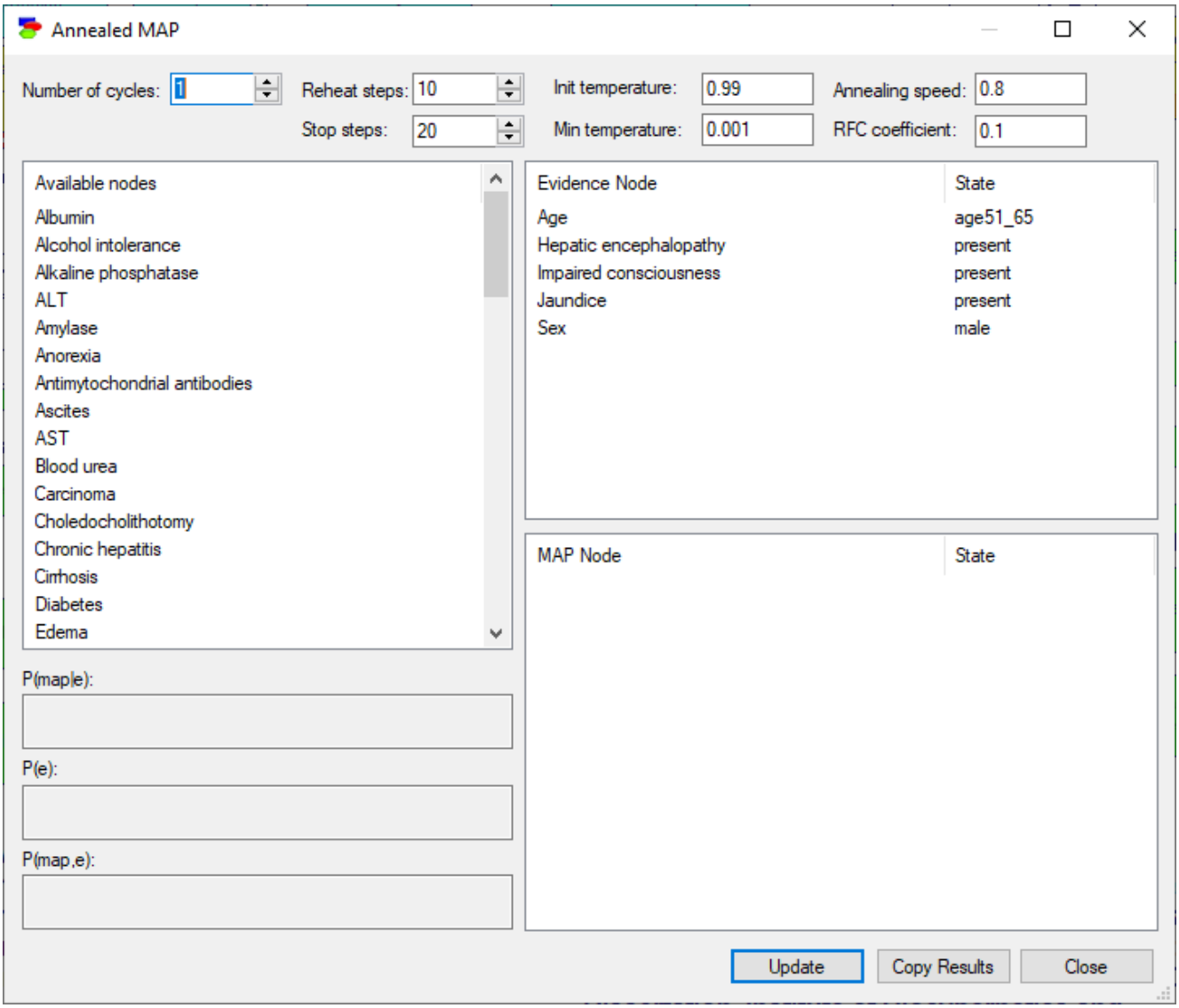
The *Annealed MAP* algorithm (Yuan et al., 2004) solves the problem of finding the most likely configuration of values of a set of nodes given observations of another subset of nodes. This problem is often called *Maximum A posteriori Probability* (*MAP*). The *Annealed MAP* algorithm is approximate and solves the problem by means of an approximate optimization procedure called simulated annealing. While the solution is approximate, it performs well in practice and it gives an idea of the order of magnitude of the true maximum. The *Annealed MAP* algorithm drastically extends the class of MAP problems that can be solved.

To invoke the algorithm dialog, select *Annealed MAP* from the *Network Menu*



Annealed MAP dialog

The Annealed MAP dialog that appears shows three window panes and several user-settable parameters.

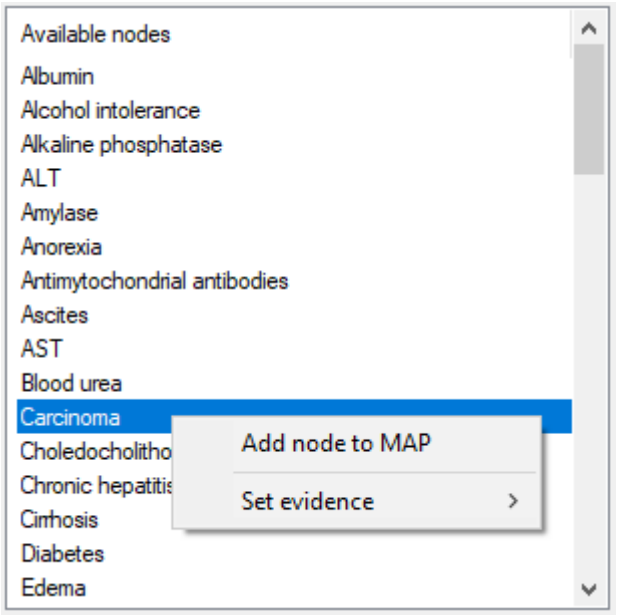


The upper-left window pane contains *Available nodes*, which are all nodes in the current network that have not been observed.

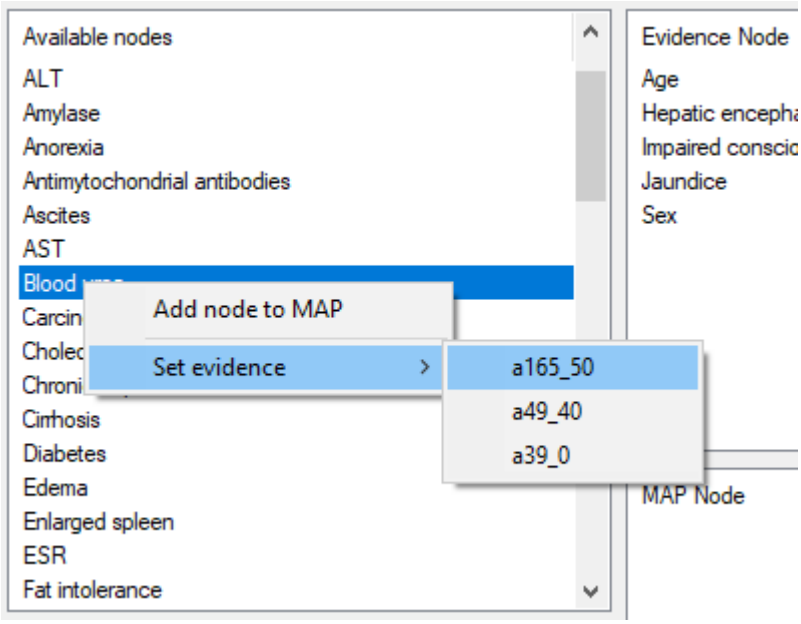
The upper-right window pane contains *Evidence Nodes* and their *States*. In this case, we have five observed evidence nodes.

The lower-right window is meant to list the nodes in the MAP set (*MAP Nodes*) and will be filled by the algorithm with their *States*.

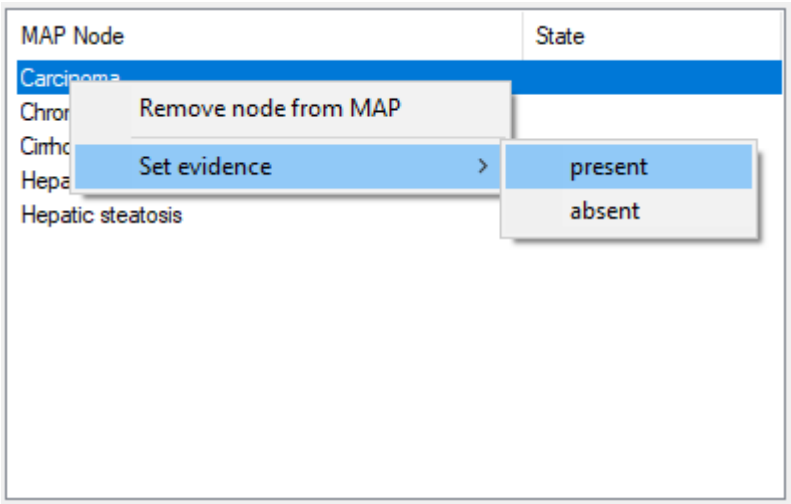
Nodes in the Available nodes set can be observed or added to the MAP set. To add a node to the MAP set, right-click on a node name and select *Add node to MAP* from the pop-up menu.



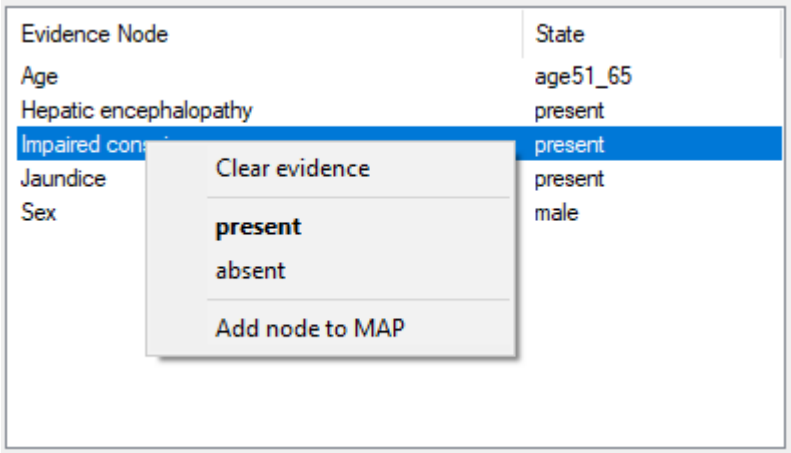
To add a node to the evidence set, right-click on a node name, select *Set evidence*, and then the observed state from the pop-up menu.



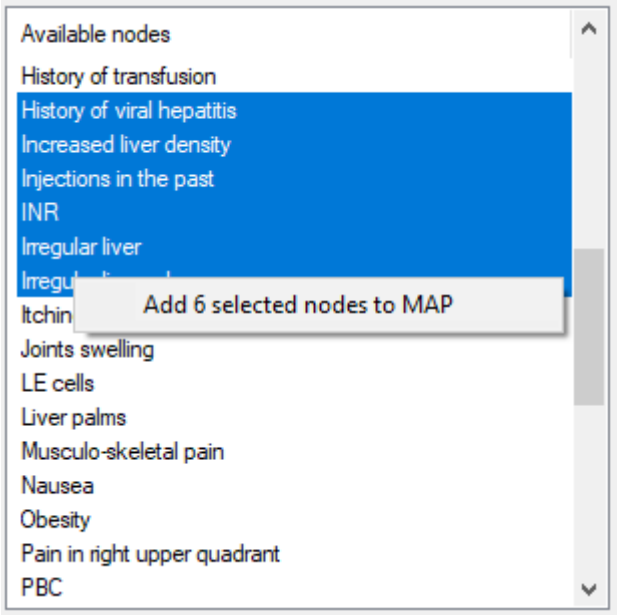
To move a node back from the *MAP Node* set or to move it to the evidence set, right-click on the node name and make an appropriate selection from the pop-up menu.



The same can be done to nodes in the Evidence Node set:



Nodes can be moved between windows in groups as well - just select multiple nodes before right-clicking:



Parameters

Annealed MAP algorithm has seven parameters:

- Number of cycles (default 1)
- Reheat steps (default 10)
- Stop steps (default 20)
- Init temperature (default 0.99)
- Min temperature (default 0.001)
- Annealing speed (default 0.8), and
- RFC coefficient (default 0.1)

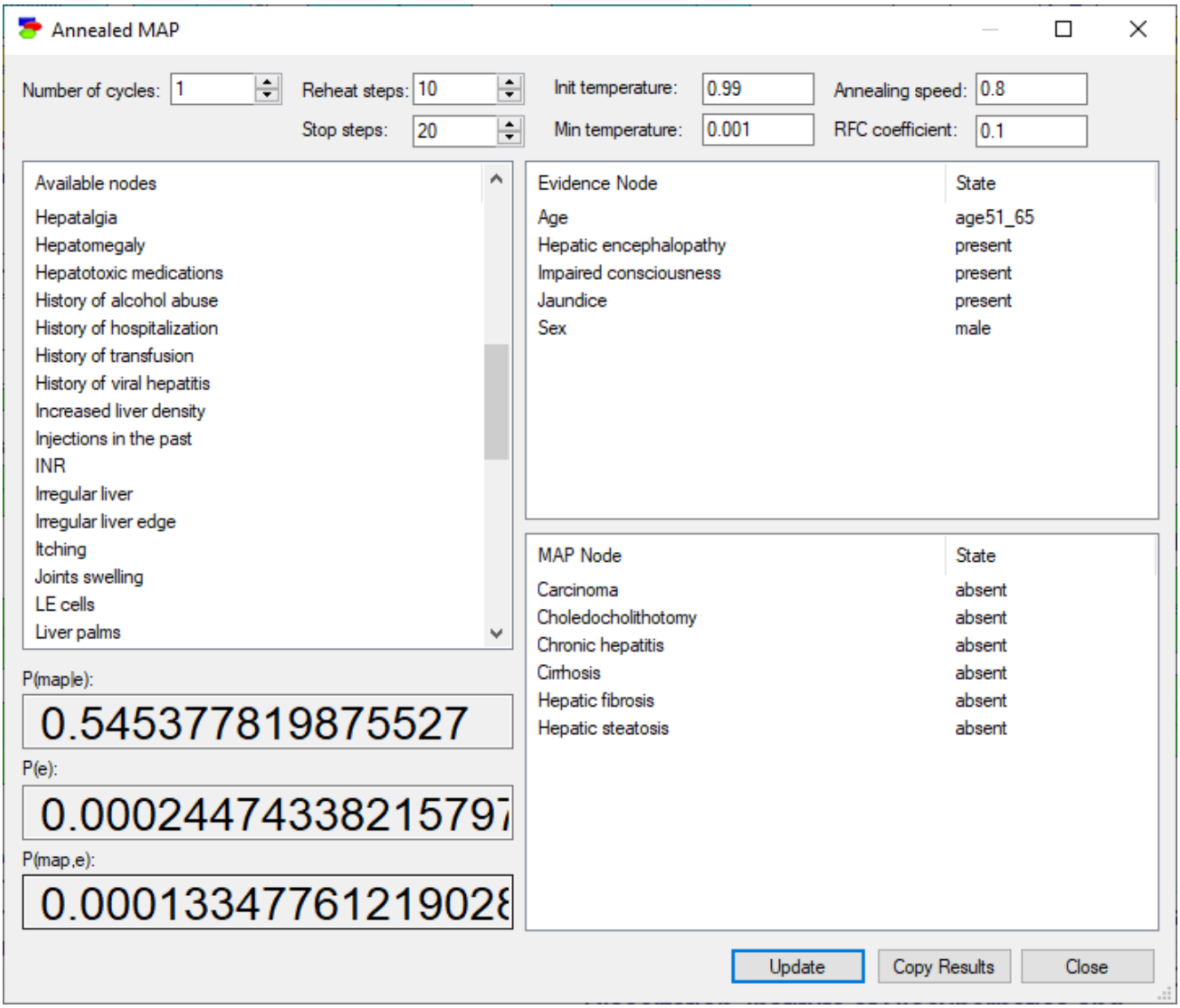
These parameters can be fine tuned to obtain better results. More information about the meaning of these parameters can be found in (Yuan et al., 2004).

Algorithm execution and results

Pressing the *Update* button starts the Annealed MAP algorithm, which finds the maximum *a posteriori* probability assignment of states to the MAP Node set and calculates the following three probabilities:

- P(MAP|E): The probability of the MAP assignment given the set of evidence

- P(E): The probability of evidence
- P(MAP,E): The joint probability of MAP assignment and the evidence



We can read from the results that the most likely combination of states of the six diseases chosen for the *MAP Node* set is that they are all absent. The probability of this combination of states is roughly P(MAP|E)=0.5454.

Pressing *Copy Results* copies the most important results to the clipboard. The results can be pasted into a different program. The paste operation in the above example gives the following result:

```
P(map|e)=0.545377819875527
P(e)=0.000244743382157977
P(map,e)=0.000133477612190281

MAP nodes:
Carcinoma    absent
```

```
Choledocholithotomy   absent
Chronic hepatitis absent
Cirrhosis   absent
Hepatic fibrosis  absent
Hepatic steatosis absent

Evidence nodes:
Age    age51_65
Hepatic encephalopathy present
Impaired consciousness present
Jaundice    present
Sex    male
```

Please remember to press the *Update* button whenever you have changed any of the Annealed MAP algorithm parameters, performed any new observations, or moved any nodes between the *Available nodes* and *MAP Nodes* windows.

Pressing *Close* closes the *Anneal MAP* dialog.

5.7.7 Influence diagrams algorithms

5.7.7.1 Policy evaluation

This *policy evaluation algorithm* is the main algorithm for solving influence diagrams in GeNIe. It's implementation is based on the algorithm proposed by Cooper (1988). The policy evaluation algorithm solves an influence diagram by first transforming it into a Bayesian network and then finding the expected utilities of each of the decision alternatives by performing repeated inference in this network. The algorithm will result in a full set of expected utilities for all possible policies in the network. This may be a computationally intensive process for large influence diagrams. If you are not interested in the values of expected utilities, but would just like to know the optimal decision, consider using the algorithm for finding the best policy. This having said, GeNIe is very fast, so you can use this algorithm until the unlikely event that it will become too slow for your influence diagrams.

GeNIe does not require the user to specify the temporal order among decision nodes in influence diagrams. However, if the order is not specified by the user, and it cannot be inferred from causal considerations (please note that directed paths starting at a decision node are necessarily causal), GeNIe will assume an order arbitrarily and make it explicit by adding arcs between decisions placing an appropriate message in the console window. GeNIe does not require the user to create non-forgetting arcs in order to avoid obscuring the structure of the model. However, it behaves as if they were there, assuming their existence from the temporal order among the decision nodes.

Finally, the policy evaluation algorithm uses the default Bayesian network algorithm specified by the user. This may have an impact on both the computational performance and the accuracy of the computation.

If you are more concerned about finding what is the optimal decision at the highest level rather than the actual utilities for each decision, algorithm for finding the best policy is a better choice. It is generally much faster than the policy evaluation algorithm.

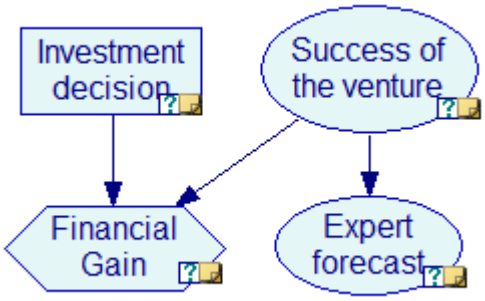
5.7.7.2 Find Best Policy

Sometimes the only thing that we might be interested in is the optimal decision at the top level of the influence diagram. Since the policy evaluation algorithm computes the expected utilities of all possible policies, it may be

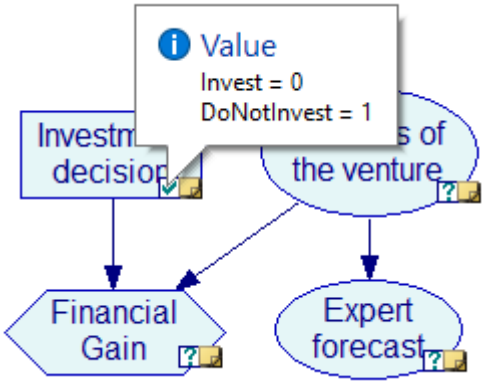
doing unnecessary work. For those cases, SMILE and GeNIe provide a simplified but very fast algorithm proposed by Shachter and Peot (1992). The algorithm instantiates the first decision node to the optimal decision alternative but does not produce the numerical expected utility of this or any other decision option. In order for this algorithm to be run, all informational predecessors of the first decision node have to be instantiated.

Here is an example of the finding the best policy algorithm in action.

Consider the influence diagram from the *Building an influence diagram* section (included among the example models as `VentureID.xdsl`).



After selecting the *Find Best Policy* algorithm from the *Network Menu* and updating the network, only the decision node, *Investment decision* is updated. The algorithm does not produce the expected utilities but rather indicates which decision option will give the highest expected utility (*DoNotInvest* in this case).



If we had other decision nodes in the model, they would not be updated at this stage. This is because the *Find Best Policy* algorithm only finds the best policy for the next decision node in the network. To find the best policy for other decision nodes in the network, you first need to set the decision for the first decision node and then update the network again.

If you need to see the expected utilities for the decisions, policy evaluation algorithm is a better choice. Generally, the *Find Best Policy* algorithm is most suitable for autonomous agents that just need to know what to do next and are not interested in expected utilities, their relative values, or sensitivities of the optimal policies to various elements of the model.

5.7.8 Algorithms for continuous and hybrid models

5.7.8.1 Introduction

This section describes inference in continuous and hybrid Bayesian networks, i.e., networks including both discrete and continuous variables. There are two algorithms that GeNIe relies on in such networks: (1) hybrid forward sampling and (2) discretization.

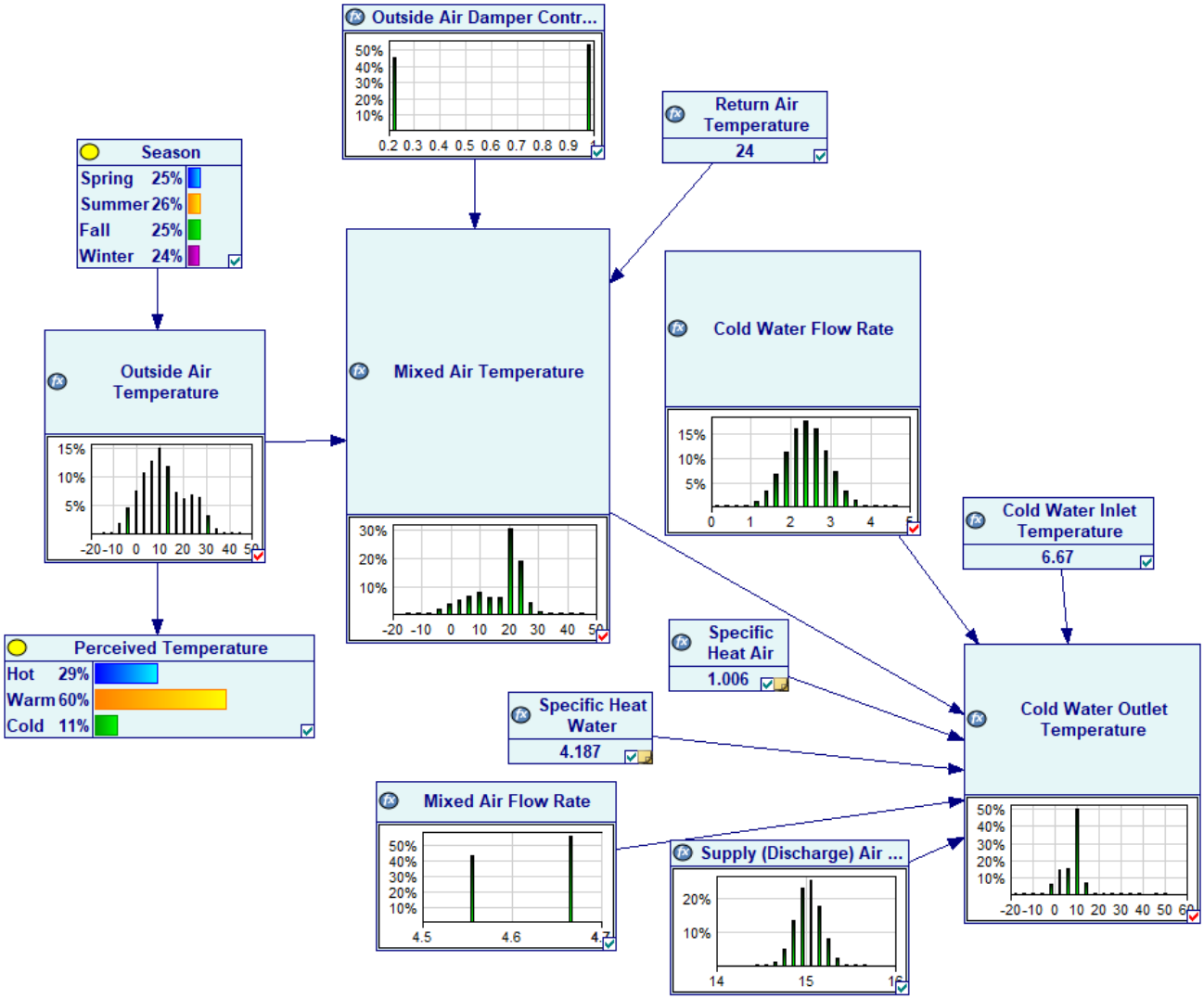
Hybrid forward sampling is used whenever there is no evidence in the network or all evidence is in parent-less nodes. In that case, stochastic forward sampling is very efficient and its complexity is polynomial in the number of nodes. Whenever there is evidence in nodes with parents, forward sampling runs into the problem of very unlikely evidence. Samples involving the observed values become extremely unlikely, approaching zero probability.

GeNIe's answer to this problem is converting the hybrid network into a discrete Bayesian network for the purpose of inference. This is accomplished by discretizing every continuous variable in the model, according to the discretization intervals specified by the user or, if no discretization is specified, offering a crude discretization that the user can refine further. The discretized network offers insight into the continuous posterior marginal distributions over the variables of interest. GeNIe puts no limitations on the functional forms of the interactions between variables in the model, offering its user a complete modeling freedom.

The following two sections describe the details of these two approaches.

5.7.8.2 Hybrid Forward Sampling

Hybrid forward sampling is a simple modification of the *Probabilistic Logic Sampling* algorithm that works in hybrid Bayesian networks, i.e., networks including both discrete and continuous variables. The algorithm draws samples in the forward direction, i.e., first it samples from parents-less nodes and then, once samples have been drawn from all parents of a child node, from the child node. Samples are drawn from both continuous and discrete nodes, according to the nodes' definitions. The results of the hybrid forward sampling algorithm are shown as histograms of the samples. Here is a hybrid Bayesian network (included among the example networks as `Heat Equations Autodiscretized Hybrid.xdsl`) updated by the hybrid forward sampling algorithm:

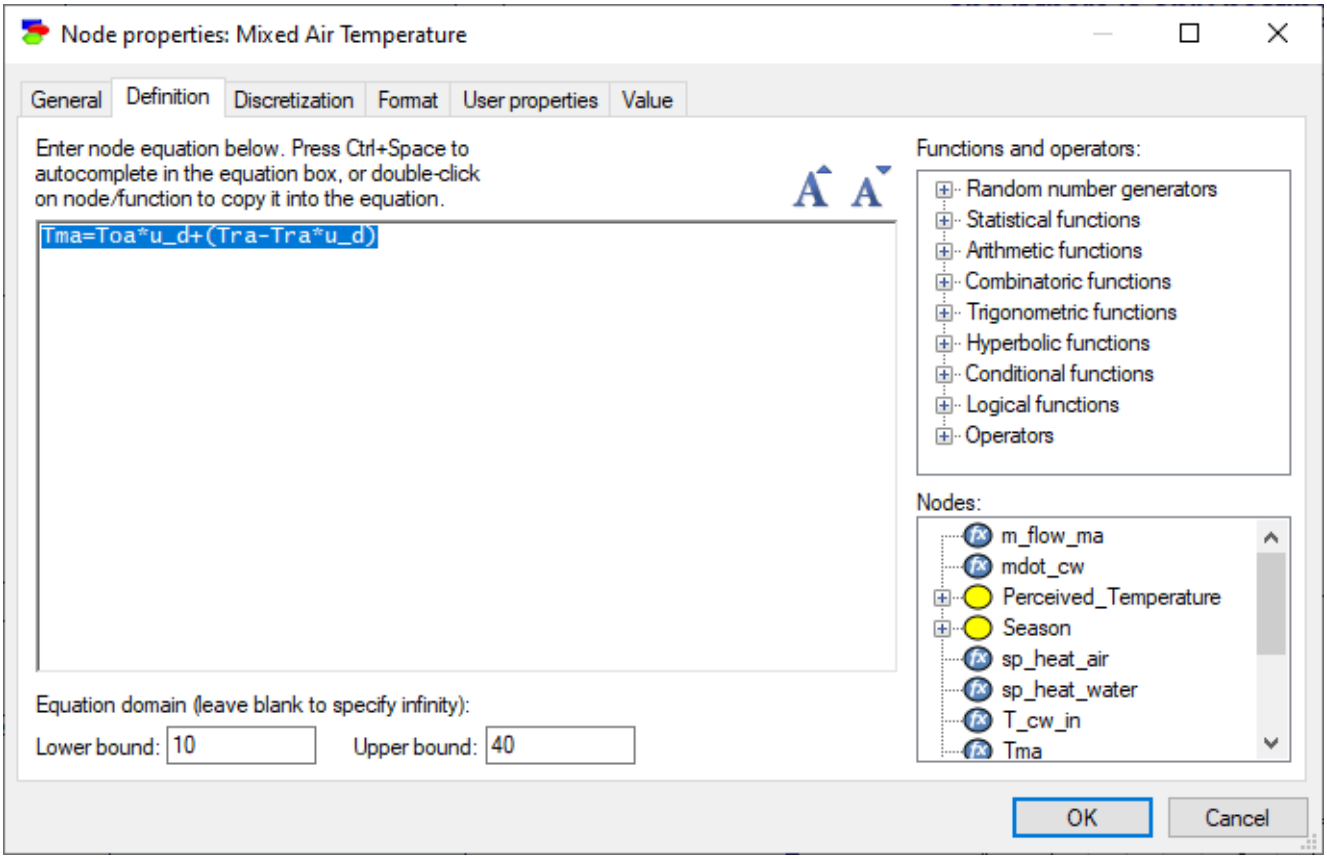


Please note that the hybrid forward sampling algorithm derives the marginal probability distributions over the variables in a hybrid network according to their types. In both cases, probability distribution are histograms of samples drawn during the sampling.

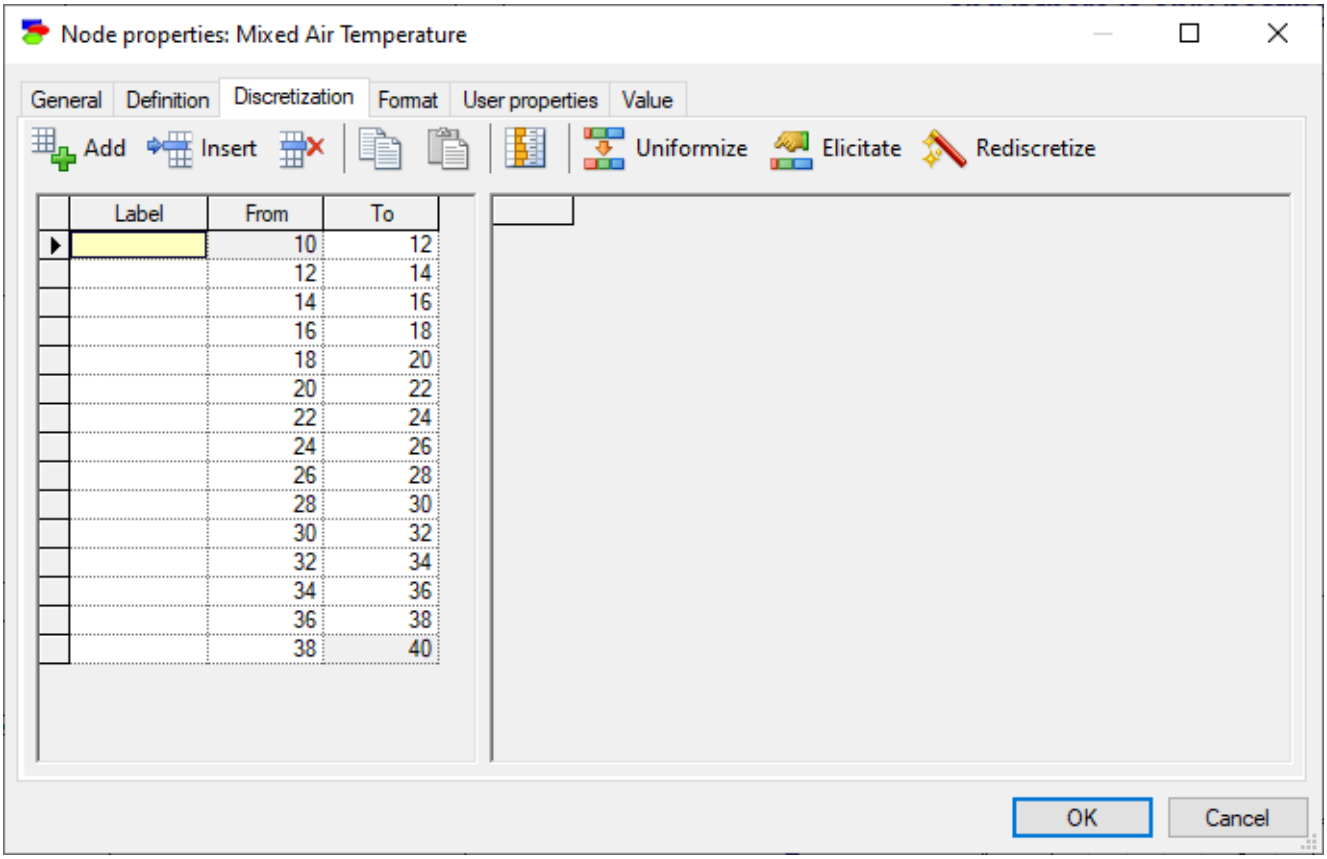
5.7.8.3 Autodiscretization

Autodiscretization is an approach to solving hybrid Bayesian networks in which there have been observations made in nodes that have parents. In its intermediate stage, the algorithm converts a hybrid Bayesian network into a discrete Bayesian network. For any continuous variable, the user can specify a discretization to follow in this conversion. Once the network has been converted, it behaves like a discrete Bayesian network. The big advantage of this algorithm is that is converts the original hybrid Bayesian network into a discrete Bayesian network only for the purpose of inference, preserving the modeling freedom.

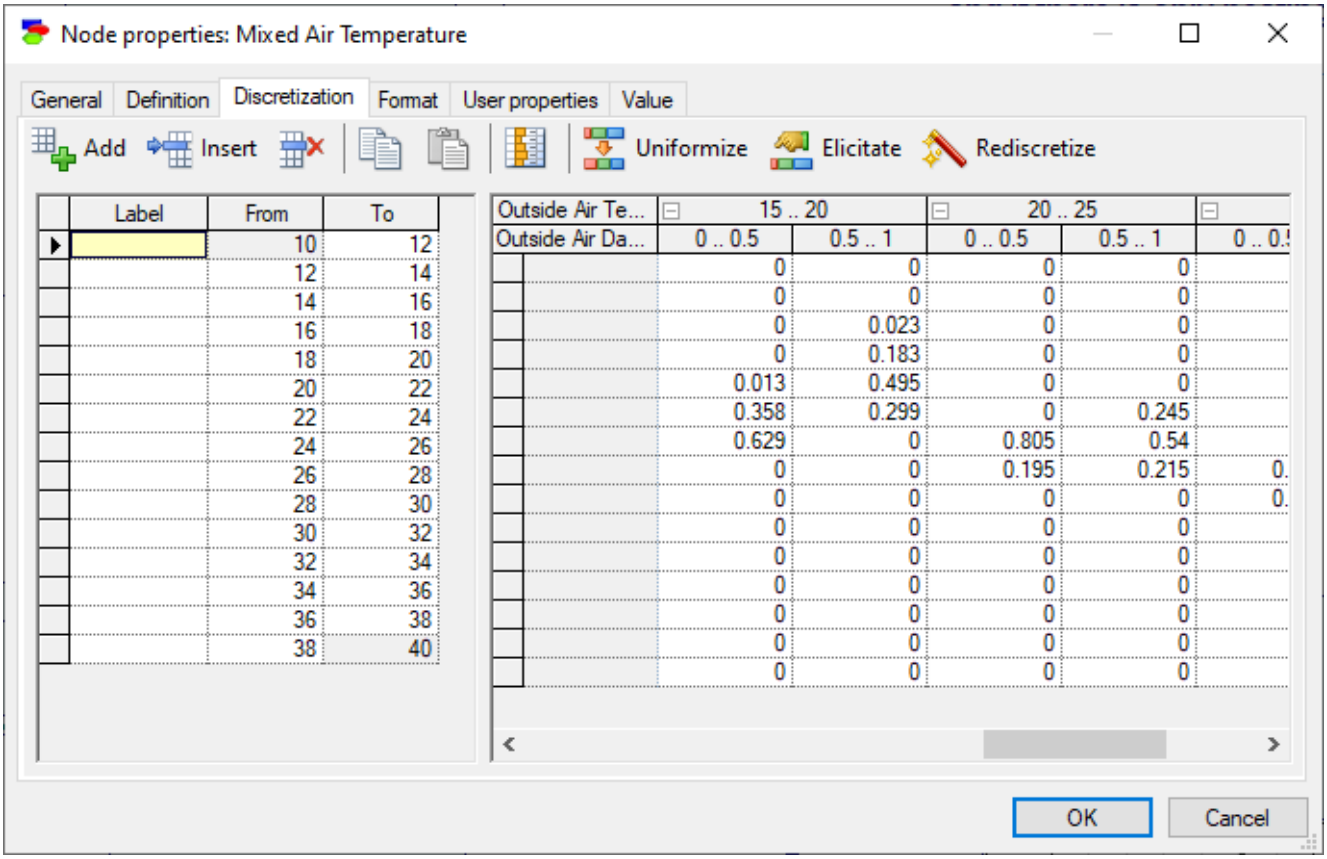
A continuous node in GeNIe is not bound by any constraints and its definition can be a general equation, including any function. Here is the definition of a node *Tma* (*Mixed Air Temperature*):



Every continuous node in GeNIe has a *Discretization* tab. The *Discretization* tab specifies how the variable should be discretized for the purpose of inference. Here is a possible discretization for the node *Tma*:



Clicking on the button *Rediscretize* ( ) invokes a simple and local forward sampling algorithm that, based on the discretization of the node and the discretizations of its parents, derives the discrete conditional probability distribution for the node. Here is a distribution derived for the node *Tma*:



Label	From	To
	10	12
	12	14
	14	16
	16	18
	18	20
	20	22
	22	24
	24	26
	26	28
	28	30
	30	32
	32	34
	34	36
	36	38
	38	40

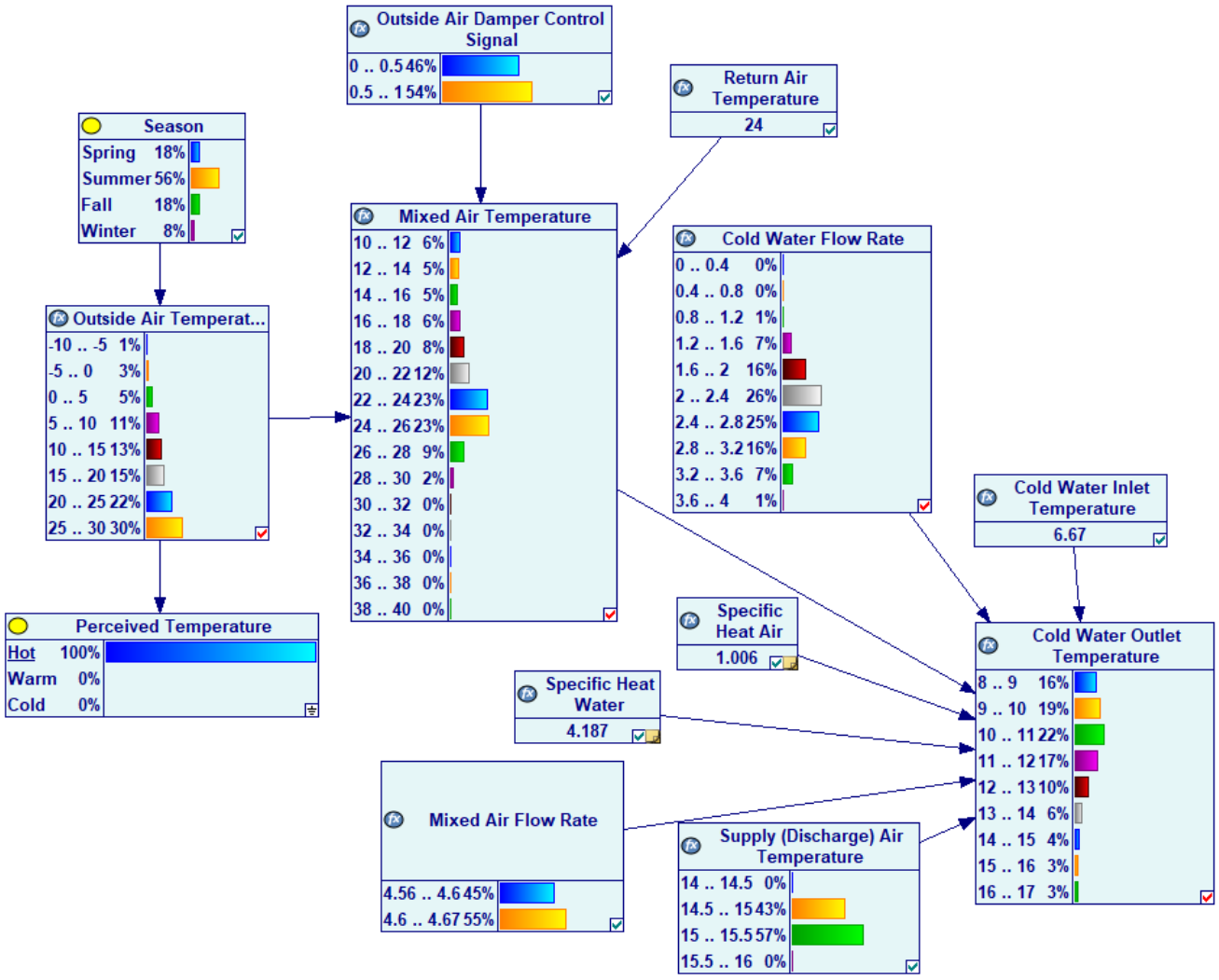
Outside Air Te...	15 .. 20		20 .. 25		
Outside Air Da...	0 .. 0.5	0.5 .. 1	0 .. 0.5	0.5 .. 1	0 .. 0.!
	0	0	0	0	
	0	0	0	0	
	0	0.023	0	0	
	0	0.183	0	0	
	0.013	0.495	0	0	
	0.358	0.299	0	0.245	
	0.629	0	0.805	0.54	
	0	0	0.195	0.215	0.
	0	0	0	0	0.
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	

It may happen in the process of auto-discretization that a warning appears in the *Output window* of the following type:

```
Discretization problem in node Tma:
Underflow samples: 3000, min=-9.80332 loBound=10
Total valid samples: 13000 of 16000
CPT configs with no valid samples: 1 of 16
```

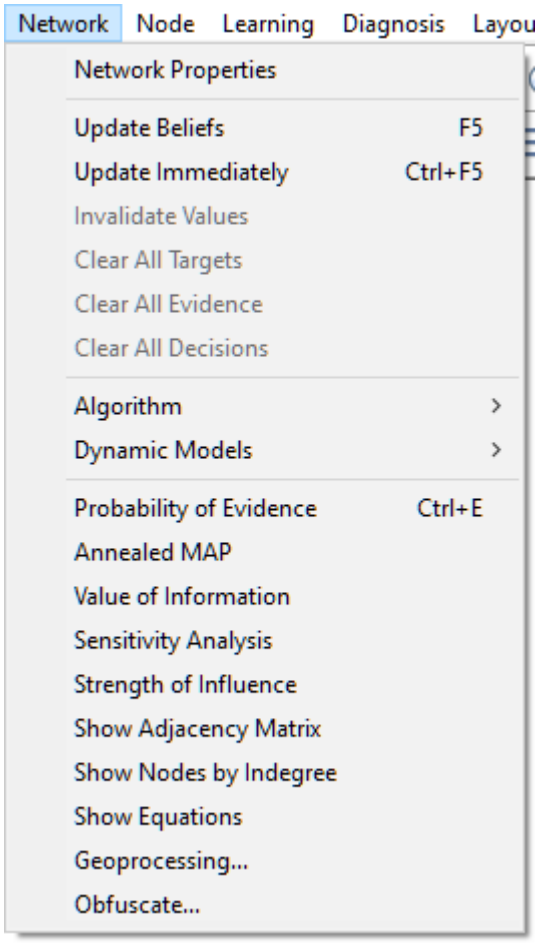
Generally, there is no need to worry about this message and the fact that there were underflow or overflow samples has minimal impact on the discretized network. Essentially, when performing discretization, SMILE generates samples from the domains of the parent nodes. Since the child node is specified by an equation, sometimes such samples fall outside of the domain of the child node. This is when underflow/overflow happens. One can observe this in continuous network when the Reject out-of-bound and invalid samples flag is checked (in *Network Properties*, *Inference* tab). Sample rejection in forward sampling (both inference in continuous networks and in discretization amounts to forward sampling) is just fine. So, our advice is that when the number of invalid/overflow samples is large, please check if the domain of the variable is not too tight. If the number of invalid/overflow samples is small and the domain bounds are reasonable, we suggest that this message is ignored.

Consider the following hybrid network (included among the example networks as `Heat Equations Autodiscretized Hybrid.xdsl`) in which an observation has been made in the node *Perceived Temperature*. Because the node *Perceived Temperature* has a parent, GeNIe invokes the autodiscretization algorithm. Please note that each of the continuous variables has been discretized. The network behaves like a discrete Bayesian network.

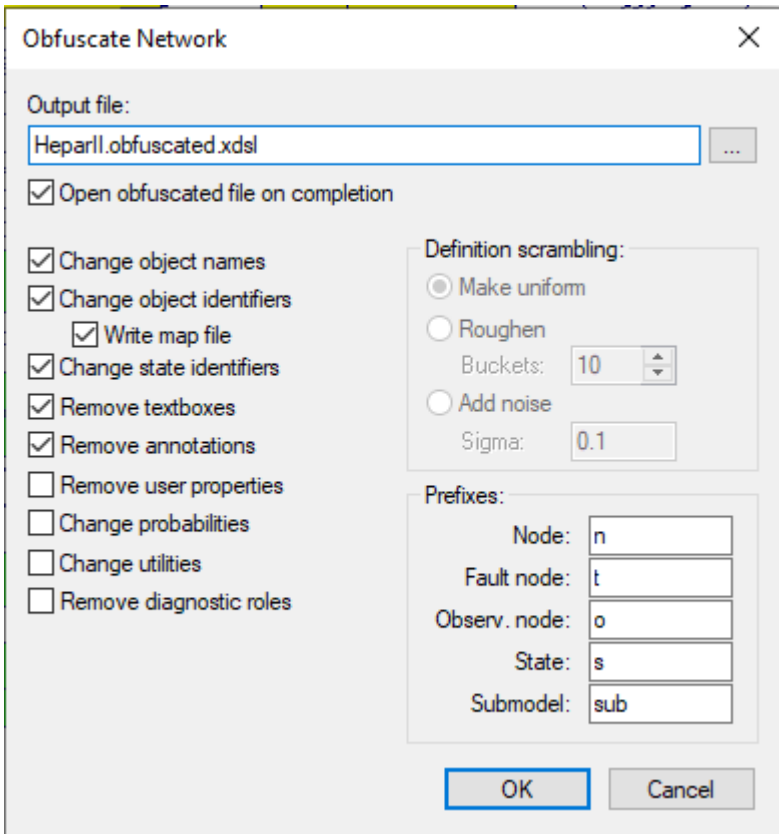

Outside Air Damper Control Signal
0 .. 0.5 46%
0.5 .. 1 54%
Return Air Temperature
24
Season
Spring 18%
Summer 56%
Fall 18%
Winter 8%
Mixed Air Temperature
10 .. 12 6%
12 .. 14 5%
14 .. 16 5%
16 .. 18 6%
18 .. 20 8%
20 .. 22 12%
22 .. 24 23%
24 .. 26 23%
26 .. 28 9%
28 .. 30 2%
30 .. 32 0%
32 .. 34 0%
34 .. 36 0%
36 .. 38 0%
38 .. 40 0%
Cold Water Flow Rate
0 .. 0.4 0%
0.4 .. 0.8 0%
0.8 .. 1.2 1%
1.2 .. 1.6 7%
1.6 .. 2 16%
2 .. 2.4 26%
2.4 .. 2.8 25%
2.8 .. 3.2 16%
3.2 .. 3.6 7%
3.6 .. 4 1%
Outside Air Temperat...
-10 .. -5 1%
-5 .. 0 3%
0 .. 5 5%
5 .. 10 11%
10 .. 15 13%
15 .. 20 15%
20 .. 25 22%
25 .. 30 30%
Cold Water Inlet Temperature
6.67
Perceived Temperature
Hot 100%
Warm 0%
Cold 0%
Specific Heat Air
1.006
Specific Heat Water
4.187
Cold Water Outlet Temperature
8 .. 9 16%
9 .. 10 19%
10 .. 11 22%
11 .. 12 17%
12 .. 13 10%
13 .. 14 6%
14 .. 15 4%
15 .. 16 3%
16 .. 17 3%
Mixed Air Flow Rate
4.56 .. 4.6 45%
4.6 .. 4.67 55%
Supply (Discharge) Air Temperature
14 .. 14.5 0%
14.5 .. 15 43%
15 .. 15.5 57%
15.5 .. 16 0%

5.8 Obfuscation

Sometimes we may want to share a model with others but want to preserve the intellectual property that goes into building it. This happens, for example, when sharing your models with our Support Department. The *Obfuscate...* command can be used to create a new network from an existing network with an option to change various model properties while maintaining the original structure of the model.



The command invokes the following dialog



Output file allows the user to choose the location for the obfuscated network. *Browse* (...) button is convenient in finding room for the file on the local disk.

Obfuscation options

The check boxes on the left-hand side fulfill the following function:

- *Open obfuscated file on completion* leads to opening the obfuscated network in GeNIe.
- *Change object names* changes the names of objects in the model. These are typically names of nodes, submodels, and the network.
- *Change object identifiers* changes object identifiers in the model. These are typically identifiers of nodes, submodels, and the network. Because changing the identifiers may lead to communication problems with the recipient of the obfuscated network, *Write map file* option creates a file that maps the original with the scrambled identifiers.
- *Change state identifiers* changes the state names within discrete nodes in the model.
- *Remove textboxes* leads to removal of all text boxes that contains notes documenting the model.

- *Remove annotations* removes all annotations from the model (these are on arcs, nodes, node states, individual probabilities).

- *Remove user properties* removes all user properties defined for the model and its nodes.

- *Change probabilities* scrambles the numerical probabilities in the definitions of *Chance* nodes. See below for your choices of scrambling.

- *Change utilities* scrambles all numerical utilities in the definitions of *Utility* nodes. See below for your choices of scrambling.

- *Remove diagnostic attributes* removes any diagnostic properties defined in the model.

Definition scrambling

When at least one of the two options, *Change probabilities* or *Change utilities*, is checked, obfuscation involves scrambling the numerical parameters. There are three possibilities here:

Make uniform leads to replacing all probability distributions with uniform distribution

Roughen leads to rounding each of the parameters. Precision of the rounding is controlled by the number of Buckets parameter. When it's value is 10 (the default), all probabilities are rounded to the first digit after the decimal point.

Add noise leads to distorting all parameter with random noise. The process is controlled by the parameter Sigma, which is the standard deviation of the Normal distribution from which the noise is drawn. Adding noise is conducted by first transforming the parameter to odds, adding a random number from the Normal(0,sigma) distribution, and transforming the parameter back to probability. The entire distribution is re-normalized after this process to ensure that the sum of all probabilities is 1.0.

Prefixes

When a model is obfuscated, its elements will be assigned artificial names. You can control to some degree what these names will look like through specifying prefixes. Here are the default prefixes:

Node: n

Target node: t

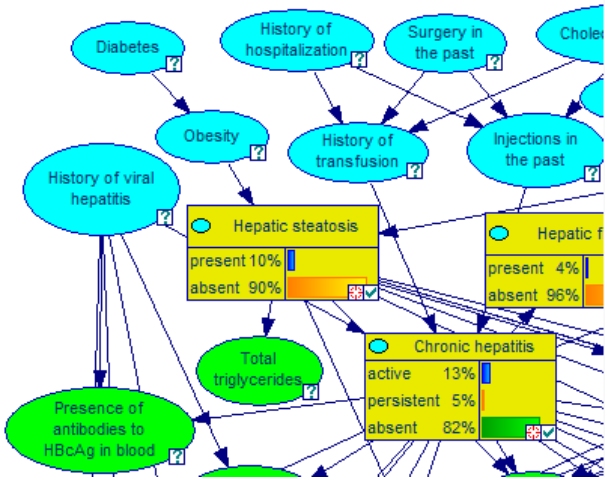
Observ. node: o

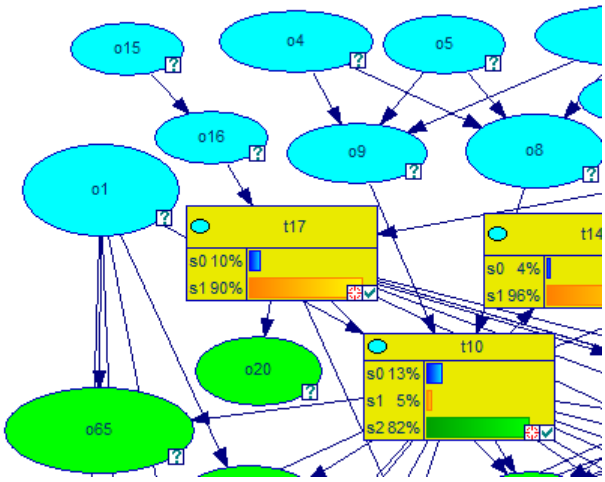
State: s

Submodel: sub

Example

The following screen shots show the Hepar II network before (left) and after (right) obfuscation with the default settings.





Definitions of the node *Chronic hepatitis* before (left) and after (right) obfuscation with the default settings are shown below:

History of transf...	present			
History of viral ...	present		absent	
Injections in th...	present	absent	present	absent
active	0.20942408	0.46153846	0.06	0.13043478
persistent	0.0052356	0.30769231	0.06	0.04347826
absent	0.78534031	0.23076923	0.88	0.82608696

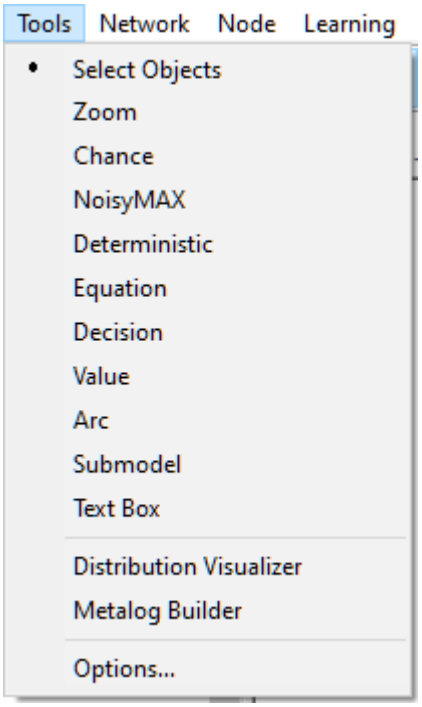
o9	s0			
o1	s0		s1	
o8	s0	s1	s0	s1
s0	0.20942408	0.46153846	0.06	0.13043478
s1	0.0052356	0.30769231	0.06	0.04347826
s2	0.78534031	0.23076923	0.88	0.82608696

Please note that the default settings do not scramble the numerical parameters.

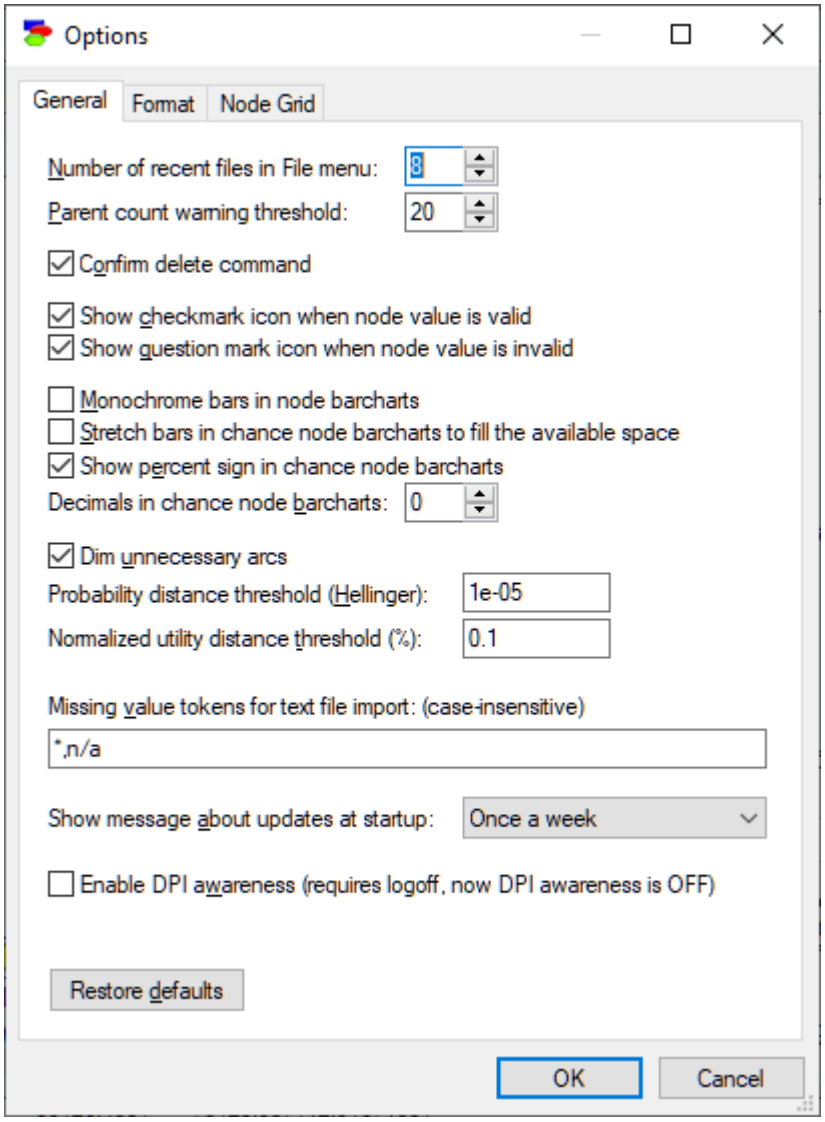
The obfuscation process creates two files on the disk: The obfuscated model (suffix *.obfuscated.xdsl*) and the mapping file (suffix *.obfuscated.map*). The latter is a text file containing a list of pairs of node IDs, the original and the obfuscated ID. When sharing the obfuscated model with somebody (e.g., with BayesFusion support folks), please do not send the map file, just the obfuscated model, as sending the mapping file along defies the purpose of obfuscation.

5.9 Program options

To change program options, select *Options*... from the *Tools Menu*



The following dialog appears



General tab

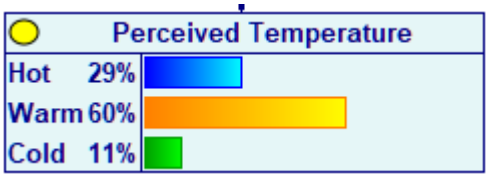
Number of recent files in File Menu determines the number of file names and locations in the *File Menu* saved between sessions.

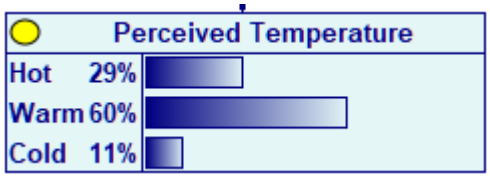
Parent count warning threshold is an important practical setting, especially for beginning modelers. As you increase the number of parents of a node, the node's CPT grows exponentially. With 10 binary parents, the CPT contains 1,024 columns, with 20 binary parents, this number is 1,048,576. As the number of parents grows, at some point, this will exhaust all available computer memory. This setting determines when a model author should be given a warning that the number of parents is getting out of hands.

Confirm delete command, when set, issues a warning when the user issues a *Delete* command.

Show checkmark icon when node value is valid and *Show question mark icon when node value is invalid* control two important node status icons.

Monochrome bars in node barcharts changes the colorful bars displaying the probabilities of various outcomes into monochrome bars. Some users find it more natural, especially given that GeNIe chooses the colors automatically and the colors of bars in a node are not related in any way to the same colors in other nodes. Compare the following two: color bars (left) and monochrome bars (right):





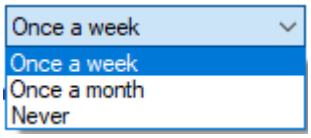
Stretch bars in chance nodes barcharts to fill the available space controls the appearance of the node bar chart view.

Show percent signs in chance node barcharts and *Decimals in chance node barcharts* control the display of numerical posterior probabilities in the node bar chart view.

Dim unnecessary arcs is an important modeling tool. When the probability distributions in a CPT are such that a parent's state makes no difference, the arc between the parent and the node is not necessary. This is often the case when building a model - because GeNIe makes sure that a model is always correct, it puts uniform distributions in all columns of the node's CPTs. Whenever an arc is added, distributions are copied and are identical for all states of the parent node. Because unnecessary arcs are dimmed, it is clearly visible which arcs still need modeling attention. Without this cue, it is easy to overlook node definitions. Indeed, when we converted some sizeable Bayesian network models to GeNIe format, we typically noticed several dimmed arcs, which means that the modelers have overseen them when eliciting parameters. Real numbers are rarely identical, so the *Probability distance threshold (Hellinger)* is a setting that allows for approximate equality of distributions. When two distributions are equal up to the threshold, they are considered equal. Utilities are not distributions, so when they are compared, the second setting (*Normalized utility distance threshold (%)*) is used. When two utilities differ less than the indicated percentage, they are considered equal.

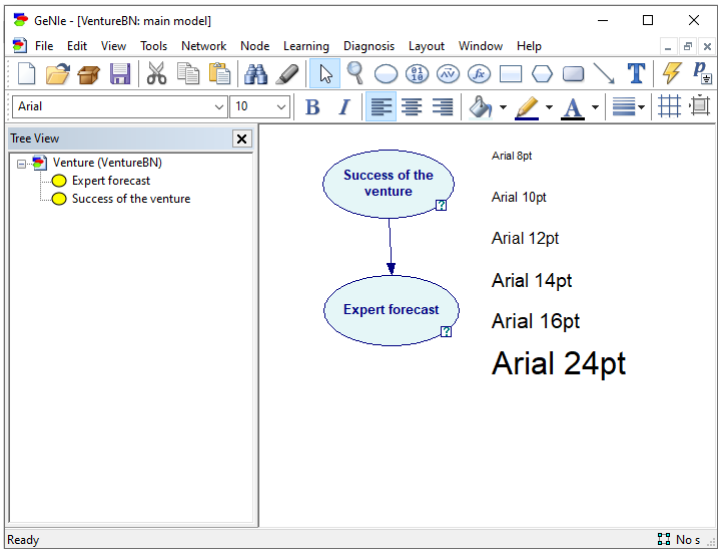
Missing value tokens for text file import: (case-insensitive) (default value "*,n/a") allows for specifying the text that will be recognized in input files as denoting missing values.

Show message about updates at startup controls the frequency with which GeNIe informs its users about availability of updates. Information about updates appears always in the *About GeNIe* dialog but also as a message in the *Output window*, which is what this option controls. Frequency of messages can be set to one of three: *Once a week*, *Once a month*, and *Never*.

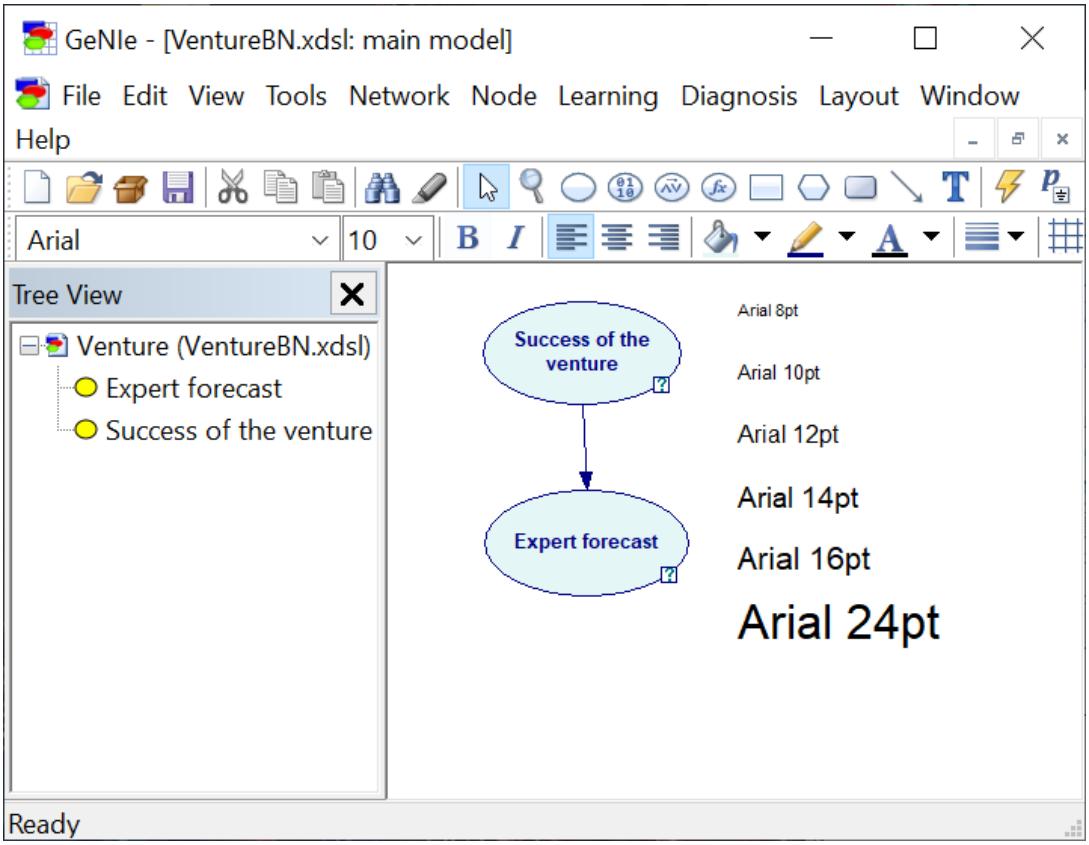


Windows allows for changing the size of text in its *Display Preferences* panel. The size is set to its default "100%" value unless you have high DPI (dots per inch) display. With the value greater than 100% set in Windows, you can change GeNIe's appearance, as shown in the screen shots below, using the *Enable DPI awareness* option.

To get an idea of the impact of this option on the display, please look at the following model with Windows 10 at 150% scaling, with *Enable DPI awareness* OFF:



and with *Enable DPI awareness* ON:

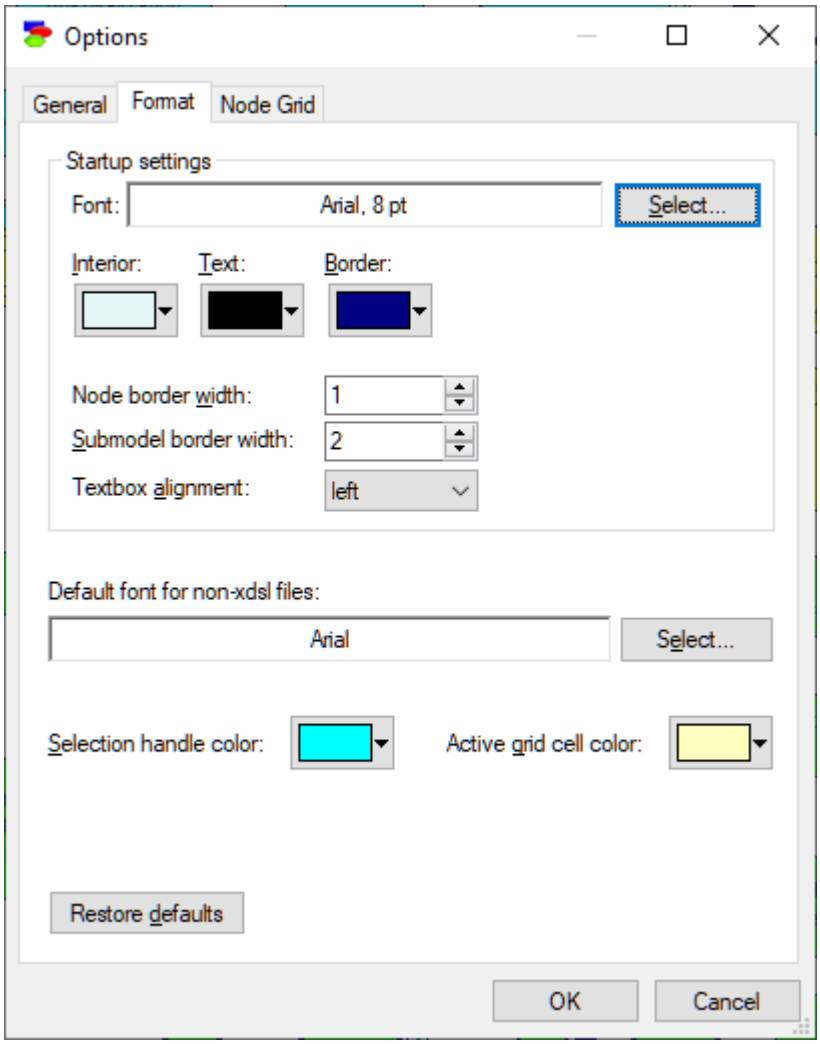


Clearly, perceived font size and clarity differ. Please note that relative size of the fonts in GeNIe's *Graph View* are unaffected. We leave it up to the user whether to set this option or not. Our preference is for keeping this option OFF.

Please note that the GeNIe's graphical network view always assumes 96 DPI. Also note that our experience with Windows 10 indicates that changing *Enable DPI awareness* option and restarting the program does not always have the desired effect, hence we recommend logging off the current Windows session before checking the effect of *Enable DPI Awareness* on GeNIe.

Format tab

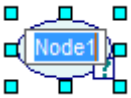
The *Format* tab allows for setting the default format for objects in the *Graph View*. The format can be changed later but here is where we set the initial values.



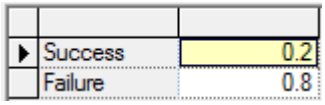
The top part of the *Format* tab concerns nodes, submodels, and text boxes. Model elements in most of the screen shots in this document use the default settings pictured in the above *Format* dialog.

XDSL files contain font specifications. *Default font for non-xdsl files* is for all those file formats that do not specify the font explicitly.

Selection handle color determines the color of node handles when nodes are selected, like the eight handles around the *Chance* node below.

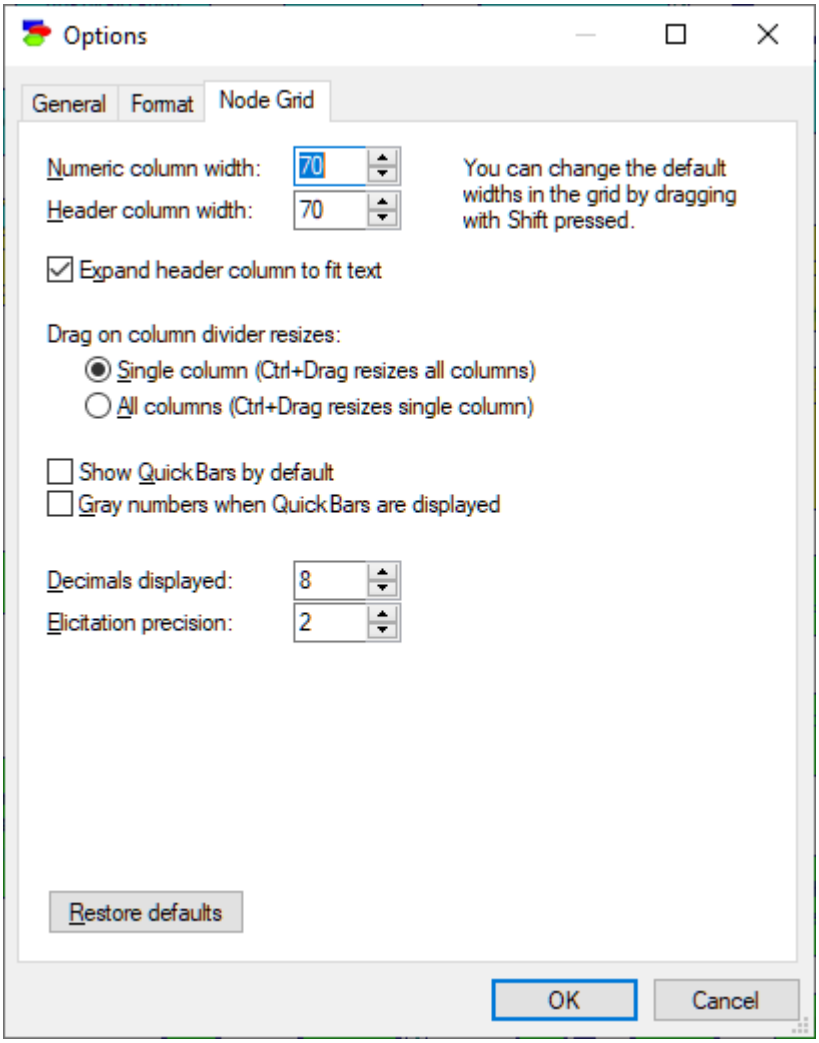


Active grid cell color is the color or a CPT cell that is active, i.e., that is selected by the cursor, like the cell for the state *Success* below.



Node Grid tab

The *Node Grid* tab contains the settings of the CPT grid in the node *Definition* tab.



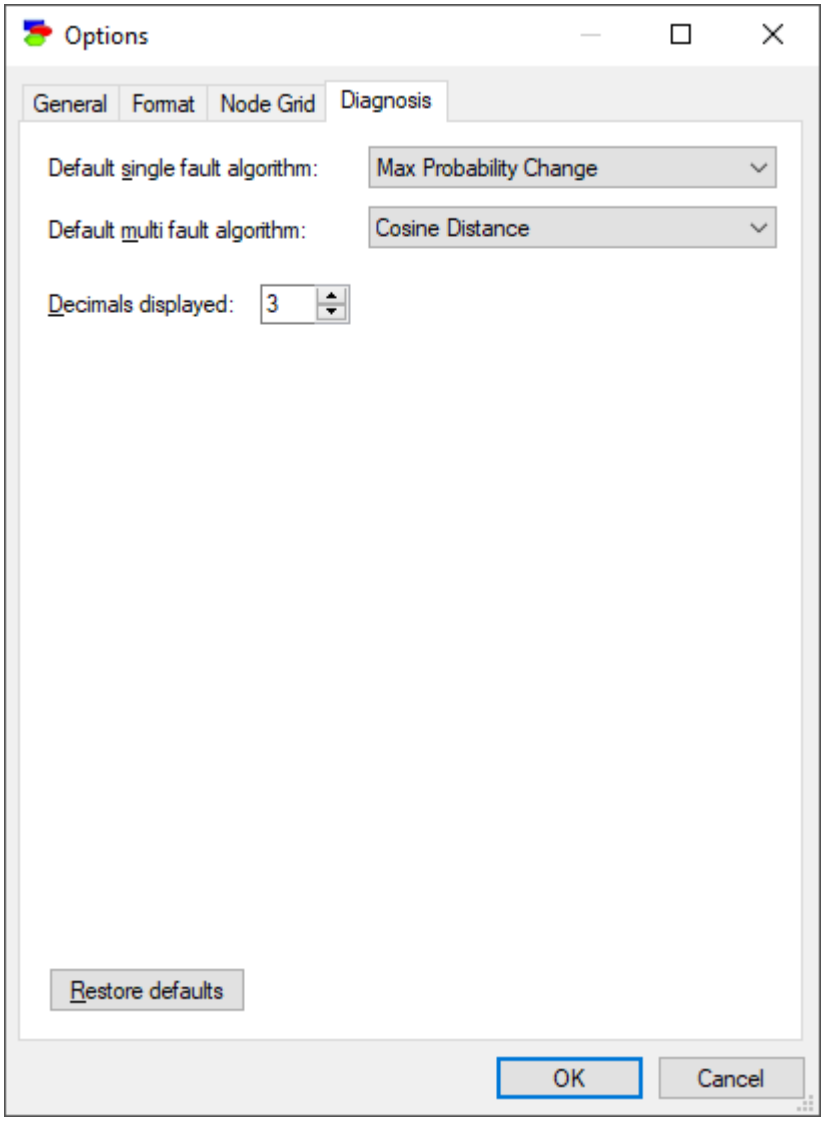
Most of the settings control the physical appearance of the node *Definition* grid and are self-explanatory.

Decimals displayed (default 6) control precision of numerical probabilities in the CPTs.

Elicitation precision (default 2) controls the precision of graphical methods of probability elicitation (pie chart and bar chart).

Diagnosis tab

The *Diagnosis* tab allows for choosing three default program parameters:



Default single fault algorithm for diagnostic value of information computation.

Default multi fault algorithm for diagnostic value of information computation.

Decimals displayed when displaying the marginal probability of faults and also diagnostic values.

5.10 Keyboard shortcuts

File operations

CTRL+N: Create a new network

CTRL+O: Open an existing network

CTRL+SHIFT+O: Open a network from BayesBox

CTRL+S: Save the currently active file to disk

CTRL+P: Print the current network

CTRL+W: Close the current network

Layout of elements in the *Graph View*

CTRL+G: Toggle display of grid lines

CTRL+L: Highlight selected model elements

CTRL+SHIFT+G: Toggle auto alignment of elements to grid

F8: View nodes as bar charts

SHIFT+F8: View nodes as icons

Finding / selecting nodes / spreadsheets

CTRL+F: Find a node

CTRL+A: Select all elements

CTRL+SHIFT+A: Select all nodes

CTRL+ALT+A: Select all submodels

SHIFT+F3: Find next/previous (data spreadsheet)

CTRL+H: Replace (data spreadsheet)

Show / hide windows

CTRL+T: Toggle display of the *Tree View* window

CTRL+U: Toggle display of the *Output* window

CTRL+ALT+C: Toggle display of *Case Manager* pane

CTRL+ALT+B: Toggle display of *Temporal Beliefs* pane

F7: Show *Diagnosis* window

F11: View network full-screen (hides all views, menus, and toolbars)

F12: Zoom to fit window

*CTRL+F12 / CTRL +**: Zoom to 100%

CTRL+PLUS: Zoom in

CTRL+ MINUS: Zoom out

Updating the network

F5: Update beliefs

CTRL+F5: Toggle *Update Immediately* switch

CTRL+F8: View nodes as bar charts and update beliefs

CTRL+E: Calculate probability of evidence

Editing

F2: Rename object

CTRL+B: Bold font

CTRL+I: Italic font

CTRL+C: Copy

CTRL+V: Paste

CTRL+X: Cut

Del: Delete

This page is intentionally left blank.

Using GeNIe

6 Using GeNIe

6.1 Introduction

This section presents various modules of GeNIe from the point of view of their function. It is an alternative view to the one presented in the previous section, which focused on GeNIe's building blocks.

6.2 Bayesian networks

6.2.1 Building a Bayesian network

Building a Bayesian network in GeNIe is demonstrated step for step in section *Hello GeNIe!*

6.2.2 Structural analysis

One of the important elements of probabilistic modeling is the ability of directed probabilistic graphs to represent the causal structure of the modeled domain. The structure itself is very valuable and is an important source of insight. Models built by means of GeNIe can be examined structurally. An important element of this analysis is the structure itself, viewing the strengths of influences and pathways through the graph. We describe viewing the strengths of influences in the *Strength of influences* section. This section describes tools for the analysis of the graph structure.

Dimming unnecessary arcs

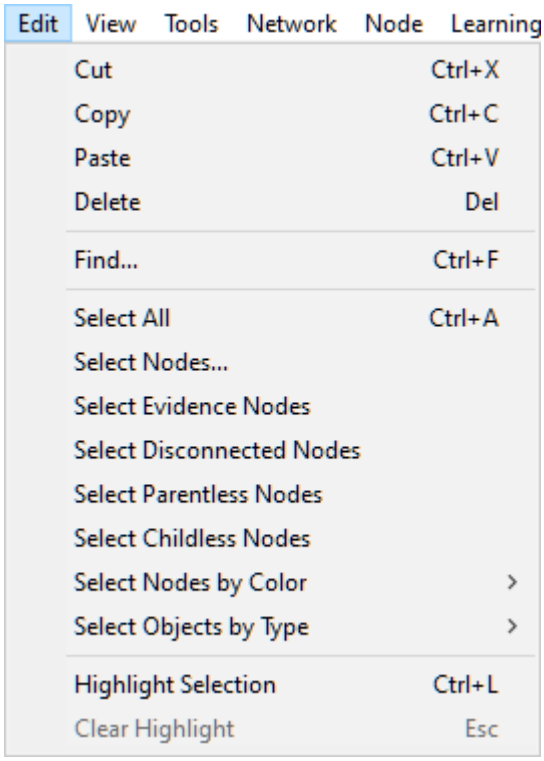
Dimming unnecessary arcs, discussed earlier in this manual, is a simple but important modeling tool that allows for finding one class of modeling errors, errors of omission. When the probability distributions in a conditional probability table are such that a parent's state makes no difference, the arc between the parent and the node is not necessary. This is often the case when building a model - because GeNIe makes sure that a model is always correct (this was our important design principle), it puts uniform distributions in all columns of the node's conditional probability table. Whenever an arc is added, distributions are copied and are identical for all states of the parent node. When the model under construction is sufficiently large, it is very easy to forget about refining them. Because unnecessary arcs are dimmed, it is clearly visible which arcs still need modeling attention. Without this cue, in a sufficiently large model under construction, it would be easy to forget about refining node definitions.

Real numbers are rarely identical, so the *Probability distance threshold (Hellinger)* is a setting that allows for approximate equality of distributions. When two distributions are equal up to the threshold, they are considered equal. Utilities are not distributions, so when they are compared, the second setting (*Normalized utility distance threshold (%)*) is used. When two utilities differ less than the indicated percentage, they are considered equal. The details of these settings are described in section *Program options*, *General Tab*.

Whenever you see a dimmed arc, please have a look at the definition of the child node - chances are that it is not that the arc is unnecessary but rather that you have forgotten to define the interaction between the child node and its parents.

Locating disconnected nodes

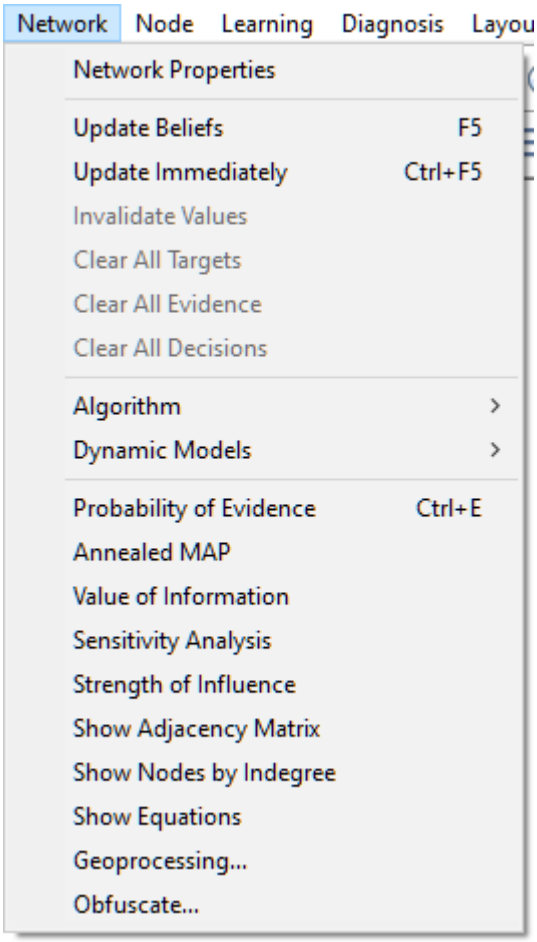
When modeling, many a user create a collection of nodes and then connect them by means of arcs. When the model under construction is sufficiently large, it is not uncommon to forget about some of the nodes and leave them unconnected to the rest of the network. Locating disconnected nodes is a functionality that helps to find such nodes so that we can give them more attention. To find disconnected nodes, please select *Select Disconnected Nodes* from the *Edit Menu*:



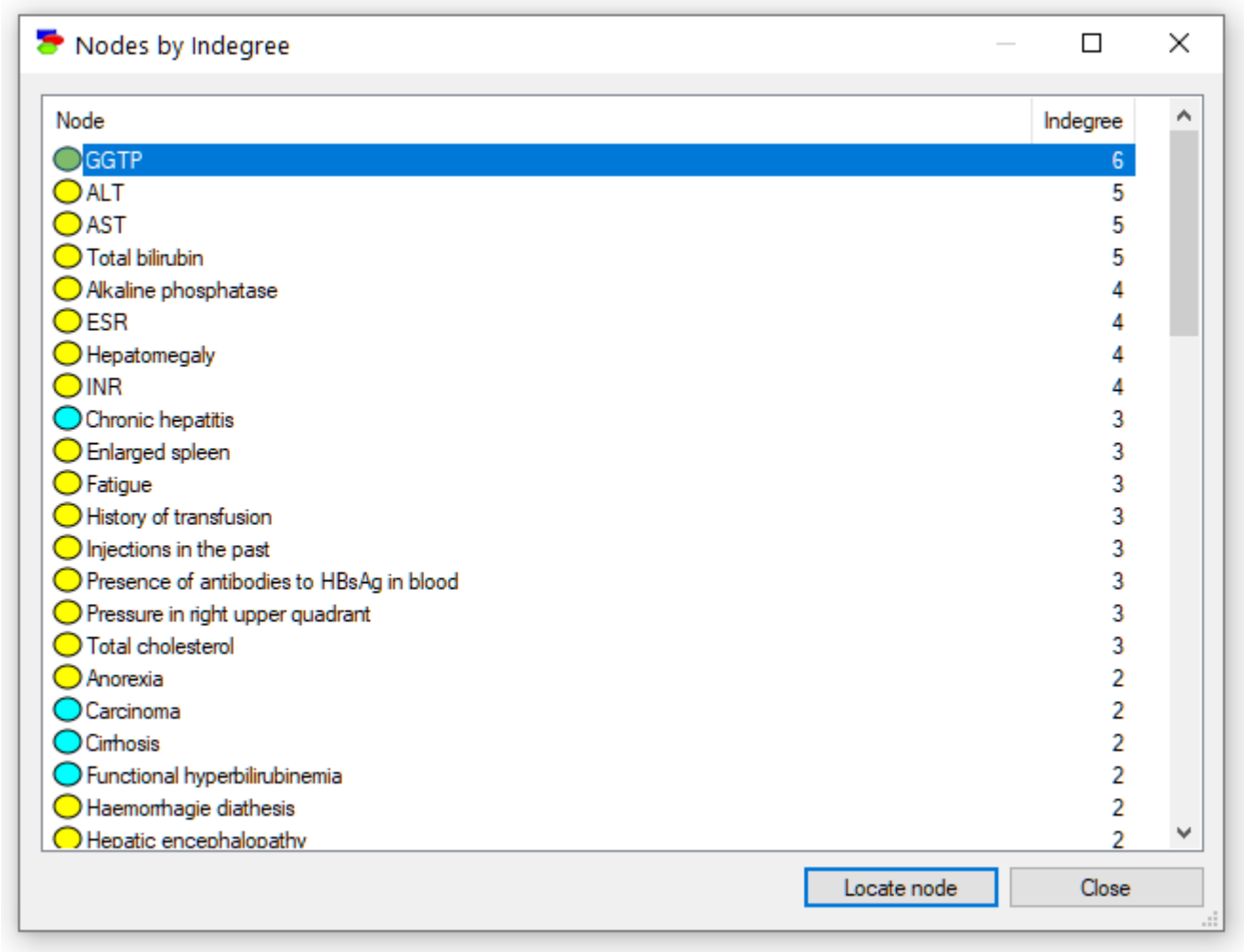
If there are no disconnected nodes in the model, this choice is going to be grayed out. Nodes selected can be subsequently highlighted (choice *Highlight Selection* or *Ctrl-L* in the *Edit Menu* or pressing the *Highlight selection* button) and easily located in the *Graph View* visually.

Finding nodes with high in-degree

Inference in Bayesian networks is worst-case NP-hard, which typically means that algorithms updating a Bayesian network will require an exponential amount of memory or computation time. One factor that influences the complexity of inference is the network connectivity. An important factor in the network connectivity is node in-degree (the number of parents of a node) because the size of the conditional probability table in a node is exponential in the number of parents of that node. GeNIe allows for finding nodes with high in-degree through the function *Show Nodes by Indegree* in the *Network Menu*.



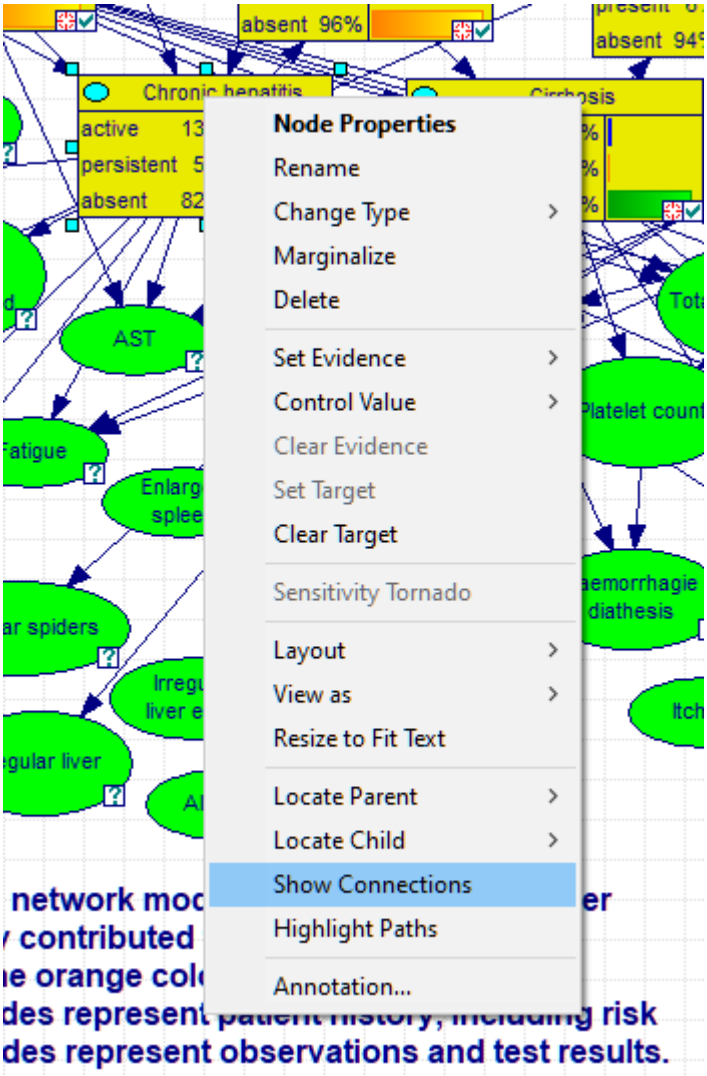
This function displays the following dialog



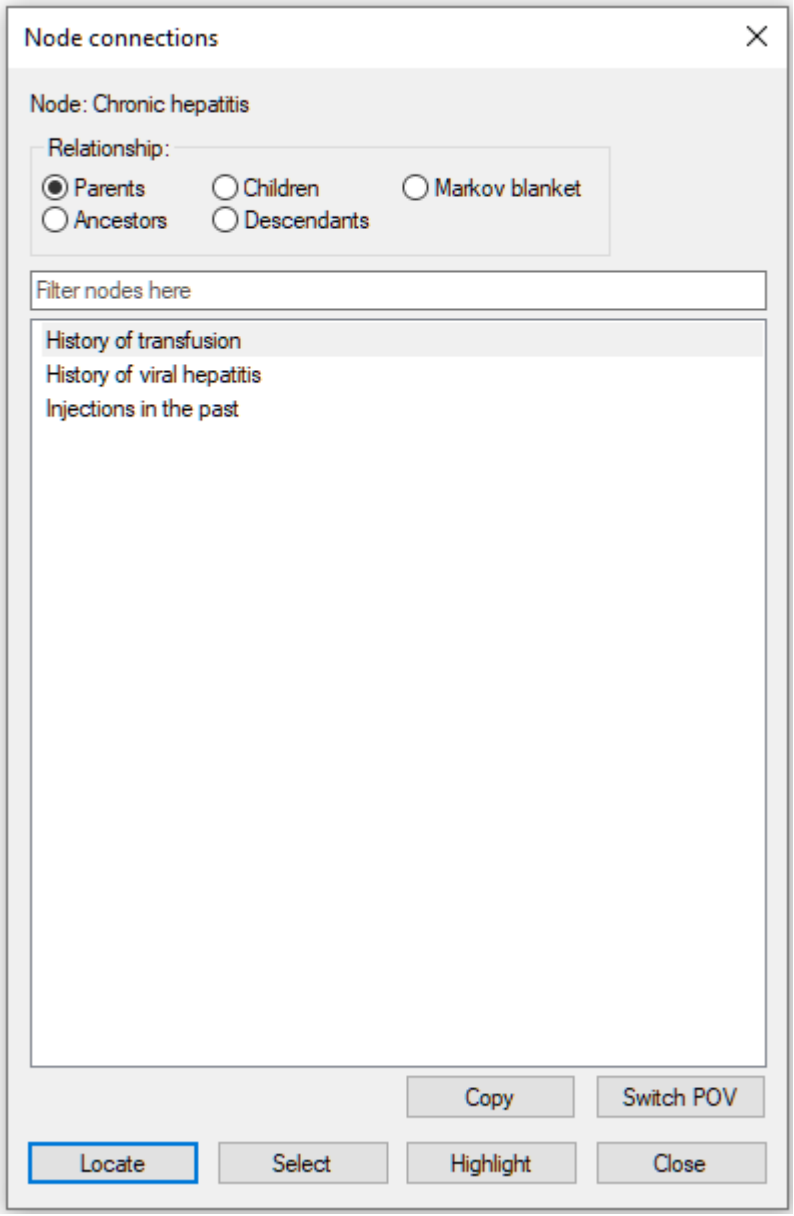
The dialogs shows a list of all nodes in the model with their in-degree. The nodes are sorted by their in-degree (from the highest to the lowest). Selecting any node on the list and pressing the *Locate node* button (or just double-clicking on the node) finds the node in the *Graph View* of the model. In case of any problems with inference, it is advisable to find those nodes that have high in-degree and to try to reduce their number of parents. Often this is possible by removing weaker connections or by a technique known as "parent divorcing," (mentioned in Section on *Computational complexity*).

Neighborhoods

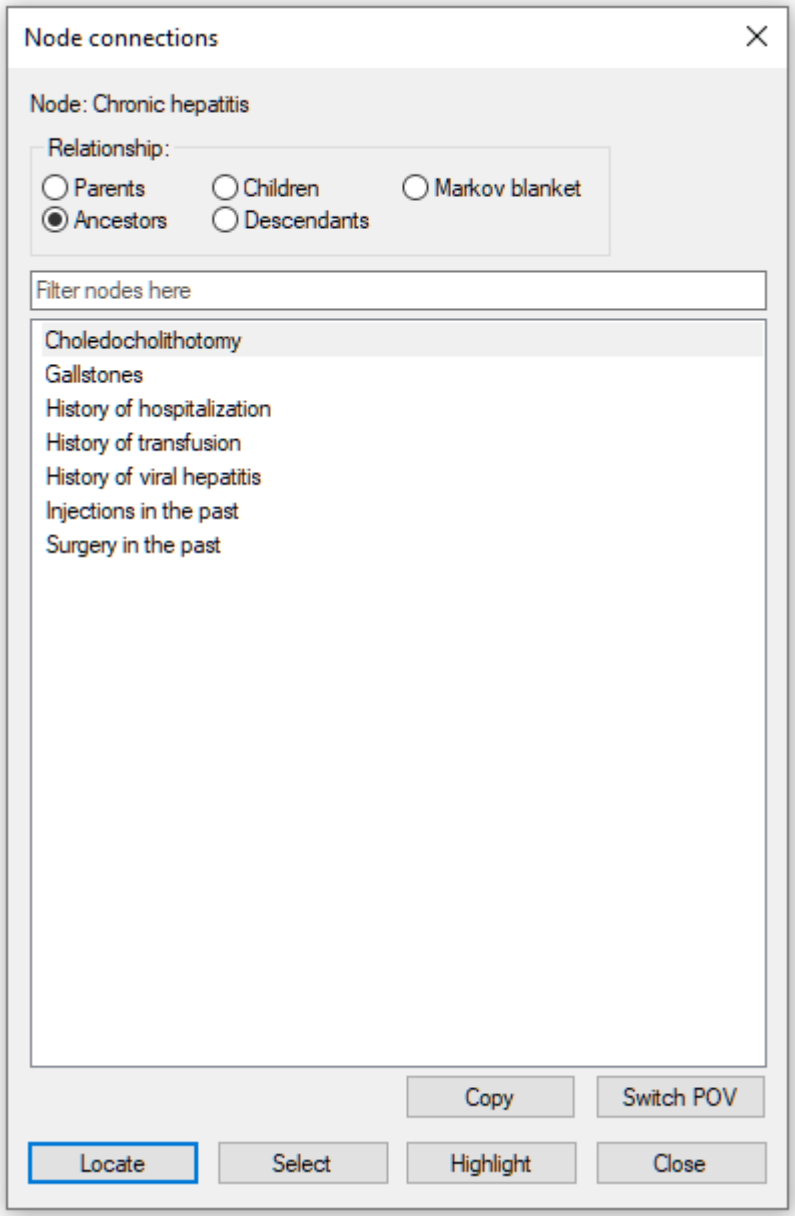
There are several useful functions that help with analysis of connections and pathways through the directed graph. One group of such functions is showing connections of a selected node. To open the connections dialog, please select *Show Connections* from the context menu of the node in question. The image below show invoking the dialog for the node *Chronic hepatitis* in the *Hepar II* network.



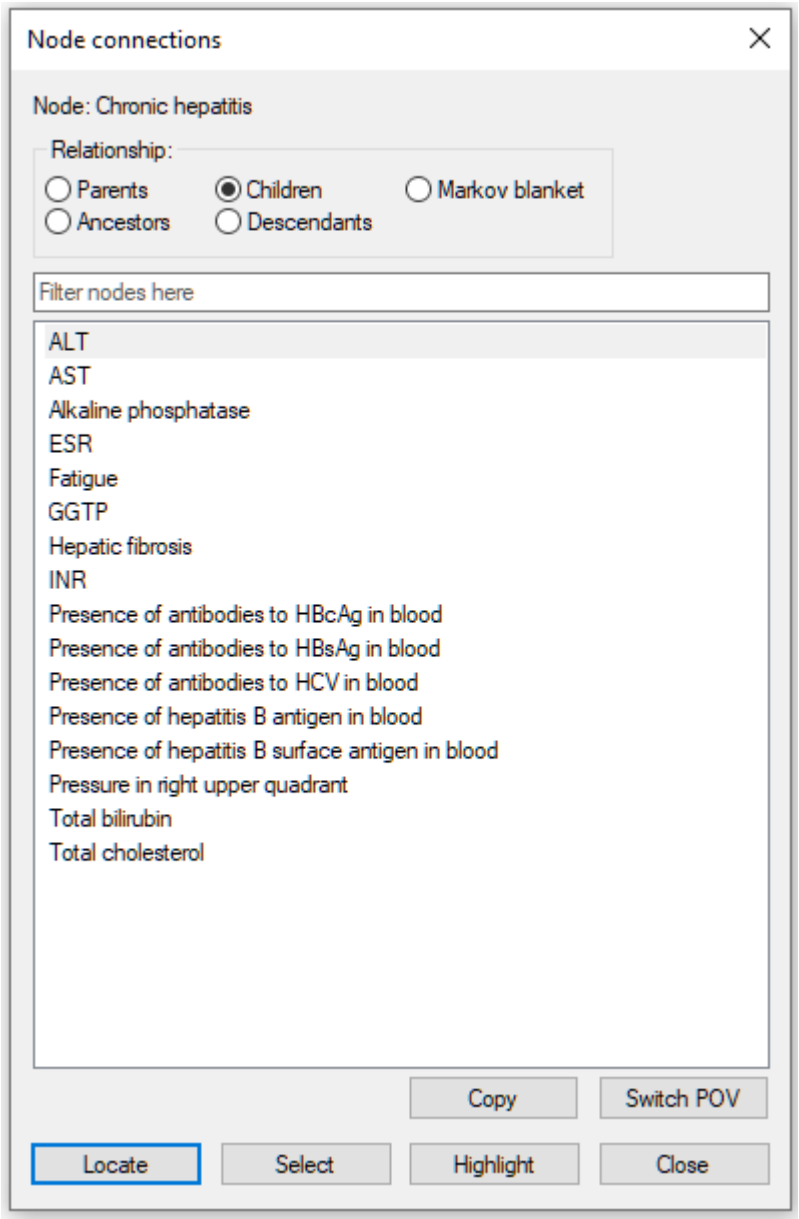
The dialog allows for selecting parents (direct predecessors in the graph), children (direct successors in the graph), family (parents and the node itself), ancestors (all predecessors in the graph), descendants (all successors in the graph), and the node's Markov blanket (the set of nodes that make the node in question independent of the other nodes in the graph). There are three parents of the node *Chronic hepatitis* in the graph.



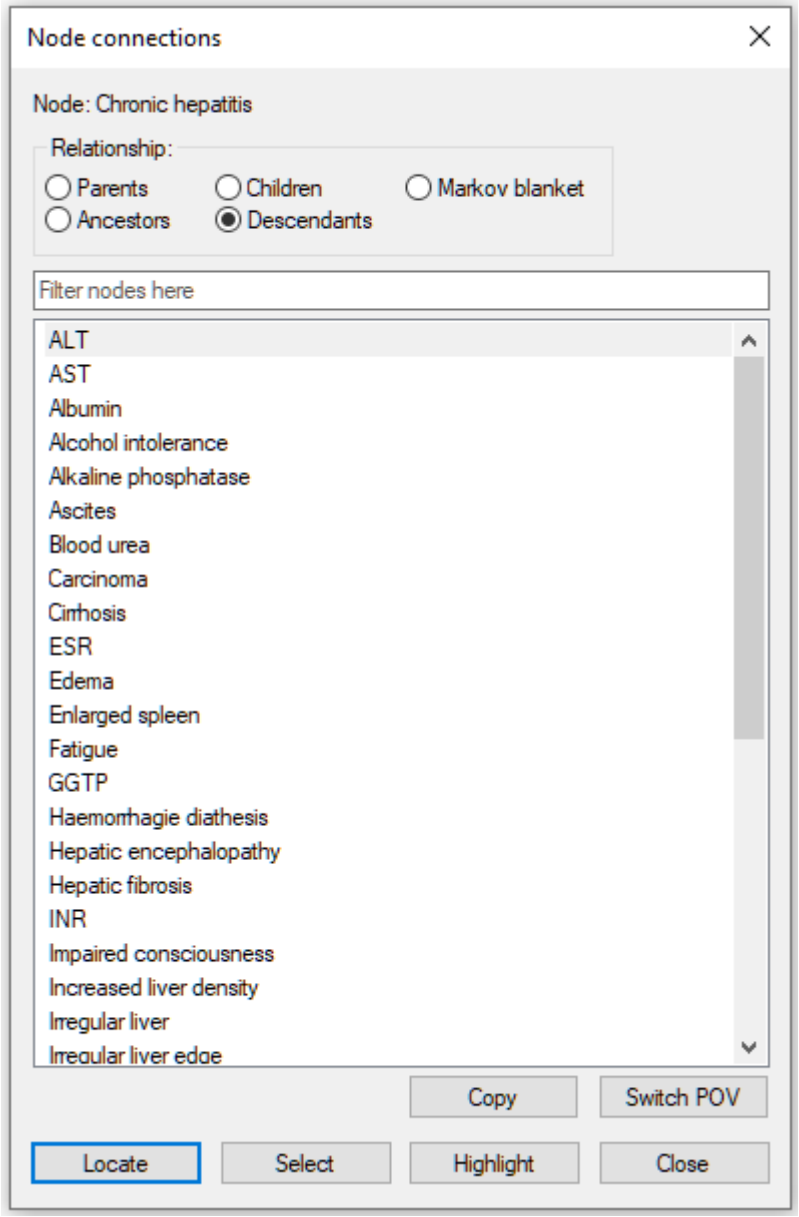
The three nodes belong also to the ancestors of the node *Chronic hepatitis*.



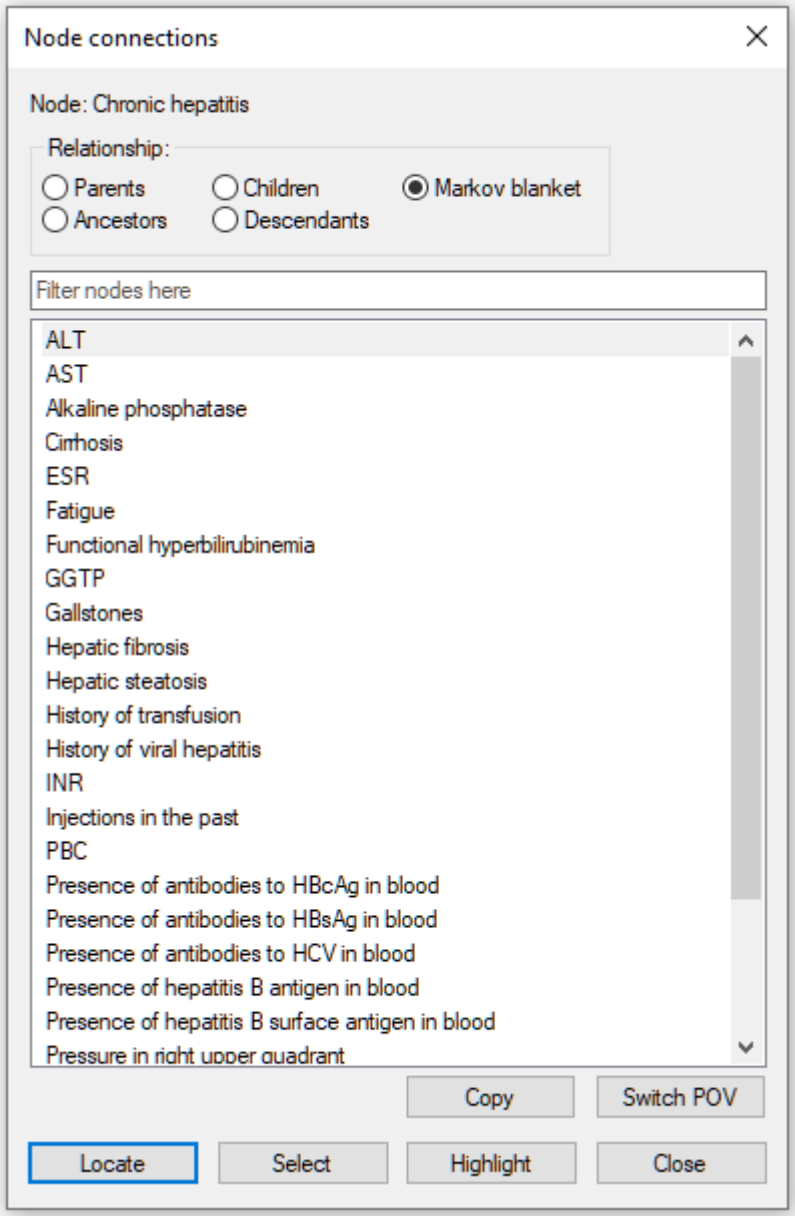
The node *Chronic hepatitis* has 16 children:



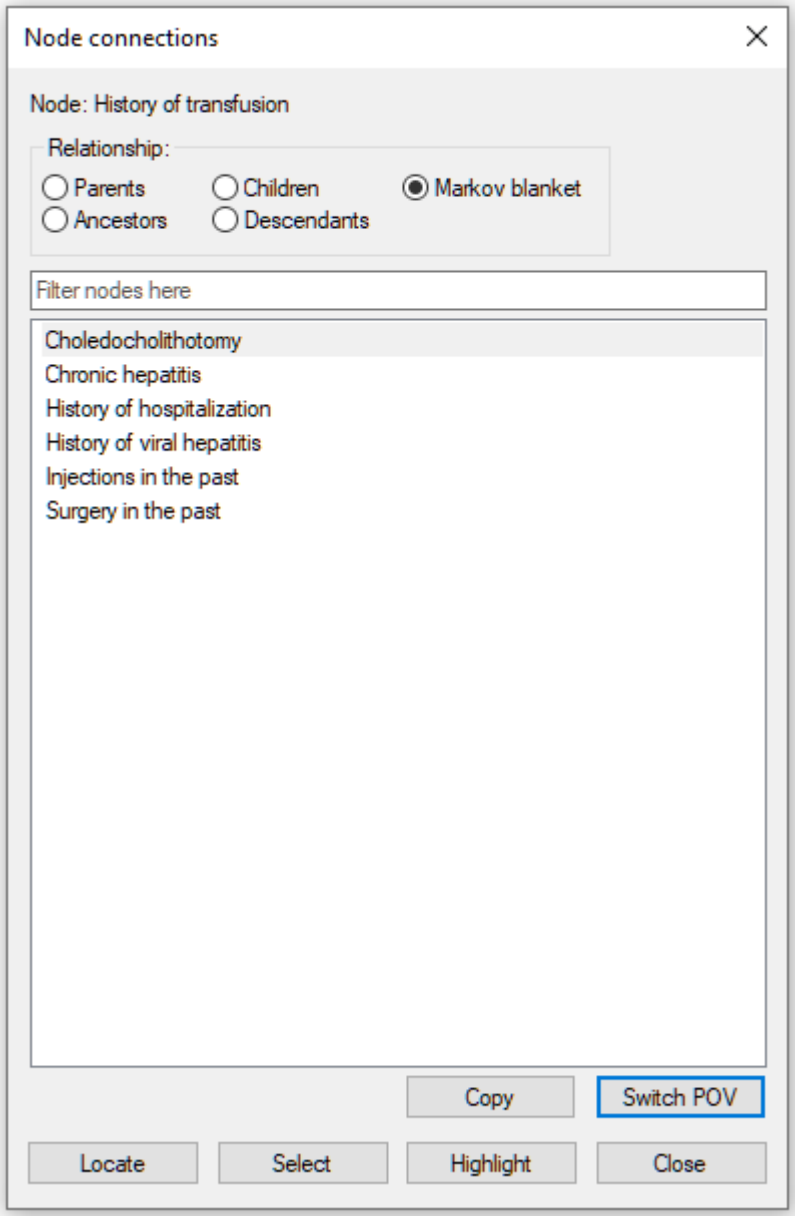
They both belong to the descendants of *Chronic hepatitis*:



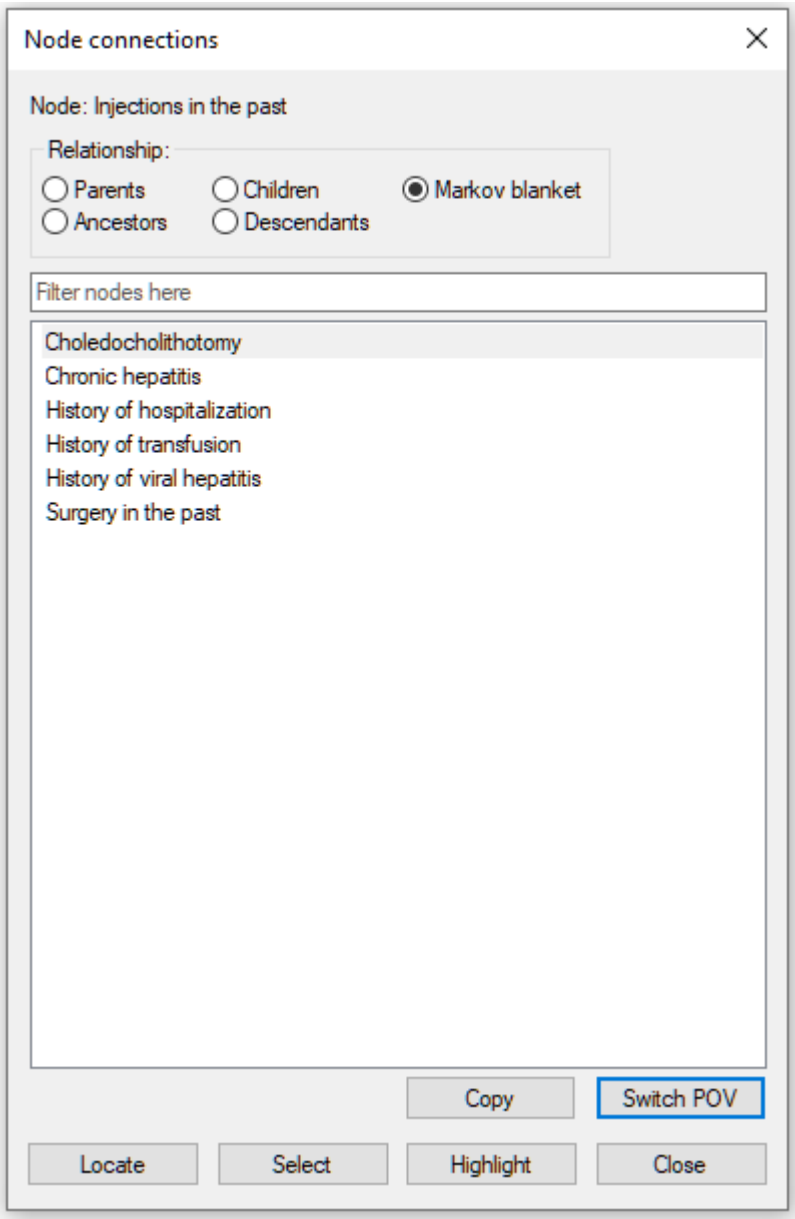
Parents, children, and the parents of those children belong to the Markov blanket of the node *Chronic hepatitis*:



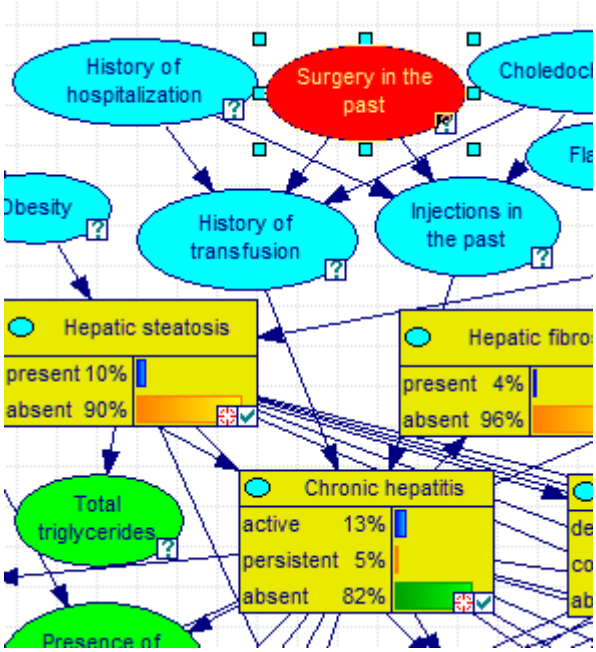
The dialog allows for traversing the graph through changing the focus of the analysis. A new focus can be chosen by selecting it from the list of nodes and pressing the *Switch POV* (*Point of View*) button in the lower-right corner of the dialog. Let us select *History of transfusion* and press the *Switch POV* button.



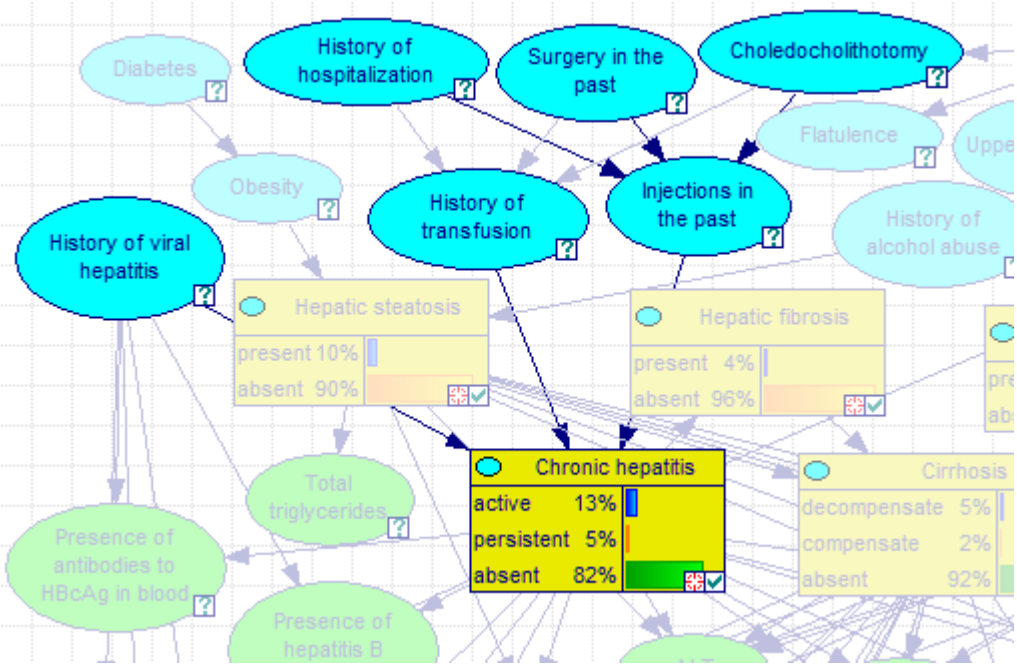
Selecting *Injections in the past* and pressing the *Switch POV* button leads to refocusing the neighborhood to the node *Injections in the past*.



Selecting any of the nodes on the list and pressing *Locate* button locates the node in the *Graph View* and flashes the node three times. Double-clicking on the selected node has the same effect. Selecting *Surgery in the past* and pressing the *Locate* button yields:

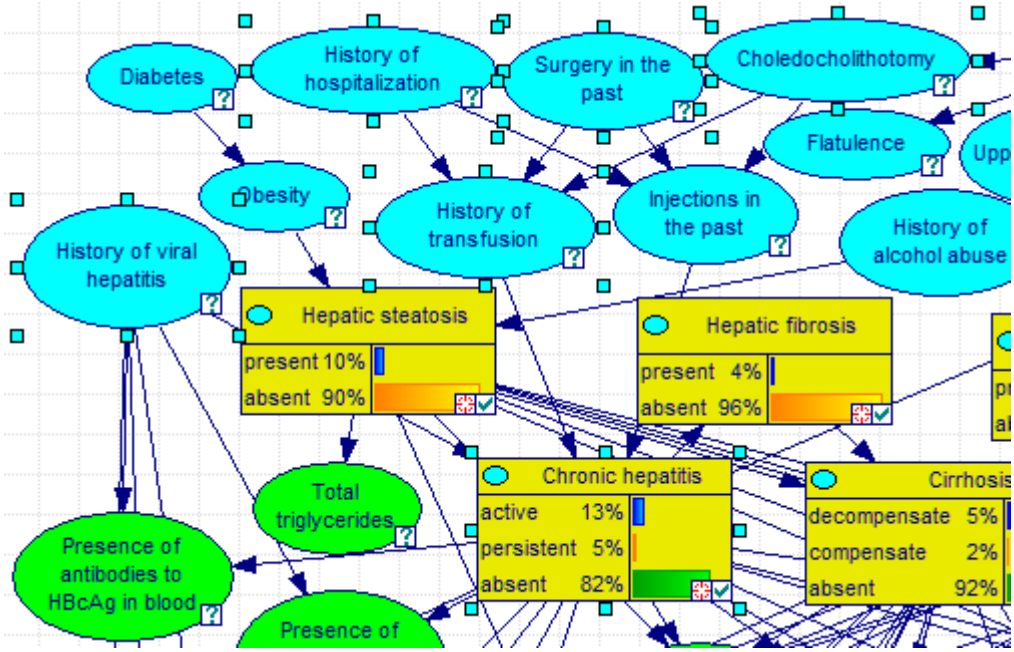


Pressing the *Highlight* button exits the dialog and highlights all nodes on the list, including the POV node and all arcs connecting the nodes. In the image below, *Highlight* button was pressed when the *Node connections* dialog showed the *Markov blanket* of the node *Injections in the past*.



Pressing *ESC* or choosing *Clear Highlight* from the network pop-up menu will clear the selection.

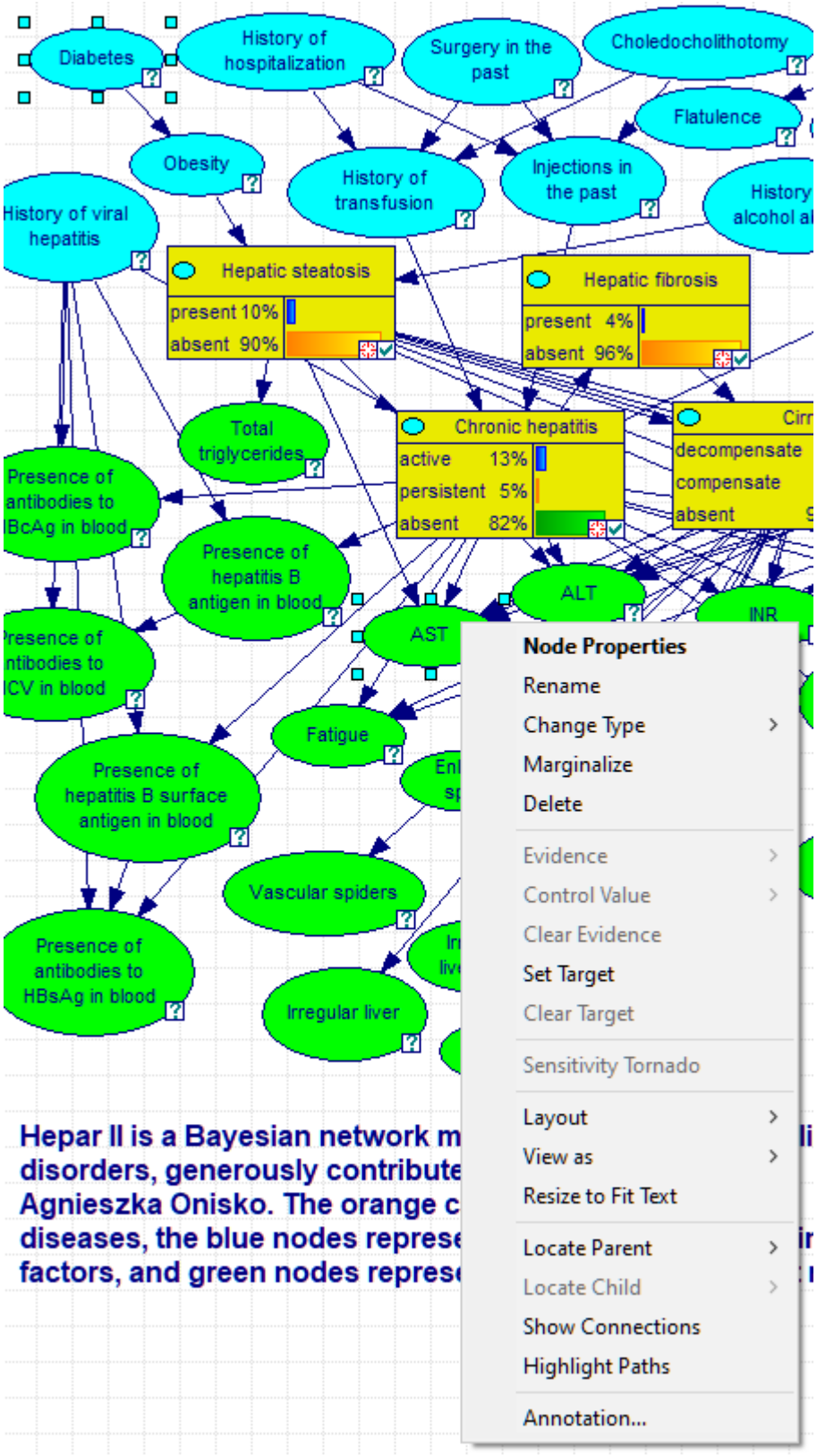
Pressing *Select* button in the *Show Connections* dialog selects all nodes on the list (selection does not include the POV node!) in the *Graph View*. In the image below, *Select* button was pressed when the *Node connections* dialog showed the *Markov blanket* of the node *Injections in the past*.



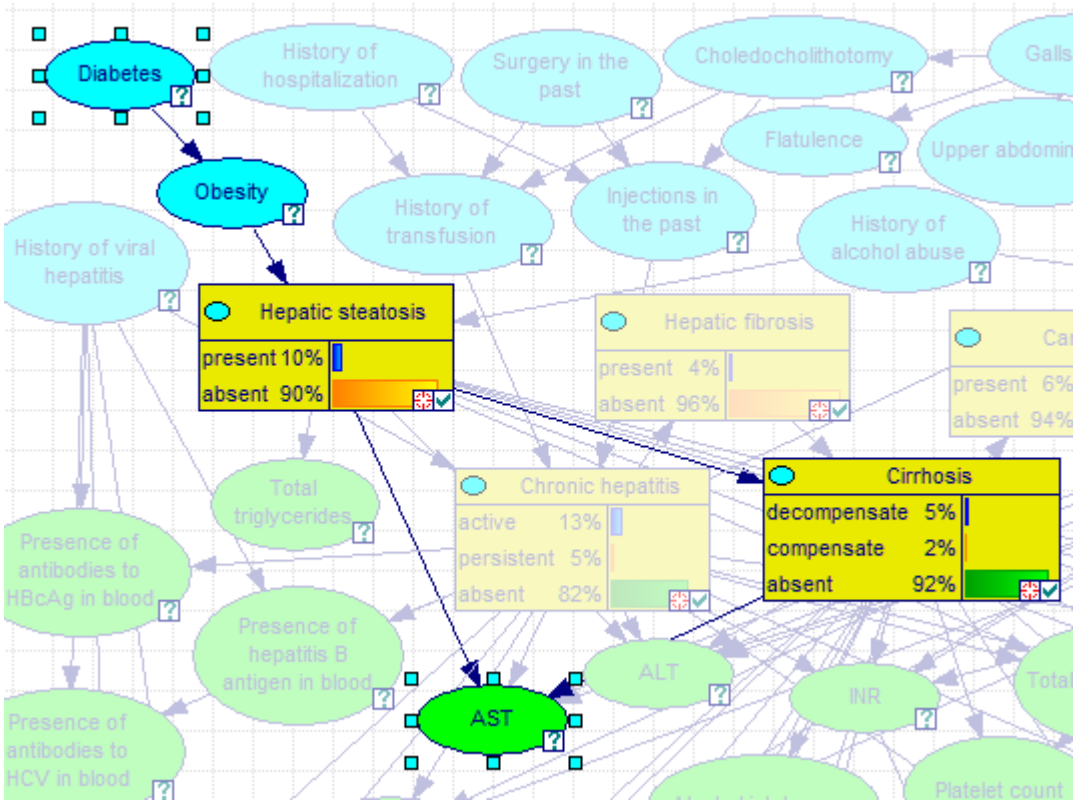
This selection can be further enhanced through other ways of selecting nodes and arcs. Please see the *Select Nodes...* dialog, described in the *Selection of model elements* section.

Pathways of information flow

GeNIe allows for showing active paths through which information flows between a pair of nodes. To show all paths between two nodes A and B, please select these two nodes and choose *Highlight Paths* from the node context menu of one of the two nodes. The following image shows how to invoke highlighting paths between the nodes *Diabetes* and *AST*.


Diabetes
History of hospitalization
Surgery in the past
Choledocholithotomy
Flatulence
Obesity
History of transfusion
Injections in the past
History of viral hepatitis
Hepatic steatosis
present 10%
absent 90%
Hepatic fibrosis
present 4%
absent 96%
Total triglycerides
Chronic hepatitis
active 13%
persistent 5%
absent 82%
decompensate
compensate
absent
Presence of antibodies to HBcAg in blood
Presence of hepatitis B antigen in blood
ALT
AST
INR
Presence of antibodies to HCV in blood
Fatigue
Presence of hepatitis B surface antigen in blood
Vascular spiders
Presence of antibodies to HBsAg in blood
Irregular liver
Node Properties
Rename
Change Type
Marginalize
Delete
Evidence
Control Value
Clear Evidence
Set Target
Clear Target
Sensitivity Tornado
Layout
View as
Resize to Fit Text
Locate Parent
Locate Child
Show Connections
Highlight Paths
Annotation...
Hepar II is a Bayesian network m
disorders, generously contribute
Agnieszka Onisko. The orange c
diseases, the blue nodes represe
factors, and green nodes represe

The result of choosing *Highlight Paths* above is



Pressing ESC or choosing *Clear Highlight* from the *Network pop-up* menu will clear the selection.

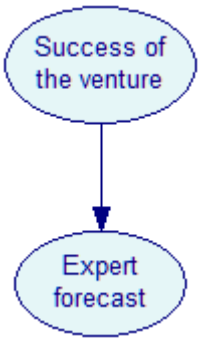
6.2.3 Useful structural transformations

This section reviews two useful structural transformation of Bayesian networks that preserve the join probability distribution represented by the network: arc reversal and node marginalization.

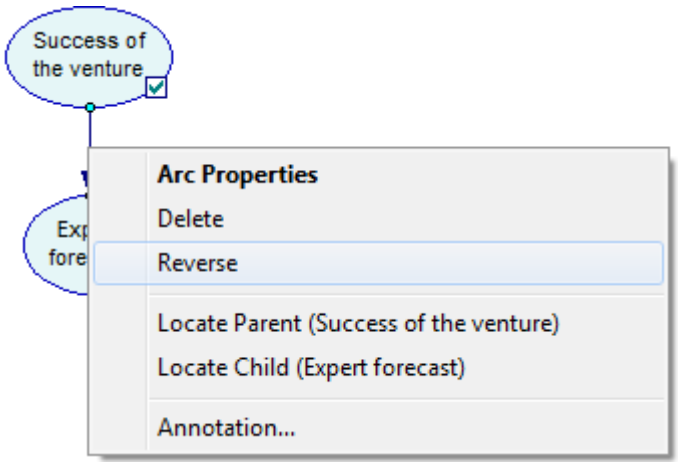
Arc reversal

Formally speaking, the lack of an arc between two nodes *x* and *y* that denotes (possibly conditional) independence between *x* and *y*. Less formally, but more intuitively, an arc in Bayesian networks denotes direct probabilistic influences. It is a good idea to draw arcs in causal direction and let them denote causal relationships. It may happen in the process of building a Bayesian network that you draw an arc from a node *x* to a node *y* and later realize that you would rather have the arc oriented from *y* to *x*. Another situation in which you may want to change the direction of an arc is when performing expert elicitation and realizing that while the causal direction is from *x* to *y*, P(y|x) is easier to estimate. The operation of arc reversal is a structural transformation of a Bayesian network that allows for changing the direction of an arc but at the same time preserves the joint probability distribution represented by the network.

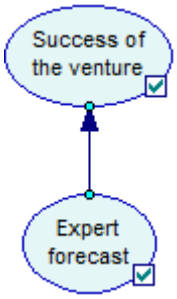
Consider the simple Bayesian network used in the *Hello GeNIe!* example



To change the direction of the arc, right-click on it and choose *Reverse*.



The result is a Bayesian network that represent the same joint probability distribution between the two variables but has the arc pointing in the opposite direction.



Compare the CPT of the node *Success of the venture* before (left) and after (right) the operation of arc reversal.

Success	0.2
Failure	0.8

Expert forecast	Good	Moderate	Poor
Success	0.5	0.25	0.076923077
Failure	0.5	0.75	0.92307692

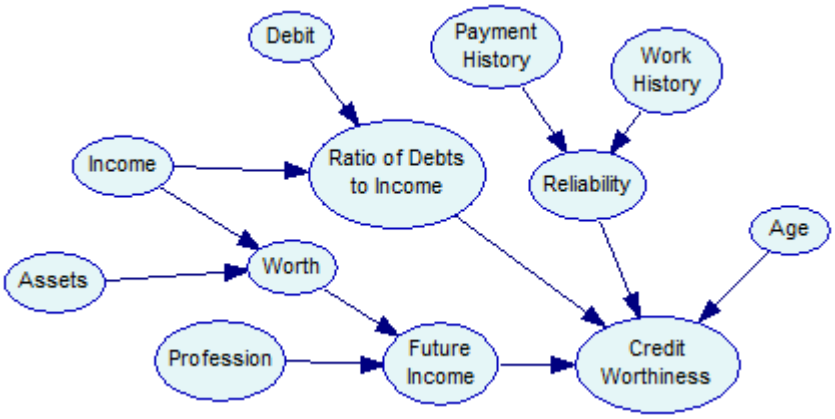
Similarly, different is the CPT of the node *Expert forecast* before (left) and after (right) the operation of arc reversal.

Success of the...	Success	Failure
Good	0.4	0.1
Moderate	0.4	0.3
Poor	0.2	0.6

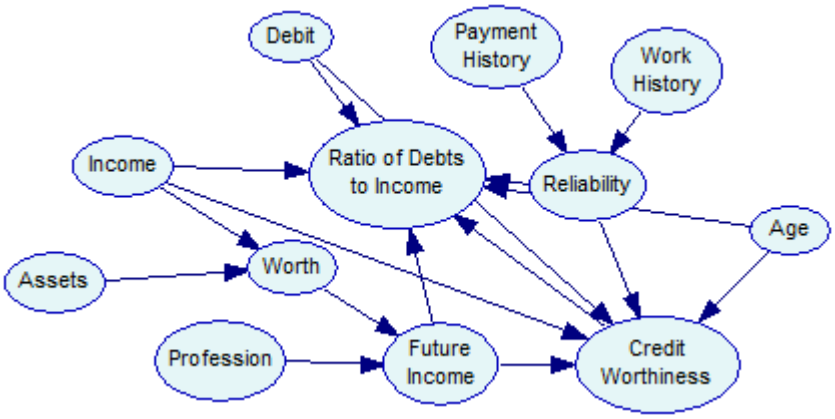
Good	0.16
Moderate	0.32
Poor	0.52

The CPTs are recalculated in such a way that they yield the same joint probability distribution over all nodes in the network.

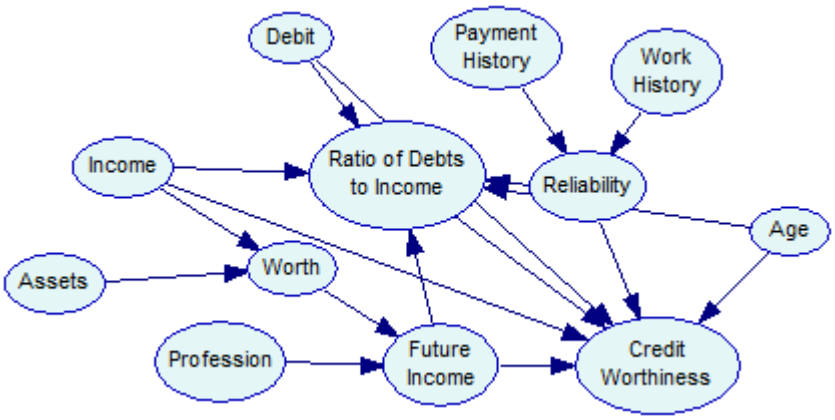
There is an additional complication related to reversing an arc: Because the structure of the network encodes explicitly independences in the domain, reversing an arc violates the pattern of independences. To account for this, in general the two nodes at the opposite ends of the arc need to inherit their parents. In a network with many nodes and connections among them, the operation of arc reversal may, thus, lead to a significantly more complex structure. Consider the following Bayesian network:



Reversing the arc between the nodes *Ratio of Debts to Income* and *Credit Worthiness* transforms the graph to the following form



It happens so that the operation of arc reversal loses some structural information about the network and, in particular, about the independences that the structure of the network represents. Because of this, reversing the same arc back does not necessarily restore the original graph

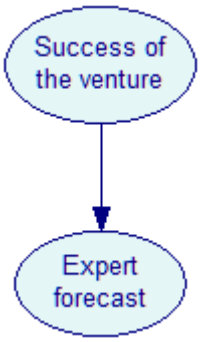


Our advice is to use this operation rarely, if possible. It preserves the joint probability distribution represented by the model but it may lead to loss of a very important property, notably the structure of the underlying graph.

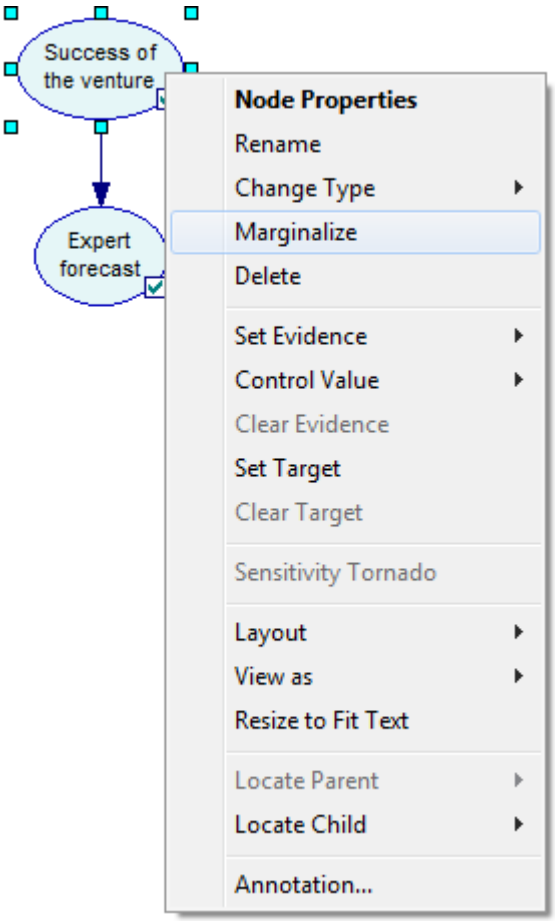
Node marginalization

It may happen that we want to simplify a model by means of removing a variable from it. If we want to preserve the joint probability distribution over the nodes in the network, we can use the operation of marginalization. Deleting the variable instead of marginalizing it in would lead to loss of the numerical properties of the network.

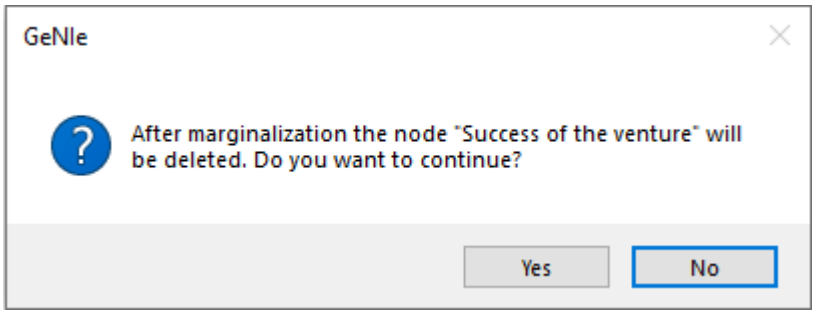
Consider again the simple Bayesian network used in the *Hello GeNIe!* example



To marginalize the node *Success of the venture*, right-click on it and choose *Marginalize*.



GeNIe issues a warning that the marginalized node will be deleted from the network. Pressing *No* gives us a chance to retract from the operation.

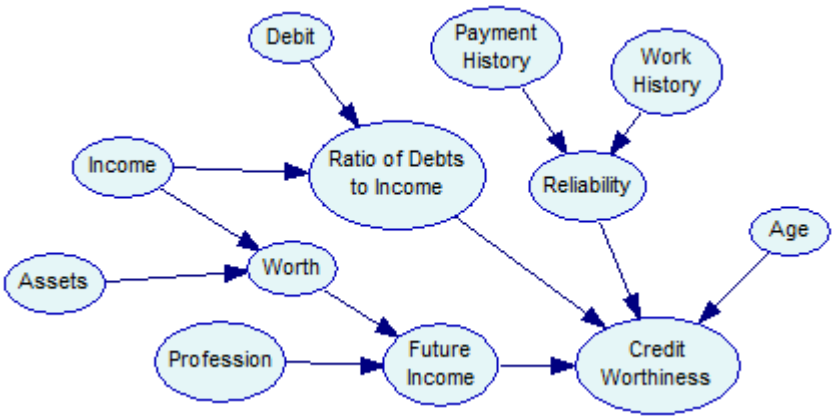


Pressing *Yes*, however, removes the node *Success of the venture* while modifying the CPTs of the remaining nodes so that they preserve their ability to represent the joint probability distribution. The CPT of the node Expert forecast is transformed to the following

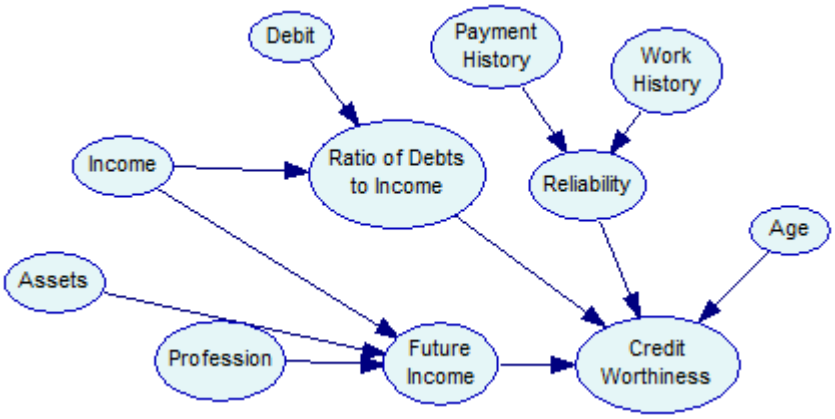
Good	0.16
Moderate	0.32
Poor	0.52

which is what would be its marginal distribution with the node *Success of the venture* present.

Sometimes, when the marginalized node is involved in complex interactions with other nodes, the operation of marginalization may introduce additional arcs, which have as a goal preservation of the dependencies that are lost by removing the node. Consider the following Bayesian network:



Marginalizing the *Worth* transforms the graph to the following form



Please note the new arcs between *Income* and *Future Income* and *Assets* and *Future Income*.

Marginalization of a node may in addition lead to change of the type of its children. When a child of the marginalized node is *Deterministic* or *NoisyAdder*, it will be converted to a chance general node. The same may happen with a child node that is a *NoisyMax* - whenever a new arc is added to that node, it is no longer a Noisy-Max node in general.

The operation of marginalization is powerful but it leads to loss of information, so we advise caution in using it.

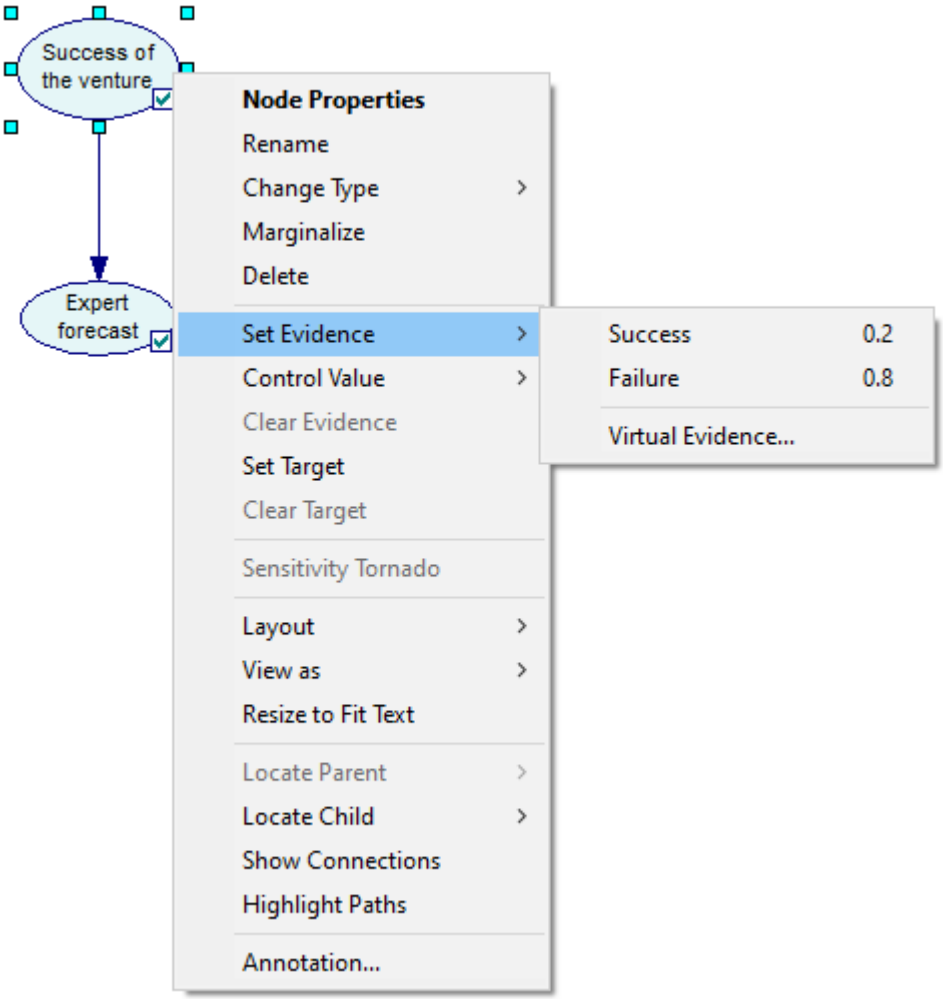
6.2.4 Entering and retracting evidence

Entering observations (evidence) is one of the basic operations on a probabilistic model. It amounts to adjusting the model to a new situation, one in which more information is available. It allows to query the system

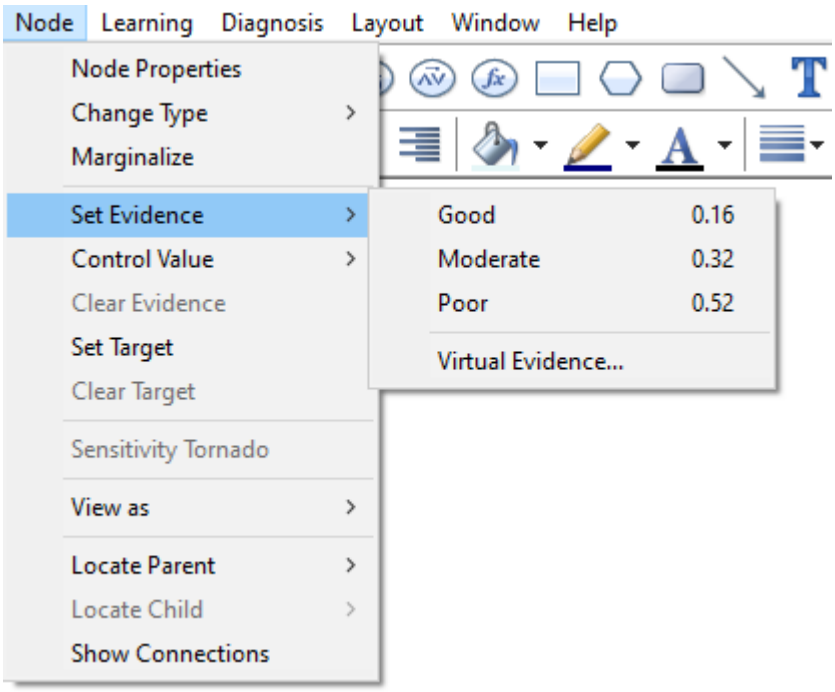
subsequently about the new, posterior probability distributions. If you have gone through the *Building a Bayesian network* tutorial introduction, you might remember that you have already entered evidence for the *Expert forecast* node. Let us go through this again.

You may load the model *VentureBN.xdsl* created in *Building a Bayesian network* from the *Example Networks* folder.

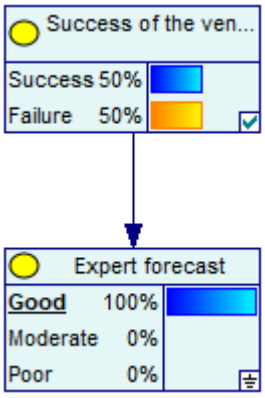
To enter evidence into your model, right-click on the node in question (in the picture below, node *Success of the venture*) and choose *Set Evidence*.



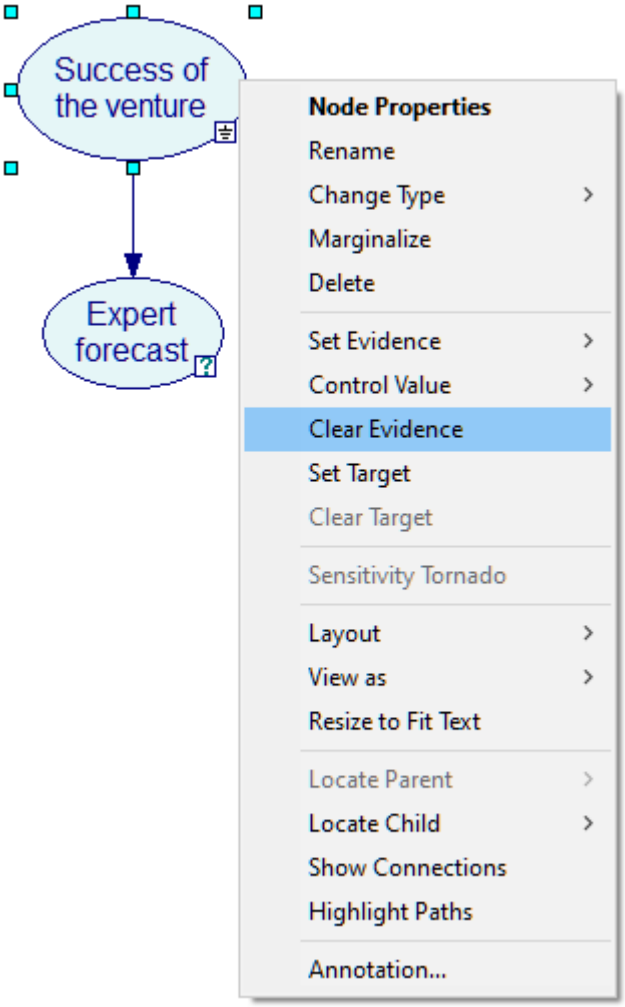
Alternatively, select the node in question and choose *Set Evidence* submenu from the *Node Menu*.



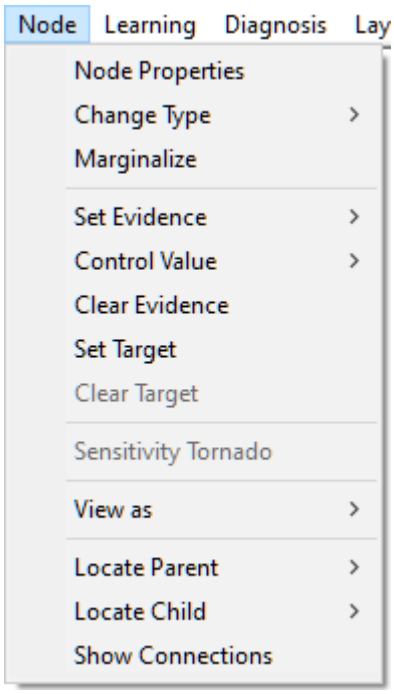
Yet another way is double-clicking on the observed state in the *Bar chart view* of the node



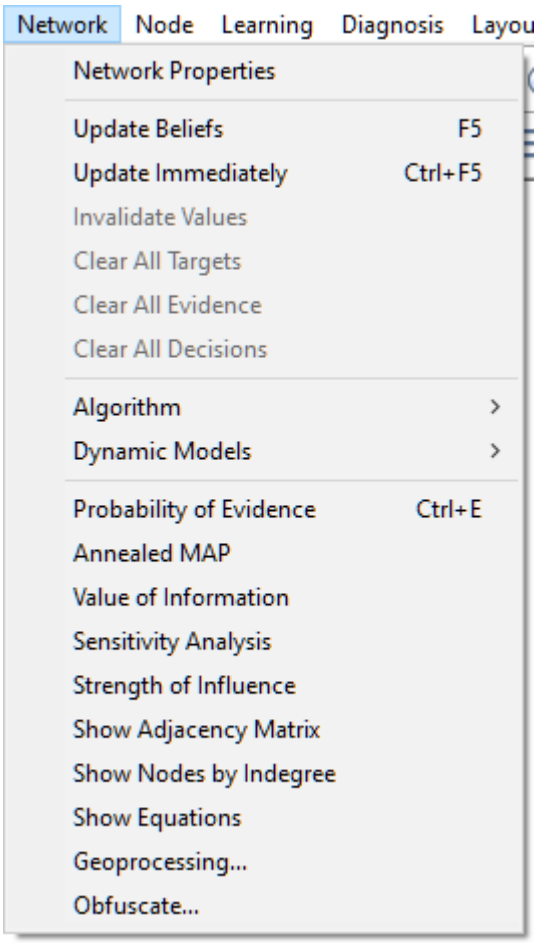
To retract evidence, right-click on the node with previously entered evidence (in the picture below, node *Success of the venture*) and choose *Clear evidence*.



Alternatively, select the node in question and choose *Clear evidence* from the *Node Menu*.



