

GeNle Modeler

USER MANUAL

Version 2.1.1, Built on 1/27/2017

BayesFusion, LLC

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Table of Contents

3

1. Read me first	9
2. Hello GeNle!	11
3. Introduction	29
3.1 Guide to GeNle manual	30
3.2 GeNle Modeler	30
3.3 SMILE Engine	31
3.4 Distribution information	32
3.5 GeNle on a Mac	36
3.6 Copyright notice	38
3.7 Disclaimer	39
3.8 Acknowledgments	39
4. Decision-theoretic modeling	41
4.1 Decision analysis	42
4.2 Discrete and continuous variables	42
4.3 Probability	43
4.4 Utility	44
4.5 Bayesian networks	45
4.6 Influence diagrams	47
4.7 Bayesian updating	48
4.8 Solving decision models	49
4.9 Changes in structure	50
4.10 Decision support systems	50
5. Building blocks of GeNle	55
5.1 Introduction	56
5.2 GeNle workspace	57
5.2.1 Introduction	57
5.2.2 The menu bar	59
5.2.3 Graph view	60
5.2.4 Tree view	73
5.2.5 Status bar	76
5.2.6 Case manager	78
5.2.7 Output window	82
5.2.8 Help menu	83

5.3 Components of GeNIe models	85
5.3.1 Node types	85
5.3.2 Canonical models	87
5.3.3 Multi-attribute utility nodes	100
5.3.4 Submodels	105
5.3.5 Arcs	115
5.3.6 Node status icons	116
5.3.7 Text boxes	119
5.3.8 Annotations	120
5.4 Model and component properties	123
5.4.1 Network properties	123
5.4.2 Submodel properties	133
5.4.3 Tools menu and Standard toolbar	136
5.4.4 Node properties	138
5.5 Visual appearance, layout, and navigation	173
5.5.1 Introduction	173
5.5.2 Viewing nodes in the Graph View	173
5.5.3 Zooming and full screen mode	176
5.5.4 Format toolbar and Layout menu	177
5.5.5 Graph layout functions	182
5.5.6 Selection of model elements	184
5.5.7 Model navigation tools	186
5.6 Saving and loading models in GeNIe	188
5.6.1 Introduction	188
5.6.2 File menu	193
5.6.3 XDSL file format	197
5.6.4 DSL file format	197
5.6.5 Ergo file format	198
5.6.6 Netica file format	199
5.6.7 BN interchange format	200
5.6.8 Hugin file format	200
5.6.9 KI file format	200
5.7 Inference algorithms	200
5.7.1 Introduction	200
5.7.2 Immediate and lazy evaluation	203
5.7.3 Relevance reasoning	204
5.7.4 Node menu	207
5.7.5 Network menu	209
5.7.6 Bayesian networks algorithms	211
5.7.6.1 Exact algorithms	211

5.7.6.1.1	Clustering algorithm	211
5.7.6.1.2	Relevance-based decomposition	213
5.7.6.1.3	Polytree algorithm	213
5.7.6.2	Stochastic sampling algorithms	214
5.7.6.2.1	Probabilistic Logic Sampling	214
5.7.6.2.2	Likelihood Sampling	214
5.7.6.2.3	Self-Importance Sampling	214
5.7.6.2.4	Backward Sampling	214
5.7.6.2.5	AIS algorithm	215
5.7.6.2.6	EPIS Sampling	215
5.7.6.3	Special algorithms	216
5.7.6.3.1	Probability of evidence	216
5.7.6.3.2	Annealed MAP	220
5.7.7	Influence diagrams algorithms	227
5.7.7.1	Policy evaluation	227
5.7.7.2	Find Best Policy	228
5.7.8	Algorithms for continuous models	229
5.7.8.1	Introduction	229
5.7.8.2	Autodiscretization	230
5.7.8.3	Hybrid LW	235
5.7.8.4	Hybrid Logic Sampling	235
5.7.8.5	Hybrid LBP	235
5.7.8.6	Hybrid EPIS	236
5.8	Obfuscation	236
5.9	Program options	241
5.10	Keyboard shortcuts	246
6.	Using GeNle	249
6.1	Introduction	250
6.2	Bayesian networks	250
6.2.1	Building a Bayesian network	250
6.2.2	Useful structural transformations	250
6.2.3	Entering and retracting evidence	256
6.2.4	Virtual evidence	261
6.2.5	Viewing results	263
6.2.6	Strength of influences	270
6.2.7	Controlling values	273
6.2.8	Sensitivity analysis in Bayesian networks	276
6.3	Influence diagrams	281
6.3.1	Building an influence diagram	281
6.3.2	Viewing results	290
6.3.3	Sensitivity analysis in influence diagrams	291
6.3.4	Value of information	296

Table of Contents

6

6.4	Support for diagnosis	303
6.4.1	Introduction	303
6.4.2	Diagnosis menu	304
6.4.3	Diagnosis toolbar	307
6.4.4	Enabling diagnostic extensions	308
6.4.5	Spreadsheet view	317
6.4.6	Testing window	324
6.4.7	Diagnostic case management	330
6.4.8	Cost of observation	334
6.5	Learning	339
6.5.1	Accessing data	339
6.5.2	Data menu	351
6.5.3	Cleaning data	353
6.5.4	Knowledge editor	373
6.5.5	Pattern editor	376
6.5.6	Structural learning	379
6.5.6.1	Introduction	379
6.5.6.2	Bayesian Search	381
6.5.6.3	PC	383
6.5.6.4	Essential Graph Search	389
6.5.6.5	Greedy Thick Thinning	391
6.5.6.6	Tree Augmented Naive Bayes	393
6.5.6.7	Augmented Naive Bayes	395
6.5.6.8	Naive Bayes	398
6.5.7	Learning parameters	400
6.5.8	Generating a data file	407
6.5.9	Validation	411
6.6	Dynamic Bayesian networks	432
6.6.1	Introduction	432
6.6.2	Creating DBN	433
6.6.3	Inference in DBNs	440
6.6.4	Learning DBN parameters	450
6.7	Equation-based models	456
6.7.1	Introduction	456
6.7.2	Constructing equation-based models	456
6.7.3	Inference	462
6.7.4	Viewing results	470
6.7.5	Structural changes	477
7.	Resources	483
7.1	Books	484

Table of Contents

7

7.2	Research papers	484
7.3	Conferences	485
7.4	Model repositories	486
7.5	Social Media	486
7.6	References	487
Index		495

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Read me first

1 Read me first

Welcome to GeNIe manual, Version 2.1.1, Built on 1/27/2017.

This manual is available in CHM, PDF and HTML formats, all available at
<http://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!](#)^[12] section. If you are an advanced user, please browse through the *Table of Contents* or search for the topic of your interest.



Hello GeNle!

2 Hello GeNIE!

This section offers an informal introduction to GeNIE, similar to the light introduction to the C programming language offered by Brian Kernighan 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. Once you have made yourself familiar with GeNIE in this informal way, you can proceed with the *Elements of GeNIE* chapter, which offers a thorough introduction to various elements 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 what you can do with GeNIE. 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](#)⁴⁵ 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*.

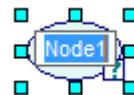
If you have not already started GeNIE, start it now.

The [Tool Menu](#)¹⁷⁶ shows a list of different types of nodes that you can create. These are also displayed as buttons on the [Standard Toolbar](#)¹⁷⁶.

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

Select *Chance* button () from the [Standard Toolbar](#)¹⁷⁶ or [Tool Menu](#)¹⁷⁶.

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](#)⁶⁰) 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 a node button instead of single-clicking it the first time. 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.

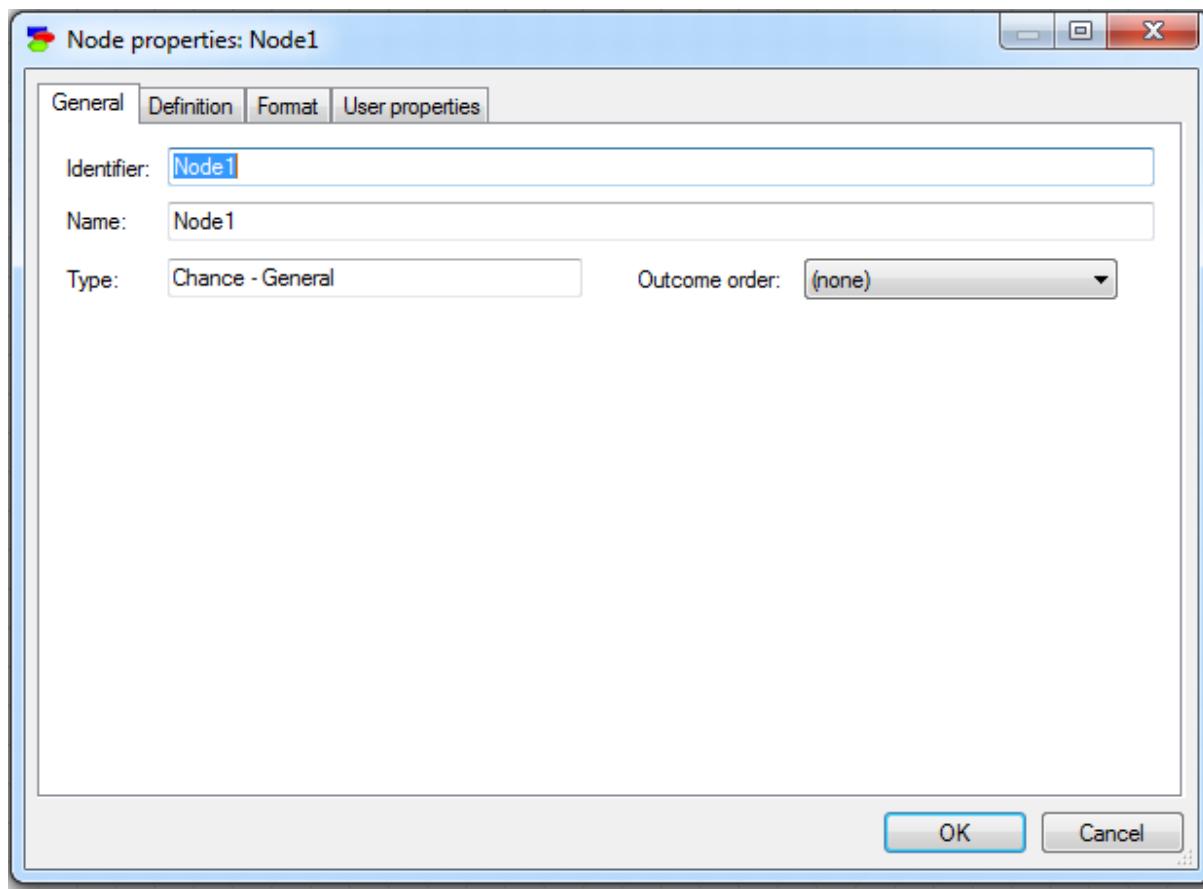
Once you have drawn the node on the [Graph View](#)⁶⁰, the *Chance* button on the toolbar will become normal again and the *Select Objects* button will become recessed.

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. Names are simply strings of characters with no limitations. GeNIE assigned the node that you have just created identifier and 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 name for the newly created node.

Double click on the node *Node1*.

GeNIE will display the following dialog box:



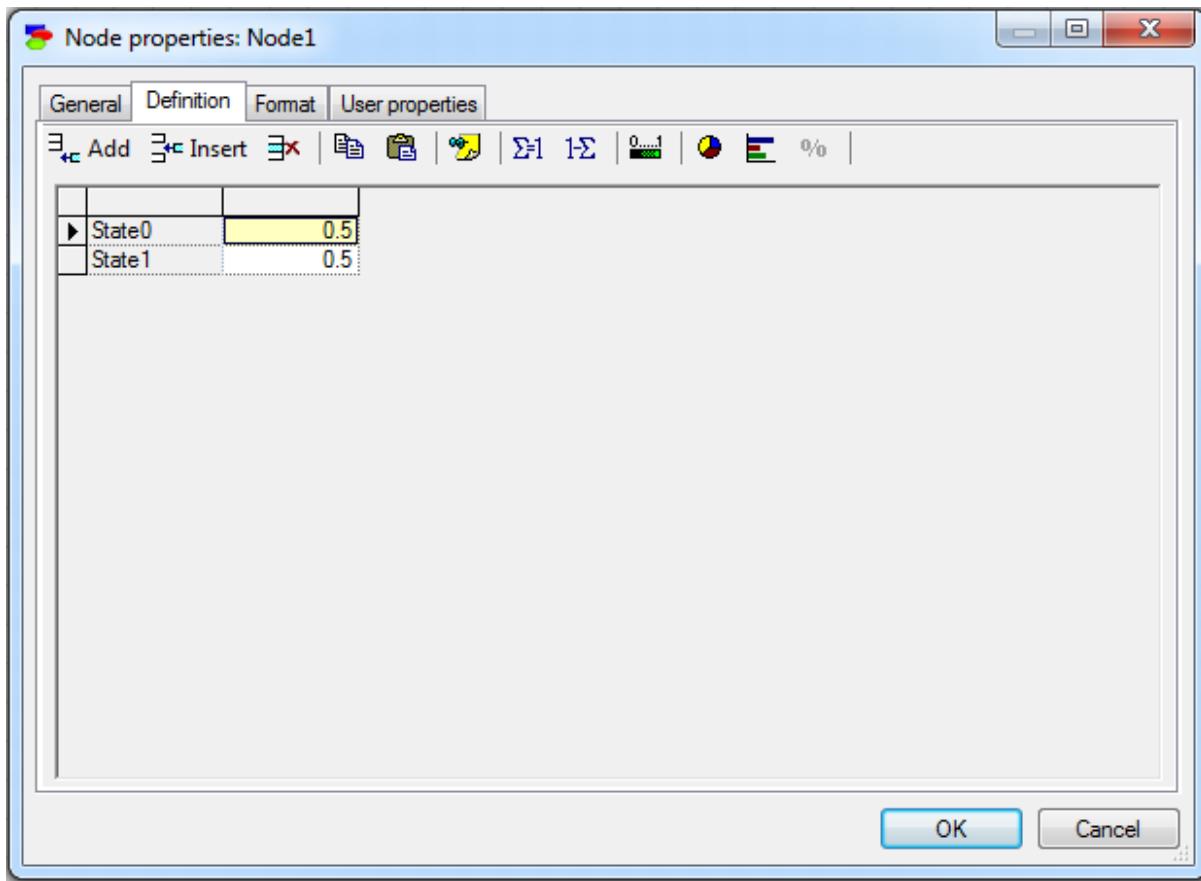
These are the [Node Property Sheets](#)¹³⁸, which are used to specify various properties of the node.

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](#)¹³⁸ sheets.

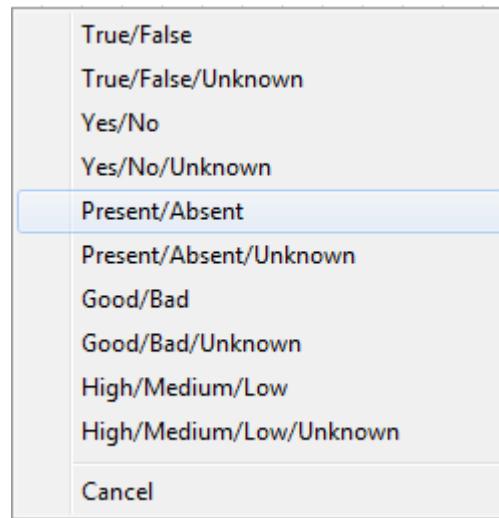
1. Click on the *Definition* tab

GeNIE will display the following dialog:

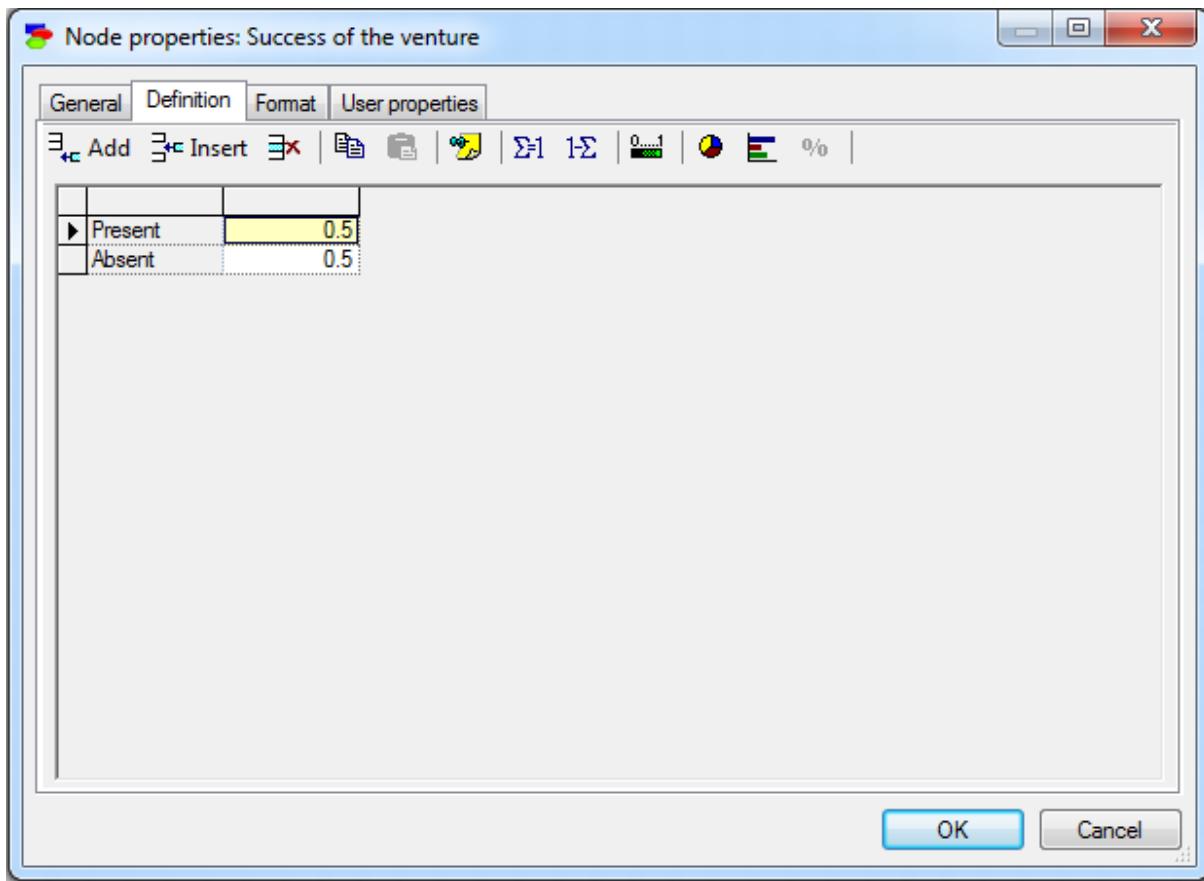


2. Double-click on the identifier of the first state, and change its name to *Success*
3. Similarly rename *State1* into *Failure*.

You may also select the states of the node from a *Quick States* list at the time of laying a new chance node on the *Graph View*. To do so, select  (*Chance button*) from the [Standard Toolbar](#)¹⁷⁶ or [Tool Menu](#)¹⁷⁶. 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 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 remain 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 $P(\text{Success})$ to 0.2 and $P(\text{Failure})$ 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:

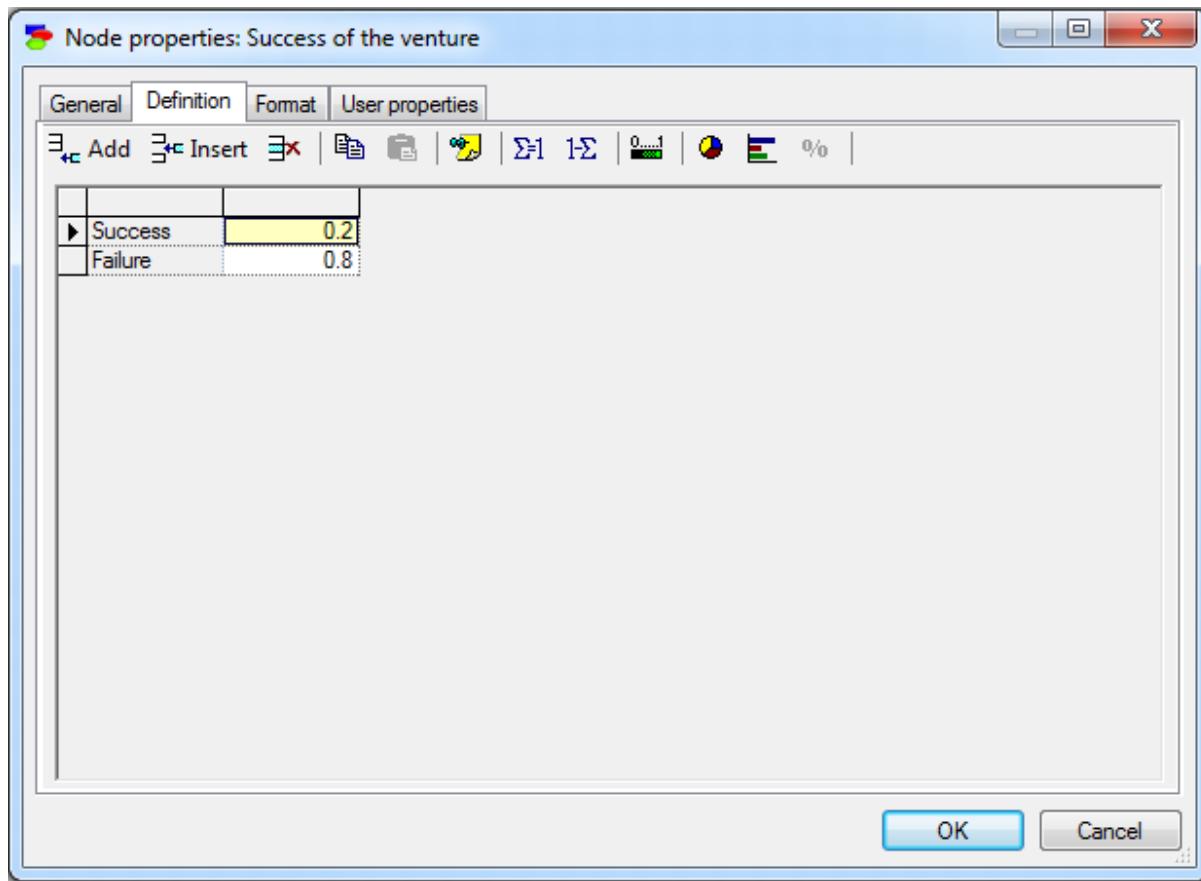
4. Double click on the value field for the *Success* state and enter 0.2.
5. Select the probability for the *Failure* state (currently 0.5).
6. Click on the *Complement* () button. This will set the value field to 0.8, which is 1-0.2, so that the sum of probabilities is equal to 1.0.

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* (). 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](#)⁶⁰.

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](#)⁶⁰.

Your *Graph View* will look similar to this:



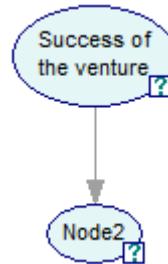
The label for the "Succ..." node is not completely displayed because we have changed the text inside the properties tab but not adjusted the node size. You can adjust the size of the node to fit the text by selecting it and dragging one of the small squares. GeNIE offers also a semi-automatic way of adjusting node size. To do so, right click on the node and select *Resize to Fit Text* from the *Node Popup 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:

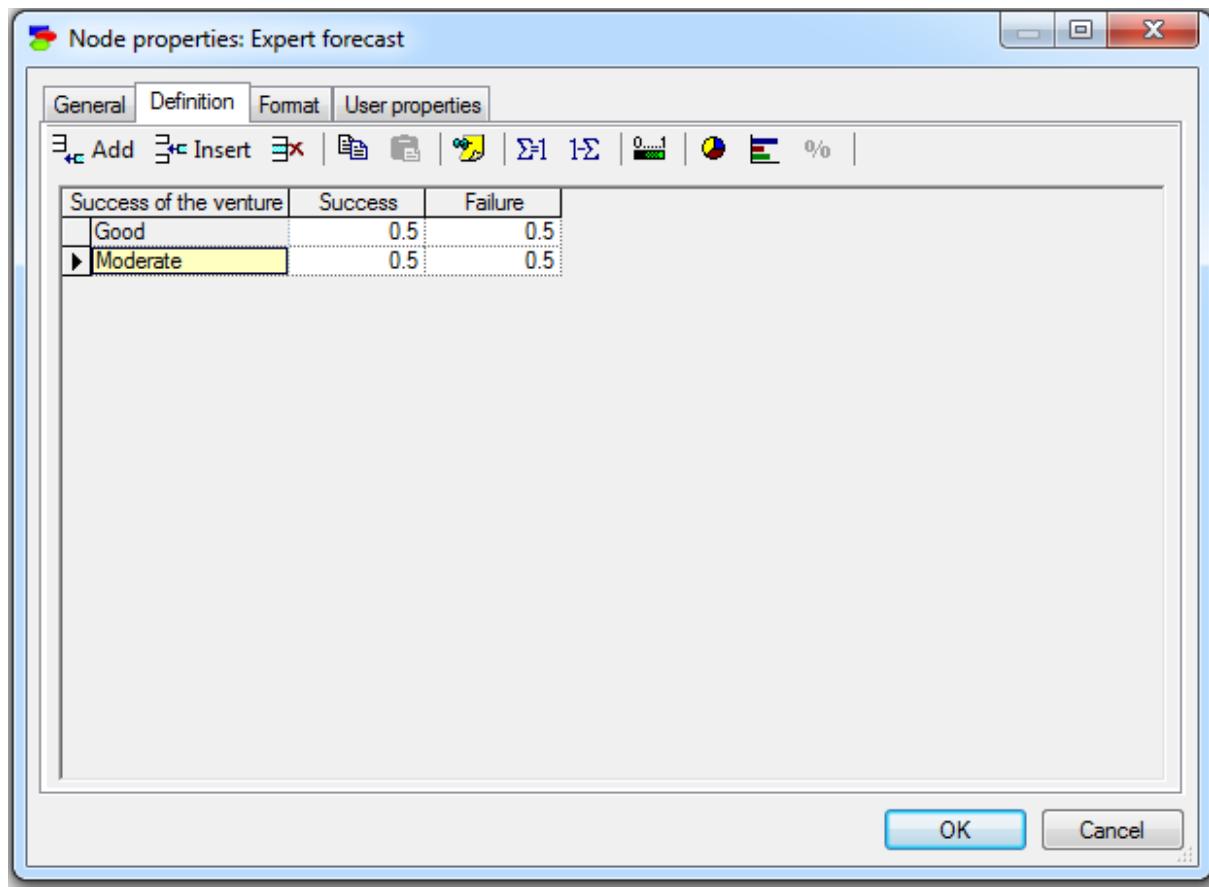


The arc between the two nodes means that whether or not the venture is going to be successful makes a difference for the probability distribution over various statements made by the expert (this is going to be expressed by the conditional probability distributions over *Node2*).

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

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

7. Now enter the probabilities for each state combination. Use the values shown below:

The screenshot shows the 'Node properties' dialog box for an 'Expert forecast' node. The 'General' tab is selected. Below it is a table titled 'Success of the venture' with three rows: 'Good', 'Moderate', and 'Poor'. The table has three columns: 'Success' and 'Failure'. The 'Success' column contains values 0.4, 0.4, and 0.2 respectively. The 'Failure' column contains values 0.1, 0.3, and 0.6 respectively. The dialog box also features standard Windows-style buttons for 'OK' and 'Cancel' at the bottom right.

Success of the venture	Success	Failure
► Good	0.4	0.1
Moderate	0.4	0.3
Poor	0.2	0.6

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.

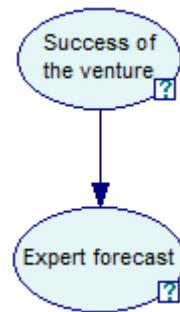
8. Click *OK* to accept all changes and return to the [Graph View](#)⁶⁰.

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* (.ALIGN CENTER) button on the [Format Toolbar](#)¹⁷⁷.

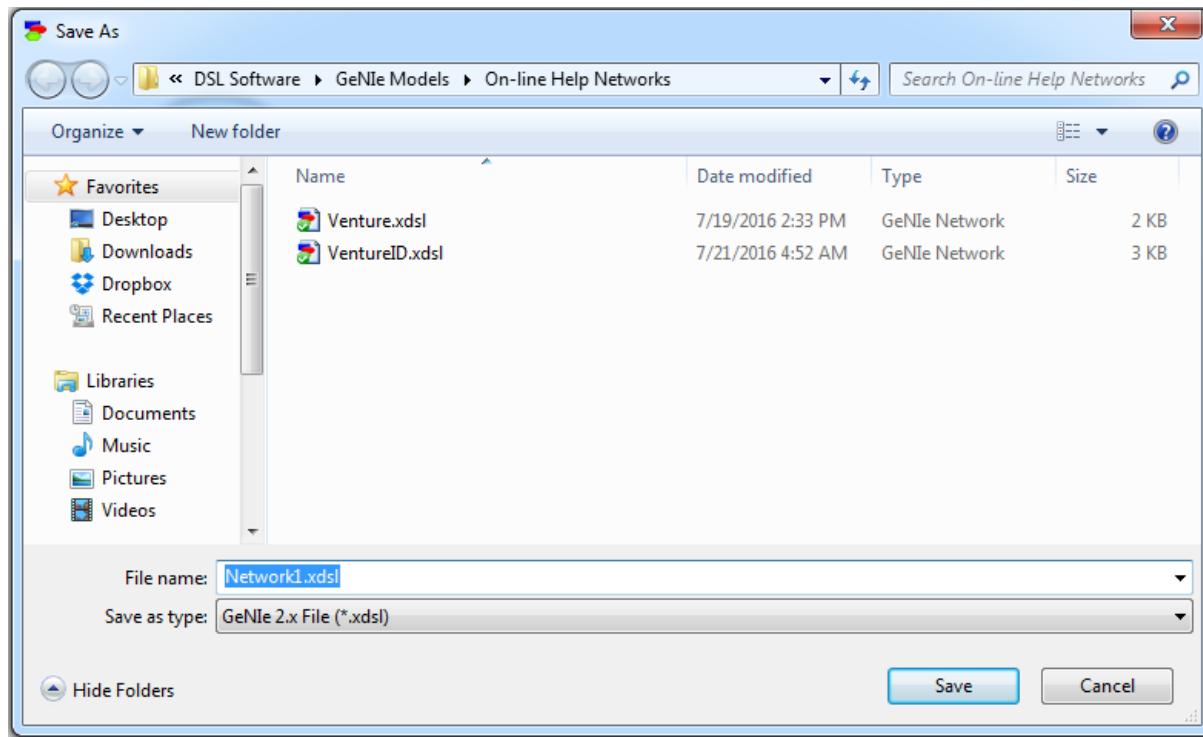
Your network should look like this:



- G.** At this point you should save your work.

- 1.** Click on *Save* button (FILE) on the [Standard Toolbar](#)¹⁷⁶.

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

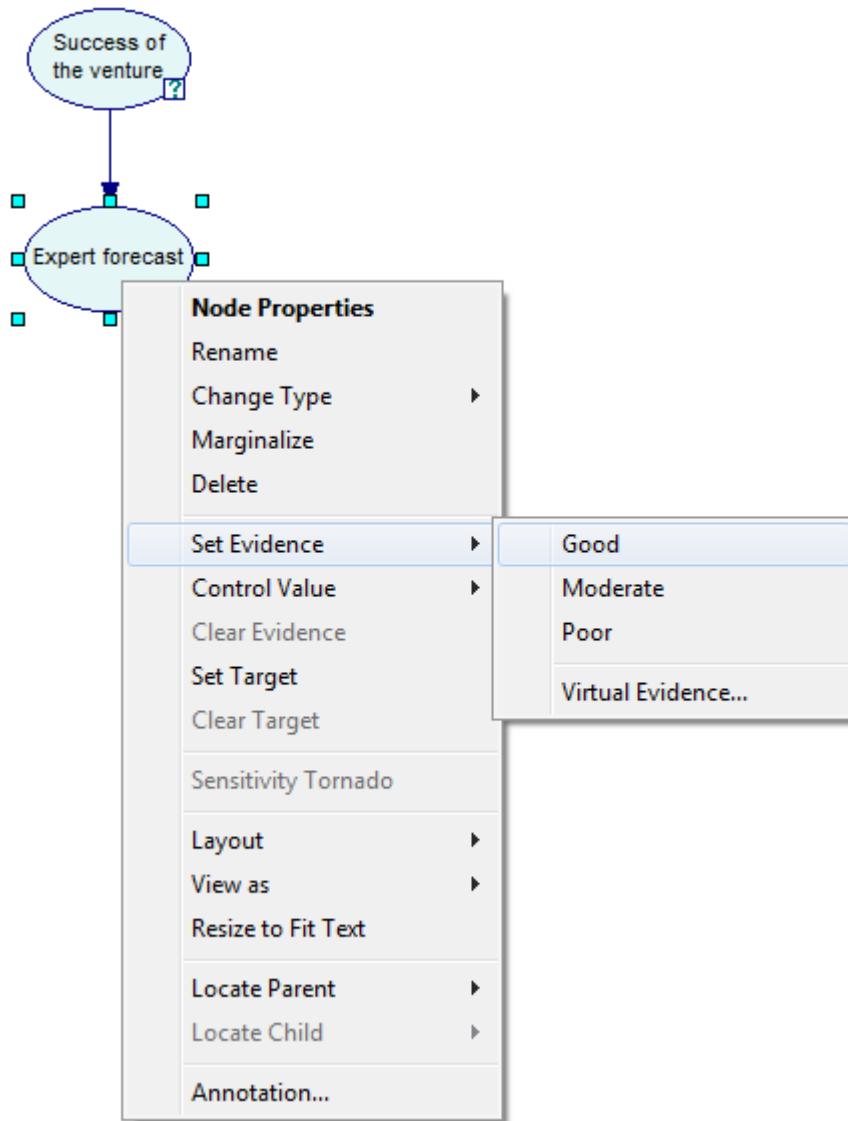


2. Enter *VentureBN* as the *File name* and click on *Save*.

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*.



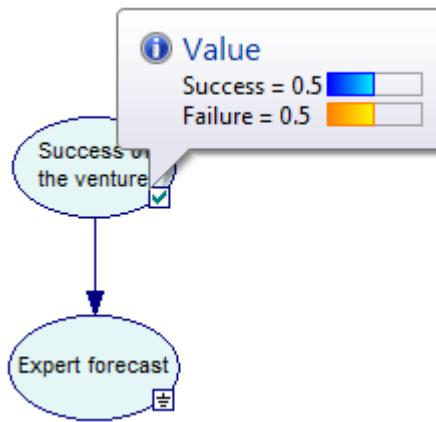
Note that the status icon on the bottom right of the node changes from to . This indicates that the node has been observed.

- 2.** Click on the *Update* tool () on the [Standard Toolbar](#)¹⁷⁶.

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.

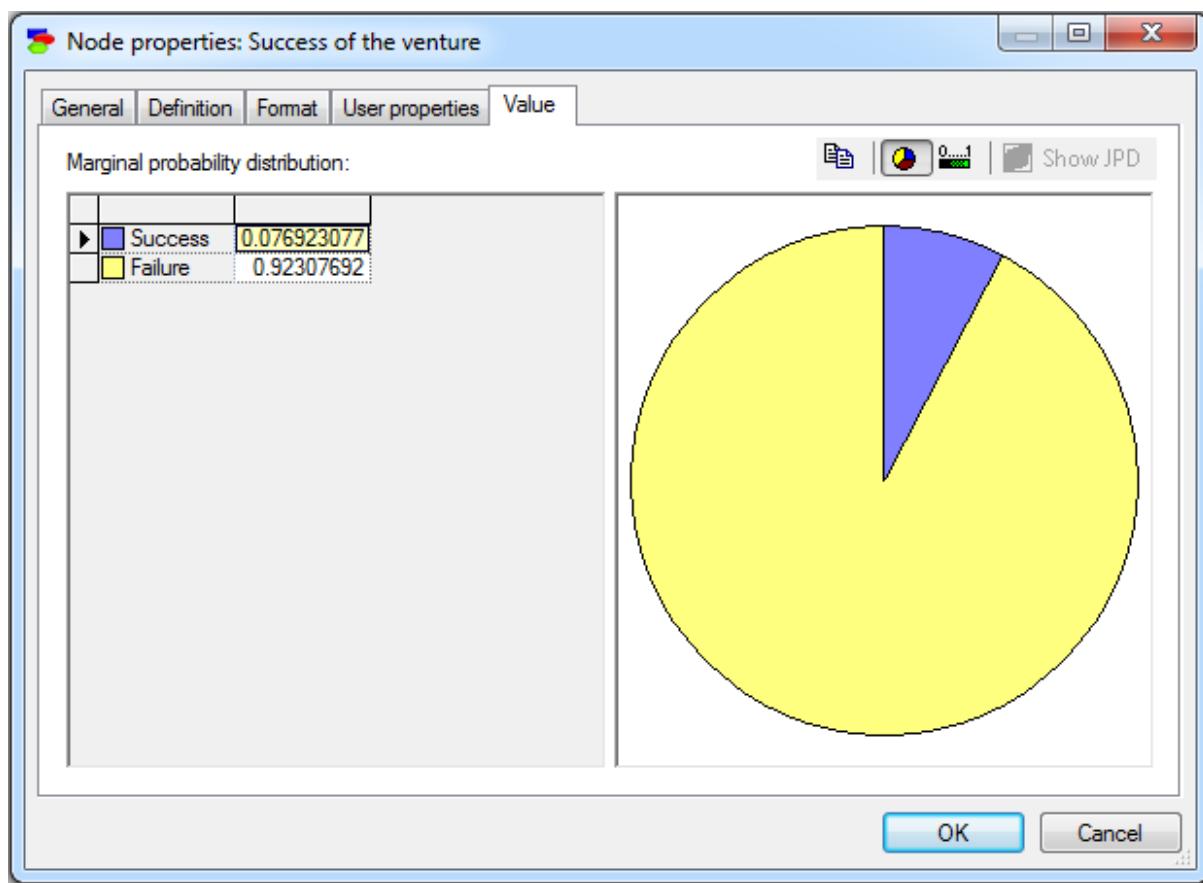
To answer the second question: "What if he judges them to be poor?" we will set the evidence in the node *Expert forecast* to *Poor*, update the model, and observe that the probability of success is now less than 0.08.

Results of [Bayesian updating](#)⁴⁸ can be also viewed by double-clicking on a node and selecting the *Value* tab.

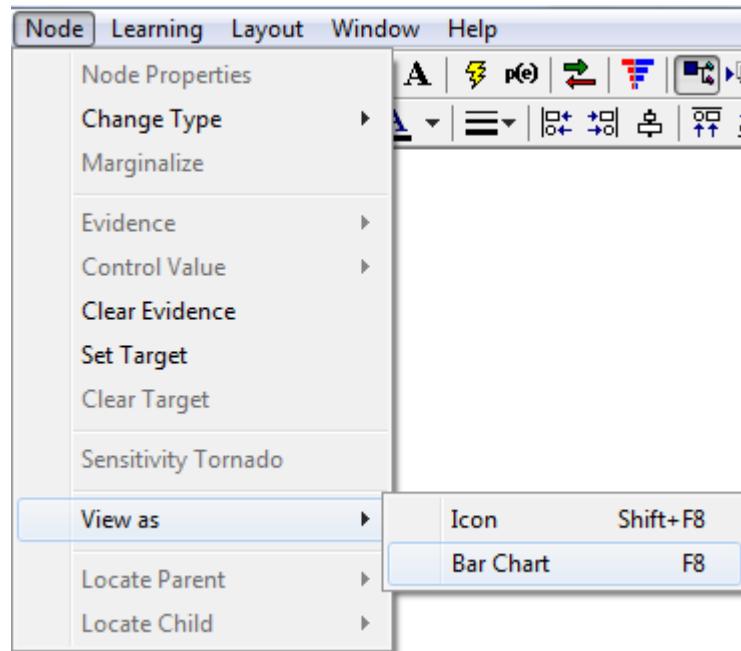
4. Double click on the *Success of venture* node.

5. Select *Value* tab from the [Node Property](#)¹³⁸ sheets

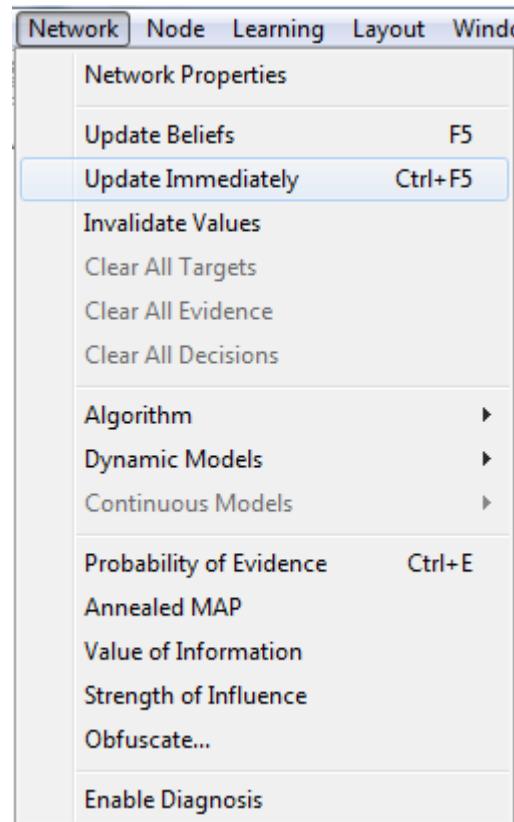
The result should look as follows:



Entering evidence and displaying results can be done much simpler. To try this, select all nodes (*CTRL-A* or select them with a mouse) and change their appearance from *Icon* to *Bar Chart*.

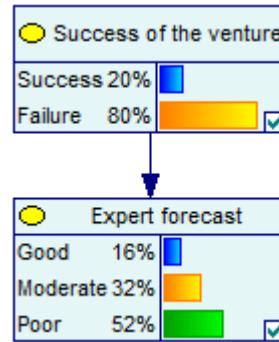


Also, set the *Update Immediately* flag:

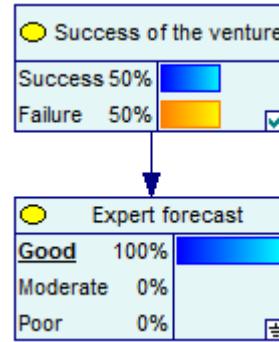


The *Update Immediately* flag, when set, invokes Bayesian inference as soon as any change happens to the model. It is convenient to have it turned off as we build a model (to avoid computation when the model is incomplete and it does not make much sense yet) and turned on when the model is ready.

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. Most of the tasks in GeNIE are intuitive and conform to the general standards of well designed user interfaces. However, 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 <http://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. In 2015, we created a company, BayesFusion, LLC, and acquired a license for GeNIE from the University of Pittsburgh. 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's name and its uncommon capitalization originates from the name Graphical Network Interface, given to the original simple interface to SMILE^[31], our library of classes for graphical probabilistic and decision-theoretic models. GeNIE is an outer shell to SMILE.

GeNIE is 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) under Wine. Please see [GeNIE on a Mac](#)^[36] section for more information on

running GeNIe on a macOS. Our users have also reported running GeNIe on Linux using Wine.

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, using SMILE, which is fully portable.

GeNIe 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 both programs. Because of their versatility and reliability, GeNIe and SMILE have become incredibly popular and became de-facto standards in academia, while being embraced by a number of government, military, and commercial users.

The strongest element of GeNIe, 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 orders of magnitude and it will lead to a fast return of the investment in its licensing fees. SMILE is not far behind and belongs to the easiest to learn and use, most reliable, and fastest libraries for graphical models.

3.3 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](#)^[45], [influence diagrams](#)^[47], 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) and .NET (SMILE.NET) 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 R, Matlab, Python 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 only on Windows computers, although 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 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 and SMILE have become incredibly popular and became *de facto* standards in academia, while being embraced by a number of government, military and commercial users.

The strongest element of SMILE, one that distinguishes 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 done very nicely in UAI speed competitions). Speed especially is crucial, as most calculations in probabilistic graphical models are exponential in nature.

3.4 Distribution information

Hardware and software requirements

Disk space

Full installation of GeNIE requires less than 20 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.

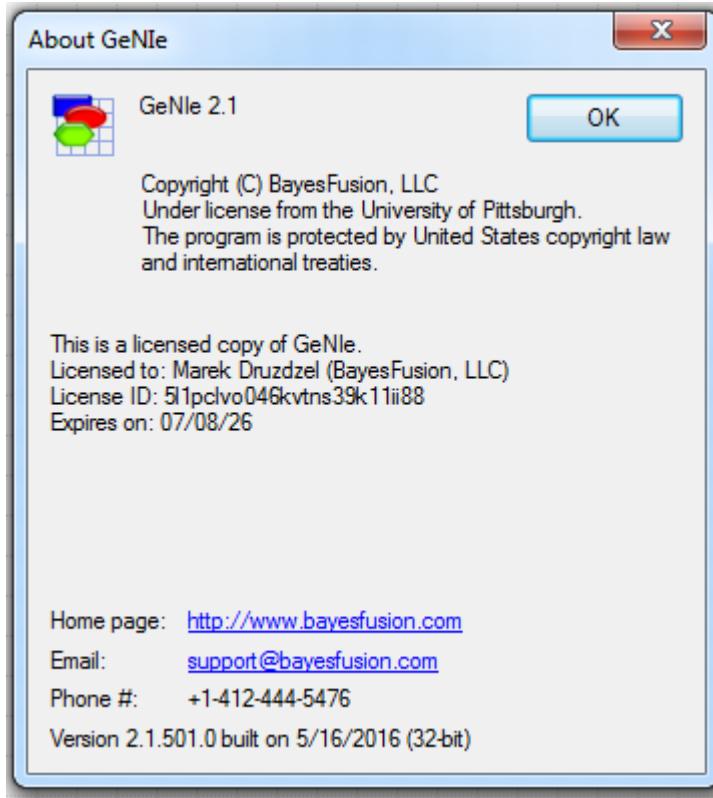
Operating system

GeNIE is 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) under Wine. Please see [GeNIE on a Mac](#)³⁶ section for more information on running GeNIE on a macOS. Our users have also reported running GeNIE on Linux using Wine.

GeNIE version

To determine the version of GeNIE that you have installed, select *About GeNIE* from the [Help Menu](#)⁸³. The version number is listed in the small frame of the following window:



Example Networks

GeNIE installation creates a directory named *Example Networks*, containing the following models:

A.xdsl

The *A* network is a randomly generated network contributed to the community by Alex Kozlov and used for several benchmarks in the UAI literature. Even though *A* contains only 54 nodes, it is rather densely connected and is known to be

daunting to exact algorithms. The network has been first mentioned in (Kozlov & Singh, 1996).

Alarm.xdsl

The Alarm network has been developed for on-line monitoring of patients in intensive care units and generously contributed to the community by Ingo Beinlich and his collaborators. The model has first appeared in (Beinlich et al. 1989).

Animals.xdsl

A simple animal guessing game modeled by and made available to the community by Noetic, Inc., the developers of Ergo. The network will guess which of the five animals you have in mind, as you provide information about habitat and characteristics of the animal. The network illustrates well the interaction between probability and propositional logic.

Asia.xdsl

This is an example graphical model useful in demonstrating basics concepts of Bayesian networks in diagnosis. It first appeared in (Lauritzen & Spiegelhalter 1988).

AsiaSmoking.xdsl

This is the Asia network due to Lauritzen and Spiegelhalter (1988), extended by us to an influence diagram. It is useful in demonstrating basics concepts of influence diagrams. The diagnostic part (the decision and the value node) helps in deciding whether or not to smoke.

B.xdsl

The *B* network is a randomly generated network contributed to the community by Alex Kozlov and used for several benchmarks in the UAI literature. Even though *B* contains only 18 nodes, it is rather densely connected and is known to be daunting to exact algorithms. The network has been first mentioned in (Kozlov & Singh 1996).

Coma.xdsl

The *Coma* (also known as *Cancer*) network appeared first in Greg Cooper's doctoral dissertation (Cooper, 1984).

Credit.xdsl

A simple network for assessing credit worthiness of an individual, developed by Gerardina Hernandez as a class homework at the University of Pittsburgh.

Disease-Test.xdsl

A simple network useful in demonstrating the simplest possible application of Bayes theorem.

Hailfinder2-5.xdsl

Hailfinder is a normative system that forecasts severe summer hail in northeastern Colorado. It has been generously contributed to the community by Ward Edwards and Bruce Abramson. *Hailfinder* was described in (Abramson et al. 1996).

MAUNetwork.xdsl

A simple, incomplete diagram demonstrating Multi-Attribute Utility nodes in GeNIe.

Pinball.xdsl

An influence diagram has been developed by Michael T. Filipiak, Charles Neville and Joseph C. Wynn as a class project at Carnegie Mellon University. It models a real business decision related to a hobby (collecting pinball machines) turned into a business.

Tank.xdsl

A simple network for diagnosing the possibility of a possible explosion in a tank, developed by Gerardina Hernandez as a class homework at the University of Pittsburgh.

VentureBN.xdsl

This is a simple Bayesian network from the [Hello GeNIe!](#)^[12] section.

VentureID.xdsl

This is an example of an influence diagram used in the [Creating influence diagrams](#)^[281] and [Value of information](#)^[296] sections.

Command Center.xdsl

A simple model demonstrating how to control values in GeNIE from the [Controlling values](#)²⁷³ section.

VentureID Sensitivity.xdsl

A simple model demonstrating sensitivity analysis in GeNIE from the [Sensitivity analysis](#)²⁹¹ section.

Umbrella.xdsl

A run of the mill umbrella problem, found in most decision analysis textbooks with a weather forecast.

VentureID Submodel.xdsl

A tutorial network demonstrating the use of submodels in GeNIE.

3.5 GeNIE on a Mac

Running GeNIE on macOS (formerly Mac OS X) with Wine.

1. Download and install Wine on your Mac. Follow the instructions at Wine website (<https://wiki.winehq.org/MacOSX>)
2. Download GeNIE on your Mac. Double-click on GeNIE Installer icon to start the setup wizard.
3. After installing GeNIE, run Wine from Launchpad. This opens the *Terminal* window configured to run Windows programs. Do not try to launch GeNIE by locating its icon and double-clicking on it.
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 2.1"
```

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

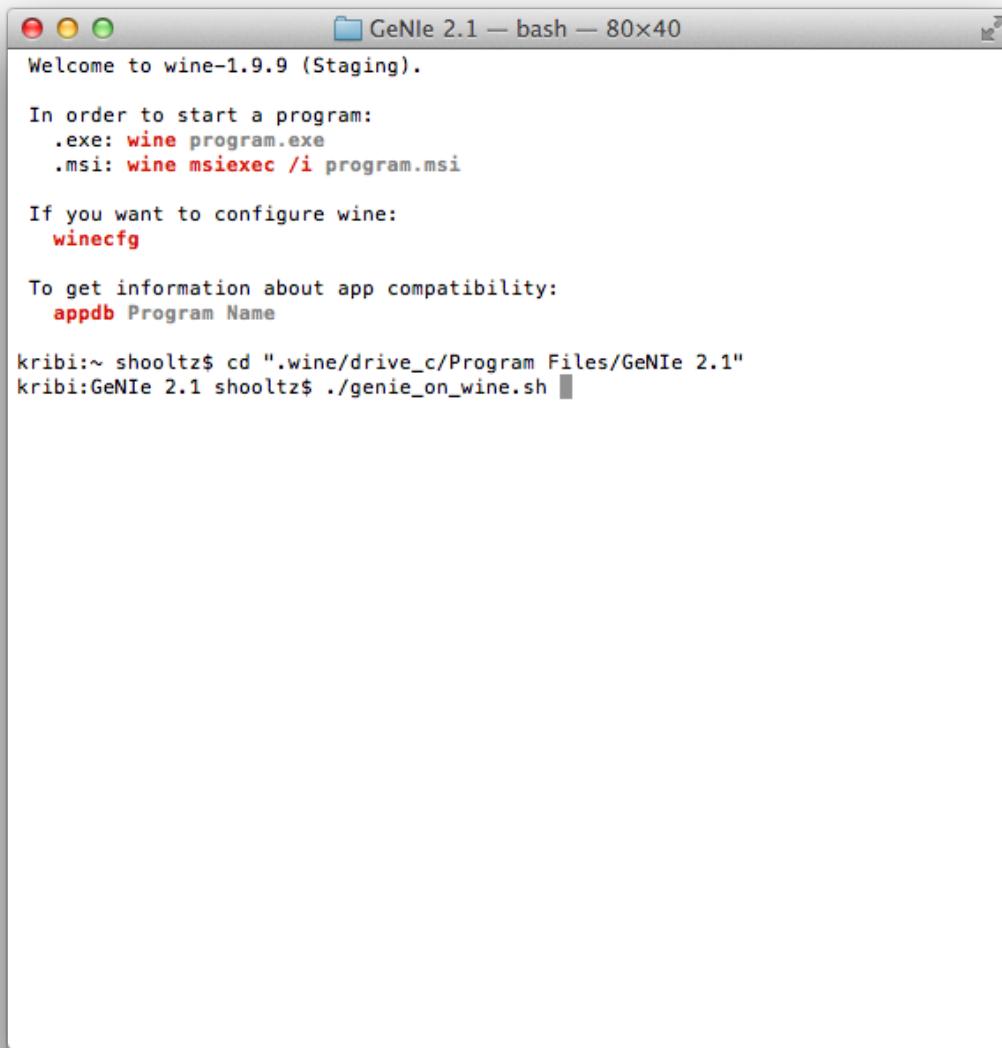
GeNIe Academic installs to "GeNIe 2.1 Academic" by default.

5. After changing the directory, run the `genie_on_wine.sh` script:

```
./genie_on_wine.sh
```

This starts GeNIe using appropriate Wine configuration.

See the screen shot below, illustrating steps 3-5.



```
Welcome to wine-1.9.9 (Staging).