You can also retract all evidence by choosing *Clear all evidence* from the *Network Menu*:

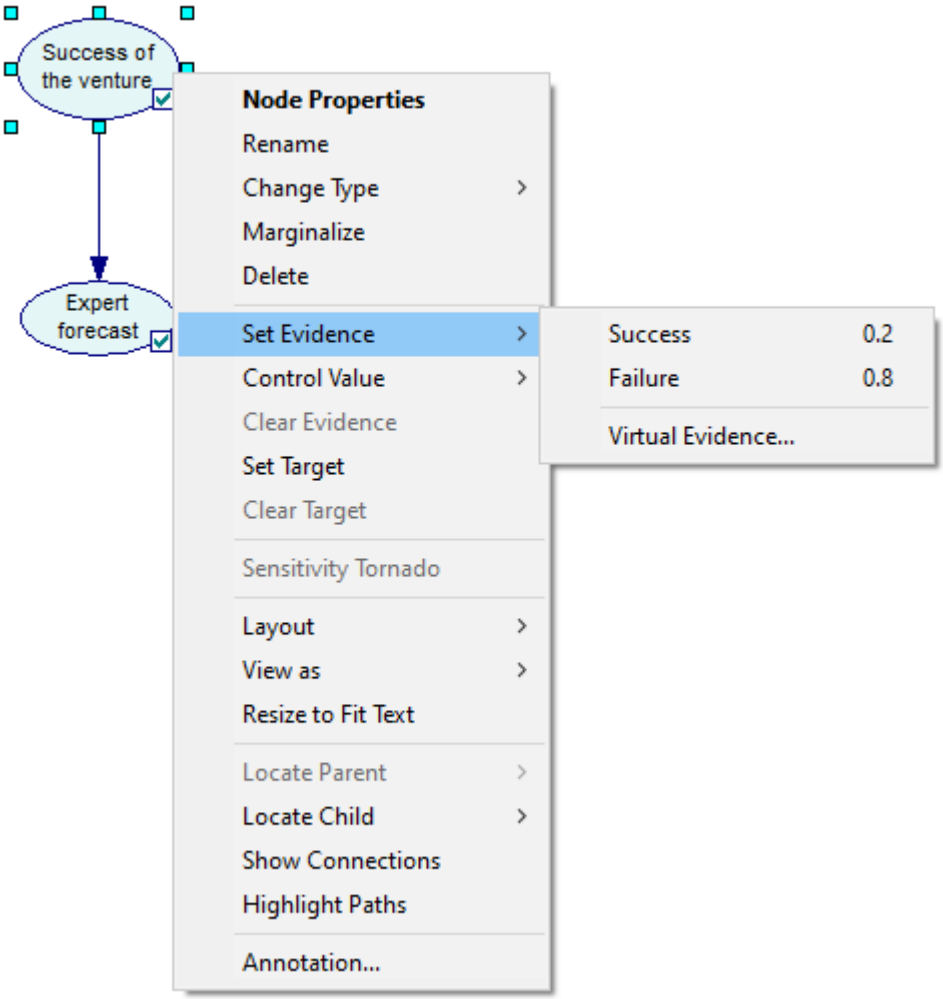


To enter different evidence instead of retracting evidence altogether, just set different evidence from the *Set Evidence* sub-menu.

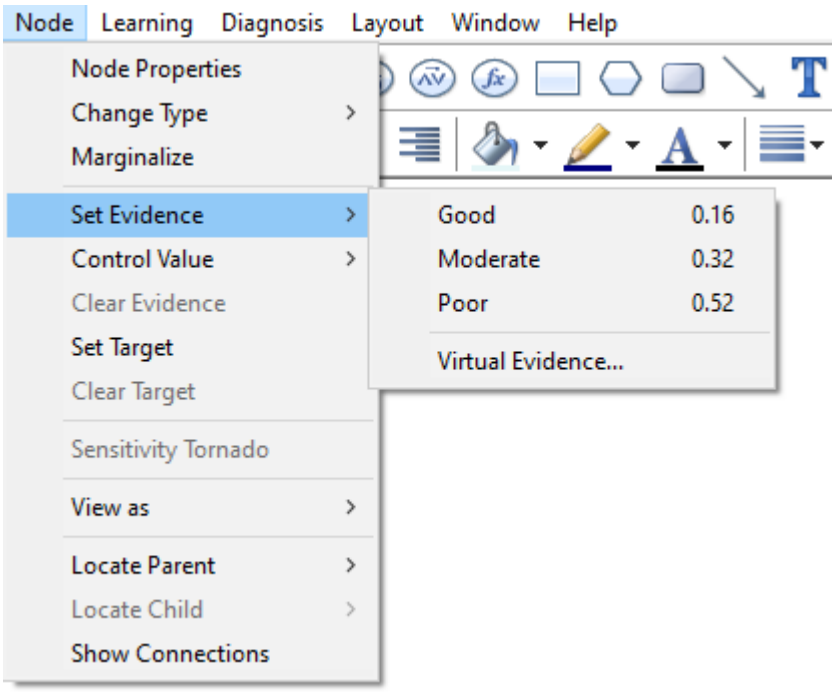
6.2.5 Virtual evidence

Virtual evidence is a term used for a shortcut that allows for entering evidence for normally unobservable variables. A typical modeling practice in such cases is that we model such variables but next to them variables that are observable and can provide us with information about the unobservables. For example, we are typically unable to determine whether a disease is present or not but can model a medical test, which when observed will provide evidence for or against the disease. The test result is readily observable. Virtual evidence allows us for entering uncertain observation (in form of probability distribution over the possible states of the observation) directly into the normally unobservable variable. Some modelers use this construct to modify the prior probability distribution over a variable, although this works only when the variable does not have parents.

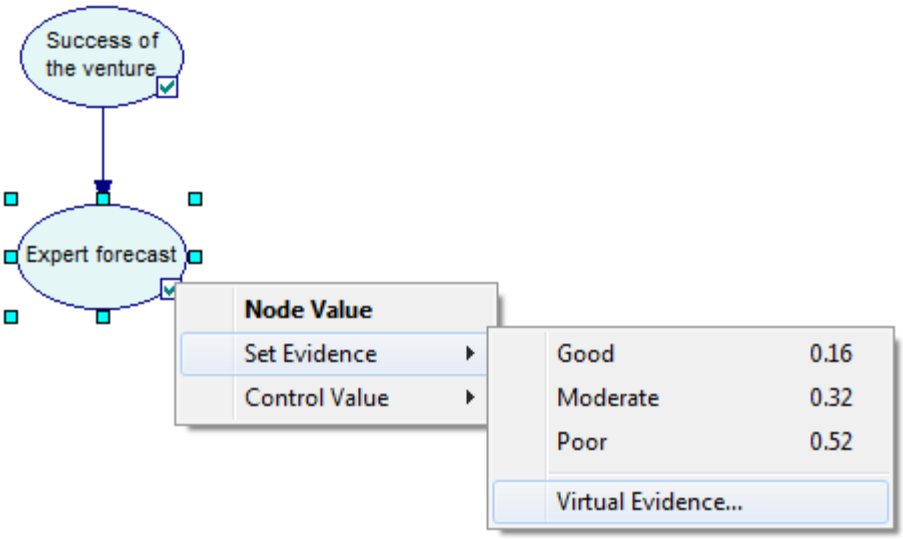
Entering virtual observations (evidence) is similar to entering evidence. The main difference is that instead of observing a state of a node, we enter a probability distribution over all states of the node. You may load the model *VentureBN.xdsl* created in *Building a Bayesian network* from the *Example Networks* folder. To start the virtual evidence dialog, right-click on the node in question (in the picture below, node *Expert forecast*) and choose *Set Evidence*, followed up by *Virtual Evidence...*



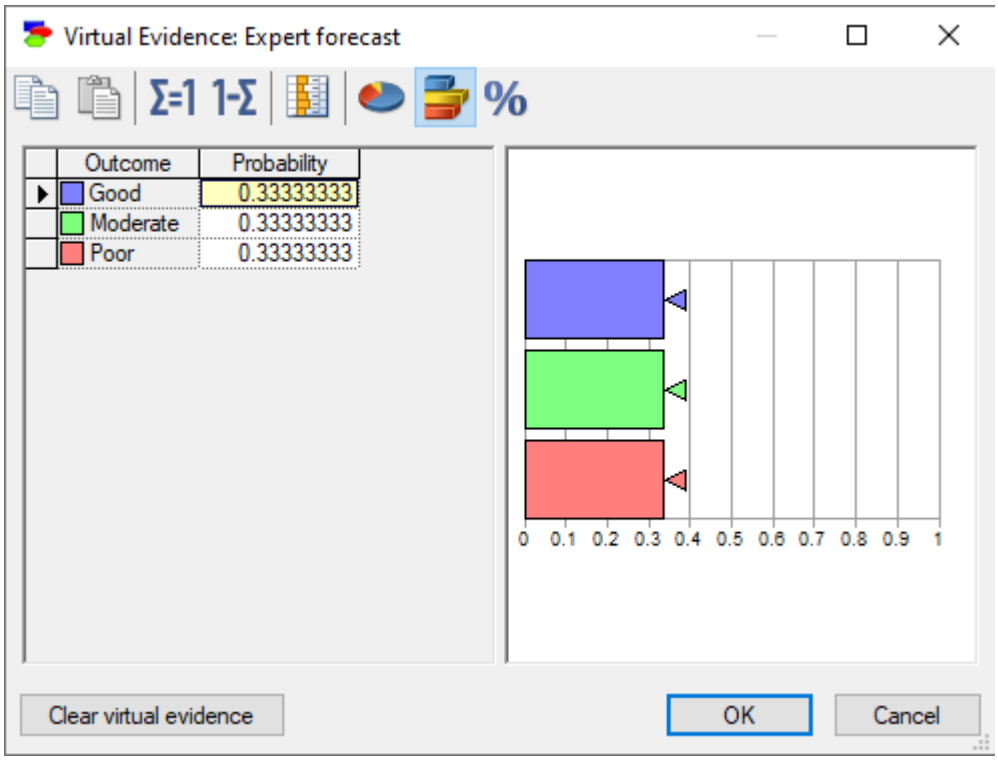
Alternatively, select the node in question and choose *Set Evidence-Virtual Evidence...* sub-menu from the *Node Menu*.



Yet another way is right-clicking on the updated icon of the node and using the context menu that pops up.



In each of these three cases, this invokes the following dialog for entering virtual evidence:



Virtual evidence can be entered numerically or graphically, using a probability wheel or a bar chart (pictured in the above screen shot). Keep in mind that virtual evidence is a probability distribution over the states of the observed variable and individual probabilities in the distribution have to add up to 1.0.

The internal implementation of virtual evidence in GeNIe is simple and equivalent to creating a temporary child node for each node with virtual evidence and populating its CPT with values based on the values entered as virtual evidence. The theoretical interpretation of this temporary node is that it provides uncertain information about its parent, which is what virtual evidence is about.

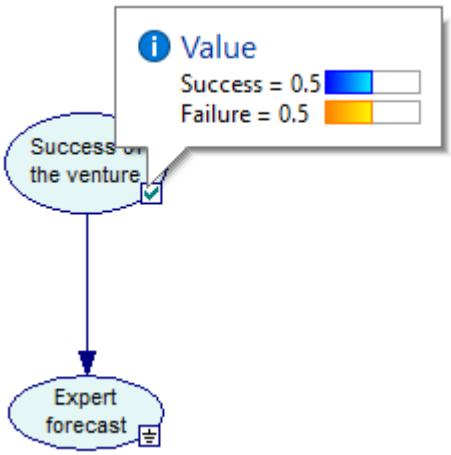
Retracting virtual evidence is identical to retracting evidence and has been described in section *Entering and retracting evidence*.

To enter different evidence instead of retracting evidence altogether, invoke the *Virtual Evidence Dialog* and set different evidence.

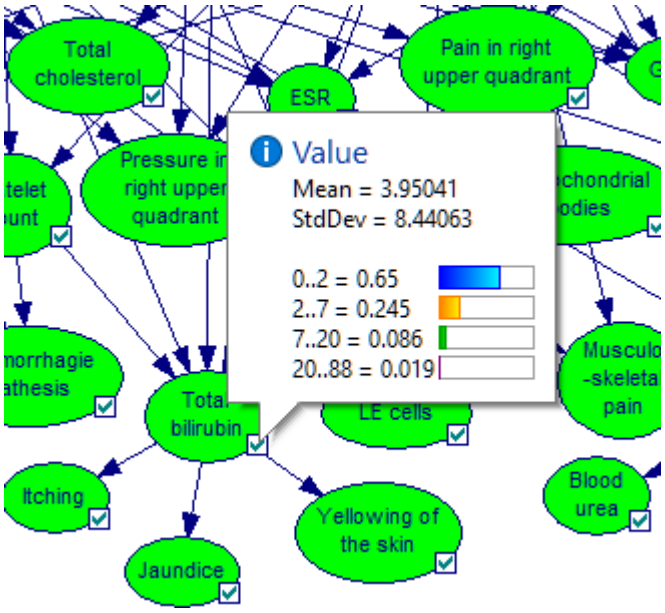
6.2.6 Viewing results

Marginal probability distributions

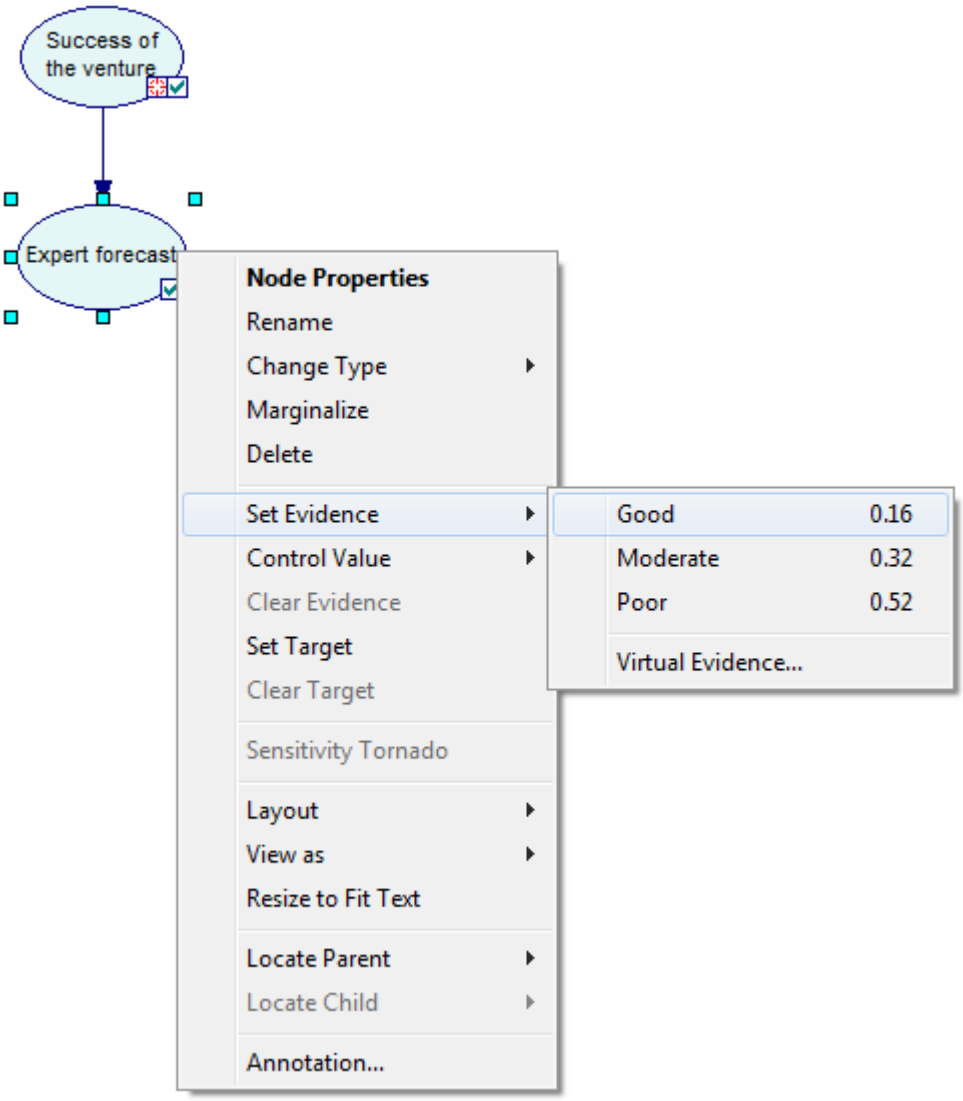
Once a model is evaluated, you are ready to view the results computed by GeNIe. This can be done in several ways. Posterior marginal probability distribution over any node in a Bayesian network can be viewed by hovering the cursor over the status icon when it is *Updated* (☑). The probabilities of the various states of the node will be displayed as follows:



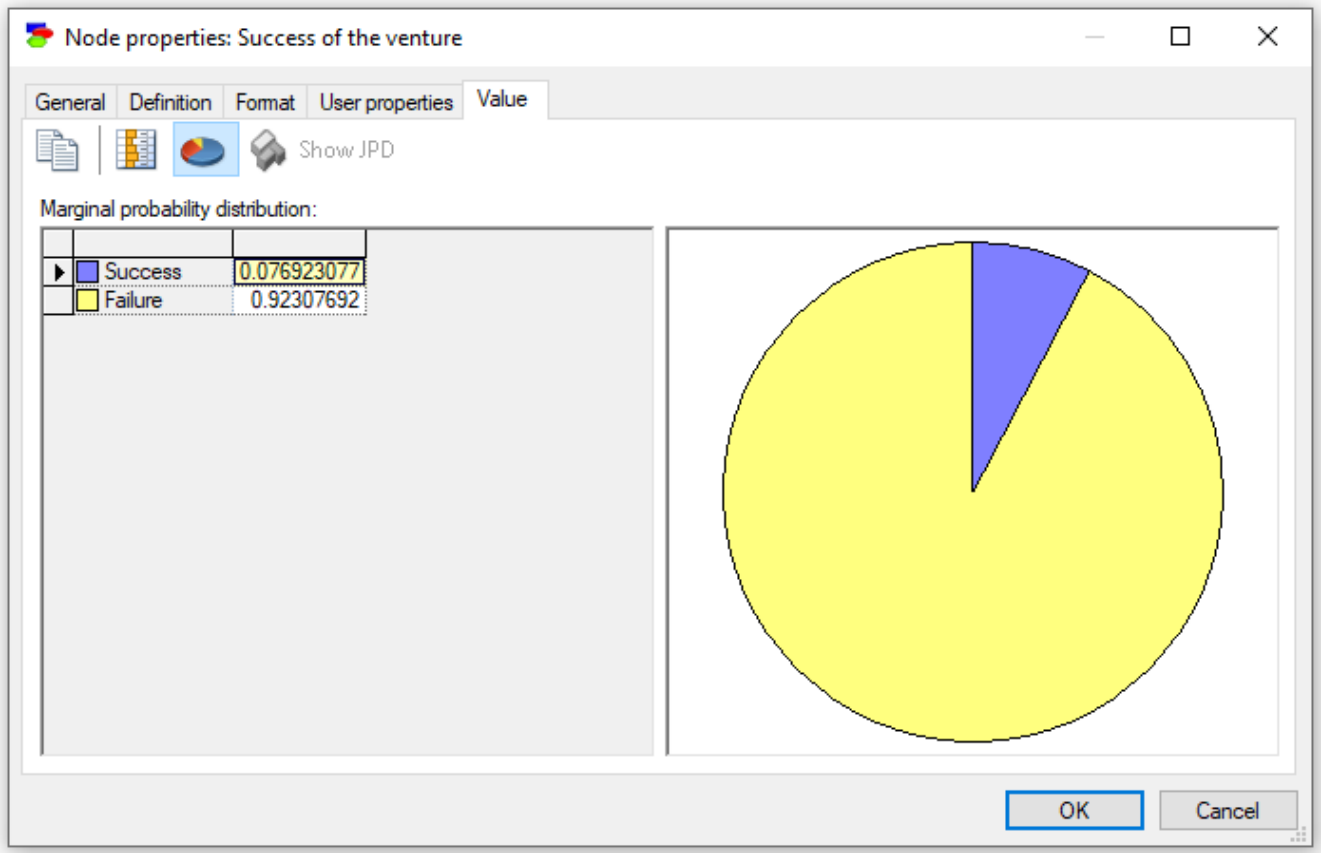
For nodes that are discrete numerical, GeNIe shows the probability distribution over their states but also calculates the mean and standard deviation over their domains.



You can also right-click on the node and choosing *Set Evidence*. GeNIe shows a list of states of the node along with their probabilities, when these are available.



An alternative way of viewing the posterior probability distribution is to choose the *Value* tab from the *Node Property Sheet*. Shown below is the *Value* tab when the *Expert forecast* is *Poor*.

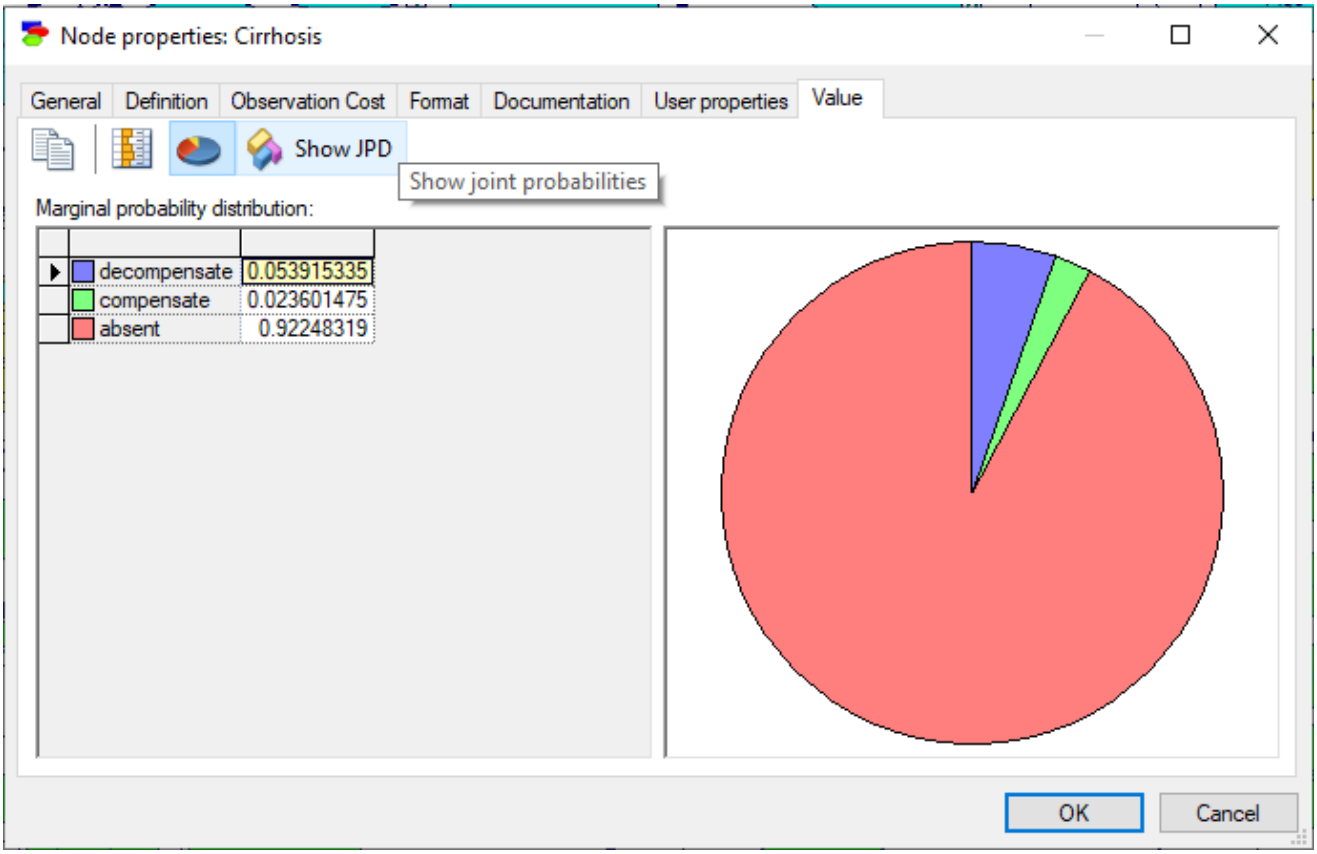


Posterior probability distribution in influence diagrams are more detailed. In cases when a node's probability distribution is affected by a decision or by a node that precedes a decision node, the posterior probability distribution is indexed by the outcomes of these nodes. In such cases, right clicking on a node icon will display a message *The result is a multidimensional table, double click on the icon to examine it*.

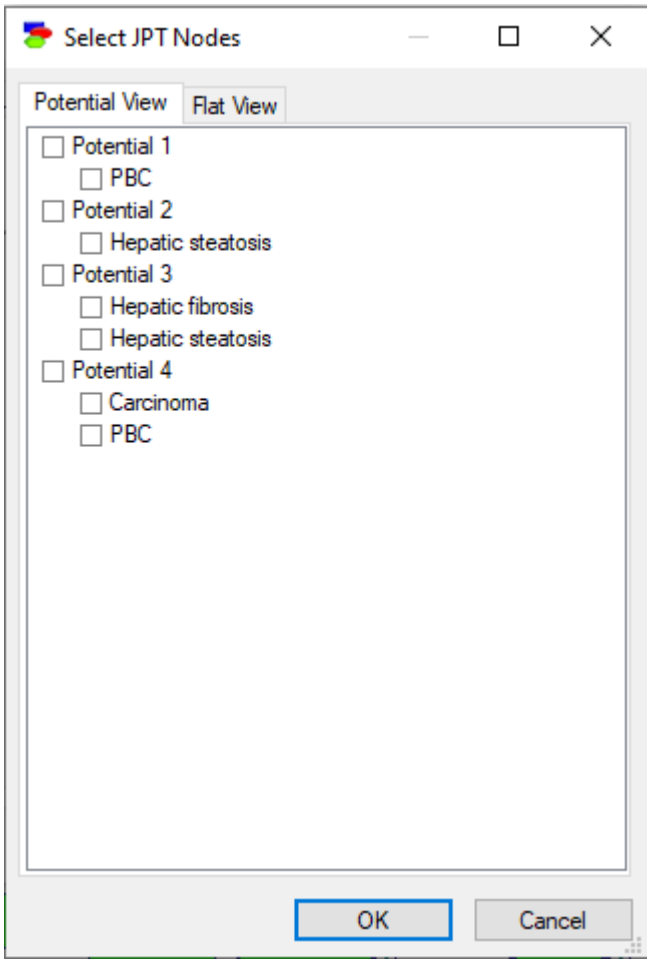
The only way to see the entire distribution, is by using the *Value Tab* of the decision node or the value node in question.

Joint probability distributions

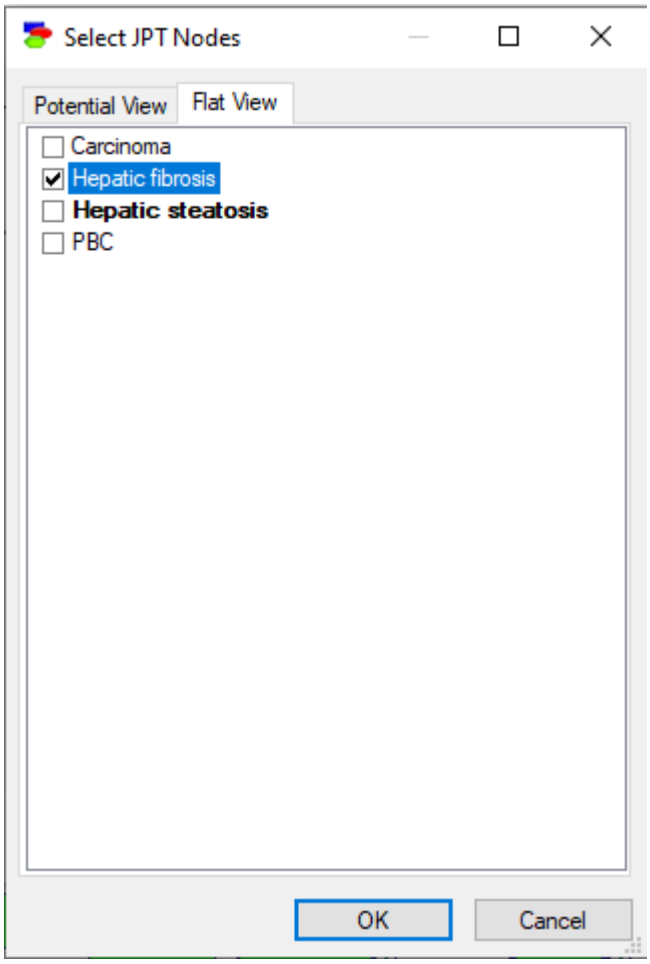
Preservation of clique potentials allows for viewing joint probability distribution over those variables that are located within the same clique. Should you wish to derive the joint probability distribution over any variable set, just make sure that they are in the same clique before running the clustering algorithm. One way of making sure that they are in the same clique is creating a dummy node that has all these variables as parents. In any case, when the *Preserve Clique Potentials* flag is on, there is an additional button in the *Value* tab of *Node Properties* dialog, *Show JPD* ( ).



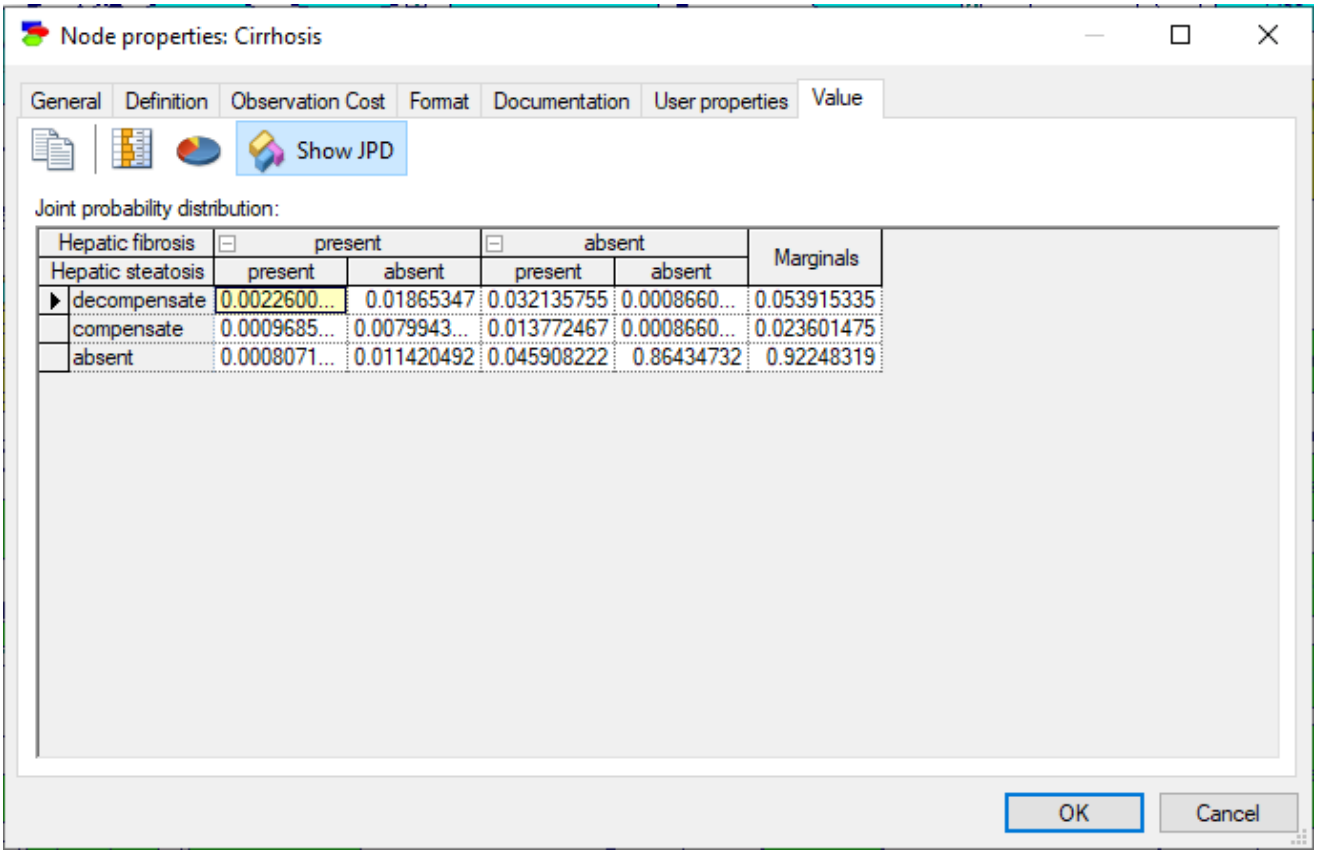
Pressing the *Show JPD* button invokes the following dialog:



Only one potential can be viewed at the same time, although if the potential is large, one can select some variables within the same potential. The *Flat view* tab allows for a variable list view of the potentials.



Selecting *Potential 3* and pressing *OK* yields the following view with the joint probability distribution over variables *Hepatic fibrosis* and *Hepatic steatosis*.



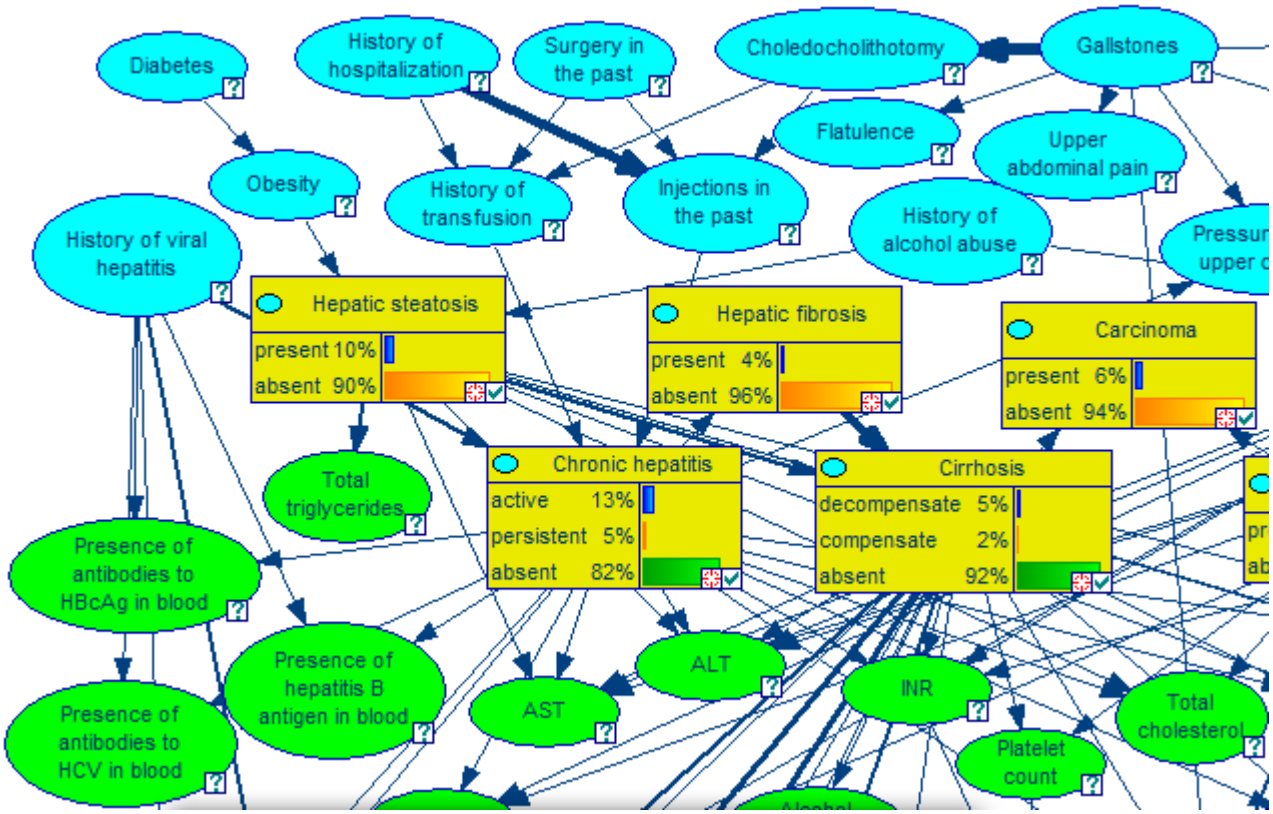
The clustering algorithm is GeNIe's default algorithm and should be sufficient for most applications. Only when networks become very large and complex, the clustering algorithm may not be fast enough. In that case, we suggest the *Relevance-based decomposition* or a stochastic sampling algorithms, such as EPIS-BN (Yuan & Druzdzel, 2003), which is quite likely the best stochastic sampling algorithm available for discrete Bayesian networks.

6.2.7 Strength of influences

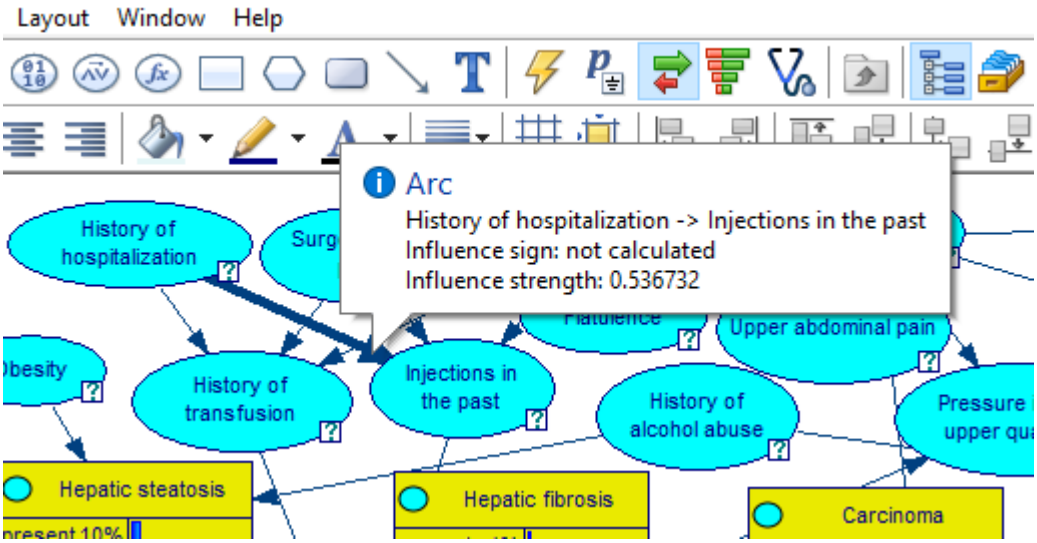
Clicking on the *Strength of Influence* ( ) tool from the *Standard Toolbar*, brings up the *Influence Toolbar* and also changes the appearance of the arcs in the network. The arc have different thickness, dependent on the strength of influence between the nodes that they connect. Strength of influence is calculated from the CPT of the child node and essentially expresses the distance between various conditional probability distributions over the child node conditional on the states of the parent node. This is a GeNIe only feature, i.e., there is no equivalent of this functionality in SMILE. It serves the purpose of visual analysis and verification of models during their construction.

Basic functionality

Here is a fragment of the Hepar II network with the *Strength of Influence* tool ( ) pressed.



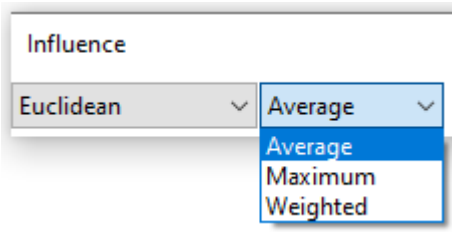
If the mouse is placed on the head of the arrow, information relating the strength of influence is shown in a comment box as shown below.



The *Influence Toolbar* allows to choose various options related to the calculation and display of strengths of influence. It is by default detached from the tool bars and can be moved to any position on the screen.

Selection of display mode

Thickness of arcs can be based on one of the three: *Average* (default), *Maximum*, and *Weighted*

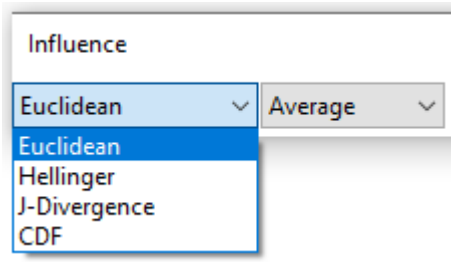


Maximum uses the largest distance between distributions, *Average* takes the plain average over distances, and *Weighted* weighs the distances by the marginal probability of the parent node.

The *Normalize* ( ) button toggles between the normalized and non-normalized mode. In normalized mode, the thickest possible arc is given to that arc that has the highest strength of influence. The thicknesses of all other arcs are calculated proportionally to the thickest arc. In the non-normalized mode (default), thickness is based on the absolute value of the distance.

Measures of distance between distributions

There are four measures of distance between distributions used by this functionality: *Euclidean* (default), *Hellinger*, *J-Divergence*, and *CDF*. A good source of information about the four measures is (Koiter, 2006).

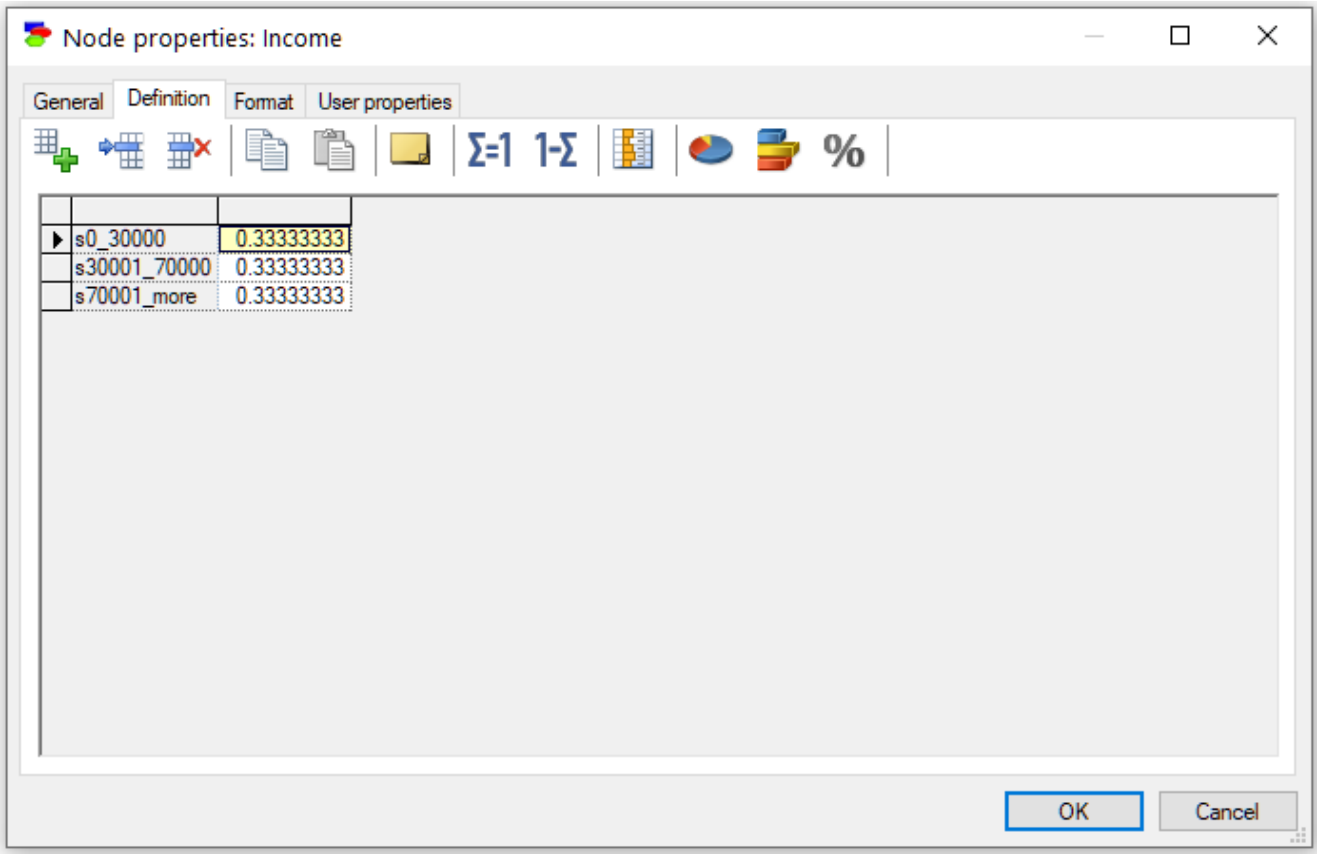


The J-Divergence has an additional setting, which can be changed by pressing the *Alpha* ( ) button. The alpha parameter controls normalization of the J-Divergence.

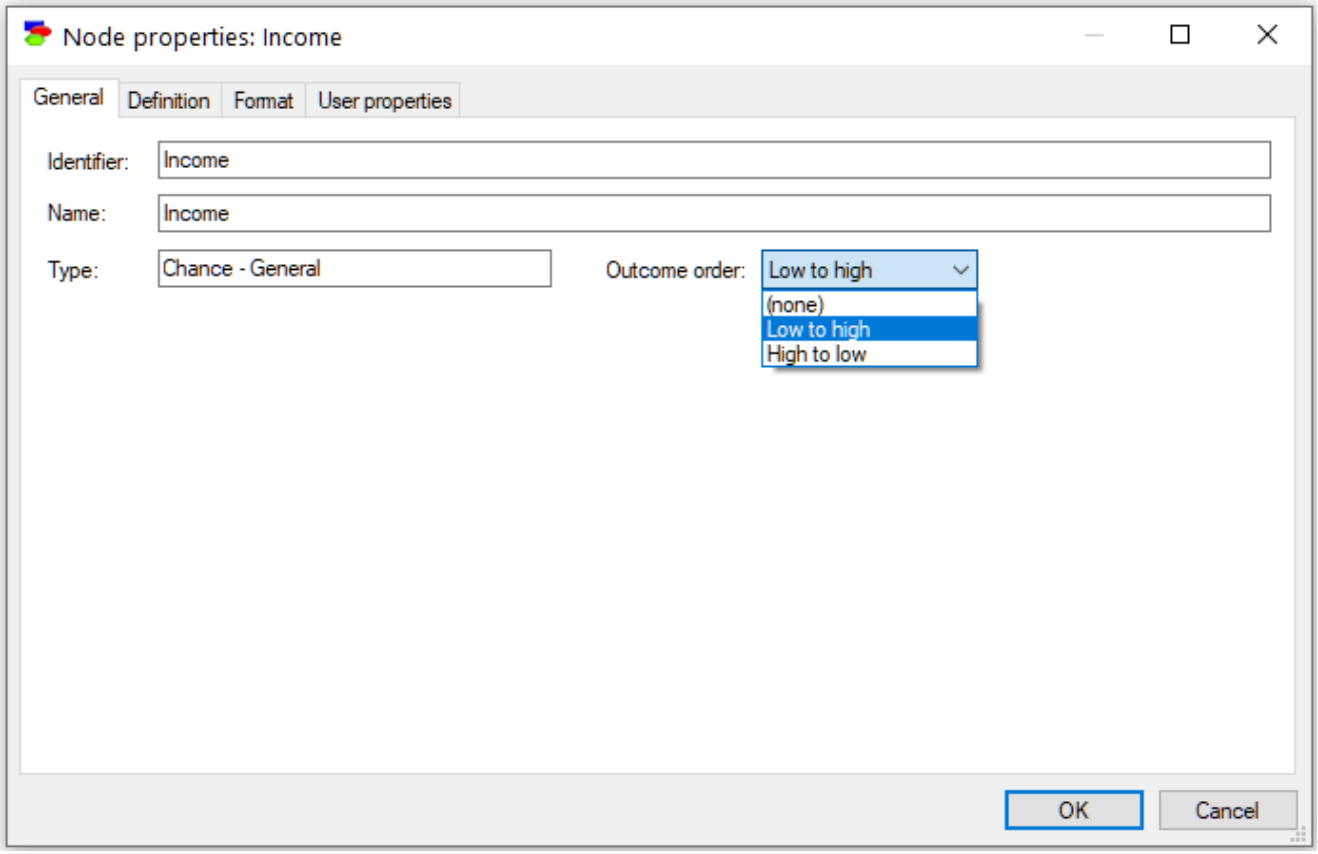
Other settings

Influence shown can be *Static* or *Dynamic*. The *Static* mode only makes use of the conditional probability tables present in the model and is, therefore, not context-dependent. The thickness of the arcs indicates the strength of influence that a parent has on a child, while the colors of the arcs shows the sign of that local influence. The *Dynamic* mode, invoked by pressing the *Show dynamic influence* button ( ) is context-specific and essentially shows the potential influence that two directly connected nodes can have on each other. In the dynamic mode, the *Recalculate influence* ( ) button can be used to recalculate the thickness and coloring of arcs. This is needed when, for example, a new piece of evidence has been observed.

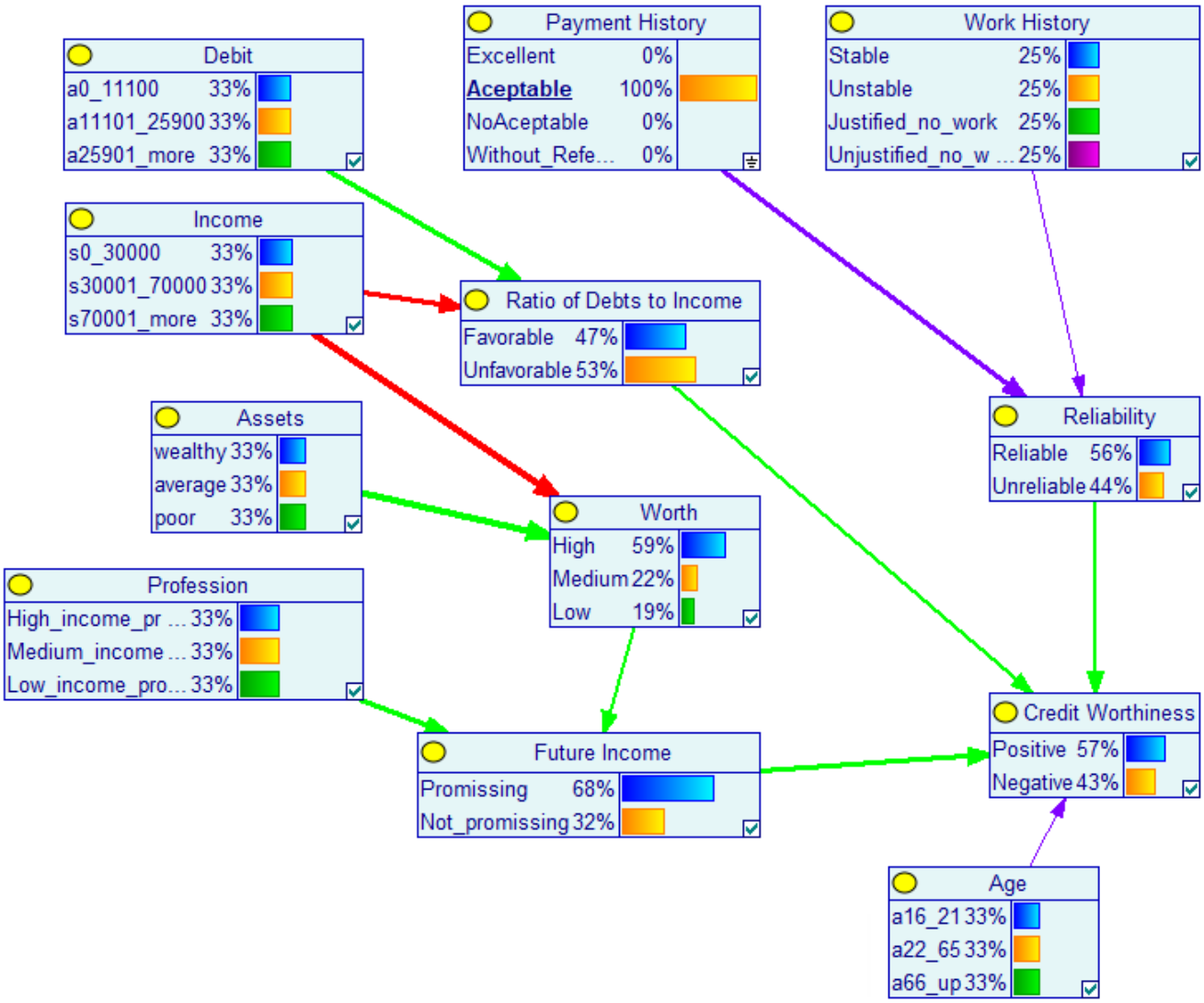
Use arc thickness to show strength ( ) button allows for turning the arc thickness on and off from the *Influence* toolbar. When the *Use color to show sign* ( ) button is pressed, coloring of the arcs is activated. Colors indicate the sign of influence. This sign can be positive (green), negative (red), null (gray), or ambiguous (purple). There are different colors for static and dynamic modes. For the signs of influences and, hence, colors to be calculated, one has to specify first the *Outcome order* on the child node's *Node properties*'s *General tab* (either *Low to high* or *High to low*). Consider, for example the variable *Income* in the `Credit.xdsl` model. Its definition is as follows:



GeNIe needs to know the meaning of the order of outcomes for it to compute the sign of influence. The order of outcomes (clearly from low to high in this case) can be set on the *General* tab of *Node properties*:

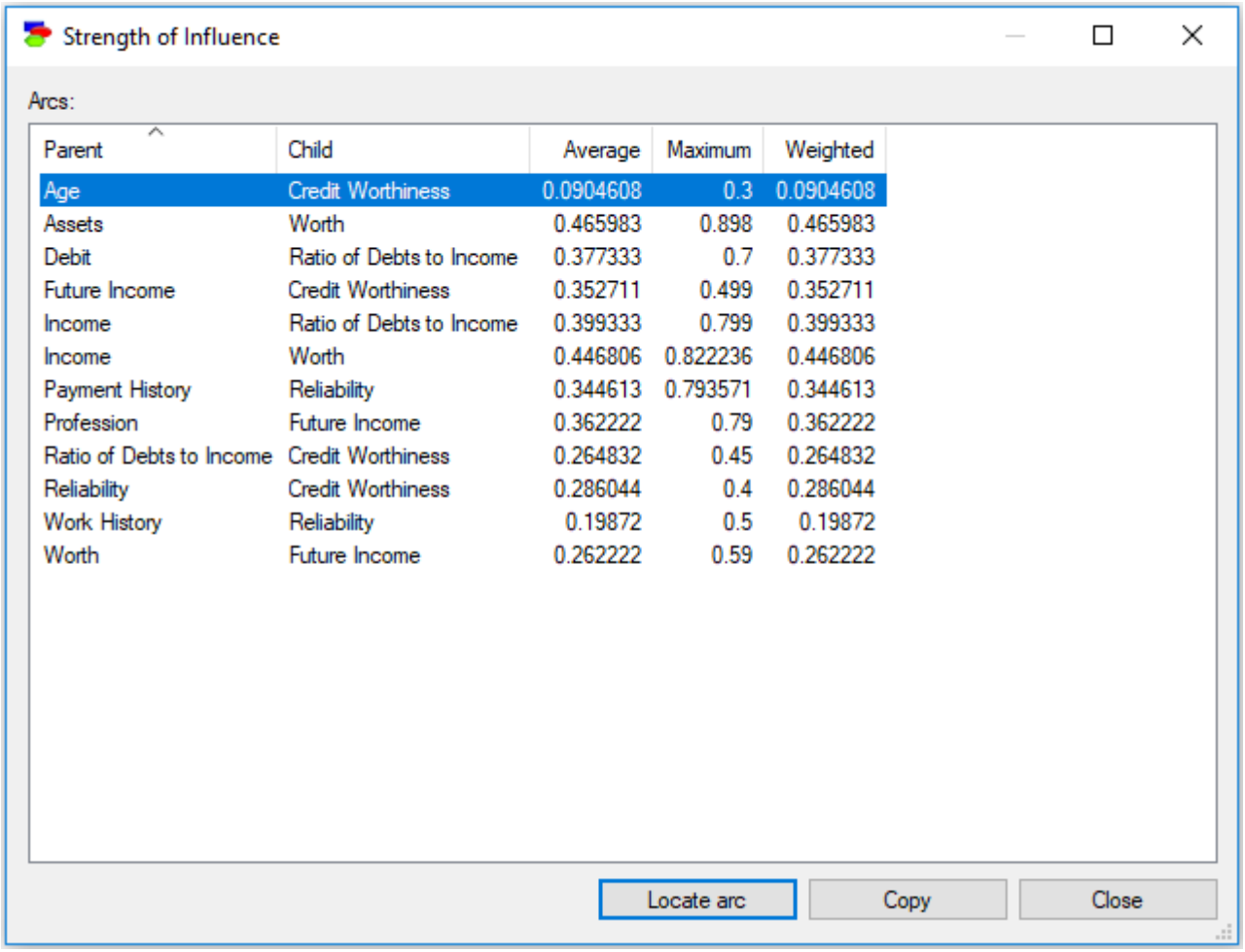


Once we do this for all nodes in the model, GeNIe will be able to calculate and display the signs of individual influences. The following screen shot displays colored arcs for the `Credit.xdsl` model



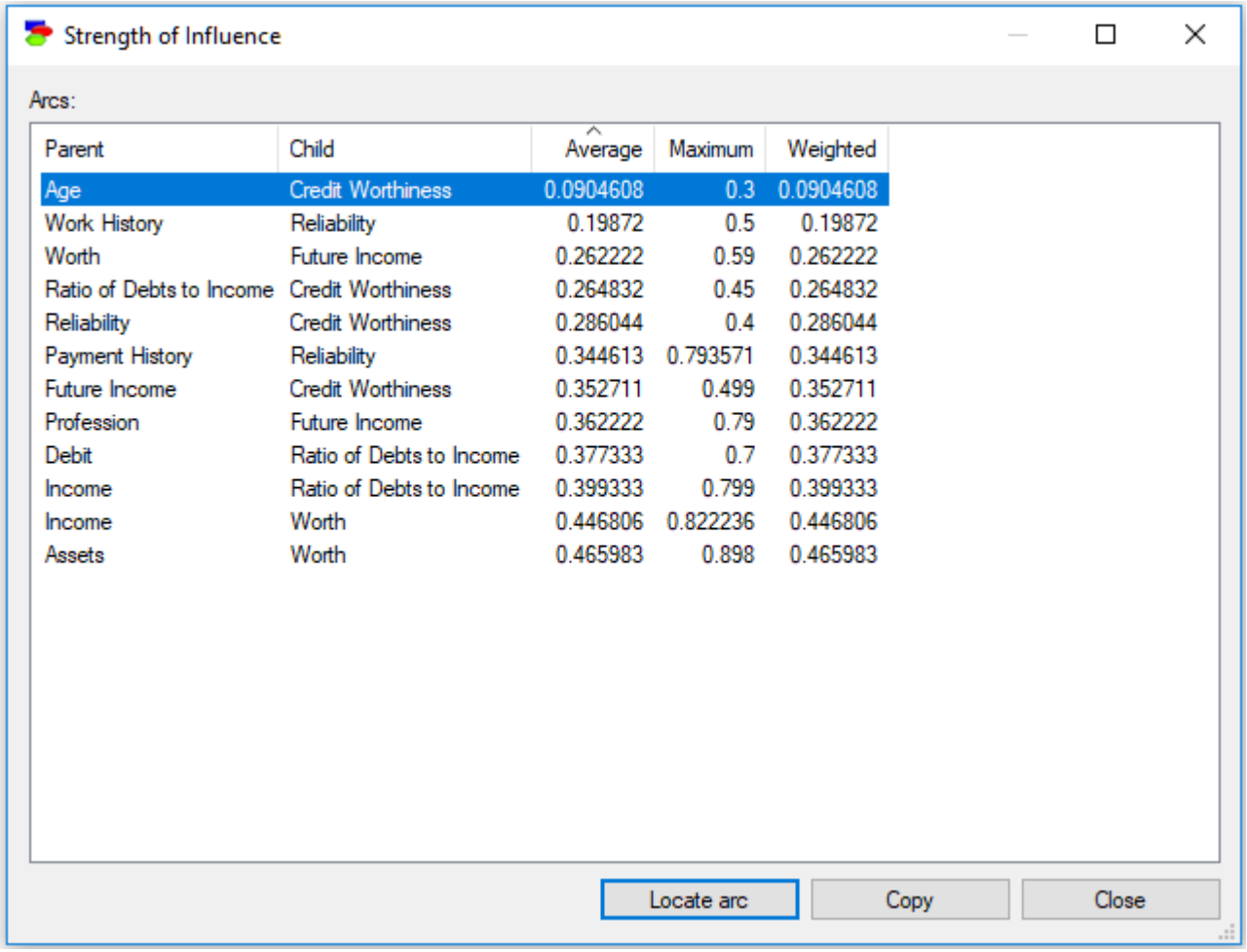
List of arcs

Pressing the *Show arc list* button ( ) invokes the *Strength of Influence* dialog, showing the list of all arcs in the model, along with their strengths



Parent	Child	Average	Maximum	Weighted
Age	Credit Worthiness	0.0904608	0.3	0.0904608
Assets	Worth	0.465983	0.898	0.465983
Debit	Ratio of Debts to Income	0.377333	0.7	0.377333
Future Income	Credit Worthiness	0.352711	0.499	0.352711
Income	Ratio of Debts to Income	0.399333	0.799	0.399333
Income	Worth	0.446806	0.822236	0.446806
Payment History	Reliability	0.344613	0.793571	0.344613
Profession	Future Income	0.362222	0.79	0.362222
Ratio of Debts to Income	Credit Worthiness	0.264832	0.45	0.264832
Reliability	Credit Worthiness	0.286044	0.4	0.286044
Work History	Reliability	0.19872	0.5	0.19872
Worth	Future Income	0.262222	0.59	0.262222

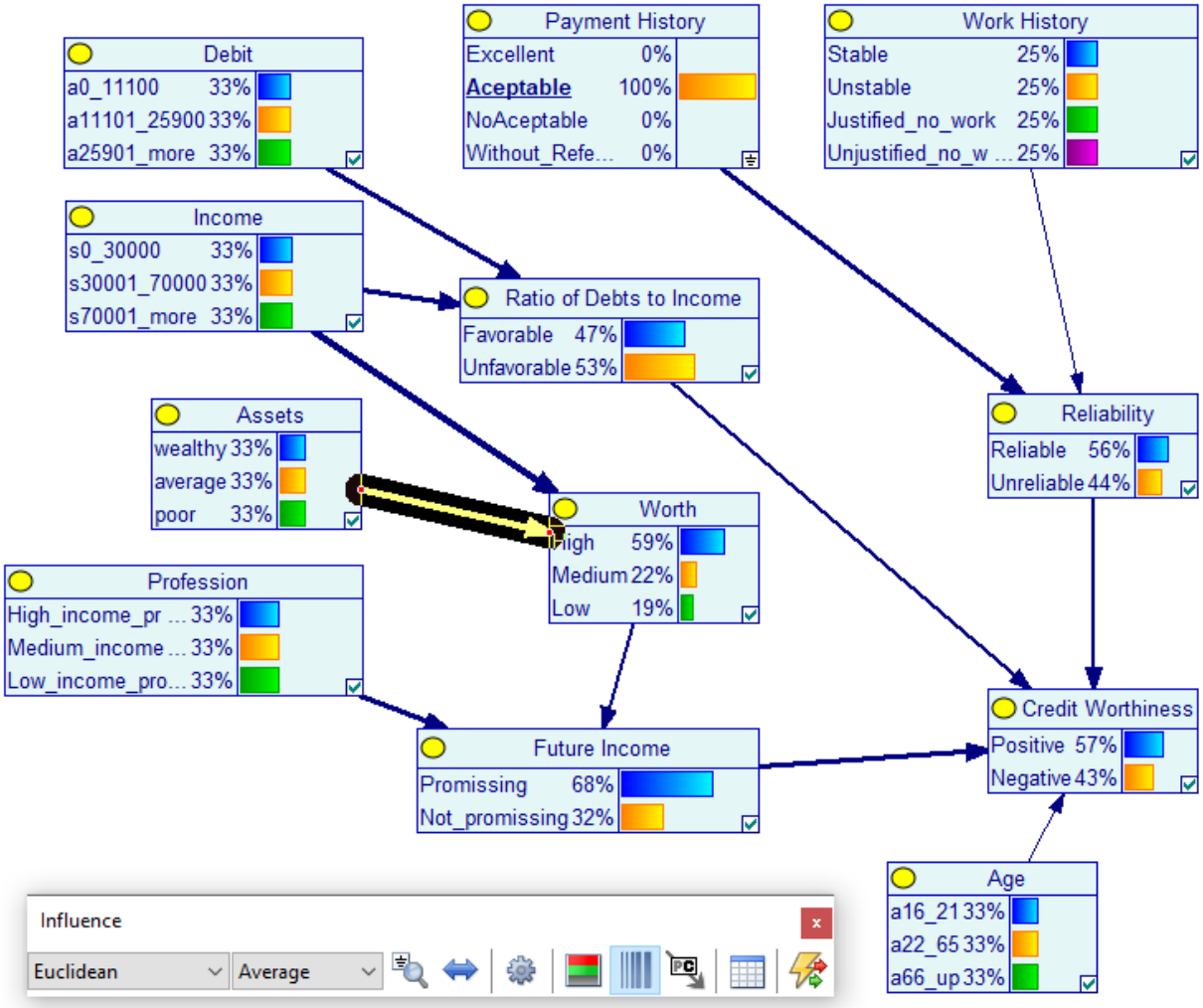
To sort the list by any of the columns, click on the column header. Clicking again will change the order from increasing to decreasing.



Parent	Child	Average	Maximum	Weighted
Age	Credit Worthiness	0.0904608	0.3	0.0904608
Work History	Reliability	0.19872	0.5	0.19872
Worth	Future Income	0.262222	0.59	0.262222
Ratio of Debts to Income	Credit Worthiness	0.264832	0.45	0.264832
Reliability	Credit Worthiness	0.286044	0.4	0.286044
Payment History	Reliability	0.344613	0.793571	0.344613
Future Income	Credit Worthiness	0.352711	0.499	0.352711
Profession	Future Income	0.362222	0.79	0.362222
Debit	Ratio of Debts to Income	0.377333	0.7	0.377333
Income	Ratio of Debts to Income	0.399333	0.799	0.399333
Income	Worth	0.446806	0.822236	0.446806
Assets	Worth	0.465983	0.898	0.465983

The *Copy* button allows for copying the content of the table to be copied to the clipboard for a later paste into a text editor. Individual elements of the table are separated by TAB characters, which makes it easy to paste the table into a spreadsheet program like Microsoft Excel, for example.

Pressing the *Locate arc* button locates the arc in the *Graph View*.



6.2.8 Controlling values

In causal probabilistic models, there is an additional class of inference problems: Predicting the effects of external intervention. In the context of Bayesian networks, computing the effects of observations amounts to belief updating after setting evidence for the observed network variables. The effect of intervention, on the other hand, is a change in the network structure, related to external manipulation of the system modeled by the network, followed by setting the values of the manipulated nodes and updating beliefs.

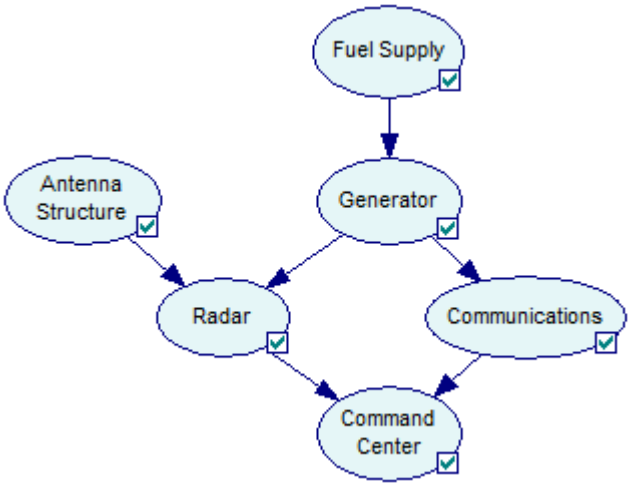
We will explain control values with the help of the following example:

Consider a model for the operational status of a *Command Center*. The command center depends on the status of *Communications* and *Radar*. *Radar* depends on the *Antenna Structure* and the power supplied by the *Generator*. *Communications* rely on the power supplied by *Generator*. The *Generator* relies on *Fuel Supply* to generate power.

We have created a Bayesian network that models these causal relations.

This network is included among the example models as `CommandCenter.xdsl`.

Once this network is loaded, you should see the following diagram in the *Graph View*.



Now suppose the model is an enemy's command center and the decision maker's objective is to disrupt its operations. We can act on the *Communications* by, for example, jamming its outgoing signal with noise. This can be viewed as an external intervention that results in *Communications* not working, i.e., essentially setting the value of variable *Communications* to state *Absent*.

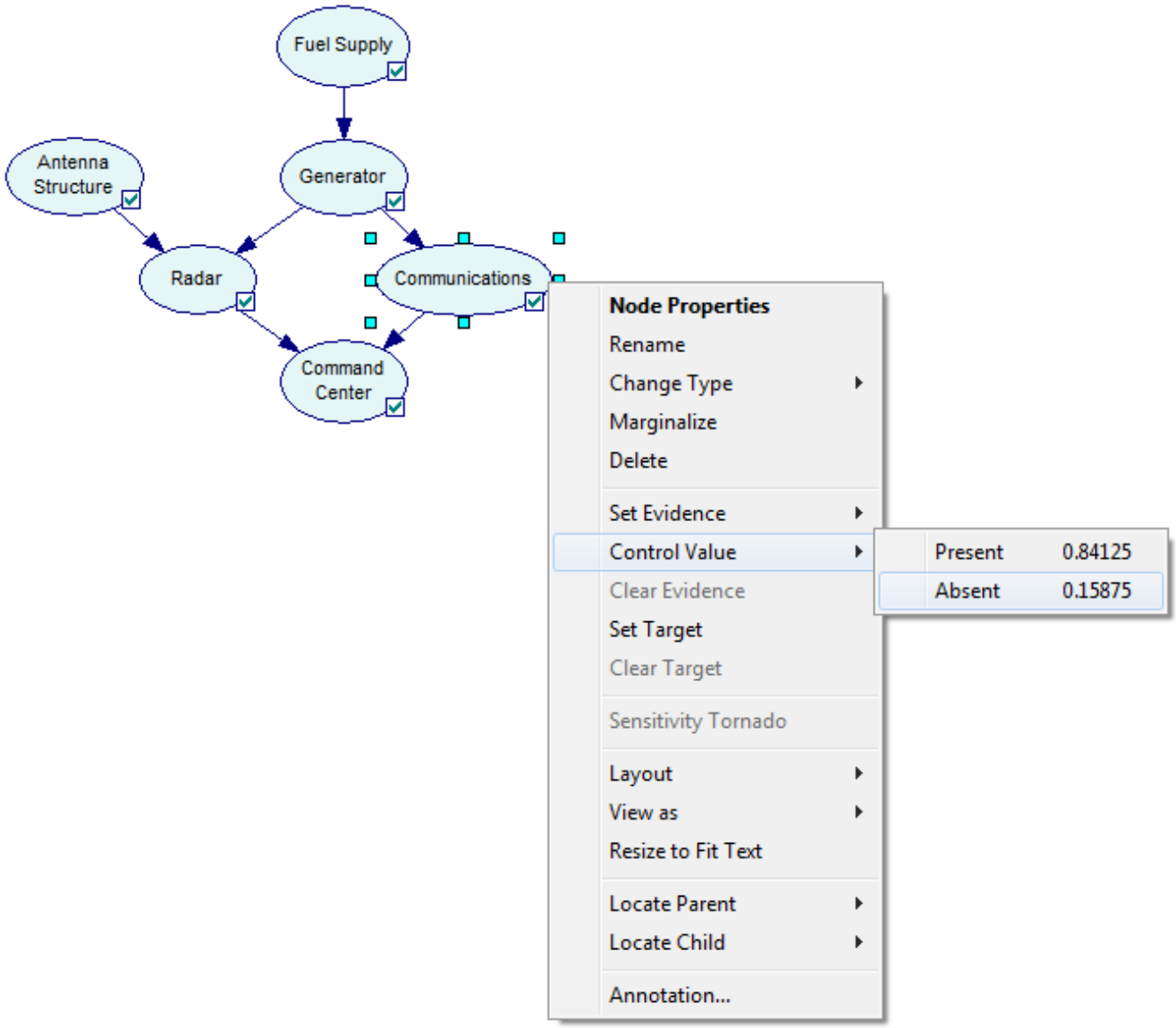
Communications will be *Absent* regardless of the values of other variables in the network (and, in particular, its parent variable, *Generator*).

Control Value is used to model such type of situations.

Let us control the value of the *Communications* node to state *Absent*.

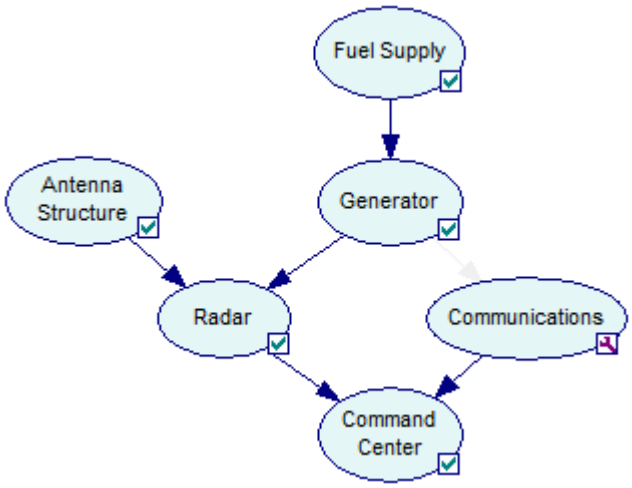
1. Right click on *Communications* node and select *Control Value* from the node's pop-up menu that appears.

2. Select *Absent* from the *Control Value* sub-menu.



This action has the following effects:

Since the state of a controlled node does not depend on the value of its parents, there is a temporary change in the network structure. GeNIe shows this by dimming the arc connecting the parent nodes to the controlled node. In our case, the arc connecting *Generator* and *Communications* is dimmed as shown below:



Now that we have controlled the value of the *Communications* node, GeNIe sets the value of the controlled node to the value imposed by the manipulation. So the value of *Communications* is set to *Absent*. To indicate that the node is controlled, GeNIe displays the status icon on the node. Note that the *Communications* node has the status symbol.

To inspect the effects of intervention, we will need to update the model. After updating the model, you can view the values of each node.

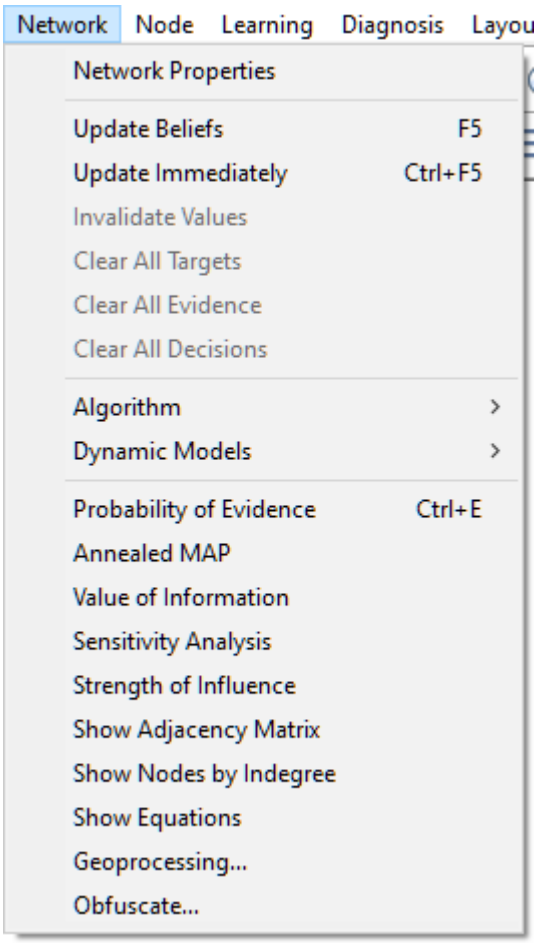
Notice that the intervention only changes the posterior probabilities of the descendants of the controlled node. The control value operation is not available for those nodes that have observed or manipulated descendants. Controlling the value of a descendant of an observed node would lead to a theoretical problem, which one could summarize briefly as a desire to modify the past.

6.2.9 Sensitivity analysis in Bayesian networks

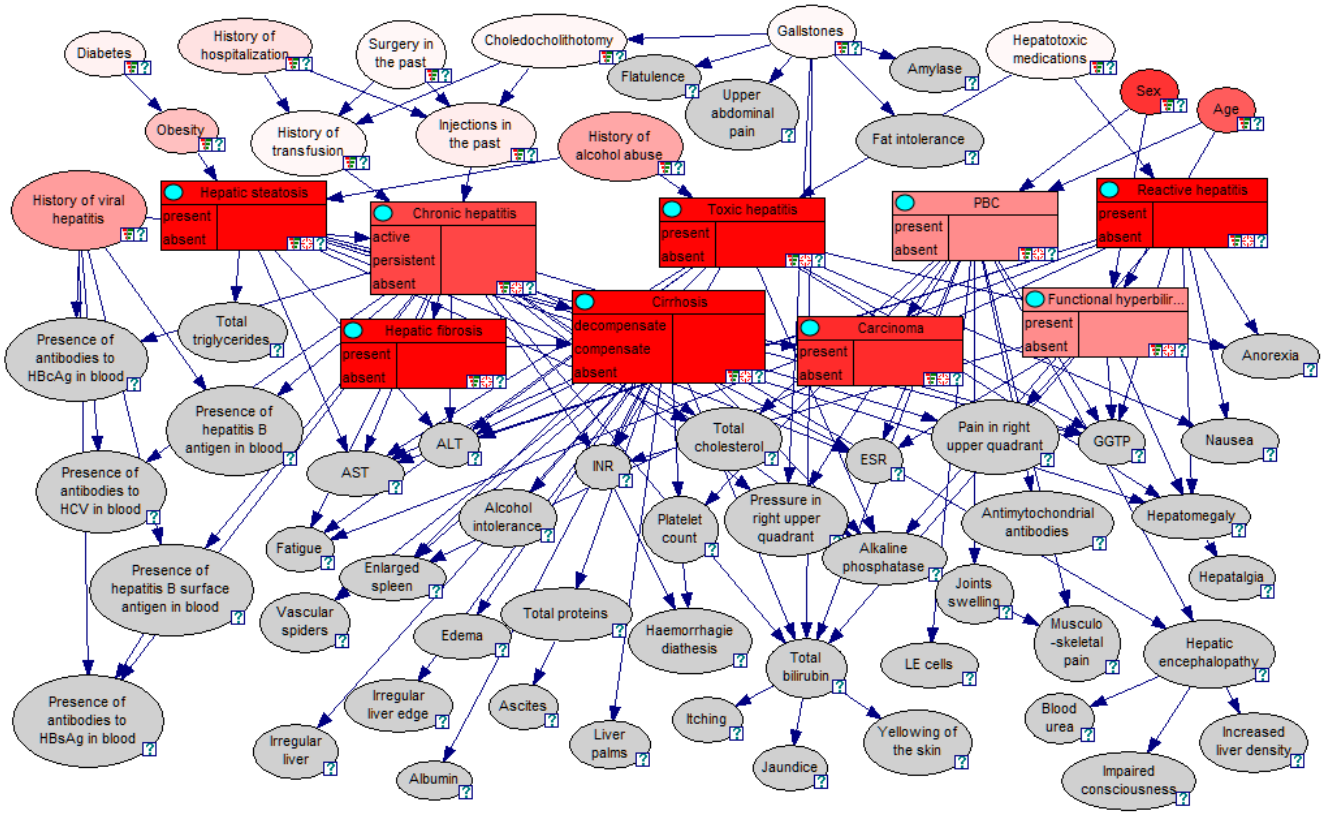
Sensitivity analysis (Castillo et al., 1997) is technique that can help validate the probability parameters of a Bayesian network. This is done by investigating the effect of small changes in the model's numerical parameters (i.e., prior and conditional probabilities) on the output parameters (e.g., posterior probabilities). Highly sensitive parameters affect the reasoning results more significantly. Identifying them allows for a directed allocation of effort in order to obtain accurate results of a Bayesian network model.

GeNIe implements an algorithm proposed by Kjaerulff and van der Gaag (2000) that performs simple sensitivity analysis in Bayesian networks. Roughly speaking, given a set of target nodes, the algorithm calculates efficiently a complete set of derivatives of the posterior probability distributions over the target nodes over each of the numerical parameters of the Bayesian network. These derivatives give an indication of importance of precision of network numerical parameters for calculating the posterior probabilities of the targets. If the derivative is large for a parameter p, then a small change in p may lead to a large change in the posteriors of the targets. If the derivative is small, then even large changes in the parameter make little difference in the posteriors.

To invoke sensitivity analysis in a Bayesian network, choose *Sensitivity Analysis* from the *Network Menu*

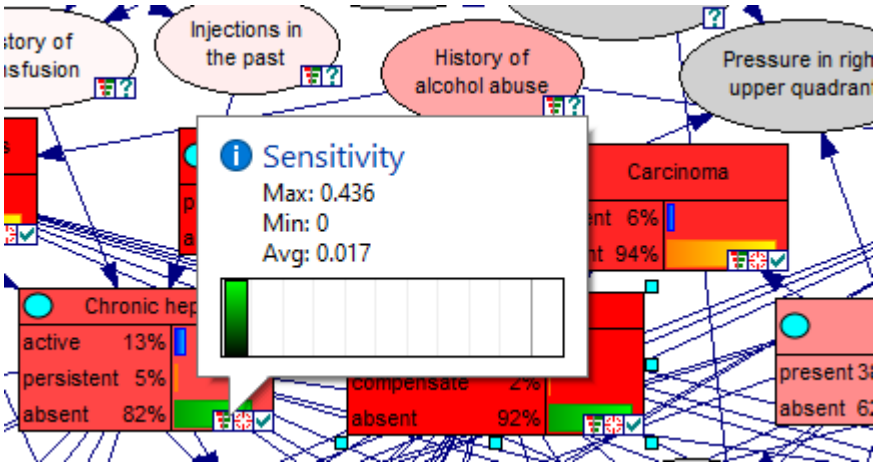


or press the *Sensitivity analysis* ( ) tool on the *Standard Toolbar*. This leads to changing the coloring of the network to indicate where the sensitive parameters are located. Please note that the sensitivity analysis algorithm cannot be invoked unless there is at least one target node defined.

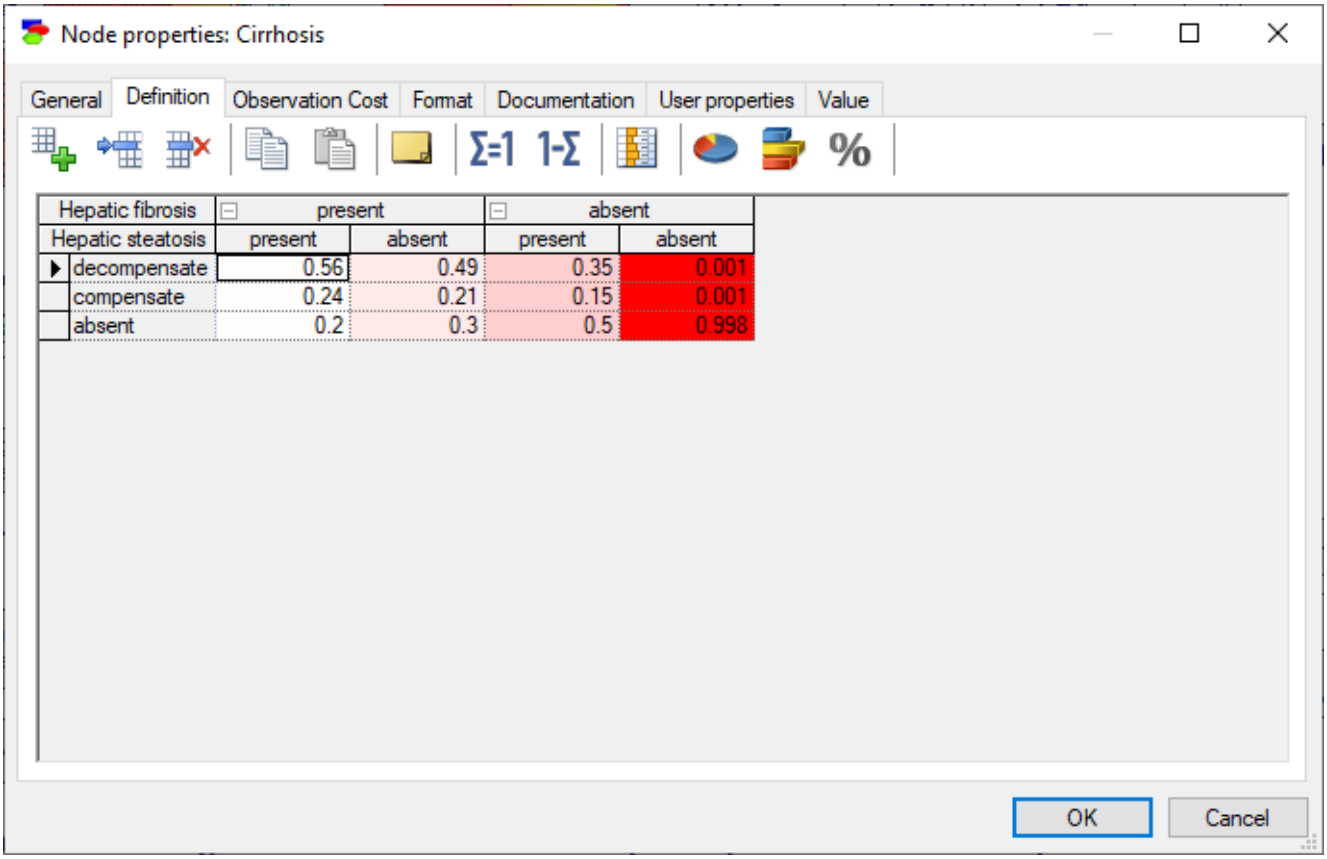


Nodes colored in red contain parameters that are important for the calculation of the posterior probability distributions in those nodes that are marked as targets (in the screen shot above, the rectangular nodes represent various liver disorders and have been marked as targets). Gray-colored nodes do not contain any parameters that are used in the calculation of the posterior probability distributions over the target variables. Sensitivity of any of these parameters is zero and is determined qualitatively, before any computation takes place, based on GeNIe relevance computation layer. It is important to understand that the sensitivity analysis algorithm gives results that are context dependent - the value of derivatives calculated depend on the current target set and the set of observations made in the network. You may perform further observations and clear existing observation, which will prompt the algorithm to recalculate the derivatives and to recolor the graph.

Hovering over the tornado icon in individual nodes shows summary information for those nodes

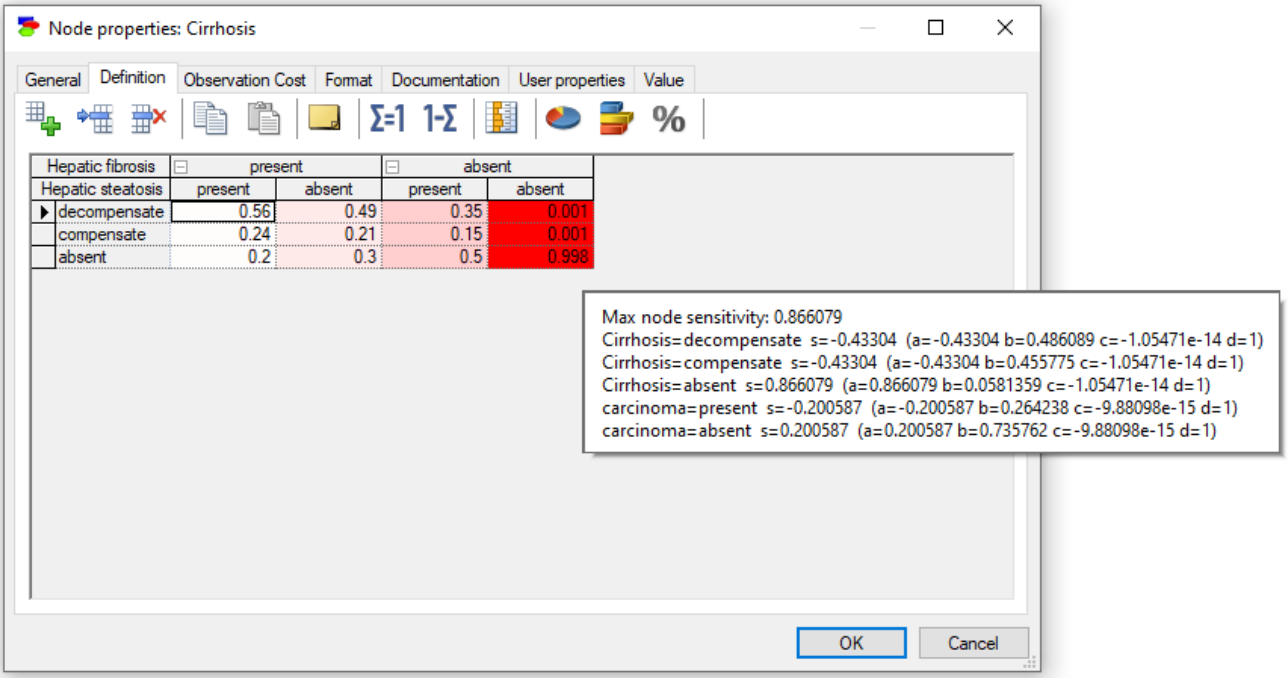


Double-clicking on the node *Cirrhosis* (one of the nodes in red) shows the following view of the definition tab

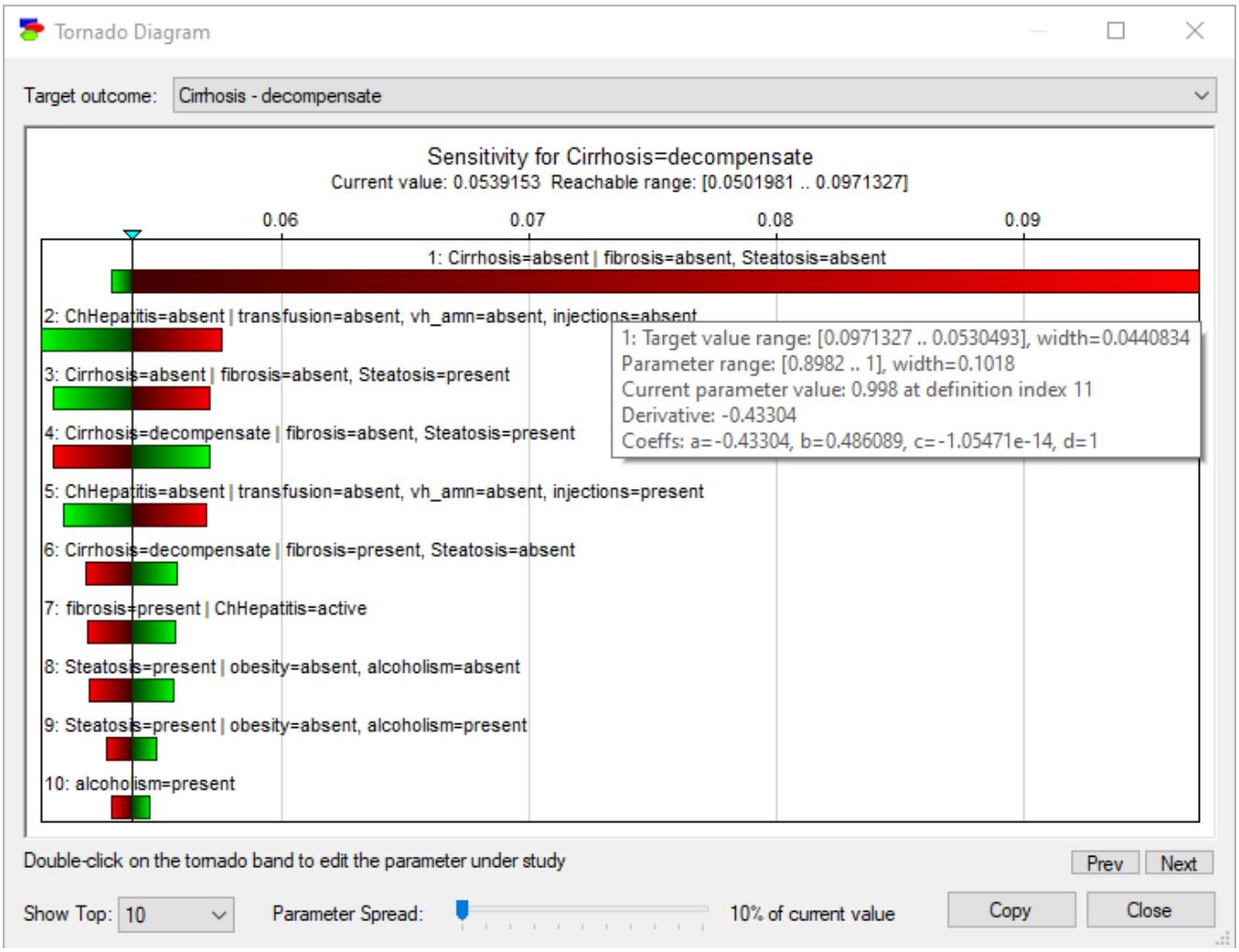


Hepatic fibrosis	present		absent	
Hepatic steatosis	present	absent	present	absent
decompensate	0.56	0.49	0.35	0.001
compensate	0.24	0.21	0.15	0.001
absent	0.2	0.3	0.5	0.998

The coloring of the individual elements of the definition shows those individual parameters that are important. Hovering over them shows additional information from which we can read the numerical values of the computed derivatives



Double-clicking on the small *Tornado* ( ) icon in any target node opens the *Tornado Diagram* dialog. The following screen shot shows the dialog box that opens after clicking on the tornado icon on the node *Cirrhosis*



The diagram shows the most sensitive parameters for a selected state of the target node (in this case, *decompensate* state of the node *Cirrhosis*) sorted from the most to least sensitive. For each parameter, we see its precise location in the model (node and its state conditional on the parents and their states). The bar shows the range of changes in the target state as the parameter changes in its range (in this case 10% of its current value up and 10% down). The color of the bar shows the direction of the change in the target state, red expresses negative and green positive change.

The diagram can be copied (by pressing the *Copy* button or right-clicking and selecting *Copy*) and pasted into another application. Please keep in mind that GeNIe copies the information related to the diagram and it is your responsibility to paste it in the format that you require. The default paste depends on the destination application and this may depend on the application's preferences as set by the user. If you wish to control the paste format, we advise to use *Paste Special*, which allows you to choose a picture or a text describing the sensitivity.

The tornado diagram plot can be customized. We can select the number of parameters shown in the graph between *Top 10* (the default) and *All*. The slider on the bottom of the dialog (*Parameter Spread*) allows us to vary the percentage of change in all parameters (the default is 10%). Clicking on the *Copy* button or right-clicking on the tornado diagram and choosing *Copy* copies the entire diagram to the clipboard for a later *Paste* into other programs, such as Microsoft Word or PowerPoint in bitmap or picture format, explained in the *Graph*

view section. The horizontal axis shows the absolute change in the posterior probability of *Cirrhosis=decompensate* when each of the parameters changes by that percentage.

Hovering over any of the bars shows the exact numerical sensitivities for that bar. In the screen shot above, the gray rectangle shows the parameters for the first bar from the top (*Cirrhosis=absent|fibrosis=absent*, *Steatosis=absent*). Here is a brief explanation of the displayed parameters.

Target value range shows the minimum and maximum posterior probability values for the selected *Target outcome* (in the screen shot above, it is the state *decompensate* of the target node *Cirrhosis*). These minimum and maximum posterior probability values depend directly on the *Parameter Spread* selected.

Parameter range show the minimum and maximum parameter value. Again, these depend directly on the *Parameter Spread*.

Current parameter value shows the nominal value of the probability in the CPT of the node in question. The probability is identifiable uniquely by the states of the conditioning variables (in this case, *fibrosis=absent*, *Steatosis=absent*).

Derivative is the value of the first derivative of the posterior probability T of the selected state of the target node (at its current value) over the parameter p in question. The posterior probability is represented by the following general general linear rational functional form:

$$T=(a * p + b) / (c * p + d) .$$

The sensitivity analysis algorithm calculates the four coefficients (a, b, c, and d). Once we know these, it is trivial to obtain the derivative (which is the basic measure of sensitivity) and target posterior range (see above). *Coeffs* lists the calculated values of a, b, c, and d. The formula for the derivative is:

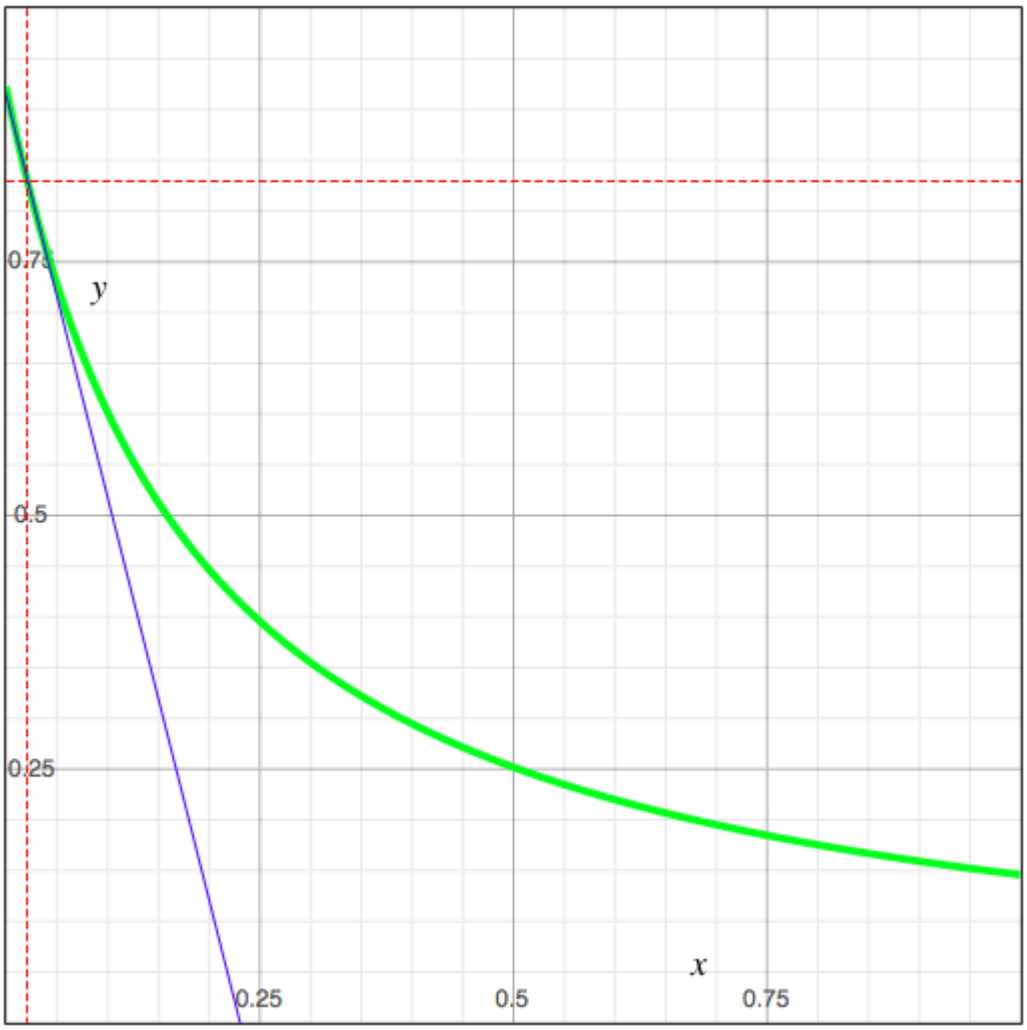
$$D=(a*d - b*c) / (c * p + d)^2 .$$

The denominator is positive, therefore the sign of the derivative is constant for all values of p and, hence, the function is either monotonic or constant. By substituting 0 and 1 for p in the first formula (note that p is a probability), we can calculate how much the posterior will change if p is modified in its entire range. The range is defined by:

$$p_1=b/d$$

$$p_2=(a+b)/(c+d)$$

The sign of a*d-b*c determines which value (of p_1 and p_2) is the minimum and which is the maximum. The following example figure shows a visualization of how the sensitivity is calculated and what it means. x in the plot stands for p, the value of the parameter under study, y stands for T, the posterior probability of the selected state of the target node. The green plot shows this posterior probability as a function of the value of the parameter under study, the blue line, tangent to the green line at the current value of the two parameters, illustrates the derivative.



Sensitivity analysis can be also performed in influence diagrams. It is implemented in such as way that GeNIe executes multiple sensitivity analyses, one for each combination of the indexing parents for the terminal utility node, which is by definition the target. There is no need to set it to be a target (in fact, it is impossible to set it to be a target) - it is a target by default. The captions over the tornado bars help in identifying the scenario.

Sensitivity analysis is essentially an art with few standard procedures. Refining a model involves a search for the most important parameters and paying attention to their precision. The sensitivity analysis, as implemented in GeNIe, is a good first step in this process.

6.3 Influence diagrams

6.3.1 Building an influence diagram

While Bayesian networks allow us to quantify uncertain interactions among random variables and use this quantification to determine the impact of observations, influence diagrams allow us to capture a decision maker's decision options and preferences and use these to determine the optimal decision policy. In order to build a decision model, a decision maker should clearly frame both the problem and the decision to be made.

Decision analysis rests on an empirically verified assumption that while it is relatively easy for humans to specify elements of decisions, such as available decision options, relevant factors, and payoffs, it is much harder to combine these elements into an optimal decision. This assumption suggests strongly that decisions should be supported by means of models. A model supports a decision by computing the expected value (or expected utility) of each decision alternative. The decision alternative with the highest expected gain is, by definition, optimal and should be chosen by the decision maker.

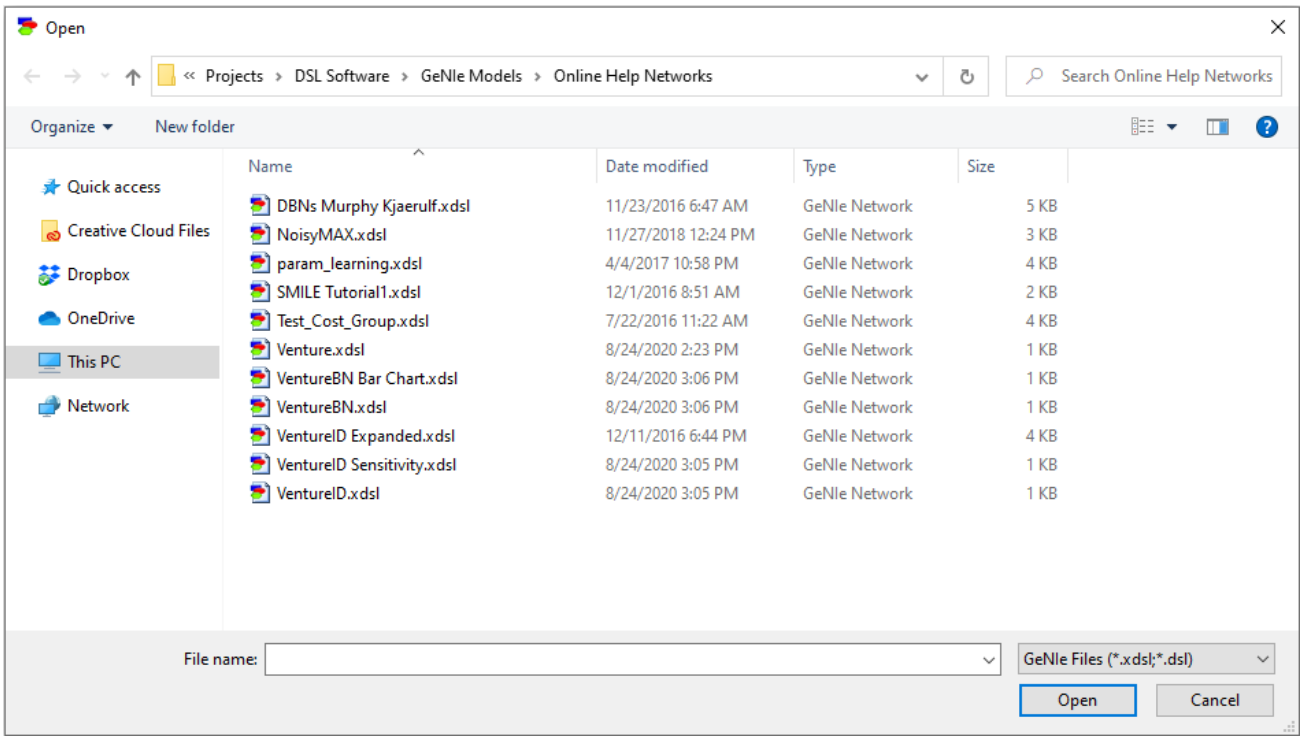
Consider the following scenario based on the example used in the *Hello GeNIe!* section. Let our venture capitalist consider investing $5,000 in the venture. She knows that if the venture is successful, she will earn an additional $10,000 on it. If it is not successful, she will lose her entire investment of $5,000. If she does not invest, her gain will be $500 from a risk-free investment in a bank. We will assume that our capitalist is interested only in financial gain. In case other factors play a role, such as intangible values or non-linearities in the intrinsic value of money, they can be captured by a measure known as utility. GeNIe supports expected utility calculation but leaves it to the user to learn how to measure and represent utility.

We will extend the simple Bayesian network built in the *Hello GeNIe!* section into an influence diagram by adding to it a decision node and a utility node. This is a general principle that is worth remembering: Bayesian networks describe the world, with its complexities and uncertainties, and influence diagrams describe what actions we can take in relation to this world and what values we can expect from these actions. We will use this influence diagram to evaluate two available policy options: *Invest* and *DoNotInvest*.

A. Open the Bayesian network created in the *Hello GeNIe!* *section*. You can find a copy of this Bayesian network in the *Example Networks* folder. It is named *VentureBN.xdsl*.

1. Click on the *Open network* ( ) button on the *Standard Toolbar*.

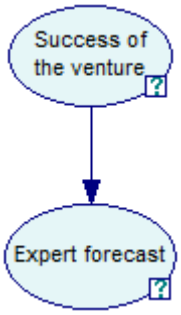
GeNIe will display the *Open* dialog as shown below:



2. Click on *Example Networks* directory.

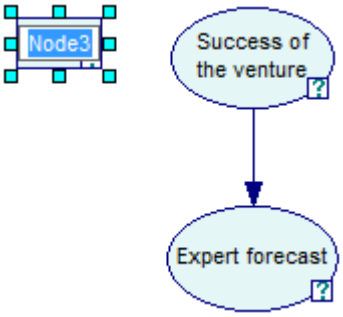
3. Select *VentureBN.xdsl* from the list and click on *Open*.

You should have the following network loaded in *Graph View*:



B. We will create a decision node and define its states. Start by selecting the *Decision* ( ) tool from the *Tools Menu* or the *Standard Toolbar* and click on some empty space near the network.

You can always move the new node around for a more pleasant and readable layout by clicking and dragging it on the screen. Your screen will look as follows

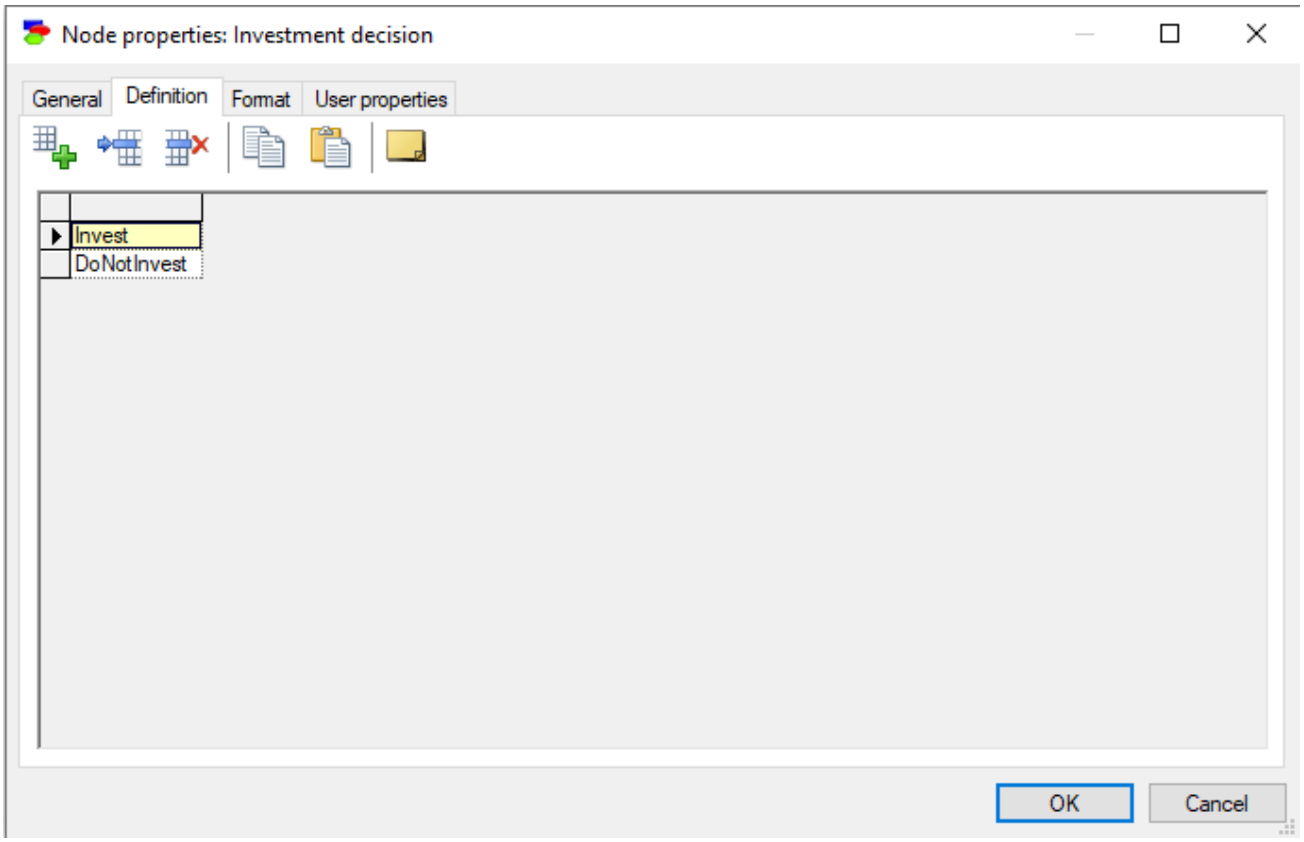


1. Double-click on the created rectangle to open its *Node Properties* sheet.

2. Change the *Identifier* of the new node to *Invest* and its *Name* to *Investment decision*.

3. Click on the *Definition Tab* and change the names of the decision options to *Invest* and *DoNotInvest*.

You will obtain the following:



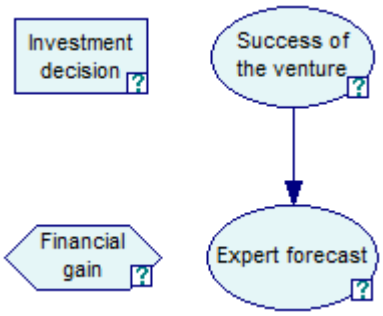
4. Click *OK* to return to *Graph View*.

C. We need to create a value node to represent the utility values for each decision and link it to the diagram.

1. Select the Value ( ) tool from the *Tools Menu* or the *Standard Toolbar* and click on some other empty space near the network.

2. Double click on the node, change its identifier to *Gain* and its name to *Financial gain* and click *OK*.

You should obtain the following:

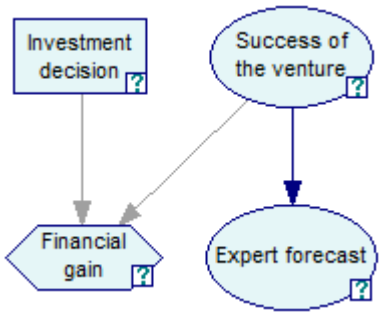


We need to tie the two new nodes (nodes *Investment decision* and *Financial gain*) with the original Bayesian network (nodes *Success of the venture* and *Expert forecast*) using arcs.

The financial gain from the investment depends clearly on whether the investment is made or not and on whether the investment will succeed. We reflect this by adding arcs.

3. Draw an arc from node *Invest* to node *Gain* and from node *Success* to node *Gain*.

The resulting influence diagram should look as follows:

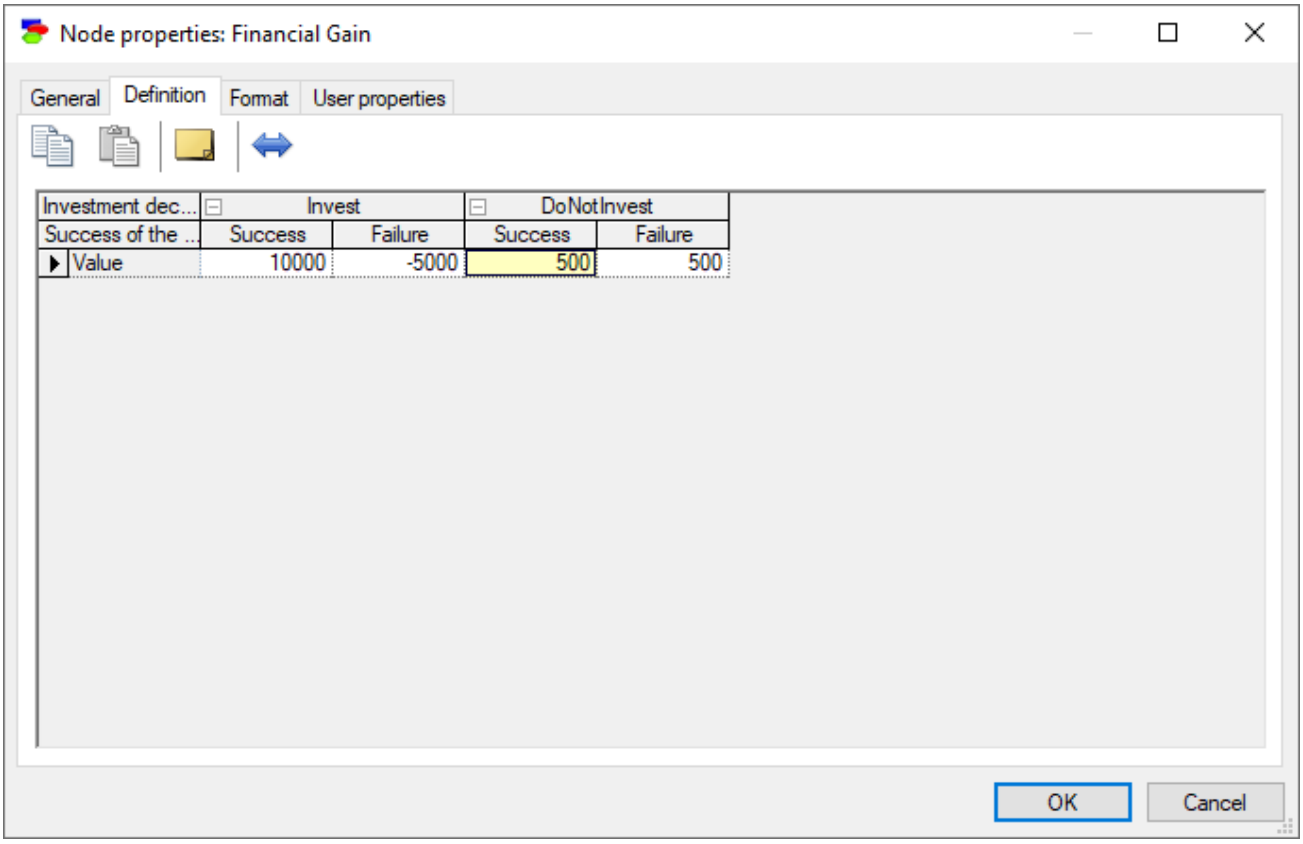


In as much as presence of an arc between two nodes denotes a direct dependence between them, absence of an arc represents independence. And so, the investment decision in this model has no impact on the success or failure of the venture. The only impact that the expert forecast has on the financial gain is indirect, by changing our belief in the success of the venture.

D. Now we are ready to enter the definition of the value node *Gain*.

1. Open the *Node properties* sheet for *Financial gain* by double clicking on the node

2. Click on the *Definition* tab, and enter the different values of gains, as shown below



Please note that *Financial gain* is indexed by both *Investment decision* and *Success of the venture*: we have specified the monetary gain for each possible combination of their values. For example, if the capitalist decides to invest (outcome *Invest*) and the venture ends up in a success (outcome *Success*), the gain is $10,000.

E. We are now ready to solve the diagram, i.e., to determine which of the decision options (*Invest* or *DoNotInvest*) leads to the highest expected gain. Similarly to reasoning in Bayesian networks, we will use the *Update* tool.

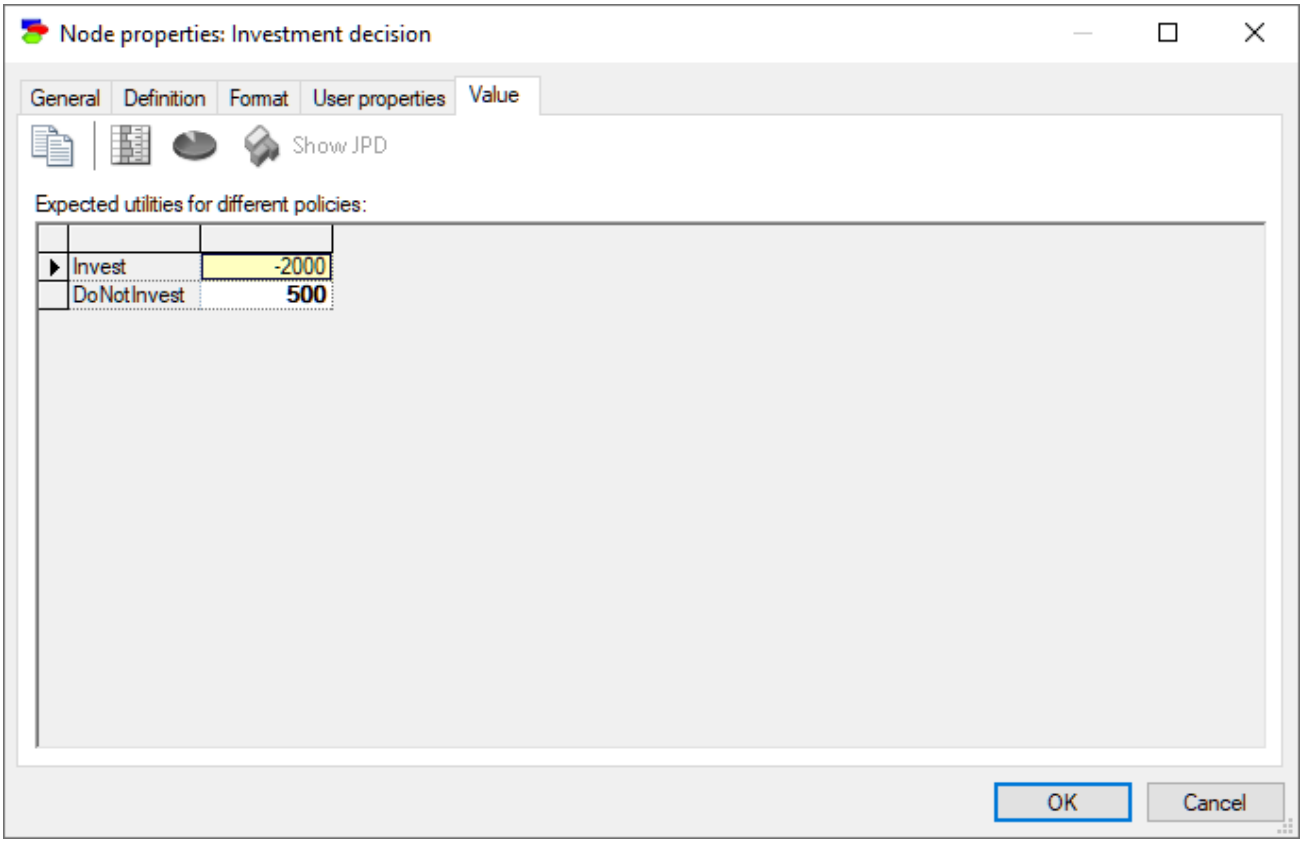
Click on the Update ( ) tool from the *Standard Toolbar*.

This solves the influence diagram.

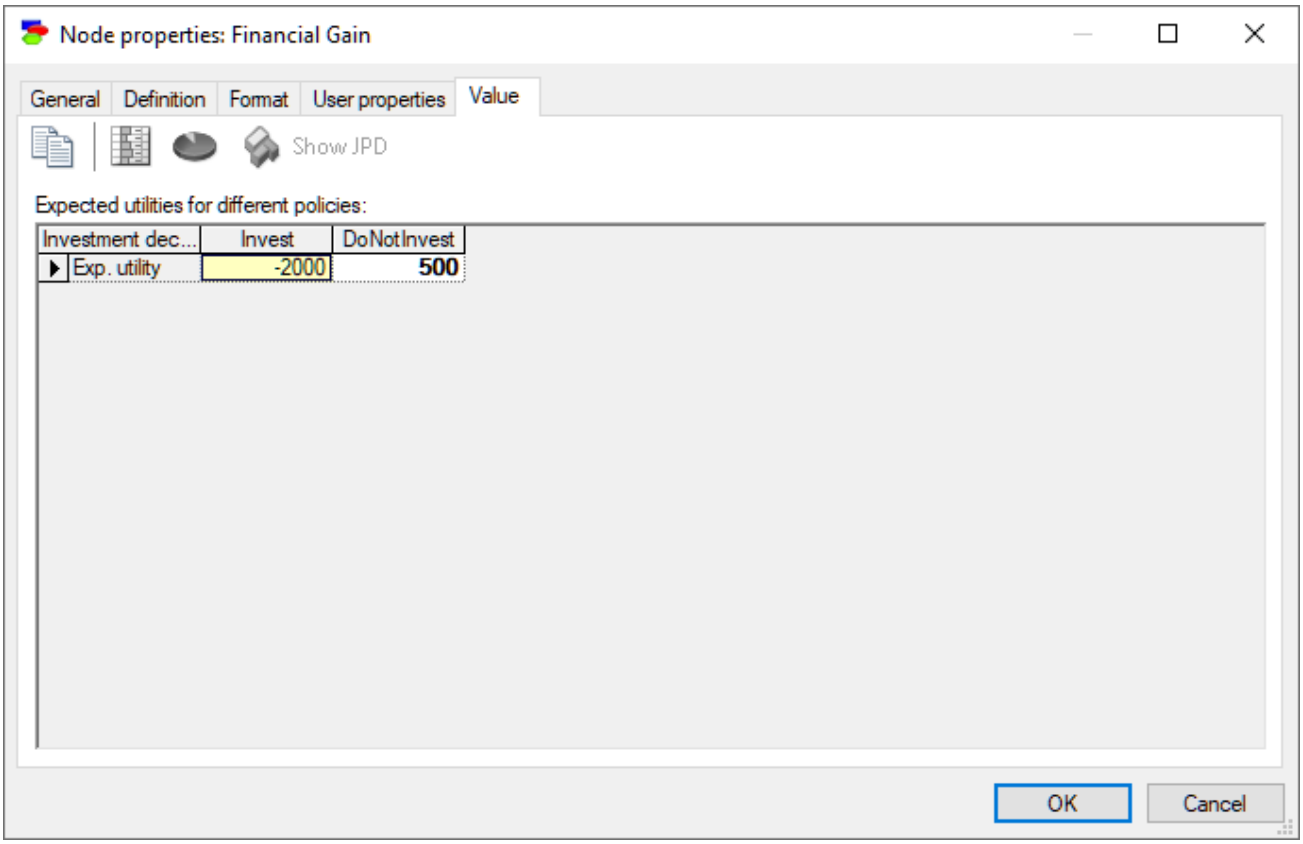
You can examine the solution by double-clicking on either the decision node (*Investment decision*) or the value node (*Financial gain*) and choosing the *Value* tab.

There is a difference in what you will see in terms of the result in each of these, but this difference materializes only in influence diagrams containing multiple decision nodes.

In our diagram, the decision node shows the result as follows:



This result states essentially that the expected value of investment is a loss of $2,000, while the expected value of not investing is a gain of $500. If expected financial gain is the only investment criterion, our venture capitalist should not invest. The value node shows essentially the same result.



GeNIe offers two algorithms for solving influence diagrams: *Policy Evaluation* (default) and *Find Best Policy*, both listed in the *Network Menu*.

To choose an algorithm, click on the appropriate option in the *Network Menu*. GeNIe displays a bullet beside the currently active algorithm.

The *Policy Evaluation* algorithm solves the entire model, exploring all possible combinations of decision nodes and observations. For each of these combinations, it also calculates the posterior distributions of all those nodes in the network that are impacted by them. All this information may be not necessary for some applications, for example all those in which it is enough to identify the best decision option for the next decision step. If the focus of reasoning is finding the best decision option rather than computing the expected values (or expected utilities) of a set of decision options, we suggest that the *Find Best Policy* algorithm be used. The algorithm calculates this best choice much faster than when evaluating all policies. To use this algorithm, set the default influence diagram algorithm to *Find Best Policy* and update beliefs. The algorithm will only calculate the best choice for the first undecided decision node. Once the network is updated, the best choice for the first undecided decision node will be the one containing a "1" in the node value. The remaining choices will contain 0s. The algorithm is applicable only to those influence diagrams, whose first undecided decision node has no undecided/unobserved parents.

You can find the above model among the example models under the name *VentureID.xdsl* included in the distribution version of GeNIe.

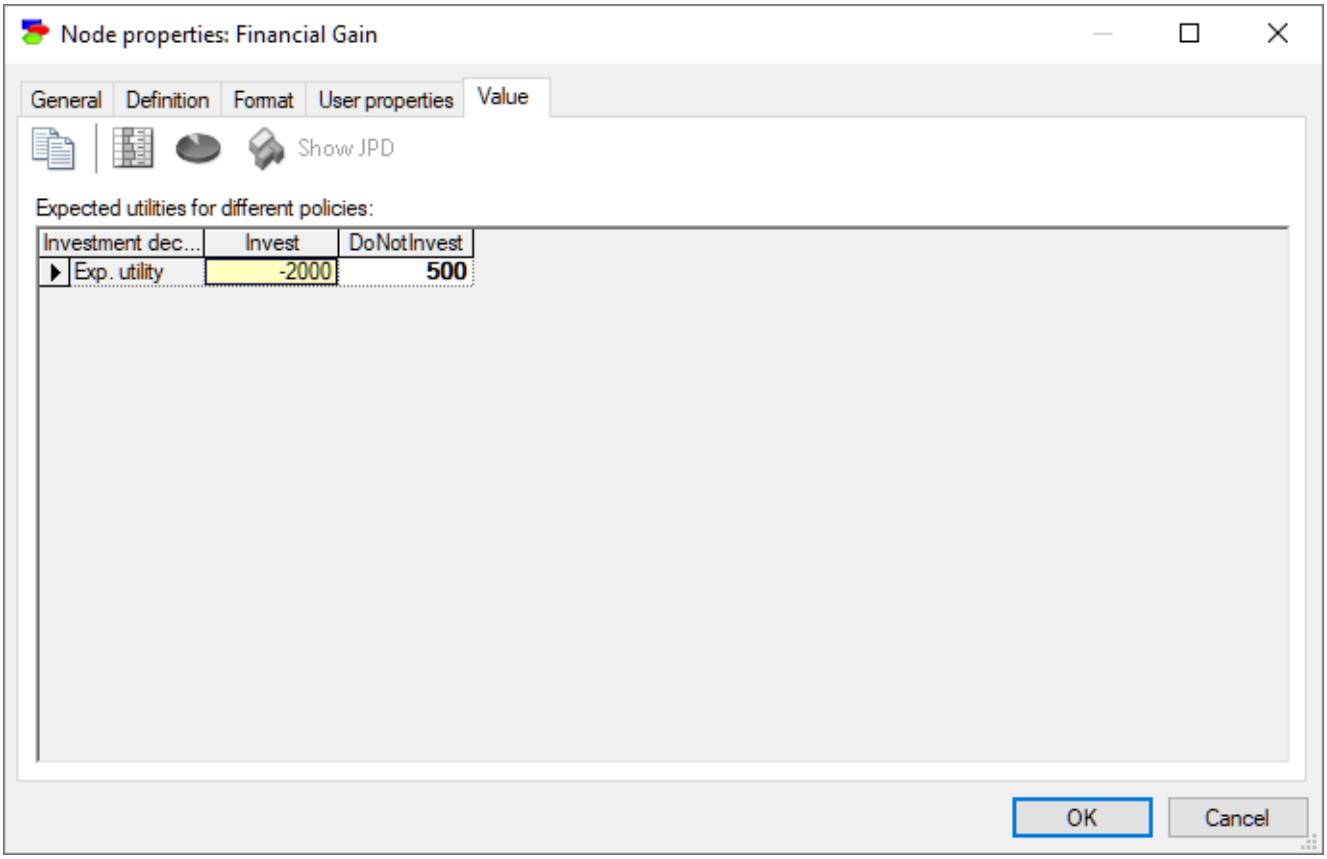
Be careful while saving the model that you have just worked on, because if you choose the *Save* option, then the *VentureBN* model will be overwritten. If you want to save the model that you have just created, choose *Save As* option from the *File Menu*.

6.3.2 Viewing results

Once a model is evaluated, you are ready to view the results computed by GeNIe. This can be done in several ways. Viewing the results of probabilistic inference, captured in chance nodes of the underlying Bayesian networks, has been described in section *Viewing results* in the section on Bayesian networks.

Posterior probability distribution in influence diagrams are more detailed. In cases when a node's probability distribution is affected by a decision or by a node that precedes a decision node, the posterior probability distribution is indexed by the outcomes of these nodes. In such cases, right clicking on a node icon will display a message *The result is a multidimensional table, double click on the icon to examine it*.

The only way to see the entire distribution, is by using the *Value* tab of the decision node or the value node in question. Shown below is the value node for the influence diagram example from the section:



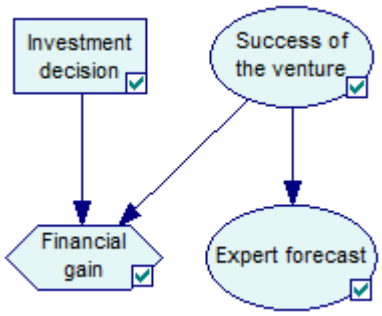
There is a difference in what you will see in terms of the result in each of these, although this difference materializes only in those influence diagrams that contain multiple decision nodes. The value node will show the expected utilities of all combinations of decision alternatives. The decision node will show the expected utilities

of its alternatives, possibly indexed by those decision nodes that precede it. In case decision nodes have predecessors, these predecessors will index the result if they have not been observed before making the decision.

6.3.3 Sensitivity analysis in influence diagrams

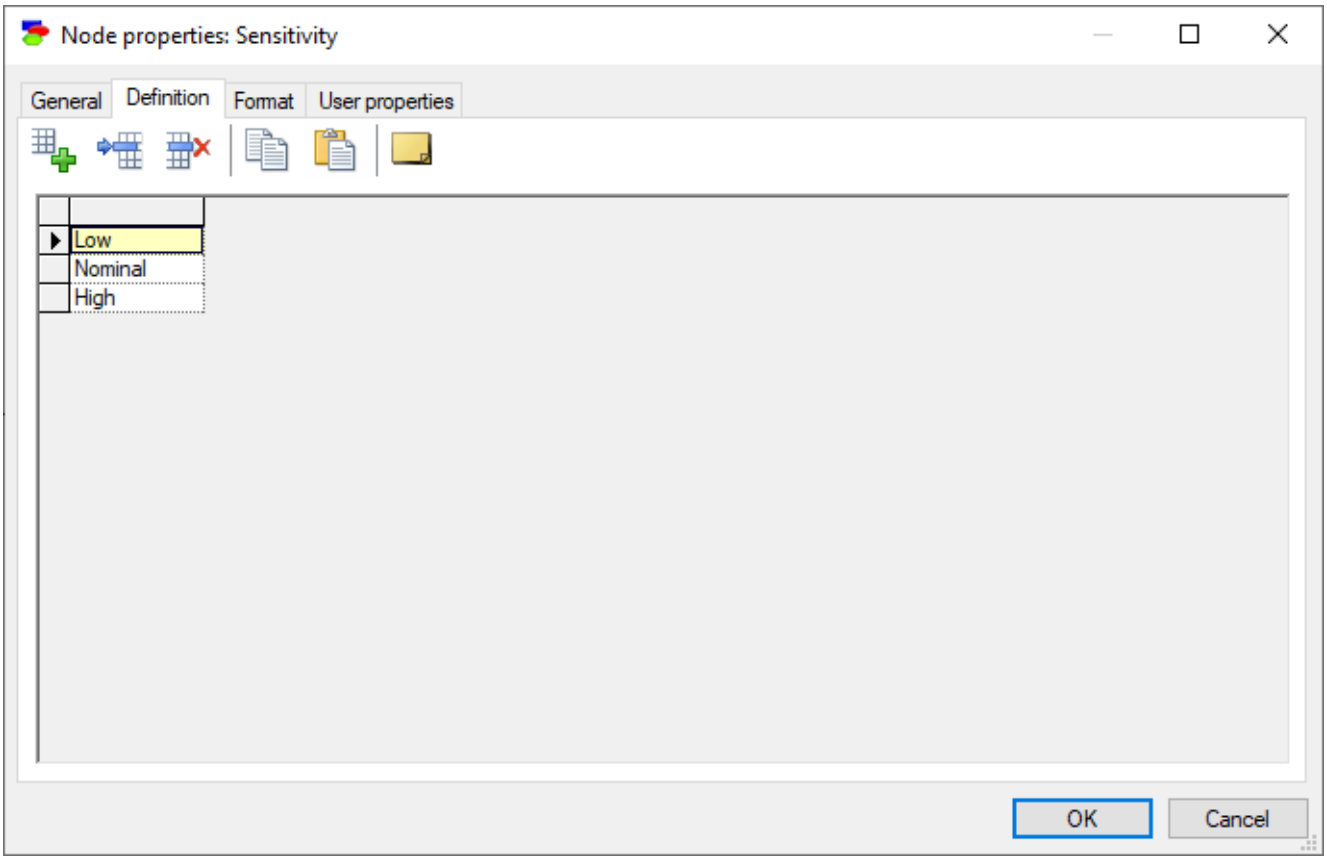
GeNIe supports simple sensitivity analysis in graphical models. To perform sensitivity analysis, add an additional indexing variable that will index various values of parameters in question and have GeNIe compute the impact of these values on the results. We will demonstrate this idea on the example diagram introduced in the *Building an influence diagram* section. If you do not have it saved, you can find a copy in the *Example Networks* folder (it is named `VentureID.xdsl`).

You should have the following network loaded in *Graph View*:

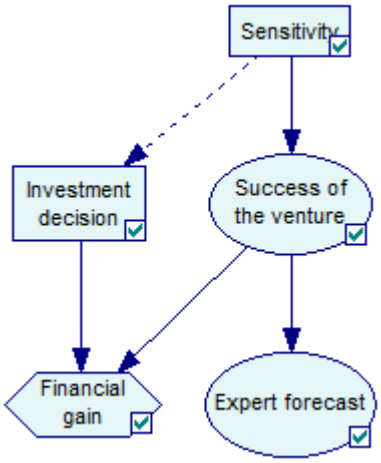


Suppose that we are uncertain as to the actual probability of the success of the venture. Believing that the nominal value of 0.2 is approximately right, we feel that it can be as low as 0.1 and as high as 0.35. To express this, we will add a *Decision* node called *Sensitivity* with three states: *Low*, *Nominal* and *High*.

Create a *Decision node* by selecting the *Decision* ( ) tool from the *Tools Menu* or the *Standard Toolbar* and click on some empty space near the network. Name the newly created node *Sensitivity*, define three states for the node, and name them *Low*, *Nominal* and *High*.

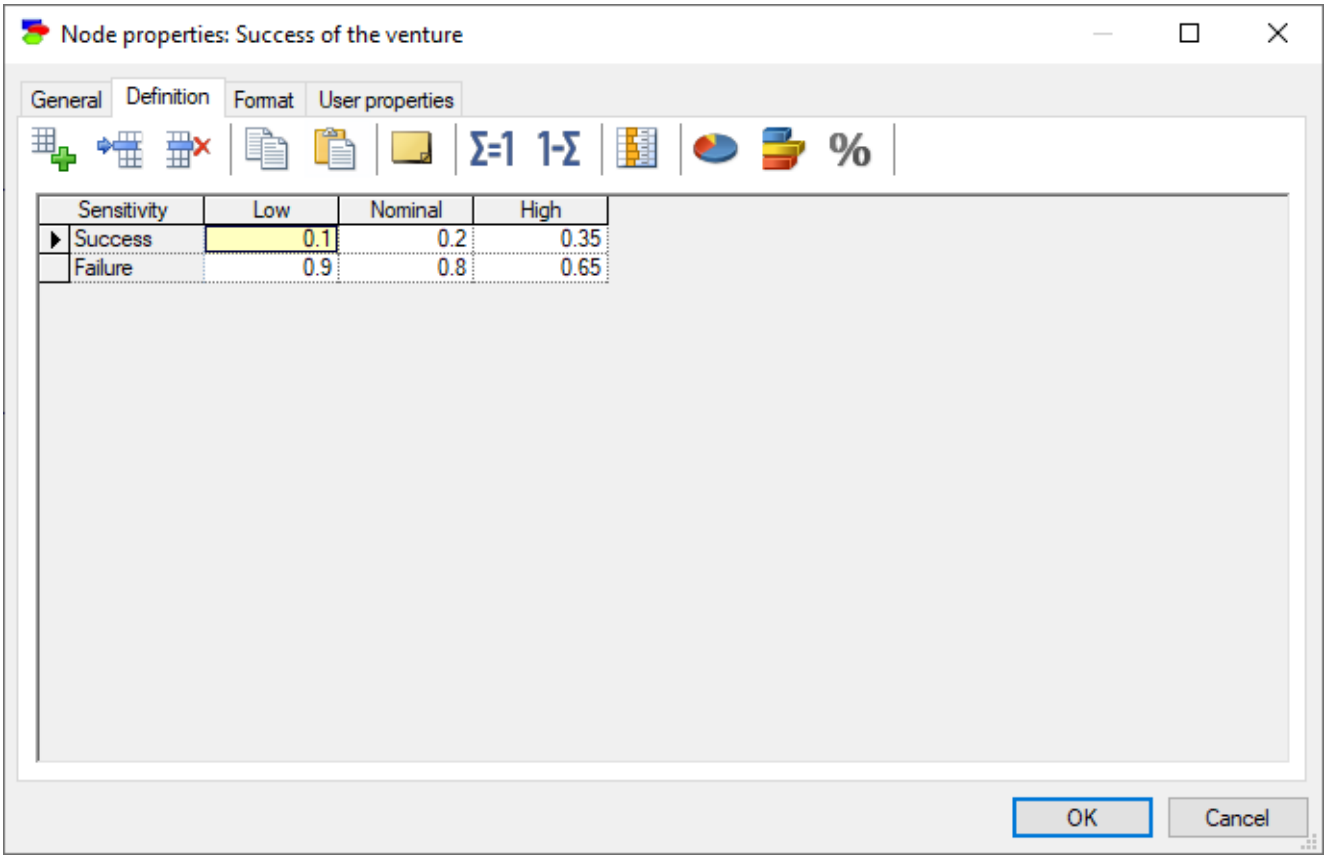


We will use this newly created decision node (*Sensitivity*) to index the *Success of the venture* node and, by this, define a range of values of probability of success. To this effect, we need to add a directed arc from *Sensitivity* to *Success of the venture*. We may add an arc from *Sensitivity* node to *Investment decision* node in order to introduce an explicit temporal order between the two decision nodes. It will be an information arc and it will be dashed. An arc between two decision nodes practically signifies temporal order between the nodes. The model, included among the example networks as `VentureIDSensitivity.xdsl`, will take the following form:



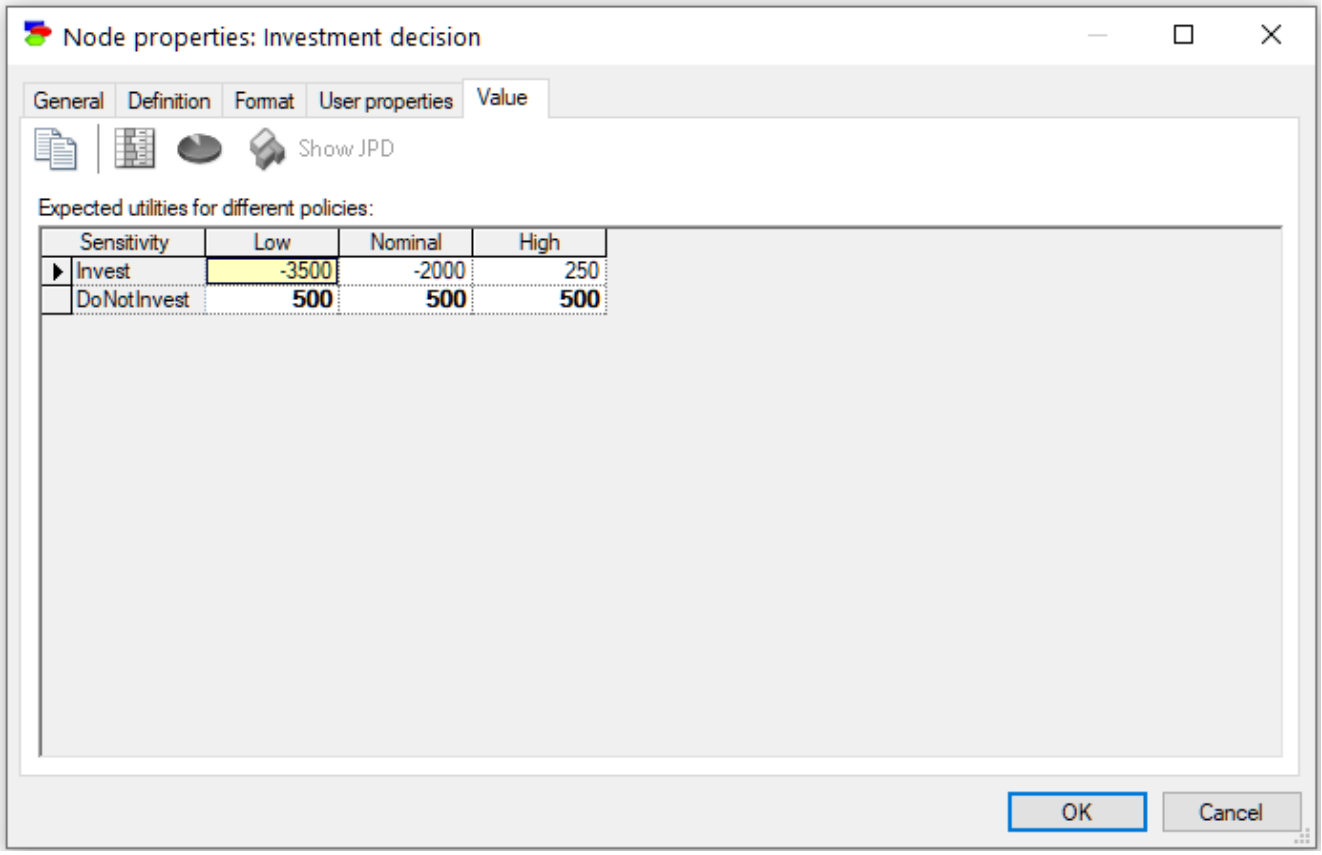
If you forget to add the arc between *Sensitivity* and *Investment decision*, do not worry, because GeNIe will automatically assume the temporal order between the two and draw an arc for you between *Sensitivity* and *Investment decision*.

The states of node *Sensitivity* will index the parameters of *Success of the venture* and will allow to specify their low, nominal, and high values. We enter the low, nominal, and high values for the probability of outcome *Success* in the conditional probability table of the node *Success*. The table should look as follows:



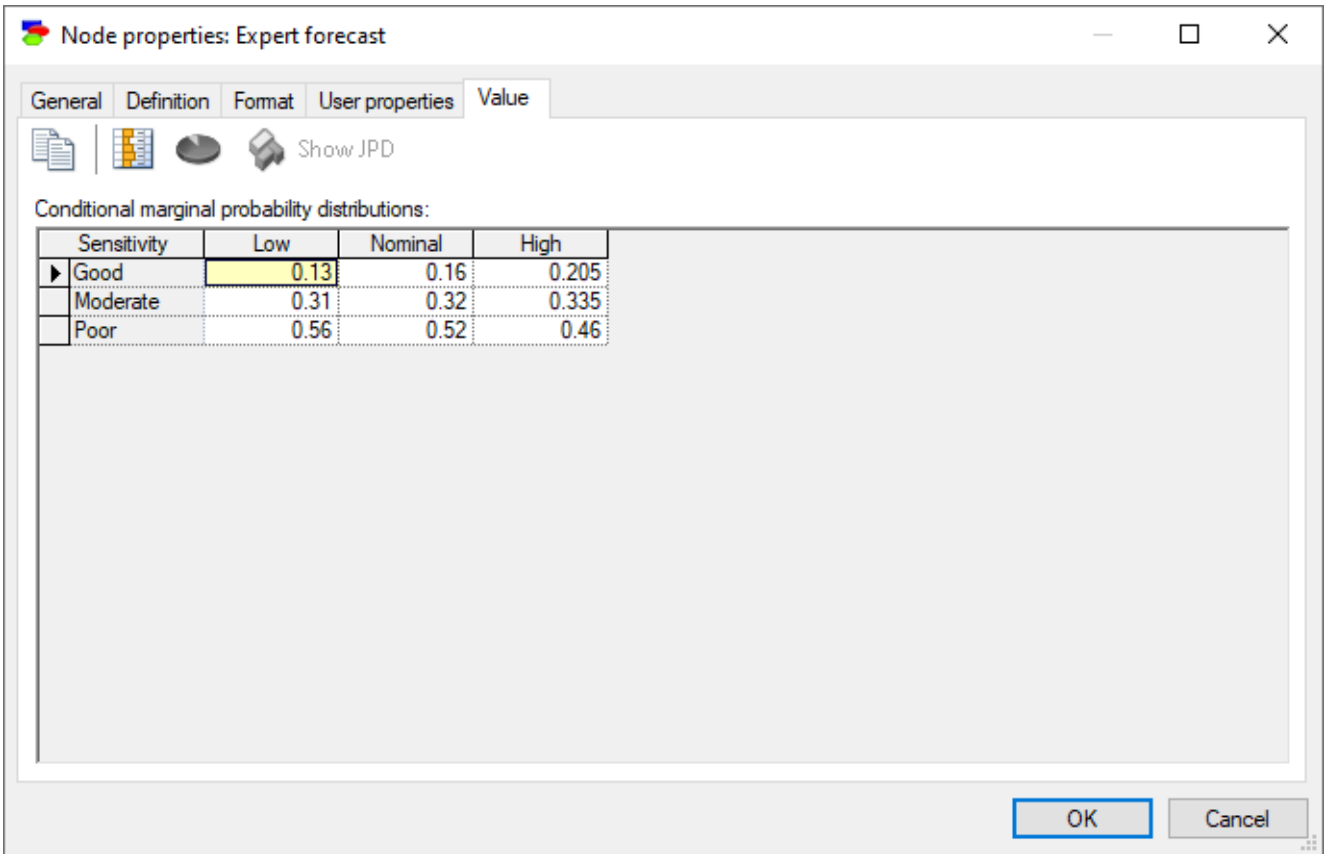
Sensitivity	Low	Nominal	High
Success	0.1	0.2	0.35
Failure	0.9	0.8	0.65

Now we are ready to update the model and observe the results. Update the model (e.g., by pressing the *Update* icon on the *Standard Toolbar*) and open the *Value* tab of *Node Properties Sheet* of the *Investment decision* node. You should see the following:



We can see that even the most optimistic value of the probability of success still does not make *Invest* an attractive option, so our decision is not sensitive to the value of the probability of success.

We can also observe the impact of uncertainty over the probability of success on the posterior probability distribution of any node in the network. In the example above, we can examine the posterior probability of the node *Expert forecast* and see its marginal probability distribution as a function of our initial estimates of the probability of success:



The modified influence diagram for sensitivity analysis is saved as `VentureID_Sensitivity.xdsl` in the *Example Networks* folder.

There is an alternative algorithm for sensitivity analysis that the user may want to choose. It is described in the section *Sensitivity analysis in Bayesian networks*. This algorithm can be executed for influence diagrams as well, in which case, the utility node (in case of a complex structure of utility nodes involving MAU and/or ALU nodes, the terminal MAU/ALU node) is by default the target node. There is no need to set this node to be the target - GeNIe sets it to be the target by default. In influence diagrams, GeNIe runs multiple sensitivity analysis, one for each combination of the indexing parents for the terminal utility node. Interpretation of the results is described in the *Sensitivity analysis in Bayesian networks* section. Captions over the tornado bars should help in identifying the individual scenario (combinations of values of the indexing nodes).

6.3.4 Value of information

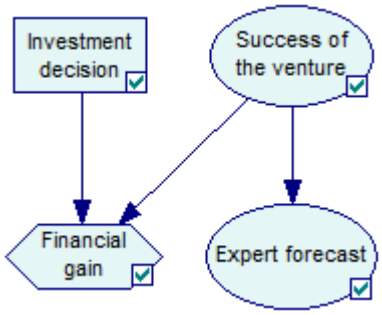
GeNIe allows its user to compute expected value of information (VOI), i.e., the expected value of observing the state of a node before making a decision.

Definition of VOI and the formula for calculating it can be found in any decision-analysis textbook. An up to date list of textbooks covering the field of decision analysis can be found on BayesFusion's web site. Plainly speaking, VOI for an uncertain node x is the expected difference in expected utility (EU) for the two situations: (a) the node x has been observed, and (b) the node x is unobserved. We calculate the expected EU (yes, this is a double "expected") because we do not know a-priori which state of the variable we will observe. There are two important properties of VOI that can be proven easily: (1) Expected EU for (a) can never be smaller than EU for (b). The

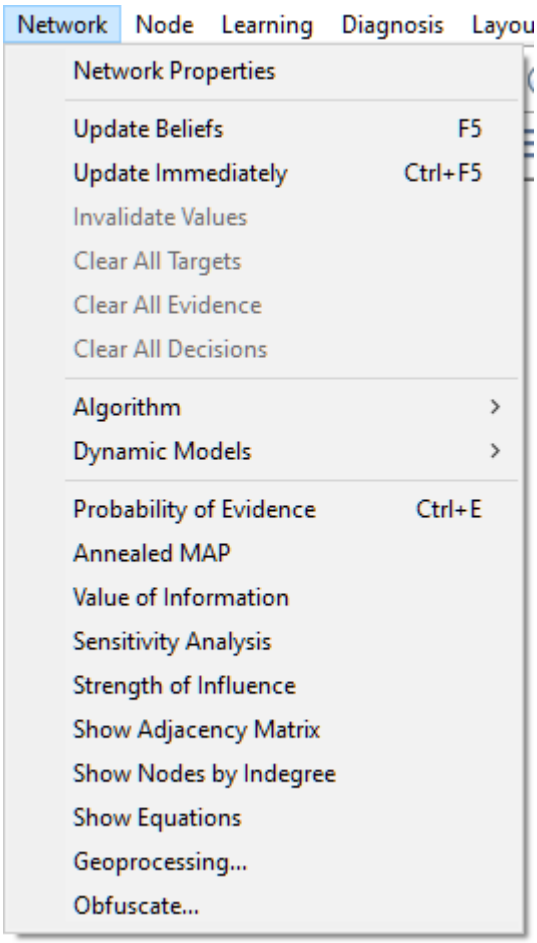
intuition behind it is that it is always better to know than not to know when making a decision. (2) It is always better to gather the information earlier than later, especially in earlier means before making a decision. Vito Corleone's *consigliere*, Tom Hagen, a character in Francis Ford Coppola's interpretation of Mario Puzo's *Godfather*, expresses this property eloquently in a conversation with Mr. Jack Waltz: "*Don Corleone is a man, who insists on hearing bad news immediately*."

We will use the influence diagram created in the *Building an influence diagram* section. If you do not have it saved, you can find a copy in the *Example Networks* folder (it is named `VentureID.xdsl`).

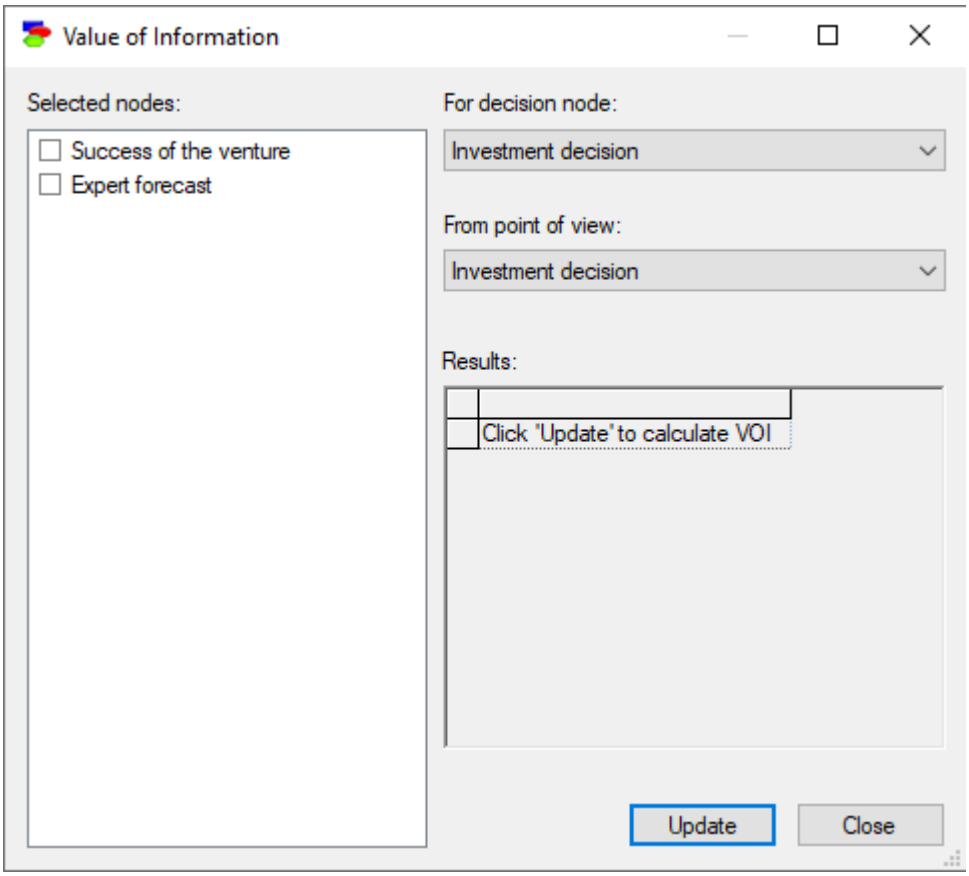
You should have the following network loaded in *Graph View*:



Select the *Value of Information* option from the *Network Menu*:



GeNIe will display the *Value of Information* dialog box as shown below:



The left pane displays all chance and deterministic nodes in the model (our model contains only two of these).