In order to start a program:
  .exe: wine program.exe
  .msi: wine msisexec /i program.msi

If you want to configure wine:
  winecfg

To get information about app compatibility:
  appdb Program Name

kribi:~ shooltz$ cd ".wine/drive_c/Program Files/GeNIe 2.1"
kribi:GeNIe 2.1 shooltz$ ./genie_on_wine.sh
```

3.6 Copyright notice

Copyright (C) [BayesFusion, LLC](#), under license from the [University of Pittsburgh](#). All rights reserved. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, without an explicit written permission of BayesFusion, LLC.

We would like to acknowledge the following trademarks:

Netica and Norsys are trademarks of Norsys Software Corp.,

Hugin is a trademark of Hugin, A.G.,

Ergo is trademark of Noetic Systems, Inc.,

Microsoft and Windows are registered trademarks of Microsoft, Inc.

macOS and OS X are registered trademarks of Apple, Inc.

GeNIe, [SMILE](#)³¹ and all accompanying graphics and manuals are copyrighted (1996-2015) by University of Pittsburgh, used under license by BayesFusion, LLC, and cannot be copied or distributed without permission. The only legal way of obtaining the programs is directly from BayesFusion, LLC. We require that interested individuals contact us directly or visit our web site for the most recent copy of the programs. This ensures the quality and completeness of the programs and accompanying manuals. It also allows us to keep track of who is using the programs and to notify the users about updates and new releases.

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 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 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, <http://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, <http://www.bayesfusion.com/>.

3.7 Disclaimer

GeNIe and [SMILE](#)^[31] are made available on an "as is" basis. We have performed extensive tests of the software, which has been used in hundreds 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.

3.8 Acknowledgments

Support for the development GeNIe and [SMILE](#)^[31] 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 over the last 20 years, the principal developers of GeNIe 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, and *Decision Support and Expert Systems* at the University of Alaska, Anchorage provided us with useful feedback and suggestions.

GeNIE 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](#) has been an inspiration and a role model for us. Analytica's user interface has been 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. Knowledge Industries' WinDX software has been 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 have been used in describing the learning component of GeNIE.

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*. Yet another is *Financial gain* assuming three 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 algorithms for [Bayesian networks](#)^[45] and [influence diagrams](#)^[47] 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.

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 as *Low, Normal, Fever, High fever*. Experience in decision analytic modeling has taught that representing continuous variables by their three to five point discrete approximations perform very 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 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, a reasonable prior belief will lead to a much faster convergence.

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.

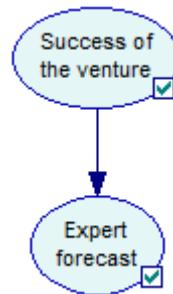
Utility is by assumption subjective: various decision makers facing the same choice and even sharing the same set of beliefs about the world may choose differently because of their different preference structure and different utility functions. A utility function for any decision problem needs to be 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](#)⁴⁷, 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.

Nodes of a Bayesian network are usually drawn as circles or ovals. The following simple Bayesian network, discussed in detail in the [Hello GeNIE!](#)¹² section, represents two variables, *Success of the venture* and *Expert forecast*, and expresses the fact that they are directly dependent on each other.



A Bayesian network also represents the quantitative relationships among the modeled variables. Numerically, it represents the joint probability distribution among them. This distribution is described efficiently by exploring the probabilistic independences among the modeled variables. Each node is described by a probability distribution conditional on its direct predecessors. Nodes with no predecessors are described by prior probability distributions. For example, node *Success of the venture* in the example network above will be described by the prior probability distribution over its two outcomes: *Success* and *Failure*.

	Success	Failure
► Success	0.2	
Failure	0.8	

Node *Expert forecast* will be described by a probability distribution over its outcomes (*Good*, *Moderate*, *Poor*), conditional on the outcomes of its predecessor (node *Success of the venture*, outcomes *Success* and *Failure*).

Success of the venture	Success	Failure
► Good	0.4	0.1
Moderate	0.4	0.3
Poor	0.2	0.6

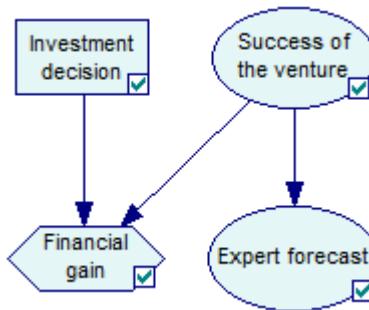
Both, the structure and the numerical parameters of a Bayesian network, can be elicited from an expert. They can also be learned from data, as the structure of a Bayesian network is simply a representation of independences in the data and the numbers are a representation of the joint probability distributions 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.

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](#)⁴³) and this subjective probability distribution can be updated in the light of new evidence using Bayes theorem.

4.6 Influence diagrams

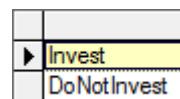
Influence diagrams (Howard & Matheson 1984), also called *relevance diagrams*, are acyclic directed graphs representing decision problems. The goal of influence diagrams is to choose a decision alternative that has the highest expected gain (expected utility^{[44](#)}).

Similarly to [Bayesian networks](#)^{[45](#)}, 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, discussed in detail in the [section on influence diagrams](#)^{[281](#)}, 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*.



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.

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).

Value nodes, usually drawn as diamonds (such as node *Financial gain* above), represent [utility](#)⁴⁴, 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		Do Not Invest	
	Success	Failure	Success	Failure
Value	10000	-5000	500	500

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 Bayesian updating

[Bayesian networks](#)⁴⁵ 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 between a fraction of a second and a few seconds.

4.8 Solving decision models

In brief, solving decision-theoretic models amounts to computation of the expected utility⁴⁴ 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 diagrams⁴⁷ that transforms an influence diagrams into a Bayesian network⁴⁵ 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.

4.9 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](#)²⁷³ 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](#)²⁷³ section of this document for additional details.

4.10 Decision support systems

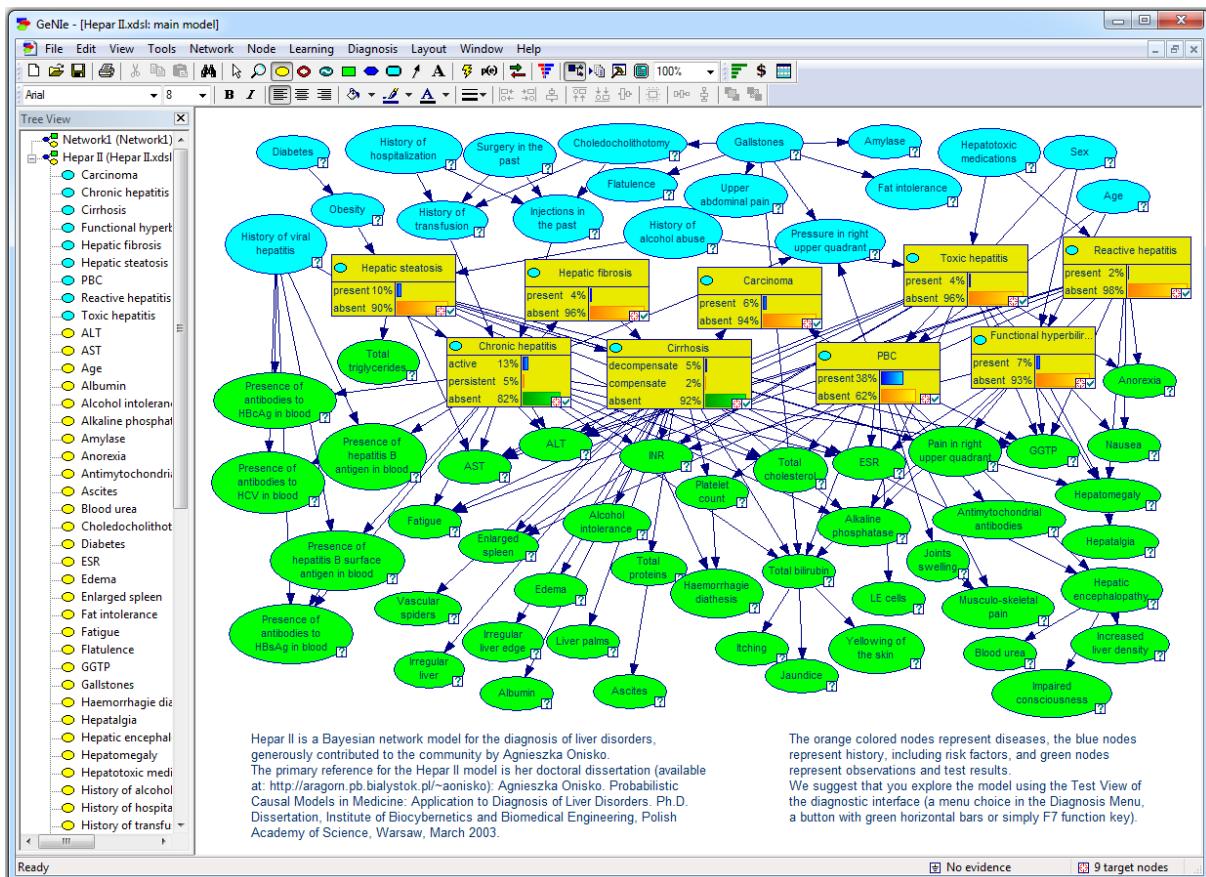
The principles of decision-analytic decision support, implemented in GeNIE and [SMILE](#)³¹ can be applied in practical decision support systems (DSSs). In fact, quite a number of probabilistic decision support systems have been developed, in which GeNIE plays the role of a developer's environment and SMILE plays the role of the reasoning engine. A decision support system based on SMILE can be equipped with a dedicated user interface.

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.

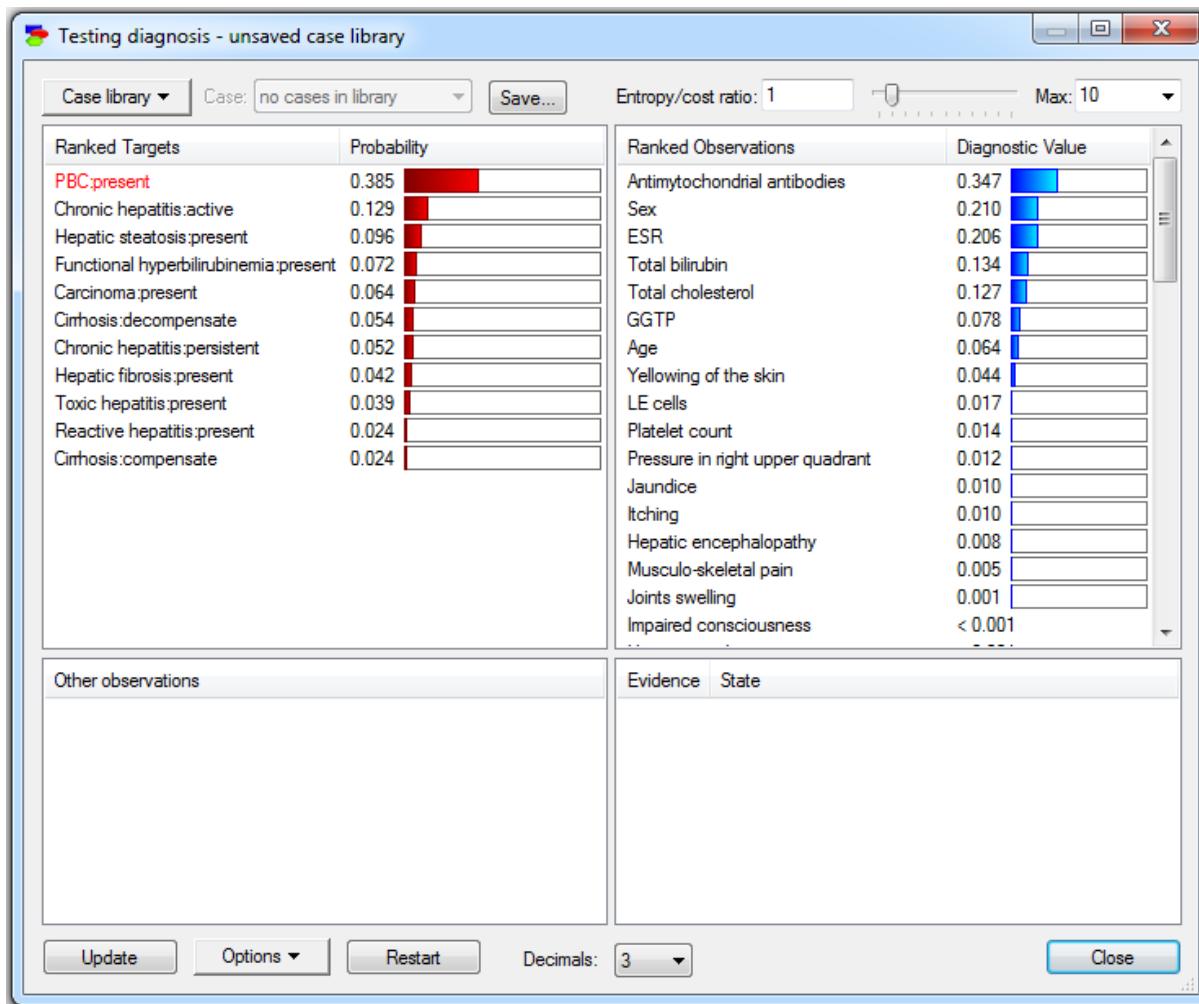
Probabilistic DSSs are based on a philosophically different principle than rule-based expert systems. While the latter attempt to model the reasoning of a human expert, the former use an axiomatic theory to perform computation. The soundness of probability theory provides a clear advantage over rule-based 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). 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 is currently used both as a diagnostic aid and a training tool.

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Building blocks of GeNle

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 typically found in one of the menus.

Dialogs

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

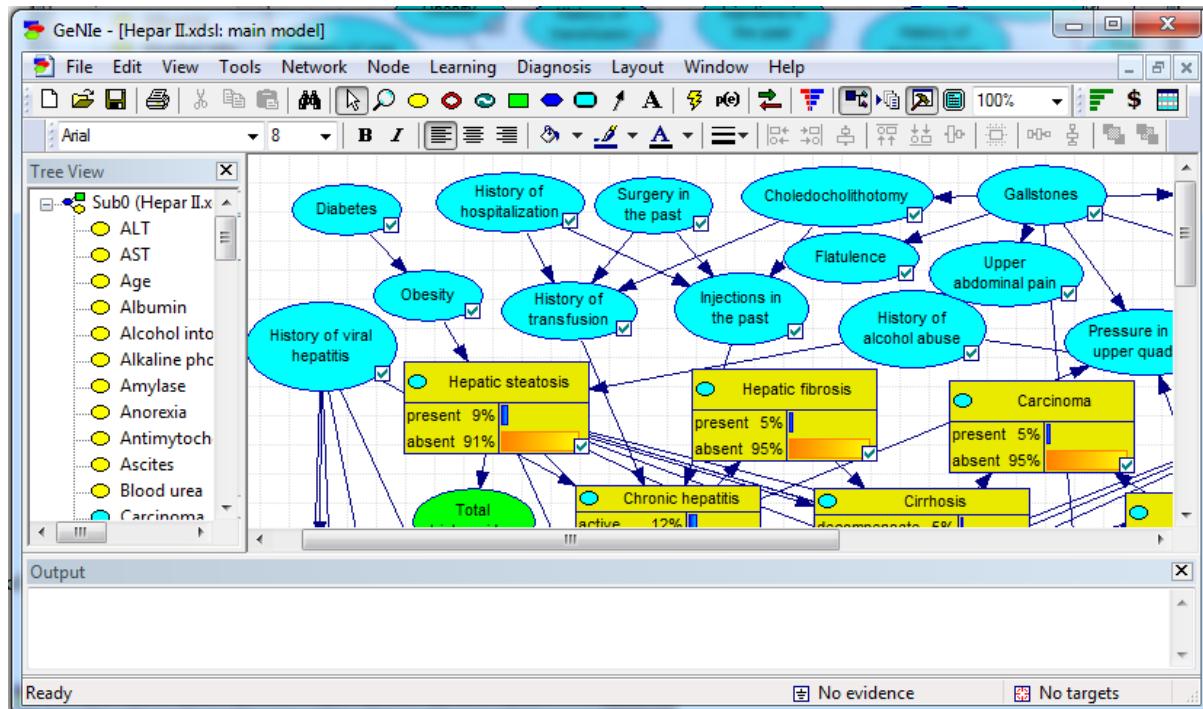
Keyboard shortcuts

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](#)⁶⁰ and the [Tree View](#)⁷³.

When diagnostic features of GeNIE are enabled, two additional views become available, these are the [Cost Graph View](#)³³⁴, and the [Spreadsheet View](#)³¹⁷. Documentation for the [Spreadsheet View](#)³¹⁷ is located in the [Support for Diagnosis](#)³⁰³ section.

[Status Bar](#)⁷⁶ displays the number of evidence and target nodes that are set for the active model.

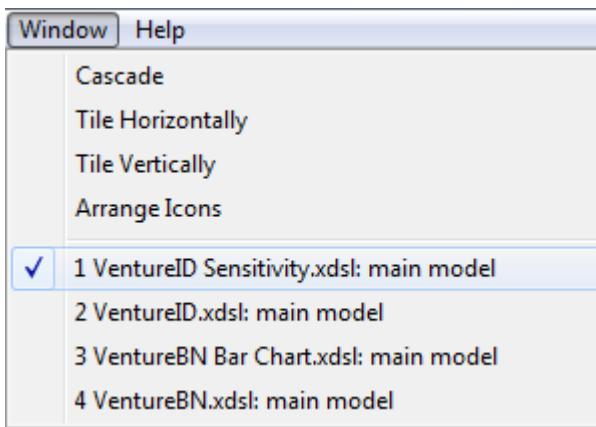
[Case Manager](#)⁷⁸ allows users to create cases for different combinations of observed evidence.

The [Output Window](#)⁸² 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](#)⁵⁶ 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](#)⁶⁰ and the [Tree View](#)⁷³.

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.

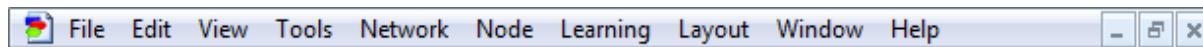
5.2.2 The menu bar

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:

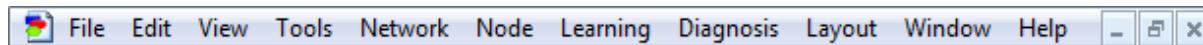
- (1) when no network is open,



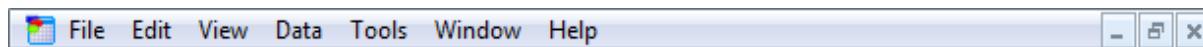
- (2) when a network is open in the workspace,



- (3) when a network is open and the diagnostic features are enabled,



- and (4) when a data file is open.



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 creating a new model, opening, saving, closing, and printing a model.

Edit Menu has commands for cutting, copying and pasting elements of the model, searching for an element and selecting multiple 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 commands 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 for displaying network properties, updating the model, setting the number of samples, clearing target nodes and evidence, selecting the belief updating algorithm.

Node Menu has commands for displaying Node properties, changing node type, setting evidence, decision, controlling node value, setting *Targets*, clearing evidence, selecting view type of the node, and locating relations of the node.

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.

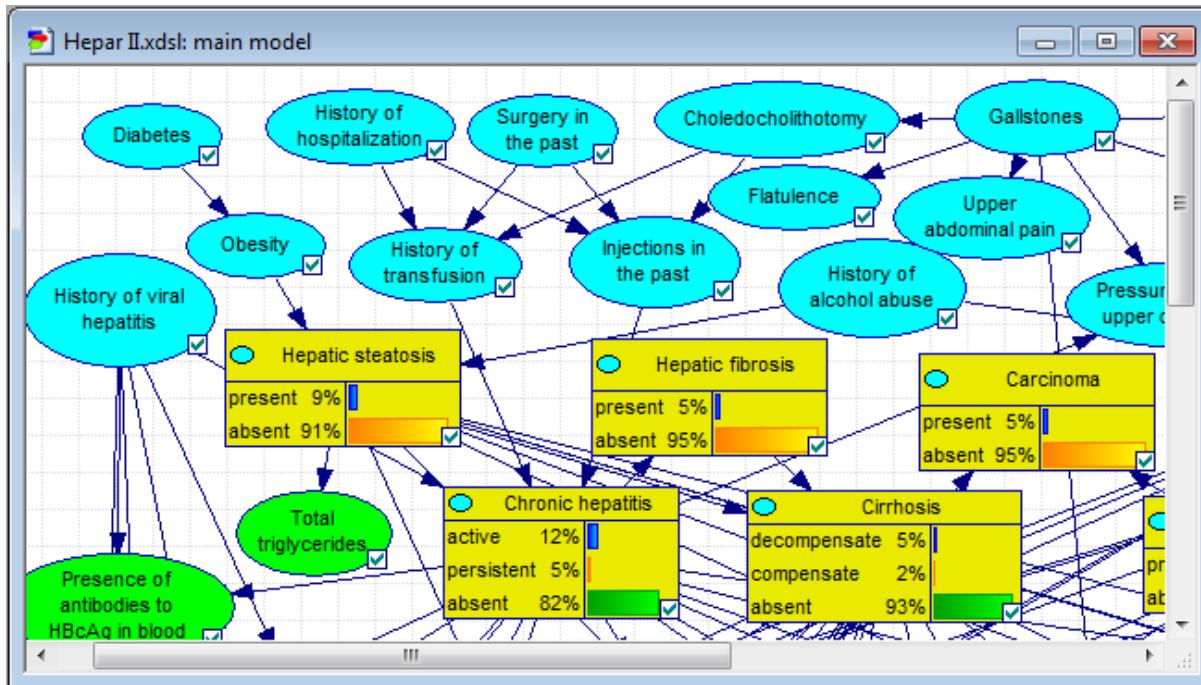
Help Menu has commands to display online help and modify help settings.

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

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 a separate section.

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 GeNle [submodels](#)¹⁰⁵ to learn more about it.

The *Layout Menu* and buttons on the [Format Toolbar](#)¹⁷⁷ 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](#)¹⁷⁷ 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](#)¹⁷⁷ for more information.

Opening a Graph View window

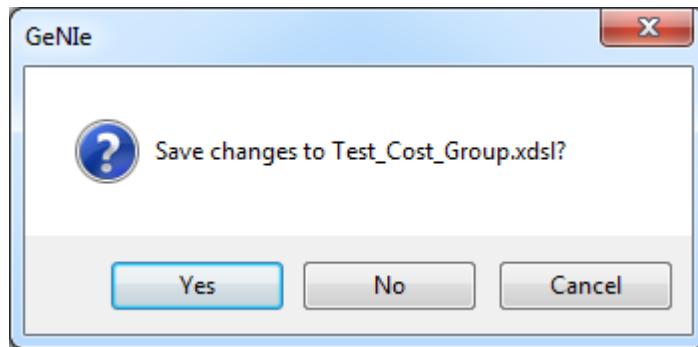
The graph view window is always open by default whenever a new model is opened or created. It is a large sheet with variables placed at user-designated locations. You can select the *Open* option from the [File Menu](#)¹⁹³ to open a saved network file. You can create a new network by selecting the *New* option from the [File Menu](#)¹⁹³.

Closing a Graph View window

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

- By clicking on the *Close* (X) button at the top right of the *Graph View* window.
- By selecting the *Close* option in the [File Menu](#)¹⁹³.
- 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:

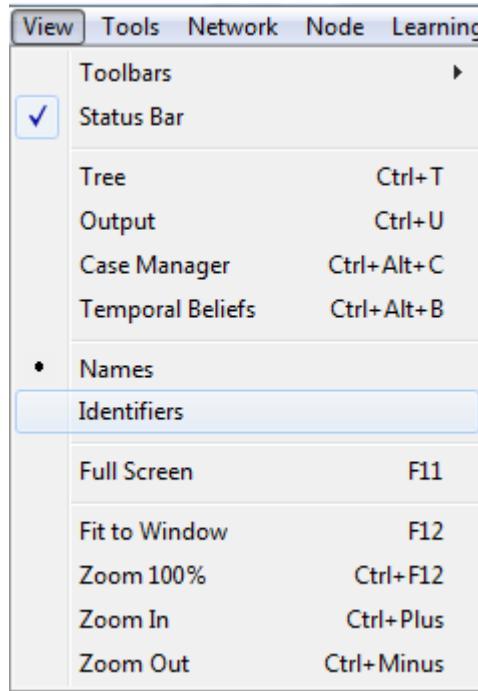
Each network is opened in a separate graph view sheet in the workspace. Double clicking on a clear area of the graph view sheet will open the [Network Property Sheet](#)¹²³. Right clicking on any clear area of the graph view sheet will display the *Network Pop-up* menu, which can be used to modify various 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 [Tool Menu](#)¹⁷⁶ or clicking on the appropriate button on the [Standard Toolbar](#)¹⁷⁶.

Double clicking on any node will open its [Node Properties Sheet](#)¹³⁸. 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:

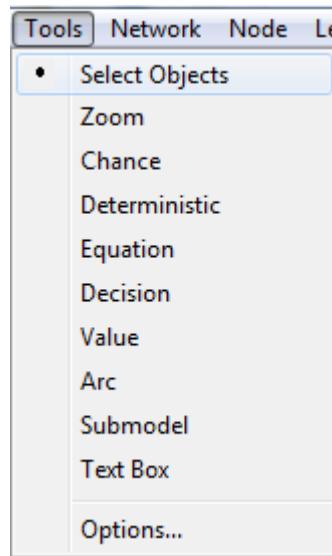
Double clicking on any submodel in the *Graph View* will open a *Graph View* for that submodel. You can go back to the main network by either minimizing or closing the submodel using the buttons on the top right of the submodel window. Right clicking on the submodel will display the *Submodel Pop-up 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](#)¹⁷⁶ 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.

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](#)¹² and [Building an influence diagram](#)²⁸¹.

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**

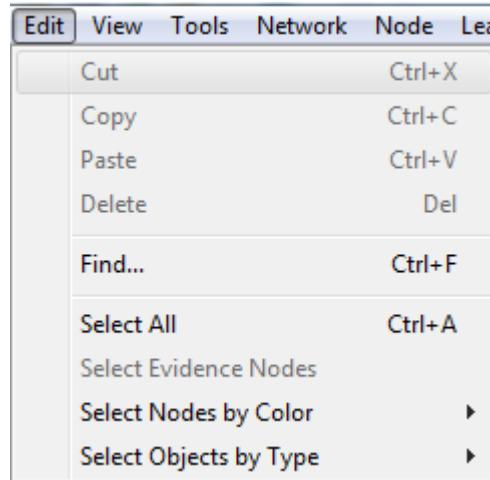
You can select a group of nodes by clicking on an empty area of the *Graph View* and dragging a selection rectangle in any direction that you wish. Any node completely within the rectangle will be selected.

- **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 *CTRL* key while clicking on them. This selection process acts as a toggle, i.e., nodes that are currently selected will be de-selected.

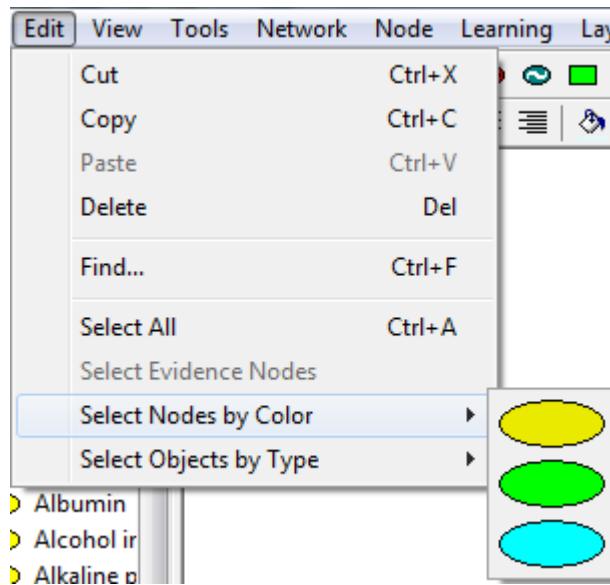
- **Group selection**

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*.



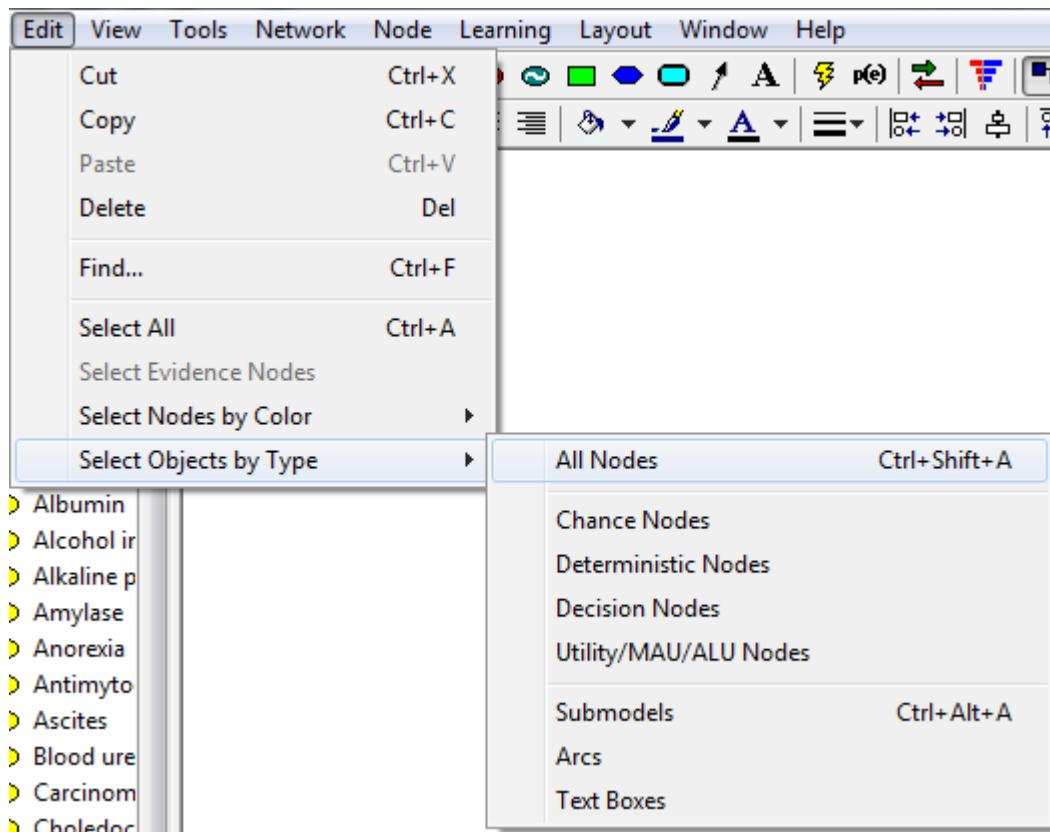