The *For decision node* drop list displays the list of all decision nodes in the model. Because the model contains only one decision node, *Investment decision*, it is the choice. This drop list indicates the decision that will immediately succeed the observation.

The *From point of view* drop list displays the list of all decision nodes in the model. Because the model contains only one decision node, *Investment decision*, it is the choice. This drop list is used to select the node that is the reference point, i.e., from which point of view the value of information is being asked. This field is of importance only when there are more than one decision node and the decision to be made after observing the information is not the first decision to be made.

The *Results* pane displays the results of VOI calculation, once the *Update* button is clicked.

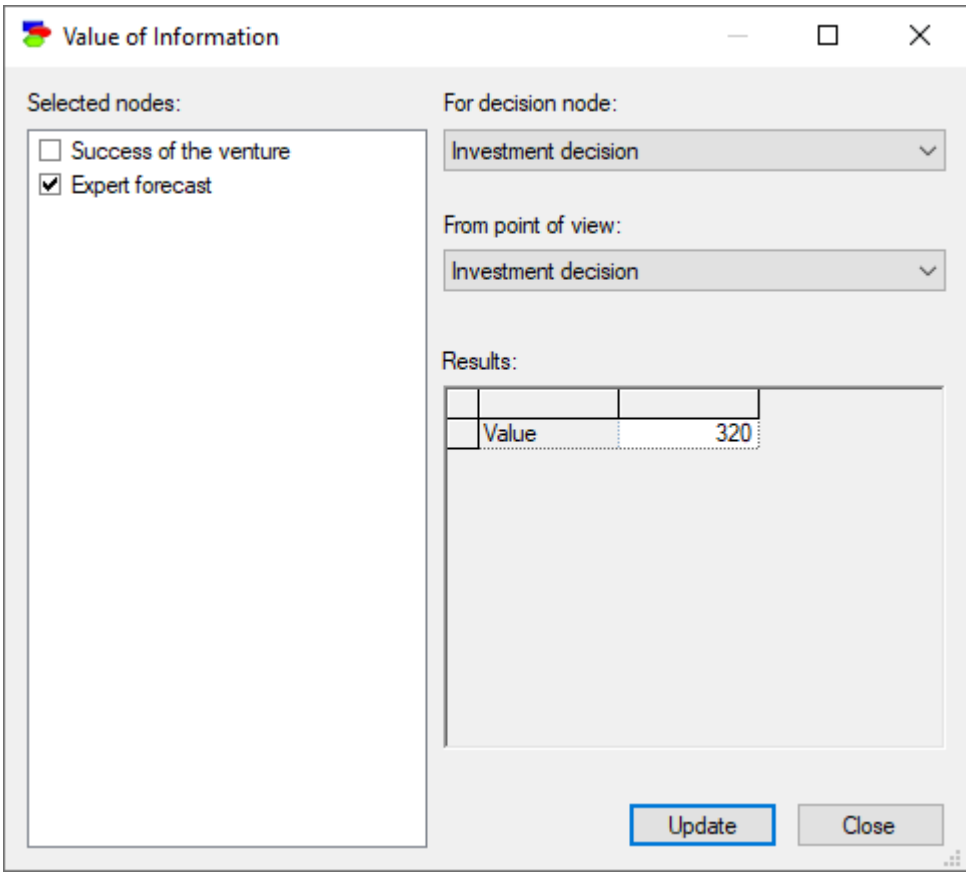
Suppose we want the value of information of the node *Expert forecast*. We proceed as follows:

1. Check the box next to *Expert forecast* in the left pane.

Because there is only one decision node, it is automatically selected in the *For decision node* and *From point of view* drop lists.

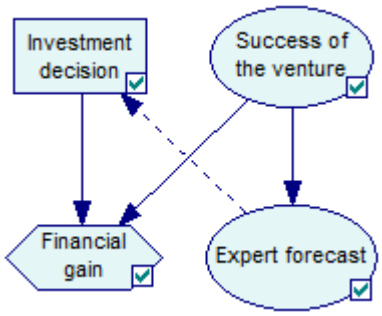
2. Click on the *Update* button to calculate VOI.

GeNIe will display the calculated value of information for the node *Expert forecast*, i.e., value of observing it before making the decision *Investment decision*, in the lower-right pane, as shown below:



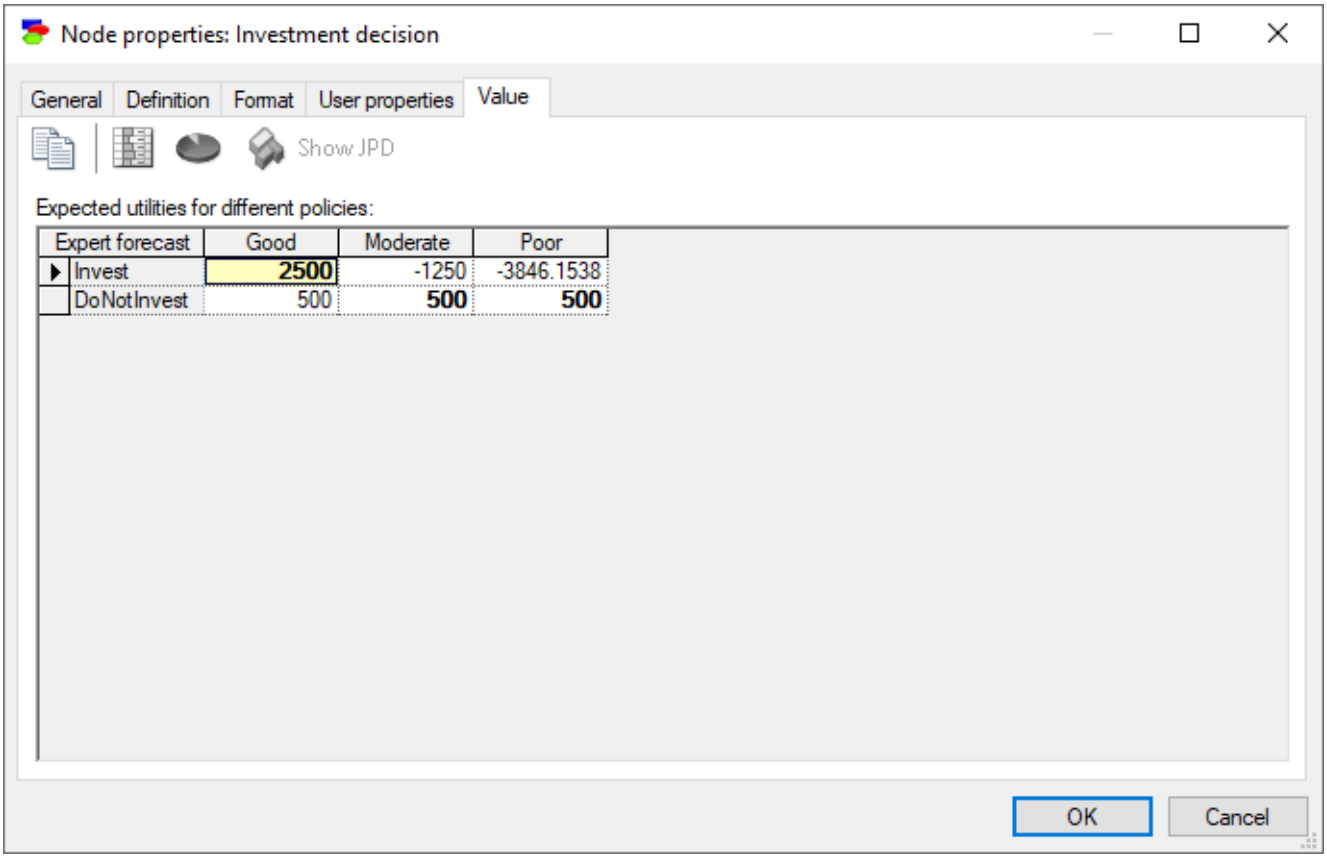
The forecast of our expert is worth $320 to our investor. This is the difference between the expected value of the optimal decision given perfect information about the node *Expert forecast* and the expected value of the optimal decision given no information about the node *Expert forecast*. A positive value means that knowing the value of the node *Expert forecast* will improve the expected value of the decision. A zero value would mean that learning the value of the node *Expert forecast* before making the decision has no impact on the decision. It can be proven that VOI calculation always yields a non-negative value - the intuition behind this is that we are never worse off by obtaining more information (in other words, it never makes sense not to know something that we could know).

Let us now examine the decisions that the investor will face after she has heard the forecast. To do so, we need to add an arc between the node *Expert forecast* and the node *Investment decision*, obtaining the following diagram:



Please note that the arc between the nodes *Expert forecast* and *Investment decision* is dashed. Arcs entering decision nodes have a special meaning in influence diagrams - they are informational arcs and denote the fact that the decision maker will know the outcomes of the nodes at the tails of the informational arcs before she makes the decision.

Update the values using *Update* ( ) button from *Standard Toolbar*. After updating the values we observe the following result in the *Value* tab of the node *Investment decision:*



Expert forecast	Good	Moderate	Poor
Invest	2500	-1250	-3846.1538
DoNotInvest	500	500	500

Our investor should ask expert opinion if expert opinion costs less than $320 (please note that this was the calculated value of information for the node *Expert forecast*) and in case the forecast is *Good*, she should invest

in the venture. If the forecast is *Moderate* or *Poor*, however, she should not invest, as safe investment in a bank yields a higher expected value.

The above procedure computes the expected value of perfect information (EVPI). In order to compute the expected value of imperfect information (EVII) for a node *N*, you need to specify the reliability of the source of information about *N* by an additional node *M*, a direct descendant of *N* and then compute the EVPI for *M*. This will be equal to the EVII for *N*. This is, in fact, precisely what is being done by nodes *Success of the venture* and *Expert forecast*. We would love to know the value of the node *Success of the venture*. However, this is unattainable and the only thing we can obtain is imperfect information through *Expert forecast*. Node *Expert forecast* is such an imperfect source of information about the node *Success of the venture*. This imperfection is characterized by the probability distribution conditional on node *Success of the venture*. Within an influence diagram, there is no distinction between the value of perfect and imperfect information. Value of information, as calculated in an influence diagram is always value of perfect information. It is the relationship between the observed variable and the variable that we are really interested in that make information perfect or imperfect.

Finally, we would like to caution our users against two common misconceptions of VOI that we have encountered in the past and that we responded to in our *Forum*. The first misconception relates to calculation of VOI for nodes that have deterministic marginal probability distributions in the sense of all but one of the probabilities being zeros. This happens, for example, in case of deterministic parent-less nodes (these are generally a bad modeling practice anyway) or in case of chance nodes that have been observed. VOI calculated for such nodes will be zero. The intuition behind this is that there is no uncertainty over such nodes and learning what we already know is essentially worthless from the point of view of decision making. The second misconception occurs when we ask for the value of information for a node that is a descendant of the decision node. Such a case is difficult to interpret theoretically and is essentially dismissed by GeNIe with a VOI equal to zero. On the one hand, the decision influences the marginal probability distribution over all of its descendants. On the other hand, when calculating the VOI over any of these nodes, these marginal will have to take part in the calculation. This is circular and, as we said above, difficult to interpret theoretically. A viable possibility in such a case is converting the influence diagram to so called *canonical form*, described in detail in a paper by Heckerman and Shachter (1995). In the canonical form, all descendants of the decision nodes are deterministic and all uncertainty resides in their ancestors. This modeling trick allows for asking VOI of these ancestors, which is typically the users' intention.

6.4 Support for diagnosis

6.4.1 Introduction

Diagnosis is one of the most successful applications of Bayesian networks. The ability of probabilistic knowledge representation techniques to perform a mixture of both predictive and diagnostic inference makes it very suitable for diagnosis. Bayesian networks can perform fusion of observations such as patient (or equipment) history and risk factors with symptoms and test results. This section reviews special features of GeNIe that support diagnostic applications.

A diagnostic model built using GeNIe represents various components of a system, possible faulty behaviors produced by the system (symptoms), along with results of possible diagnostic tests. The model essentially captures how possible defects of the system (whether it is a natural system, such as human body, or a human-made device, such as a car, an airplane, or a copier) can manifest themselves by error messages, symptoms, and test results. Using such a model, GeNIe produces a ranked list of the most likely defects and a ranked list of the most informative and cost-effective tests. The following sections assume that you are already familiar with using the plain version of GeNIe.

In section *Enabling diagnostic extensions*, we will learn how to enable the diagnostic features of GeNIe and how to define individual nodes of a model and their properties for the purpose of diagnosis.

Section *Spreadsheet View* discusses a special extension of GeNIe that is useful in rapid model building - all properties of every variable are listed in one window and the user specifying a model can move rapidly between variables and enter their specifications into the model.

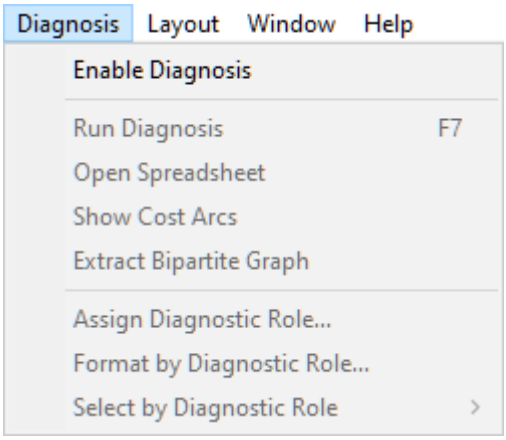
Section *Diagnosis window* describes a special dialog window that allows the user to use a diagnostic model on real cases. The window allows for observing symptoms and signs, entering test results, and seeing GeNIe's suggestions as to what tests to perform next and what the probabilities of various faults are.

Section *Diagnostic case management* discusses how diagnostic cases can be saved to and retrieved from permanent storage (disk files).

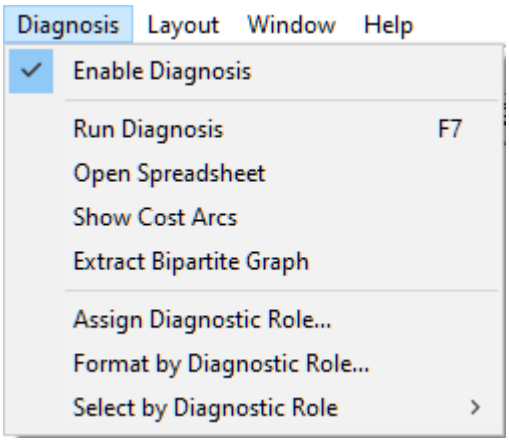
Finally, section *Cost of observation* covers encoding and using costs of diagnosis as part of the model.

6.4.2 Diagnosis menu

Diagnosis menu is mostly grayed out, except for the choice *Enable Diagnosis*, which turns on GeNIe's diagnostic extensions



Diagnostic extensions are turned on or off for each open model individually. They are turned on automatically for those newly loaded models that have diagnostic information specified but are turned off for any other, including those newly created models. When diagnostic extensions are turned on, all other items on the *Diagnosis menu* are activated, the *Run diagnosis* ( ) button becomes active, and node property sheets are also extended with special diagnostic functionality.



Run Diagnosis command (shortcut *F7*) opens the *Diagnosis window*, which allows for performing diagnosis interactively with the current model.

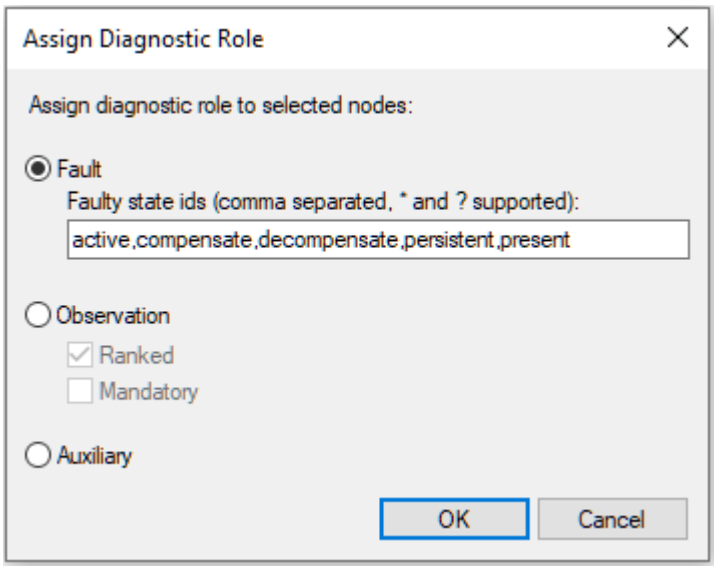
Open Spreadsheet command opens the *Spreadsheet View* for the current network in a separate window. The *Spreadsheet View* is a special extension of GeNIe that is useful in rapid building of diagnostic models - all properties of every variable of a model are listed in one window and the user specifying a model can move rapidly between variables and enter or modify their specifications.

Show Cost Arcs command toggles the *Graph View* to *Cost Graph View*. The *Cost Graph View* allows for specifying conditional costs of performing tests and making observations. See *Cost of observation* section for more information.

Extract Bipartite Graph command extracts from a diagnostic model a two layer network, in which nodes in the upper layer are composed of all *Fault* nodes, all nodes in the lower layer are *Observation* nodes, and there are arcs connecting *Fault* nodes with *Observation* nodes, i.e., there are no arcs between nodes in the same layer. No changes are made to the original model by this operation. Bipartite graphs are structurally simpler than general

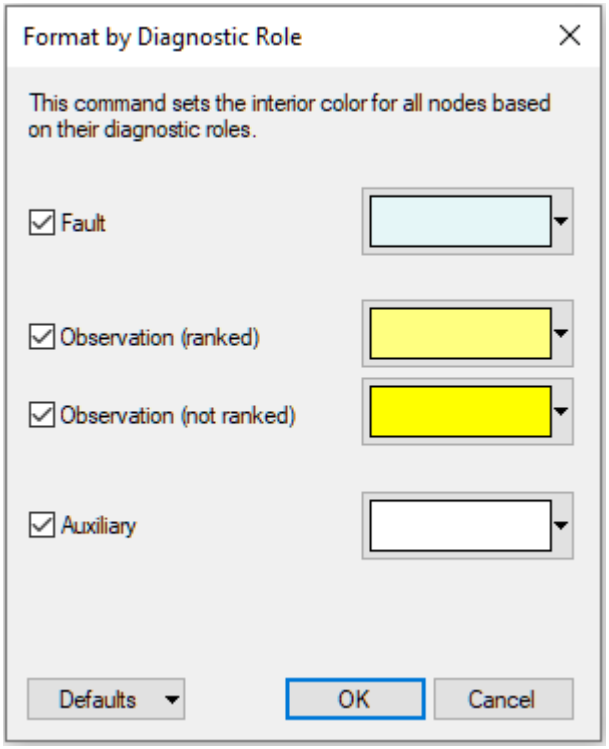
graphs, which often goes together with reduced computational complexity, and are of interest to some knowledge engineers building diagnostic models.

The *Assign Diagnostic Role...* dialog invokes a dialog that allows for setting diagnostic roles for groups of nodes.

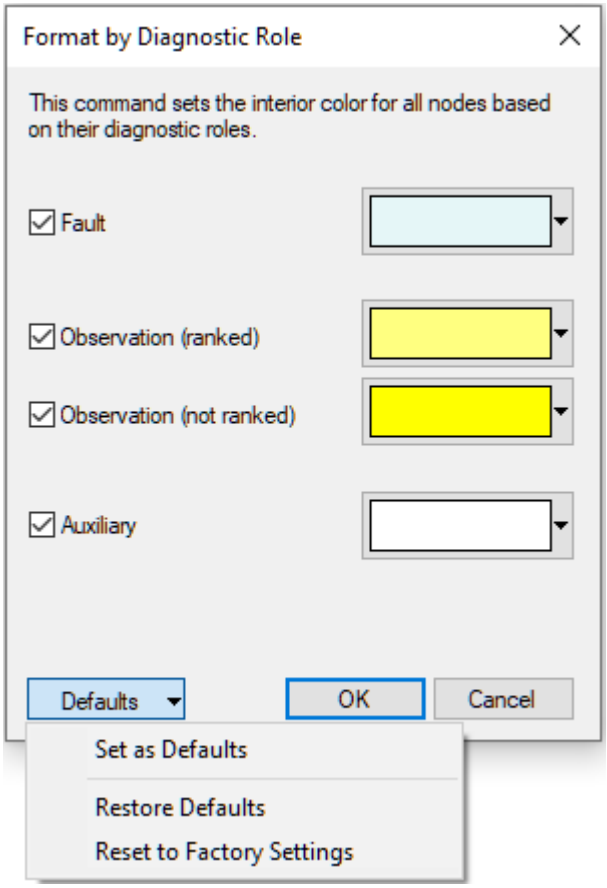


All nodes selected at the time that the dialog has been invoked will be assigned the selected diagnostic role. In case of *Faults*, one has to select those states of the nodes that represent faulty states. The small window right below the selection allows for specifying state IDs that denote faulty states. The window is pre-filled with state IDs that are currently designated as fault states and accepts wild card characters, such as * and ?, with role identical to the role that they play in most computer systems. In case of *Observations*, one can select whether they should be Ranked and/or Mandatory. The meaning of these two properties as well as the diagnostic roles in general are explained in the *Diagnosis Window* section.

The *Format by Diagnostic Role...* option invokes a dialog that allows to specify coloring of nodes based on their diagnostic properties.



Clicking on the *Defaults* displays the following pop-up menu:

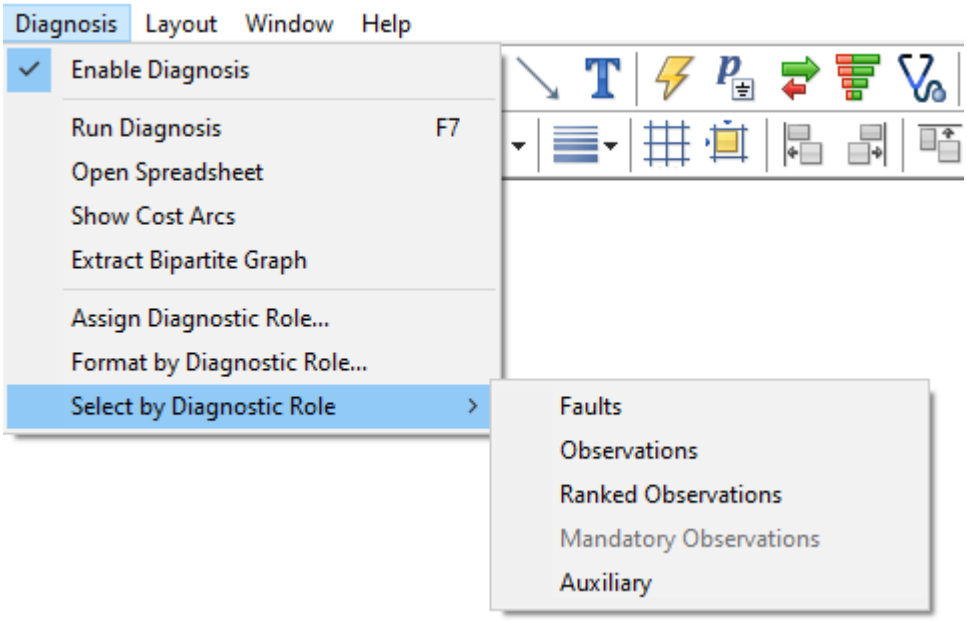


Set as Defaults sets the current selection of colors as the defaults colors for *Format by Diagnostic Role*.

Restore Defaults restores the color settings to the default settings.

Reset to Factory Settings restores the color settings to the factory settings that come with GeNIe.

Select by Diagnostic Role sub-menu allows for selecting all nodes belonging to one of the three types of diagnostic nodes: *Faults*, *Observations*, *Ranked Observations*, *Mandatory Observations*, or *Auxiliary* nodes.



The typical application of this selection is joint editing of each type of nodes, for example coloring or displaying as bar charts.

6.4.3 Defining diagnostic information

Diagnostic features of GeNIe are generally accessed through the *Diagnosis Menu* and they are typically set through the *Diagnosis* Tab in *Node properties* tabs.

Setting properties of model variables for diagnostic applications

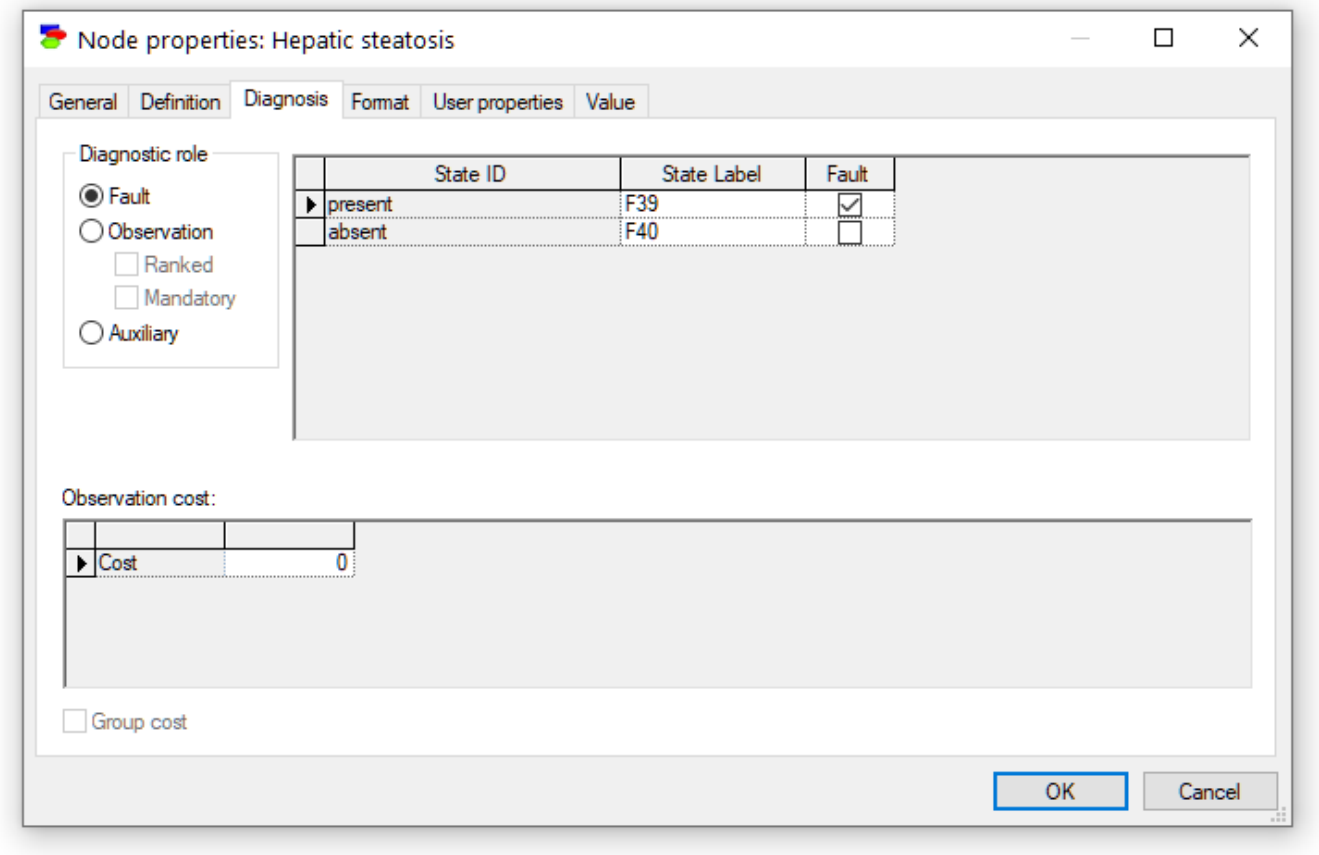
The critical element of diagnostic extensions is setting the roles that various nodes play in diagnosis. Each defect, error message, symptom, and test is represented as a separate node of a graphical causal model that is at the foundation of a Bayesian network model. Since a general purpose Bayesian network model does not make a distinction between the meaning of different types of nodes, the modeler has to indicate which of them represent defects, which are observations, and which are possible tests. For example, nodes representing components can have states labeled *OK* and *Defective*. For nodes representing symptoms or error messages, the states can be *Present* and *Absent*. For test nodes, the states could be labeled *Passed* and *Failed*. Nodes are connected together in the Bayesian network using directed links. The links typically follow the causal direction, i.e., go from the nodes that represent possible faults to nodes representing observations, error messages, tests, and symptoms. A link from a given component to a symptom can indicate that the symptom can be caused by the defects of the component. A link from a given component to a test can indicate that the test can be used to determine whether or not the component is defective.

Nodes in GeNIe's diagnostic extensions are divided into three classes: *Faults* (these are abnormalities such as machine failures, defects, or diseases), *Observations* (these are observable symptoms or risk factors), and *Auxiliary* (these are additional variables, which are neither faults or observables but are useful in specifying the model). All nodes in a diagnostic model must be *Chance* type nodes.

The *Diagnosis* tab in the *Node properties* sheet can be used to enter new properties or alter current properties for a single node.

Diagnosis tab

Shown below is the snapshot of the *Diagnosis* tab of the *Node properties* sheet (please note that the diagnostic extensions need to be enabled for this tab to be visible).



Fault nodes represent machine failures, hardware defects, disorders, etc. At least one state of a *Fault* node must be designed as a *Fault* state (check-box on the right-hand side). A fault state is the defective state, so all states that are not designated to be *Faults* are normal states by default.

Observation nodes represent variables that are observable in the process of diagnosis, such as historical data, error messages, symptoms, or test results.

Auxiliary nodes are all other nodes and are neither faults nor observations. They are typically used for modeling convenience.

In addition to the three node types, there are also two node subtypes of *Observation* nodes: *Ranked* and *Mandatory*.

Observation Ranked nodes are the most common observation nodes. They are ranked by the diagnostic *Diagnosis Window* according to how informative they are with respect to the faults. They represent observations, whose states are unknown in advance but that are useful in diagnosis. The purpose of a *Observation Ranked* node is to find out how an observation ranks relative to other possible observations before it is performed by displaying a list ranked in terms of its effectiveness in troubleshooting. These tests are ranked in the *Ranked-Observation* pane of the diagnostic *Diagnosis Window*.

Observation Mandatory represents information that needs to be provided in advance. It may represent an action, a testing condition, or any other factor that needs to be performed or observed before the troubleshooting begins.

Faults appear always on the list of ranked faults in the *Diagnosis Window*. *Fault* components are ranked in terms of their probability. Each of the *Fault* states will be listed along with its probability.

All of the above can also be edited in the *Spreadsheet window*.

The following table lists the diagnostic types with their typical uses and the *Ranked/Mandatory* options:

Diagnostic Type	Typical use	Ranked/Mandatory status
Fault	Faulty and defective components or medical disorders	*Ranked*
Observation	Error messages, symptoms or tests	None / *Ranked* / *Mandatory*
Auxiliary	No specific use, only for modeling convenience	None

State Label field allows to define an additional state label, which can be referred to in embedded diagnostic applications. GeNIe and SMILE provide only means for storing, retrieving, and editing this field.

Note : *State IDs must be unique within the same node.*

Node property tabs is just one way of entering diagnostic information. For existing models, diagnostic roles can be set by means of the *Assign Diagnostic Role* dialog, which can be invoked through the *Diagnosis Menu*. The *Assign Diagnostic Role* dialog allows for doing it in batches for all nodes in the *Graph View* that have been selected before invoking the dialog. Please look at the description of the *Assign Diagnostic Role* dialog in the *Diagnosis Menu* section. Yet another way of entering diagnostic information is described in the *Spreadsheet View* section.

Cost of observation

The *Observation cost* allows for entering the cost of observing the value of the current node. The cost of performing a test can be expressed on some scale, e.g., currency or time in minutes. The cost entered can be any real number or can be *N/A* (or a shorthand *na* or *NA*) when the cost is not applicable. Costs can also be entered in the *Spreadsheet View*.

It is possible to enter negative values as costs. Negative values indicate costs that are so inexpensive that they should always be performed. For example, it may cost close to nothing to determine the biological sex of a patient and doing so gives some information about the likelihood of various disorders. The same holds, for example, for car make and type or engine version in case of diagnosing a car - they are all readily available. Negative costs can be identified in the *Diagnosis Window* when the option *Enable Quick Tests* is on.

Note : The *Group Cost* check box is enabled only if the current node has more than one child. For more information on how to use *Group Costs* see *Group Costs* section of *Cost of Observation* section.

For more information on how to use cost of observation in your network see *Cost of Observation* section.

6.4.4 Single and multiple fault diagnosis

When all fault states of the entire system are present in one and the same node, they are by definition mutually exclusive (please note that states of a random variable are mutually exclusive) and we are dealing with a commonly used assumption of single fault diagnosis, i.e., that there is only one fault possible at any given time.

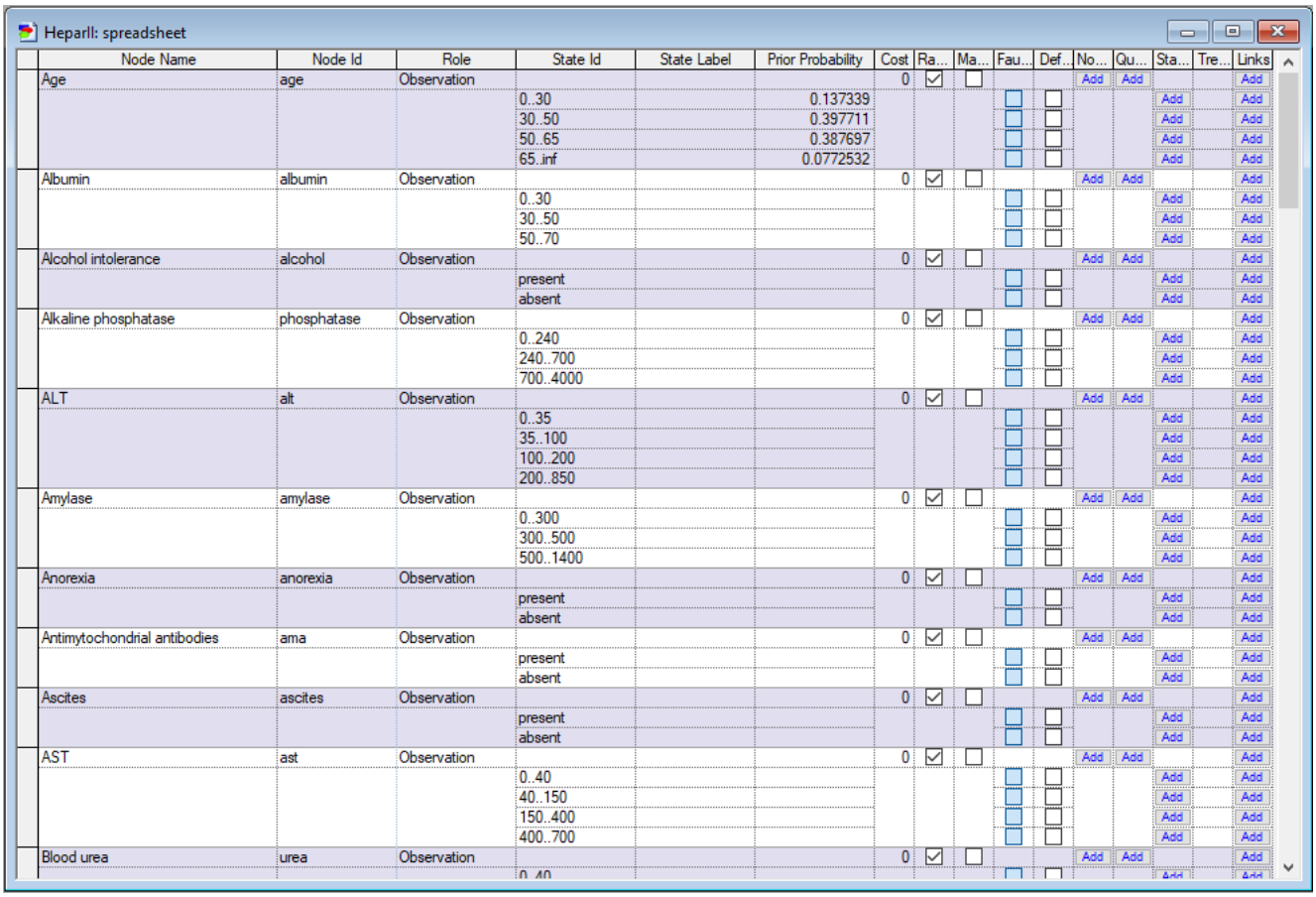
Diagnosis transforms naturally into multiple fault diagnosis when faults are present in more than one node. In that case, faults are no longer mutually exclusive and presence of multiple faults is theoretically allowed. No longer will the probabilities of various faults add up to 1.0.

6.4.5 Spreadsheet view

The *Spreadsheet View* is a special extension of GeNIe that is useful in rapid building of diagnostic models - all properties of every variable of a model are listed in one window and the user specifying a model can move rapidly between variables and enter or modify their specifications.

To open the *Spreadsheet View*, the user must select *Open Spreadsheet* command from the *Diagnosis Menu*.

The *Spreadsheet View* consists of several columns of information that describe individual nodes (represented by rows) in detail. The information in the *Spreadsheet View* is also available in the *Node Property* sheets, although the *Node Properties Sheet* contains additional information that is not available in the *Spreadsheet View*, e.g., conditional probability tables.



HeparII: spreadsheet

Node Name	Node Id	Role	State Id	State Label	Prior Probability	Cost	Ra...	Ma...	Fau...	Def...	No...	Qu...	Sta...	Tre...	Links
Age	age	Observation				0	☑	☐			Add	Add			Add
			0..30		0.137339								Add		Add
			30..50		0.397711								Add		Add
			50..65		0.387697								Add		Add
			65..inf		0.0772532								Add		Add
Albumin	albumin	Observation				0	☑	☐			Add	Add			Add
			0..30										Add		Add
			30..50										Add		Add
			50..70										Add		Add
Alcohol intolerance	alcohol	Observation				0	☑	☐			Add	Add			Add
			present										Add		Add
			absent										Add		Add
Alkaline phosphatase	phosphatase	Observation				0	☑	☐			Add	Add			Add
			0..240										Add		Add
			240..700										Add		Add
			700..4000										Add		Add
ALT	alt	Observation				0	☑	☐			Add	Add			Add
			0..35										Add		Add
			35..100										Add		Add
			100..200										Add		Add
			200..850										Add		Add
Amylase	amylase	Observation				0	☑	☐			Add	Add			Add
			0..300										Add		Add
			300..500										Add		Add
			500..1400										Add		Add
Anorexia	anorexia	Observation				0	☑	☐			Add	Add			Add
			present										Add		Add
			absent										Add		Add
Antimytochondrial antibodies	ama	Observation				0	☑	☐			Add	Add			Add
			present										Add		Add
			absent										Add		Add
Ascites	ascites	Observation				0	☑	☐			Add	Add			Add
			present										Add		Add
			absent										Add		Add
AST	ast	Observation				0	☑	☐			Add	Add			Add
			0..40										Add		Add
			40..150										Add		Add
			150..400										Add		Add
			400..700										Add		Add
Blood urea	urea	Observation				0	☑	☐			Add	Add			Add
			0..40										Add		Add

The columns of the Spreadsheet are described below in the order from left to right.

Node Name

Node Name is simply the name of the node that is being described. A node name can be changed manually by clicking on the box and typing the different name. The column can be sorted alphabetically by clicking on the *Node Name* label. In the figure above, the nodes are listed alphabetically starting with *ALT* and then *AST*.

Node ID

The *Node ID* column specifies a unique (among all nodes in the network) identifier for each node. Clicking on *Node ID* will change the order of the nodes from ascending to descending alphabetical order. *Node IDs* consist of a string of alphanumerical characters with no spaces (underscore characters are allowed).

Role

Role describes the diagnostic role of the node. By clicking on *Role*, the rows can be sorted according to their diagnostic roles. The diagnostic role can be changed manually by clicking on the box and by typing in a different name.

You can also change a node's diagnostic role by selecting the node (or a group of nodes) and selecting the *Assign Diagnostic Role...* from the *Diagnosis Menu*. This opens the *Assign Diagnostic Role* dialog described in the *Diagnosis Menu* section.

For more information on the functions of the various types of nodes, see *Defining diagnostic information* section.

State ID

The *State ID* column lists the node's states. For example, node *Alcohol intolerance* has two states: *present* and *absent*. *ALT*, which is a numerical node specified by means of intervals, has four states (intervals): *0..35*, *35..100*, *100..200* and *200..850*. Each of node states is described further in the spreadsheet. A state ID, which consist of a string of alphanumerical characters with no spaces (underscore characters are allowed), can be changed manually by clicking on a corresponding cell and typing in a different ID. Please remember that state IDs have to be unique within a node.

State Label

The *State Label* is a unique identifier for each state of a node. The *State Label* can be changed manually by clicking on a corresponding cell and typing in a different label. *State label* consist of a string of alphanumerical characters with no spaces (underscore characters are allowed) and should be unique in the entire model.

Prior Probability

Prior Probability column displays the prior probability distribution over the states of root nodes (i.e., nodes without parents). Many nodes in diagnostic models fall into this category and including this column in the spreadsheet is a compromise between having all distributions and avoiding the complexity of multi-dimensional conditional probability tables. Prior probabilities for components indicate how likely the components are to fail (it can be based on expert opinion or on objective data, such as average frequency of failure across the fleet). For example: a given component, *A*, may have a prior probability of being in the state *Defective* equal to *0.001* and of being in the state *OK* equal to *0.999*. This prior probability means that only in one locomotive in a 1000 will the component *A* be defective per shop visit. To change the prior probabilities of a faulty component, locate the row of the node for which the probability is to be changed. Then locate the rows representing the states of the node. After finding the appropriate row, scroll right until you reach the column in the *Spreadsheet View* labeled *Prior Probabilities*. Click on the box representing the probability of the state and type in the new probability. Enter the probabilities for the other states as well. The sum of prior probabilities over all states of a given node has to be equal to 1.0. In the figure above, the states of node *Age* are *age65_100*, *age51_65*, *age31_50*, and *age0_30* and have prior probabilities of 0.0772532, 0.387697, 0.397711, and 0.137339, respectively. If the prior probability of a node representing a faulty component is changed to zero then this fault is effectively eliminated from the process of diagnosis. In some cases, the faulty component nodes are dependent upon other nodes, such as nodes representing versions of the hardware, e.g., engine type. To change the probability of failure of such nodes, one needs to enter new values into the conditional probability table in the *Definition Tab* of *Node Property* sheets.

See *Definition* tab section of *Node Property* sheets for more information on how to enter data into the conditional probability tables (CPTs).

Cost

The *Cost* column is defined for *Observation* nodes only. Cost contains a simple cost of a tests or an observation represented by the node. The *Observation* cost can also be specified from the *Node Property* sheets for the node. See the *Cost of observation* section for more information.

Ranked & Mandatory

The *Ranked* and *Mandatory* columns are used to represent subtypes of each node role. Checking and unchecking the boxes in corresponding columns specifies their subtype. More than one subtype may be selected for a given node role, i.e., a node can be both *Ranked* and *Mandatory*. Legal combinations of the two are described in the *Defining Diagnostic Information* section. If a box has been grayed out, then it does not pertain to the particular node role and cannot be changed.

The *Ranked* and *Mandatory* status can also be specified in the *Diagnosis Tab* in the *Node Property* sheets.

Fault State & Default State

The *Fault State* check box specifies whether a state is a disorder/malfunction or not. The *Default State* points out a state that is the default observation, i.e., other information lacking, the node is set to that state. If the region is gray, then the column in question does not pertain to the node.

The *Fault* and *Default* status can also be specified from the *General Tab* in the *Node Properties Sheet*. See the *General Tab* section of *Node Property* sheets for more information on how to do this.

Node Description

The *Node Description* column contains a short text defining the node and its states. To edit the *Node Description* column, click the button labeled *Edit* or *Add*. *Add* indicates that there is no description available for the node. *Edit* indicates that a description has been already entered for the node and it can be edited. Clicking on *Edit* or *Add* buttons results in a new window, in which the user can add the description.

The *Node Description* can also be specified from the *Documentation tab* in the *Node Property* sheets. See the *Documentation Tab* section of *Node Property* sheets for more information on how to do this.

Question

The *Question* column applies to *Observation* nodes only and describes in the form of a question what the observation is supposed to answer. The contents of the *Question* column is edited similarly to the *Node Description* column. The *Question* can also be specified in the *Documentation tab* in the *Node Property* sheets. See the *Documentation Tab* section of *Node Property* sheets for more information on how to do this.

State Description

The *State Description* column contains a short text describing the state. The contents of the *State Description* column is edited similarly to the *Node Description* column. The *State Description* can also be specified from the *Documentation Tab* in the *Node Property* sheets. See the *Documentation Tab* section of *Node Property* sheets for more information on how to do this.

Treatment

The *Treatment* column describes how to treat the defect represented by the state and applies to *Fault* states only. It will be grayed out for all other states. The contents of the *Treatment* column is edited similarly to the *Node Description* column. The *Treatment* can also be specified from the *Documentation Tab* in the *Node Property* sheets.

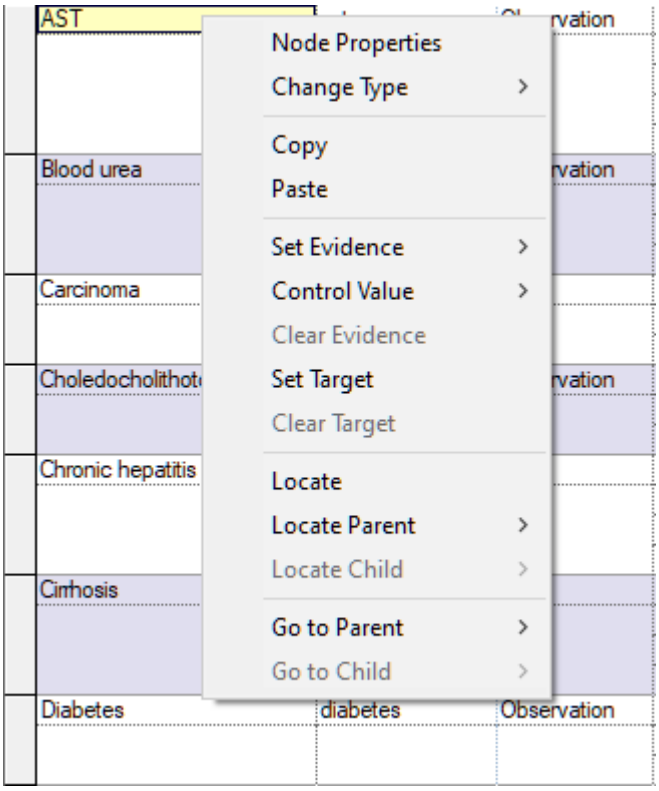
Links

The *Links* column, similarly to the preceding columns, contains a short text consisting of hyperlinks. These links in most cases are pointing to documents with additional information, such as further documentation for the nodes and states. These documents can be used to repair procedures, schematics, block diagrams, manuals, etc. Links have names describing what the documents contain and the address of the document. The contents of the *Links* column is edited similarly to the *Node Description* column.

The *Links* can also be specified from the *Documentation Tab* in the *Node Property* sheets.

Node properties menu in spreadsheet view

Right-clicking on a node in the spreadsheet view brings up the *Node Pop-up* menu



Most of the choices are the same as in the *Node Pop-up* menu in the *Graph View*.

Copy and *Paste* copy currently selected text onto the clipboard and pastes current contents of the clipboard, respectively. The pair is useful for copying text between cells.

Locate is used to locate the selected node in the *Graph View*. Once located, the node is flashed several times on the screen to make it visible to the user.

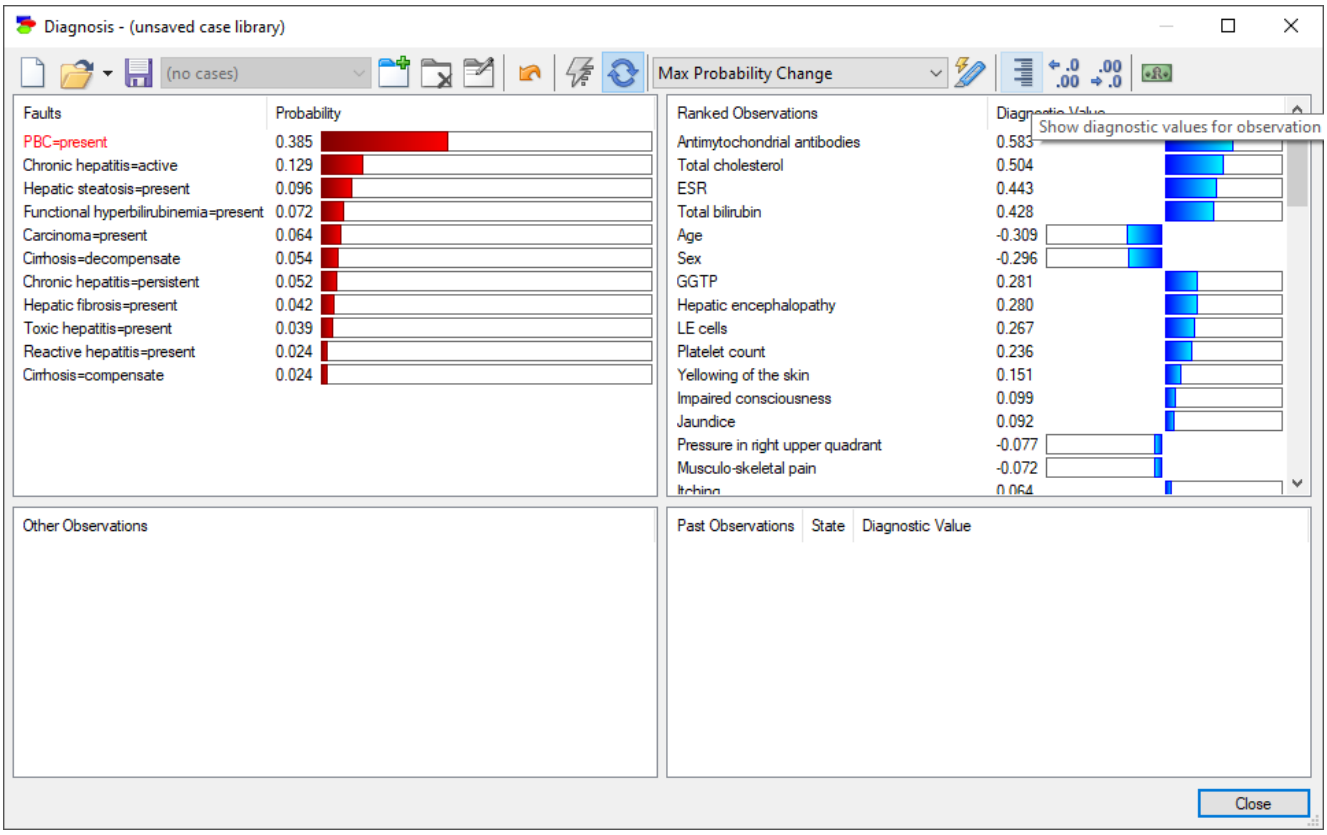
Go to Parent and *Go to Child* allow for moving around in the *Spreadsheet View*. They locate the selected parent or child and move the cursor to the corresponding line in the spreadsheet.

6.4.6 Diagnosis window

The *Diagnosis Window* is a special GeNIe module that supports diagnostic applications.

Composition of the Diagnosis Window

One of the example models with diagnostic extensions is the Hepar II model. Once the model has been loaded and diagnostic extensions enabled, please click on the *Run diagnosis* ( ) button on the toolbar, or press F7. Once the *Diagnosis window* has opened, it will appear as shown below:



The *Diagnosis Window* has four panes, which can be described as follows:

- The top-left pane is for the *Faults*, which are non-observable and ranked nodes, typically representing various faulty components. The fault in red is the focus of the current diagnostic session. When the *Diagnosis Window* is just started, the most likely fault is the one selected to be the focus.

- The top-right pane is for the *Ranked Observations*, which are observable nodes that have not yet been observed, ranked from the most to least informative (from the highest to the lowest *Diagnostic Value*).

- The bottom-left pane lists *Other Observations*, which are those observations that have been designated as *Mandatory* and are not yet observed. Their designation as *Mandatory* indicates that they are straightforward to observe or are otherwise part of the first steps and need to be observed first. They are displayed in red font in order to draw user's attention.

- The bottom-right pane contains all those nodes from among both the *Ranked Observations* and the *Other Observations*, that have been observed along with their diagnostic value, i.e., their contribution to the.

We will discuss various buttons placed on the top of the diagnostic window later in this section, in the context of the window's fundamental functionalities. Here we would like to point out three general purpose buttons:

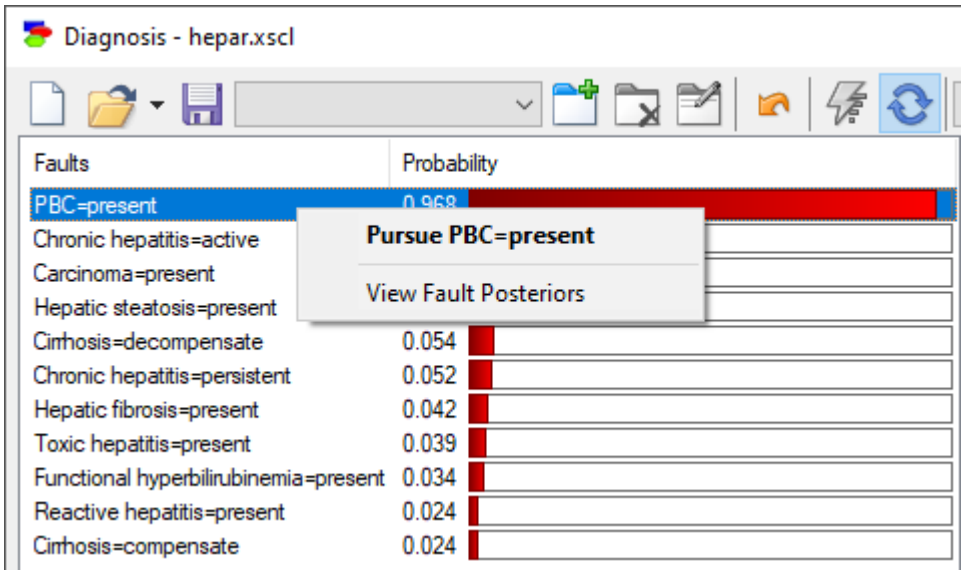
Restart diagnosis button ( ) clears all observations made so far and starts with the clear observation slate.

Update button ( ) and *Toggle immediate update* button ( ) control updating the window panes after making observations. We advise that the immediate update is on by default, unless update takes longer than desired. Update immediately, when set, makes GeNIe recalculate the ranking as soon as any change is made. If immediate update is off, you will need to press the *Update* button to recalculate the ranking. Please note that the calculations of diagnostic information value are computationally very complex and while SMILE is very fast and we have not experienced long wait times, it may happen that this calculation takes too long.

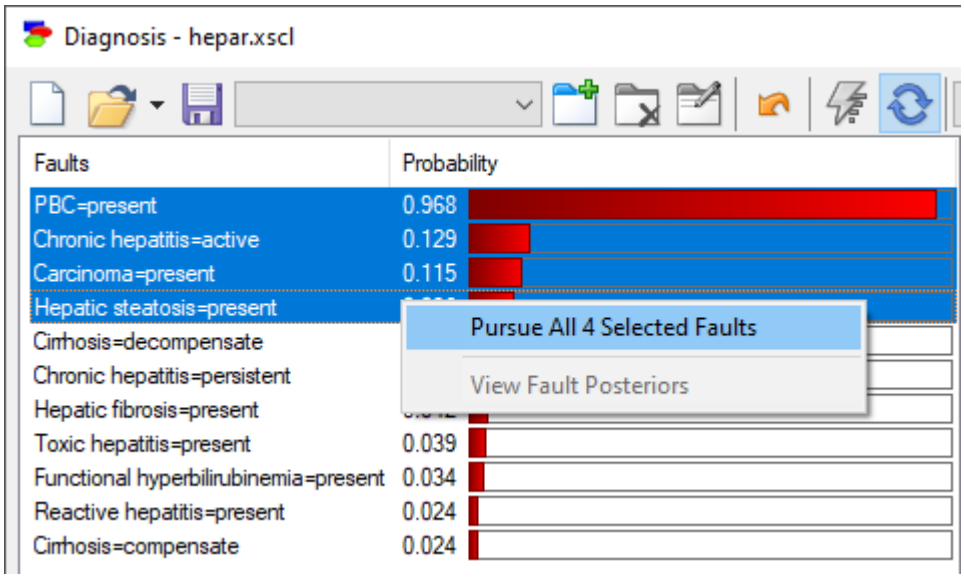
Enable relevance button ( ) becomes useful when networks are very large and it switches on a heuristic that ranks only those tests and faults that are relevant (in terms of being connected to) the observed set of evidence nodes. When this flag is off, some of the faults may be present and ranked in the diagnostic window purely because of their high prior probability.

Focus of diagnostic inference: Pursuit of a fault

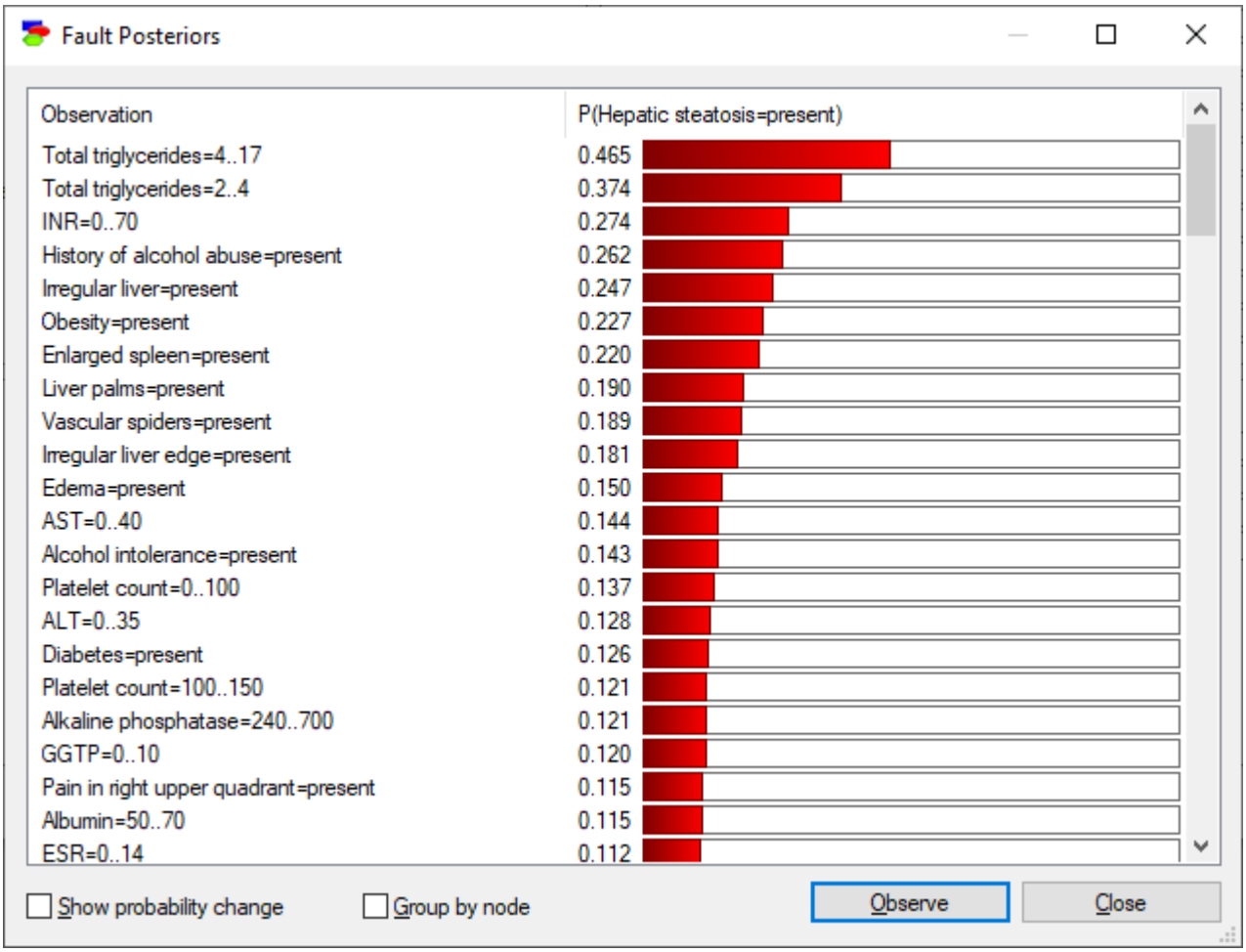
The first and foremost setting used in the calculation of cross-entropy is setting the focus of reasoning. The focus of reasoning is the fault displayed in red. When the *Diagnosis Window* is just started, the most likely fault is the one selected to be the focus. You can change the focus by selecting a fault state (or a set of fault states), right-clicking on the selection, and choosing *Pursue* (in the illustration below, *Pursue PBC=present*).



The selected fault will become the focus and will be displayed in red. The top-right pane will show a ranked list of observations that are relevant to the focus. We can select a group of fault states and pursue them all, in which case, the top-right pane will show a ranked list of observations that help us in differentiating among the selected states.

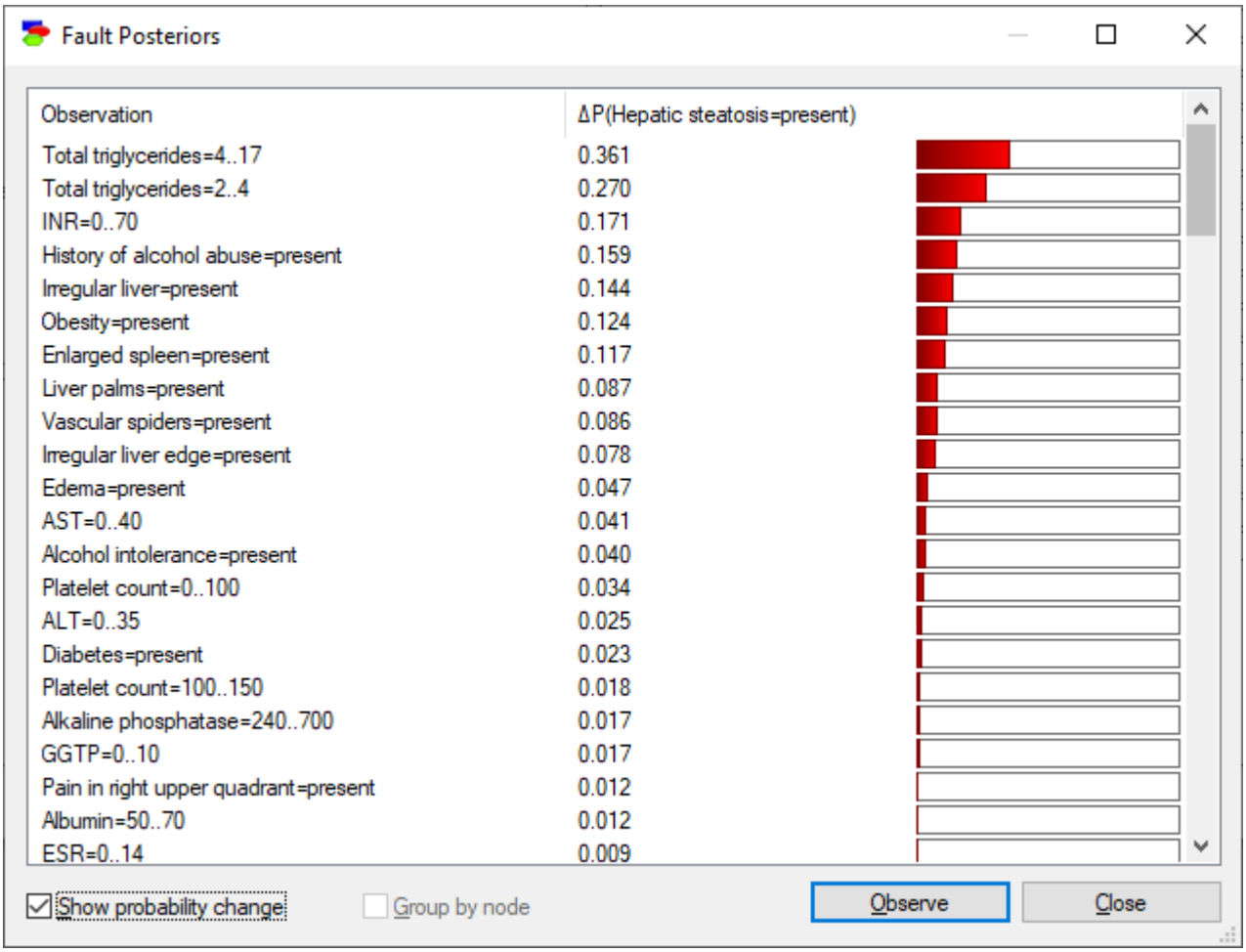


For the fault that is currently being pursued, it is possible to display its probability conditional on various observations. To view that list, please select *View Fault Posteriors* from the context menu. This will work also when pursuing multiple faults but the dialog always shows the probabilities for the fault that is selected by right-clicking on it.

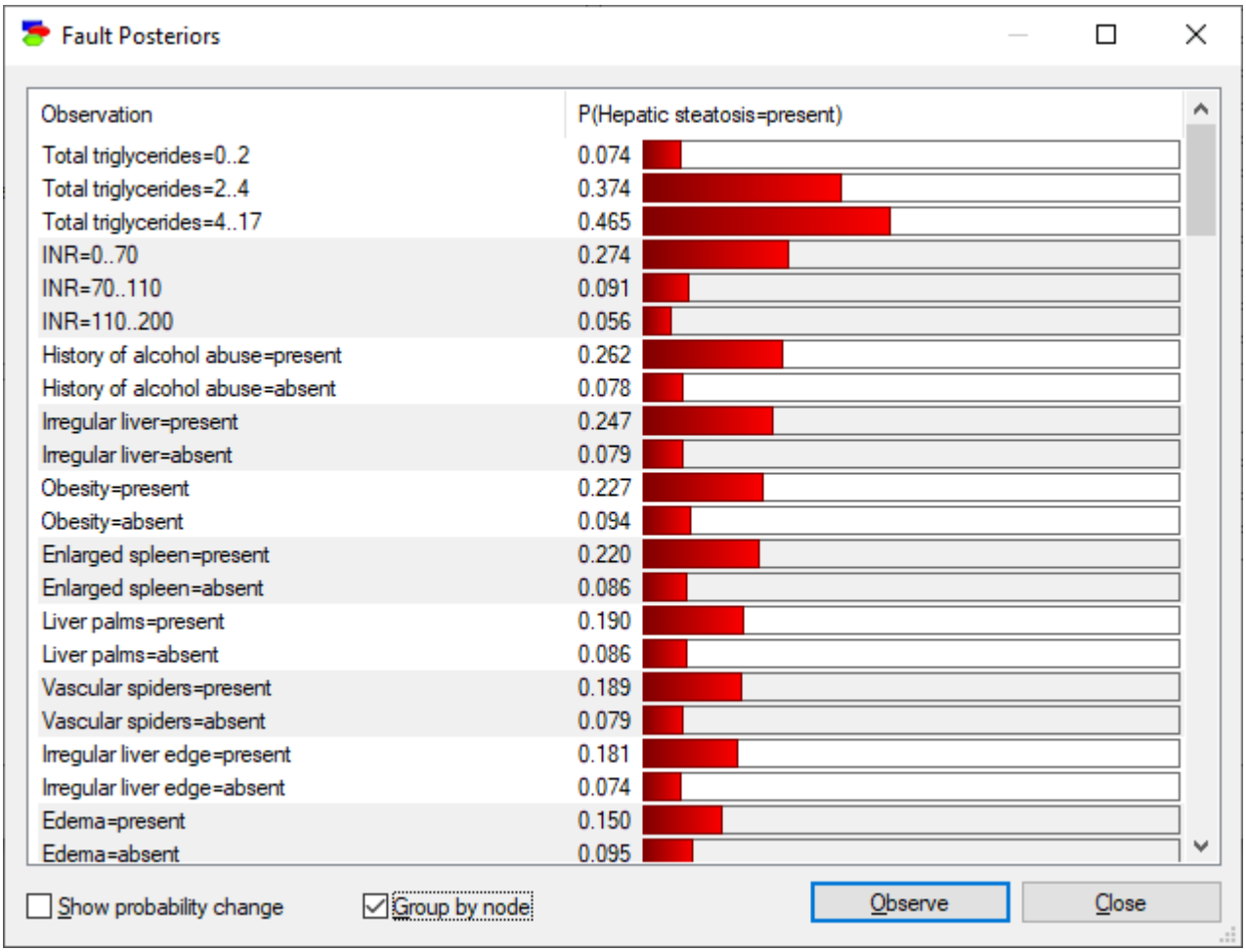


The ensuing dialog shows, in addition to the posterior probabilities of the pursued fault, two options: *Show probability change* and *Group by node*.

Show probability change displays the (absolute) change in probability of the selected pursued fault as a function of the possible observations:



Group by node sorts the list in such a way that all outcomes of a single node are placed next to each other:

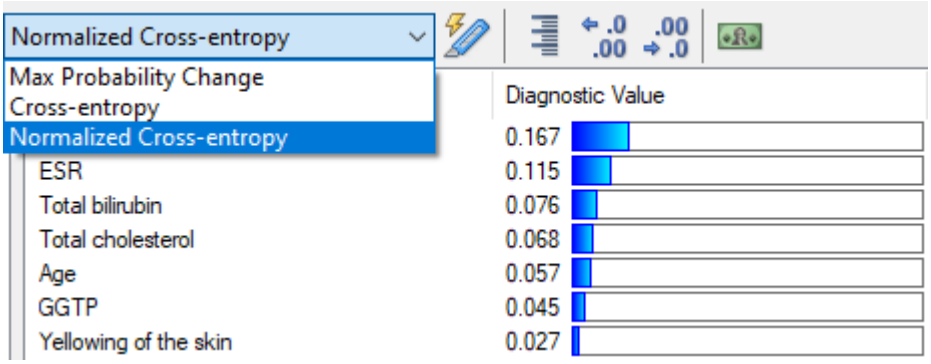


The list of observations is still sorted by the largest posterior probability of the pursued node.

Diagnostic Value of Information

The top-right pane shows a list of *Observation* nodes that have not yet been observed, ranked from the most to the least informative. The ranking is based on one of the selected measures of information, which we will discuss below. The measure used is displayed in the central part of the dialog header (the default, although as we will discuss later, not necessarily the measure that gives most insight, is *Max Probability Change*). The default measure (algorithm for diagnostic value of information computation) can be selected in *Diagnosis* tab of the *Options* dialog, available from *Tools* menu.

We will change the algorithm to *Normalized Cross-entropy*, the default measure in previous versions of GeNIe, by clicking on the *Diagnostic value algorithm* pop-up menu:



Cross-entropy is an information-theoretic measure that expresses, for each *Observation* node *X* individually, the expected reduction in entropy of the probability distribution over the fault nodes in red font in the top-left pane after observing *X*. Cross-entropy is a utility-free measure of value of information and it gives a good idea about the value of the observations for diagnosing the disorder in question.

Ranking the possible observations can be used as a guide for diagnostician what to do next. Each measure of the diagnostic value, including cross-entropy, is dynamic and it depends strongly on the collection of fault nodes and also on other observations. Calculation of each measure is computationally complex and involves a series of runs of a belief updating algorithm. It is useful to view the calculation of diagnostic value in GeNIe as a flexible alternative to pre-cooked list of questions. We all know how irritating it is to a customer to hear a phone support employee follow a standard "once-size-fits-all" set of questions that are typically irrelevant to the problem at hand. The ranked list calculated by GeNIe is always up-to-date and tells us at each step what to do next and what question makes the most sense.

Here is an example set of ranked observation nodes:

Ranked Observations	Diagnostic Value
Sex	0.167
ESR	0.115
Total bilirubin	0.076
Total cholesterol	0.068
Age	0.057
GGTP	0.045
Yellowing of the skin	0.027
LE cells	0.009
Platelet count	0.008
Pressure in right upper quadrant	0.007
Itching	0.006
Jaundice	0.006
Hepatic encephalopathy	0.004
Musculo-skeletal pain	0.003
Joints swelling	< 0.001
Impaired consciousness	< 0.001
Hepatomegaly	< 0.001
Blood urea	< 0.001

For each observation node, the window displays its diagnostic value numerically. Precision of this number is controlled by the buttons *Increase decimal places* (.0 .00) and *Decrease decimal places* (.00 .0) buttons.

While the graphical display is less precise, is is very convenient for quick judgment of relative value.

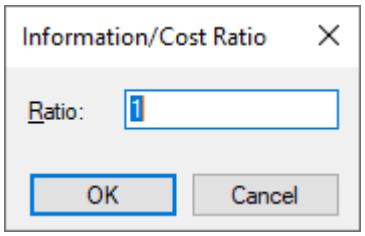
When a focus has never been selected explicitly, GeNIe will keep selecting the most likely fault to be the focus. When the user selects a focus explicitly, that focus stays throughout the diagnostic session.

Information/Cost Ratio

While cross-entropy is a unit-free measure, in some diagnostic applications diagnosticians may be more interested in monetary costs of the performed tests and benefits of observations. As you remember, one of the items specified for any observation node is the cost of observation. GeNIe offers a simple formula for combining cross-entropy with costs, notably a weighted additive scheme. Total benefits are V=*cross-entropy*-wC/wE * *costs*. The ratio α=wE/wC is called *entropy/cost ratio* and the formula for diagnostic value that includes α is as follows:

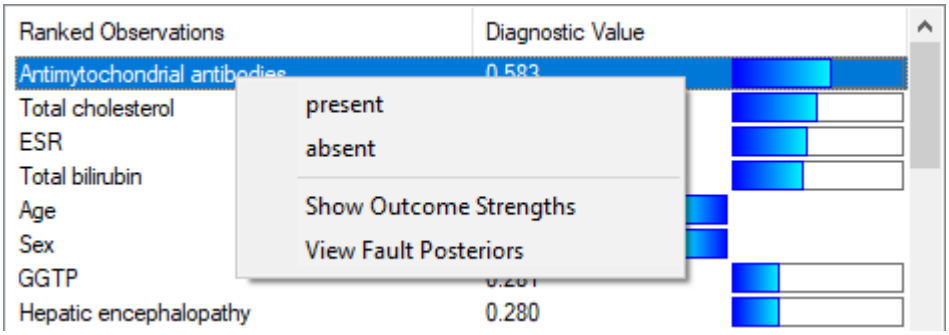
$$V = \textit{cross-entropy} - \alpha * \textit{costs}$$

This ratio allows for a simple way of taking costs into account when looking at the value of observation. The default *entropy/cost ratio* is 1 but it can be changed easily through a simple dialog invoked by the *Modify information/cost ratio* button ( ) at the top-right of the *Diagnosis Window*. Pressing on this button invokes a simple dialog:

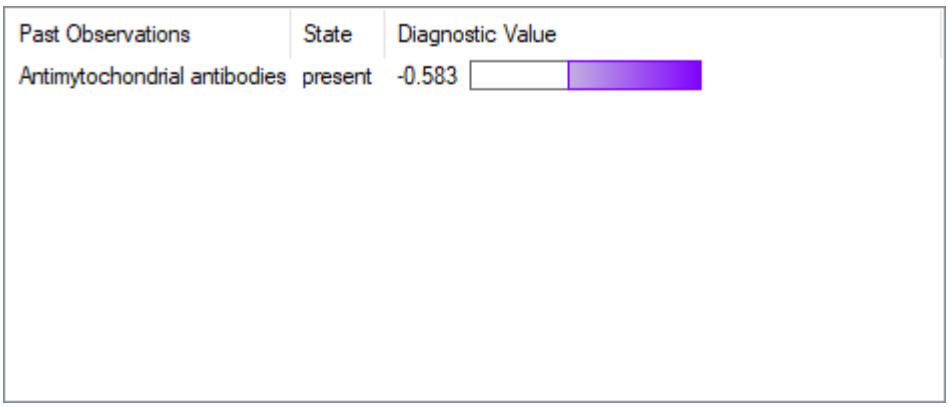


The *Information/cost ratio* is a real number ranging between 0 and 99999. The contents of the box in the *Information/Cost Ratio* dialog represents the value α and can be changed from 0 to 99999. If the *Entropy/cost ratio* is zero, then the ranking of *Observations* is based solely on cross-entropy and disregards to the costs of testing. Any higher number increases the role of cost in the ranking. As we change the *Entropy/cost* ratio, the ranking of *Observations* may change, reflecting the role of costs in the ranking.

To record the value of an observation node (both the *Ranked Observations* and the *Other Observations*), right-click on the node name and select the observed state on the pop-up menu that show up.

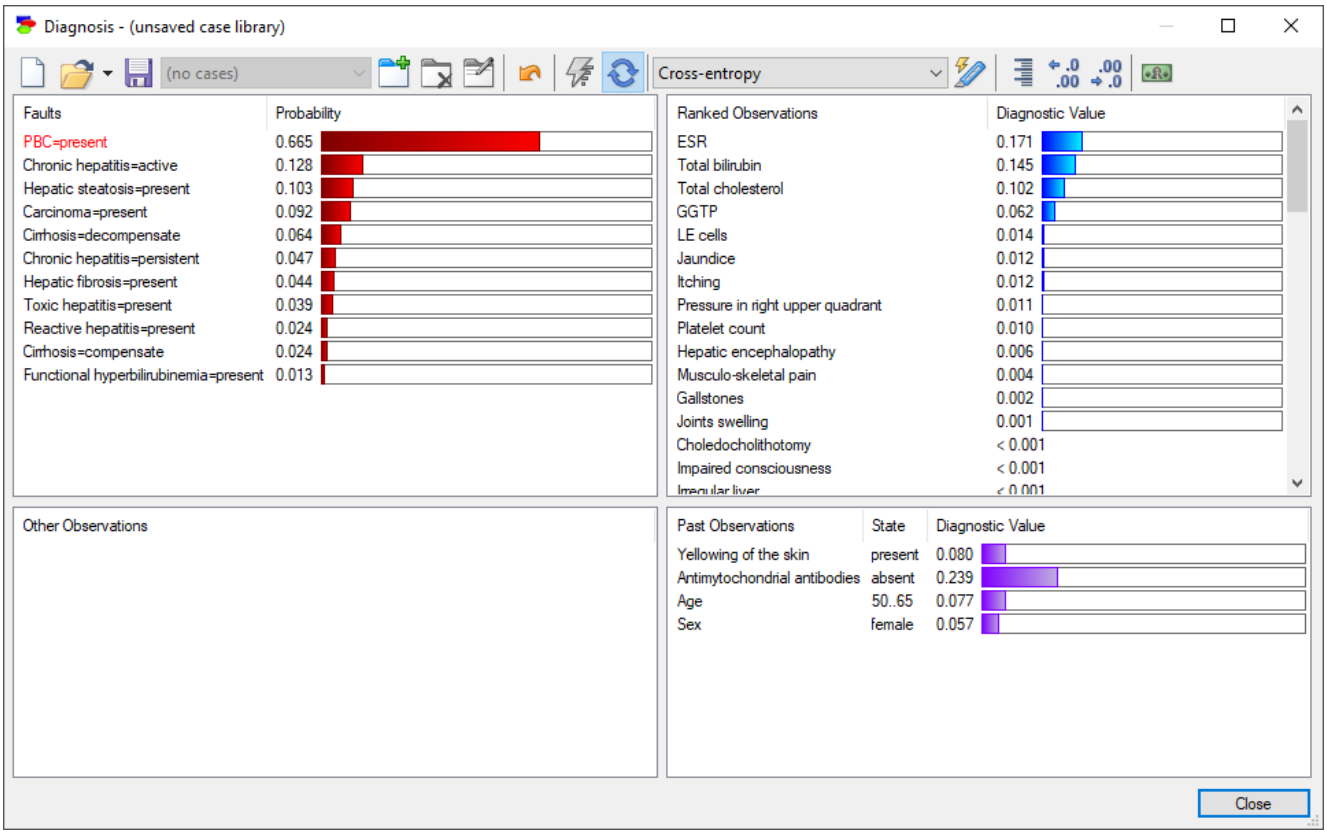


Please note that the node will then show up in the bottom-right pane of the dialog window.

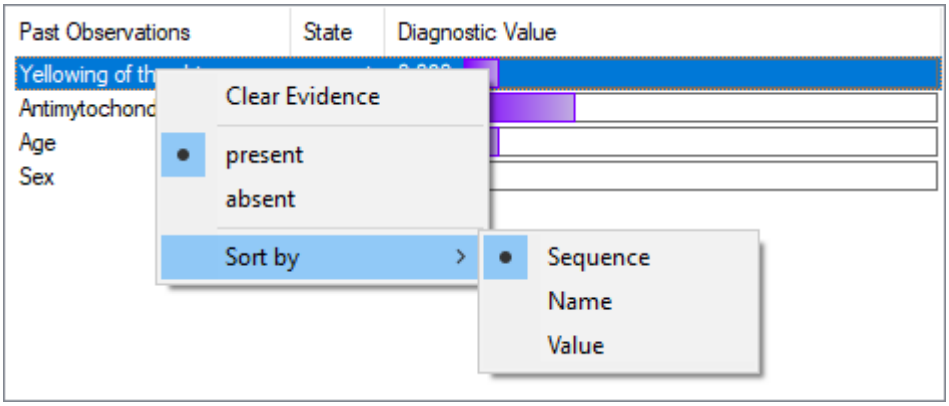


Past Observations Pane

The bottom-right pane contains all those nodes from among both the Ranked Observations and the Other Observations, that have been observed along with their diagnostic value, i.e., their contribution to the probability of the pursued fault(s).



The diagnostic value of the past observations is a dynamic measure and the contribution of each observation changes in the light of other observations, whether new or past. The list of past observations can be sorted by their name, diagnostic value, or order of observations.

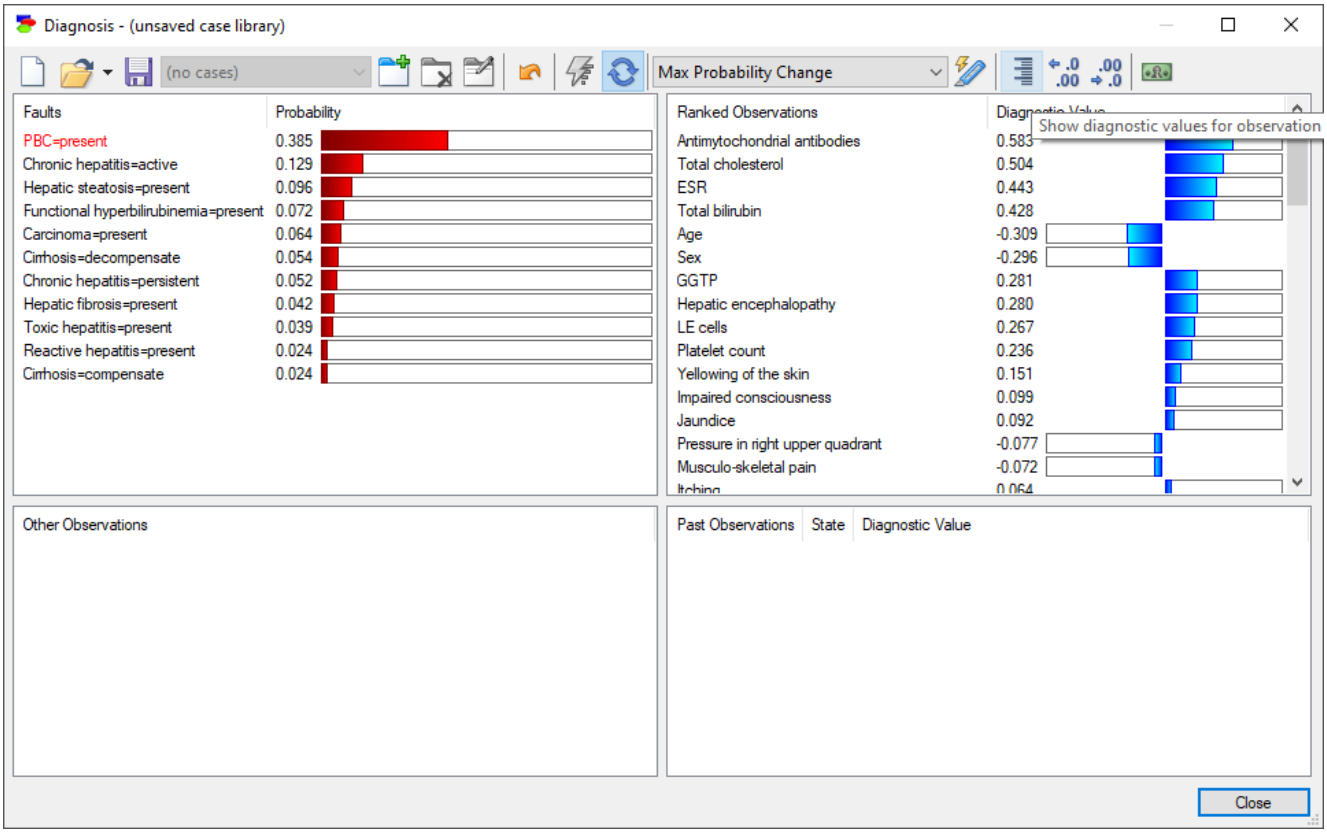


6.4.7 Measures of diagnostic Value of Information (VOI)

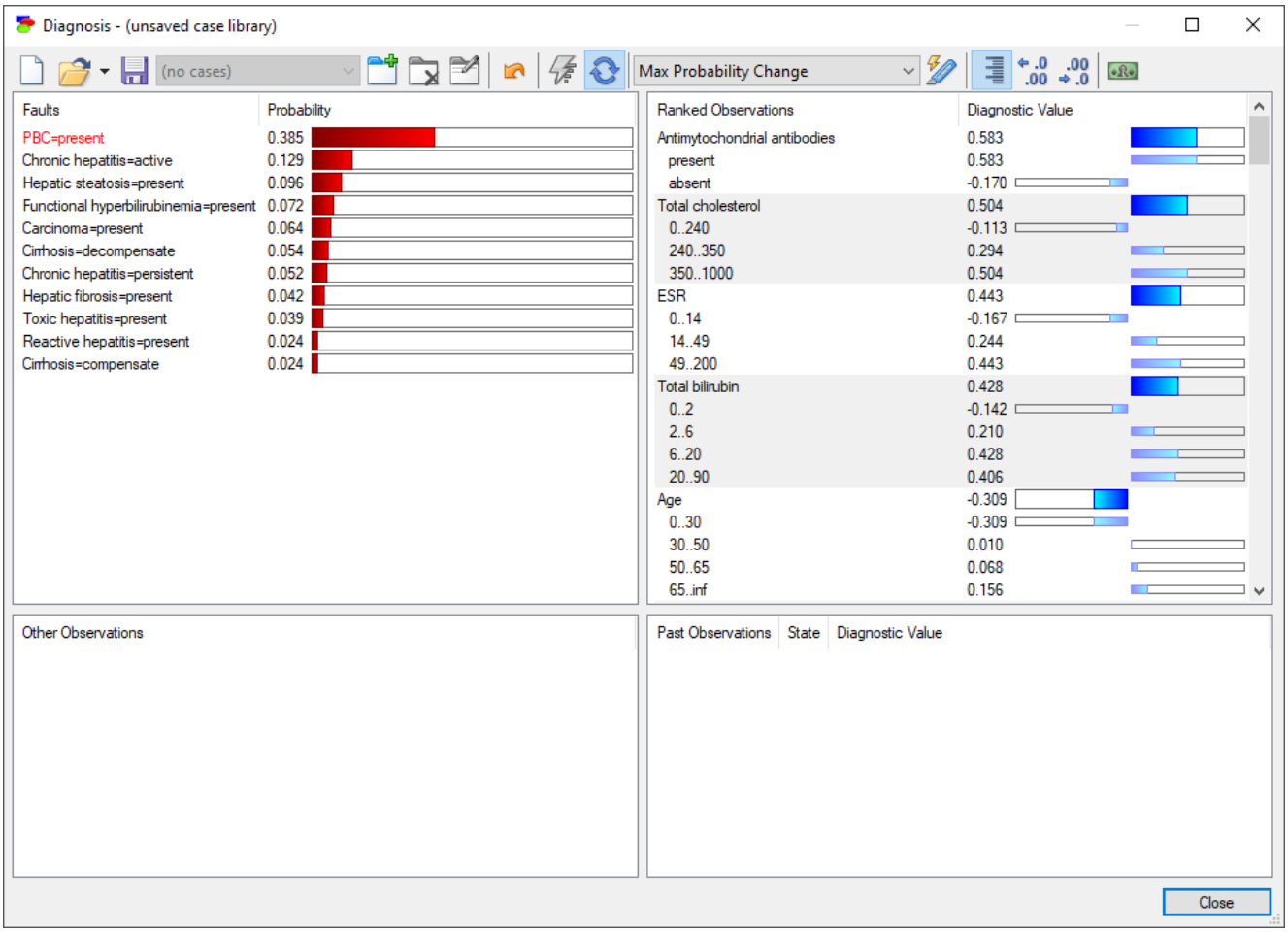
GeNIe and SMILE include three basic measures of diagnostic value of information: (1) Max Probability Change, (2) Cross-entropy, and (3) Normalized Cross-entropy. Each of them produces some insight into the value of observing any of the observation nodes but it is hard to argue for any of them as being the best. We discuss the three measures below in the context of pursuing a single fault. A discussion of how these measures are extended to multiple faults will follow.

Max Probability Change

This measure lists for every state of an observation node the change in the probability of the fault that is being pursued. In the default view, we see for each observation node the maximum probability change.



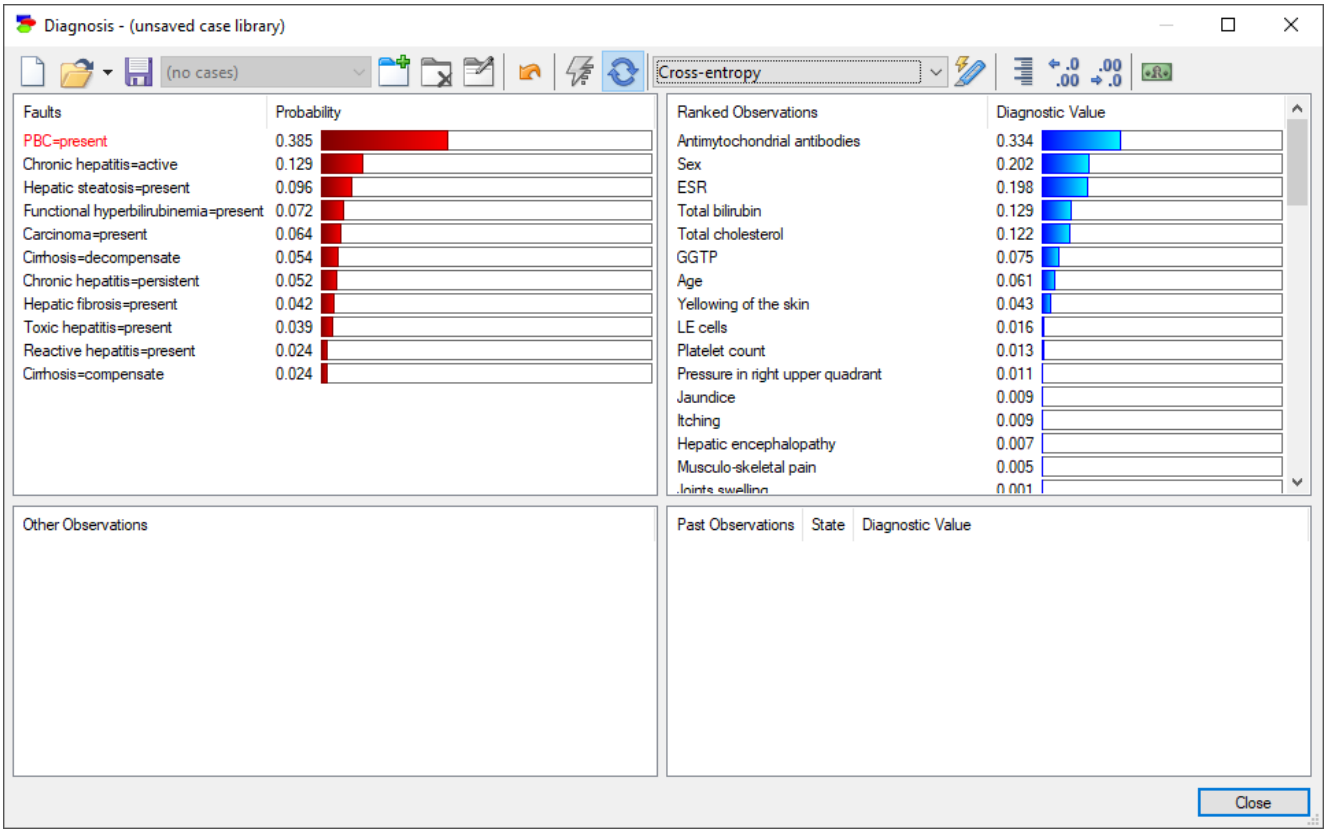
And so, there exists a state of the observation node *Antimychondrial antibodies* for which the probability of the state *present* of the node *PBC* (currently pursued fault) will increase by 0.583. Similarly, there exists a state of the node *Age* for which the probability of the currently pursued fault will decrease by 0.309. The maximum change is taken over the absolute values of the change in probability. Pressing the *Show diagnostic values for observation outcomes* button ( ) displays the probability change for each of the individual states of the observation nodes.



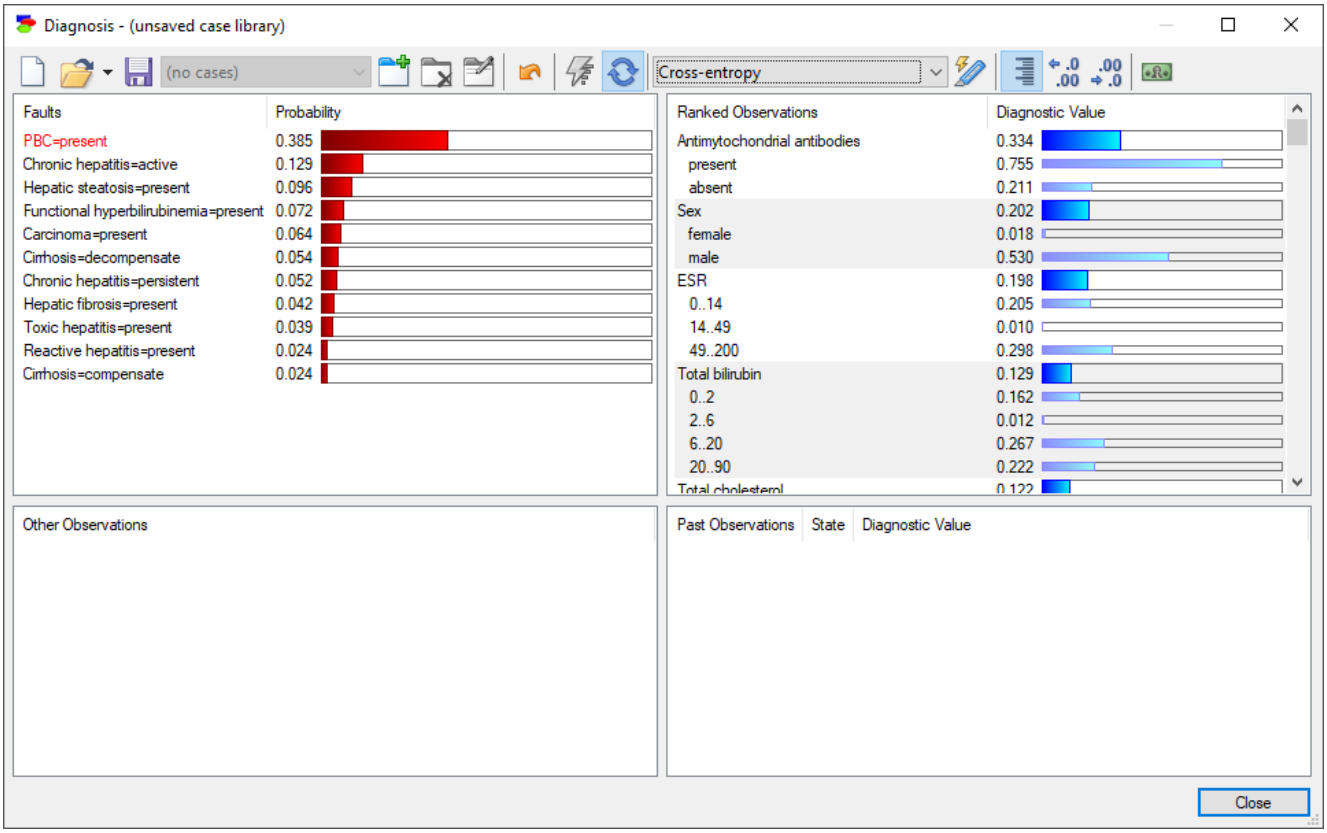
The expanded list allows to see which state exactly is capable of making the largest change in the probability of the pursued fault. And so, we can now see that the state *present* of the node *Antimychondrial antibodies* can increase the probability of the currently pursued fault by 0.583, while state absent can decrease it by 0.17. Similarly, state 0..30 of node *Age* will decrease the probability of the pursued fault by 0.309, while other states will generally increase the probability.

Cross-entropy

The insight from the *Max Probability Change* is limited in the sense of not telling us a key piece of information how likely each of the changes happens. For example, a positive test result for cancer will make a huge change it the probability of cancer. However, the probability of seeing a positive test may be very small in a generally healthy person. So, effectively the expected amount of diagnostic information from performing this test is rather small. Cross-entropy is an information-theoretic measure that takes into account both the amount of information flowing from observing individual states of an observation variable and the probabilities of observing this states. A high cross-entropy indicates a high expected contribution of observing a variable to the probability of the pursued fault. Here is the diagnostic window with cross-entropy of the individual observations:



Pressing the *Show diagnostic values for observation outcomes* button ( ) displays the entropy change for each of the individual states of the observation nodes.



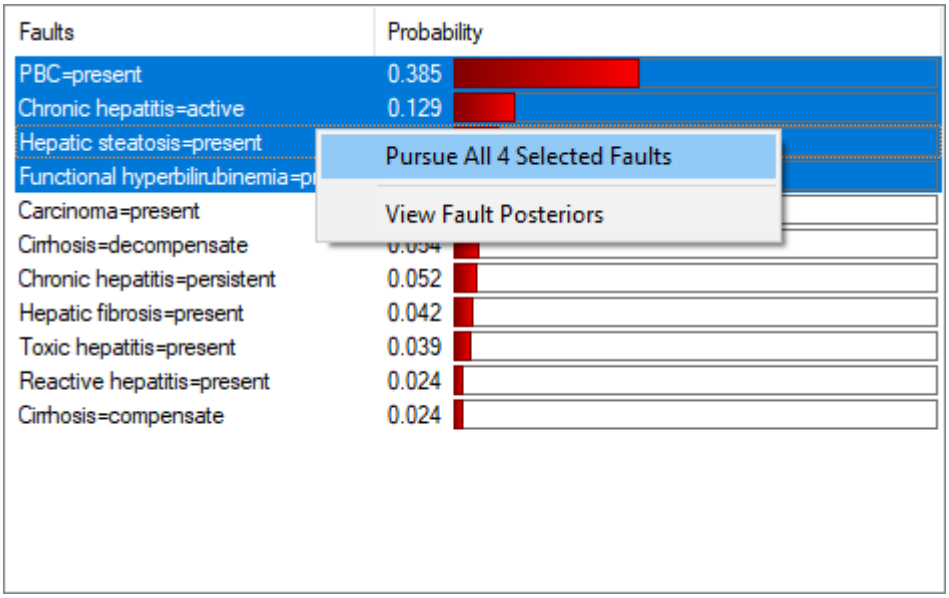
We can see that cross-entropy, the expected change in entropy, is always in-between the changes in entropy due the individual states.

Normalized Cross-entropy

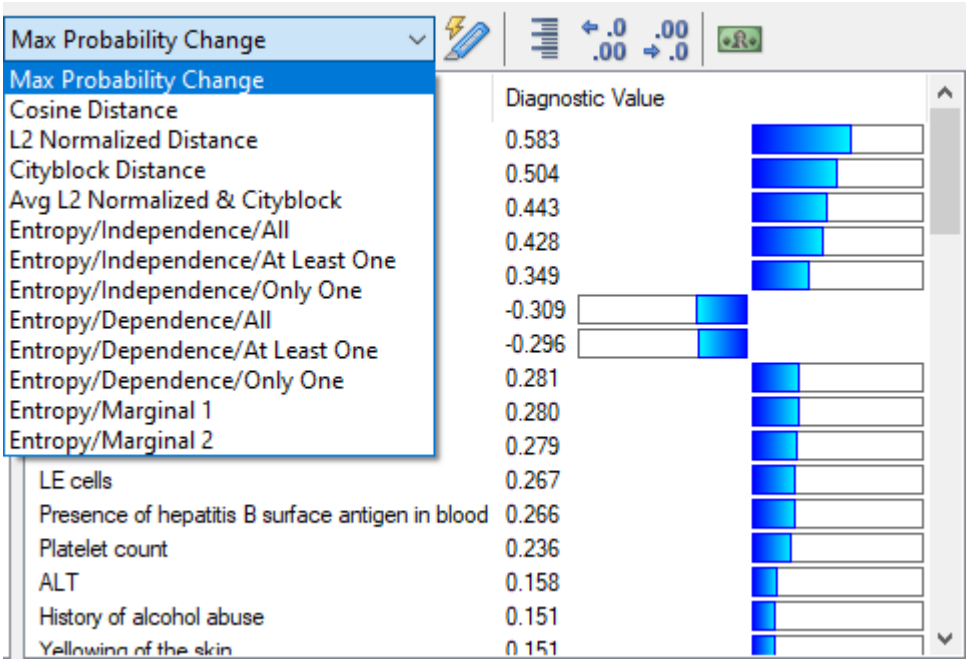
Normalized cross-entropy is essentially entropy divided by the current value of the entropy of the focus node. It is very similar to cross-entropy.

Measures of Diagnostic Value of Information for Multiple Pursued Faults

The meaning of each of the three measures is quite clear in case of pursuing a single fault or pursuing multiple faults that are various states of a single node. Additionally, the calculation of cross-entropy is exact in this case. When the pursued faults are located in various nodes, the calculation of exact cross-entropy is computationally hard, as it would require calculating the joint probability distribution over the pursued fault nodes. In this section, we will introduce the diagnostic measures of information used for multiple fault diagnosis. Multiple fault diagnosis is invoked by pursuing more than one fault.



Once multiple faults are pursued, the selection of measures changes to a longer list



We will discuss each of these choices individually.

Max Probability Change

The extension of this measure to multiple faults amounts to taking the maximum change in the probability of a pursued fault over all pursued faults.

Cosine Distance

Cosine distance (also referred to as cosine similarity, https://en.wikipedia.org/wiki/Cosine_similarity) measures the distance between two vectors of numbers, in this case vectors of marginal probabilities of pursued faults, before and after making an observation. The larger the distance, the larger the impact of the observation. Cosine distance operates on vectors of probabilities and it naturally fits pursuit of multiple faults.

L2 Normalized Distance

L2 norm, also called Euclidean distance, Euclidean length, or L^2 norm (https://en.wikipedia.org/wiki/Norm_(mathematics)#Euclidean_norm) calculates the distance between two vectors in Euclidean space. This measure also naturally fits pursuit of multiple faults.

Cityblock Distance

City block distance (https://en.wikipedia.org/wiki/City_block) calculates the distance between two points in multidimensional space. The points are expressed by two vectors of numbers, in this case vectors of marginal probabilities of pursued faults, before and after making an observation. The larger the distance, the larger the impact of the observation. City block distance is close to Euclidean distance but is better suited for discrete/rounded probabilities. It operates on vectors of probabilities and it naturally fits pursuit of multiple faults.

Avg L2 Normalized & Cityblock

This measure of distance takes the average between the *L2 Normalized Distance* and *Cityblock Distance*.

Entropy-based Measures

Because entropy-based measures require calculation of the joint probability distribution over all pursued faults, which is computationally prohibitive, it is necessary to use approximations of the joint probability distribution. The approximations are based on two strong assumptions about dependencies among them: (1) complete independence (this is taken by the first group of approaches) and (2) complete dependence (this is taken by the second group of approaches). Each of the two extremes is divided into three groups: (1) *At Least One*, (2) *Only One*, and (3) *All*. These refer to different partitioning of the combinations of diseases in cross-entropy calculation.

In practice, each of the approximations will give a reasonable order of tests but in cases where it is important to be precise about the order of tests, it may be a good idea to try all three because all three are approximations and it is impossible to judge which of the approximations is the best without knowing the joint probability distribution over the faults.

Entropy/Independence/All

The assumption is that the pursued faults are independent of each other, *All* means opposing the event that all faults are present against the event that at least one fault is absent.

Entropy/Independence/At Least One

The assumption is that the pursued faults are independent of each other, *At Least One* means opposing the event that at least one of the faults is present against the event that none of the faults are present.

Entropy/Independence/Only One

The assumption is that the pursued faults are independent of each other, *Only One* means opposing the event that exactly one fault is present against all other possibilities (i.e., multiple faults or no faults present).

Entropy/Dependence/All

The assumption is that the pursued faults are perfectly dependent on each other, *All* means opposing the event that all faults are present against the event that at least one fault is absent.

Entropy/Dependence/At Least One

The assumption is that the pursued faults are perfectly dependent on each other, *At Least One* means opposing the event that at least one of the faults is present against the event that none of the faults are present.

Entropy/Dependence/Only One

The assumption is that the pursued faults are perfectly dependent on each other, *Only One* means opposing the event that exactly one fault is present against all other possibilities (i.e., multiple faults or no faults present).

Entropy/Marginal 1/2

The *Marginal* probability-based approach is much faster than the independence/dependence-based joint probability distribution approaches but it is not as accurate because it makes a stronger assumption about the joint probability distribution. Entropy calculations in this approach are based purely on the marginal probabilities of the pursued faults. The two algorithms that use the *Marginal Probability Approach* differ essentially in the function that they use to select the tests to perform. Both functions are scaled so that they return values between 0 and 1.

Entropy/Marginal 1 uses a function without the support for maximum distance and its minimum is reached when all probabilities of the faults are equal to 0.5.

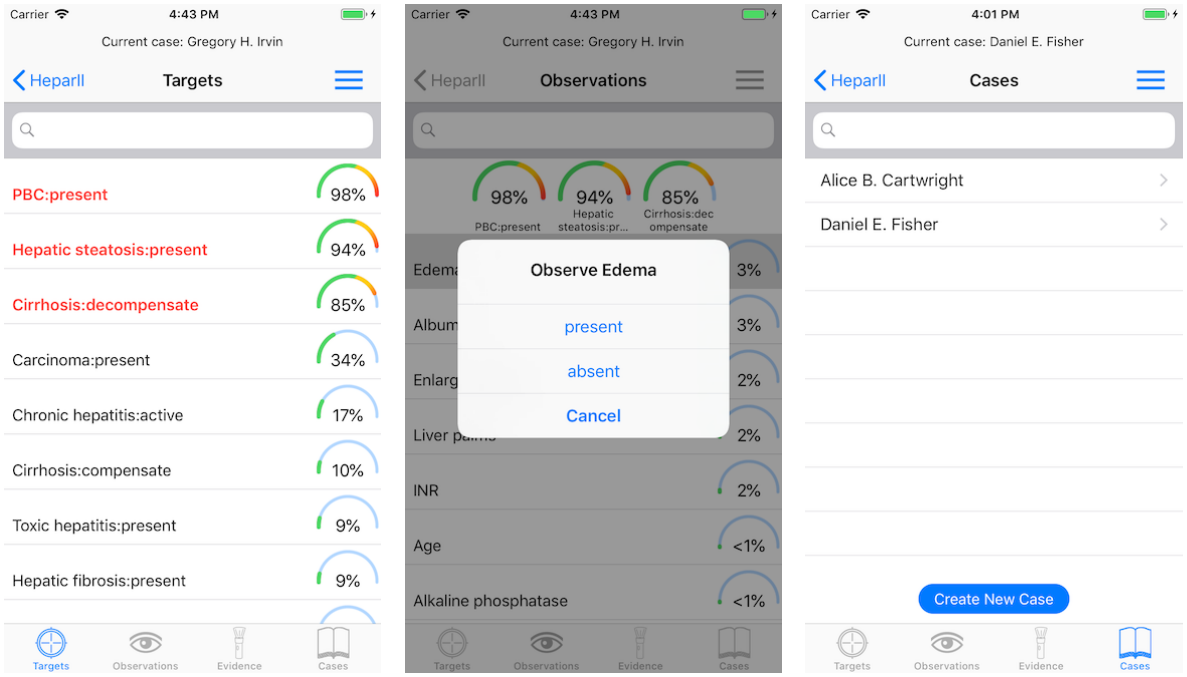
Entropy/Marginal 2 uses a function that has support for maximum distance and is continuous in the domain [0,1].

We would like to stress that none of the measures of distance supported by GeNIe (or any other systems for that matter) will offer optimality guarantees, i.e., guarantees that the order of tests suggested in the diagnosis dialog is optimal. Each of the measures will produce similar but not identical results and their optimality will depend on the model in which they are used (practically, depending on how well the model fits the assumptions made by each method). Users performing diagnosis interactively are encouraged to try several measures and pursue those observations that are indicated as the most informative consistently by various measures. When the models are embedded in end-user systems, we advise to select the measure that performs best in a given model. We recommend entropy-based methods in both single-fault and multiple fault diagnosis.

6.4.8 BayesMobile

BayesMobile is a mobile app that allows for interactive use of diagnostic GeNIe models. The app comes with several built-in example models from BayesFusion model repository. With an in-app purchase (at really minimal costs), any models can be loaded – either from device's local storage or from the cloud. BayesMobile has been developed for our clients' convenience and is intended to support rapid deployment of diagnostic models by our clients throughout their organizations.

Similarly to the *Diagnosis Window*, the BayesMobile app has four tabs, displaying (1) the marginal posterior probabilities of all fault variables, (2) observations that can be made rank-ordered according to their diagnostic value of information, (3) observations/evidence entered so far, and (4) library of diagnostic cases stored by the user. Cases can be imported and exported. Here are a few screen shots of BayesMobile screens:

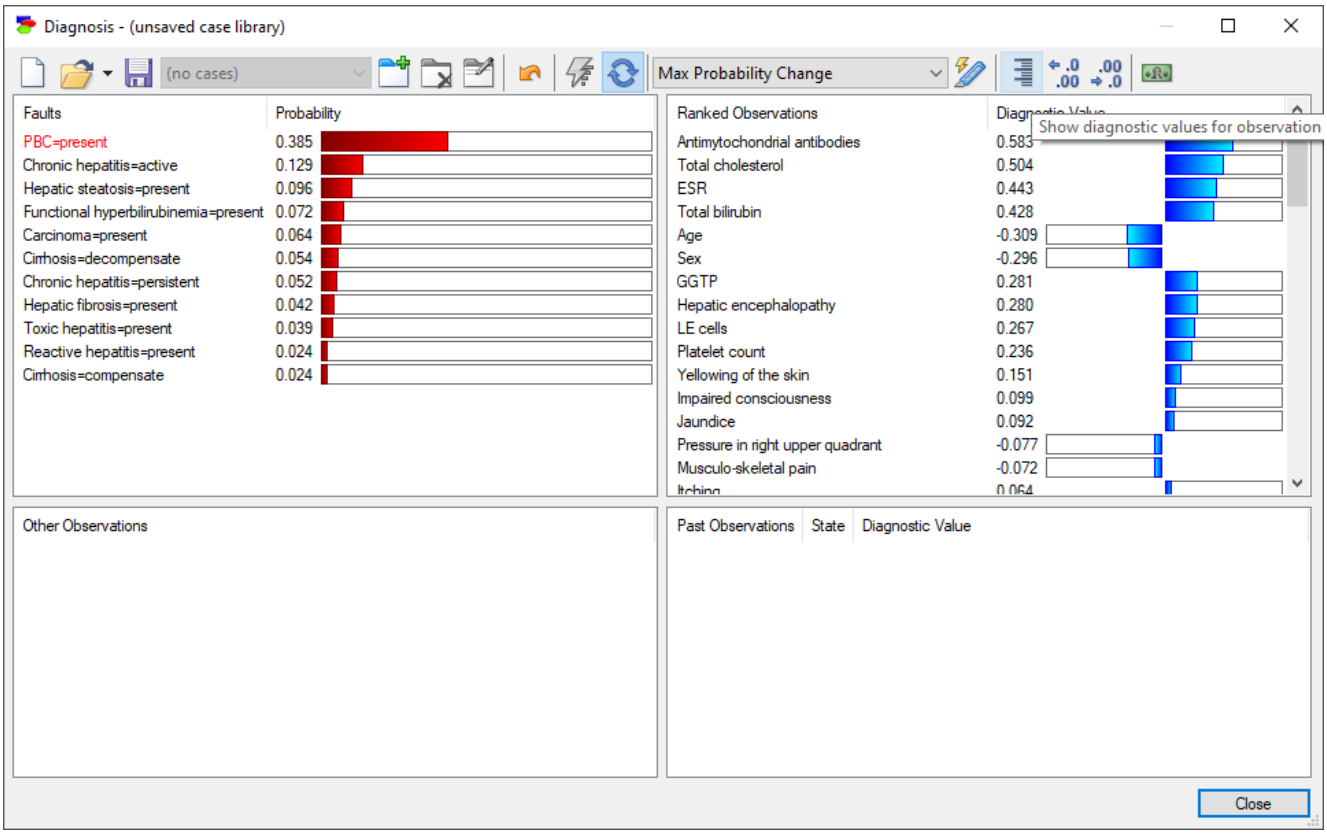


The Bayesian inference in the app is provided by SMILE. SMILE's low memory footprint and high performance make it a perfect fit for mobile development. At BayesFusion, we have the capacity to adjust BayesMobile to clients' needs or to develop a new mobile app altogether.

Click here to open BayesMobile page in AppStore.

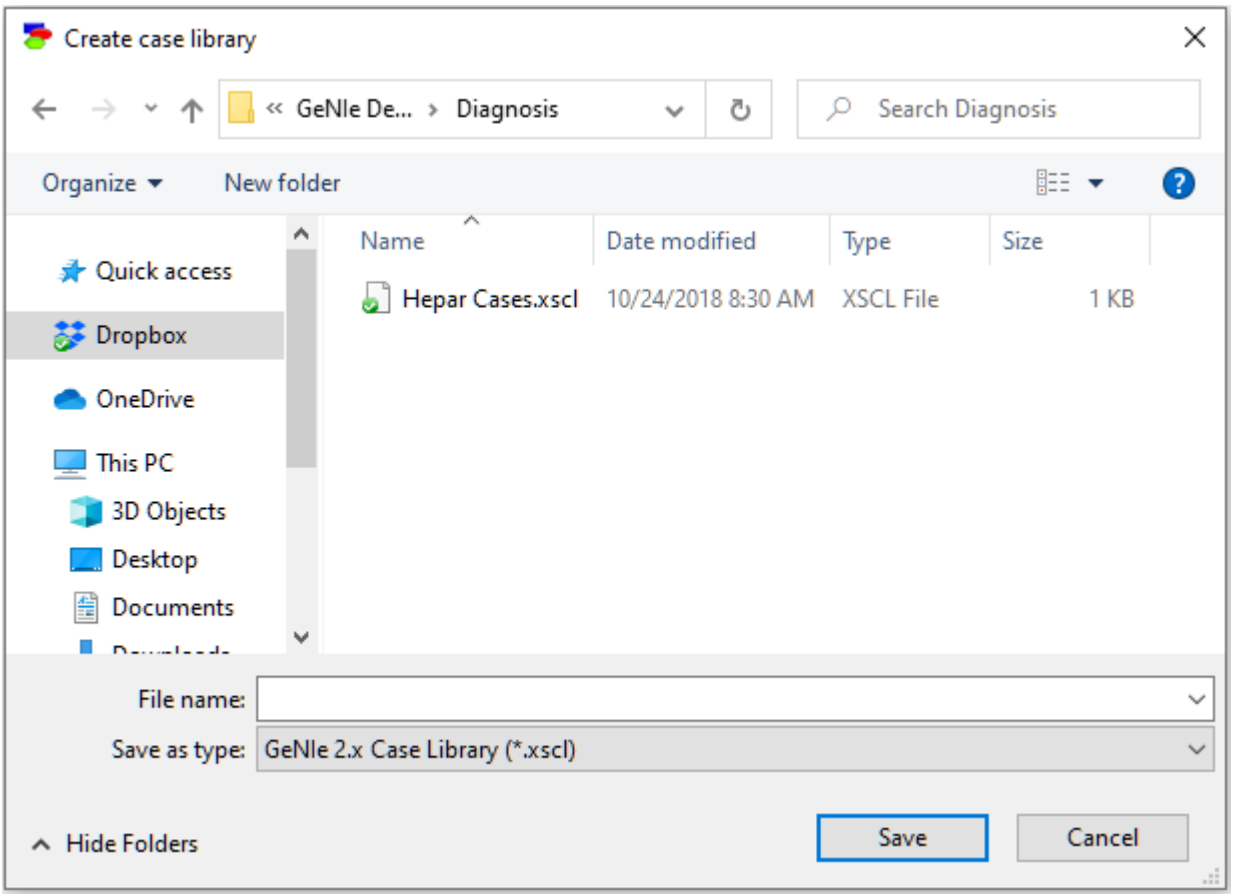
6.4.9 Diagnostic case management

The *Diagnostic window* allows users to store and preserve diagnostic cases that they worked on using the model through the *Case library* pop-up menu in the upper-left corner.



This functionality is similar to the *Case manager* view in the GeNIe workspace. The buttons on the left-hand side of the dialog header can be used to create ( ), open ( ), or save ( ) a case library.

The *New Case Library* button ( ) is used to create a new case library. It will open the *Create case library* dialog, which is a standard *Save file* dialog, as shown below:

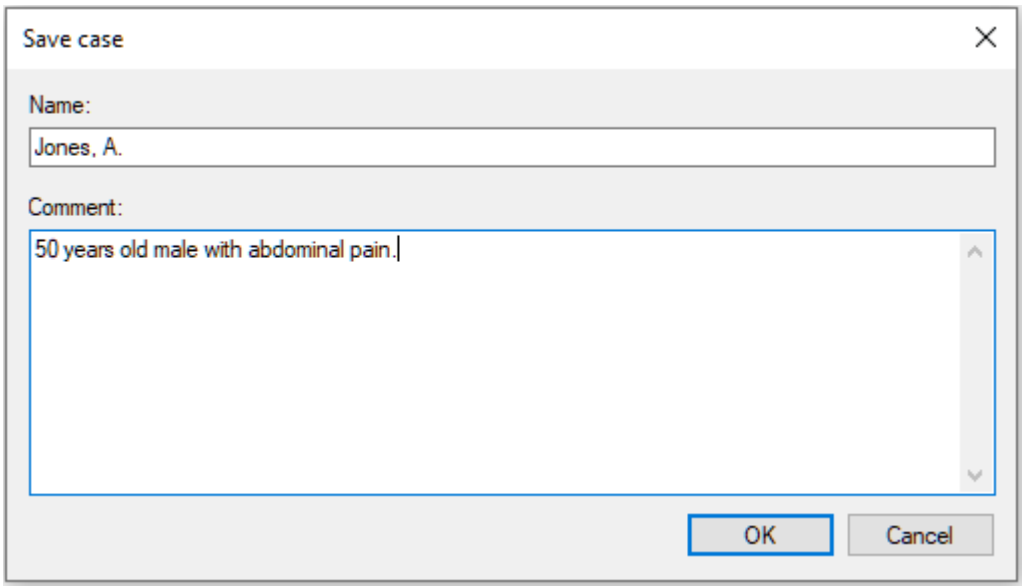


The *Open Case Library* button ( ) is used to open an existing case library. It will open a standard *Open file* dialog, which can be used to select the case library to open. Case libraries in GeNIe are saved as files with the extension *.xscl*.

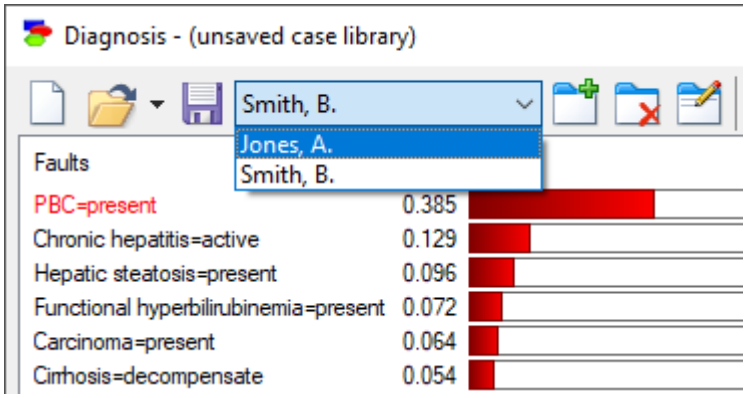
Each case library can contain multiple cases. While multiple case libraries can be created for any model, typically one case library is created for every network, and different cases for the network are saved within that case library. A case library is essentially a folder, within which you can organize your cases.

Adding/saving a case

Diagnostic cases can be added/saved. A new case will include *the case name*, *network used*, *entropy/cost ratio details*, *evidence nodes*, *pursued faults*, and an open ended *Comment*. Most of this information is available automatically but the *Case name* and the *Comment* are entered when creating the case. In order to save a case, a case library should be open. Once a case library is open the *Add case* ( ) button becomes active. After pressing it, the following dialog appears:

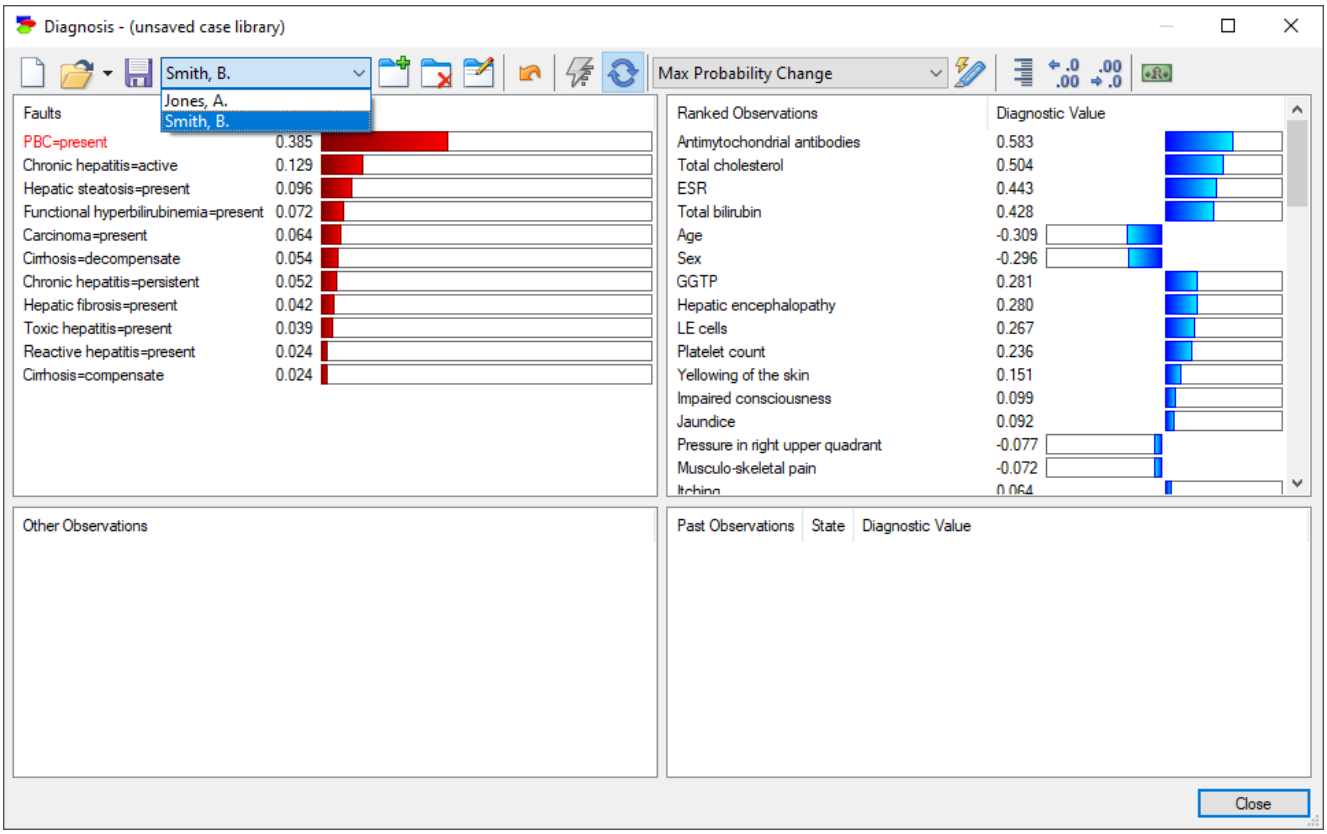


Enter the case *Name* and *Comment* and click *OK* to save the case within the currently open case library. When we click on the *Case list* pop-up menu, the list of cases in the library becomes visible:



Loading a diagnostic case

To load a case from a currently open case library, use the *Case list* pop-up menu. When clicked, the button shows a list of saved cases, from which we can select one. Once a case has been selected, it is loaded, which means that all evidence saved in the case is instantiated.

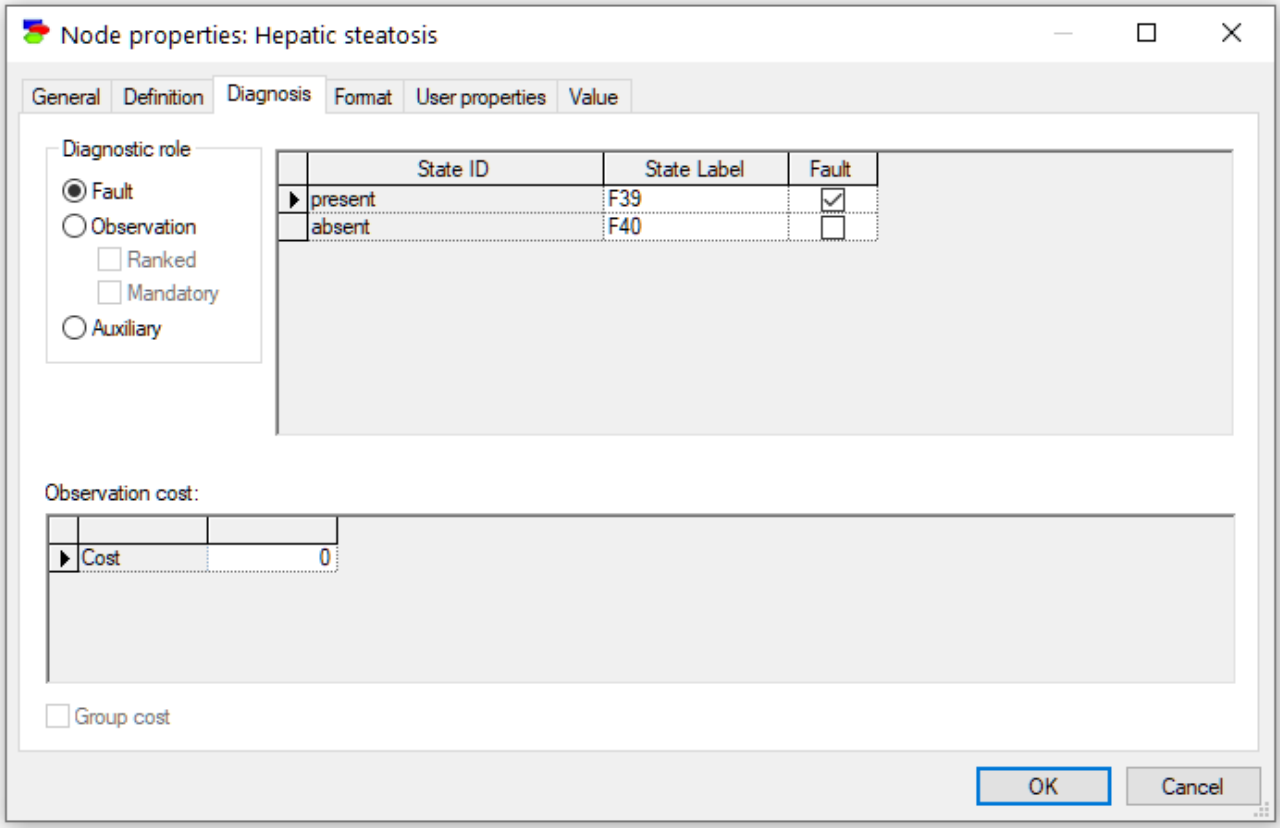


6.4.10 Cost of observation

Observing the value of a variable is often associated with cost. For example, in order to observe the platelet count, one has to draw a blood sample and subject it to examination by professionals. Measuring the temperature of an air conditioner exhaust unit requires a technician's time. In order to perform an optimal diagnosis, one has to take into account the value of information along with the cost of obtaining it. GeNIe allows for entering the cost of observing the value of a variable.

Simple costs

Simple cost is used when the cost of observing a node is independent of whether other nodes are observed or not. Simple cost could be the cost of performing a diagnostic test. This cost is set in the *Diagnosis* tab in *Node Properties* dialog:

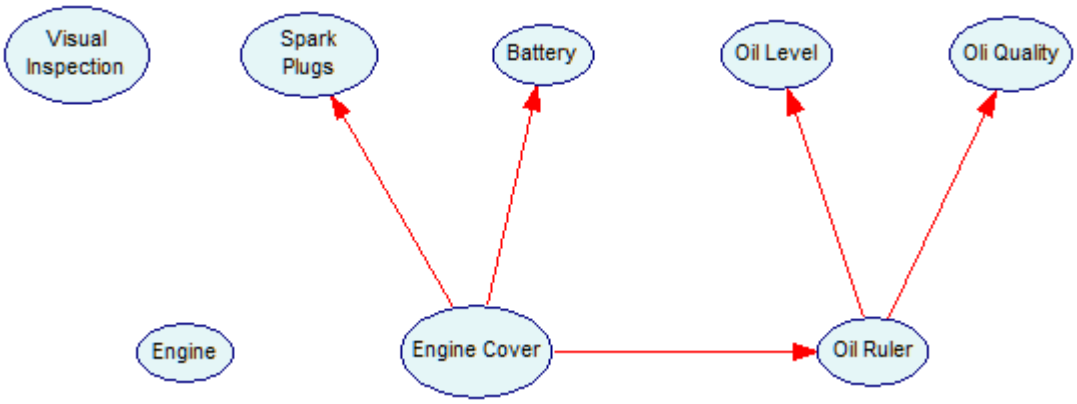


Simple costs can be also entered in the *Spreadsheet View*:

Conditional costs

Sometimes, costs of observing a variable are not independent of observing other variables. For example, once a blood sample is taken, performing additional tests on it is cheaper than performing these tests when no blood sample is available. The cost of measuring some parameter of a locomotive engine depends on whether the locomotive is in the shop or in the field. It may be much lower when the locomotive is in the shop. Taking off a locomotive cover may take a few hours but once it is removed, many tests become inexpensive. GeNIe represents conditional costs by means of an acyclic directed graph, available in the *Cost Graph View*.

The *Cost Graph View* is a modified version of the *Graph View* that shows the cost dependencies between the nodes of the network. The snapshot below is of the *Cost Graph View*.



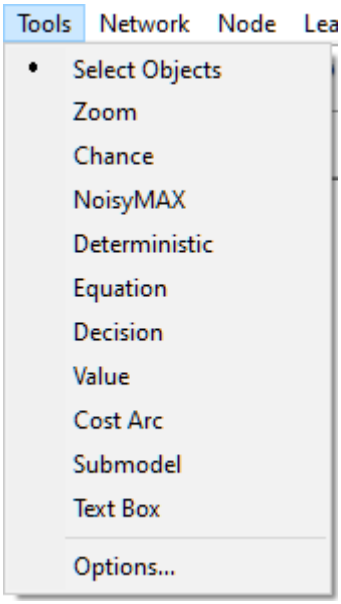
The *Cost Graph View* can be invoked by choosing the *Show Cost Arcs* switch from the *Diagnosis* menu. You can return to *Graph View* by choosing the *Show Cost Arcs* switch from the *Diagnosis* menu again.

The following changes will occur when the *Cost Graph View* is invoked:

1. All arcs of your graphical model (dark navy colored by default) will disappear and only the cost arcs (red colored by default) will appear between the nodes. If the cost arcs are not visible, then probably none are yet defined (this is the initial state for any model).

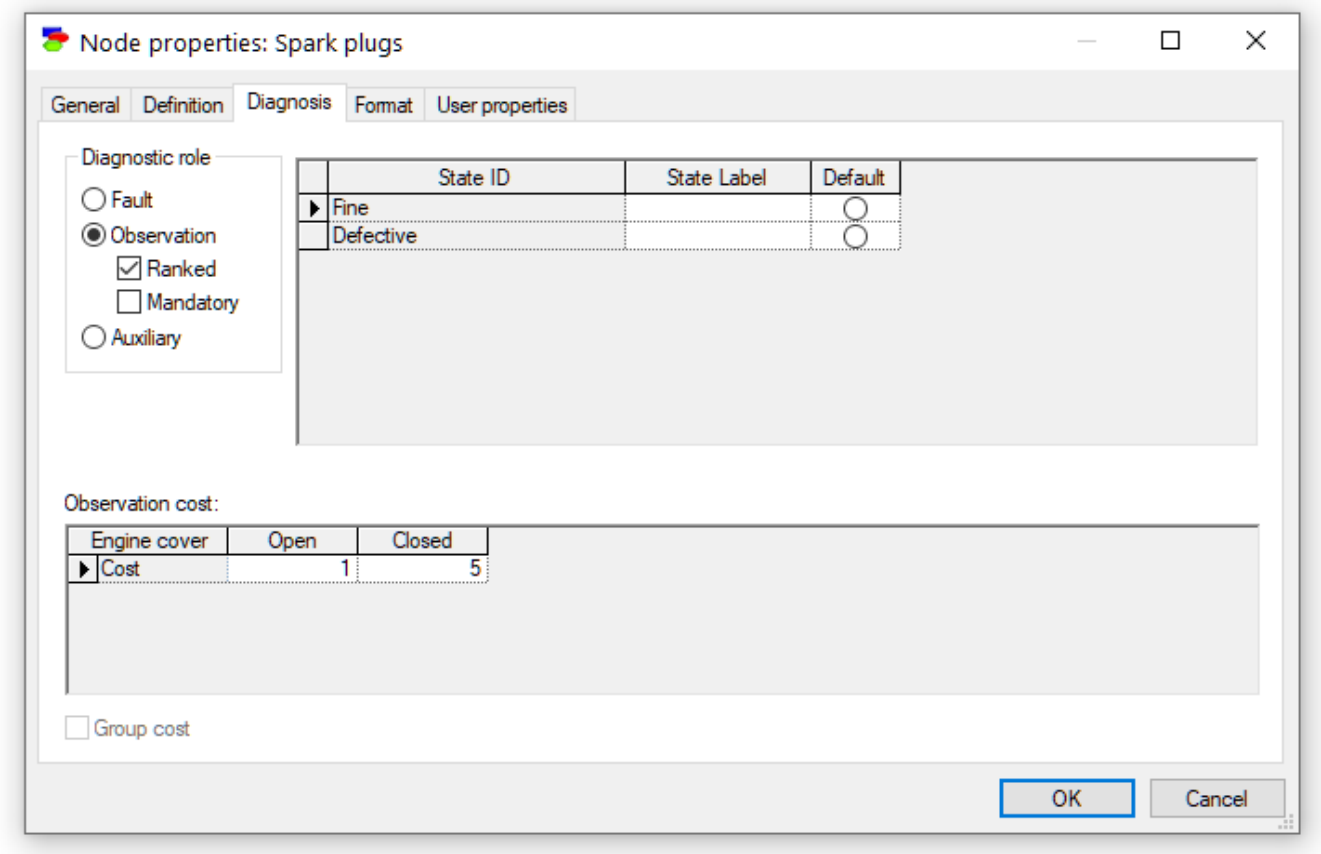
2. The *arc* ( ) button changes into a *cost arc* button ( ).

3. The *Arc* menu item in the *Tools Menu* changes to *Cost Arc*.

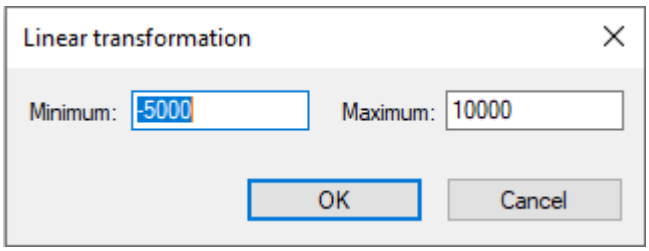


You can select the *Cost Arc* tool from either the *Tools Menu* or by clicking on the *Cost arc* button from the *Standard Toolbar* and add cost dependencies between nodes just as you add normal dependencies between nodes.

Once we have added a cost arc from the node *Engine cover* to the node *Spark plugs*, the cost tab shows two costs, one for when the engine cover is *Open* and one for when the engine cover is *Closed*.



The *Linear transformation* button allows for specifying the minimum and the maximum cost for a node in case of conditional costs. When pressed, it invokes the following dialog

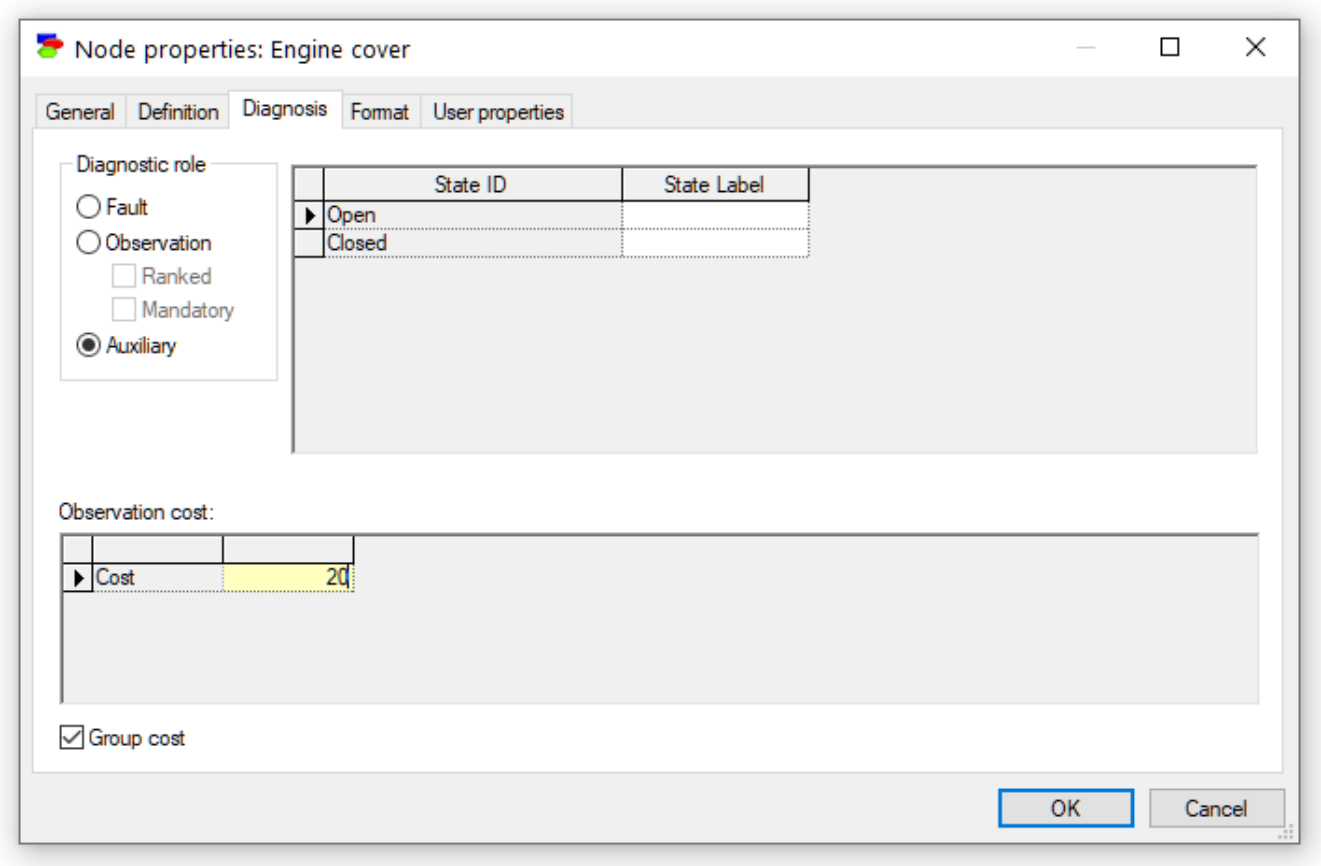


It performs a linear transformation of the costs, similarly to the transformation performed on utilities.

Group costs

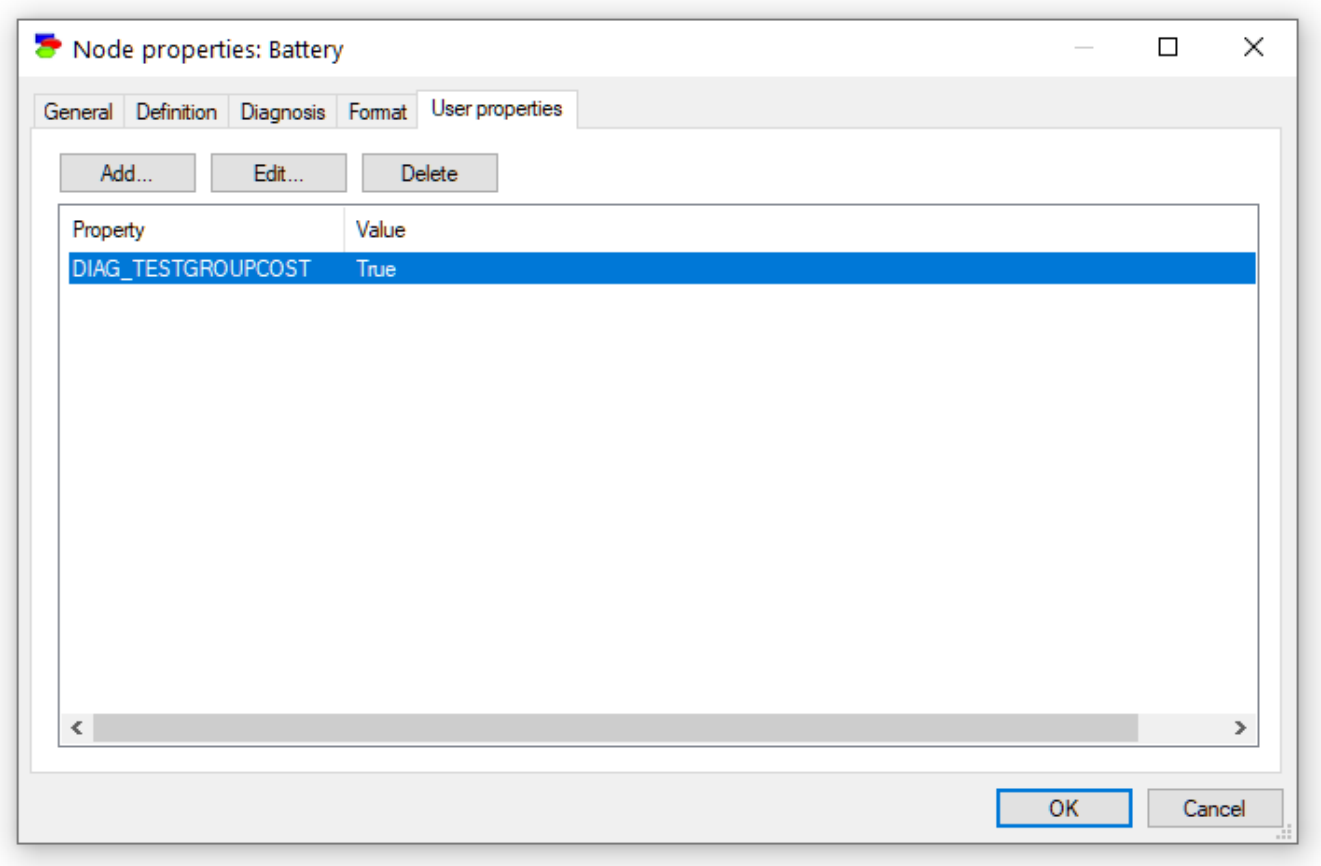
Group cost is used when the cost of observing a variable incurs a constant preparatory cost in addition to the cost of observing the variable. A typical example of a group cost is taking a blood sample, in which the cost of the blood sedimentation rate test incurs the cost of first drawing a blood sample from a patient. Another example is performing a test of an internal part of a locomotive engine, which involves removing the engine cover of a locomotive to access engine parts. Once a blood sample is drawn or the locomotive cover is open, any other tests will incur only simple costs. A group of nodes with a common group cost can be defined in the *Node properties* window. *Group cost* is one time cost associated with performing the first test of a particular group. The group cost node also designates the cost group.

To add a group cost to a node, the user must check the *Group cost* check box



The *Group cost* check box is enabled only if the current node has more than one child.

Subsequently, go to the *User Properties* tab and *Add* a property named, for example, *DIAG_TESTGROUPCOST*



The nodes that have the *DIAG_TESTGROUPCOST* property defined will incur the cost associated with opening the *Engine Cover*. As mentioned, this cost will only be applied to the nodes as long as none of the nodes of that group have been instantiated. Assume that the group cost of opening the *Engine Cover* is set to 20 units. In the example, the cost of observing the state of *Spark Plugs* is set at 2 units and the cost of checking the *Battery* is set at 5 units. As long as neither of the nodes is observed, the total cost of checking *Spark Plugs* is 22 units and the cost of checking *Battery* is 25 units. If either node, *Spark Plugs* or *Battery*, is set to a state, then the group cost will be set to 0 units. For example, if *Battery* is set to *Good*, the cost of performing the *Spark Plugs* test will now be 2 units. The 20 unit cost associated with the node *Engine Cover* no longer applies.

6.5 Learning

6.5.1 Data format

The data used by all GeNIe learning functionality has to conform to certain format constraints. Columns in data files correspond to variables and rows (records) to various values of these variables. Variable names (the first row in the data file) should be strings of alphanumerical characters with no spaces and no special characters (with the exception of underscore characters). Values of variables should either be strings of alphanumerical characters with no spaces and no special characters (with the exception of underscore characters) or, in case of continuous variables, numbers. Letters are a-z and A-Z but also all Unicode characters above codepoint 127, which allows using characters from other alphabets than the Latin alphabet.

GeNIe will be somewhat tolerant to minor departures from these requirements but we advise strict adherence because departure from the required format will lead to GeNIe replacing illegal characters, which may lead to later mismatch between a model and the data from which it was learned. Even if you depart from the strict adherence to the format, please make sure that you avoid some characters in the variables names. We have found the that underlying database does not like the "/" and "." characters.

Users often ask us about limitations on the size of the data file: (1) *What is the largest number of records in a data file that GeNIe can handle?*, and (2) *What is the minimum required number of records in a data file?* Both questions do not have clear-cut answers.

There is never too many data records in learning and as their number increases, the quality of the learned structures and parameters will increase as well. GeNIe deals with data quite efficiently and usually compresses them before or during learning. Should the number of data records cause a slowdown of learning (we have not encountered this but it is theoretically possible), we suggest reducing the data set size by selecting a representative sample of the records. As long as the sample reflects the properties of the original data, the learning procedure will be sound. To ensure that the sample is representative, one has to give every member of the population (the original huge set of records) an equal chance of being selected.

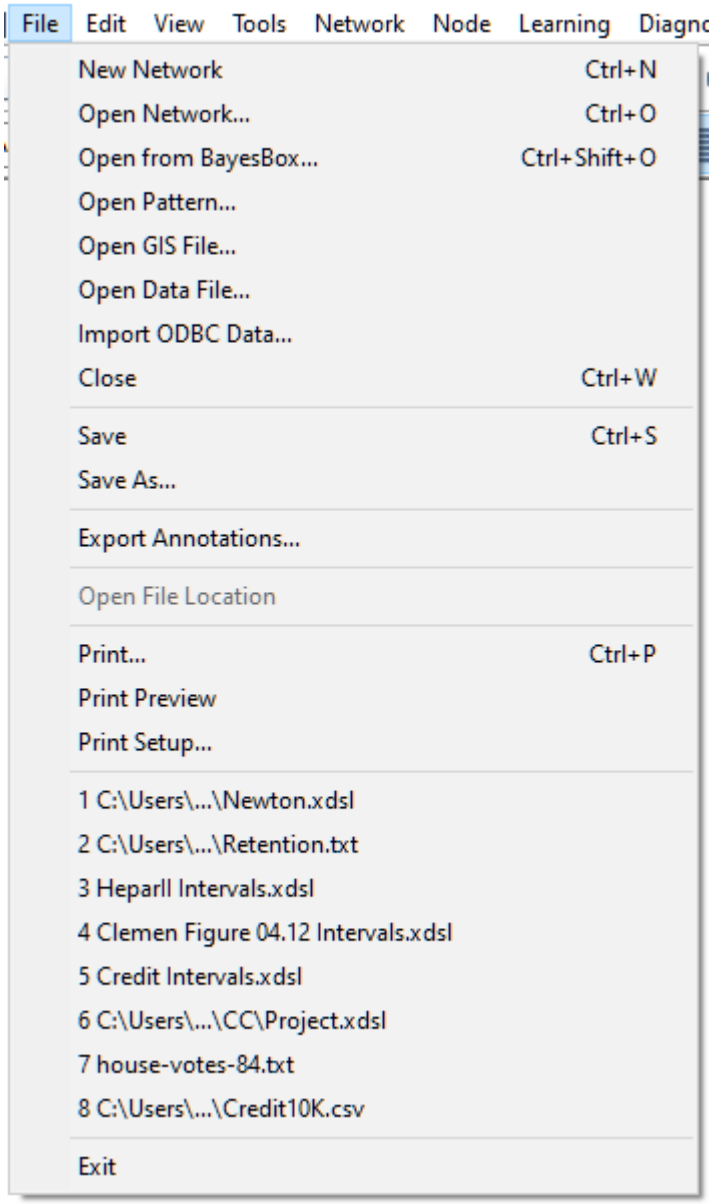
Too few records in a data file is a much more common problem in practice. Here, there is no clear-cut answer of what the minimum should be, as it depends on the properties of the underlying model. We have seen a heuristic "at least ten times as many records as the number of variables" but it is really just a heuristic that can be easily criticized. One way of estimating the number of records that one needs to learn a network is to look at the largest CPT in the model. When learning parameters, all records in the data have to be distributed among the columns of that largest CPT. Suppose we have 100 records in the data file and the largest CPT consists of 32 columns. The hundred records will have to be distributed among the 32 columns, which gives roughly three records per column on the average. It is hard to learn a probability distribution from just three data points. Another difficulty is that the distribution among the 32 columns is not going to be uniform and there will be many columns with zero records. This is a rough, order of magnitude argument but it should give an idea of how many records one needs to achieve a reasonable accuracy.

6.5.2 Accessing data

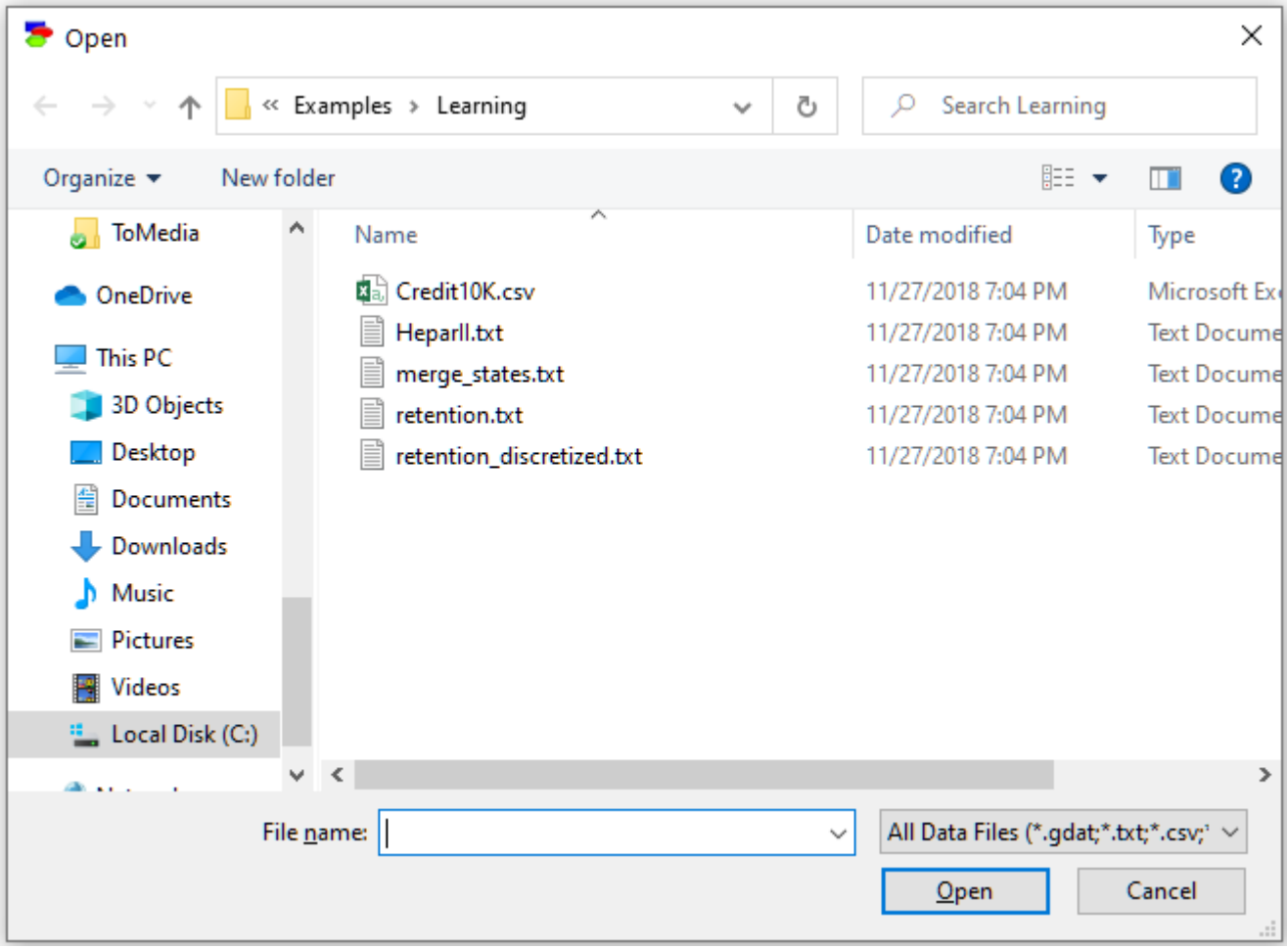
GeNIe can access data from three sources: text files, ODBC databases, and the native GeNIe data format. They will be subject of the following three sections.

Text Format (*.txt, *.dat, *.csv)

The simplest data format used by GeNIe is text format. Data in the text format consist of rows of records in text format, where values are separated by commas (*.csv format) or TAB characters (*.txt and *.dat formats). The first row in the data file contains variable IDs. Each of these IDs has to start with a letter, followed by letters, digits, and underscore characters. Letters are a-z and A-Z but also all Unicode characters above codepoint 127, which allows using characters from other alphabets than the Latin alphabet. The popular CSV format (used, among others, in Microsoft Excel), conforms to this standard. To access data stored in a text file select *File-Open Data File...*



Subsequently, select the data file that you wish to load.



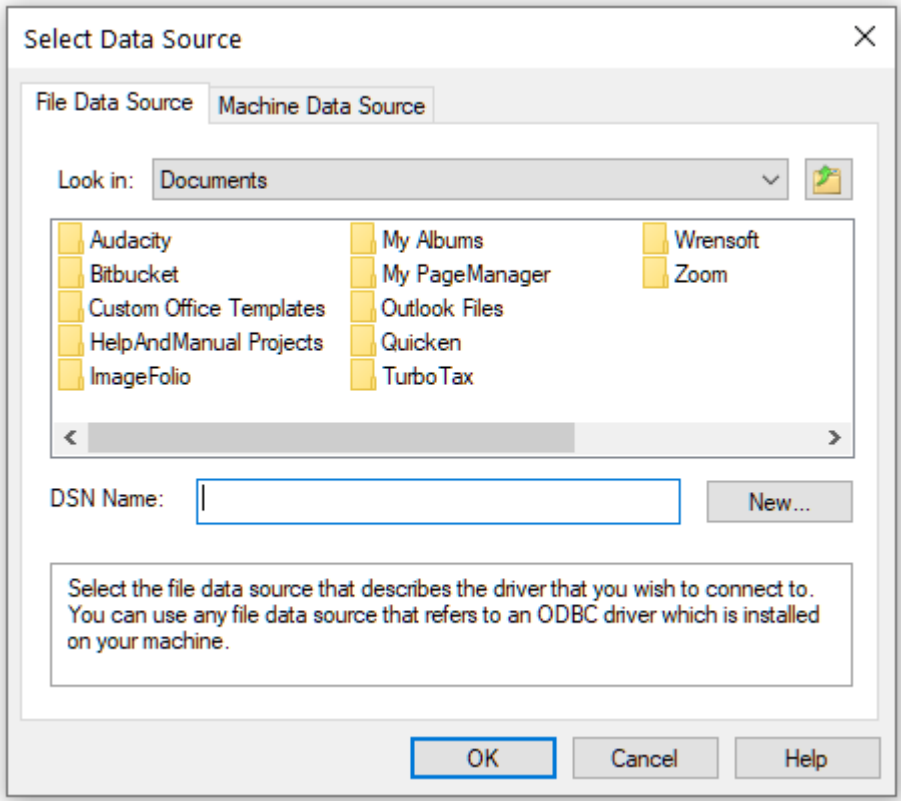
Data, once loaded, should look as follows:

spend	apret	top10	rejr	tstsc	pacc	strat	salar
7057	55.25	17	24.379	59.063	44.251	21.2	58200
16848	77.75	48	26.69	75.938	27.187	9.2	63000
18211	91	87	76.681	80.625	51.164	12.8	74400
21561	69.25	58	44.702	76.25	26.689	9.2	75400
20667	65	68	22.995	75.625	28.038	11	66200
10684	61.75	26	8.774	66	33.99	9.5	52900
11738	74.25	32	25.449	66.875	27.701	12	63400
10107	74	43	11.315	71	29.096	16.2	66200
7817	65.75	36	33.709	64.25	52.548	17.7	54600
7050	26	11	0	55.313	55.651	18.8	59500
9082	83.5	73	64.668	77.375	43.185	13.6	66700
11706	60	56	16.937	73.75	39.479	12.7	62100
7643	49.25	23	36.635	62.813	39.302	18.7	57700
25734	90	77	67.758	80.938	44.133	10	80200
20155	86	84	69.31	79.688	48.766	17.6	74000
29852	94.5	84	75.009	81.313	51.363	10.6	74100
7980	68.5	34	9.122	63.875	35.294	16.3	53100
8446	57	23	29.65	64.625	36.181	14.8	63200
24636	92.75	88	70.653	81.875	43.464	12.8	80300
7396	68.75	34	13.469	63.889	39.05	14.8	51900
24256	81.25	68	35.556	75	26.736	11.5	68200
7263	54	28	49.583	68.125	42.149	13.4	48839
7005	46.75	50	36.236	68.188	33.875	22.5	59600
10454	77.75	34	23.784	67.5	33.333	11.2	70000
13396	66.75	39	26.458	75	25.907	12.9	67100
18366	89.5	70	68.439	77.188	49.909	12.3	74000
10127	55.25	68	38.006	74.75	40.754	19.6	67100
7604	26.75	9	35.082	54.938	54.165	19.6	58000

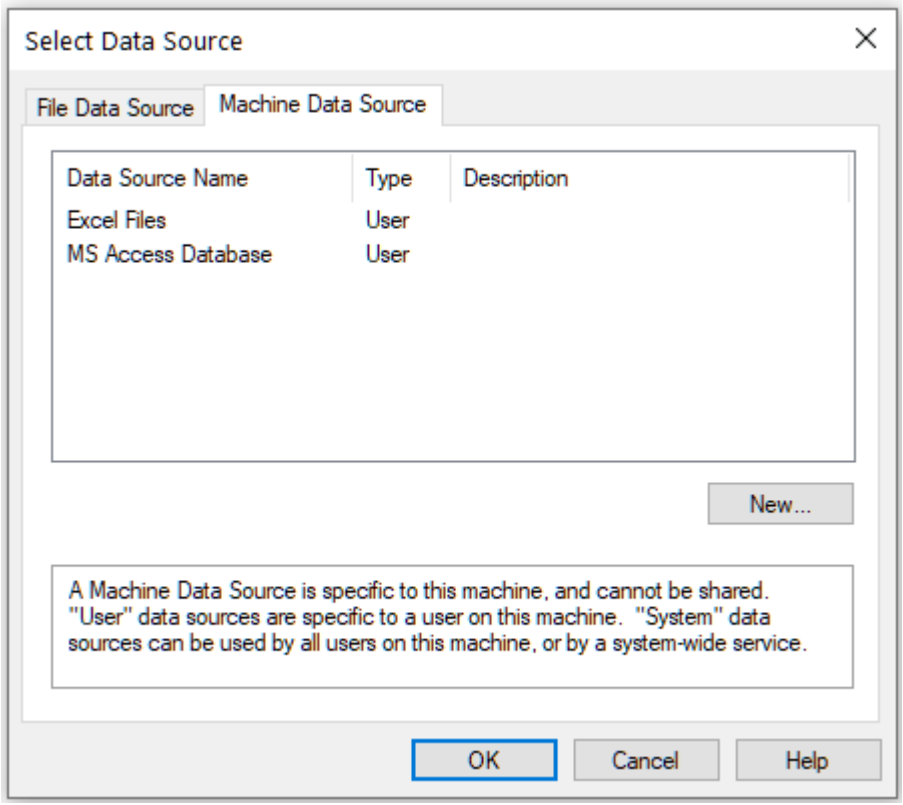
Row 1 of 170

ODBC Data

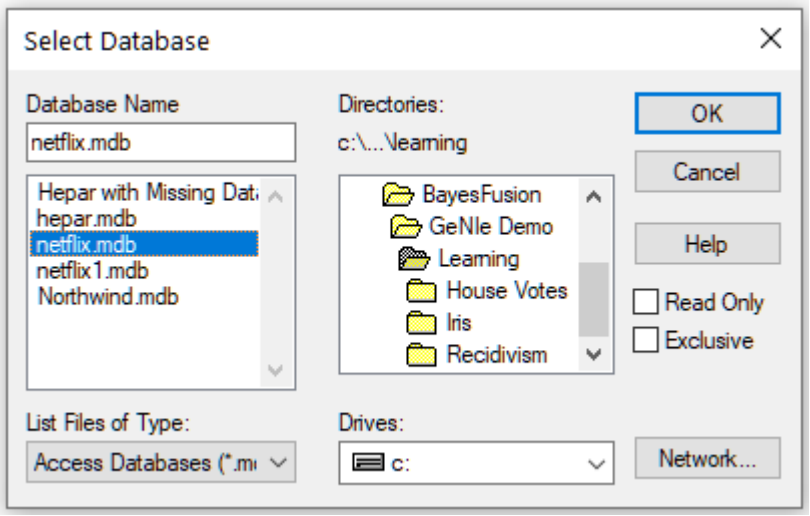
ODBC (Open Database Connectivity) is a standard application programming interface (API) for accessing database management systems (DBMS). ODBC is independent of the details of any concrete database system and operating systems. GeNIe implements the ODBC standard, which allows it to connect to most DBMS. In this section, we will open a Microsoft Access database. To access the data from a database select *File-Import ODBC Data...*, which will open the *Select Data Source* dialog.



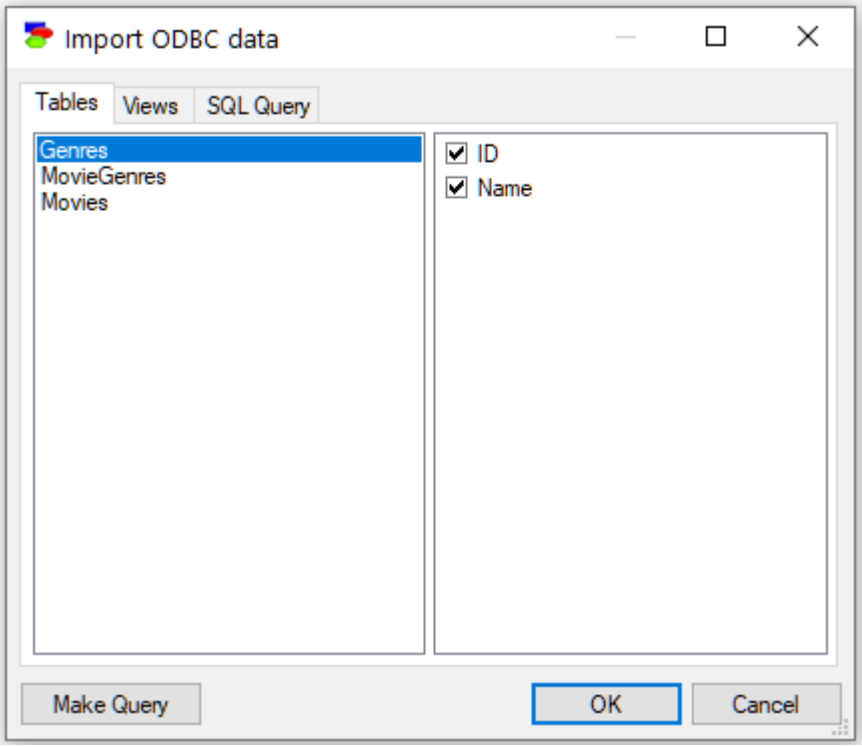
If you have never created a data source before, you will have to create a new one. It is most convenient to create a new data source that covers all files originating from a Windows application, which is a *Machine Data Source*. We will create a data source for Microsoft Access.



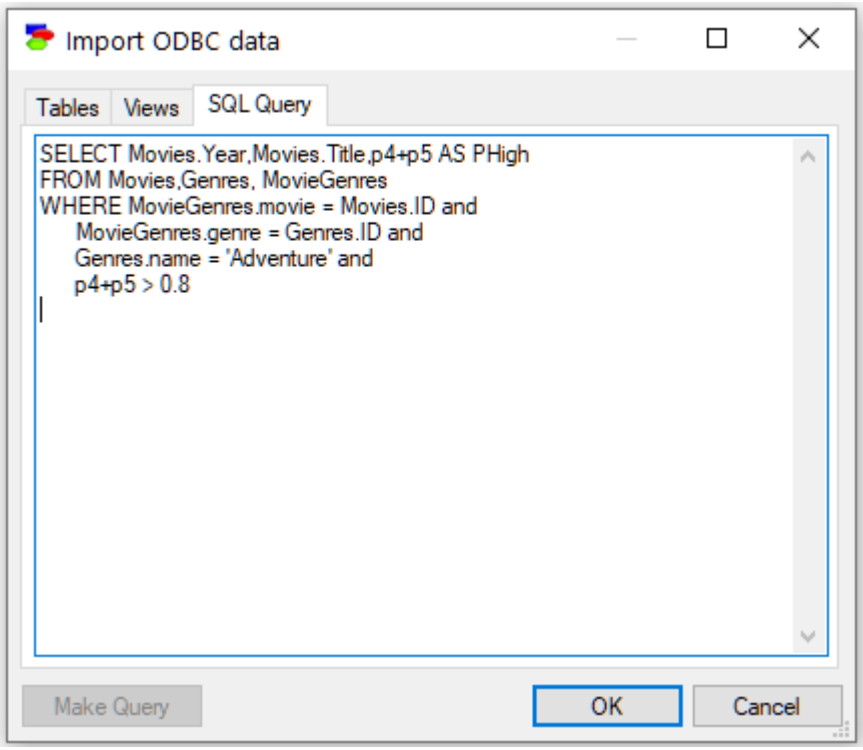
We select *Microsoft Access Database* and press *OK*. GeNIe will display a dialog box that allows for selecting data, that should look as follows:



We will open the *netflix.mdb* database. The ensuing dialog shows the tables (or views, if you select the *Views* tab) present in the database. Table *MovieGenres* contains two variables, *movie* and *genre*.



You can select a table, a view, or create a new table through an SQL query that you can type in the *SQL Query* tab.



Pressing *OK* runs the query and opens the result in GeNIe:

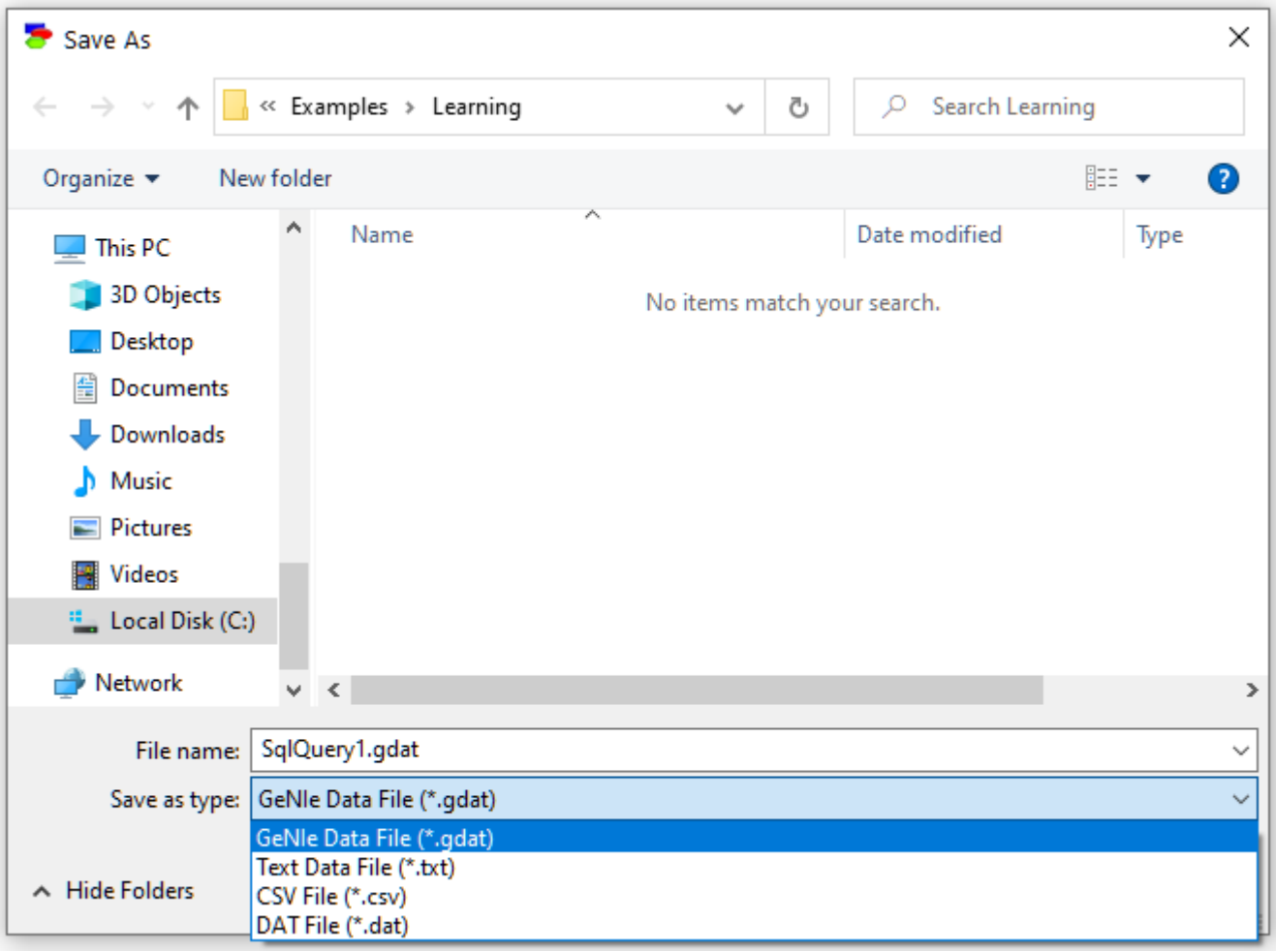
SqlQuery2

Year	Title	PHigh
2004	Lost: Season 1	0.941647
2004	Battlestar Galactica: Season 1	0.93818
1981	Raiders of the Lost Ark	0.908969
2004	Samurai Champloo	0.8825
2004	Smallville: Season 4	0.874619
2004	Case Closed: Season 5	0.866141
2004	Teen Titans: Season 2	0.863637
1995	Toy Story	0.858613
2002	Alias: Season 2	0.857559
1989	Indiana Jones and the Last Cru	0.852522
1990	Star Trek: The Next Generation	0.84949
2003	Alias: Season 3	0.849061
1992	Star Trek: The Next Generation	0.846714
2001	Alias: Season 1	0.846131
2001	Farscape: Season 3	0.844224
2004	The Incredibles	0.842713
2002	Farscape: Season 4	0.840428
1993	Star Trek: The Next Generation	0.836429
1991	Star Trek: The Next Generation	0.834596
1987	The Princess Bride	0.832354
2004	Farscape: The Peacekeeper	0.831399
2005	Batman Begins	0.829358
2003	Read Or Die	0.827824
1988	Star Trek: The Next Generation	0.824505
1963	The Great Escape	0.816221
2003	Smallville: Season 3	0.813642
2000	Farscape: Season 2	0.81321
2002	Smallville: Season 2	0.80966
1959	North by Northwest	0.809471
1999	Toy Story 2	0.809113
2004	Harry Potter and the Prisoner of	0.807584
2000	Gladiator	0.805826
1995	Ninja Scroll	0.804775
1954	Seven Samurai	0.800669

Row 1 of 34

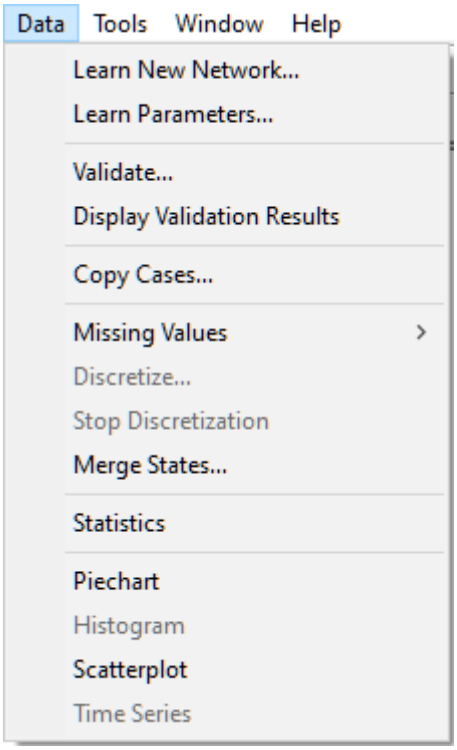
GeNIe Data Format (*.gdat format)

GeNIe allows to save data in a binary internal format that we call GeNIe Data Format (*.gdat). The biggest advantage of this data format is that it allows for saving all useful information, such as the original values in the data, the replaced missing values, discretization information, and even column widths. Because the format includes the original data, it is always possible to reverse all data preprecessing operations, such as discretization. To save your data in *GeNIe Data Format*, select *File-Save As...*



6.5.3 Data menu

Once a data set is open, *Menu Bar* is extended with *Data* menu.



The *Data Menu* offers the following functions, all discussed in various sections of this chapter:

Learn New Network... starts a dialog for learning a new graphical structure from the data set. See *Structural learning* section for more information.

Learn Parameters... allows for learning the parameters of an existing network from the data set. See *Learning parameters* section for more information.

Copy Cases... allows for importing cases from the data into a network. This function is described in the *Case manager* section.

Data pre-processing commands

The remaining choices in the Data Menu offer various functions for viewing and pre-processing data. They are all described in detail in the *Cleaning data* section.

Missing Values submenu gives various options for dealing with missing values in the data.

Discretize... opens a dialog for discretizing the data.

Stop Discretization returns the data to their original values, reversing the discretization.

Merge States... opens a *Merge States* dialog that allows the user to merge states of a variable and assign a new name to the merged states.

Statistics opens a *Statistics Dialog* that displays useful statistics for the variables included in the data.

Piechart displays the distribution of data for the selected variable on a piechart graph.

Histogram displays the distribution of data for the selected variable on a histogram graph.

Scatterplot displays the scatterplot graph for two selected variables.

Time Series displays a plot of values of the selected variable indexed by the record number.

6.5.4 Cleaning data

Before the structure of a Bayesian network (or the numerical parameters of an existing network) can be learned, it is a good idea to interactively examine the data. GeNIe allows for interactive editing data files in several aspects described in sections below. For the purpose of this section, we will be using data file *retention.txt*, which can be found in the examples sub-folder of the GeNIe installation folder.

Data Grid view

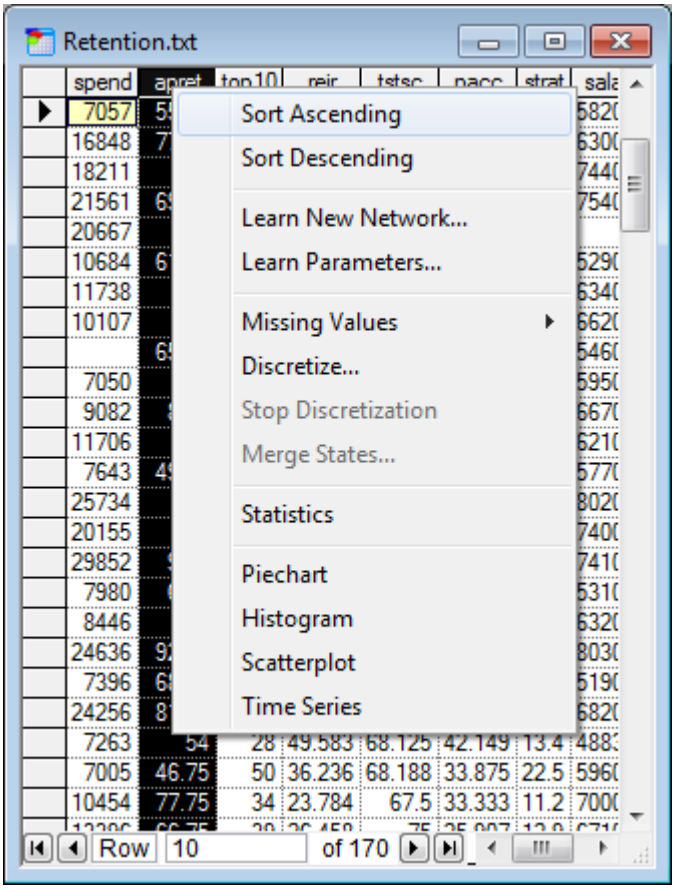
GeNIe uses the *Data Grid* view to display the contents of a data file. The grid is similar to a spreadsheet, like Microsoft Excel, and allows the users to analyze and clean the data. The bottom of the window shows the number of the row in which the cursor resides, along with the total number of rows. It is possible to type the desired row number and press *Return*, which will place the cursor in the row with that number.

The *Data Grid* view allows also for zooming in and out, similarly to the *Graph View*.

spend	apret	top10	rejr	tstsc	pacc	strat	salar
7057	55.25	17	24.379	59.063	44.251	21.2	58200
16848	77.75	48	26.69	75.938	27.187	9.2	63000
18211	91	87	76.681	80.625	51.164	12.8	74400
21561	69.25	58	44.702	76.25	26.689	9.2	75400
20667	65	68	22.995	75.625	28.038	11	66200
10684	61.75	26	8.774	66	33.99	9.5	52900
11738	74.25	32	25.449	66.875	27.701	12	63400
10107	74	43	11.315	71	29.096	16.2	66200
7817	65.75	36	33.709	64.25	52.548	17.7	54600
7050	26	11	0	55.313	55.651	18.8	59500
9082	83.5	73	64.668	77.375	43.185	13.6	66700
11706	60	56	16.937	73.75	39.479	12.7	62100
7643	49.25	23	36.635	62.813	39.302	18.7	57700
25734	90	77	67.758	80.938	44.133	10	80200
20155	86	84	69.31	79.688	48.766	17.6	74000
29852	94.5	84	75.009	81.313	51.363	10.6	74100
7980	68.5	34	9.122	63.875	35.294	16.3	53100
8446	57	23	29.65	64.625	36.181	14.8	63200
24636	92.75	88	70.653	81.875	43.464	12.8	80300
7396	68.75	34	13.469	63.889	39.05	14.8	51900
24256	81.25	68	35.556	75	26.736	11.5	68200
7263	54	28	49.583	68.125	42.149	13.4	48839
7005	46.75	50	36.236	68.188	33.875	22.5	59600
10454	77.75	34	23.784	67.5	33.333	11.2	70000
13396	66.75	39	26.458	75	25.907	12.9	67100
18366	89.5	70	68.439	77.188	49.909	12.3	74000
10127	55.25	68	38.006	74.75	40.754	19.6	67100
7604	26.75	9	35.082	54.938	54.165	19.6	58000

Row 1 of 170

The columns hold variables (their IDs are in the first, grayed row), rows contain data records, which are simultaneous measurements of the states of the variables.



Similarly to Excel, *Data Grid* view allows for sorting data in ascending and in descending order. It is possible to sort using more than one column - just select them with *CTRL* or *SHIFT*. Please note that the order of selecting columns impacts the sort criteria. For example, with three columns (*A*, *B*, and *C*), if you click on *C* first and *A* second (with *CTRL* key down), C will be used first in sorting. The value of *A* will be relevant only if multiple records contain the same value in *C*. Sorting over multiple columns is useful if we want to find records in the data file containing a certain combination of values of variables. They are fairly easy to identify in a file sorted by the relevant columns.

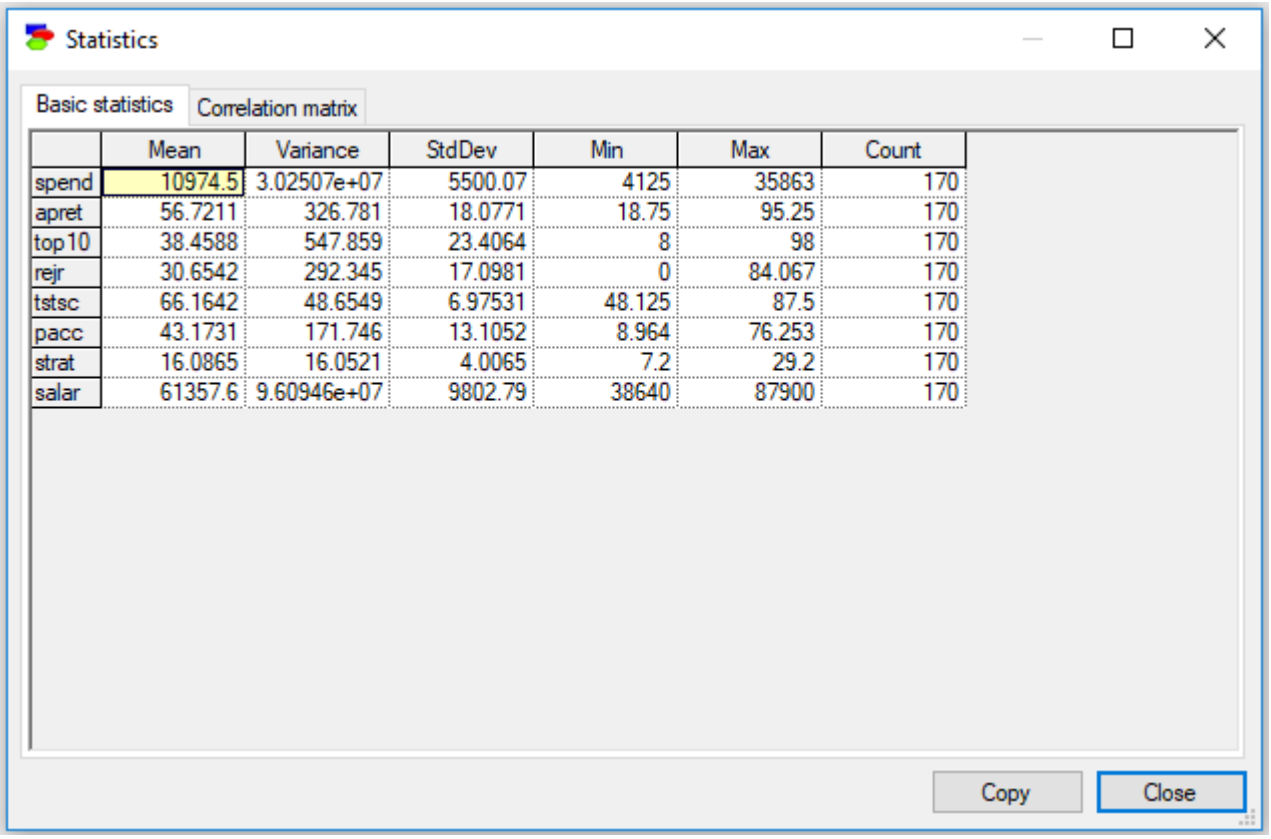
The values of cells in the *Data Grid* view can be edited, similarly to cells in Excel. Rows can be deleted, which corresponds to removing data records from analysis. Groups of cells can be selected, copied, and subsequently pasted, both within the *Data Grid* view and between *Data Grid* view and external applications. Commands of the *Edit Menu*, such as *Find* and *Replace* can also be used in the *Data Grid* view, although *Replace All* takes effect only on the selected column.

Currently the *Data Grid* does not allow for adding, deleting, and renaming columns. Some of our users perceive this as a limitation and we are planning to add this functionality to one of our future releases. Deleting a column is easy to cope with currently, as learning always allows for selecting columns, so those columns that one might want to delete can be just omitted from the selection. Adding new columns can be simulated by creating a few additional dummy columns before data are read into GeNIe. Then, if an additional column is needed, one can paste data into one of those dummy columns and use is subsequently as a newly added column.

Changes to the *Data Grid* view are not transferred to the data file. In order to make the changes permanent, you need to explicitly save the data file.

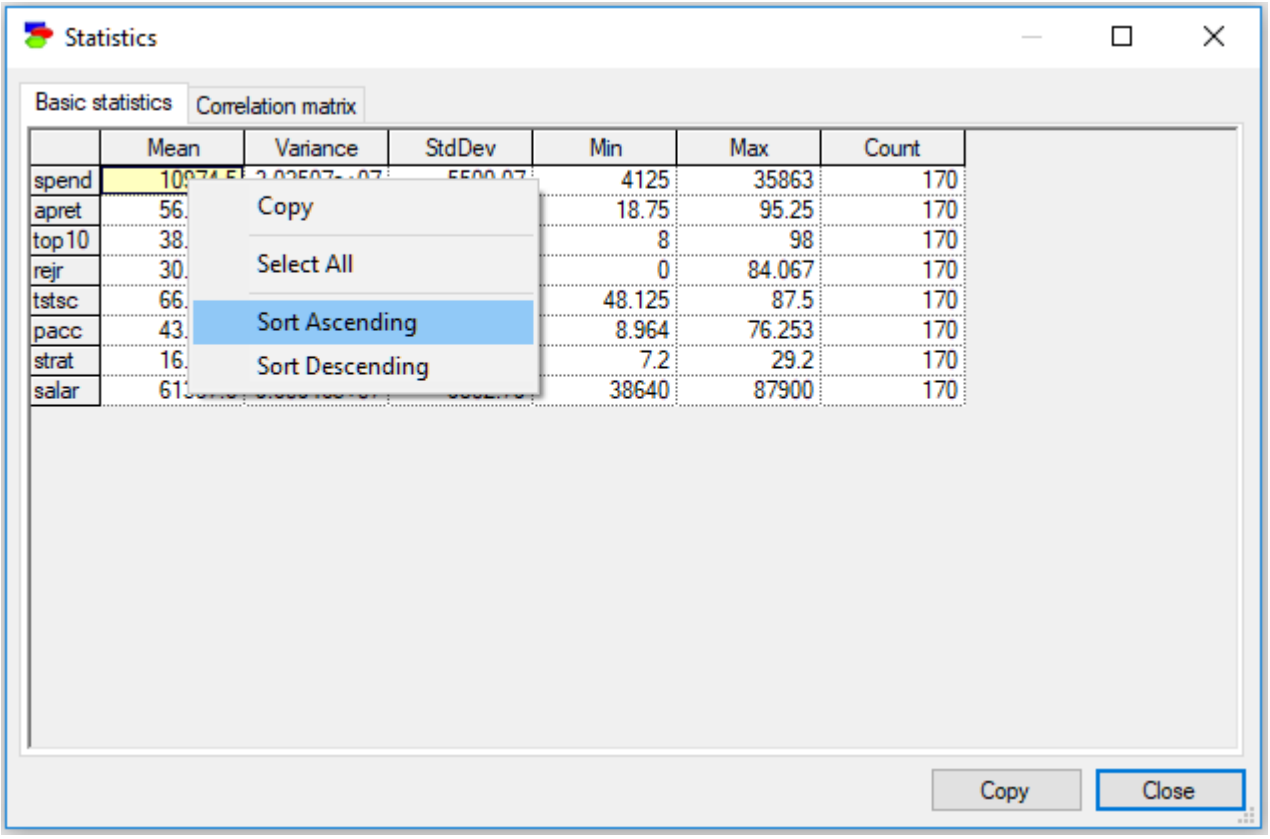
Statistic

GeNIe allows to display basic statistics for the variables in the data set. These include: mean, variance, standard deviation, minimum and maximum value, and count (number of values in the column). To invoke the *Statistics* dialog, select *Data-Statistics*.

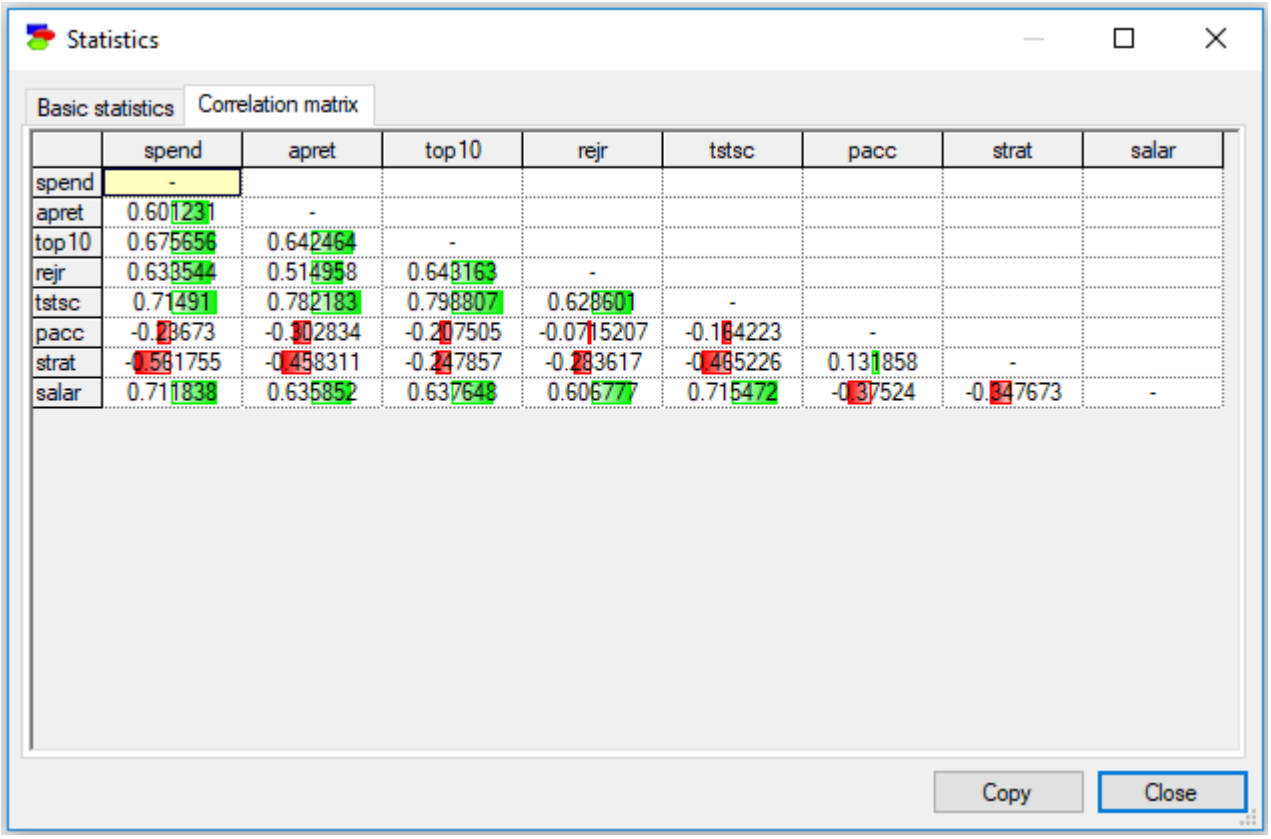


	Mean	Variance	StdDev	Min	Max	Count
spend	10974.5	3.02507e+07	5500.07	4125	35863	170
apret	56.7211	326.781	18.0771	18.75	95.25	170
top10	38.4588	547.859	23.4064	8	98	170
rejr	30.6542	292.345	17.0981	0	84.067	170
tstsc	66.1642	48.6549	6.97531	48.125	87.5	170
pacc	43.1731	171.746	13.1052	8.964	76.253	170
strat	16.0865	16.0521	4.0065	7.2	29.2	170
salar	61357.6	9.60946e+07	9802.79	38640	87900	170

Values in various columns of the Basic statistics table can be sorted

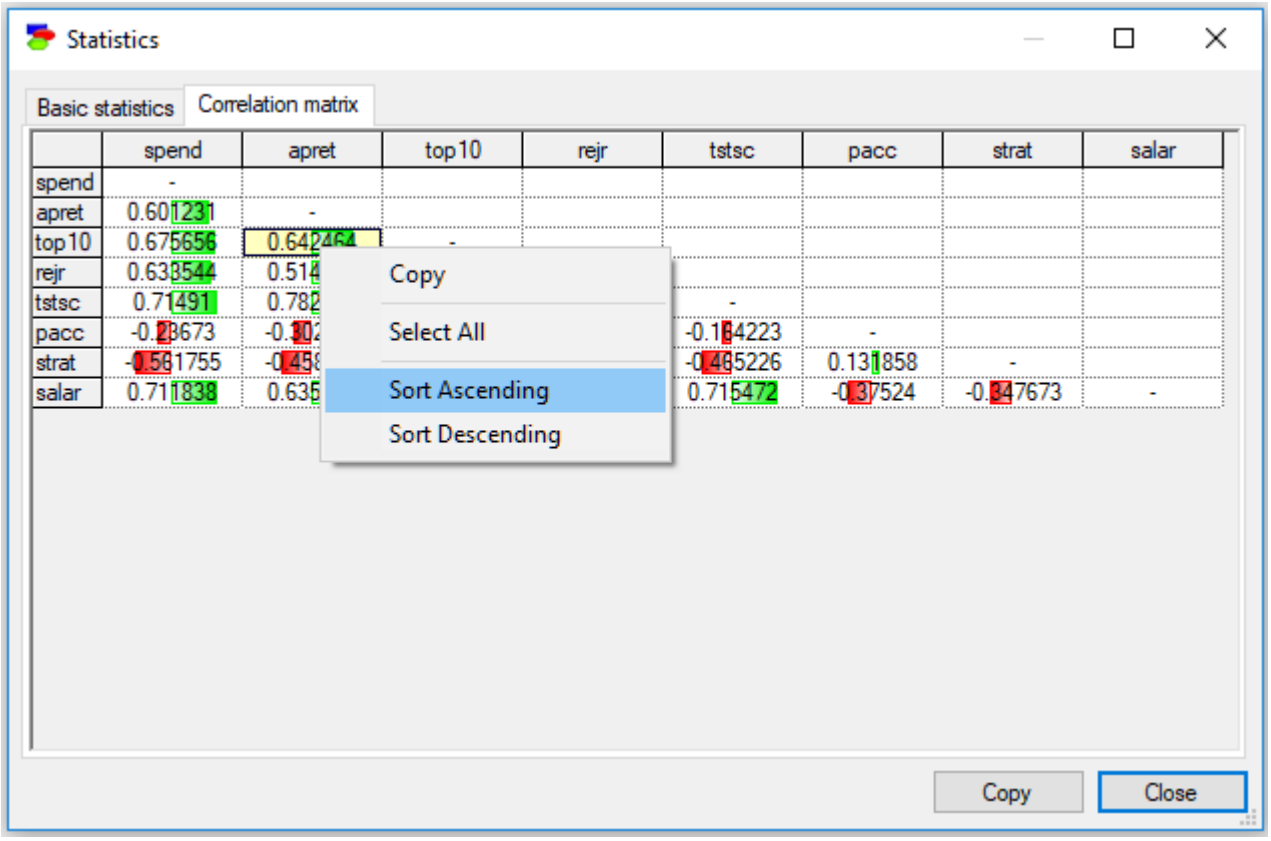


In case of continuous variables following the multi-variate Gaussian distribution, *Correlation matrix* (second tab) offers also insightful information, showing correlations between pairs of variables. These correlations are indications of strength of (possibly indirect) relationships. Horizontal bars inside cells show the correlations graphically for easier visual identifications, green bars represent positive and red bars represent negative correlations.

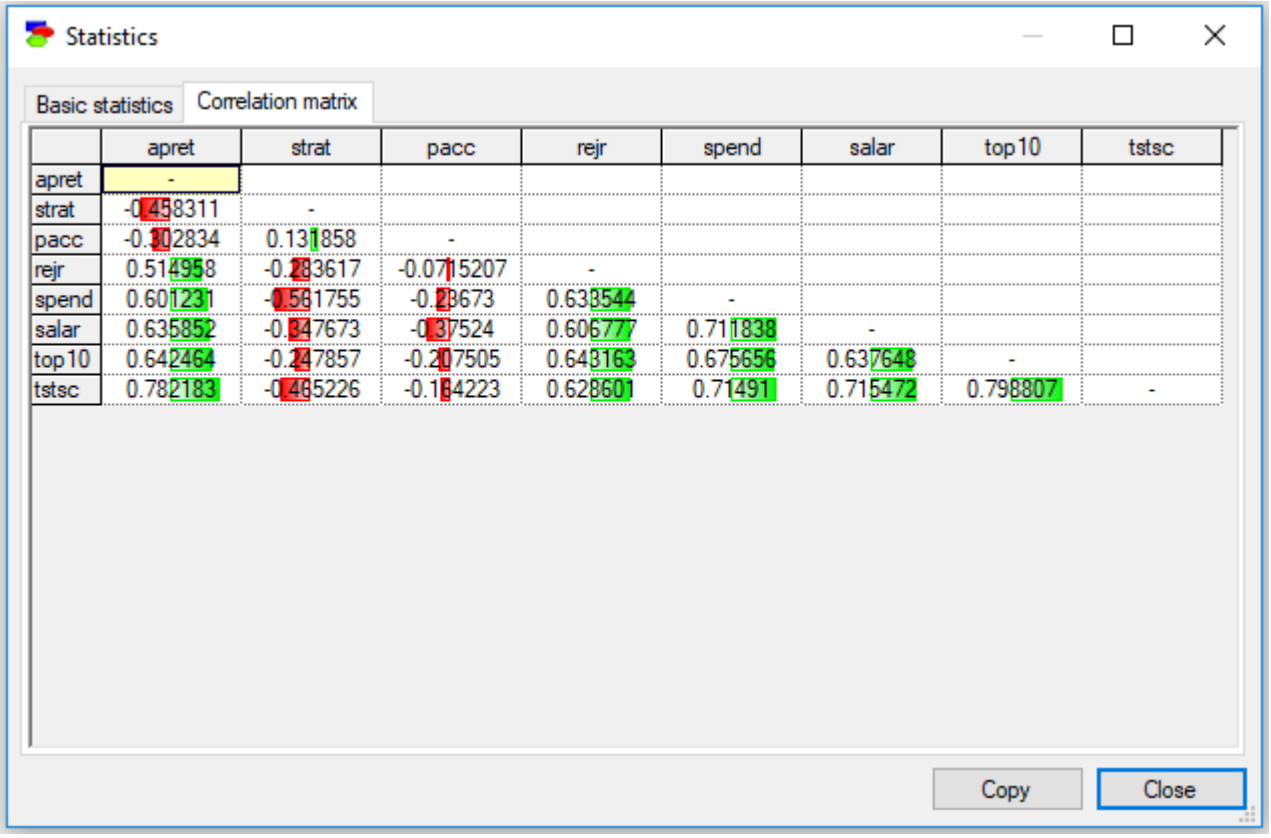


	spend	apret	top10	rejr	tstsc	pacc	strat	salar
spend	-							
apret	0.601231	-						
top10	0.675656	0.642464	-					
rejr	0.638544	0.514958	0.643163	-				
tstsc	0.71491	0.782183	0.798807	0.628601	-			
pacc	-0.23673	-0.302834	-0.207505	-0.0715207	-0.164223	-		
strat	-0.561755	-0.458311	-0.247857	-0.283617	-0.465226	0.131858	-	
salar	0.711838	0.635852	0.637648	0.606777	0.715472	-0.37524	-0.347673	-

This tab can be sorted by individual columns as well and this operation is very useful in case of large tables - we may be able to find highest correlations between a chosen variable and the remaining variables.



In this case, the variable that we are sorting on (*apret* in the screen shot above), is moved to the left-most column, so that all correlations with the variable can be shown in one column while preserving the triangularity of the matrix.



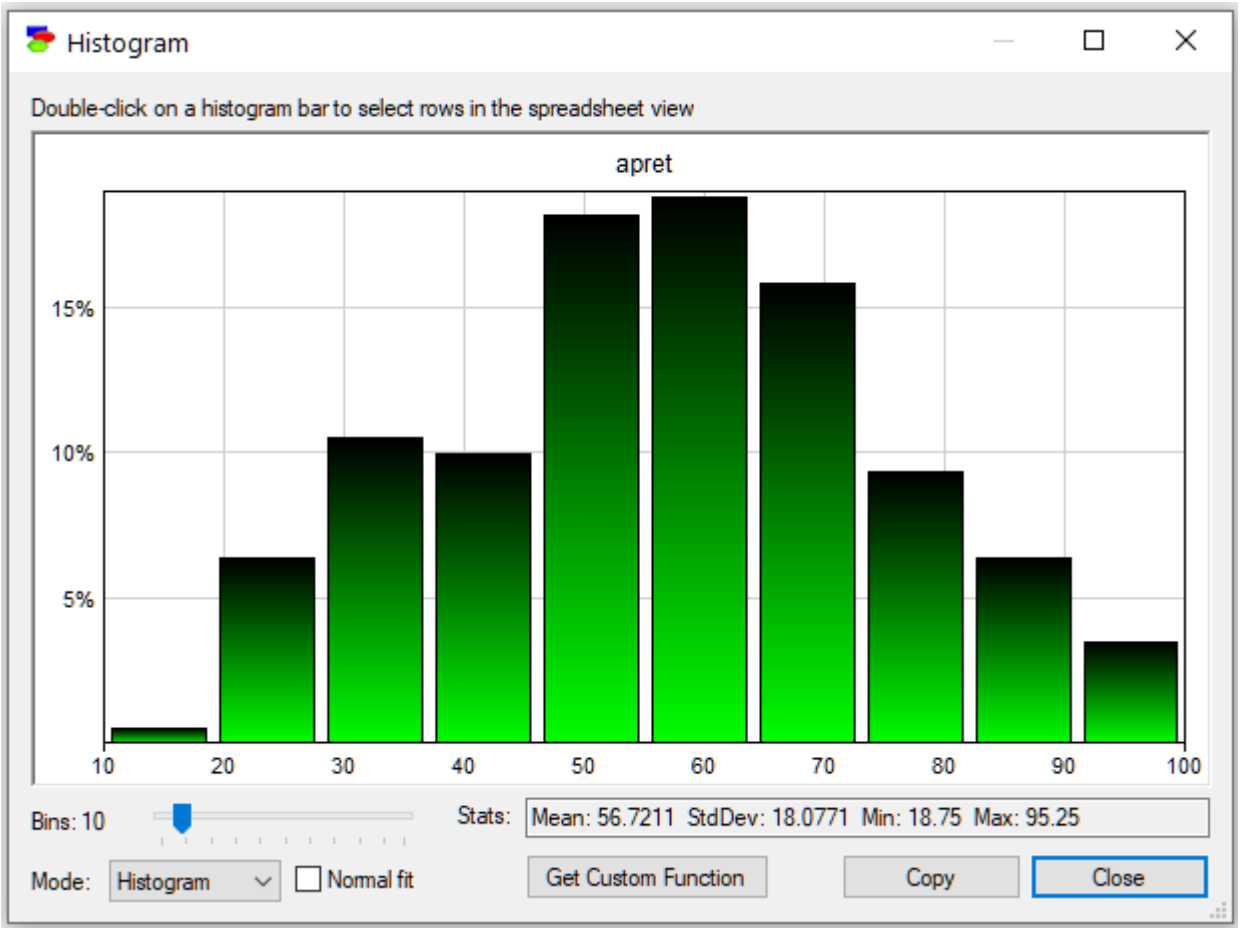
	apret	strat	pacc	rejr	spend	salar	top10	tstsc
apret	-							
strat	-0.458311	-						
pacc	-0.302834	0.131858	-					
rejr	0.514958	-0.283617	-0.0715207	-				
spend	0.601231	-0.581755	-0.28673	0.638544	-			
salar	0.635852	-0.347673	-0.37524	0.606777	0.711838	-		
top10	0.642464	-0.247857	-0.207505	0.643163	0.675656	0.637648	-	
tstsc	0.782183	-0.465226	-0.164223	0.628601	0.71491	0.715472	0.798807	-

It is now convenient to notice that the variables with the highest correlation with *apret* are *tstsc* (78.2%) and *top10* (64.2%).

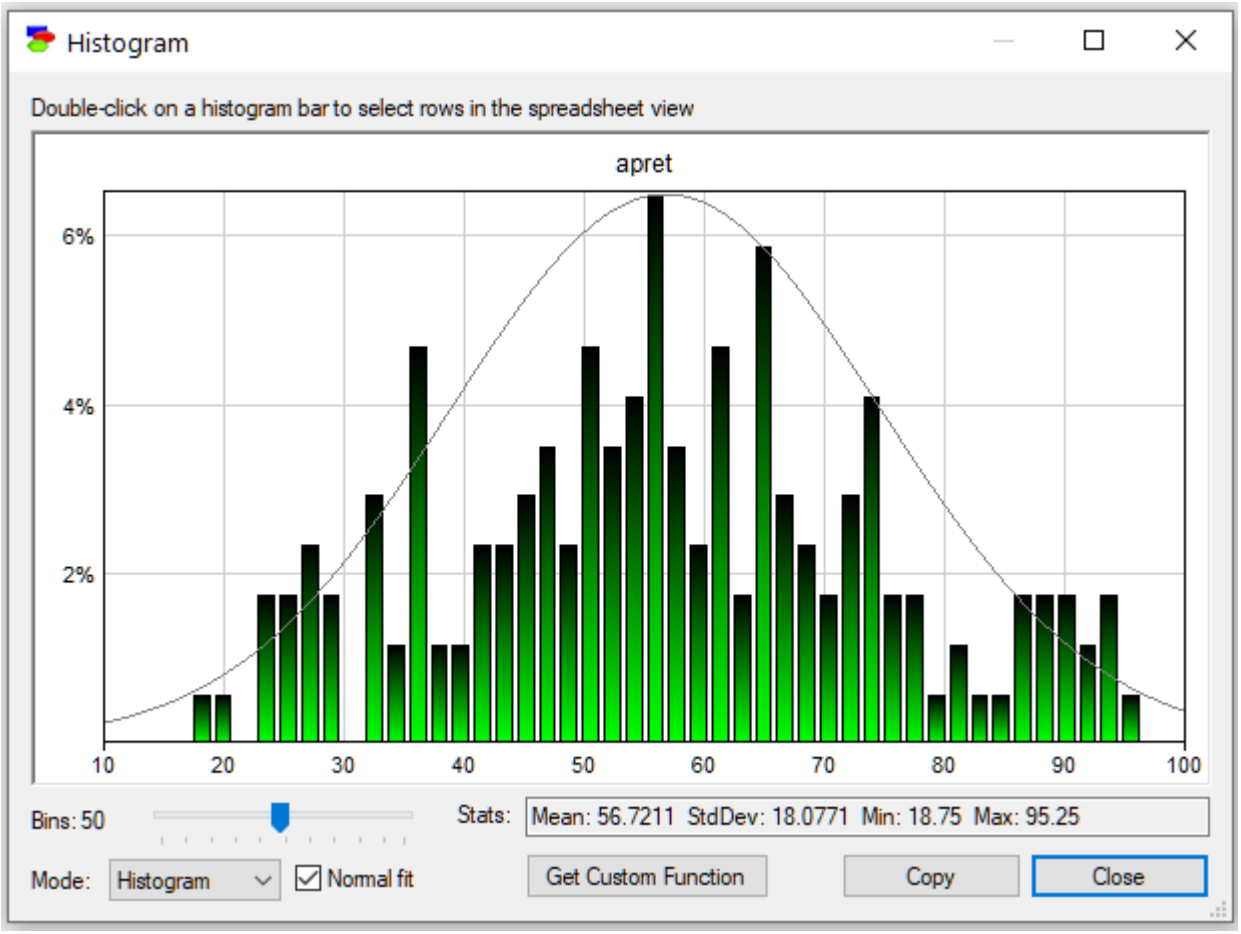
In both, the *Basic statistics* and *Correlation matrix* tabs, scrolling the mouse wheel forward and backward with *CTRL* key pressed zooms in and out, respectively. Also, in both tabs, the *Copy* button places the contents of the grid (along with the headers) on the clipboard. Selecting parts of the grid and using the context menu to copy the selection will never copy the headers. The copied text can be pasted to other text editors.

Histogram

To see the distribution of the values in a column on a histogram graph select *Data-Histogram* after selecting a column or, at least, placing the cursor inside one of the columns.



GeNIe displays a handful of parameters of the distribution (*Mean*, *StdDev*, *Min* and *Max*). It is well known that the shape of a histogram depends strongly on the number of bins selected. GeNIe allows you to change the number of bins interactively (the Bins slides in the lower-left corner). The following image shows the histogram of the same variable (*apret*) when the number of bins is 50 instead of the default 10 on the previous picture:



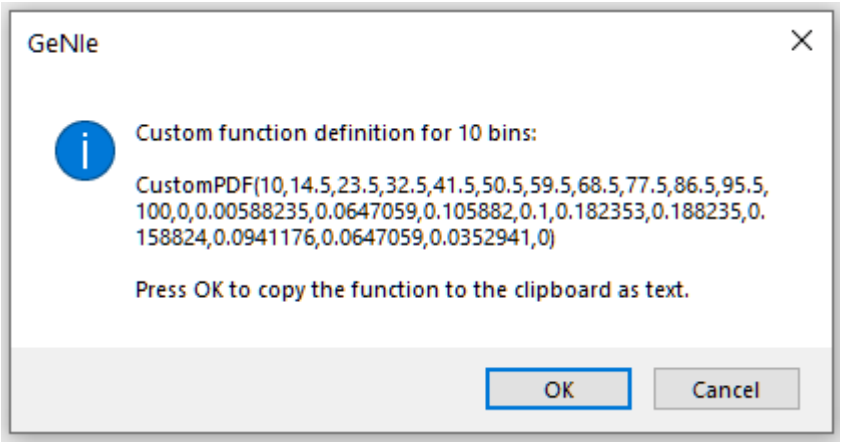
The histogram shows one more feature: Normal fit to the data, invoked by the *Normal fit* check box.

You can double-click a bar on the histogram to select all data rows that correspond to this bar. Histogram is always drawn of the selected rows (if no rows are selected, the histogram is of all rows). You can drill down the data by double-clicking on bars and then displaying histograms of the data that correspond to the selected bars.

You can copy the image of the histogram by clicking on the *Copy* button or right-clicking on it and selecting *Copy* from the pop-up menu that shows. To paste it into an external program as an image (bitmap or picture format, explained in the *Graph view* section), please use *Paste* or *Paste Special*. *Paste*'s output is a text listing bin boundaries and counts. For the first histogram shown above, pasting will yield the following text:

```
apret
10      19      1
19      28      11
28      37      18
37      46      17
46      55      31
55      64      32
64      73      27
73      82      16
82      91      11
91      100     6
```

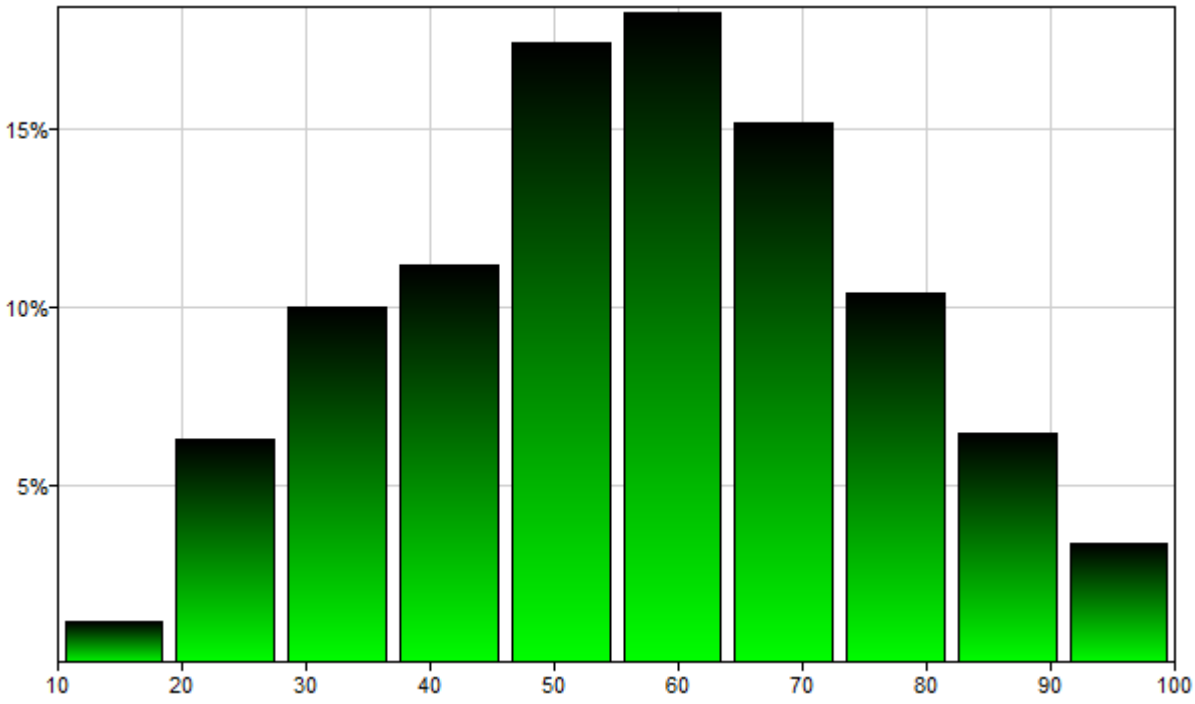
Get Custom Function button allows for learning a continuous probability distribution from the data describing the variable. For the first histogram above (with 10 bins), *Get Custom Function* opens the following dialog



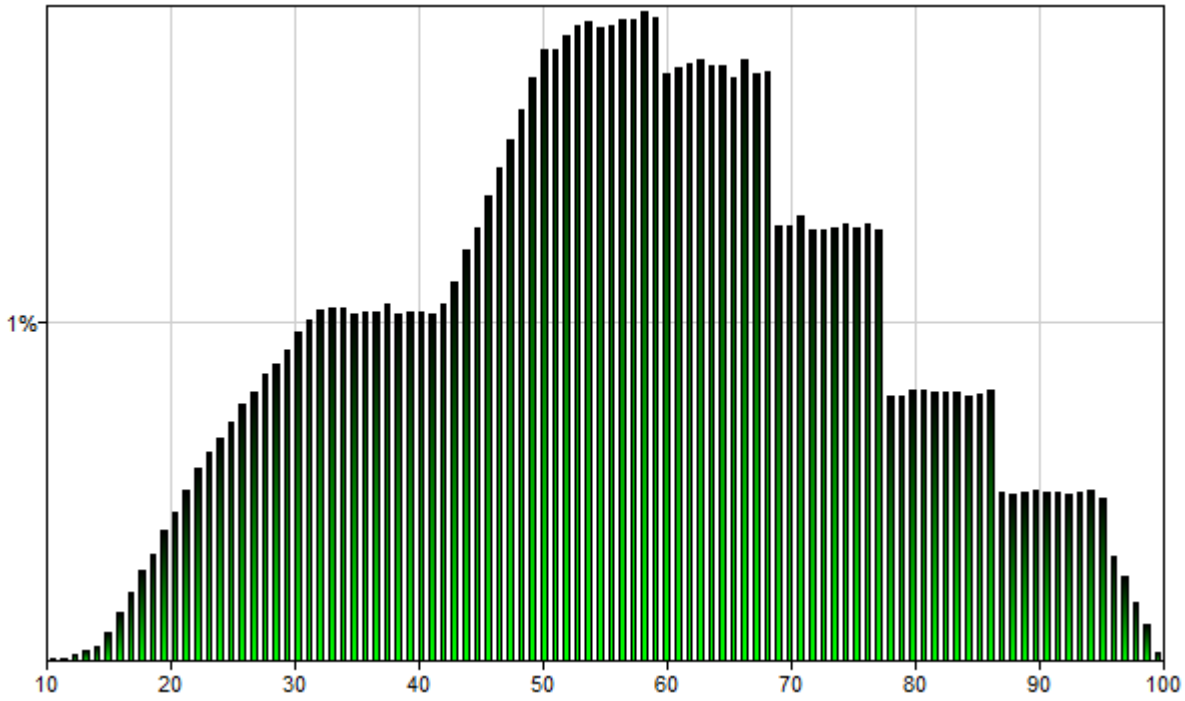
Pressing *OK* puts the following text on the Clipboard:

```
CustomPDF(10,14.5,23.5,32.5,41.5,50.5,59.5,68.5,77.5,86.5,95.5,100,
0,0.00588235,0.0647059,0.105882,0.1,0.182353,0.188235,0.158824,0.0941176,0.0647059,0.0352941,0)
```

The text describes a custom PDF function that can be subsequently used in the definition of a continuous node. The `CustomPDF` function's first argument is the number of intervals, followed by two lists, the breakpoints and their values. A large number of samples from this distribution displayed in a histogram with 10 bins looks as follows.

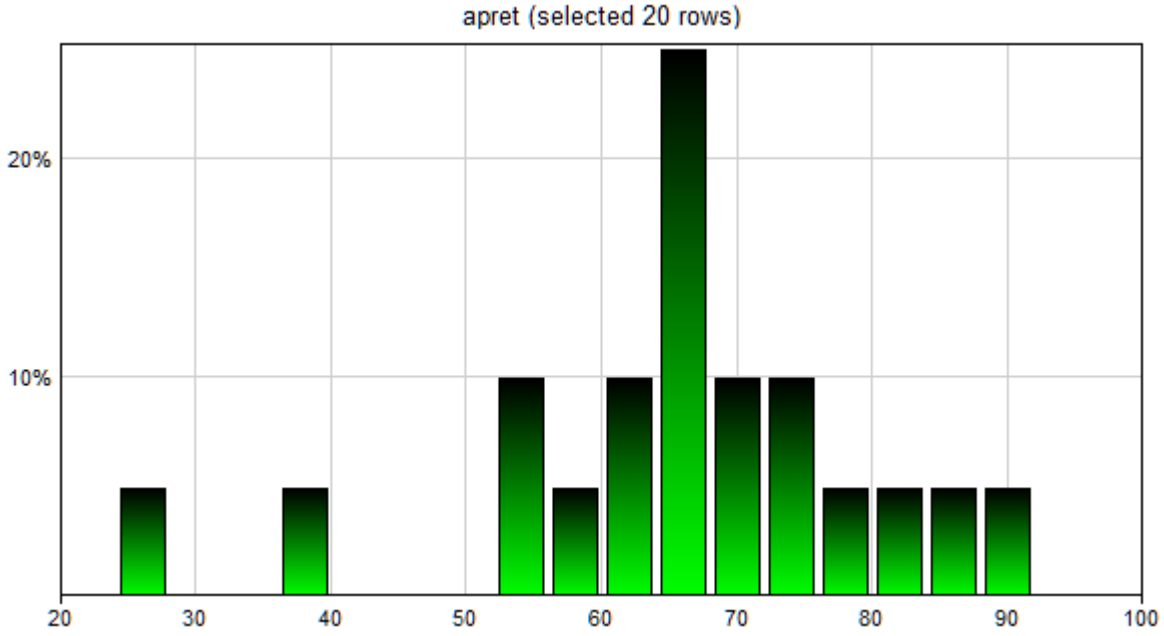


Please note that the shape of this histogram resembles to some degree the original histogram. The same histogram displayed with 100 bins looks as follows.



Please note that the shape of this histogram typically does not show steps but rather smoother transitions.

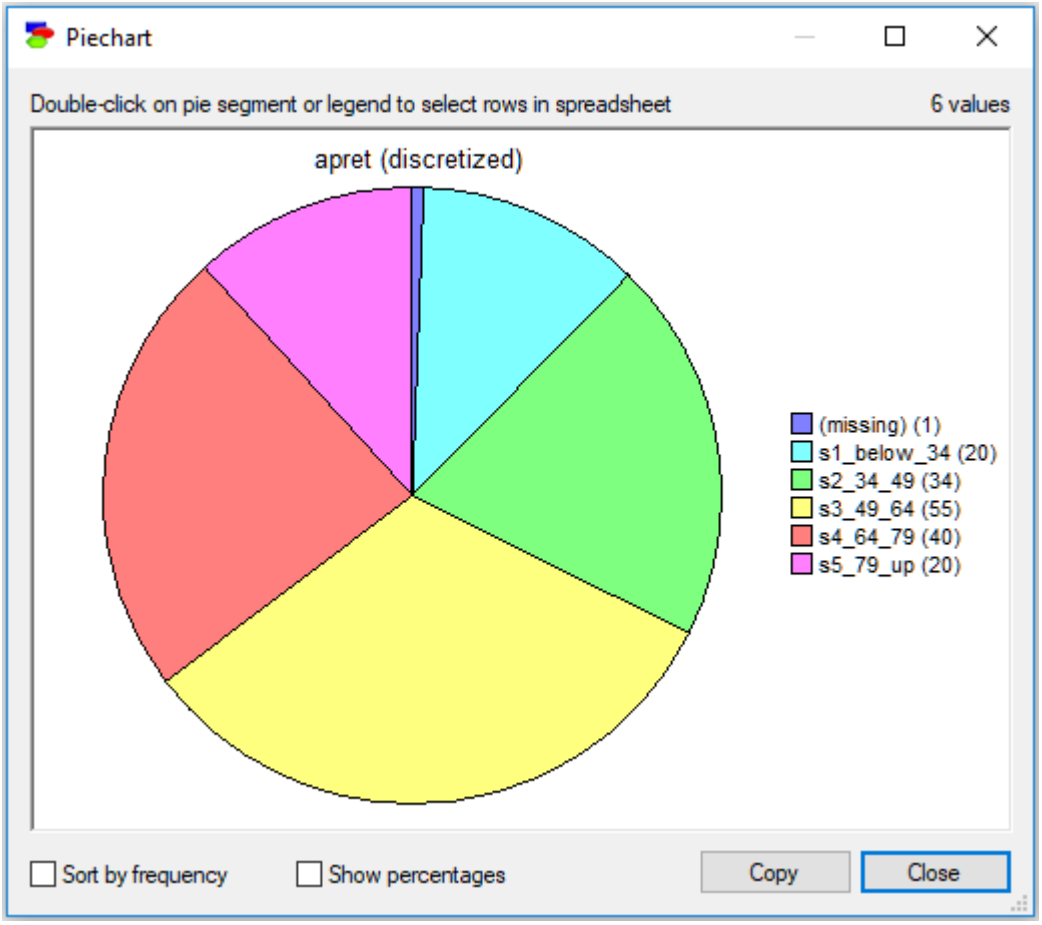
Histogram may be conditional. When a set of records is selected in the *Data Grid* view, the histogram will be based on just those records. Histogram of the variable *apret* in the *retention.txt* data file with the first 20 records selected will look as follows



Please note that this functionality allows for learning conditional probability distributions. It is sufficient to make sure that the records selected are those fulfilling a specified condition. Once the (conditional) histogram has been displayed, we can follow the procedure for learning `CustomPDF` distribution, as described above.

Pie chart

To see the distribution of the values in a discrete (or discretized) column on a pie chart graph, select *Data-Piechart*. This will invoke the following dialog.



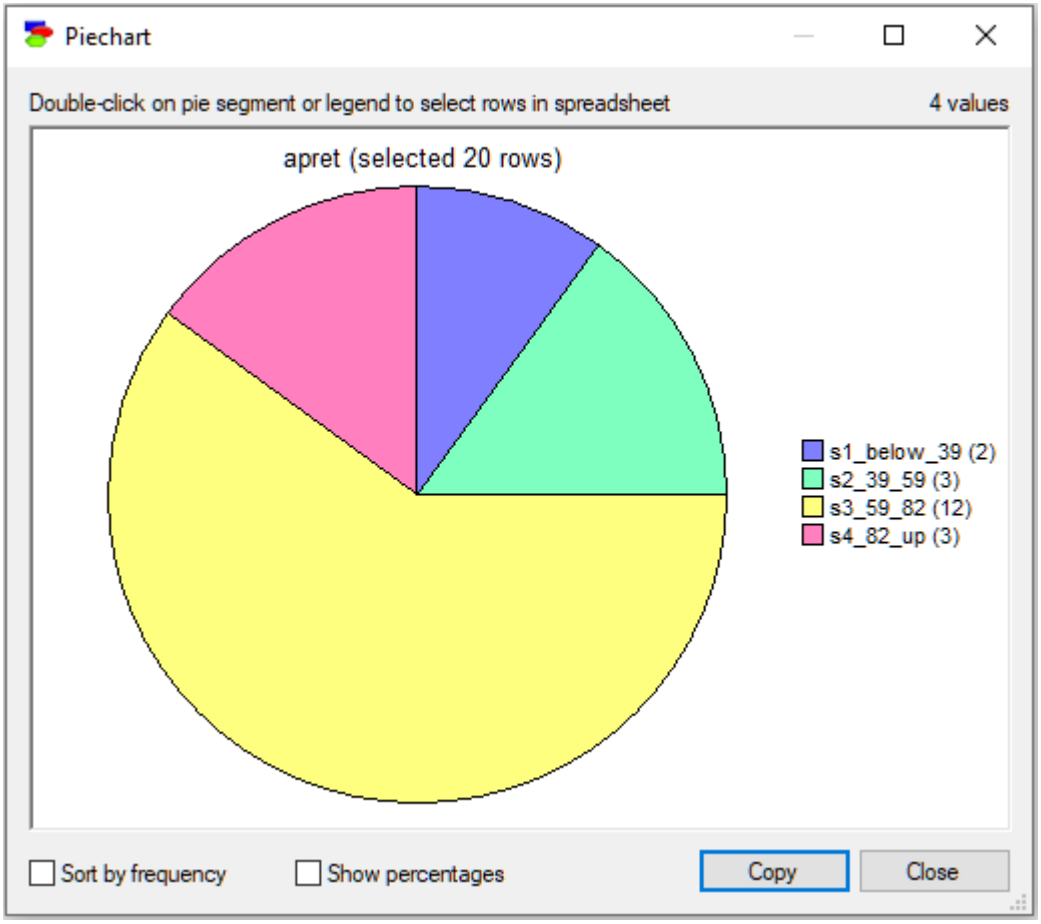
The *Sort by frequency* check box sorts the states in the legend on the right-hand side from the most to the least frequent values. This is convenient when some of the states are very unlikely and hard to identify.

Show percentages check box causes GeNIe to show percentages rather than counts next to the value.

Double-clicking on any segment of the pie chart or any of the small squares in the legend on the right-hand side selects records in the *Data Grid* view that correspond to the value represented by the segment.

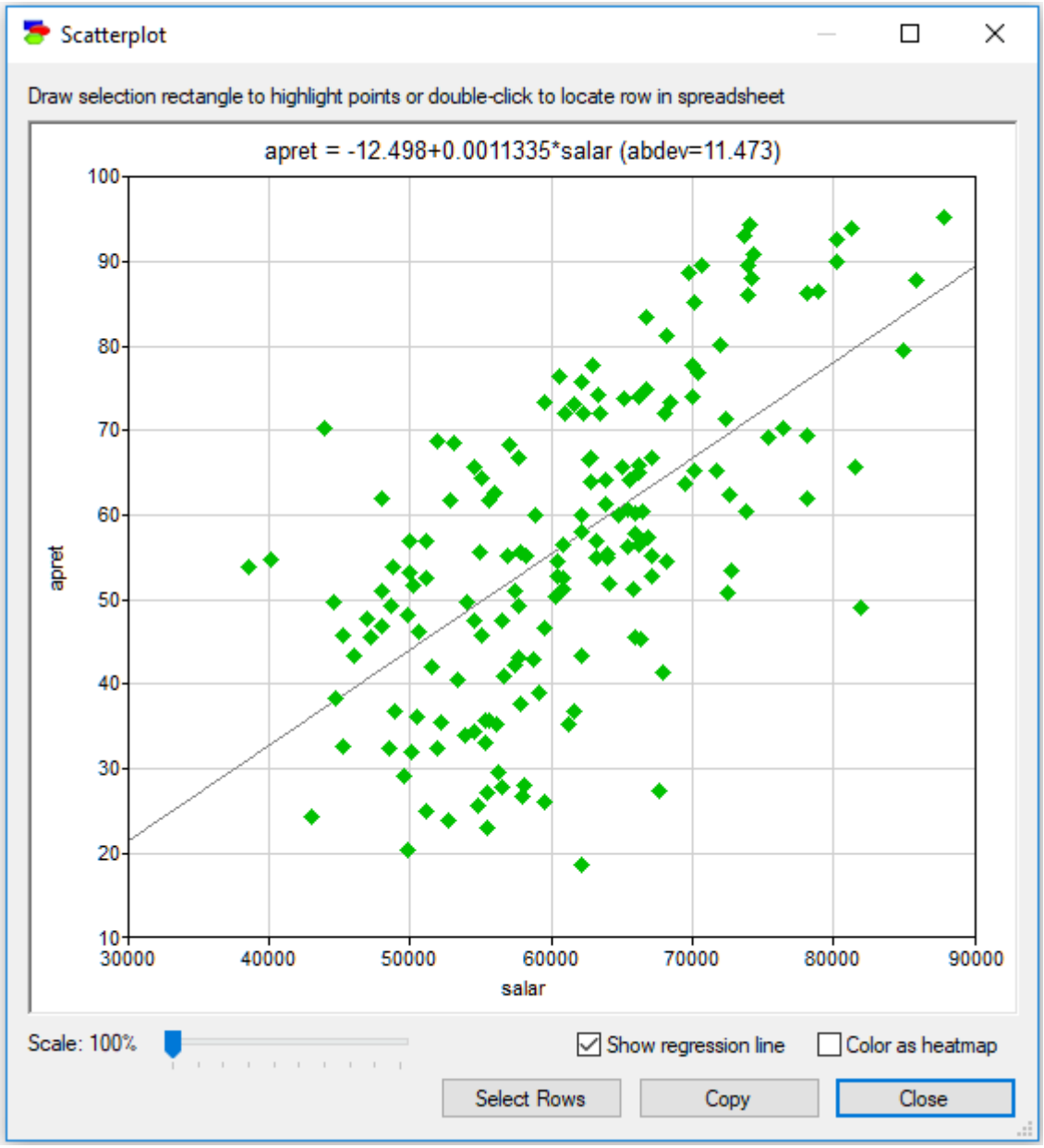
You can copy the image of the pie chart by clicking on the *Copy* button or right-clicking on it and selecting *Copy* from the pop-up menu that shows. To paste it into an external program as an image bitmap or picture format, explained in the *Graph* view section, please use *Paste Special*.

Pie charts may also be conditional. When a set of records is selected in the *Data Grid* view, the pie chart will be based on just those records. The pie chart of the variable *apret* in the *retention_discretized.txt* data file with the first 20 records selected will look as follows



Scatterplot

To see the joint distribution of two variables, select two (numerical) columns (by clicking on their IDs with *CTRL* key pressed) and select *Data-Scatterplot*. This functionality displays a scatterplot of the two variables, with the variable selected first being represented by the x-axis and the variable selected second being represented by the y-axis. Scatterplot between the variables *salar* and *apret* in the *retention.txt* data file will look as follows:



Color as heatmap option is useful in case of dense scatterplots and shows the density of the points on the plot.

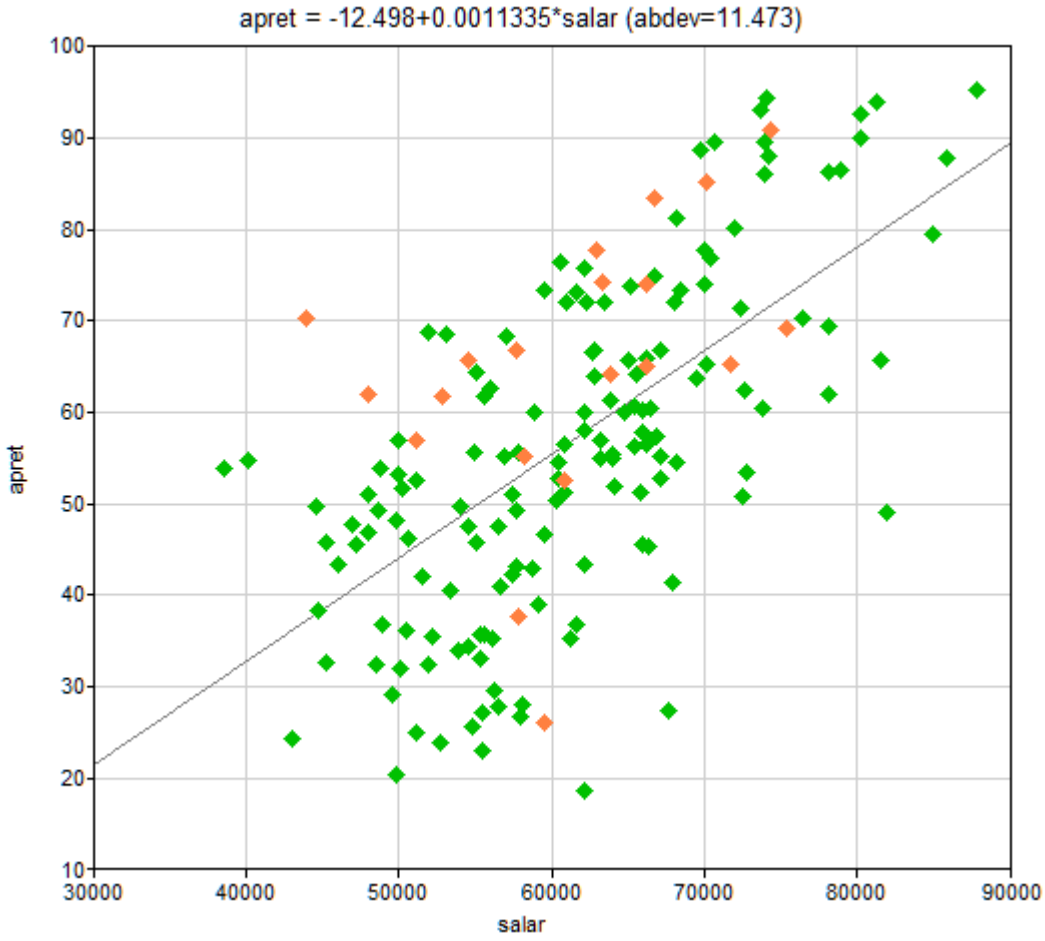
Show regression line is useful in case of linear relationships between the variables. When this option is checked, GeNIe runs a linear regression of the variable on the y-axis against the variable on the x-axis. The equation of the line fitting the data points is displayed above the scatterplot along with the mean absolute deviation (*abdev*) in y of the points from the fitted line.

There is a close connection between the *Scatterplot* and the *Data Grid*. The *Scatterplot* displays rows that were selected before invoking it in orange color. Selecting any points on the *Scatterplot* and clicking on the *Select Rows* button exits the *Scatterplot* window and selects the rows corresponding to the selected points. Double-clicking on a point selects the corresponding data row in the *Data Grid*. This functionality is very convenient, for example, in case of identifying and removing outliers.

Scale allows for focusing on parts of the graph. Scaling the *Scatterplot* (*Scale* slider) helps in distinguishing points that appear close on the plot but in reality are distinct.

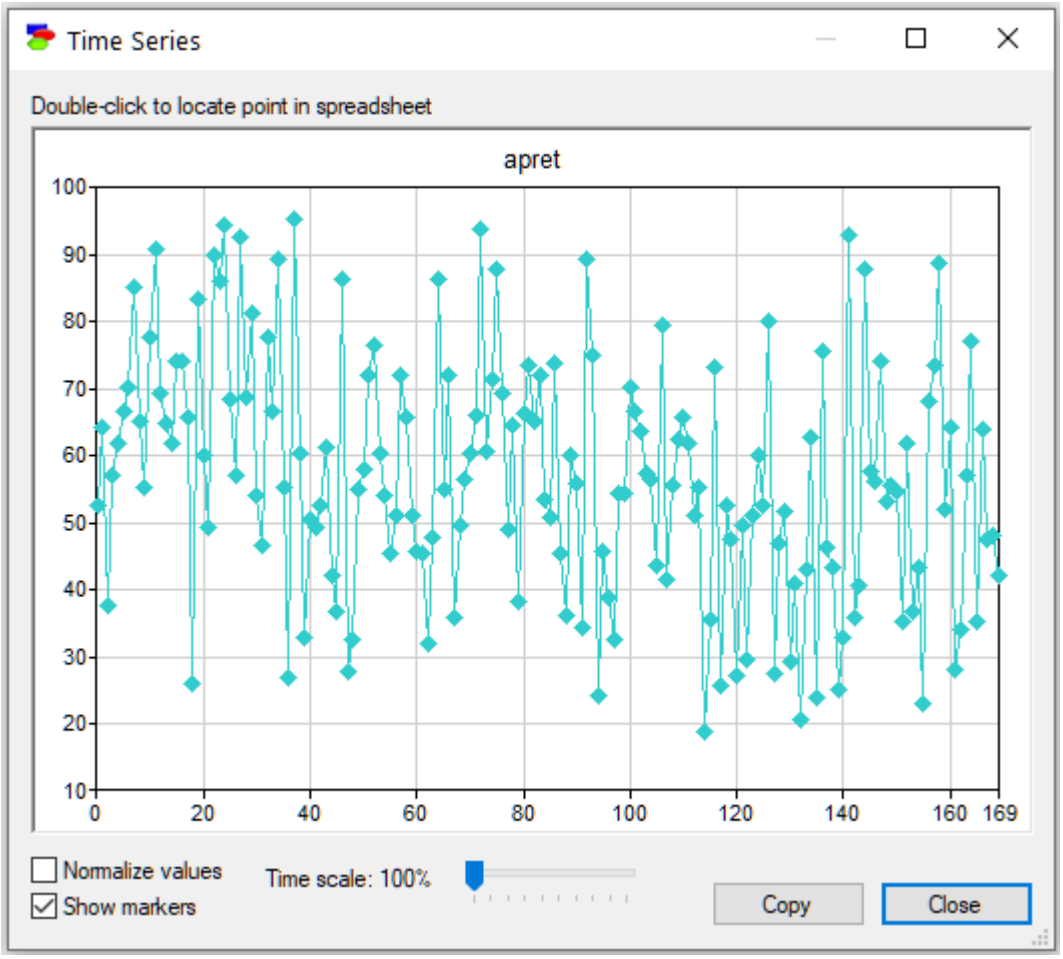
You can copy the image of the scatterplot by clicking on the *Copy* button or right-clicking on it and selecting *Copy* from the pop-up menu. The image can be then pasted into an external program, *such as MS Word*, in bitmap or picture format, explained in the *Graph* view section.

Scatterplots may be conditional. When a set of records is selected in the *Data Grid* view, the scatterplot will display the selected records in orange. Scatterplot between the variables *salar* and *apret* in the *retention.txt* data file with the first 20 records selected will look as follows:



Time Series

Finally, some data are time series and are best viewed in the original order. To see a plot of a single variable as a time series, select *Data*-*Time Series*.

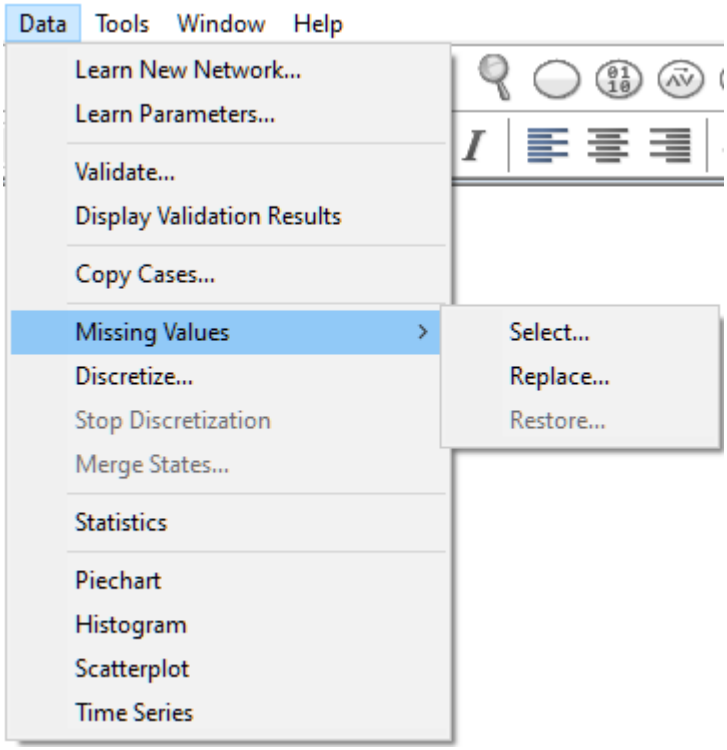


To show markers corresponding to the data points, select the *Show markers* check box. To normalize the values in the data so as the highest and lowest values take the highest and lowest points on the plot, select *Normalize values* check box. Time scale slider allows you to focus on parts of the plot in greater detail. Double click a point to locate the corresponding point in the data spreadsheet.

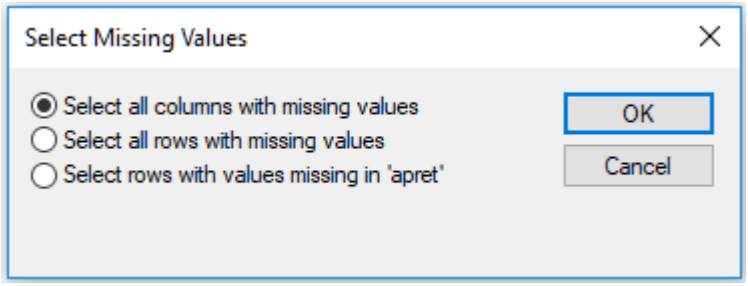
You can copy the image of the time series by clicking on the *Copy* button or right-clicking on it and selecting *Copy* from the pop-up menu that shows. To paste it into an external program as a bitmap image or picture format, explained in the *Graph* view section, please use *Paste Special*.

Missing Values

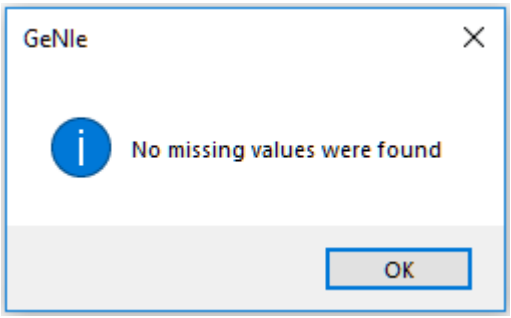
Missing values show as empty cells. To select rows that contain missing values (for instance, for deletion) select *Data-Missing Values-Select...*



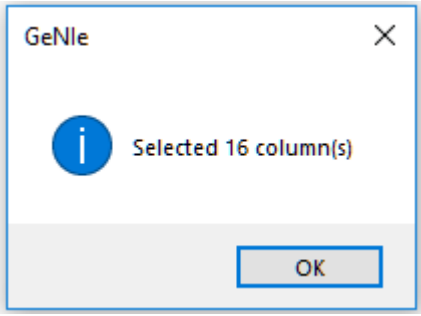
You will be given three options: (1) *Select all columns with missing values*, (2) *Select all rows with missing values*, and (3) *Select rows with values missing only in 'apret'* (the currently selected column):



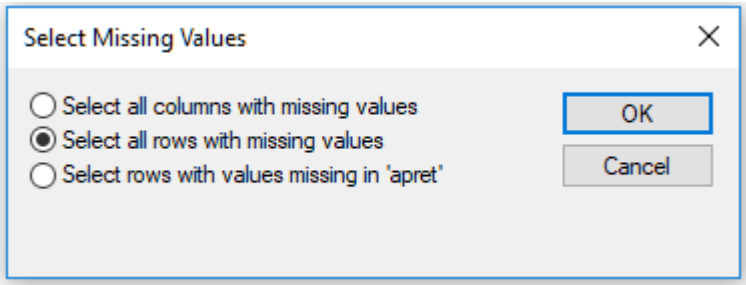
Selection of columns helps in identifying variables that have missing values. If the data file did not contain any missing values, GeNIe will inform you about that:



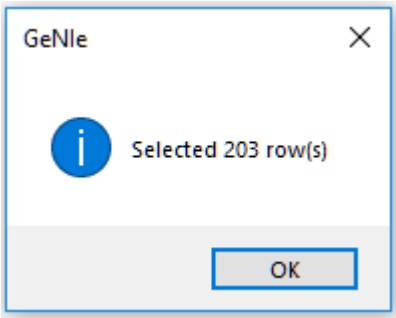
Otherwise, GeNIe will confirm how many columns it selected:



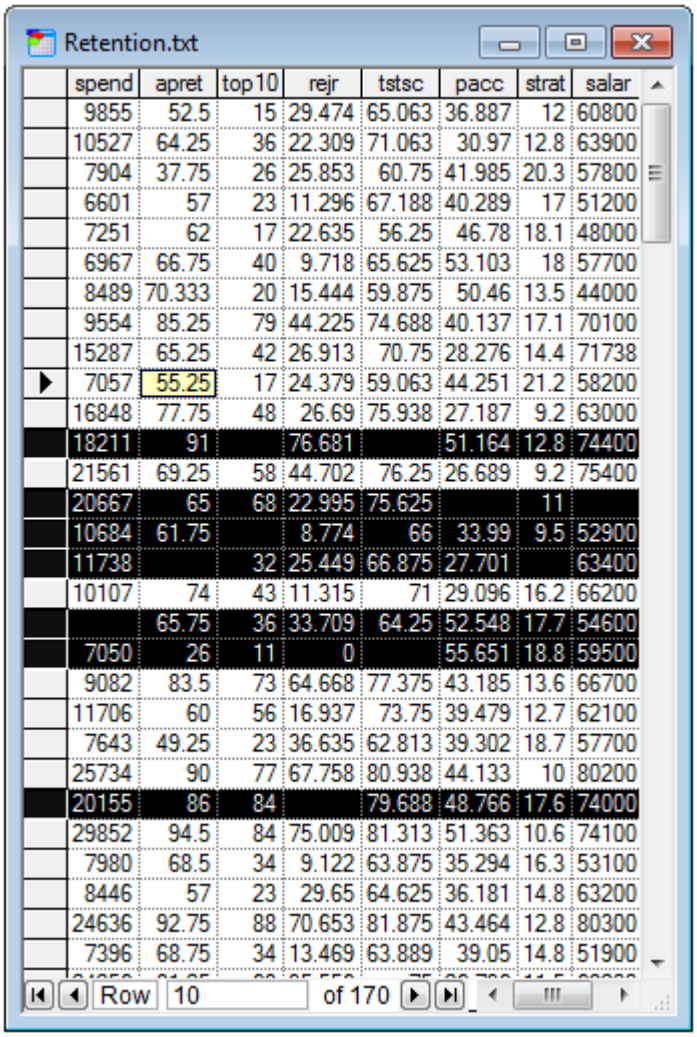
The second and the third choice allow you for selecting data records with missing values in any cells and in the currently selected variable respectively. A column in the data grid is considered selected if the cursor is placed in any of its cells.



If the data file did not contain any missing values, GeNIe will inform you about that. Otherwise, GeNIe will confirm how many rows it selected.



Selected rows will be highlighted in the *Data Grid View*:



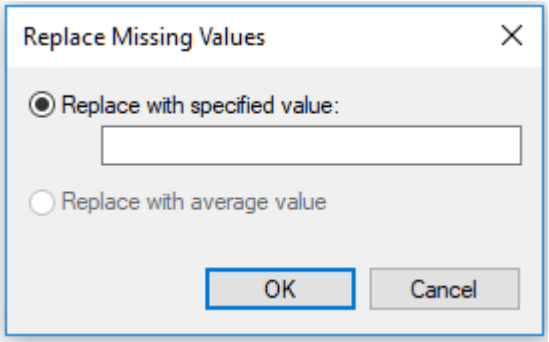
Retention.txt

spend	apret	top10	rejr	tstsc	pacc	strat	salar
9855	52.5	15	29.474	65.063	36.887	12	60800
10527	64.25	36	22.309	71.063	30.97	12.8	63900
7904	37.75	26	25.853	60.75	41.985	20.3	57800
6601	57	23	11.296	67.188	40.289	17	51200
7251	62	17	22.635	56.25	46.78	18.1	48000
6967	66.75	40	9.718	65.625	53.103	18	57700
8489	70.333	20	15.444	59.875	50.46	13.5	44000
9554	85.25	79	44.225	74.688	40.137	17.1	70100
15287	65.25	42	26.913	70.75	28.276	14.4	71738
7057	55.25	17	24.379	59.063	44.251	21.2	58200
16848	77.75	48	26.69	75.938	27.187	9.2	63000
18211	91		76.681		51.164	12.8	74400
21561	69.25	58	44.702	76.25	26.689	9.2	75400
20667	65	68	22.995	75.625		11	
10684	61.75		8.774	66	33.99	9.5	52900
11738		32	25.449	66.875	27.701		63400
10107	74	43	11.315	71	29.096	16.2	66200
	65.75	36	33.709	64.25	52.548	17.7	54600
7050	26	11	0		55.651	18.8	59500
9082	83.5	73	64.668	77.375	43.185	13.6	66700
11706	60	56	16.937	73.75	39.479	12.7	62100
7643	49.25	23	36.635	62.813	39.302	18.7	57700
25734	90	77	67.758	80.938	44.133	10	80200
20155	86	84		79.688	48.766	17.6	74000
29852	94.5	84	75.009	81.313	51.363	10.6	74100
7980	68.5	34	9.122	63.875	35.294	16.3	53100
8446	57	23	29.65	64.625	36.181	14.8	63200
24636	92.75	88	70.653	81.875	43.464	12.8	80300
7396	68.75	34	13.469	63.889	39.05	14.8	51900

Row 10 of 170

Very often, selection is the first step to deleting records, which is one way of dealing with missing values in structural learning. This approach works if the number of records with missing values is relatively small.

If the data file contains any missing values, you may choose to replace them with something - this is one way of dealing with missing values. To do that select *Data-Missing Values-Replace...* You will be given a choice to replace (1) with a specific value or (2) with an average of the selected column. The values will be replaced only in the currently selected column.



The replaced values will be distinguished with a red color, like on the screen shot below (the missing values were replaced in column *top10* with the value 9999):

Retention.txt

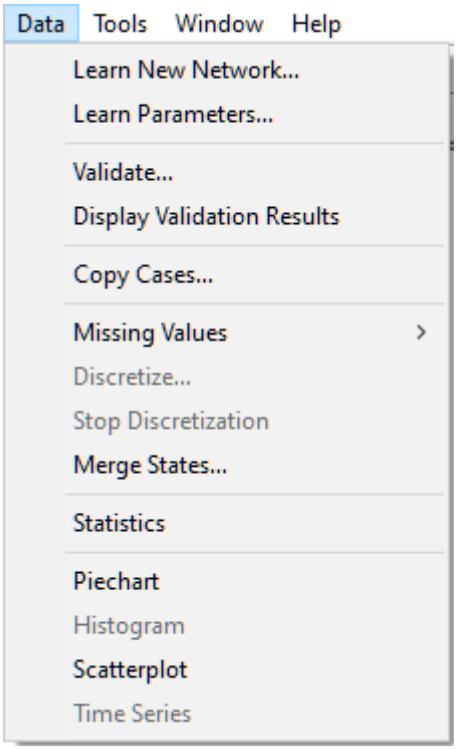
spend	apret	top10	rejr	tstsc	pacc	strat	salar
9855	52.5	15	29.474	65.063	36.887	12	60800
10527	64.25	36	22.309	71.063	30.97	12.8	63900
7904	37.75	26	25.853	60.75	41.985	20.3	57800
6601	57	23	11.296	67.188	40.289	17	51200
7251	62	17	22.635	56.25	46.78	18.1	48000
6967	66.75	40	9.718	65.625	53.103	18	57700
8489	70.333	20	15.444	59.875	50.46	13.5	44000
9554	85.25	79	44.225	74.688	40.137	17.1	70100
15287	65.25	42	26.913	70.75	28.276	14.4	71738
7057	55.25	17	24.379	59.063	44.251	21.2	58200
16848	77.75	48	26.69	75.938	27.187	9.2	63000
18211	91	9999	76.681		51.164	12.8	74400
21561	69.25	58	44.702	76.25	26.689	9.2	75400
20667	65	68	22.995	75.625		11	
10684	61.75	9999	8.774	66	33.99	9.5	52900
11738		32	25.449	66.875	27.701		63400
10107	74	43	11.315	71	29.096	16.2	66200
	65.75	36	33.709	64.25	52.548	17.7	54600
7050	26	11	0		55.651	18.8	59500
9082	83.5	73	64.668	77.375	43.185	13.6	66700
11706	60	56	16.937	73.75	39.479	12.7	62100
7643	49.25	23	36.635	62.813	39.302	18.7	57700
25734	90	77	67.758	80.938	44.133	10	80200
20155	86	84		79.688	48.766	17.6	74000
29852	94.5	84	75.009	81.313	51.363	10.6	74100
7980	68.5	34	9.122	63.875	35.294	16.3	53100
8446	57	23	29.65	64.625	36.181	14.8	63200
24636	92.75	88	70.653	81.875	43.464	12.8	80300
7396	68.75	34	13.469	63.889	39.05	14.8	51900

Row 1 of 170

You can rollback all the replace actions on any of the columns by selecting the column itself (putting the cursor in one of its cells) and selecting *Data-Missing Values-Restore...* The inserted values will be removed from the data grid.

Discretization

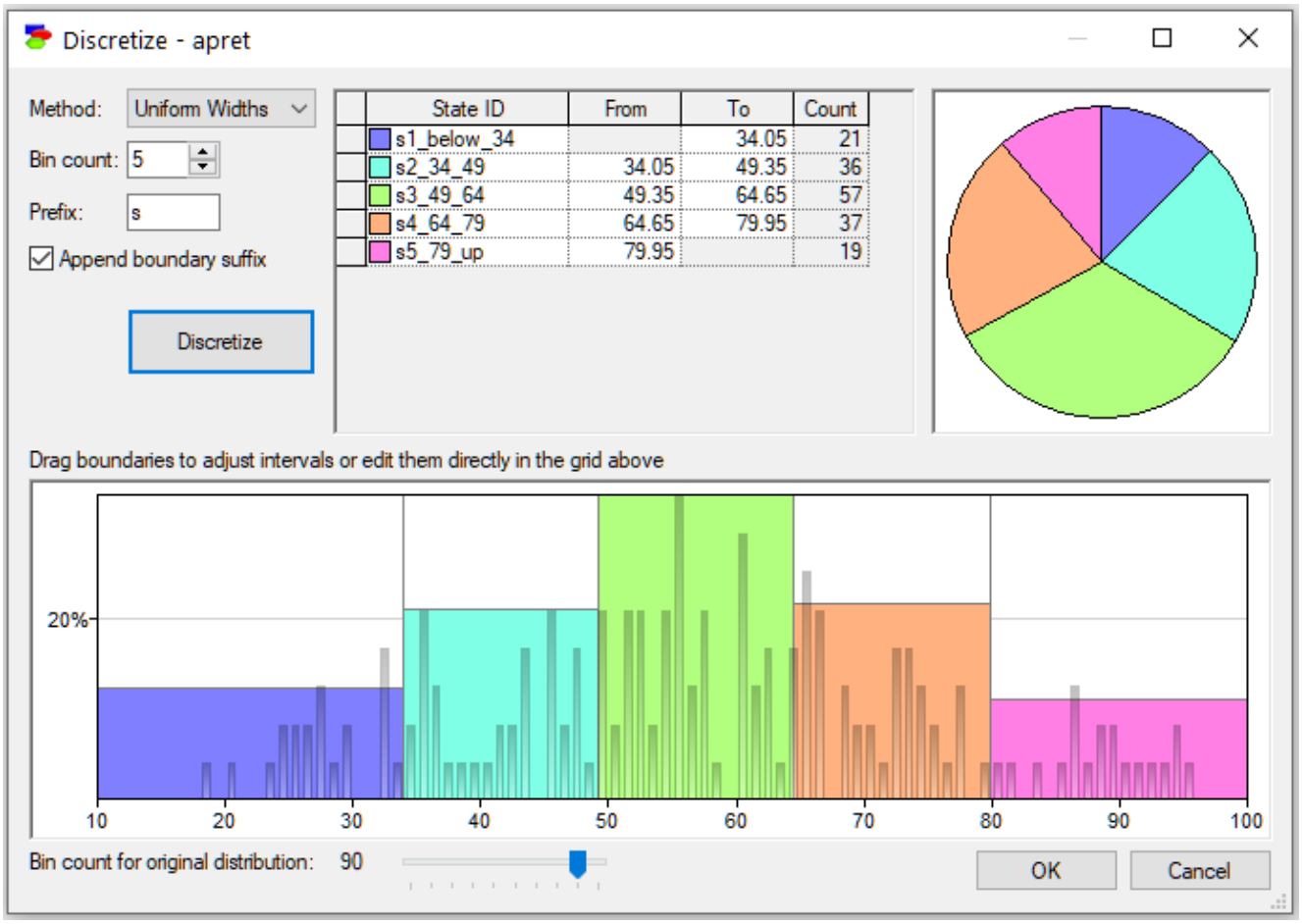
GeNIe offers a powerful interface for interactive discretization of continuous variables. To invoke the discretization interface, select *Data-Discretize...*



The interface gives you a choice of discretization method (*Method*), the number of discretization intervals (*Bin count*), and a *Prefix* for the automatically assigned labels for the intervals. There are three discretization methods implemented in GeNIe now: *Uniform Widths*, which makes the widths of the discretization intervals the same, *Uniform Counts*, which makes the number of values in each of the discretization bins the same, and *Hierarchical*, which is an unsupervised discretization method related to clustering. We do do not have the literature reference handy but here is a sketch of the algorithm implemented in GeNIe:

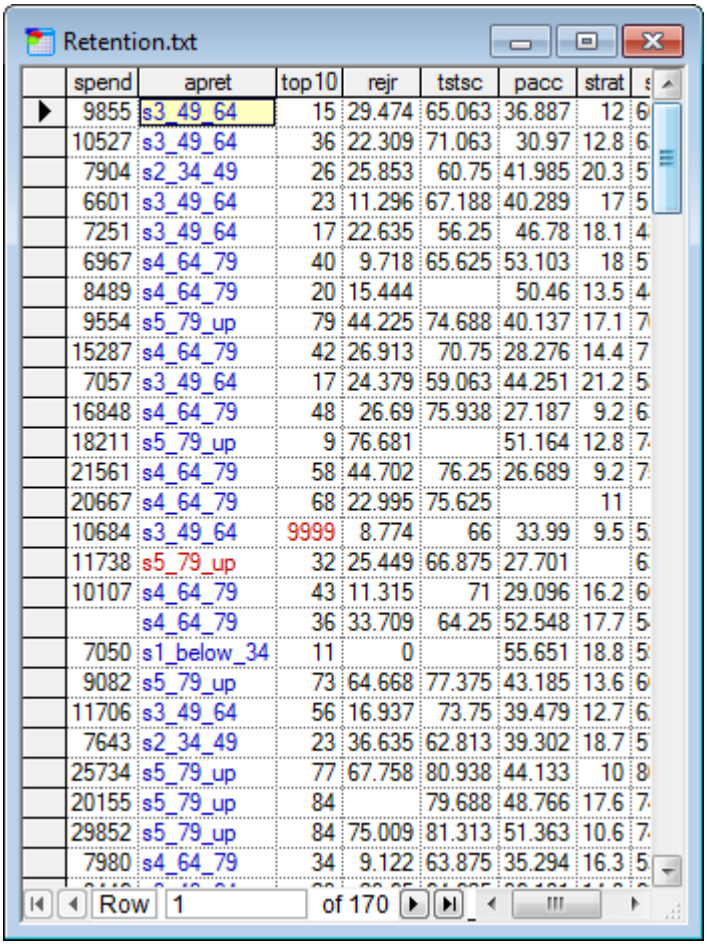
```
Input: N=# of records, K=# of desired bins
1. Let k denote the running number of bins, initialized to k=N (each record starts in its own cluster)
2. If k=K quit, else set k=k-1 by combining the two bins whose mean value has the smallest separation
3. Repeat 2
```

Discretization takes place once you press the *Discretize* button. The interface displays the distribution of records among the new intervals (as a pie chart) and a probability mass function with the histogram of the original continuous data in the background. The colors of the intervals correspond to the colors in the pie chart and in the histogram. Please note that, similarly to the histogram interface, you can modify the bin count for the histogram. You can modify the discretization boundaries by entering the new values directly into the table or by dragging the interval boundaries on the data histogram.



The pictorial representation of the discretization can be copied and later pasted into another application for the purpose of documentation or reporting. To copy the pictorial representation of the discretization, right-click on it, select *Copy*, and subsequently choose *Paste Special* in the destination application.

Once you accept the changes (by clicking on the *OK* button) the discretized column will be shown in blue font in the *Data Grid View*. You can reverse discretization at any time by selecting *Data-Stop Discretization* option from the main menu.

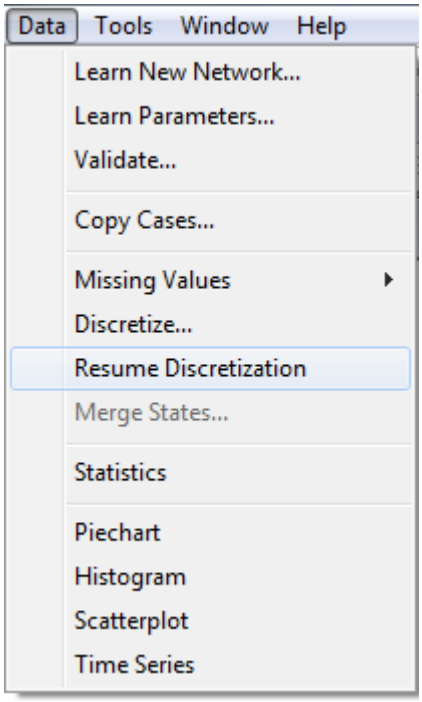


Retention.txt

spend	apret	top10	rejr	tstsc	pacc	strat	s
9855	s3_49_64	15	29.474	65.063	36.887	12	6
10527	s3_49_64	36	22.309	71.063	30.97	12.8	6
7904	s2_34_49	26	25.853	60.75	41.985	20.3	5
6601	s3_49_64	23	11.296	67.188	40.289	17	5
7251	s3_49_64	17	22.635	56.25	46.78	18.1	4
6967	s4_64_79	40	9.718	65.625	53.103	18	5
8489	s4_64_79	20	15.444		50.46	13.5	4
9554	s5_79_up	79	44.225	74.688	40.137	17.1	7
15287	s4_64_79	42	26.913	70.75	28.276	14.4	7
7057	s3_49_64	17	24.379	59.063	44.251	21.2	5
16848	s4_64_79	48	26.69	75.938	27.187	9.2	6
18211	s5_79_up	9	76.681		51.164	12.8	7
21561	s4_64_79	58	44.702	76.25	26.689	9.2	7
20667	s4_64_79	68	22.995	75.625		11	
10684	s3_49_64	9999	8.774	66	33.99	9.5	5
11738	s5_79_up	32	25.449	66.875	27.701		6
10107	s4_64_79	43	11.315	71	29.096	16.2	6
	s4_64_79	36	33.709	64.25	52.548	17.7	5
7050	s1_below_34	11	0		55.651	18.8	5
9082	s5_79_up	73	64.668	77.375	43.185	13.6	6
11706	s3_49_64	56	16.937	73.75	39.479	12.7	6
7643	s2_34_49	23	36.635	62.813	39.302	18.7	5
25734	s5_79_up	77	67.758	80.938	44.133	10	8
20155	s5_79_up	84		79.688	48.766	17.6	7
29852	s5_79_up	84	75.009	81.313	51.363	10.6	7
7980	s4_64_79	34	9.122	63.875	35.294	16.3	5

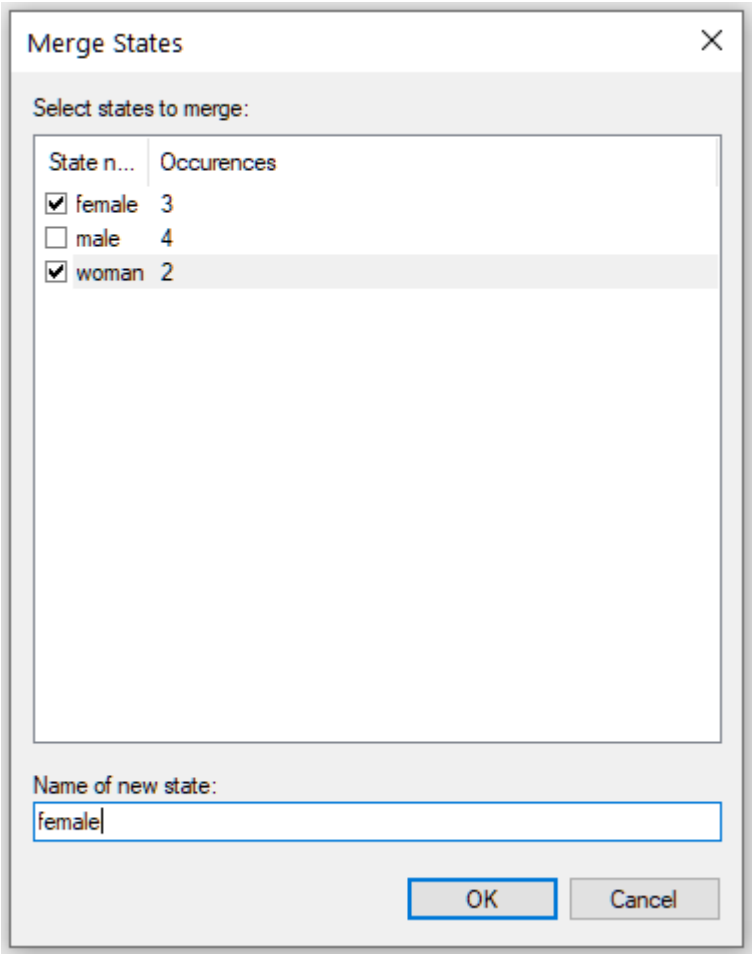
Row 1 of 170

You can resume a stopped discretization at any time by selecting *Data-Resume Discretization*. The values will be restored to the last successful discretization.



Merging States

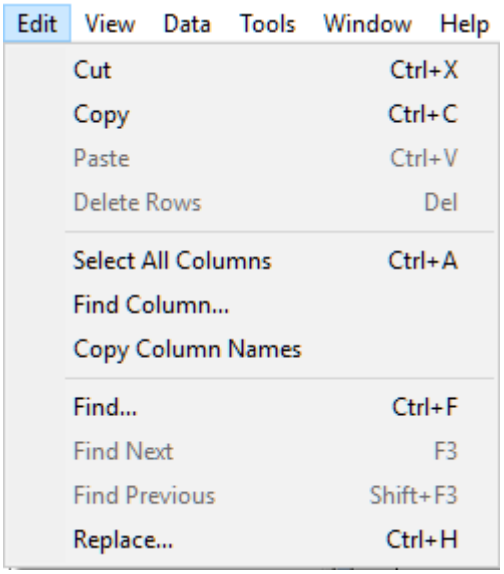
Sometime, through an error in the data collection or encoding, two or more states may denote the same value. For example, *female* and *woman* may all refer to the same value. GeNIe allows to merge such states into one through the *Merge States...* functionality from the *Data Menu*. To merge states of a variable, select the column that represents it and select *Data-Merge States...*



Select the states that you want to merge together and provide the name for the resulting state. You can see that GeNIe gives you information about the number of occurrences of each of the states of the selected variable. The effect of this operation will be that all five states (*female* and *woman*) will be changed into *female*. The *Merge States* command can be viewed a convenient shortcut for a series of *Replace All* commands.

Edit Menu for Data Spreadsheets

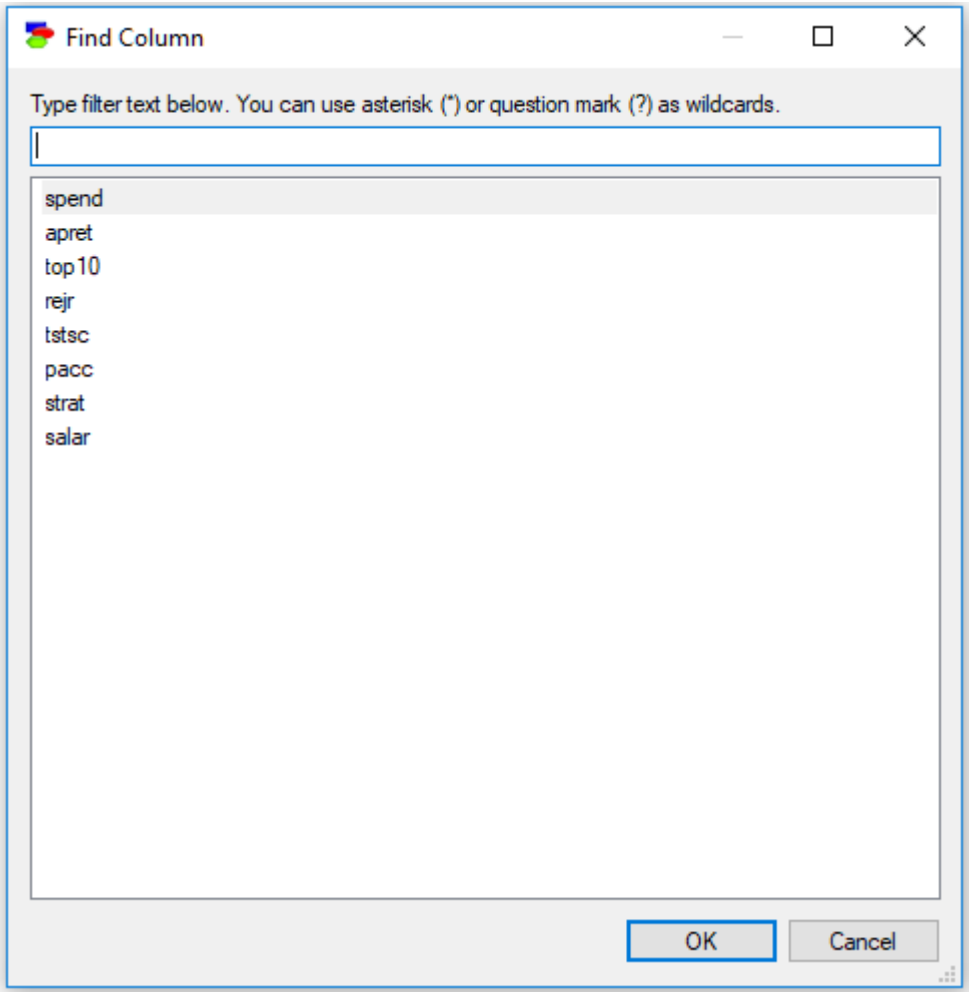
Edit Menu has additional choices for data spreadsheets:



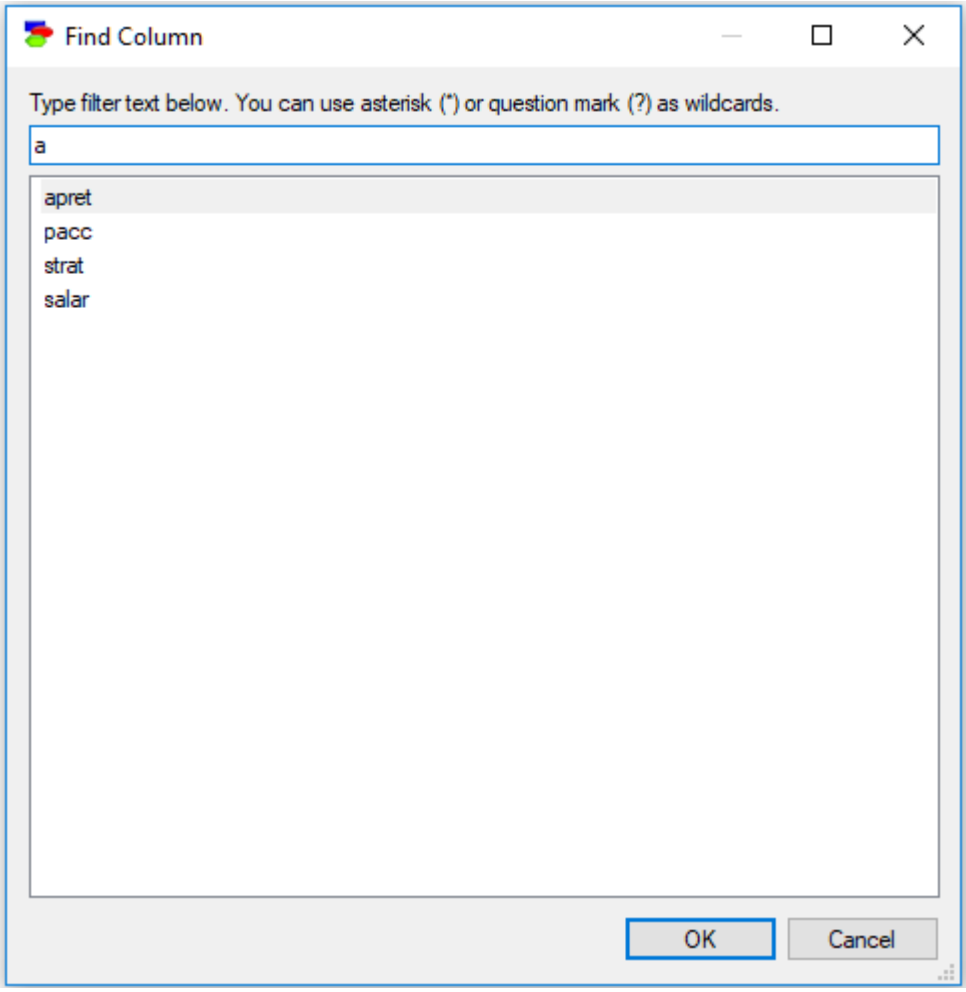
While the commands like *Cut*, *Paste*, *Find*, and *Replace* have obvious meanings for data spreadsheets, the three additional commands deserve a brief explanation.

Select All Columns selects all columns in the data spreadsheet. This is useful as a preparation for the *Copy Column Names*, which copies the names of the selected columns. This functionality is useful in creating a list of columns to be pasted into a text editor or pasting them in the *Learn New Network* dialog.

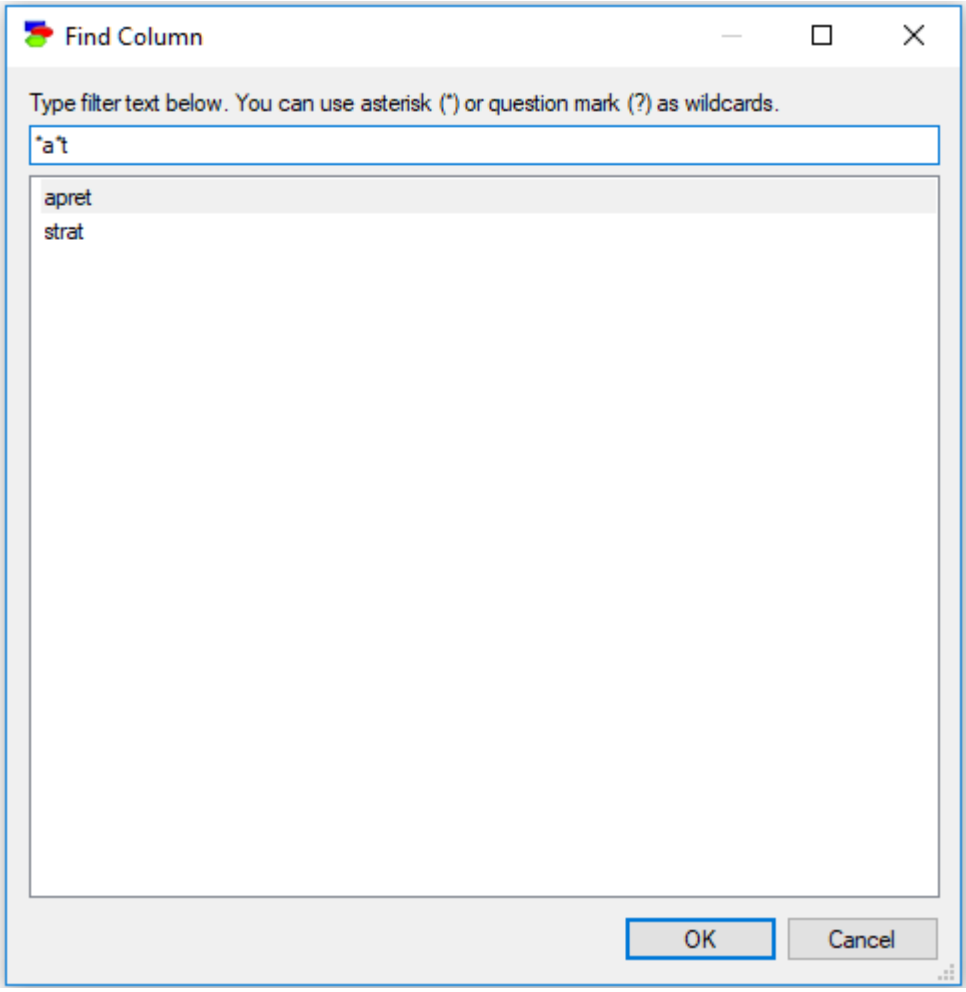
Find Column is useful when the data file has many variables/columns. When their number is large, scrolling the screen sideways to find a desired column may be challenging.



Typing anything in the filter field causes the dialog to limit the names of columns to those that match the filter. For example, typing an a into the above dialog will lead to the following selection:



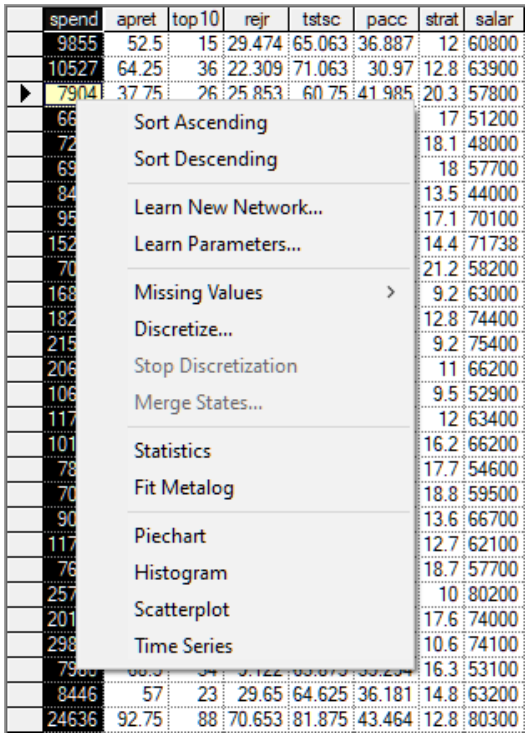
Letter case is ignored, so a lower case letter (e.g., "a") is equivalent to an upper case letter (i.e., "A"). In addition to regular characters, the filter field interprets wildcard characters, such as an asterisk (*) or a question mark (?) similarly to Windows. Typing "*a*t" will select only those column names that have an "a" character and end with a "t":



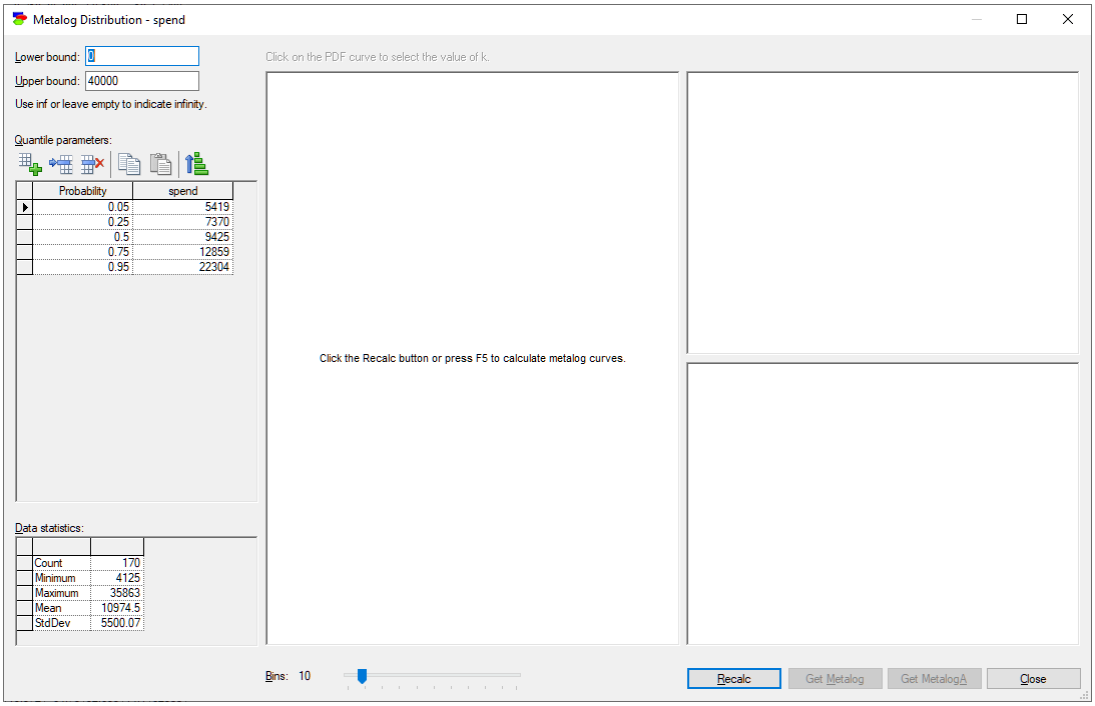
Double-clicking on any of the column names or selecting a column name and pressing OK will select that column in the data spreadsheet.

6.5.5 Fitting metalog distribution to data

One added functionality of the data spreadsheet interface is learning the (marginal or conditional) probability distributions that have generated the data. We have described the simple functionality of learning the `CustomPDF` distribution through the histogram interface in the *Cleaning data* section. There is an even more powerful functionality for learning the metalog distribution from data. To invoke this functionality, please select *Fit Metalog* from a column context menu:



This will invoke the following dialog

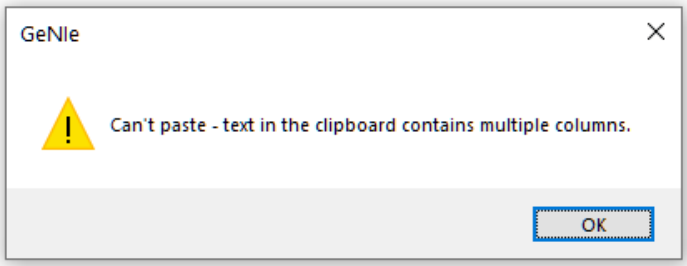


Lower and upper bounds are extracted from the data, although they are somewhat generous, rounded down and up respectively. Both can be modified by the user. To make the bounds infinite, please enter `-inf` and `inf`

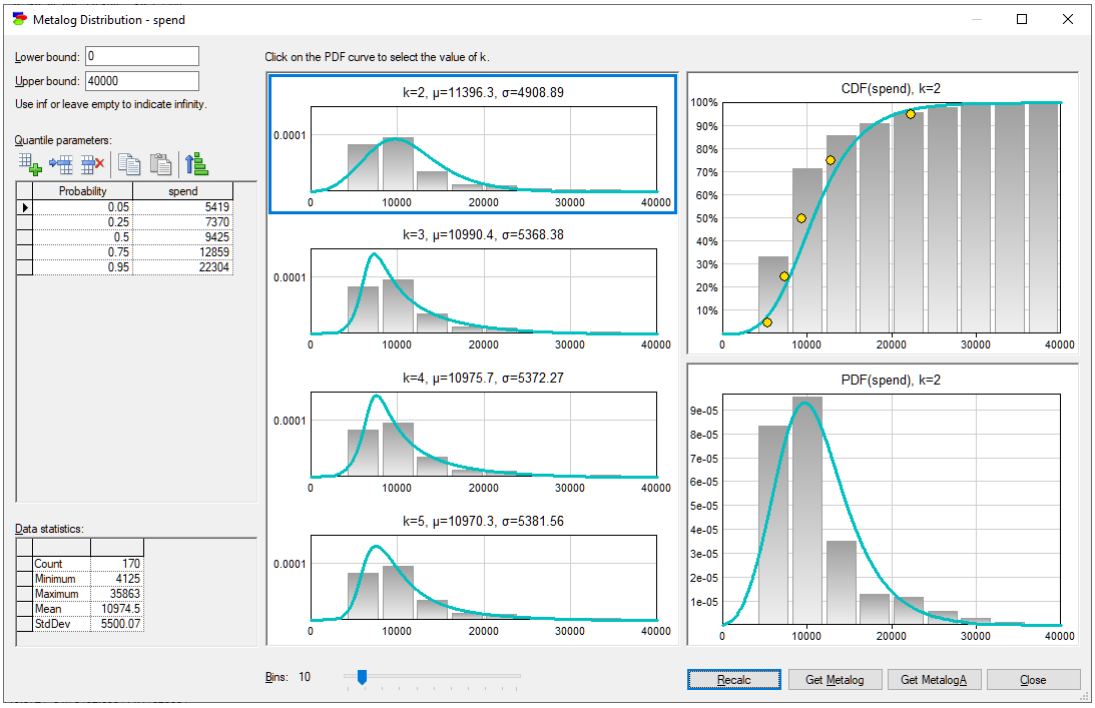
respectively or leave the bound field empty. The bottom-left corner shows basic statistics of the selected data column.

Input parameters are probabilities and the corresponding quantiles. The default probabilities are 0.05, 0.25, 0.5, 0.75, and 0.95. Generally, the higher the number of quantiles specified, the more complex and flexible the distribution. However, high number of quantiles carries the danger of overfitting the data, so we advise prudent caution, looking at the distributions generated for different values of k, and selecting a not-too-high value of k that produces the desired distribution.

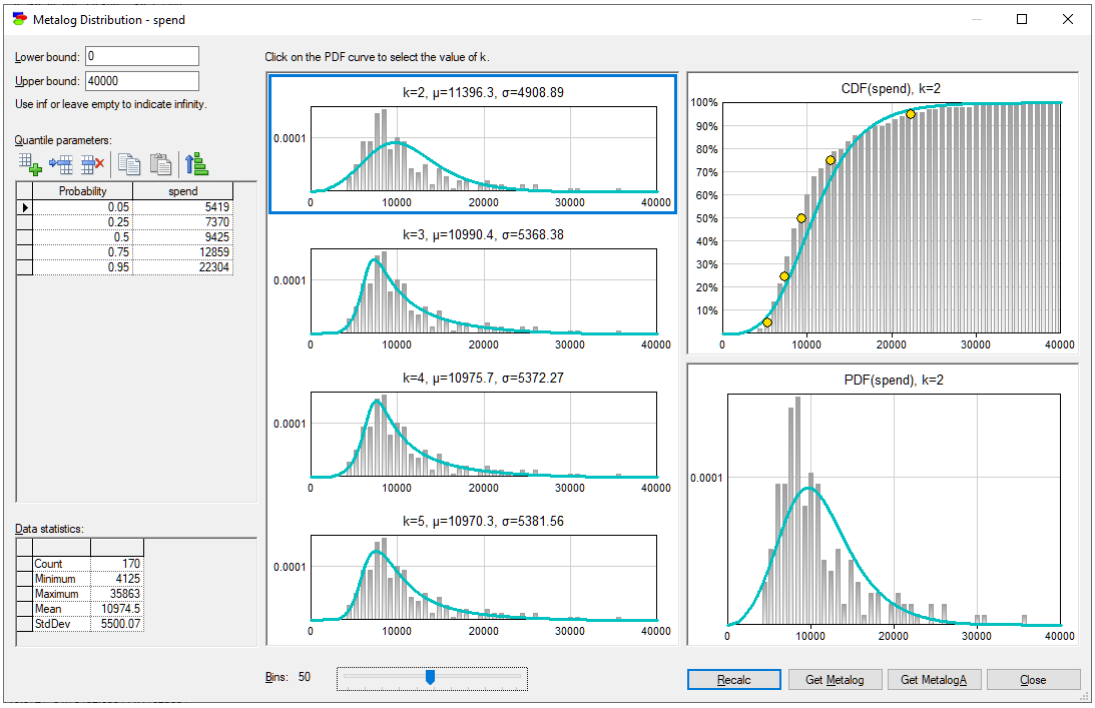
Editing the table comes with a similar set of tools as editing probability tables (*Add*, *Insert*, and *Delete* buttons). Probabilities do not need to be sorted a-priori, as GeNIe will validate and sort them for you before it starts with the process of fitting metalog distributions to data. You can sort the entries explicitly by pressing the *Sort* button (). GeNIe is also tolerant to empty rows between values. The two columns are inter-dependent given the data and changing the value in one column will lead to recalculating the value in the other column. Both columns can be copied and pasted. Pasting is restricted to one column because of the interdependence of the two columns. Pasting a text block that has more than one column does not make sense and will result in an error message.



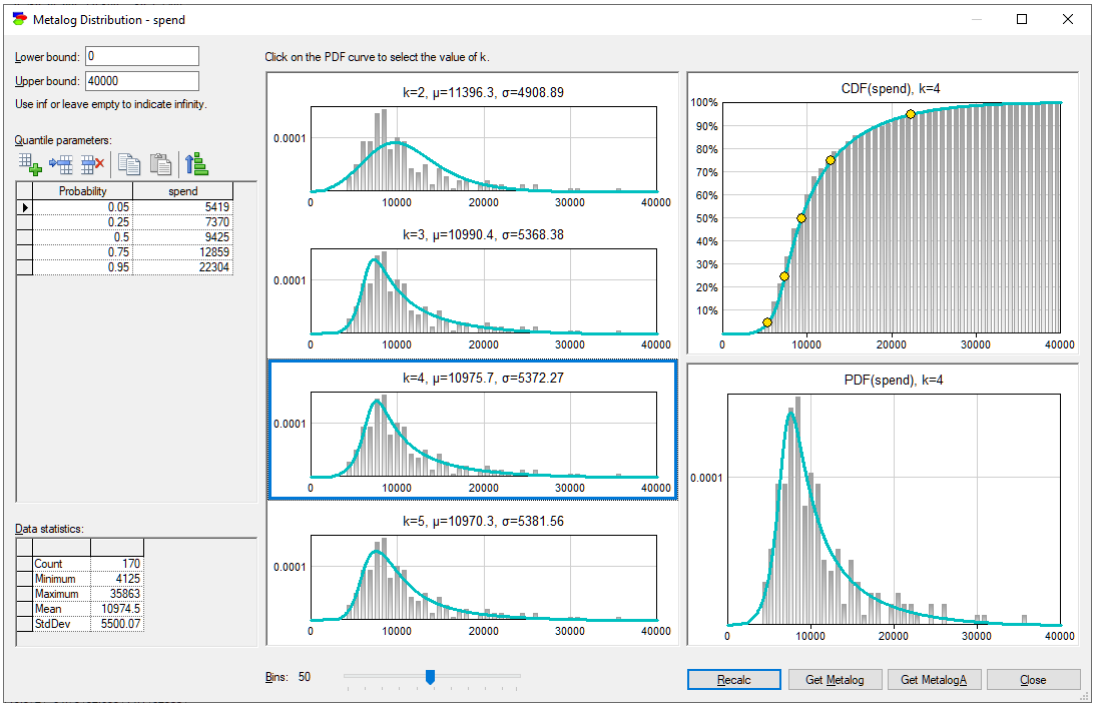
Pressing *Recalc* button shows a set of plots of the metalog distributions for k=2 through n that fit the data, where n is the number of input parameters (quantiles of the distribution). Only the feasible metalog distributions are shown, so the number of distributions may be smaller than n-1. As the number of quantiles specified in the example is 5 and all four metalog distributions are feasible, the dialog displays distributions for k=2, 3, 4, and 5. Histograms of the data are shown in the background so that the user can make a judgment of the goodness of fit. Generally the higher the value of k, the more flexible the distribution but it is worth looking at the distributions generated for different values of k to make a conscious choice. Mean and standard deviation of every metalog distribution are shown above each plot.



Changing the number of bins (effectively, the bin size, as all bins are equal width) often leads to additional insights about the goodness of fit.

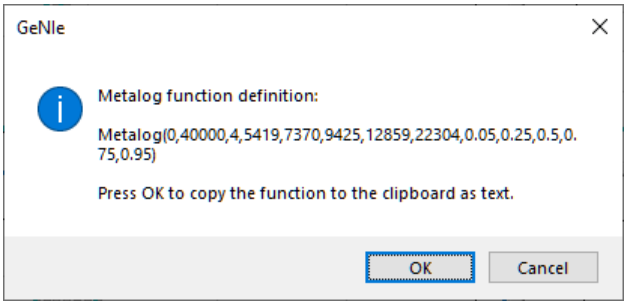


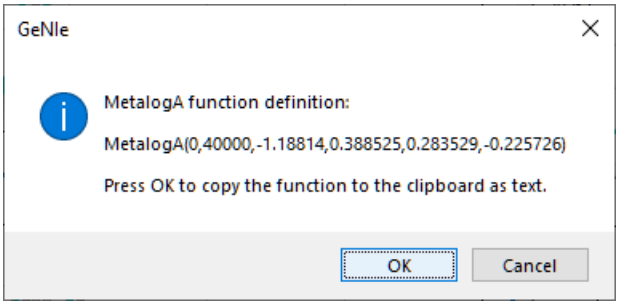
Clicking on any of the four plots displays the CDF and PDF of the distribution on the right-hand side. The CDF plot shows the input percentiles. Obviously, the closer the points are to the CDF line, the better the fit.



Any histogram in the dialog can be copied and later pasted as a picture into any other Windows application by right-clicking on the histogram and choosing *Copy*.

The buttons *Get Metalog* and *Get MetalogA* copy the formal description of the chosen metalog function for pasting elsewhere (usually in the definition of some node in your model).



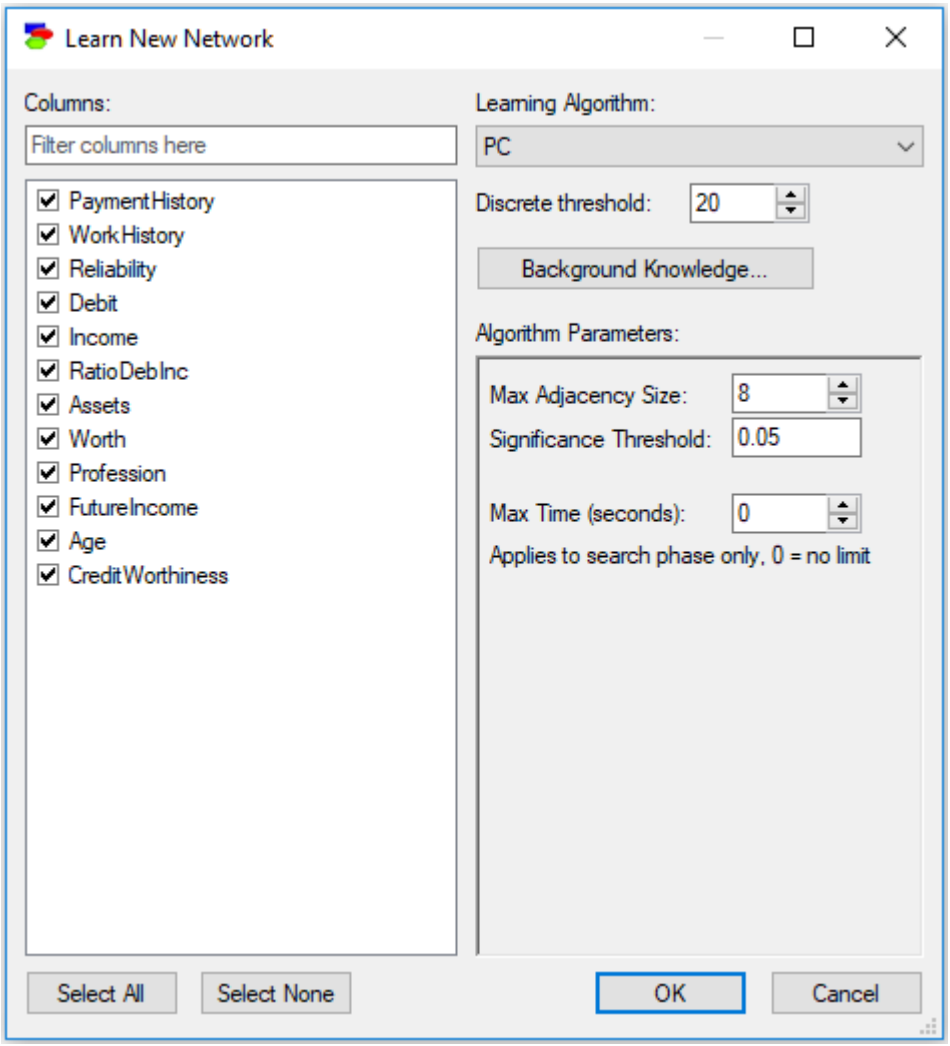


As explained in the section on *Random number generators*, the difference between the `Metalog` and `MetalogA` functions is that the latter uses internal metalog coefficients that do not have easily interpretable meaning. `Metalog`, on the other hand, uses parameters that are percentiles of the distribution. One might expect that `MetalogA` is more efficient in sample generation, as it skips the whole process of deriving the distribution from which it subsequently generates a sample. However, GeNIe has an efficient caching scheme that makes `Metalog` equally efficient in practice.

For more information about the metalog distribution, please look at the comprehensive article on the topic on Wikipedia (https://en.wikipedia.org/wiki/Metalog_distribution), the Metalog Distribution web site created by Tom Keelin (http://metalogdistributions.com/) or the Metalog Distributions YouTube channel, educational videos (https://www.youtube.com/channel/UCyHZ5neKhV1mSsedzDBoqyA). These sources provide access to articles by Tom Keelin and colleagues on the topic of the metalog distribution.

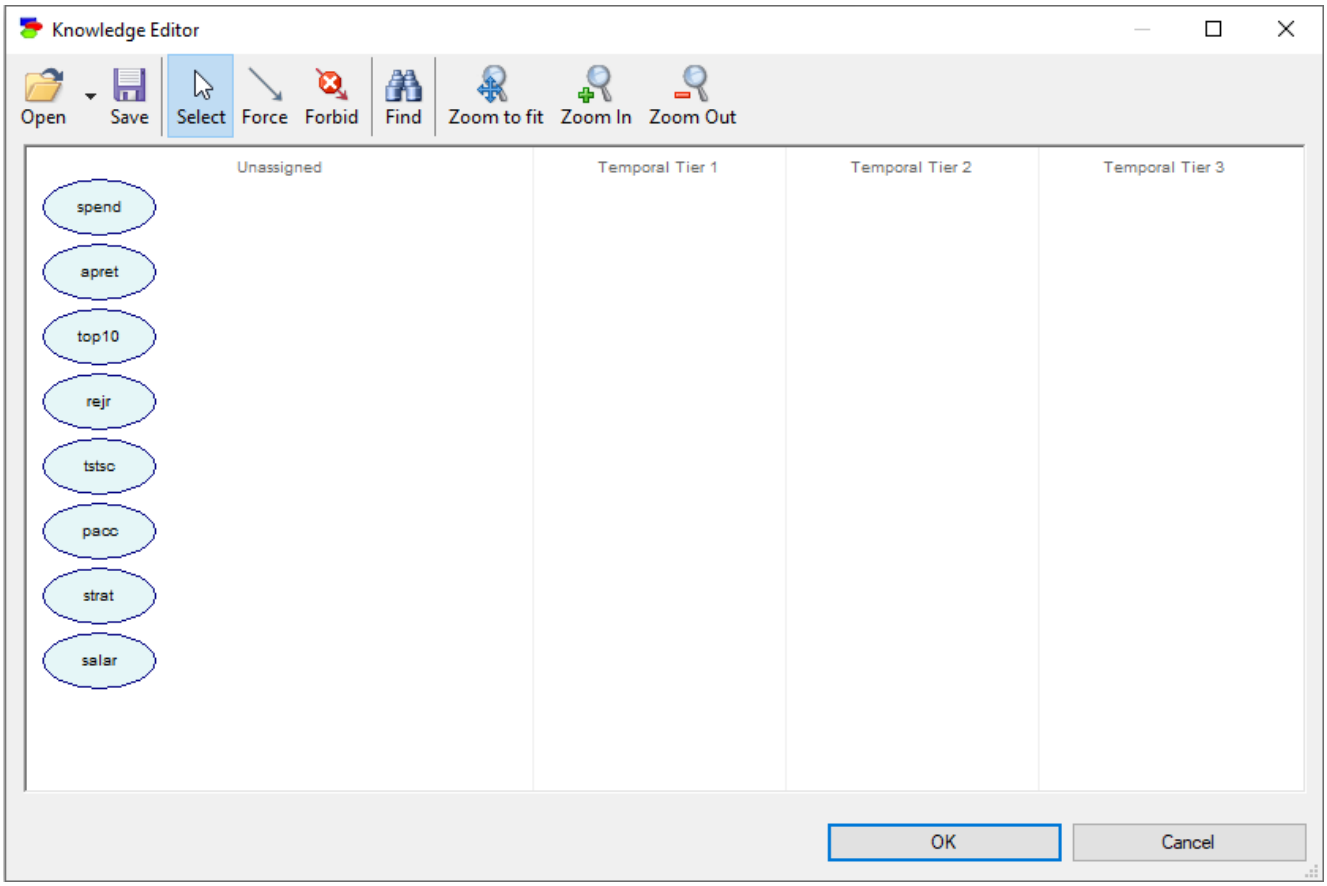
6.5.6 Knowledge editor

To invoke the *Knowledge Editor* dialog, click on the *Background Knowledge* button in the *Learn New Network* dialog.



Entering and editing domain knowledge

The *Knowledge Editor* dialog allows for entering outside (in the sense of not being present in the data) domain knowledge that will aid in the structure learning.



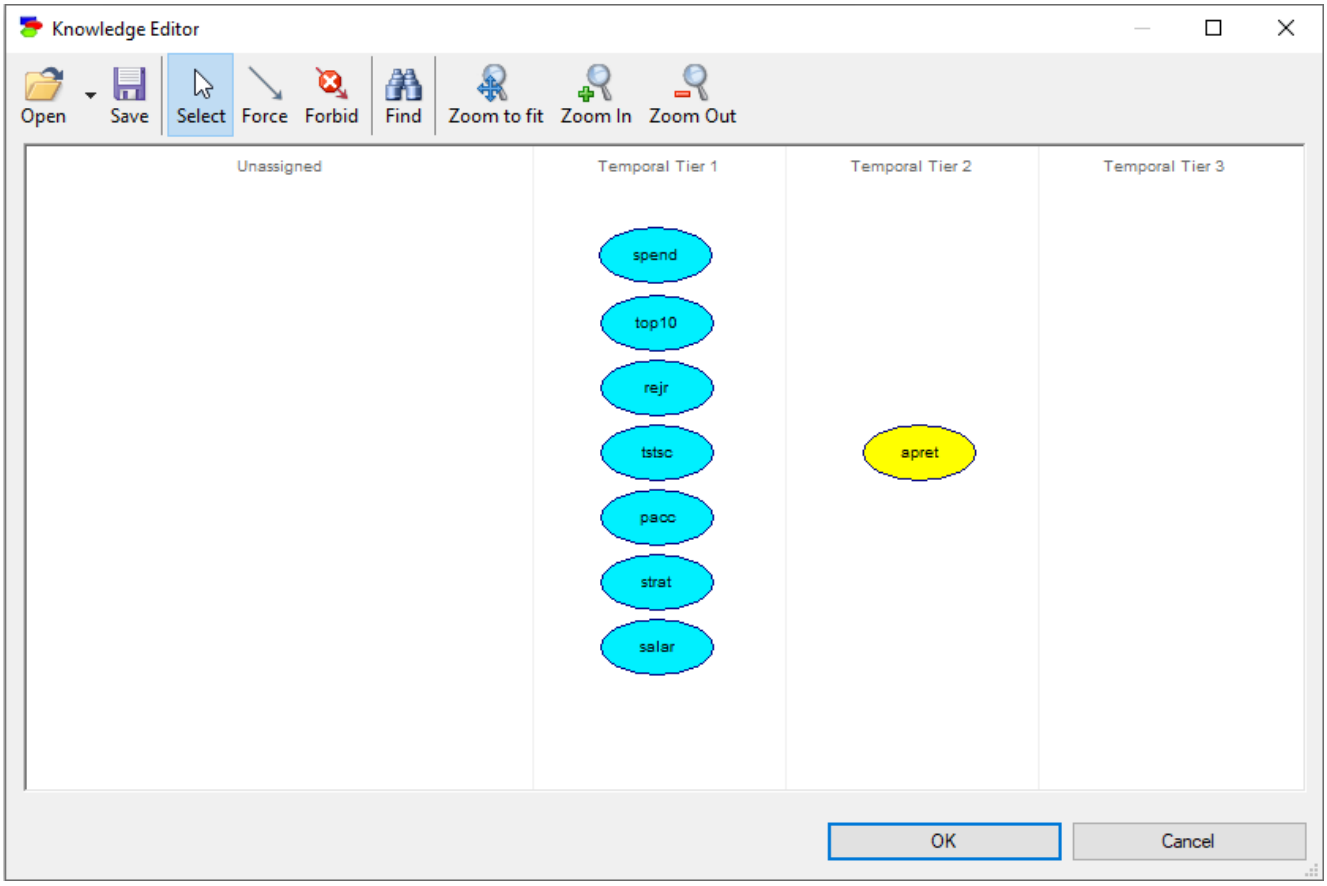
The *Knowledge Editor* dialog allows you to:

- force arcs (these arcs are guaranteed to appear in the learned structure)
- forbid arcs (these arcs are guaranteed to be absent in the learned structure), and
- assign variables to temporal tiers

Forced and forbidden arcs can be drawn by clicking on the origin and dragging the arc to the destination node, similarly to the way we draw arcs in the *Graph View*. We would like to caution the reader to be careful with specifying forced arcs. Even though parameters of the structure learning algorithms may put a maximum on the number of parents of a single node, forced arcs specified in the *Knowledge Editor* dialog override that constraint. We have encountered in the past users who introduced unnecessary complexity into their models by forcing too many arcs. It is useful to think of the forced and forbidden arcs as expressions of knowledge that is so certain and robust that we do not want it be overridden by the data. Because few pieces of knowledge should override data, we advice that these be used sparingly.