- **Select Nodes by Color sub-menu**

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 to another location by dragging it. 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.

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, but you should 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 stores data simultaneously in three formats:

1. Native GeNIE format for cut, copy, and paste operations between models
2. Standard text, for example names of selected nodes and comments
3. Standard bitmap, which is used for selected objects that have a graphical component

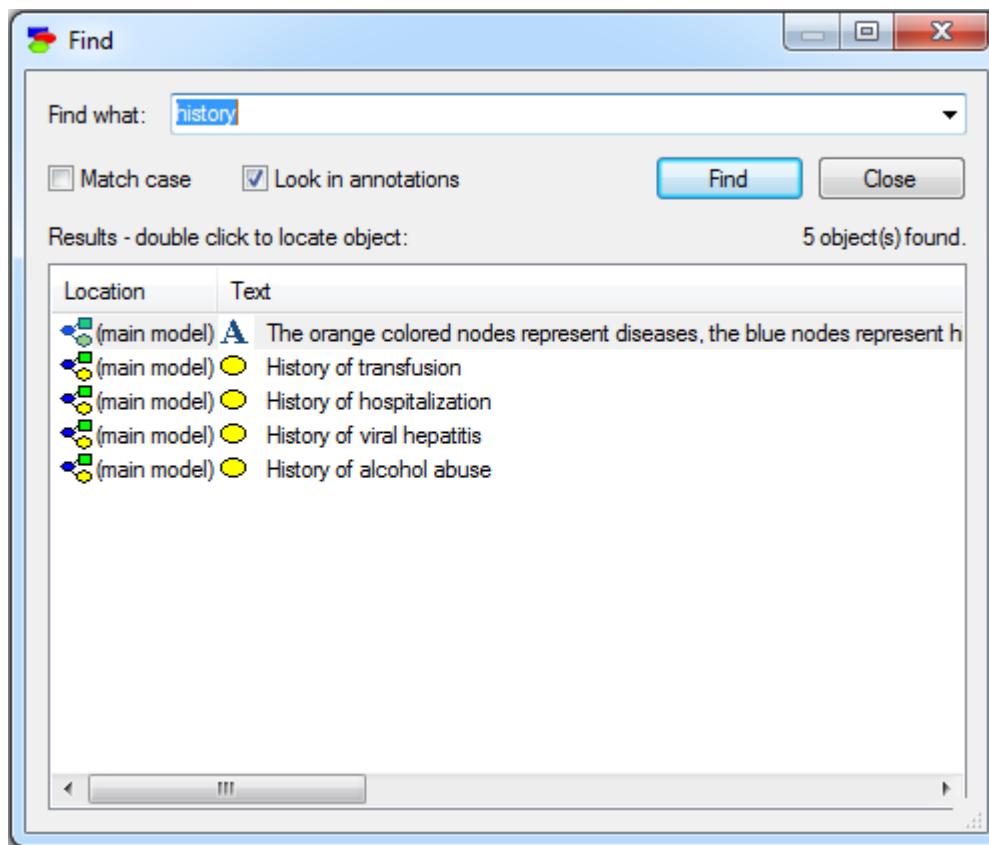
Each time you invoke *Copy* or *Cut*, data in all three formats are sent to the *Clipboard*. When pasting into a different application, all you need to do is select *Paste special*

from that application's *Edit* menu (e.g., in *Word* or *PowerPoint*). This should bring the dialog box with all format names. Plain Paste command does different things in different programs - it pastes data as text into Word by default but may paste a bitmap image in *Paint*.

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* in *Word*. If you use *Photo Editor* or any graphics editing program, *Paste* will paste the model image by default.

Text-based search for model elements (Find command)

 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

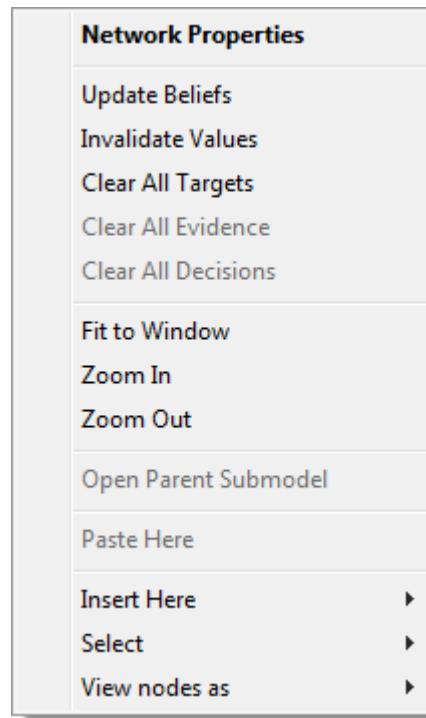


The *Find* dialog allows for finding the text string specified in the *Find what* box in the names and identifiers of all elements of the models and submodels. It will also search

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](#)⁶⁰, 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](#)¹²³ 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](#)¹⁷⁶. 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](#)²⁰⁰ section.

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*.

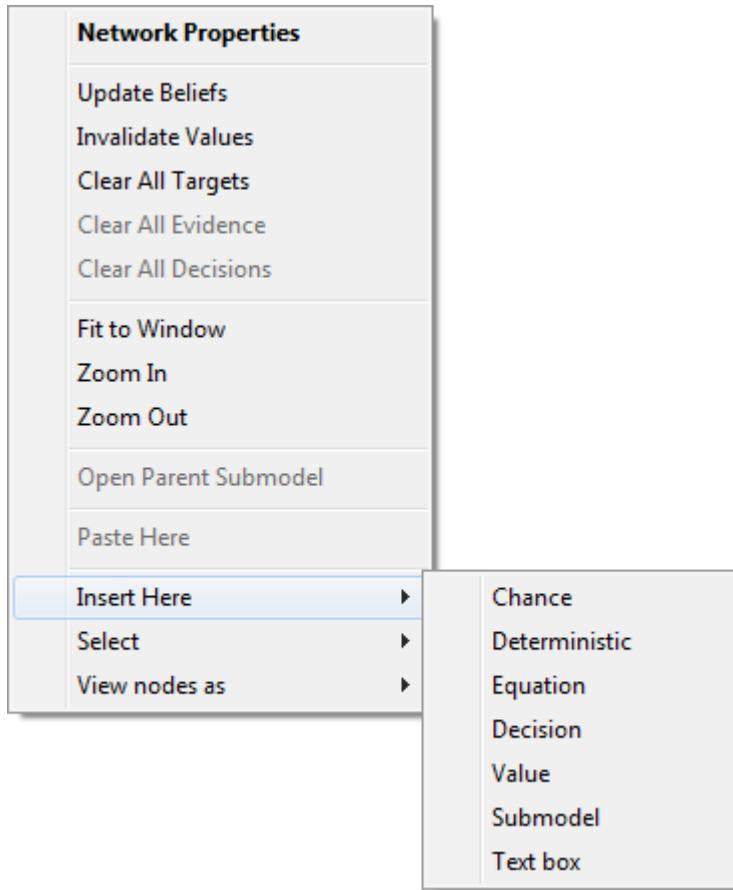
Zoom In zooms into the network. Every application of this command increases the zoom by 25%. The current zoom percentage is displayed on the top right of the [Standard Toolbar](#)¹⁷⁶. A similar effect can be obtained by using the zoom tool from the [Standard Toolbar](#)¹⁷⁶ or the [Tool Menu](#)¹⁷⁶.

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](#)¹⁷⁶.

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.

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.

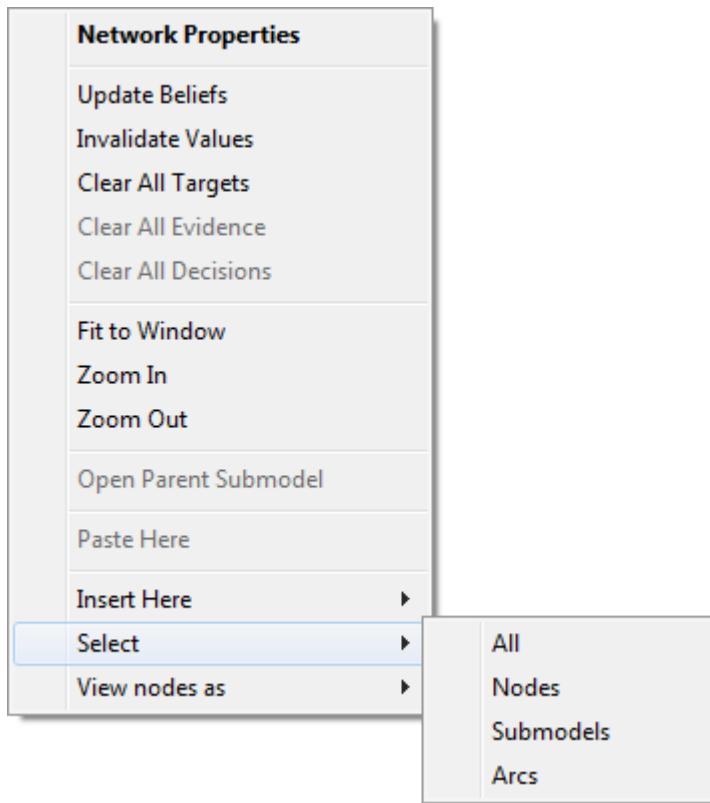
Insert Here submenu



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.

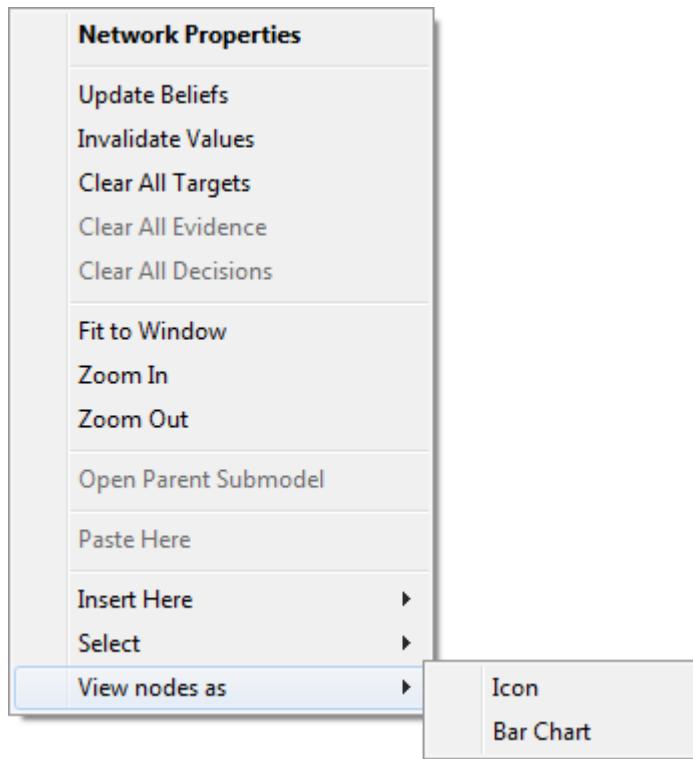
Select submenu

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*), and *Arcs* (select all arcs in the *Graph View*).

View nodes as submenu

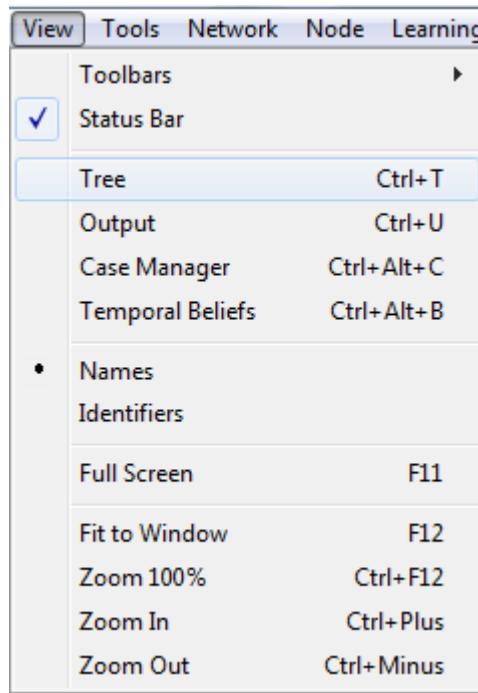


The *View nodes as* submenu is used to select how the nodes should be displayed in the *Graph View*. It is similar to the *View As* submenu in the [Node](#)²⁰⁷ menu but it applies to all nodes rather than to the selected nodes.

5.2.4 Tree view

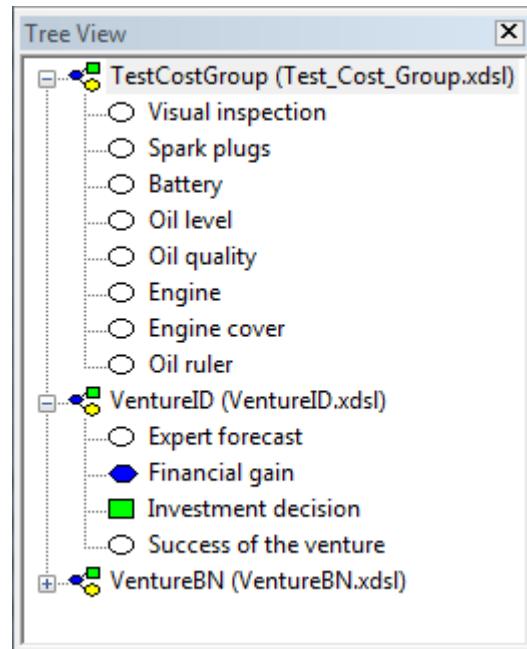
GeNIE provides an alternative method of model navigation known as *Tree View*. The *Tree View* in GeNIE is very similar to Windows tree view. It shows a hierarchical, alphabetically sorted list of all networks currently open, and all the nodes in the network. Most operations available in the [Graph View](#)⁶⁰ can be also performed in the *Tree View*. Whatever changes are made in the *Tree View*, they are reflected immediately in the [Graph View](#)⁶⁰. The *Tree View* can be also used to navigate in the [Graph View](#)⁶⁰, 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 submodel¹⁰⁵ can be expanded or collapsed by clicking on  or  beside its name. *VentureBN* is not expanded. (hence, 

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 perform these operations.

Moving nodes between networks and submodels

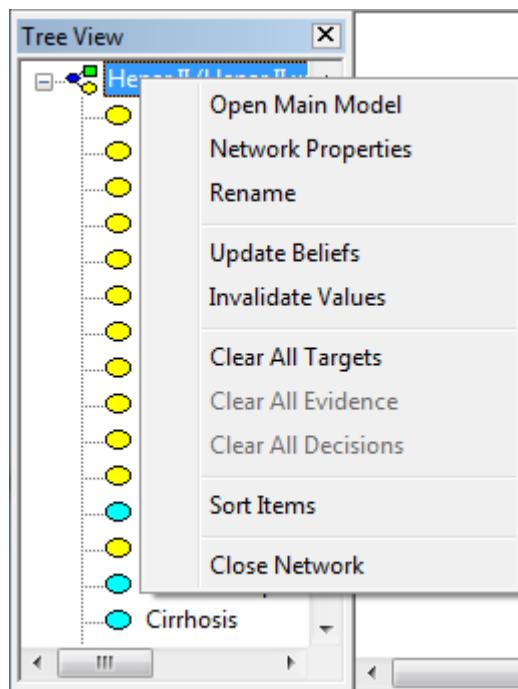
You can select any number of nodes and submodels in the *Tree View* and drag and drop them into any model or submodel in the *Tree View* or the *Graph View*. You can perform the drag and drop operations between different networks.

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](#)⁶⁰.

Open Main Model opens the main network in the [Graph View](#)⁶⁰.

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.

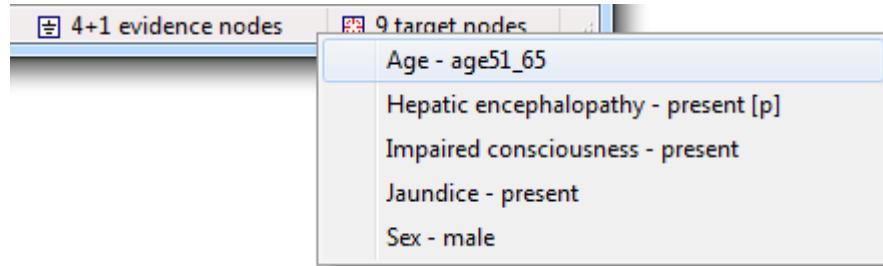
Close Network closes the network file. If any changes have been made to the network, then GeNIE will warn you that you may lose the changes. You can also close the network by clicking on the *Close* button at the top right of the [Graph View](#)⁶⁰ window or selecting *Close* option from the [File Menu](#)¹⁹³.

5.2.5 Status bar

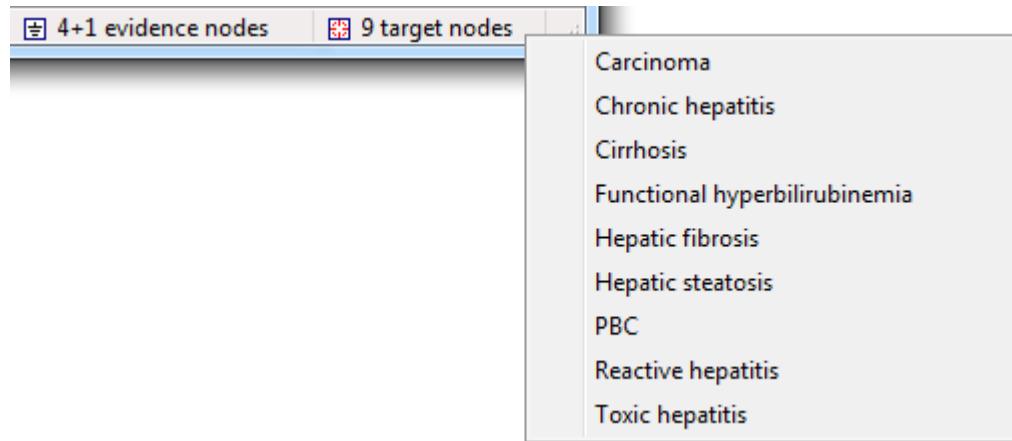
The *Status Bar* is a horizontal bar located at the very bottom of the main GeNIE window. The *Status Bar* shows a short description of the command to be executed by the selected menu item or a tool on a toolbar on the left side and lists the number of evidence nodes and target nodes present in the network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



If there are any [evidence](#)²⁵⁶ or target nodes (see [Relevance reasoning](#)²⁰⁴ 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:

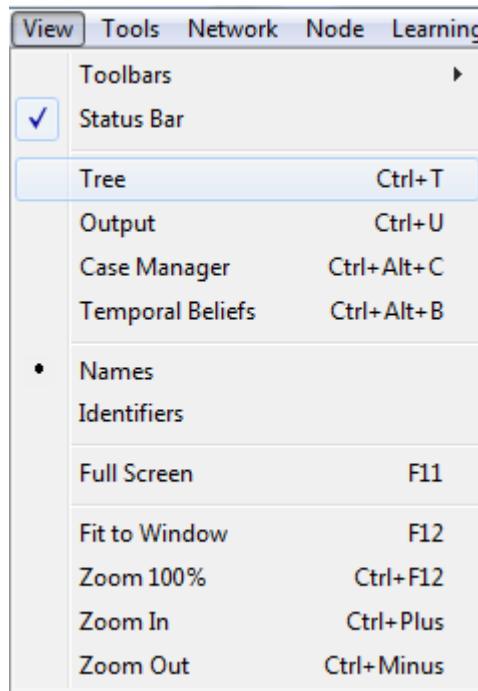


So can the list of target nodes:



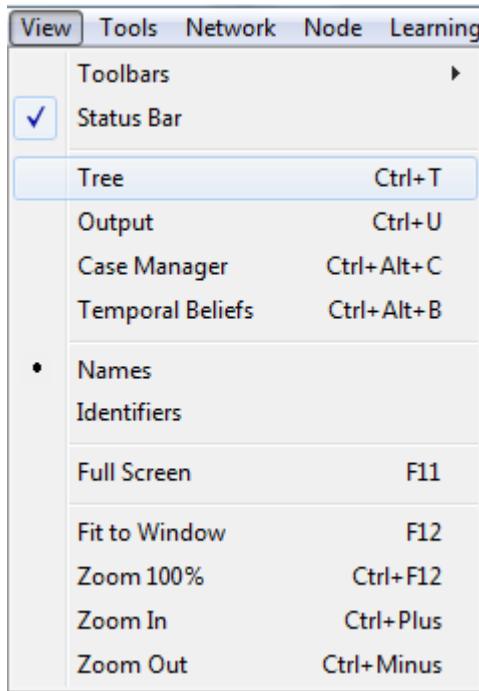
Any propagated evidence will be listed with a suffix *[p]*. To display only real evidence, right-click on the status bar when holding *CTRL* key. To display only propagated evidence, right-click on the status bar when holding the *SHIFT* key. Clicking a node name on the list of evidence or target nodes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

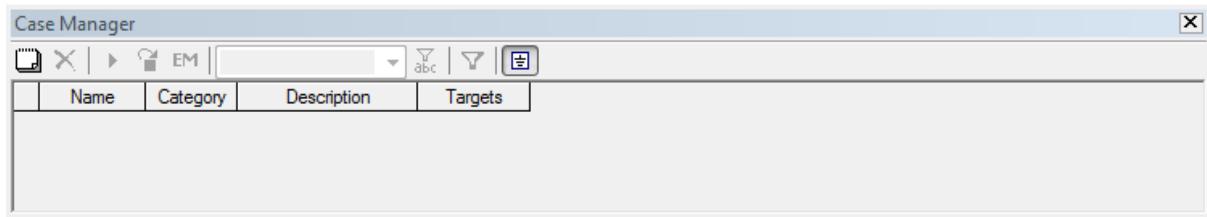


5.2.6 Case manager

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*:

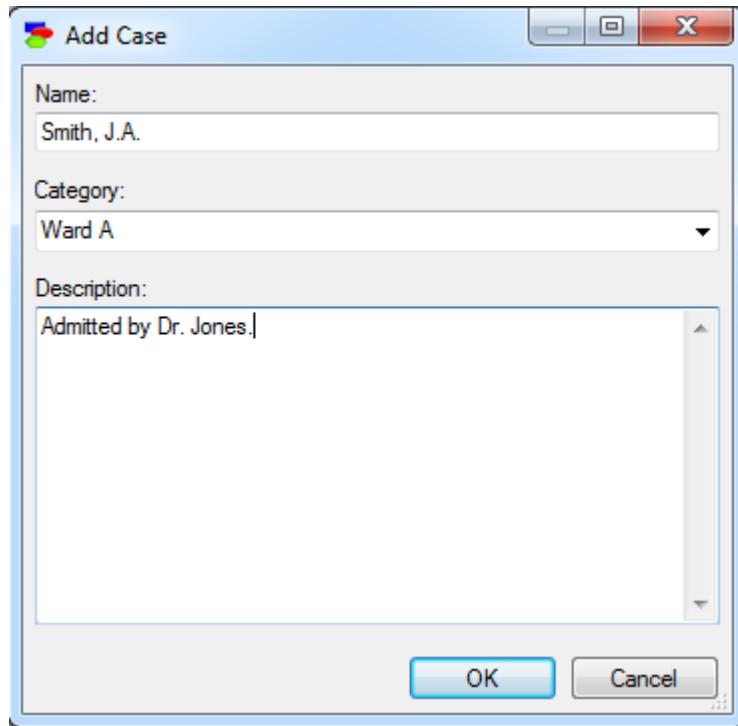


Case Manager window looks initially as follows:



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 (New icon). This results in the following dialog, which allows for entering case details.



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

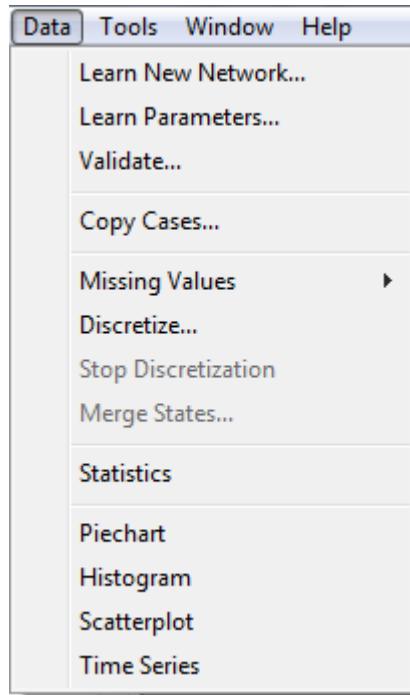
Case Manager									
	Name	Category	Description	Targets	Age	Hepatic encephalopathy	Impaired consciousness	Jaundice	Sex
▶	Smith, J.A.	Ward A	Admitted by Dr. Jones.	9: Toxic ...	age51_65	present	present	present	male

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.

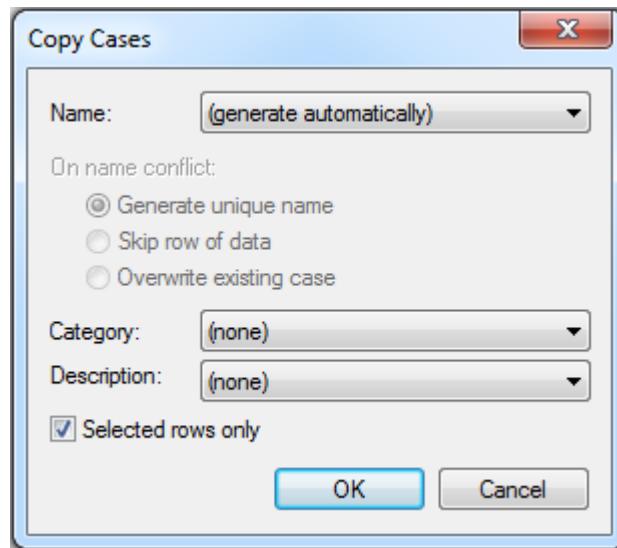
Case Manager									
	Name	Category	Description	Targets	Age	Hepatic enc...	Impaired co...	Jaundice	Sex
Smith, J.A.	Ward A	Admitted by Dr. Jones.	9: Toxic ...	age51_65	present	present	present	male	
Miller, N.	Ward B	Emergency room ad...	9: Toxic ...	age31_50		absent	absent	female	
▶ Humpfrey, ...	Ward A	Mild bowel complaints.	9: Toxic ...	age31_50	absent			male	

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](#)³⁵¹.



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 (). Pressing the *Filter cases by text* () button displays cases that match the string. *Show filtered cases* () button toggles between all cases and cases matching the filtering text.

5.2.7 Output window

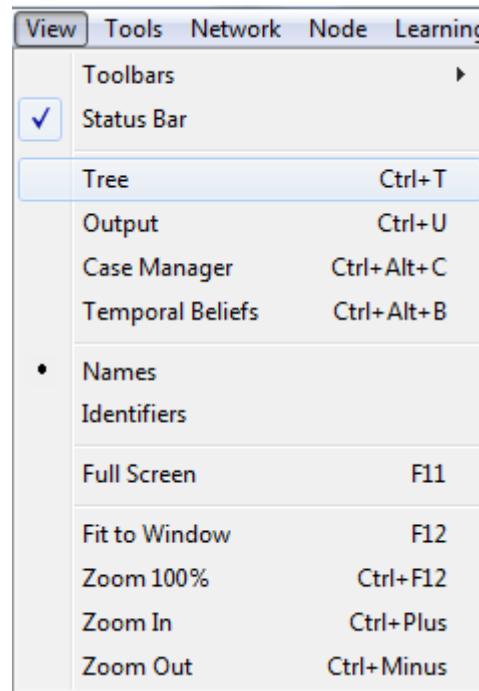
GeNIE includes a *Output Window* that is used for notifying the user about possible problems with the model or program errors. The *Output Window* is usually shown in the bottom part of the screen, but can be moved to any location by the user.



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.

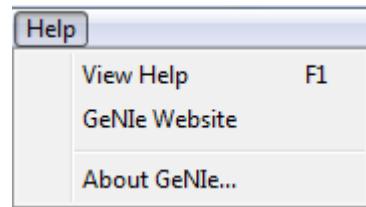


The *Output* window can be hidden or made visible by the user by changing the *Output* flag in the *View Menu* (shortcut *CTRL-U*).



5.2.8 Help menu

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

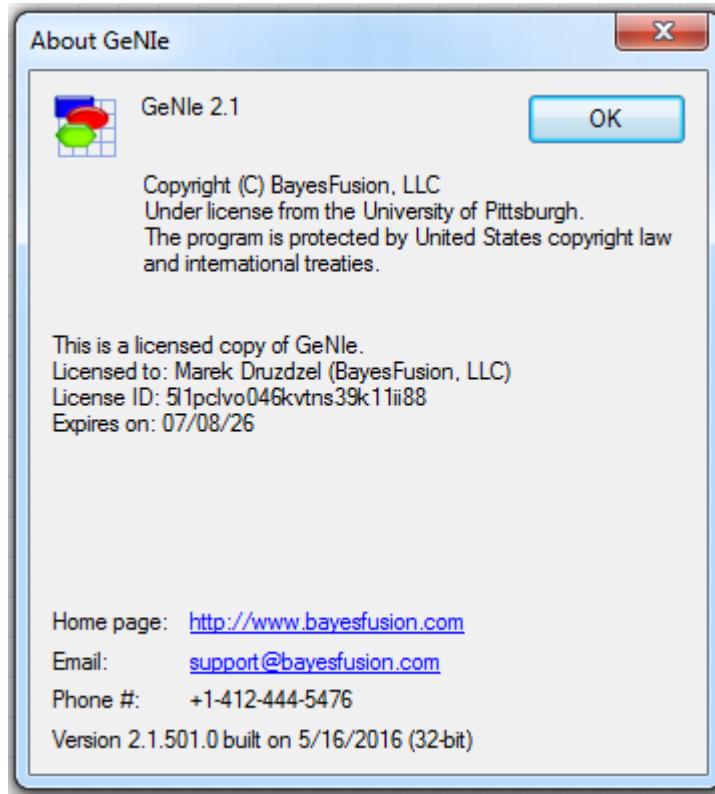


It contains three commands:

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
<http://www.bayesfusion.com/>

About GeNIE shows the following simple window:



It displays a copyright notice, license holder's name and institution, license number, and its expiration date. Version number of your build of GeNIE can be found on the very bottom.

5.3 Components of GeNIE models

5.3.1 Node types

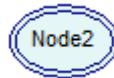
GeNIE supports the following node types:

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](#)⁸⁷ section for more information.

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).



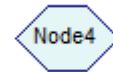
There is a popular error, made by novices in the area of probabilistic graphical models, to equip models with deterministic parentless nodes. This is a bad practice that is perhaps not changing the numerical properties of the model but it serves no purpose, obscure the picture, and make the model larger, 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](#)¹¹⁹ or [annotations](#)¹²⁰. 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](#)²⁹⁶ (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 GeNle are always discrete and specified by a list of possible states/actions.

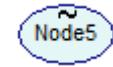


Value nodes (also called [Utility](#)⁴⁴ nodes), drawn as hexagons, denote variables that contain information about the decision maker's goals and objectives. They express the decision maker's preferences over the outcomes over their direct predecessors.

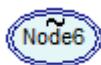


There are two fundamental types of value nodes: *Utility* and *Multi-Attribute Utility*. The latter include a special case of *Additive Linear Utility* (ALU) functions. There is no distinction between the two in the graph view, as they differ only in the way they specify the utility functions. *Utility* 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](#)¹⁰⁰ section for more information.

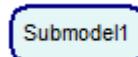
Equation nodes, which are relatives of chance nodes, drawn as ovals with a wave symbol, denoting that they can take continuous values. Instead of a conditional probability distribution table, which describes the interaction of a discrete node with its parents, an equation node contains an equation that 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](#)^[12] and [Building an influence diagram](#)^[281].

5.3.2 Canonical models

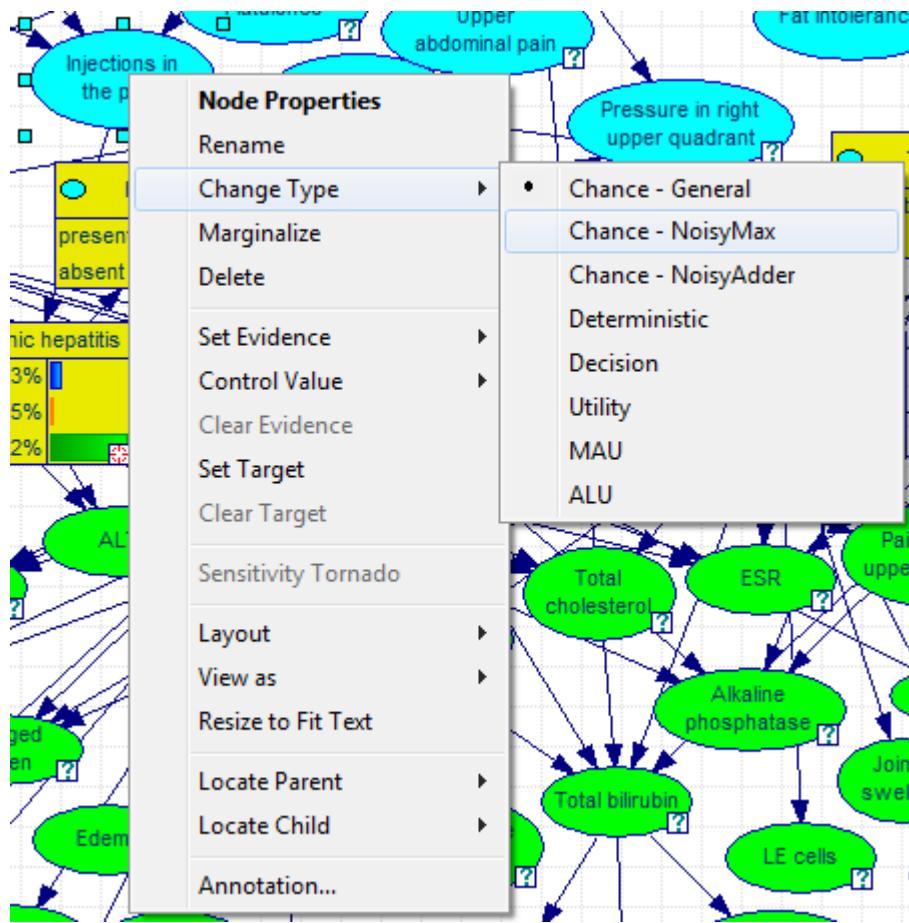
Canonical probabilistic nodes, such as *Noisy-MAX/OR*, *Noisy-MIN/AND*, and *Noisy-Average* gates, implemented by GeNle, are convenient knowledge engineering tools widely used in practical applications. In case of a general *Chance* binary node with n binary parents, the user has to specify $2n$ parameters, 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.

This section gives a brief introduction to *Noisy-OR*, *Noisy-AND*, and *Noisy-Average* gates and assumes a basic knowledge of the principles applied in these gates.

Noisy-OR/MAX and Noisy-AND/MIN gates

These gates were 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). GeNle implementation allows using both parameterizations, proposed by Diez (1993) and Henrion (1989) respectively.

Both, *Noisy-OR* and *Noisy-AND* types of nodes can be modeled using *Noisy-MAX* nodes implemented in GeNle. To change the type of a node from general to *Noisy-MAX*, right-click on a node, select *Change Type* and then *NoisyMAX*.



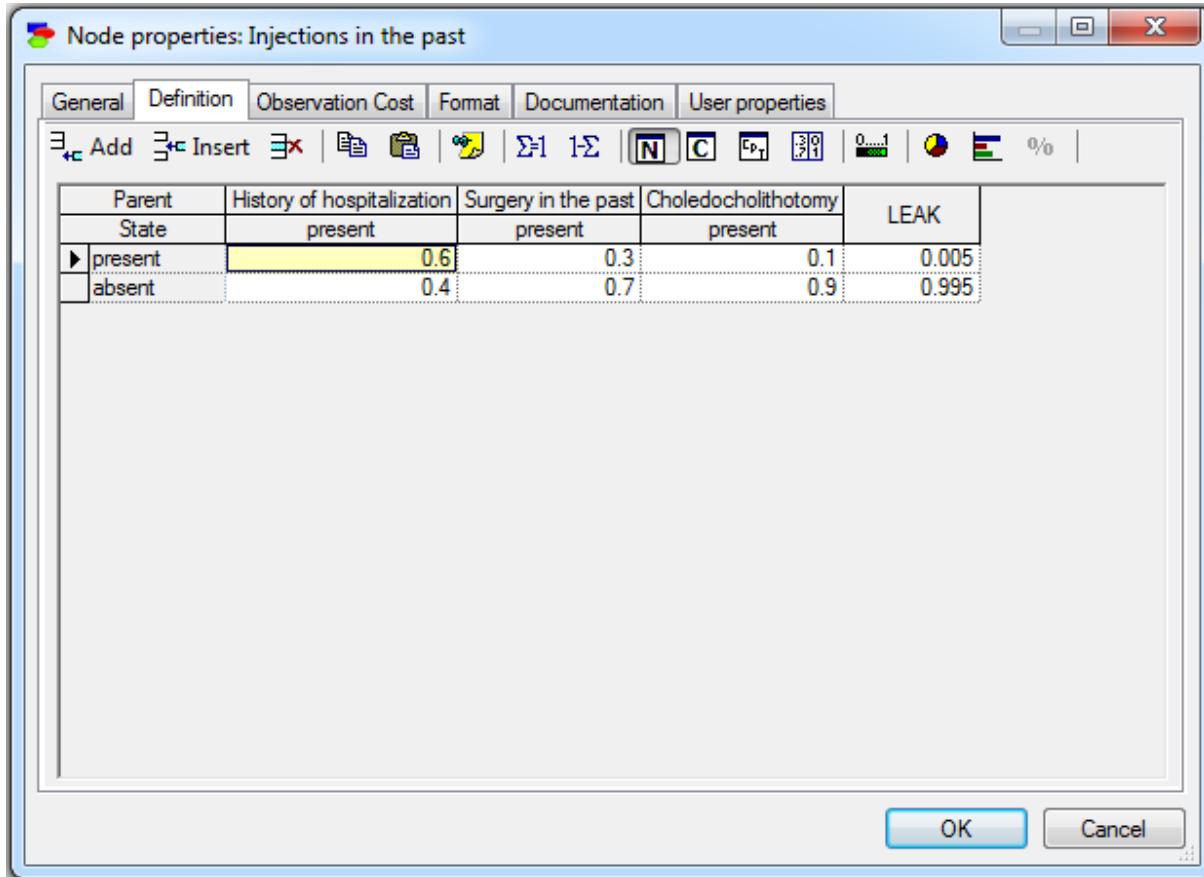
At the moment, GeNIE does not take any computational advantage from *Noisy-MAX* nodes in its reasoning algorithms and uses them purely as a useful knowledge engineering tool. This we plan to change in the future, so using *Noisy-MAX* nodes is a good idea from the point of view of both, ease of model building and future computational efficiency.

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, the *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.

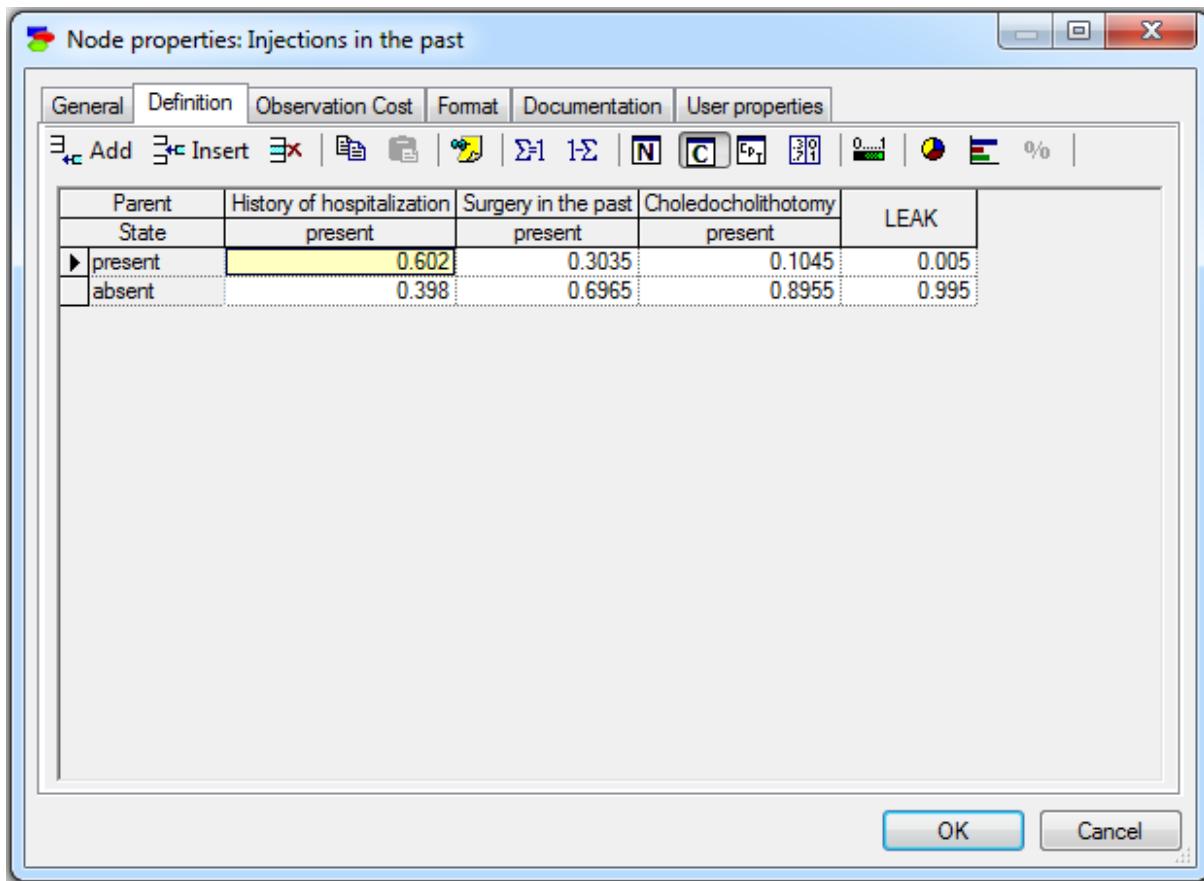
There are two different (but mathematically equivalent) parameterizations of the *Noisy-OR/MAX* gates. They are often referred to by names of the researchers who proposed them, Henrion (1989) and Diez (1993). The two parameterizations are in the forms of $P(Y|X_1)$ and $P(Y|\sim X_1, \dots, X_i, \dots, \sim X_n, X_L)$. 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.

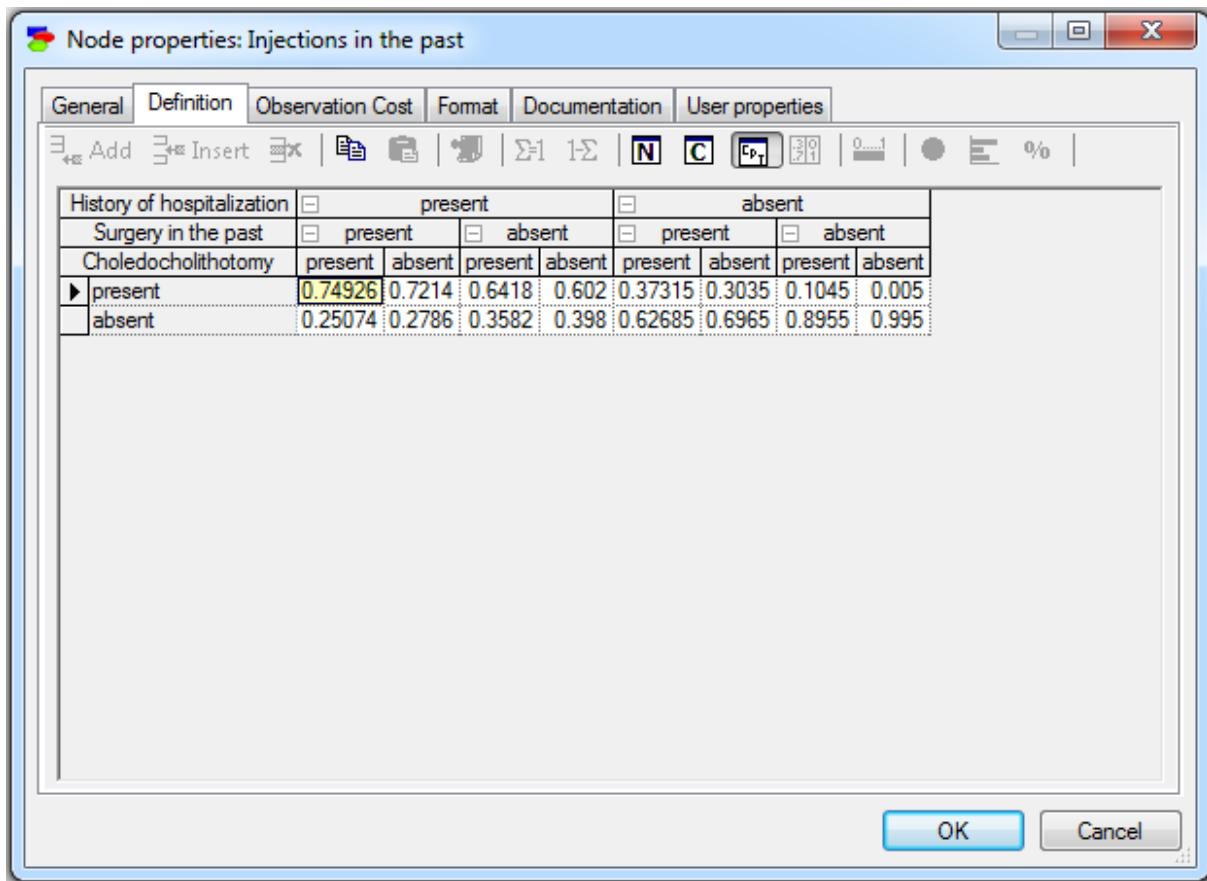
To examine the node definition of a Noisy-Max gate, open the *Definition* tab among the node's *Node Properties*.



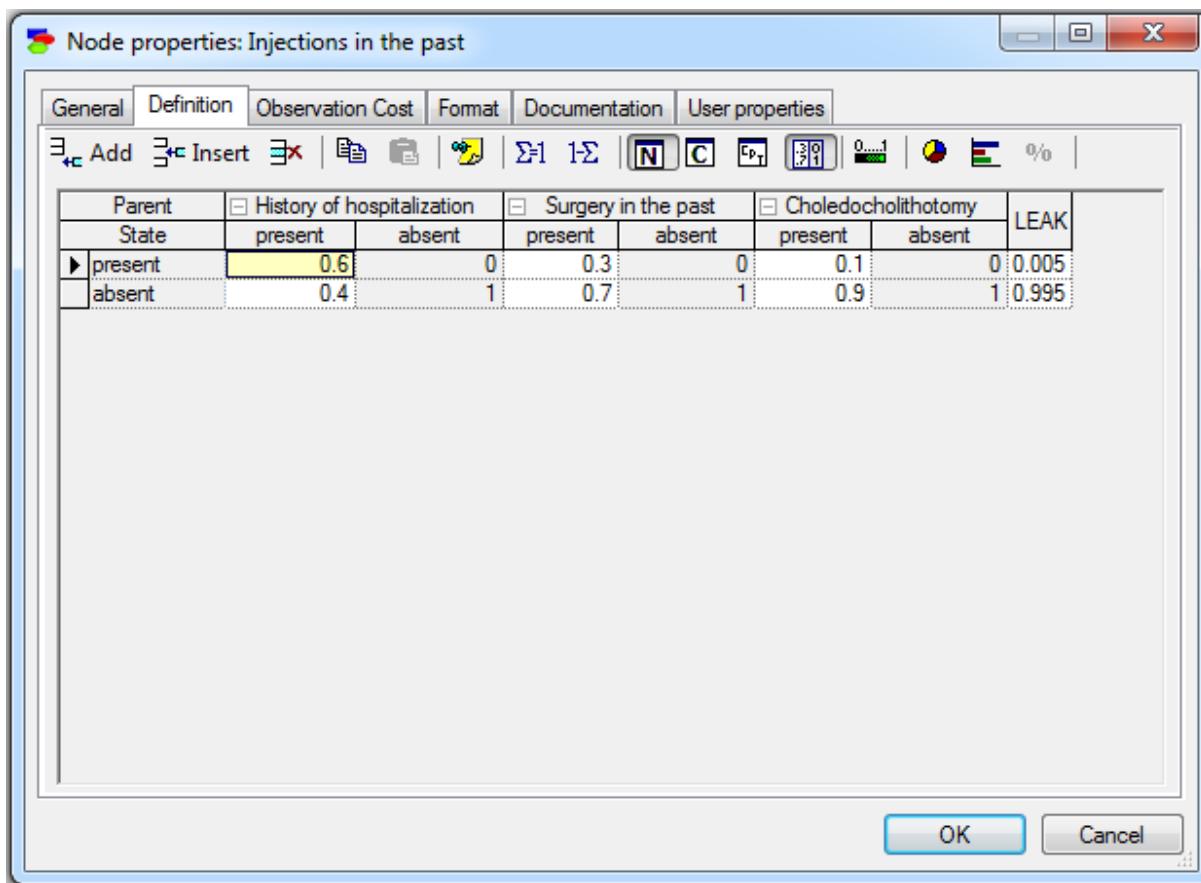
The default definition format for Noisy-MAX nodes in GeNIE is *net*. Net probabilities in a *Noisy-MAX* node for a parent X_i express the probability of the effect happening when the cause X_i is present and none of the other causes of the effect, whether modeled or unmodeled is present. It is possible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



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



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.

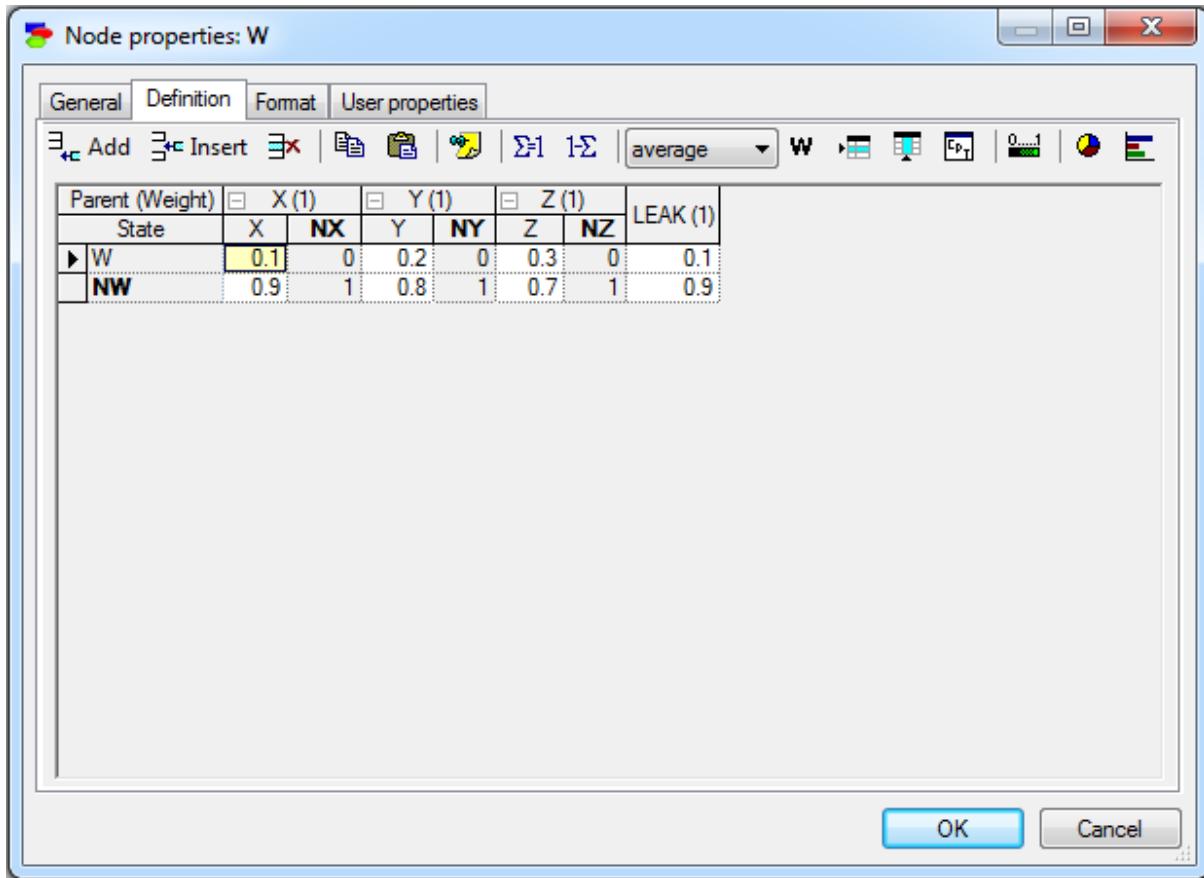


The regions shown with a gray background denote constraints on the *Noisy-MAX* parametrization - the last state of each parent is assumed to have distribution {0.0, 0.0, ..., 0.0, 1.0}. States of the node and of its parents can be ordered, which amounts to a negation. *Noisy-MAX* implementation in GeNle 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 and 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.

Noisy-Adder gates

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*.

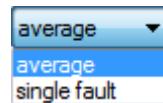


There are two additional buttons on the *Definition* tab that allow for specifying which of the states of the current node and the parents are the distinguished states.

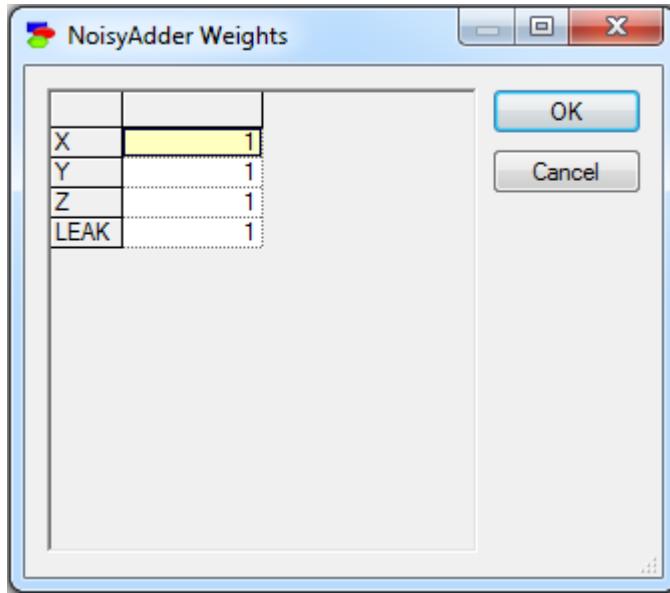
The *Set distinguished state of this node* (button) allows for choosing the distinguished state of the current node. The state with the cursor will become the distinguished state after clicking the button.

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

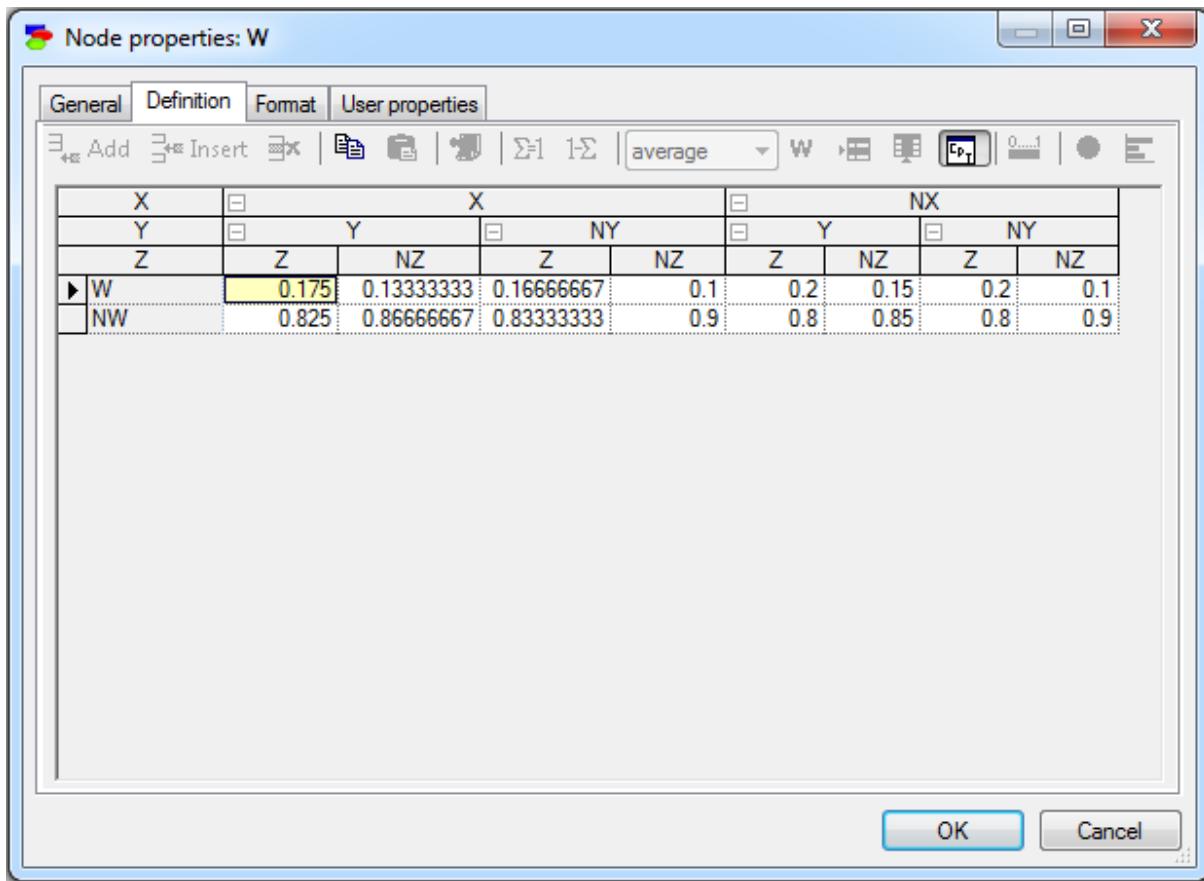


The *average*-type *Noisy-Adder* has weights associated with each of the parents. These weights can be edited by invoking the *NoisyAdder Weights* dialog through the *Set weights* () button.

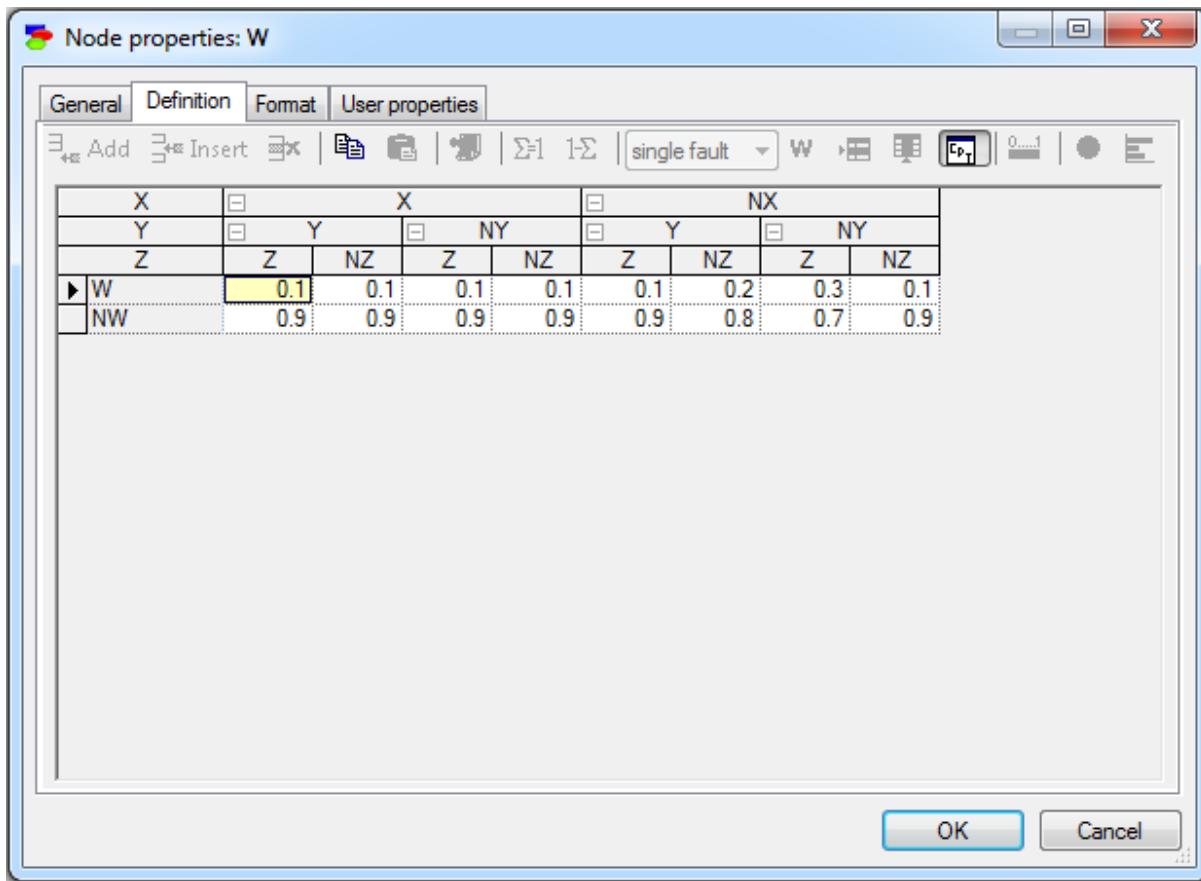


The weight are all equal to one by default and they are displayed in the header of the definition table.

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

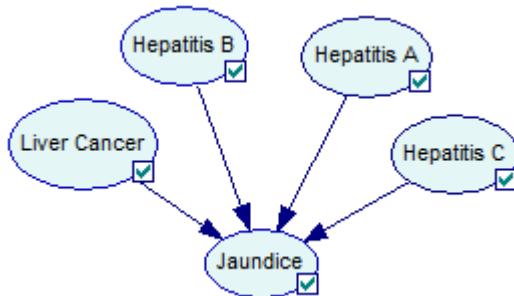


The *single fault* type *Noisy-Adder* node assumes that only one of the parent nodes will be in the non-distinguished state. If none of the modeled causes of the effect is active, the *Leak* node will be active. The CPT corresponding to the definition above looks as follows



Noisy-MAX example

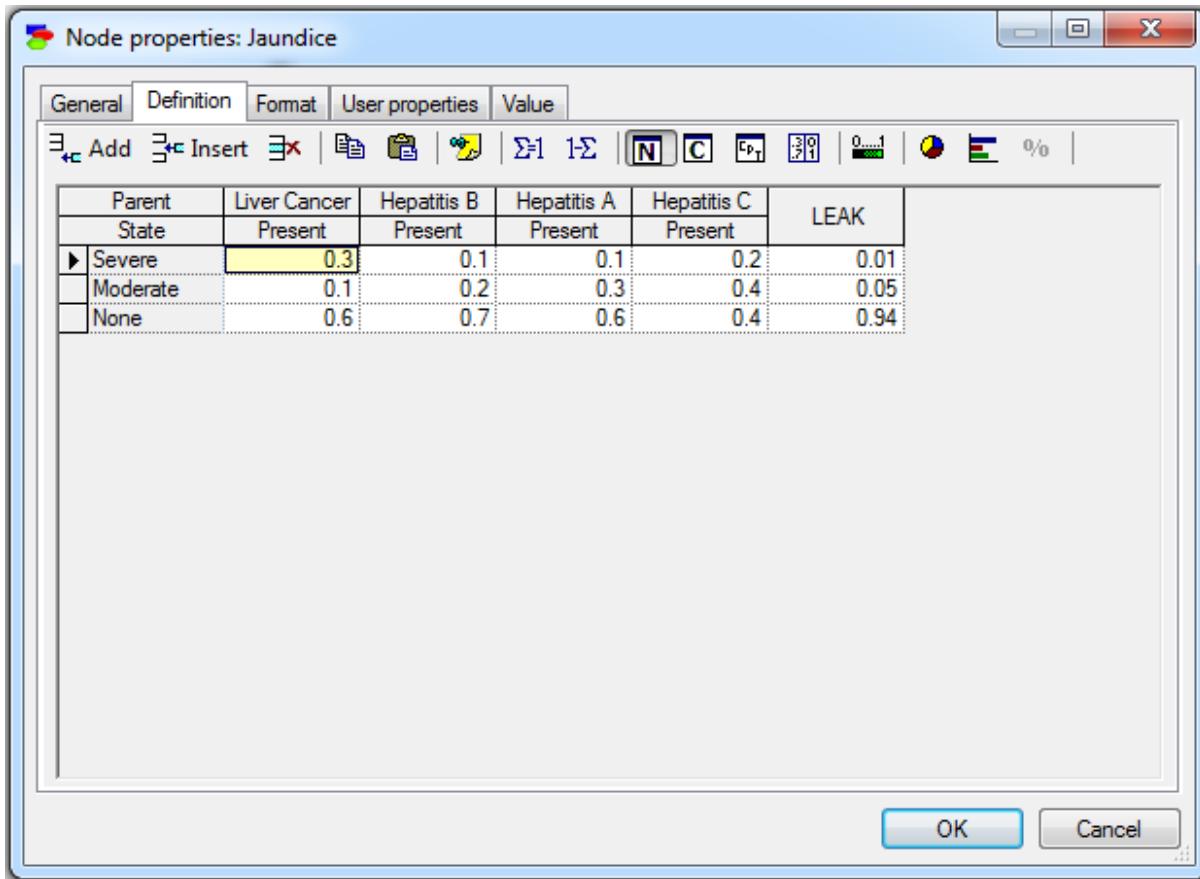
Consider the following Bayesian network modeling the interaction of various liver disorders (*Liver Cancer*, *Hepatitis A*, *Hepatitis B* and *Hepatitis C*) in producing *Jaundice*. We start by creating the structure of the network as follows.



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.



The probability numbers in the table are specified for each non-distinguished 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 the current causes and only the current cause is present. The above table encodes the belief that 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 that *Jaundice* will be *Severe* is 0.3. When none of the four explicitly modeled causes of *Jaundice* are present, there is still 0.01 chance for *Severe* and 0.05 chance of *Moderate Jaundice* (because of all possible unmodeled causes of *Jaundice*), as specified in the last (*LEAK*) column.

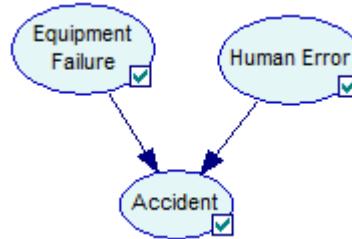
It is important that the outcomes of the node *Jaundice* are ordered from the highest to the lowest. The outcomes of the parent nodes are recommended should also be ordered from the highest to the lowest (in term of causal influence on the child node),

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.

The advantages of canonical gates become apparent when the number of parents of a node becomes large. Then the savings in terms of the number of probability elicitations may be dramatic. 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 (2005) summarize the theory behind the *Noisy-MAX* and other canonical gates.

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*. We start by creating the structure of the network as follows.

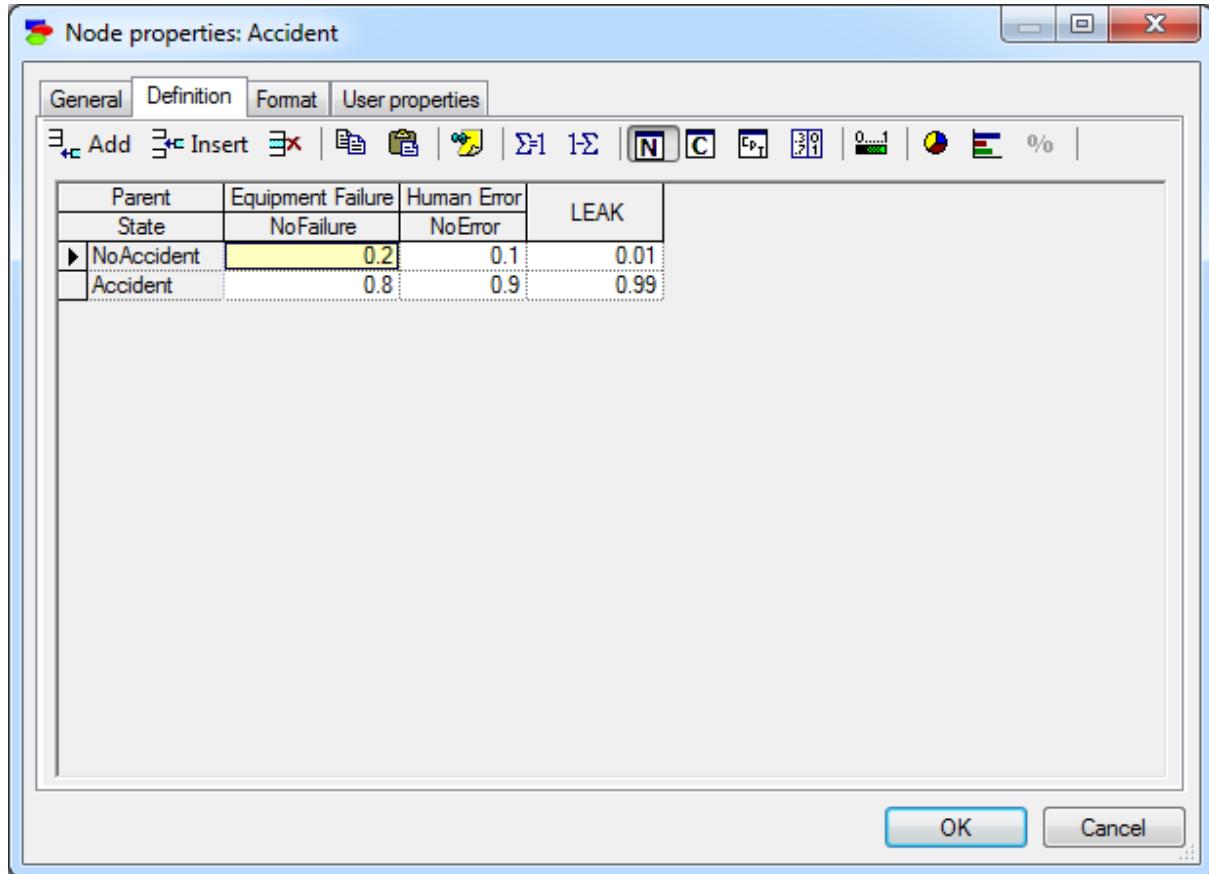


We create the node *Accident* as a general *Chance* node and subsequently change its type into a *Noisy-MAX* node. To this effect, we right click on the node, and choose *Change Type*, from the *Node Context Menu*, then *Chance - NoisyMax* and click *OK*. It can be reasonably assumed that each of the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

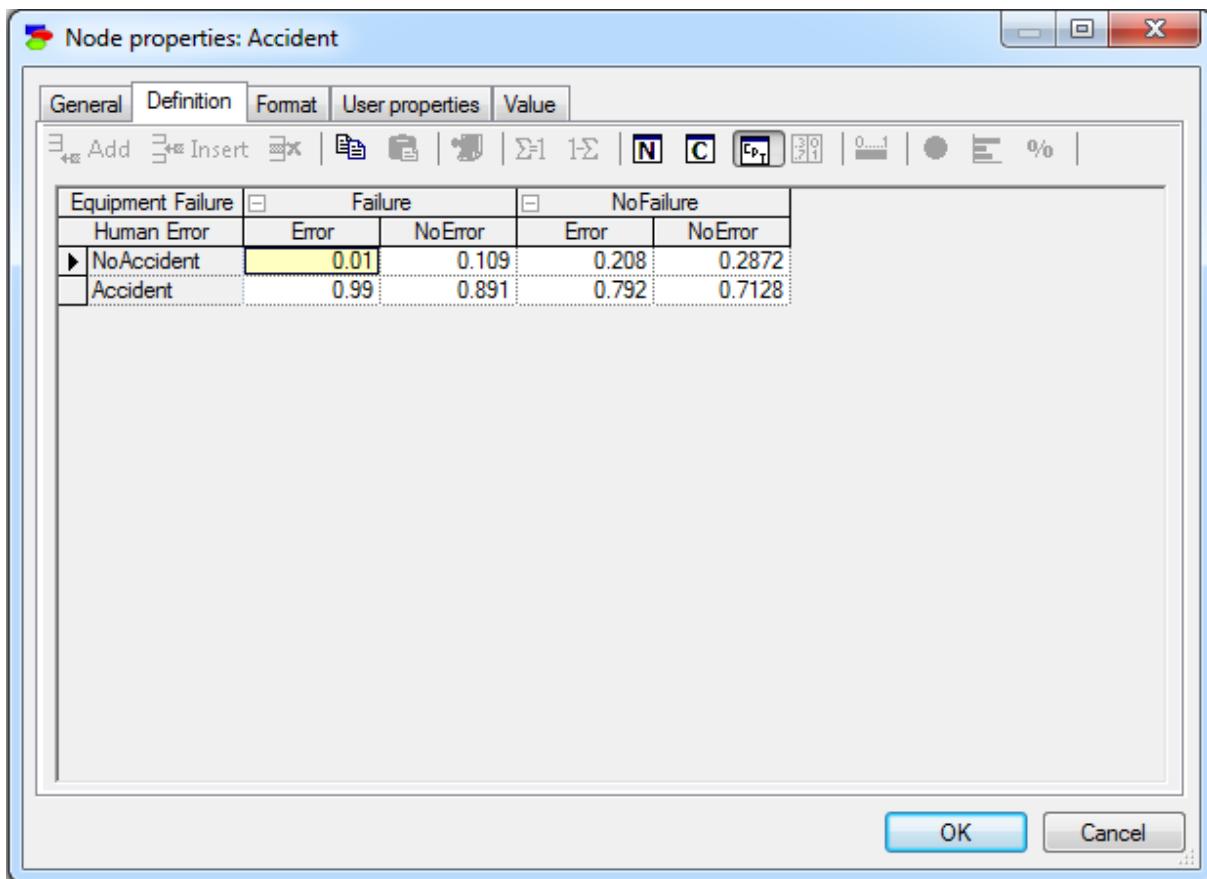
Noisy-MAX implementation in GeNIE does not require a separate implementation of *Noisy-AND/MIN* nodes. This is because a *Noisy-AND* node can be modeled by a *Noisy-OR* node and negation, based on DeMorgan's laws. Hence, having *Noisy-OR* model and the ability to negate inputs and output, one can implement *Noisy-AND* gate. Negation corresponds to reversing order of outcomes (or causal strengths), what can be easily done in GeNIE.

Noisy-AND nodes work in a similar way to *Noisy-OR* nodes, but their 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. 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*. Assume that with probability 0.8 the *Accident* will take place if *Equipment Failure*

does not happen but *Human Error* does. Similarly, assume that with probability 0.7 the *Accident* will take place if *Human Error* does not happen but *Equipment Failure* does. The following table shows the parametrization of the node *Accident* for this case.



The resulting CPT will take the following form:



Noisy-Adder example

TO BE SUPPLIED

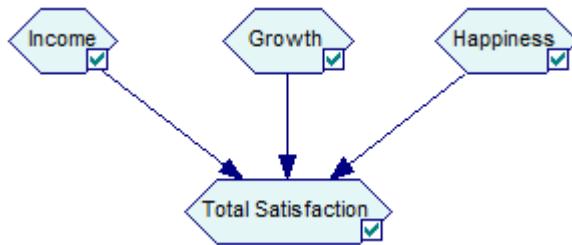
5.3.3 Multi-attribute utility nodes

Some outcomes of decision problems involve several, possibly conflicting attributes. For example, outcome of a business decision may optimize production 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.

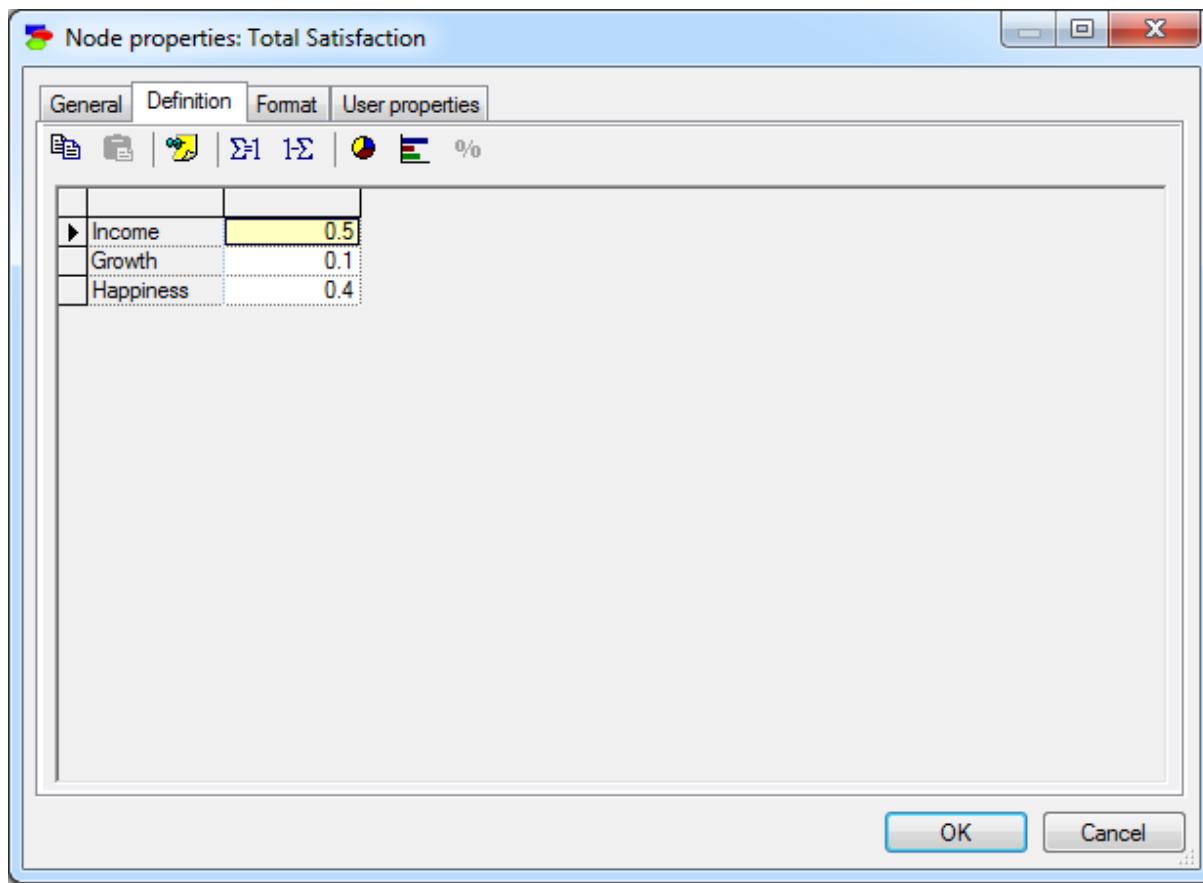
GeNle supports two types of multi-attribute utility nodes: (1) general *Multi-Attribute Utility* (MAU) nodes, which allow any explicitly framed multi-attribute utility function, and (2) a special case of MAU, the *Additive Linear Utility* (ALU) functions.

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*).



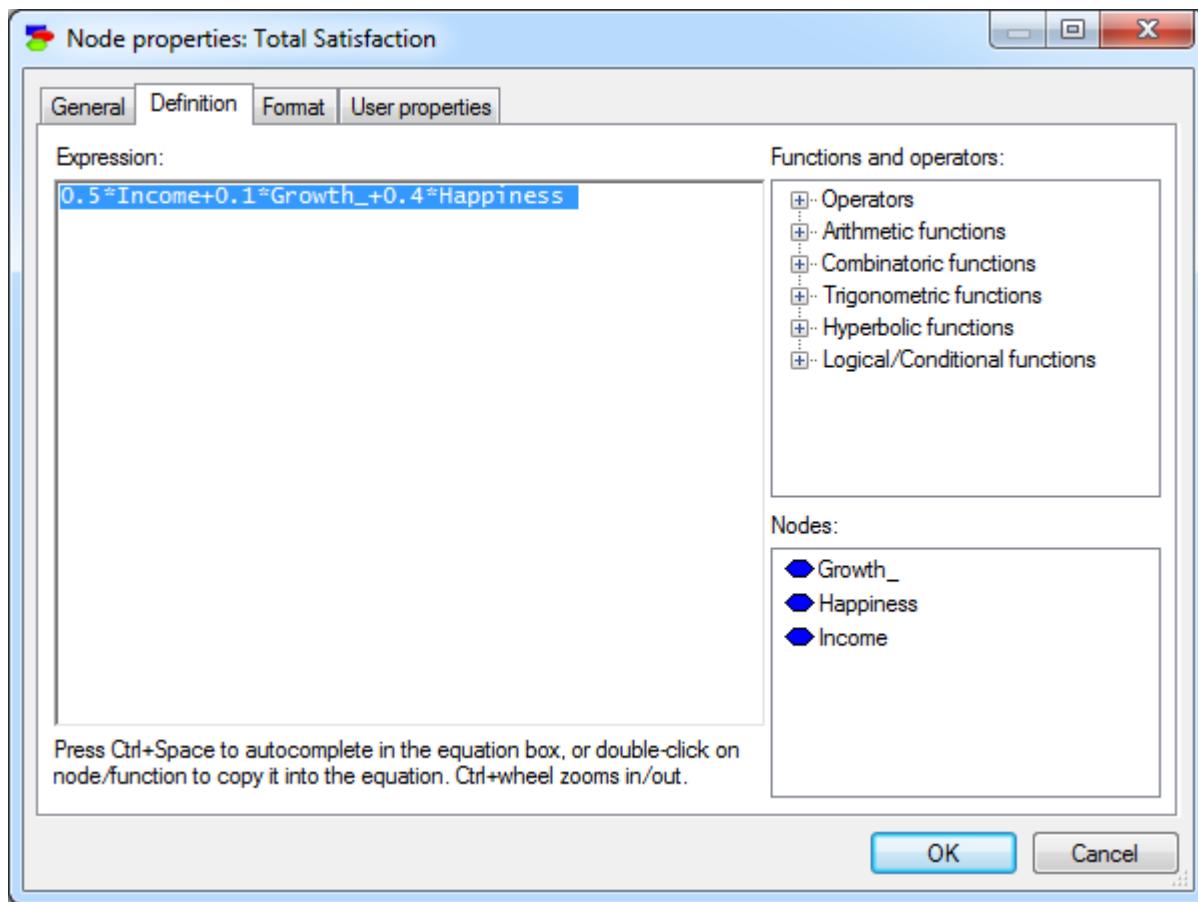
Let us define the coefficient (weights) of the linear multi-attribute additive linear utility function (*ALU*) in the following way:



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:

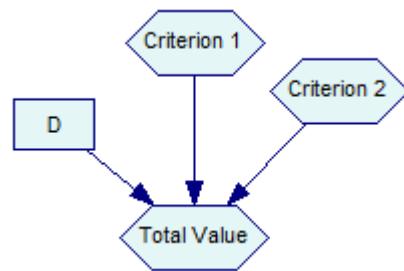
$$\text{TotalSatisfaction} = 0.5 * \text{Income} + 0.1 * \text{Growth} + 0.4 * \text{Happiness}$$

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

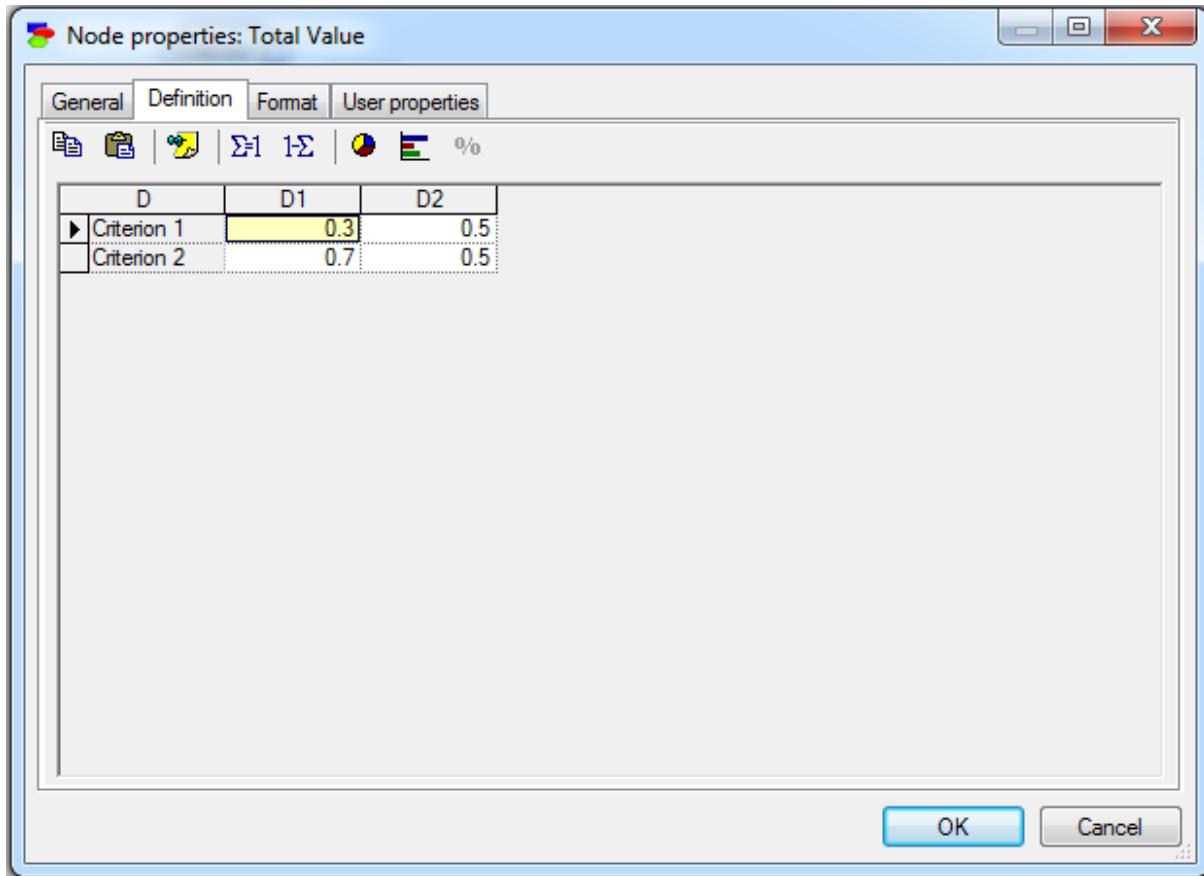


The definition is an equation involving the three parent utility nodes. There are no constraints on this equation, so any functional form can be used here, for example one that expresses the interaction between the three parent utility nodes as multiplicative. MAU nodes are thus as general as they get.

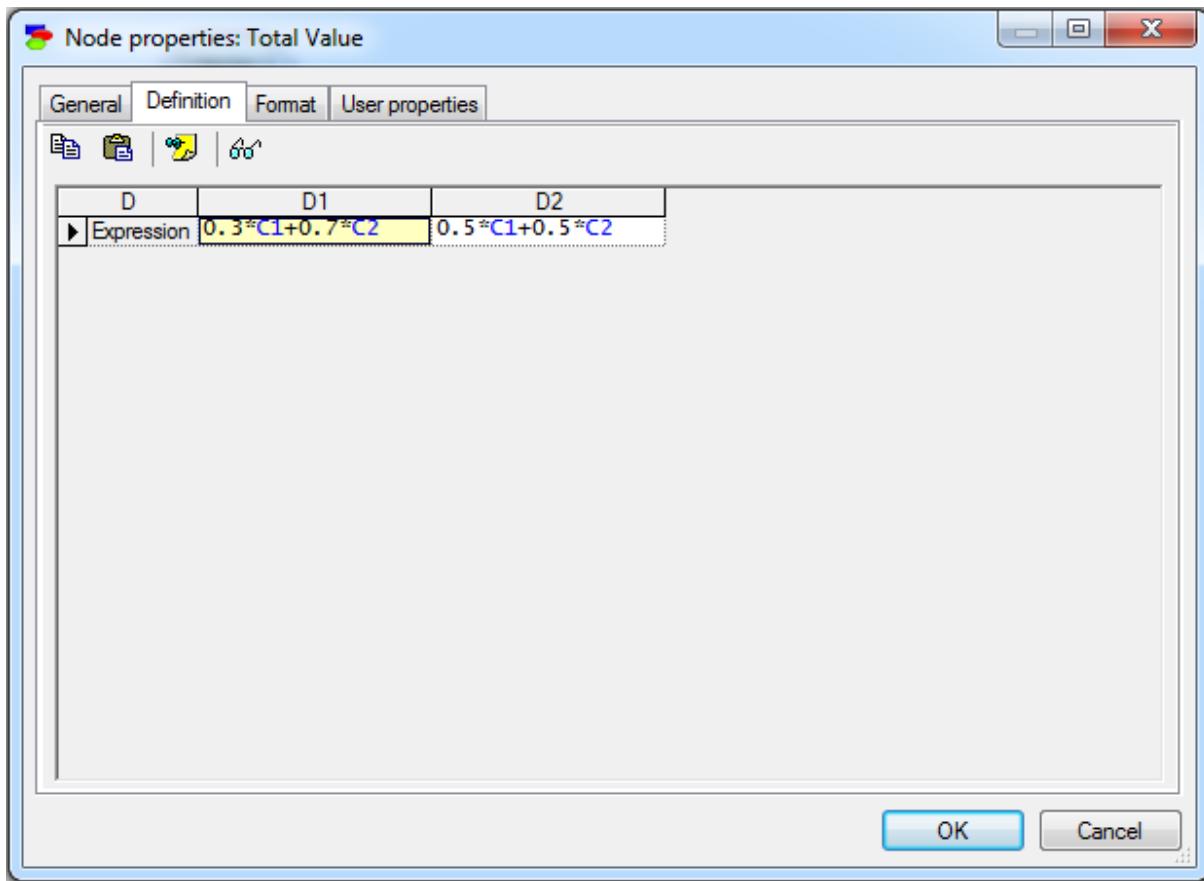
The only parents of ALU and MAU nodes can be utility or other ALU or MAU nodes. There is one exception that is allowed by GeNle - decision nodes can also be parents of ALU and MAU nodes. Consider the following network fragment



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](#)⁶⁰ 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.

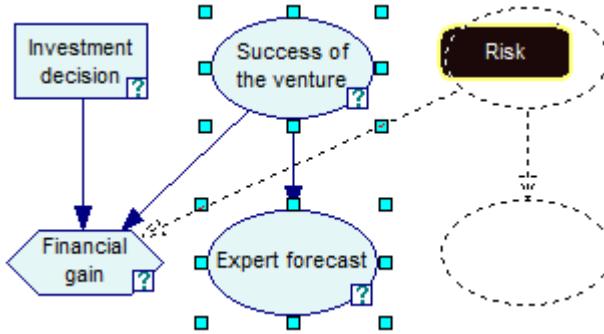
Creation of submodels, moving nodes, navigating through submodels

To create a submodel in GeNle, select *Submodel* from the [Tool Menu](#)¹⁷⁶ or the *Submodel* () tool from the [Standard Toolbar](#)¹⁷⁶ and click on the *Graph View*. You will see a new submodel.

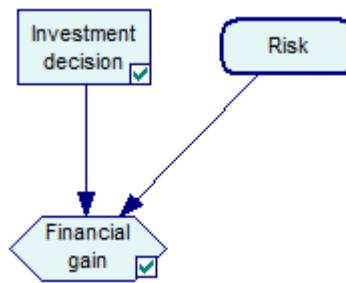


Submodel windows can be opened by double-clicking on the *Submodel* icon or right-clicking on the submodel icon and choosing *Open Submodel* from the *Submodel properties* menu.

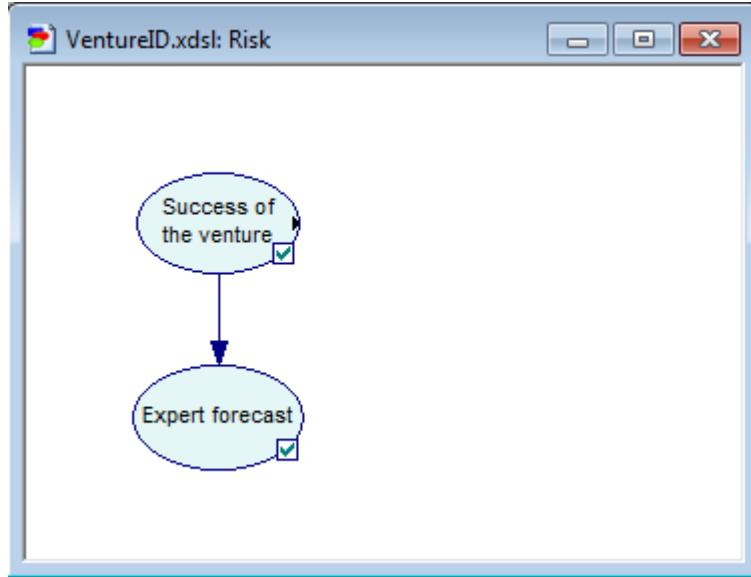
Nodes can be moved between submodels by selecting them in the source submodel, dragging, and dropping them in the destination submodel. For example, we might want to create a submodel for the variables *Success* and *Forecast* in the [influence diagram](#)⁴⁷ model used in the [Creating an influence diagrams](#)²⁸¹ 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.



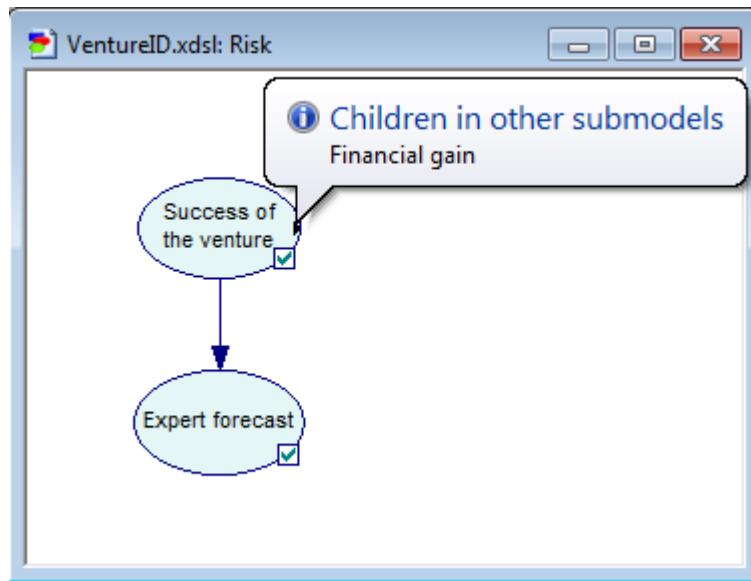
The resulting model will look as follows:



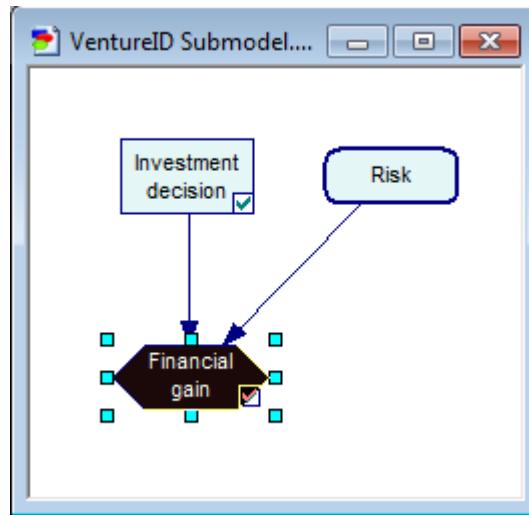
Submodels are opened by double-clicking on them. Double-clicking on the submodel *Risks* yields the following:



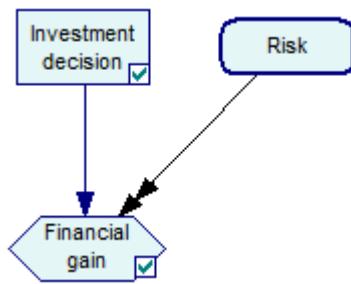
One thing that becomes less clear in submodels is the connections that the submodel has with the external world. GeNIE does not show arcs that are coming from outside or that go to the outside world. It does let the user know that there are such connections. First of all, by showing these connections 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* submenu to flash the child on the screen as shown below:



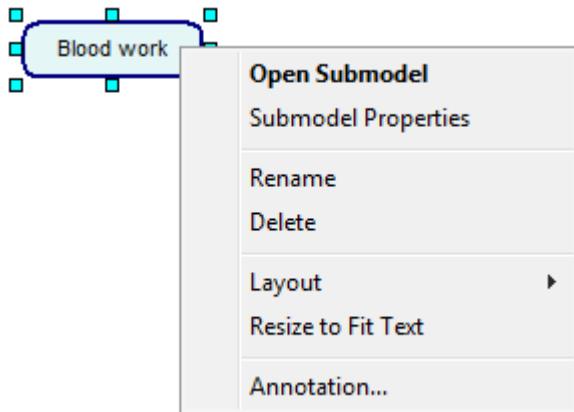
It is possible to add arcs between nodes that are located in different submodels in the very same way that arcs are added between nodes in the same submodel. When more than one arc is drawn between a submodel and a node, then GeNIE draws a double arrow arc from the submodel to the node as shown below:



All the above functions can be also performed through GeNIE *Tree View*.

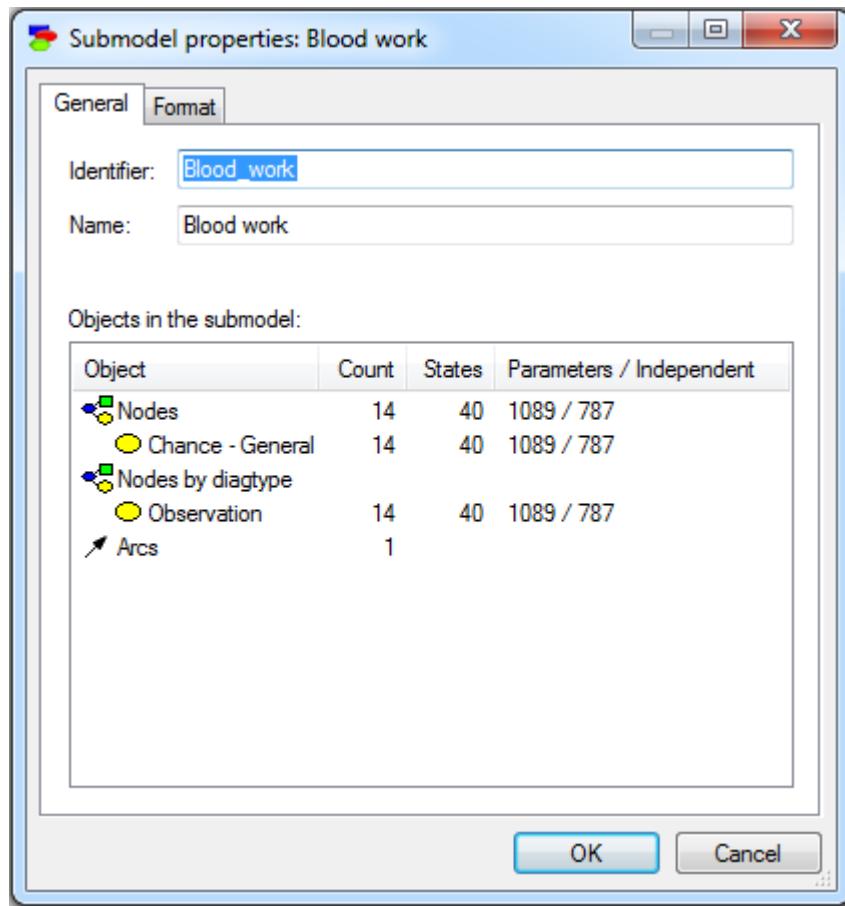
Submodel properties

Submodel properties sheet can be displayed by right clicking on the name of the submodel in the [Tree View](#)⁷³ or right clicking on the submodel icon in the [Graph View](#)⁶⁰. 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.

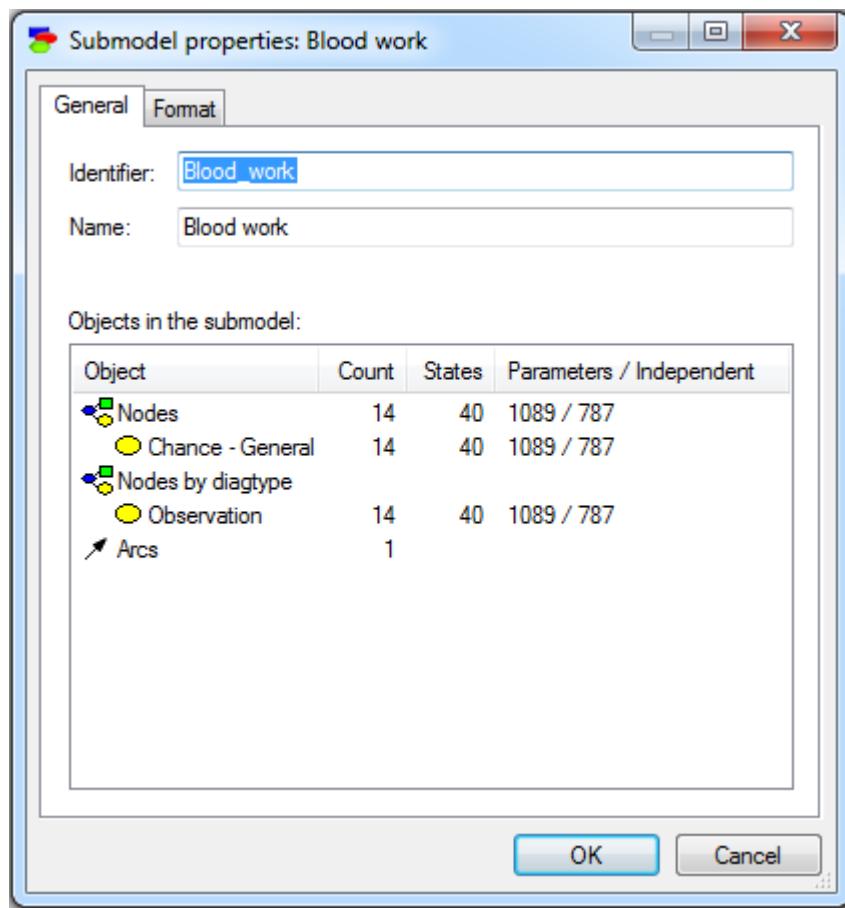
The *Submodel properties* sheet consist of two tabs: *General* and *Format*.



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.



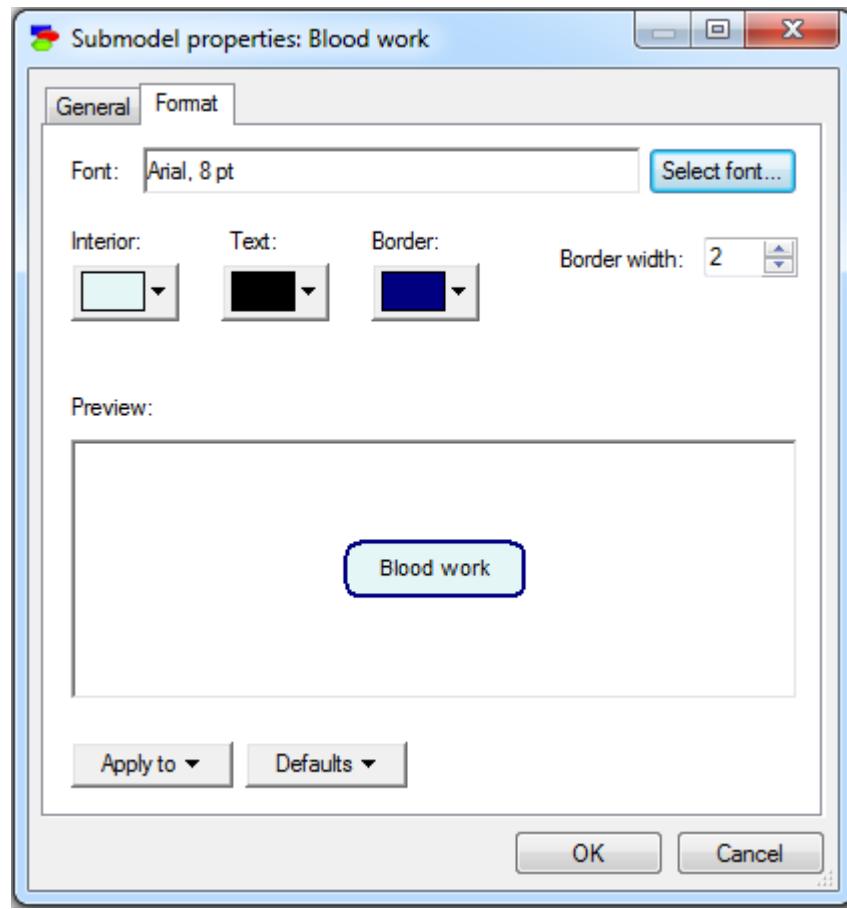
Identifier displays the identifier for the submodel, which is user-specified. Identifiers must start with a letter, and can contain letters, digits, and underscore (_) characters. The identifier for the network shown above is *Blood_work*.

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*.



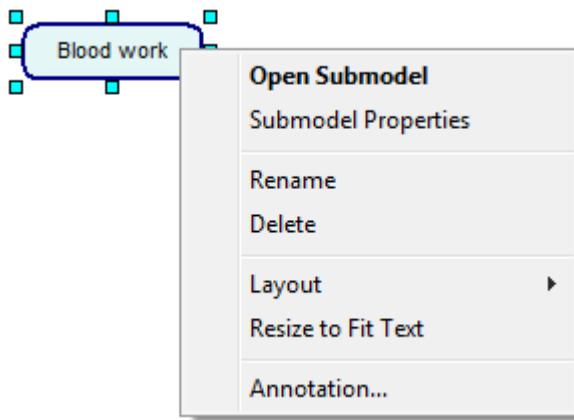
The *Format* tab is similar in function to the *Format* tab of the [Node properties](#)¹³⁸ sheet.

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](#)⁶⁰ can be displayed by right clicking on the submodel icon in the [Graph View](#)⁶⁰.



Open Submodel opens the submodel in a new [Graph View](#)^[60] window.

Submodel Properties opens the *Submodel Properties* sheet.

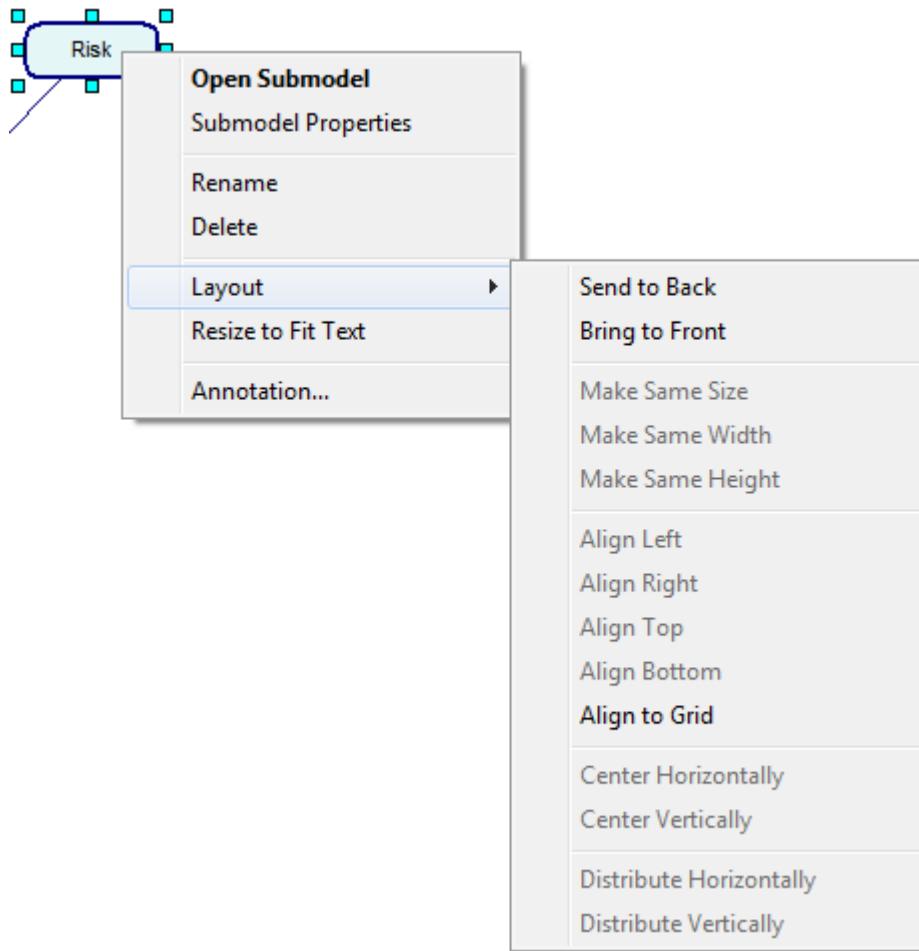
Rename allows you to rename the submodel by placing the submodel icon in edit mode. You can also rename a submodel by modifying the *Name* field in *Submodel Properties* sheet.

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](#)^[120] section for more information).

Layout submenu



Most of the commands on the *Layout* submenu are the same as those in the [Layout](#)¹⁷⁷ menu. The only commands here that are not found in the *Layout* menu are:

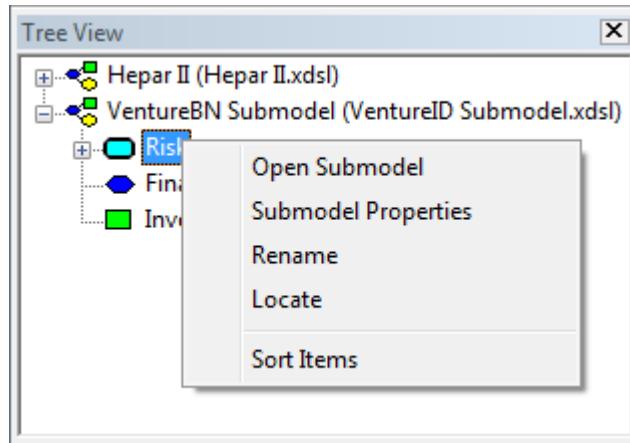
Make Same Size (enabled only if two or more items are selected in the *Graph View*) resizes the selected items so that they are the same size as the item that was right clicked.

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](#)⁷³ can be displayed by right clicking on the submodel name.



The two commands that are not available in the [Graph View](#)⁶⁰ are:

Locate locates the submodel in the [Graph View](#)⁶⁰ 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](#)⁴⁵ or an [influence diagram](#)⁴⁷ 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 S_1 to a sub-model S_2 means that there is at least one node in S_2 that depends on at least one node in S_1 . 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.

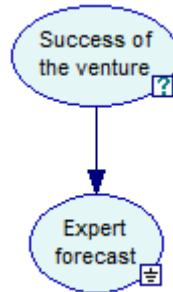
5.3.6 Node status icons

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.

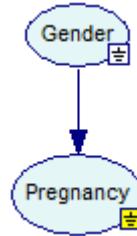
Observed Status Icon

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:



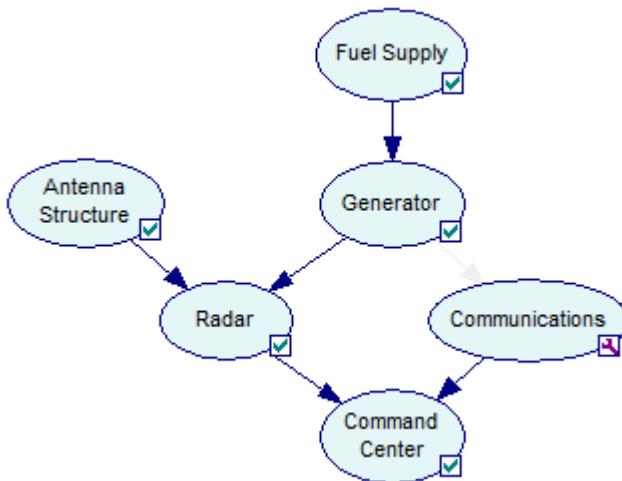
Implied Status Icon

Sometimes, observing a node implies the values of other nodes. For example, observing that a patient is 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*.



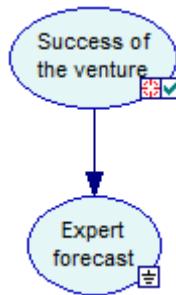
Controlled Status Icon

Controlling the value of a node is different from observing it (see [Changes in structure](#)⁵⁰ 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

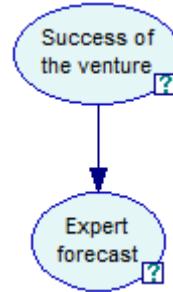
The *Target* status icon (☒) has to do with GeNIE's support for relevance reasoning. Very often in a decision support system, only a small number of variables are of interest to the user. When the model used by the system is large, the amount of computation to update all variables may be prohibitive 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

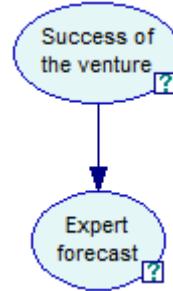
The *Invalid* status icon (☒) marks those nodes that need updating, i.e., their values (or the probability distributions) are invalid and cannot be inquired. *Invalid* mark can appear together with the *Target* mark, but never with the *Observed* and *Implied* marks (note that when a node is observed, 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:



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* (A) button from the 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*⁶⁰ and click the left mouse button. You will see a rectangle appear on the screen:

Type your text here

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:

This is an example text in a Graph View window text box

You can change the font type, style, size, and color using appropriate tools from the [Format Toolbar](#)¹⁷⁷. Shown below are some effects of using the [Format Toolbar](#)¹⁷⁷:

This is an example text in a
Graph View window text box

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.

This is an example text in a
Graph View window text box

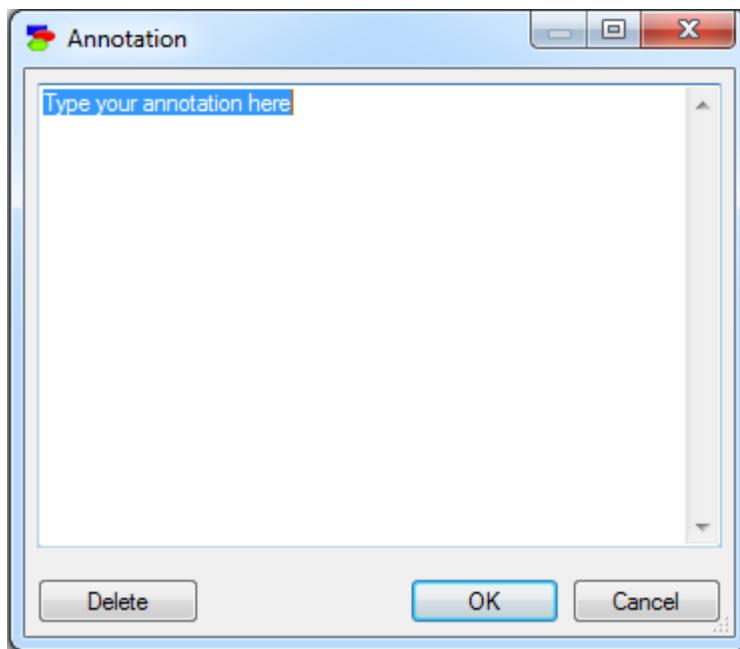
You can re-size a selected box by dragging on one of these small squares. You can delete it by pressing the *Delete* key. You can drag the box to a new location by clicking on it and holding the mouse button down while moving it to a new location.

5.3.8 Annotations

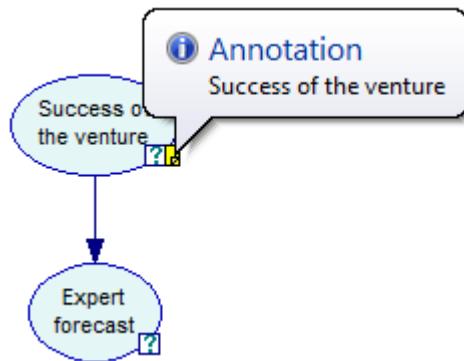
GeNle supports annotations for nodes and states of nodes. Following the idea that one of the main goals of a model is documenting the decision making process, annotations are useful for explaining function of nodes and states, or to note down just about anything the user feels is important regarding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

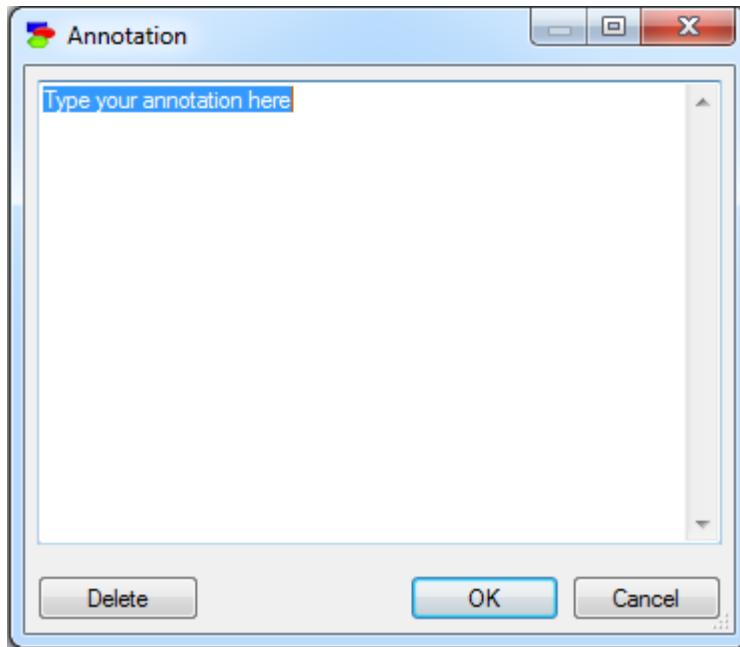


Double click on the note to display the annotation box, which will allow you to edit the annotation.

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

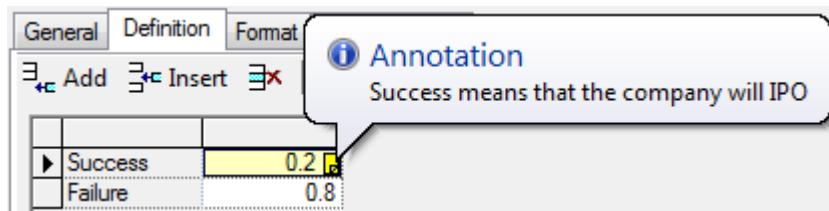
Annotations for states

You can annotate individual states of a node too. To annotate a selected state of a node open the [Node Properties](#)¹³⁸ 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:

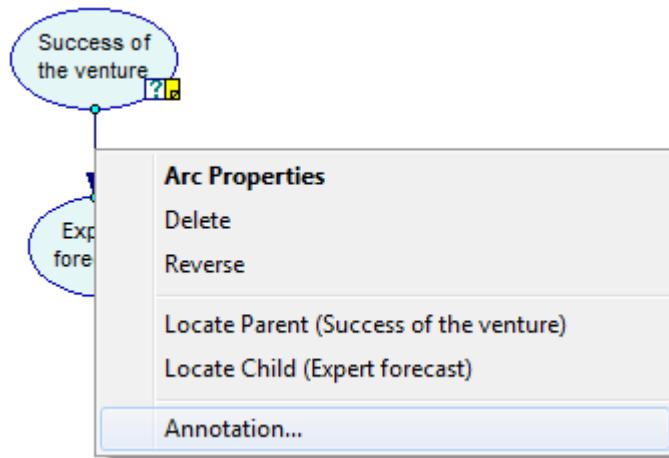
Hovering the mouse over the note icon will display the annotation text as shown below:



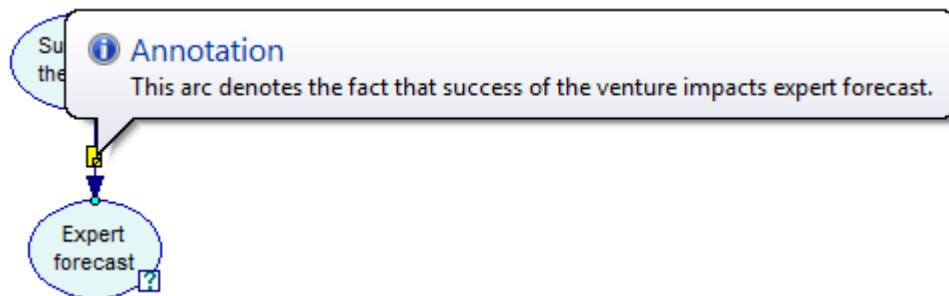
To edit the annotation click on the note.