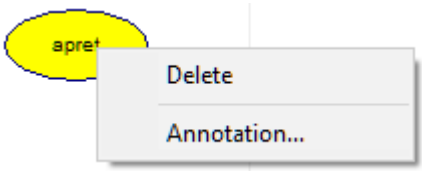
The rule about temporal tiers is simple: They are specifying the temporal order among the variables. In the resulting network, there will be no arcs from variables that occur later in time (in higher tiers) to nodes happening earlier in time (in lower tiers). It is useful to think about the learned structure in terms of causation: Causality never works backwards and we forbid arcs to go from variables in later temporal tiers to variables in

earlier temporal tiers. The following specification expresses that the variable *apret* occurs in time later than all the other variables:



Knowledge Editor node context menu

There is also a context menu for individual elements (nodes and arcs), which is invoked by right-clicking on a node or an arc:



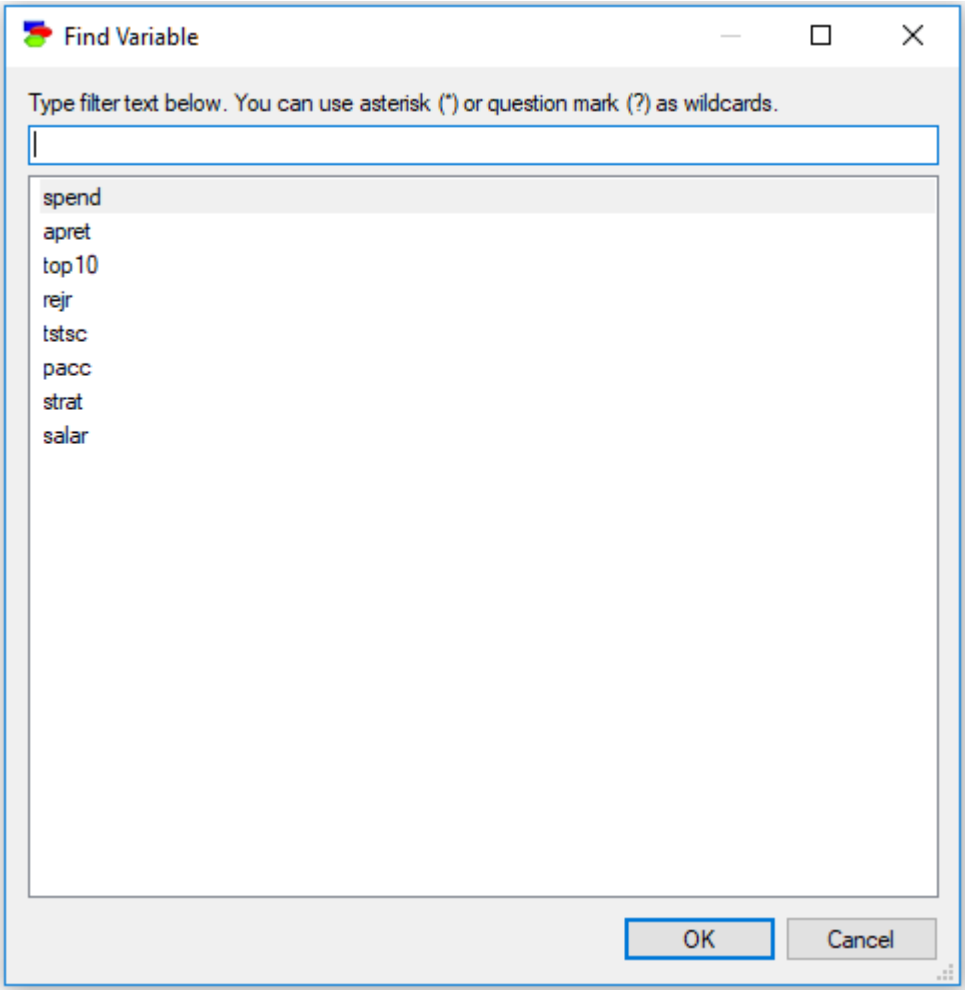
Delete removes the element (a node or an arc) from the knowledge and *Annotation* allows for adding annotations to individual nodes and arcs. These annotations are useful in preserving background information between sessions, when the knowledge is saved on the disk. Annotations on nodes and forced arcs carry over to the network that is learned with use of the background knowledge. Annotations on forbidden arcs are placed in the annotation in the destination node.

Saving and loading knowledge patterns

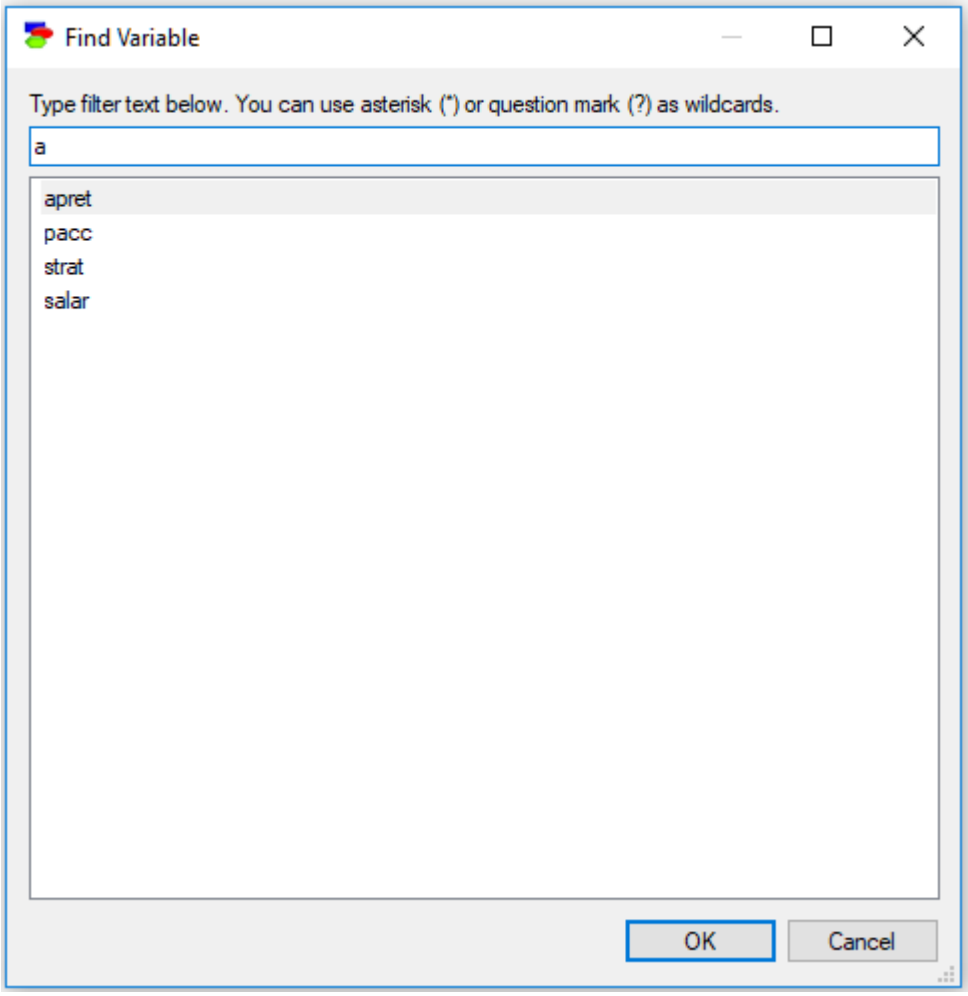
It is possible to save and later retrieve knowledge entered by means of the *Knowledge Editor*. Knowledge files have suffix **.gkno* for *G*eNIe *KNO*wledge.

Finding variables within the knowledge pattern

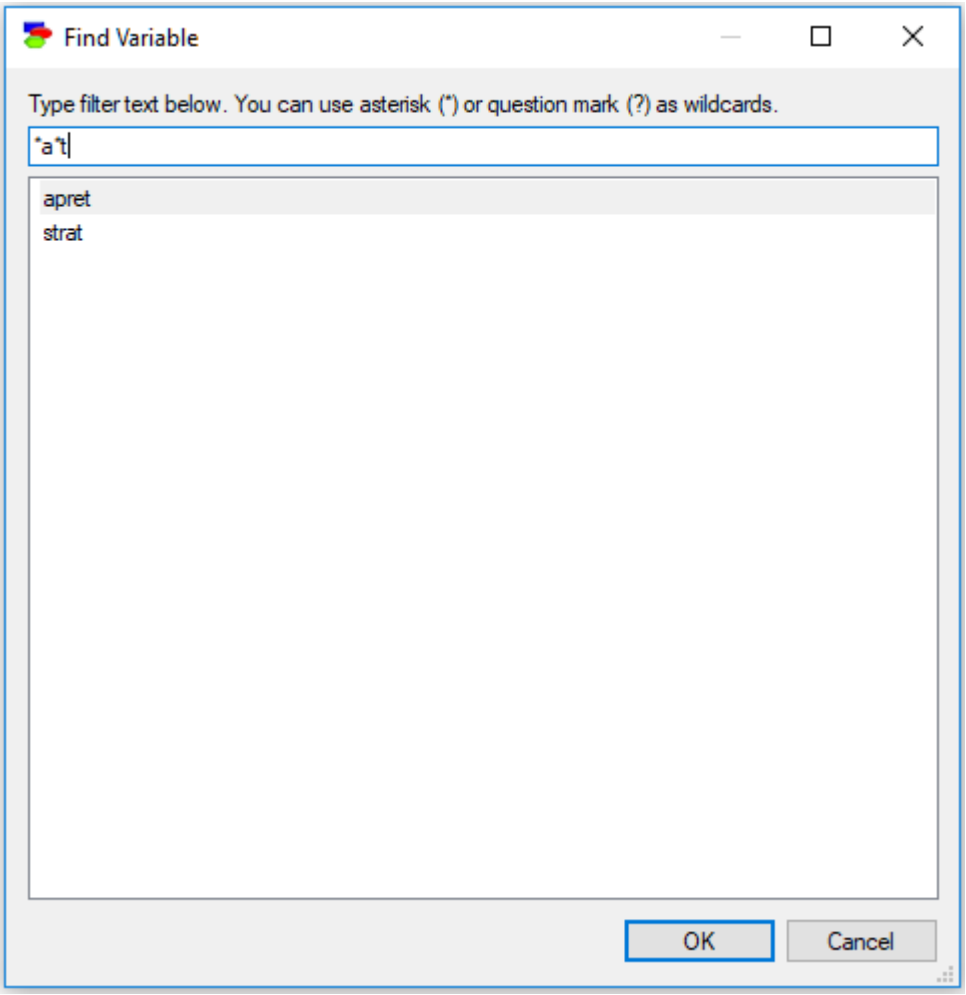
When a data set contains many variables (we have seen in our work with clients as many as 500 variables), locating a variable in the *Knowledge Editor* may be daunting. *Find* button ( ) invokes the *Find Variable* dialog:



The dialog works similarly to the *Find Column* dialog. Typing anything in the filter field causes the dialog to limit the names of variables to those that match the filter. For example, typing an a into the above dialog will lead to the following selection:



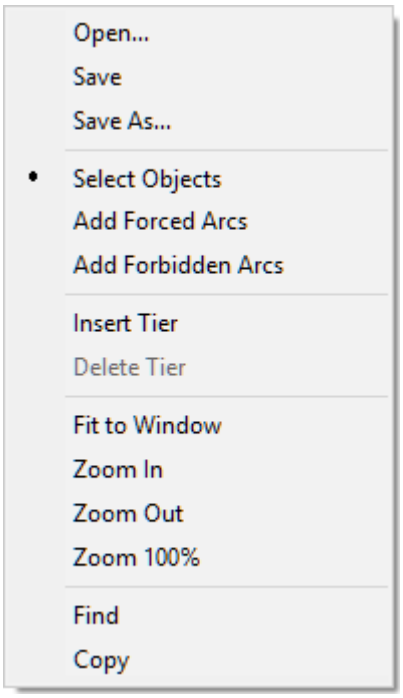
Letter case is ignored, so a lower case letter (e.g., "a") is equivalent to an upper case letter (i.e., "A"). In addition to regular characters, the filter field interprets wildcard characters, such as an asterisk (*) or a question mark (?) similarly to Windows. Typing "*a*t" will select only those column names that have an "a" character and end with a "t":



Double-clicking on any of the variable names or selecting a variable name and pressing OK will select that variable in the *Knowledge editor* window.

Knowledge Editor context menu

The *Knowledge Editor* is also equipped with a context menu, which is invoked by right-clicking on the background:

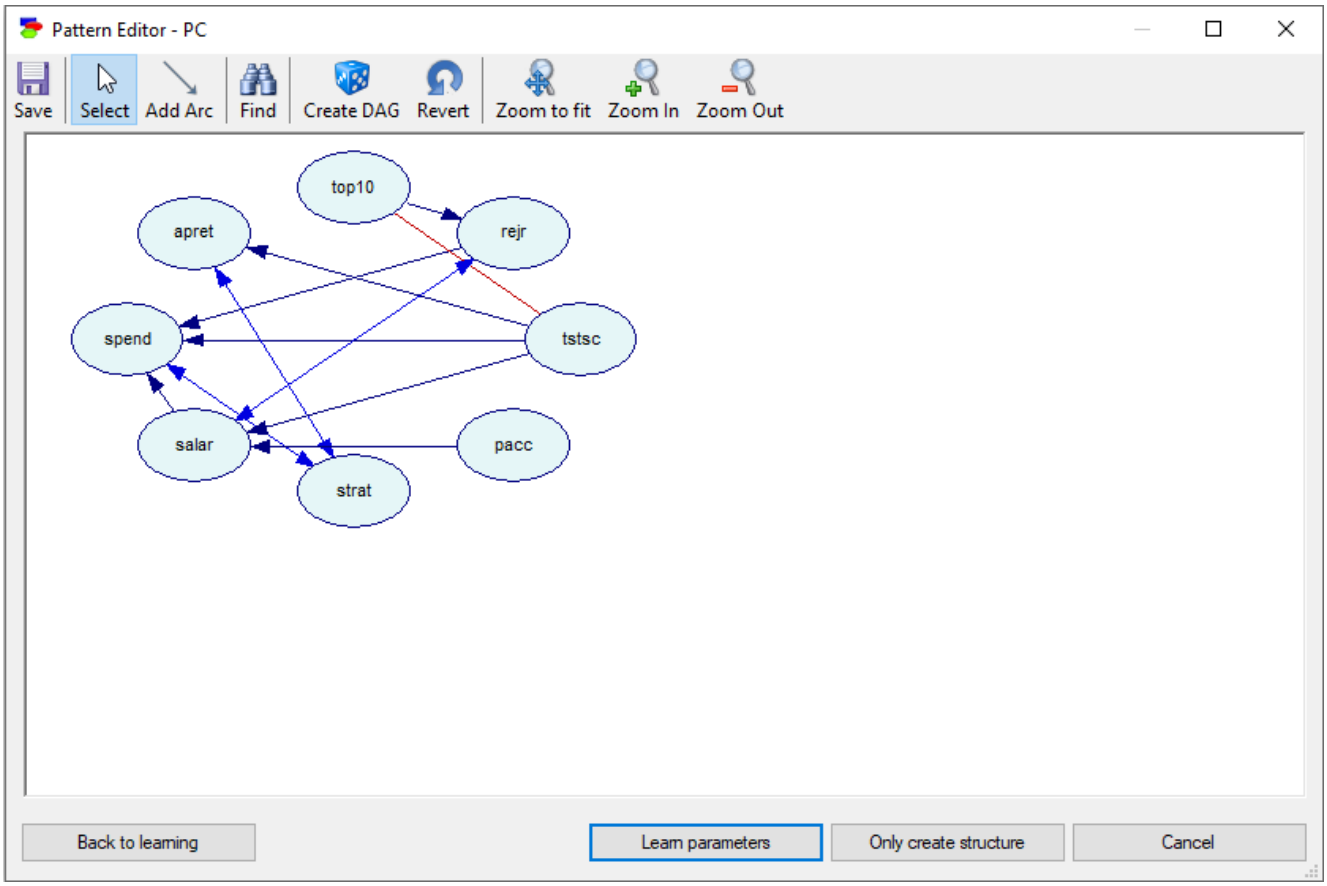


Most of the choices mimic the buttons on the top of the *Knowledge Editor*. *Zoom 100%*, absent among the buttons, restores the original 100% magnification of the nodes. *Copy* places the entire collection of the nodes on the clipboard (bitmap format or picture format, explained in the *Graph view* section) for a possible later paste into an external application.

Insert Tier creates a new temporal ties after the tier from which the context menu was invoked. *Delete Tier* deletes the temporal tier from which the context menu was invoked. If there are any variables in the tier, they are moved to the *Unassigned* tier.

6.5.7 Pattern editor

The output of some of the structure learning algorithms, e.g., the *PC* algorithm, is the *Pattern Editor* dialog.



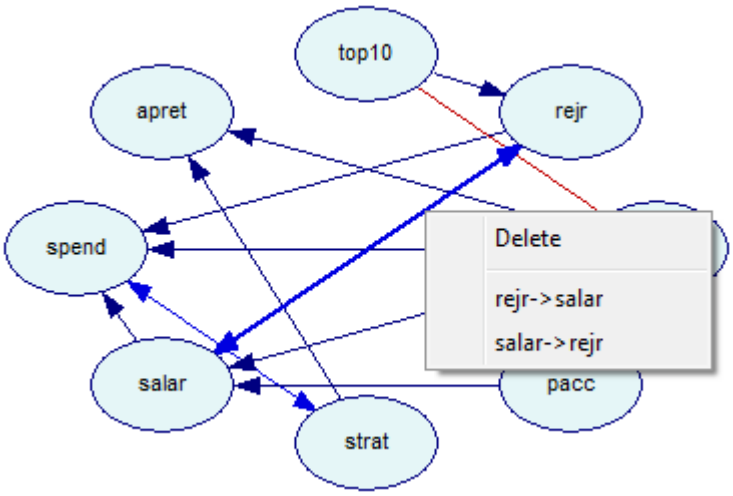
The *Pattern Editor* dialog displays the equivalence class of Bayesian networks that has been learned from data. There are three types of arcs in the *Pattern Editor* dialog: undirected (red), directed (black), and doubled-headed (blue) arcs. Their meaning is: an arc that could not be oriented based purely on data, a directed arc, and an arc that that could not be oriented based purely on data with a possibility of a hidden common cause, respectively. The pattern can be interpreted causally: All arcs that are oriented denote a possible causal relationship between the pair of variables that it connects. In the *retention.txt* data set, used in this section, the variable of interest is *apret*, which seems to be correlated with every variable in the data set. The structure learned suggests that most of the correlations are indirect and only two variables in this data set are directly related to *apret*: *strat* and *tstsc*.

Finding variables within the pattern

Find button () invokes the *Find Variable* dialog, which has an identical functionality to that in the *Knowledge editor*.

Manual modifications to the pattern

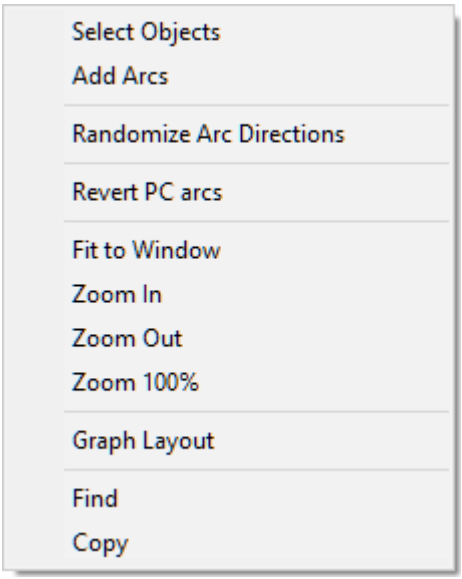
In order to turn this structure into an acyclic directed graph (a Bayesian network), we need to choose the direction for each undirected and double-headed arc. This can be done through right-clicking on an arc and choosing the direction:



or by clicking on the *Create DAG* ( ) button, which picks the direction of each undirected arc randomly to yield an acyclic directed graph.

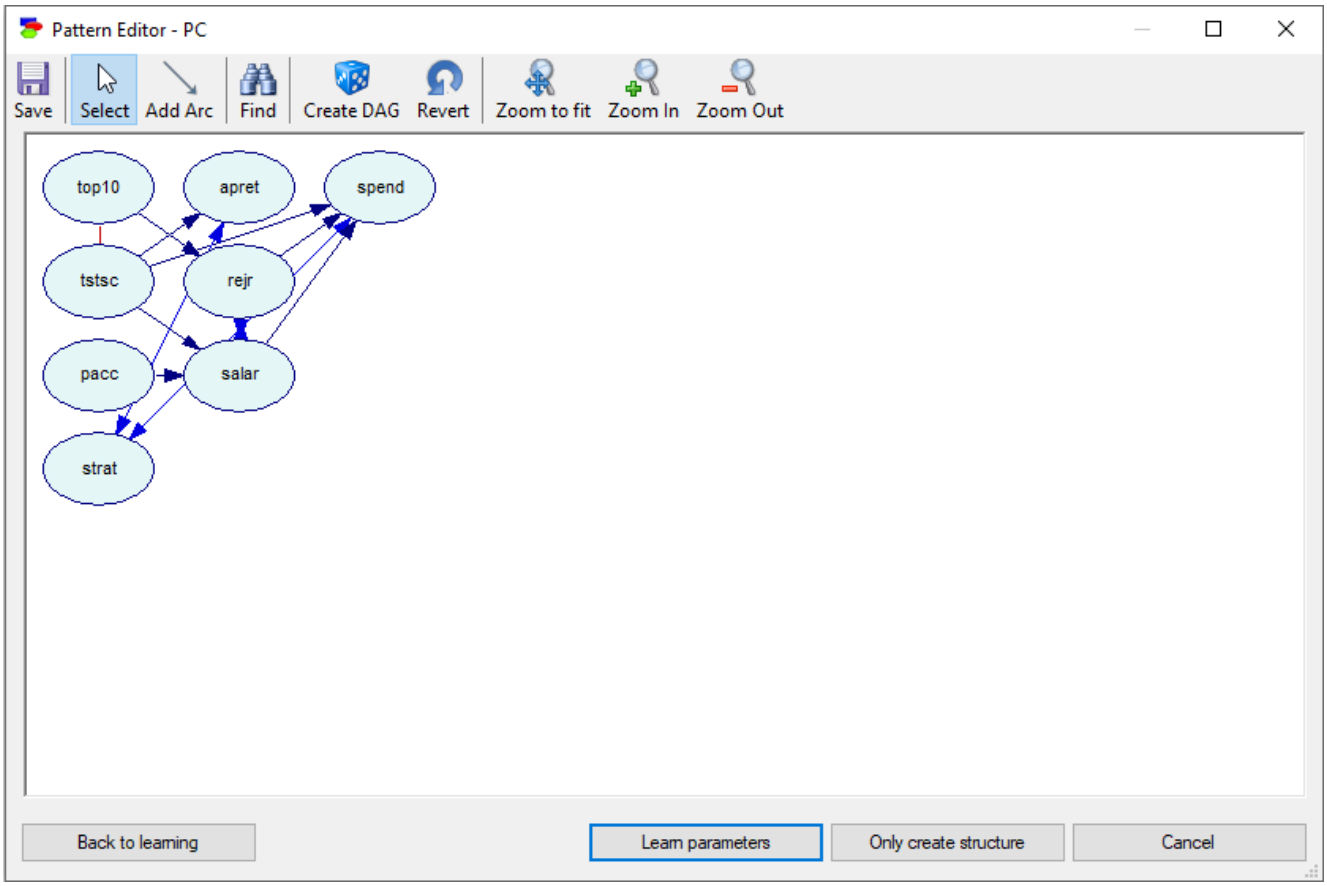
Pattern editor context menu

The *Pattern Editor* is also equipped with a context menu, which is invoked by right-clicking on the background:



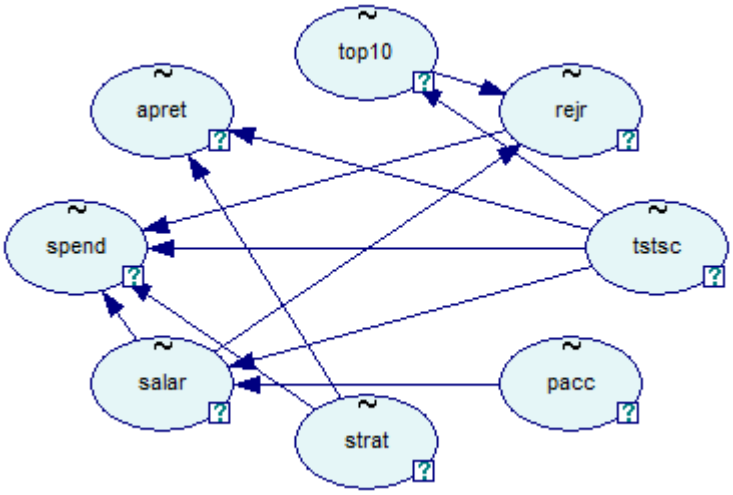
Most of the choices mimic the buttons on the top of the *Pattern Editor*. *Zoom 100%*, absent among the buttons, restores the original 100% magnification of the pattern. *Copy* places the entire pattern on the clipboard (bitmap or picture format, explained in the *Graph view* section) for a possible later paste into an external application.

Graph Layout applies the Parent Ordering layout algorithm to the pattern. This helps with analyzing the causal connections among the variables.



Finalizing the learning process

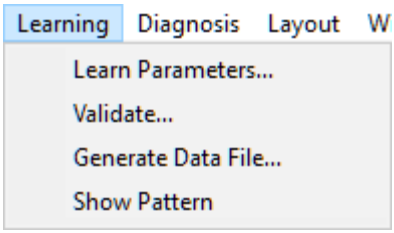
Clicking on the *Learn parameters* button yields a fully parametrized Bayesian network:



Which is typically the final product of any learning algorithm. Pressing the *Only create structure* button instead creates a Bayesian network without learning its parameters. The newly created network will have parameters but they will be all uniform distributions.

The *Back to learning* button brings back the *Learn New Network* dialog. This is especially useful if we want to modify the parameters of the learning algorithm. PC algorithm, for example, is very sensitive to the *Significance Threshold* and it is a good idea to run it at several significance thresholds.

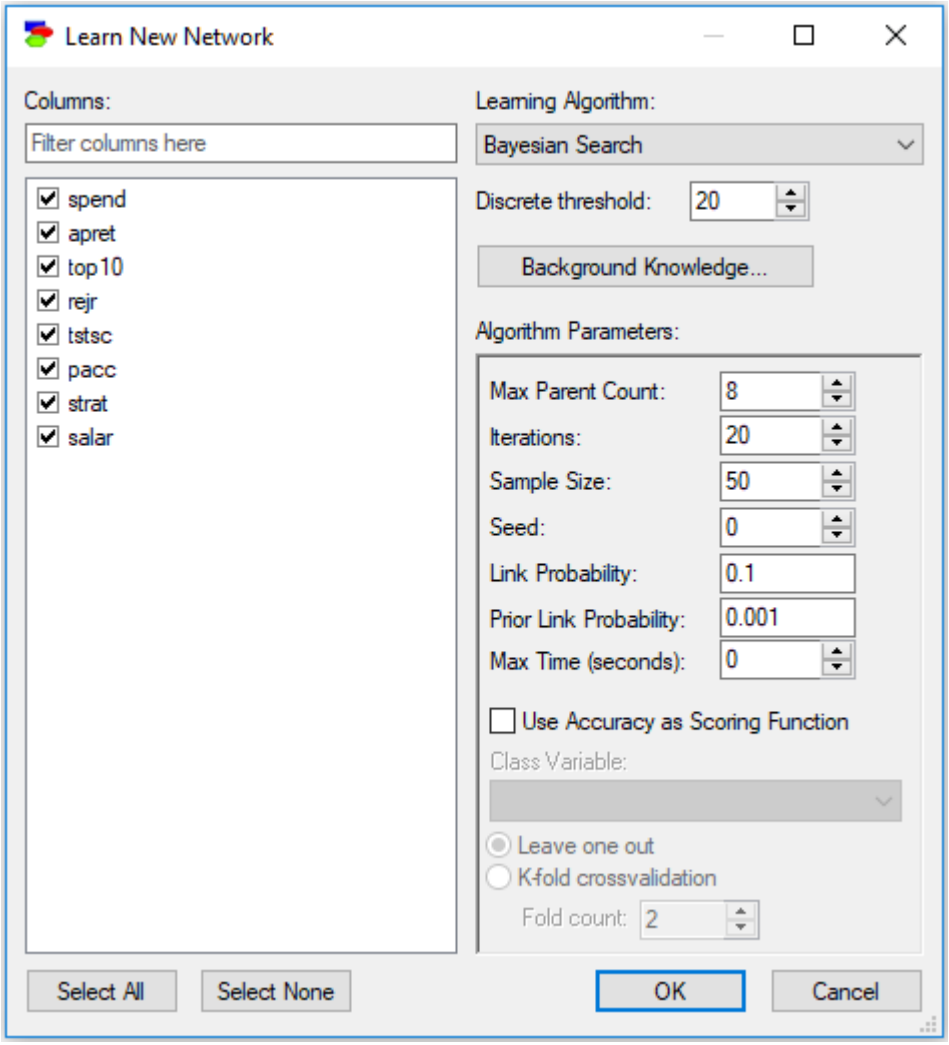
It is possible to come back from an acyclic directed graph of a Bayesian network to the *Pattern Editor* dialog by choosing *Show Pattern* from the *Learning* menu:



6.5.8 Structural learning

6.5.8.1 Introduction

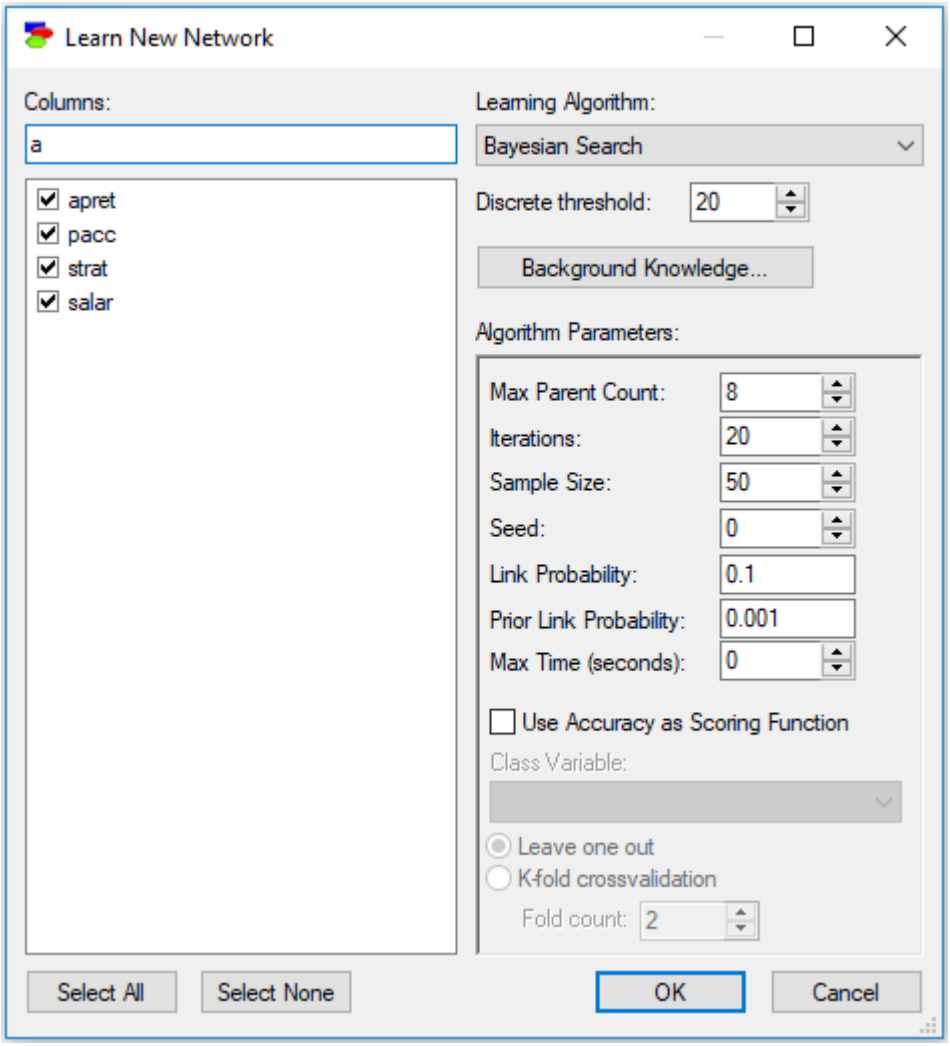
To invoke the structure learning dialog, select *Data-Learn New Network...* The ensuing dialog allows you to choose variables that will participate in learning, enter background knowledge, and choose one of the available learning algorithms and their parameters.



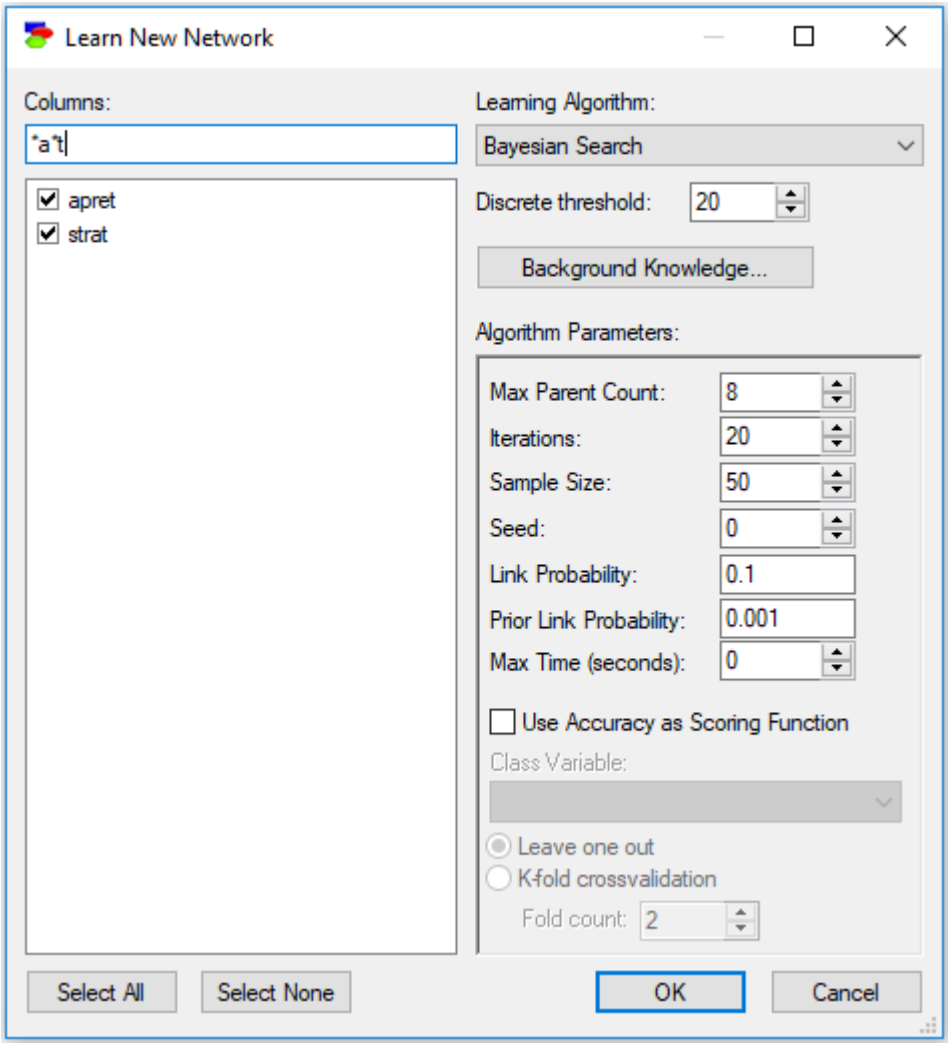
Selecting data columns for structure learning

Check boxes next to variable names on the left-hand side allow us for selecting those variables that will take part in learning. All variables with the check-box checked will be used in structure learning. Given that a practical data set may contains many variables (we have seen in our work with clients as many as 500 variables), locating and selecting columns for learning may be daunting. The filter field above the list of data columns serves for selecting columns.

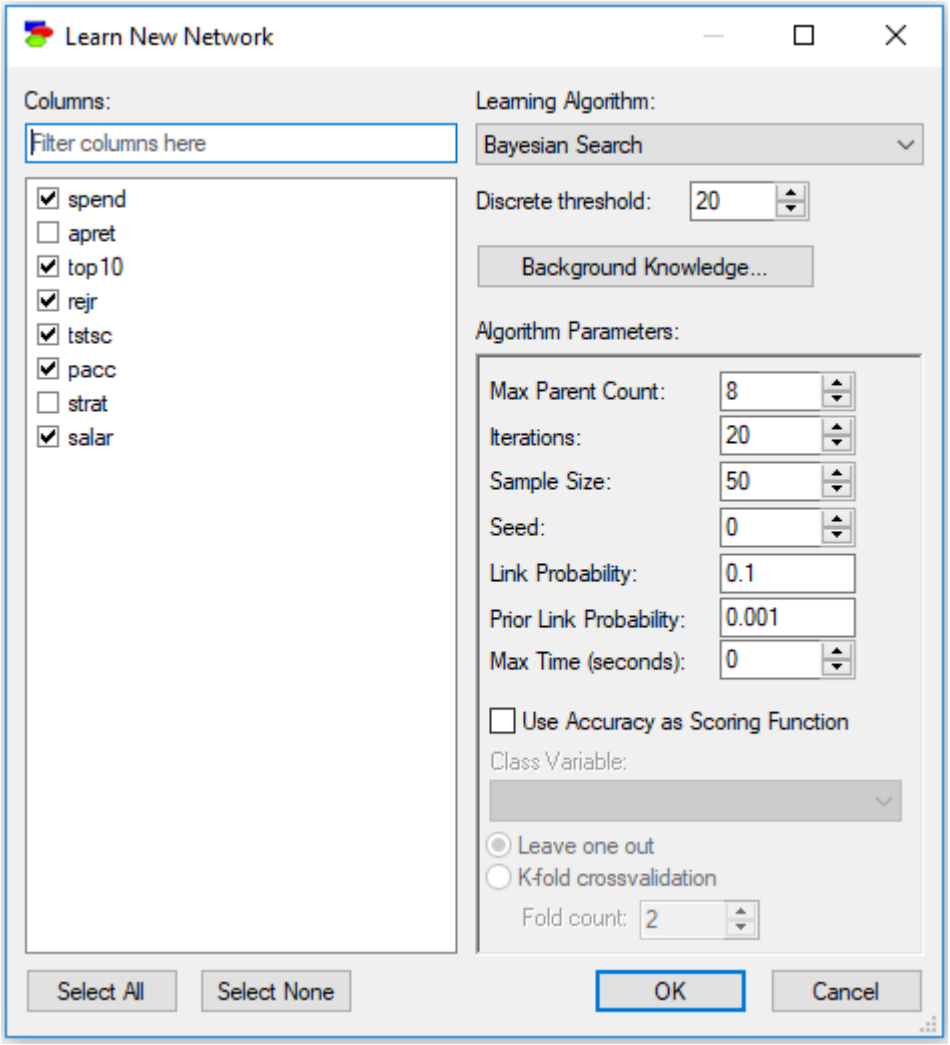
The dialog works similarly to the *Find Variable* dialog in the *Knowledge editor*. Typing anything in the filter field causes the dialog to limit the names of data columns to those that match the filter. For example, typing an a into the above dialog will lead to the following selection:



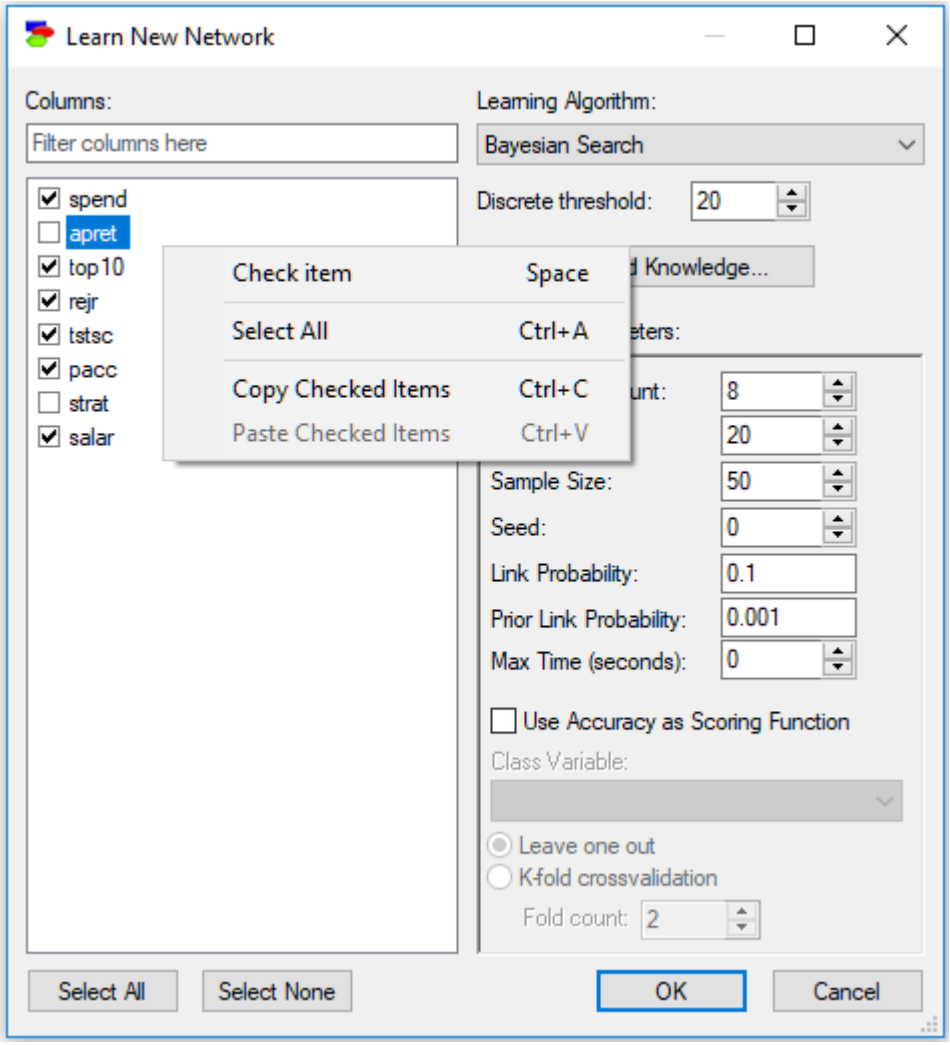
Letter case is ignored, so a lower case letter (e.g., "a") is equivalent to an upper case letter (i.e., "A"). In addition to regular characters, the filter field interprets wildcard characters, such as an asterisk (*) or a question mark (?) similarly to Windows. Typing "*a*t" will select only those column names that have an "a" character and end with a "t":



Any of the columns visible in the list can be selected or de-selected. Pressing one of the two buttons, *Select All* and *Select None*, will select or de-select columns visible in the column list. The buttons will have no effect on the remaining variables in the data set. If we press *Select None* button, thus and remove the filter, the effect will be



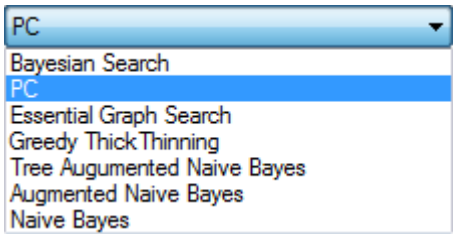
Finally, the interface allows for copying and pasting the list of selected columns. To that effect, please invoke the context menu on any element of the list



Copy Checked Items will place the list of the selected columns (text format) on the clipboard. The clipboard format is multi-line text containing the names of items in the list. If, at some later stage, we selected and copy the text list outside of GeNIe and choose *Paste Checked Items*, GeNIe will select only the columns on the list. *Paste Checked Items* command will remove the current check marks, unless the *SHIFT* key is pressed when command is invoked. This functionality is very convenient in case the list of columns in the data file is large and repetitive column selection process is laborious. The same format is used by *Copy Column Names* (see *Edit Menu for Data Spreadsheets* in *Cleaning Data*), which means that one can select columns directly in the data spreadsheet, use the *Copy Column Names* command, and then paste the names into the column list to select the same columns in the dialog.

Selecting a structure learning algorithm

The *Learning Algorithm* pop-up menu allows for selection of the learning algorithm



Discrete threshold parameter allows for selecting the minimum number of different states in a variable to be considered continuous. With its default value of 20, any variable that has more than 20 different values will be considered continuous.

Background knowledge

The *Background knowledge* button invokes the *Knowledge Editor* dialog, which allows for entering domain knowledge that will aid in the structure learning.

General properties of structure learning algorithms

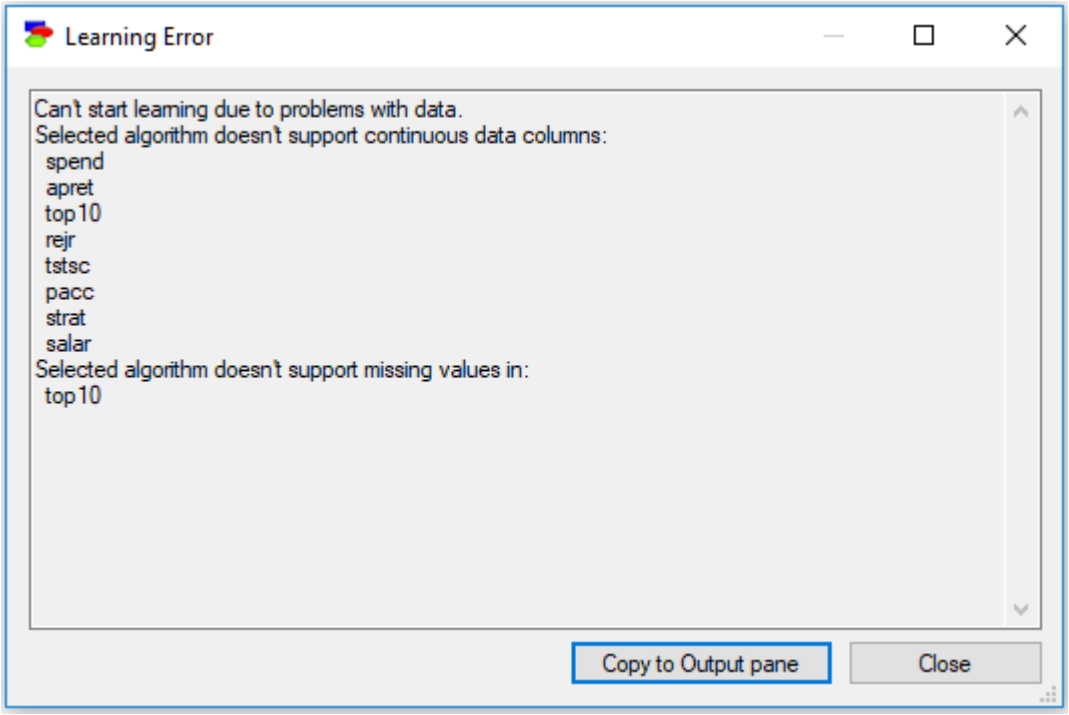
There are three major obstacles to structure learning that are tested at the beginning of the run of each of the algorithms:

- Each of the structure learning algorithms is capable of structural learning when all variables are categorical. In addition, the PC algorithm allows for learning the structure when all variables are continuous and their joint distribution is multivariate normal. None of the algorithms allows for learning from a mixture of discrete and continuous variables, so if there is a discrete variable in the learning set, it is necessary to discretize all continuous variables. In case the data set contains a mixture of discrete and continuous variables, GeNIe will complain about this and list all discrete and all continuous columns.

- None of the structure learning algorithms (except for the Naive Bayes algorithm, which does not learn the model structure but rather creates it based on strong independence assumption) is capable of learning the structure of a model when there are missing values in the records. In case there are any missing values in the data set, GeNIe will complain about this and list all variables with missing values. Please see also the *Missing Values* in *Cleaning data* section.

- None of the structure learning algorithms allows for learning with constant values, i.e., variables (columns in your data) containing the same value across all the records. Constant values are generally useless in learning the structure of a model. The common sense of this is the following: If a variable x takes the same value in each of the records, then it cannot be a predictor for any other variable. No matter what values the other variables take, x will take the same value anyway. There is, thus, no basis for judgment during the learning process what relationship x has to the remaining variables. In situations when one wants to include x in the model, one could enhance the model afterward by adding x and making a judgment of how x is connected to the rest of the model, including the parameters that describe these connections. In any case, constant variables should not be used in learning and GeNIe will complain if they are included in the learning set.

GeNIe generates a single error report as soon as the algorithm is invoked, so that the user can see in one place what needs to be done to the data for the algorithm to run correctly.



This report can be copied from the dialog and later pasted into an outside program, which is helpful in preparing the data for a new attempt at structure learning.

The remaining parameters are specific to the algorithm selected.

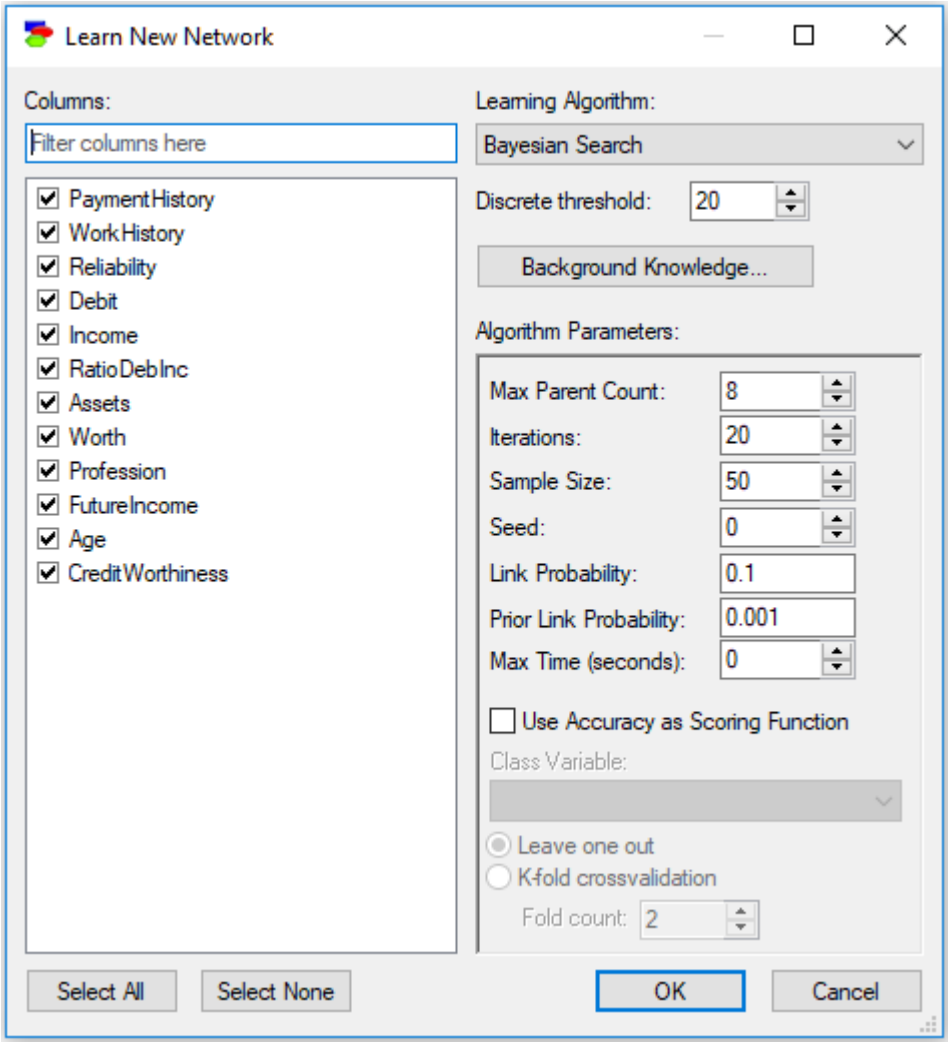
Each of the algorithms produces a graph, which is passed through the simple *Parent Ordering* layout algorithm for the purpose of readable positioning of nodes on the screen. As the *Parent Ordering* algorithm may not be able to result in an intuitive layout, the user is encouraged to improve the layout manually.

Variable/feature selection

There are currently no variable (feature) selection algorithms implemented in GeNIe (please stay tuned - we have plans to add them!). However, a user facing the need for variable selection is not left without tools supporting it. The easiest way is to run the *Naive Bayes* algorithm (as many times as there are class variables, for each class variable in separation) and then analyzing *Strength of influences*, both graphically and as a list of influences. Those variables having the strongest connections to the class variable are likely to be among the most important features. Another useful method is running a fast structure discovery algorithm, such as the *Greedy Thick Thinning* algorithm or the *PC* algorithm with a time limit (this ensures that the algorithm returns within the set time limit) and a very low significance threshold (this results in sparser graphs). Those nodes directly connected to the class variables are the most likely to include important features.

6.5.8.2 Bayesian Search

The *Bayesian Search* structure learning algorithm is one of the earliest and the most popular algorithms used. It was introduced by (Cooper & Herkovitz, 1992) and was refined somewhat by (Heckerman, 1995). It follows essentially a hill climbing procedure (guided by a scoring heuristic, which in GeNIe is the log-likelihood function) with random restarts. Here is the *Learn New Network* dialog for the *Bayesian Search* algorithm:



The *Bayesian Search* algorithm has the following parameters:

- *Max Parent Count* (default 8) limits the number of parents that a node can have. Because the size of conditional probability tables of a node grow exponentially in the number of the node's parents, it is a good idea to put a limit on the number of parents so that the construction of the network does not exhaust all available computer memory.

- *Iterations* (default 20) sets the number of restarts of the algorithm. Generally, the algorithm is searching through a hyper-exponential search space and its goal can be compared to searching for a needle in a haystack. Restarts allow for probing more areas of the search space and increase the chance of finding a structure that will fit the data better. We advise to make this number as large as we can afford it in terms of running time. The default number of iterations should give you an idea of how long the algorithm will take when the number of iteration is larger. The computing time is roughly linear in the number of iterations.

- *Sample size* (default 50) takes part in the score (*BDeu*) calculation, representing the inertia of the current parameters when introducing new data.

- *Seed* (default 0), which is the initial random number seed used in the random number generator. If you want the learning to be reproducible (i.e., you want to obtain the same result every time you run the algorithm), use the same *Seed*. *Seed* equal to zero (the default) makes the random number generator really random by starting it with the current value of the processor clock, so please try a different value for reproducibility of results.

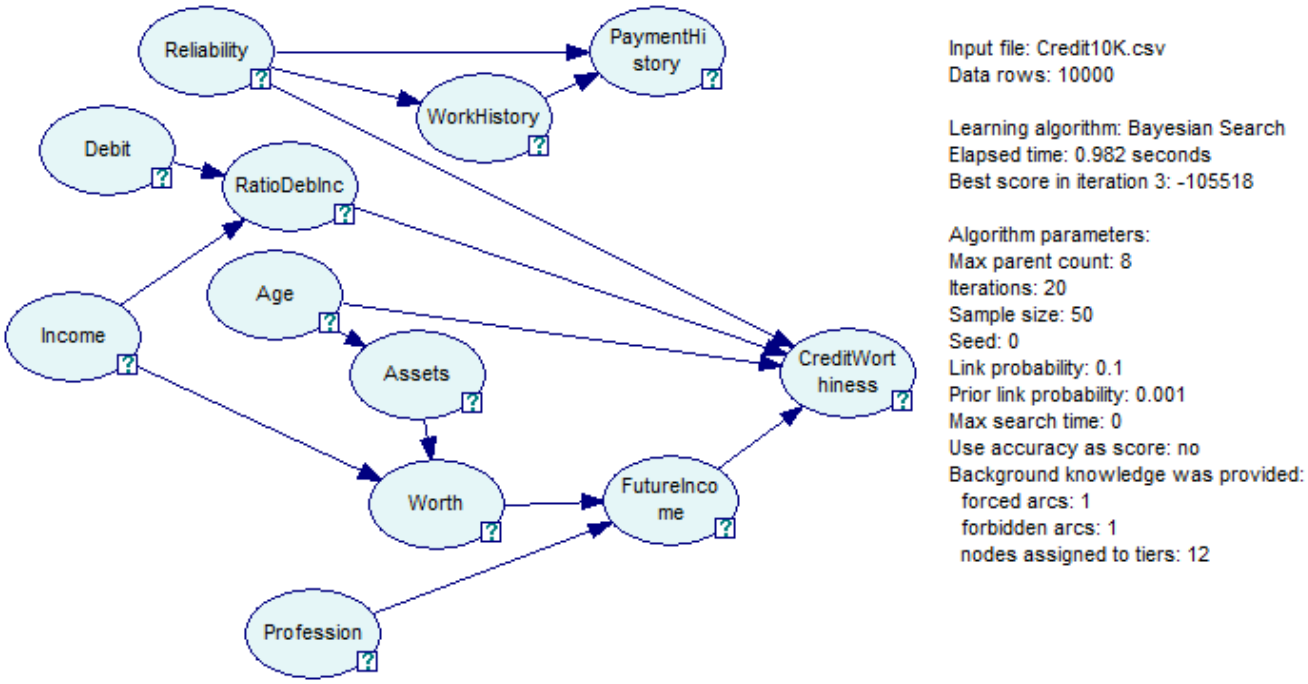
- *Link Probability* (default 0.1) is a parameter used when generating a random starting network at the outset of each of the iterations. It essentially influences the connectivity of the starting network.

- *Prior Link Probability* (default 0.001) influences the (*BDeu*) score, by offering a prior over all edges. It comes into the formula in the following way: log Posterior score = log marginal likelihood (i.e., the BDeu) + |parents| *log(pll) + (|nodes|-|parents|-1)*log(1-pll)).

- *Max Time (seconds)* (default 0, which means no time limit) sets a limit on the run time of the algorithm. It is a good idea to set a limit for any sizable data set so as to have the algorithm terminates within a reasonable amount of time.

- *Use Accuracy as Scoring Function* (default off). When checked, the algorithm will use the classification accuracy as the scoring function in search for the optimal graph. The user has to specify the class variable and the cross-validation technique (*Leave one out* or *K-fold cross-validation*, the former being a special case of the latter, when the *Fold count* is equal to the number of records in the data set). When this option is used, GeNIe uses classification accuracy only to compare the networks generated by (independent) iterations of BS. The actual inner work of a single BS iteration is not influenced by this option.

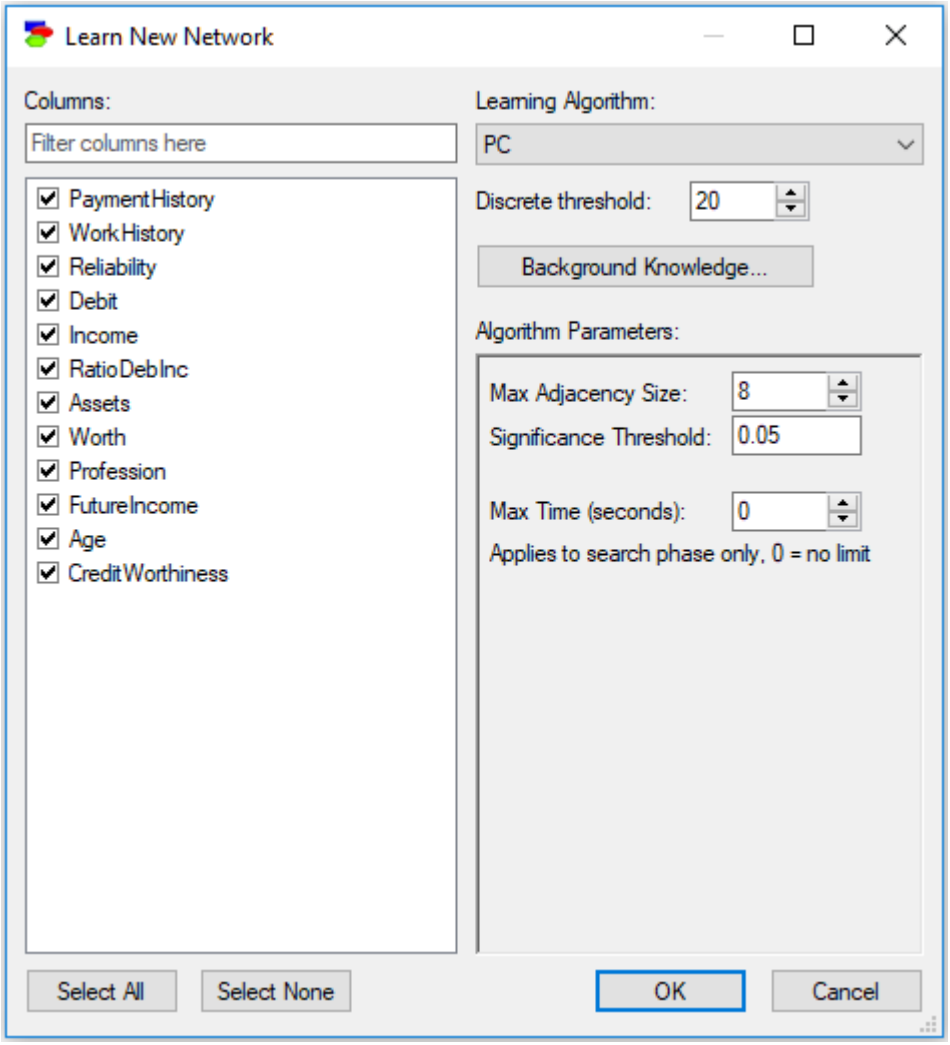
The algorithm produces an acyclic directed graph that achieves the highest score. The score is proportional to the probability of the data given the structure, which, assuming that we assign the same prior probability to any structure, is proportional to the probability of the structure given the data. The algorithm produces an on-screen text box that includes the settings of all parameters of the BS algorithms.



Because the *Bayesian Search* algorithm produces an acyclic directed graph, it is a good idea to investigate the theoretical limits to what it can identify based on the data. To this effect, we advise the user to transform the acyclic directed graph of a Bayesian network to the *Pattern Editor* dialog**.**

6.5.8.3 PC

The *PC* structure learning algorithm is one of the earliest and the most popular algorithms, introduced by (Spirtes et al., 1993). It uses independences observed in data (established by means of classical independence tests) to infer the structure that has generated them. Here is the *Learn New Network* dialog for the *PC* algorithm:



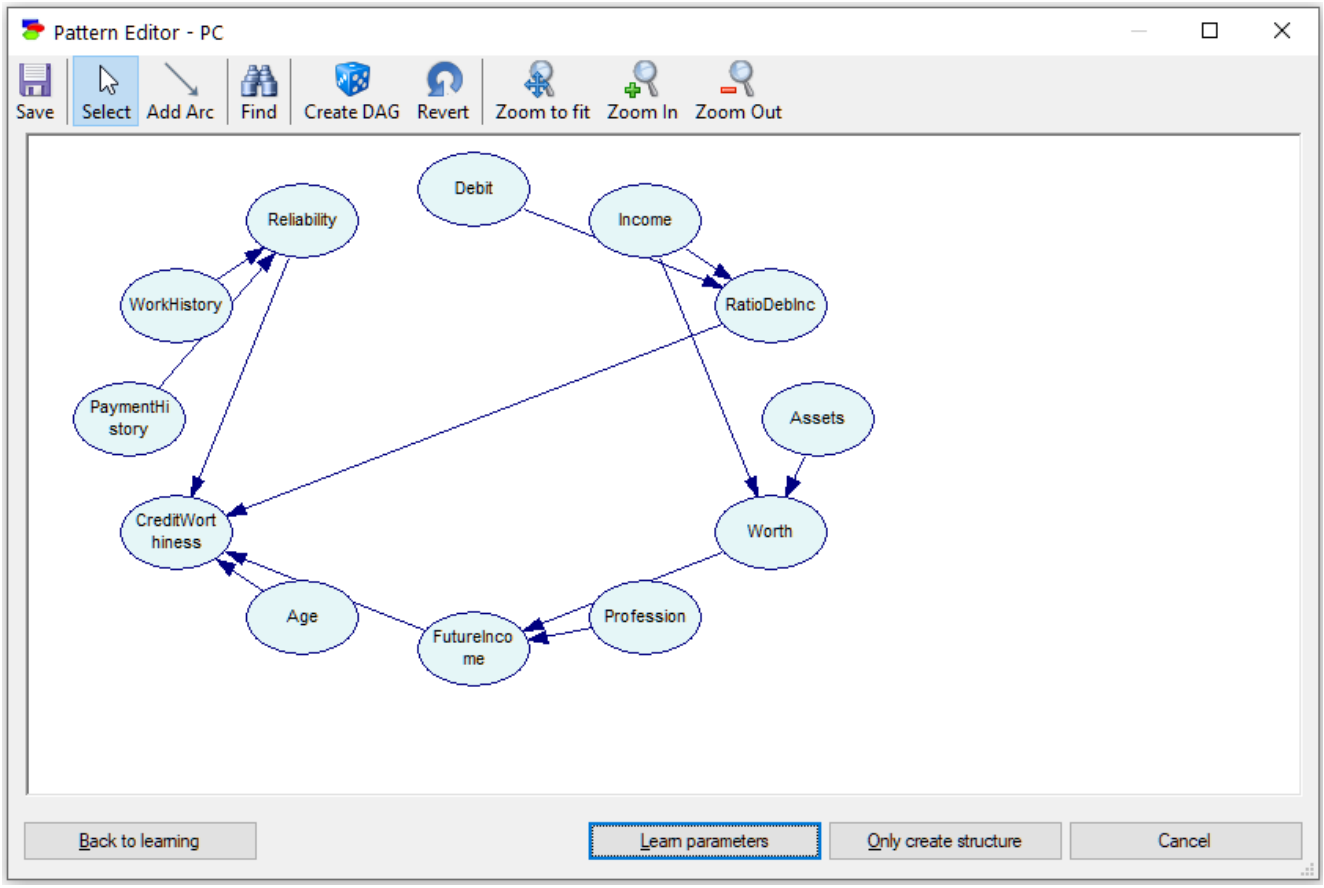
The PC algorithm has three parameters:

- *Max Adjacency Size* (default 8), which limits the number of neighbors of a node. In practice, this parameter is important for limiting the number of parents that a node corresponding to a variable can have. Because the size of conditional probability tables of a node grow exponentially in the number of the node's parents, it is a good idea to put a limit on the number of parents so that the construction of the network does not exhaust all available computer memory.

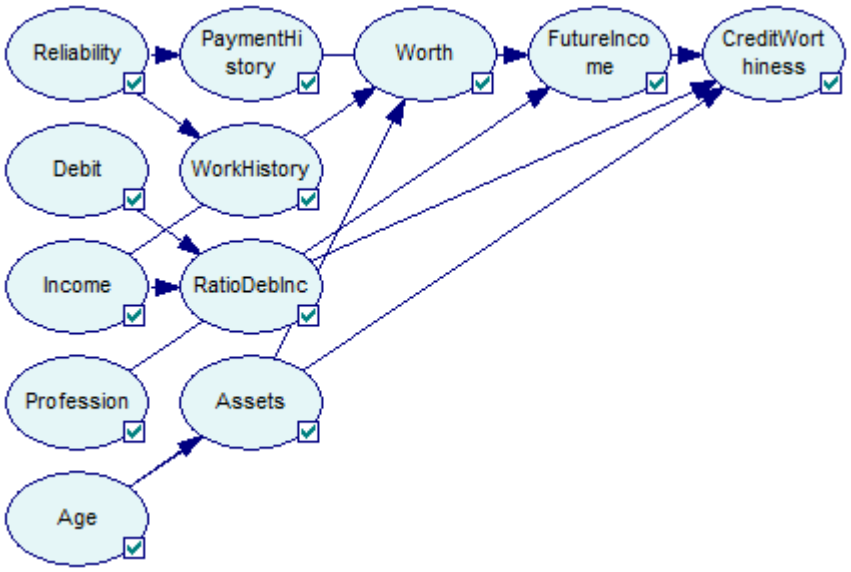
- *Significance Level* (default 0.05) is the alpha value used in classical independence tests on which the PC algorithm rests.

- *Max Time (seconds)* (default 0, which means no time limit) sets a limit on the search phase of the PC algorithm. It is a good idea to set a limit for any sizable data set so as to have the algorithm terminate within a reasonable amount of time.

Pressing *OK* and then *OK* in the *Learn New Network* dialog starts the algorithm, which ends in the *Pattern Editor* dialog.

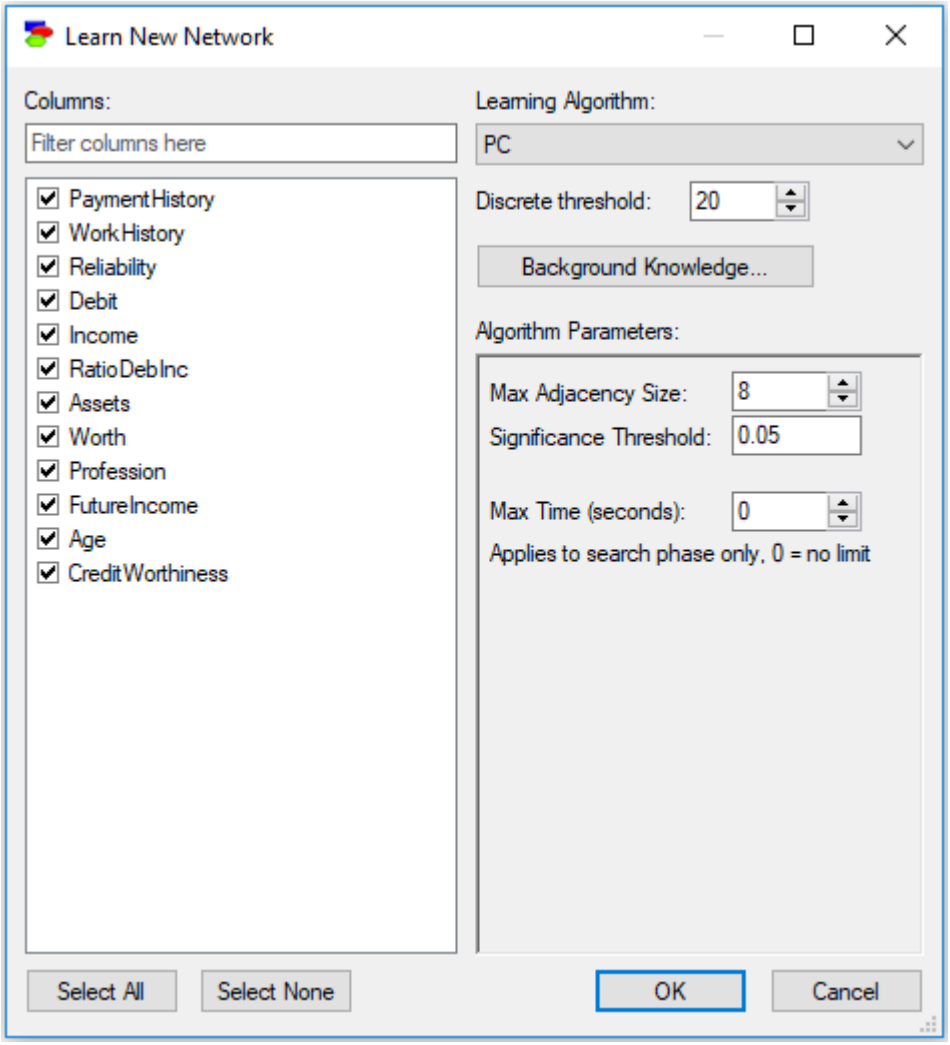


There is one double-edged arc in the pattern (between *Assets* and *Worth*). Clicking on the *Create DAG* ( ) button, which picks the direction of each undirected arc randomly to yield an acyclic directed graph, clicking on the *Learn parameters* button, and then running a graph layout algorithm (*Parent Ordering* from left to right) results in the following Bayesian network:

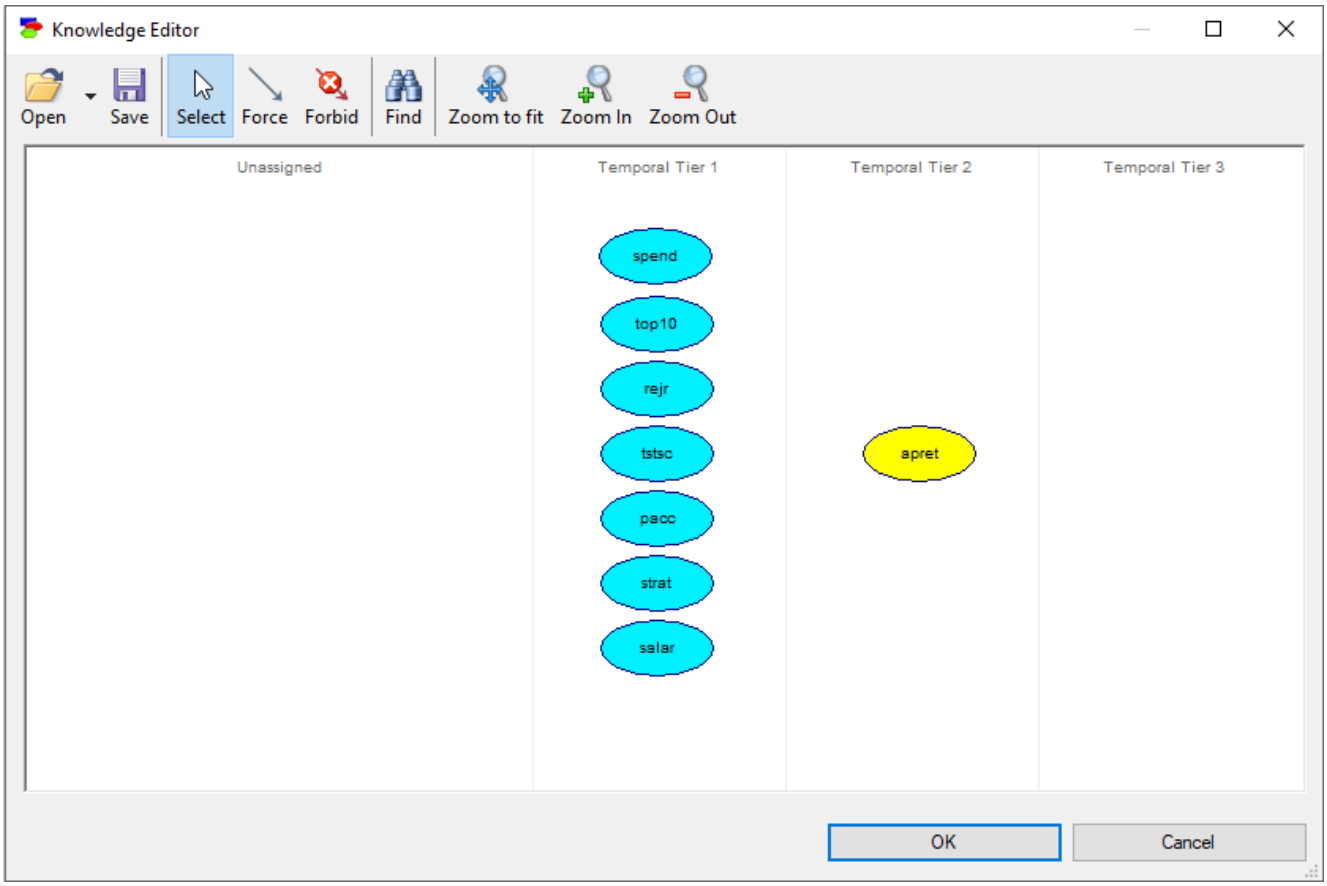


The PC algorithm is the only structure learning algorithm in GeNIe that allows for continuous data. The data have to fulfill reasonably the assumption that they come from multivariate normal distribution. To verify this assumption, please check that histograms of every variable are close to the Normal distribution and that scatter plots of every pair of variables show approximately linear relationships. Voortman & Druzdzel (2008) verified experimentally that the PC algorithm is fairly robust to the assumption of multi-variate Normality.

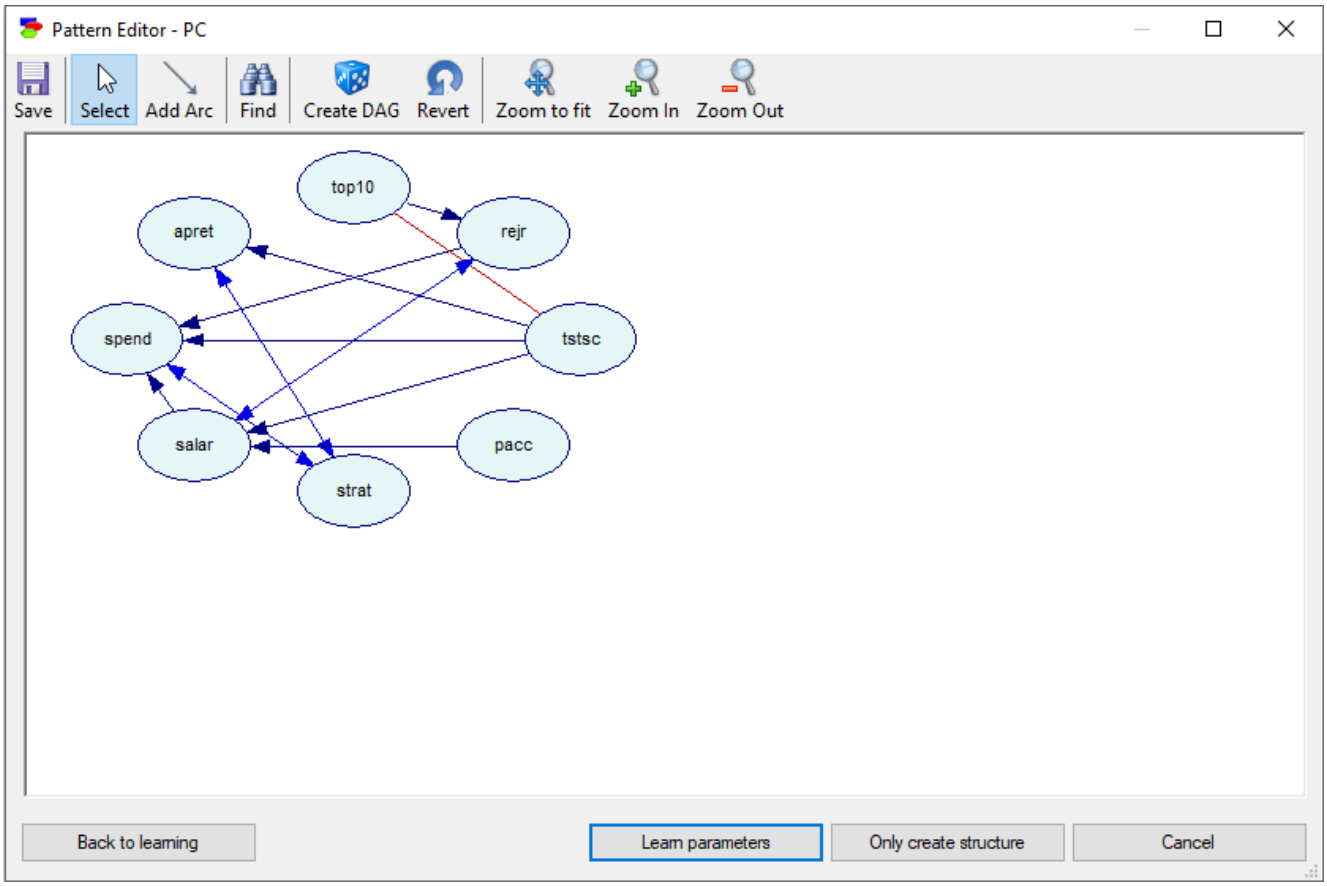
Let us demonstrate the working of the PC algorithm on a continuous data set retention.txt, included among the example data sets.



Here is the suggested specification of prior knowledge about the interactions among the variables in the data set:



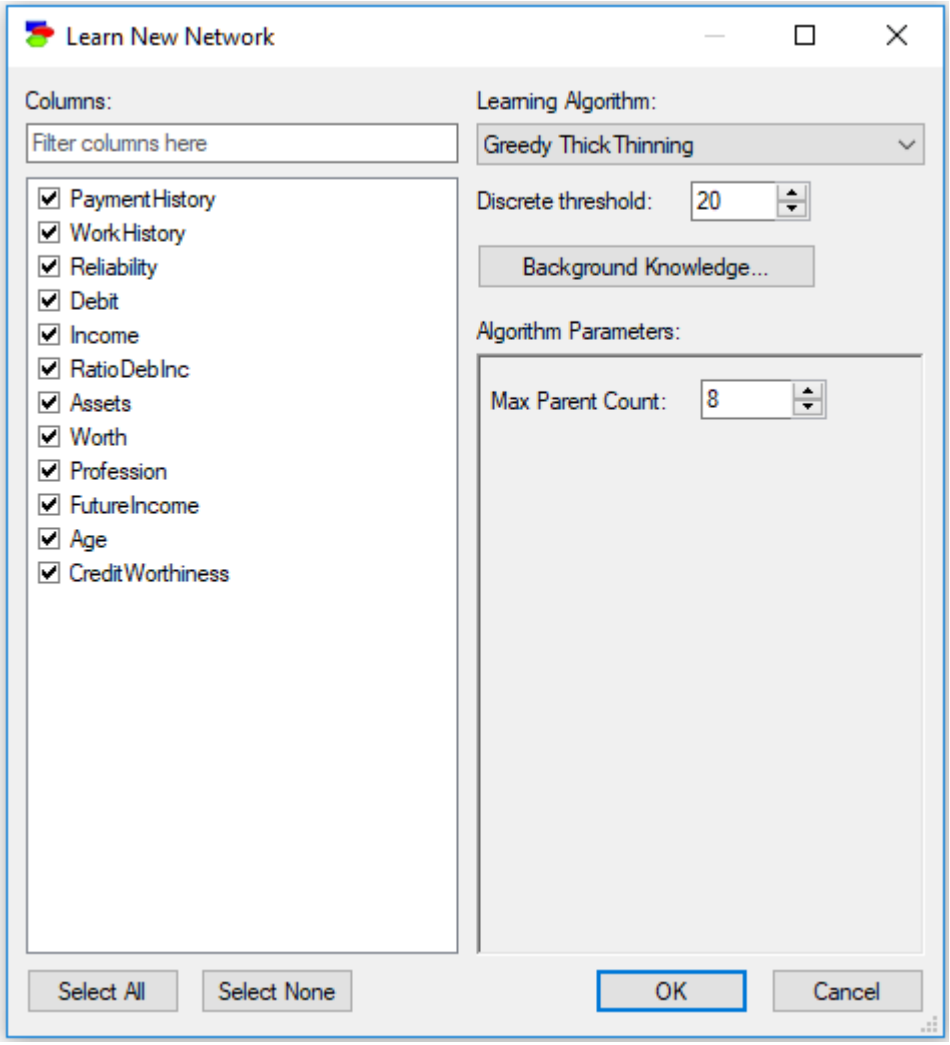
Pressing *OK* and then *OK* in the *Learn New Network* dialog starts the algorithm, which ends in the *Pattern Editor* dialog.



The only two causes of the variable *apret* (average percentage of student retention) are *tstsc* (average standardized test scores of incoming students) and *strat* (student-teacher ratio). The connection between *strat* and *apret* disappears when we repeat the learning with the *Significance Level* parameter set to *p=0.01*. This example is the subject of a paper by Druzdzel & Glymour (1998), who concluded that the only direct cause of low student retention at US universities is the quality of incoming students. This study is one of the successful examples of causal discovery, and its conclusion was verified empirically later in a real-life experiment by Carnegie Mellon University.

6.5.8.4 Greedy Thick Thinning

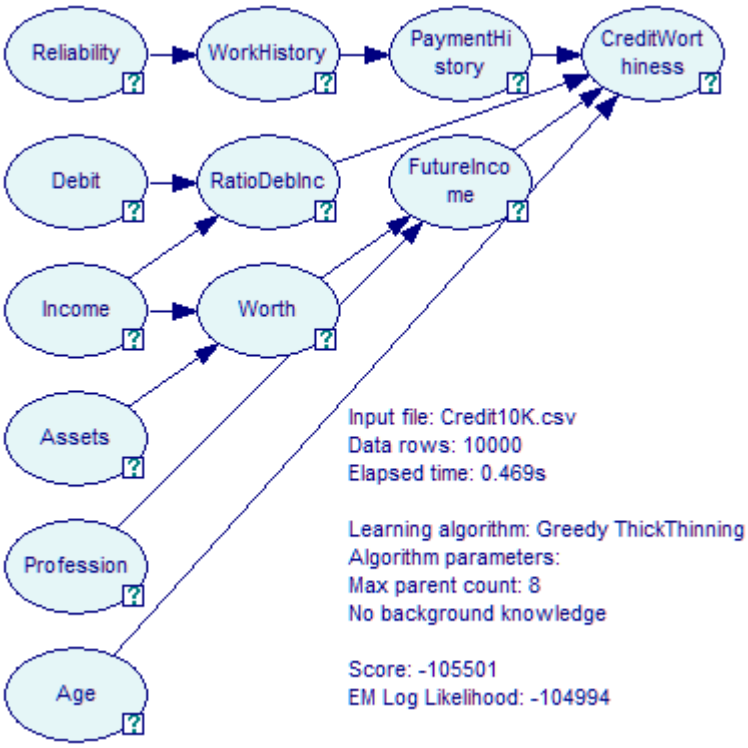
The *Greedy Thick Thinning* (*GTT*) structure learning algorithm is based on the *Bayesian Search* approach and has been described in (Cheng et al., 1997). *GTT* starts with an empty graph and repeatedly adds the arc (without creating a cycle) that maximally increases the marginal likelihood *P(D|S)* until no arc addition will result in a positive increase (this is the thickening phase). Then, it repeatedly removes arcs until no arc deletion will result in a positive increase in *P(D|S)* (this is the thinning phase). It is an approximate but a very fast algorithm that yields quite good structures. Here is the *Learn New Network* dialog for the *Greedy Thick Thinning* algorithm:



The *Greedy Thick Thinning* algorithm has only one parameter:

- *Max Parent Count* (default 8) limits the number of parents that a node can have. Because the size of conditional probability tables of a node grow exponentially in the number of the node's parents, it is a good idea to put a limit on the number of parents so that the construction of the network does not exhaust all available computer memory.

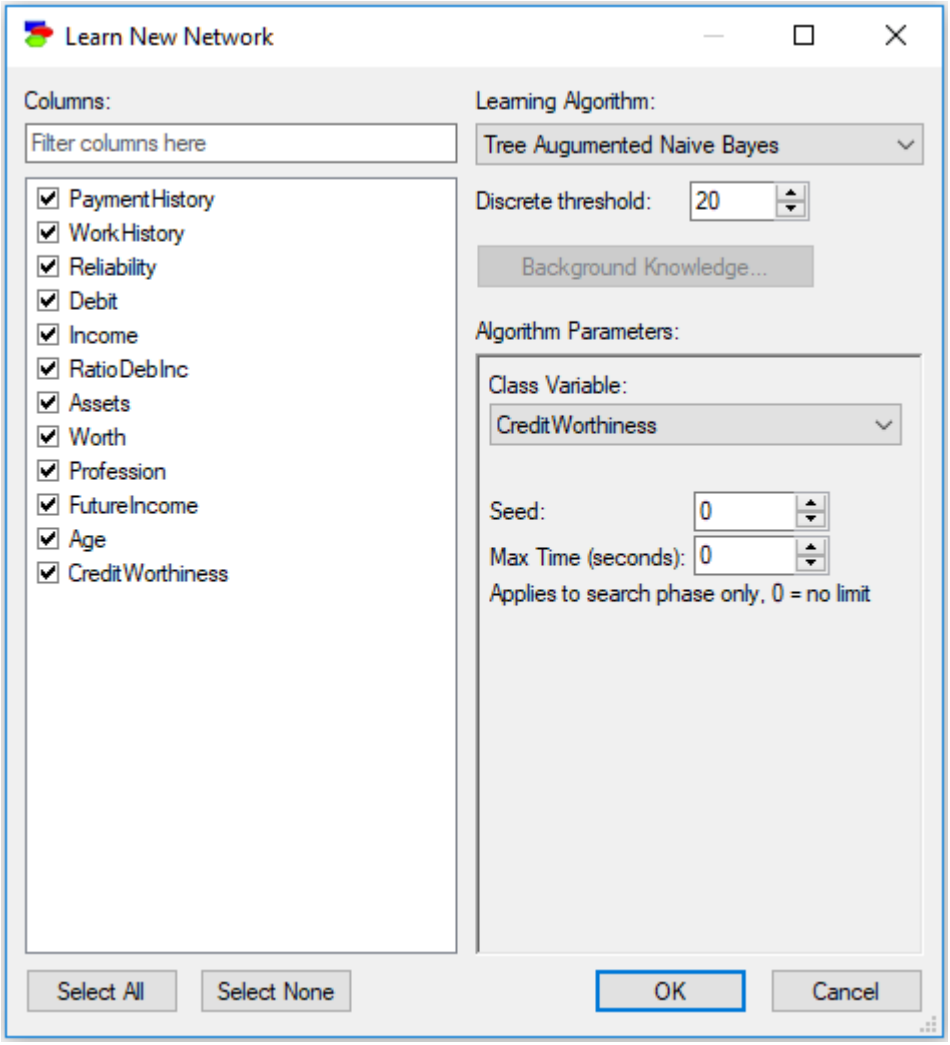
The algorithm produces an acyclic directed graph that gives the maximum score. The score is proportional to the probability of the data given the structure, which, assuming that we assign the same prior probability to any structure, is proportional to the probability of the structure given the data.



Because the *Greedy Thick Thinning* algorithm produces an acyclic directed graph, it is a good idea to investigate the theoretical limits to what it can identify based on the data. To this effect, we advise the user to transform the acyclic directed graph of a Bayesian network to the *Pattern Editor* dialog**.**

6.5.8.5 Tree Augmented Naive Bayes

The *Tree Augmented Naive Bayes* (*TAN*) structure learning algorithm is a semi-naive structure learning method based on the *Bayesian Search* approach, described and thoroughly evaluated in (Friedman et al., 1997). The *TAN* algorithm starts with a *Naive Bayes* structure (i.e., one in which the class variable is the only parent of all remaining, feature variables) and adds connections between the feature variables to account for possible dependence between them, conditional on the class variable. The algorithm imposes the limit of only one additional parent of every feature variable (additional to the class variable, which is a parent of every feature variable). Please note that the *Naive Bayes* structure assumes that the features are independent conditional on the class variable, which leads to inaccuracies when they are not independent. The *TAN* algorithm is simple and has been found to perform reliably better than *Naive Bayes*. Here is the *Learn New Network* dialog for the *TAN* algorithm:



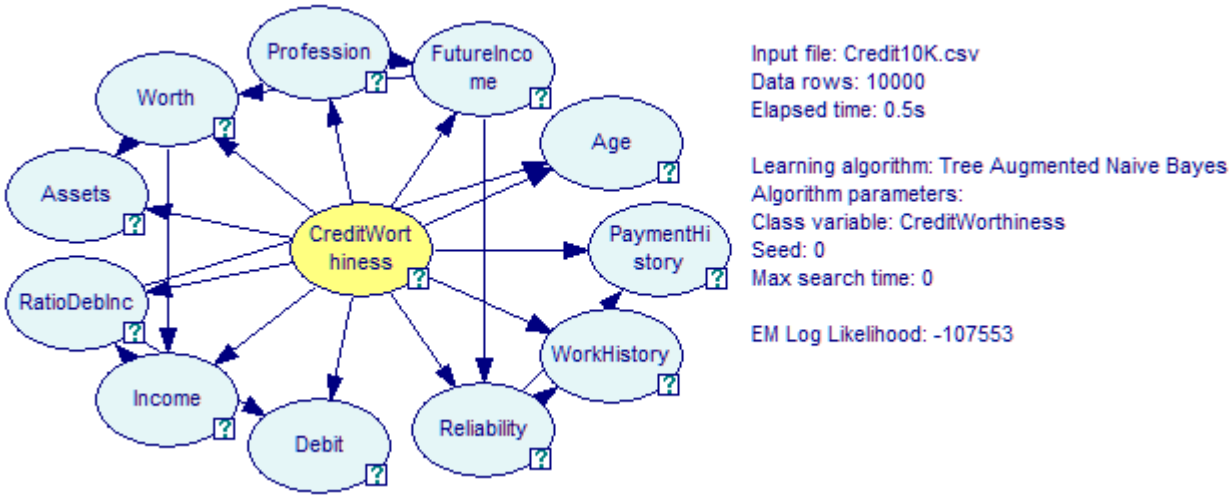
The *TAN* algorithm has the following parameters:

- *Class Variable*, which is a pop-up menu forcing the user to select one of the variables as the class variable. This is a crucial choice, as it determines the structure of the graph. Only those variables that are selected in the *Columns* list appear on the *Class Variable* pop-up menu.

- *Seed* (default 0), which is the initial random number seed used in the random number generator. If you want the learning to be reproducible (i.e., you want to obtain the same result every time you run the algorithm), use the same *Seed*. *Seed* equal to zero (the default) makes the random number generator really random by starting it with the current value of the processor clock.

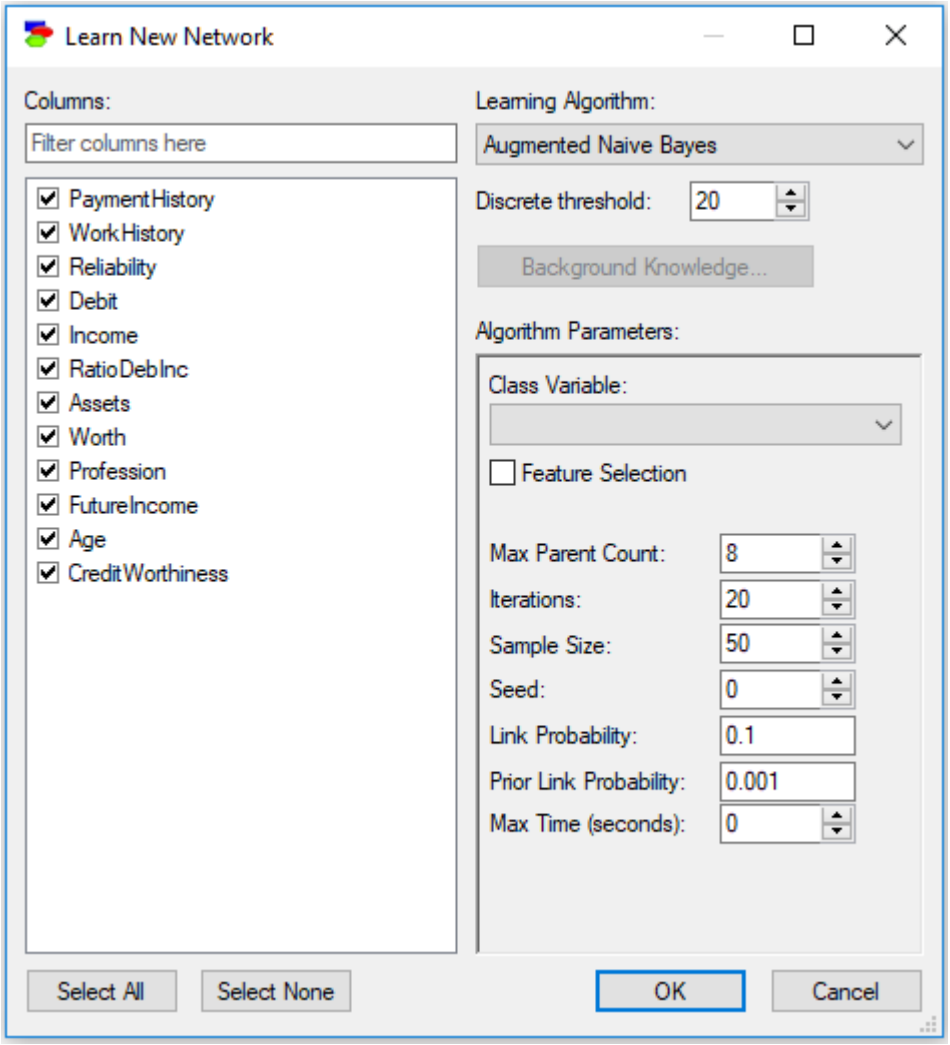
- *Max Time (seconds)* (default 0, which means no time limit) sets a limit on the run time of the search phase of the algorithm. It is a good idea to set a limit for any sizable data set so as to have the algorithm terminates within a reasonable amount of time, although it has to be said that the *TAN* algorithm is very simple and it is rather unheard of that it does not terminate within a reasonable amount of time.

The algorithm produces an acyclic directed graph with the class variable being the parent of all the other (feature) variables and additional connections between the feature variables. The structure learned is one with the maximum score, similarly to other algorithms based on *Bayesian Search*. The score is proportional to the probability of the data given the structure, which, assuming that we assign the same prior probability to any structure, is proportional to the probability of the structure given the data.



6.5.8.6 Augmented Naive Bayes

The *Augmented Naive Bayes* (*ANB*) structure learning algorithm is a semi-naive structure learning method based on the *Bayesian Search* approach, described and thoroughly evaluated in (Friedman et al., 1997). The *ABN* algorithm starts with a *Naive Bayes* structure (i.e., one in which the class variable is the only parent of all remaining, feature variables) and adds connections between the feature variables to account for possible dependence between them, conditional on the class variable. There is no limit on the number of additional connections entering each of the feature variable, unless it is imposed by one of the algorithm's parameters (*Max Parent Count*). Please note that this is the main difference between the ABN and the TAN algorithm, which sets the limit on the number of parents to 2. The *Naive Bayes* structure assumes that the features are independent conditional on the class variable, which leads to inaccuracies when they are not independent. The *ANB* algorithm is simple and has been found to perform reliably better than *Naive Bayes*. Here is the *Learn New Network* dialog for the *ANB* algorithm:



The *ANB* algorithm has a number of parameters, most of which mimic the parameters of the *Bayesian Search* algorithm that the *ANB* algorithm is based on:

- *Class Variable*, which is a pop-up menu forcing the user to select one of the variables as the class variable. This is a crucial choice, as it determines the structure of the graph. Only those variables that are selected in the *Columns* list appear on the *Class Variable* pop-up menu.

- *Feature Selection*, when checked, invokes an additional function that removes from the feature set those features that do not contribute enough to the classification.

- *Max Parent Count* (default 8) limits the number of parents that a node can have. Because the size of conditional probability tables of a node grow exponentially in the number of the node's parents, it is a good idea to put a limit on the number of parents so that the construction of the network does not exhaust all available computer memory. When *Max Parent Count* is equal to 2, the Augmented Naive Bayes algorithm deteriorates in principle to the Tree Augmented Naive Bayes algorithm, which imposes the maximum number of parents of the feature nodes equal to 2.

- *Iterations* (default 20) sets the number of restarts of the algorithm. Generally, the algorithm is searching through a hyper-exponential search space and its goal can be compared to searching for a needle in a haystack. Restarts allow for probing more areas of the search space and increase the chance of finding a structure that will fit the data better. We advise to make this number as large as we can afford it in terms of running time. The default number of iterations should give you an idea of how long the algorithm will take when the number of iteration is larger. The computing time is roughly linear in the number of iterations.

- *Sample size* (default 50) takes part in the score (*BDeu*) calculation, representing the inertia of the current parameters when introducing new data.

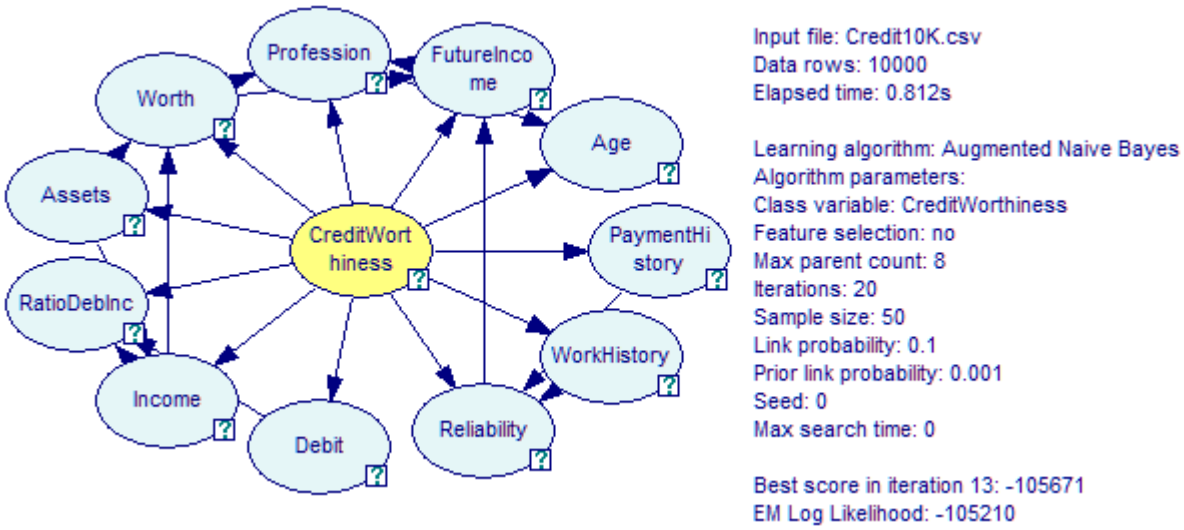
- *Seed* (default 0), which is the initial random number seed used in the random number generator. If you want the learning to be reproducible (i.e., you want to obtain the same result every time you run the algorithm), use the same *Seed*. *Seed* equal to zero (the default) makes the random number generator really random by starting it with the current value of the processor clock.

- *Link Probability* (default 0.1)

- *Prior Link Probability* (default 0.001)

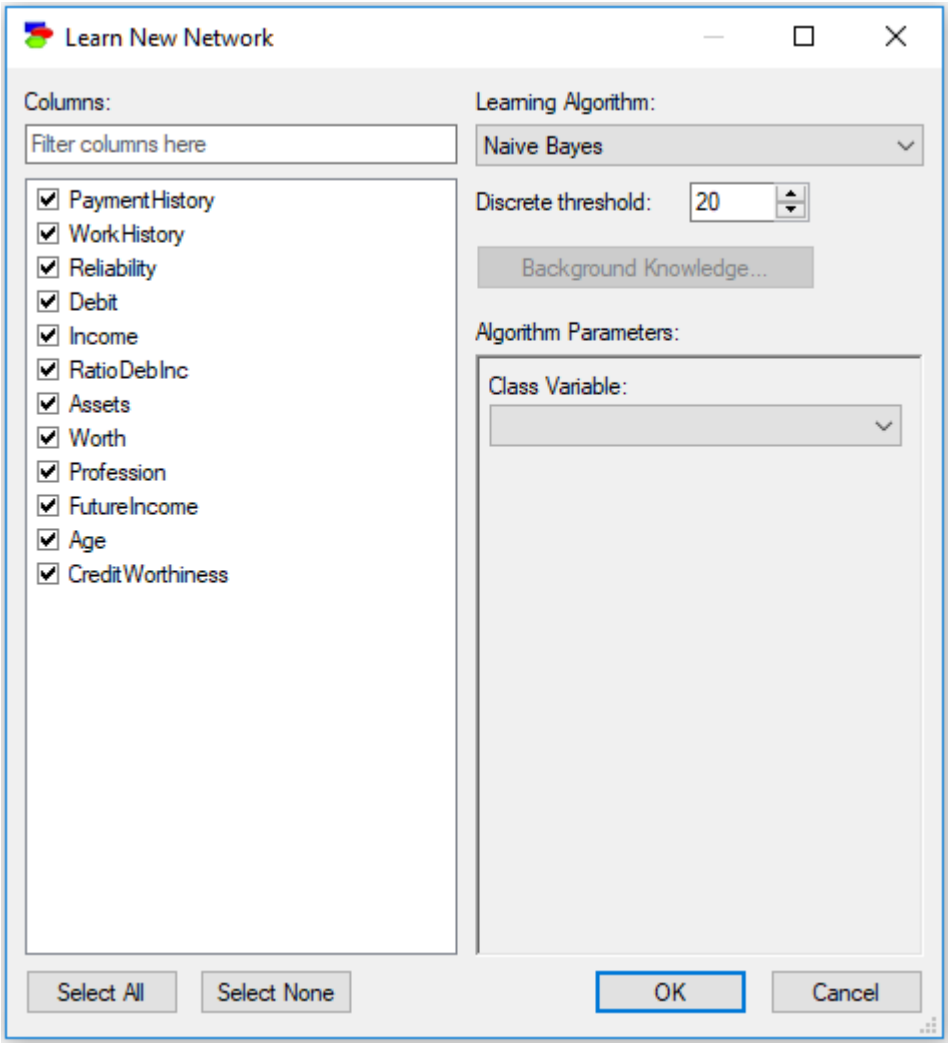
- *Max Time (seconds)* (default 0, which means no time limit) sets a limit on the run time of the search phase of the algorithm. It is a good idea to set a limit for any sizable data set so as to have the algorithm terminates within a reasonable amount of time, although it has to be said that the *ANB* algorithm is quite simple and it is rather unheard of that it does not terminate within a reasonable amount of time.

The algorithm produces an acyclic directed graph with the class variable being the parent of all the other (feature) variables and additional connections between the feature variables. The structure learned is one with the maximum score, similarly to other algorithms based on *Bayesian Search*. The score is proportional to the probability of the data given the structure, which, assuming that we assign the same prior probability to any structure, is proportional to the probability of the structure given the data.



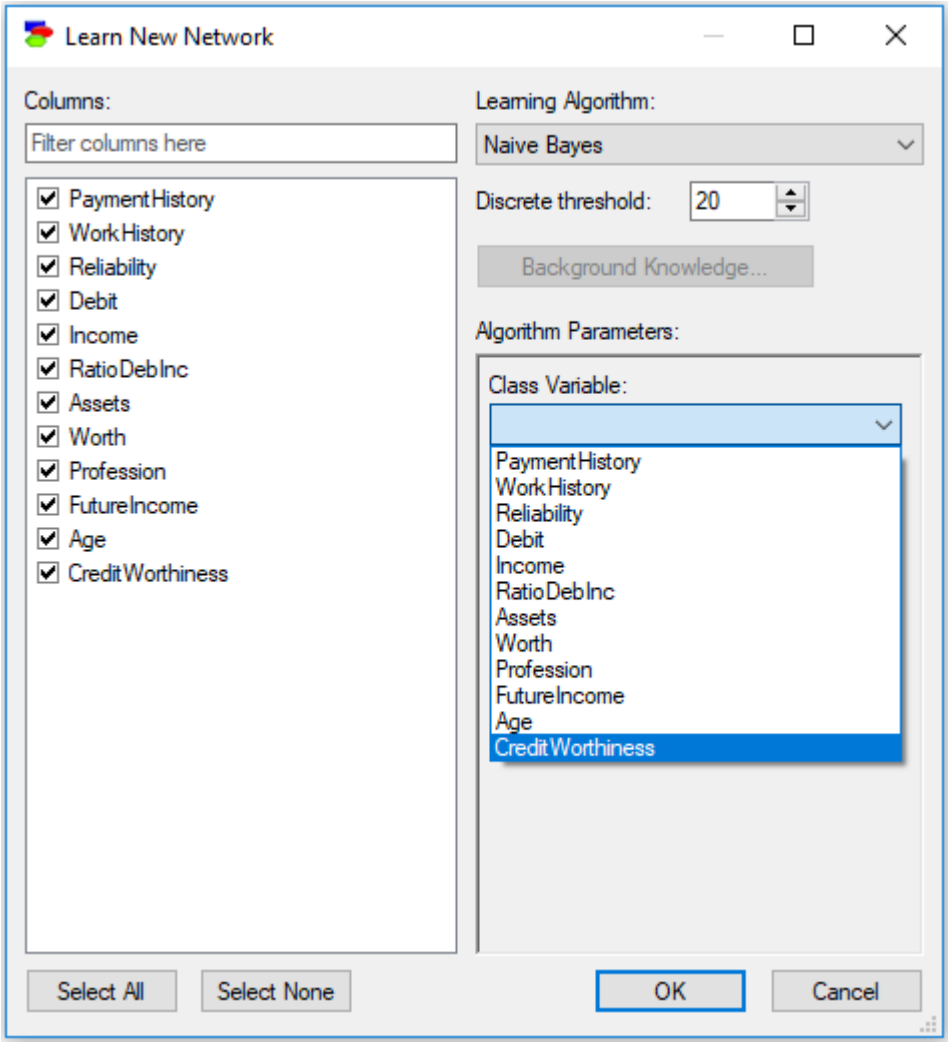
6.5.8.7 Naive Bayes

The *Naive Bayes* learning algorithm is a naive structure learning method that is included in the category of structure learning algorithms only because it creates a Bayesian network, including its structure and parameters directly from data. The structure of a Naive Bayes network is not learned but rather fixed by assumption: The class variable is the only parent of all remaining, feature variables and there are no other connections between the nodes of the network. The *Naive Bayes* structure assumes that the features are independent conditional on the class variable, which leads to inaccuracies when they are not independent. If you believe that this assumption does not hold, please try one of the improvements on the Naive Bayes algorithm, the *TAN* of the *ANB* algorithms. The *Naive Bayes* algorithm is incredibly simple and has been found to perform reasonably well, even for small data sets. Here is the *Learn New Network* dialog for the *Naive Bayes* algorithm:

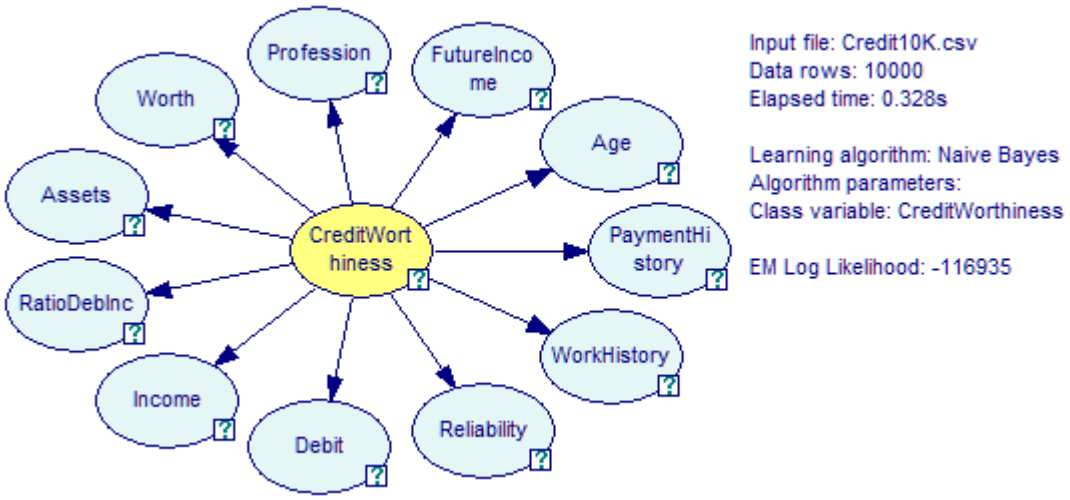


The *Naive Bayes* algorithm has only one algorithm specific parameter:

- *Class Variable*, which is a pop-up menu forcing the user to select one of the columns in the data file (variables) as the class variable. This is a crucial choice, as it determines the structure of the graph. Only those variables that are selected in the *Columns* list appear on the *Class Variable* pop-up menu.

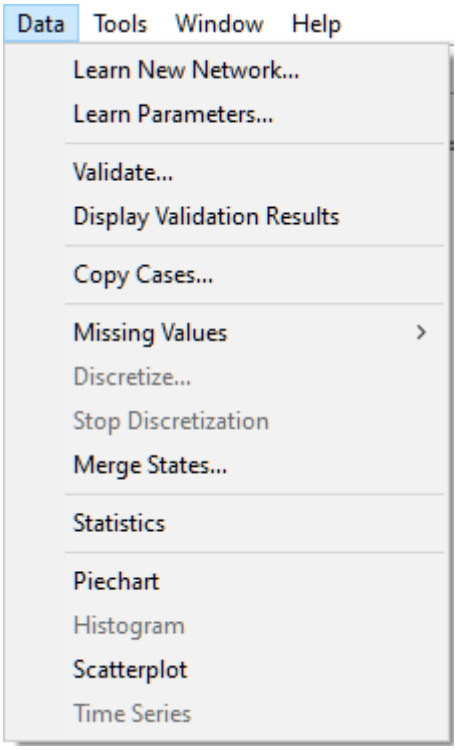


The algorithm produces an acyclic directed graph with the class variable being the parent of all the other (feature) variables and no other connections between the nodes. In case the *Feature Selection* option is checked, those nodes that are independent of the class variable are disconnected from it.

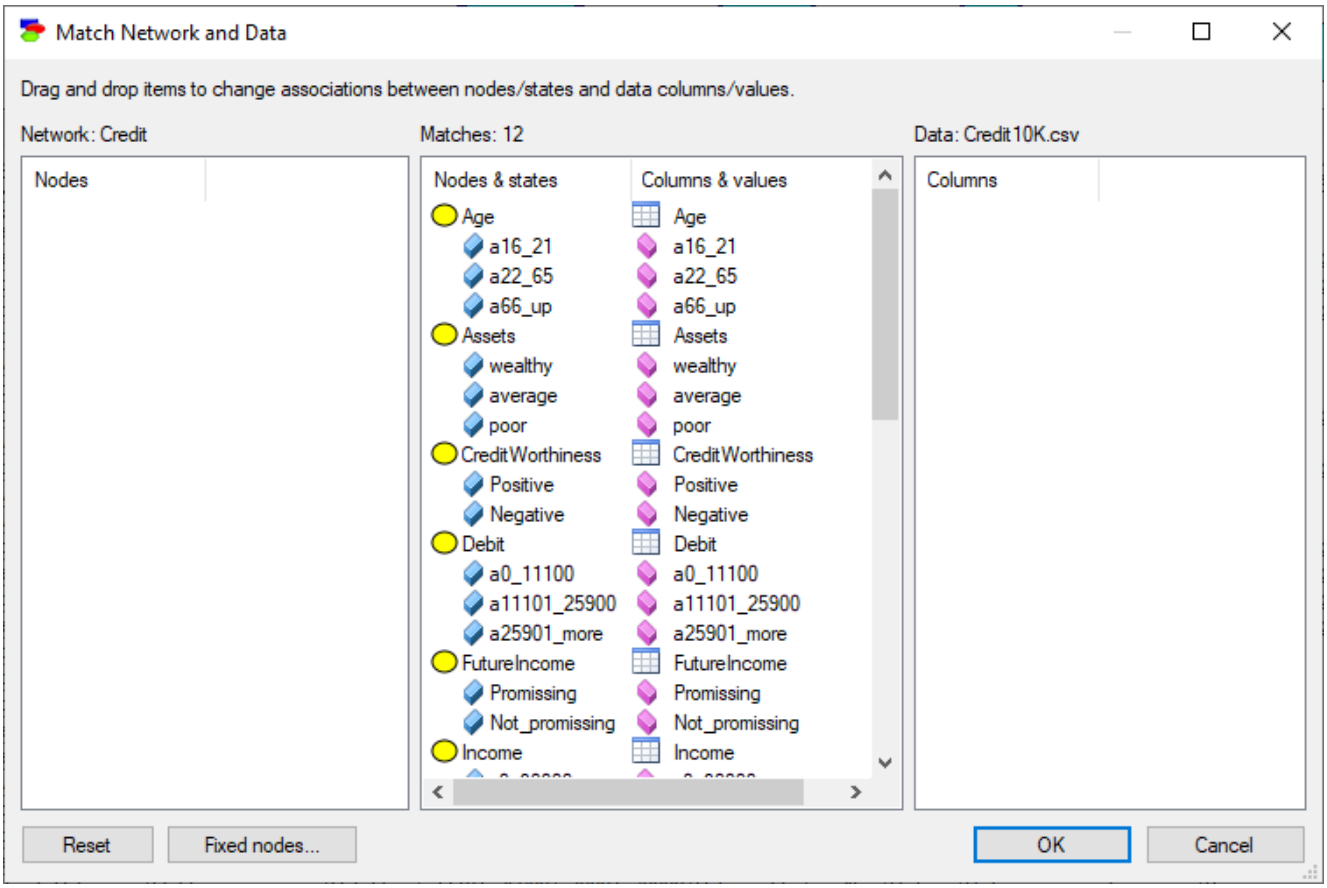


6.5.9 Learning parameters

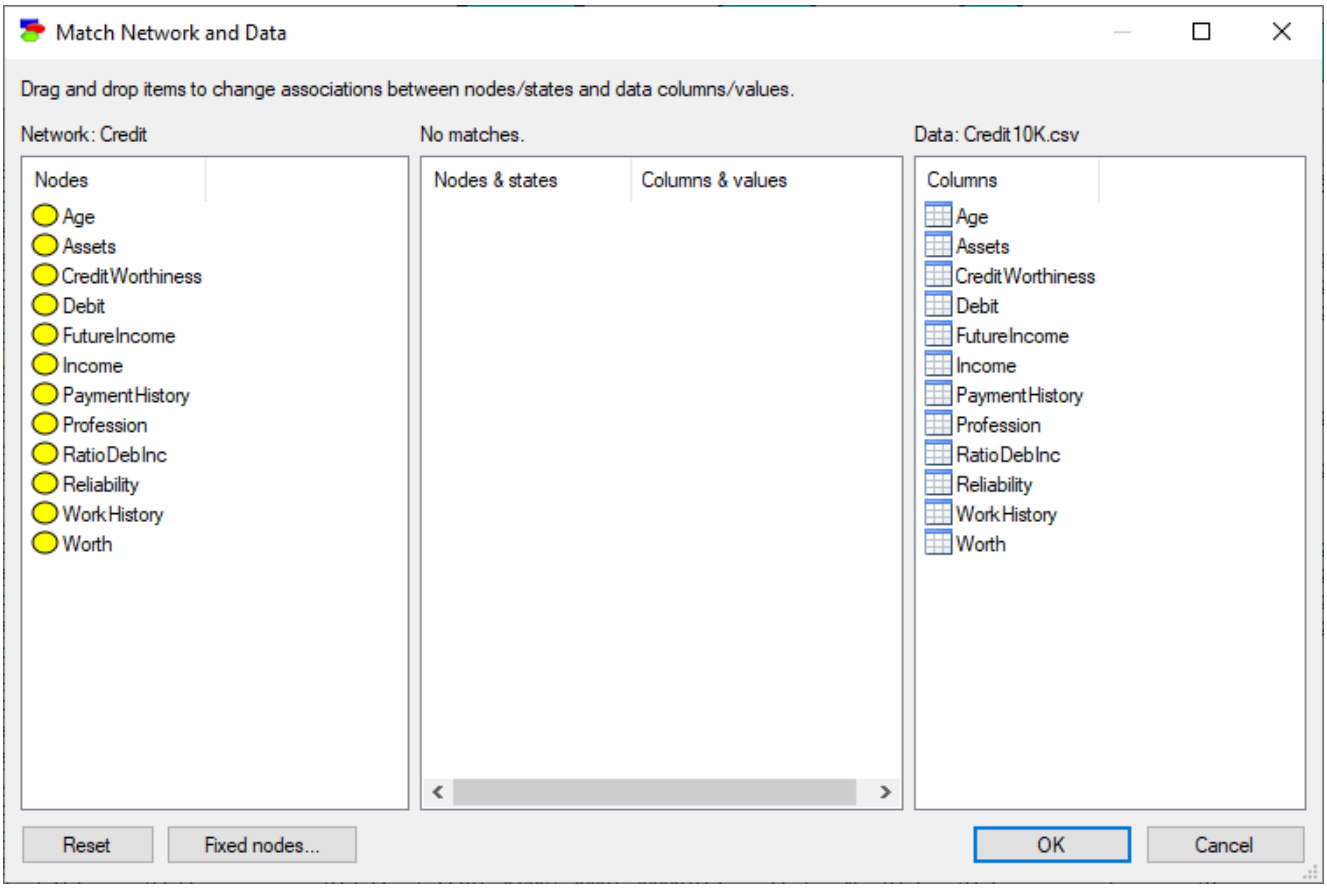
To learn parameters of an existing network (i.e., one for which the structure is already defined), you will need both, a data file and the network open. We will demonstrate the procedure of learning the parameters of a Bayesian network from data on the network *Credit.xdsl* and a data file *Credit10K.csv*. Both are available among the example files included with GeNIe. Once you have opened both, select *Data-Learn Parameters*...



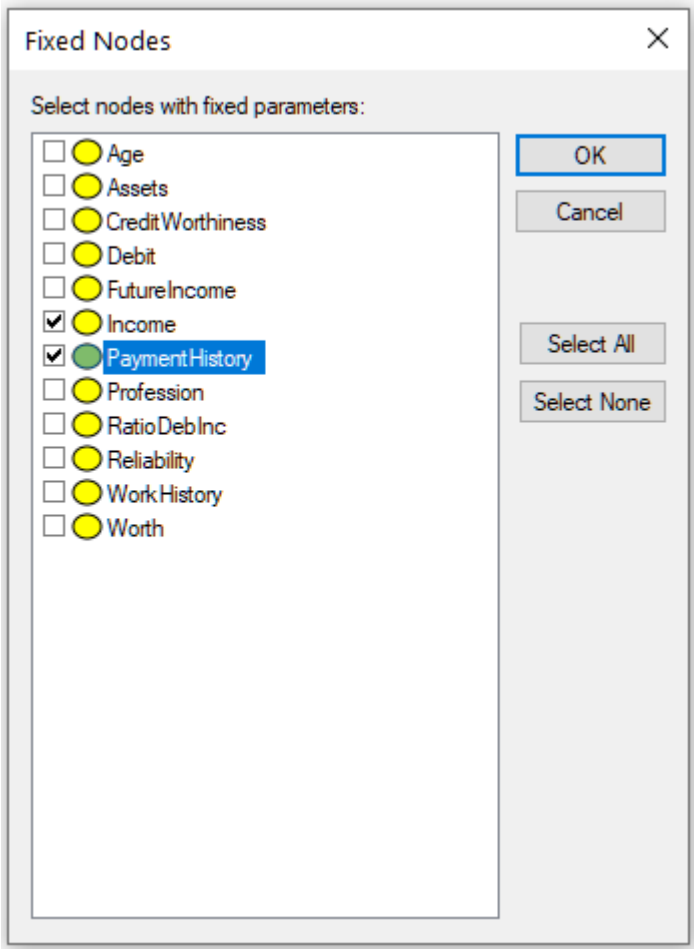
This will invoke the *Match Network and Data* dialog that serves to create a mapping between the variables defined in the network (left column) and the variables defined in the data set (right column).



Both lists of variables are sorted alphabetically. The *Match Network and Data* dialog does text pre-matching and places in the central column all those variables and their states that match (have identical or close to identical names - GeNIe is rather tolerant in this respect and, for example, ignores spaces and special characters). If there is any disparity between them, GeNIe highlights the differing labels by means of a yellow background. Manual matching between variables in the model and the data can be performed by dragging and dropping (both variables and their outcomes). To indicate that a variable (or its state in the middle column) in the model is the same as a variable in the data, simply drag-and-drop the variable (or its state in the middle column) from one to the other column. To start the matching process from scratch, use the *Reset* button, which will result in the following matching:

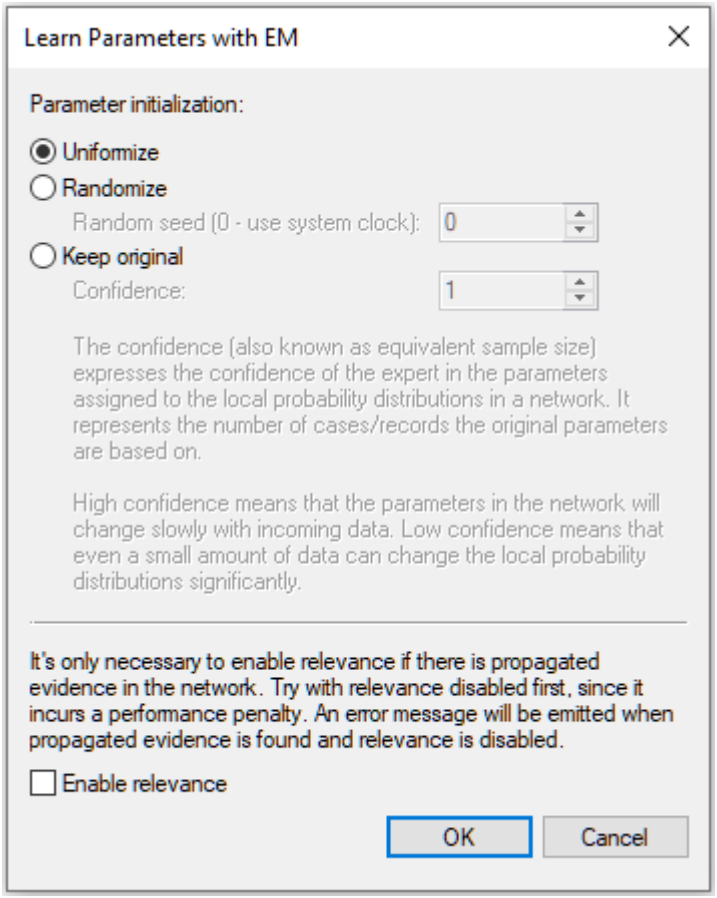


Fixed nodes... button invokes a dialog that allows for excluding nodes from the learning process:



Nodes selected in this dialog (in the dialog above, nodes *Income* and *PaymentHistory*) will not be modified by the learning process and will preserve their original CPT.

Once you have verified that the model and the data are matched correctly, press OK, which will bring up the following dialog:



In learning probability distributions, GeNIe uses the EM algorithm (Dempster et al., 1977; Lauritzen, 1995), which is capable of learning parameters from data sets that contain missing values (this is, unfortunately, often the case). The algorithm has several parameters:

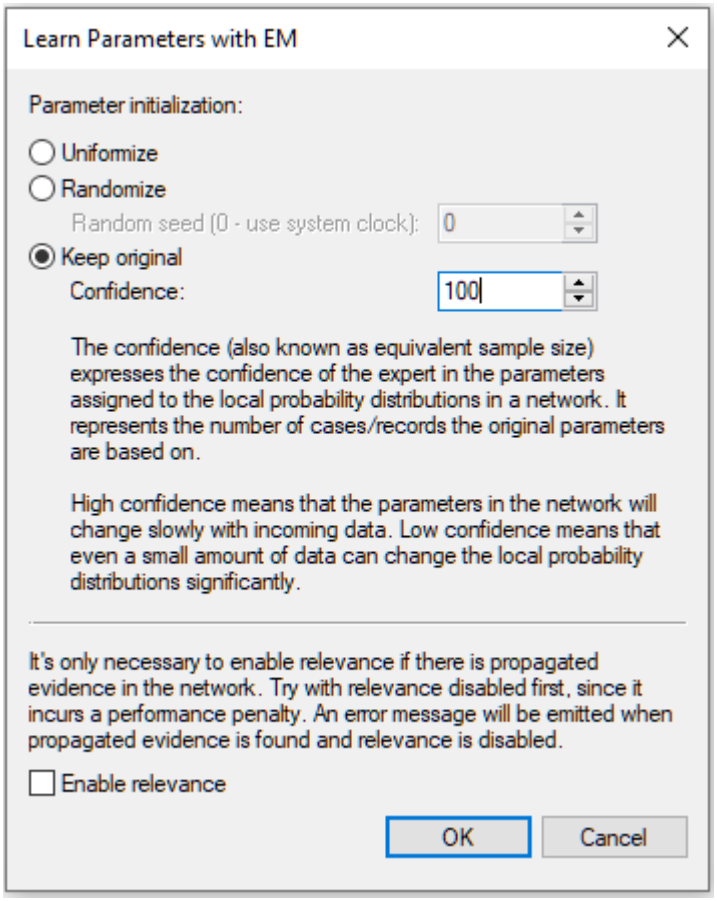
The *Parameter initialization* group allows for choosing one of the three possible starting points of the EM algorithm.

Uniformize, when set, causes the algorithm to start with all parameters in the network taken from the Uniform distribution. This is a typical option that one should use when one wants to disregard the existing parameters. The *Confidence* assigned by the algorithm in this case is equal to 1.

Randomize allows for picking random values for parameters, which inserts some randomness in the algorithm's search for the optimal values of parameters. Interestingly, uniformizing all distributions prior to learning does not necessarily lead to better quality parameters. Any way you set the initial values of the parameters will just provide a starting point in EM's search for the parameter set that maximizes the probability of data given the model. Using the *Randomize* option may be especially useful when learning parameters with latent variables. The EM algorithm is more likely in such cases to avoid the local maximum around uniform distributions. *Random number seed* is the initial seed passed to the random number generator. Using the same seed each time you perform learning makes the results perfectly reproducible, unless the seed is equal to zero (the default value), in which case GeNIe uses the system clock as the seed and the random number sequence is really random.

Keep original allows for starting with the original parameters. This option should be used only if we use the new data set as an additional source of information over the existing network. Keeping the original parameters and learning from the same data file that they were extracted from will lead to over-fitting the data.

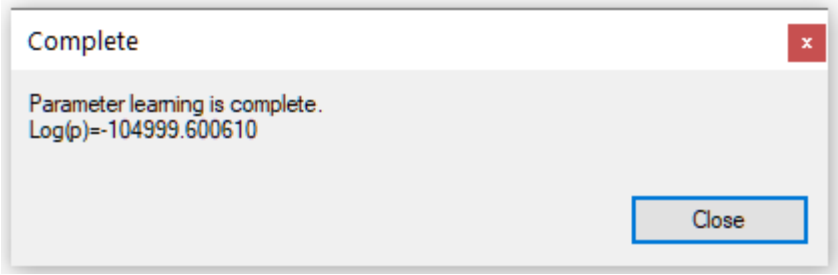
When keeping the original probabilities in the network (*Keep original* option), *Confidence* becomes important. *Confidence* is also known as the equivalent sample size (ESS), which can be interpreted as the number of records that the current network parameters are based on. The interpretation of this parameter is obvious when the entire network or its parameters have been learned from data - it should be equal to the number of records in the data file from which they were learned. The *Confidence* in the screen shot below is set to 100.



When the parameters in the network have been elicited from an expert, we can view them as the number of cases that the expert has seen before providing us with the current parameters. The larger the ESS, the less weight is assigned to the new cases, which gives a mechanism for gentle refinement of model numerical parameters. ESS expresses the confidence of the expert in the parameters assigned to the local probability distributions in the network. High confidence means that the parameters in the network will change slowly with incoming data. Low confidence means that even a small amount of data can change the local probability distributions significantly. In establishing a value for ESS, we advise to reflect on the number of records/cases on which the current parameters are based. This will naturally combine with the number of new records, a quantity that is known by the algorithm in the learning process.

Enable relevance option makes the algorithm faster by speeding up the Bayesian inference part. We suggest that this be checked only in the rare cases that the algorithm takes a long time.

Once we press *OK*, the EM algorithm updates the network parameters following the options chosen and comes back with the following dialog:



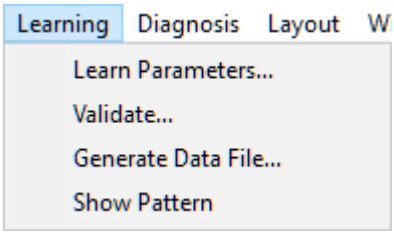
Log(p), ranging from minus infinity to zero, is a measure of fit of the model to the data.

A remark on the network structure and also on existing evidence. Learning parameters functionality focuses on learning parameters, not the structure, which is assumed fixed and will be unaffected. Existing evidence in the network is ignored and has no effect on the learned parameters.

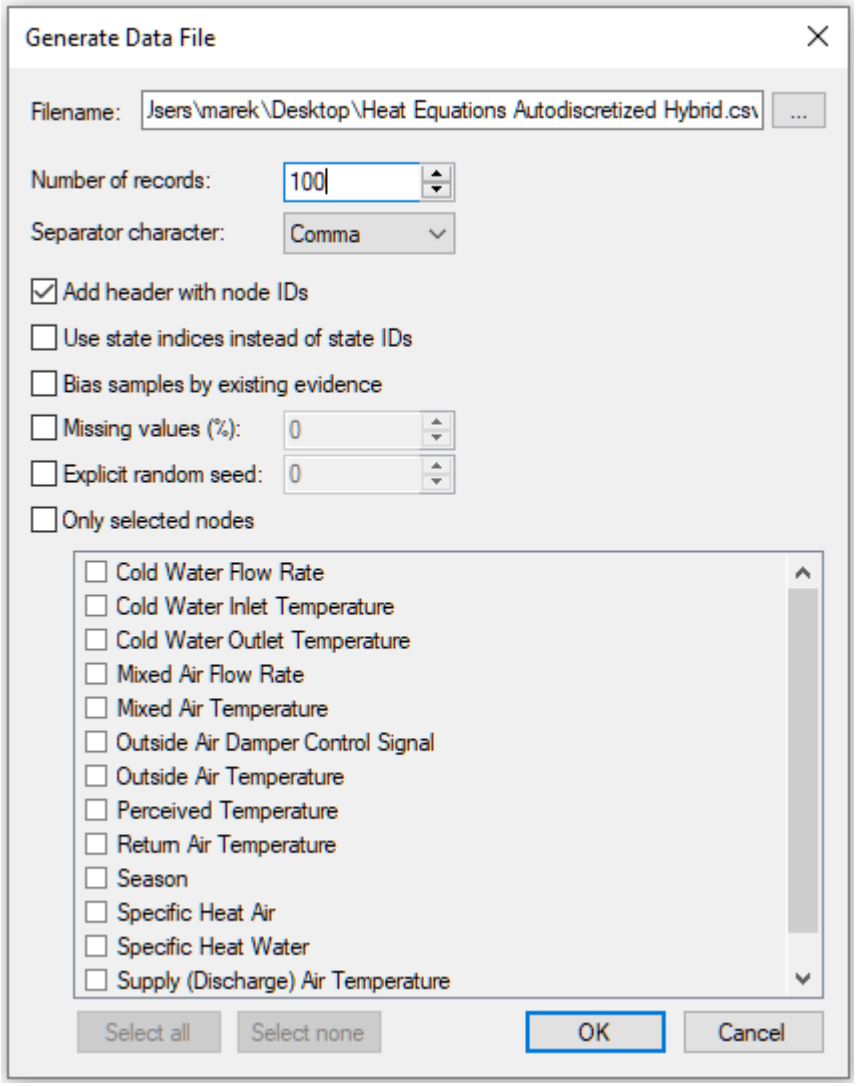
Finally, a remark on a limitation in learning parameters of continuous, multi-variate Gaussian models. Our implementation of the EM algorithm in this case does not allow for missing values. An extension of this implementation is on our development agenda, so please stay tuned!

6.5.10 Generating a data file

A Bayesian network model is a representation of the joint probability distribution over its variables. Given this distribution, it is sometimes useful to generate a data file from it. Such data file can be subsequently used, for example, to test a learning algorithm. GeNIe allows for generating a data file from a model through the *Generate Data File...* command from the *Learning* menu.



The following dialog appears

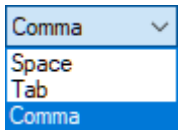


The *Generate Data File* command generates a text file containing records that are representative for the network in the sense of coming from a joint probability distribution modeled by the network. The individual records of the output file contain values of the nodes randomly generated from the joint distribution modeled by the network.

Filename specifies the location for the data file to be stored. Browse (...) button invokes *Save As* dialog, which helps with finding a location to save the file.

Number of records specifies the number of records to be generated.

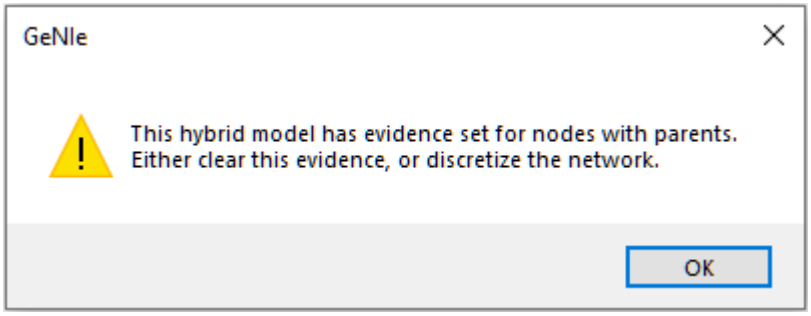
Separator character allows for selecting a character that separates individual node states in records. The choices are *Space*, *Tab* and *Comma*.



Add header with node ID's, when checked, makes the first record of the output file contain IDs of the nodes. If the file is to be read into GeNIe, this option should be checked.

Use state indices instead of state ID's leads to saving records with the state indices (0, 1 , 2, etc.) for discrete variables instead of state IDs or state names.

Bias samples by existing evidence, when checked, generates a data file from the posterior joint probability distribution (i.e., biased by the observations) rather than from the original joint probability distribution. This option will not work for continuous or hybrid networks with evidence in nodes with parents. In such cases, GeNIe will signal an error



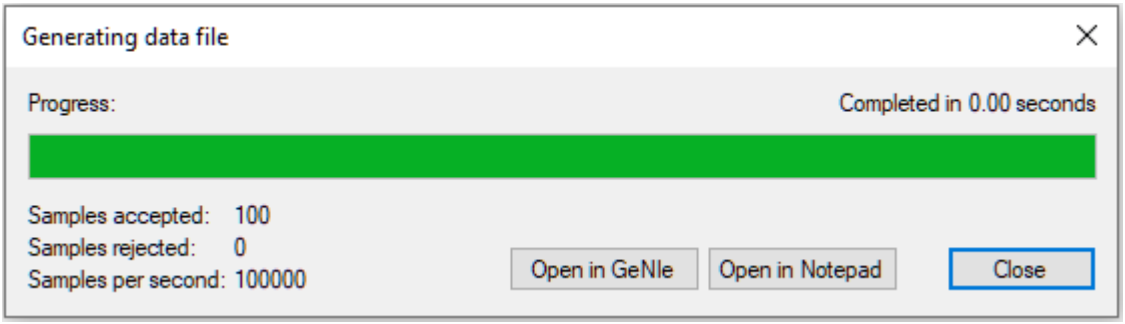
Missing values (%), when checked, produces an output file with missing values. The values are *Missing At Random* (this is also known as the *MAR* assumption). Percentage specifies the percentage of values missing.

Explicit random seed allows for reproducibility of the record generation process. Unchecked or checked with zero random number seed (default) leads to using the system clock, which means that the records generated are truly random.

Only selected nodes, when enabled, allows for selective contents of the output file. The user can, in this case, select nodes from the list in the window pane below, to generate records comprising of only the selected nodes. With the option enabled, two buttons, *Select all* and *Select none*, help with the selection process by allowing to select all nodes or clearing the selection, respectively.

It is also possible to select nodes in the *Graph* view before invoking the *Generate Data File* command. In this case, the nodes selected in the *Graph* view will be selected in the list of selected nodes. If nothing is selected in the *Graph* view, GeNIe recreates the last selection used in generating a data file.

Pressing the *OK* button starts the generation process, which ends with the following dialog



In addition to reporting the time taken by the generation process, the number of samples generated (these are divided into *Samples accepted* and *Samples rejected*, which becomes of essence only when generating a data file biased by observed evidence - in this case samples incompatible with the evidence are rejected), and generation speed (*Samples per second*), the dialog allows for opening the newly generated file in GeNIe and in Notepad.

Open in GeNIe shows the newly generated file in a GeNIe *Data View* window.

Season	Toa	u_d	Tra	Tma	m_flow_ma	sp_heat_air	Tsa	mdot_cw	sp_heat_water	T_cw_in	T_cw_out	Perceived_Temperature
Winter	-3.81616	0.2	24	18.4368	4.67	1.006	14.9562	2.11763	4.187	6.67	8.51421	Cold
Summer	26.3714	1	24	26.3714	4.56	1.006	15.2605	1.86406	4.187	6.67	13.2005	Warm
Fall	8.0615	1	24	8.0615	4.67	1.006	14.8786	4.09328	4.187	6.67	4.80129	Warm
Fall	10.8029	1	24	10.8029	4.67	1.006	14.9836	2.39149	4.187	6.67	4.7085	Warm
Fall	12.2824	1	24	12.2824	4.56	1.006	15.1079	2.23681	4.187	6.67	5.28604	Warm
Fall	6.90302	0.2	24	20.5806	4.56	1.006	14.9691	2.15605	4.187	6.67	9.52154	Warm
Fall	10.3377	1	24	10.3377	4.56	1.006	15.1132	2.03699	4.187	6.67	4.10144	Warm
Fall	16.5907	0.2	24	22.5181	4.56	1.006	14.9091	1.67217	4.187	6.67	11.6555	Warm
Winter	-0.803386	1	24	-0.803386	4.56	1.006	15.1517	2.6199	4.187	6.67	-0.00231027	Warm
Spring	16.5504	0.2	24	22.5101	4.56	1.006	14.9321	1.54953	4.187	6.67	12.0282	Warm
Spring	13.0857	0.2	24	21.8171	4.67	1.006	14.943	2.75	4.187	6.67	9.47478	Warm
Spring	21.1045	0.2	24	23.4209	4.56	1.006	15.311	3.36008	4.187	6.67	9.3144	Warm
Summer	34.9542	0.2	24	26.1908	4.56	1.006	14.8908	1.94729	4.187	6.67	13.0278	Hot
Spring	7.51965	0.2	24	20.7039	4.67	1.006	14.869	3.06073	4.187	6.67	8.80907	Hot
Fall	20.6666	1	24	20.6666	4.56	1.006	14.8215	2.76371	4.187	6.67	8.98717	Warm
Summer	19.6751	0.2	24	23.135	4.67	1.006	15.231	2.36775	4.187	6.67	10.4156	Hot
Winter	0.741886	1	24	0.741886	4.56	1.006	15.0559	1.99904	4.187	6.67	-1.17516	Warm
Summer	26.4214	1	24	26.4214	4.56	1.006	14.8238	3.67887	4.187	6.67	10.1239	Hot
Fall	10.5777	1	24	10.5777	4.67	1.006	14.8695	2.17756	4.187	6.67	4.45856	Hot
Fall	13.2602	1	24	13.2602	4.67	1.006	15.0668	2.92328	4.187	6.67	5.97657	Warm
Summer	27.594	1	24	27.594	4.56	1.006	14.9794	1.12801	4.187	6.67	18.9224	Hot
Fall	8.56677	1	24	8.56677	4.67	1.006	14.8176	1.17815	4.187	6.67	0.716864	Warm

Please note that the network used in this example was hybrid and this is reflected in the data, which contains two columns with categorical values and the remainder of the columns numerical, reflecting the types of variables in the model.

Please remember about checking the *Add header with node ID's* option in the *Generate Data File* dialog. If this is not checked, the generated file will not conform to GeNIe requirement that the first record in the file contain variable names. The same file opened in Notepad looks as follows

Heat Equations Autodiscretized Hybrid.csv - Notepad

File Edit Format View Help

```
Season,Toa,u_d,Tra,Tma,m_flow_ma,sp_heat_air,Tsa,mdot_cw,sp_heat_water,T_cw_in,T_cw
_out,Perceived_Temperature
Spring,12.1397,0.2,24,21.6279,4.67,1.006,15.1655,2.64599,4.187,6.67,9.41046,Warm
Spring,0.949313,1,24,0.949313,4.56,1.006,14.8248,2.55254,4.187,6.67,0.71427,Warm
Winter,-6.83879,1,24,-6.83879,4.56,1.006,15.1538,2.93203,4.187,6.67,-1.54805,Warm
Spring,13.9773,0.2,24,21.9955,4.67,1.006,15.2322,1.7216,4.187,6.67,11.0779,Warm
Winter,4.81518,1,24,4.81518,4.67,1.006,15.2216,2.35863,4.187,6.67,1.71943,Cold
Fall,13.9334,1,24,13.9334,4.67,1.006,14.9033,2.37106,4.187,6.67,6.21103,Warm
Fall,8.53831,1,24,8.53831,4.67,1.006,15.2693,2.99693,4.187,6.67,4.14993,Warm
Summer,30.7852,0.2,24,25.357,4.56,1.006,14.8355,2.25875,4.187,6.67,11.7735,Hot
Summer,29.7299,1,24,29.7299,4.67,1.006,15.2445,3.6381,4.187,6.67,11.1375,Hot
Summer,15.4366,0.2,24,22.2873,4.67,1.006,15.1815,3.12555,4.187,6.67,9.22094,Hot
Fall,13.472,0.2,24,21.8944,4.56,1.006,15.0792,2.38316,4.187,6.67,9.80318,Cold
```

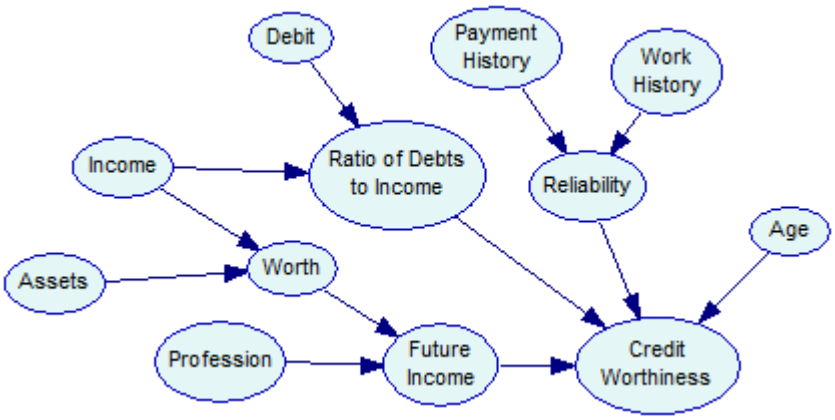
6.5.11 Validation

An crucial element of learning is validation of the results. We will show it on an example data set and a Bayesian network model learned from this data set.

Suppose we have a file `Credit10K.csv` consisting of 10,000 records of customers collected at a bank (this data file is not included in the distribution but can be easily generated from the model `Credit.xdsl`, available among the example models, through the *Generate Data File* functionality, described in the *Generating a data file* section). Each of these customers was measured on several variables, *Payment History*, *Work History*, *Reliability*, *Debit*, *Income*, *Ratio of Debts to Income*, *Assets*, *Worth*, *Profession*, *Future Income*, *Age* and *Credit Worthiness*. The first few records of the file (included among the example files with GeNIe distribution) look as follows in GeNIe:

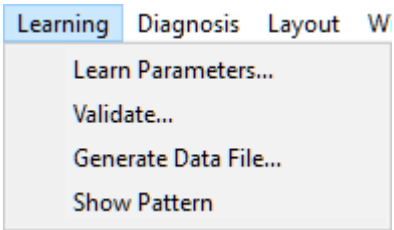
PaymentHistory	WorkHistory	Reliability	Debit	Income	RatioDebInc	Assets	Worth	Profession	FutureIncome	Age	CreditWorthiness
Without_Reference	Unstable	Unreliable	a0_11100	s70001_more	Favorable	wealthy	High	Medium_income_profession	Promissing	a16_21	Negative
Aceptable	Unjustified_no_work	Unreliable	a0_11100	s70001_more	Favorable	average	High	Medium_income_profession	Promissing	a66_up	Negative
Aceptable	Unstable	Reliable	a25901_more	s30001_70000	Unfavorable	wealthy	High	Low_income_profession	Not_promissing	a16_21	Negative
Excellent	Unstable	Reliable	a25901_more	s30001_70000	Unfavorable	average	Medium	Medium_income_profession	Not_promissing	a16_21	Negative
Excellent	Unjustified_no_work	Unreliable	a11101_25900	s0_30000	Unfavorable	average	Low	Medium_income_profession	Not_promissing	a66_up	Negative
Without_Reference	Stable	Reliable	a0_11100	s30001_70000	Favorable	average	High	Medium_income_profession	Promissing	a16_21	Positive
NoAceptable	Stable	Unreliable	a0_11100	s70001_more	Favorable	wealthy	High	Medium_income_profession	Promissing	a66_up	Positive
Excellent	Stable	Reliable	a0_11100	s70001_more	Favorable	wealthy	High	Low_income_profession	Promissing	a66_up	Positive
Excellent	Stable	Reliable	a25901_more	s70001_more	Unfavorable	poor	High	Low_income_profession	Not_promissing	a16_21	Negative
NoAceptable	Stable	Unreliable	a0_11100	s30001_70000	Favorable	average	Medium	Medium_income_profession	Promissing	a22_65	Positive
Without_Reference	Justified_no_work	Reliable	a25901_more	s70001_more	Unfavorable	poor	High	Low_income_profession	Not_promissing	a16_21	Negative
NoAceptable	Unstable	Unreliable	a25901_more	s30001_70000	Unfavorable	wealthy	High	Medium_income_profession	Promissing	a16_21	Negative
NoAceptable	Justified_no_work	Unreliable	a25901_more	s30001_70000	Unfavorable	wealthy	High	High_income_profession	Promissing	a22_65	Negative
Excellent	Stable	Reliable	a11101_25900	s0_30000	Unfavorable	average	Low	Medium_income_profession	Not_promissing	a16_21	Negative
Aceptable	Stable	Unreliable	a25901_more	s0_30000	Unfavorable	wealthy	Medium	Medium_income_profession	Not_promissing	a66_up	Negative
Without_Reference	Unjustified_no_work	Unreliable	a0_11100	s0_30000	Favorable	poor	Low	Low_income_profession	Not_promissing	a66_up	Positive
Aceptable	Unstable	Reliable	a11101_25900	s30001_70000	Unfavorable	average	Medium	Low_income_profession	Not_promissing	a66_up	Negative
Without_Reference	Unstable	Unreliable	a0_11100	s30001_70000	Unfavorable	average	High	Medium_income_profession	Promissing	a16_21	Negative

Supposed we have learned or otherwise constructed a Bayesian network model that aims at capturing the joint probability distribution over these variables. The main purpose of constructing this model is to be able to predict *Credit Worthiness* of a new customer applying for credit. If this customer comes from the same population as previous customers, we should be able to estimate the probability of *Positive Credit Worthiness* based on the new customer's characteristics. Let the following be the model (it is actually model `Credit.xdsl`, available among the example models):

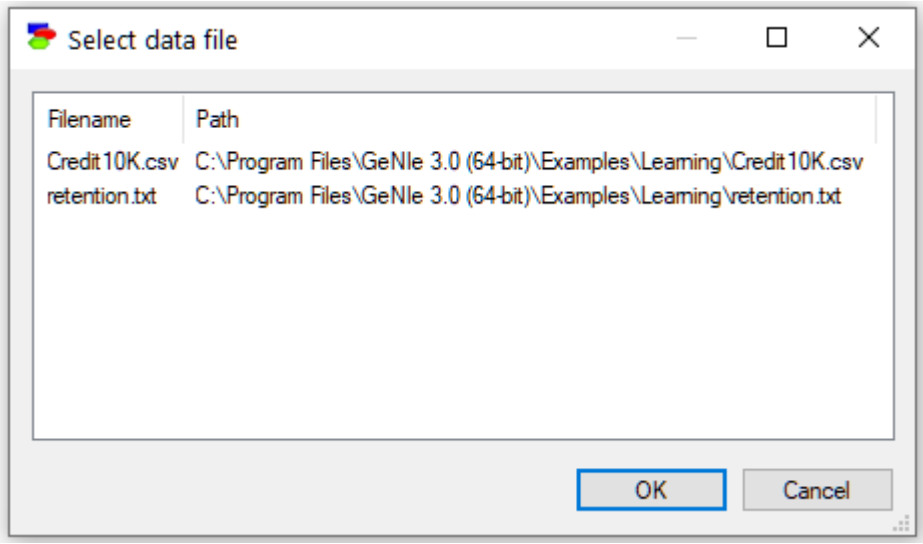


Running validation

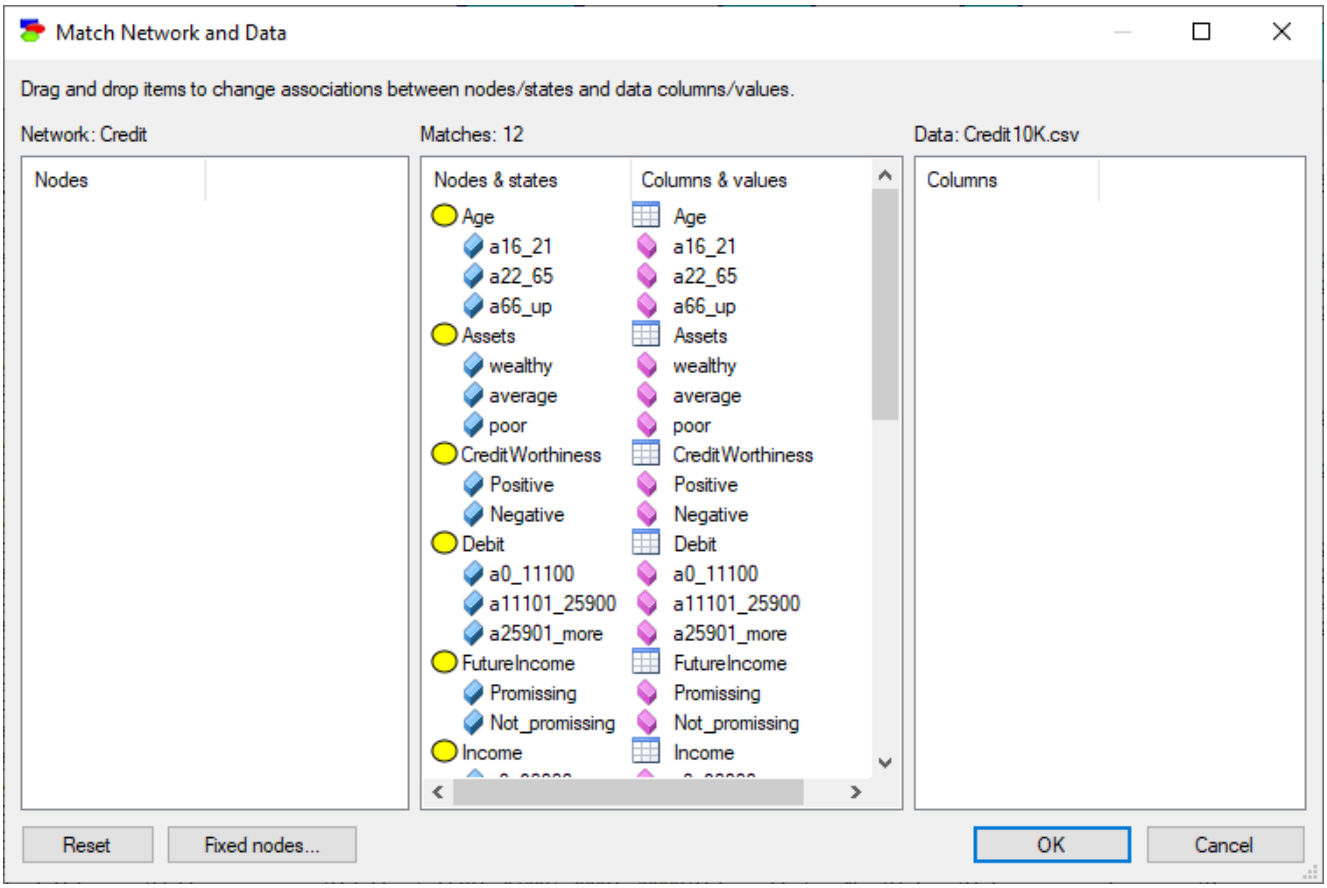
We can perform evaluation of the model by choosing *Validate...* from the *Learning* menu



If only one model and one data file are open, it is clear that we want to evaluate the model with the data file. If there are more than one data file open in GeNIe, the following dialog pops up

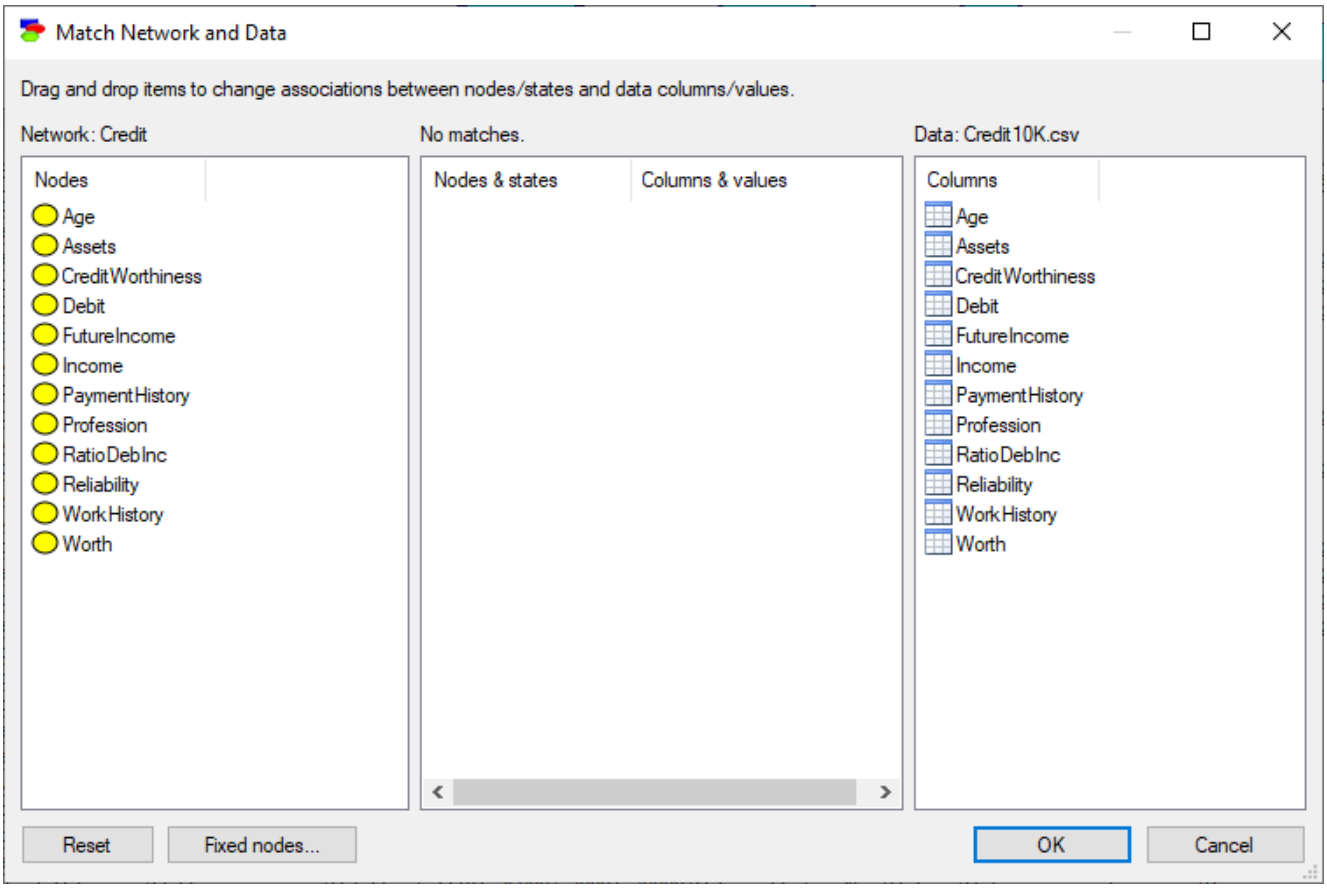


After selecting the file `Credit10K.csv`, we proceed with the following (*Match Network and Data*) dialog, whose only function is to make sure that the variables and states in the model (left column) are mapped precisely to the variables defined in the data set (right column). This dialog is identical to the dialog appearing when learning model parameters from data.