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.

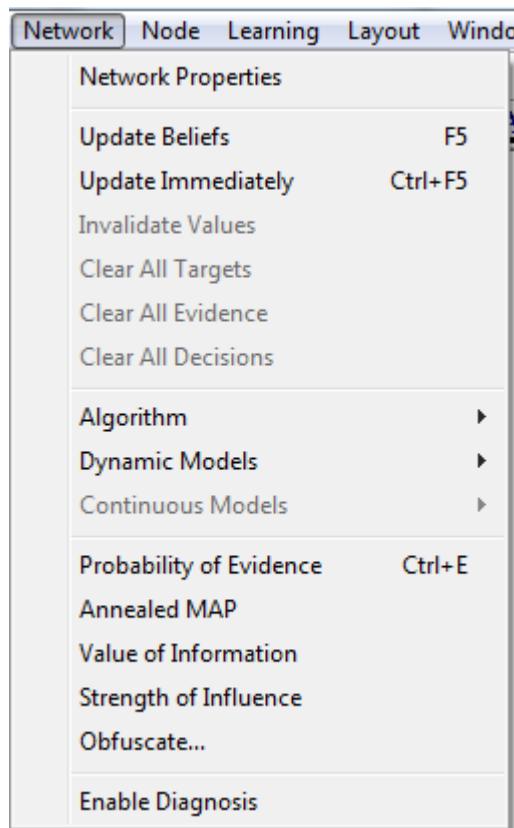


5.4 Model and component properties

5.4.1 Network properties

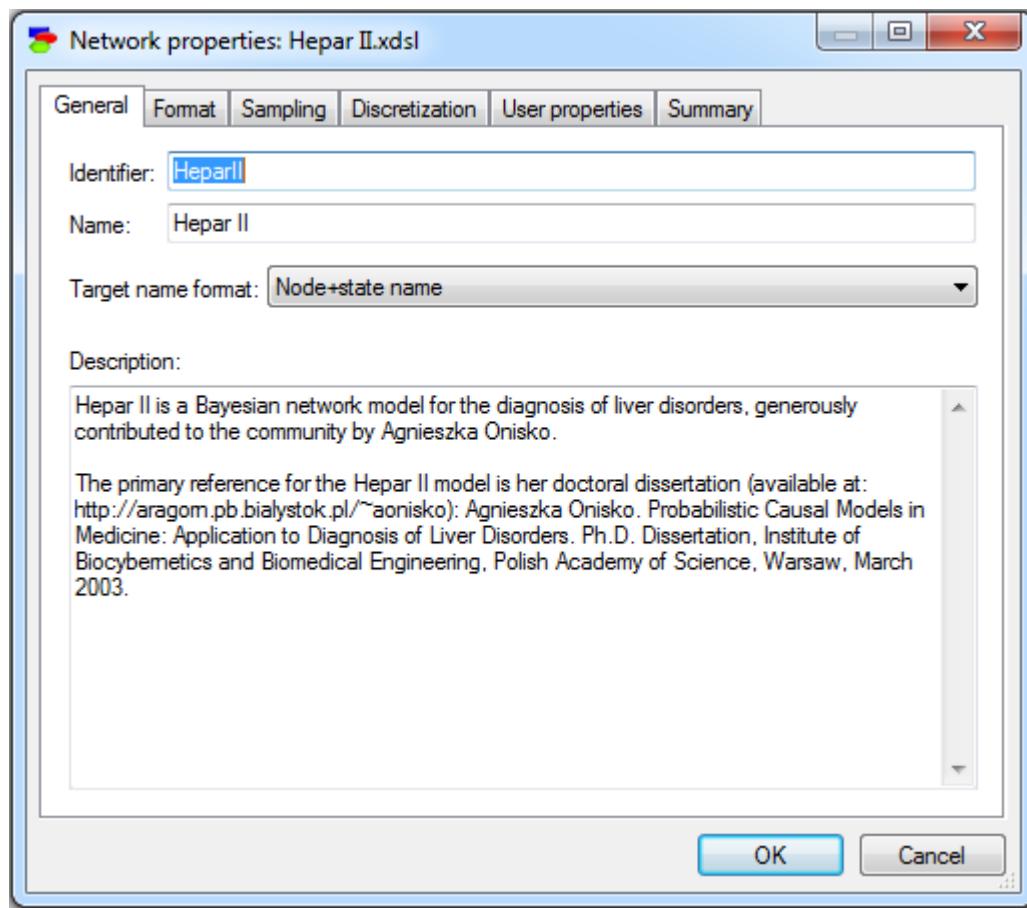
Network properties sheet summarizes all properties that are specified at the network level. It can be invoked in three ways:

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](#)²⁰⁹ 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

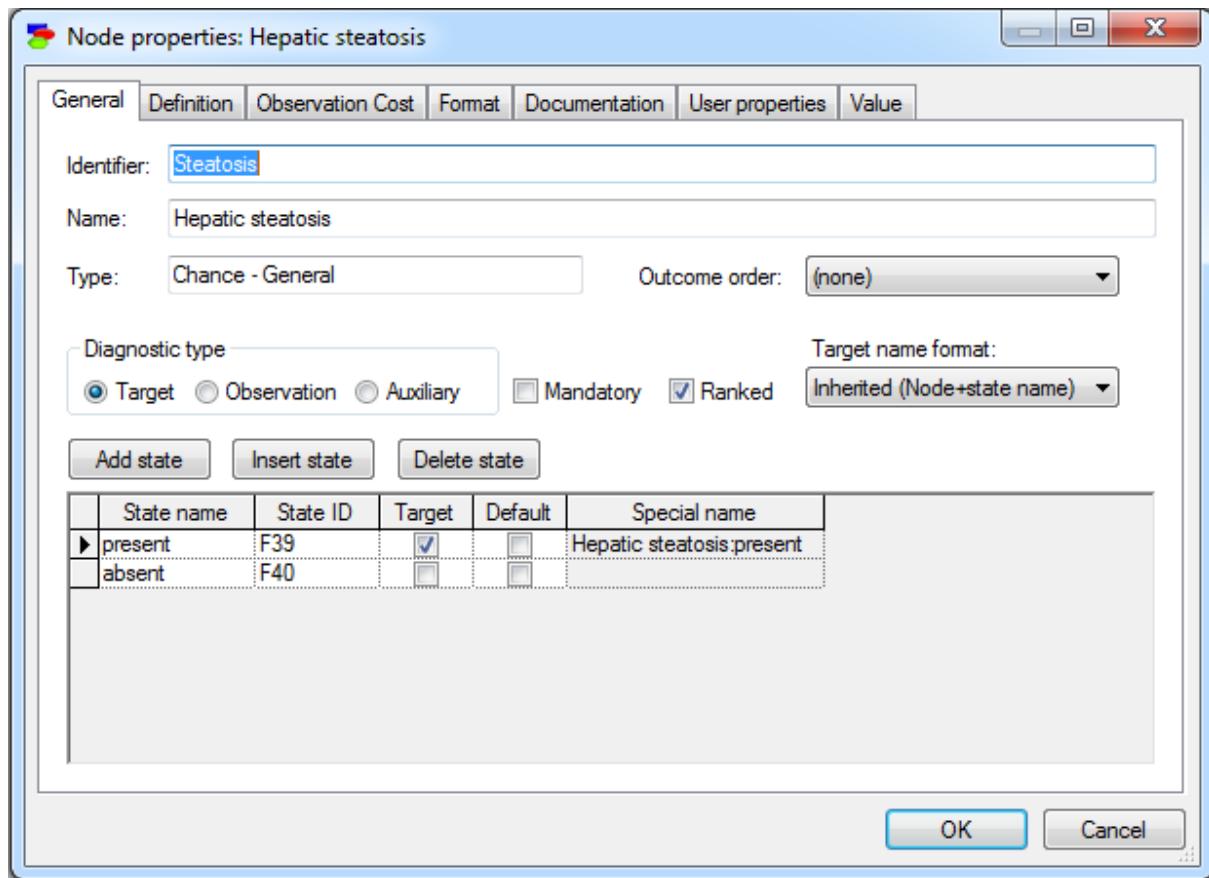
The *General* tab of *Network properties* (shown above) consists of the following fields:

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*.

Target name format is used to define the format for the special name given to states in target nodes. The special name is set on the General tab of the [Node Properties Sheet](#)¹³⁸ (see below), when diagnostic properties are enabled (see [Enable Diagnosis](#)³⁰⁸ for more information). When the *Target name format* is set to

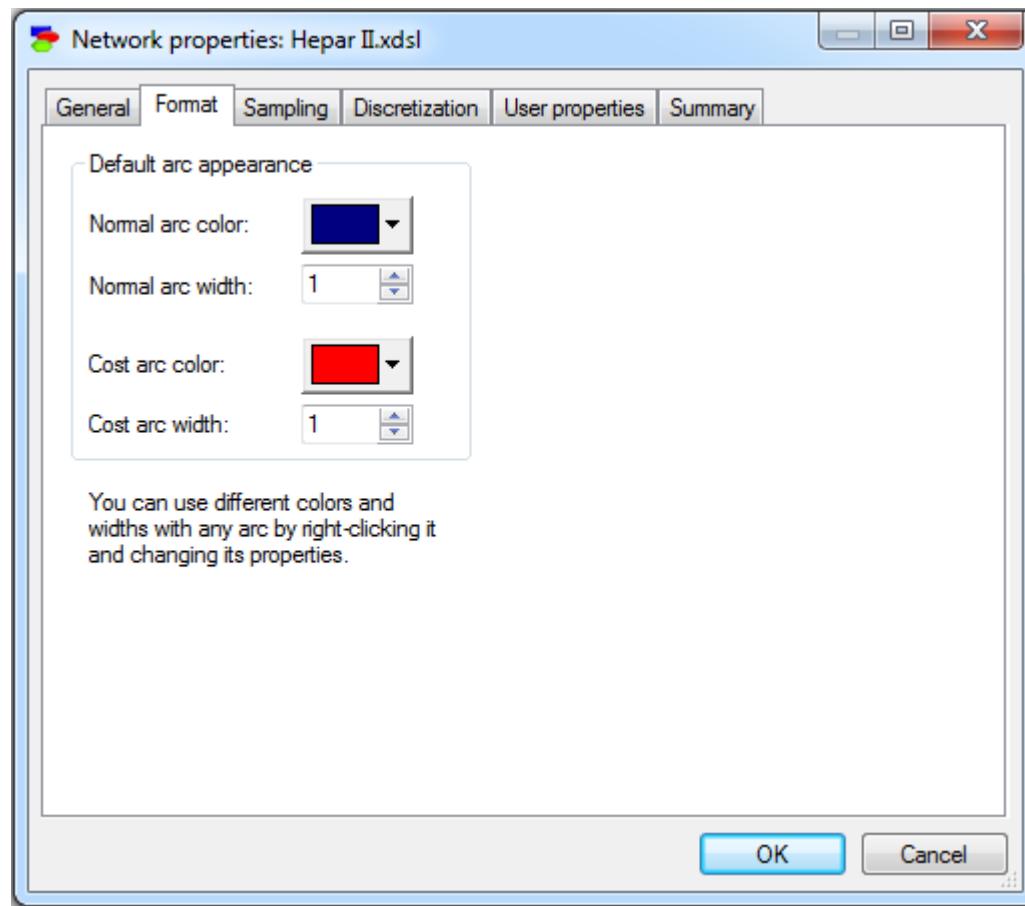
Inherited, the *Target name format* from *Network properties*, determines the format of the special name of targets.



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

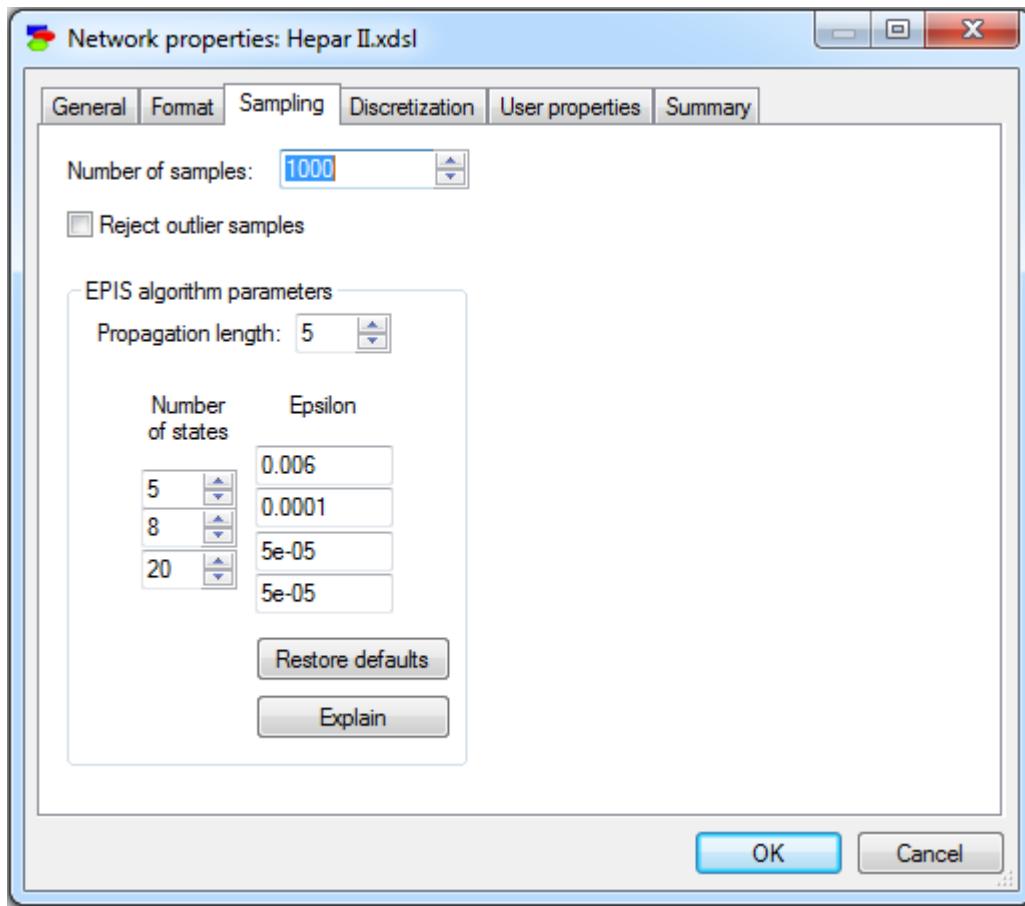
Format tab

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.



Sampling tab

The *Sampling* tab allows the user to set the various parameters of the [sampling algorithms](#)²⁰⁰ used in GeNIE.



Number of samples is quite likely the most important parameter and it sets the number of samples used in each execution of a sampling algorithm. 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). 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 invalidate all values (see the *Invalidate values* command above) and by this force GeNIE to recompute them during the next run of the algorithm.

Reject outlier samples check box essentially 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.

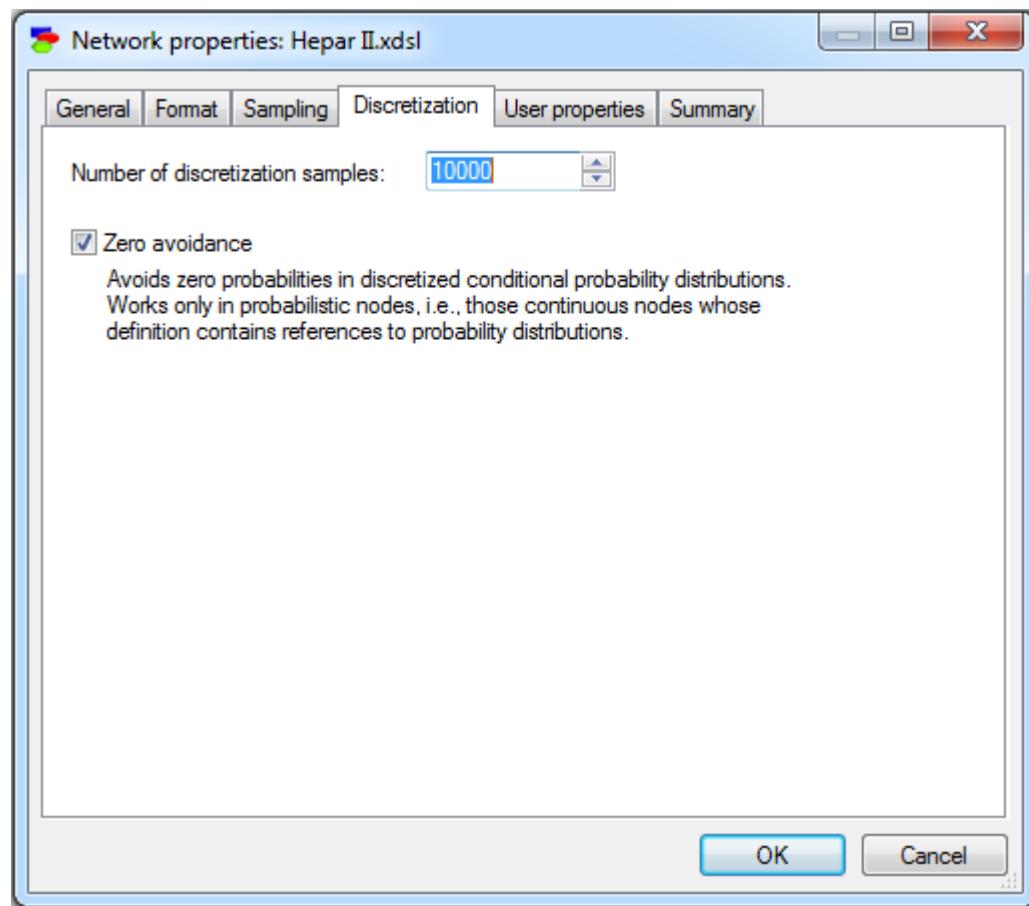
EPIS algorithm parameters are used to fine tune the EPIS algorithm execution. The EPIS algorithm uses *Loopy Belief Propagation* (LBP), an algorithm proposed

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.

Discretization tab

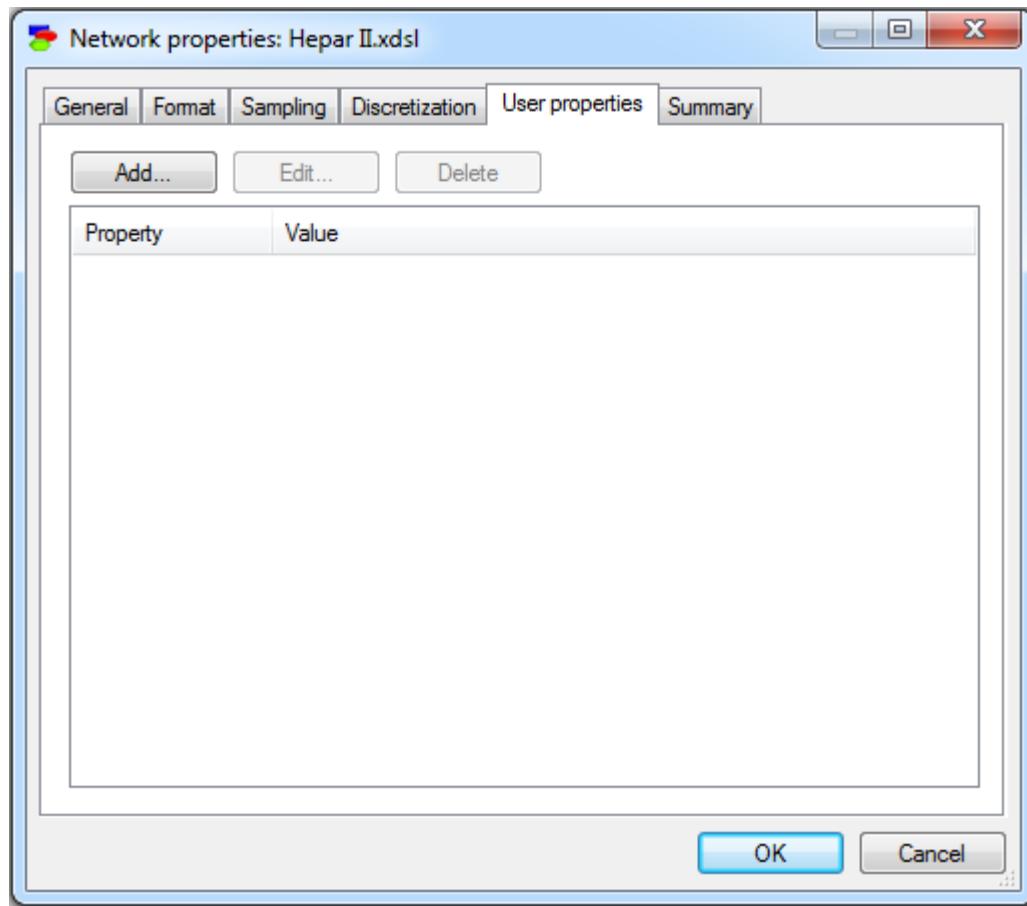
Discretization tab allows the user to define two important parameters of the [auto-discretization algorithm](#)²³⁰ for continuous models:

- *Number of discretization samples*, which determines the number of samples used in deriving the conditional probability tables in auto-discretized models.
- *Zero avoidance*, when checked, avoids 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).



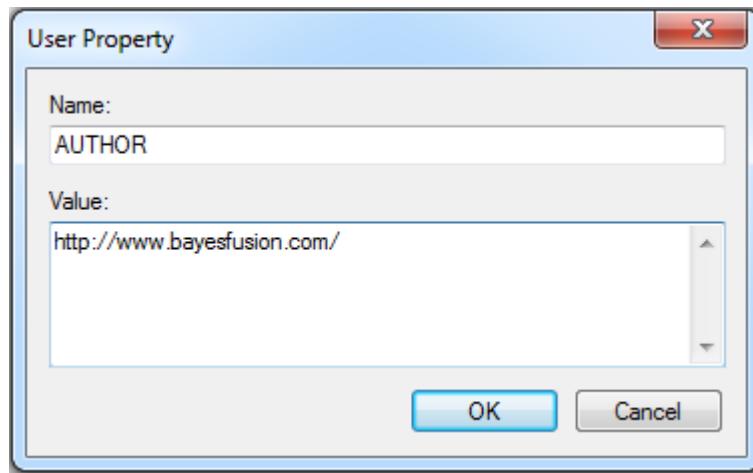
User properties tab

User properties tab allows the user to define properties of the model that can be later retrieved by an application program using [SMILE³¹](#).



For example, we can add a property *AUTHOR* with the value "<http://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 only allow for editing them.

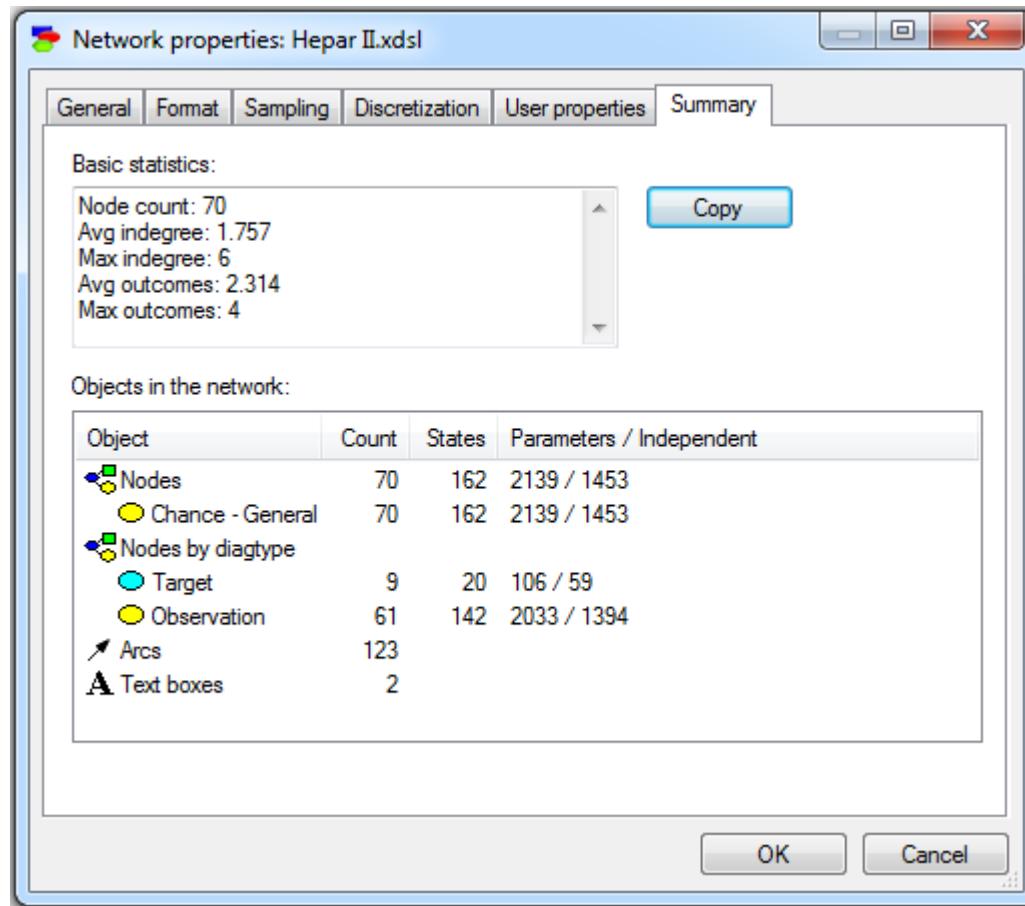
Button *Add* invokes the following dialog that allows for defining a new user property:



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

Summary tab

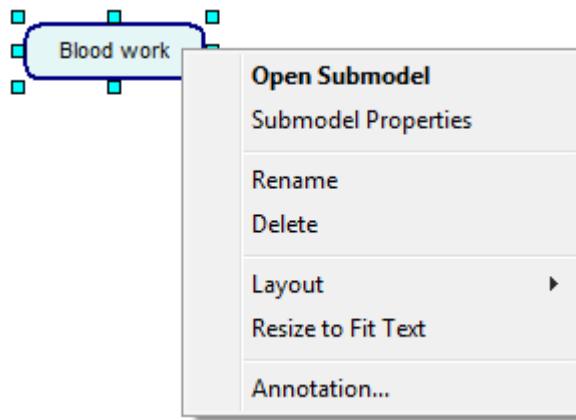
Summary tab contains summary statistics of the network, as illustrated below:



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 type, 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 target 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.

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](#)⁷³ or right clicking on the submodel icon in the [Graph View](#)⁶⁰. This will display the *Submodel Pop-up Menu*. Select *Submodel Properties* from the menu.

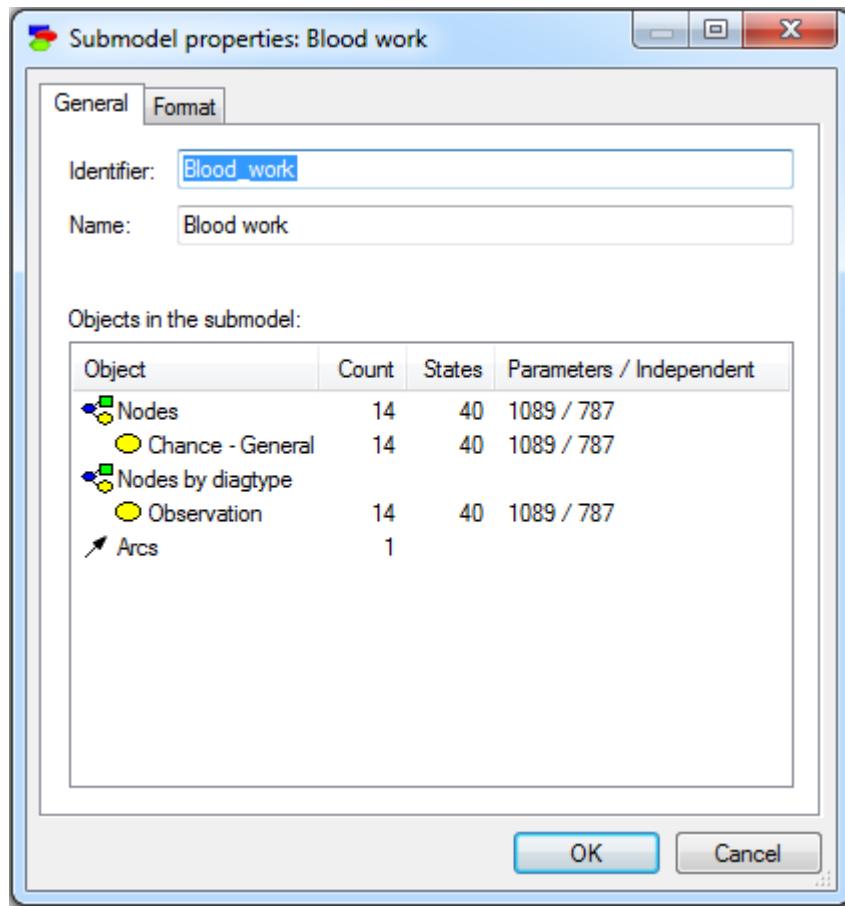


Note : Double clicking on the submodel will open the graph view of the submodel, it will not open the Submodel properties sheet.

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.



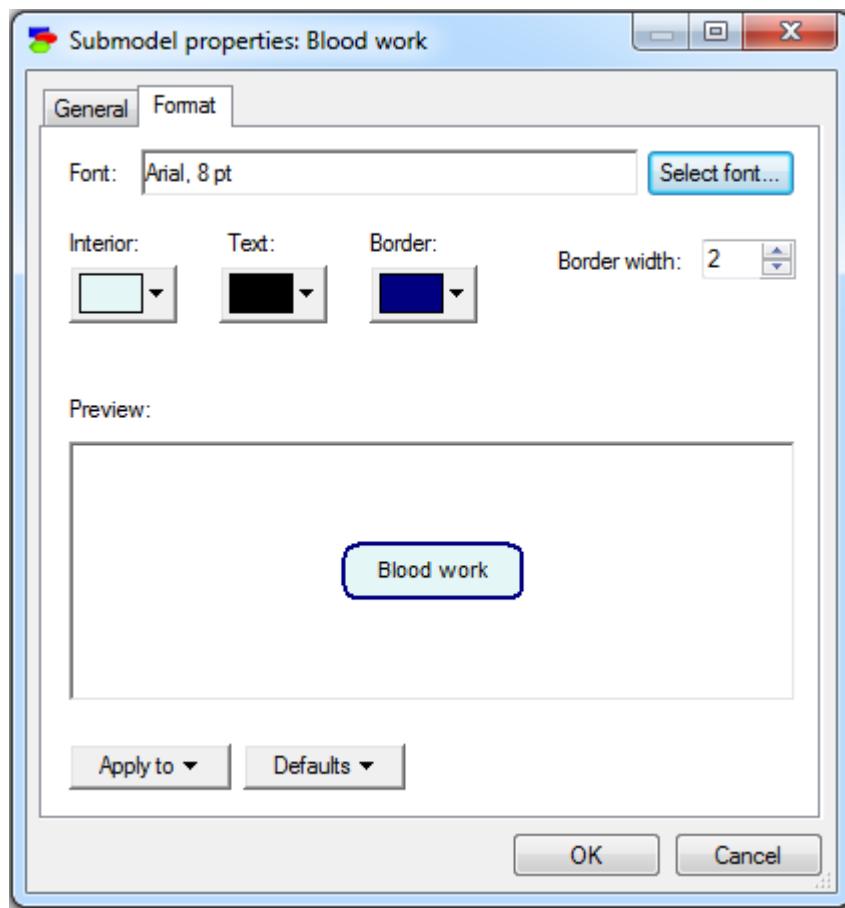
Identifier displays the identifier for the submodel, which is user-specified. Identifiers must start with a letter, and can contain letters, digits, and underscore (_) characters. The identifier for the network shown above is *Blood_work*.

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

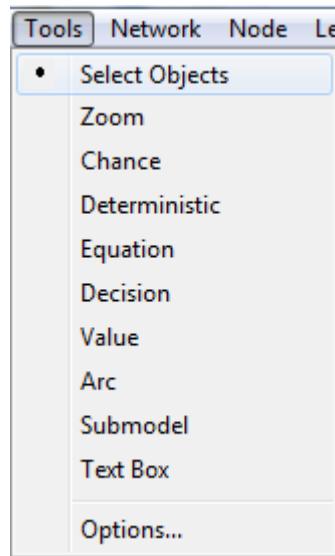
The *Format* tab allows to modify the visual properties of the submodel icon, i.e., how the submodel icon is displayed in the *Graph View*.



The *Format* tab is similar in function to the *Format* tab of the [Node properties](#) sheet.¹³⁸

5.4.3 Tools menu and Standard toolbar

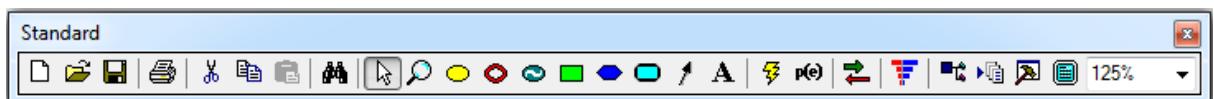
Tools menu is the main bag of tools for building models.



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 *Tool Menu*.

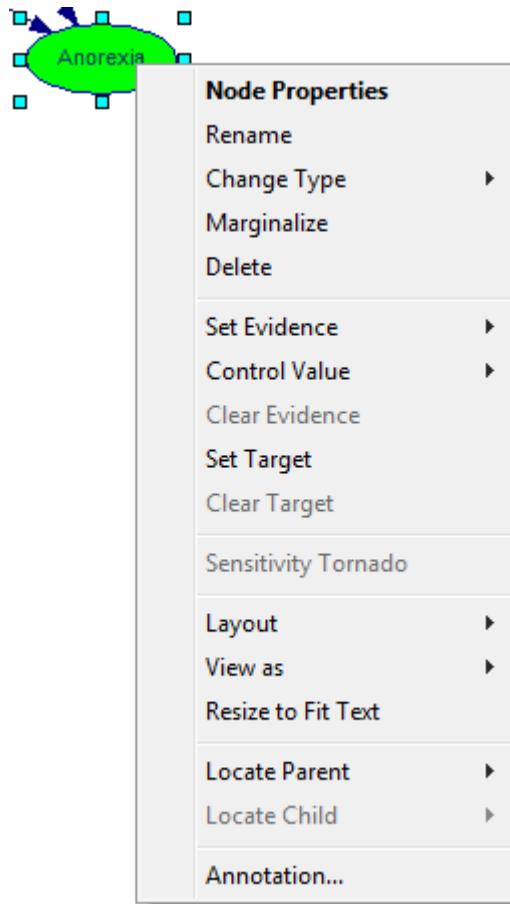
We review here *Standard Toolbar* tools that mimic the *Tools* menu tools and allow for creating objects in the *Graph View* window. The drawing tool currently selected is marked by a dot on the left side of its name in the *Tools* menu. Each of the *Tool* menu tools can be also selected using a corresponding *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* () , *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.

5.4.4 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](#) ⁶⁰.
2. Right click on the name of the node in the [Tree View](#) ⁷³ or right click on the icon of the node in the [Graph View](#) ⁶⁰. This will display the *Node Pop-up* menu. Select *Node Properties* from the menu.

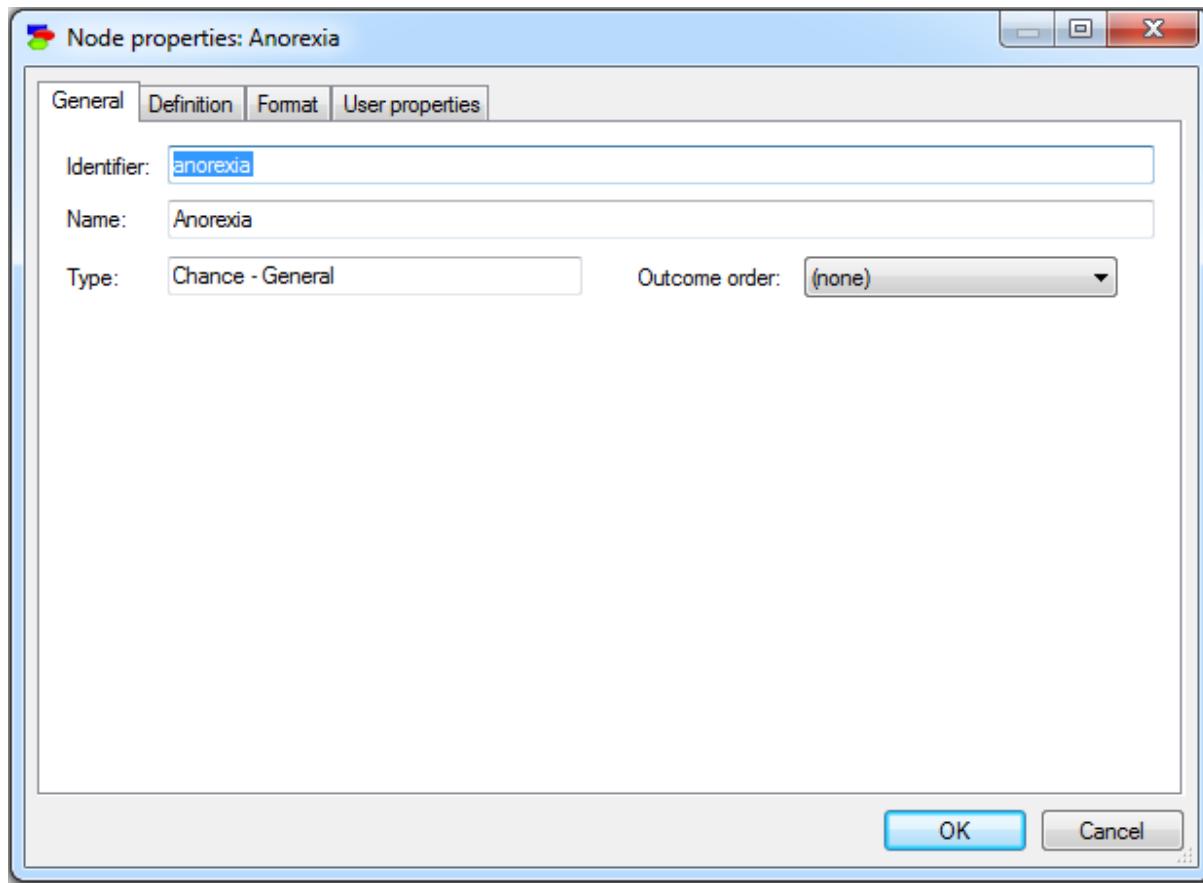


The *Node properties* consist of several tabs. While we will discuss all of the tabs in this section, not all of the tabs appear among the *Node properties* at the same time.

The *Value* tab appears only when the value of the node is available. Some tabs appear only when the diagnostic features of GeNIE are enabled.

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:

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. The identifier for the network shown above is *anorexia*.

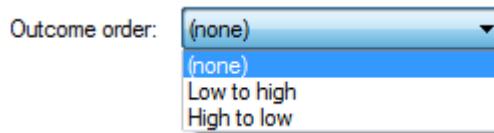
Name displays the name for the network, which is user-specified. There are no limitations on the characters that can be part of the name. The name for the network shown above is *Anorexia*.

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.

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](#)¹⁷⁶). It can be changed after creation by either right clicking on the node in the [Graph View](#)⁶⁰ 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](#)²⁰⁷ section for more information.

Note: *Chance Noisy nodes have to be created by first creating the Chance - General node and then changing its type using the method shown above.*

Outcome order 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](#)⁸⁷.

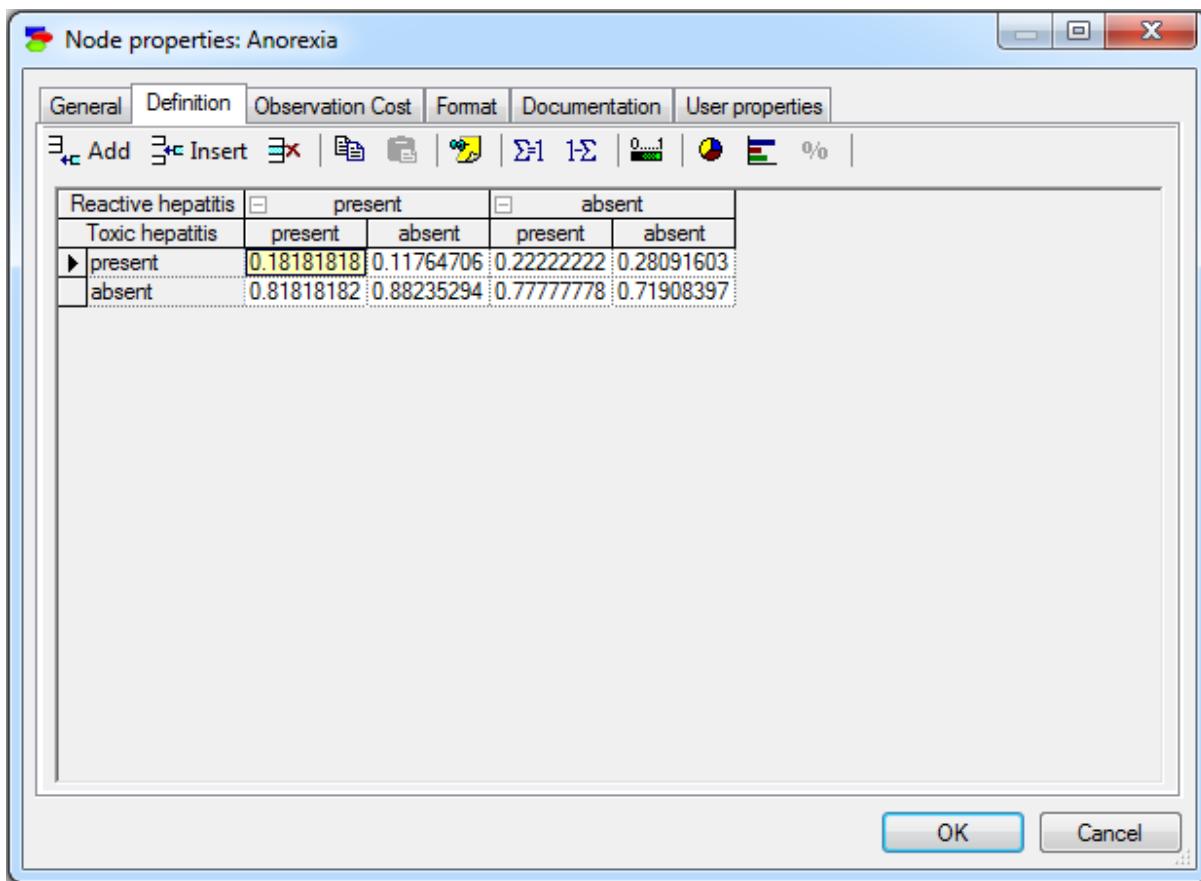


Definition tab

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-General nodes

Chance-general are nodes modeling discrete random variables. 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).



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 $2 \times 2 = 4$ 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 by clicking on the Node name, or an entire column by clicking on the state name.

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.

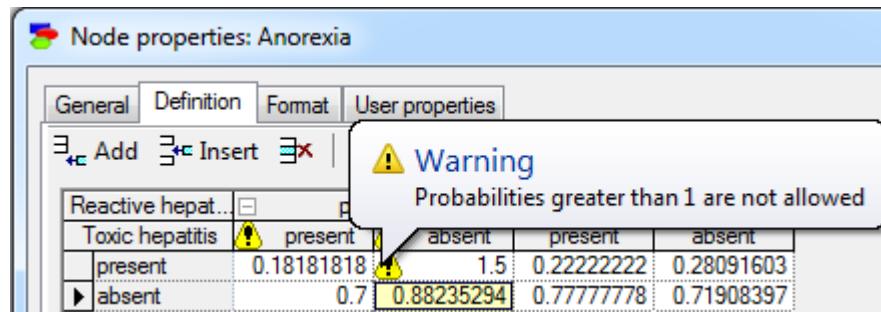
The *Complement* button (shortcut *CTRL+O*) can be pressed after selecting one or more cells (they do not have to be cells in the same column). When pressed, it fills in the selected cell the complement of the remaining cells in the column, i.e., a number that will make the sum equal precisely to 1.0. 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 0.7, 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.

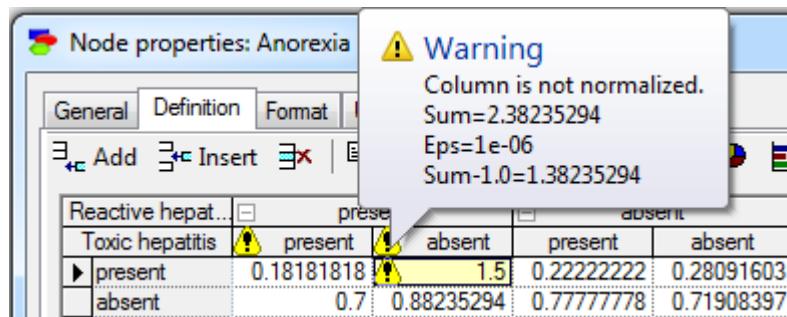
GeNIE warns the user of problems in the probability distribution tables whenever these contain incorrect distributions. Any number outside of the range [0..1] causes GeNIE to raise a flag in the cell. Also, when the sum of probabilities in a column is not 1.0, GeNIE places a flag on the column.

		present		absent	
		present	absent	present	absent
Toxic hepatitis	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:

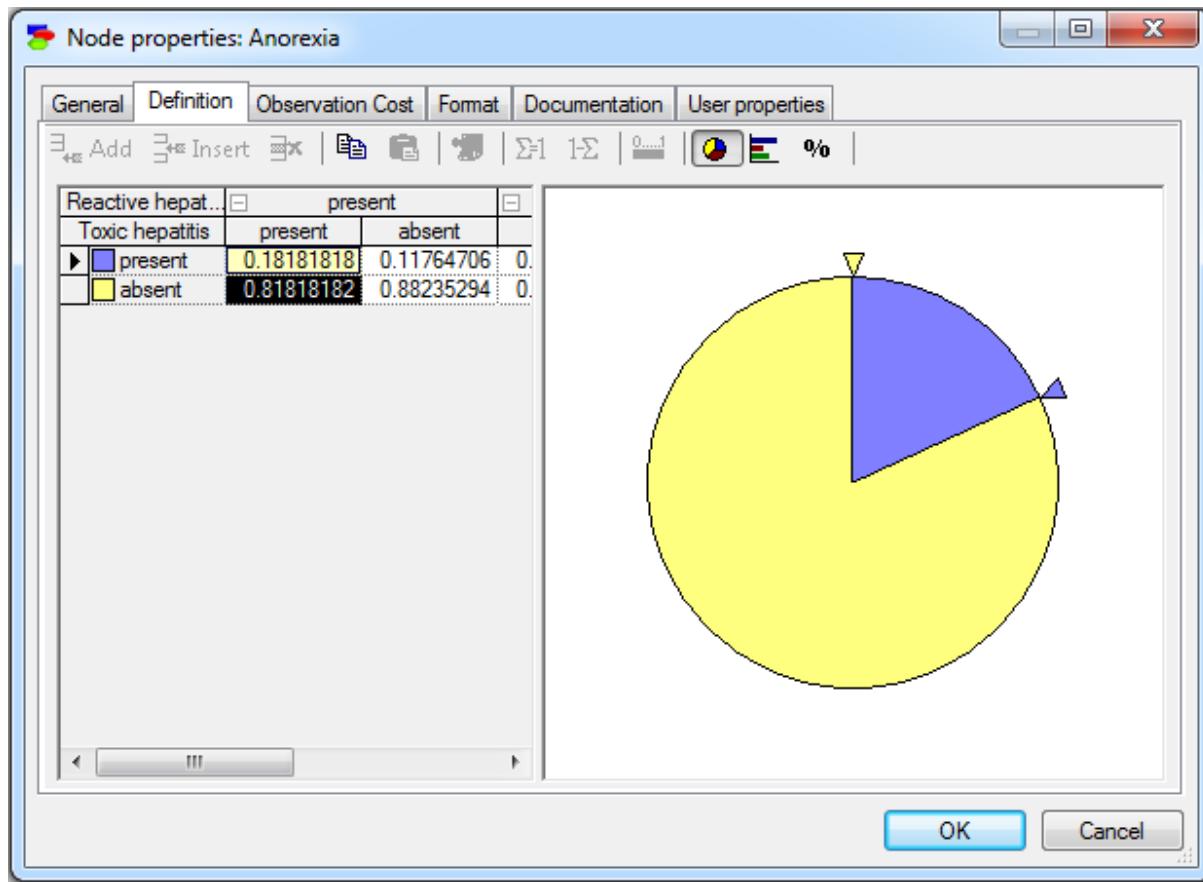


The warning displays the sum of probabilities, the tolerance threshold value (in the above example $Eps=1e-06=0.000001$), and the difference between the sum and the

theoretically enforced sum of 1.0. The tolerance threshold is adaptive and is never larger than the smallest probability value in the CPT.

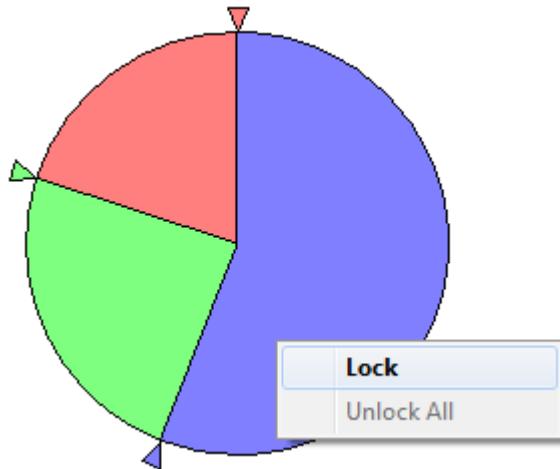
GeNIE will not allow to exit the *Node properties* dialog if the definition is incorrect.

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:

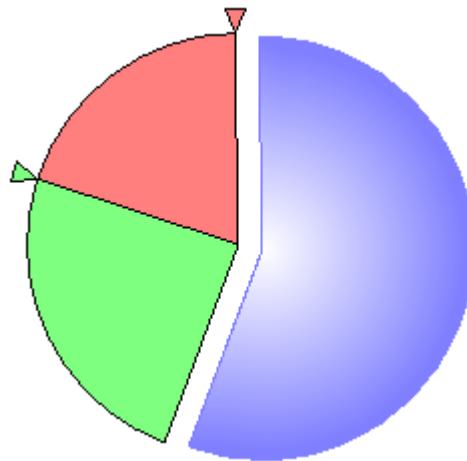


You can modify the probability distribution graphically using the small triangles. Drag a triangle around the circumference of the pie to increase or decrease the probability distribution for that state. As we adjust the size of one section of the piechart, the remaining section change proportionally.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

Once you have finished modifying a particular state, you can freeze it by double clicking on the pie area for that state. It will result in that portion of the pie to separate as shown below.

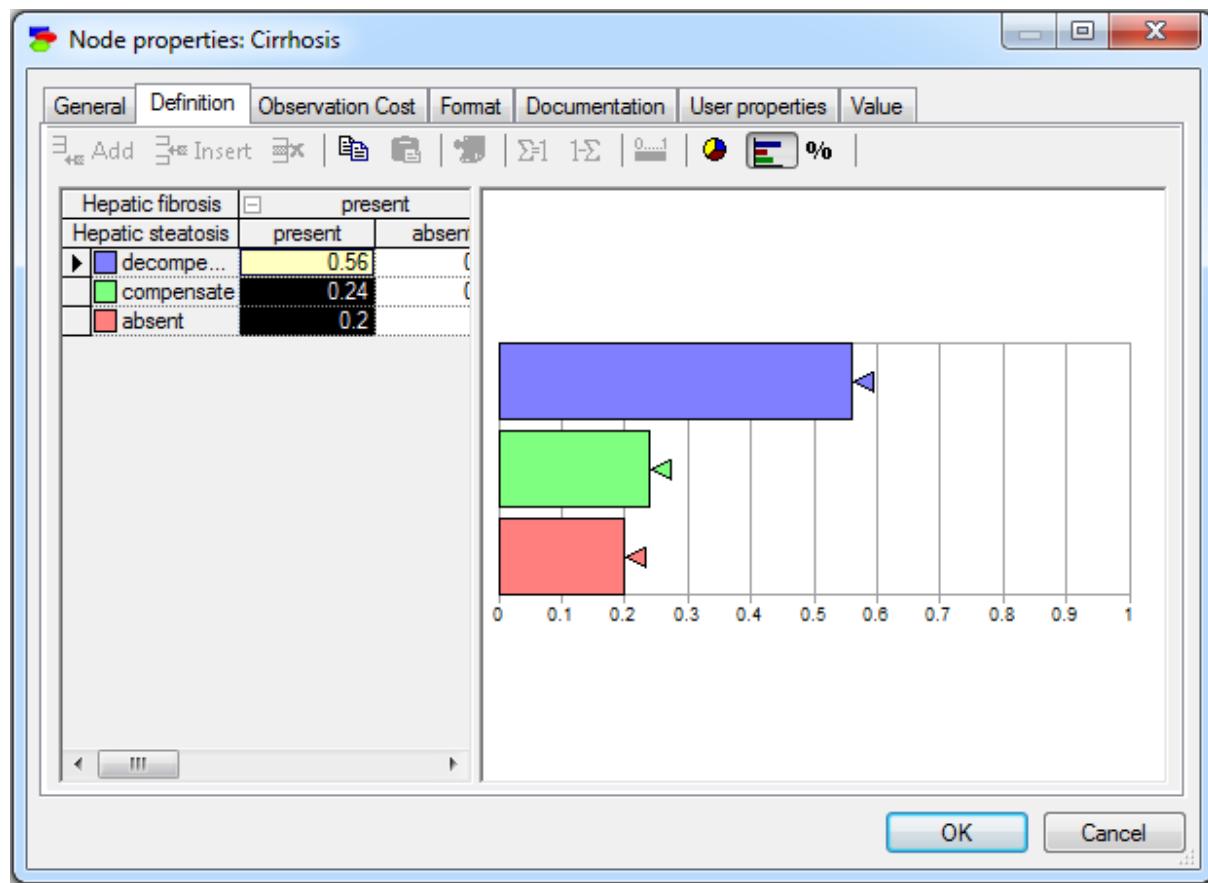


The effect of this is that the locked part no longer takes part in elicitation and remains constant until released.



It is possible to lock multiple parts.

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

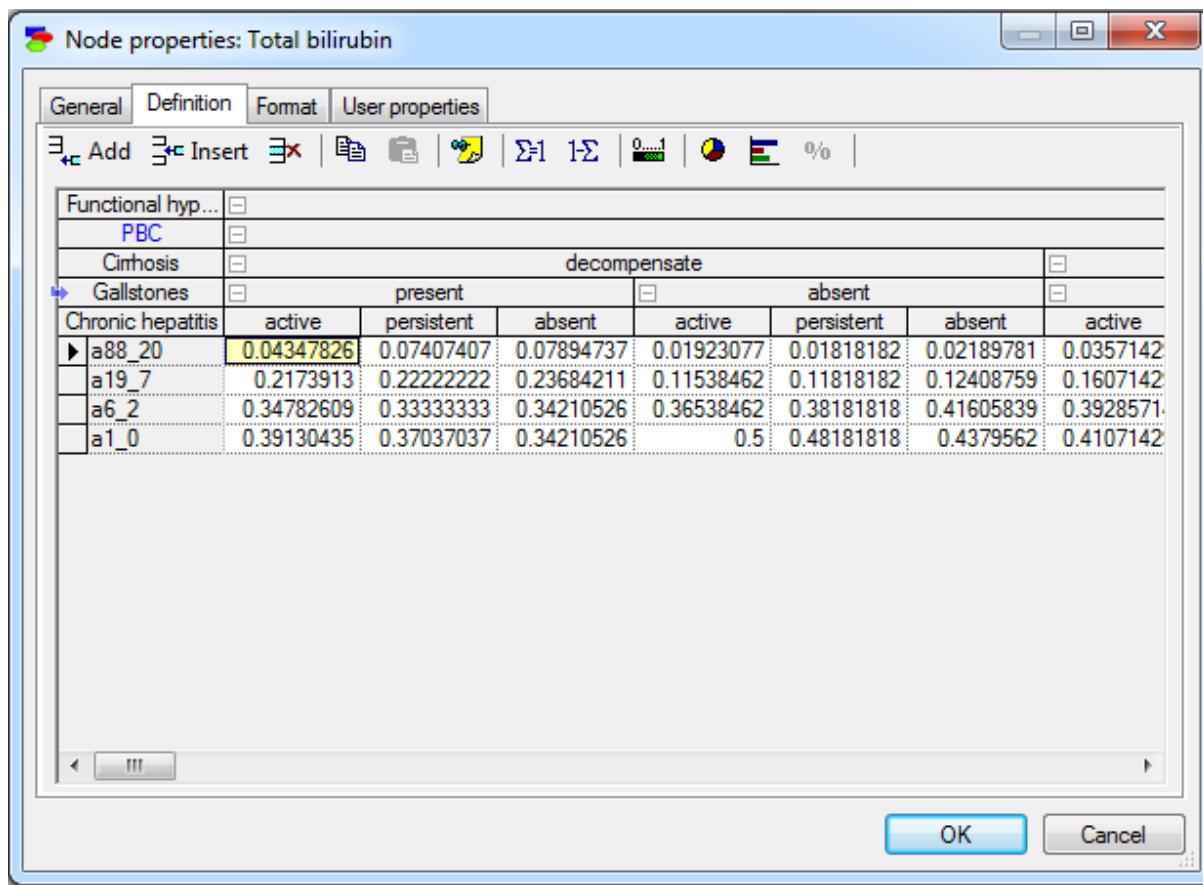


Here also you can change the size of the individual bars by dragging the small triangles on their right-hand side. You can lock those bars that we are happy with to prevent them from further changing.

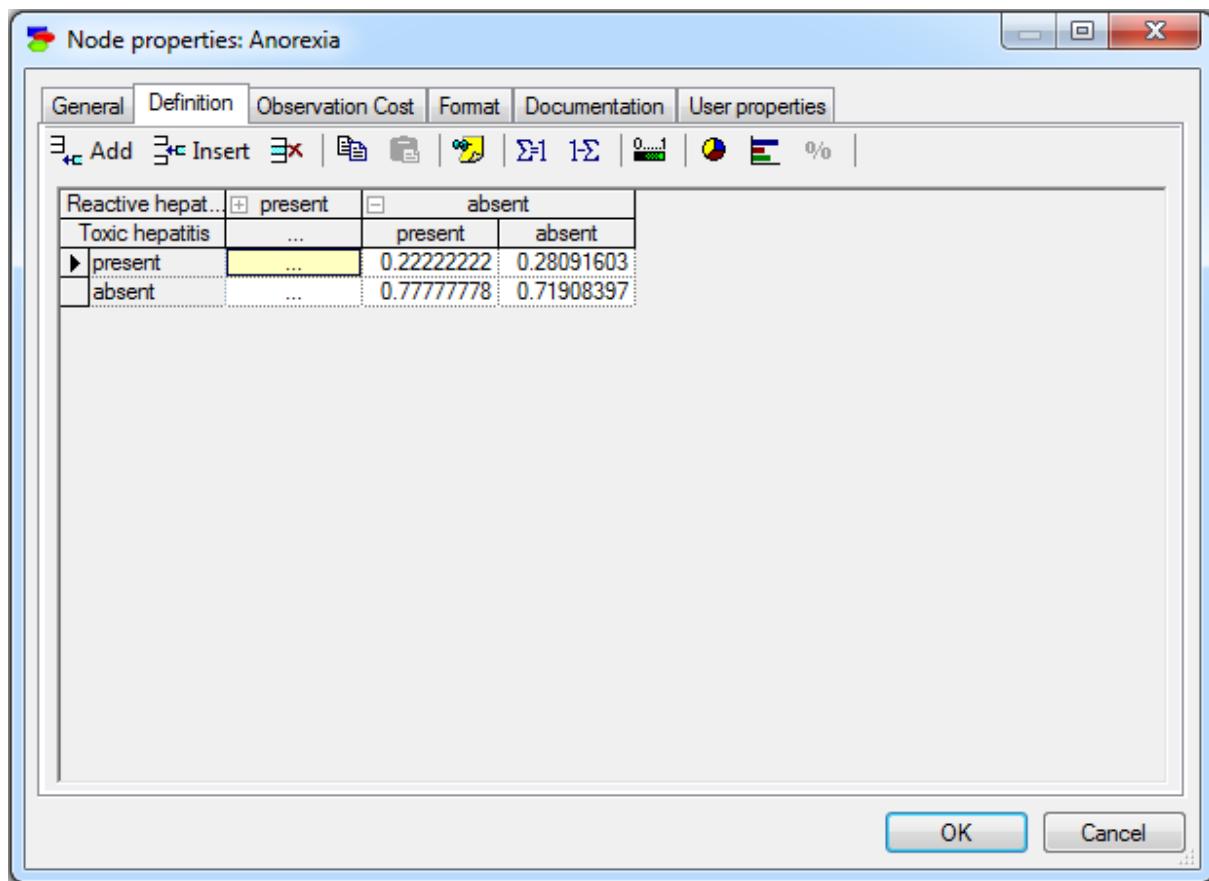
The *Show percentages in the elicitation chart* (button turns on a tool tip that shows the numerical value of the modified probability. There has been empirical research showing that viewing the probabilities during graphical elicitation may not be the best idea in terms of leading to a decreased accuracy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

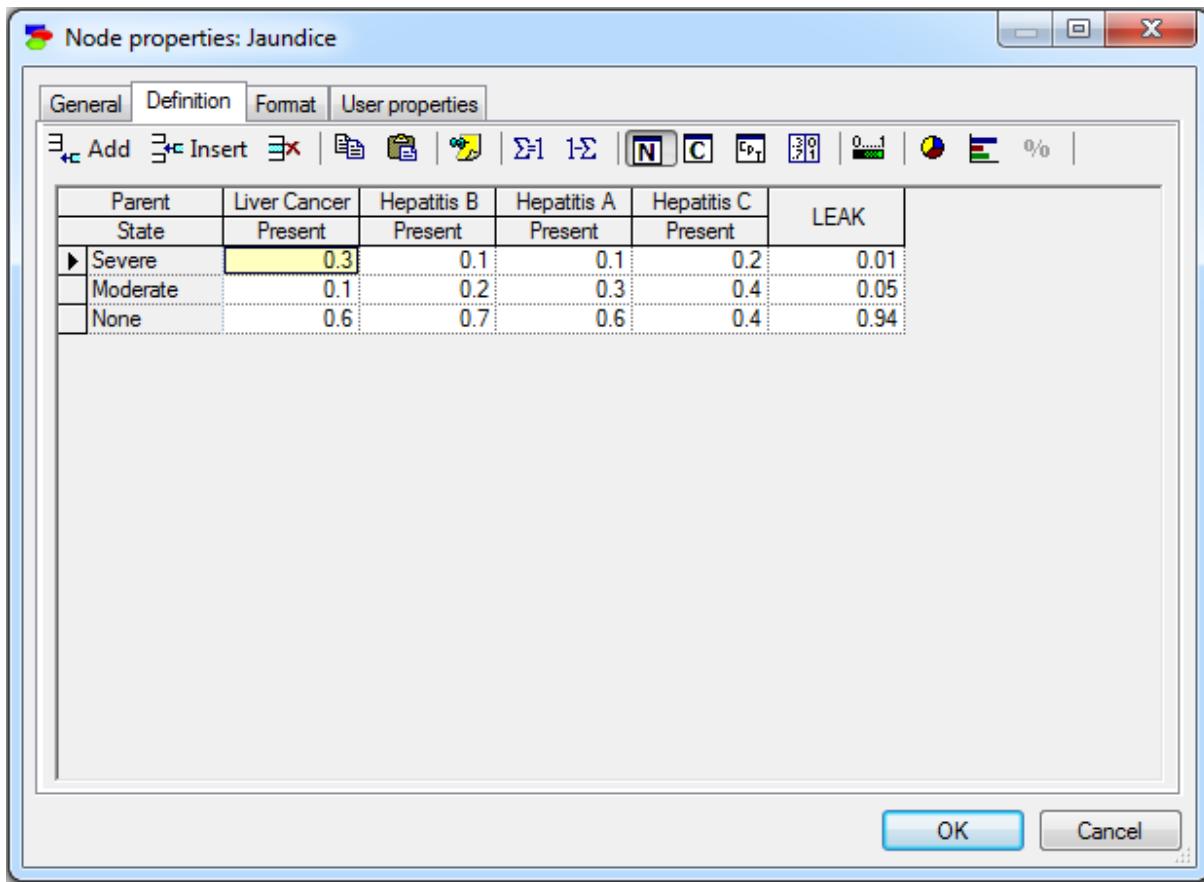


When the CPTs are very large, it is useful to shrink part of them. To that effect, please use the small buttons in the header lines (and).



Chance-NoisyMax nodes

Chance-NoisyMax are a special case of chance nodes modeling discrete random variables using the Noisy-Max assumption. 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).



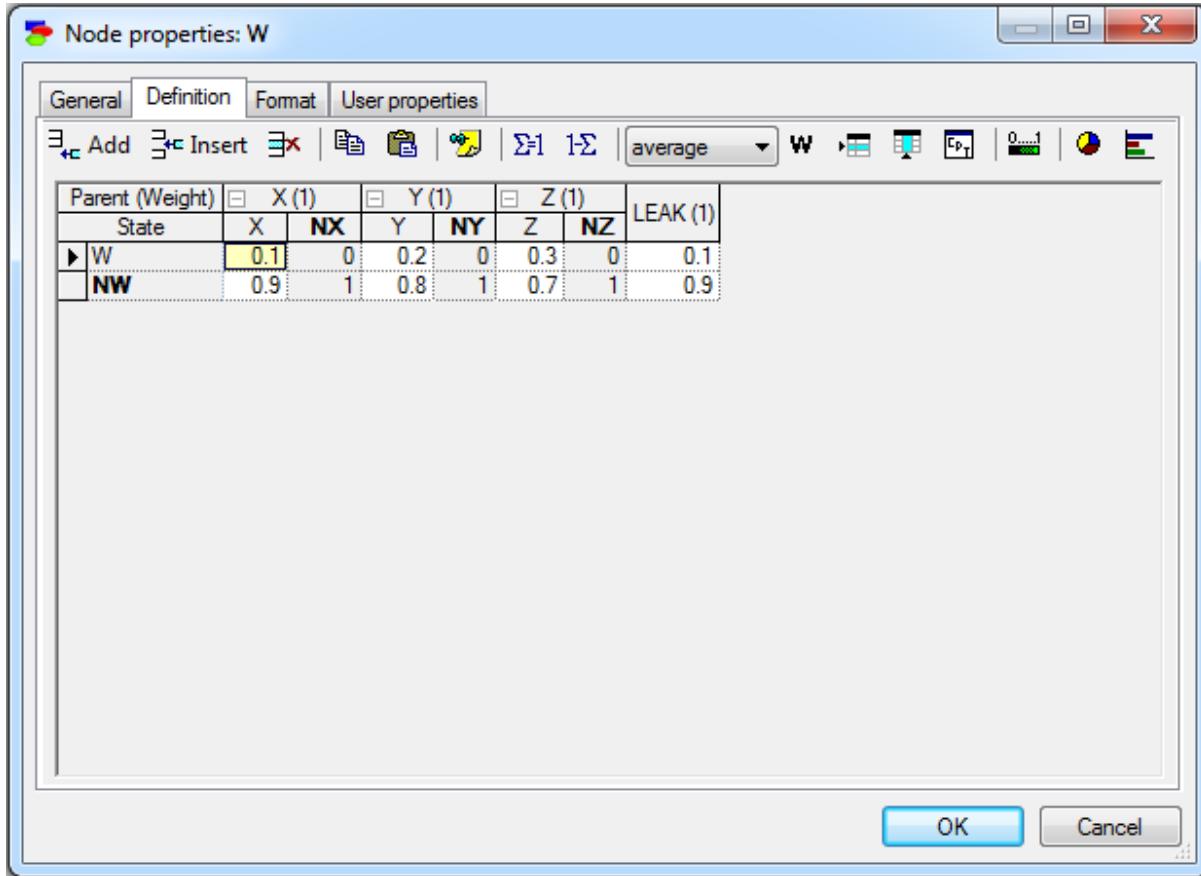
Most of this dialog resembles the dialog for the *Chance-General* nodes and we will not touch upon it. The additional functionality, which we will discuss below, allows for *NoisyMax*-specific activities. The buttons *Show net parameters* () and *Show compound parameters* () switch between two representations of the Noisy-MAX parameters: net and loaded. Only one of the two buttons can be pressed at a time.

The *Show 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 for the purpose of this interaction and not in the definition of the parent variable. Please see the [Canonical models](#)⁸⁷ section for a tutorial-like introduction to the Noisy-MAX gate.

Chance-NoisyAdder nodes

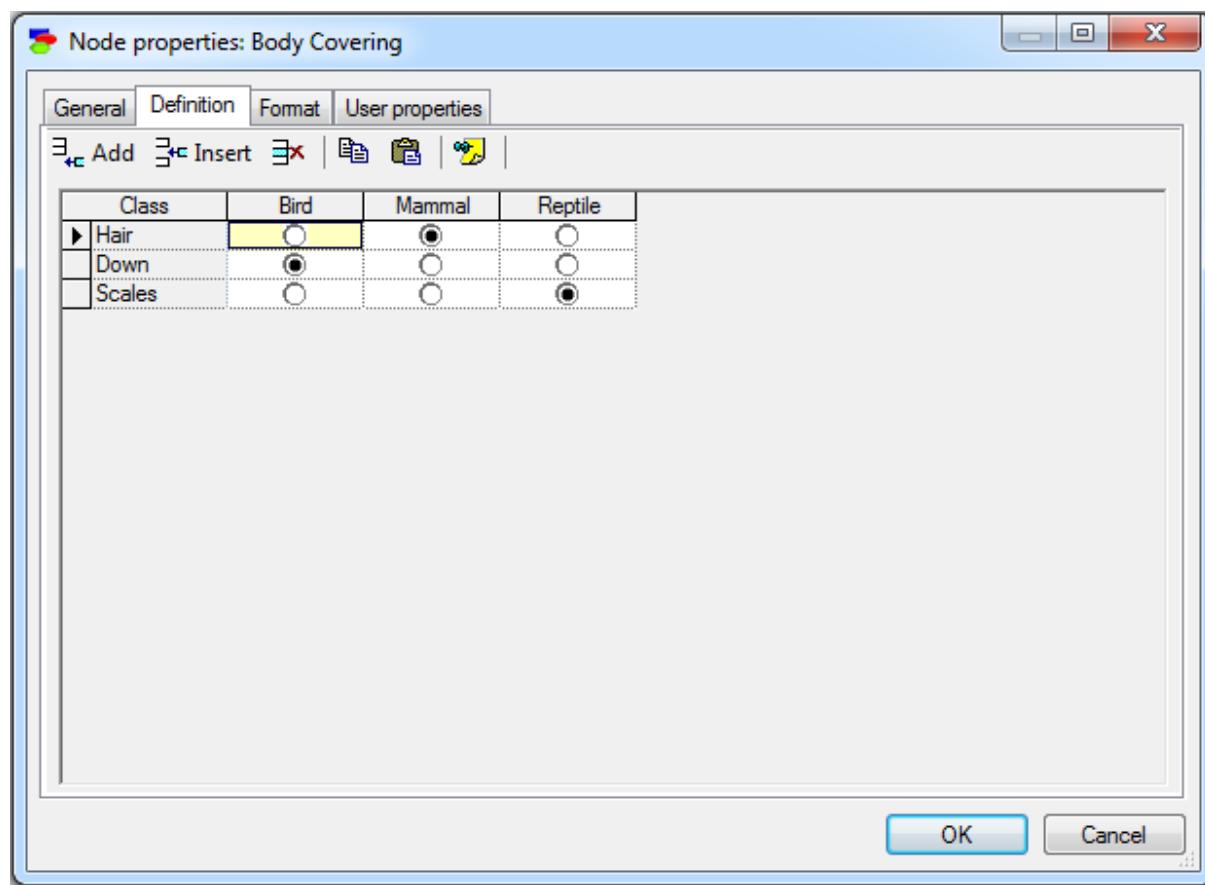
The *Noisy-Adder* nodes are a special case of chance nodes modeling discrete random variables using the noisy-adder assumption. Noisy-adder is a non-decomposable model that derives the probability of the effect by taking the average of probabilities

of the effect given each of the causes in separation. Similarly to the *NoisyMax* 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 [Canonical models](#)⁸⁷ section for a tutorial-like introduction to the *Noisy-Adder* nodes.



Deterministic nodes

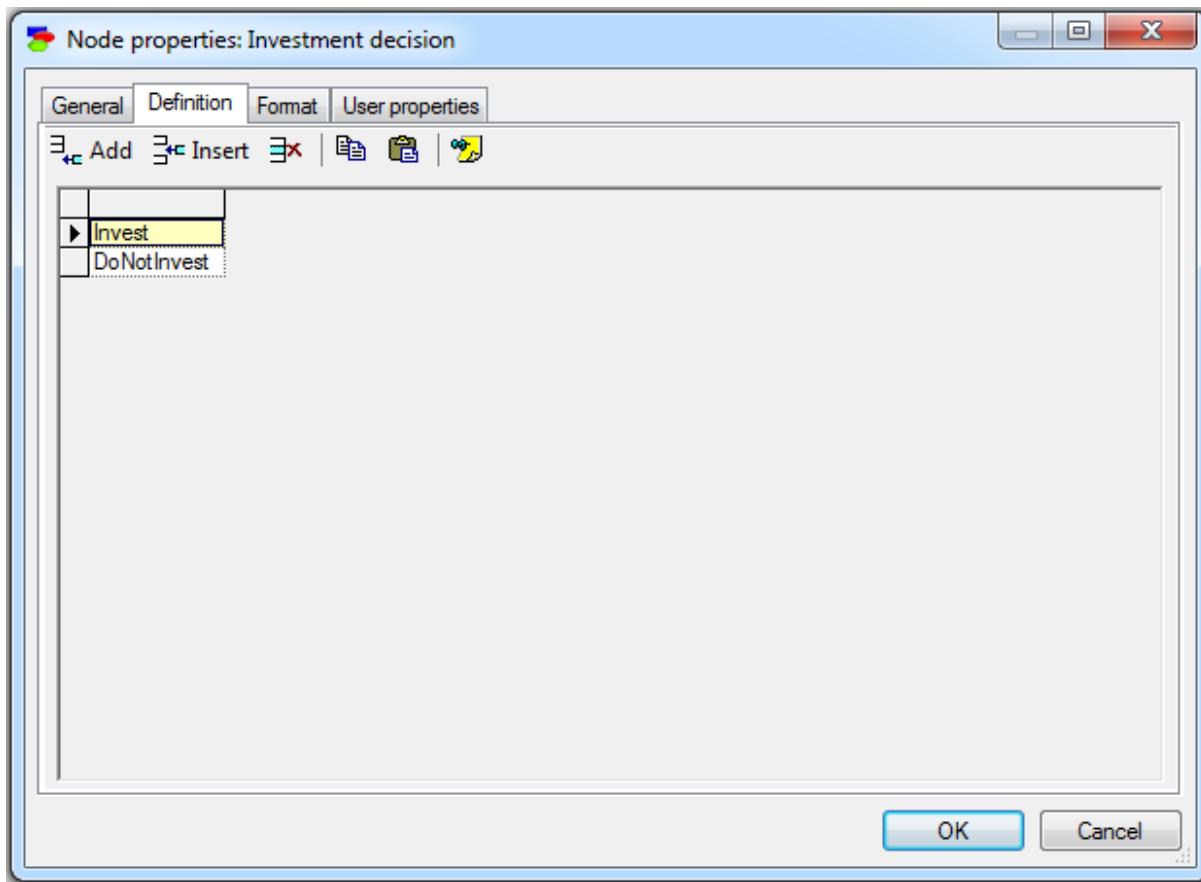
Deterministic nodes are discrete nodes that have no noise in them, i.e., we know their state with certainty if we know the states of their parents. The definition of a deterministic node is a truth table - knowing the values of the parents of a deterministic node defines its state. The only difference 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 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

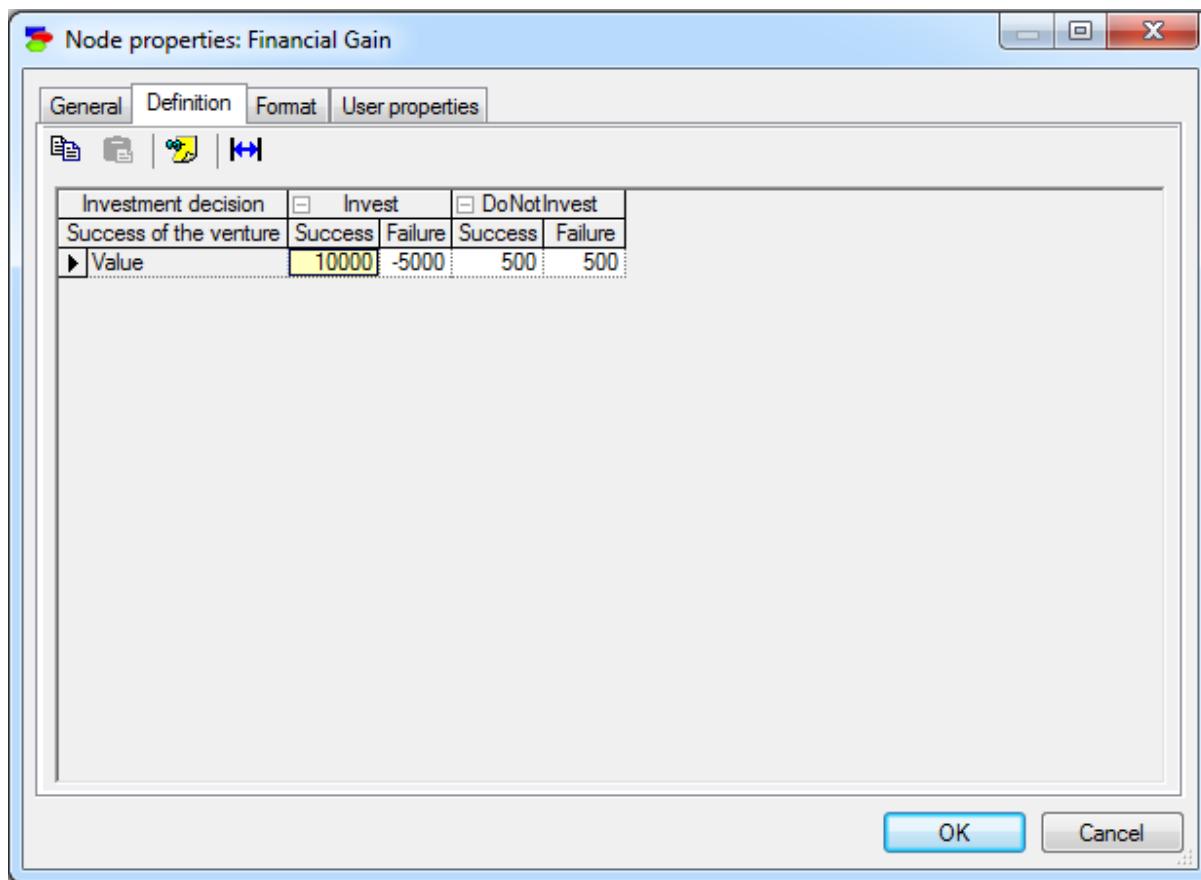
Decision nodes are discrete nodes that are under control of the decision maker and, hence, have no parents influencing them and no numerical specification. The definition of a *Decision* node is a list of labels - one of these labels will be chosen by the decision maker.



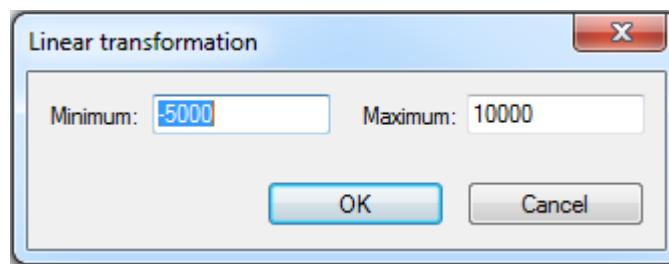
All editing actions are the same as in *Chance-general* nodes.

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.



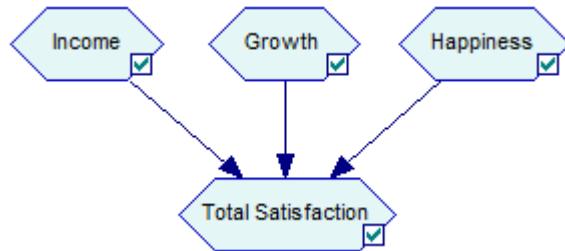
In the example above, node *Financial Gain* expresses the monetary gain for all combinations of states of the nodes Investment decision and Success of the venture. In expected utility theory, decisions are optimal when they maximize the expected utility. It turns out that the maximization process is 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:



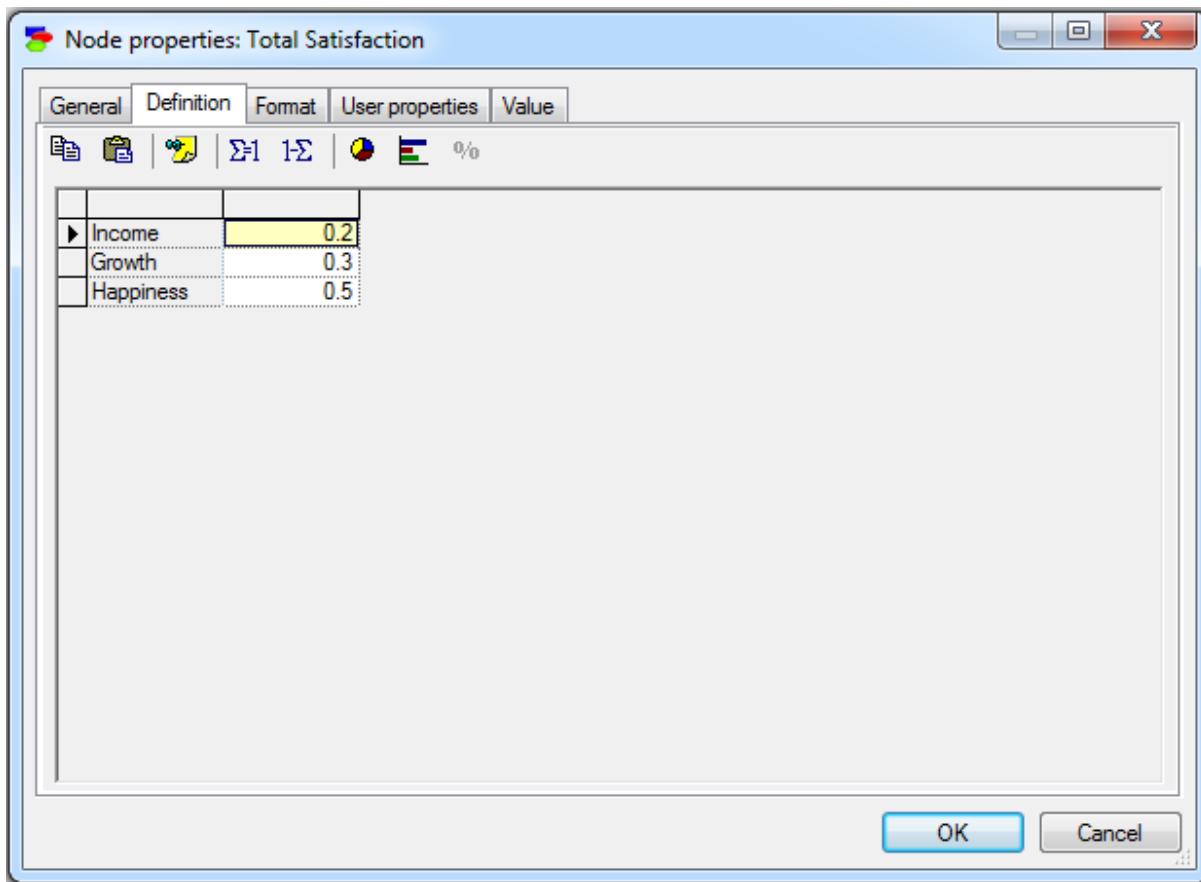
The fields *Minimum* and *Maximum* contain the lowest and the highest value in the table, respectively. When these values are edited and the *OK* button is pressed, the values in the table are linearly transformed to the new interval. This operation is useful in case the numbers in the table are utilities. 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*.



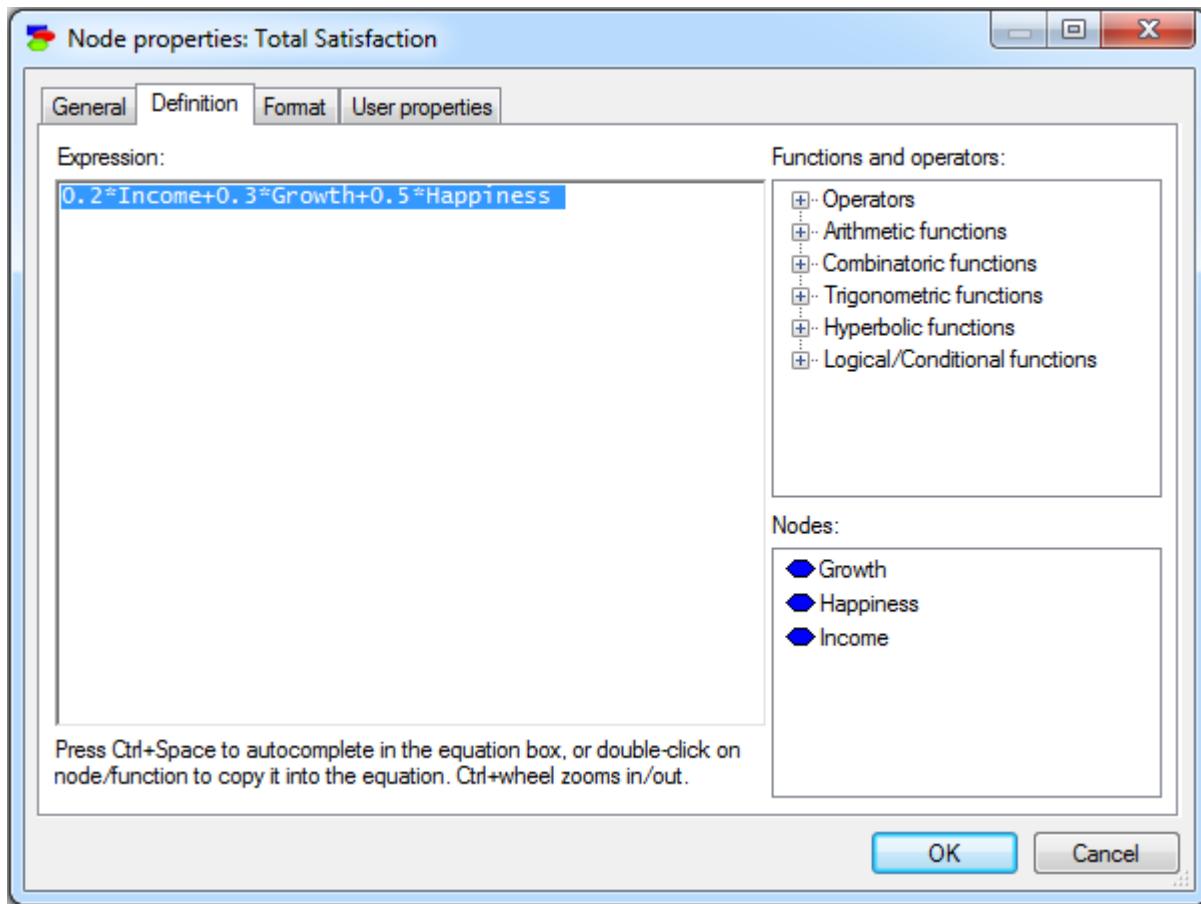
Let us assume that these combine linearly with weights 0.2, 0.3, and 0.5, respectively, i.e., we are dealing with the following function of the utilities of the three components: $\text{Total Satisfaction} = 0.2 \text{ Income} + 0.3 \text{ Growth} + 0.5 \text{ Happiness}$. The following definition of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