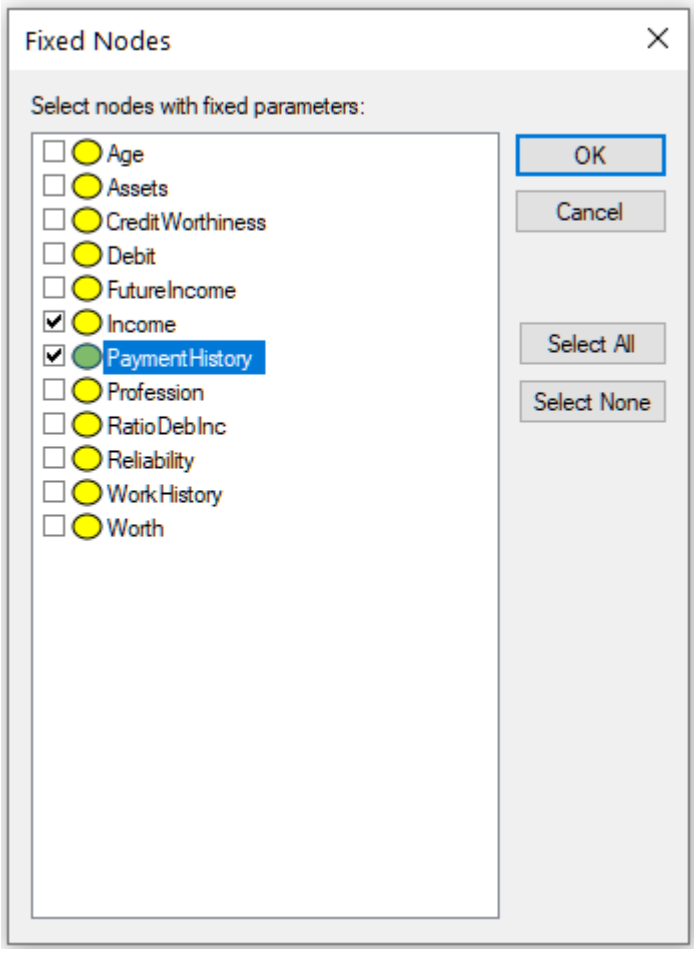


Both lists of variables are sorted alphabetically. The *Match Network and Data* dialog does text pre-matching and places in the central column all those variables and their states that match (have identical or close to identical names - GeNIe is rather tolerant in this respect and, for example, ignores spaces and special characters). If there is any disparity between them, GeNIe highlights the differences by means of a yellow background, which makes it easy to identify disparities. Manual matching between variables in the model and the data is performed by dragging and dropping. To indicate that a variable in the model is the same as a variable in the data, simply drag-and-drop the variables from one to the other column.

To start the matching process from scratch, use the *Reset* button, which will result in the following matching:

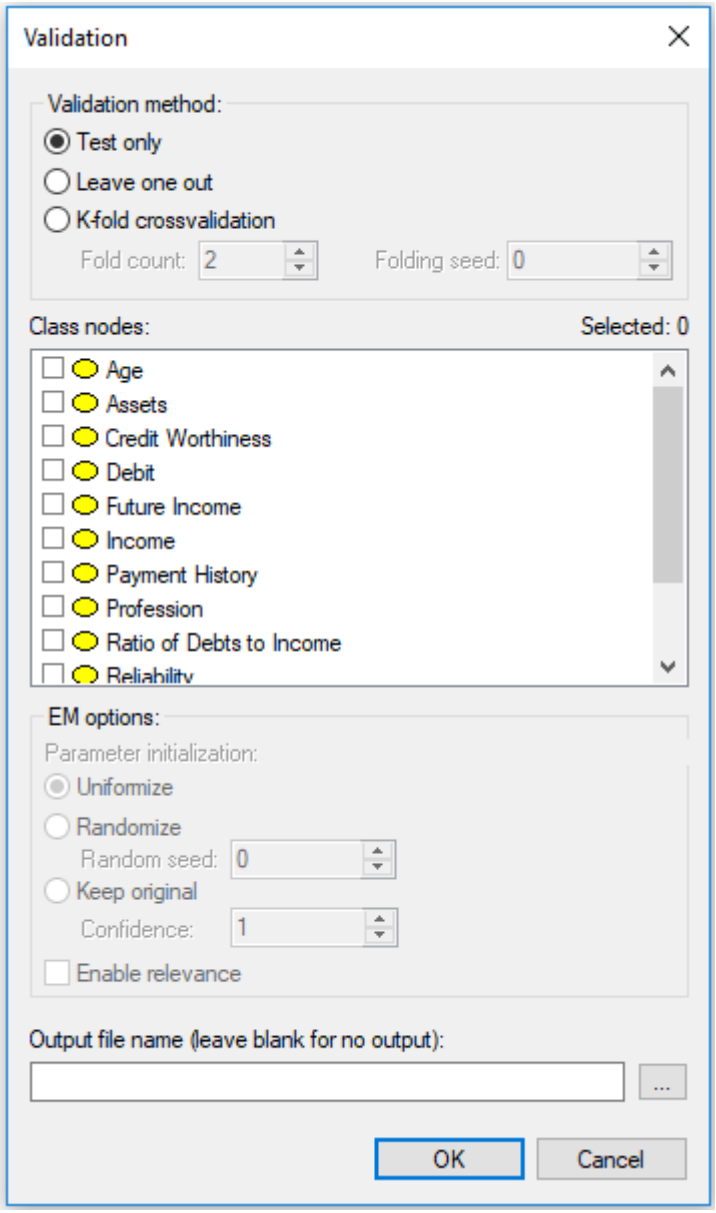


Fixed nodes... button invokes a dialog that allows for excluding nodes from the learning process in cross-validation



Nodes selected in this dialog (in the dialog above, nodes *Income* and *PaymentHistory*) will not be modified by the learning stages of cross-validation and will preserve their original CPTs for the testing phase.

Once you have verified that the model and the data are matched correctly, press OK, which will bring up the following dialog:



There are two important elements in this dialog.

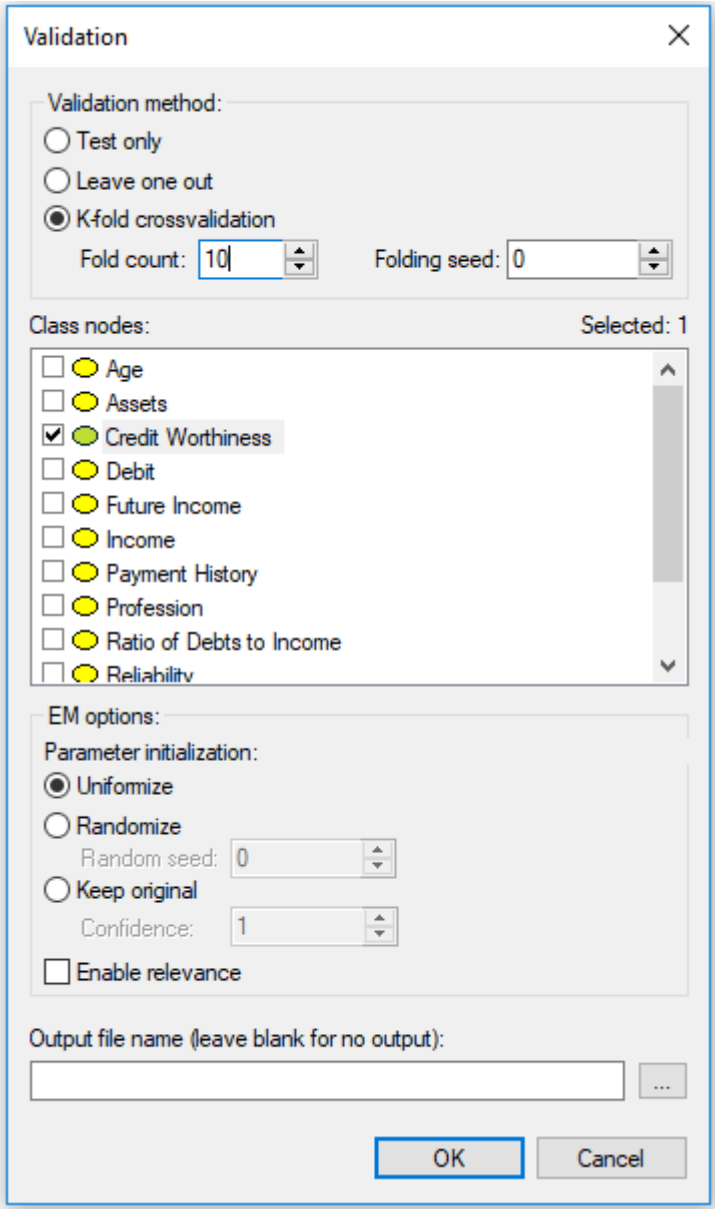
The first element is *Validation method*. The simplest evaluation is *Test only*, which amounts to testing the model on the data file. This is suitable to situations when the model has been developed based on expert knowledge or when the model was learned from a different data set and we want to test it on data that it has never seen (e.g., a holdout data set). More typically, we want to both learn and evaluate the model on the same data set. In this case, the most appropriate model evaluation method is cross-validation, which divides the data into two subsets: training and testing. GeNIe implements the most powerful cross-validation method, known as *K-fold crossvalidation*, which divides the data set into *K* parts of equal size, trains the network on *K*-1 parts, and tests it on the last, *K*th part. The process is repeated *K* times, with a different part of the data being selected for testing. *Fold count* allows for setting the number of folds. *Folding seed* is used in setting up random assignment of records to different folds. Setting the *Folding seed* to anything different than zero (the default) allows for making the process of evaluation repeatable. Zero *Folding seed* amounts to taking the actual random number seed from the system clock and is, therefore, truly random. The *Leave one out* (LOO) method is an extreme case

of *K-fold crossvalidation*, in which K is equal to the number of records (*n*) in the data set. In LOO, the network is trained on *n*-1 records and tested on the remaining one record. The process is repeated *n* times. We advise to use the LOO method, as the most efficient evaluation method, whenever it is feasible in terms of computation time. Its only disadvantage is that it may take long when the number of records in the data set is very large. Let us select *K-fold crossvalidation with* K=10 for our example. Once we have selected a cross-validation technique, EM options become active. The model evaluation technique implemented in GeNIe keeps the model structure fixed and re-learns the model parameters during each of the folds. The default EM options are suitable for this process. Should you wish to explore different settings, please see the description of EM options in the *Learning parameters* section.

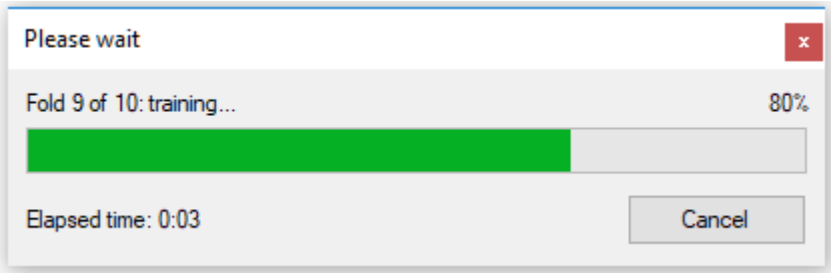
The second important element in the dialog is selection of the *Class nodes*, which are nodes that the model aims at predicting. At least one of the model variables has to be selected. In our example, we will select *Credit Worthiness*.

It is possible to produce an output file during the validation process. The output file is an exact copy of the data file with columns attached at the end that contain the probabilities of all outcomes of all class nodes. This may prove useful in case you want to explore different measures of performance, outside of those offered by GeNIe. If you leave the *Output file name* blank, no output file will be generated.

Here is the *Validation* dialog again with the settings that we recommend for our example:



Pressing *OK* starts the validation process. We can observe the progress of the process in the following dialog:

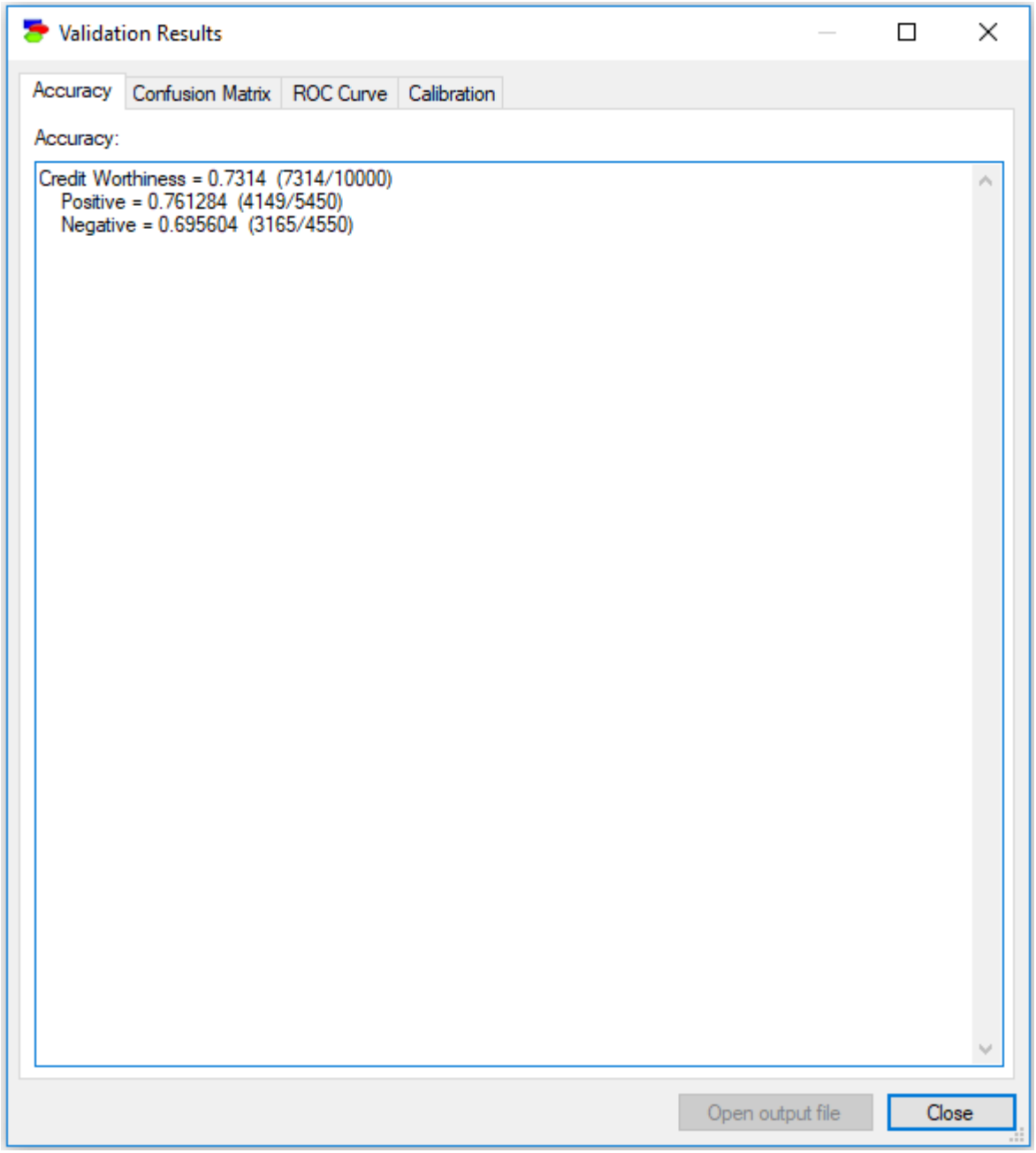


Should the validation take more time than planned for, you can always *Cancel* it and restart it with fewer folds.

Validation results for a single class node

Accuracy

When finished has finished, the following dialog appears:

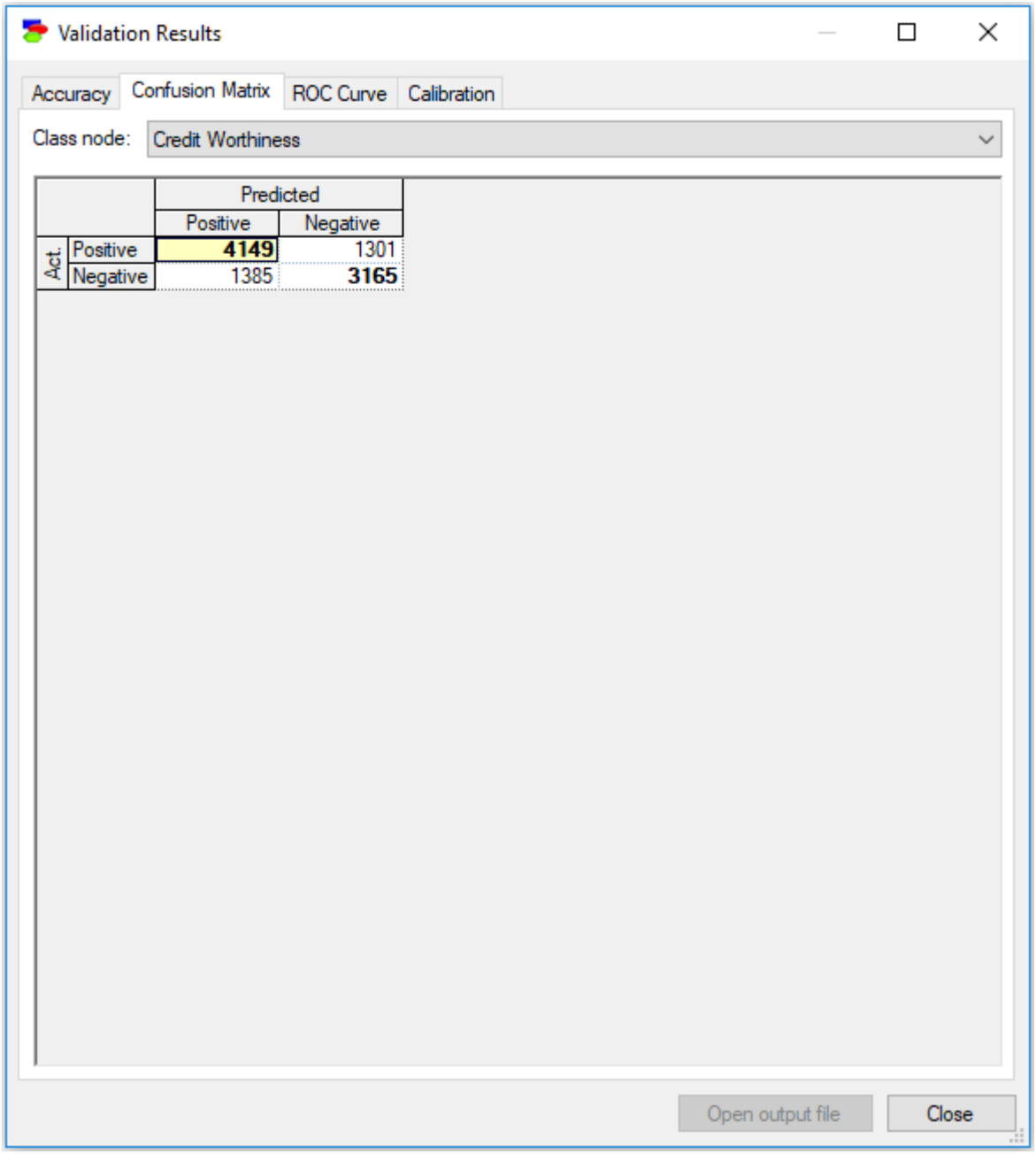


The first tab, *Accuracy*, shows the accuracy that the model has achieved during the validation. In this case, the mode achieved 73.14% accuracy in predicting the correct *Credit Worthiness* - it guessed correctly 7,314 out of the total of 10,000 records. It is important to know that during the process, GeNIe chooses for each record the state of the class node that is most probable over all other states. The tab also shows sensitivity and specificity of the model, although it is up to the user to name them. Sensitivity of the model in detecting the *Positive Credit Worthiness* is roughly 76.13% (4,149 records out of all 5,450 records for which the *Credit Worthiness* was *Positive*), with specificity of roughly 69.56% (3,165 records out of all 4,550 records for which the *Credit Worthiness* was *Negative*). One could look at this result as showing 69.56% sensitivity and 76.13% specificity in detecting *Negative Credit Worthiness*.

CTRL-A will select the entire contents of the tab, possibly to be pasted in an editor.

Confusion Matrix

The *Confusion Matrix* tab shows the same result in terms of the number of records correctly and incorrectly classified. Here the rows denote the actual state of affairs and the columns the model's guess. The diagonal of the confusion matrix (marked by bold numbers) shows the numbers of correctly identified instances for each of the classes. Off-diagonal cells show incorrectly identified classes.



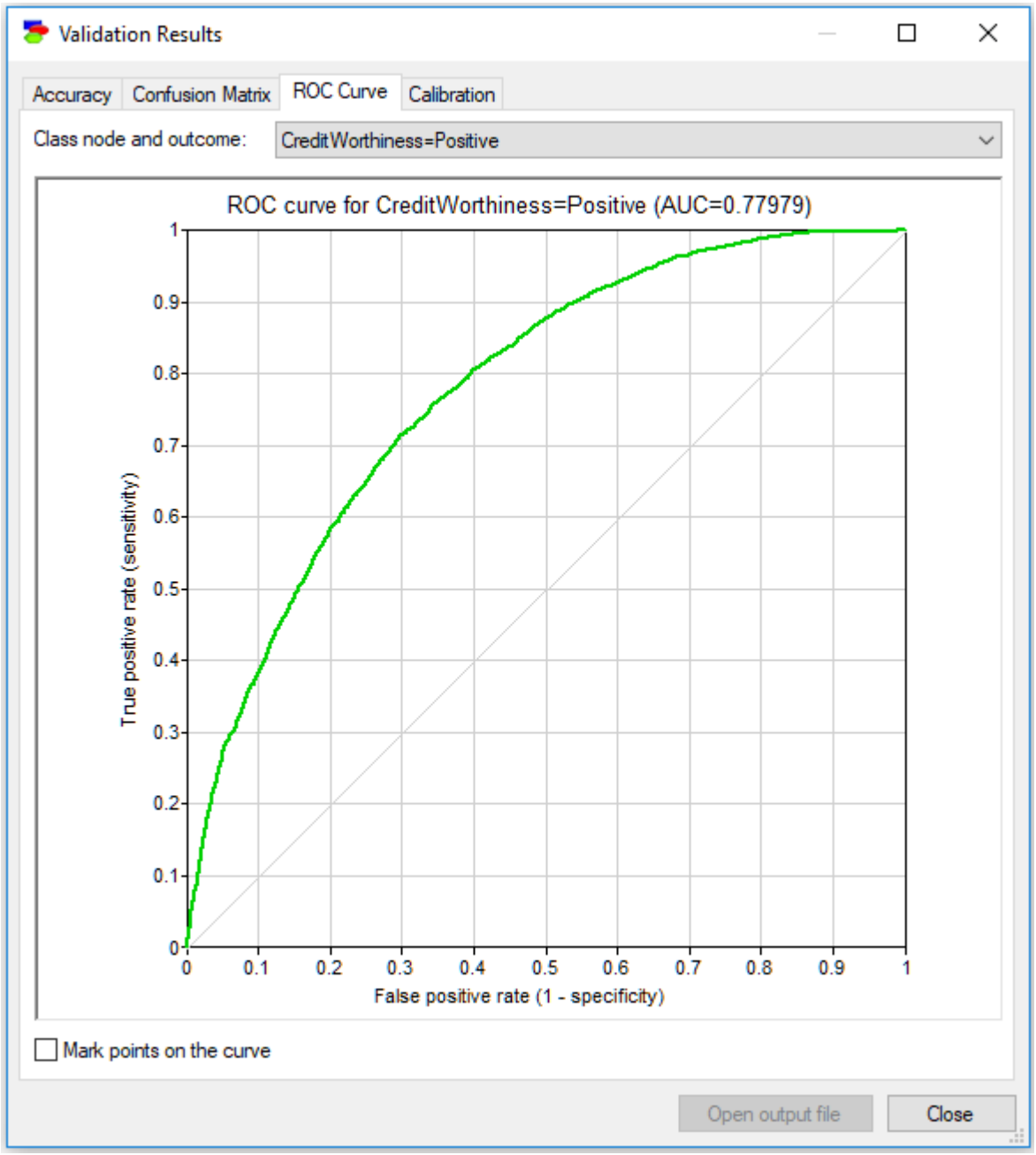
You can copy the contents of the confusion matrix by right-clicking *Copy* on the selected cells (or right-clicking and choosing *Select All*). You can paste the selected contents into other programs.

ROC Curve

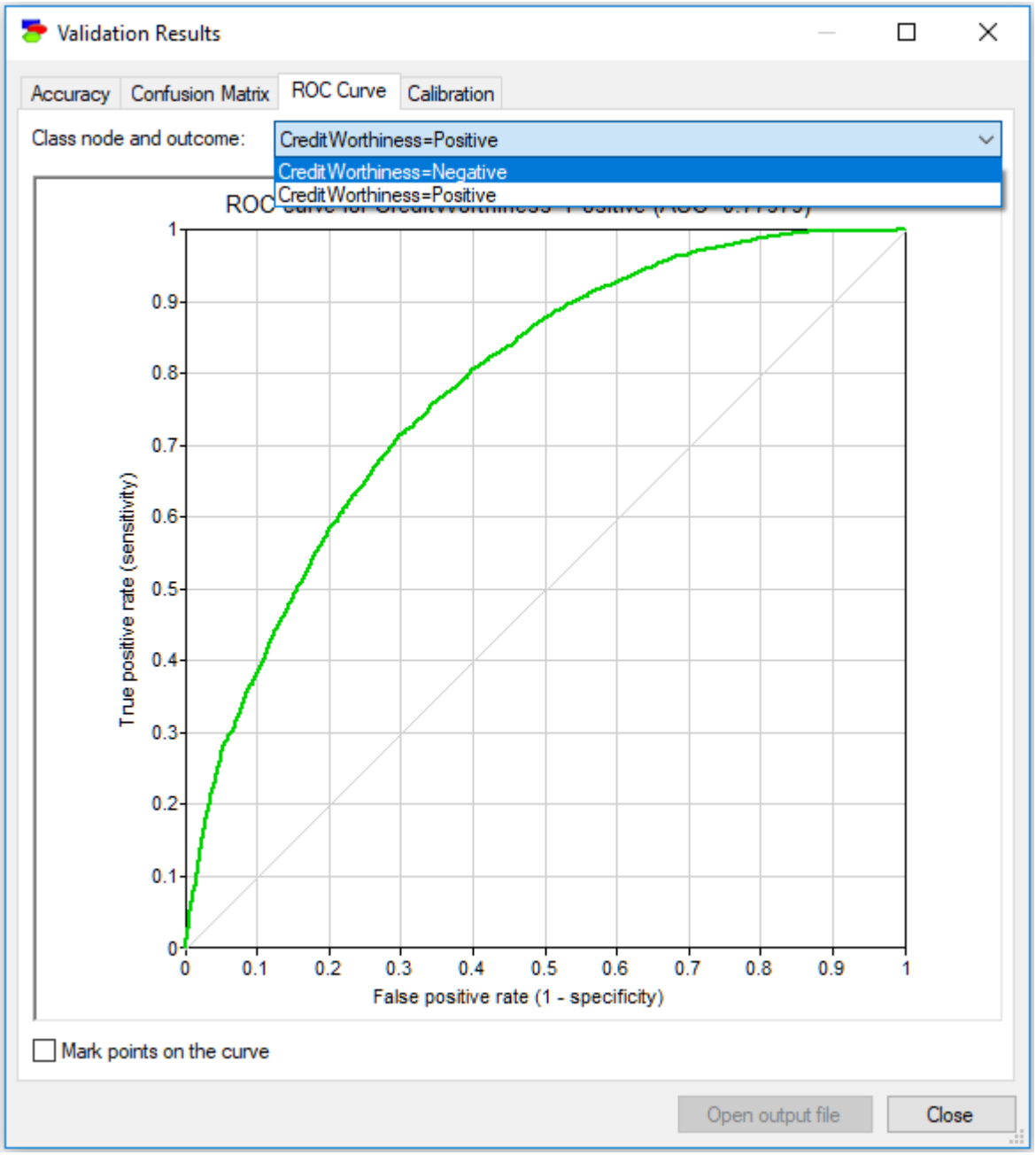
The *ROC Curve* tab shows the Receiver Operating Characteristic (ROC) curves for each of the states of each of the class variables. ROC curves originate from Information Theory and are an excellent way of expressing the quality of a model independent of the classification decision (in case of GeNIe validation, this decision is based on the most likely state, which in case of a binary variable like *Credit Worthiness* amounts to a probability threshold of 0.5). The ROC curve is capable of showing the possible accuracy ranges, and the decision criterion

applied by GeNIe is just one point on the curve. Choosing a different point will result in a different sensitivity and specificity (and, hence, the overall accuracy). The ROC curve gives insight into how much we have to sacrifice one in order to improve the other and, effectively, helps with choosing a criterion that is suitable for the application at hand. It shows the theoretical limits of accuracy of the model on one plot.

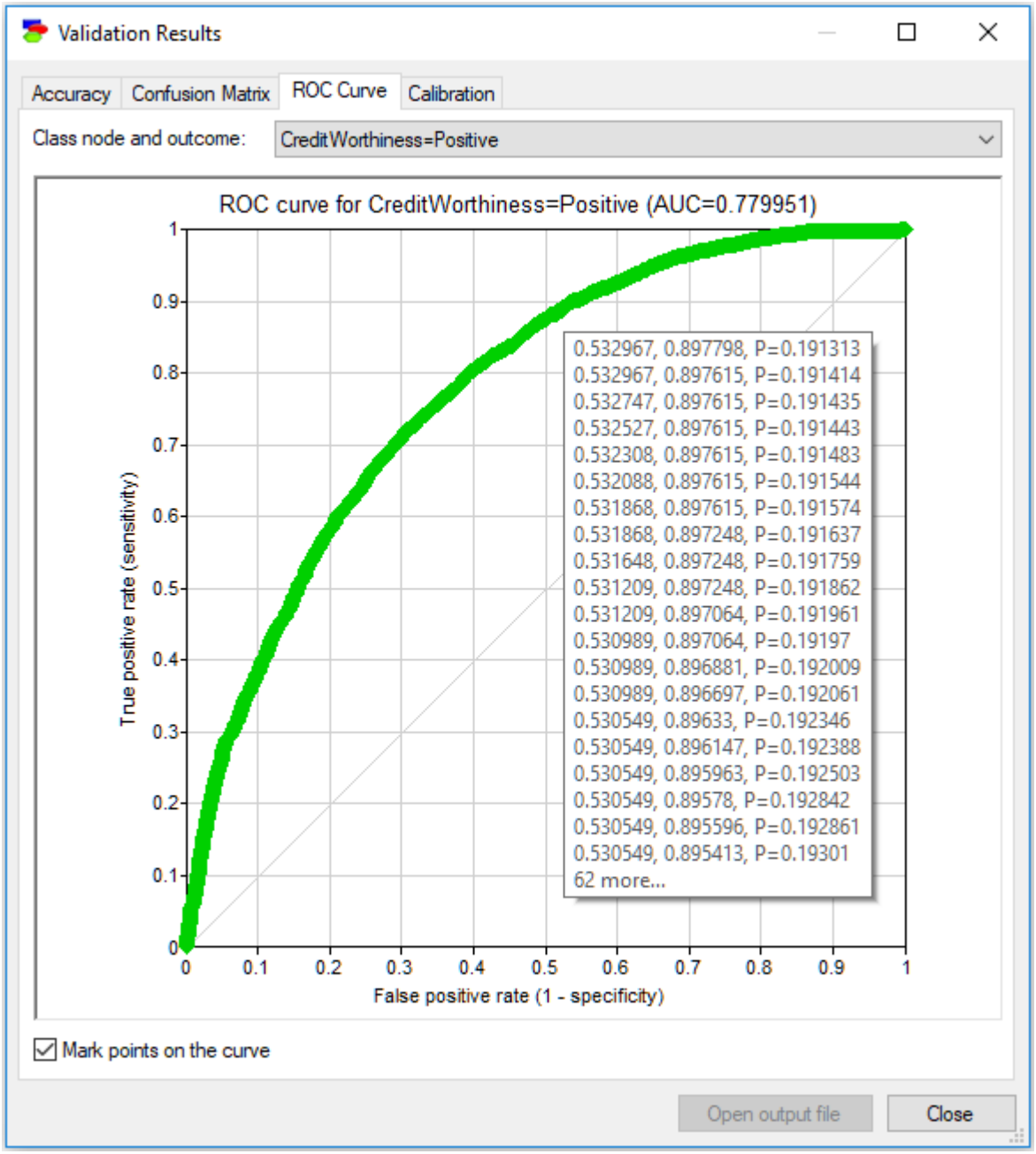
The following ROC curve is for the state *Positive* of the variable *Credit Worthiness*. The dim diagonal line shows a baseline ROC curve of a hypothetical classifier that is worthless. A classifier that does its job will have its ROC curve above this diagonal line. Above the curve, we see the Area Under the ROC Curve (*AUC*) displayed. AUC is a simple, albeit imperfect way of expressing the quality of the model by means of one number.



The ROC curve assumes that the class node is binary. When the class node is not binary, GeNIe changes it into a binary node by taking the state of interest (in the ROC curve above, it is the state *Positive*) and lumping all remaining states into the complement of the chosen state. There are, thus, as many ROC curves for each of the class nodes as there are states. The pop-up menu in the upper-right corner of the dialog allows for choosing a different class variable and a different state. The following screen shot shows the ROC curve for the state *Negative* of the variable *Credit Worthiness*.



Finally, the ROC curve is drawn based on a finite number of points, based on the data set used for the purpose of verification. When the number of points is small (typically, this occurs when the data file is small), the curve is somewhat rugged. It may be useful to see these points on the curve. The *Mark points on the curve* check box turns these points on an off. The following plot shows this idea.

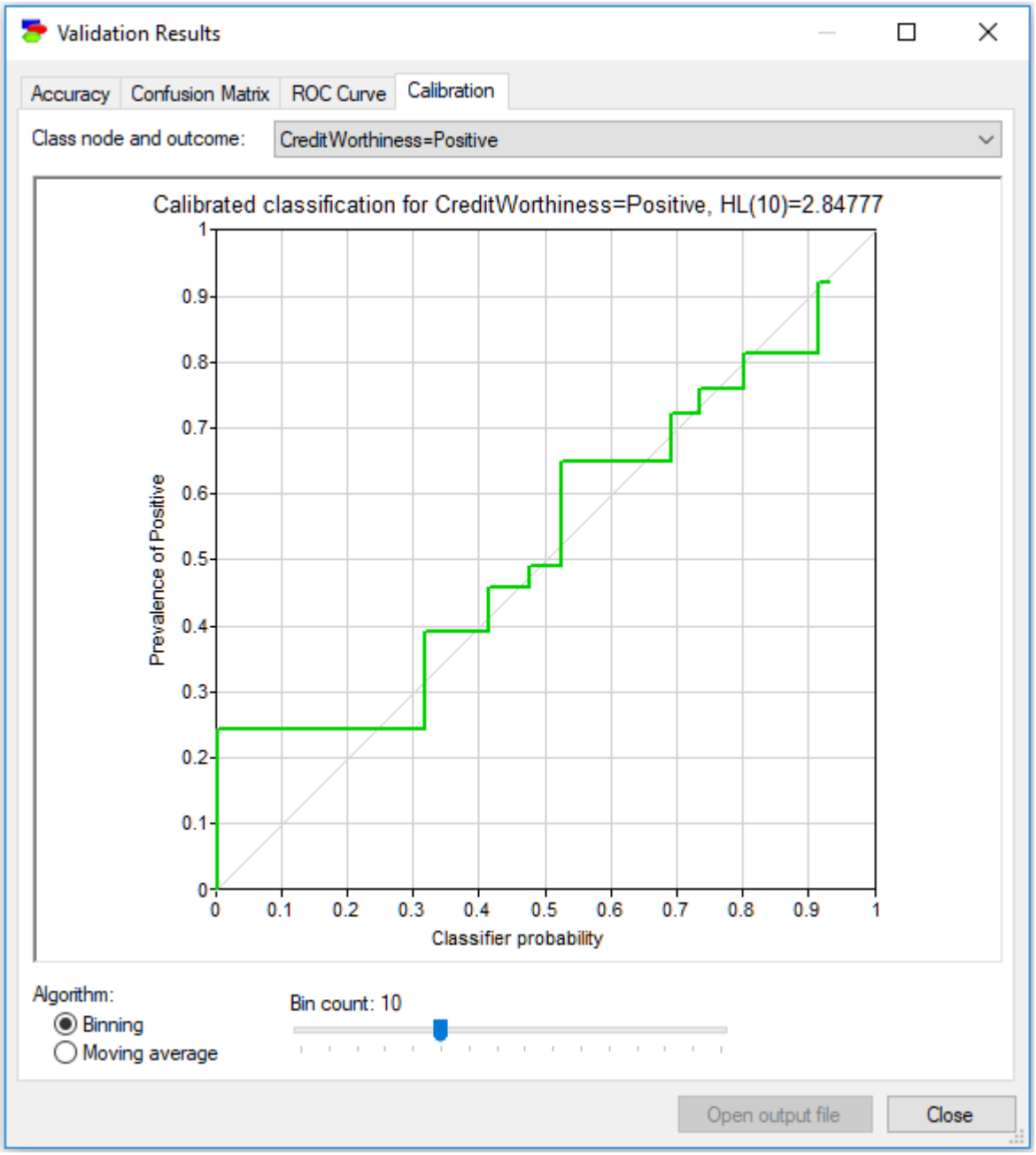


Hovering over any of the points shows the probability threshold value that needs to be used to achieve this point along with the resulting (1-specificity) and sensitivity. In the example above, when the probability threshold is p=0.6, the model achieves sensitivity of 0.7 and specificity of 1-0.237143=0.762857. You can also copy and paste the numbers behind the ROC curve by right-clicking anywhere on the chart, selecting *Copy* and then pasting the results as text into any text editor (such as Notepad or Word). Pasting into any image editor (or *pasting special* into a text editor such as Word) results in pasting the image of the ROC curve, in bitmap or picture format, explained in the *Graph* view section.

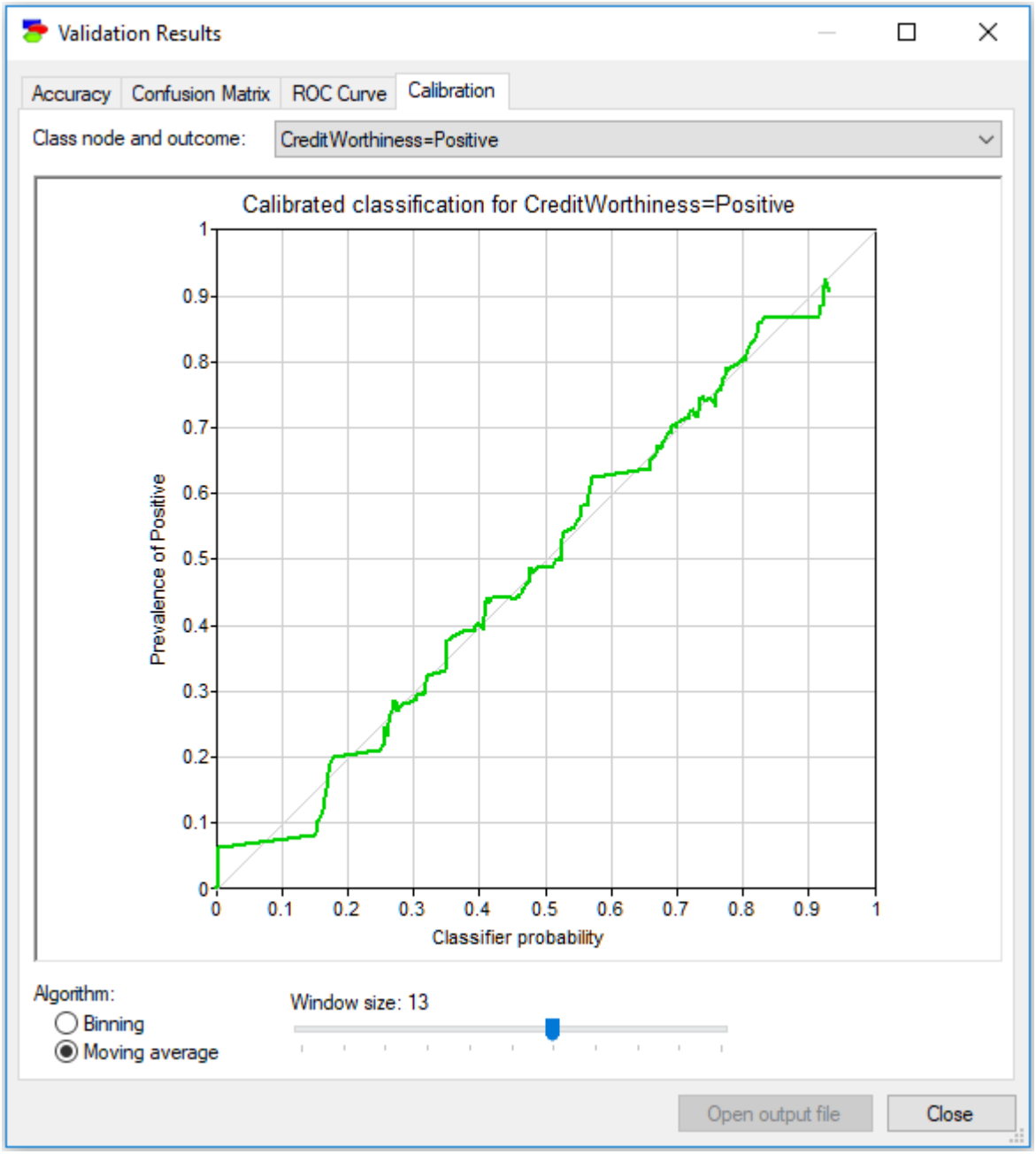
ROC curve is a fundamental and useful measure of a model's performance. For those users, who are not familiar with the ROC curves and AUC measures, we recommend an excellent article on Wikipedia (https://en.wikipedia.org/wiki/Receiver_operating_characteristic).

Calibration curve

The final tab, *Calibration*, shows a very important measure of performance of a probabilistic model, notably the calibration curve. Because the output of a probabilistic model is a probability and this probability is useful in decision making, ideally we would like it to be as accurate as possible. One way of measuring the accuracy of a model is comparing the output probability to the actually observed frequencies in the data. The calibration curve shows how these two compare. For each probability *p* produced by the model (the horizontal axis) for the class variable, the plot shows the actual frequencies in the data (vertical axis) observed for all cases for which the model produced probability *p*. The dim diagonal line shows the ideal calibration curve, i.e., one in which every probability produced by the classifier is precisely equal to the frequency observed in the data. Because *p* is a continuous variable, the plot groups the values of probability so that sufficiently many data records are found to estimate the actual frequency in the data for the vertical axis. There are two methods of grouping implemented in GeNIe: *Binning* and *Moving average*. *Binning* works similarly to a histogram - we divide the interval [0..1] into equal size bins. As we change the number of bins, the plot changes as well. It is a good practice to explore several bin sizes to get an idea of the model calibration.



Moving average (see the screen shot below) works somewhat differently. We have a sliding window that takes always the neighboring *k* output probabilities on the horizontal axis and shows the class frequency among the records in this sliding window on the vertical axis. Here also, as we change the *Window size*, the plot changes as well. It is a good practice to explore several window sizes to get an idea of the model calibration.

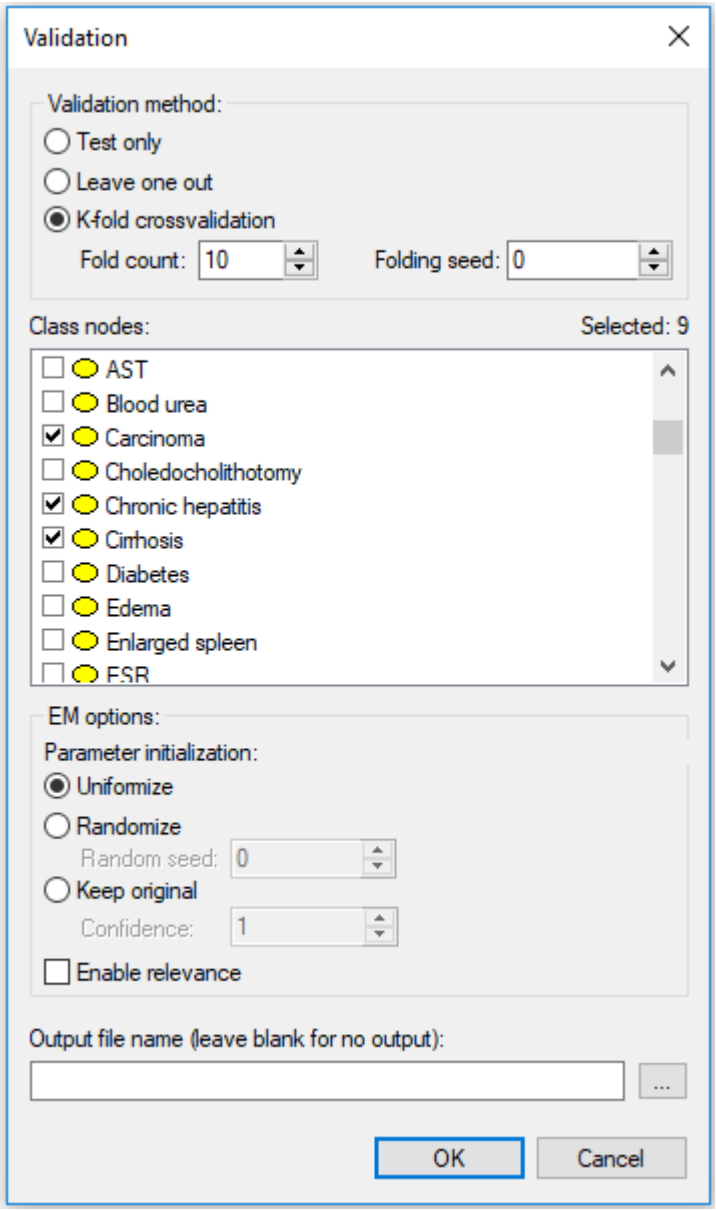


Similarly to the ROC curve, you can copy and paste the coordinates of points on the calibration curve by right-clicking anywhere on the chart, selecting *Copy* and then pasting the results as text into any text editor (such as Notepad or Word). Pasting into any image editor (or *pasting special* into a text editor such as Word) results in pasting the image of the calibration curve in bitmap or picture format, explained in the *Graph* view section.

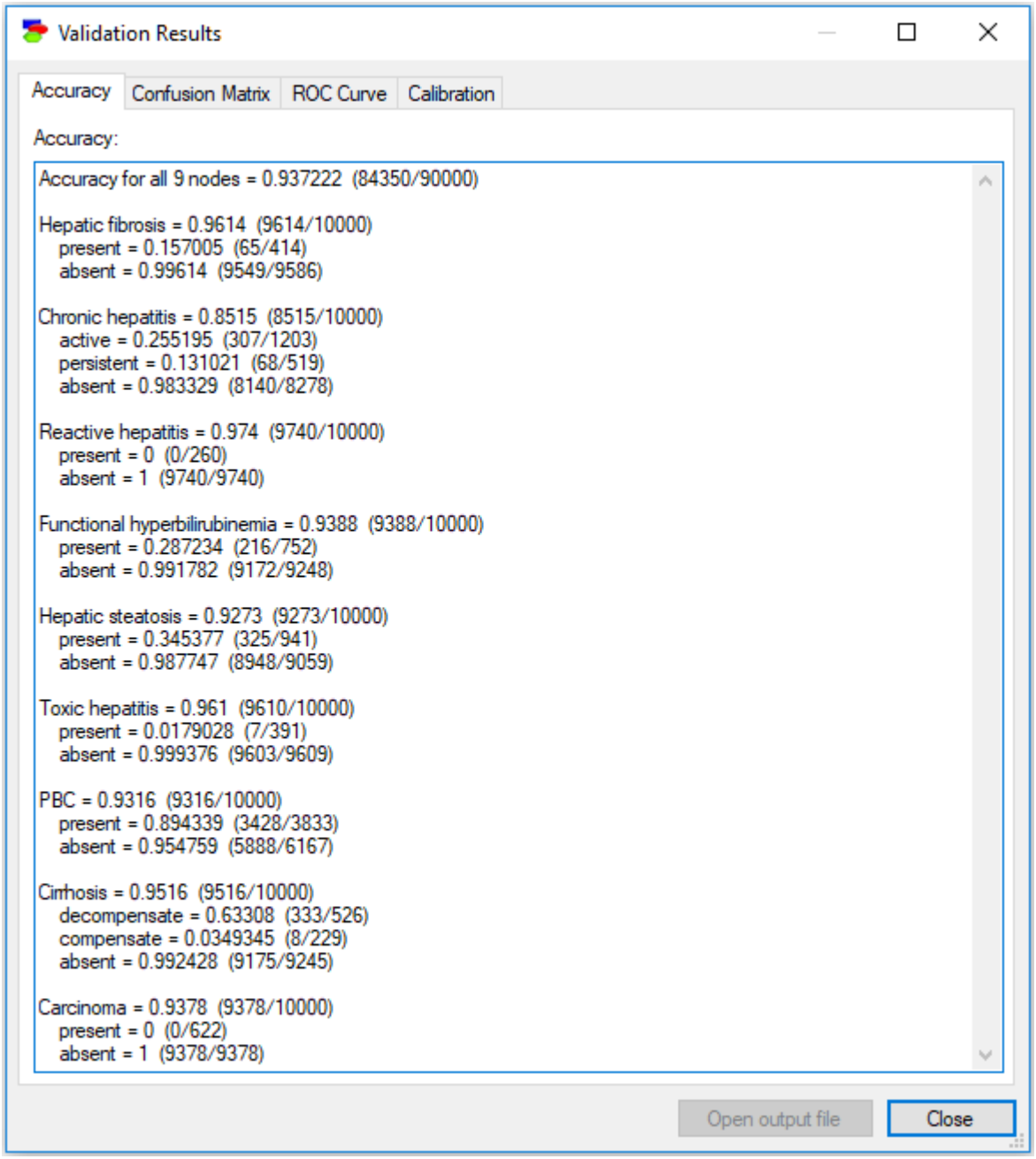
Validation for multiple target nodes

The above example involved one class node, *Credit Worthiness*. It happens sometimes that there are several class nodes, for example in multiple disorder diagnosis, when more than one problem can be present at the same time. The Hepar II model (Onisko, 2003) contains nine disorder nodes (*Toxic hepatitis*, *Chronic hepatitis*, *PBC*,

Hepatic fibrosis, *Hepatic steatosis*, *Cirrhosis*, *Functional hyperbilirubinemia*, *Reactive hepatitis* and *Carcinoma*). When validating the model, we should select all of these.



The results show accuracy for each of the class nodes in separation

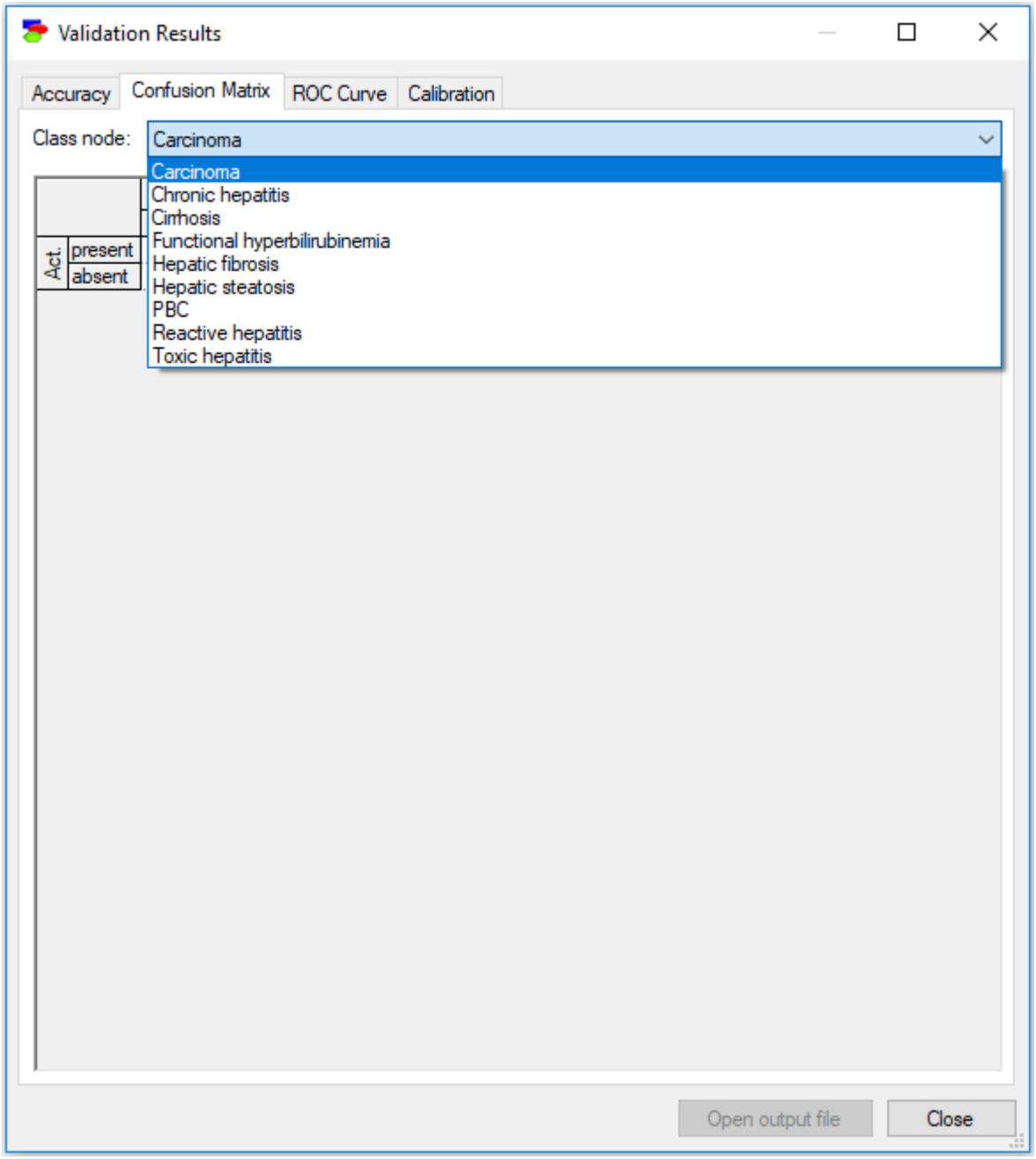


The accuracy tab lists the accuracy for each of the nodes in separation (as opposed to the accuracy in terms of combinations of values of class nodes). Should you wish to calculate the accuracy of the model in pinpointing a combination of values of several nodes, you can always perform a structural extension of the model by creating a deterministic child node of the class nodes in question that has states corresponding to the combinations of interest. Accuracy for those states is equal to the accuracy of the combinations of interest.

It is important to know that when testing a model with multiple class nodes GeNIe never instantiates any of the class nodes. This amounts to observing only non-class nodes and corresponds to the common situation in which we do not know any of the class nodes (e.g., we do not know for sure any of the diseases). Should you wish to calculate the model accuracy for a class node when knowing the value of other class nodes, you will need to run

validation separately and select only the class node tested. In this case, GeNIe will use the values of all nodes that are not designated as class nodes.

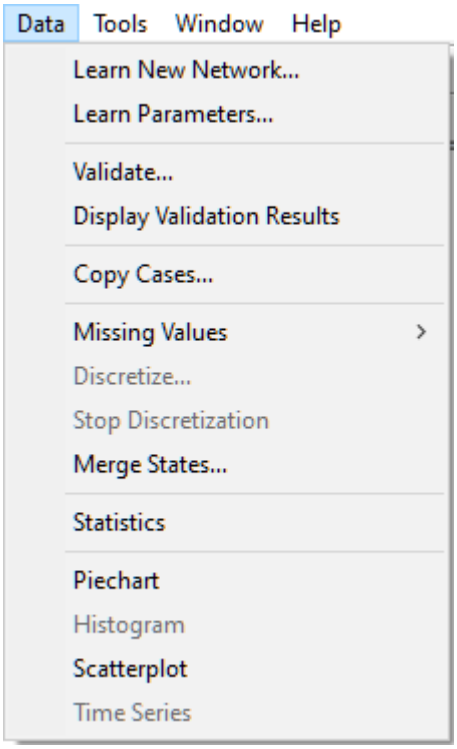
The *Confusion Matrix* tab requires that you select one of the class nodes - there are as many confusion matrices as there are class nodes.



The remaining two result tabs (*ROC Curve* and *Calibration*) require selecting a state of one of the class nodes.

Displaying results based on a validation output file

The cross-validation results file contains all the information that are required to show confusion matrix, calculate accuracy, display the ROC and the calibration curves, etc. GeNIe allows for opening an existing validation output file for this purpose. The output data file can be opened exactly the way one opens a data file. Once an output data file is open, please select the *Display Validation Results* command from the *Data Menu*.



This will invoke the *Validation Results* dialog, as described above in this section. No corresponding network needs to be open but the dialog rests on the assumption that the validation output file contains additional columns, as created by GeNIe, i.e., for every class variable *X* there are corresponding columns *X_State1*, *X_State2*, ..., *X_StateN*, and *X_predicted*, where *State1* through *StateN* are all states of *X*. This is precisely what GeNIe places in its output files, so any output file produced by GeNIe is amenable for opening later and displaying validation results.

6.6 Dynamic Bayesian networks

6.6.1 Introduction

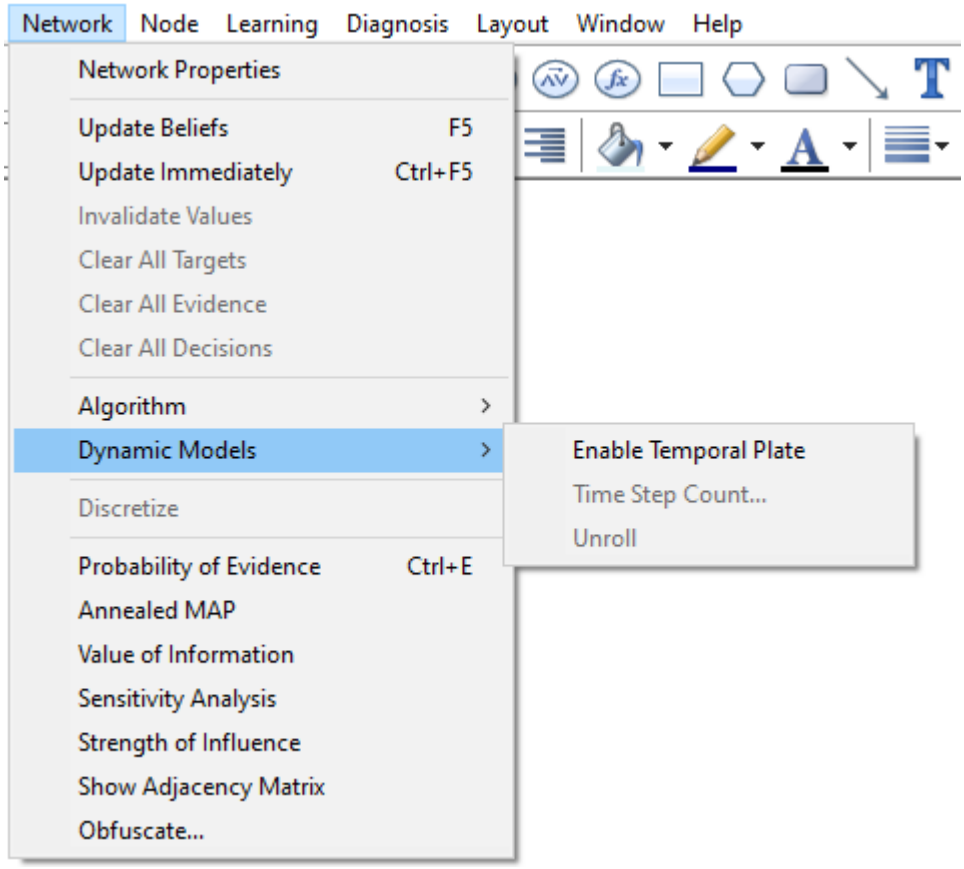
A Bayesian network is a snap shot of the system at a given time and is used to model systems that are in some kind of equilibrium state. However, most systems in the world change over time and sometimes we are interested in how these systems evolve over time more than we are interested in their equilibrium states. Whenever the focus of our reasoning is change of a system over time, we need a tool that is capable of modeling dynamic systems.

A *dynamic Bayesian network* (*DBN*) is a Bayesian network extended with additional mechanisms that are capable of modeling influences over time (Murphy, 2002). We assume that the user is familiar with DBNs, Bayesian networks, and GeNIe. The temporal extension of BNs does not mean that the network structure or parameters changes dynamically, but that a dynamic system is modeled. In other words, the underlying process, modeled by a DBN, is stationary. A DBN is a model of a stochastic process. The implementation of DBNs in GeNIe is based on a M.Sc. thesis by Joris Hulst (2006).

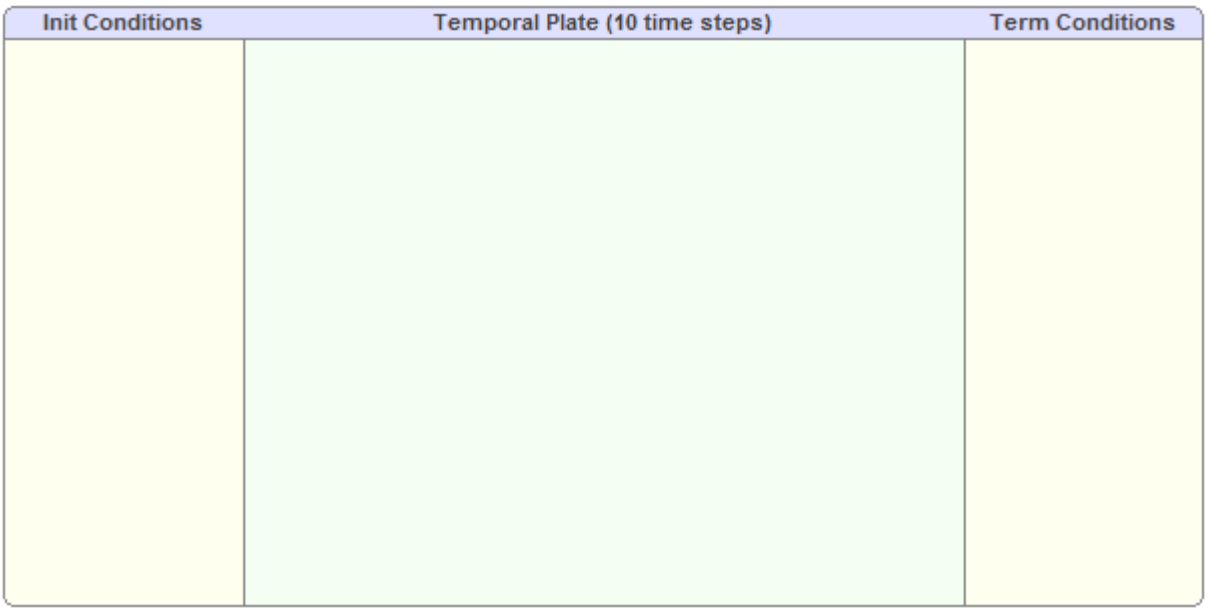
6.6.2 Creating DBN

Consider the following example, inspired by (Russell & Norvig, 1995), in which a security guard at some secret underground installation works on a shift of seven days and wants to know whether it is raining on the day of her return to the outside world. Her only access to the outside world occurs each morning when she sees the director coming in, with or without, an umbrella. Furthermore, she knows that the government has two secret underground installations: one in Pittsburgh and one in the Sahara, but she does not know which one she is guarding. For each day *t*, the set of evidence contains a single variable $Umbrella_t$ (observation of an umbrella carried by the director) and the set of unobservable variables contains $Rain_t$ (a propositional variable with two states *true* and *false*, denoting whether it is raining) and *Area* (with two possible states: *Pittsburgh* and *Sahara*). The prior probability of rain depends on the geographical location and on whether it rained on the previous day.

We will use GeNIe to model this example. We start with enabling the *Temporal Plate*, which is a special construct in the *Graph View* that allows for building dynamic models



The effect of enabling temporal plate in the *Graph View* is the following workspace addition to the *Graph* view window.



The *Temporal Plate* divides the *Graph* view into four areas:

- *Contemporals*, which is the part of the *Network View* window that is outside of the temporal plate. All nodes in that area are static.

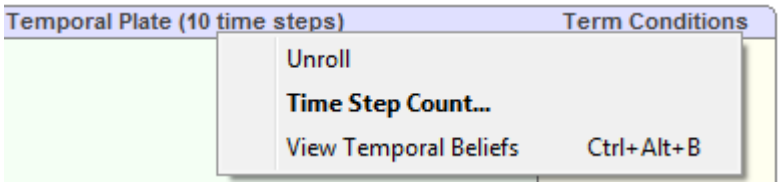
- *Init Conditions*, which is the part of the network area where, so called, *anchor nodes* are stored. An *anchor node* is a node outside of the temporal plate that has one or more children inside the temporal plate. Anchor nodes are similar to static nodes outside of the temporal plate but they are only connected to nodes (their children) in the first time-slice of the network.

- *Temporal Plate*, which is the main part representing the dynamic model. Nodes in the *Temporal Plate* are the only nodes that are allowed to have *Temporal Arcs*. This area also shows the number of time-slices for which inference will be performed.

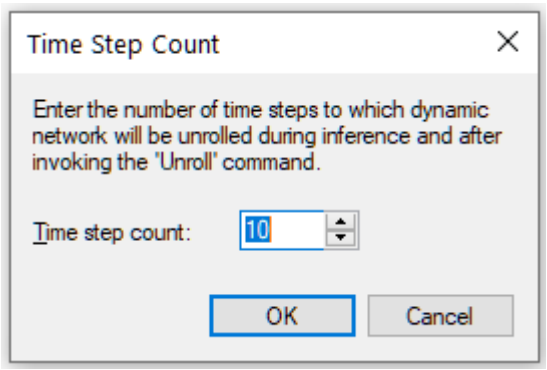
- *Term Conditions*, which is the part of the network area where the *terminal nodes* are stored. A terminal node is a node outside of the temporal plate that has one or more parents inside the temporal plate. *Terminal nodes* are only connected to nodes (their parents) in the last time-slice of the network.

The size of the *Temporal Plate* can be changed by clicking and dragging its edges and so can the sizes of its three areas (*Init Conditions*, *Temporal Plate* and *Term Conditions*). There is a small subtlety in resizing the three. If you click and drag the extreme right or extreme left edge of the temporal plate, it is the middle part (the Temporal Plate) that gets resized and the sizes of *Init Conditions* and *Temporal Plate* remain the same. Pressing the *SHIFT* button when dragging the edges has the effect that the size of the Temporal Plate remains the same and the sizes of *Init Conditions* and *Temporal Plate* change.

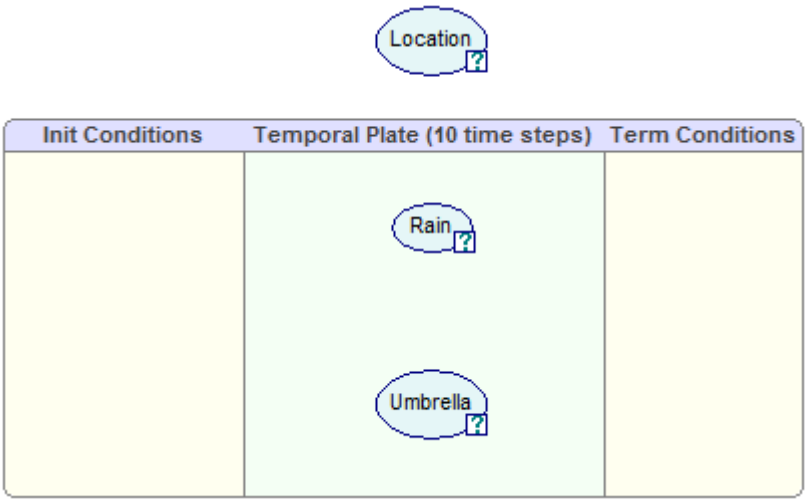
For our example, we set the number of time steps to 8. We can either double-click or right-click on the header of the *Temporal Plate*



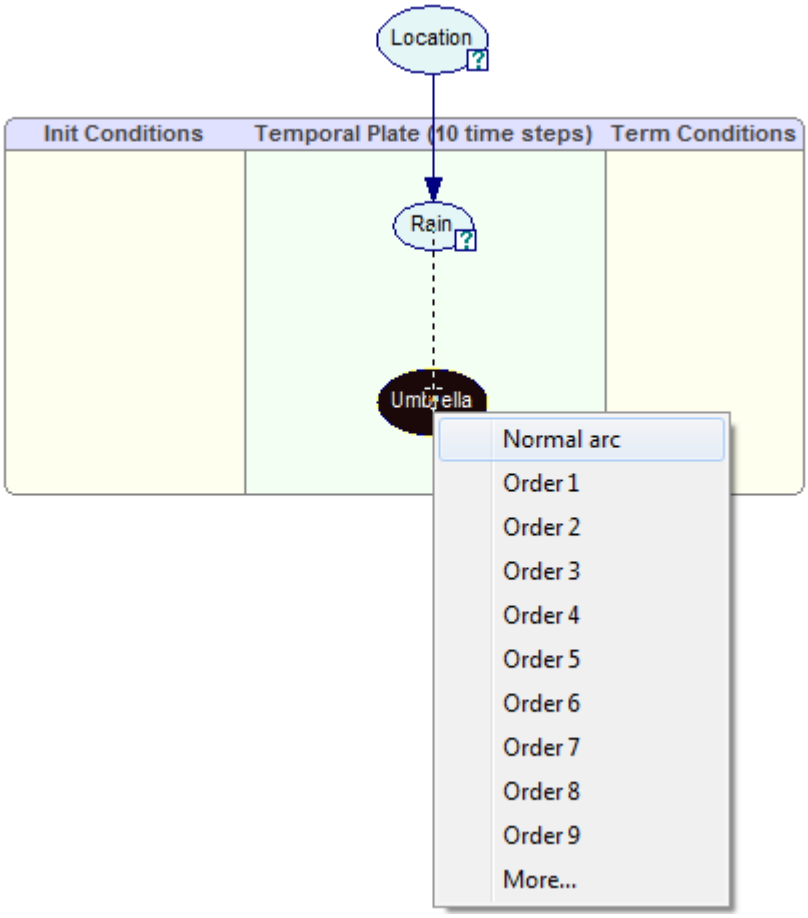
which will invoke the *Time Step Count* dialog that allows to change the *Time step count*



We create the following nodes: *Location* in the *Contemporals*, *Rain* and *Umbrella* in the *Temporal Plate* and set their states as described in the example above.

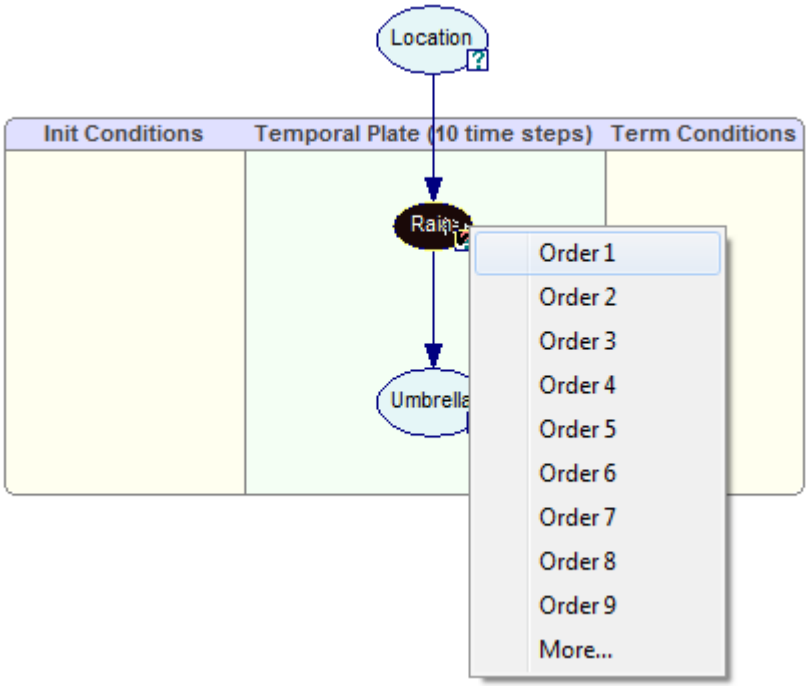


The next step is to connect these nodes. We draw an arc from *Location* to *Rain* and then from *Rain* to *Umbrella*. The first arc is a regular Bayesian network arc but in case of the second arc, we are connecting nodes inside the *Temporal Plate* and need to indicate the time order of the arc drawn



Because the influence of *Rain* on whether or not the director carries an umbrella with her is instantaneous, we choose *Normal arc*. We draw an arc from the node Rain to itself, which will represent the impact on *Rain* on a

given day that *Rain* on the prior day has. To achieve this, select the *Arc* tool, click and hold on the *Rain* node, move the cursor outside of the node and back into it, upon which the node becomes black, and release the cursor, which will cause the arc order menu to pop up. In this case, we choose *Order 1*, which indicates that the impact has a delay of 1 day: The state of the variable *Rain* on the previous day impacts the state of *Rain* today.



This operation will result in a temporal arc of order 1 being created from the node *Rain* to itself. Please note that the restriction that the graph of a Bayesian network be acyclic does not hold inside the *Temporal Plate*. Cycles represent temporal processes and are allowed only for temporal arcs. Similarly to static networks, normal arcs are not allowed to form cycles, even inside the *Temporal Plate*.

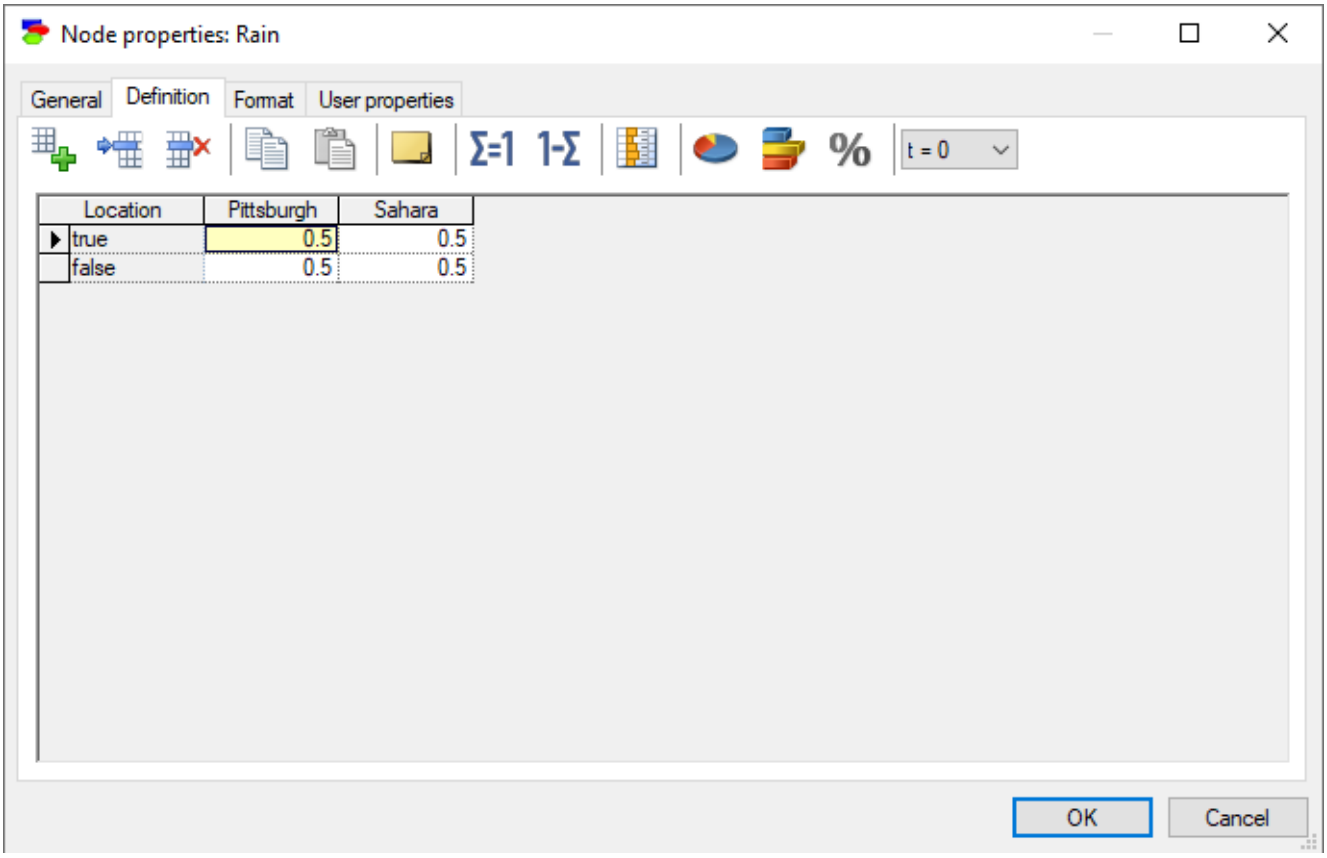
Please note that a DBN, as implemented in GeNIe, can have temporal arcs of any order, which means that DBNs in GeNIe can model dynamic processes of any order.

We define each of the nodes in terms if their states and numerical probabilities. Definitions of the nodes *Location* and *Umbrella* are identical to the definitions of nodes in Bayesian networks:

Pittsburgh	0.5
Sahara	0.5

Rain	true	false
true	0.9	0.2
false	0.1	0.8

However, the definition of the node *Rain* is specific to DBNs, as one of the incoming arcs is temporal



Please note an additional pop-up menu on the right-hand side, marked *t=0*. This menu allows us to traverse through the conditional probability tables that compose the definition of node Rain. For *t=0*, we enter the prior probability of rain on day *0*:

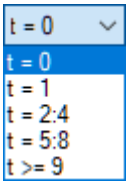
Location	Pittsburgh	Sahara
true	0.7	0.01
false	0.3	0.99

For *t=1*, which denotes any day that has an observation of rain on the prior day, we enter the prior probability of rain on day *1* as a function of the *Location* and rain on the previous day:

Location	Pittsburgh		Sahara	
(Self) [t-1]	true	false	true	false
true	0.7	0.3	0.001	0.01
false	0.3	0.7	0.999	0.99

This menu shows a list of definitions of a node that has incoming temporal arcs. In the above example, there are just two definitions, for t=0 and t=1.

It may happen that there are more definitions and that these definitions are not for consecutive time steps. Node *Phase* in the `MenstrualCycle.xdsl` model has four incoming temporal arcs, of orders [1], [2], [5] and [9]. The node has five different definitions: one for t=0, one for t=1, one for t ranging between 2 and 4 (i.e., t=2, t=3, and t=4), one for t ranging between 5 and 8 (i.e., t=5, t=6, t=7, and t=8), and finally one for t>=9 (i.e., t=9, t=10, ...).



The reason for these varying definitions is that even though the model has a temporal influence of order 9, for t<9, this influence will not occur. Similarly, at time t=4, there will be no influences of 5th and 9th order, so the definition has to different and involve influences of order smaller than 5 (in case of this model, influences of 1st and 2nd order only). To observe this in action, try unrolling this network and you will see that there is just no way higher order influences can appear in the first few steps of the unrolled network.

This concludes the creation and specification of the DBN modeling the problem. There is another way of creating a DBN. Rather than constructing it directly in the *Temporal Plate*, we can construct a BN in the *Graph View* window, drag it into the *Temporal Plate*, and adding temporal links. This has to be done cautiously, as the order of dragging can make a difference. For example, if we drag the node *Rain* into the *Temporal Plate*, GeNIe will remove the arc between the nodes *Rain* and *Umbrella*. This is because it is not allowed to have arcs from temporal plate enter nodes in the *Contemporals* plate. To avoid that, we can drag both the *Umbrella* and *Rain* nodes into the temporal plate, in which case no links will be deleted.

6.6.3 Inference in DBNs

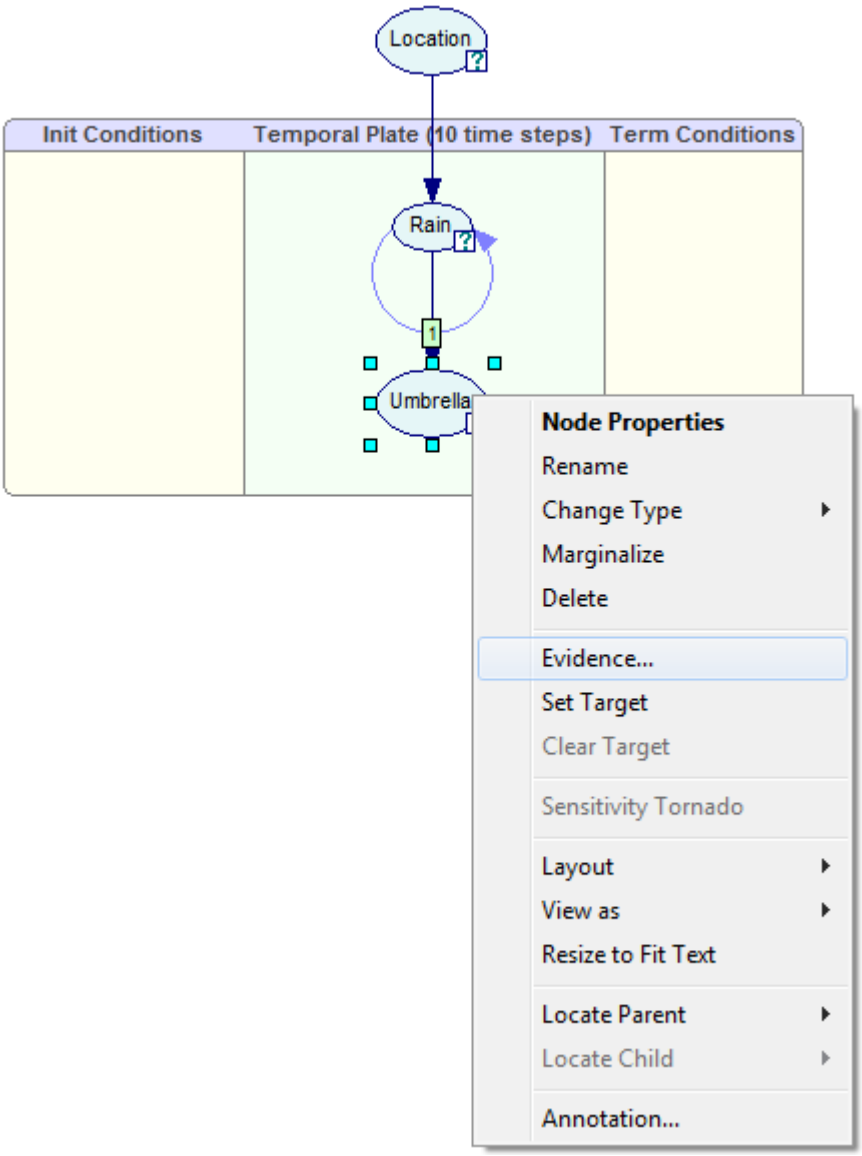
Inference in a DBN, similarly to inference in a BN, amounts to calculating the impact of observation of some of its variables on the probability distribution over other variables. The additional complication is that both evidence and the posterior probability distributions are indexed by time. We will go through the example used in the previous sections to demonstrate setting evidence, running an algorithm, and viewing the results.

Setting temporal evidence

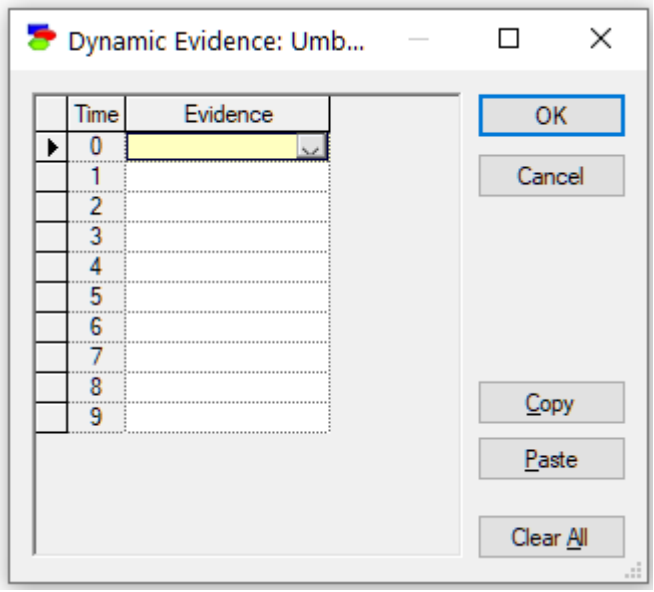
Suppose that during her week-long shift, the guard observes an umbrella on every day except for day three (day count starts with zero), when she was sure she did not see any umbrella and day four, when she forgot whether she saw an umbrella or not. This means that the evidence vector for the node *Umbrella* is as follows:

Umbrella[0:6]= [*true*, *true*, *true*, *false*, --, *true*, *true*].

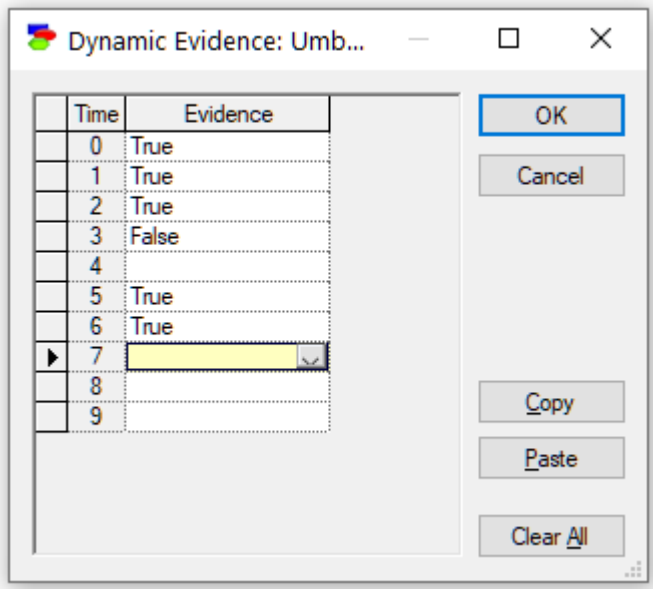
To enter this evidence, we right-click on the *Umbrella* node and select *Evidence*...



This invokes the *Dynamic Evidence* dialog



We enter the evidence vector as specified above:

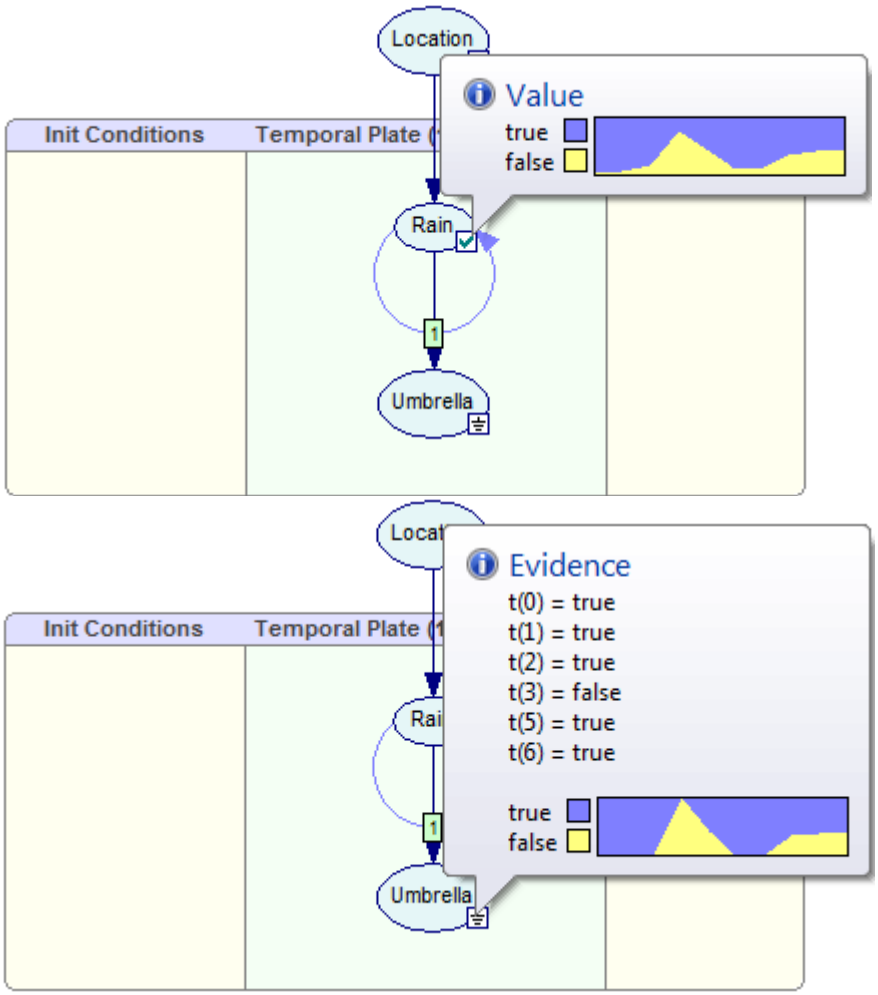


Running the belief updating algorithm

Running the belief updating algorithm is identical to doing so in Bayesian networks. We press the *Update* ( ) button or select *Update Beliefs* from the *Network Menu*. GeNIe converts the DBN into a Bayesian network (this is called unrolling - see below) and updates the beliefs using the selected belief updating algorithm.

Viewing the results: Temporal posterior beliefs

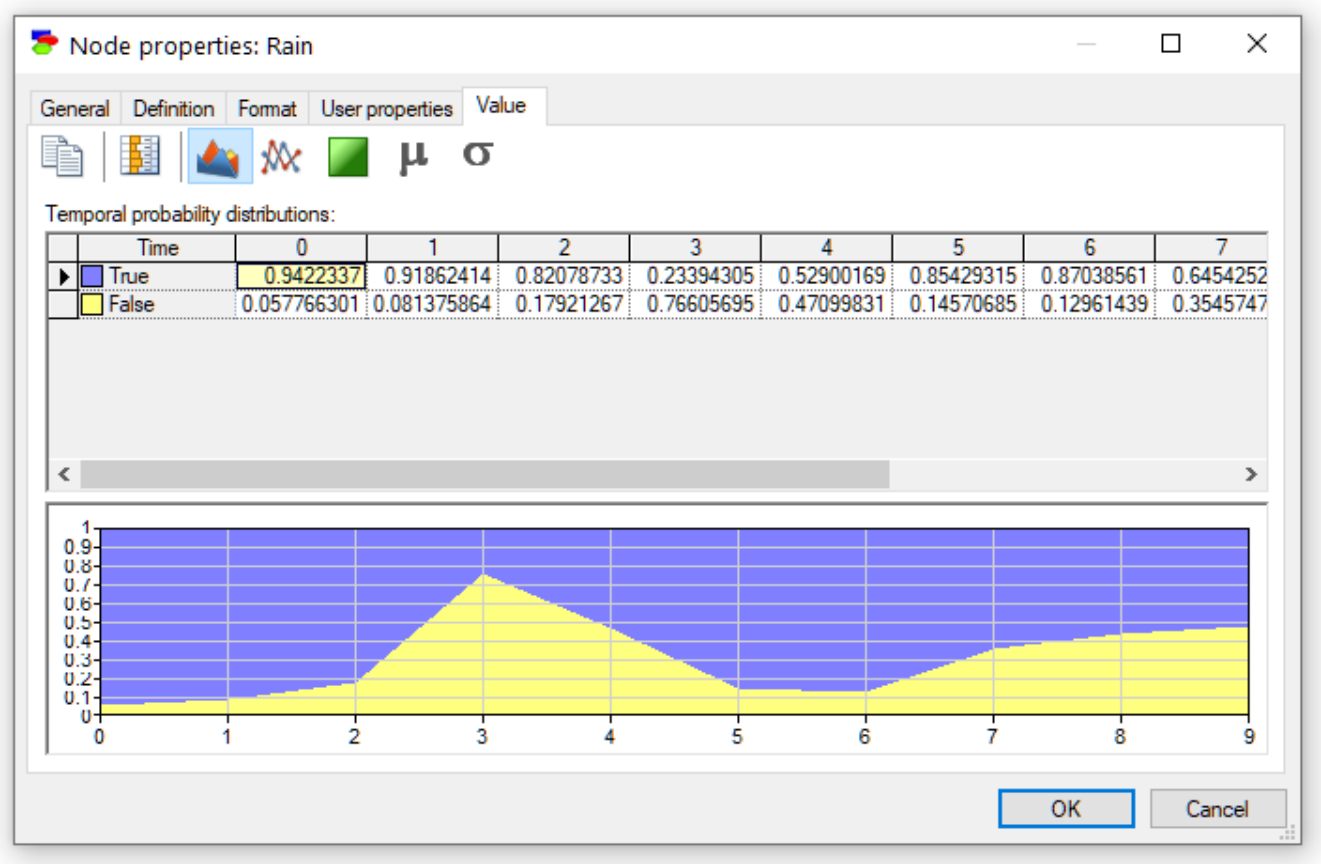
After the network has been updated, we can view its temporal beliefs, which are marginal posterior probability distributions as a function of time. Hovering the mouse over the status icon yields the following views:



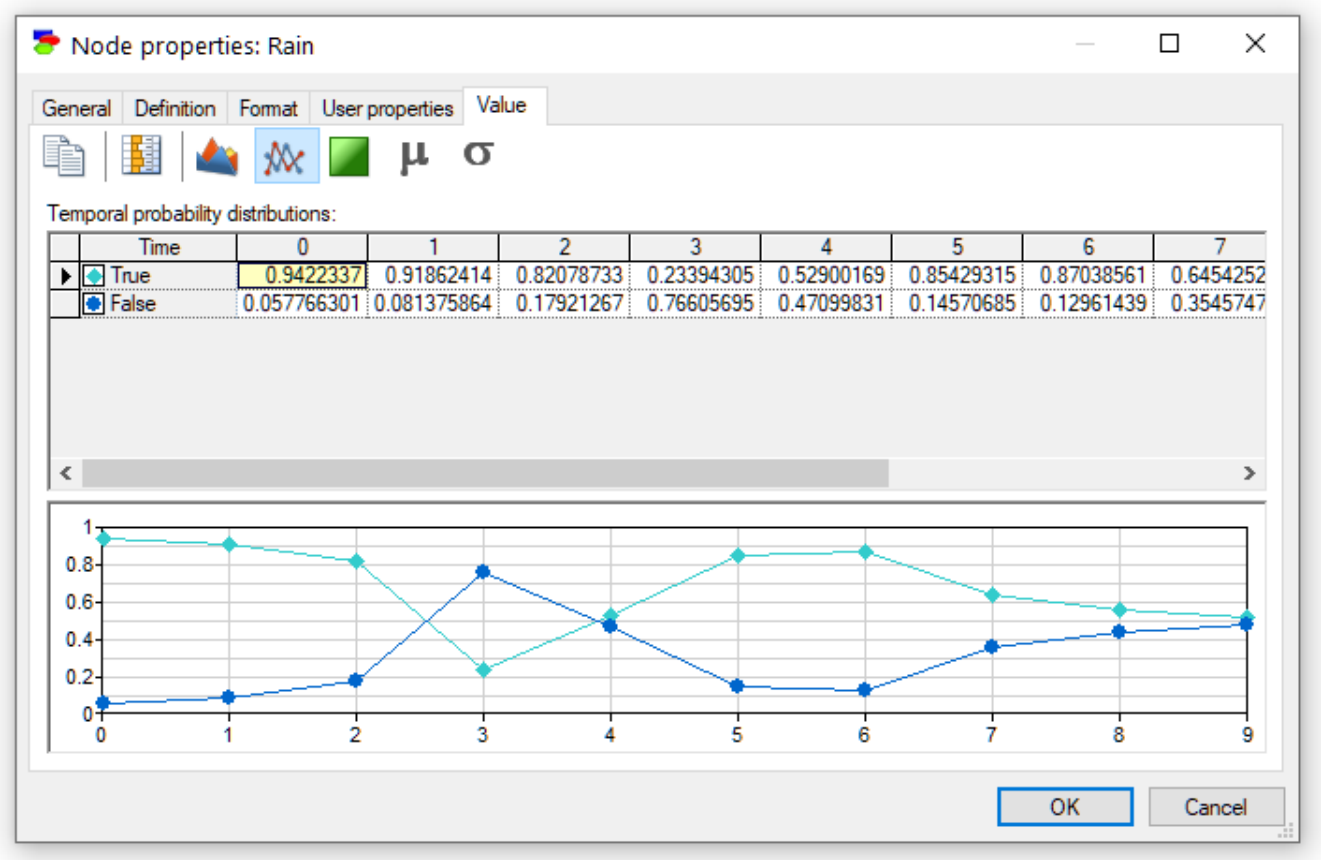
Please note that the *Umbrella* node is an evidence node. Its temporal beliefs are also well defined, albeit for those time slots for which there are observations, they are constant.

We can view the temporal beliefs in the *Value Tab* of a node as a spreadsheet indexed by the time steps. Selecting cells in the results spreadsheet and pressing the *Copy* ( ) button copies the cells for use outside of GeNIe.

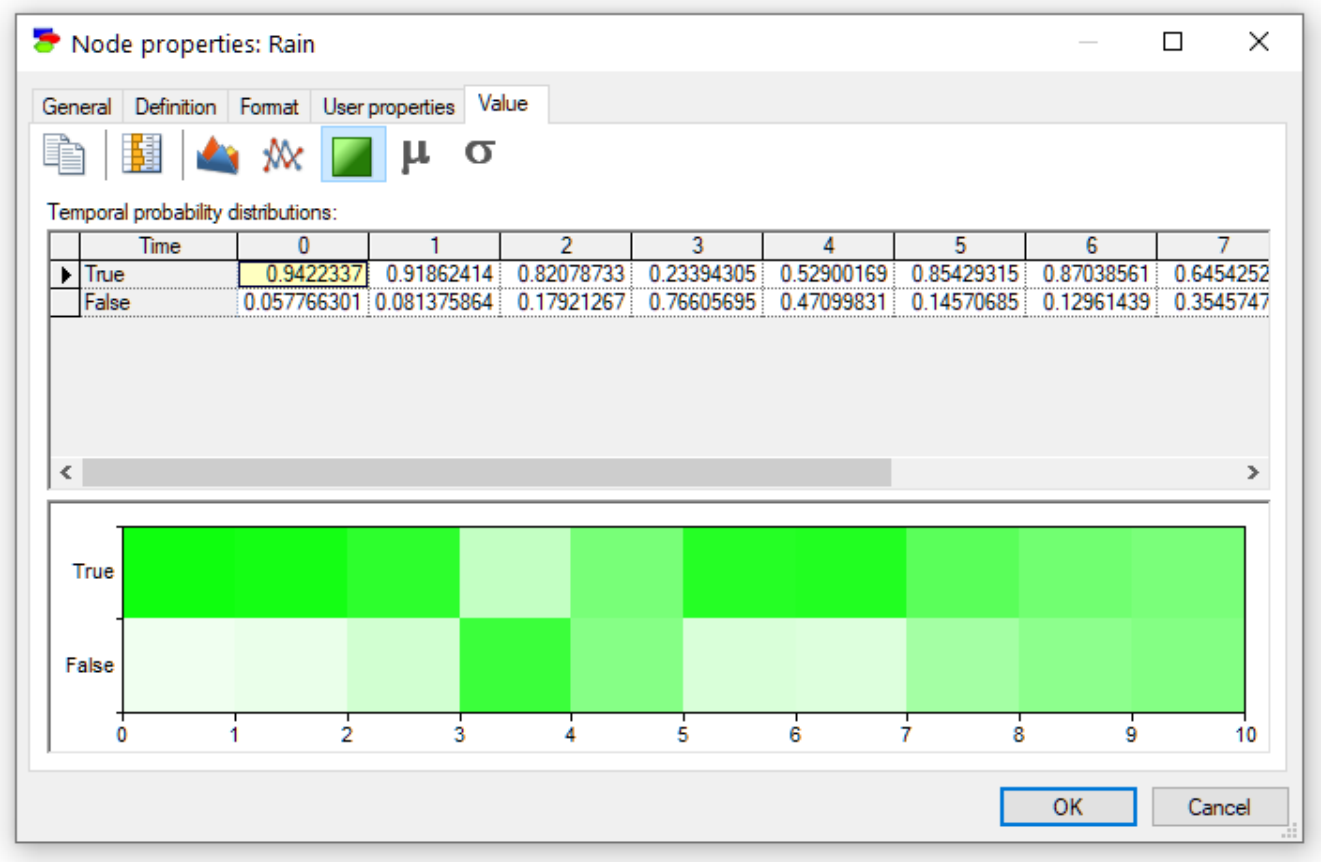
In addition to the numerical values of the marginal posterior probabilities over time, we can view the results graphically. Pressing the *Area chart* ( ) button displays the posterior marginal probabilities graphically:



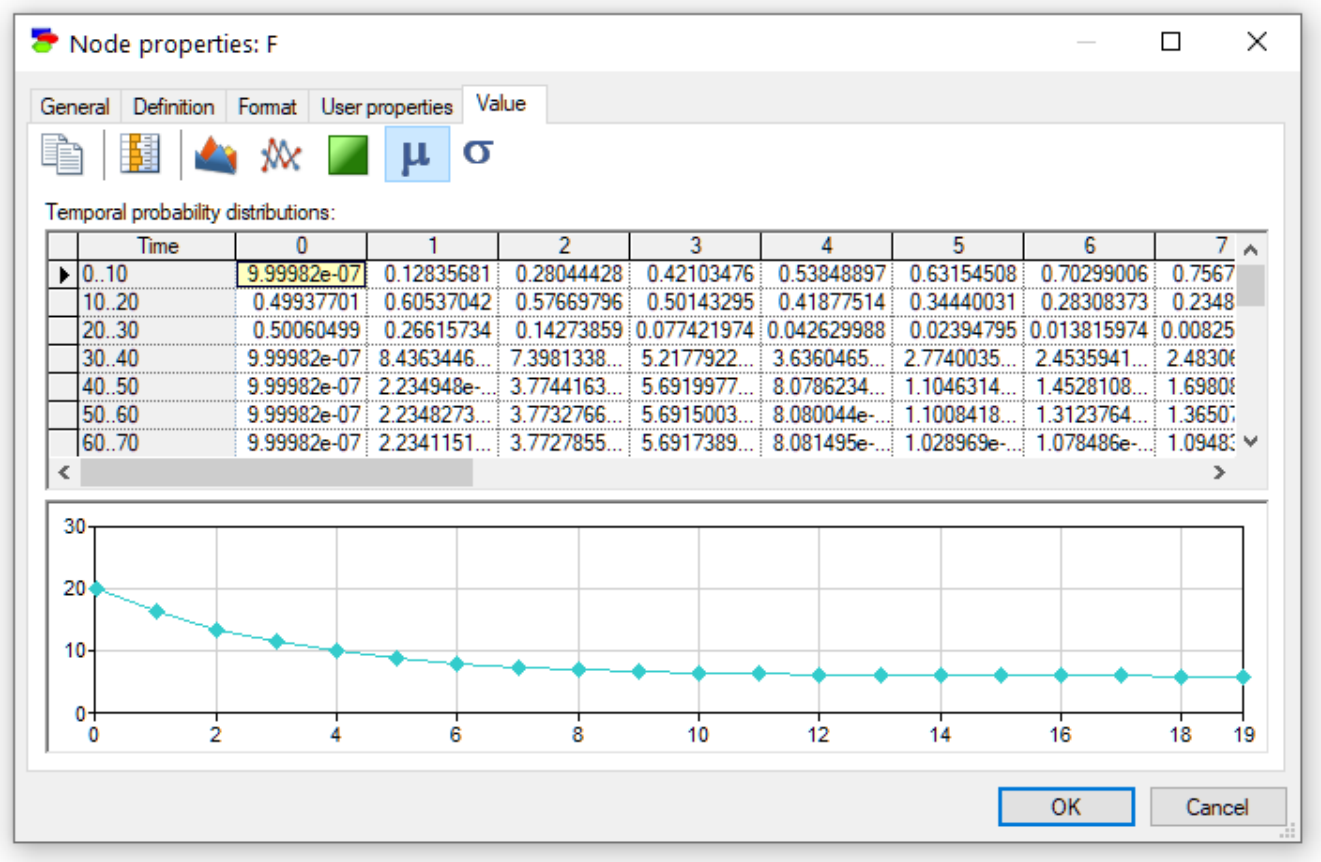
Pressing the *Time series* ( ) button shows the posteriors as a time series plot (a curve for every state of the variable):



Finally, pressing the *Contour plot* ( ) button displays the posterior marginal beliefs as a contour plot with probabilities displayed by colors. Hovering over individual areas shows the numerical probabilities corresponding to the areas/colors. The *Contour plot* is especially useful when the variable has many states and shows graphically the weight of probability mass.

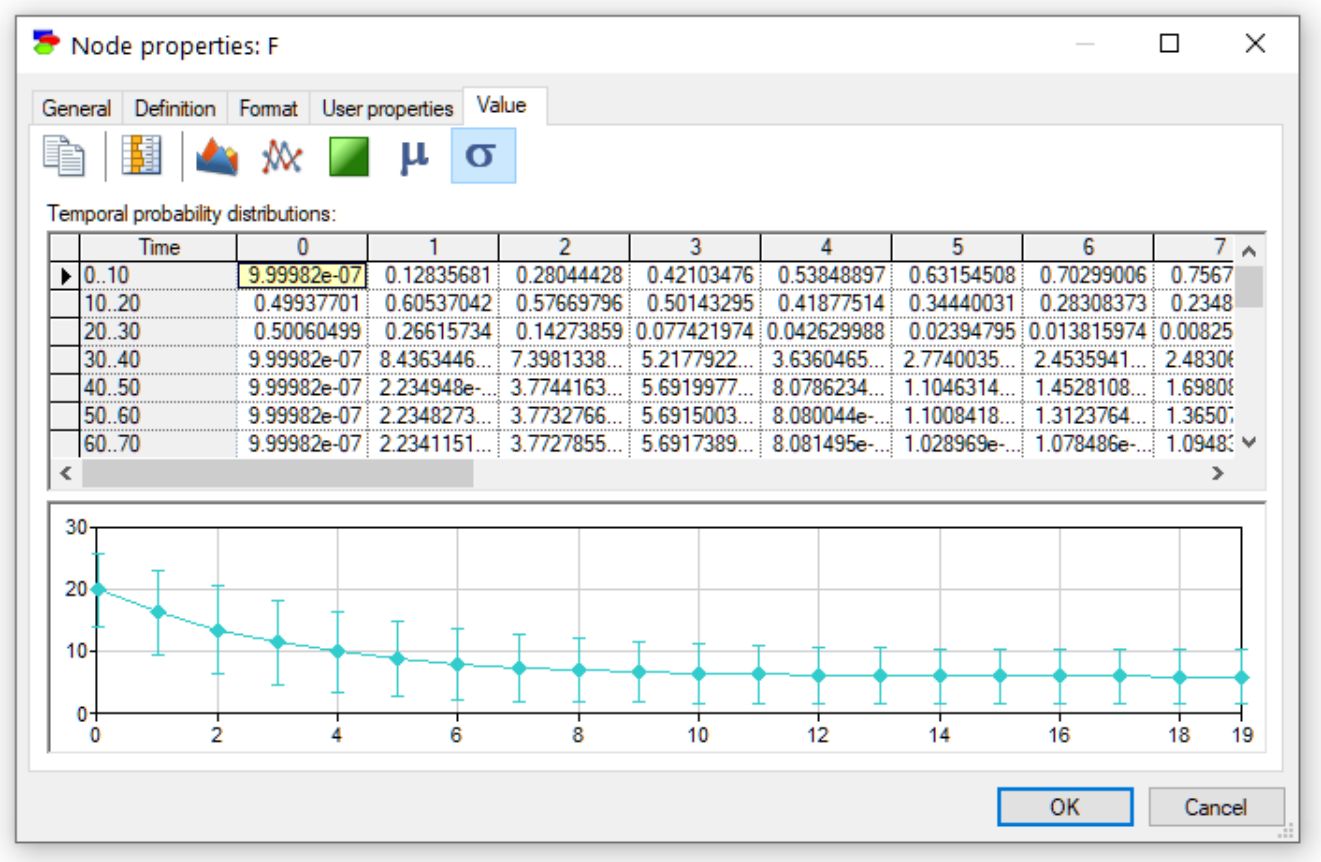


There is one more plot available in case of discrete numerical variables, notably plot of the mean of the variable as a function of time. To show that plot, please press the *Mean button* ().



Time	0	1	2	3	4	5	6	7
0..10	9.99982e-07	0.12835681	0.28044428	0.42103476	0.53848897	0.63154508	0.70299006	0.7567
10..20	0.49937701	0.60537042	0.57669796	0.50143295	0.41877514	0.34440031	0.28308373	0.2348
20..30	0.50060499	0.26615734	0.14273859	0.077421974	0.042629988	0.02394795	0.013815974	0.00825
30..40	9.99982e-07	8.4363446...	7.3981338...	5.2177922...	3.6360465...	2.7740035...	2.4535941...	2.48306
40..50	9.99982e-07	2.234948e-...	3.7744163...	5.6919977...	8.0786234...	1.1046314...	1.4528108...	1.69808
50..60	9.99982e-07	2.2348273...	3.7732766...	5.6915003...	8.080044e-...	1.1008418...	1.3123764...	1.36507
60..70	9.99982e-07	2.2341151...	3.7727855...	5.6917389...	8.081495e-...	1.028969e-...	1.078486e-...	1.09483

The plot can be enhanced by displaying the standard deviation around the mean. We can accomplish this by pressing the Standard deviation button (σ)

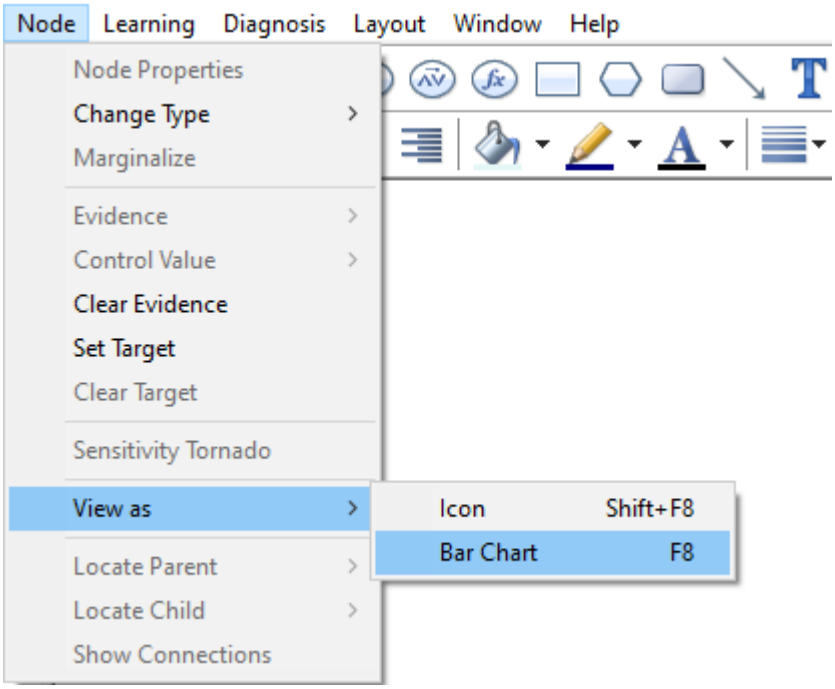


Time	0	1	2	3	4	5	6	7
0..10	9.99982e-07	0.12835681	0.28044428	0.42103476	0.53848897	0.63154508	0.70299006	0.7567
10..20	0.49937701	0.60537042	0.57669796	0.50143295	0.41877514	0.34440031	0.28308373	0.2348
20..30	0.50060499	0.26615734	0.14273859	0.077421974	0.042629988	0.02394795	0.013815974	0.00825
30..40	9.99982e-07	8.4363446...	7.3981338...	5.2177922...	3.6360465...	2.7740035...	2.4535941...	2.48306
40..50	9.99982e-07	2.234948e-...	3.7744163...	5.6919977...	8.0786234...	1.1046314...	1.4528108...	1.69808
50..60	9.99982e-07	2.2348273...	3.7732766...	5.6915003...	8.080044e-...	1.1008418...	1.3123764...	1.36507
60..70	9.99982e-07	2.2341151...	3.7727855...	5.6917389...	8.081495e-...	1.028969e-...	1.078486e-...	1.09483

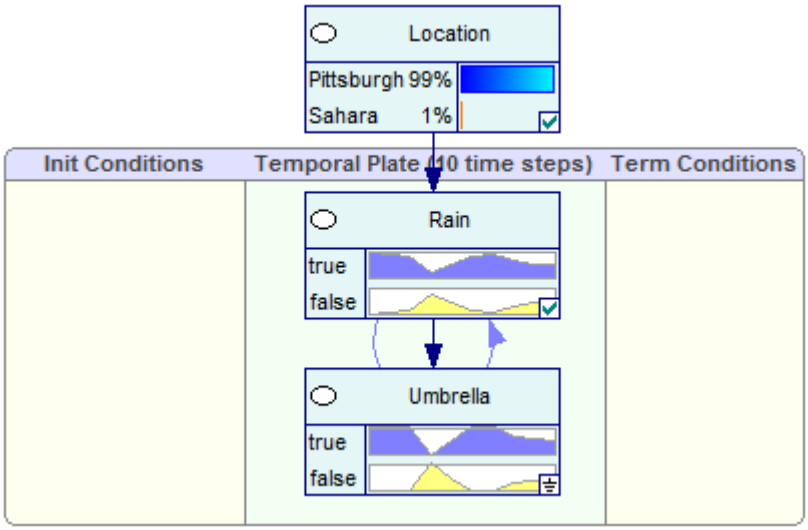
The pictorial representation of the temporal probability distributions can be copied and later pasted into another application for the purpose of documentation or reporting. To copy the pictorial representation of the temporal probability distributions, right-click on it, select *Copy*, and subsequently choose *Paste Special* in the destination application.

One should add that the relative size of the table and the plot area can be in each of the above plots adjusted by clicking and dragging on the divider line.

Marginal posterior probabilities can be also shown on the screen permanently by the changing the node view to *Bar chart*. This can be accomplished by selecting nodes of interest and then changing the view of the nodes through the *Node-View as-Bar Chart* option.

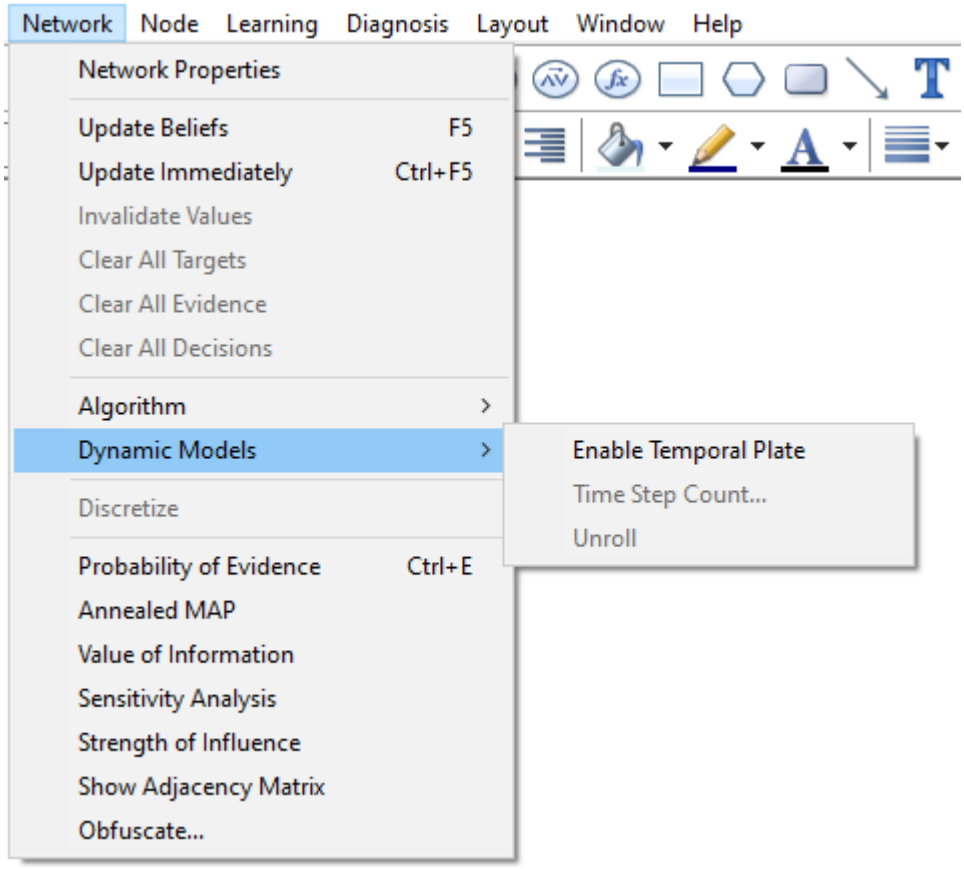


The *Bar chart* view allows for displaying the temporal posterior marginal probabilities on the screen permanently.

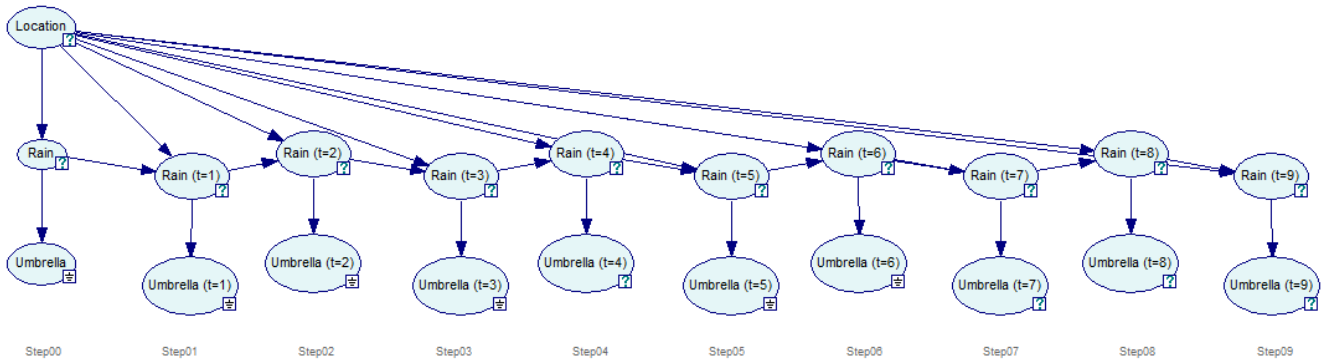


Unrolling the DBN

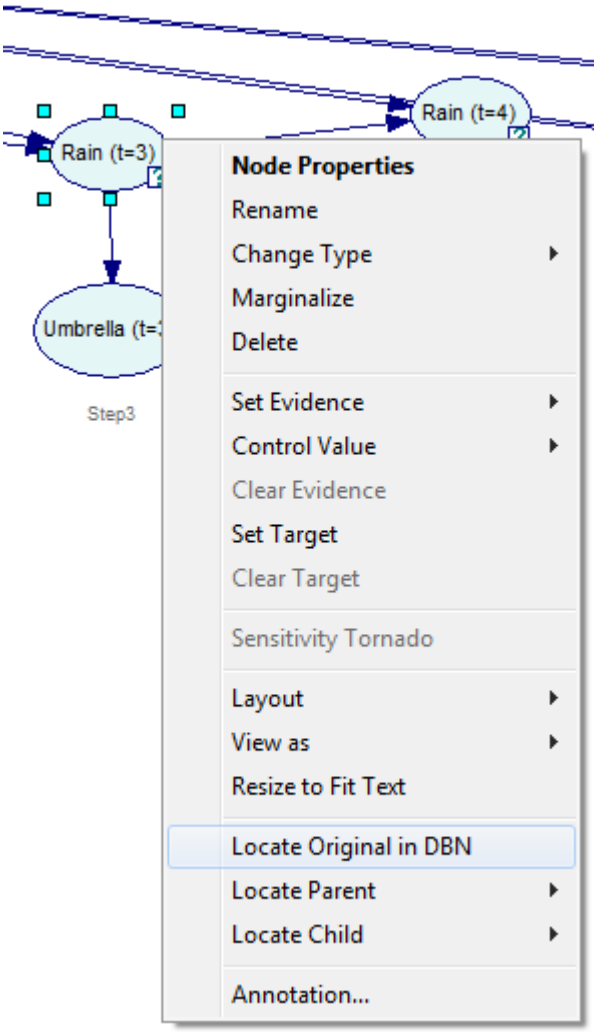
As we mentioned above, for the purpose of inference, GeNIe converts the DBN into a Bayesian network and updates the beliefs using the selected belief updating algorithm. It can be useful, for example for model debugging purposes, to explicitly unroll a temporal network. GeNIe provides this possibility through the *Network-Dynamic Models-Unroll* option.



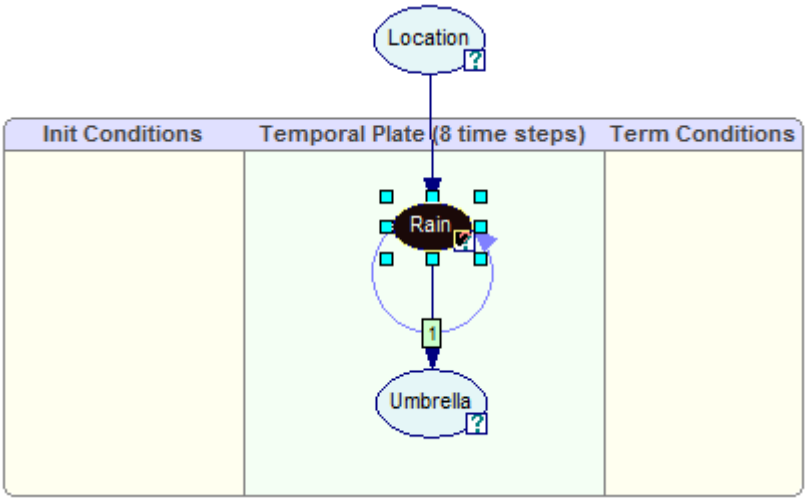
GeNIe creates a new network that has the temporal network unrolled for the specified number of time-slices. The unrolled network that is a result from unrolling the temporal network is cleared from any temporal information whatsoever. It can be edited, saved and restored just like any other static network. The screen shot below shows the unrolled network representation of a temporal network and how the original DBN can located back from the unrolled network.



It is possible to locate a node in the temporal network from the unrolled network by right-clicking on the node in the unrolled network and selecting *Locate Original in DBN* from the context-menu.



The node will be identified in the original DBN:



6.6.4 Learning DBN parameters

While GeNIe structure learning algorithms do not allow for learning the structure of dynamic models, it is possible to learn the parameters of existing DBNs from time series.

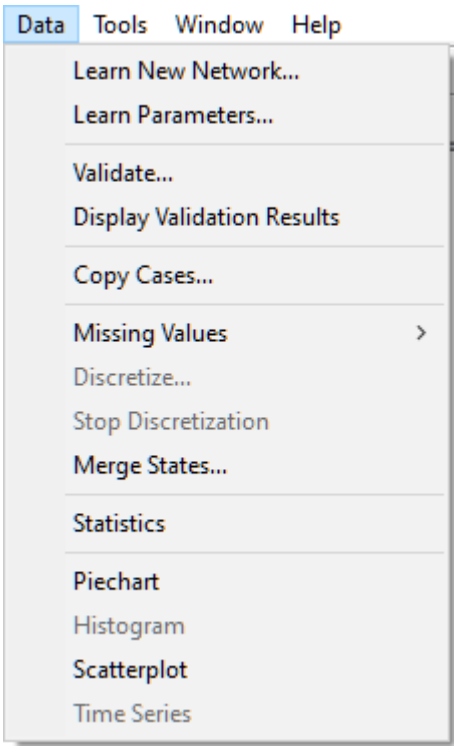
To learn parameters of an existing DBN (i.e., one for which the structure is already defined), you will need both, a data file and a network open. We will demonstrate the procedure of learning the parameters of a DBN from data on the network created in the section *Creating DBNs* and a corresponding data file, both available in the example models directory. Here is a screen shot of the data file opened in GeNIe:

R&N DBN Example.txt

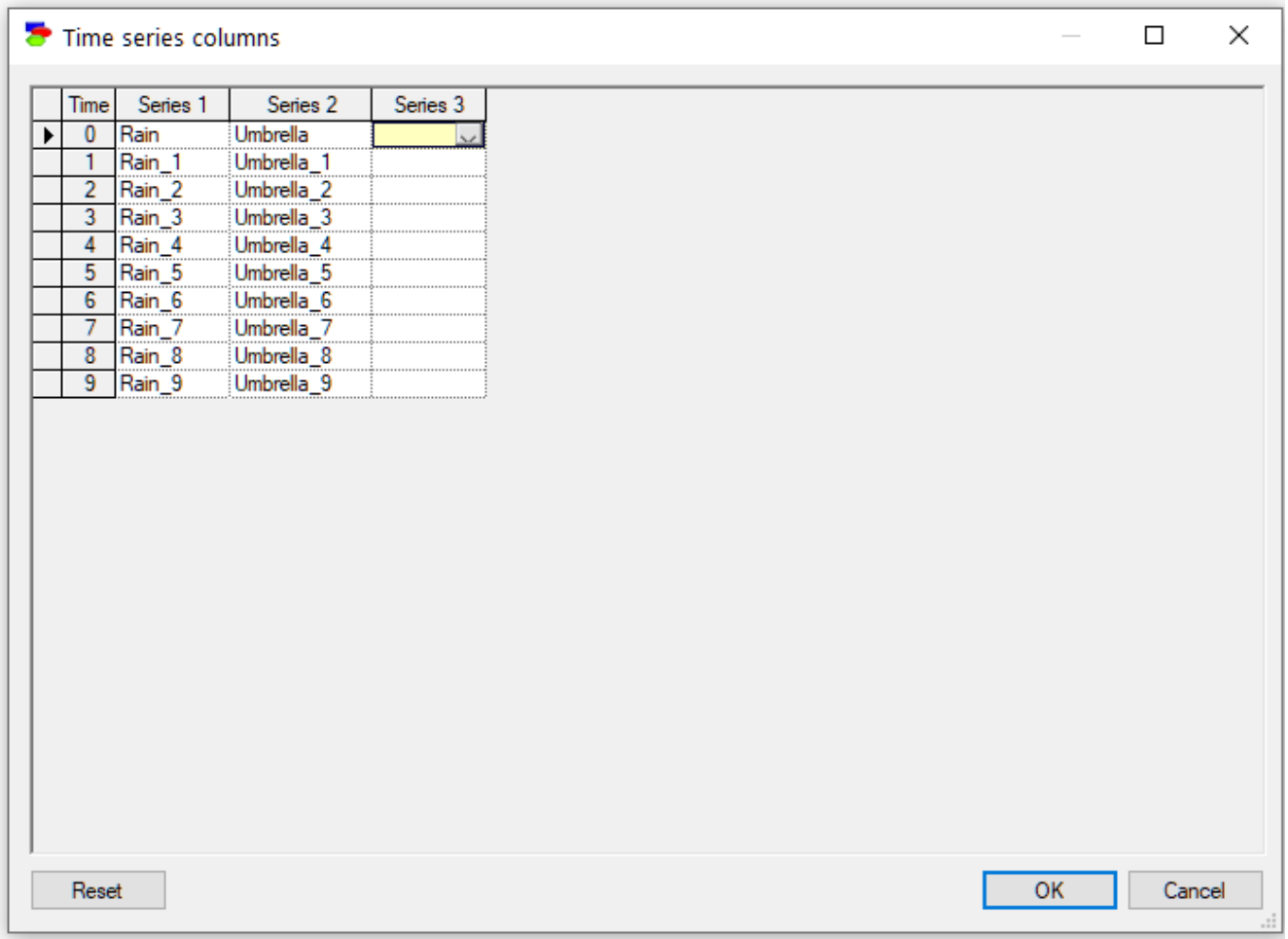
Location	Rain	Rain_1	Rain_2	Rain_3	Rain_4	Rain_5	Rain_6	Rain_7	Rain_8	Rain_9	Umbrella	Umbrella_1	Umbrella_2	Umbrella_3	Umbrella_4	Umbrella_5	Umbrella_6	Umbrella_7	Umbrella_8	Umbrella_9
Pittsburgh	true	true	true	true	true	true	true	false	false	true	false	true	true	true	true	true	false	false	false	true
Pittsburgh	false	false	false	false	false	true	true	true	true	true	true	true	true	true	false	true	true	false	true	false
Sahara	false	false	false	false	false	false	false	false	false	false	true	false	false	true	true	true	true	false	false	true
Sahara	false	false	false	false	false	false	false	false	false	false	false	false	true	false	true	false	true	false	false	false
Pittsburgh	false	true	true	false	false	false	true	false	false	false	false	true	true	false	false	false	true	false	false	false
Pittsburgh	true	false	true	false	false	false	true	true	true	true	true	false	true	true	true	false	true	true	true	true
Sahara	true	false	false	false	false	false	false	false	false	false	true	false	true	false	false	false	false	false	false	false
Sahara	false	false	false	false	false	false	false	false	false	false	false	false	false	true	true	true	false	false	false	false
Sahara	false	false	false	false	false	false	false	false	false	false	true	true	false	false	false	false	false	false	true	false
Pittsburgh	false	true	true	true	true	true	true	true	false	false	false	false	true	true	false	true	true	true	false	false
Sahara	false	false	false	false	false	false	false	false	false	false	false	false	false	true	true	false	false	false	false	false
Pittsburgh	false	true	false	false	false	false	true	true	false	false	false	true	false	false	false	false	true	true	false	false
Pittsburgh	true	false	true	true	true	true	true	true	true	false	true	false	true	true	true	true	true	true	true	false
Sahara	false	false	false	false	false	false	false	false	false	false	true	true	true	false	false	true	false	false	false	false
Pittsburgh	true	false	false	false	false	false	false	false	true	true	true	false	false	false	false	true	false	false	true	true
Sahara	false	false	false	false	false	false	false	false	false	false	false	true	false	false	false	false	false	true	false	false
Pittsburgh	true	false	false	true	false	false	false	false	true	true	true	false	false	true	false	false	false	false	true	true
Pittsburgh	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true
Sahara	false	false	false	false	false	false	false	false	false	false	false	false	false	false	true	false	true	false	false	true
Sahara	false	false	false	false	false	false	false	false	false	false	false	false	false	false	true	false	false	false	false	false
Sahara	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	false	true

The file used for learning the parameters of a DBN has to follow a simple but strict format. The labels have to correspond to the IDs of the nodes in the network. Measurements taken at different time steps have to be labeled by the node ID with an underscore character followed by the time step number (except for time step zero, which has no time step mark).

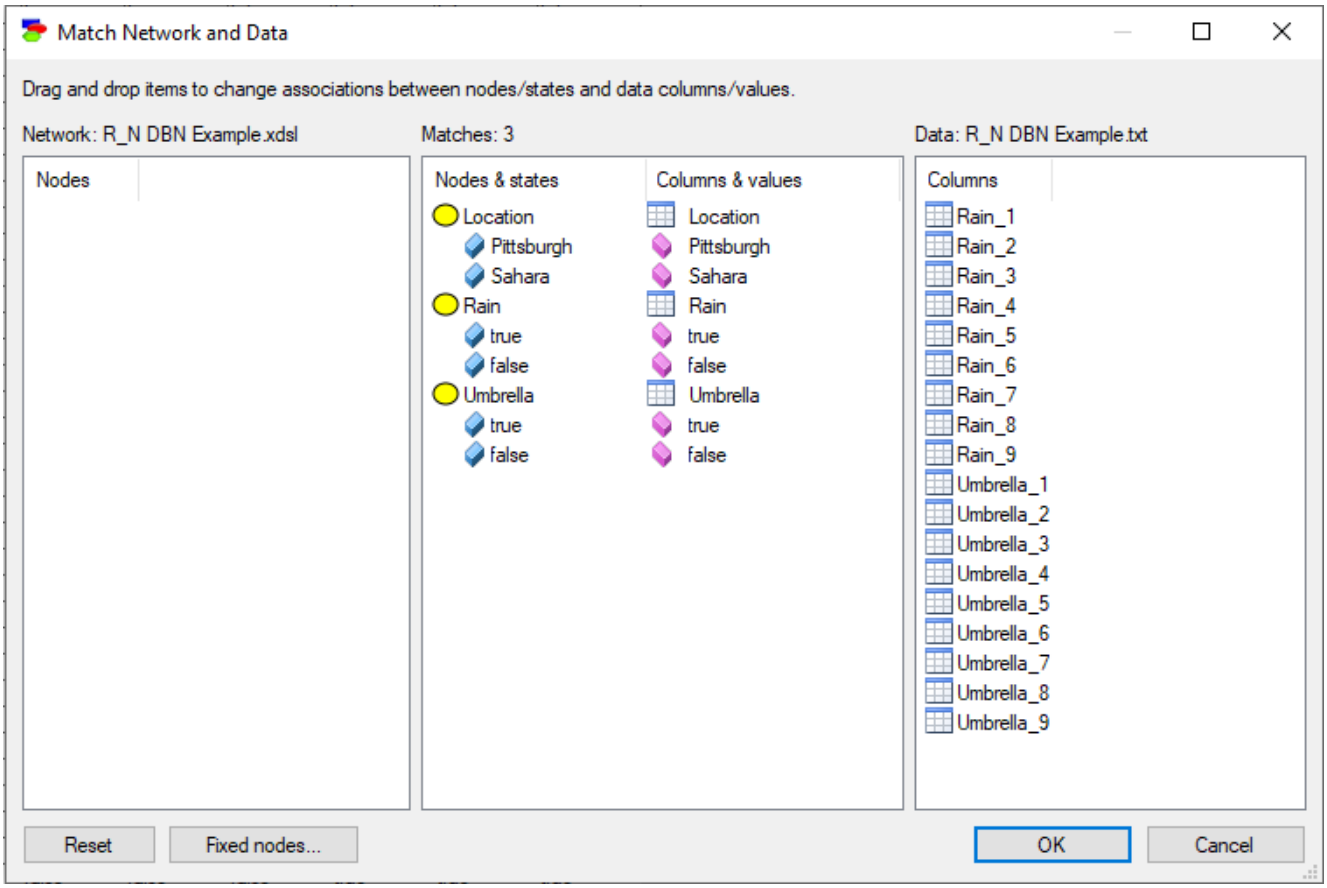
Once you have opened both the model and the data, select *Learn Parameters...* from the *Data* menu.



This will invoke the *Time series columns* dialog that allows for double-checking the matching between time steps and labels in the data set.

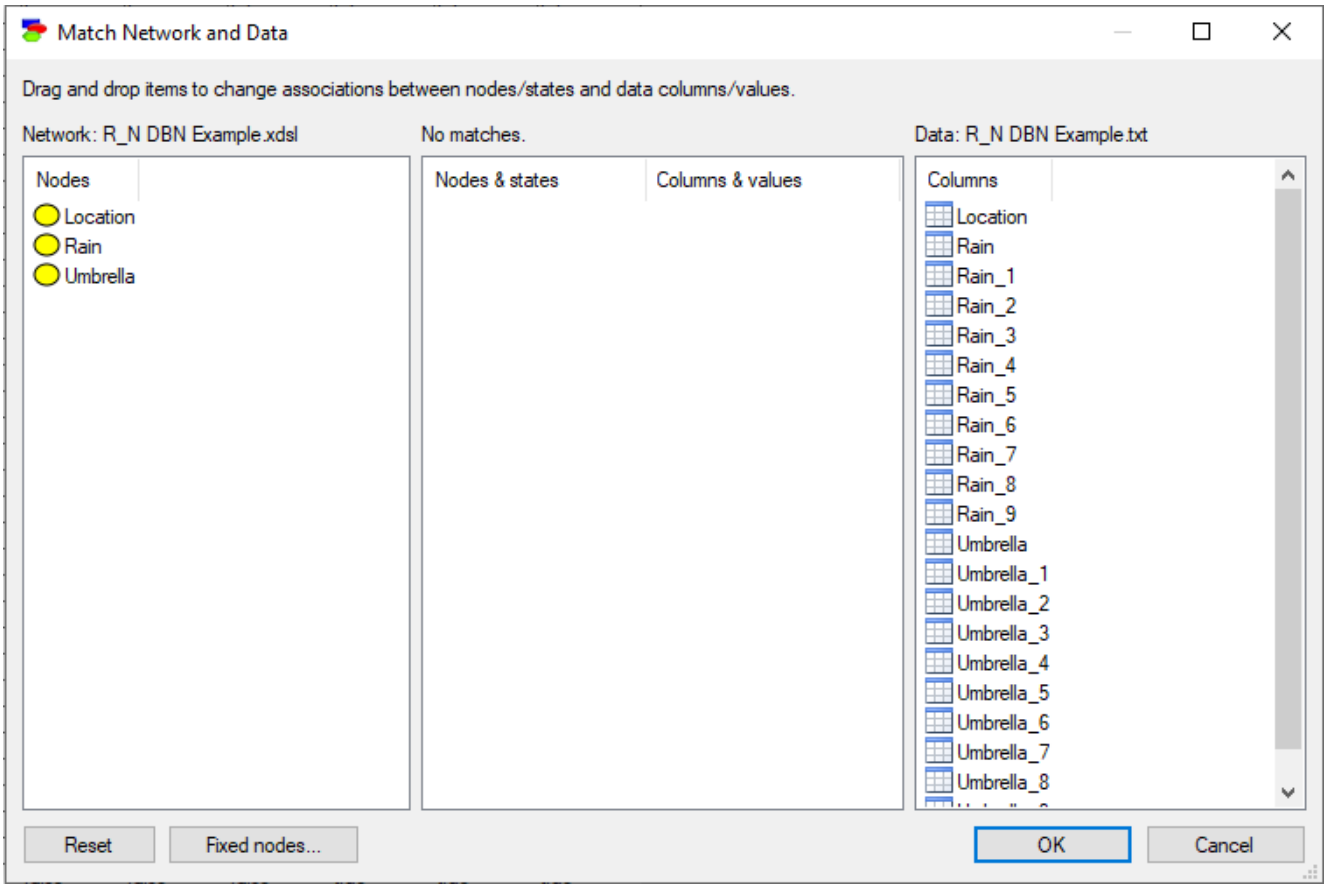


Closing this dialog invoke the *Match Network and Data* dialog that serves to create a mapping between the variables defined in the network (left column) and the variables defined in the data set (right column).



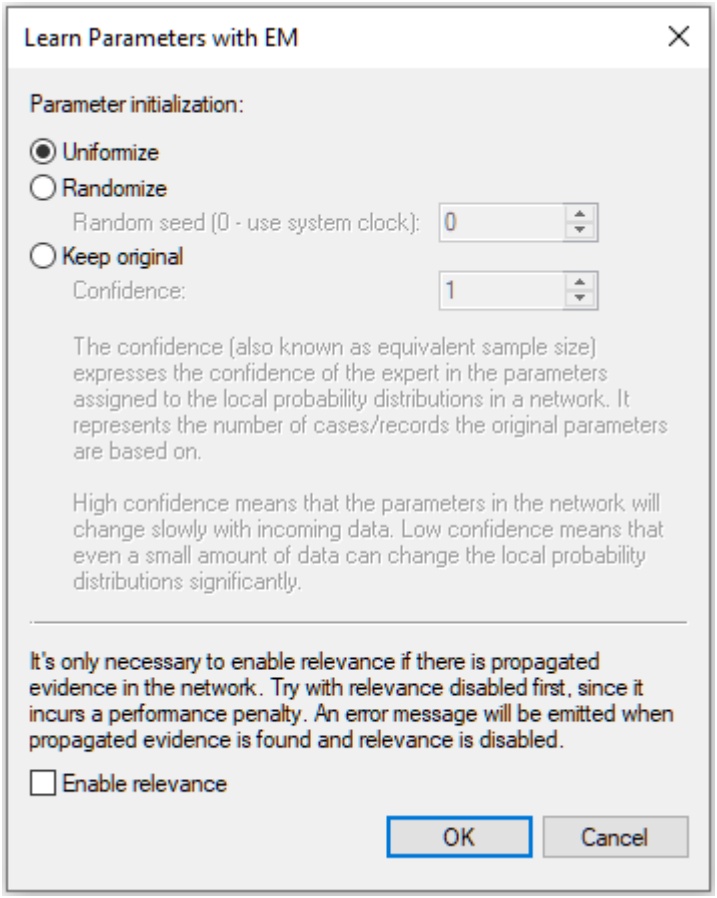
Both lists of variables are sorted alphabetically. The *Match Network and Data* dialog does text pre-matching and places in the central column all those variables and their states that match (have identical or close to identical names). If there is any disparity between them, GeNIe highlights the differences by means of a yellow background, which makes it easy to identify disparities. Manual matching between variables in the model and the data is performed by dragging and dropping (both variables and their outcomes). To indicate that a variable (or its state or its state in the middle column) in the model is the same as a variable in the data, simply drag-and-drop the variable (or its state in the middle column) from one to the other column. Please note that it is only necessary to match the variables in the model to the columns that correspond to the time step zero in the data. Other times steps (double-checked in the *Time series columns* dialog) will follow.

To start the matching process from scratch, use the *Reset* button, which will result in the following matching:



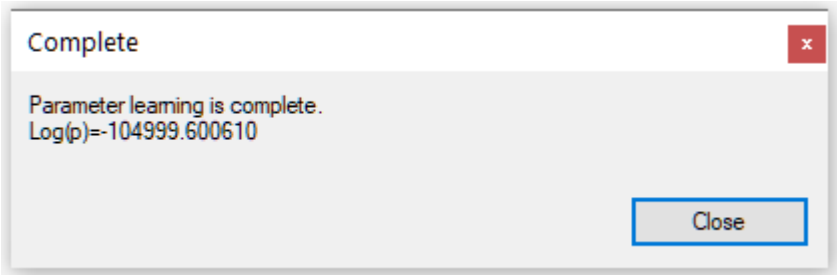
Fixed nodes... button, described in detail in section *Learning parameters*, invokes a dialog that allows for excluding nodes from the learning process.

Once you have verified that the model and the data are matched correctly, press *OK*, which will bring up the following dialog:



This dialog is described in detail in section *Learning parameters*.

Once we press *OK*, the EM algorithm updates the network parameters following the options chosen and comes back with the following dialog:



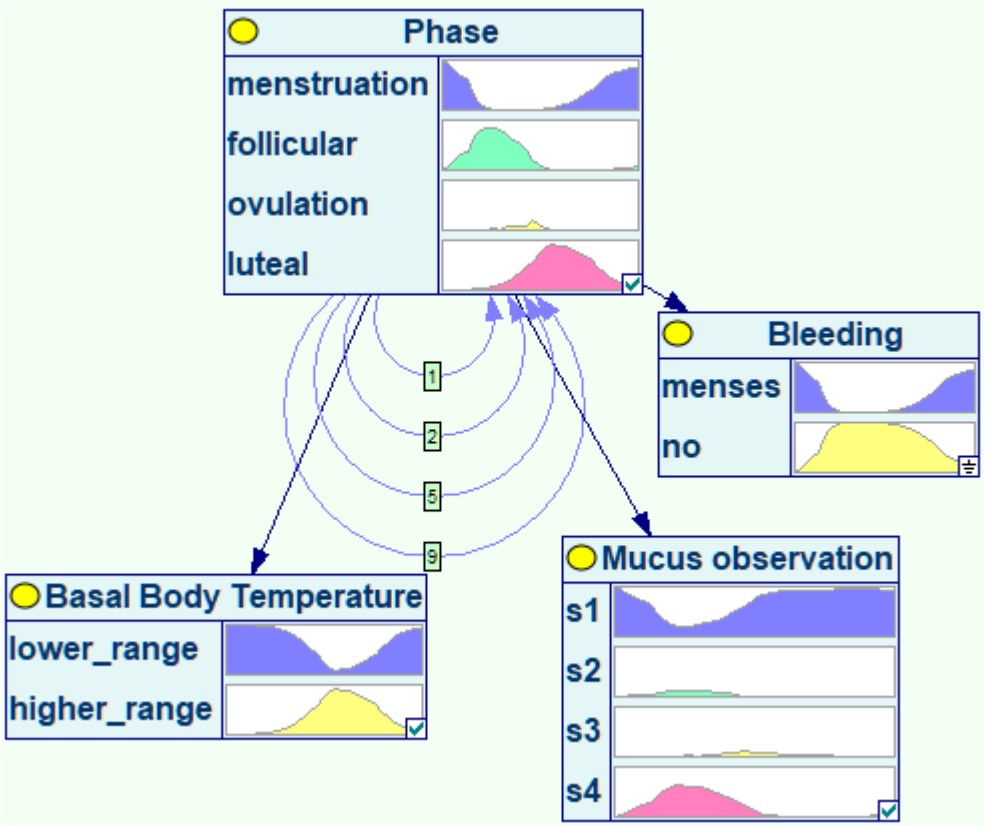
Log(p), ranging from minus infinity to zero, is a measure of fit of the model to the data.

A remark on the network structure and also on existing evidence. Learning parameters functionality focuses on learning parameters, not the structure, which is assumed fixed and will be unaffected. Existing evidence in the network is ignored and has no effect on the learned parameters.

6.6.5 Higher order influences

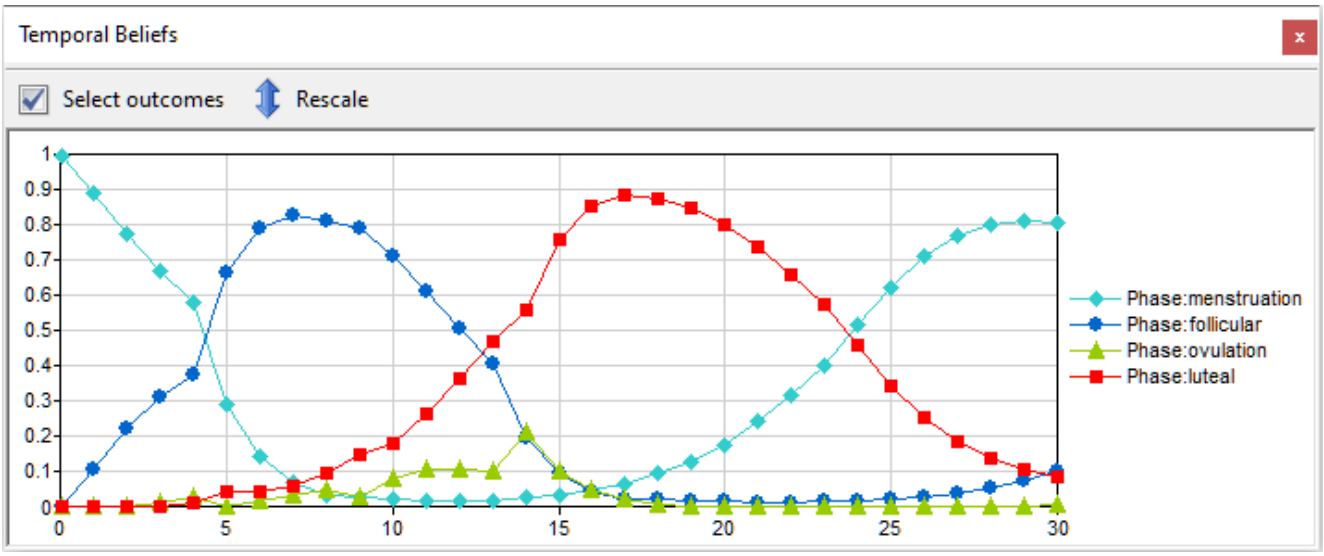
DBN, as implemented in GeNIe, allow for temporal arcs of any order, which means that DBNs in GeNIe can model dynamic processes of any order. This, to our knowledge, is unique among graphical modeling software.

The following example of a DBN originates from the doctoral research conducted by Dr. Anna Lupinska-Dubicka (2014) and is available from BayesFusion's interactive on-line model repository. It models woman's monthly cycle and helps with predicting the day of ovulation, with applications to fertility awareness and difficulties with conception.



Woman's monthly cycle is a dynamic system with memory spanning of the period of typically 28 days (it varies among women and for the woman modeled the the above model, it is 30). It means that to model it correctly, one has to relax the Markovian assumption that the current state of the system is determined only the state of the system one step back. On the first day of the monthly cycle (typically, it is the first day of the monthly bleeding), it is more or less known when ovulation will happen and when the cycle will end (this is by definition the last day before the next monthly bleeding). The above model uses influences of the 1st, 2nd, 5th, and 9th order, which turned out to be sufficient to model transitions between the four phases of the monthly cycle and to predict both ovulation and the end of the cycle.

The following plot, derived in GeNIe, shows the probability distribution over the states of the variable Phase as a function of time. The only evidence entered into the network is bleeding (*menses*) on Day zero. The plot resembles qualitatively typical plots of concentrations of various hormones in woman's body, indicative of the current phase of the cycle.


Temporal Beliefs
Select outcomes
Rescale
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
Phase:menstruation
Phase:follicular
Phase:ovulation
Phase:luteal

6.7 Equation-based and hybrid models

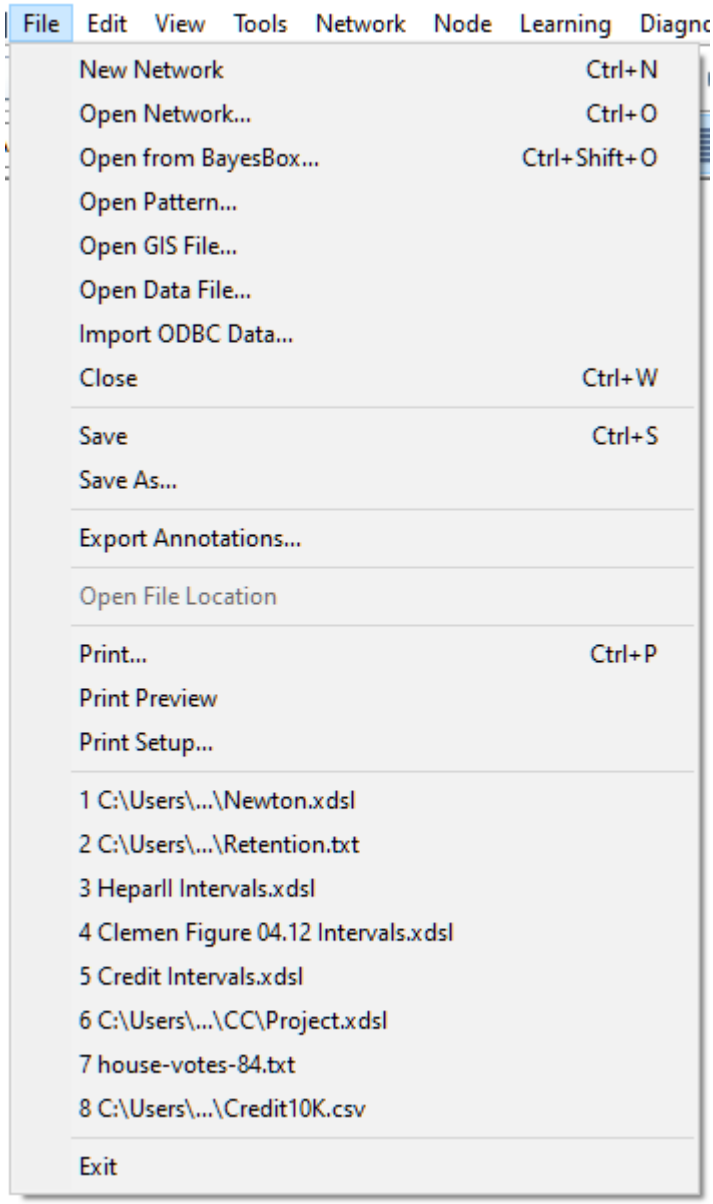
6.7.1 Introduction

It is often forgotten that graphical models, such as Bayesian networks, are not necessarily built of only discrete variables. They are, in fact, close relatives of systems of simultaneous structural equations. GeNIe allows for constructing models that consist of equation nodes. These are alternative, graphical representations of systems of simultaneous structural equations.

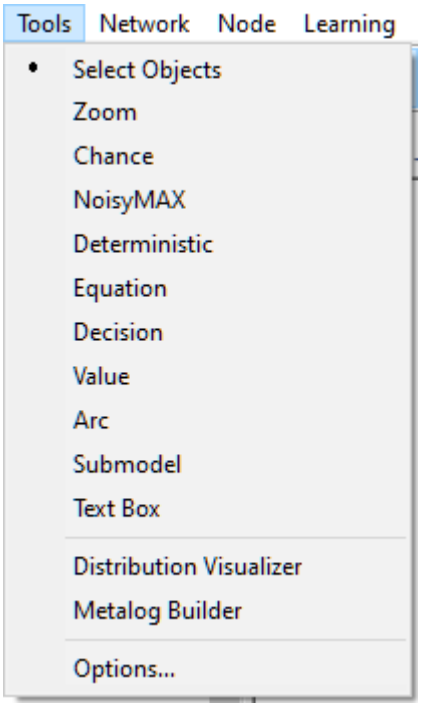
The following sections cover the process of constructing, inference, and viewing results in equation-based and hybrid models, i.e., models that contain both discrete and equation (continuous) nodes.

6.7.2 Constructing equation-based models

Equation-based models consist of *Equation* nodes. To construct an equation-based model, add *Equation* nodes to the *Graph View* and add connections between them. Let us create a simple model describing an object of mass m under influence of a force f, receiving acceleration a, which is governed by Newton's 2nd law of motion. We start by selecting *New Network* from the *File Menu*

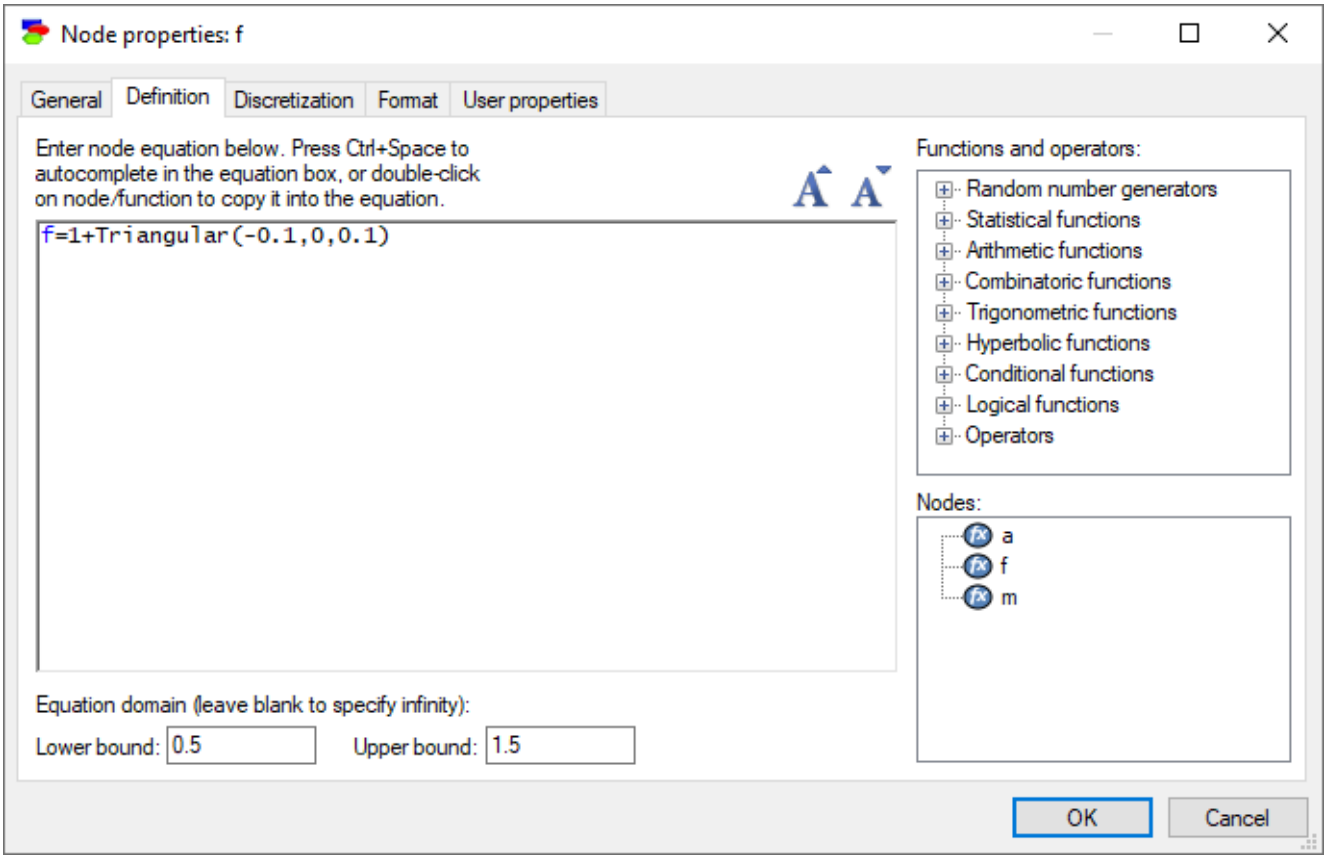


We proceed by dropping three *Equation* nodes in the *Graph View* window using the *Tools Menu*



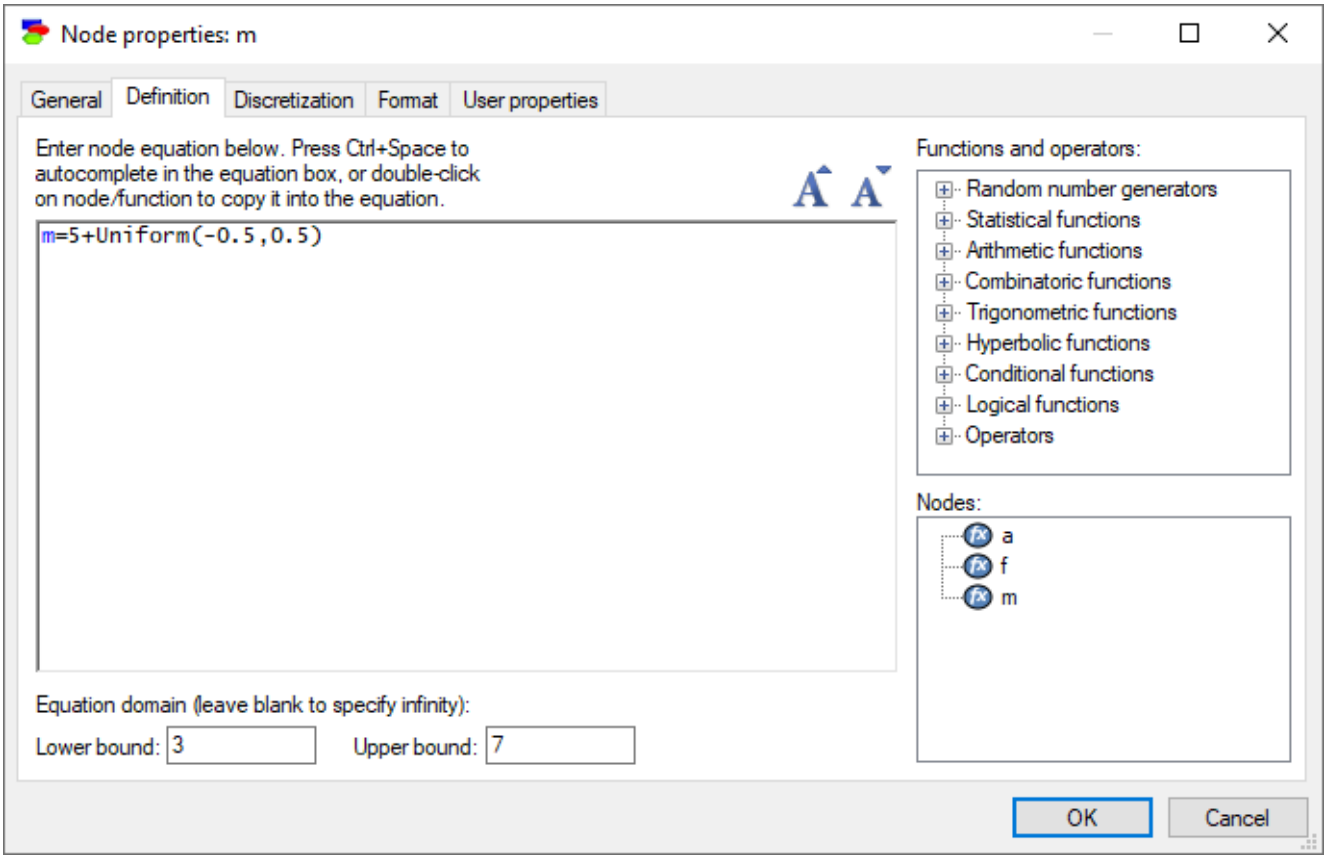
or the *Equation* ( ) button from the [Standard Toolbar](). We name them *f*, *m*, and *a*.

We define them as follows. Let the force be a constant (*f*=1) with some noise that we express by means of a *Triangular(-0.1,0,0.1)* distribution.



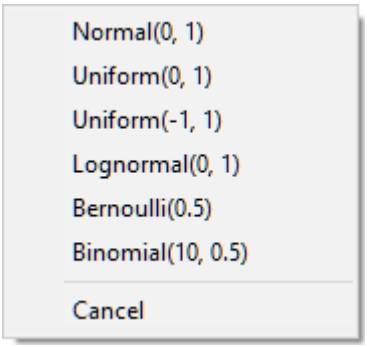
We specify the plausible domain of values of *f* to be between 0.5 and 1.5 Newtons.

Let the mass be a constant (*m*=5) with some noise that we express by means of a *Uniform(-0.5,0.5)* distribution.



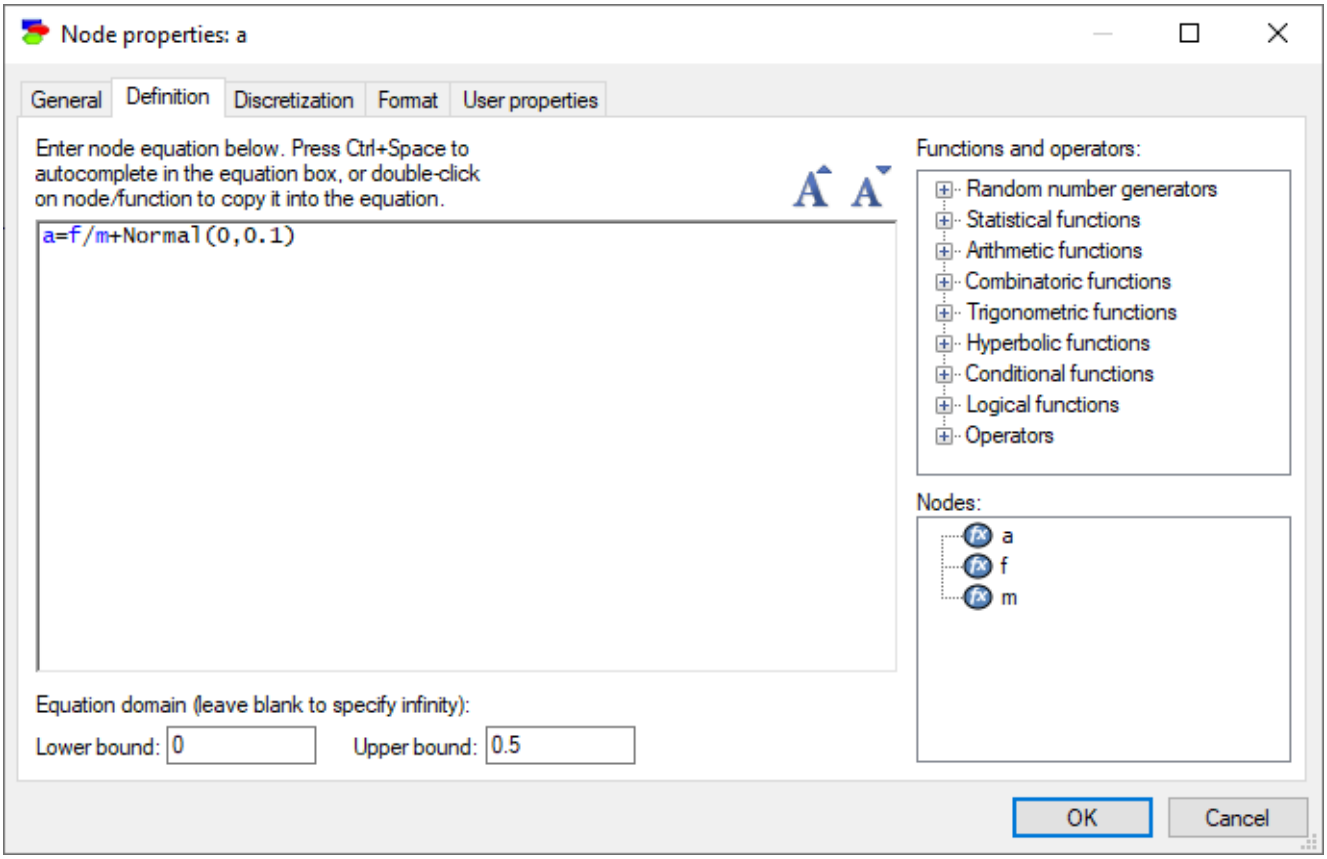
We specify the plausible domain of values of *m* to be between 3 and 7 kilograms.

You may also select the definition of the node from a *Quick definitions* list at the time of creating a new equation node in the *Graph View*. To do so, select (*Equation button*) from the *Standard Toolbar* or *Tools Menu*. The *Equation button* will become recessed and the cursor will change to an arrow with an ellipse (with a wave sign) in bottom right corner. Move the mouse to a clear portion of the screen inside GeNIe window, hold the *SHIFT* key on the keyboard and click the left mouse button. The following menu with *Quick definitions* list for the nodes will pop up:

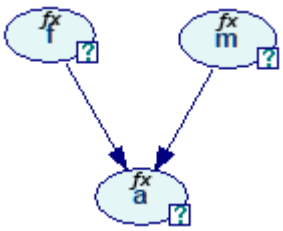


Selecting any of the items on the list creates nodes with the selected definition, which can be subsequently modified.

We define the third variable, acceleration *a*, as a function of *f* and *m* using Newton's 2nd law of motion and adding some noise, expressed by means of a *Normal(0,0.1)* distribution. We estimate the plausible domain of values of *a* to be between 0 and 0.5.



The graph of the model in the *Graph View* changes - arcs are added from the variables *f* and *m* to *a*.

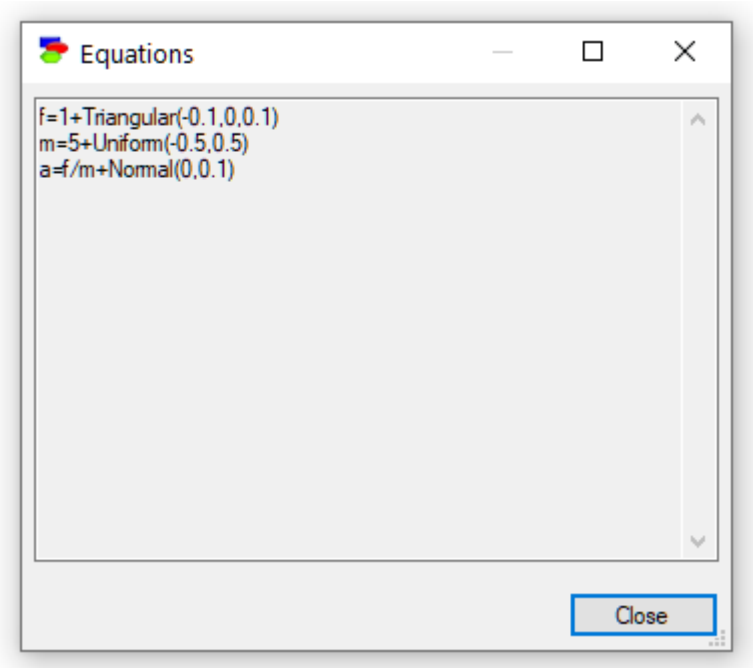


The model constructed corresponds to the following system of six simultaneous structural equations with six variables:

```
ε_f=Triangular(-0.1,0,0.1)
ε_m=Uniform(-0.5,0.5)
ε_a=Normal(0,0.1)
f=1+ε_f
m=5+ε_m
a=f/m+ε_a
```

The system could be simplified to three structural equations with three variables if we replaced variables ε with just references to probability distributions, like we did in the Bayesian network model constructed in this section. The distributions used (*Triangular*, *Uniform*, and *Normal*) may not be physically plausible in this example. We just wanted to use them to show that GeNIe puts no limitations on the functional form and the distributions used in the equations. Any function and any distribution available in the *Functions and operators* pane on the right-hand side can be used in the definition.

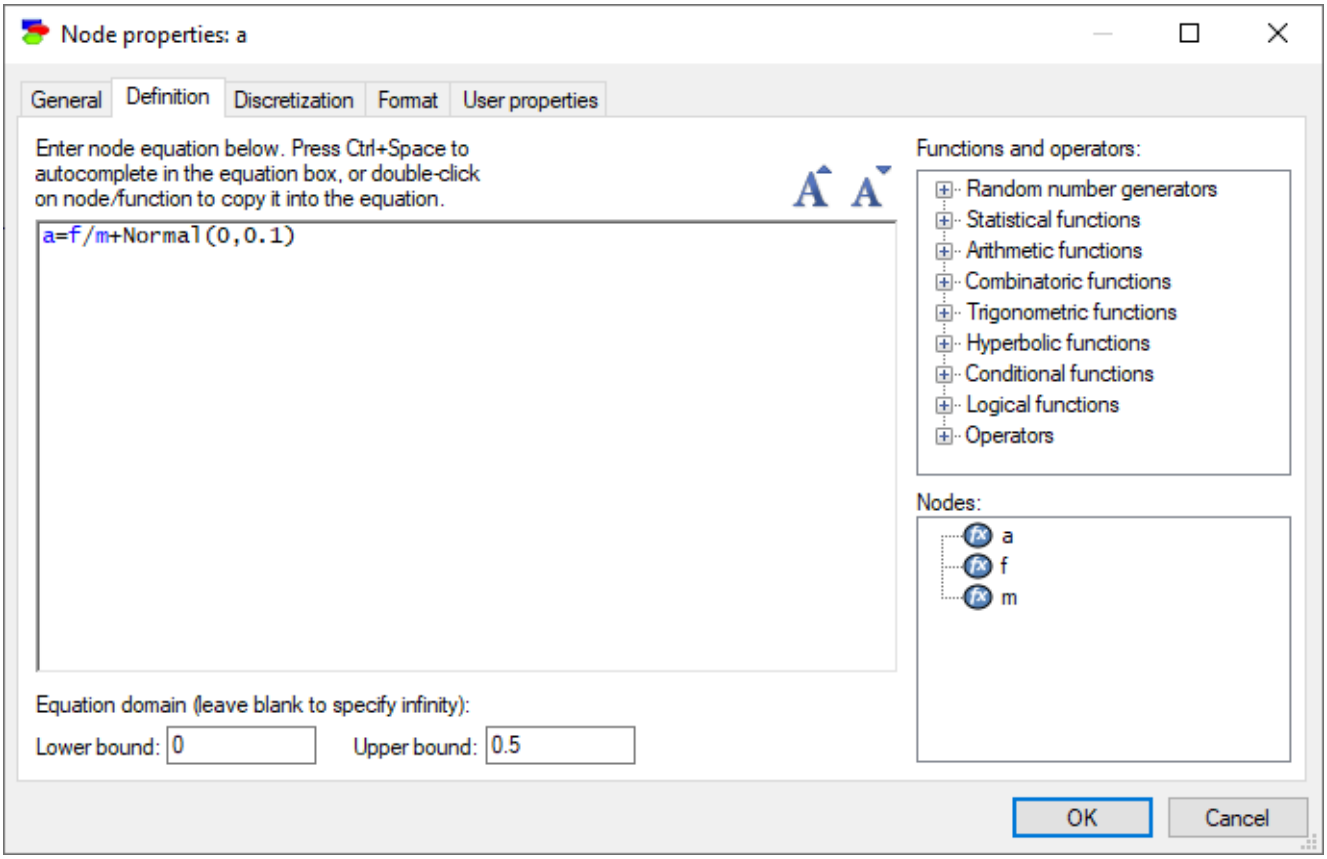
One can display the equations embedded into a hybrid model by choosing *Show Equations* from the *Network Menu*. This will display a dialog with all equations (in their simplified form) extracted from the model. The dialog invoked for the above model will look as follows:



6.7.3 Writing equations in GeNIe

6.7.3.1 Introduction

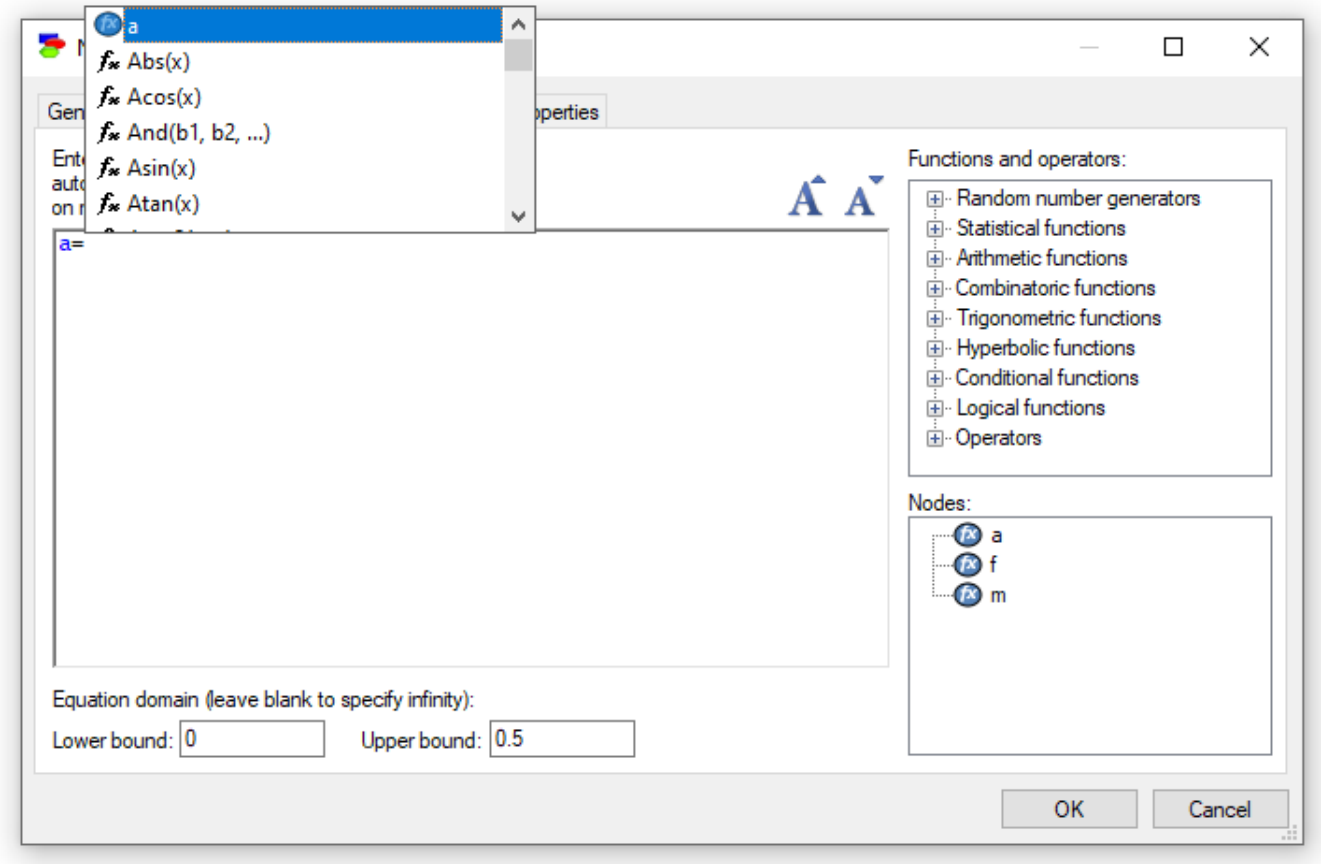
Equations in GeNIe may involve variables (nodes), their states, numerical and text constants, tied with each other by means of operators and functions. The following *Definition* dialog shows an explicit equation defining the variable *a* as a function of two variables, *f* and *m*, and a constant representing normal noise in the interaction.



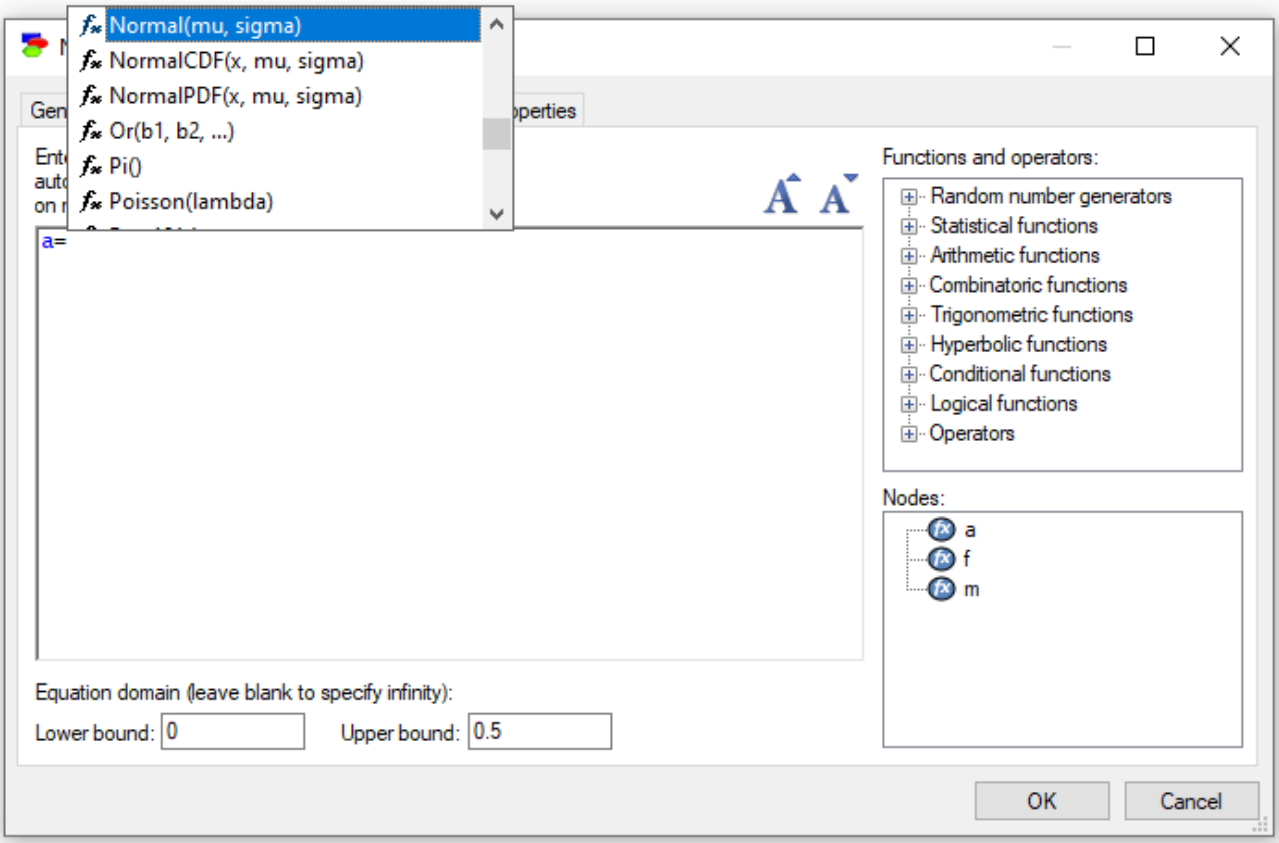
The lower-right pane of the dialog shows the list of variables/nodes involved in the interaction (this list includes the current node and all its parents). Adding another node existing in the model is possible, even if the node is currently not a parent of the current node. The upper-right pane shows a list of functions and probability distributions available in GeNIe. Selecting them (by a mouse click) places them in the dialog and allows for editing their formal parameters.

Auto-completion

Pressing *Ctrl+Space* when editing an equation prompts for allowable elements. Normally, *Ctrl+Space* will pop-up an alphabetically ordered list of GeNIe functions.

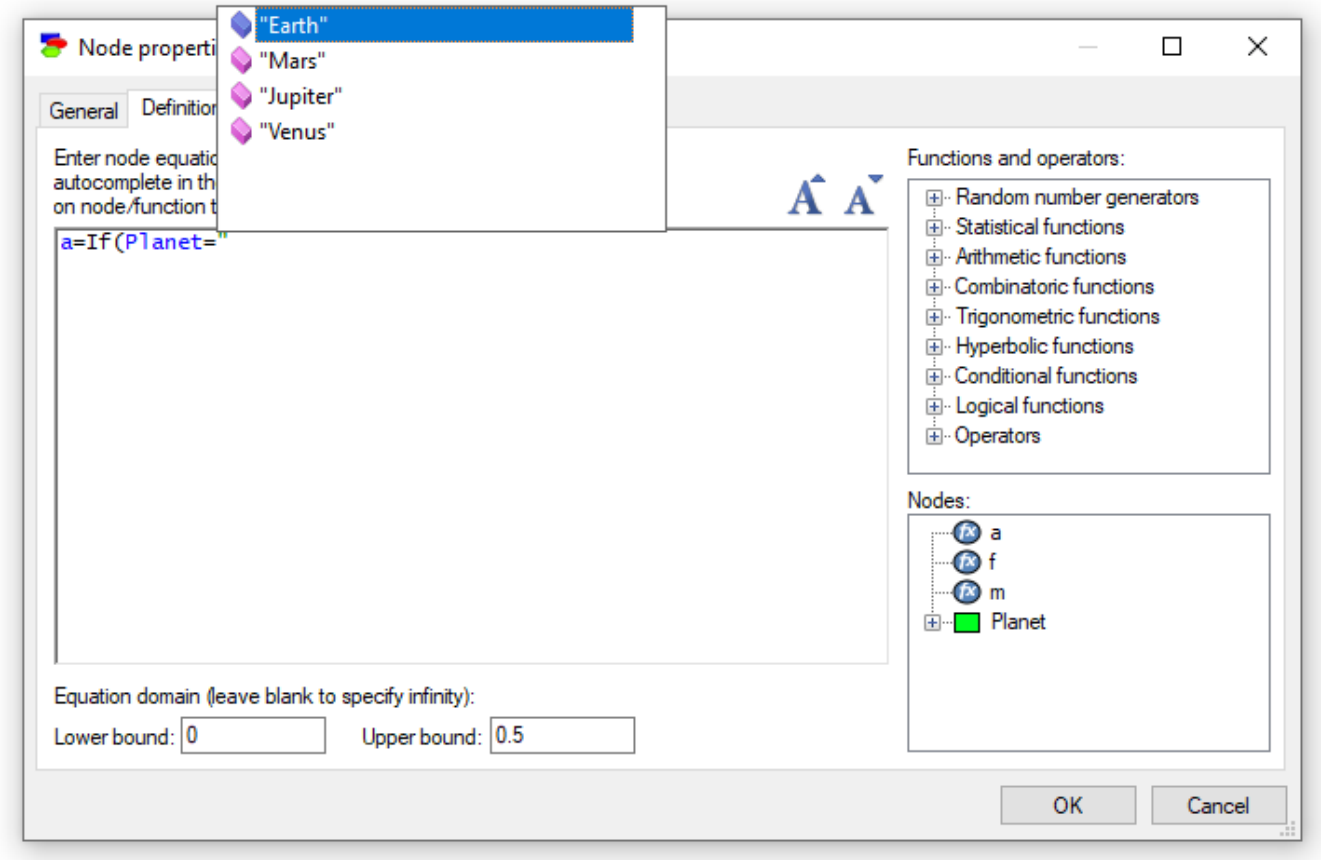


Pressing a letter will show functions starting with that letter. Pressing *n*, for example, yields



Typing several letters in a sequence will position the selection at the element starting with the prefix typed.

Ctrl+Space helps also with selecting states of discrete nodes. Let a parent of *a* be a variable *planet* with four outcomes: *Earth*, *Mars*, *Jupiter*, and *Venus*. Pressing *Ctrl+Space* after typing the double quote in the equation box opens an auto-complete list of relevant state names.



This feature does not perform parsing of the entire expression, only tokenization, so the resulting expression may not always be valid. Pressing *OK* at the end of editing will perform full validation and check whether the equation is syntactically valid.

Numerical equivalent of state IDs

While all expressions evaluate ultimately to numbers, there is one translation to numbers that deserves special attention. It is the translation of state IDs of discrete nodes to their indexes. When used in expressions, discrete state IDs translate naturally to integers between 0 and *n*-1, where *n* is equal to the number of states. The integer number corresponding to a state ID is determined by the order of states. The first state in the definition is assigned 0, the second, 1, etc. Thus,

```
If(Planet="Earth",9.81,10)
```

is equivalent to

```
If(Planet=0,9.81,10)
```

The `Switch()` function allows for its first argument to be a node ID. In that case, the current node's state evaluates to an integer. For example:

```
Switch(Planet,0,9.81,1,5.2,2,10,3,12.4,0,100)
```

will yield 9.81, 5.2, 10, and 12.4 if the current state of the variable *Planet* is *Earth*, *Mars*, *Jupiter* and *Venus* respectively and 100 (the default value) in case variable *Planet* has more than four states.

Translation of the state ID to an integer happens with all comparison operators (=, <, >, <=, >=, and <>). In case of the < and > operators, it is not the alphabetical order of IDs that is compared but their order in the node definition! This allows, for example, for *Water*>="*Medium*" to evaluate to 1 for "*Medium*" and "*High*", assuming that state *High* comes after *Medium*, regardless of the alphabetical ordering of *Medium* and *High*.

Wherever an equation expects a logical expression, equality operator can be abbreviated by omitting the variable name and the operator. For example, *Water*="*Medium*" can be abbreviated to "*Medium*". While it is perfectly legal to rely on the defaults, we do advise to be explicit in the expressions for the sake of modeling clarity. Additionally, being explicit makes models more robust to future model modifications.

At this point, GeNIe does not rename literals in the equations of the child nodes when parent definition changes, so when this happens, the user has to carefully check the literals in the child node definitions.

Equations evaluating to a constant

An equation node is considered to be a constant if it has no references to variables or distributions on the right-hand side of the equation. The following equation, for example, will evaluate to a constant:

```
x=Sin(Pi()/3)
```

There is no runtime penalty for writing an equation this way rather than `x=0.866025`, which is what `Sin(Pi()/3)` evaluates to. GeNIe evaluates constants once, before entering sampling loops, so sampling code will refer to the value `0.866025` and not `Sin(Pi()/3)`.

6.7.3.2 Functions

6.7.3.2.1 Random number generators

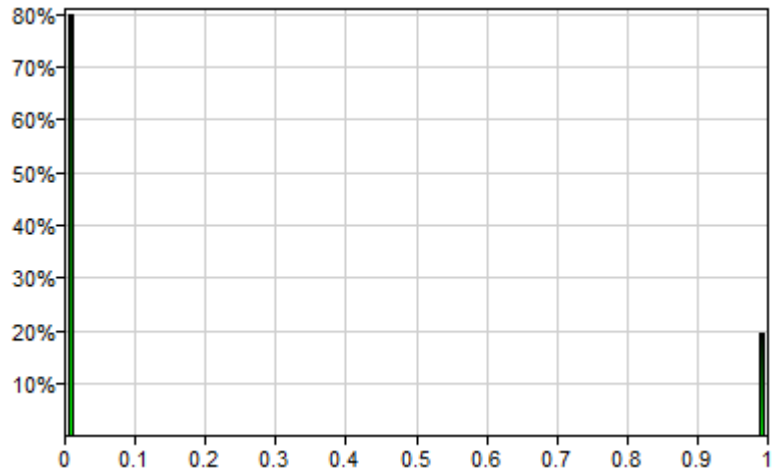
Probability distribution functions **each generate a single sample** from the distributions defined below. In most equations defined in GeNIe, they can be imagined as random noise that distorts the equation. Because the fundamental algorithm for inference in continuous and hybrid models is stochastic simulation, it is possible to visualize what probability distributions these single samples result in for each of the variables in the model.

Bernoulli(p)

Bernoulli is a discrete distribution that generates 0 with probability 1-*p* and 1 with probability *p*.

Example:

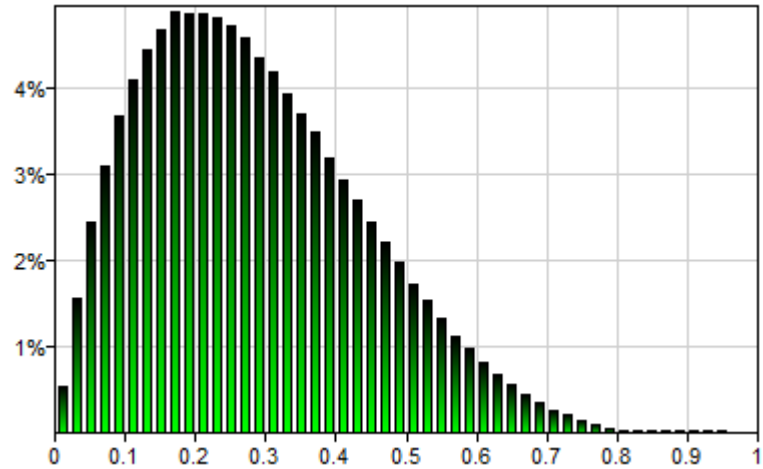
`Bernoulli(0.2)` will generate a single sample (0 or 1) from the following distribution, i.e., 1 with probability 0.2 and 0 with probability 0.8:



Beta(a,b)

The Beta distribution is a family of continuous probability distributions defined on the interval [0, 1] and parametrized by two positive shape parameters, *a* and *b* (typically denoted by α and β), that control the shape of the distribution.

`Beta(2,5)` will generate a single sample from the following distribution:

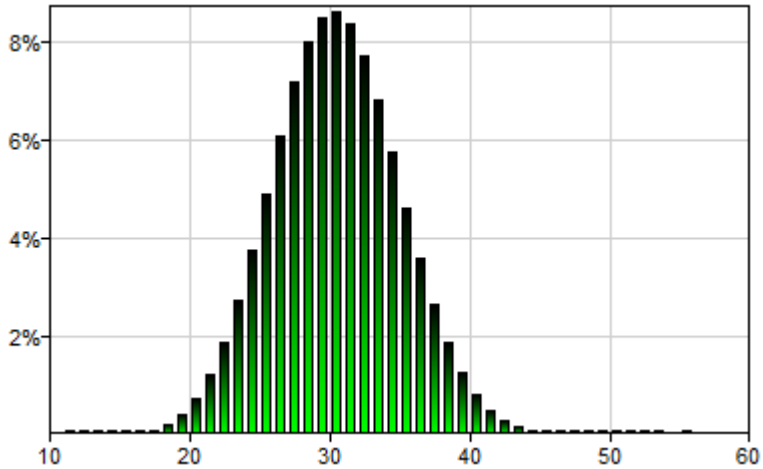


Binomial(n,p)

Binomial is a discrete probability distribution over the number of successes in a sequence of *n* independent trials, each of which yields a success with probability *p*. It will generate a single sample, which will be an integer number between 0 and *n*. A success/failure experiment is also called a Bernoulli trial. Hence, `Binomial(1,p)` is equivalent to `Bernoulli(p)`.

Example:

`Binomial(100,0.3)` will generate a single sample from the following distribution:

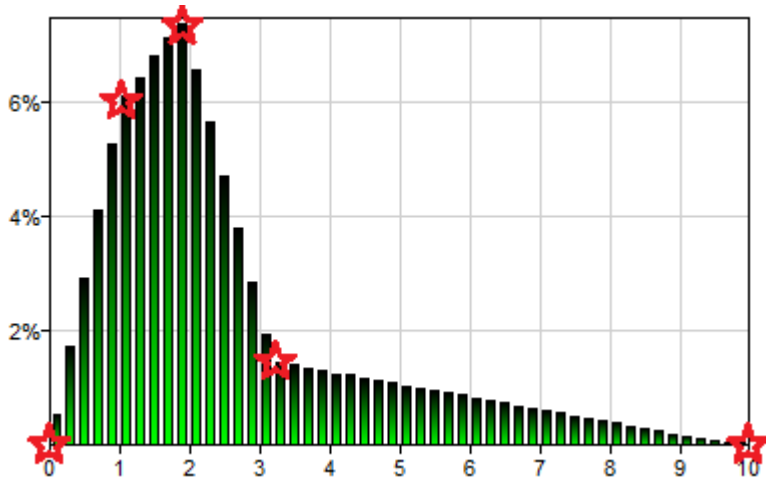


CustomPDF(x1,x2,...y1,y2,...)

The CustomPDF() distribution allows for specifying a non-parametric continuous probability distribution by means of a series of points on its probability density (PDF) function. Pairs (x_i,y_i) are coordinates of such points. The total number of parameters of CustomPDF() function should thus to be even. Please note that x coordinates should be listed in increasing order. The PDF function specified does not need to be normalized, i.e., the area under the curve does not need to add up to 1.0.

Example:

`CustomPDF(0,1.02,1.9,3.2,10,0,4,5,1,0)` generates a single sample from the following distribution:



Stars on the plot mark the points defined by the `CustomPDF()` arguments, i.e., (0, 0), (1.02, 4), (1.9, 5), (3.2, 1), and (10, 0).

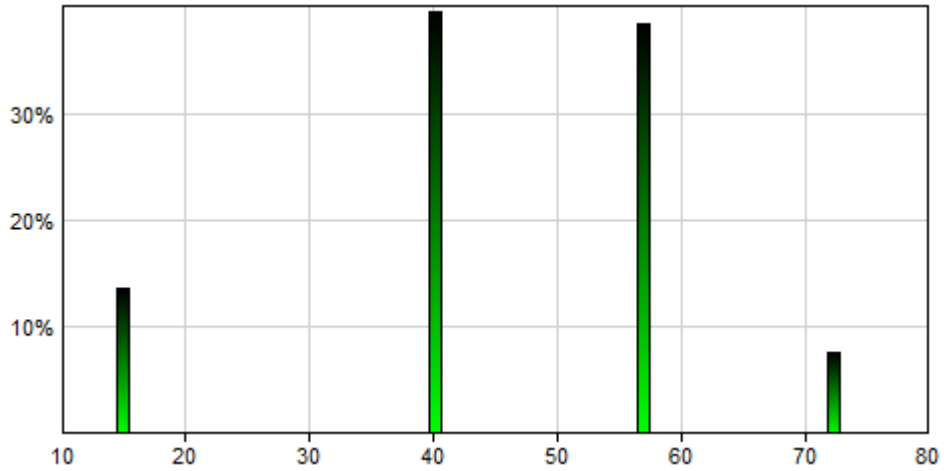
The following probability distributions are pretty much standard

Discrete(x1,x2,...,xn, p1,p2,...,pn)

The `Discrete()` distribution allows for specifying a discrete probability distribution over a collection of numerical values. It is one of the simplest random number generators, essentially replicating a discrete distribution and producing values x1, x2, ..., xn with probabilities p1, p2, ..., xn. This distribution is useful in simulating a discrete node using an equation node. The total number of parameters of `Discrete()` function should be even. Please note that `x` values should be listed in increasing order. Even though the `p` values should in theory add up to 1.0 and we advise that they do, GeNIe perform normalization, i.e., modifies them proportionally to add up to 1.0.

Example:

`Discrete(15,40,57.5,72.5,0.137339,0.397711,0.387697,0.0772532` replicates the definition of the variable `Age` in the HeparII model and generates a single sample from the following distribution:

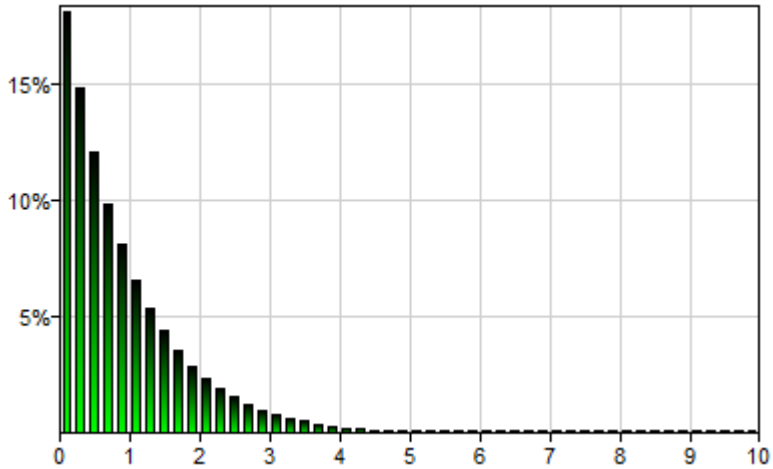


Exponential(lambda)

The exponential distribution is a continuous probability distribution that describes the time between events in a Poisson process, i.e., a process in which events occur continuously and independently at a constant average rate. Its only real-valued, positive parameter *lambda* (typically denoted by λ) determines the shape of the distribution. It is a special case of the Gamma distribution. `Exponential(lambda)` generates a single sample from the domain (0,∞).

Example:

`Exponential(1)` will generate a single sample from the following distribution:

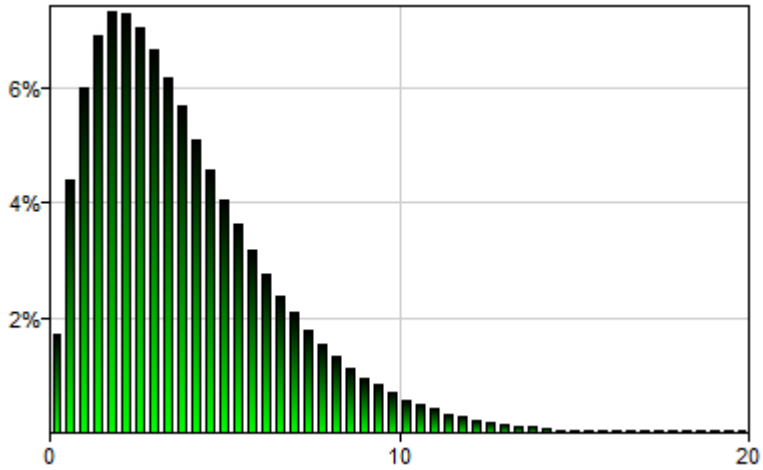


Gamma(shape,scale)

The Gamma distribution is a two-parameter family of continuous probability distributions. There are different parametrizations of the Gamma distribution in common use. GeNIe parametrization follows one of the most popular parametrizations, with *shape* (often denoted by *k*) and *scale* (often denoted by θ) parameters, both positive real numbers.

Example:

`Gamma(2.0,2.0)` will generate a single sample from the following distribution:

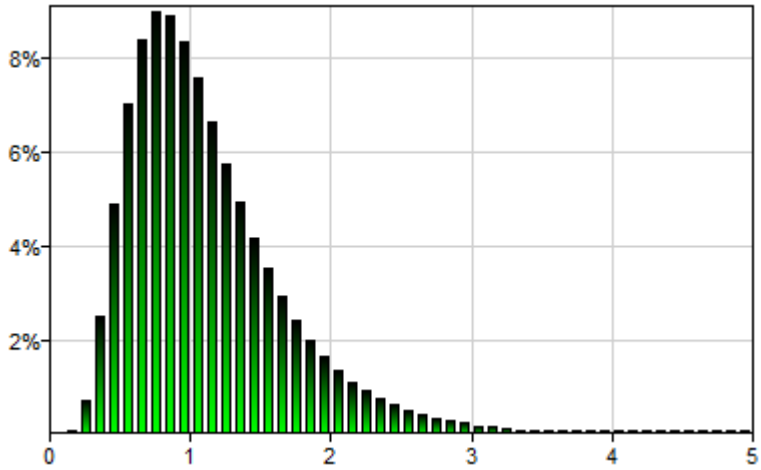


Lognormal(mu,sigma)

The lognormal distribution is a continuous probability distribution of a random variable, whose logarithm is normally distributed. Thus, if a random variable X is lognormally distributed, then a variable Y=Ln(X) has a normal distribution. Conversely, if Y has a normal distribution, then $X=e^Y$ has a lognormal distribution. A random variable which is lognormally distributed takes only positive values.

Example:

`Lognormal(0,0.5)` will generate a single sample from the following distribution:



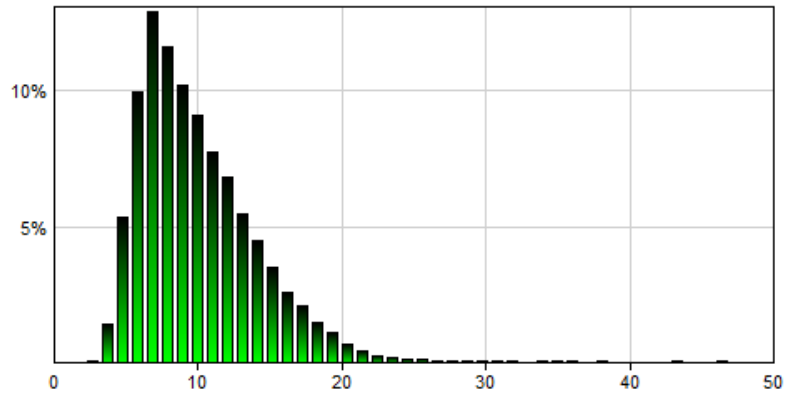
Metalog(lower,upper,k,x1,x2,...,y1,y2,...)

Metalog (also known as the Keelin) distribution is a very flexible distribution, capable of fitting many naturally occurring distributions. It can be specified by probability quantiles, which are values of the variable `xi` and their corresponding cumulative probabilities `yi`. Metalogs are able to represent distributions that are unbounded, semi-bounded, and bounded. `lower` and `upper` are the bounds of the distribution (`-Inf()` and `Inf()` denote lower and upper infinite bound respectively). `k` is a parameter of the metalog distribution, running from `2` to n, where n is the number of probability quantiles specified. Generally the higher the value of `k`, the more flexible the distribution but it is worth looking at the distributions generated for different values of `k` to find a compromise between complexity and goodness of fit. The choice of `k` is best performed interactively, looking at the family of metalog distributions generated from the probability quantiles. Please see *Metalog Builder* section for more information. Please note that we made the functionality of *Metalog Builder* available to the community through a web interface at https://metalog.bayesfusion.com/.

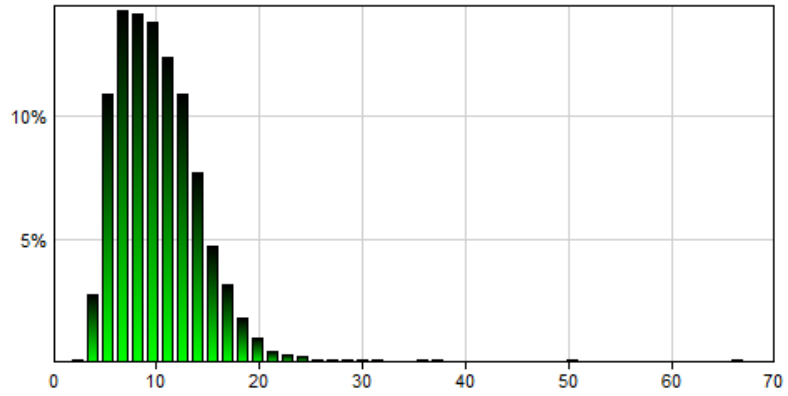
For more information about the metalog distribution, please look at the comprehensive article on the topic on Wikipedia (https://en.wikipedia.org/wiki/Metalog_distribution), the Metalog Distribution web site created by Tom Keelin (http://metalogdistributions.com/) or the Metalog Distributions YouTube channel, educational videos (https://www.youtube.com/channel/UCyHZ5neKhV1mSsedzDBoqyA).

Examples:

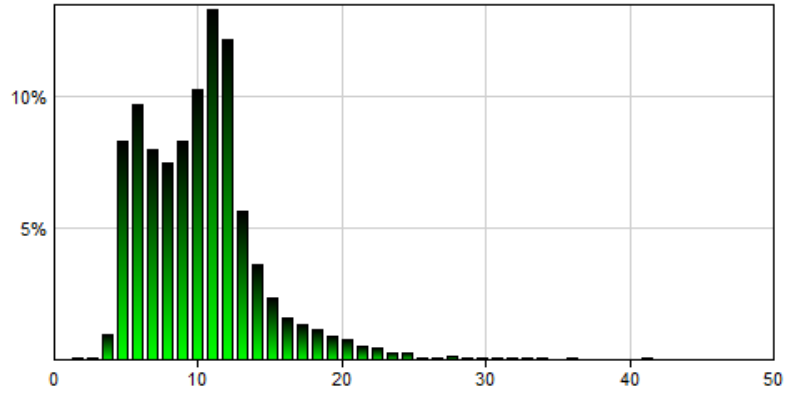
`Metalog(0,Inf(),4,3,4,5,5,7,10,12,15,18,32,0.001,0.01,0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.999)` will generate a single sample from the following distribution



`Metalog(0,Inf(),6,3,4,5,5,7,10,12,15,18,32,0.001,0.01,0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.999)` will generate a single sample from the following distribution



`Metalog(0,Inf(),8,3,4,5,5,7,10,12,15,18,32,0.001,0.01,0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.999)` will generate a single sample from the following distribution



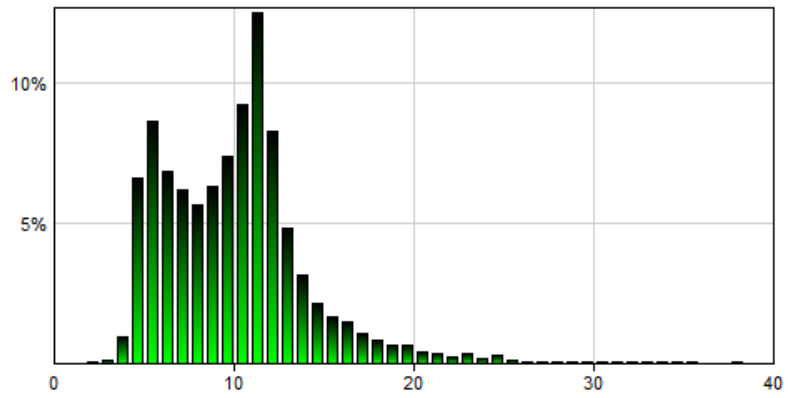
MetalogA(lower,upper,a1,a2,...)

`MetalogA` function uses what one could call internal metalog coefficients (`ai` and, additionally, the `lower` and `upper` bound of the distribution) that, contrary to percentiles of the distribution used as parameters of

`Metalog`, do not have easily interpretable meaning. One might expect that `MetalogA` is more efficient in sample generation, as it skips the whole process of deriving the distribution from which it subsequently generates a sample. However, GeNIe has an efficient caching scheme that makes `Metalog` equally efficient in practice.

Examples:

`MetalogA(0,Inf(),2.30769,0.164148,-0.731388,0.343231,0.883249,-0.170727,1.40341,2.64853)`, which is equivalent to `Metalog(0,Inf(),8,3,4,5,5,7,10,12,15,18,32,0.001,0.01,0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.999)`, will generate a single sample from the following distribution:

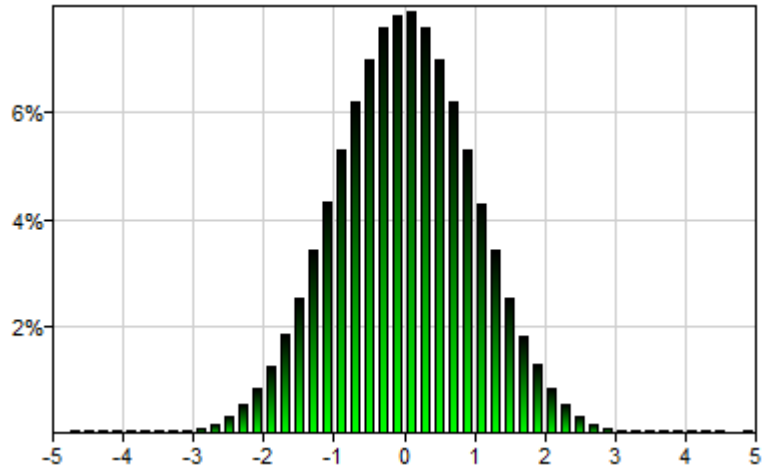


Please note that the number of `ai` parameters of `MetalogA` is equal to the parameter `k` in `Metalog`. Obtaining the parameters `ai` outside of tools like *Metalog Builder* is rather challenging.

Normal(mu,sigma)

Normal (also known as Gaussian) distribution is the most commonly occurring continuous probability distribution. It is symmetric and defined over the real domain. Its two parameters, *mu* (mean, μ) and *sigma* (standard deviation, σ), control the position of its mode and its spread respectively.

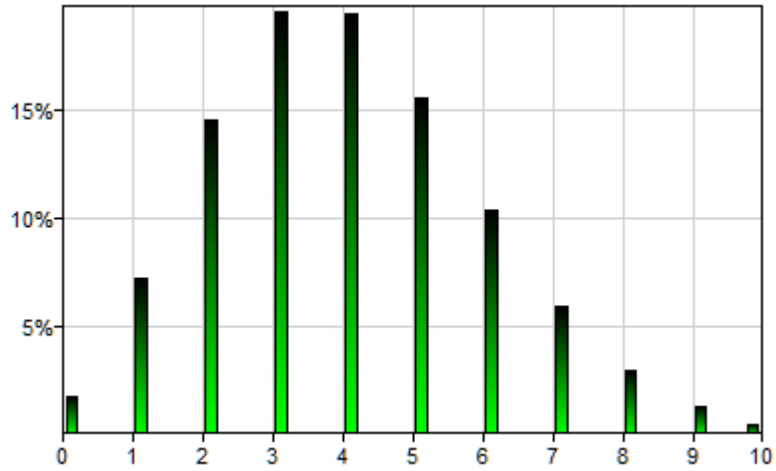
`Normal(0,1)` will generate a single sample from the following distribution:



Poisson(lambda)

Poisson distribution is a discrete probability distribution typically used to express the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant rate and independently of the time since the last event. Its only parameter, lambda, is the expected number of occurrences (which does not need to be integer).

Poisson(4) will generate a single sample from the following distribution:

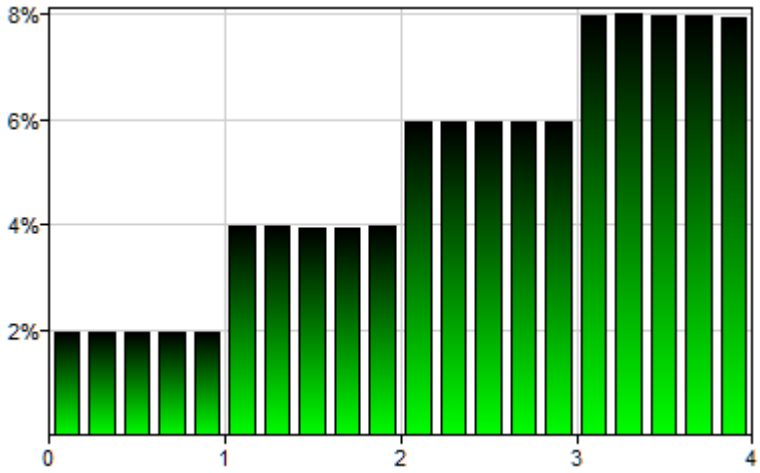


Steps(x1,x2,...y1,y2,...)

The Steps() distribution allows for specifying a non-parametric continuous probability distribution by means of a series of steps on its probability density (PDF) function. It is similar to the CustomPDF() function, although it does not specify the inflection points but rather intervals and the height of a step-wise probability distribution in each of the intervals. Because the number of interval borders is always one more than the number of intervals between them, the total number of parameters of Steps() function should be odd. Please note that x coordinates should be listed in increasing order. The PDF function specified does not need to be normalized, i.e., the area under the curve does not need to add up to 1.0.

Example:

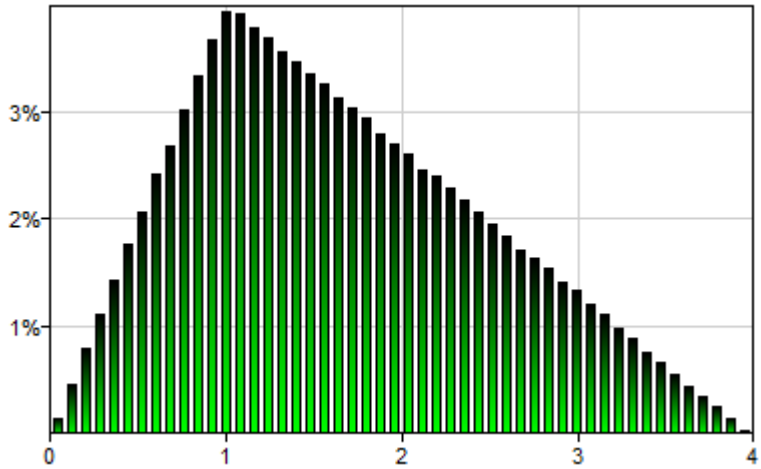
Steps(0,1,2,3,4,1,2,3,4) generates a single sample from the following distribution:



Triangular(min,mod,max)

Triangular distribution is a continuous probability distribution with lower limit *min*, upper limit *max* and mode *mod*, where *min* ≤ *mod* ≤ *max*.

`Triangular(0,1,3)` will generate a single sample from the following distribution:

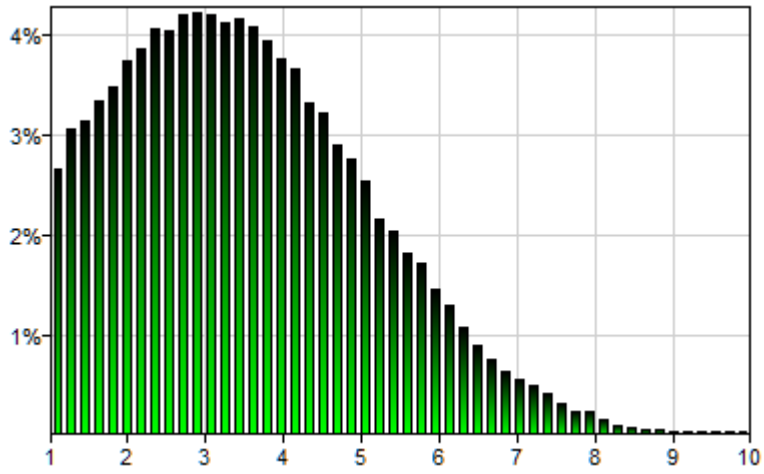


TruncNormal(mu,sigma,lower,[upper])

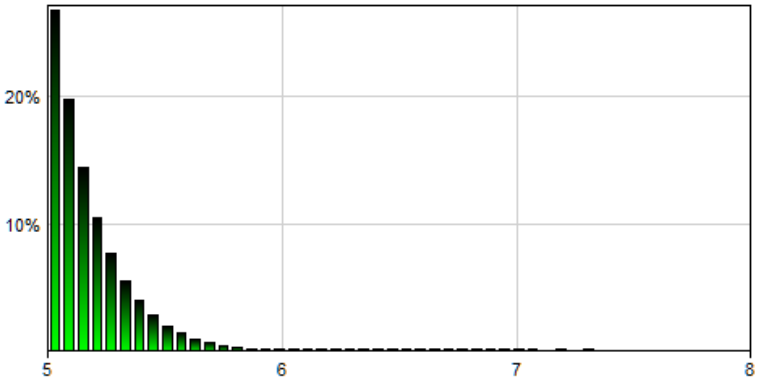
Truncated Normal distribution is essentially a Normal distribution that is truncated at the values *lower* and *upper*. This distribution is especially useful in situation when we want to limit physically impossible values in the model.

Example:

`TruncNormal(3,2,1)` will generate a single sample from the following distribution:



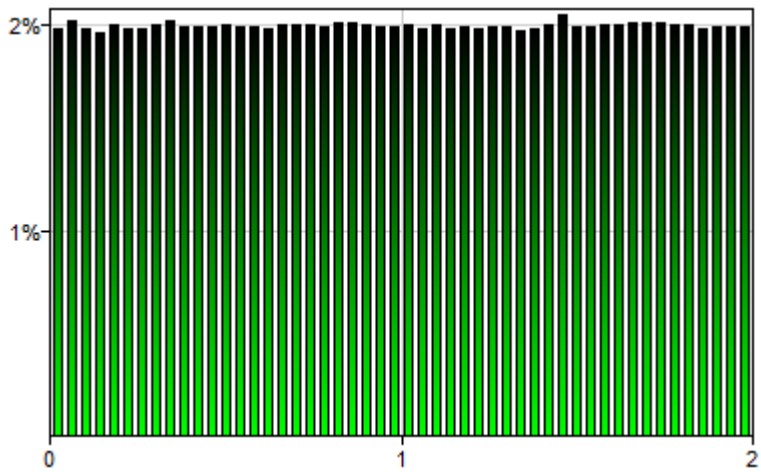
`TruncNormal(0,1,5,10)` will generate a single sample from the following distribution:



Uniform(a,b)

The continuous uniform distribution, also known as the rectangular distribution, is a family of probability distributions under which any two intervals of the same length are equally probable. It is defined two parameters, *a* and *b*, which are the minimum and the maximum values of the random variable.

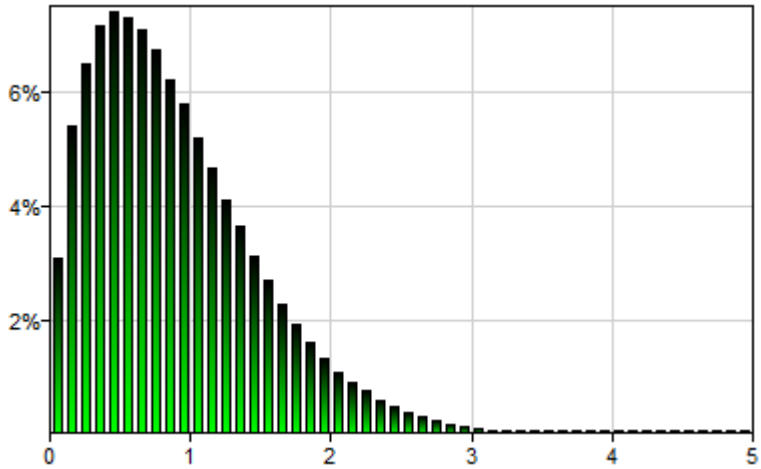
`Uniform(0,2)` will generate a single sample from the following distribution:



Weibull(lambda,k)

Weibull distribution is a continuous probability distribution named after a Swedish mathematician Waloddi Weibull, used in modeling such phenomena as particle size. Weibull is characterized by two positive real parameters: the scale parameter *lambda* (λ) and the shape parameter *k*.

`Weibull(1,1.5)` will generate a single sample from the following distribution:



6.7.3.2.2 Statistical functions

Erf(x)

Returns error function (also called the Gauss error function, inverse of the Normal PDF), e.g., `Erf(1.0) =0.842701`

NormalCDF(x, mu, sigma)

Returns the value of the Normal cumulative density function, e.g., `NormalCDF(1.0,0,1)=0.841345`

NormalCDF(x, mu, sigma)

Returns the value of the Normal probability density function, e.g., `NormalPDF(1.0,0,1)=0.241971`

6.7.3.2.3 Arithmetic functions

Abs(x)

Returns the absolute value of a number, e.g., `Abs(5.3)=Abs(-5.3)=5.3`

Exp(x)

Returns *e* (Euler's number) raised to the power of *x*, e.g., `Exp(2.2)=`$e^{2.2}$`=9.02501`

Gammaln(x)

Returns the natural logarithm of the Gamma function ($\Gamma(x)$), e.g., `Gammaln(2.2)=0.0969475`

GCD(n,k)

Returns the greatest common integer divisor of its two integer arguments *n* and *k*, e.g., `GCD(15,25)=5`

When the arguments are not integers, their fractional part is ignored.

Inf()

Returns positive infinity. For negative infinity, please use `-Inf()`.

LCM(n,k)

Returns the least common integer multiple of its two integer arguments *n* and *k*, e.g., `LCM(15,25)=75`

When the arguments are not integers, their fractional part is ignored.

Ln(x)

Returns the natural logarithm of *x*, which has to be non-negative, e.g., `Ln(10)=2.30259`

Log(x,b)

Returns the base *b* logarithm of *x*, which has to be non-negative, e.g., `Log(10,2)=3.32193`

Log10(x)

Returns decimal logarithm of *x*, which has to be non-negative, e.g., `Log10(100)=2`

Mod(x,y)

Returns the remainder of dividing `x` by `y`. This works for both integer and real arguments, e.g., `Mod(5,2)=1`, `Mod(9.2,3.7)=1.8`.

Pi()

Returns the π constant, e.g., `Pi()=3.14159`.

Pow10(x)

Returns 10 raised to the power of *x*, e.g., `Pow10(2)=10`^{2}=100

Round(x)

Returns the integer that is nearest to x, e.g., `Round(2.2)=2, Round(3.5)=4`

Sign(x)

Returns 1 if *x*>0, 0 when *x*=0, and -1 if *x*<0, e.g., `Sign(2.2)=1, Sign(0)=0, Sign(-3.5)=-1`

Sqrt(x)

Returns the square root of *x*, which has to be non-negative, e.g., `Sqrt(2)=1.41421`

SqrtPi(x)

Returns square root of π multiplied by *x*, which has to be non-negative, e.g. `SqrtPi(2)=Sqrt(Pi()*2) =2.50663`

This function is provided for the sake of compatibility with Microsoft Excel.

Sum(x1,x2,...)

Returns the sum of its arguments, e.g., `Sum(2.2, 3.5, 1.3)=7.0`

`Sum()` requires at least two arguments.

SumSq(x1,x2,...)

Returns the sum of squares of its arguments, e.g., `SumSq(2.2, 3.5, 1.3)=18.78`

`SumSq()` requires at least two arguments.

Trim(x,lo,hi)

Trims the value of the argument *x* to a value in the interval <*lo*, *hi*>. If *x*≤*lo*, the function returns *lo*, if *x*≥*hi*, the function returns *hi*, if *lo*<*x*<*hi*, the function returns *x*. The function is a shortcut to two nested conditional functions `If()` and is equivalent to `If(x<lo,lo,If(x>hi,hi,x))`. For example, `Trim(-0.5,0,1)=0`, `Trim(0.5,0,1)=0.5`, `Trim(1.5,0,1)=1`

Truncate(x)

Returns the integer part of *x*, e.g., `Truncate(2.2)=2`

6.7.3.2.4 Combinatoric functions

Combin(n,k)

Returns the number of combinations of distinct *k* elements from among *n* elements, e.g., `Combin(10,2)=45`

Fact(n)

Returns the factorial of *n*, e.g., `Fact(5)=5!=120`

`Fact()` of a negative number returns 0.

FactDouble(n)

Returns the product of all even (when *n* is even) or all odd (when *n* is odd) numbers between 1 and *n*, e.g., `FactDouble(5)=15`, `FactDouble(6)=48`

`FactDouble()` of a negative number returns 0.

Multinomial(n1,n2,...)

Factorial of sum of arguments, divided by the factorials of all arguments, e.g., `Multinomial(2,5,3)=Fact(2+5+3)/(Fact(2)*Fact(5)*Fact(3)=10!/(2!*5!*3!)=2520`

All arguments of `Multinomial()` have to be positive.

6.7.3.2.5 Trigonometric functions

Acos(x)

Returns arccosine (*arcus cosinus*) of *x*, e.g., `Acos(-1)=3.14159`

Asin(x)

Returns arcsine (*arcus sinus*) of *x*, e.g., `Asin(1)=1.5708`

Atan(x)

Returns arctangent (*arcus tangens*) of *x*, e.g., `Atan(1)=0.785398`

Atan2(y,x)

Returns arctangent (*arcus tangens*) from *x* and *y* coordinates, e.g., `Atan2(1,1)=0.785398`

Cos(x)

Returns cosine (*cosinus*) of *x*, e.g., `Cos(1)=0.540302`

Pi()

Returns constant π, e.g., `Pi()=3.14159`

Sin(x)

Returns sine (*sinus*) of *x*, e.g., `Sin(1)=0.841471`

Tan(x)

Returns tangent (*tangens*) of *x*, e.g., `Tan(1)=1.55741`

6.7.3.2.6 Hyperbolic functions

Cosh(x)

Returns the hyperbolic cosine of *x*, e.g., `Cosh(1)=1.54308`

Sinh(x)

Returns the hyperbolic sine of *x*, e.g., `Sinh(1)=1.1752`

Tanh(x)

Returns the hyperbolic tangent of *x*, e.g., `Tanh(1)=0.761594`

6.7.3.2.7 Conditional functions

Choose(index,v0,v1,...,vn)

Returns v_i if *index* is equal to *i*. When *index* evaluates to a value smaller than 0 or larger than *n*-1, the function returns 0. Examples:

```
Choose(0,1,2,3,4,5)=1
Choose(4,1,2,3,4,5)=5
Choose(7,1,2,3,4,5)=0
```

If(cond,tval,fval)

If *cond* evaluates to non-zero return *tval*, *fval* otherwise, e.g., `If(1=2,3,4)=4`, `If(1,5,10)=5`

Switch(x,a1,b1,a2,b2,...,[def])

If *x*=*a1*, return *b1*, if *x*=*a2*, return *b2*, when *x* is not equal to any of *a*s, return *def* (default value), which is an optional argument. When *x* is not equal to any of *a*s and no *def* is defined, return 0. Examples:

```
Switch(3,1,111,2,222,3,333,4,444,5,555,999)=333
Switch(8,1,111,2,222,3,333,4,444,5,555,999)=999
Switch(8,1,111,2,222,3,333,4,444,5,555)=0
```

6.7.3.2.8 Logical/Conditional functions

And(b1,b2,...)

Returns the logical conjunction of the arguments, which are all interpreted as Boolean expressions. If any of the expressions evaluates to a zero, `And()` returns 0. For example, `And(1=1,2,3)=1`, `And(1=2,2,3)=0`

Max(x1,x2,...)

Returns the largest of the arguments x_i. Examples:

```
Max(1,2,3,4,5)=5
Max(-3,2,0,2,1)=2
```

Min(x1,x2,...)

Returns the smallest of the arguments x_i. Examples:

```
Min(1,2,3,4,5)=1
Min(-3,2,0,2,1)=-3
```

Or(b1,b2,...)

Returns the logical disjunction of the arguments, which are all interpreted as Boolean expressions. If any of the expressions evaluates to a non-zero value, `Or()` returns 1. For example, `Or(1=1,0,0)=1`, `Or(1=0,0,0)=0`

Xor(b1,b2,...)

Returns logical exclusive OR of all arguments, which are all interpreted as Boolean expressions. `Xor()` returns a 1 if the number of logical expressions that evaluate to non-zero is odd and a 0 otherwise.

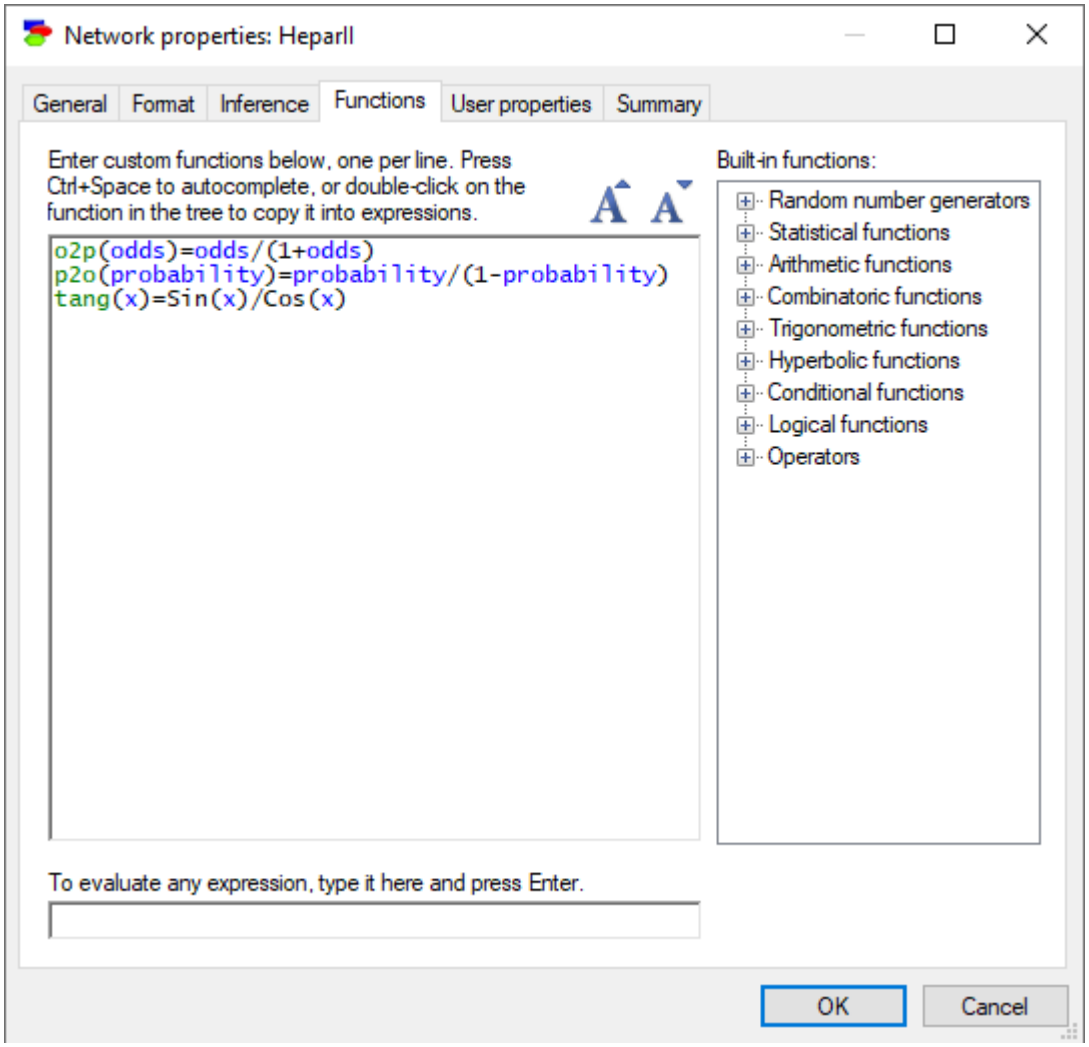
Examples:

```
Xor(0,1,2,-3,0)=1
Xor(1,1,0,2,2)=0
Xor(-5,1,1,-2,2)=1
```

6.7.3.2.9 Custom functions

User can define custom functions, which can be subsequently used in a variable definition. Custom functions are defined in the *Functions* tab of the *Network properties* dialog. The following sheet contains three definitions of functions: `o2p()`, `p2o()`, and `tang()`, used to convert odds to probabilities, probabilities to odds, and a

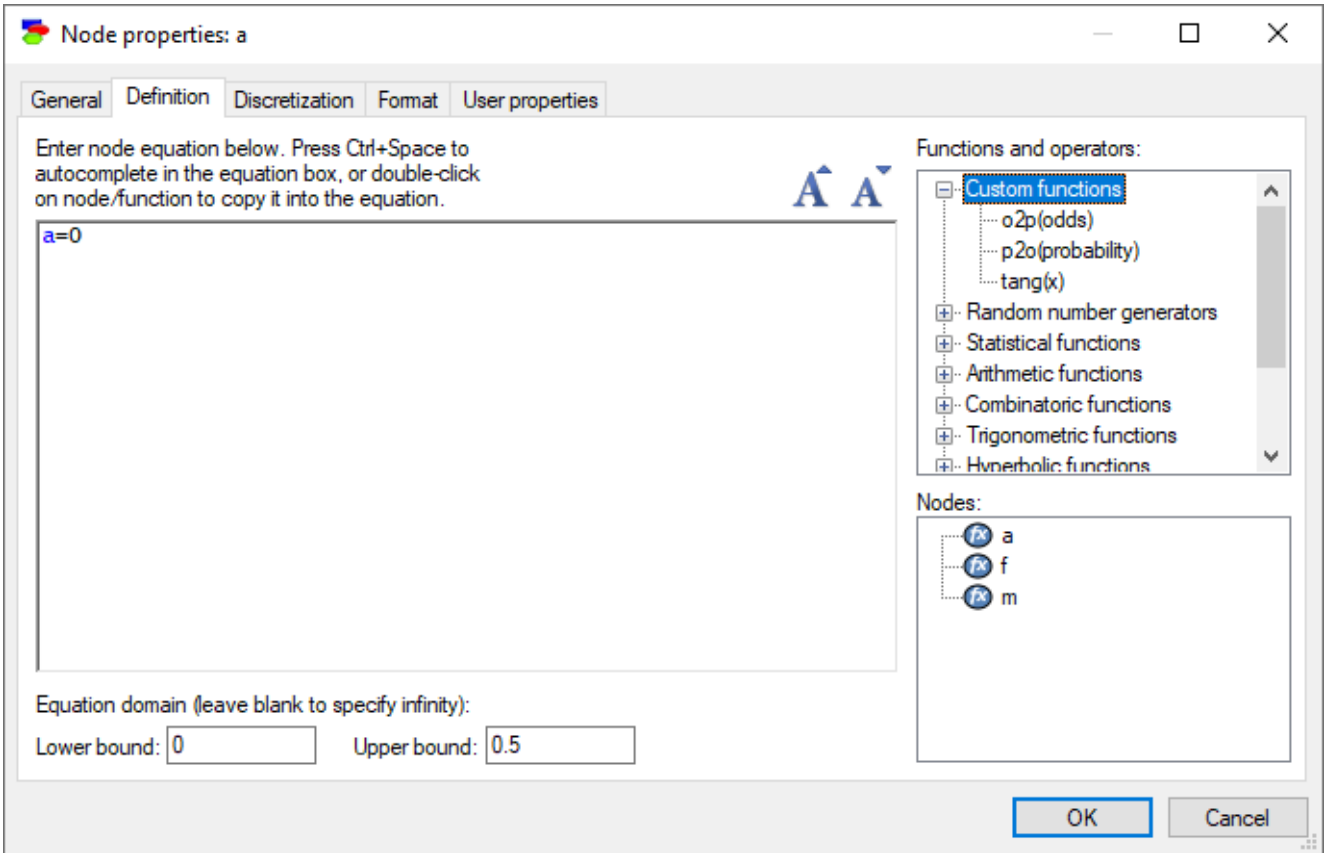
redefinition of the `tan()` tangent function. GeNIe functions may not be recursive but they may refer to functions defined earlier (in the same *Functions* tab).



It is possible to test the newly defined functions by evaluating calls to them with concrete values of parameters at the bottom of the tab. Please note that arguments of the evaluated function have to evaluate to constants (even if these constants are randomly generated by reference to a random number generator, e.g., `MyCustomFunc(Normal(0,1))`).

User-defined functions (also called *Custom functions*) are defined per network and stored along with the network. GeNIe copies user-defined functions between networks when nodes using them are copied from a source network and pasted in the destination network. In this case, conflicts may occur, for example a function with the same definition may exist already in the destination network. GeNIe performs a simple check of the name of the function, its number of parameters, and determinism status (i.e., whether the function makes calls to random number generators) and will use the definition in the destination network if these three agree. Otherwise, it will copy the definition from the source to the destination network. We recommend that meaningful names are used and different definitions of the same function in different models are avoided.

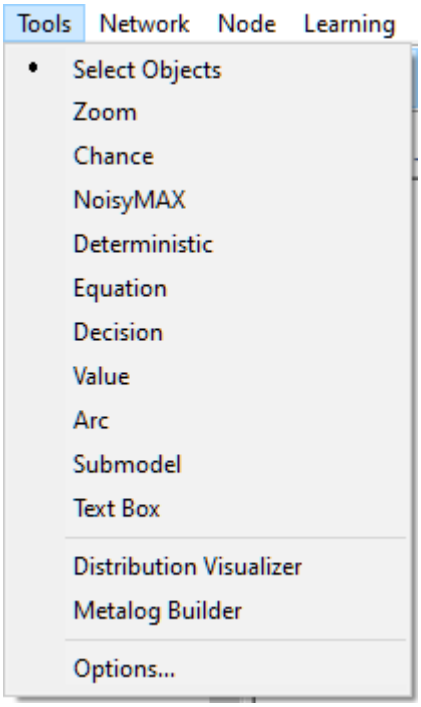
Custom functions defined at the network level appear in the *Node properties*' *Definition* tab at the top of the list of functions in the upper-right pane. They can be used in node definitions alongside of the built-in functions.



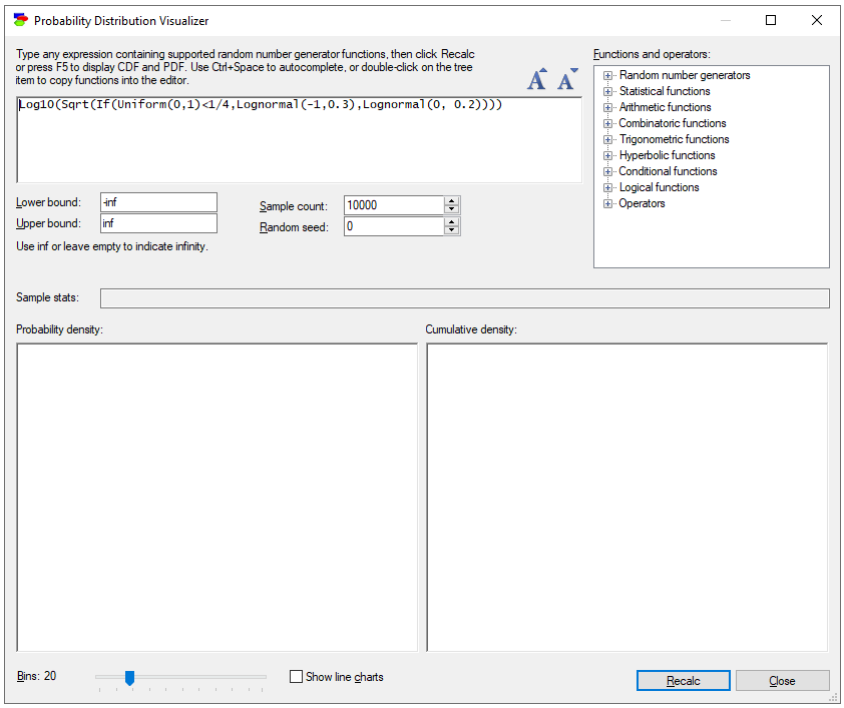
While this mechanism allows for defining complex expressions that will quite likely satisfy most, if not all, modeling needs, we encourage users to let us know when the set of GeNIe's built-in function should be extended.

6.7.3.3 Distribution visualizer

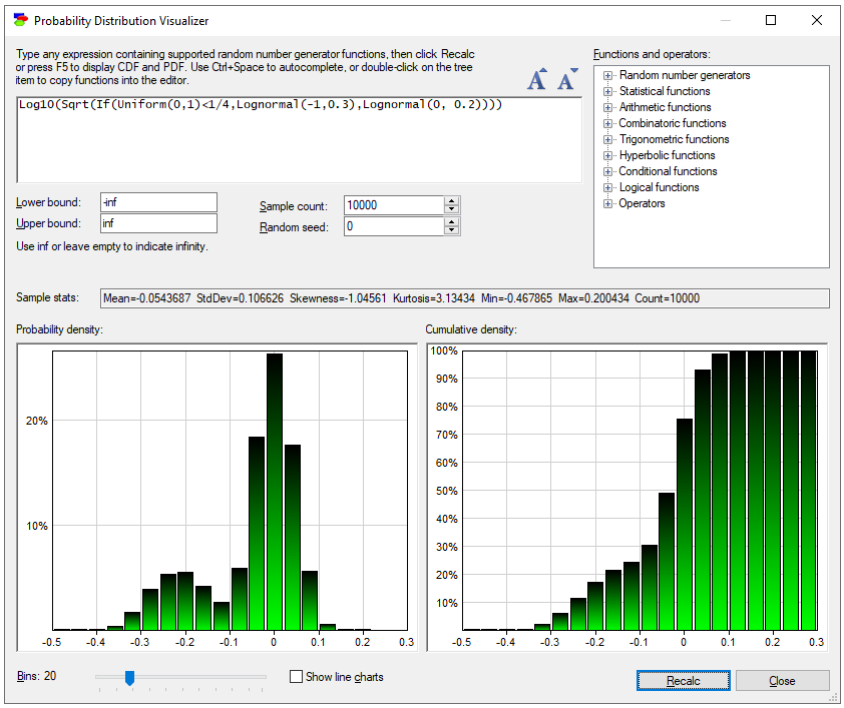
Choosing the right probability distribution over continuous data is a skill that requires some statistical insight. When the distribution is transformed by an equation expression, the task is daunting even for an experienced decision analyst. GeNIe offers an interactive tool for visualizing expressions with probability distributions through Monte Carlo simulation. This helps with selecting an expression that is most appropriate for a task at hand. To invoke the *Distribution visualizer*, please select it from the *Tools* menu.



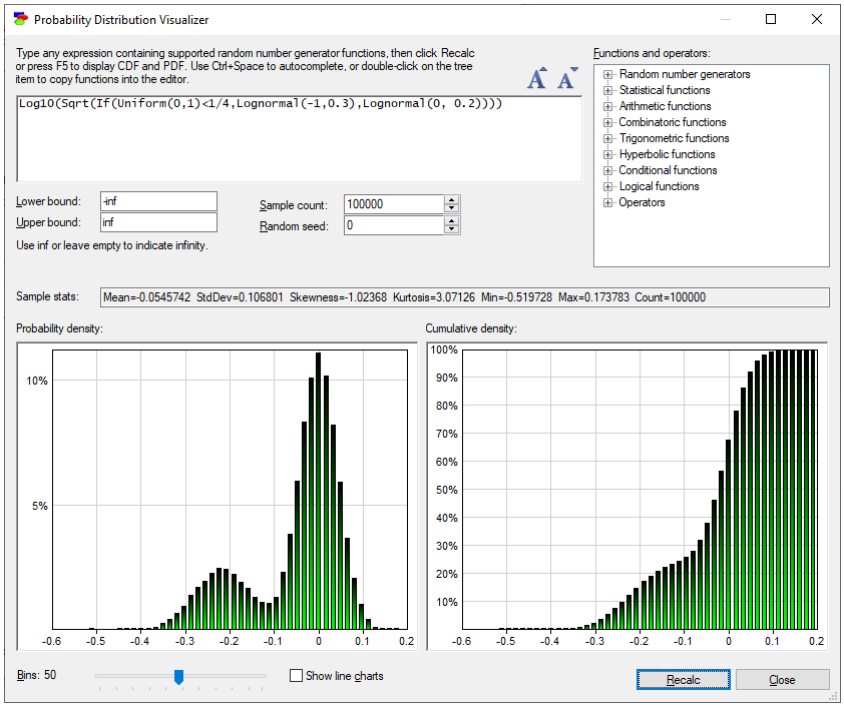
The initial dialog contains a default distribution, `Log10(Sqrt(If(Uniform(0,1)<1/4,Lognormal(-1,0.3),Lognormal(0, 0.2))))` plotted in the interval running from minus infinity to infinity based on 10,000 samples. There is nothing magical in this example distribution - it is one of infinitely many hard to visualize distributions that is a decimal logarithm of a mixture of two Lognormal distributions.



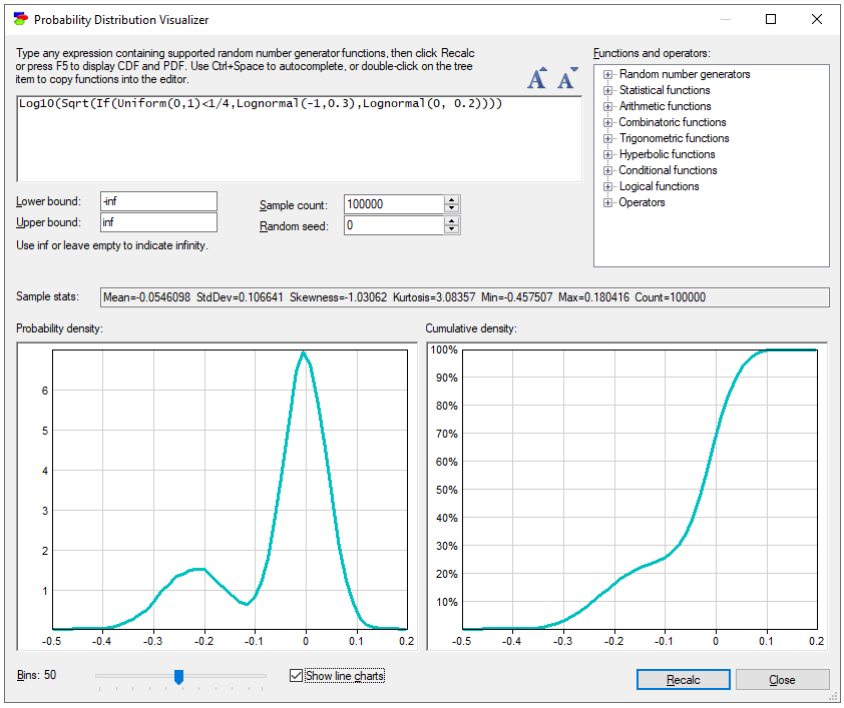
Each of the parameters of this dialog can be modified. Let us, however, start with the default values and press the *Recalc* button. This shows the probability density function (PDF) and cumulative density function (CDF) of the distribution.



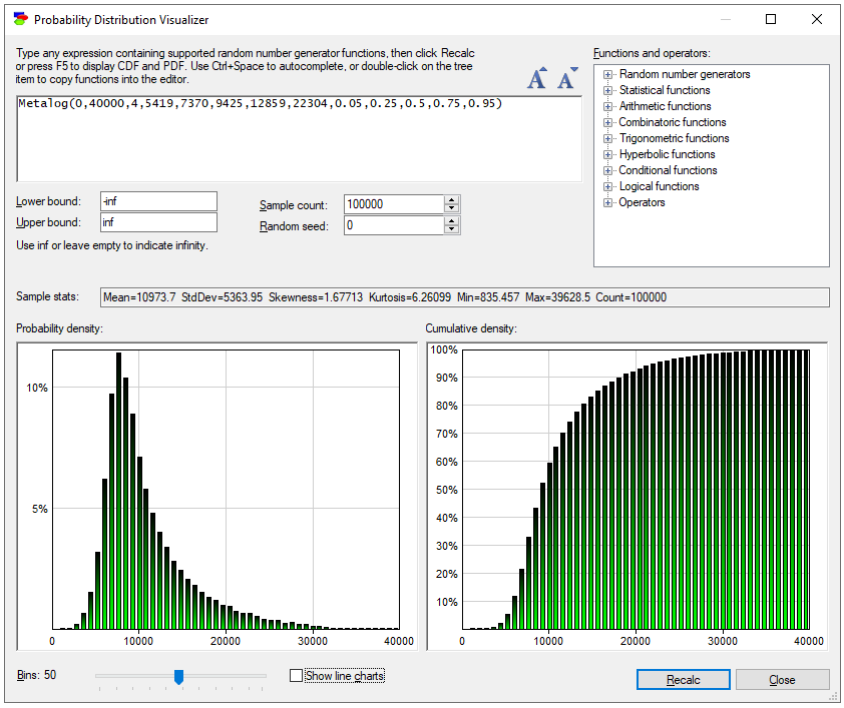
Two easy ways of obtaining more insight into the distribution is by (1) increasing the number of samples, and (2) increasing the number of bins of the histogram that shows the samples. In the following dialog, we changed the number of samples to 100,000 and the number of bins to 50.



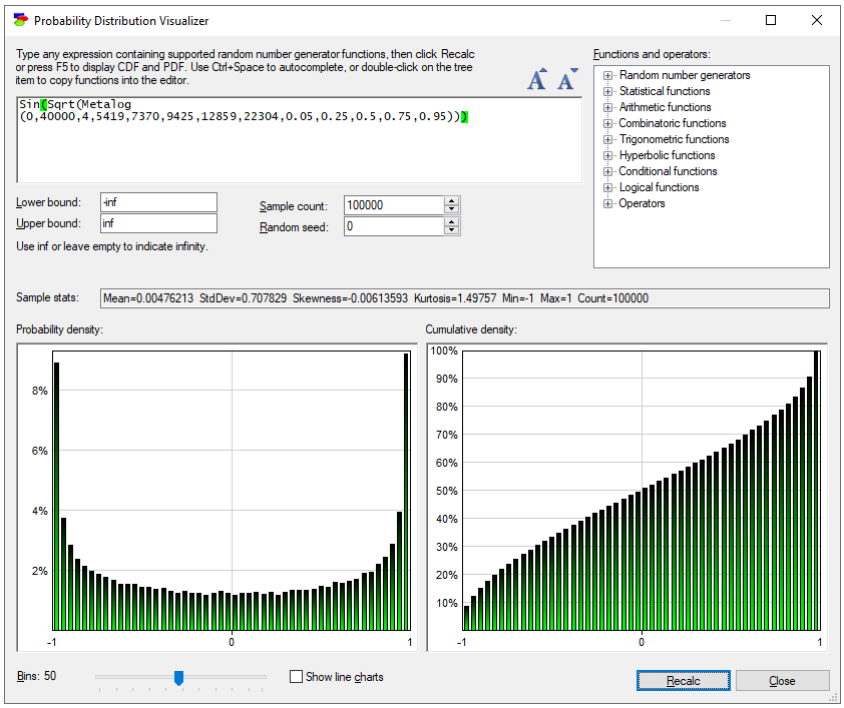
By checking the *Show line charts* check box, it is possible to view the plot as the PDF and the CDF charts fitted to the histograms.



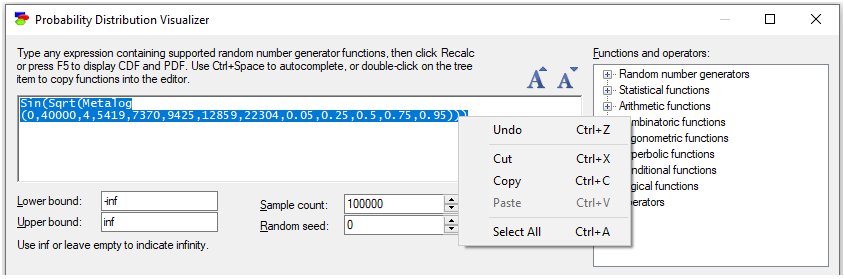
The expression dialog serves for defining the expression to be visualized. Entering a definition of a metalog distribution and pressing the *Recalc* button leads to the following plots.



We can examine what the sinus of the square root of this distribution will look like. In fact any function and any operator from those functions and operators that are known to GeNIe and displayed in the upper-right window pane can be used in the expression. If the *Distribution visualizer* is invoked in the context of a model, custom functions defined within the model will also become available.



Once we have converged on a satisfactory expression, we can select and copy the expression so that we can later paste it into the definition of a variable in our model.

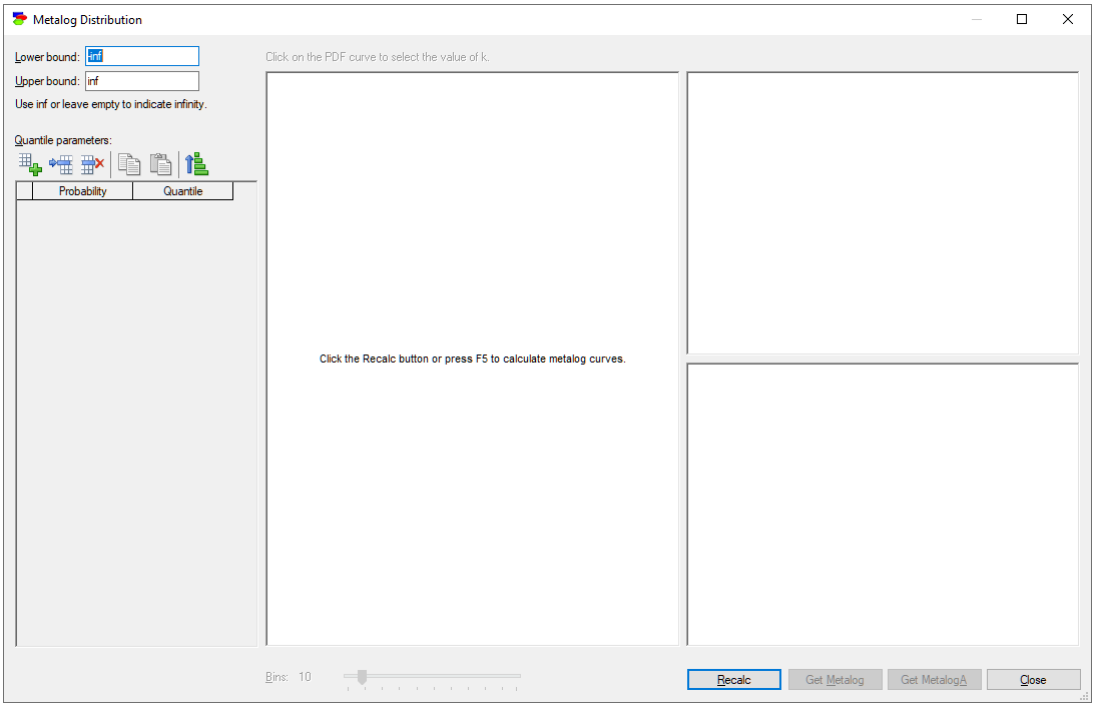


It should be added that any histogram or distribution plot in the *Distribution visualizer* can be copied and later pasted as a picture into any other Windows application by right-clicking on the histogram/plot and choosing *Copy*.

Please note that we made the functionality of *Distribution visualizer* available to the community through a web interface at https://prob.bayesfusion.com/.

6.7.3.4 Metalog builder

Metalog builder dialog parallels the *Learning Metalog distribution* dialog, in which it was possible to fit a metalog distribution to a collection of data points. *Metalog builder* also allows for selecting a metalog distribution but the starting point is a collection of probability quantiles. The initial dialog looks as follows:



It is a good idea to start with lower and upper bounds, which are any real values with the only constraint that the lower bound is lower than than upper bound. It is possible to indicate infinite bounds by entering `-inf` and `inf` respectively or leaving the bound field empty.

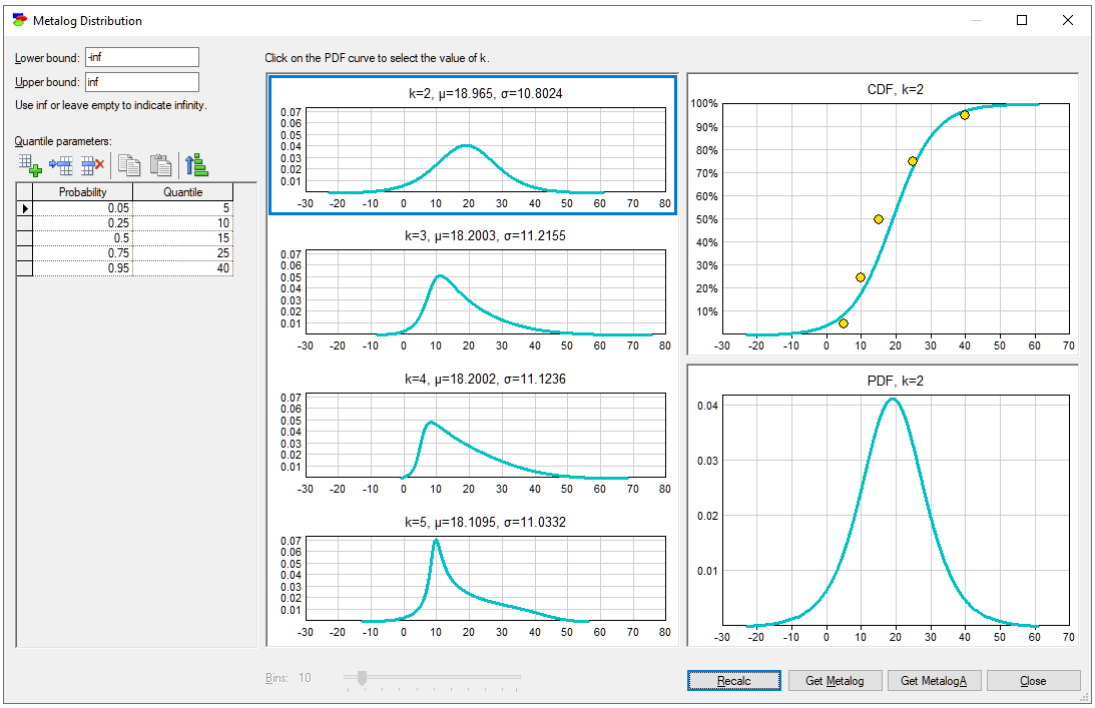
Input parameters are probabilities and quantiles captured in the *Input parameters* table. Editing the table comes with a similar set of tools as editing probability tables (*Add*, *Insert*, and *Delete* buttons). Probabilities do not need to be sorted a-priori, as GeNIe will validate and sort them for you. GeNIe is also tolerant to empty rows. Quantiles in the table are constrained to be growing with higher probability values. Both columns can be copied and pasted.

Let us leave the bounds at infinity and enter the following quantiles:

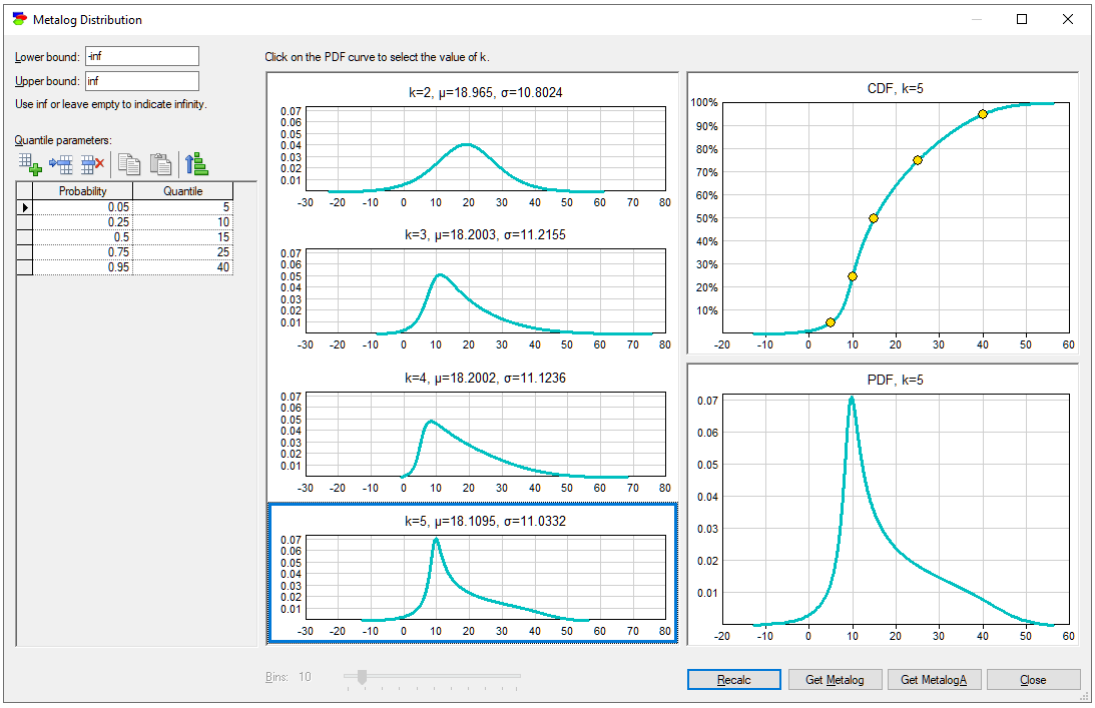
Probability	Quantile
0.05	5
0.25	10
0.5	15
0.75	25
0.95	40

Pressing *Recalc* button shows a set of plots of the metalog distributions for `k=2` through `n` that fit the data, where `n` is the number of input parameters (quantiles of the distribution). Only the feasible metalog distributions are shown, so the number of distributions may be smaller than `n-1`. As the number of quantiles specified in the example is 5 and all metalog distributions are feasible, the dialog displays distributions for `k=2`, 3, 4, and 5.

Generally, the higher the number of quantiles specified, the more complex and flexible the distribution. However, high number of quantiles carries the danger of overfitting the data, so we advise prudent caution, looking at the distributions generated for different values of `k`, and selecting a not-too-high value of `k` that produces the desired distribution.

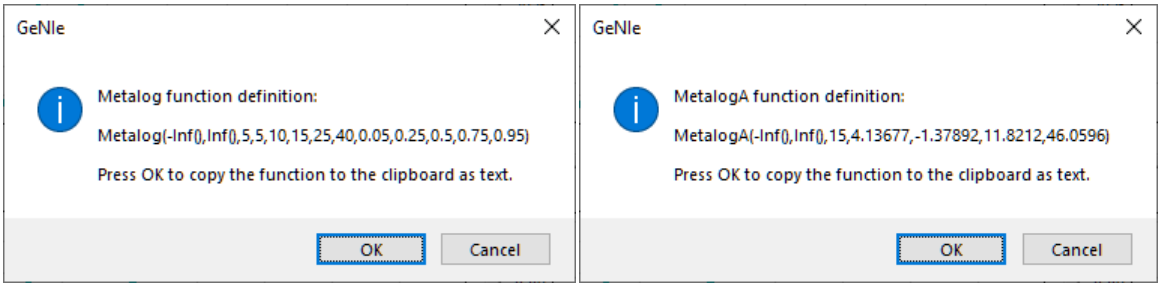


Clicking on any of the four plots displays the CDF and PDF of the distribution on the right-hand side. The CDF plot shows the input quantiles. Obviously, the closer the points are to the CDF line, the better the fit.



Any plot in the *Metalog builder* can be copied and later pasted as a picture into any other Windows application by right-clicking on the plot and choosing *Copy*.

The buttons *Get Metalog* and *Get MetalogA* copy the formal description of the chosen metalog function for pasting elsewhere (usually in the definition of some node in your model).



As explained in the section on *Random number generators*, the difference between the `Metalog` and `MetalogA` functions is that the latter uses internal metalog coefficients that do not have easily interpretable meaning. `Metalog`, on the other hand, uses parameters that are percentiles of the distribution. One might expect that `MetalogA` is more efficient in sample generation, as it skips the whole process of deriving the distribution from which it subsequently generates a sample. However, GeNIe has an efficient caching scheme that makes `Metalog` equally efficient in practice.

Please note that we made the functionality of *Metalog Builder* available to the community through a web interface at https://metalog.bayesfusion.com/.

For more information about the metalog distribution, please look at the comprehensive article on the topic on Wikipedia (https://en.wikipedia.org/wiki/Metalog_distribution), the Metalog Distribution web site created by Tom Keelin (http://metalogdistributions.com/) or the Metalog Distributions YouTube channel, educational

videos (https://www.youtube.com/channel/UCyHZ5neKhV1mSsedzDBoqyA). These sources provide access to articles by Tom Keelin and colleagues on the topic of the metalog distribution.

6.7.3.5 Operators

GeNIe allows the following operators in the definition of continuous nodes:

Arithmetic operators

+ addition, e.g., if x=3 and y=2, x+y produces 5

- subtraction and unary minus, e.g., if x=3 and y=2, x-y produces 1 and -x produces -3

^ exponentiation (a^b means a^b), e.g., if x=3 and y=2, x^y produces 9

***** multiplication, e.g., if x=3 and y=2, x*y produces 6

/ division, e.g., if x=3 and y=2, x/y produces 1.5

Comparison operators

> greater than, e.g., if x=3 and y=2, x>y produces 1

< smaller than, e.g., if x=3 and y=2, x<y produces 0

>= greater or equal than, e.g., if x=3 and y=2, x>=y produces 1

<= smaller or equal than, e.g., if x=3 and y=2, x<=y produces 0

<> not equal, e.g., if x=3 and y=2, x<>y produces 1

= equal, e.g., if x=3 and y=2, x=y produces 0

Conditional selection operator

? : ternary conditional operator like in C, C++ or Java programming languages.

This operator is essentially a shortcut to the If() function. For example, a=b?5:3 is equivalent to If(a=b, 5, 3).

Order of calculation, operator precedence, and parentheses

Expressions in GeNIe are evaluated from left to right, according to the precedence order specified below (1 denotes the highest precedence).

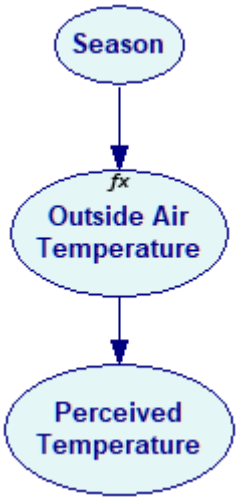
Precedence order	Operator
1	- (unary minus)
2	^ (exponentiation)
3	* and / (multiplicative operators)
4	+ and - (additive operators)
5	>, <, >=, <=, = (comparison operators)
6	?: (conditional selection)

To change the order of calculation, enclose in parentheses those parts of the formula that should be calculated first. This can be done recursively, i.e., parentheses can be nested indefinitely. For example, if x=3 and y=2, 2*y+x/3-y+1 produces 4, 2*(y+x)/(3-y)+1 produces 11, and 2*((y+x)/(3-y)+1) produces 12.

6.7.4 Hybrid models

Models consisting of both discrete and continuous (equation-based) nodes are called hybrid. Construction of discrete Bayesian network models is covered in the *Hello GeNIe* section. Construction of continuous Bayesian network models is covered in the *Constructing equation-based models* section. In this section, we will show the construction of a simple hybrid Bayesian network. In hybrid networks, the two interesting cases are when (1) a continuous node has a discrete parent, and (2) when a discrete node has a continuous parent.

Consider the following three variables involved in causal relationships with each other:



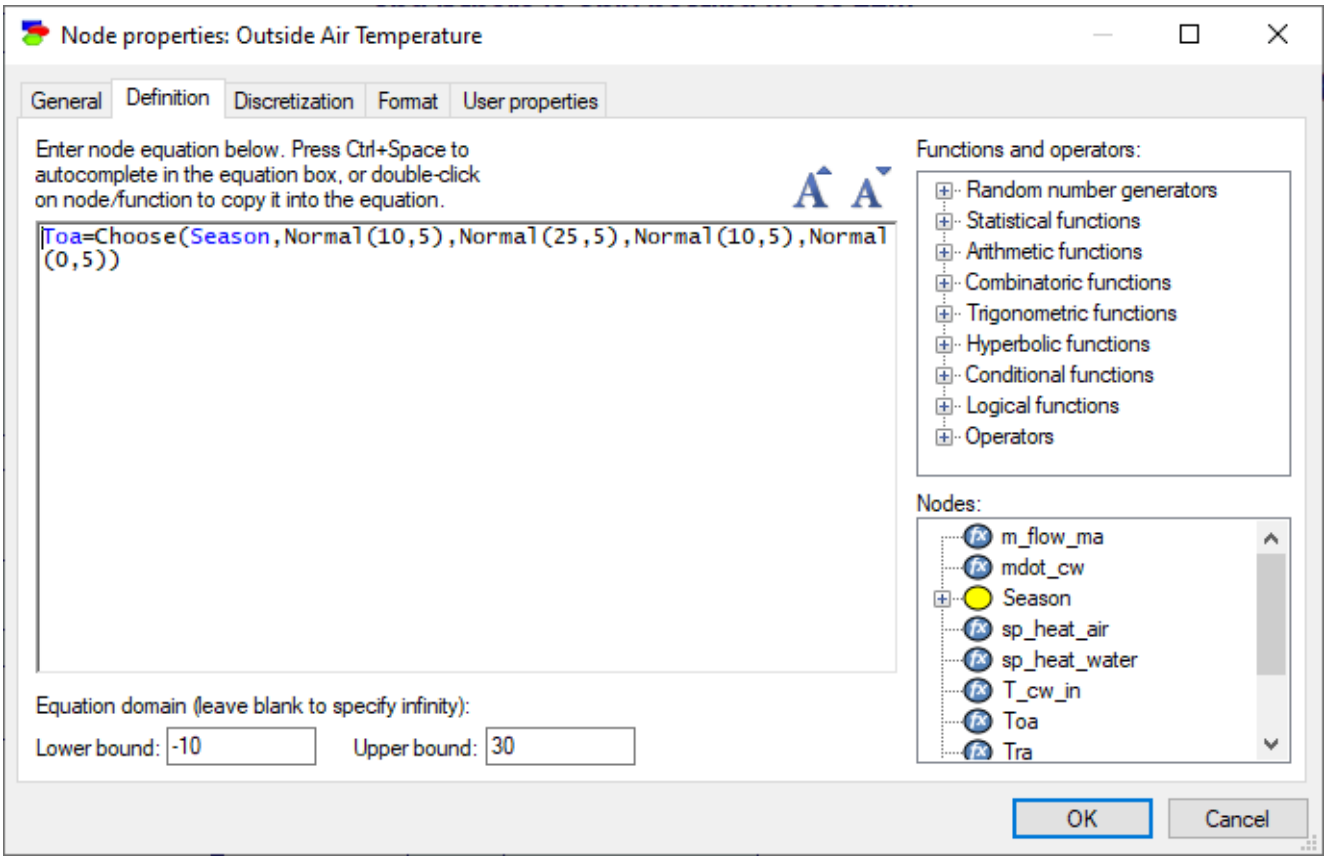
Season is a discrete variables with four states: *Spring*, *Summer*, *Fall* and *Winter*. *Perceived Temperature* is a discrete variable with three states: *Hot*, *Warm* and *Cold*. *Outside Air Temperature* is a continuous variable with probability distributions over its states distributed differently for each of the four possible seasons. The definition of the variable *Season* is a conditional probability table:

Spring	0.25
Summer	0.25
Fall	0.25
Winter	0.25

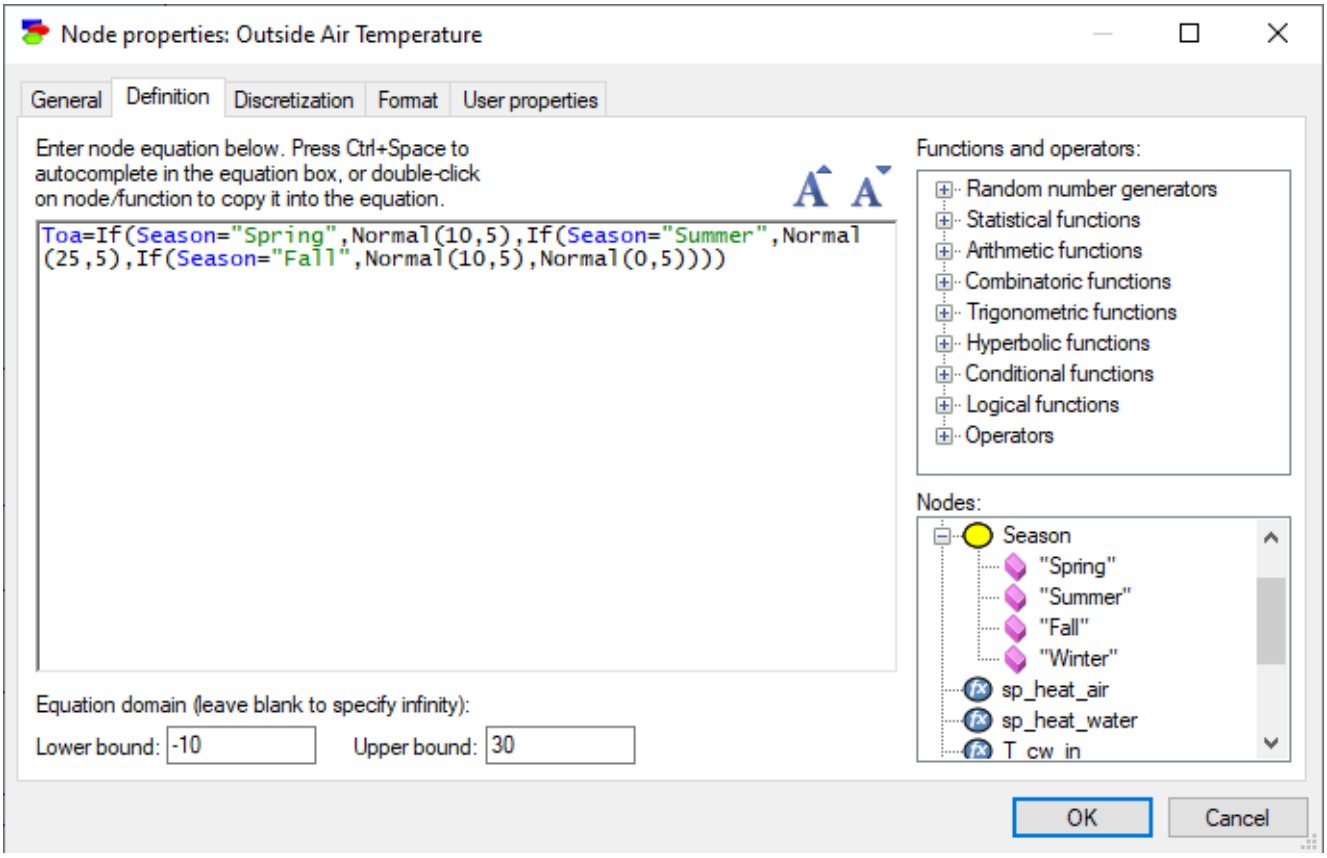
Continuous nodes with discrete parents

The case of a discrete parent of a continuous node is handled by considering as many cases as there are states in the parent. Obviously, with multiple discrete nodes being parents of a continuous nodes, we are dealing with as many cases are there are combinations of states of the discrete parents.

The definition of the variable *Outside Air Temperature* is an equation that is conditional on the variable *Season*:



Please note that we used the function *Choose()* to select a distribution over the possible temperatures depending on the state of the variable *Season*. This definition can be written in other ways, for example by means of nested *If()* function calls and referring directly to the individual states of the discrete parent:



When editing equations, you can make use of the lower-right dialog pane, which shows all variables and their states (to see the states of a discrete variable, please click on the plus sign to the left of it). Double-clicking on a variable or a state name drops that name into the current cursor position of the *Equation* editing pane.

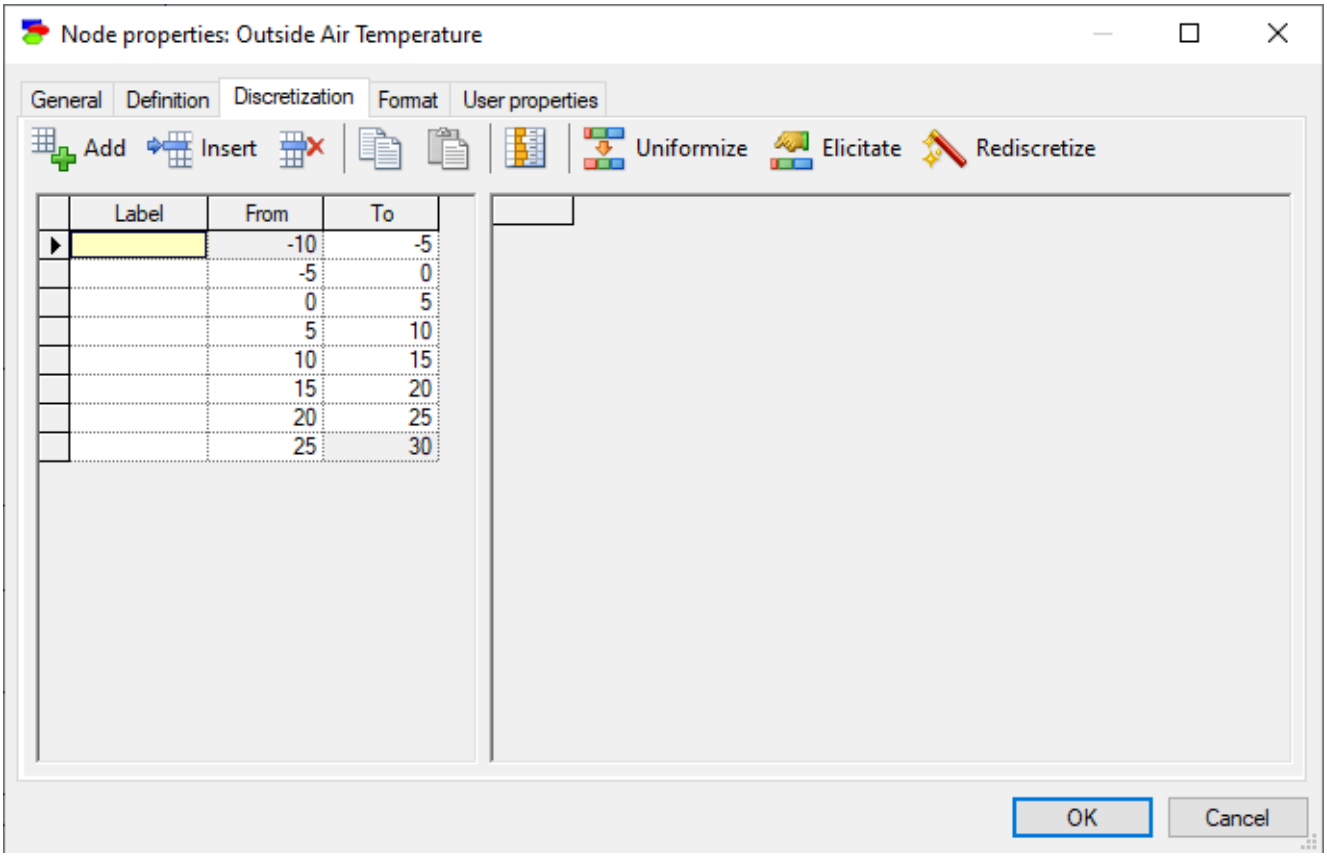
Discrete nodes with continuous parents

The case of a continuous parent of a discrete node is handled by discretization of the continuous parent. This brings the definition of the child to the familiar case of an interaction between discrete variables.

The definition of the variable *Perceived Temperature* is a conditional probability table, conditional on the variable *Outside Air Temperature*:

Outside Air Temperature	-10 .. -5	-5 .. 0	0 .. 5	5 .. 10	10 .. 15	15 .. 20	20 .. 25	25 .. 30
Hot	0.05	0.08	0.1	0.15	0.2	0.4	0.6	0.8
Warm	0.55	0.62	0.7	0.75	0.75	0.59	0.4	0.2
Cold	0.4	0.3	0.2	0.1	0.05	0.01	0	0

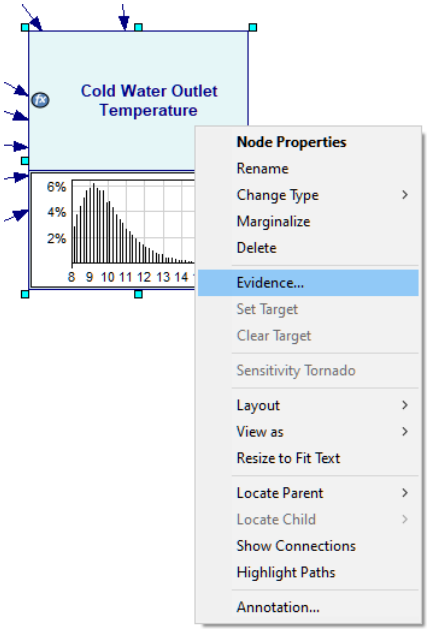
Because we need to have discrete states of the parent variable, the definition uses the discretization specified in *Outside Air Temperature*'s *Discretization* tab:



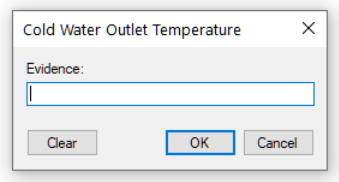
Please see the *Autodiscretization* section for the details of editing, uniformizing, rediscretizing, and eliciting discretization intervals. It deserves mentioning that in case a continuous variable has no discretization intervals defined, GeNIe creates default two discretization intervals when either an arc is drawn from the node to a discrete node or the autodiscretization algorithm is invoked.

6.7.5 Entering evidence in continuous nodes

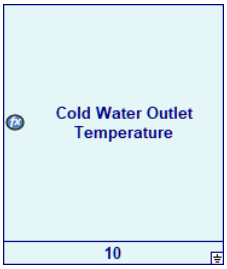
Evidence in continuous nodes can be entered by selecting *Evidence...* in the *Node Context Menu*.



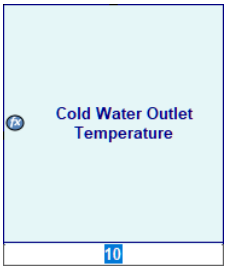
This invokes the dialog for entering evidence in continuous nodes.



Once a continuous node has evidence entered, we can see it displayed in the node



Changing the evidence or removing it altogether can be now accomplished by double-clicking on the evidence value



Next sections will cover inference and viewing results in hybrid models.

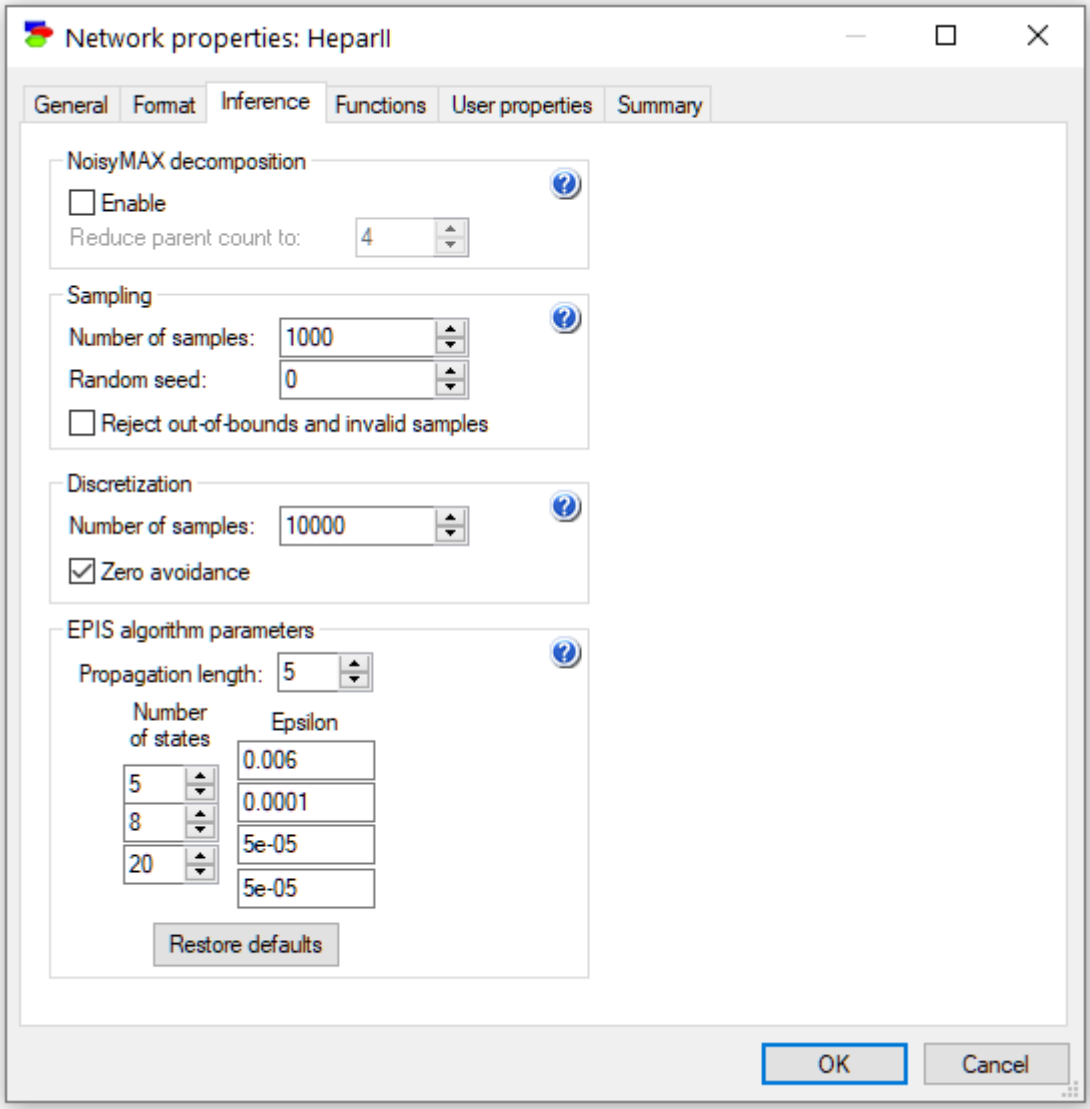
6.7.6 Inference in equation-based and hybrid models

The fundamental algorithm used for equation-based (continuous) and hybrid models is stochastic sampling. It is the preferred method when the network does not contain any evidence nodes. With evidence nodes, the situation becomes more complex and there are no universally reliable stochastic sampling algorithms. In all such cases, the algorithm for continuous and hybrid models is based on discretization.

Stochastic sampling algorithms

The reason why stochastic sampling algorithms are fundamental for equation-based models is that GeNIe puts no limitations on the models and, in particular, no limitations on the equations and distributions used in the node definitions. The modeling freedom given by GeNIe comes with a price tag - it prevents us from using any of the approximate schemes developed for special cases of continuous models. We will demonstrate the use of a stochastic sampling algorithm on the simple model used throughout this section.

Let us start with setting the number of samples to 1,000,000 from the default 10,000. We can do this in the *Sampling* pane of the *Inference* tab of the *Network properties*. We should generally choose the number of samples to be as large as possible given the constraints on the execution time. Execution time is pretty much linear in the number of samples, so mental calculation of what is admissible is easy. For the model at hand, consisting of only three variables, 1,000,000 will give us high quality results and, at the same time, it will not be noticeable in terms of execution time.



We invoke inference by pressing the *Update* ( ) tool on the *Standard Toolbar*. Section *Viewing results* discusses how to view and interpret the results of the algorithm.

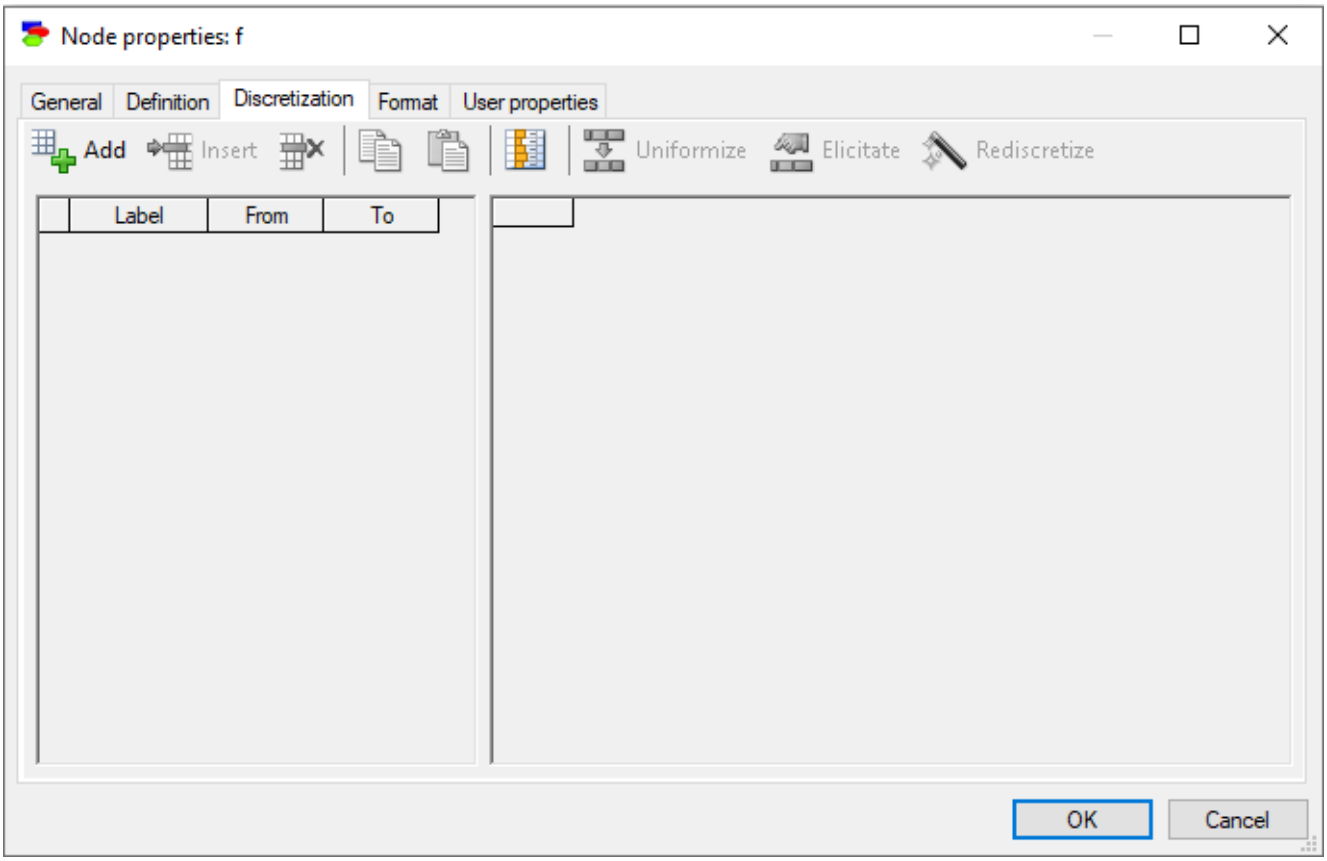
Auto-discretization

When a continuous or hybrid network contains evidence that is outside of the root (parent-less) nodes, there are no universally applicable stochastic sampling algorithms. GeNIe relies in all such cases on an algorithm that we call auto-discretization.

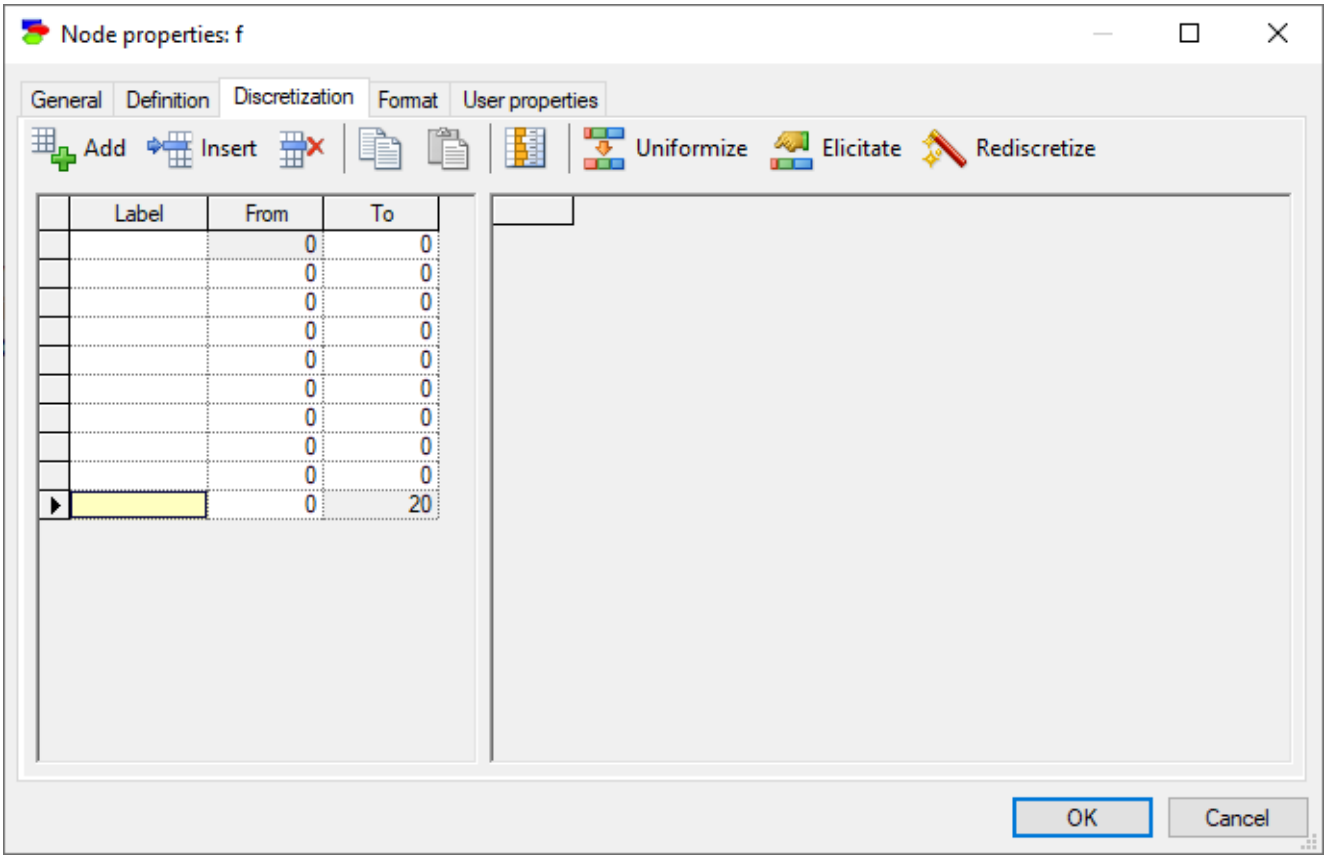
The algorithm translates the original continuous, equation-based network into a discrete Bayesian network. No changes are made to the original network definition but inference is performed in a temporary discrete Bayesian network, created solely for the purpose of inference. In order to use this algorithm, we need to enhance the definitions of the nodes in the network with a specification of the on-demand discretization. If we do not do it, GeNIe, in the spirit of preserving the syntactical correctness of every model, discretizes every variables into two

states. This, in most cases, is insufficient to obtain any insight and we advise to revisit the nodes in question in order to discretize them into more intervals.

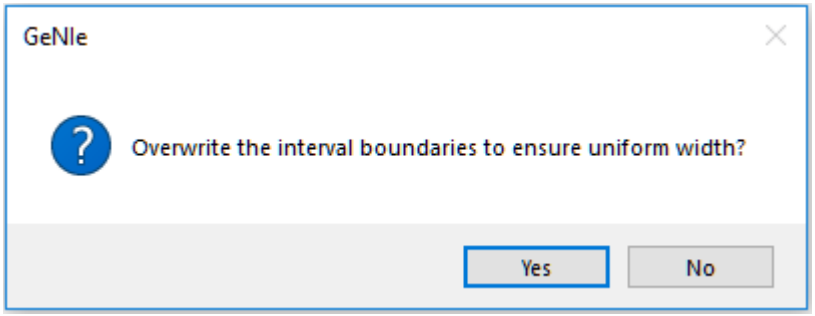
In our example, we start with the node *f*



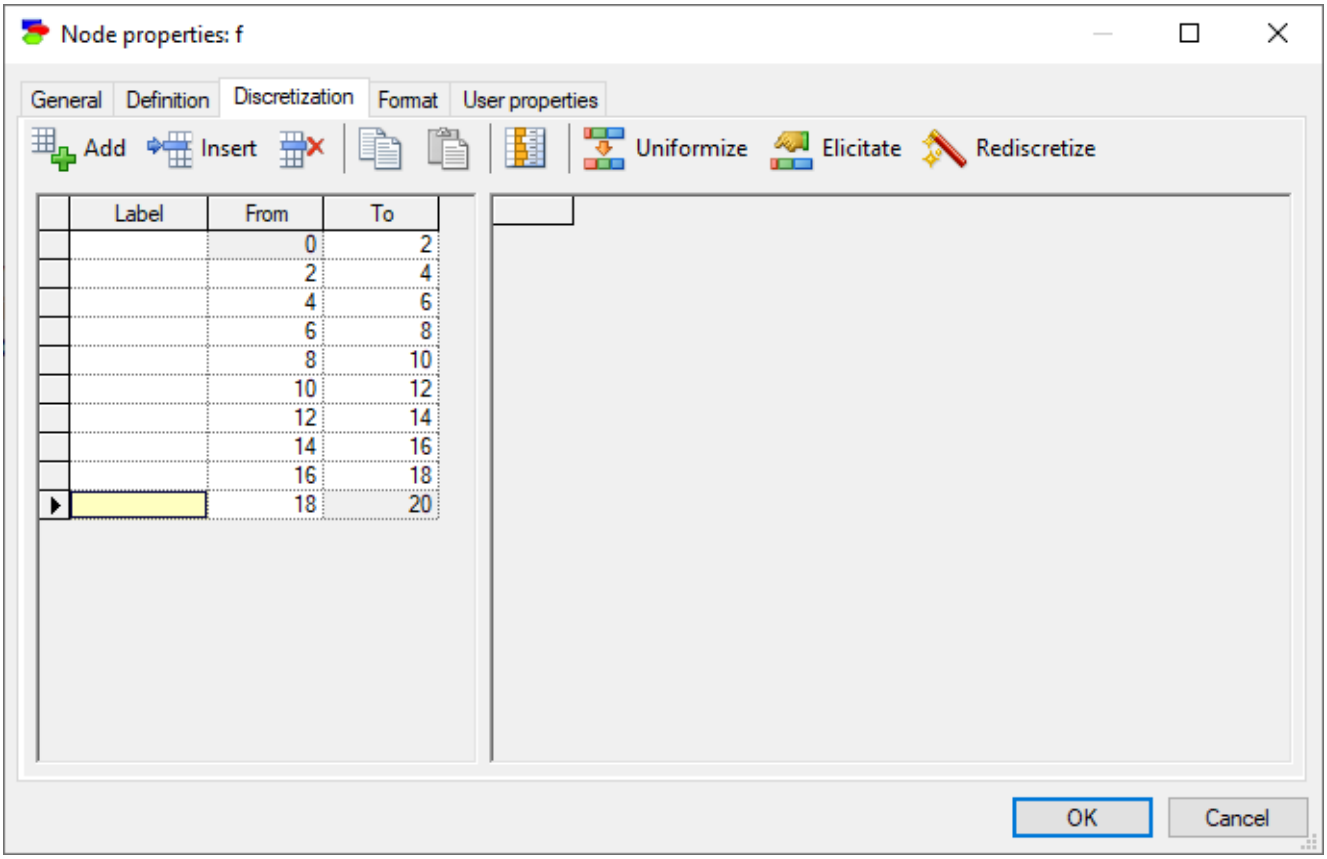
The three buttons in the top-left, *Add*, *Insert*, and *Delete* serve a similar purpose to the corresponding buttons in the *Definition* tab of discrete nodes. We add 10 intervals using the *Add interval* ( ) button



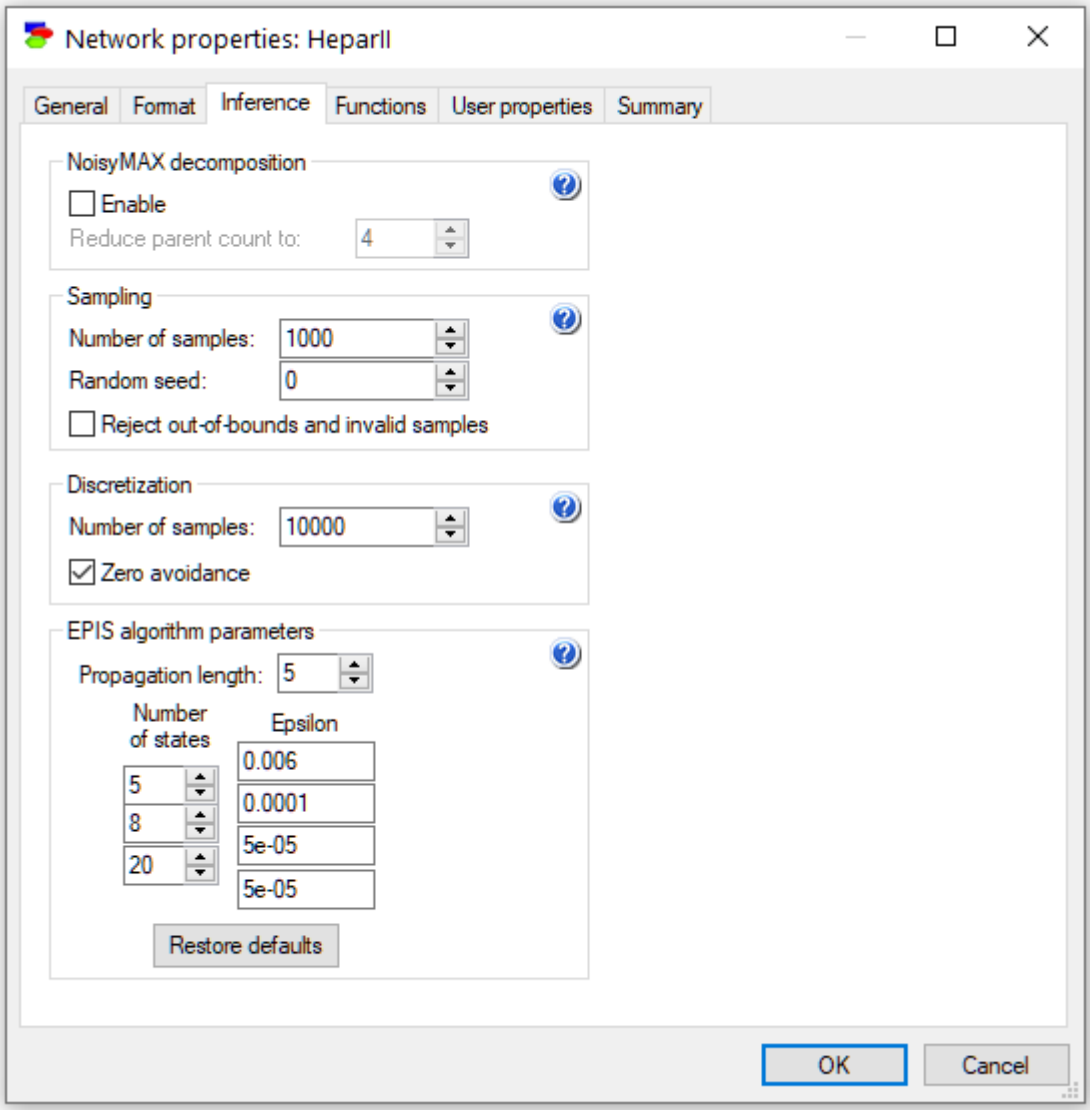
and subsequently press *Uniformize* ( ) button.