All editing actions are the same as in *Chance-general* nodes. In particular, the *Normalize* ($\Sigma 1$) and *Complement* (1Σ) 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:

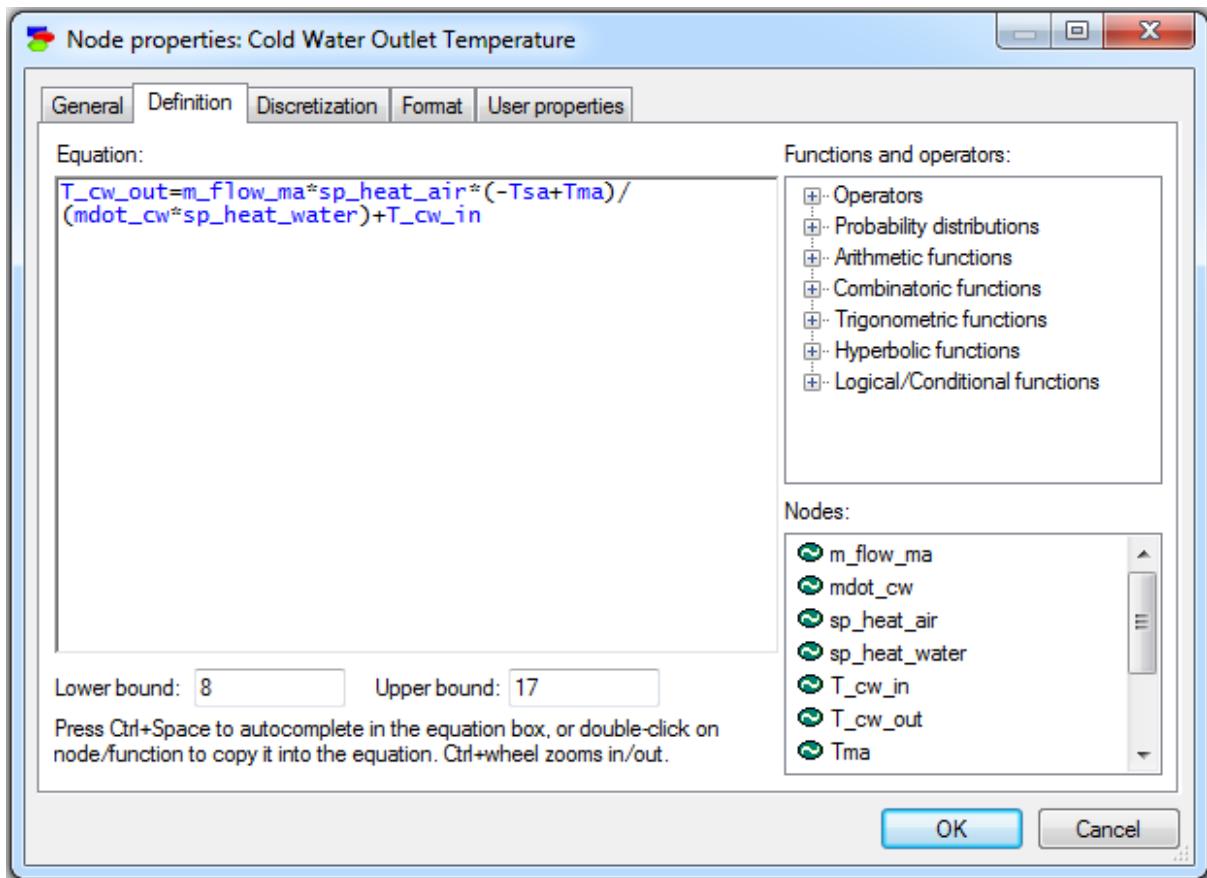


The dialog allows for entering any function of the parent utilities. The current definition shows the additive linear function corresponding to the example used for the *ALU* nodes. In addition to arithmetic operators, GeNIE offers a number of arithmetic, combinatoric, trigonometric, hyperbolic, and logical/conditional functions, which can be typed directly or selected from the lists in the right-hand side window pane.

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:

$$T_{cw_out} = mflow_{ma} * sp_heat_{air} * (-T_{sa} + T_{ma}) / (mdot_{cw} * sp_heat_{water}) + T_{cw_in}$$

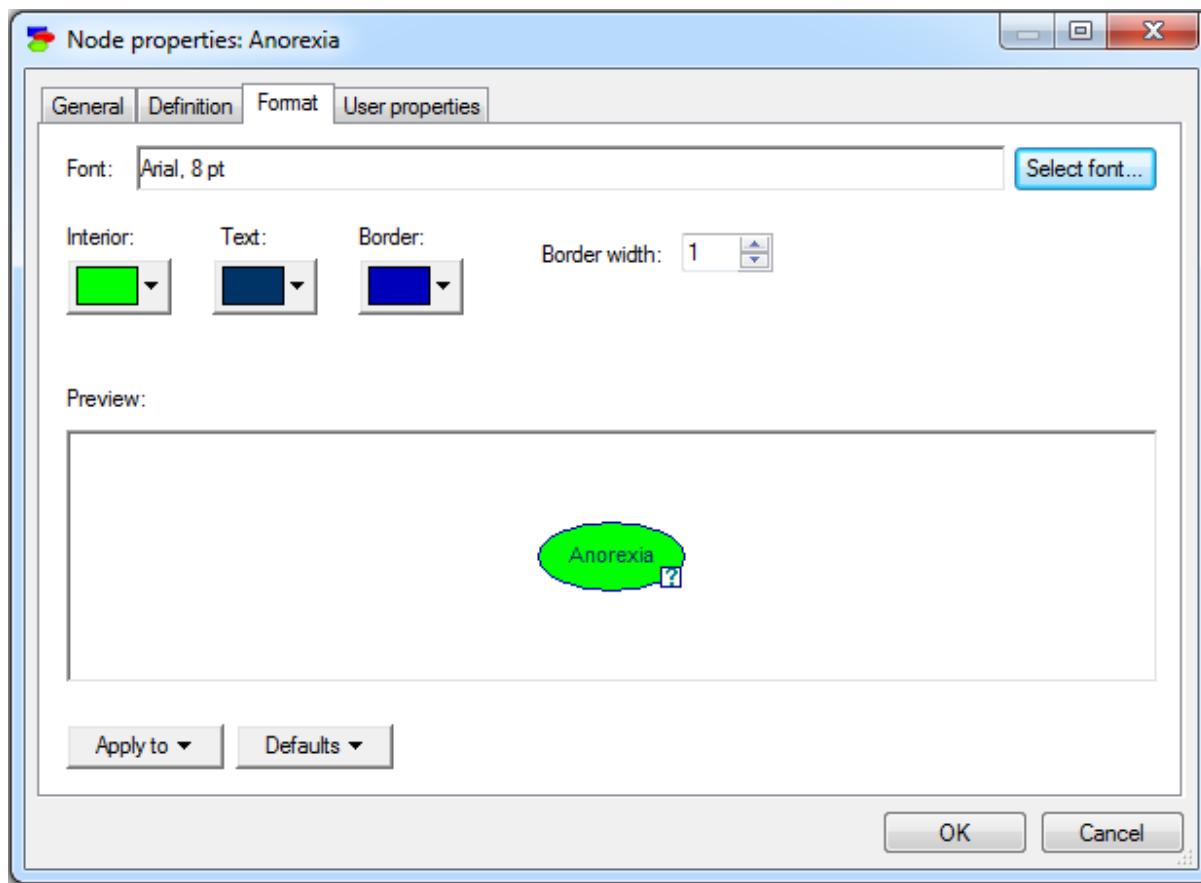


The window panes on the right-hand side of the tab contain a set of standard functions known by GeNIE and the set of nodes that participate in the equation.

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.

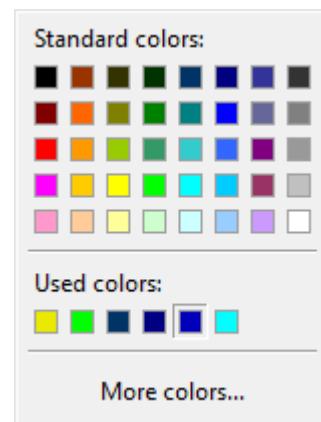
Format tab

The *Format* tab is used to specify how the node will be displayed in the [Graph View](#)^[60]. It has a preview window which displays the altered view of the node with the new settings.

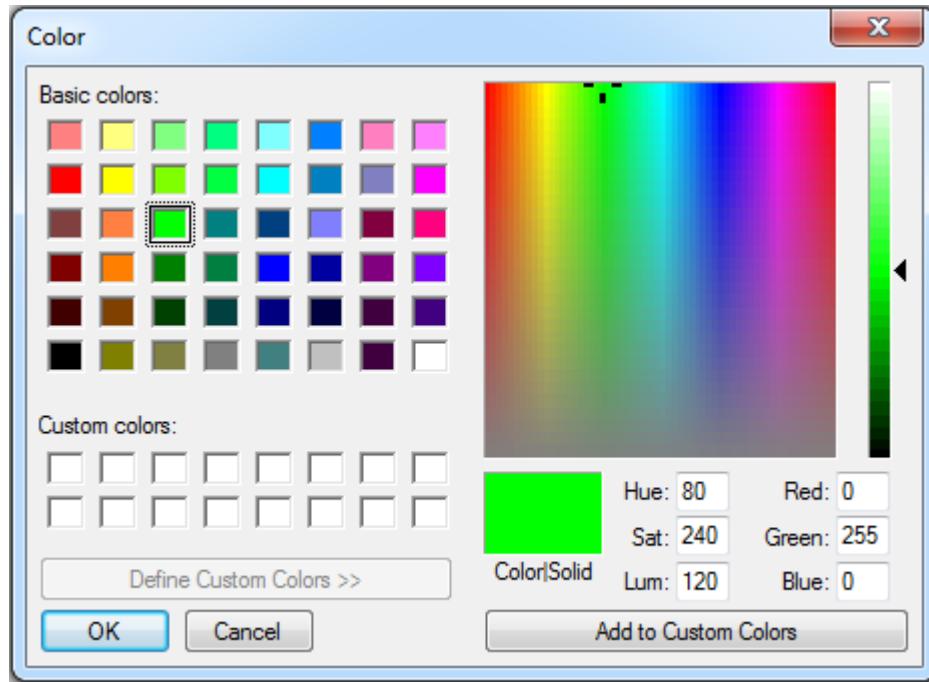


Select font button is used to select the font of the text displayed within the node. Clicking on this button will cause a standard *Font Selection* dialog box to be displayed. You can select font type, style, and size from this box.

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:

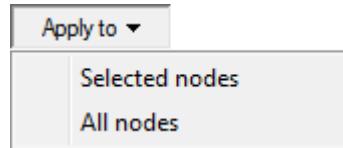


You can select any color on the rainbow-like part of the window and then add the selected color to one of the 16 custom colors (by clicking on the button *Add to Custom Colors*).

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.

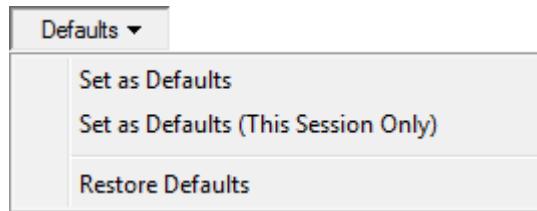
The Node properties dialog is powerful but unless you want the changes to apply only to one node that you have used to invoke the dialog, you need to make careful choices in two separate menus: the *Apply to* menu and the *Defaults* menu.

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 *Selected nodes*.

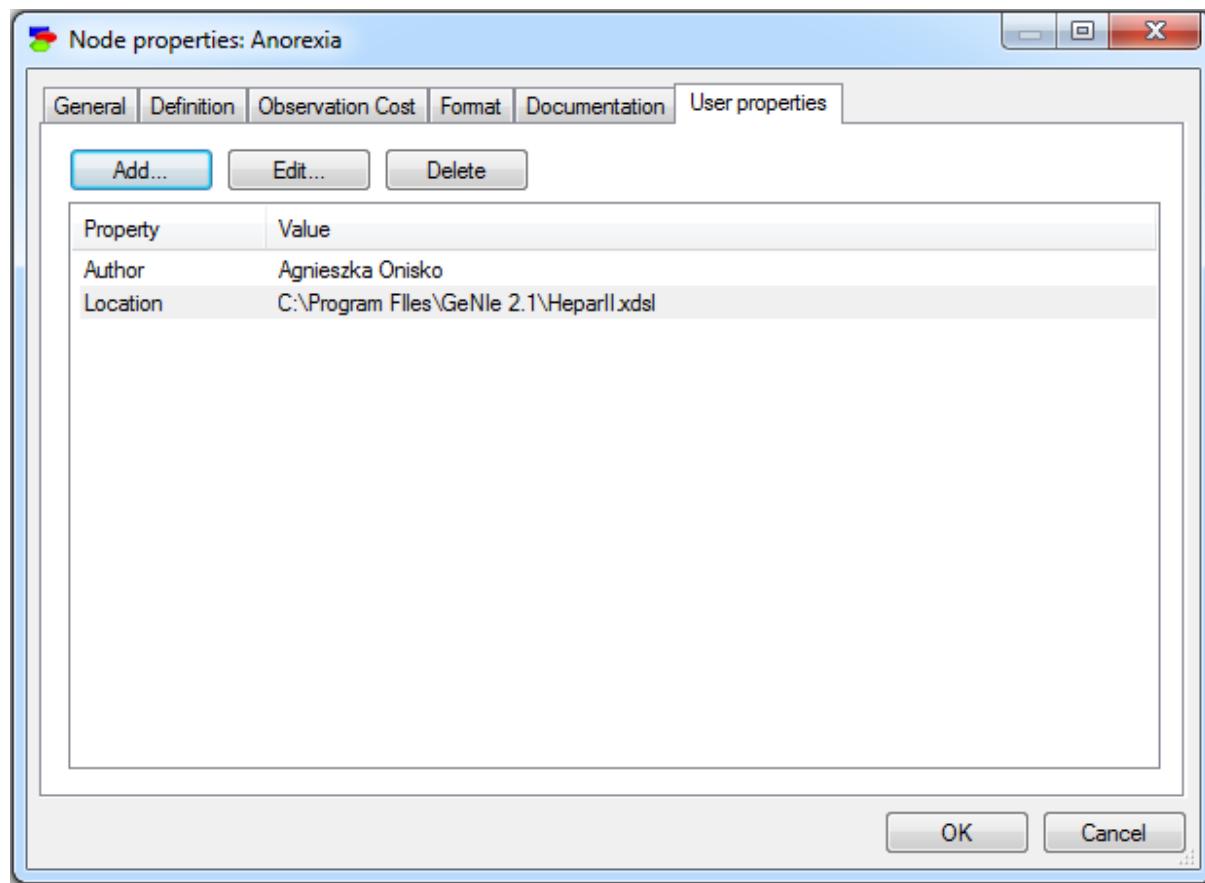
The *Defaults* menu allows you to modify the program defaults, which will make all new nodes appear the way you have specified. The menu gives you three choices: *Set as Defaults*, *Set as Defaults (This Session Only)*, and *Restore Defaults*.



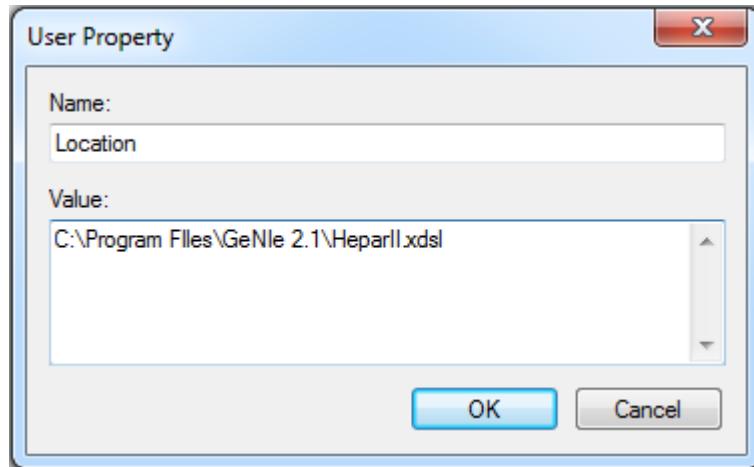
Set as Defaults set the new format as the default format for all new nodes. This default setting will be maintained between multiple sessions of GeNIE. *Set as Defaults (This Session Only)* makes the new setting a default but only for this session only. After you quit and start GeNIE again, the previous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

User properties tab

The *Properties* tab allows the user to define properties of the node that can be later retrieved by an application program using [SMILE³¹](#). 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:

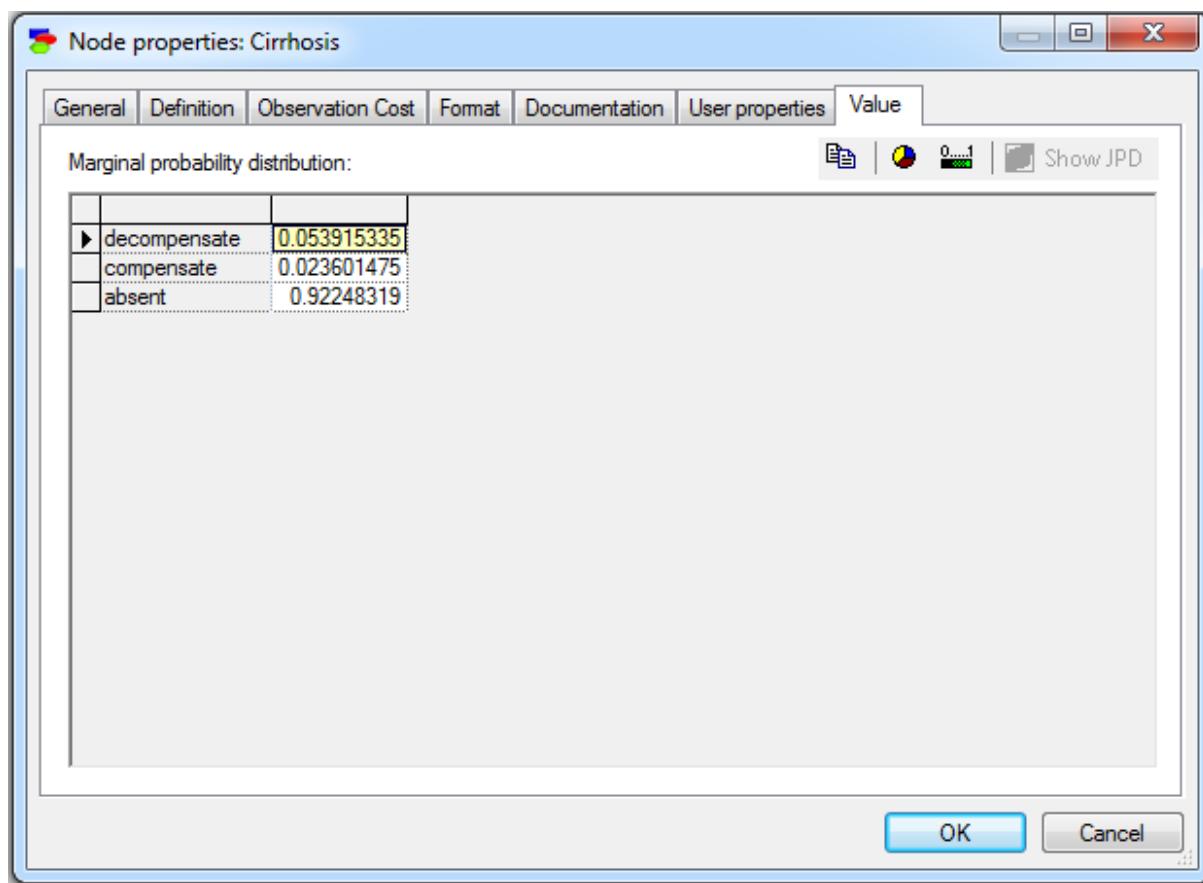


Delete removes the selected property.

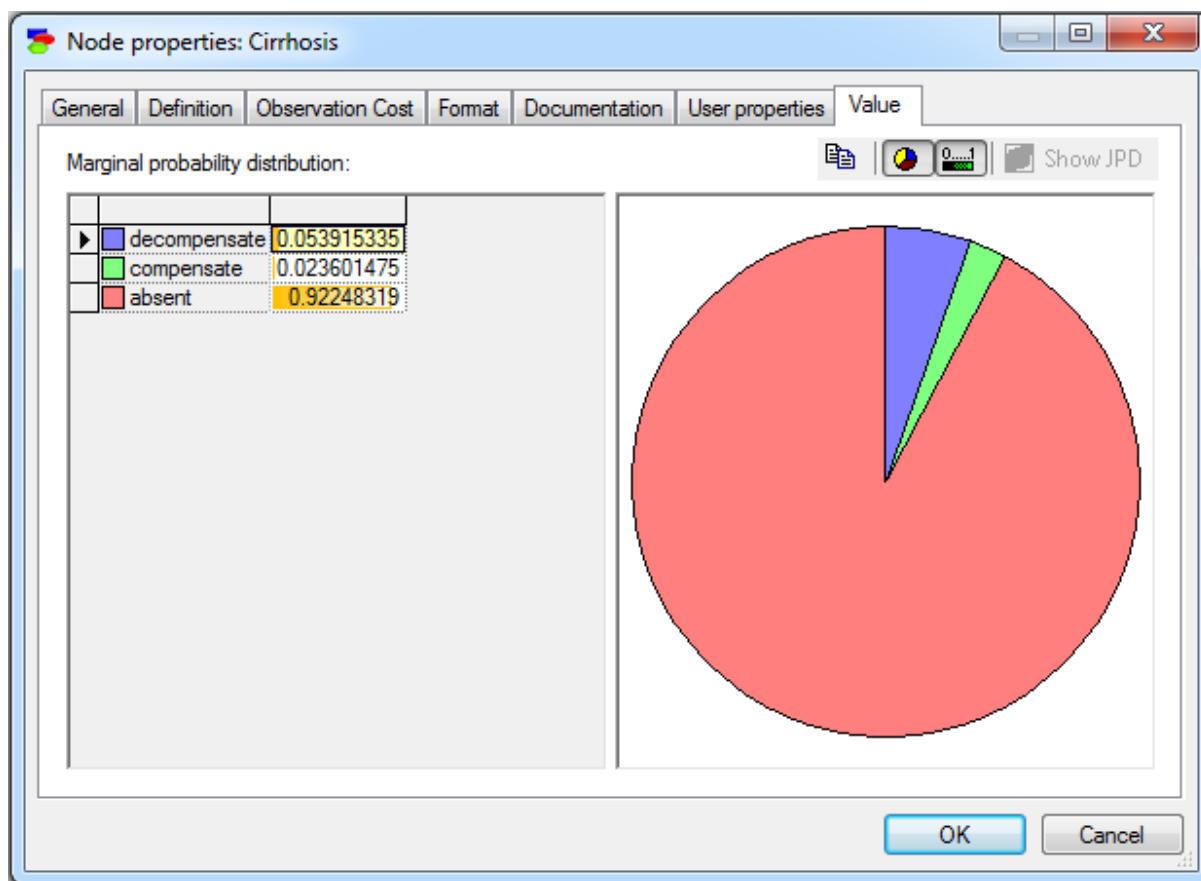
Value tab

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.

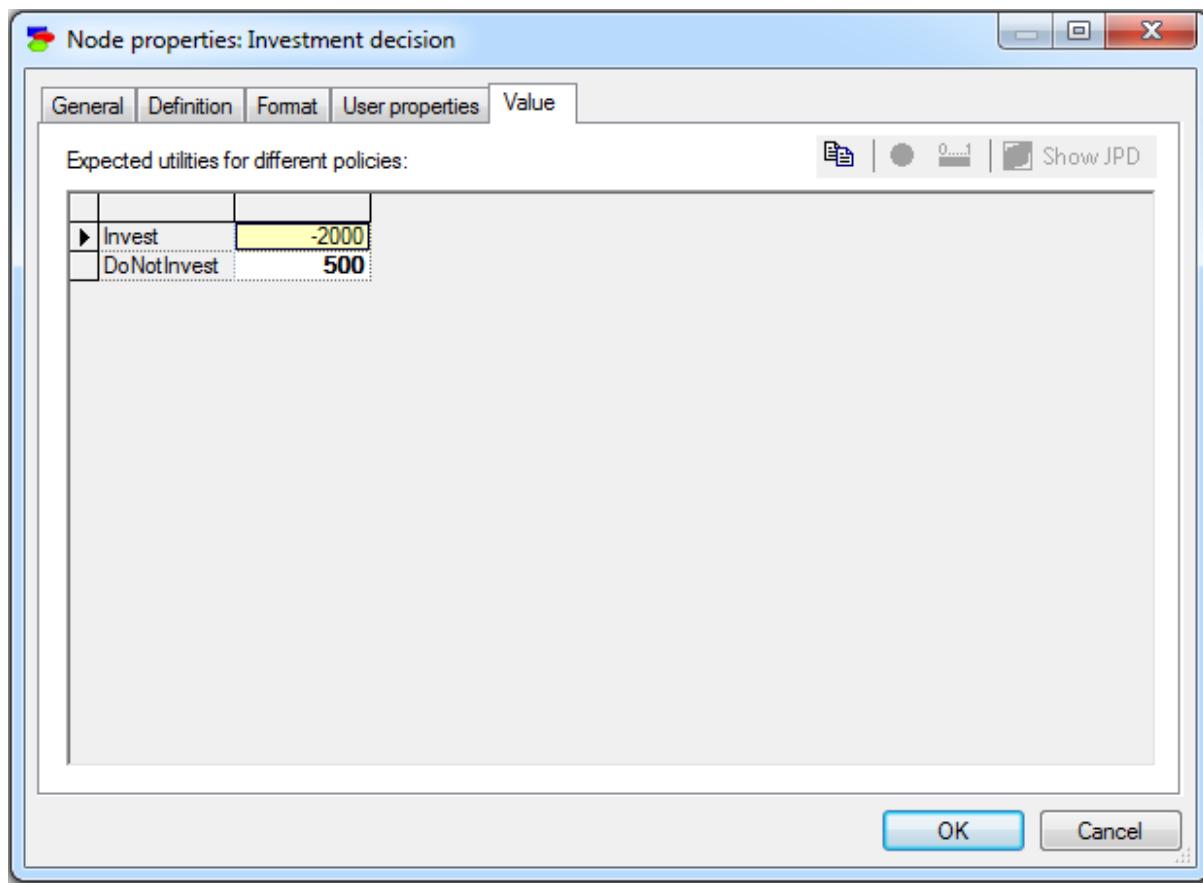
Discrete *Chance* and *Deterministic* nodes show their marginal probability distributions.



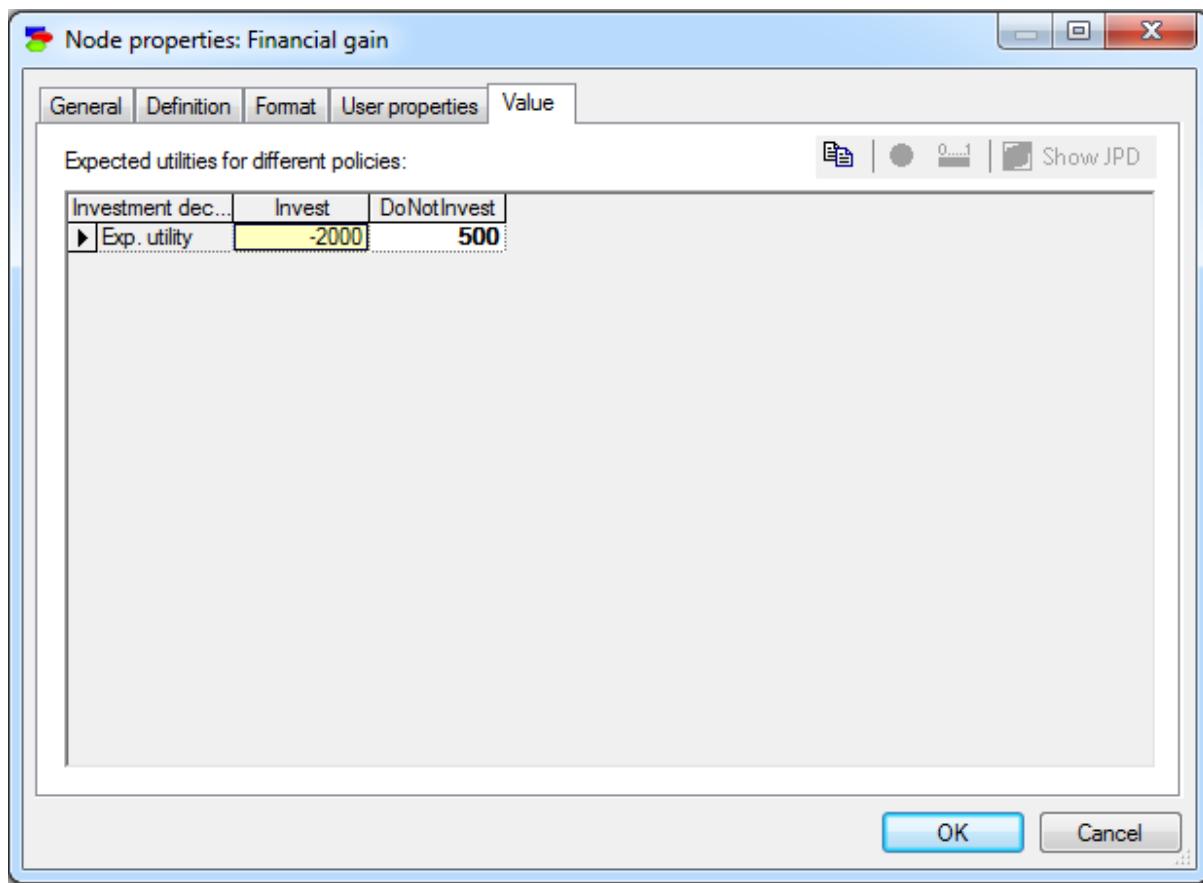
To make the display graphical, press the *Piechart* (Pie chart icon) and/or the *Show QuickBars* (QuickBars icon) buttons. They add a piechart display of the posterior marginal distribution and bars in the result spreadsheet (a list of states with their posterior probabilities) respectively.



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:

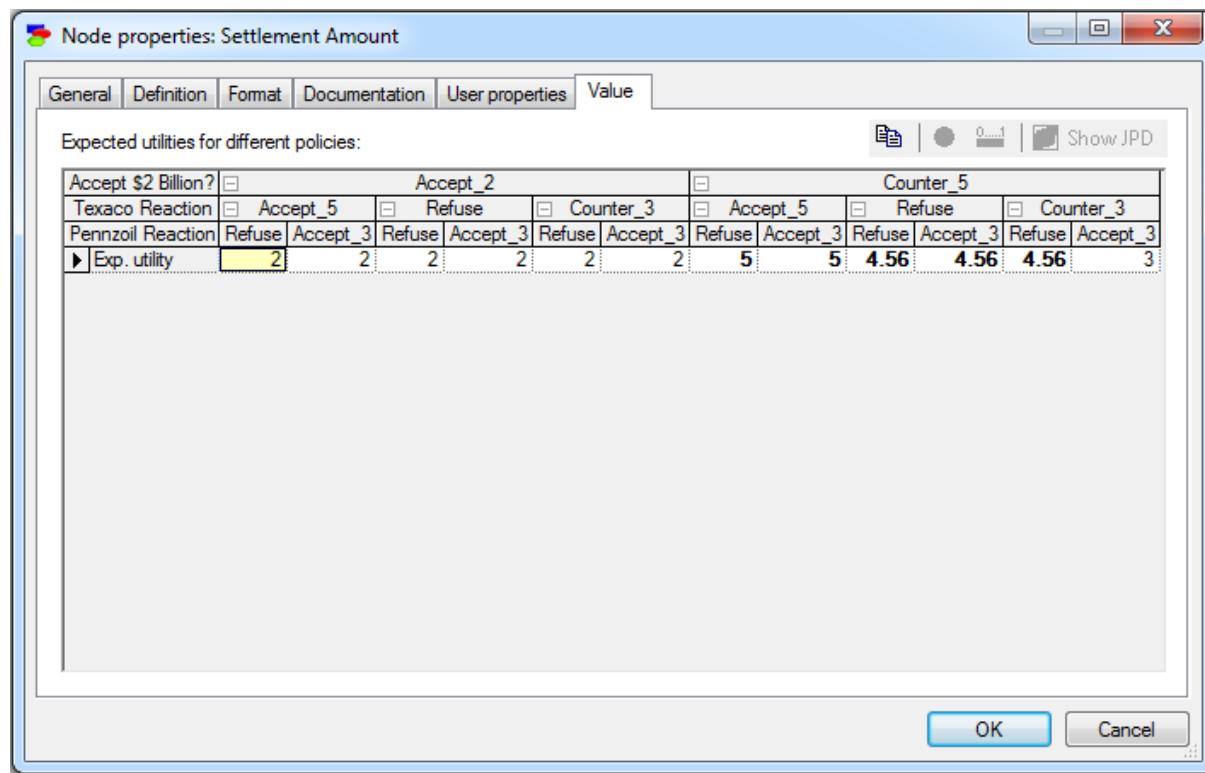


When there are unobserved decision nodes that precede the current node or when there are unobserved chance nodes that should have been observed, GeNIE shows the results as a table indexed by the unobserved nodes.

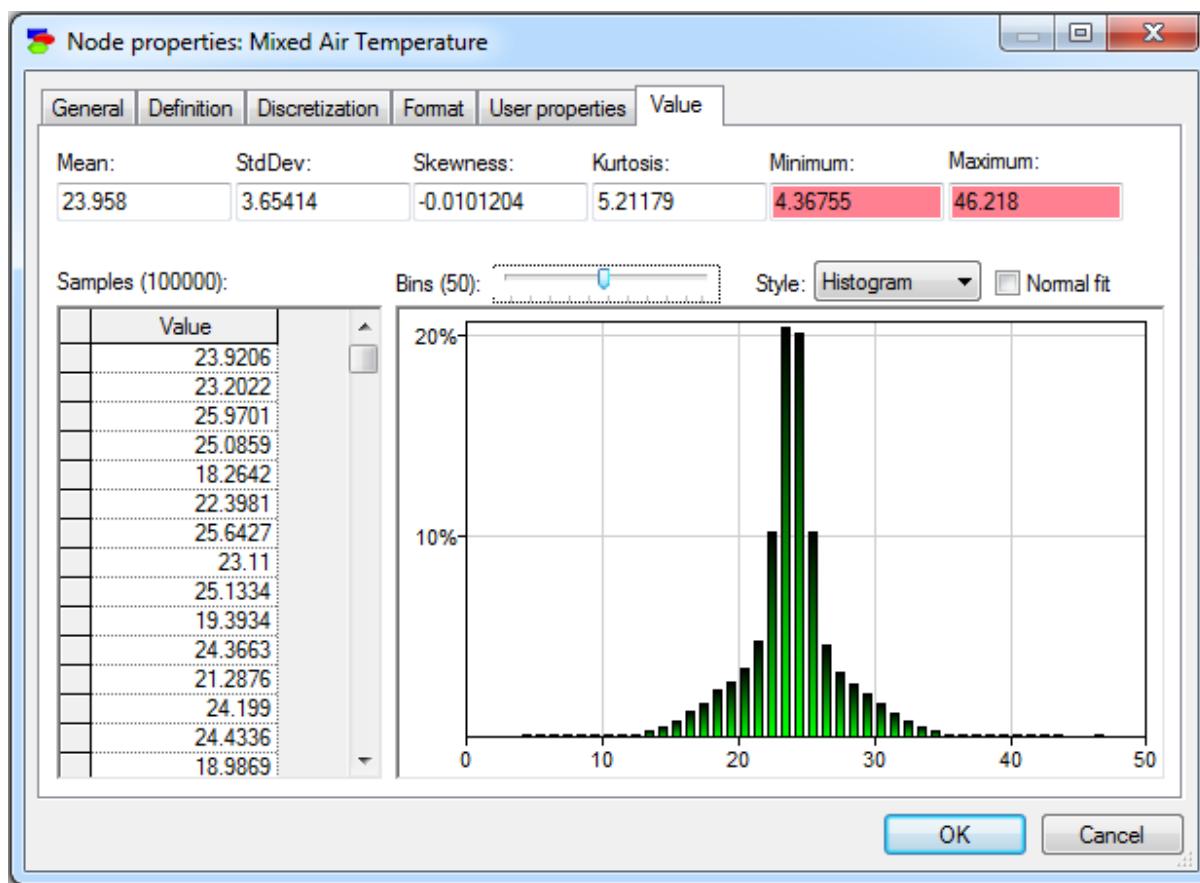
Expected utilities for different policies:						
Accept \$2 Billion?	Accept_2	Refuse	Counter_3	Accept_5	Refuse	Counter_3
Texaco Reaction	Accept_5	Refuse	Counter_3	Accept_5	Refuse	Counter_3
► Refuse	2	2	2	5	4.56	4.56
Accept_3	2	2	2	5	4.56	3

OK Cancel

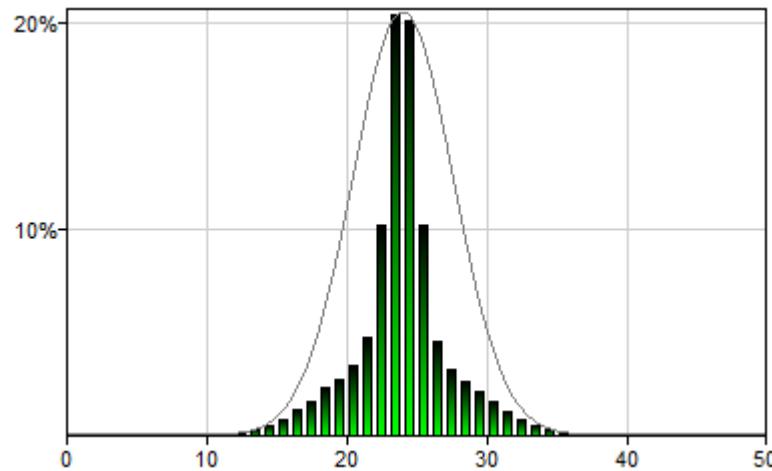
The same information shown in the *Utility* node of the same model:



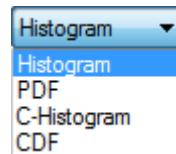
Equation nodes are continuous and show the results in form of a plot of the samples obtained during the most recent run of the sampling algorithm. The tab shows the first four moments of the distribution: *Mean*, *StdDev*, *Skewness* and *Kurtosis* along with the *Minimum* and the *Maximum*. By default,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



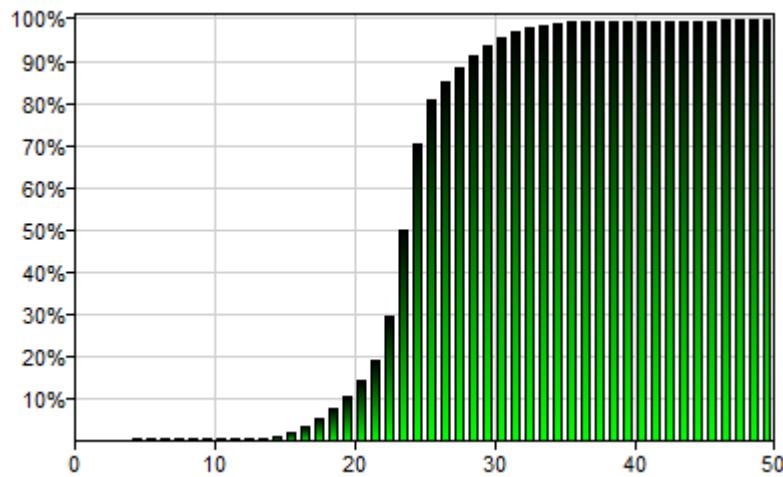
The samples themselves are preserved and displayed in the vector on the left-hand side. Similarly to the histogram interface in the data pre-processing module, the user can change the number of bins in the histogram. *Normal fit* check box draws a Normal distribution over the domain of the variable with the mean and standard deviation equal to those of the samples.



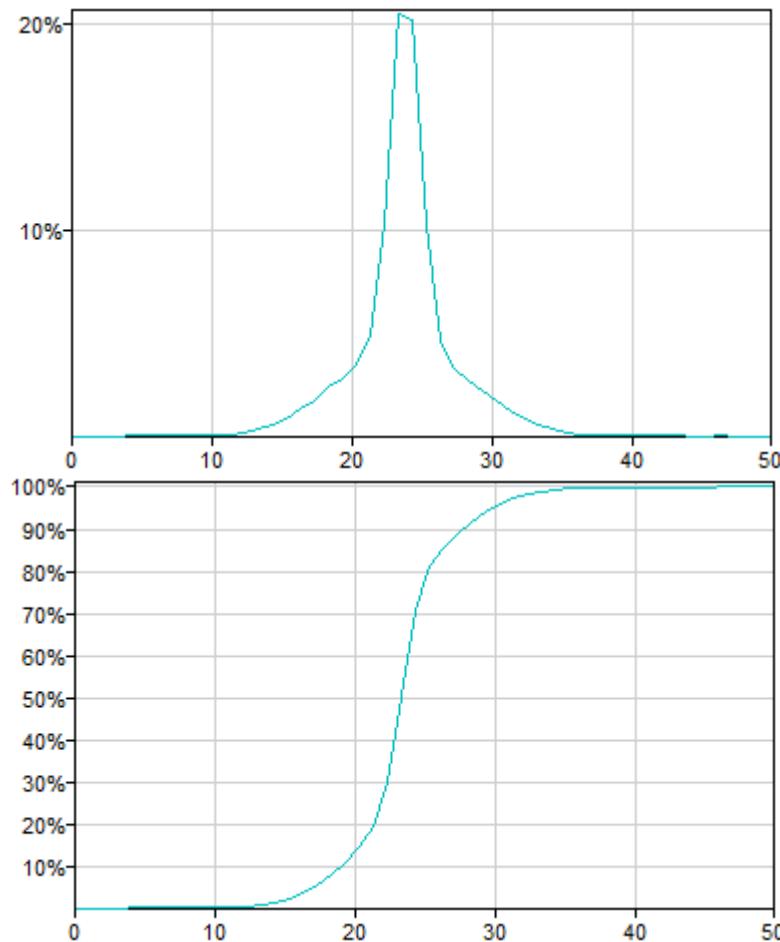
Style pop-up menu allows for choosing a different plot:



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



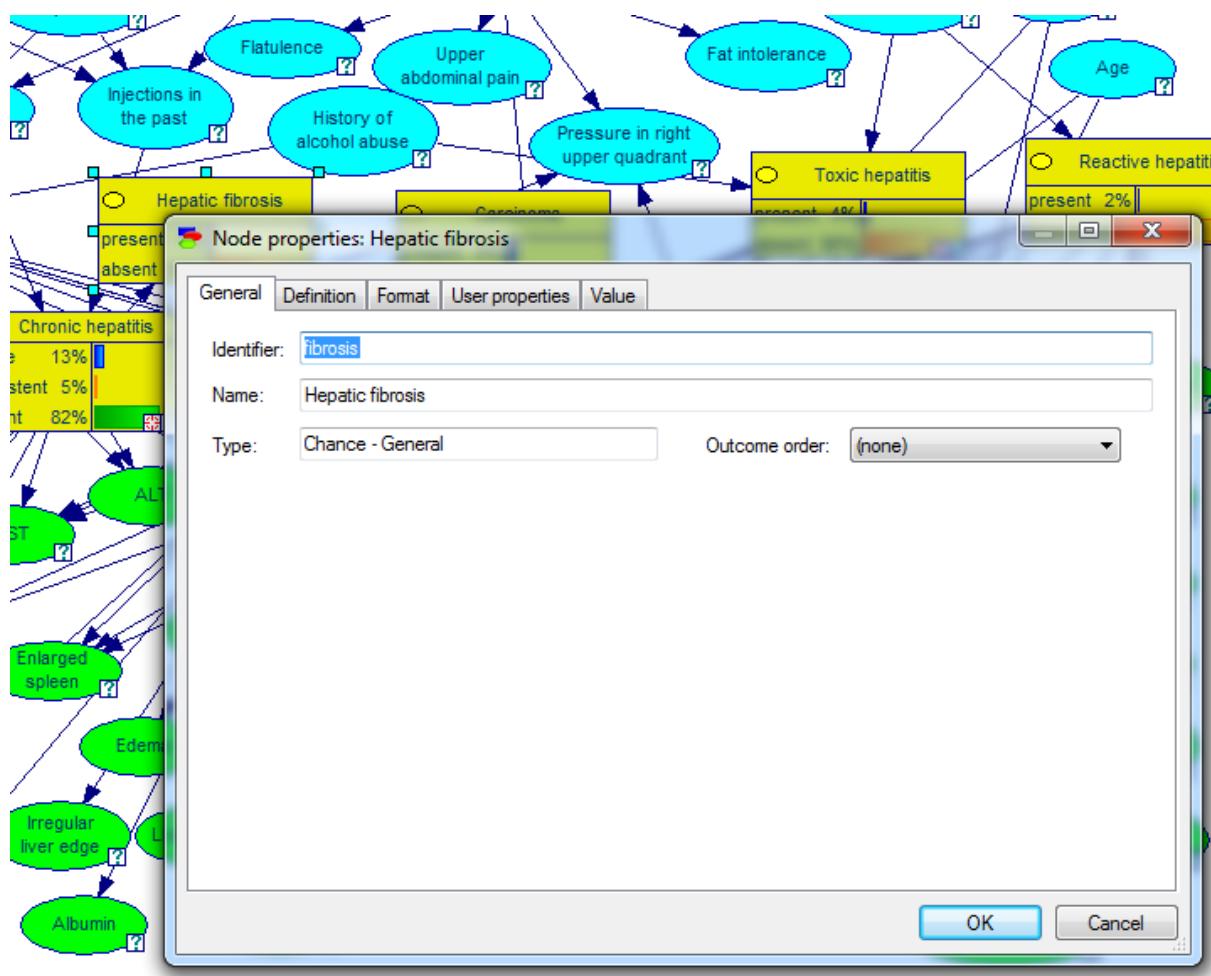
PDF and *CDF* are abstractions of the two histogram plots:



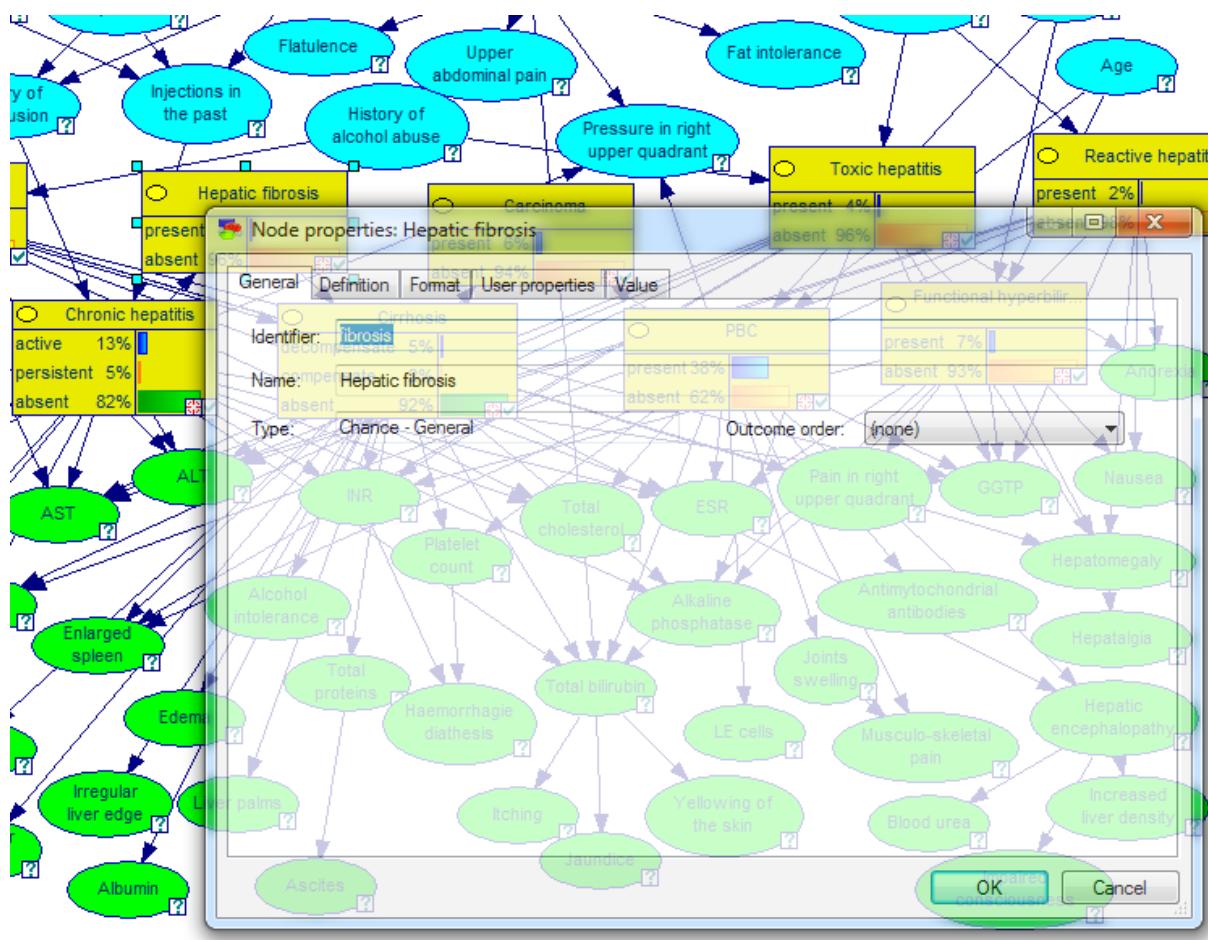
With the *AutoDiscretize* algorithm used for inference in continuous models, the *Value tab* of *Equation* nodes is identical to those of *Chance* nodes.

Transparent mode

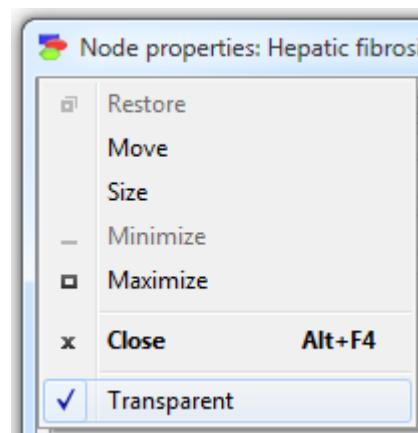
By default, the node property sheets are displayed in opaque mode, i.e., when open, they cover whatever is under them. Here is a screen shot from the Hepar II model



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 hold 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 is the bottleneck of this business, 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

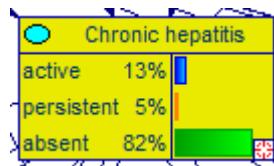
We review in this section two important aspects of viewing nodes in the *Graph View*.

Icons and bar charts

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

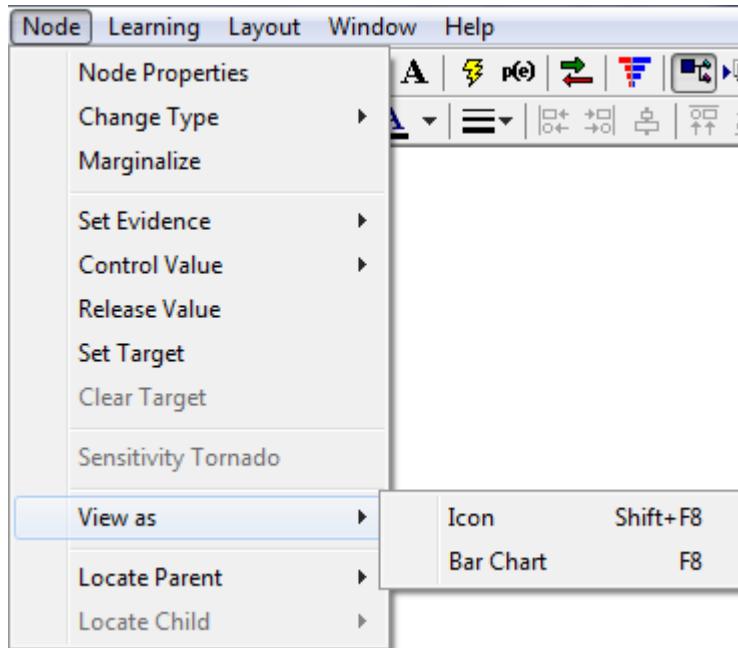


and as a bar chart, which displays graphically the node's marginal probability distribution

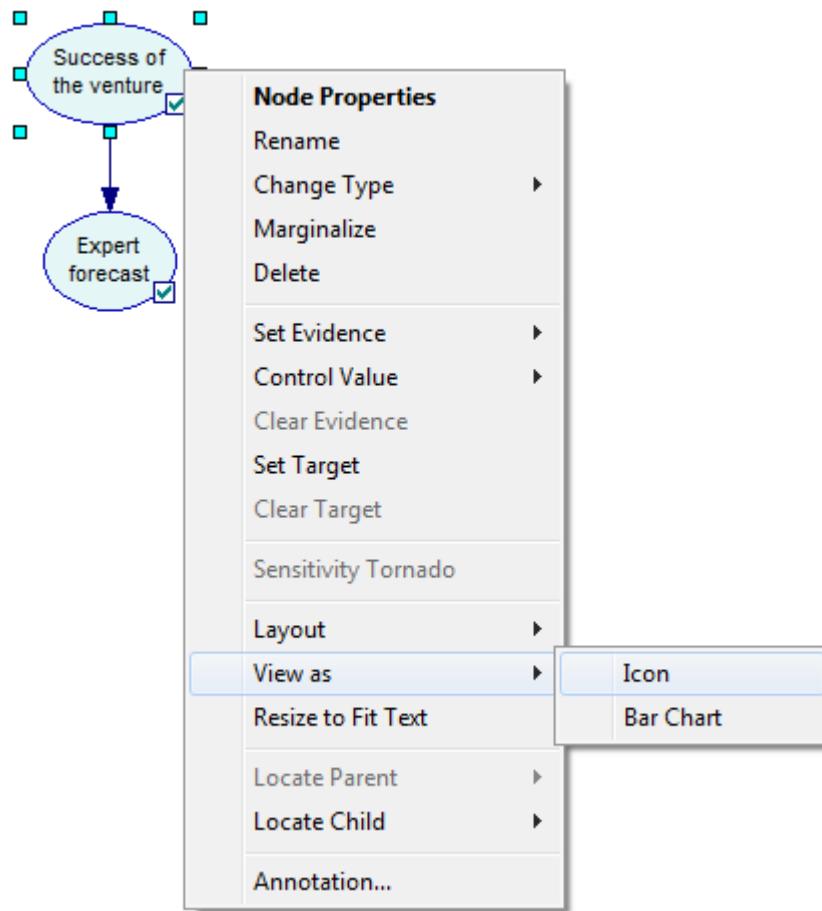


The advantage of seeing a node as a bar chart is that we can see at any point in time its marginal probability distribution. The disadvantage is that a bar chart takes more

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*

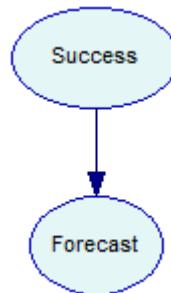


The same can be accomplished by means of the *Node Pop-up* menu

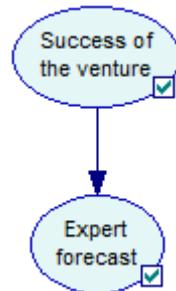


Names and identifiers

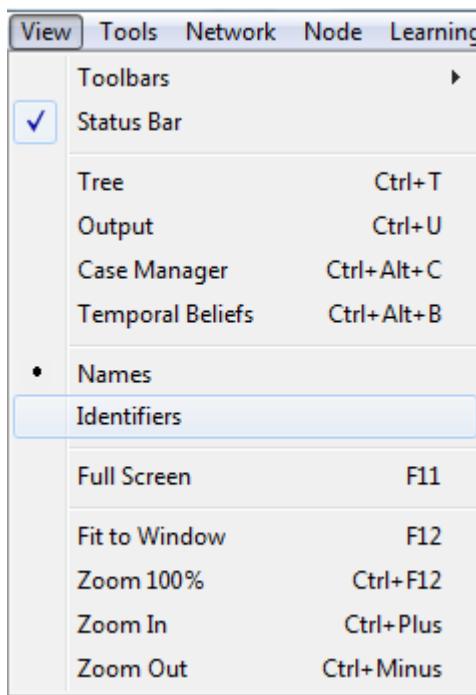
There is another important element of viewing nodes: Viewing their IDs or their names. IDs are short and play the role of variable names. GeNIE uses them for the purpose of compatibility with equation-based variables, where reference to other nodes in a node's definition has to be through a unique identifier. Identifiers have to start with a letter followed by any combination of letters, digits, and underscore characters. 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.



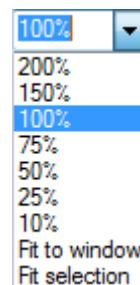
To switch between IDs and names, please use the *View Menu*



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

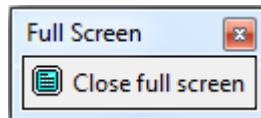
Zoom () is one of the navigational tools and allows for zooming in and out the *Graph View* by clicking with the left mouse button and the right mouse button respectively. An alternative way of zooming is through rolling the mouse wheel while holding down the *SHIFT* key. Yet another is pressing *CTRL+-* 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* (F11) 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* or *F11* keys.

5.5.4 Format toolbar and Layout menu

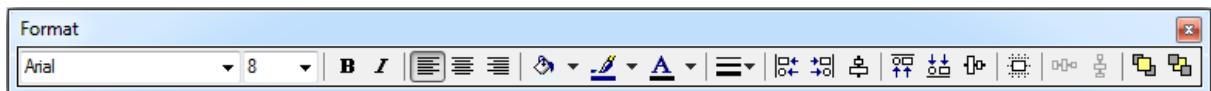
Note: All operations performed using the *Format* toolbar are applicable to the currently selected item in the *Graph View*. All new items created follow the current settings on the *Format* toolbar. *Tree View* cannot be changed using the *Format Toolbar*.

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](#)¹³⁸ sheet and in [Options](#)²⁴¹.

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.

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

Center horizontally aligns the object centers.

Align top, *Align bottom*, and *Center vertically* have analogical functions.

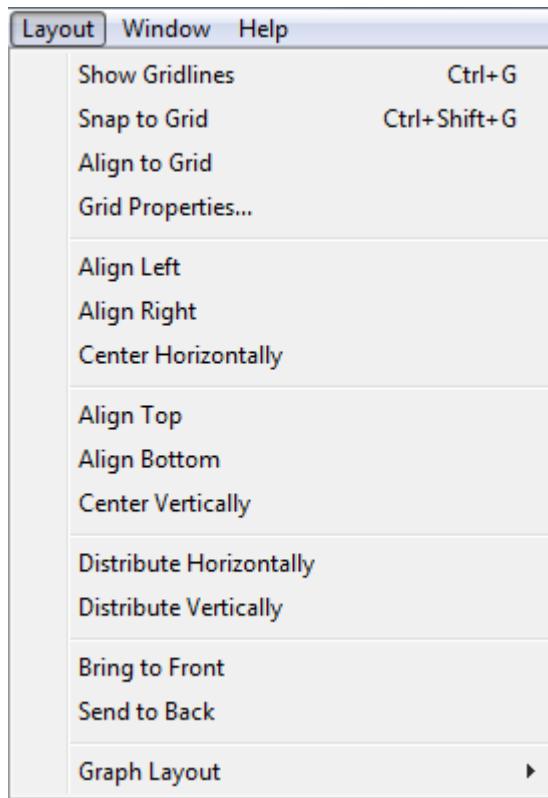
Align to grid aligns the center points of the selected objects to the nearest intersection of the grid lines. At least one object must be selected for this option to be active.

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



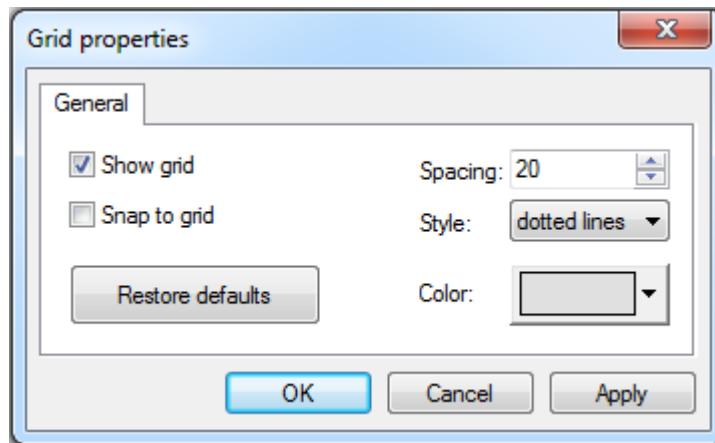
There are several additional functions offered by the *Layout* menu that are not offered by the *Format* toolbar:

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](#)⁶⁰. 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



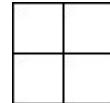
Show grid checkbox has a same function as the *Show Gridlines* option in the *Layout* menu. When checked, the grid lines are displayed in the *Graph View*.

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.

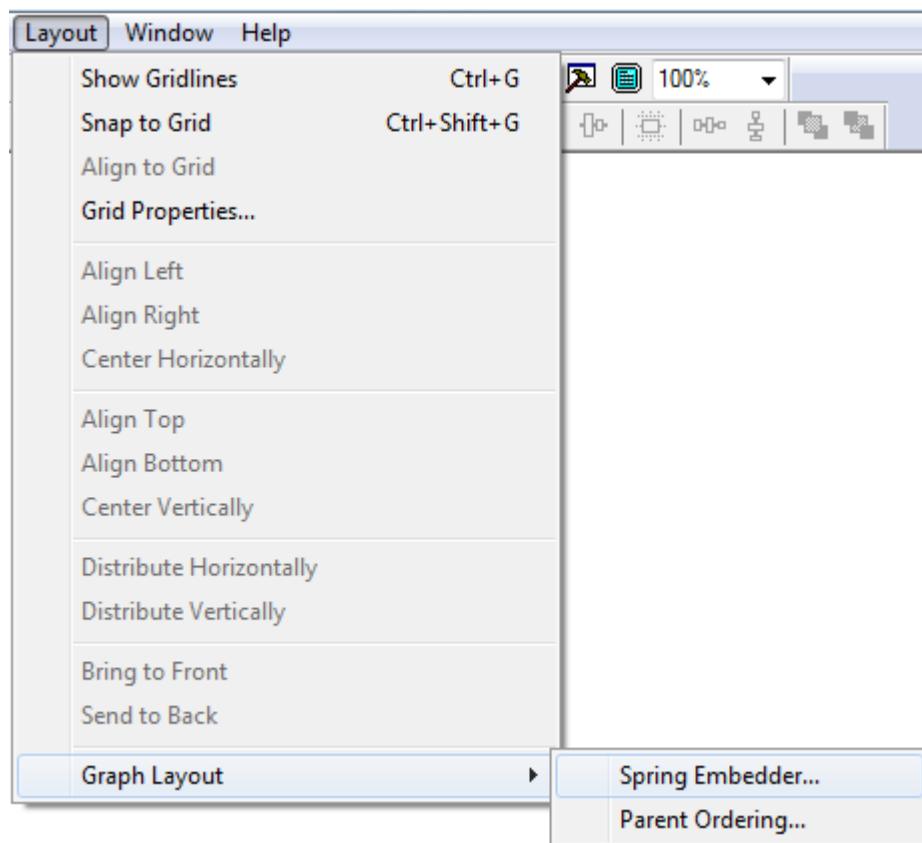
Color sets the color of the grid lines. You can select any color from the palette or define your own color. Grid color selection is similar to color selection for nodes.

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.

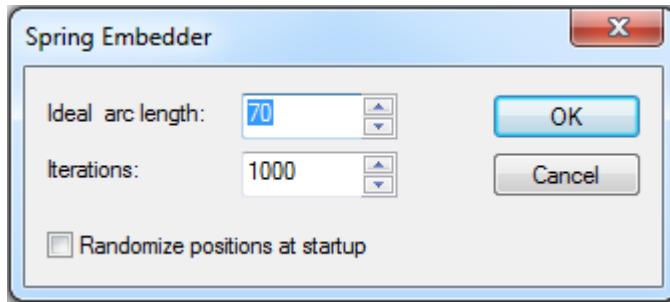


Spring embedder

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:



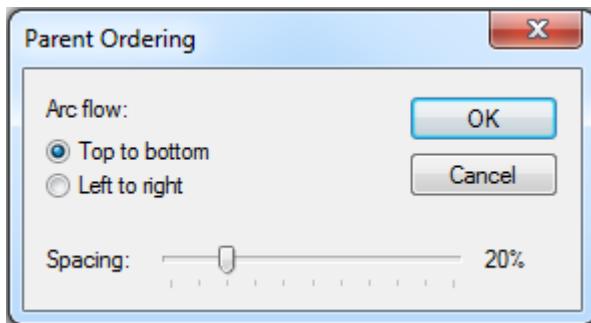
Ideal arc length specifies the ideal distance (in pixels) between two interconnected nodes. Please note that not all arcs will be of this length, the algorithm may modify this length so as to reduce overlaps.

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 receives 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.

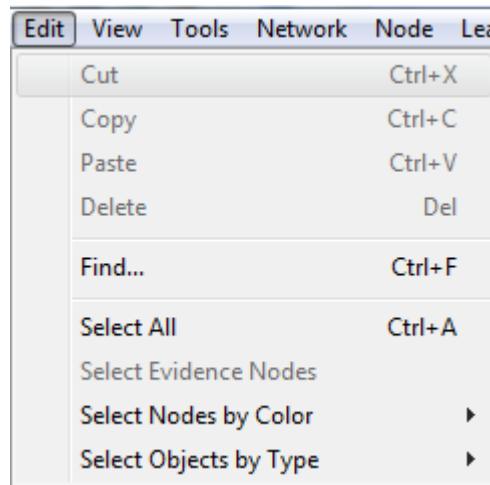
Left to Right: This places the nodes in the model from left to right on the graph view.

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.

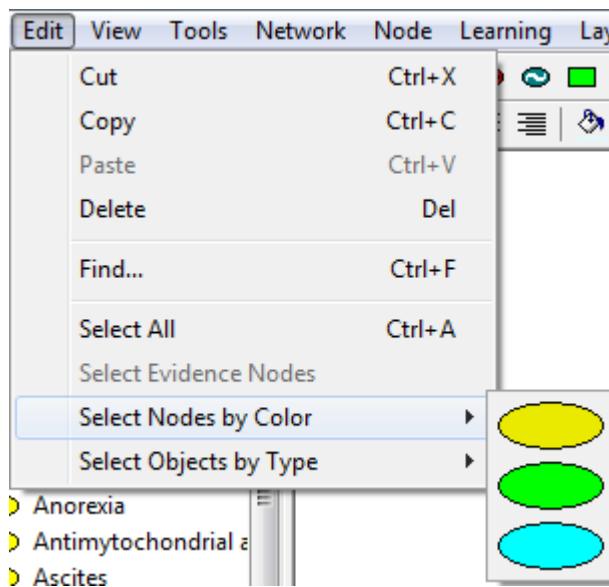
Edit Menu offers several other selection shortcuts that may make your life easier:



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.

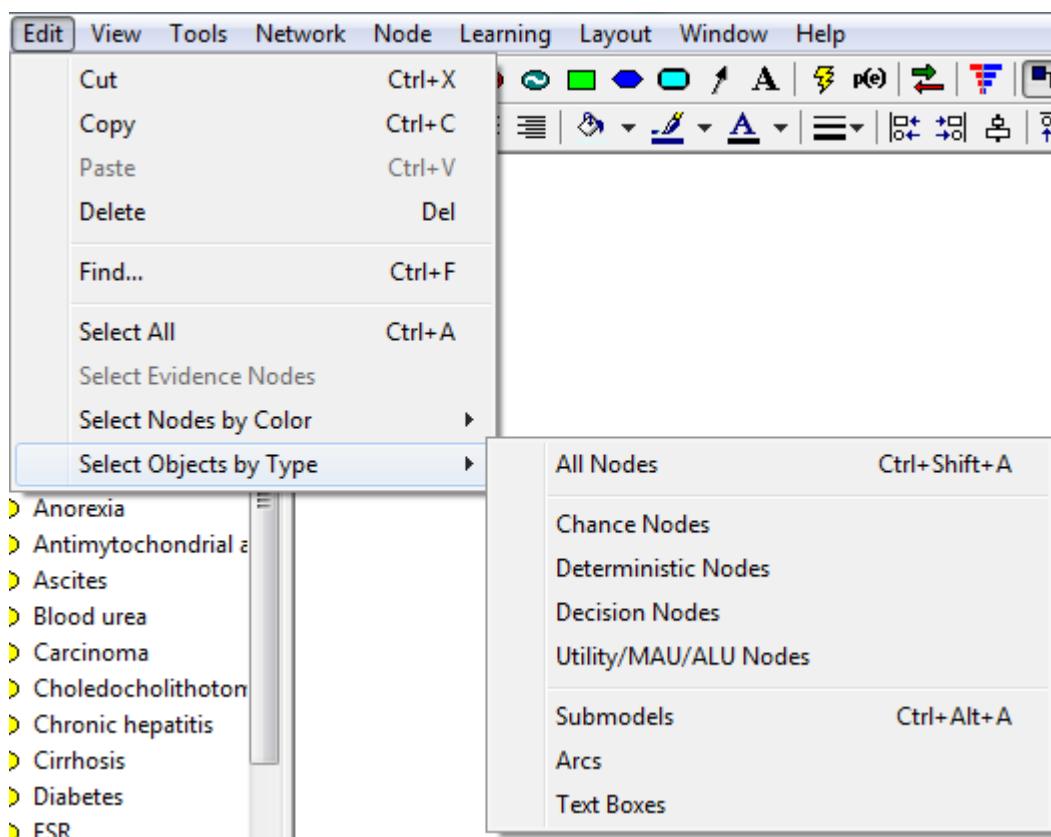
Select Evidence Nodes selects those nodes, for which the evidence has been set by the user. The function is dimmed out if no evidence is present in the model.

Select Nodes 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.

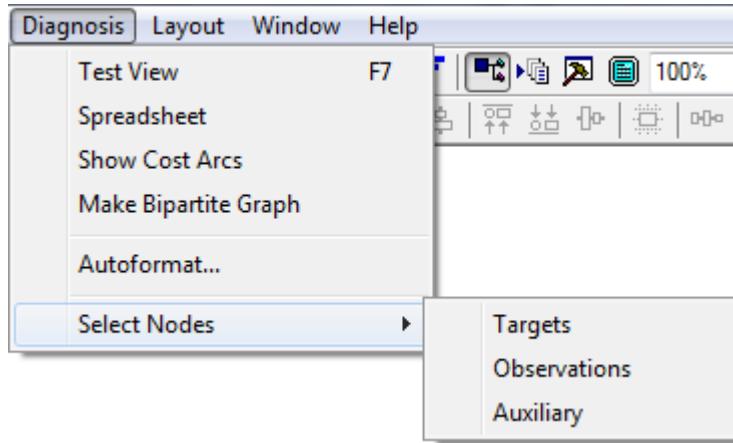
Chance Nodes, *Deterministic Nodes*, *Decision Nodes*, *Utility/MAU/ALU Nodes*, selects all nodes of the corresponding type in the current *Graph View*.

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, *Targets*, *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.

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](#)¹⁰⁵ 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](#)⁶⁰, which is the program's primary view. Some operations, however, may be more convenient in the [Tree View](#)⁷³, 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](#)¹¹⁹ 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](#)¹²⁰, 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](#)⁶⁰ 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](#)²⁷⁰ 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](#)⁷⁶ 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](#)⁷⁶ section of GeNIe workspace.

The *Output* command displays and hides the [Output Window](#)⁸². For more information on the Output window, 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 has been developed and made available on the World Wide Web, there has not been 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](#)⁴⁵ 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, KI, and Equation. For backward compatibility reason, we are also supporting the format used by GeNIe in mid-1990s (DSL file format). Should you be a software developer and want to add your file format to the formats supported by GeNIe, please [contact us](#).

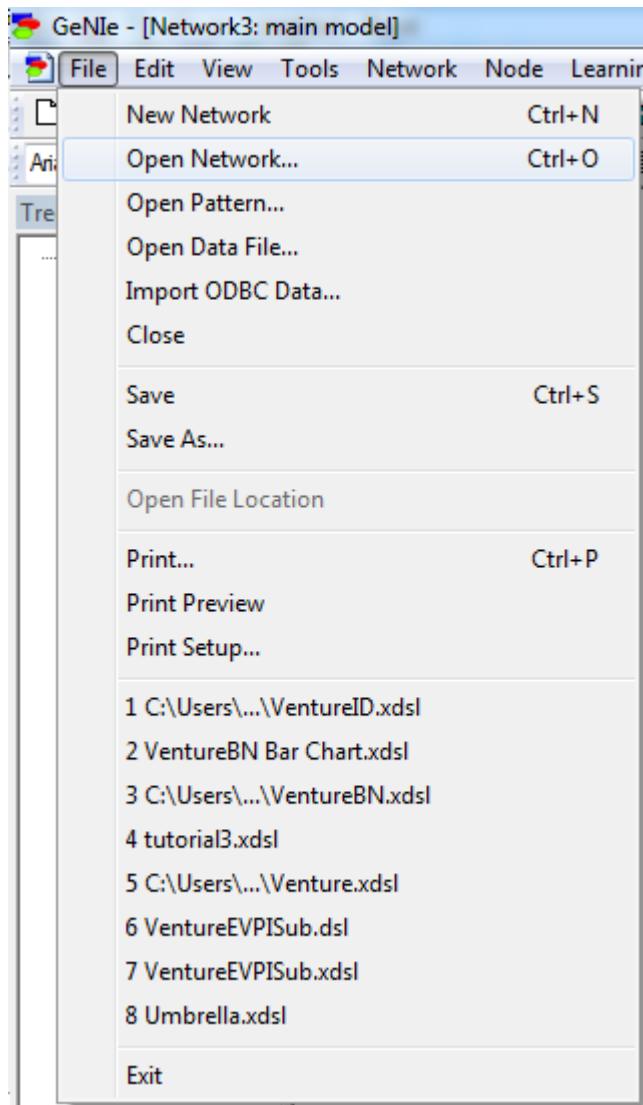
GeNIe can read and write models created by other programs. It can be thus used also as a conversion program between different formats. The user should keep in mind that various programs may have extensions and functionalities that are not supported by other programs. We would like to warn the users that 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 model in GeNIe

There are three ways in which you can open a model in GeNIe,

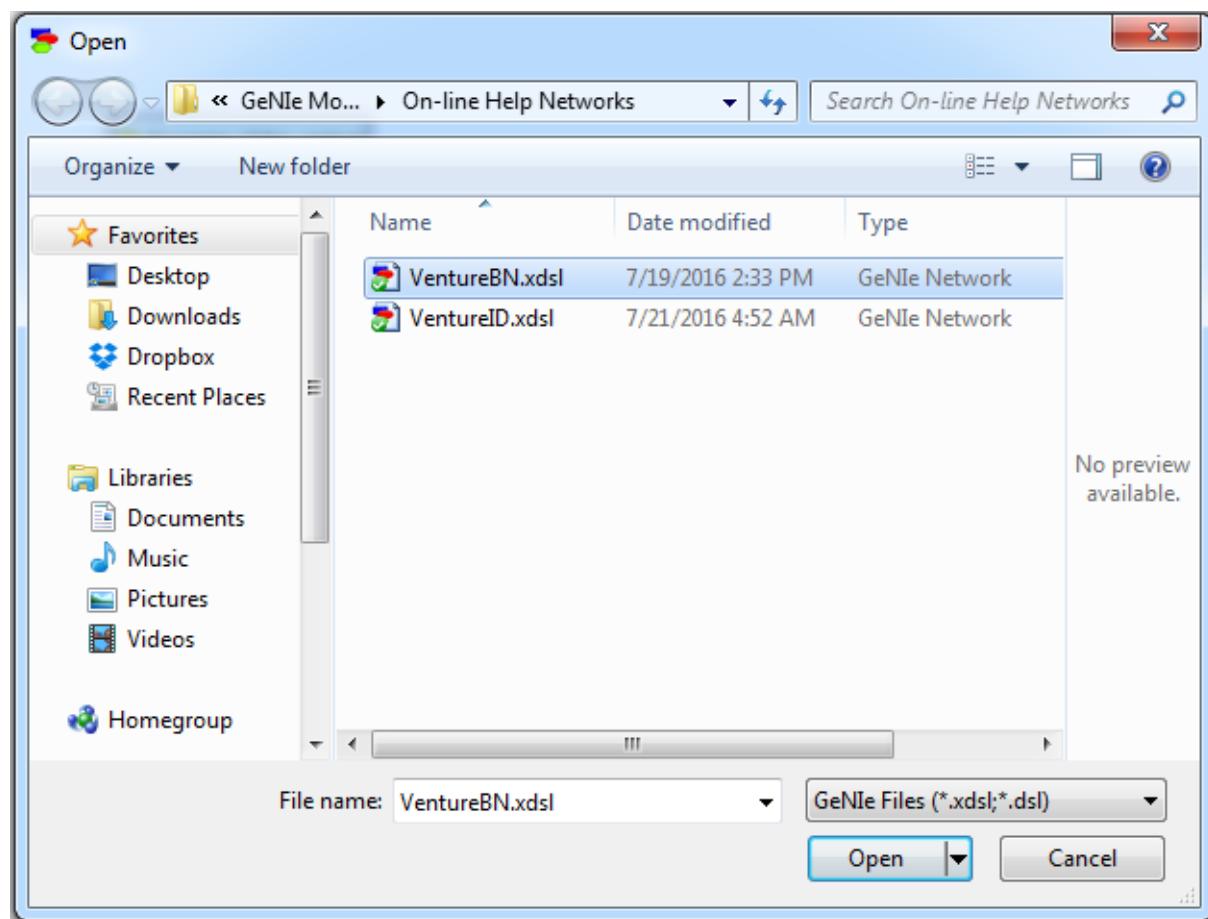
- Choose *Open Network...* from the *File* menu
- Click on *Open* () button from the *Standard Toolbar*
- Use the *CTRL+O* shortcut

Shown below is the *Open Network...* option in [File Menu](#)¹⁹³.



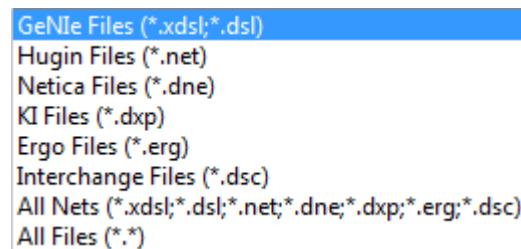
The numbered list of files at the end of the [File Menu](#)¹⁹³ are the *Most Recently Used (MRU)* file list. You can click on any of those names to re-open the file.

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 the following 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.

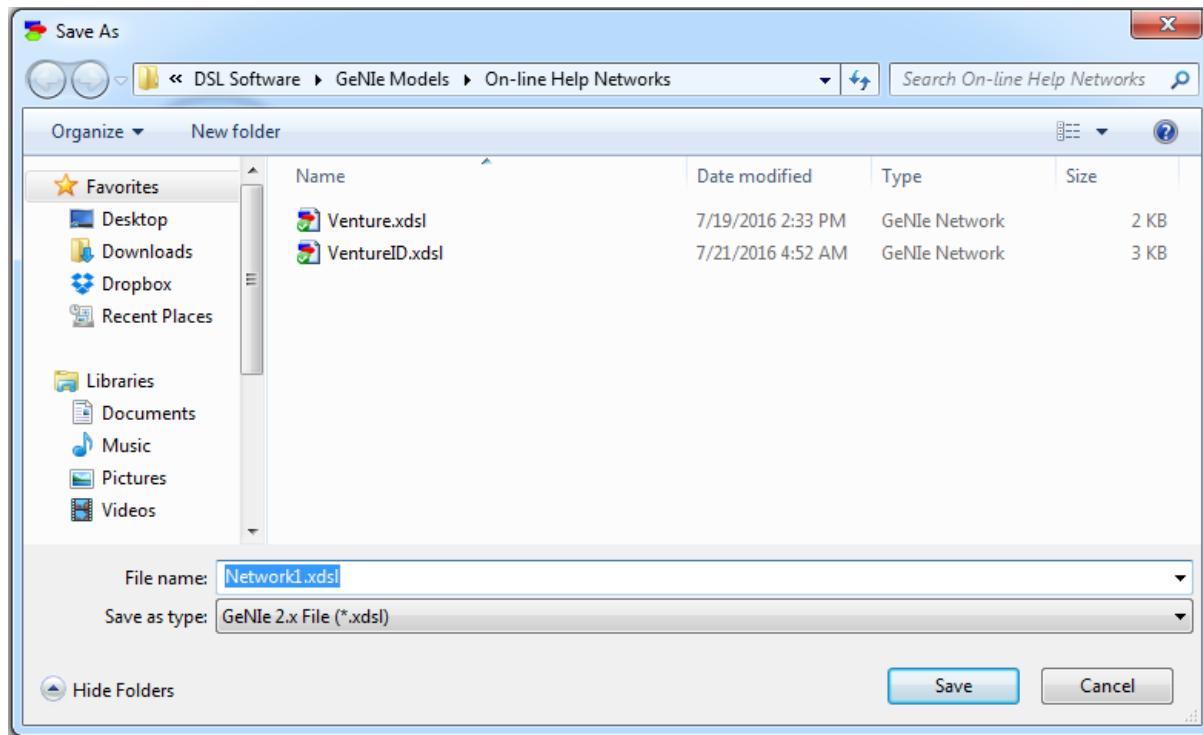
Note : You can have multiple files open at the same time.

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](#) ¹⁹³
- Click on Save (H) button from the [Standard Toolbar](#) ¹⁷⁶
- Use the *CTRL+S (Save)* shortcut

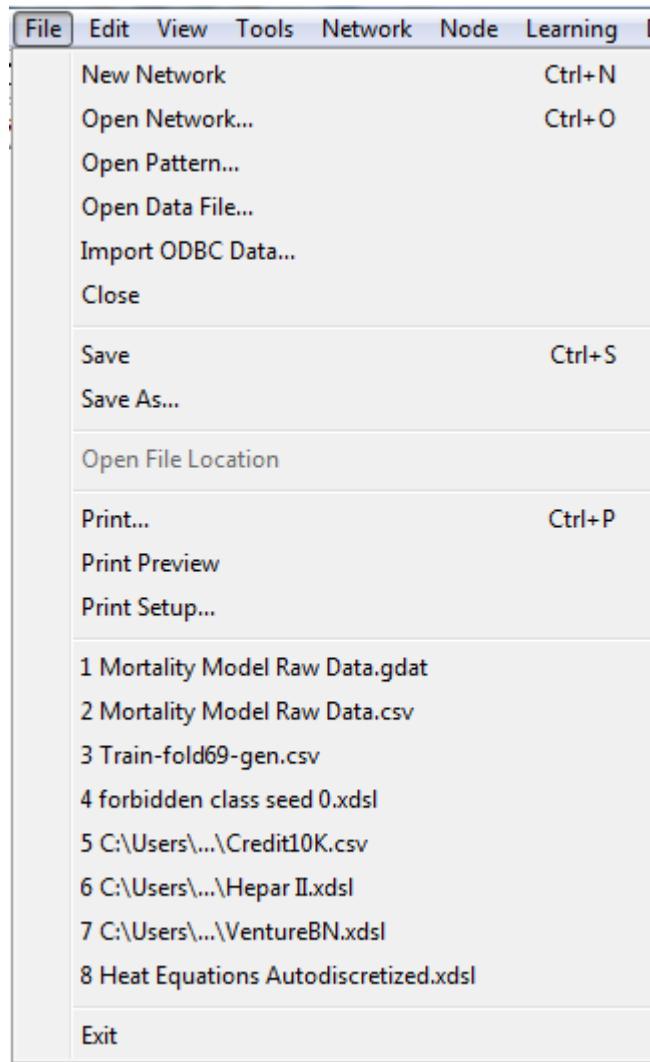
The difference between *Save* and *Save As* is that *Save As* lets you store the file under a different name, so that you can keep the original model file intact. *Save* will store the changes in the original file. However, if you are working on a new file, then *Save As* is the only option and *Save* 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](#)¹⁷⁶ 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](#)¹⁷⁶ or using the *CTRL+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](#)¹⁷⁶ 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.

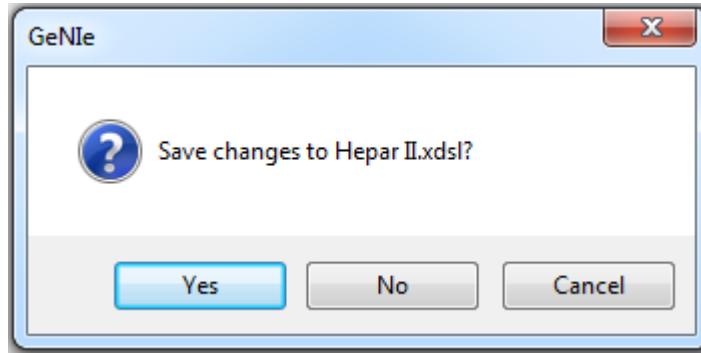
The *Open Network* and *Save/Save As* commands are discussed in detail in the [introduction](#)¹⁸⁸ to this section.

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 the [Accessing data](#)³³⁹ section.

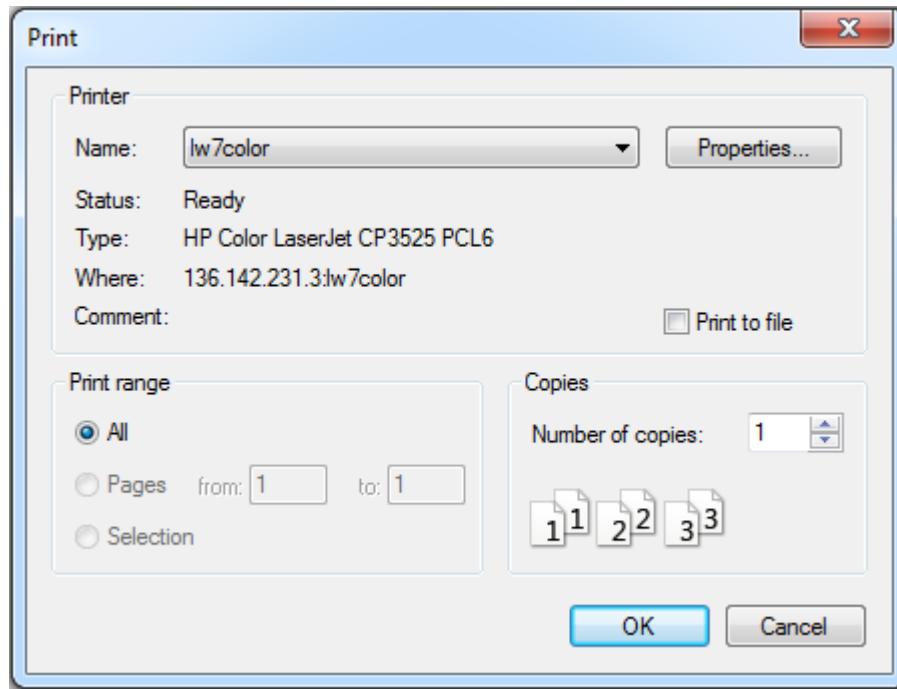
Close closes all windows of the current model. If there are any changes to the currently opened model that have not been saved, GeNIE will warn you using the following dialog box



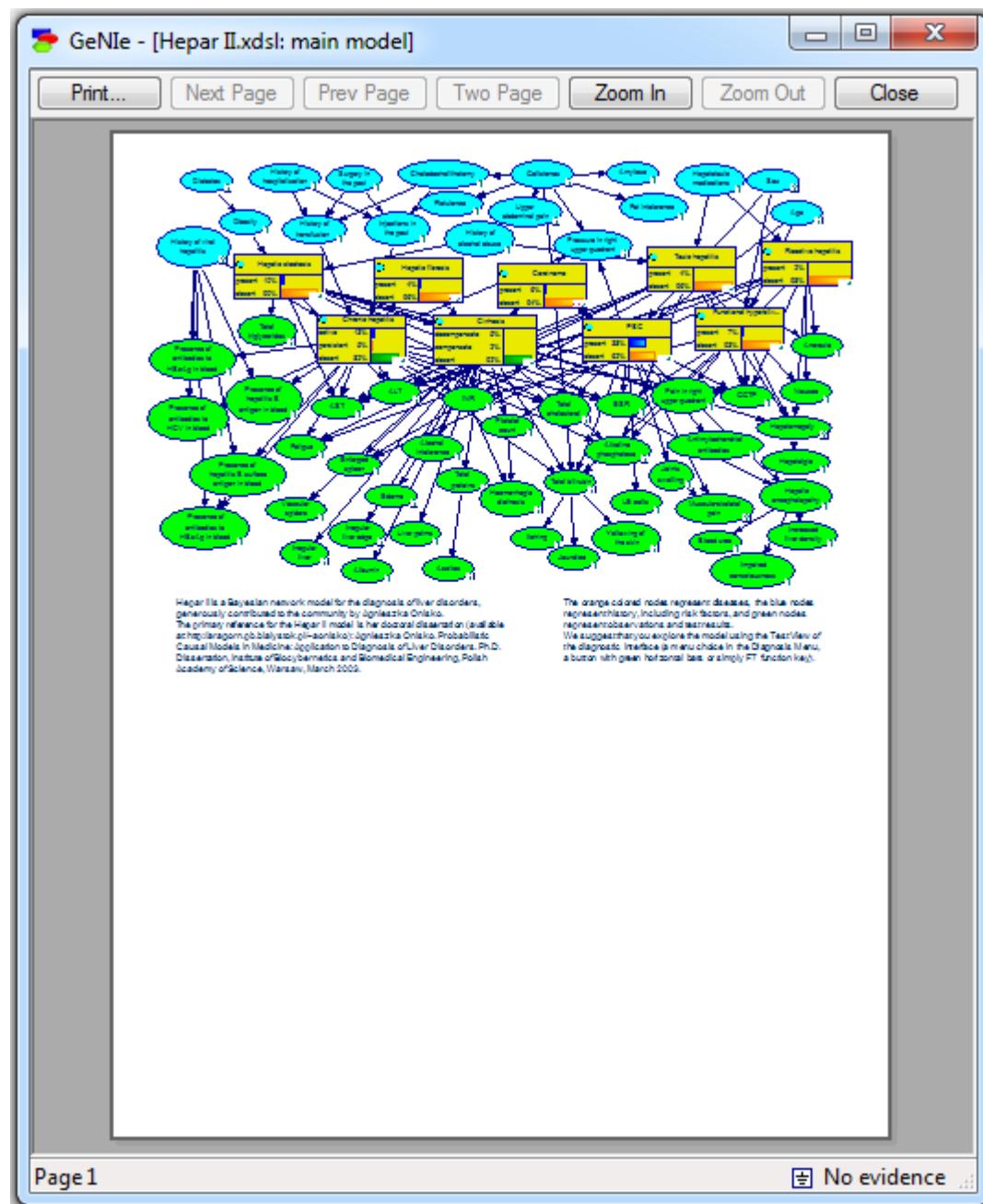
If you want to save the changes that you have made since you have last saved the model, click the *Yes* button or press *Enter*. If you want to discard them, click the *No* button. If you have second thoughts about exiting GeNIE, click *Cancel*.

Print... prints the current *Graph View* window. This command can be also invoked by pressing the *Print* (🖨) tool from the [Standard Toolbar](#)¹⁷⁶ or using the *CTRL+P* shortcut.

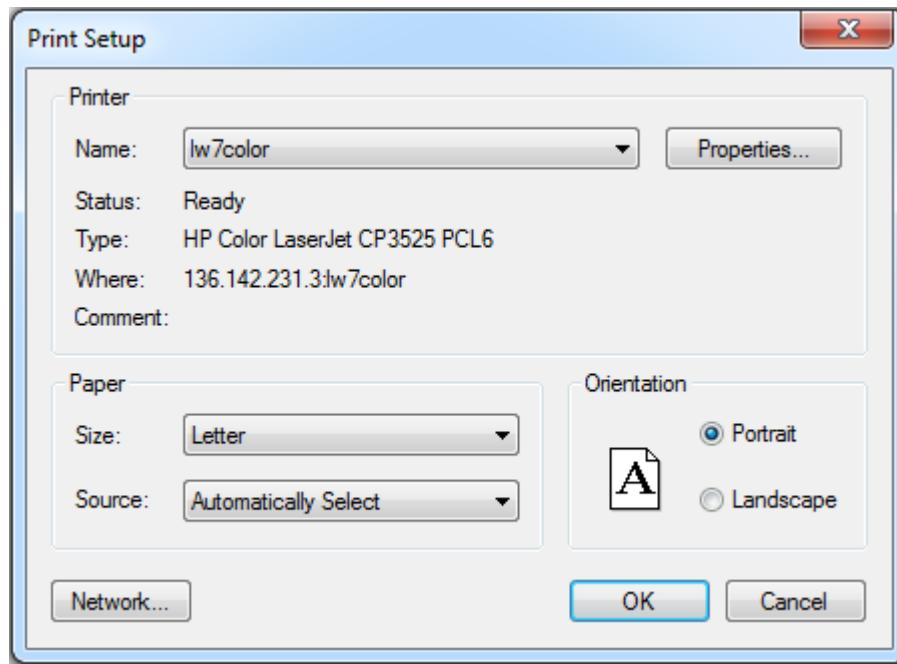
The following dialog box allows you to modify some of the printing options, such as choose the printer, the range of pages to be printed, the number of copies to be printed, and other printer properties (through *Properties...* button).



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](#)^[241]. 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: http://support.bayesfusion.com/docs/xdsl_schema.zip.

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 (_). 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 end 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](#)⁴⁵ 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](#)⁴⁵ 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          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 yet been established as a standard and our implementation is a good faith implementation of what has been agreed upon so far.⁴⁵ Our implementation allows for reading and writing [Bayesian networks](#)⁴⁵ 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

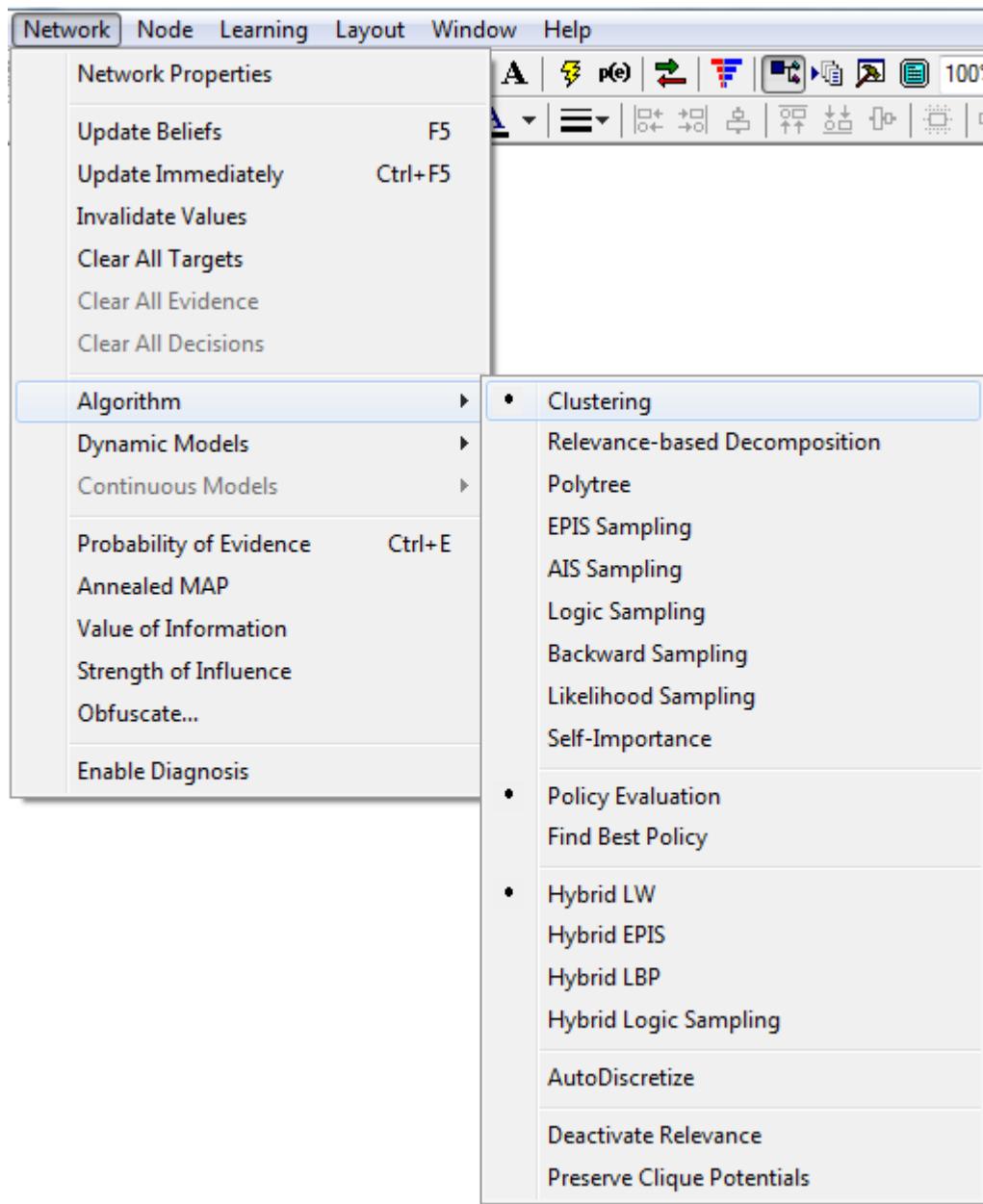
5.7.1 Introduction

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](#)²⁰⁹. [SMILE](#)³¹ and GeNIE implement several popular [Bayesian networks](#)⁴⁵ inference algorithms, including the clustering algorithm and several stochastic sampling algorithms. There are also two [influence diagrams](#)⁴⁷ 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 the fastest exact algorithm available, 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). A 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.

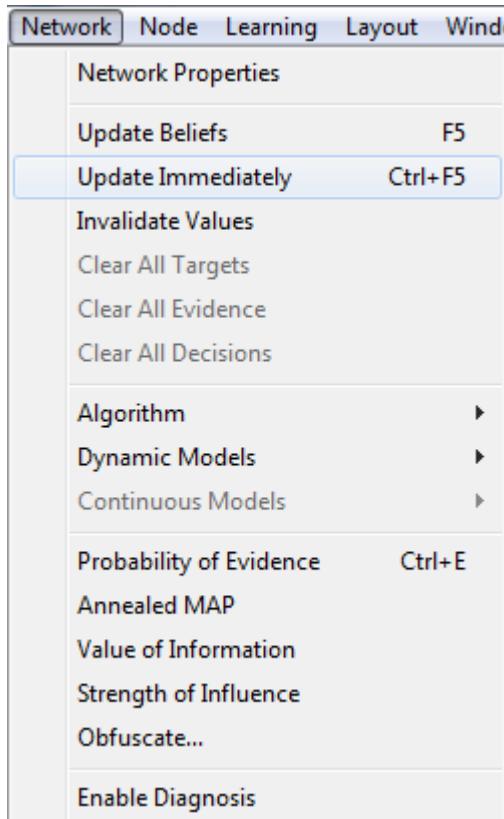
We would like to stress that the choice of algorithms has one important practical application: When a networks cannot be updated using the fastest known exact algorithm, the *Clustering* algorithm (GeNle's default), the user can still update the

model using an approximate algorithm. Tradeoff between precision and computation time can be controlled by selecting the number of samples.

5.7.2 Immediate and lazy evaluation

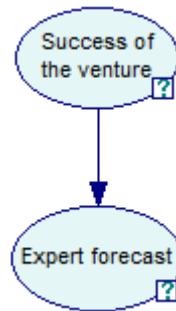
GeNIE operates in two modes: immediate and lazy updating. In lazy updating mode, every time we modify the model, enter evidence, or control the value of a node, we need to explicitly invoke an algorithm to update the values and to view the result of the changes. In immediate updating mode, GeNIE automatically 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.



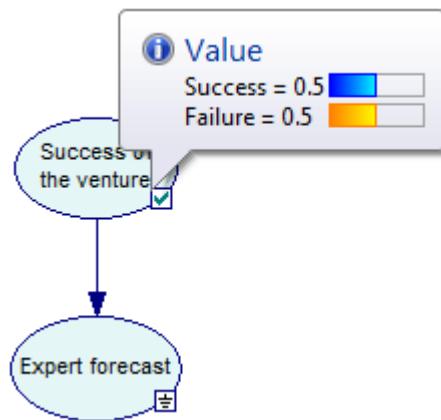
The lazy updating mode, is useful when the model is in its development stage or when it is so large that updating takes an annoyingly long time. In this case, you can run an evaluation algorithm to update the model either by choosing *Update* from the *Network* menu or, alternatively, by pressing the *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](#)¹²³ 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 hoovering over them (like in the picture above) or by opening their *Value* tab.

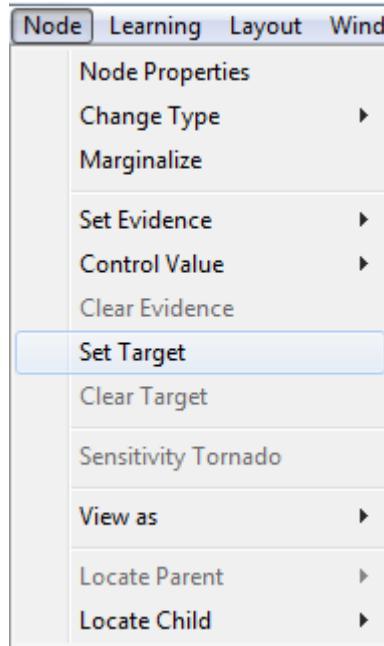
5.7.3 Relevance reasoning

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

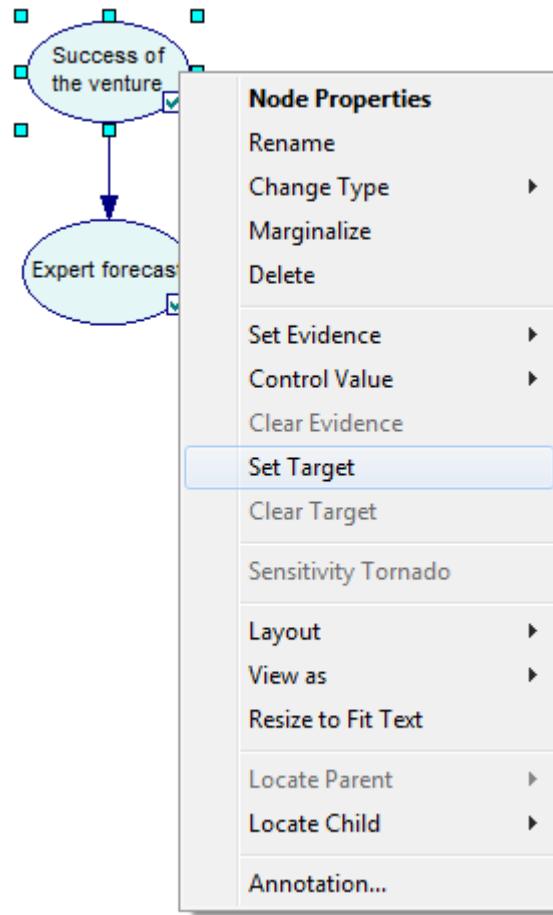
variables may be prohibitive, while potentially unnecessary. Focusing inference on those nodes that we are interested in, can save a lot of computation. Reasoning in GeNIE and the underlying [SMILE](#)³¹ is always preceded by a pre-processing step that explores structural and numerical properties of 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](#)¹¹⁶. 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](#)¹¹⁶) 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.

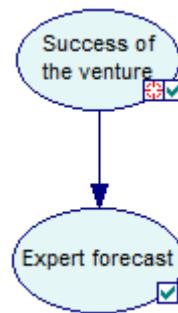
To set a node to be a target, select it and then choose *Set Target* from the [Node Menu](#)²⁰⁷.



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 (☒).

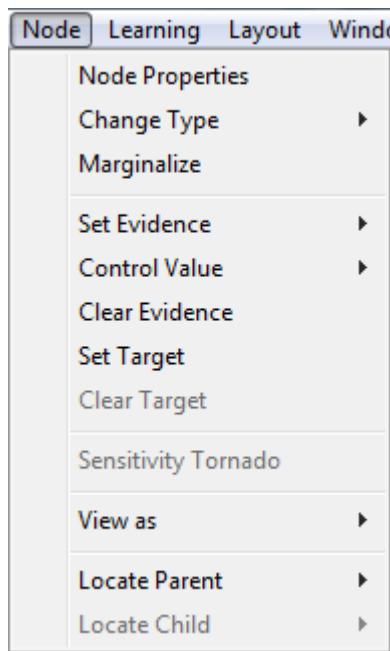


Relevance-based algorithms implemented in GeNIE are described in (Druzdzel & Suermundt 1994) and (Lin & Druzdzel 1997, 1998). Relevance algorithms usually lead to substantial savings in computation. We call this pre-processing step collectively *relevance reasoning*. Relevance reasoning is transparent to the user and the application programmer.

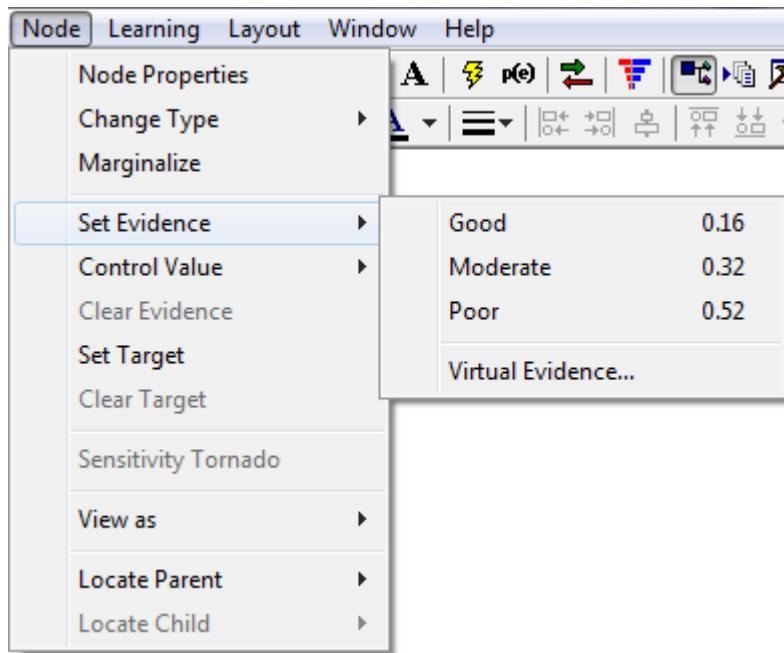
GeNIE has been originally written with teaching graphical models in mind. One of the fundamental concepts on which graphical models are built is conditional independence and relevance (Dawid 1979). In order to demonstrate how nodes are relevant to target variables and to evidence variables, GeNIE allows 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 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. GeNle'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). GeNle shows these inactive arcs as inactive by dimming them. See [Controlling values](#)²⁷³ for more information about this functionality.

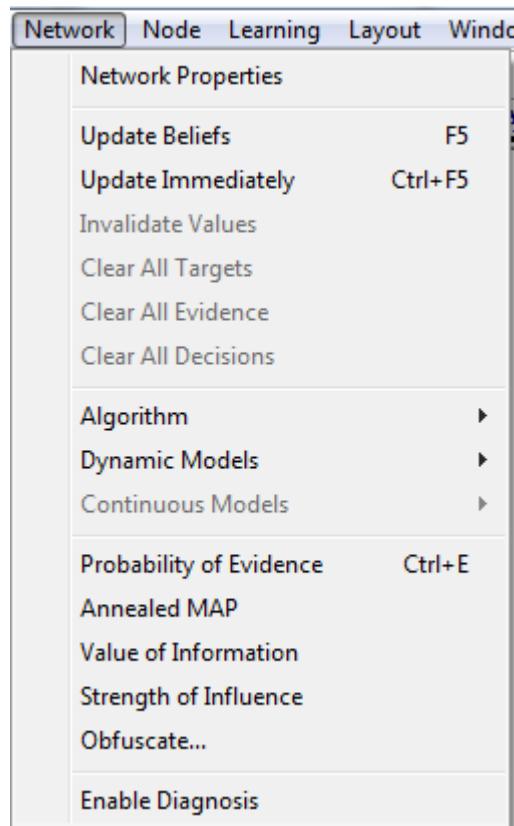
Clear Evidence command (for controlled nodes, this command is called *Release Value*) is active only if there are nodes selected in the *Graph View* and at least one of these node has been previously observed (or controlled). *Clear Evidence* un-observes a state, i.e., it reverses the effect of the *Set 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](#)¹²³ 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](#)¹⁷⁶.

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. 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 will 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](#)²⁰⁰ to the current section.

Dynamic Models submenu groups all operations on dynamic Bayesian networks. This submenu is discussed in detail in the section of *Dynamic Bayesian networks*.

Continuous Models submenu groups all operations on continuous (equation-based) models. This submenu is discussed in detail in the section of *Equation-based models*.

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](#)⁴⁷, discussed in the *Influence diagrams* section.

Strength of Influence is a model exploration technique discussed in [Strength of influences](#)²⁷⁰ section.

Obfuscate... is a special algorithm for obfuscating the network for the purpose of protecting intellectual property, discussed in [Obfuscation](#)²³⁶ 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](#)^[45]. 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, something that we have found confusing for some of our users.

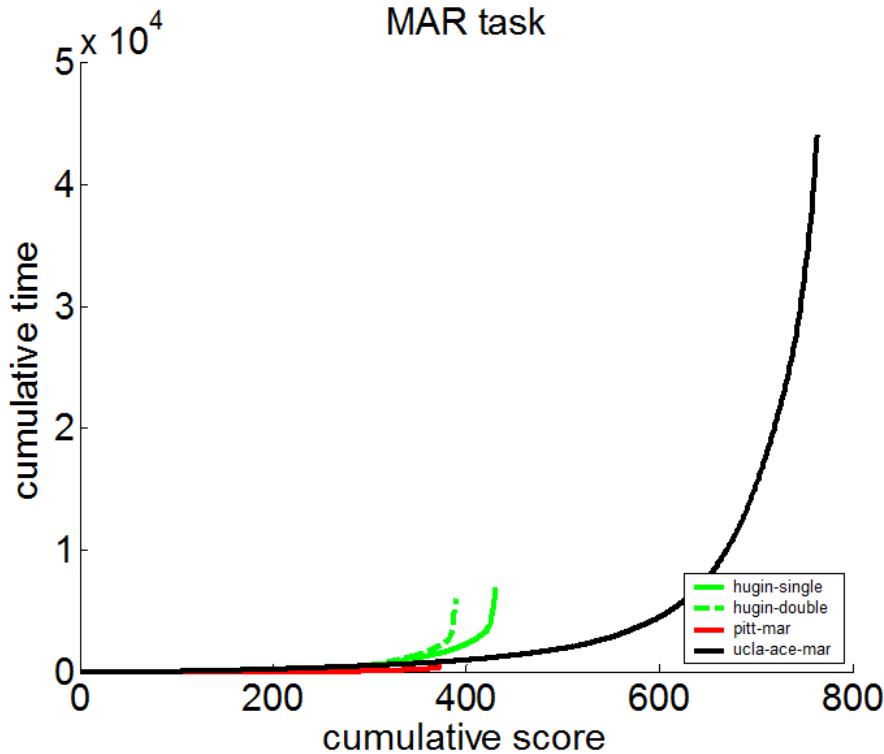
The clustering algorithm, like all of the algorithms for Bayesian networks, produces marginal probability distributions over all network nodes. In addition, it is possible to preserve clique potentials in the network, which allows for viewing joint probability distribution over those variables that are 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* ( **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, it is suggested 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).

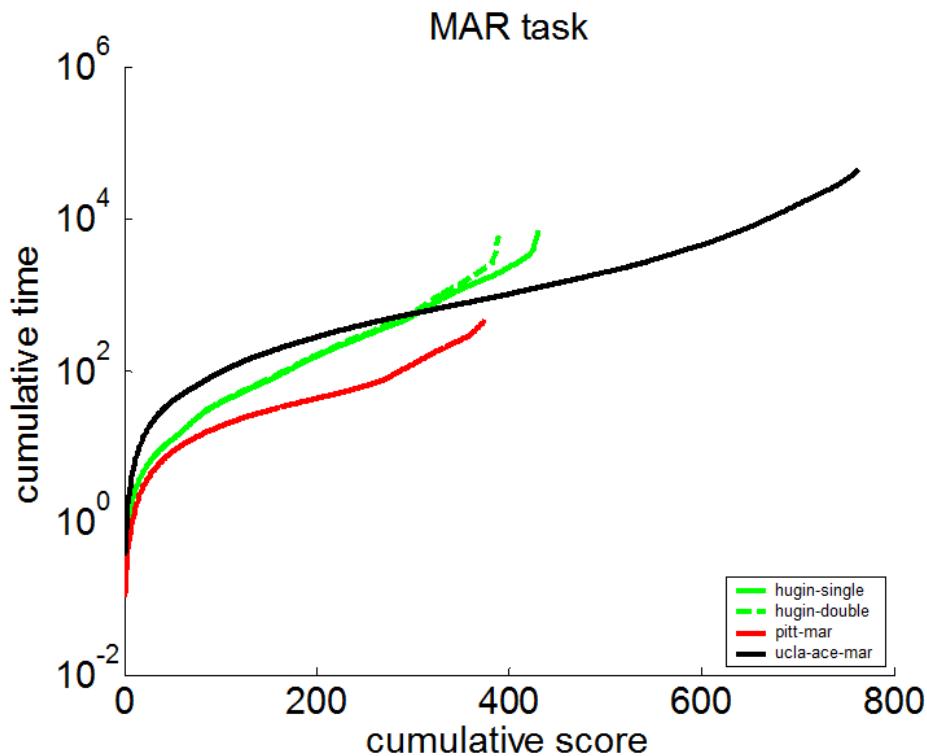
SMILE's clustering algorithm in UAI-2006 and UAI-2008 inference evaluation

SMILE's implementation of the clustering algorithm underlies the implementation of calculation of the marginal probability and the [probability of evidence](#)^[216]. 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](#)^[216] 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 <http://graphmod.ics.uci.edu/uai08/Evaluation/Report> for the full report).



The same curve in logarithmic scale



See also the evaluation results for computing the probability of evidence and approximate marginals in the [Probability of evidence](#)^[216] and [EPIS Sampling](#)^[215] 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](#)⁴⁵. 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](#)²¹⁴ 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 Self-Importance Sampling

The *Self-Importance Sampling* algorithm is described in (Shachter & Peot 1990). It controls the generation of samples to account for the bias due to disproportional sampling of cases that represent the most probable hypotheses. This algorithm may have been the first algorithm using the concept of importance sampling in Bayesian network algorithms

5.7.6.2.4 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 algorithm is an ingenious extension to forward sampling algorithms.

5.7.6.2.5 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.

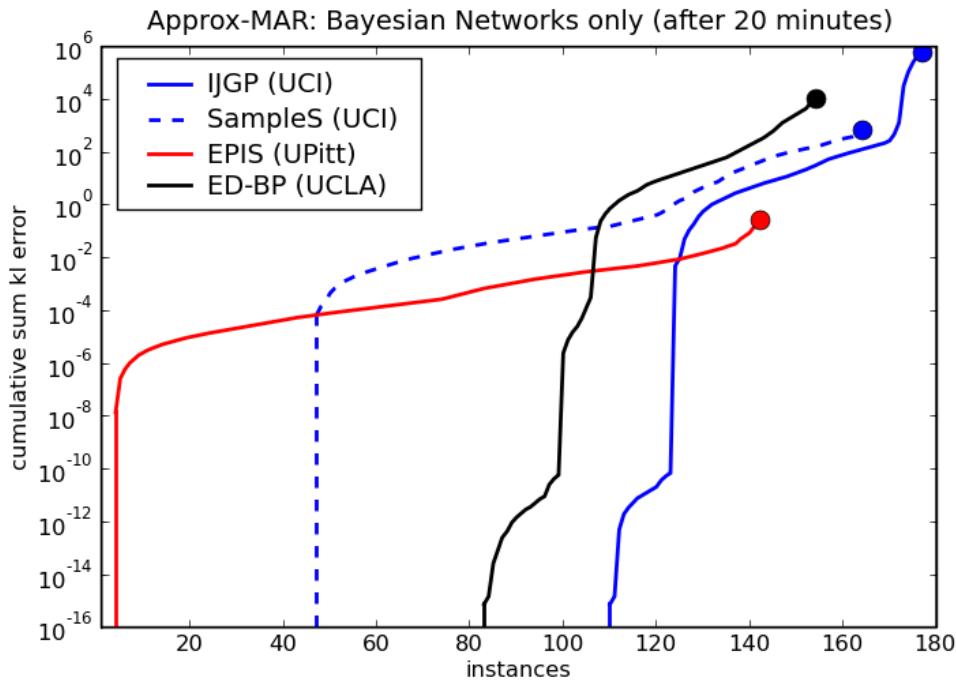
The AIS algorithm was judged to be one of the most influential developments in the area of Artificial Intelligence in 2005, receiving Honorable Mention in the 2005 IJCAI-JAIR Best Paper Prize. The IJCAI-JAIR (*International Joint Conference on Artificial Intelligence and Journal of Artificial Intelligence 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 our lab, the EPIS algorithm (Yuan & Druzdzel 2003), which sometimes offered an order of magnitude improvement over the AIS algorithm.

5.7.6.2.6 EPIS Sampling

The *Estimated Posterior Importance Sampling (EPIS)* algorithm is described in (Yuan & Druzdzel 2003). This is quite likely the best sampling algorithm 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 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.



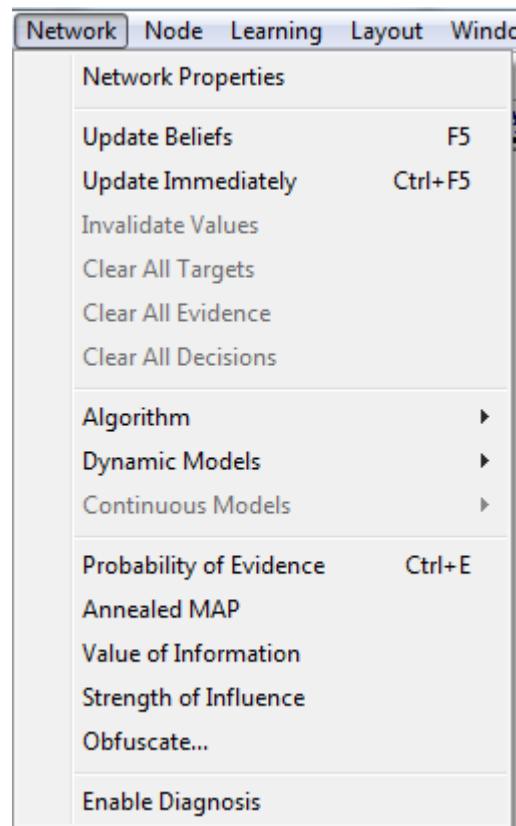
We misunderstood the rules of the competition at the time of submission and did not use exact inference whenever it was possible (for the easiest instances). EPIS Sampling algorithm ran on even the simplest networks and against algorithms that used exact inference on those networks that were computable. 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 error than the other algorithms. Please note that the increase in error between the simplest and more complex instances is not large. See <http://graphmod.ics.uci.edu/uai08/Evaluation/Report> for the full report.

See also the evaluation results for computing the probability of evidence and marginals in the [Probability of evidence](#)²¹⁶ and [Clustering algorithm](#)²¹¹ sections respectively.

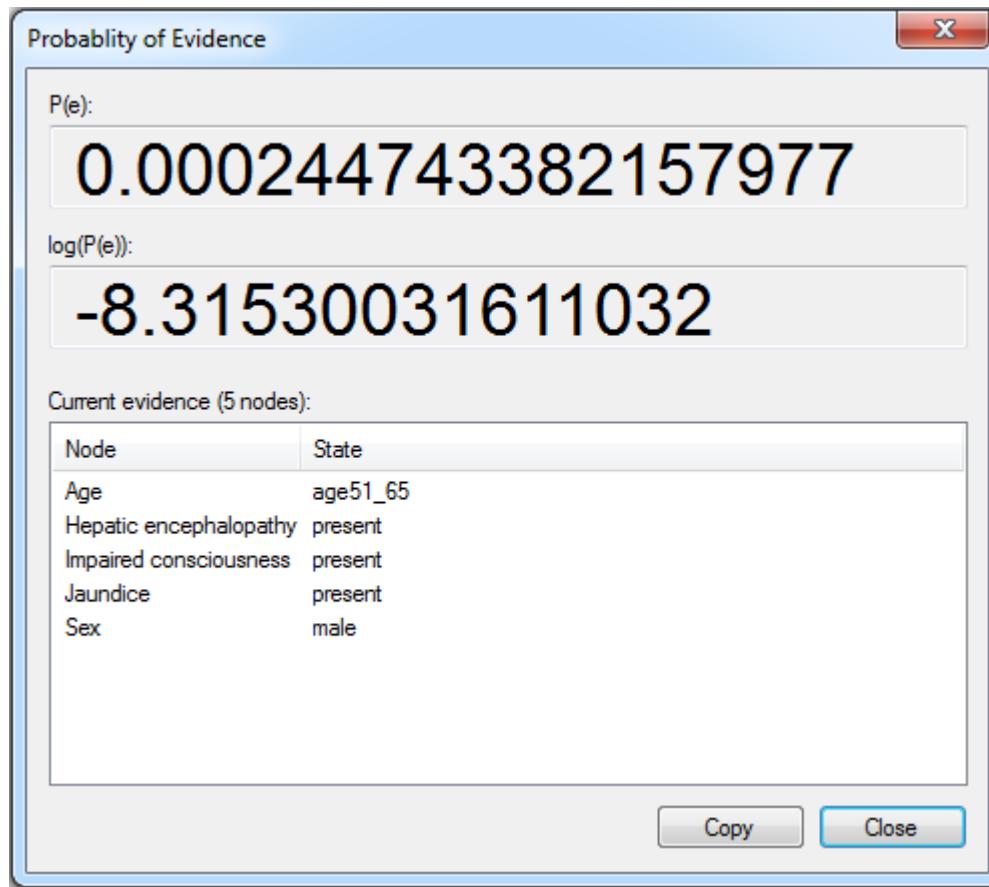
5.7.6.3 Special algorithms

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.



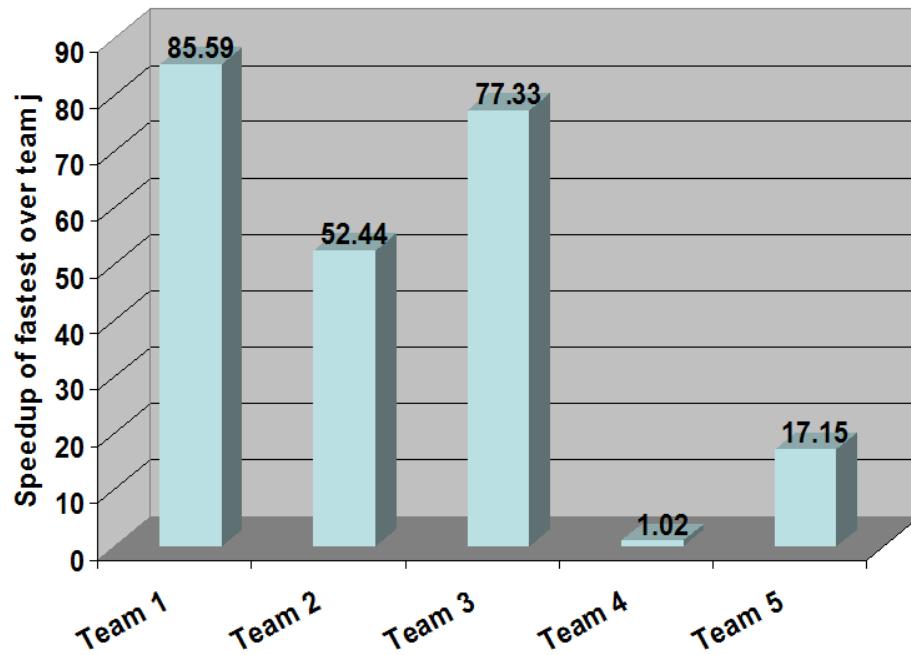
The following dialog displays the results of this calculation:



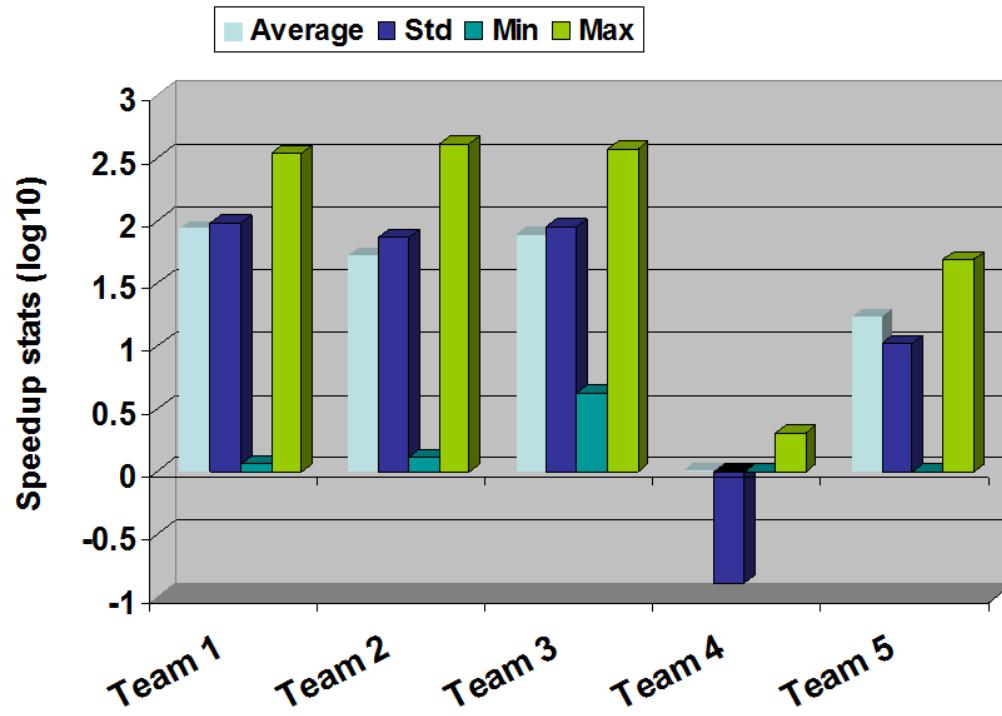
We can see that according to the model at hand, the *a-priori* probability of observing a *male* patient between the ages of *51* and *65* with *Hepatic encephalopathy*, *Impaired consciousness* and *Jaundice* is, according to the Hepar II model, roughly **0.000245**.

SMILE's probability of evidence algorithm in UAI-2006 and UAI-2008 inference evaluation

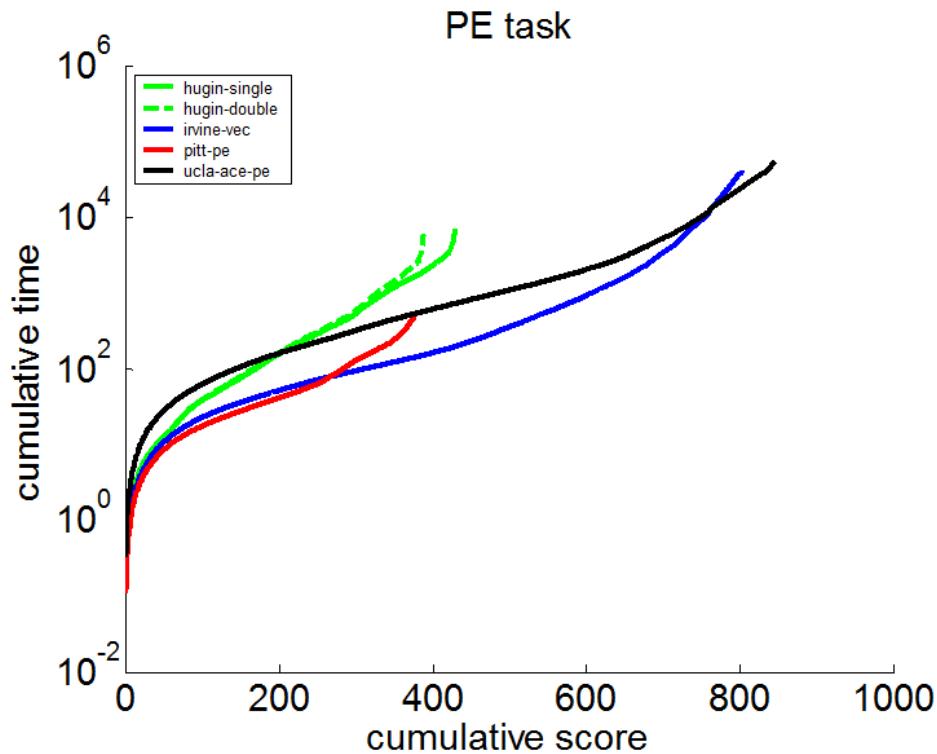
SMILE's probability of evidence algorithm participated in the UAI-2006 inference evaluation competition (see <http://melodi.ee.washington.edu/~bilmes/uai06InferenceEvaluation/>) and performed roughly two orders of magnitude faster than the algorithms belonging to the other four teams. The following plot 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 3) 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 3) was reliably the fastest with the average standard deviation of 0.01.



In the UAI-2008 inference evaluation competition (see <http://graphmod.ics.uci.edu/uai08/Evaluation/Report>), the algorithm did quite well again. SMILE typically computed very fast whatever was computable within a short time. The following plot shows the cumulative time taken over test instances.

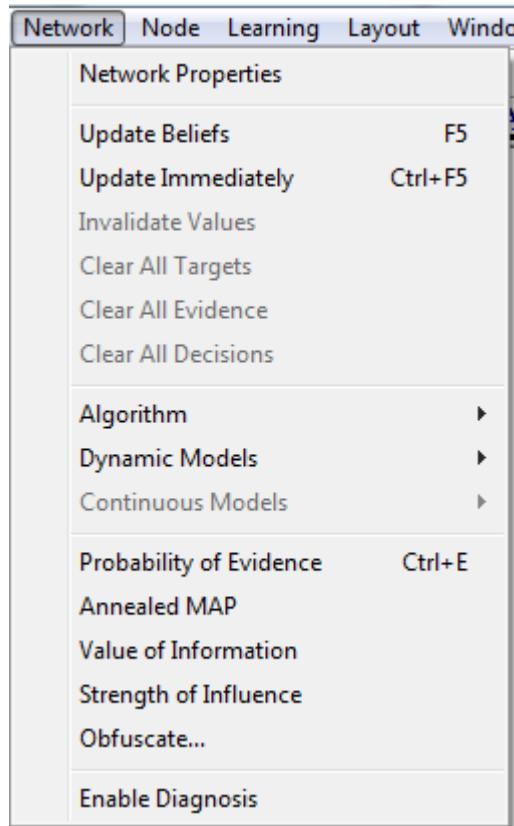


See also the results of computing the exact and approximate marginals in the [Clustering Algorithm](#)^[211] and [EPIS Sampling](#)^[215] sections respectively.

5.7.6.3.2 Annealed MAP

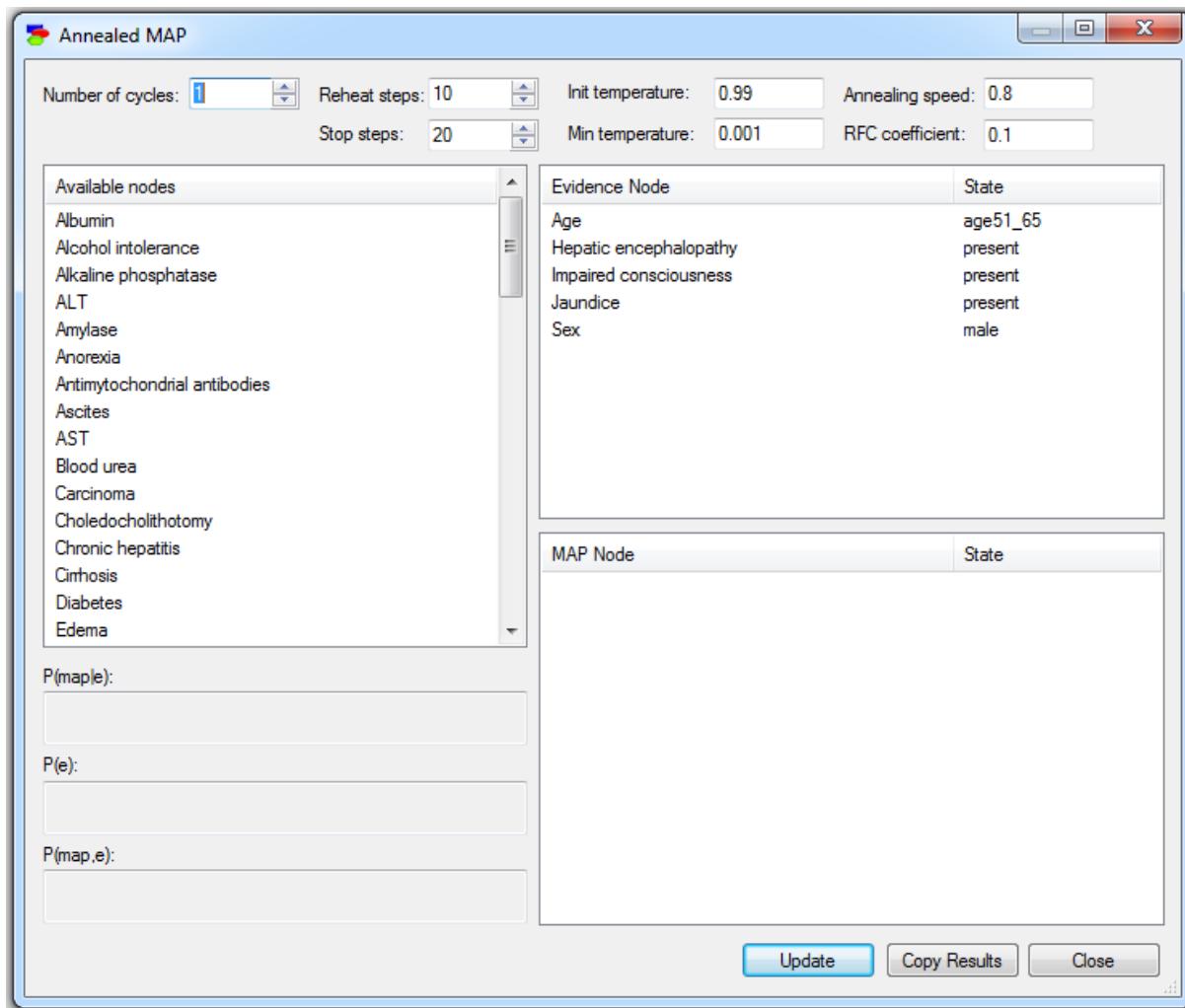
The *Annealed MAP* algorithm (Yuan et al., 2004) solves the problem of finding the most likely configuration of values of a set of nodes given observations of another subset of nodes. This problem is often called *Maximum A posteriori Probability (MAP)*. The *Annealed MAP* algorithm is approximate and 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.

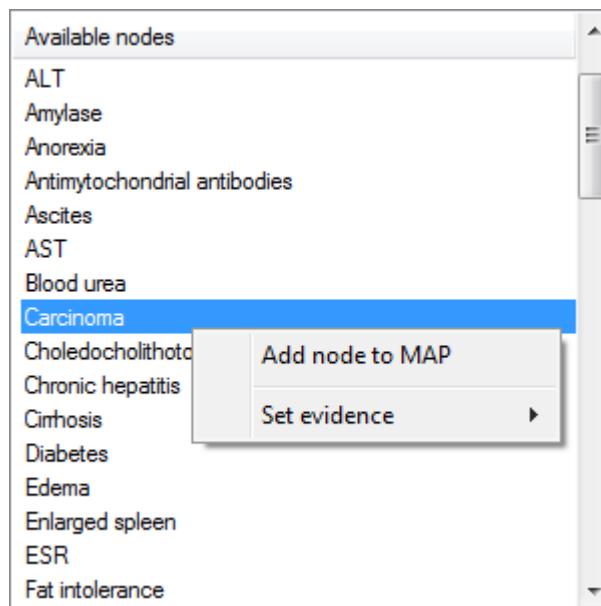