Pressing *Yes* creates new boundaries for the intervals and results in the following discretization

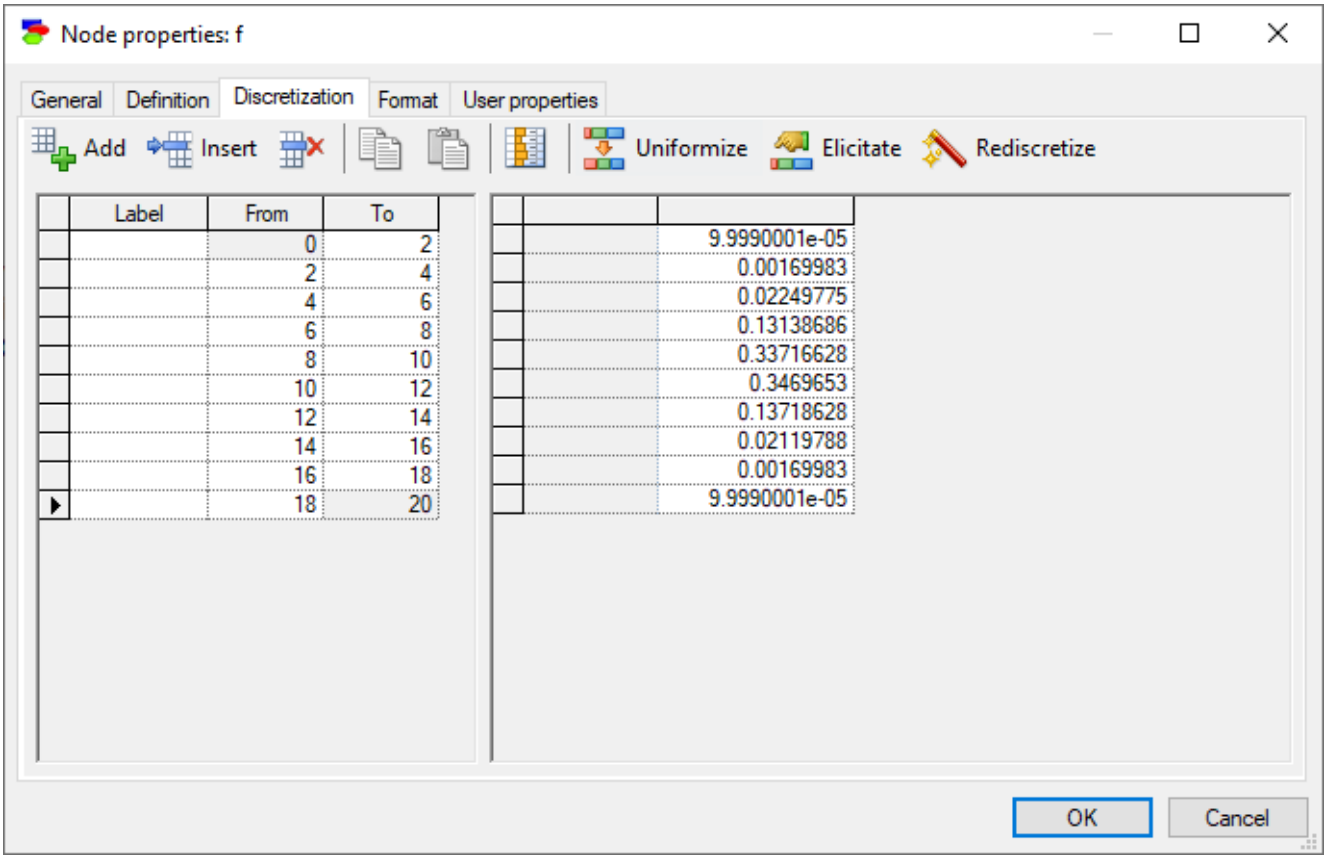


At this point, we have two choices: Elicitation and automatic derivation of the discrete distribution over the intervals from the continuous definition of the node. Pressing the *Rediscretize* ( ) button derives the probabilities from the definition of the node using stochastic sampling. The number of samples generated for the purpose of discretization is specified in the *Network* properties, *Inference* tab.



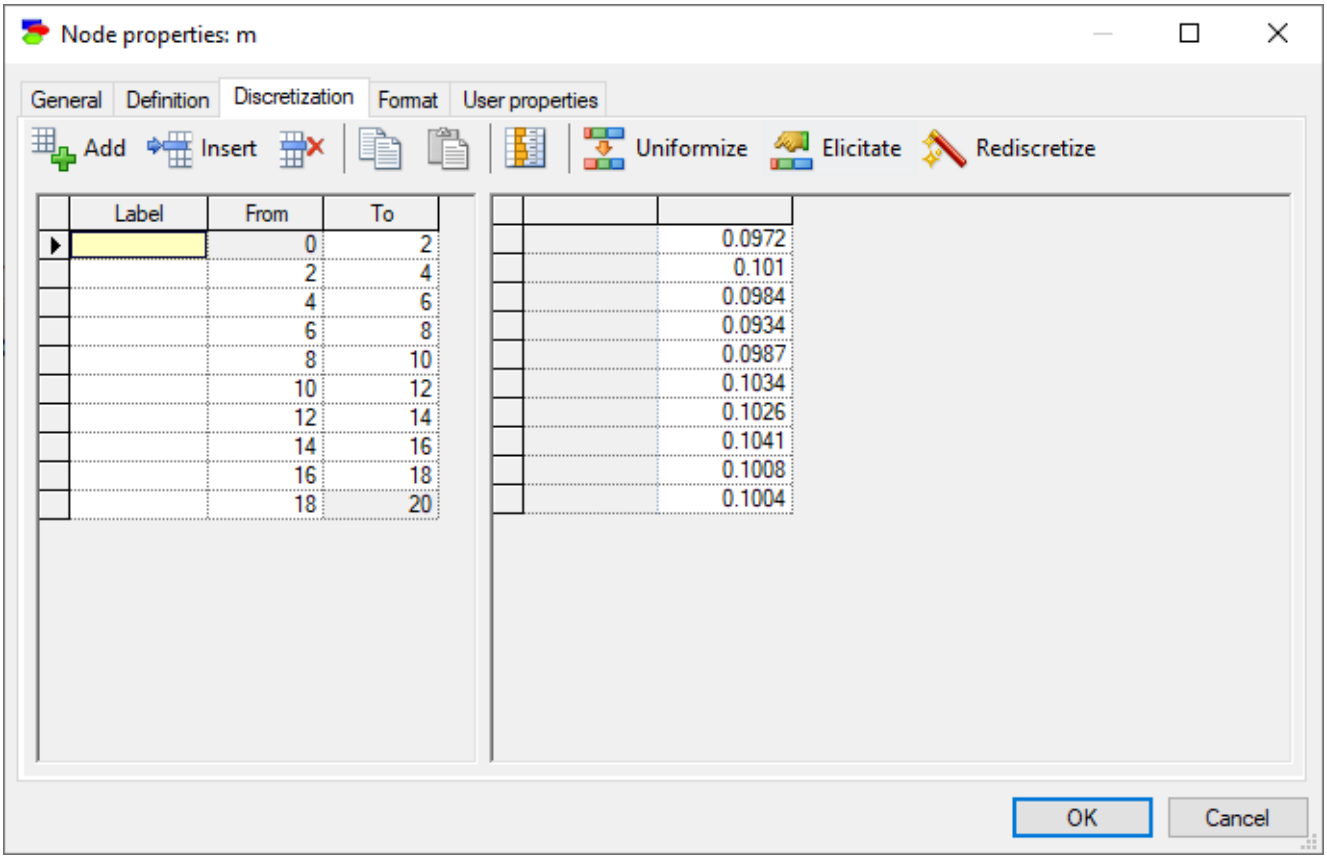
In addition to the *Number of discretization samples*, which determines the number of samples used in deriving the conditional probability tables in auto-discretized models, the tab allows the user to avoid zero probabilities in discretized conditional probability distributions. This option works only in chance nodes, i.e., nodes that contain reference to noise (expressed as calls to random number generator functions). Zeros in probability distributions lead to potential theoretical problems and should be used carefully, only if we know for sure that the probability should be zero. Once a zero, a probability cannot be changed, no matter how strong the evidence against it. We recommend (Onisko & Druzdzel, 2012) for a discussion of practical implications of this problem.

Rediscretization of the node *f* with 10,000 samples and zero avoidance should lead to the following discretized definition.



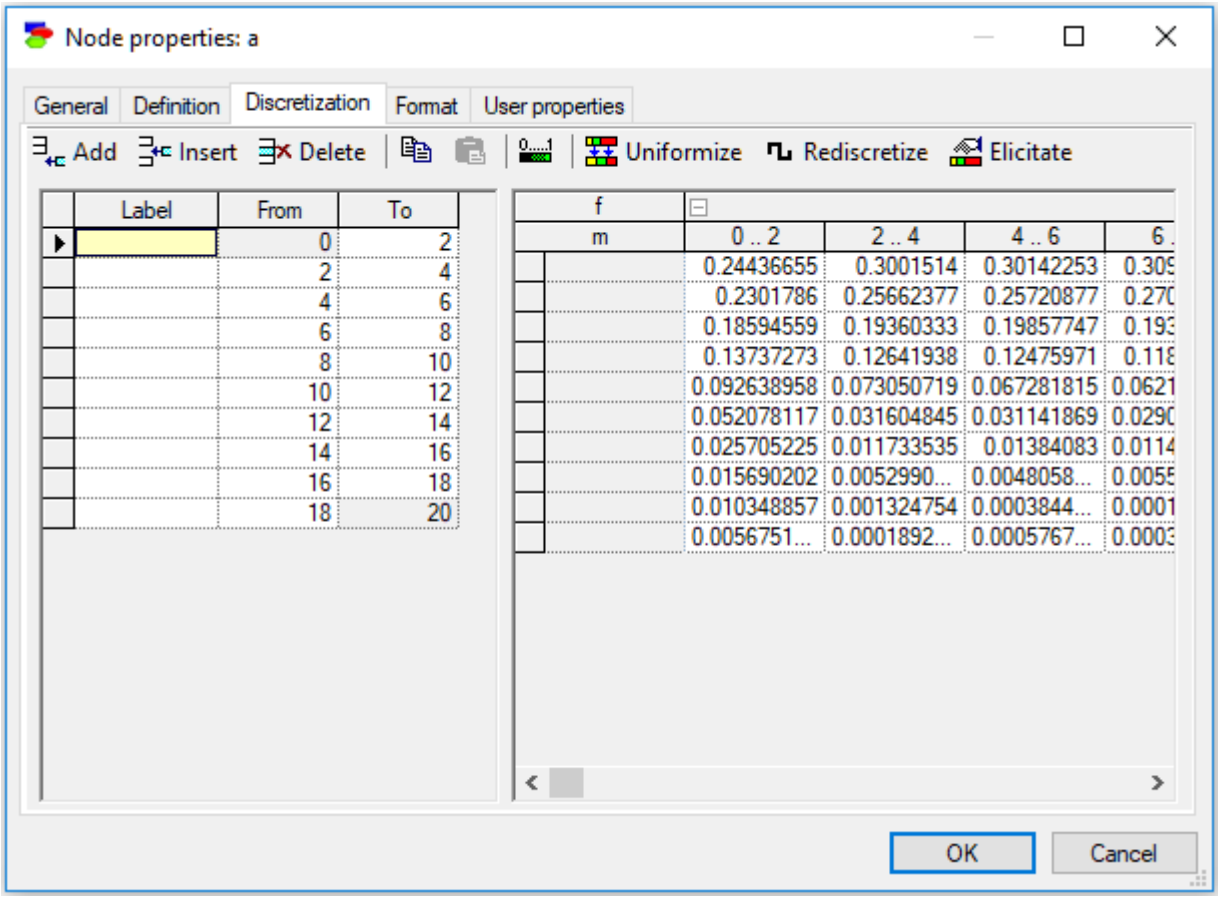
Label	From	To	
	0	2	9.9990001e-05
	2	4	0.00169983
	4	6	0.02249775
	6	8	0.13138686
	8	10	0.33716628
	10	12	0.3469653
	12	14	0.13718628
	14	16	0.02119788
	16	18	0.00169983
	18	20	9.9990001e-05

We repeat the same process for the variable *m* with also 10 intervals.



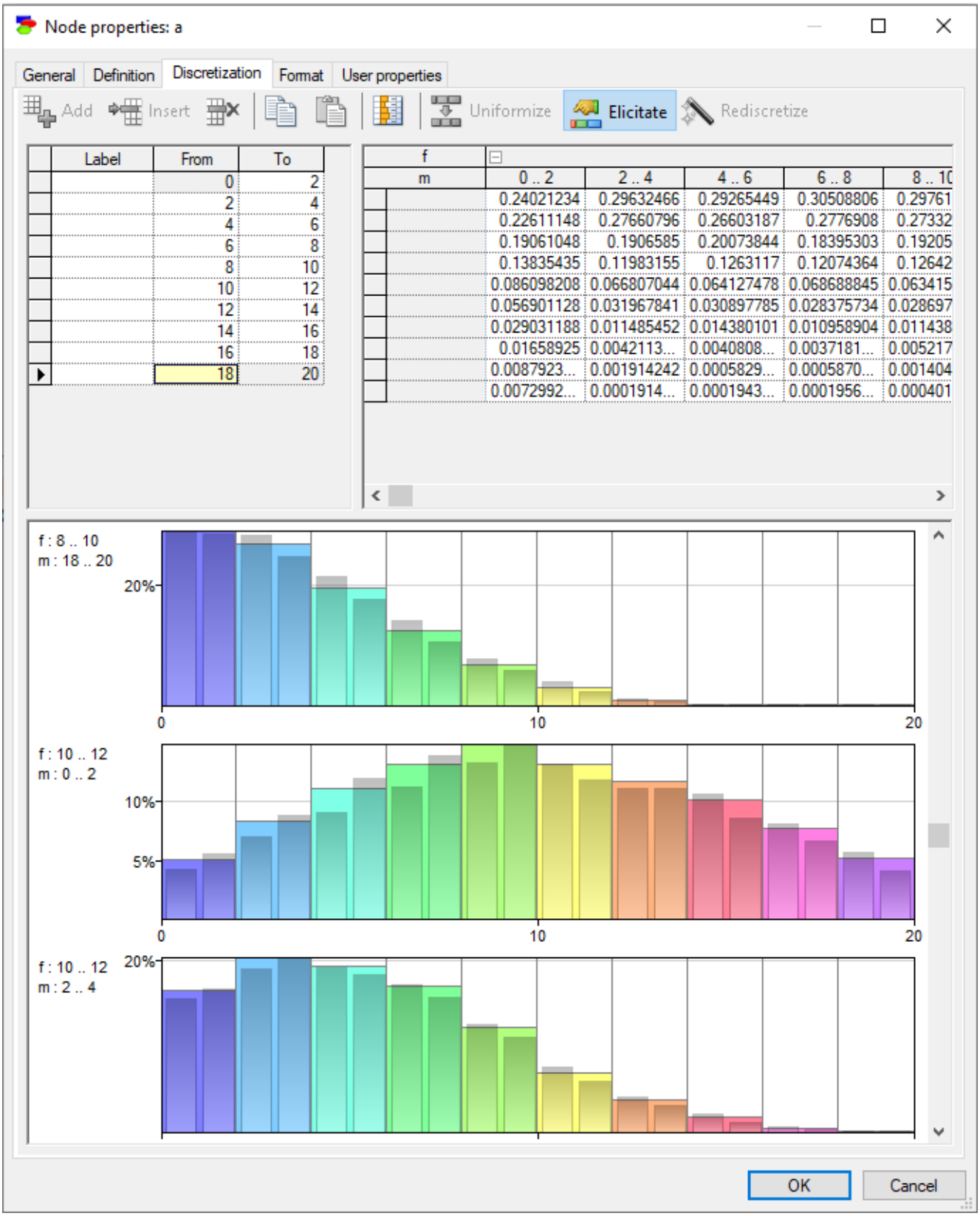
Label	From	To	
	0	2	0.0972
	2	4	0.101
	4	6	0.0984
	6	8	0.0934
	8	10	0.0987
	10	12	0.1034
	12	14	0.1026
	14	16	0.1041
	16	18	0.1008
	18	20	0.1004

And for the variable a with 10 intervals.



Please note that the CPT for node *a* is very large and contains 10x10=100 distributions, one for each combination of discretized values of the variables *f* and *m*. The screen shot above shows only its fragment.

Pressing *Elicitate* ( ) button shows the node's CPT and allows us to adjust the discretization boundaries manually.



Label	From	To
	0	2
	2	4
	4	6
	6	8
	8	10
	10	12
	12	14
	14	16
	16	18
	18	20

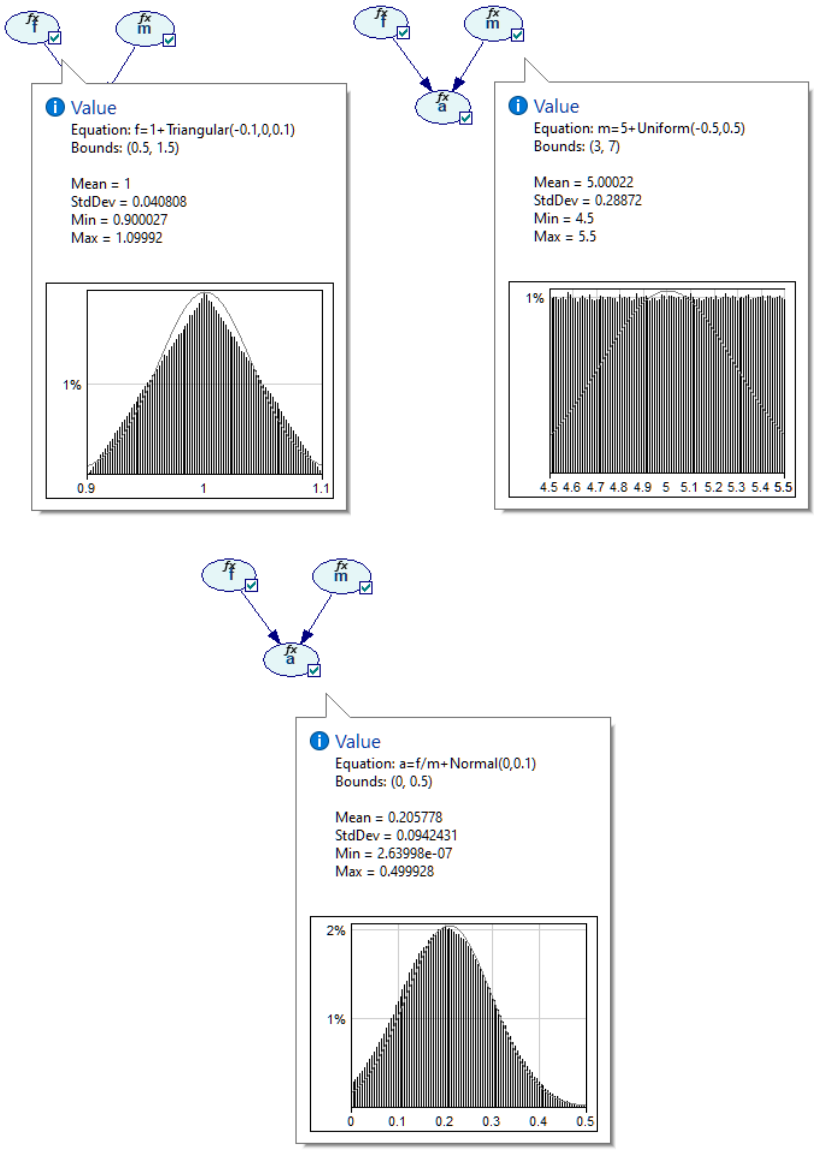
f / m	0 .. 2	2 .. 4	4 .. 6	6 .. 8	8 .. 10
	0.24021234	0.29632466	0.29265449	0.30508806	0.29761
	0.22611148	0.27660796	0.26603187	0.2776908	0.27332
	0.19061048	0.1906585	0.20073844	0.18395303	0.19205
	0.13835435	0.11983155	0.1263117	0.12074364	0.12642
	0.086098208	0.066807044	0.064127478	0.068688845	0.063415
	0.056901128	0.031967841	0.030897785	0.028375734	0.028697
	0.029031188	0.011485452	0.014380101	0.010958904	0.011438
	0.01658925	0.0042113...	0.0040808...	0.0037181...	0.005217
	0.0087923...	0.001914242	0.0005829...	0.0005870...	0.001404
	0.0072992...	0.0001914...	0.0001943...	0.0001956...	0.000401

Similarly to the histogram dialog in *Cleaning Data* section, you can adjust the boundaries of the discretization intervals manually by dragging the vertical lines that separate the samples towards right and left and observing how this impacts each of the conditional probability tables for the node (you can see each of the 100 conditional probability distributions by scrolling the dialog). The conditional probability tables are pictured as histograms of discretization samples.

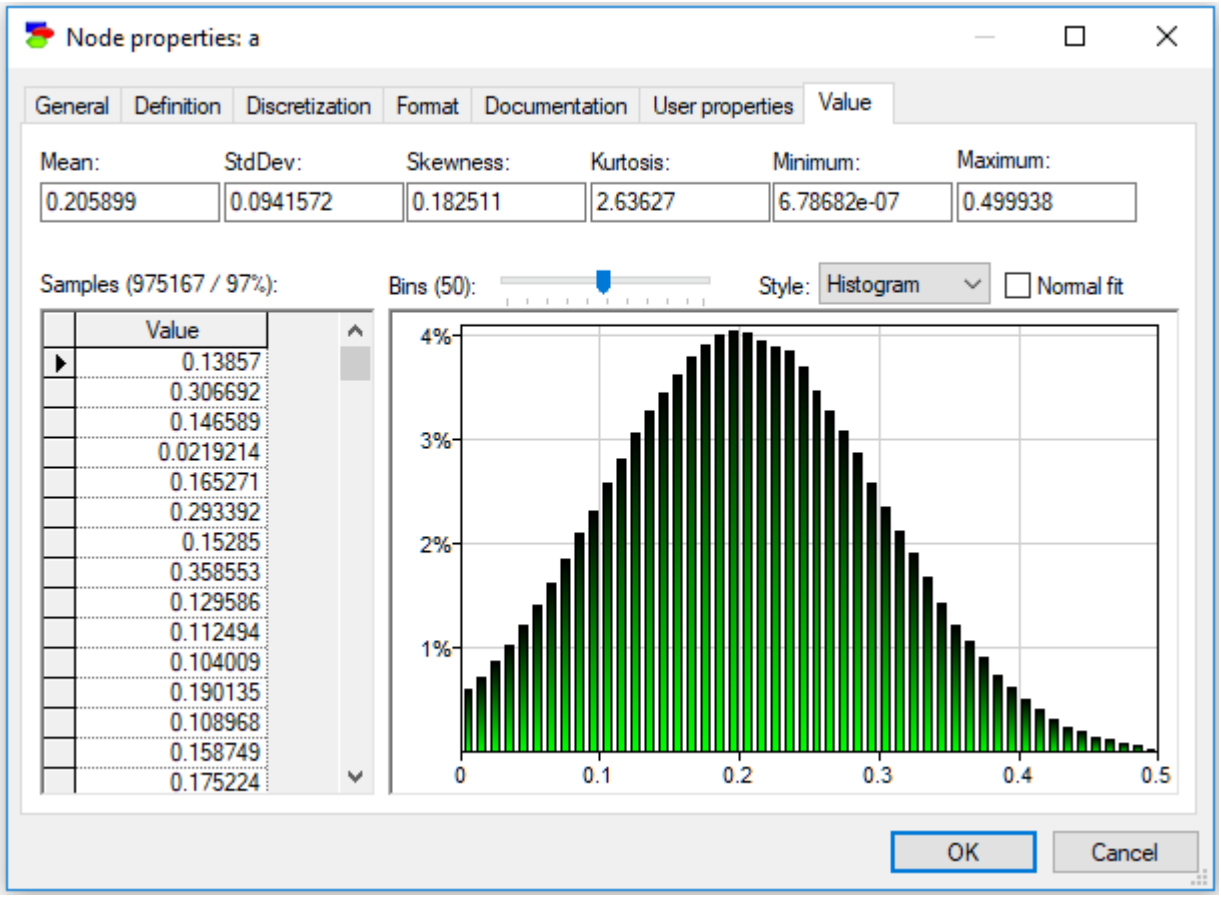
The discretization is all we need to run the auto-discretization algorithm. We run the algorithm by pressing the *Update* ( ) tool on the *Standard Toolbar*.

6.7.7 Viewing results in equation-based models

When an equation node is updated, hovering over its *Updated* ( ) icon shows the result in form of the marginal probability distribution over the node. Here are the results of the three variables in the example used throughout this section:

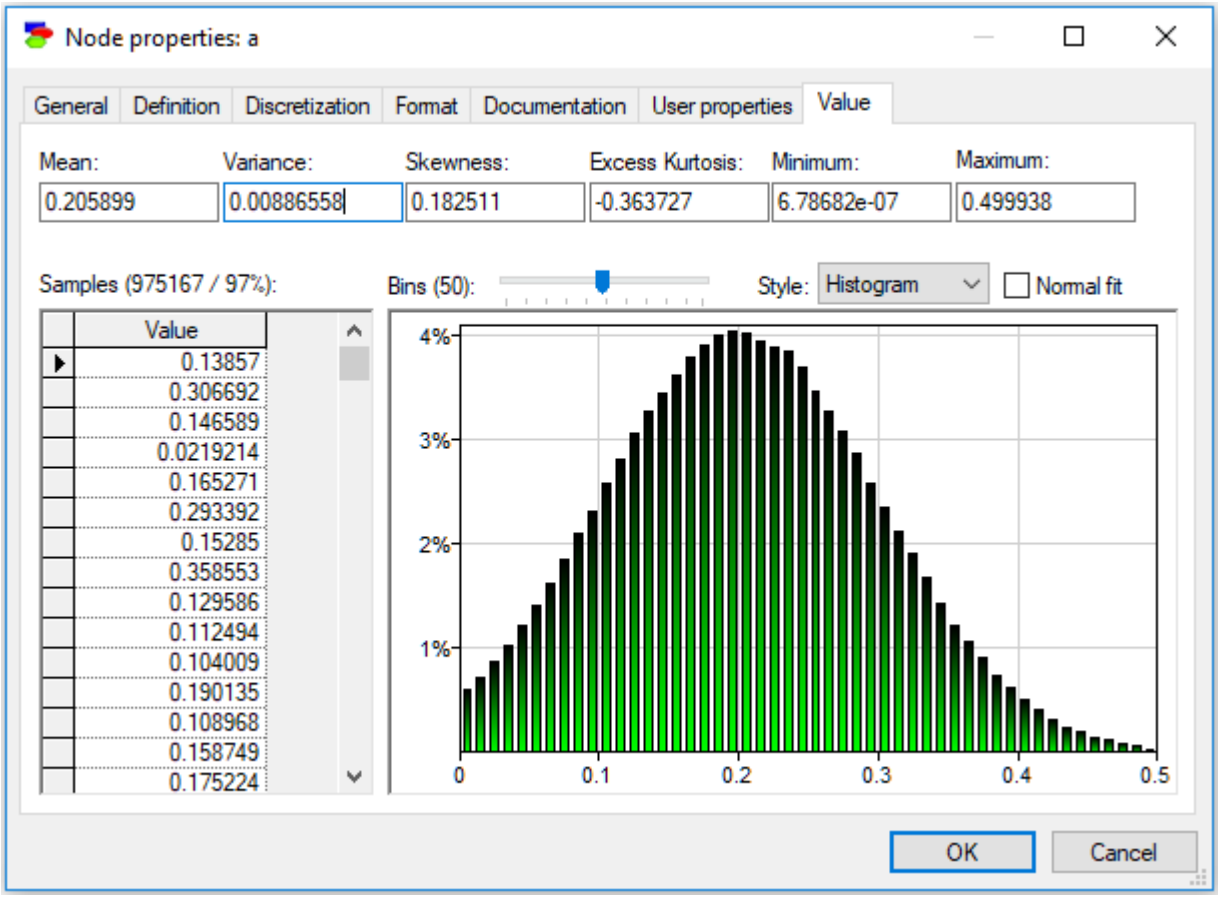


We can examine these distributions in more detail on the *Value* tab of the *Node* properties. Node *a* is most interesting from the point of view of the model.

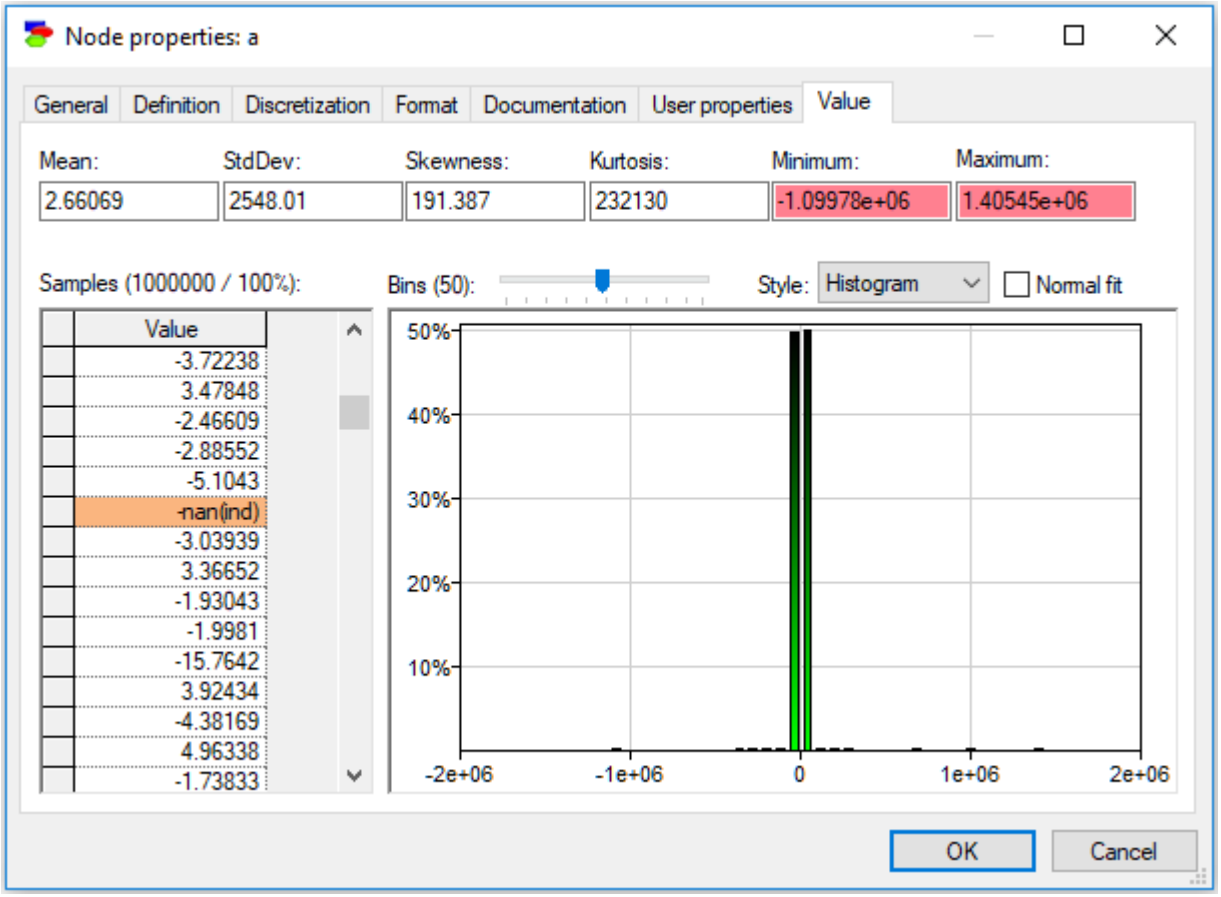


Equation nodes are continuous and show the results in form of a plot of the samples obtained during the most recent run of the sampling algorithm. The samples themselves are preserved and displayed in the vector on the left-hand side. The tab shows the first four moments of the marginal distribution over *a*: *Mean*, *StdDev*, *Skewness* and *Kurtosis* along with the *Minimum* and the *Maximum*. Similarly to the histogram interface in the data pre-processing module, the user can change the number of bins in the histogram.

A somewhat hidden feature of this dialog is that one can switch between *StdDev* and *Variance* by double-clicking on their text boxes. The same can be done for *Kurtosis*, which can be switched to *Excess Kurtosis*.

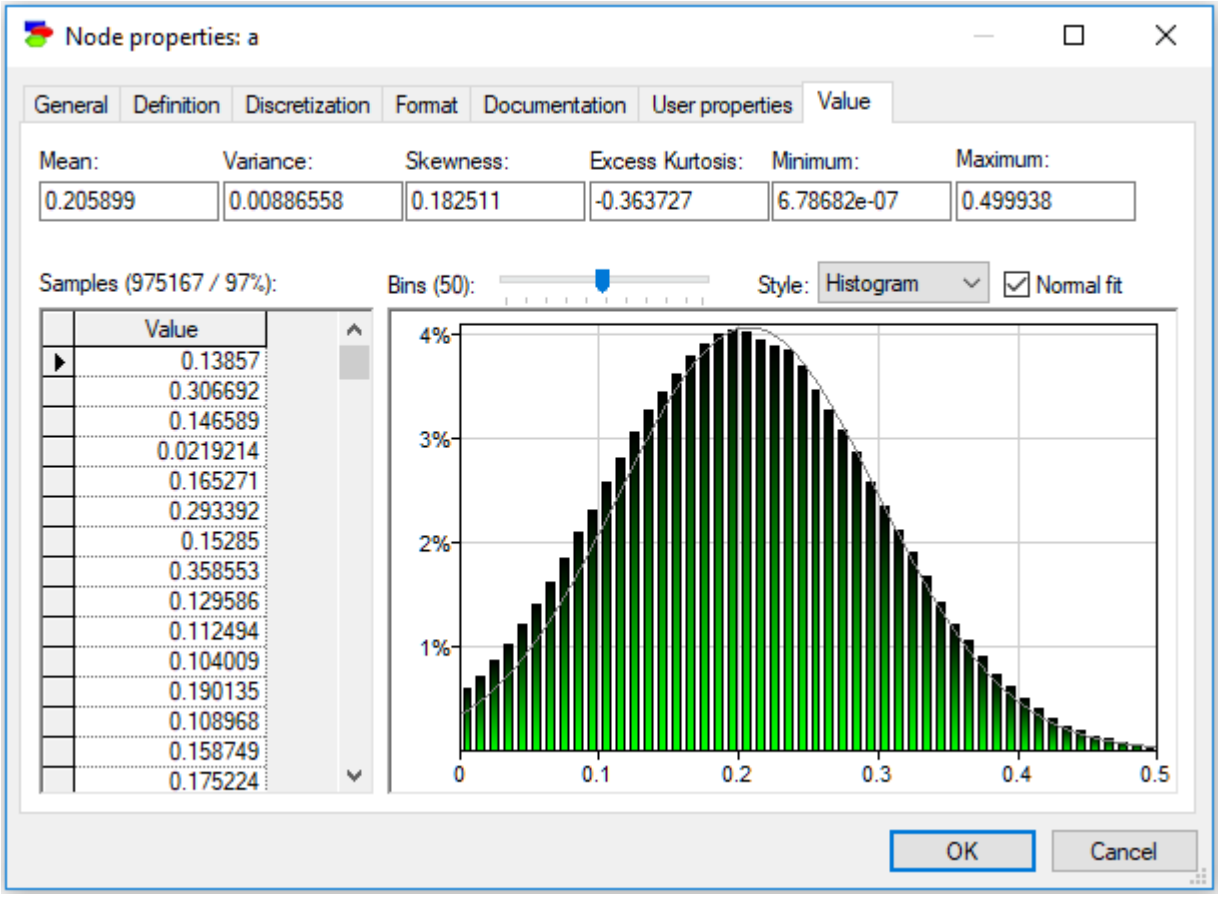


It is worth noting on the side that when a sample is invalid (*nan*, i.e., not a number), it is displayed as a *nan* and highlighted in orange. The values of *Minimum* and *Maximum* in the screen shot below are shown in red because the samples from which they are derived fall outside of the range designated on the definition tab ([0..0.5]).

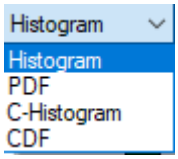


It is possible to reject all invalid and out-of-bounds sample during the sampling process by setting the *Reject out-of-bounds and invalid samples* option in the *Inference* tab of the *Network properties*. Invalid and out-of-bounds samples are treated in the same way - they are rejected as soon as they appear.

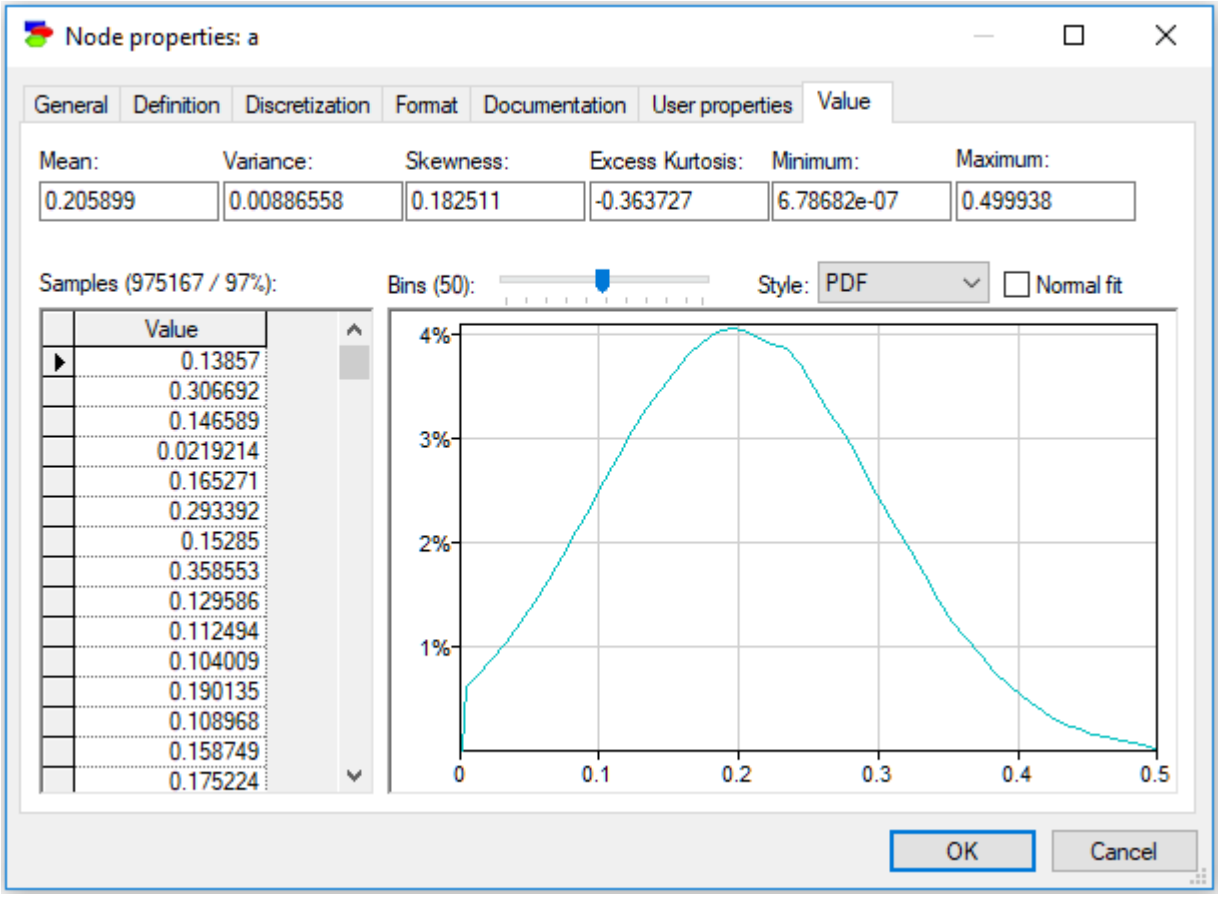
Normal fit check box draws a Normal distribution over the domain of the variable with the mean and standard deviation equal to those of the samples. This allows for judging whether the distribution is Normal or not.



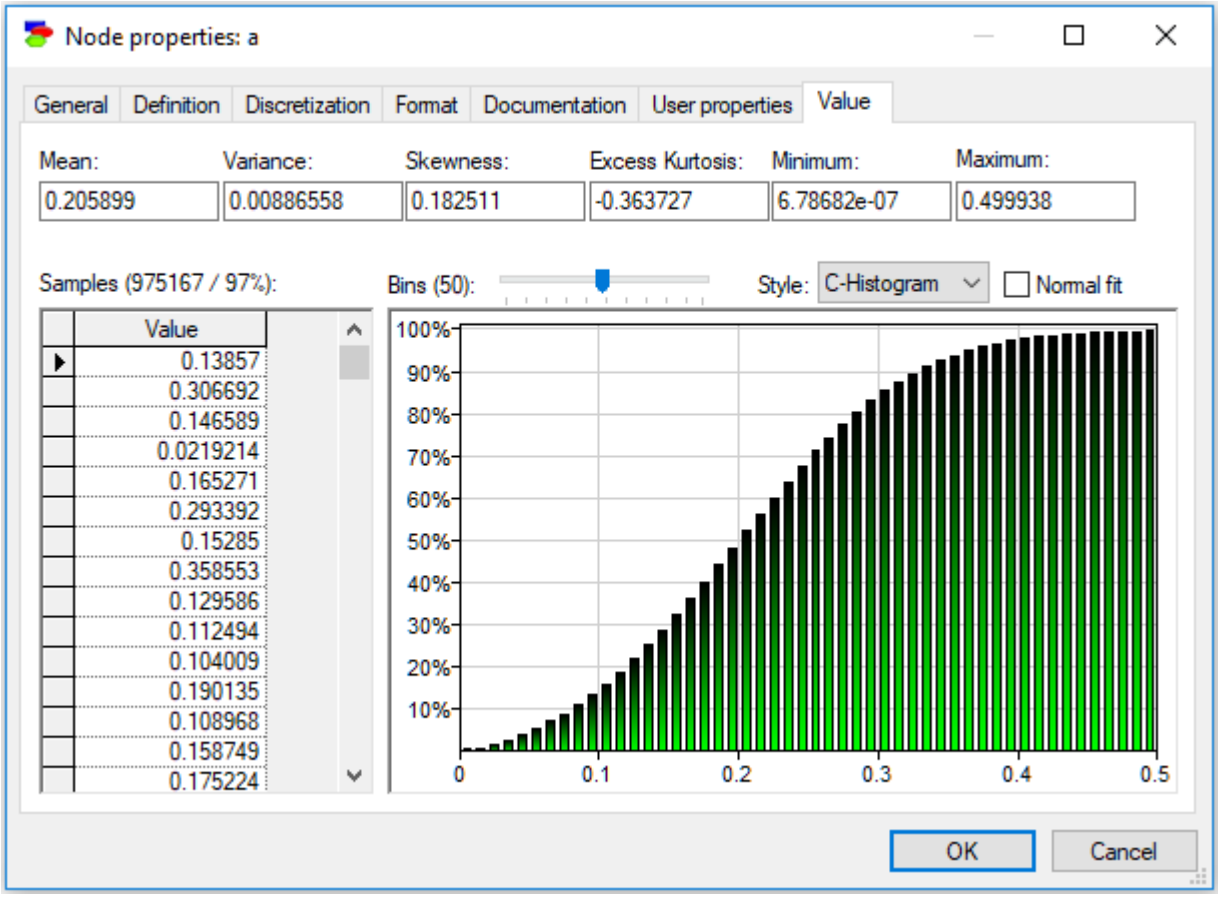
Style pop-up menu allows for choosing a different plot:



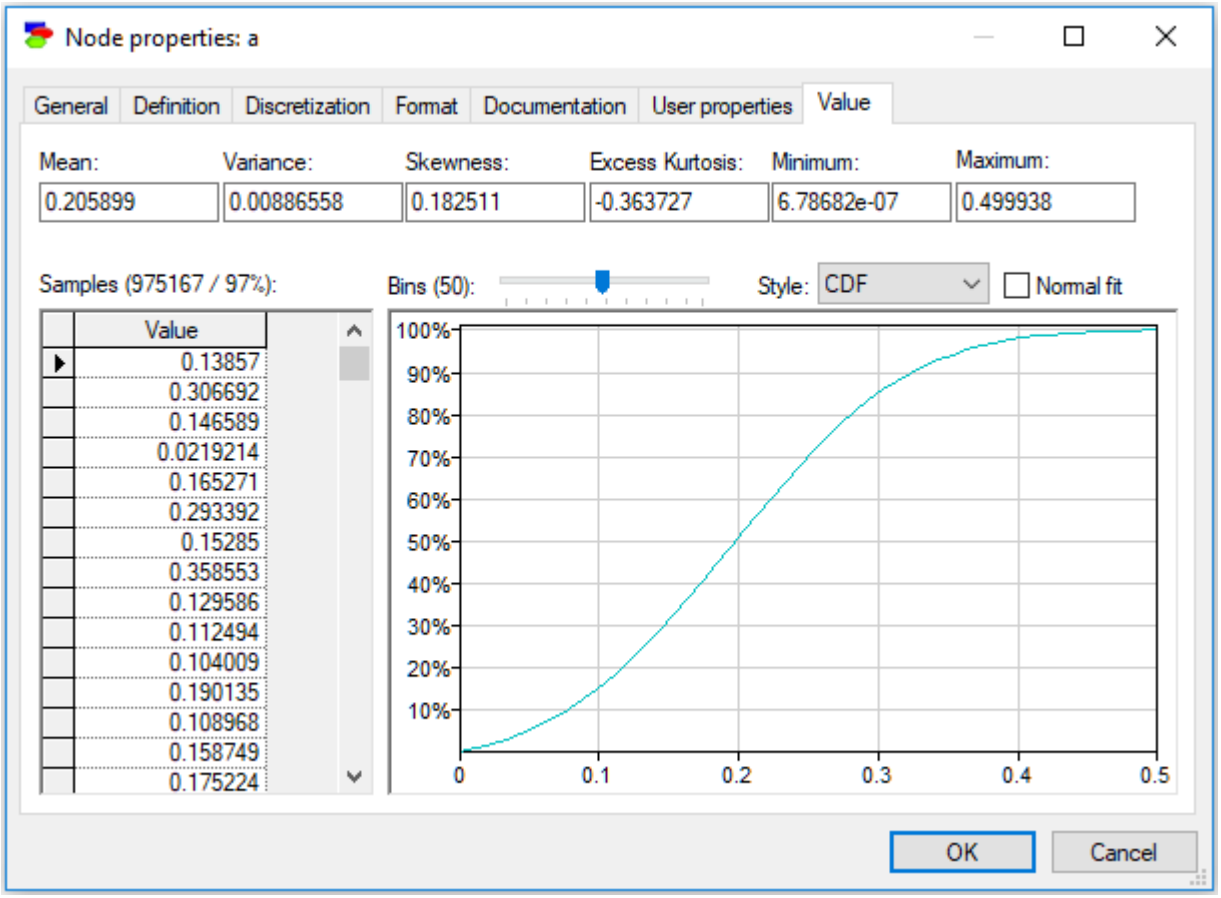
PDF is an abstraction of the histogram plot:



C-Histogram is a cumulative version of the *Histogram* plot.



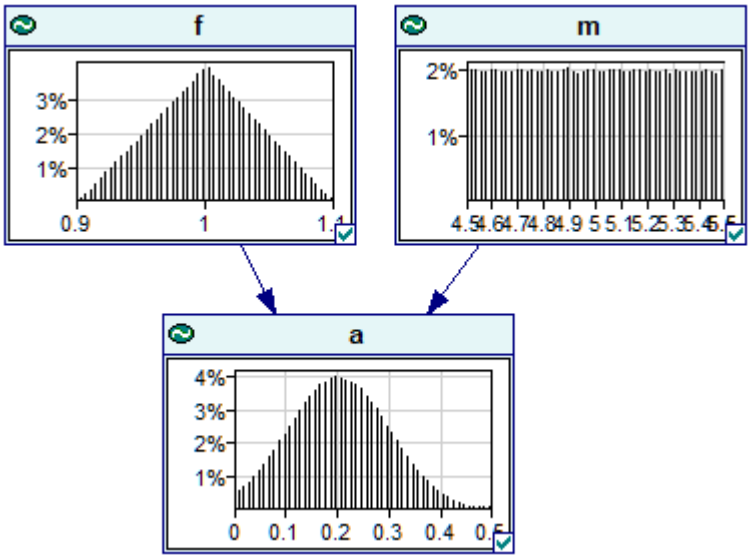
CDF is an abstraction of the cumulative histogram plot:



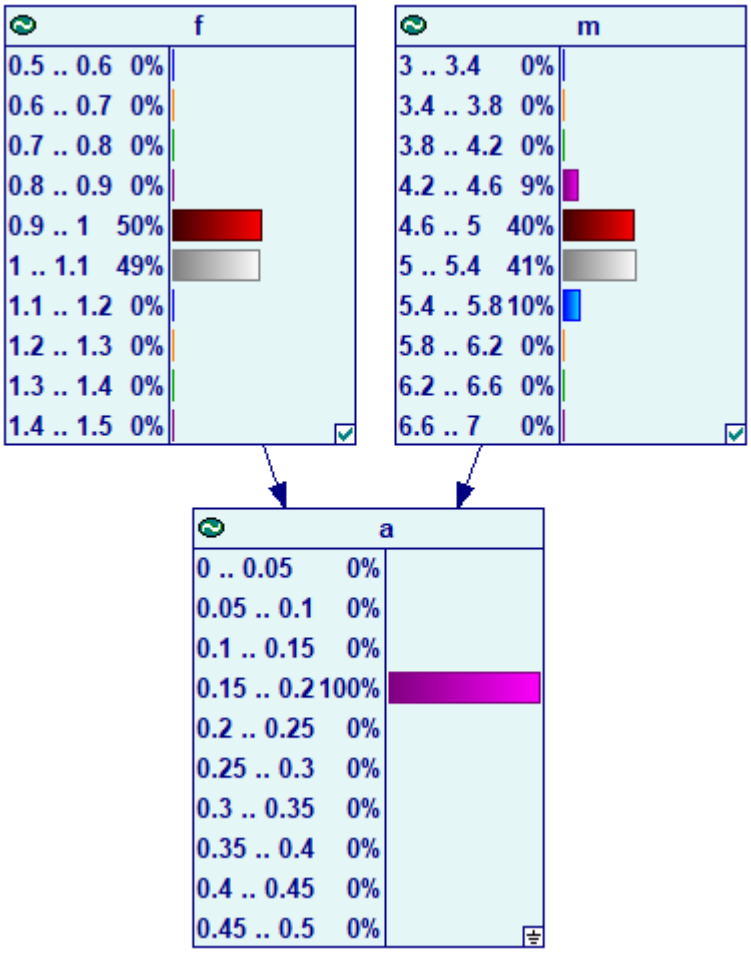
It is useful to note that the marginal probability distribution of *a* does not follow the Normal distribution - while its probability density function (PDF) follows a uni-modal bell-shaped curve, there is a visible difference between the curve and the Normal distribution plotted alongside. This is not surprising given the definition of variables *a*, *f*, and *m*.

With the *AutoDiscretize* algorithm used for inference in continuous models, the *Value tab* of *Equation* nodes is identical to those of *Chance* nodes in Bayesian networks.

It is also possible to view the marginal probabilities in continuous and hybrid models by changing the view from *Icon* to *Bar Chart*. In case of the sampling algorithm, the results are displayed as follows:



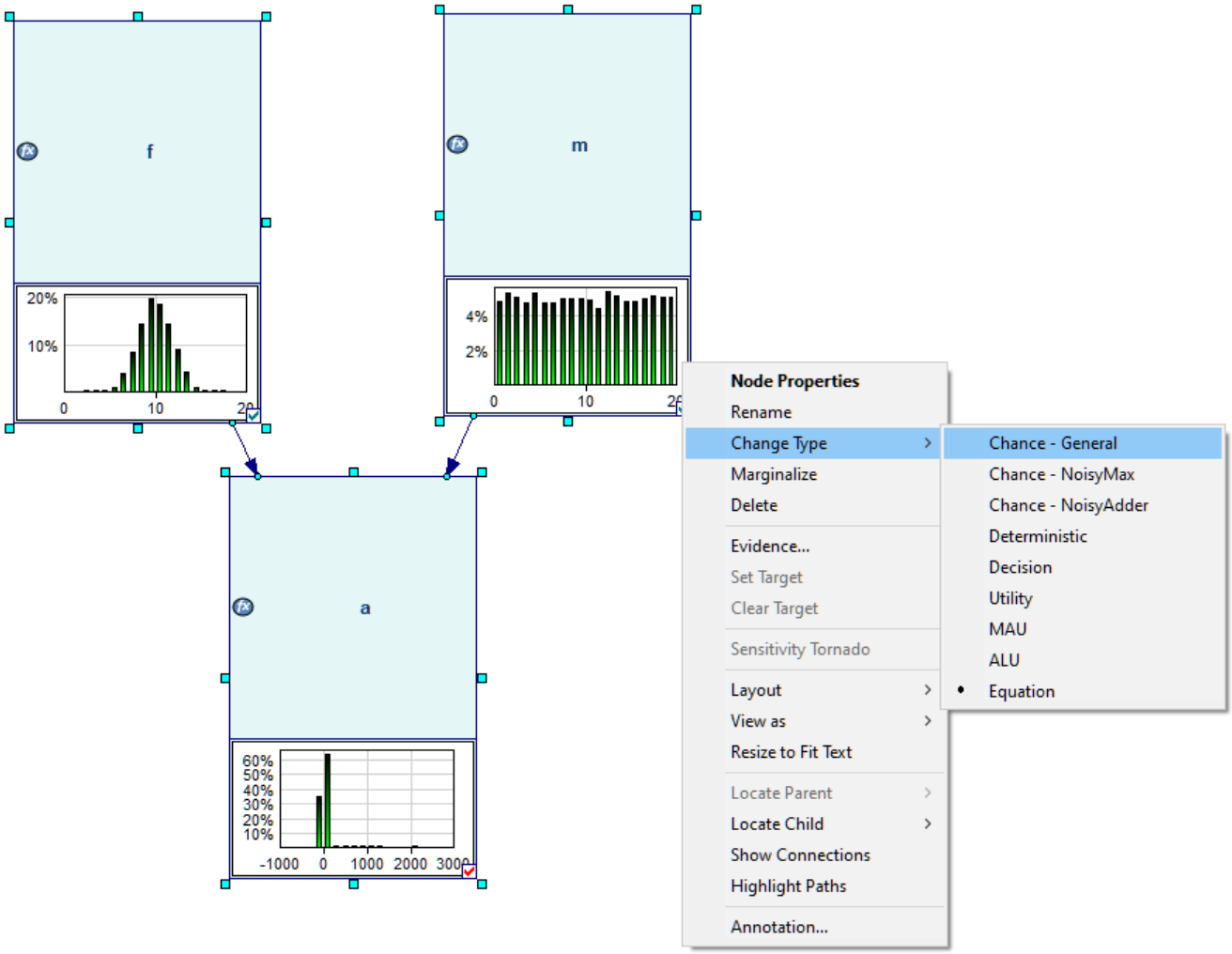
When we enter evidence in a node with parents (in this case, evidence in node *a*), GeNIe invokes the auto-discretization algorithm and the results look identically to those of discrete Bayesian networks:



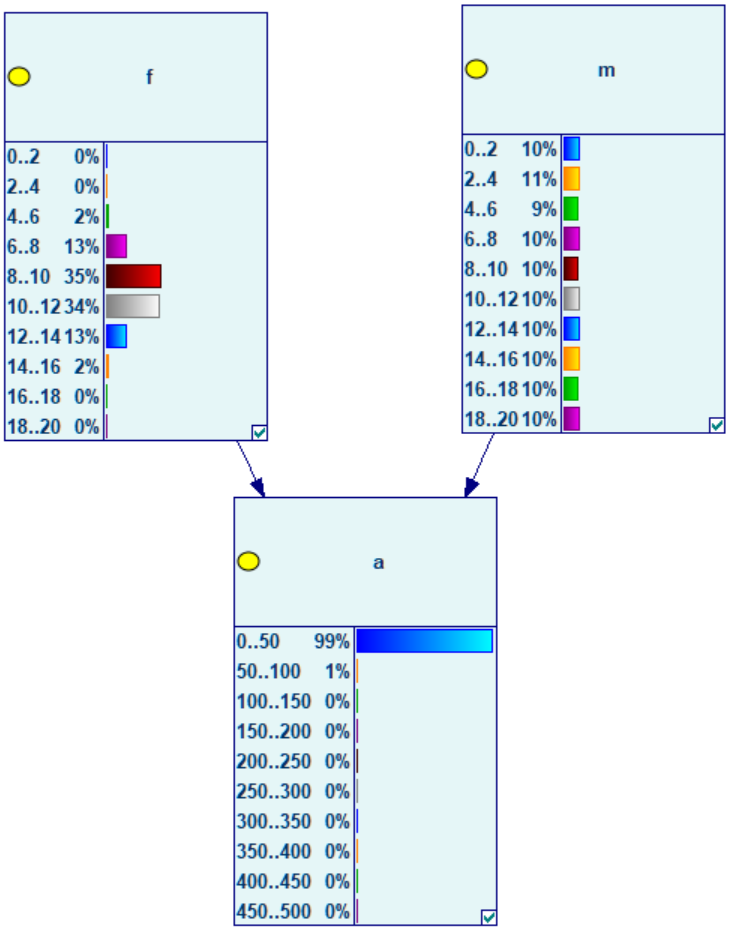
6.7.8 Discretization of a hybrid network

To convert a hybrid into a discrete Bayesian network, please select all nodes (CTRL-A) and then right-click on any of the selected nodes and select *Change Type/Chance-General* from the node context menu. The resulting network will be an auto-discretized equivalent of the original network. Of course, it is possible to convert only a part of the network by selecting a subset of nodes to be converted.

For example the following network:



will be converted into the following discrete network:


f
0..2 0%
2..4 0%
4..6 2%
6..8 13%
8..10 35%
10..12 34%
12..14 13%
14..16 2%
16..18 0%
18..20 0%
m
0..2 10%
2..4 11%
4..6 9%
6..8 10%
8..10 10%
10..12 10%
12..14 10%
14..16 10%
16..18 10%
18..20 10%
a
0..50 99%
50..100 1%
100..150 0%
150..200 0%
200..250 0%
250..300 0%
300..350 0%
350..400 0%
400..450 0%
450..500 0%

6.8 Geo-processing

6.8.1 Introduction

A growing number of users apply Bayesian networks to processing geographical data. In this case, a typical processing occurs at the level of individual cells of a raster representing a map. Processing amounts to collecting input parameters (used as evidence in a Bayesian network model) from individual maps and producing maps that contain information derived by means of the Bayesian network model. This happens at each cell of the raster map. For example, given the height above the sea level of a map point, average amount of rainfall, average temperature, etc., a Bayesian network may derive the probability of vegetation in that particular raster cell.

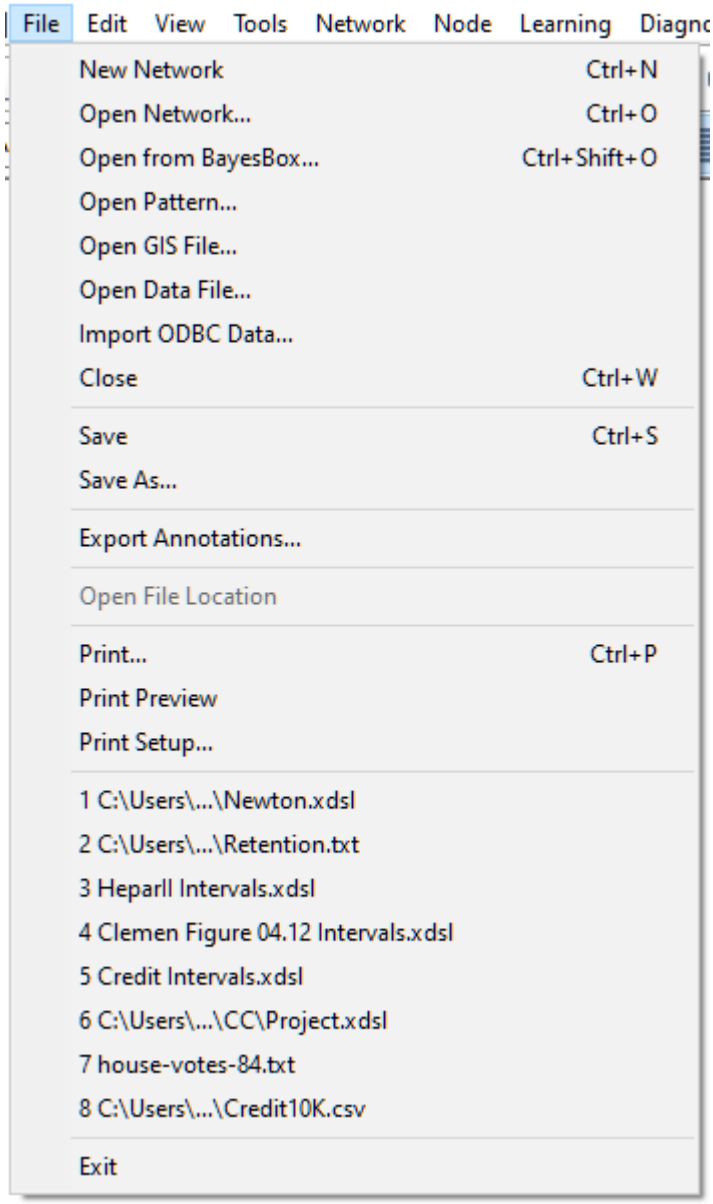
It is possible to perform these calculations using SMILE without the geo-processing extensions. In this case, one would have to read data for every raster cell from every input map, process that information using a Bayesian network model, and then produce values for the output cells. Direct support for this calculation makes processing much faster, as there is no overhead required for accessing SMILE for every single raster cell.

This section describes GeNIe's map processing capability.

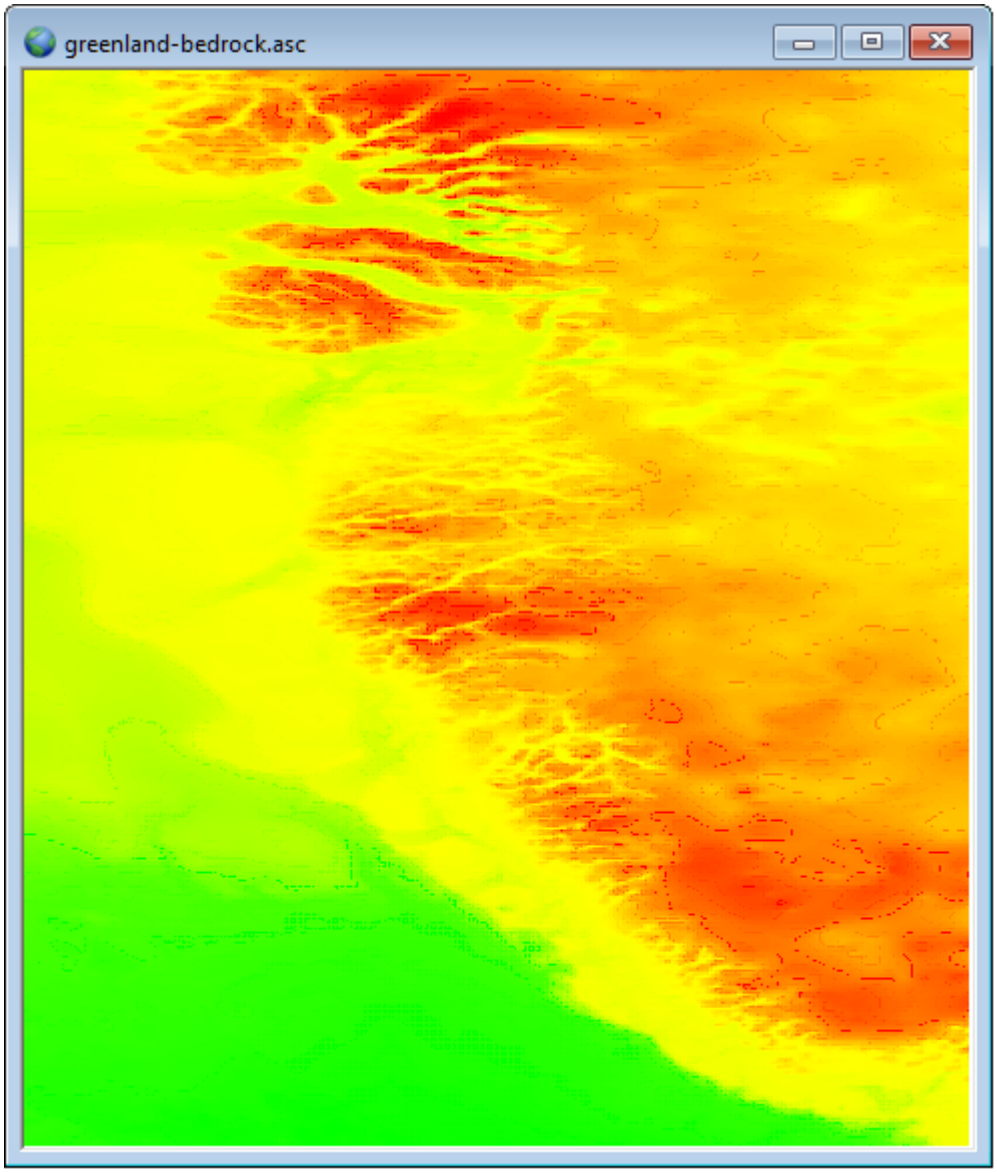
6.8.2 Map files

GeNIe supports only one map file format at the moment: ASCII (or text) raster format. While we are planning to extend the set of map file formats that are compatible with GeNIe, we understand that this is not a crucial limitation, as most GIS programs allow for converting maps to the ASCII raster format. Rather than replicating the capabilities of GIS programs, we support this simple format.

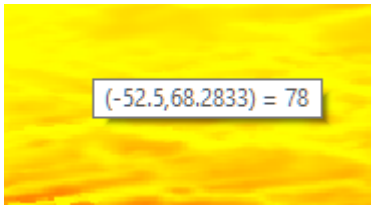
To open a map file in GeNIe, please select *Open GIS File...* from the *File Menu*:



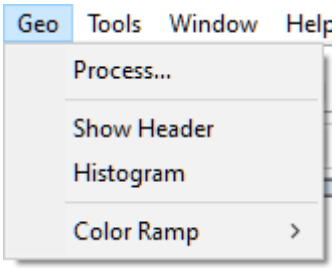
A GIS (map) file, when open, looks as follows:



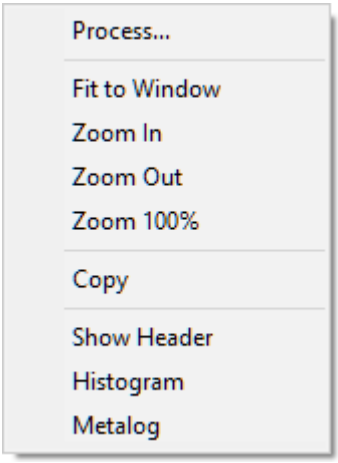
Hovering the mouse over the map shows the coordinates of the cell and the value corresponding to that cell. Please note that the text format (.asc) stores just one numerical value for each of the raster cells.



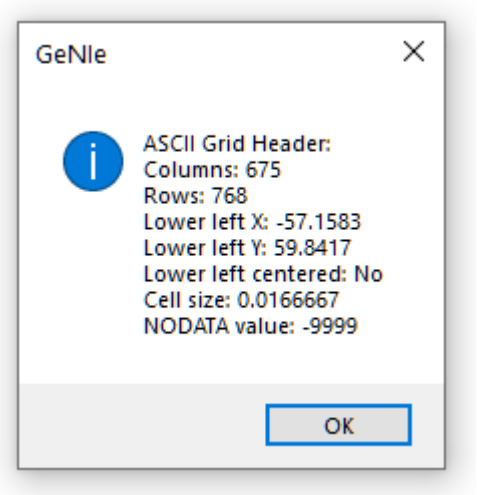
The *Geo Menu* allows for displaying the header of the map file and also a histogram of the values that the map contains



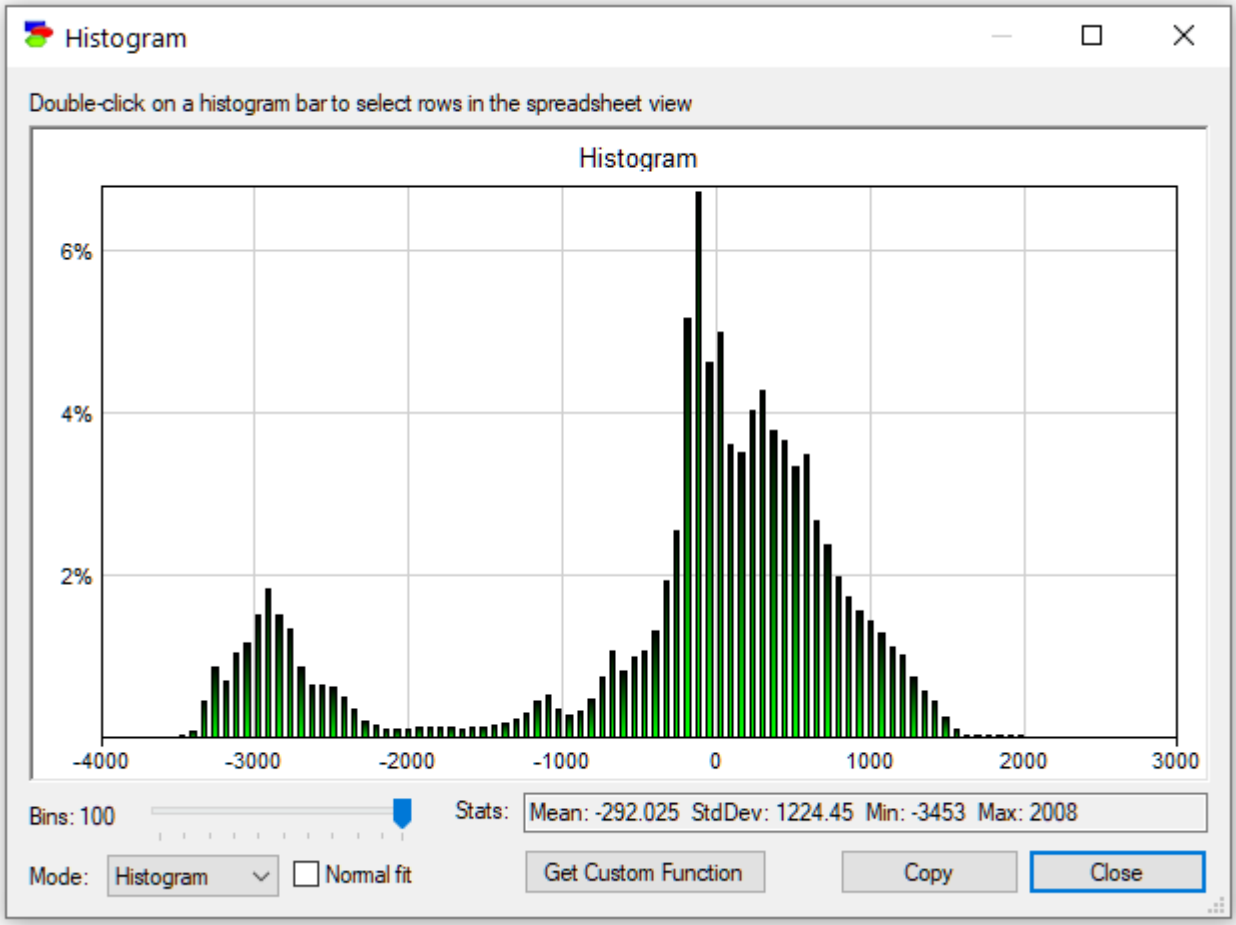
These choices are also available through the map context menu



Show header summarizes the map parameters, notably its size (in terms of the number of columns and rows in the raster file), cell size, map origin in terms of Earth coordinates of its lower-left corner, and value that denotes missing values.

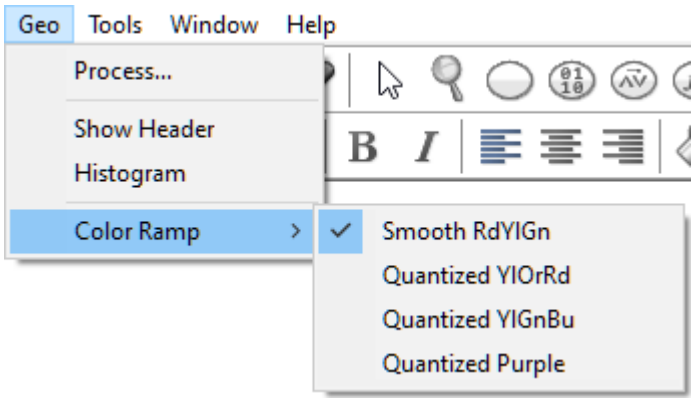


Histogram gives an idea of the occurrence of different values in the map.

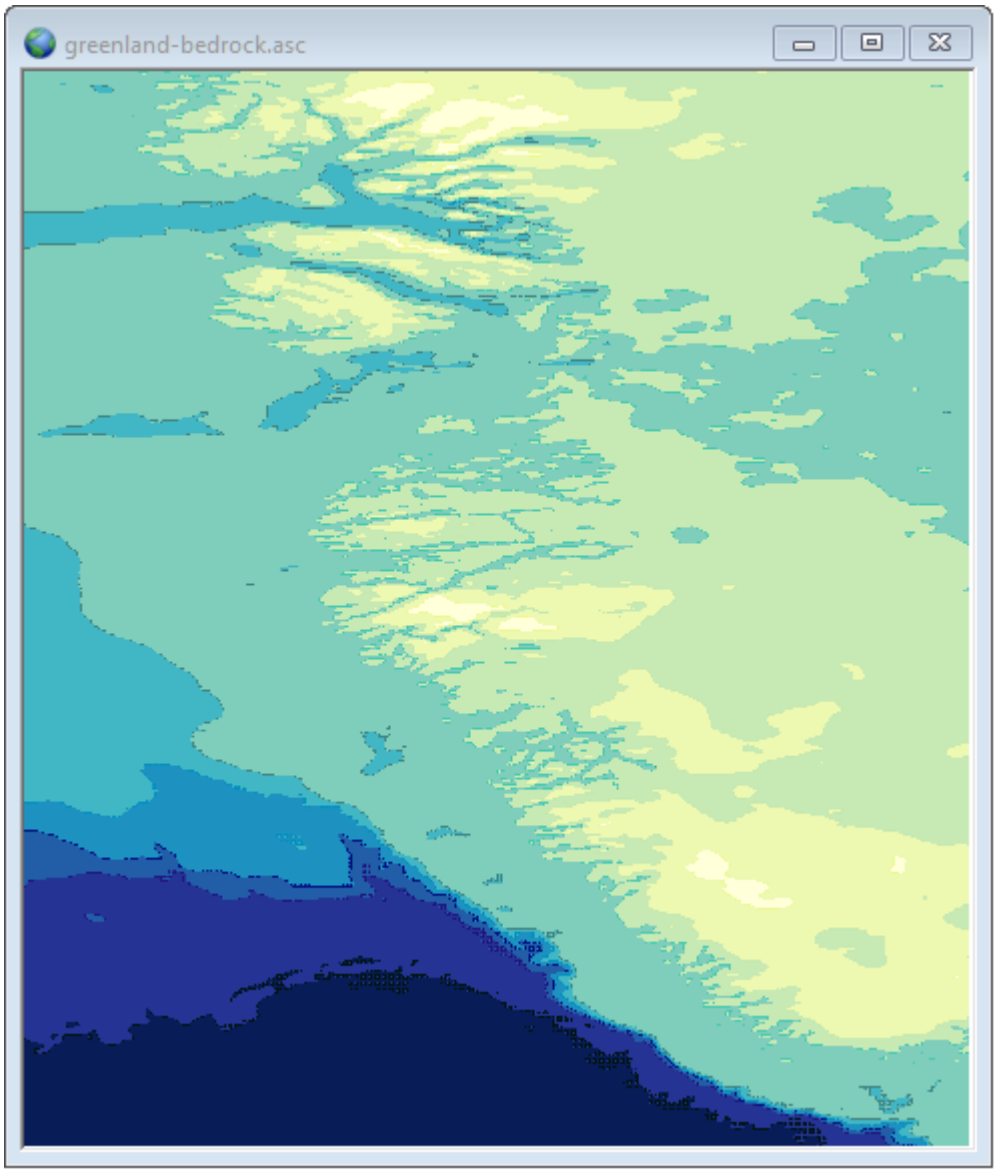


Metalog opens the *Fitting Metalog distribution to data* dialog that allows for fitting a metalog distribution to the values present in the map (in other words, fitting a metalog distribution to the histogram shown by the *Histogram* command applied to the map).

While GeNIe does not attempt to replicate the capabilities of GIS software, it is possible to modify the color ramp of the map file. Top that effect, please choose one of the ramps listed in the *Geo Menu*.



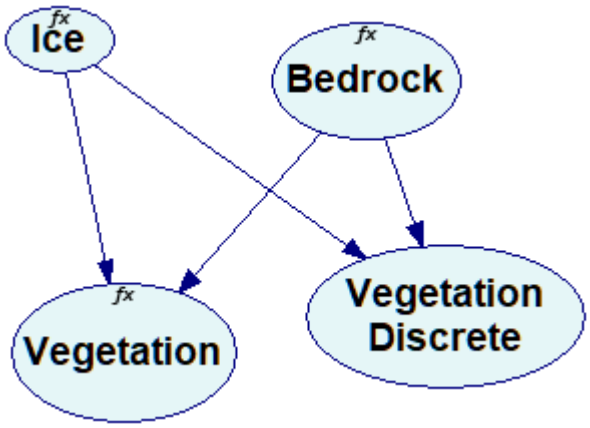
The greenland-bedrock.asc map viewed in the Quantized YIGnBu color ramp, looks as follows:



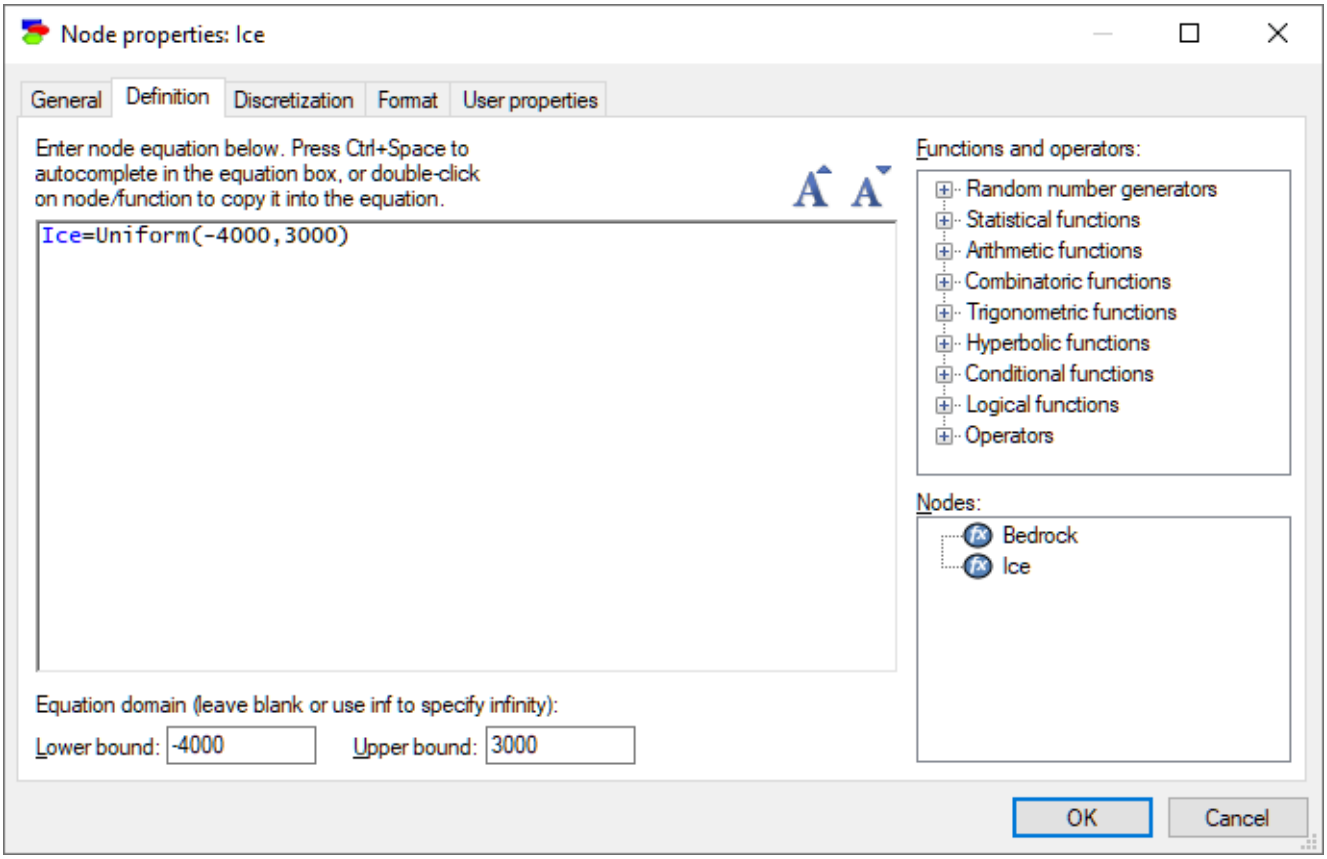
We will show how to process maps in the next section.

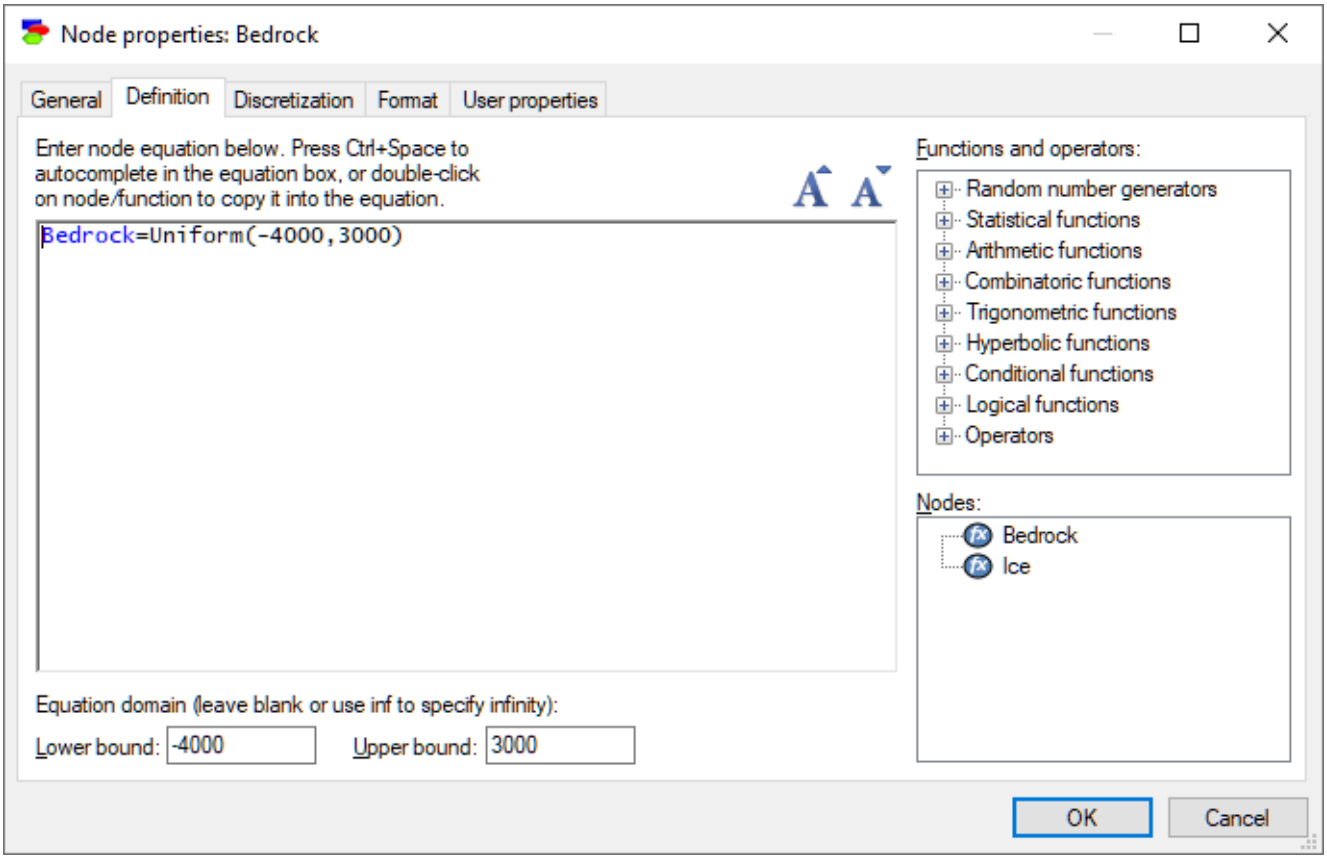
6.8.3 Map processing

We will use the following simple model that demonstrates the geo-processing capability of GeNIe:

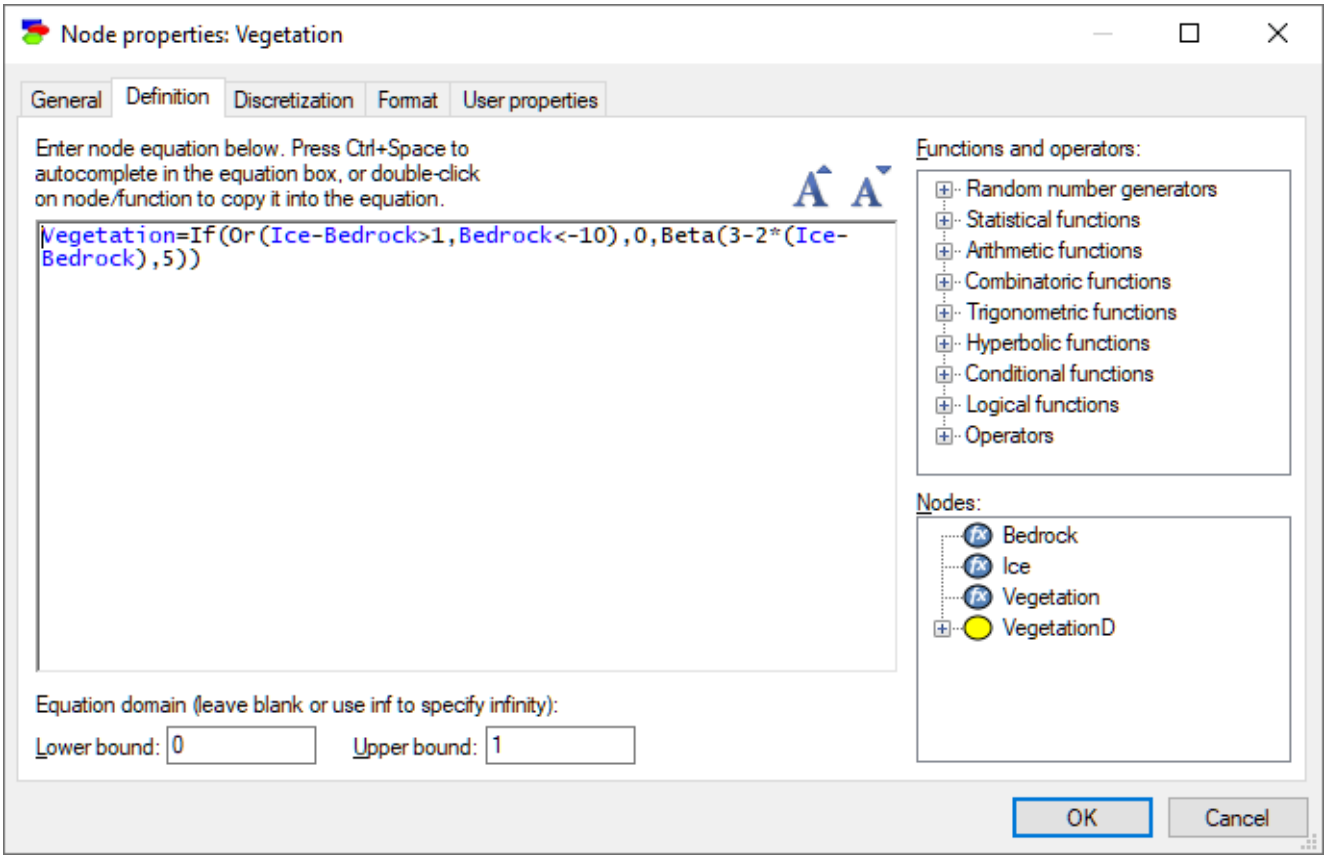


The model contains two variables that will take their values from maps (*Ice* and *Bedrock*) and two variables that will serve as values for the output maps (*Vegetation* and *Vegetation Discrete*). All variables, except for *Vegetation Discrete* are continuous. The definitions of the variables *Ice* and *Bedrock* are



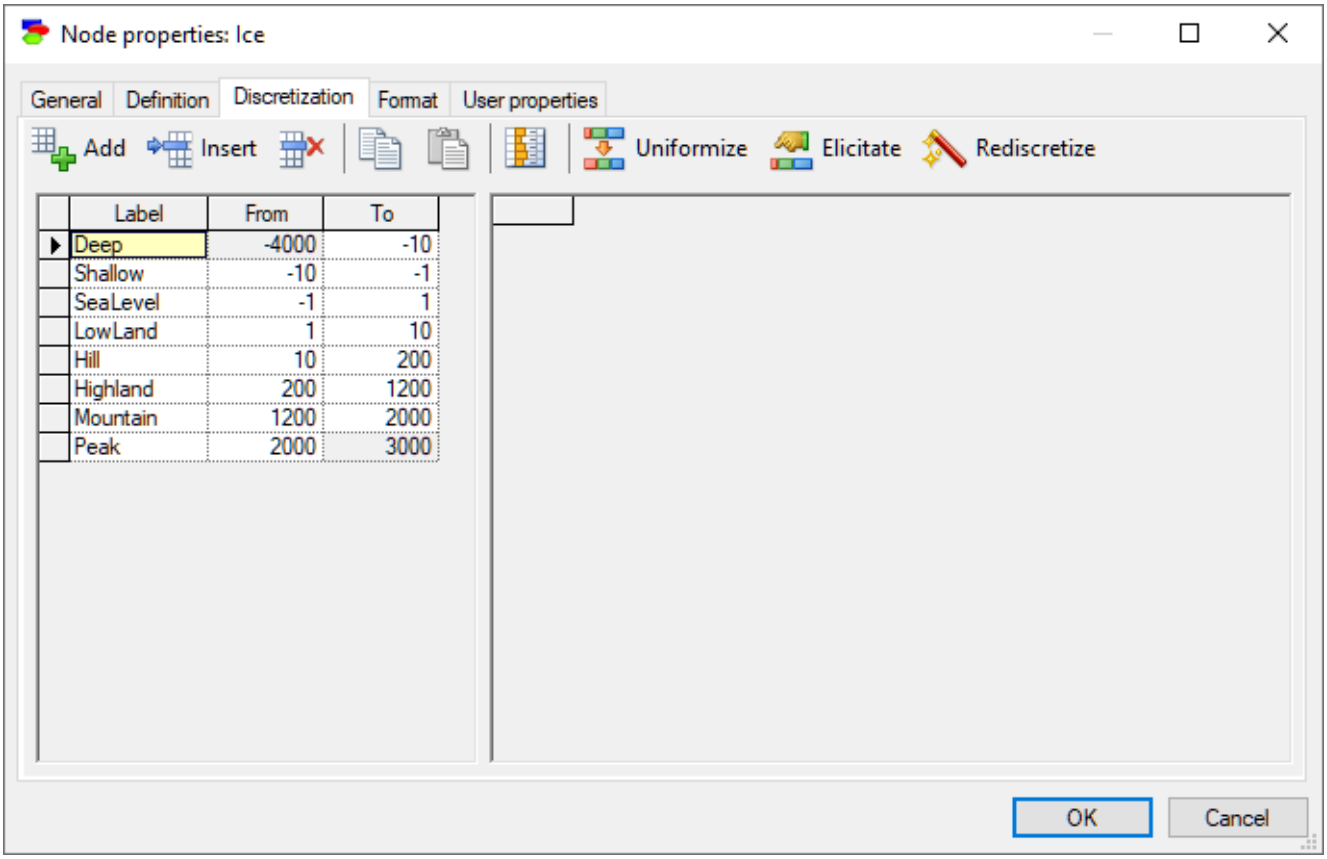


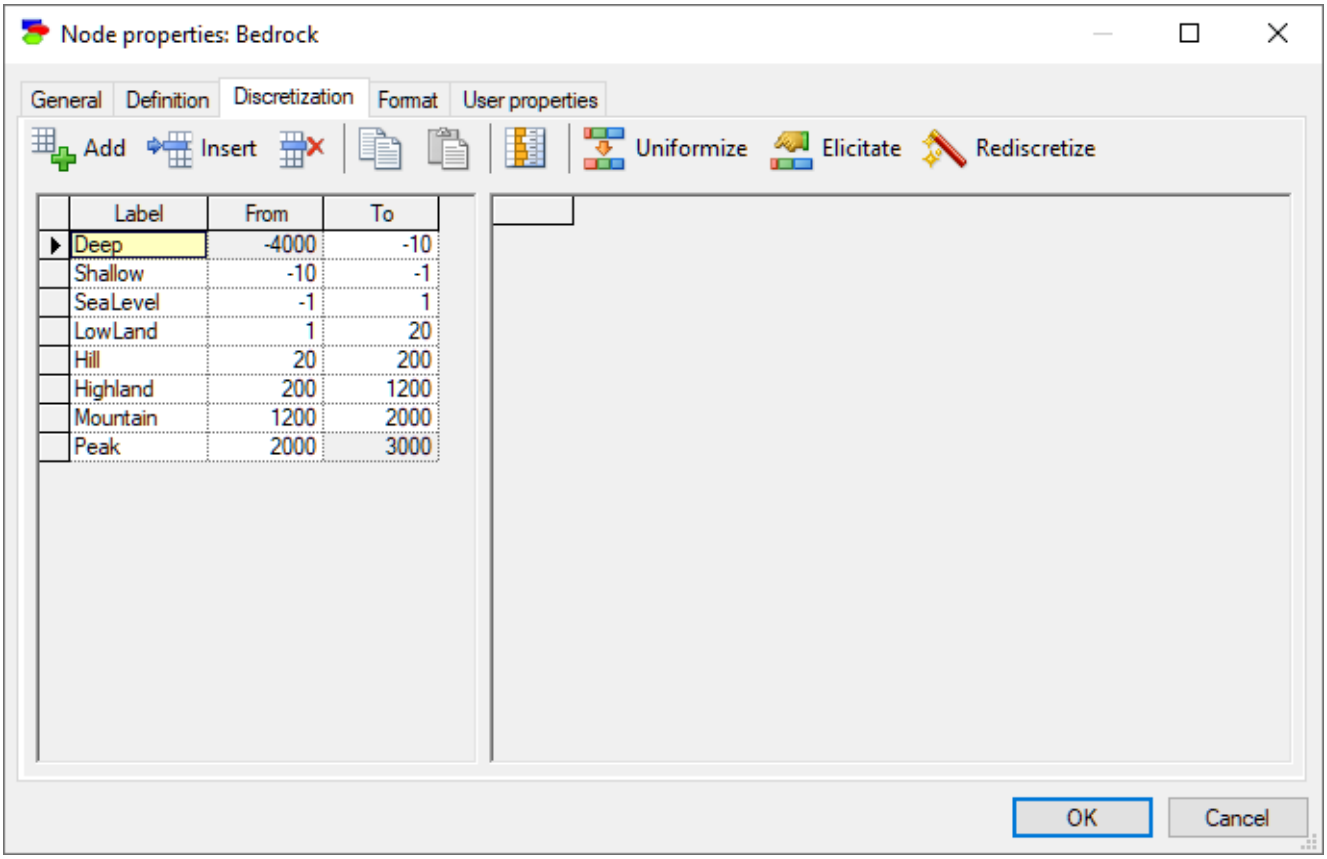
The definition of the variable *Vegetation* is as follows



Essentially, perhaps somewhat naively, we have defined the probability of vegetation as a Beta distribution with the first parameter dependent on the difference between the level of ice and the bedrock, which amounts to the thickness of ice in that location. In all those areas that are more than 10 meters below the sea level and those areas in which the thickness of the ice cover is larger than 1 meter, the probability of vegetation is defined as zero.

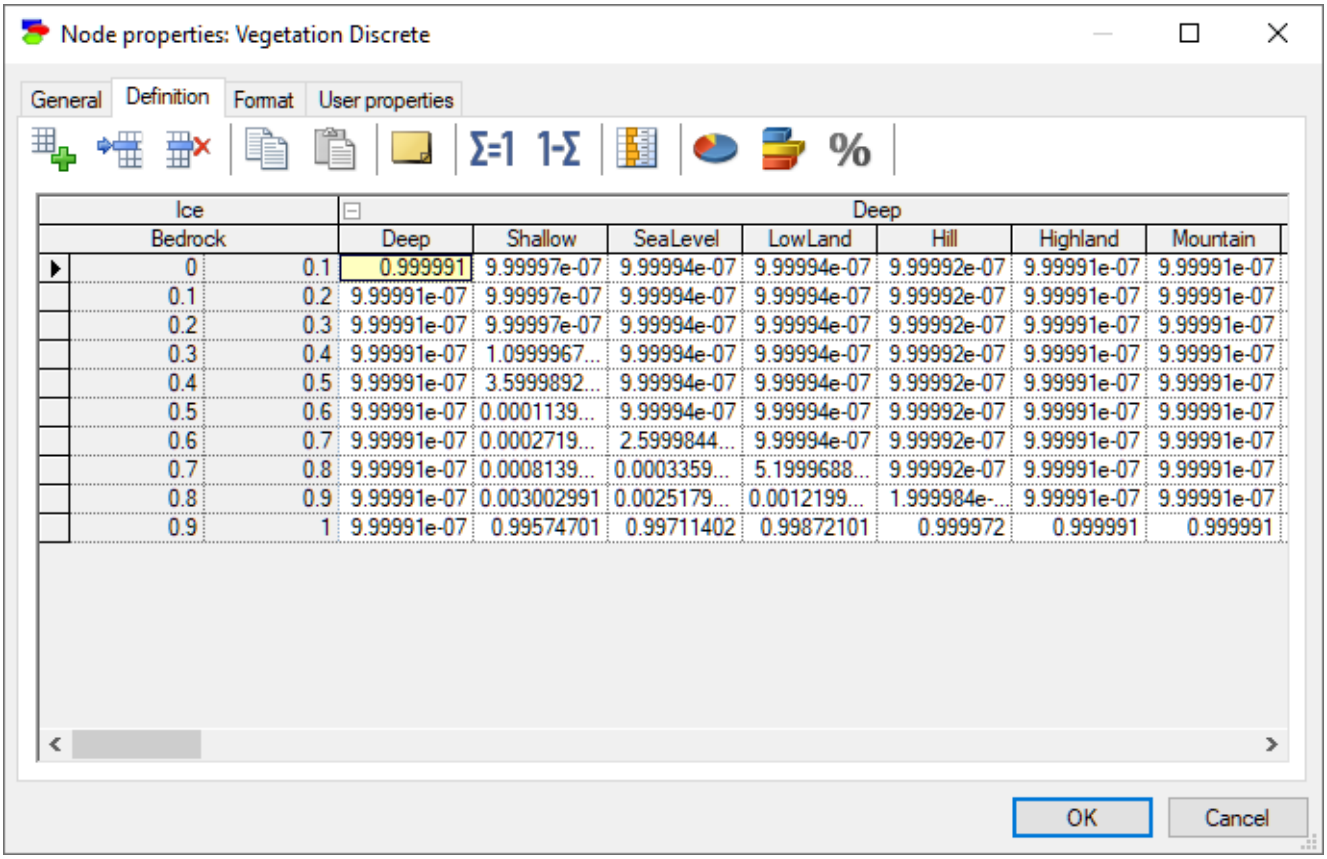
The definition of the variable *Vegetation Discrete* is a conditional probability table derived from the definition of the variable *Vegetation*. For this purpose, we have discretized the variables *Ice* and *Bedrock* as follows


Node properties: Ice
General
Definition
Discretization
Format
User properties
Add
Insert
Uniformize
Elicitate
Rediscretize
Label
From
To
Deep -4000 -10
Shallow -10 -1
SeaLevel -1 1
LowLand 1 10
Hill 10 200
Highland 200 1200
Mountain 1200 2000
Peak 2000 3000
OK
Cancel



Label	From	To
Deep	-4000	-10
Shallow	-10	-1
SeaLevel	-1	1
LowLand	1	20
Hill	20	200
Highland	200	1200
Mountain	1200	2000
Peak	2000	3000

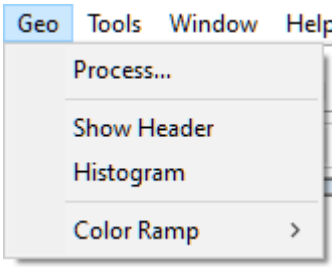
The resulting CPT of the variable *Vegetation Discrete* is as follows



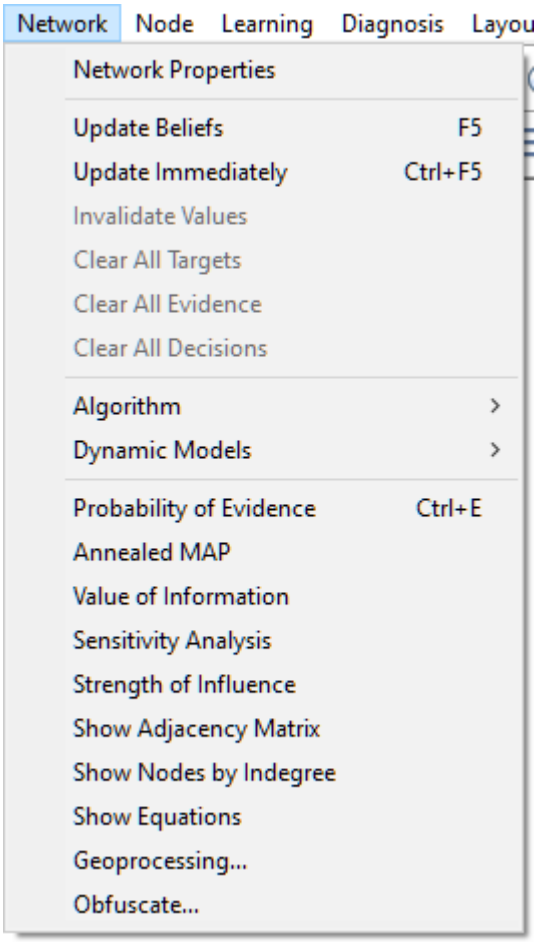
Ice		Deep						
Bedrock		Deep	Shallow	SeaLevel	LowLand	Hill	Highland	Mountain
0	0.1	0.999991	9.99997e-07	9.99994e-07	9.99994e-07	9.99992e-07	9.99991e-07	9.99991e-07
0.1	0.2	9.99991e-07	9.99997e-07	9.99994e-07	9.99994e-07	9.99992e-07	9.99991e-07	9.99991e-07
0.2	0.3	9.99991e-07	9.99997e-07	9.99994e-07	9.99994e-07	9.99992e-07	9.99991e-07	9.99991e-07
0.3	0.4	9.99991e-07	1.0999967...	9.99994e-07	9.99994e-07	9.99992e-07	9.99991e-07	9.99991e-07
0.4	0.5	9.99991e-07	3.5999892...	9.99994e-07	9.99994e-07	9.99992e-07	9.99991e-07	9.99991e-07
0.5	0.6	9.99991e-07	0.0001139...	9.99994e-07	9.99994e-07	9.99992e-07	9.99991e-07	9.99991e-07
0.6	0.7	9.99991e-07	0.0002719...	2.5999844...	9.99994e-07	9.99992e-07	9.99991e-07	9.99991e-07
0.7	0.8	9.99991e-07	0.0008139...	0.0003359...	5.1999688...	9.99992e-07	9.99991e-07	9.99991e-07
0.8	0.9	9.99991e-07	0.003002991	0.0025179...	0.0012199...	1.999984e-...	9.99991e-07	9.99991e-07
0.9	1	9.99991e-07	0.99574701	0.99711402	0.99872101	0.999972	0.999991	0.999991

The two variables *Vegetation* and *Vegetation Discrete* are almost identical, with the variable *Vegetation Discrete* being a discrete version of the variable *Vegetation*.

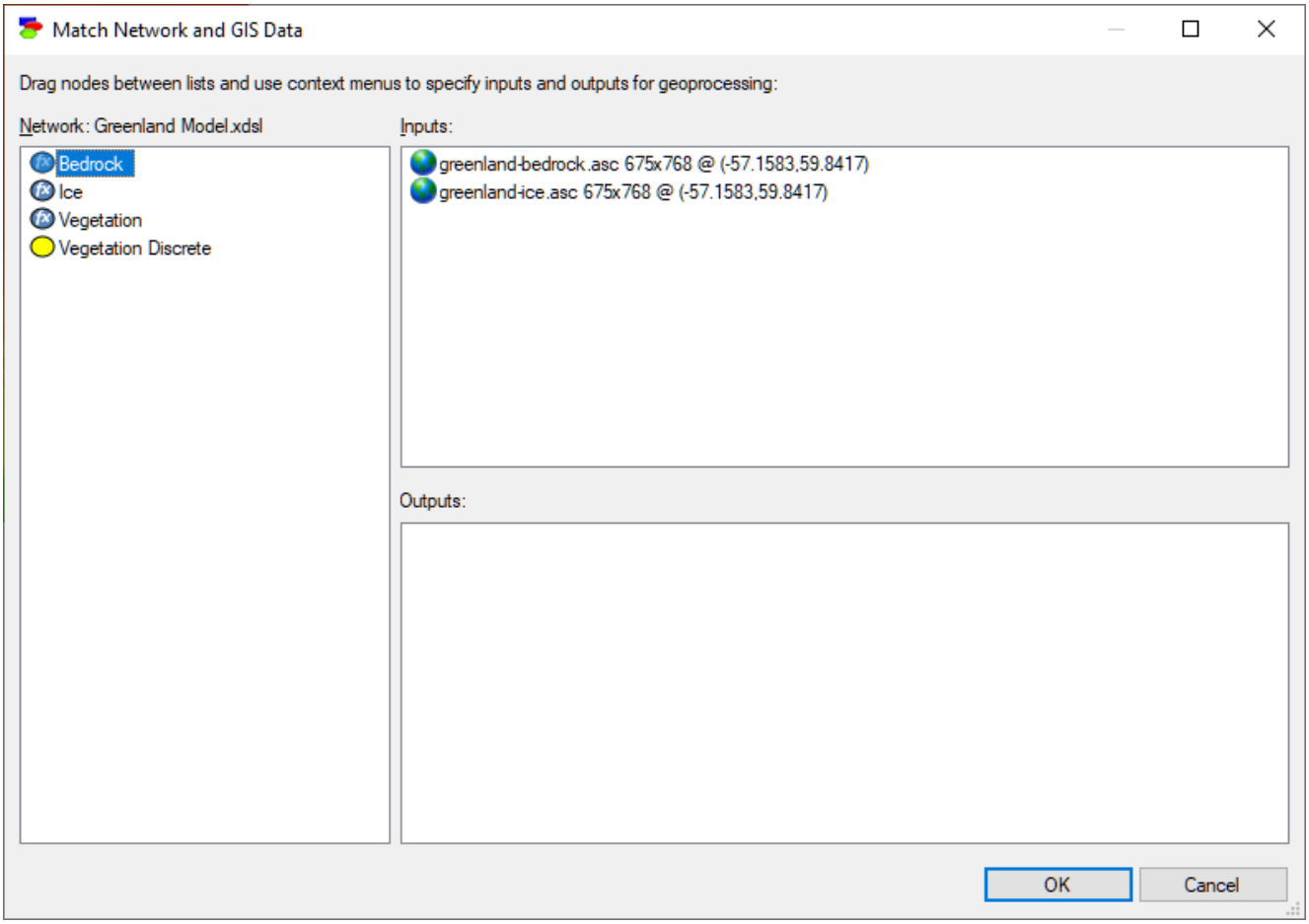
Once we have opened all input map files and a model file (the model describes the interaction between the data in the input maps and the data in the output maps), we can invoke the dialog that allows us to describe the map processing. To that effect, we select *Process...* from the *Geo Menu* (or from the map context menu):



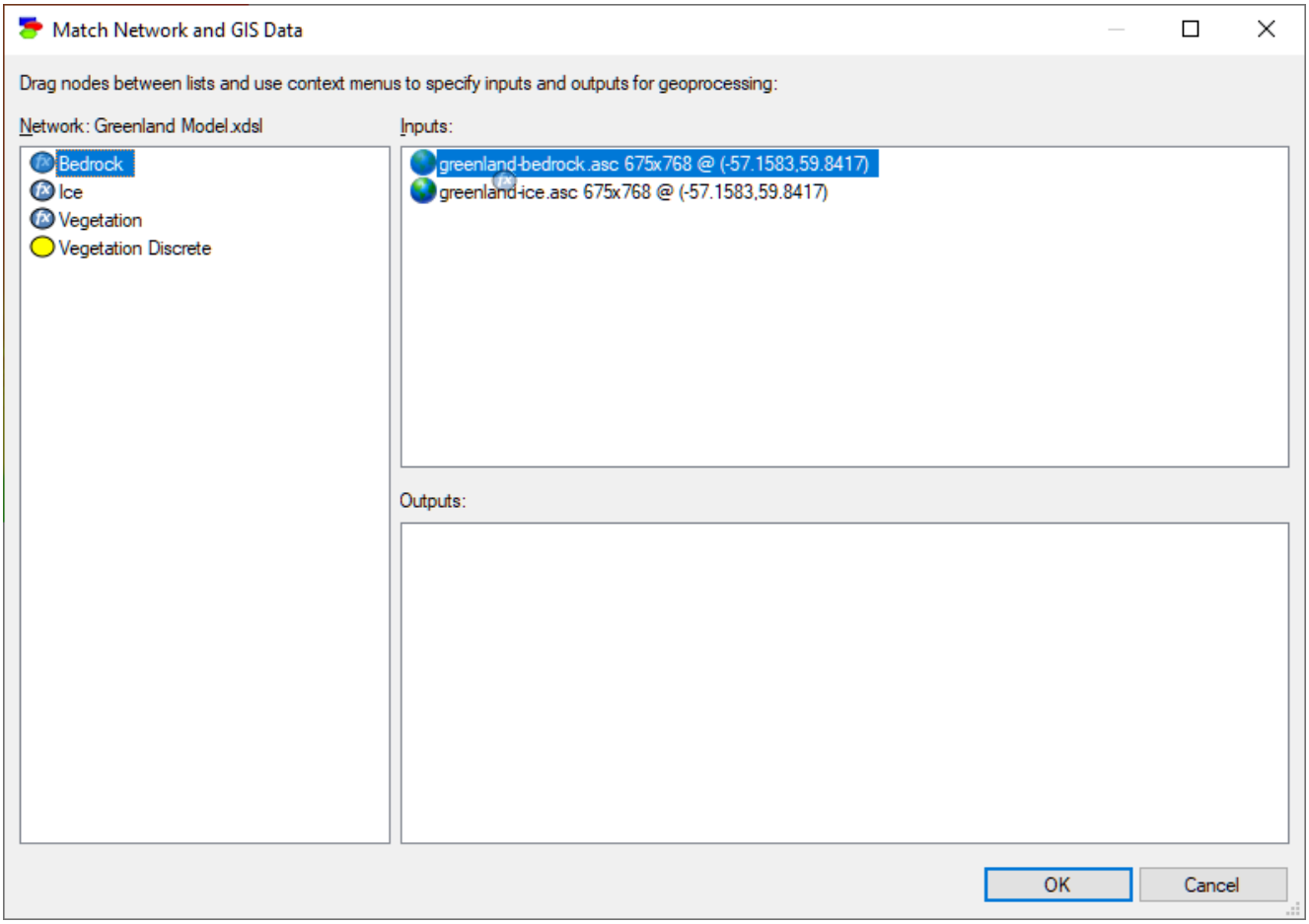
Another way of invoking the geo-processing is through the *Geoprocessing...* in the *Network Menu*.



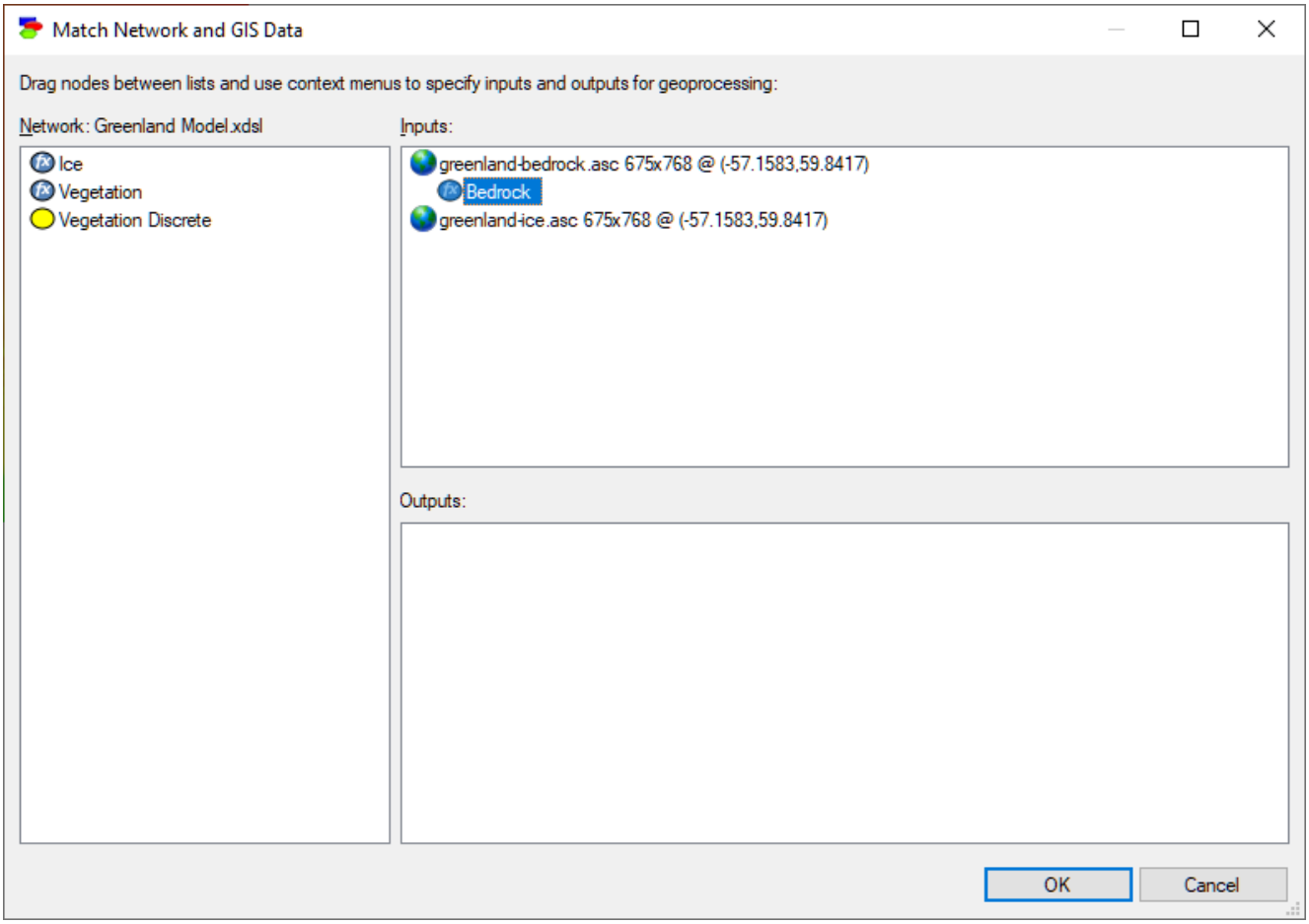
This invokes the geo-processing dialog, which essentially allows us to tie the input maps with a Bayesian network model and specify the output maps.



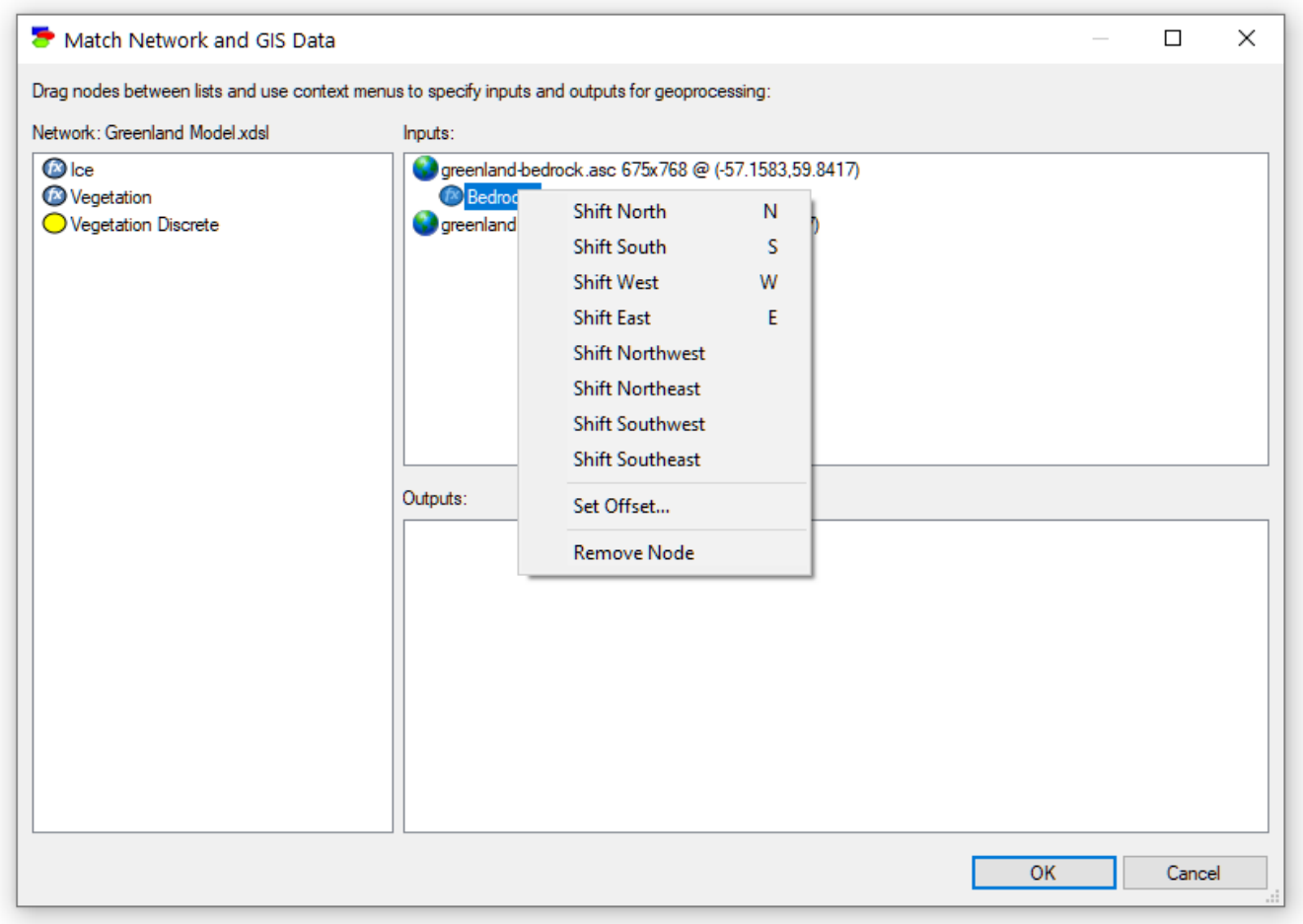
The dialog shows all maps that have been loaded into GeNIe (in this case, `greenland-ice.asc` and `greenland-bedrock.asc`) and all variables in the model chosen for geo-processing (in this case, *Bedrock*, *Ice*, *Vegetation* and *Vegetation Discrete*). To associate variables with input maps, we drag the variable name from the left-hand side pane to the upper-right-hand side pane and drop it in the corresponding map name. We drag and drop the variable *Bedrock* into the map `greenland-bedrock.asc`



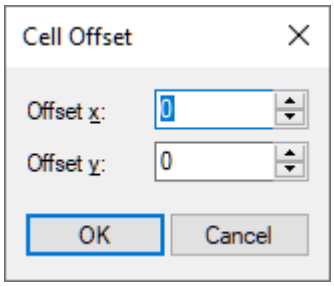
After the drop, the dialog shows an association between the variable and the map



While the meaning of this association is that the same raster cell in each of the maps is tied to the model variable, GeNIe allows also for associating model variables with neighboring map elements. This has application in all those situations in which the neighborhood of the current cell matters, for example in recognizing patterns in the maps. This way, the same map can have more than one variable associated with it, each variable associated to a different raster cell. The default association is the current cell. This can be changed by right-clicking on a variable associated with a map. For example, to change the offset of the variable *Bedrock*, please right-click on it and choose the offset (relative to the position of the current cell).

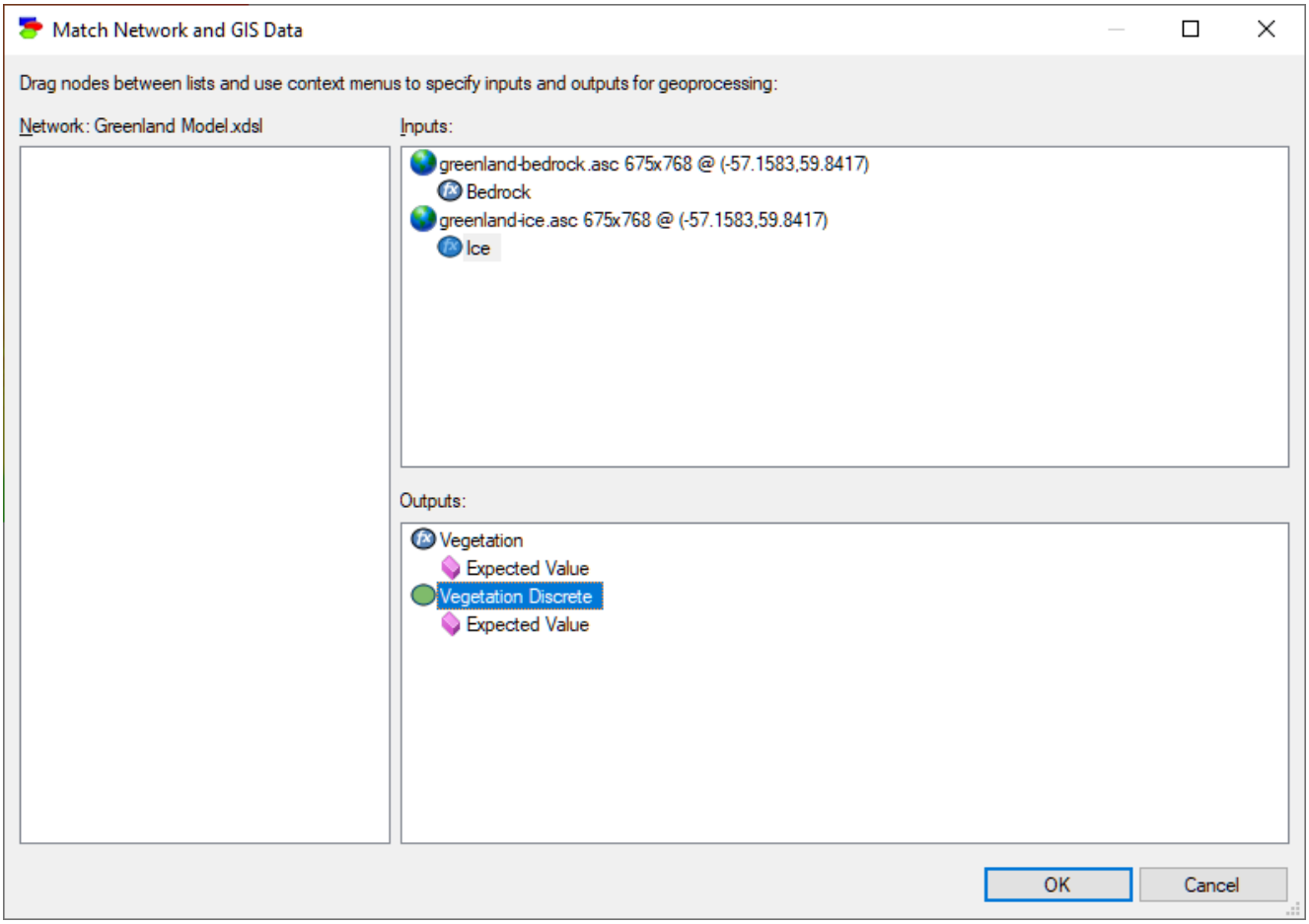


Set Offset... is the most general choice that allows for entering any offset in the x and y directions:

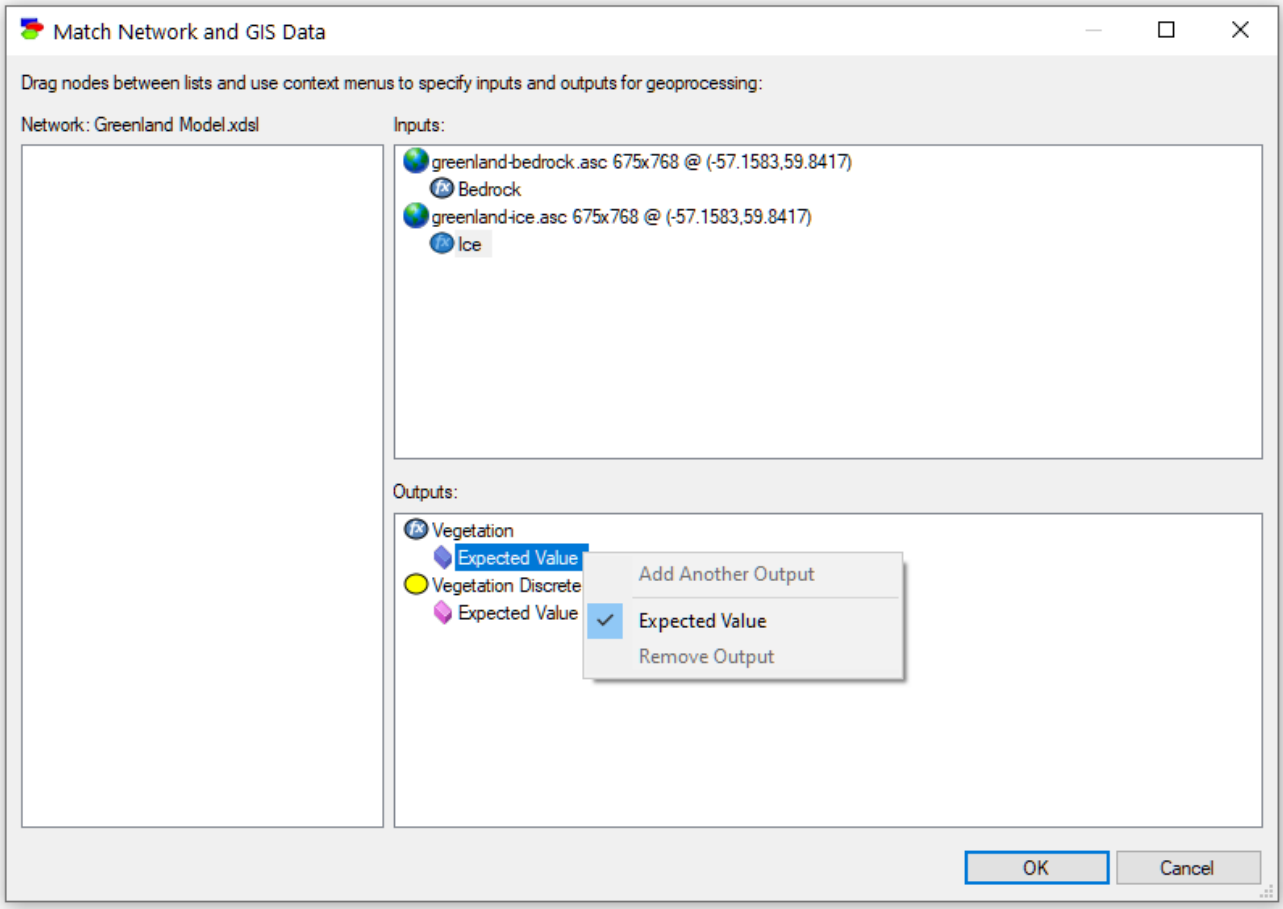


Offset (0,0) corresponds to the current cell, offset (0,1) corresponds to the cell above it, (-1,-1) to the cell to the left and below it, etc. Choices *Shift North*, *Shift South*, etc., are shortcuts that refer to the cells in the immediate neighborhood of the current cell. *North*, *South*, *West*, etc., describe a relative position with respect to the current cell. For example, *North* (*N*) is the cell above it (0,1), *West* (*W*) is the cell to the left of it (-1,0), *Northeast* is the cell up and to the right of the current cell. Since our simple example does not include any variables that are outside of the current cell, we leave the offset at (0,0).

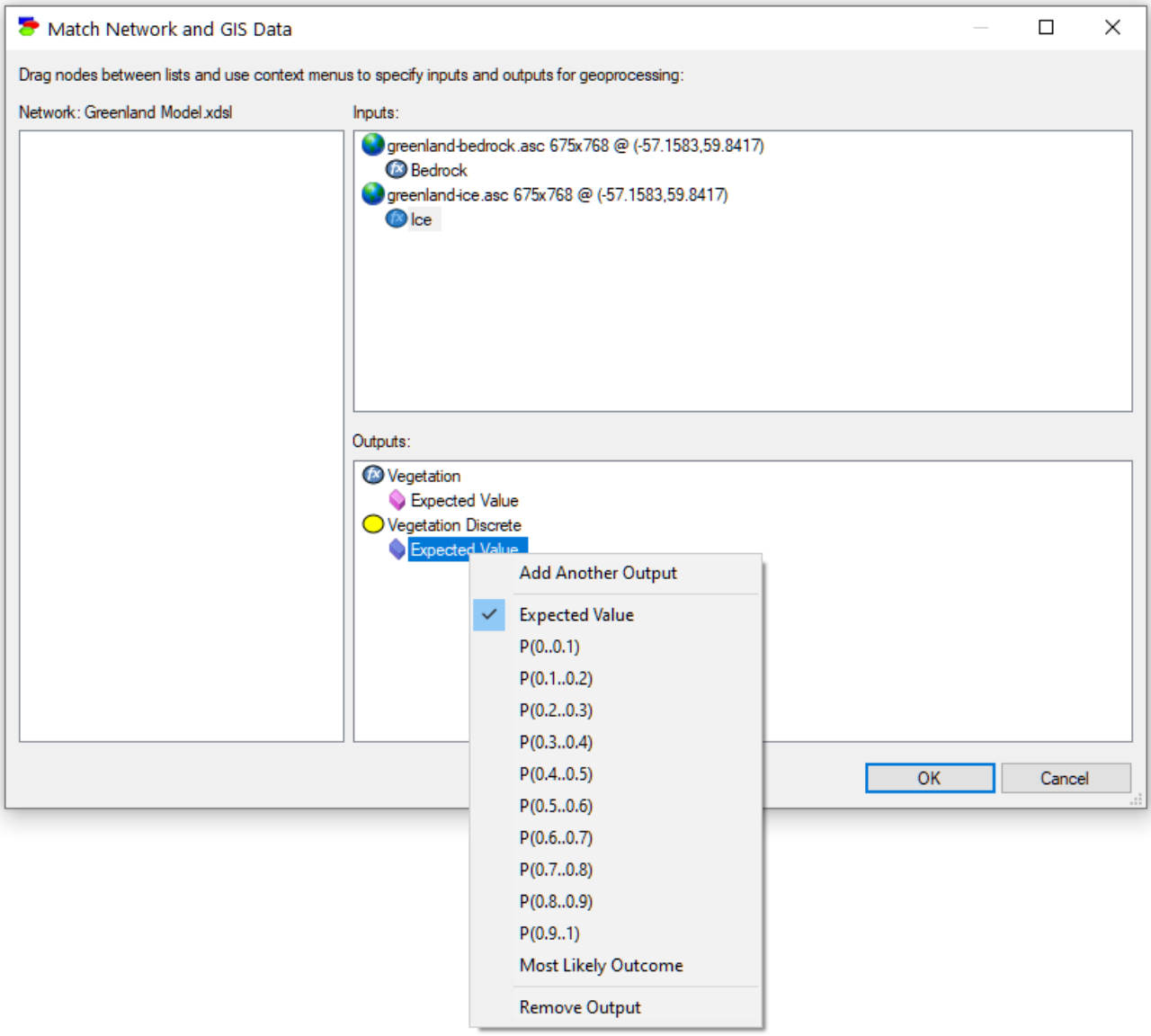
We continue by associating the variable *Ice* with the map `greenland-ice.asc` (through dragging and dropping, as was the case for variable *Bedrock*) and dropping the variables *Vegetation* and *Vegetation Discrete* into the (lower right) *Outputs* pane. The resulting dialog looks as follows



The dialog tells us that when processing the maps, `greenland-bedrock.asc` will provide input for the variable *Bedrock*, `greenland-ice.asc` will provide input to variable *Ice*, and there will be two output maps, *Vegetation* and *Vegetation Discrete*. One final step is deciding what values precisely should make it to the outputs maps. When the output variable is numerical (whether it is continuous or discrete), the default output is the expected value of the variable. In case of continuous variables, this is the only choice, as shown by right-clicking on the on the *Expected Value* text.

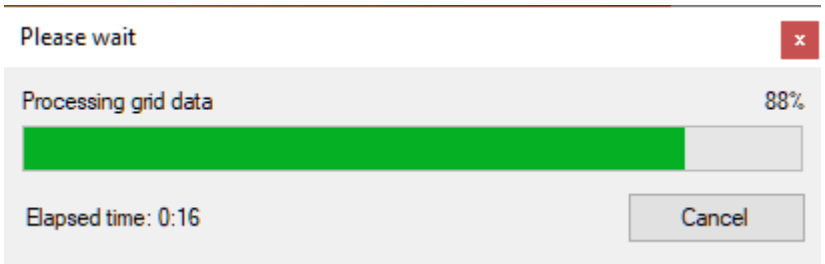


In case of discrete variables (whether numerical or categorical), there are additional choices:



In addition to expected value (numerical variables only), we can select the probability of any of the discrete states or the most likely outcome.

We will leave the output type at the *Expected Value* for both maps. Clicking OK starts the processing



which end with two additional maps shown in GeNIe: Greenland Model EV(Vegetation)

and Greenland Model EV(VegetationD)

The two maps are similar, although one can see the impact of discretization on the Greenland Model EV(VegetationD) map. Both maps show for each raster cell the probability of vegetation, as defined in the Bayesian network model. We can see that generally only coastal areas of the map have non-zero probability of vegetation. The maps can be analyzed and/or saved for further processing.

This page is intentionally left blank.

Resources

7 Resources

7.1 Books

A good source of elementary knowledge of Bayesian networks is the path-breaking book by Pearl (1988). A good, thorough synthesis of the theoretical aspects of Bayesian networks and probabilistic decision support systems, useful for readers interested in building a reasoning system from the ground up, is the book by Neapolitan (1990), regretfully out of print. Jensen's (1996) book is heartily recommended as a good source of knowledge for both builders and users of Bayesian reasoning systems. The most recent addition to the books on graphical modeling is Cowell, Dawid, Lauritzen & Spiegelhalter (1999), a heartily recommended reading for anybody who wants to develop a thorough understanding of the methods that are at the foundation of graphical probabilistic systems.

de Finetti (1970) and Savage (1954) are recommended for the foundations of Bayesian probability theory and decision theory.

Readers interested in practical aspects of decision support, especially in the context of policy making, are recommended the superb book by Morgan & Henrion (1989).

Russel an Norvig (1995) is a good textbook covering application of probabilistic methods in Artificial Intelligence.

For readers interested in graphical probabilistic models in general, the thorough book by Whittaker (1989), covering directed and undirected probabilistic graphs, may be of interest.

An up to date list of textbooks covering the field of decision analysis can be found on BayesFusion's web site.

7.2 Research papers

While there a good number of excellent papers covering the topic of decision-analytic decision support, here are some of our favorites.

An introductory paper on Bayesian networks, useful for beginners is (Charniak, 1991).

Overview papers by Horvitz et al. (1988), Cooper (1989), Henrion et al. (1991), Spiegelhalter et al. (1993) and Matzkevich & Abramson (1995) are accessible introductions to the use of probabilistic and decision-analytic methods in decision support systems.

Users interested in practical applications of Bayesian networks are directed to the March 1995 special issue of the *Communications of the ACM* journal, edited by Heckerman, Mamdani and Wellman (1995).

(Howard, 1984) is a good introduction to influence diagrams, the book in which this paper has been published, is a good collection of reading on decision analysis.

(Henrion, 1988) is a manifesto arguing convincingly for the use of probabilistic methods in artificial intelligence.

(Henrion, 1989) is a practical introduction to problems related to building probabilistic models. Another place to look at is a special issue of the *IEEE Transaction of Knowledge and Data Engineering* journal on building probabilistic models (Druzdzel & van der Gaag, 2000).

Foundations of conditional independence on which graphical models are built are outlined in (Dawid 1979).

The principles of relevance reasoning are outlined in (Druzdzel & Suermondt, 1994).

7.3 Conferences

Probably the best source for the state of the art research in graphical probabilistic models are proceedings of the annual *Conference on Uncertainty in Artificial Intelligence* (*UAI*). Proceedings of UAI conferences are available from the web site of the Association for Uncertainty in Artificial Intelligence

http://www.auai.org/

A similarly prestigious conference on the topic of probabilistic graphical models is the bi-annual European *Conference on Probabilistic Graphical Models* (*PGM*). It brings together researchers interested in all aspects of graphical models for probabilistic reasoning, decision making, and learning. The web site for the 2018 conference, listing web sites for all previous PGM conference and their on-line proceedings, is at the following location:

http://pgm2018.utia.cz/

Another conference devoted completely to Bayesian networks is the Annual Conference of the Australasian Bayesian Network Modelling Society (ABNMS). The web site for the 2019 conference, listing web sites for all previous ABNMS conference is at the following location:

http://abnms.org/conferences/abnms2019/

Other relevant conferences are *AAAI National Conferences on Artificial Intelligence* (AAAI), *International Joint Conferences on Artificial Intelligence* (IJCAI), Conferences on *Knowledge Discovery and Data Mining* (KDD), *Workshop on Artificial Intelligence and Statistics, Uncertainty Track of the Florida Artificial Intelligence Conferences* (FLAIRS), *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty* (ECSQARU), *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (IPMU), and *European Conference on Artificial Intelligence* (ECAI).

7.4 Model repositories

An important class of World Wide Web resources that may be of interest to GeNIe users are model repositories. They are a great inspiration to model builders and source of models for the purpose of testing algorithms. Here is where you can find BayesFusion's model repository:

https://repo.bayesfusion.com/

The repository is interactive and allows for entering evidence and performing Bayesian updating. Furthermore, you should be able to download any of the models included in the repository to a local disk or open it directly from GeNIe. BayesFusion's interactive model repository is powered by BayesBox, another BayesFusion's product, allowing our clients to create local and secure model repositories.

7.5 Social Media

Please visit BayesFusion's YouTube channel for useful recordings that may help you in learning more about decision-theoretic methods in intelligent systems and GeNIe.

We use BayesFusion's Twitter account to communicate with our users important news, such as new software releases. Following us on Twitter will ensure that you are up to date on what is happening at BayesFusion.

BayesFusion's Facebook account contains important news releases. We make sure that important news that are listed on our web site find their way to our Facebook page.

BayesFusion's LinkedIn account contains basic information about BayesFusion. We have started a LinkedIn group *GeNIe Users* that is an open forum for our uses.

Finally, *BayesFusion Forum* is a good place to interact with us and other GeNIe and SMILE users.

7.6 References

Bayes, Rev. Thomas (1702-1761). *An essay toward solving a problem in the doctrine of chances* (reprint). Biometrica, 45(3-4):293-315.

Castillo, E.F., J.M. Gutierrez, and A.S. Hadi (1997). *Sensitivity analysis in discrete Bayesian networks*, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 27, no. 4, pp. 412–423.

Charniak, Eugene (1991). *Bayesian networks without tears*. AI Magazine, 12(4):50-63.

Cheng, Jian & Marek J. Druzdzel (2000). *AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks*. Journal of Artificial Intelligence Research (JAIR), 13:155-188.

Cheng, Jie, David A. Bell & Weiru Liu (1997). *An Algorithm for Bayesian Belief Network Construction from Data*. Proceedings of AI & Statistics, pages 83-90.

Robert T. Clemen (1996). *Making Hard Decisions: An Introduction to Decision Analysis*, Second Edition. Duxbury Press

Cooper, Gregory F. (1988). *A method for using belief networks as influence diagrams*. Proceedings of the Workshop on Uncertainty in Artificial Intelligence, Minneapolis, Minnesota, 55-63.

Cooper, Gregory F. (1989). *Current research directions in the development of expert systems based on belief networks*. Applied Stochastic Models and Data Analysis, 5(1):39-52.

Cooper, Gregory F. (1990). *The computational complexity of probabilistic inference using Bayesian belief networks*. Artifical Intelligence, 42(2-3):393-405.

Cooper, Gregory F. & Edward Herskovits (1992). *A Bayesian method for the induction of probabilistic networks from data*, Machine Learning, 9(4):309-347.

Cowell, Robert G., A. Philip Dawid, Steffen L. Lauritzen & David J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag New York, Inc.: New York, NY.

Dagum, Paul & Michael Luby (1993). *Approximating probabilistic inference in Bayesian belief networks is NP-hard*. Artificial Intelligence, 60(1):141-153.

Dawid, A. Philip (1979). *Conditional independence in statistical theory*. Journal of the Royal Statistical Society, Series B (Methodological), 41:1-31.

Dawid, A. Philip (1992). *Applications of a general propagation algorithm for probabilistic expert systems*. Statistics and Computing, 2:25-36.

de Finetti, Bruno (1970). *Theory of Probability*. John Wiley and Sons, New York.

Dempster, A, N. Laird & D. Rubin (1977). *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society, Series B 39, 1-38.

Diez, F. Javier (1993). *Parameter adjustment in Bayes networks. The Generalized Noisy-OR gate*. In Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI-93), Morgan Kaufmann: San Mateo, CA, pages 99-105.

Diez, F. Javier & Marek J. Druzdzel (2006). *Canonical probabilistic models for knowledge engineering*. Technical Report CISIAD-06-01. UNED, Madrid, 2006 (available at http://www.ia.uned.es/~fjdiez/papers/canonical.html).

Druzdzel, Marek J. & Clark Glymour (1999). *Causal inferences from databases: Why universities lose students*. In Clark Glymour and Gregory F. Cooper (eds), Computation, Causation, and Discovery, Chapter 19, pages 521-539, AAAI Press, Menlo Park, CA.

Druzdzel, Marek J. & Henri J. Suermondt (1994). *Relevance in probabilistic models: "backyards" in a "small world."* In Working notes of the AAAI-1994 Fall Symposium Series: Relevance, New Orleans, LA, pages 60-63.

Druzdzel, Marek J. & Linda C. van der Gaag (2000). *Building probabilistic networks: `Where do the numbers come from?' Guest editors' introduction*. IEEE Transactions on Knowledge and Data Engineering, 12(4):481-486.

Eades, P. (1984). *A heuristic for graph drawing*. Congressus Numerantium, 41, page 149–160.

Friedman, Nir, Dan Geiger & Moises Goldszmidt (1997). *Bayesian network classifiers*. Machine Learning, 29, 131–163.

Fung, Robert & Kuo-Chu Chang (1990). *Weighting and integrating evidence for stochastic simulation in Bayesian networks*. In Henrion, M., Shachter, R.D., Kanal, L.N. & Lemmer, J.F. (eds.) Uncertainty in Artificial Intelligence 5. Elsevier Science Publishers B.V. (North Holland), pages 209-219.

Fung, Robert & Brendan del Favero (1994). *Backward simulation in Bayesian networks*. In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, pages 227-234.

Heckerman, David & John S. Breese (1996). *Causal Independence for Probability Assessment and Inference Using Bayesian Networks*. IEEE Transactions on Systems, Man, and Cybernetics, 26:826-831.

Heckerman, David, Dan Geiger & David M. Chickering (1995). *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data*. Machine Learning, 20, 197-243.

Heckerman, David, Abe Mamdani & Michael P. Wellman (1995). *Real-world applications of Bayesian networks*. Communications of the ACM, 38(3):24-26.

Heckerman, David & and Ross Shachter (1995). *Decision-theoretic foundations for causal reasoning*, Journal of Artificial Intelligence Research, 3:405-430.

Henrion, Max (1986). *Uncertainty in artificial intelligence: Is probability epistemologically and heuristically adequate?* In Jeryl Mumpower, Ortwin Renn, Lawrence D. Phillips & V.R.R. Uppuluri (eds.), Expert Judgment and Expert Systems, Proceedings of the NATO Advanced Research Workshop on Expert Judgment and Expert Systems, Porto, Portugal, Berlin, Germany: Springer Verlag, pages 105-129.

Henrion, Max (1988). *Propagating uncertainty in Bayesian networks by probabilistic logic sampling*. In Lemmer, J.F. and Kanal, L.N. (eds.) Uncertainty in Artificial Intelligence 2. Elsevier Science Publishers B.V. (North Holland), pages 149-163.

Henrion, Max (1989). *Some practical issues in constructing belief networks*. Kanal, L.N., Levitt, T.S. & Lemmer, J.F. (eds.), Uncertainty in Artificial Intelligence 3. Elsevier Science Publishers B.V. (North Holland), pages 161-173.

Henrion, Max (1990). *An introduction to algorithms for inference in belief nets*. In Henrion, M., Shachter, R.D., Kanal, L.N. & Lemmer, J.F. (eds.), Uncertainty in Artificial Intelligence 5, Elsevier Science Publishers B.V. (North Holland), pages 129-138.

Henrion, M., John S. Breese & Eric J. Horvitz (1991). *Decision analysis and expert systems*. AI Magazine, 12(4):64-91.

Horvitz, Eric J., John S. Breese & Max Henrion (1988). *Decision theory in expert systems and artificial intelligence*. International Journal of Approximate Reasoning. 2(3):247-302.

Howard, Ronald A. & James E. Matheson (1984). *Influence diagrams*. In Howard, R. and Matheson, J., editors, Readings on the Principles and Applications of Decision Analysis, volume II, pages 721-762. Strategic Decision Group, Menlo Park, CA.

Huang, Cecil & Adnan Darwiche (1996). *Inference in belief networks: A procedural guide*. International Journal of Approximate Reasoning, 15:225-263.

Hulst, Joris (2006). *Modeling physiological processes with dynamic Bayesian networks*. M.Sc. thesis, Delft University of Technology, Delft, The Netherlands.

Jensen, Finn V. (1996). *An Introduction to Bayesian Networks*. Springer Verlag, New York.

Jensen, Finn V., Steffen L. Lauritzen & Kristian G. Olsen (1990). *Bayesian updating in recursive graphical models by local computations*. Computational Statisticals Quarterly, 4:269-282.

Kayaalp, Mehmet & Gregory F. Cooper (2002). A Bayesian Network Scoring Metric That Is Based on Globally Uniform Parameter Priors. In Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02), Morgan Kaufmann: San Mateo, CA, pages 251-158.

Kernighan, Brian W. & Dennis M. Ritchie (1988). *The C Programming Language*. Prentice Hall PTR, 2nd edition.

Kjærulff, Uffe & Linda C. van der Gaag (2000). *Making Sensitivity Analysis Computationally Efficient*. Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI 2000), pages 317-325.

Koiter, Joost R. (2006). *Visualizing Inference in Bayesian Networks*. M.Sc. thesis, Faculty of Electrical Engineering, Mathematics, and Computer Science, Department of Man-Machine Interaction, Delft University of Technology.

Kozlov, Alexander V. & Singh, Jaswinder Pal (1996). *Parallel Implementations of Probabilistic Inference*. IEEE Computer, pages 33-40.

Lauritzen, Steffen L. & David J. Spiegelhalter (1988). *Local computations with probabilities on graphical structures and their application to expert systems* (with discussion). Journal of the Royal Statistical Society, Series B (Methological), 50(2):157-224.

Lauritzen, S.L. (1995). *The EM algorithm for graphical association models with missing data*. Computational Statistics and Data Analysis 19. 191-201.

Lin, Yan & Marek J. Druzdzel (1997). *Computational advantages of relevance reasoning in Bayesian belief networks*. In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97), Morgan Kaufmann: San Mateo, CA, pages 342-350.

Lin, Yan & Marek J. Druzdzel (1998). *Relevance-based sequential evidence processing in Bayesian networks*. In Proceedings of the Uncertain Reasoning in Artificial Intelligence track of the Eleventh International Florida Artificial Intelligence Research Symposium (FLAIRS-98), Sanibel Island, Florida, pages 446-450. (An extended version of this paper will appear in the *International Journal of Pattern Recognition and Artificial Intelligence*.)

Lupinska-Dubicka, Anna (2014). *Probabilistic Graphical Models of Time-Dependent Domains with Memory: Application to Monitoring Woman's Monthly Cycle*. Doctoral dissertation, Faculty of Computer Science, Bialystok University of Technology, Poland.

Matzkevich, Izhar & Bruce Abramson (1995). *Decision analytic networks in artificial intelligence*. Management Science, 41(1):1-22.

Morgan, M. Granger & Max Henrion (1990). *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*. Cambridge University Press, Cambridge.

Murphy, Kevin P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Doctoral dissertation, University of California, Berkeley.

Neapolitan, Richard E. (1990). *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. John Wiley & Sons, New York.

Olmsted, S. (1983). *On representing and solving decision problems*. PhD thesis, Department of Engineering-Economic Systems, Stanford University.

Agnieszka Onisko (2003). *Probabilistic Causal Models in Medicine: Application to Diagnosis of Liver Disorders*. Ph.D. Dissertation, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, Warsaw.

Onisko, Agnieszka, Marek J. Druzdzel & Hanna Wasyluk (2000). *Extension of the Hepar II model to multiple-disorder diagnosis*. In Intelligent Information Systems, M. Klopotek, M. Michalewicz, S.T. Wierzchon (eds.), pages 303-313, Advances in Soft Computing Series, Physica-Verlag (A Springer-Verlag Company), Heidelberg.

Onisko, Agnieszka, Marek J. Druzdzel & Hanna Wasyluk (2001). *Learning Bayesian network parameters from small data sets: Application of Noisy-OR gates*. International Journal of Approximate Reasoning, 27(2):165-182, 2001.

Onisko, Agnieszka & Marek J. Druzdzel (2013). *Impact of precision of Bayesian networks parameters on accuracy of medical diagnostic systems*. Artificial Intelligence in Medicine, 57(3):197-206.

Pearl, Judea (1986). *Fusion, propagation, and structuring in belief networks*. Artificial Intelligence, 29(3):241-288.

Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Quinn, N.R., Jr & M.A. Breuer (1979). *A force directed component placement procedure for printed circuit boards*. IEEE Transactions on Circuits and Systems, CAS 26, pages 377–388.

Russell, Stuart J. & Peter Norvig (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

Savage, Leonard (1954). *The Foundations of Statistics*. Dover, New York.

Shachter, Ross D. (1986). *Evaluating influence diagrams*. Operations Research. 34(6):871-882.

Shachter, Ross D. (1988). *Probabilistic inference and influence diagrams*. Operations Research, 36(4):589-604.

Shachter, Ross D. & Mark A. Peot (1990). *Simulation approaches to general probabilistic inference on belief networks*. In Henrion, M., Shachter, R.D., Kanal, L.N. & Lemmer, J.F. (eds.) Uncertainty in Artificial Intelligence 5. Elsevier Science Publishers B.V. (North Holland), pages 221-231.

Shachter, Ross D. & Mark A. Peot (1992). *Decision making using probabilistic inference methods*. In Proceedings of the Eighth Annual Conference on Uncertainty in Artificial Intelligence (UAI-92), Morgan Kaufmann Publishers: San Francisco, CA, pages 276-283.

Simon, Herbert A. (1996). *The Sciences of the Artificial*. 3rd edition. MIT Press.

Spiegelhalter, David J., A. Philip Dawid, Steffen L. Lauritzen & Robert G. Cowell (1993). *Bayesian analysis in expert systems*. Statistical Science, 8(3):219-283.

Spirtes, Peter, Clark Glymour & Richard Scheines (1993). *Causation, Prediction, and Search*. Springer Verlag Lectures in Statistics.

Srinivas, Sampath (1993). *A generalization of the Noisy-OR model*. In Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI-93), Morgan Kaufmann: San Mateo, CA, pages 208-215.

Voortman, Mark & Marek J. Druzdzel (2008). *Insensitivity of constraint-based causal discovery algorithms to violations of the assumption of multivariate normality*. In Recent Advances in Artificial Intelligence: Proceedings of the Twenty First International Florida Artificial Intelligence Research Society Conference (FLAIRS-2008), David Wilson, H. Chad Lane (eds), pages 690-695, Menlo Park, CA: AAAI Press.

Whittaker, Joe (1990). *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, Chichester.

Yuan, Changhe & Marek J. Druzdzel. *An Importance Sampling Algorithm Based on Evidence Pre-propagation*. In Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-03), Morgan Kaufmann: San Mateo, CA, pages 624-631.

Yuan, Changhe & Marek J. Druzdzel (2005). *Importance sampling algorithms for Bayesian networks: Principles and performance*. Mathematical and Computer Modeling, 43(9-10):1189-1207.

Yuan, Changhe & Marek J. Druzdzel (2006). *Hybrid loopy belief propagation*. In Proceedings of the Third European Workshop on Probabilistic Graphical Models (PGM-06), pages 317-324, Milan Studeny and Jiri Vomlel (eds.), Prague: Action M Agency.

Yuan, Changhe & Marek J. Druzdzel (2007). *Generalized Evidence Pre-propagated Importance Sampling for Hybrid Bayesian Networks*, In Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07), pages 1296-1302, Vancouver, British Columbia, Canada.

Yuan, Changhe, Tsai-Ching Lu & Marek J. Druzdzel (2004). *Annealed MAP*. In Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04), AUAI Press, Arlington, Virginia, pages 628-635.

Zagorecki, Adam T. (2010). *Local Probability Distributions in Bayesian Networks: Knowledge Elicitation and Inference*. Doctoral dissertation, School of Information Sciences, University of Pittsburgh.

- . -

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- J -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- W -

- X -

- Y -

- Z -