The upper-left window pane contains *Available nodes*, which are all nodes in the current network that have not been observed.

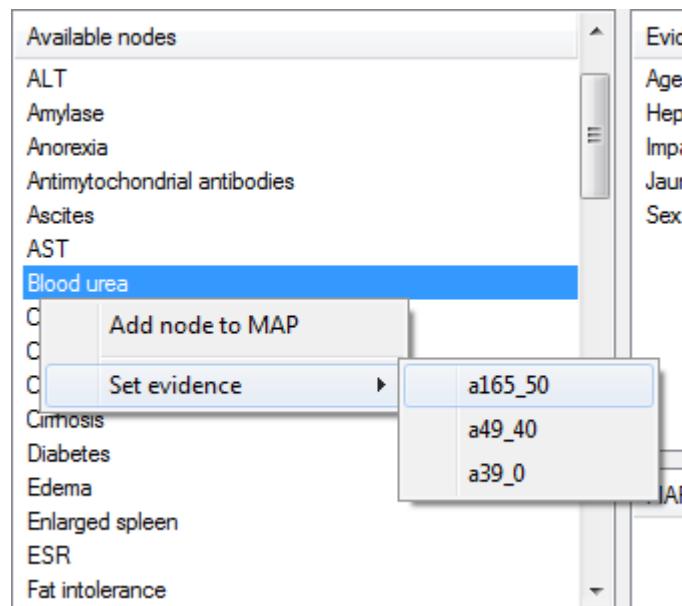
The upper-right window pane contains *Evidence Nodes* and their *States*. In this case, we have five observed evidence nodes.

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*.

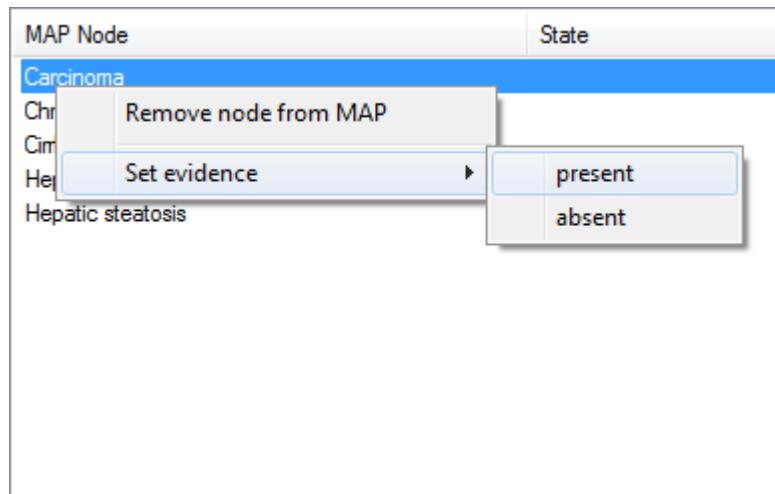
Nodes in the Available nodes set can be observed or added to the MAP set. To add a node to the MAP set, right-click on a node name and select *Add node to MAP* from the pop-up menu.



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.

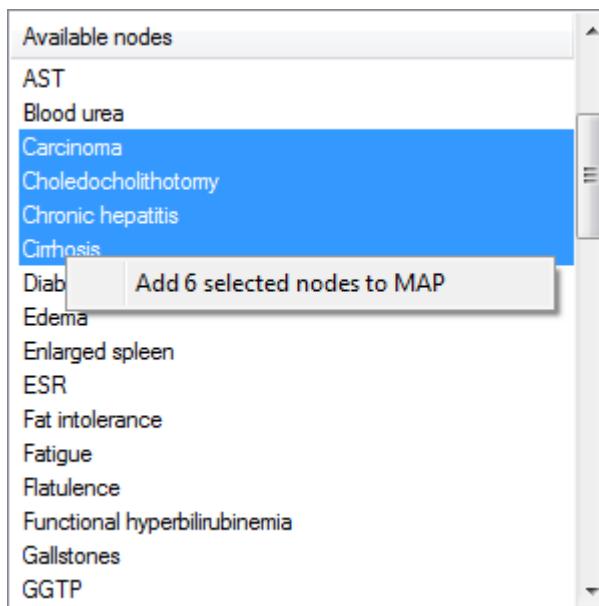


The same can be done to nodes in the Evidence Node set:

Evidence Node	State
Age	age51_65
Hepatic encephalopathy	present
Impaired consciousness	present
Jaundice	Clear evidence
Sex	present absent

A context menu is open over the node 'Impaired consciousness'. The options visible are 'Clear evidence', 'present', 'absent', and 'Add node to MAP'.

Nodes can be moved between windows in groups as well - just select multiple nodes before right-clicking:



Parameters

Annealed MAP algorithm has seven parameters:

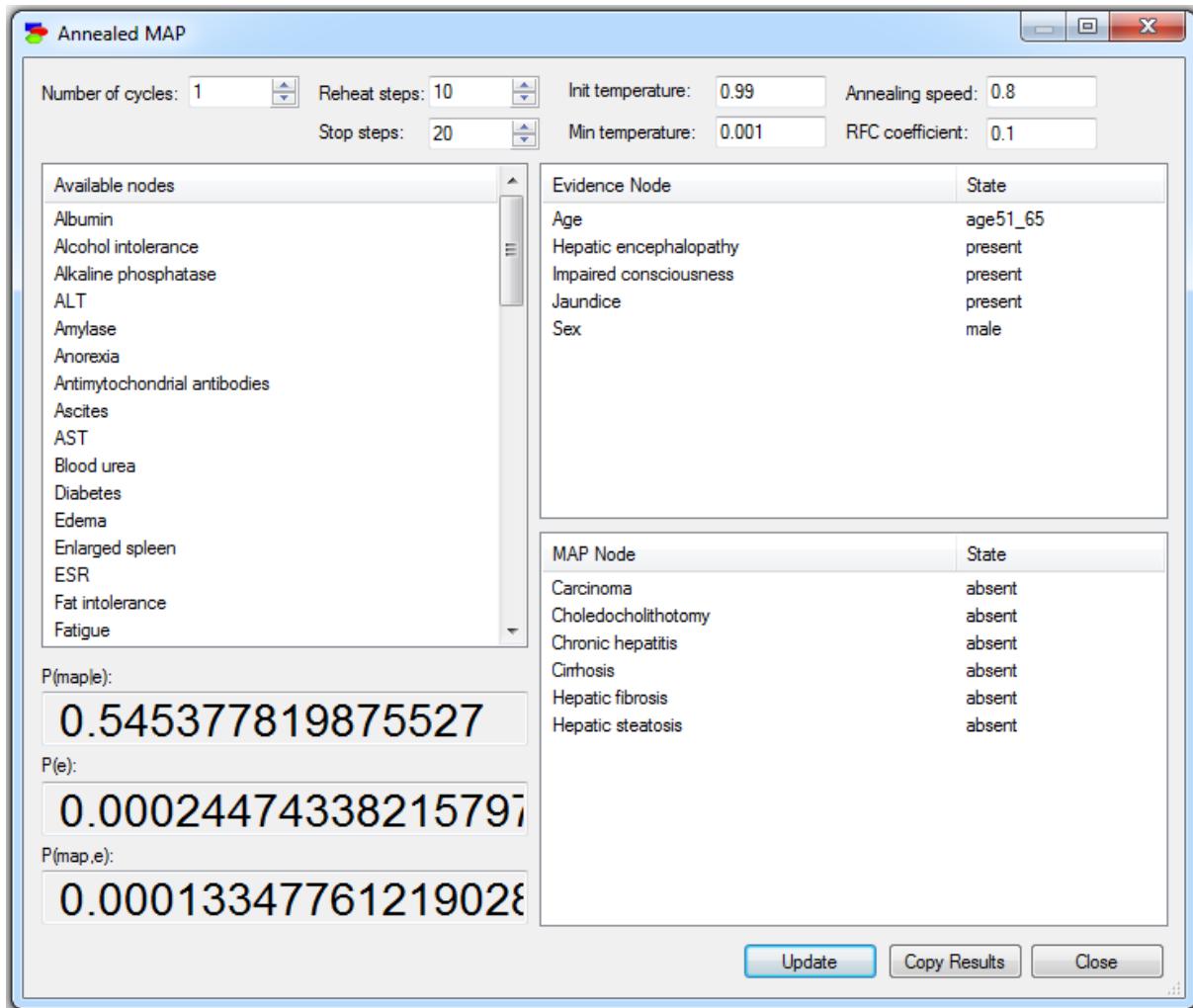
- Number of cycles (default 1)
- Reheat steps (default 10)
- Stop steps (default 20)
- Init temperature (default 0.99)
- Min temperature (default 0.001)
- Annealing speed (default 0.8), and
- RFC coefficient (default 0.1)

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:

- $P(\text{MAP} | \text{E})$: The probability of the MAP assignment given the set of evidence
- $P(\text{E})$: The probability of evidence
- $P(\text{MAP}, \text{E})$: 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 $P(\text{MAP}|E)=0.5454$.

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 . 00024474338215797
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](#)⁴⁷ in GeNIE. Its 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](#)⁴⁵ and then finding the expected [utilities](#)⁴⁴ 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](#)²²⁸. 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.

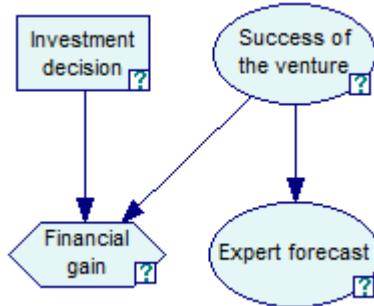
If you are more concerned about finding what is the optimal decision at the highest level rather than the actual utilities for each decision, algorithm for finding the best policy is a better choice. It is generally much faster than the policy evaluation algorithm.

5.7.7.2 Find Best Policy

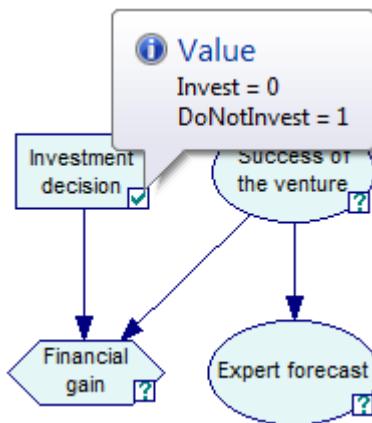
Sometimes the only thing that we might be interested in is the optimal decision at the top level of the influence diagram. Since the [policy evaluation](#)²²⁷ algorithm computes the expected utilities of all possible policies, it may be doing unnecessary work. For those cases, [SMILE](#)³¹ and GeNle 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](#)²⁸¹ section.



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 models

5.7.8.1 Introduction

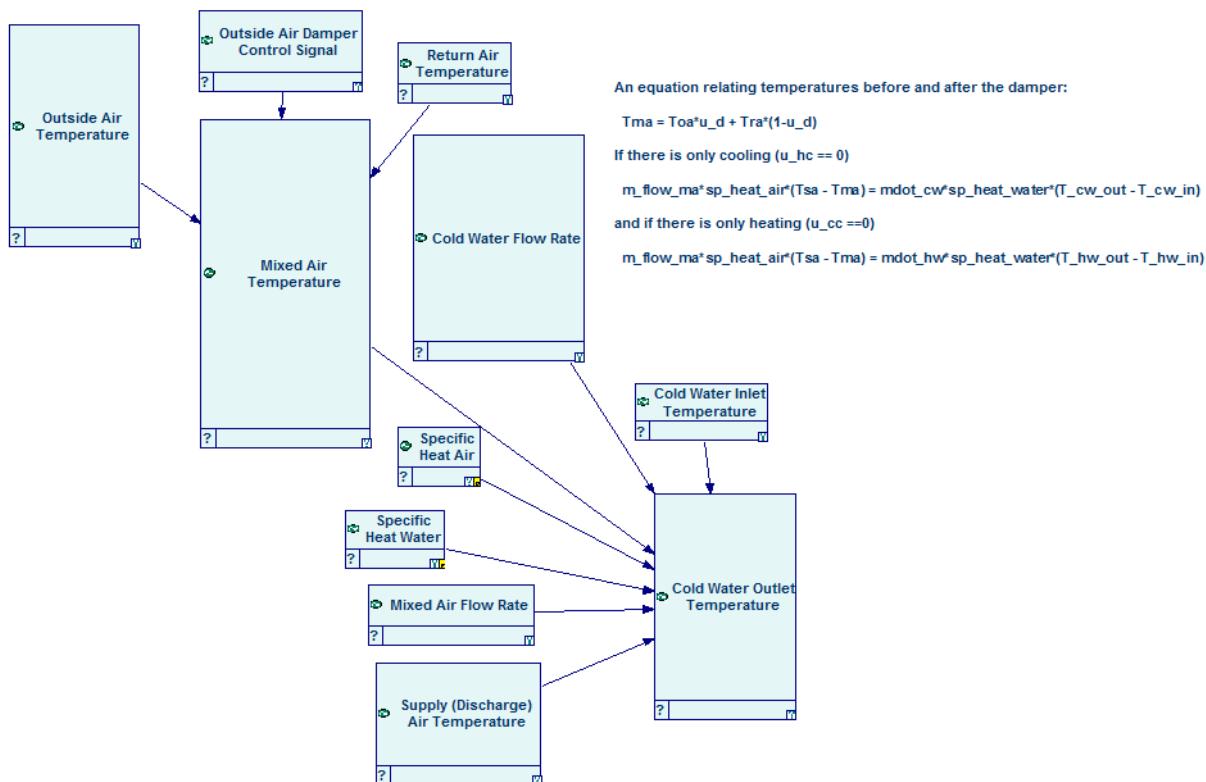
This section contains a collection of five algorithms for hybrid Bayesian networks, i.e., networks including both discrete and continuous variables. The first four of these are sampling algorithms, which we have to say, are experimental at best. The main problem with inference algorithms for hybrid Bayesian networks is that evidence in continuous variables is extremely unlikely, approaching zero probability. Hence, sampling algorithms have a hard time solving such networks. While we have included these algorithms in GeNle and [SMILE](#)³¹, we advice caution with respect to the reliability of the results.

The first algorithm, based on autodiscretization, is, we believe, the most promising direction for this work. The algorithm has an intermediate stage, in which it 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 as a discrete Bayesian network. The big advantage of this algorithm is that it converts the original hybrid Bayesian network into a discrete Bayesian network only for the purpose of inference, preserving the modeling freedom. This is the algorithm that we advise for all practical work.

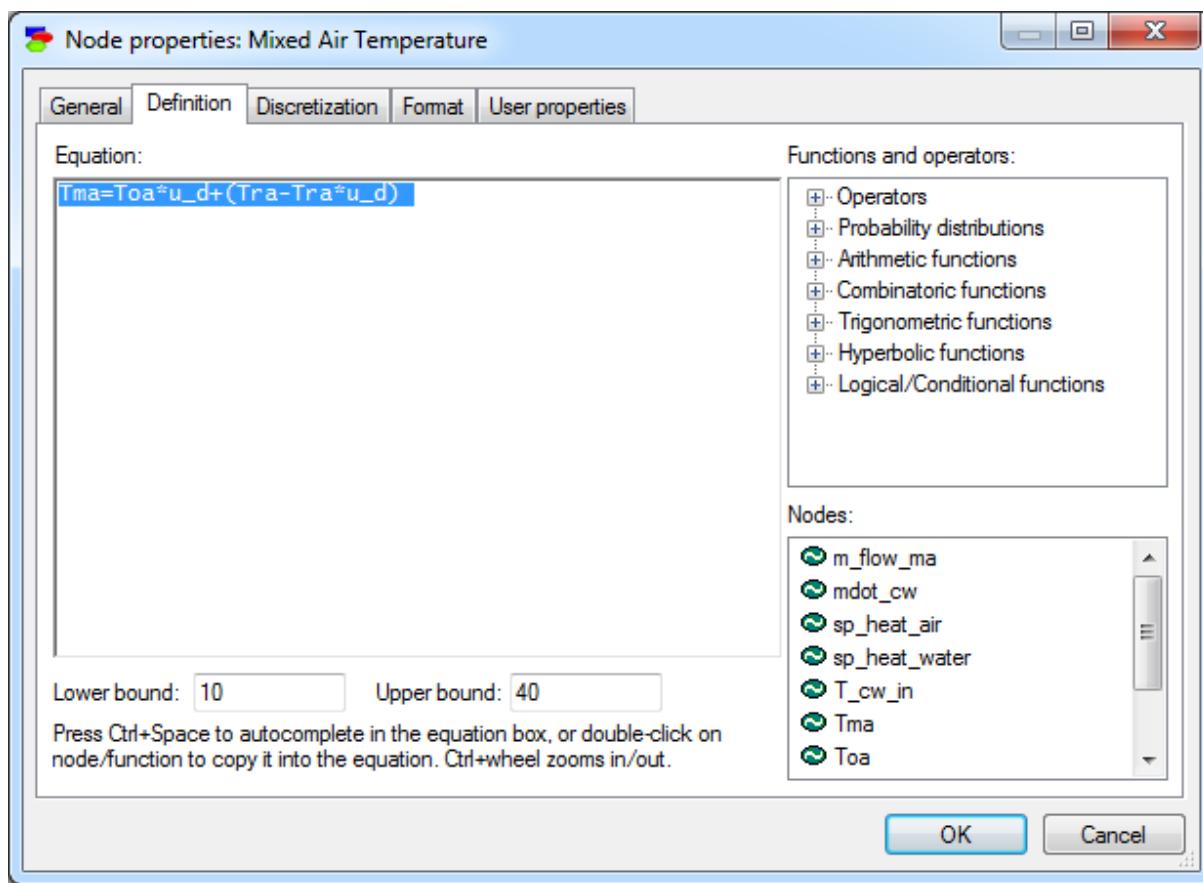
5.7.8.2 Autodiscretization

Autodiscretization is an approach to solving hybrid [Bayesian networks](#)⁴⁵, i.e., networks including both discrete and continuous variables. It offers an intermediate stage, in which it 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 as a discrete Bayesian network. The big advantage of this algorithm is that it converts the original hybrid Bayesian network into a discrete Bayesian network only for the purpose of inference, preserving the modeling freedom. This is the algorithm that we advise for all practical work.

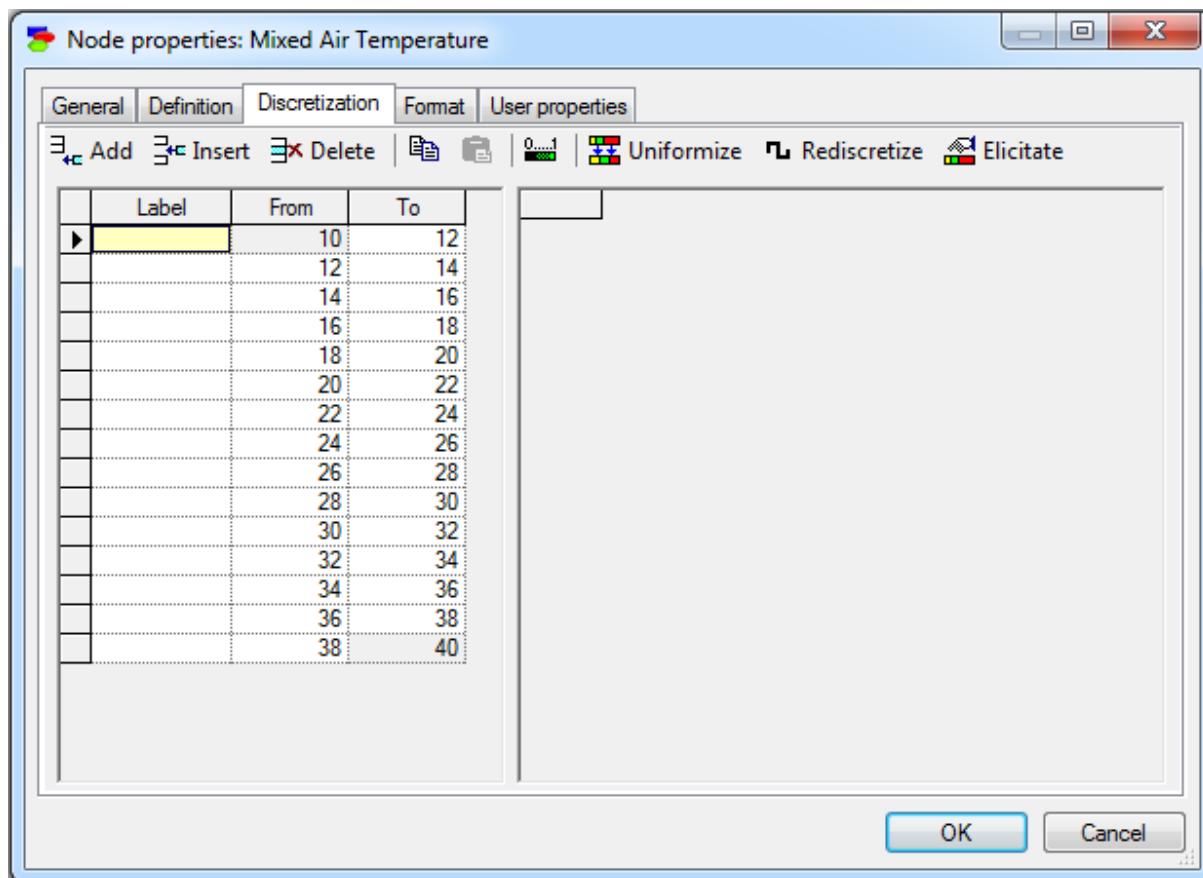
Consider the following mode consisting of continuous variables bound by equations and describing temperature flow in a heating and cooling system:



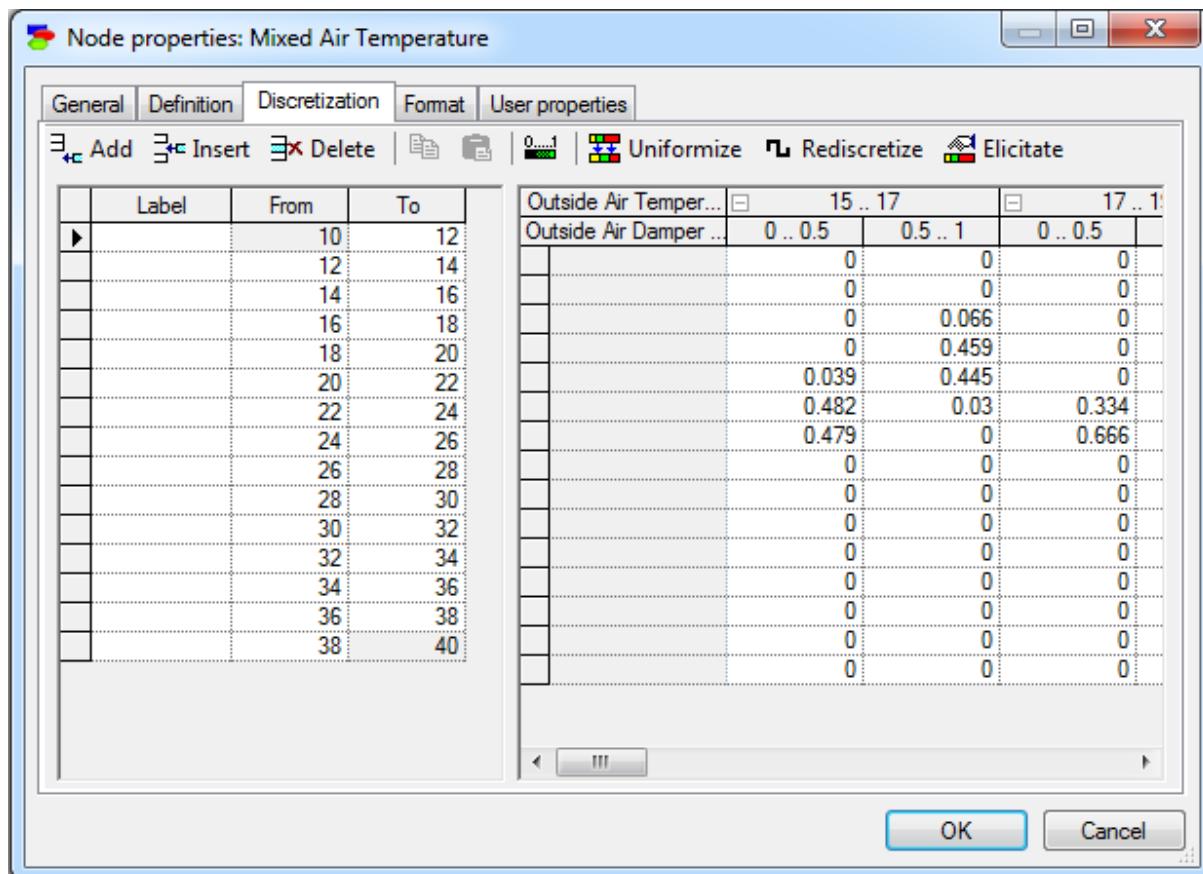
Continuous nodes in GeNle are not bound by any constraints and its definition can be a general equation, including any function. Here is the definition of the node Tma (Mixed Air Temperature):



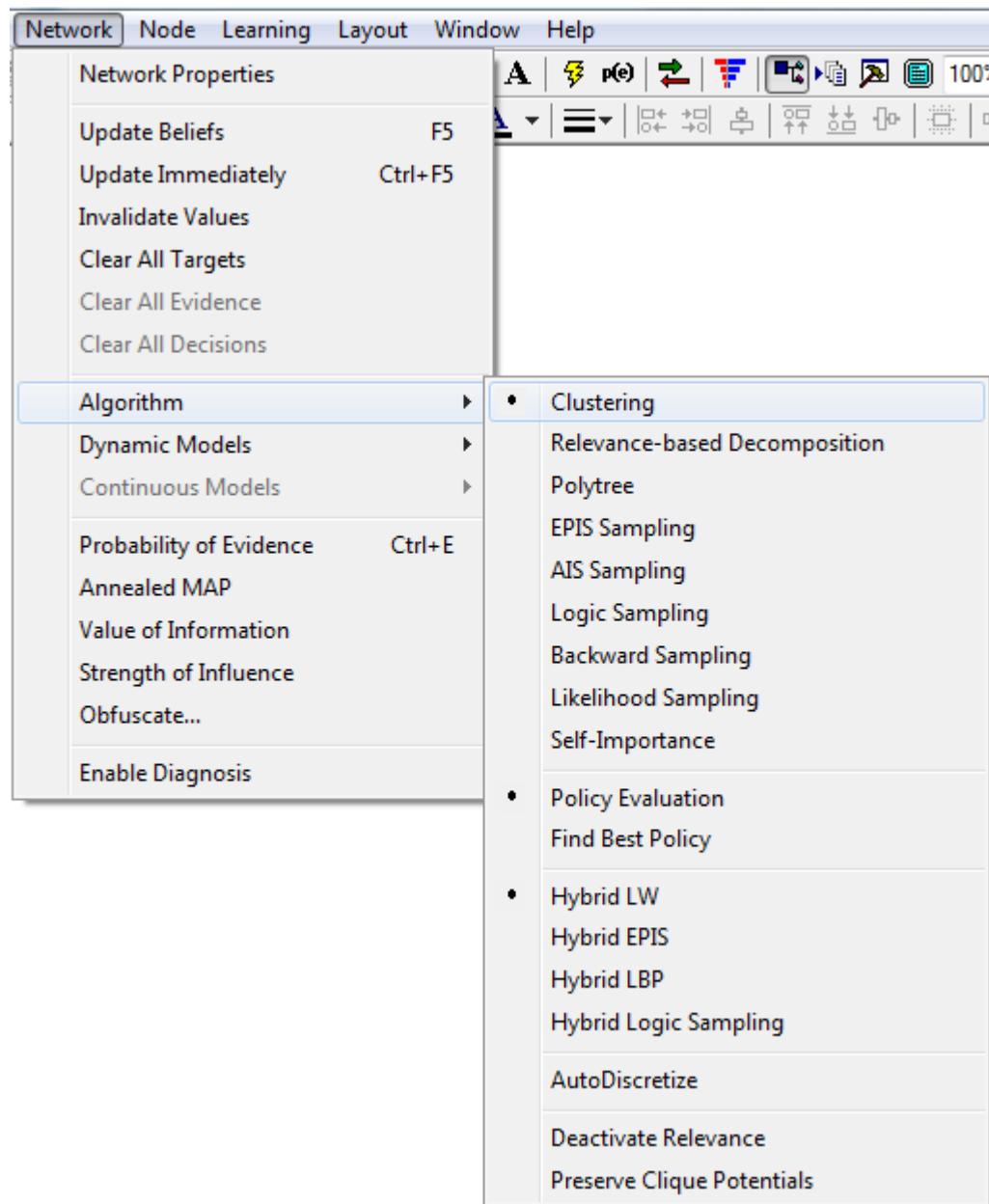
Every continuous node in GeNIE has a *Discretization* tab. The *Discretization* tab specifies how the variable should be discretized for the purpose of inference. Here is a possible discretization for the node *Tma*:



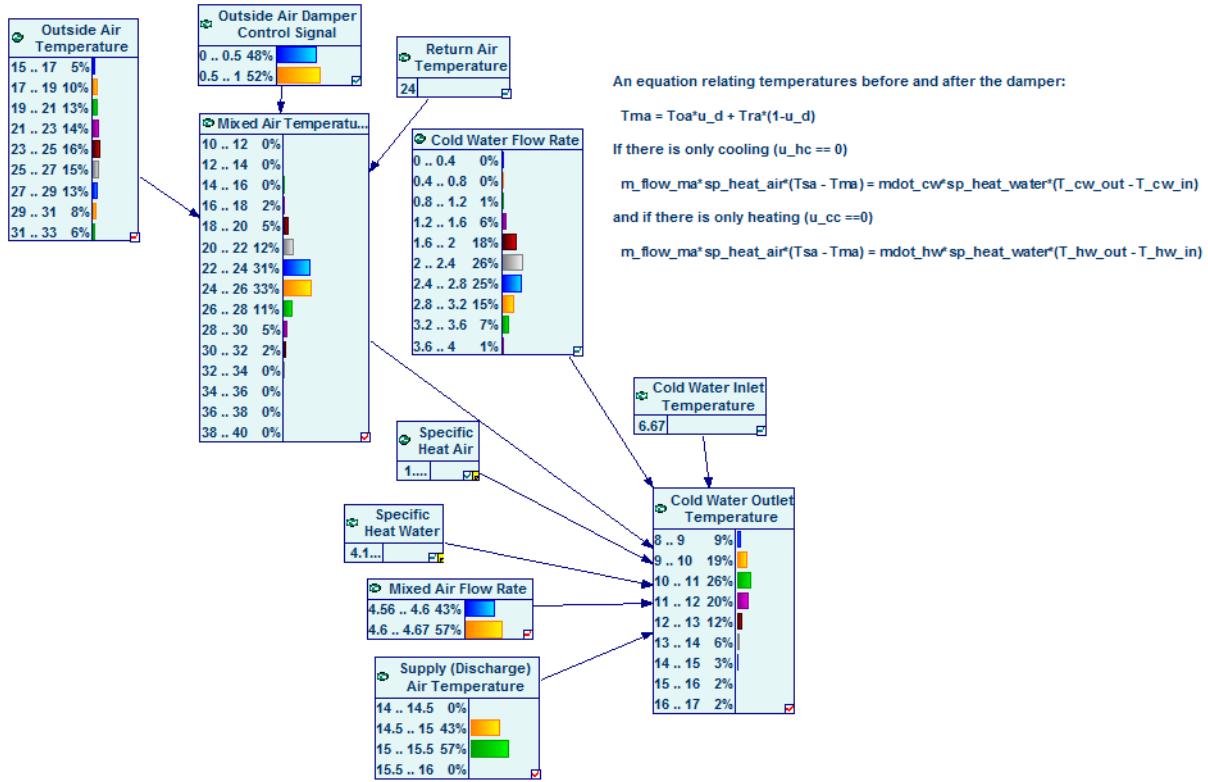
Clicking on the button *Rediscretize* (Rediscretize) invokes a simple sampling algorithm that, based on the discretization of the node and the discretizations of its parents, derives the discrete conditional probability distribution for the node. Here is a distribution derived for the node *Tma*:



We can convert the original hybrid network into a discrete Bayesian network for the purpose of inference by turning on *Autodiscretization*. To turn on *Autodiscretization*, please choose the *AutoDiscretize* from the *Network-Algorithms* menu:



The result is a (virtual) conversion of the original network into a discrete Bayesian network:



which can be used for probabilistic inference.

5.7.8.3 Hybrid LW

This is a simple modification of the [Likelihood Weighting](#)²¹⁴ sampling algorithm that works in hybrid Bayesian networks, i.e., networks including both discrete and continuous variables. The algorithm was mentioned in (Yuan & Druzdzel, 2007).

5.7.8.4 Hybrid Logic Sampling

This is a simple modification of the [Probabilistic Logic Sampling](#)²¹⁴ algorithm that works in hybrid Bayesian networks, i.e., networks including both discrete and continuous variables. The algorithm was mentioned in (Yuan & Druzdzel, 2007).

5.7.8.5 Hybrid LBP

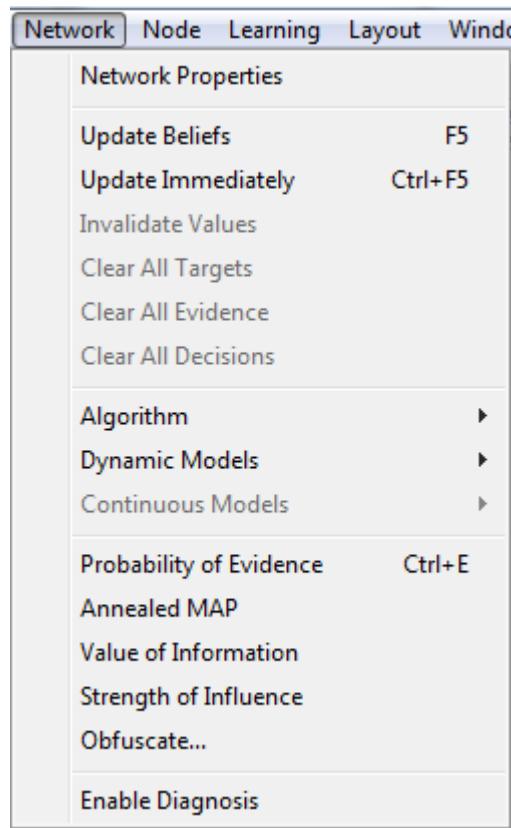
Hybrid Loopy Belief Propagation (LBP) is an algorithm for general hybrid Bayesian networks that contain mixtures of discrete and continuous variables and may represent linear or nonlinear equations and arbitrary probability distributions and naturally accommodate the scenario where discrete variables have continuous parents. Details of the *Hybrid LBP* algorithm can be found in (Yuan & Druzdzel, 2006), where it was originally proposed.

5.7.8.6 Hybrid EPIS

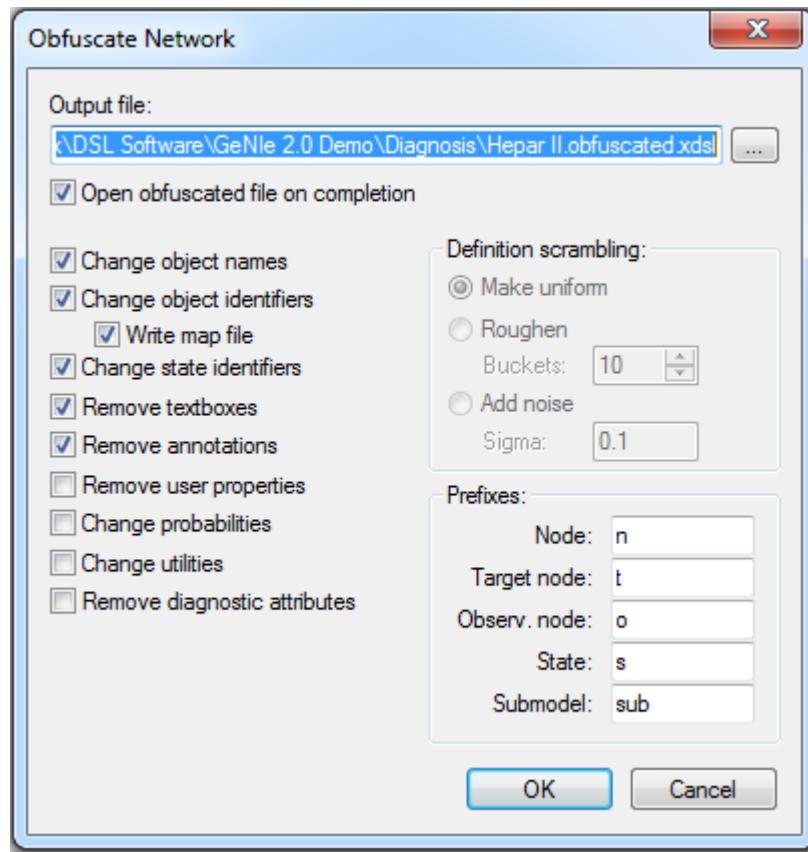
Hybrid Estimated Posterior Importance Sampling (HEPIS) is an importance sampling algorithm for hybrid Bayesian networks, i.e., networks including both discrete and continuous variables. Details of the *Hybrid EPIS* algorithm can be found in (Yuan & Druzdzel, 2007), where it was originally proposed.

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 $\text{Normal}(0, \sigma)$ 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

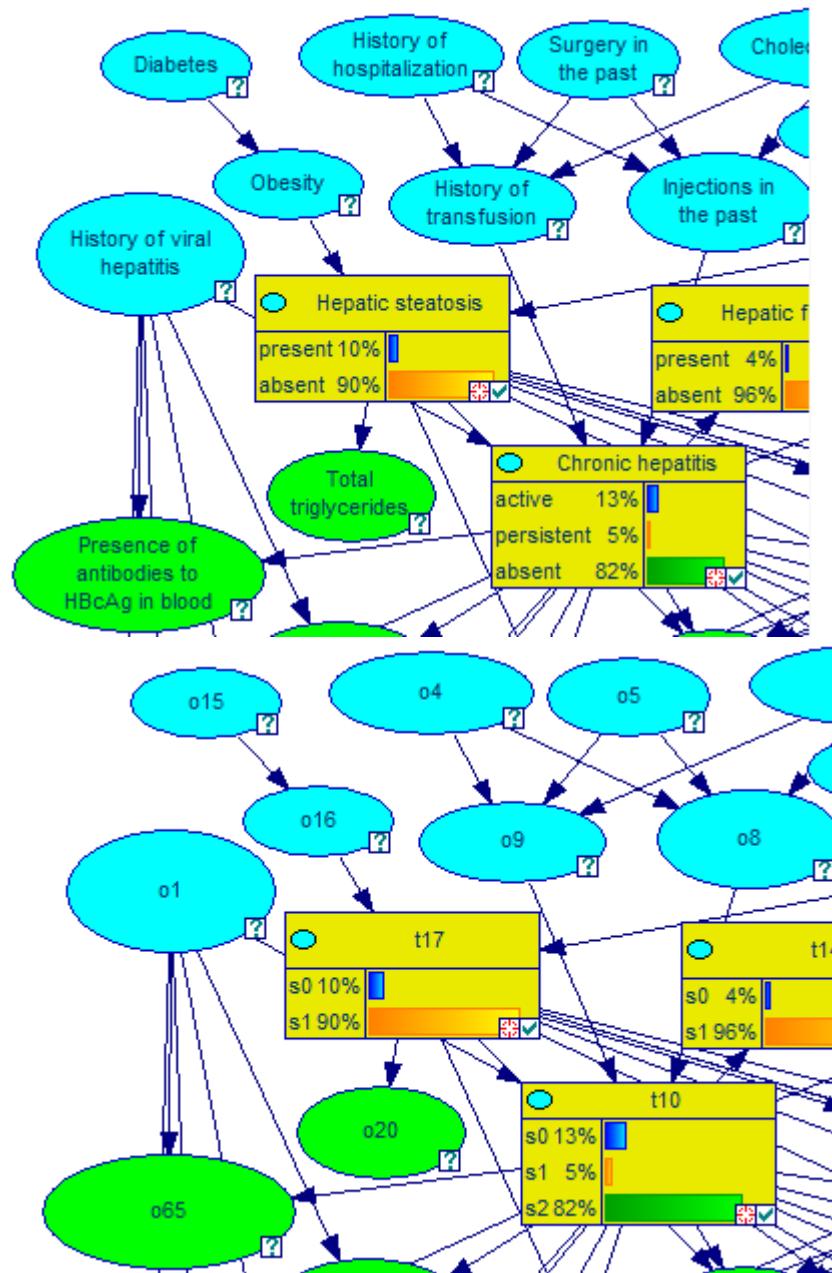
Observ. node: o

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:

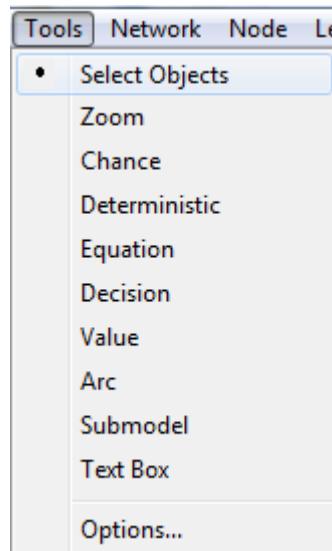
History of transf...		present			
History of viral ...		present		absent	
Injections in th...	present	absent	present	absent	
► active	0.20942408	0.46153846	0.06	0.13043478	
persistent	0.0052356	0.30769231	0.06	0.04347826	
absent	0.78534031	0.23076923	0.88	0.82608696	
o9		s0			
o1		s0	s1		
o8		s0	s1	s0	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.

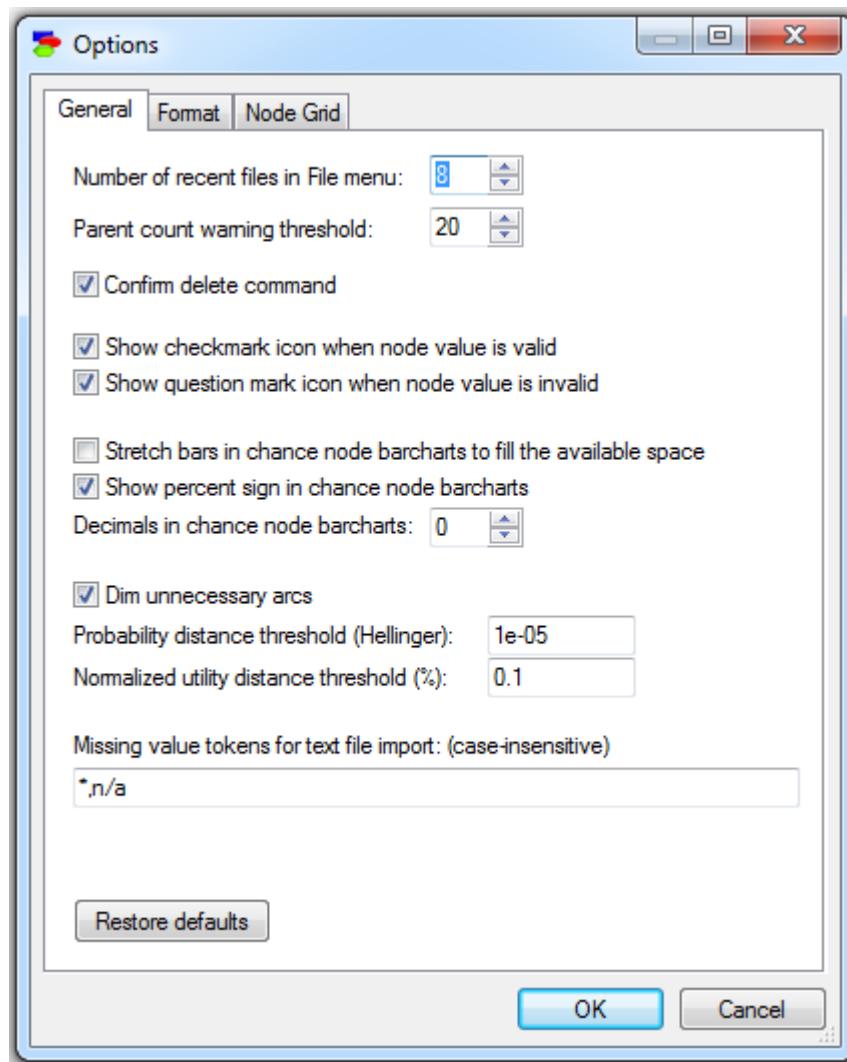
The obfuscation process creates two files on the disk: The obfuscated model (suffix *.obfuscated.xdsl*) and the mapping file (suffix *.obfuscated.map*). The latter is a text file containing a list of pairs of node IDs, the original and the obfuscated ID. When sharing the obfuscated model with somebody (e.g., with BayesFusion support folks), please do not send the map file, just the obfuscated model.

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](#)¹⁹³ 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. There are modeling tricks that a model author can apply to reduce the number of parents. 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 icons¹¹⁶](#).

Stretch bars in chance nodes barcharts to fill the available space controls the appearance of the node barchart view.

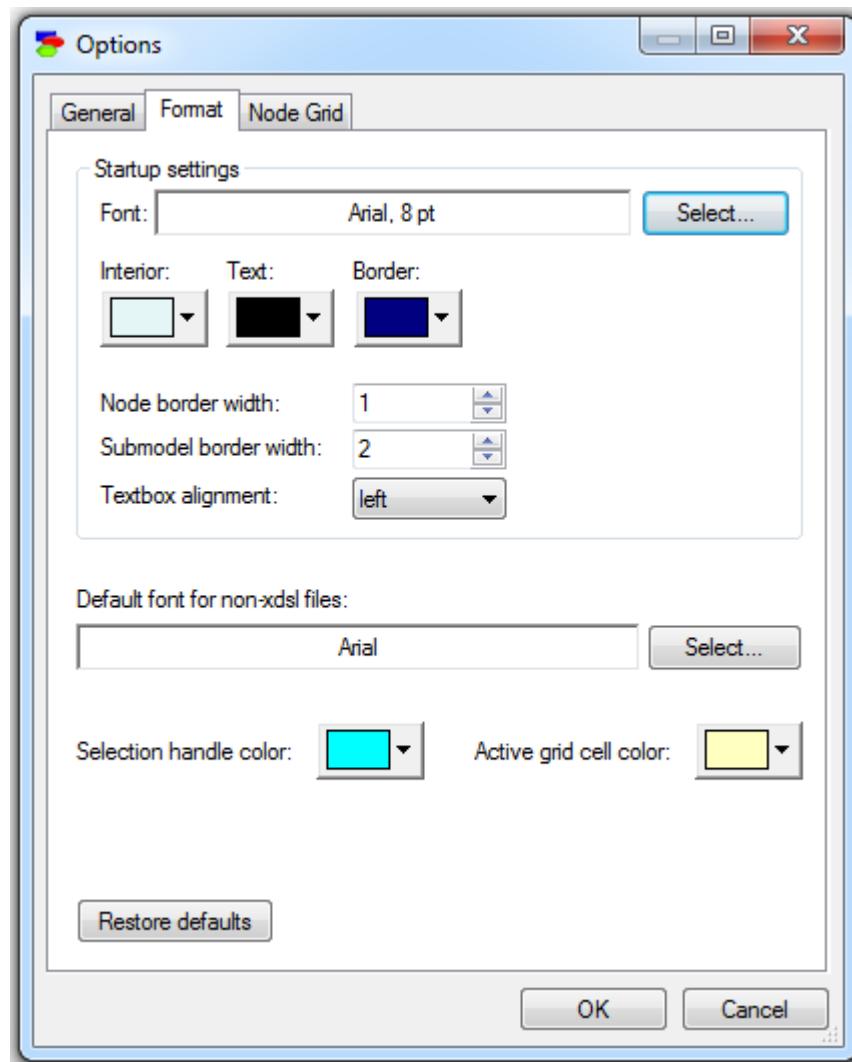
Show percent signs in chance node barcharts and *Decimals in chance node barcharts* control the display of numerical posterior probabilities in the node barchart view.

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 GeNle 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. 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.

Format tab

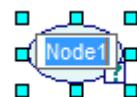
The *Format* tab allows for setting the default format for objects in the *Graph View*. The format can be changed later but here is where we set the initial values.



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.

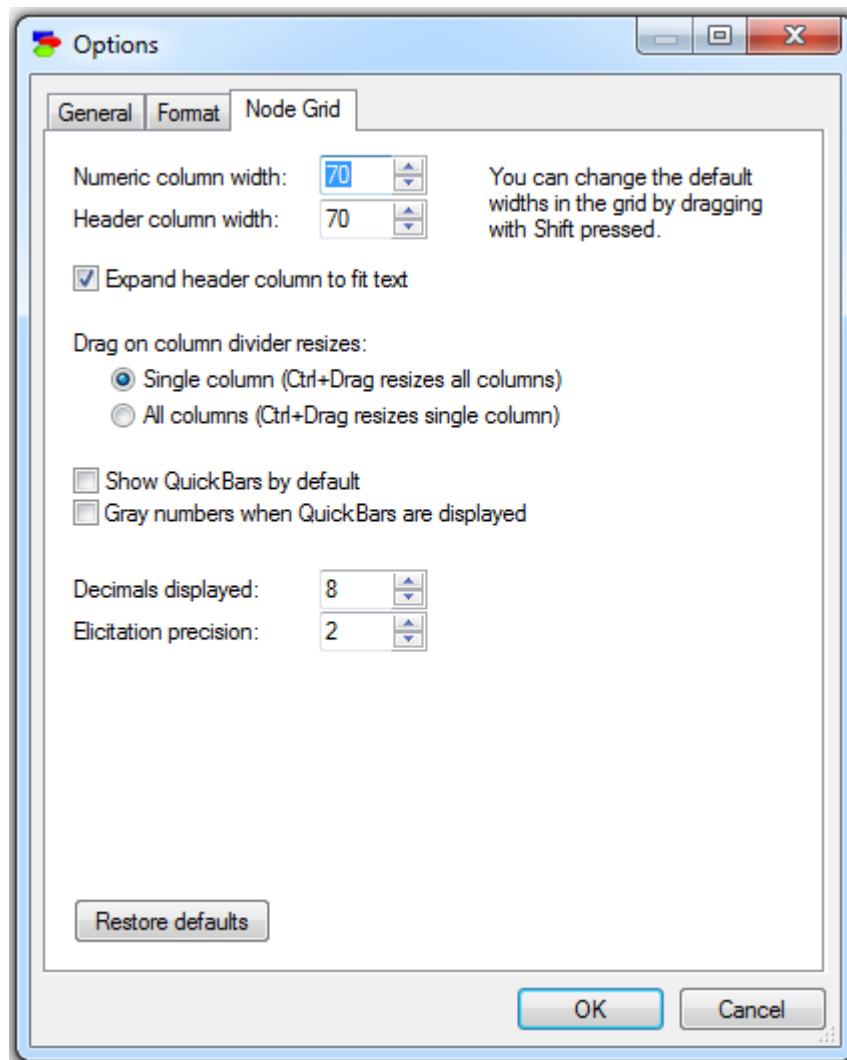


Active grid cell color is the color or a CPT cell that is active, i.e., that is selected by the cursor, like the cell for the state *Success* below.

►	Success	0.2
	Failure	0.8

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 8) control precision of numerical probabilities in the CPTs.

Elicitation precision (default 2) controls the precision of graphical methods of probability elicitation (piechart and barchart)

5.10 Keyboard shortcuts

File operations

CTRL+N: Create a new network

CTRL+O: Open an existing network

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+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

F7: Show diagnosis *Testing 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

Renaming objects

F2: Rename object

Editing

CTRL+B: Bold font

CTRL+I: Italic font

CTRL+C: Copy

CTRL+V: Paste

CTRL+X: Cut

Using GeNle

6 Using GeNle

6.1 Introduction

This section presents various modules of GeNle from the point of view of their function. It is an alternative view to the one presented in the previous section, which focused on GeNle's building blocks.

6.2 Bayesian networks

6.2.1 Building a Bayesian network

Building a [Bayesian network](#)⁴⁵ in GeNle is demonstrated step for step in section [Hello GeNle!](#)¹²

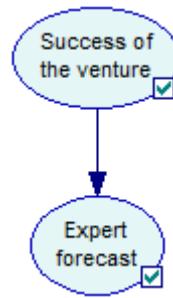
6.2.2 Useful structural transformations

This section reviews two useful structural transformation of [Bayesian networks](#)⁴⁵ that preserve the joint probability distribution represented by the network: arc reversal and node marginalization.

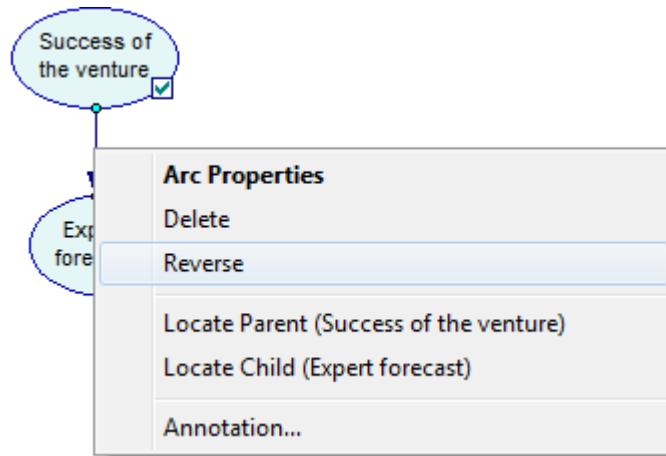
Arc reversal

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.

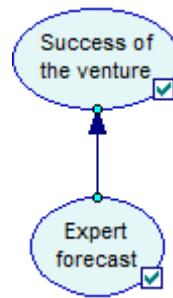
Consider the simple Bayesian network used in the *Hello GeNle!* example



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	Failure
► Success	0.2	
Failure	0.8	

	Good	Moderate	Poor
► Success	0.5	0.25	0.076923077
Failure	0.5	0.75	0.92307692

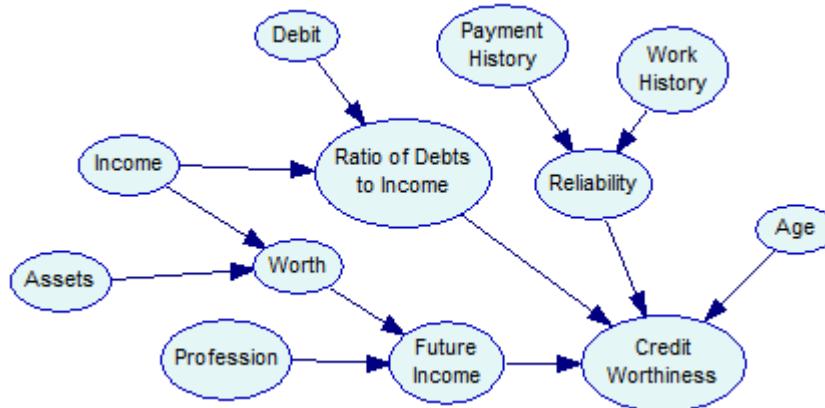
Similarly, different is the CPT of the node *Expert forecast* before (left) and after (right) the operation of arc reversal.

Success of the...	Success	Failure
► Good	0.4	0.1
Moderate	0.4	0.3
Poor	0.2	0.6

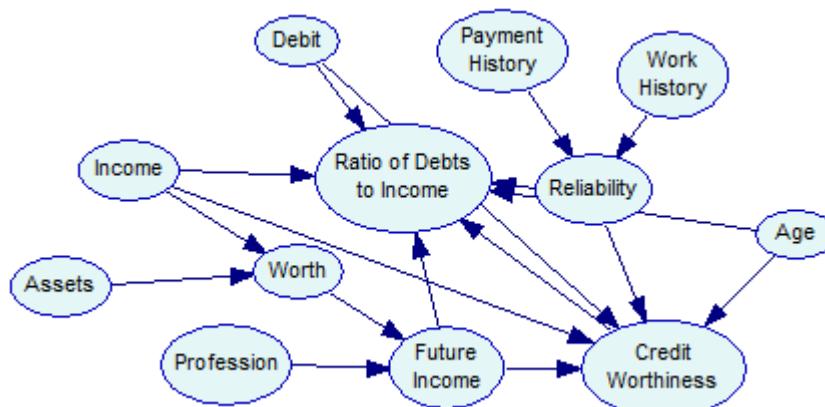
► Good	0.16
Moderate	0.32
Poor	0.52

The CPTs are recalculated in such a way that they yield the same joint probability distribution over all nodes in the network.

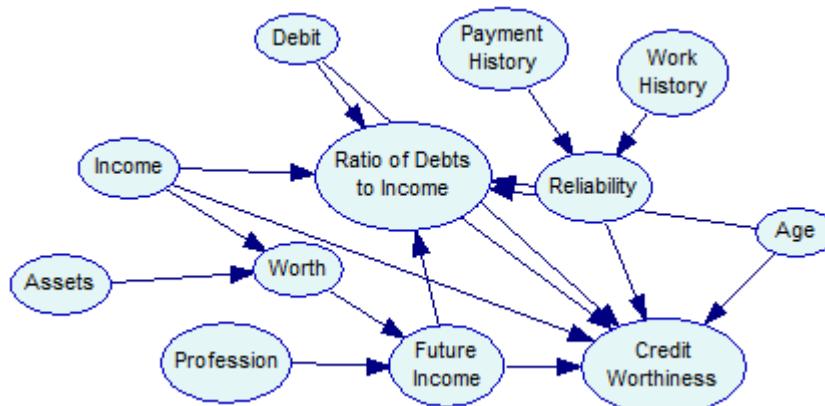
There is an additional complication related to reversing an arc: Because the structure of the network encodes explicitly independences in the domain, reversing an arc violates the pattern of independences. To account for this, in general the two nodes at the opposite ends of the arc need to inherit their parents. In a complicated network, the operation of arc reversal may, thus, lead to a 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



Reversing the same arc back does not necessarily restore the original graph

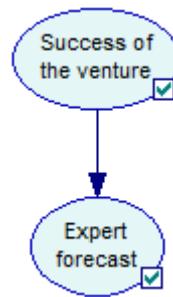


Our advice is to use this operation rarely, if possible. It preserves the joint probability distribution represented by the model but it may lead to loss of a very important property, notably the structure of the underlying graph.

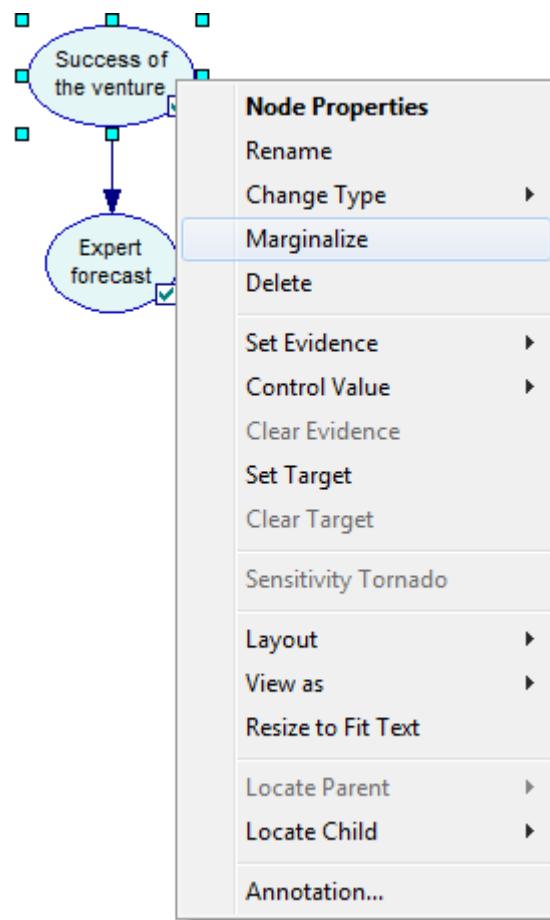
Node marginalization

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 would lead to loss of the numerical properties of the network.

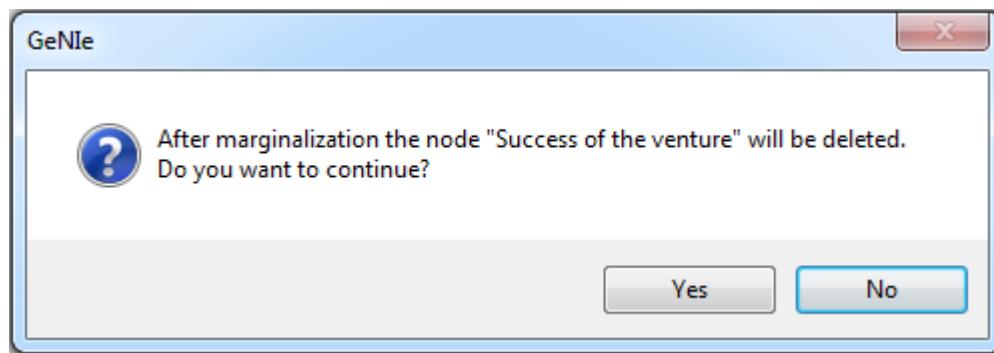
Consider again the simple Bayesian network used in the *Hello GeNle!* example



To marginalize the node *Success of the venture*, right-click on it and choose *Marginalize*.



GeNle issues a warning that the marginalized node will be deleted from the network. Pressing No gives us a chance to retract from the operation.

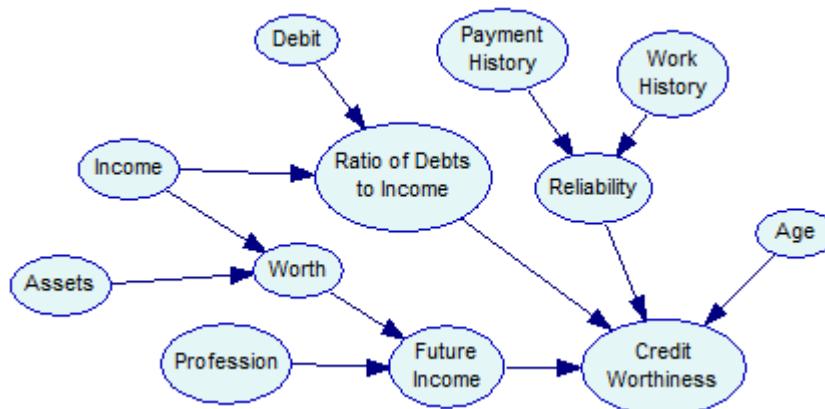


Pressing Yes, however, removes the node *Success of the venture* while modifying the CPTs of the remaining nodes so that they preserve their ability to represent the joint probability distribution. The CPT of the node *Expert forecast* is transformed to the following

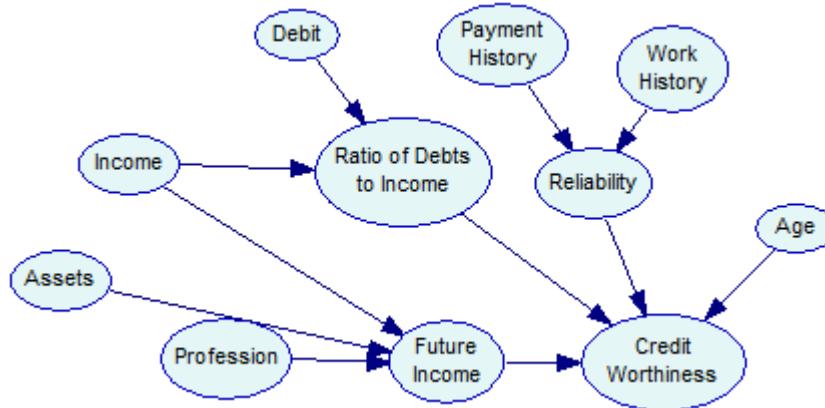
► Good		0.16
Moderate		0.32
Poor		0.52

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



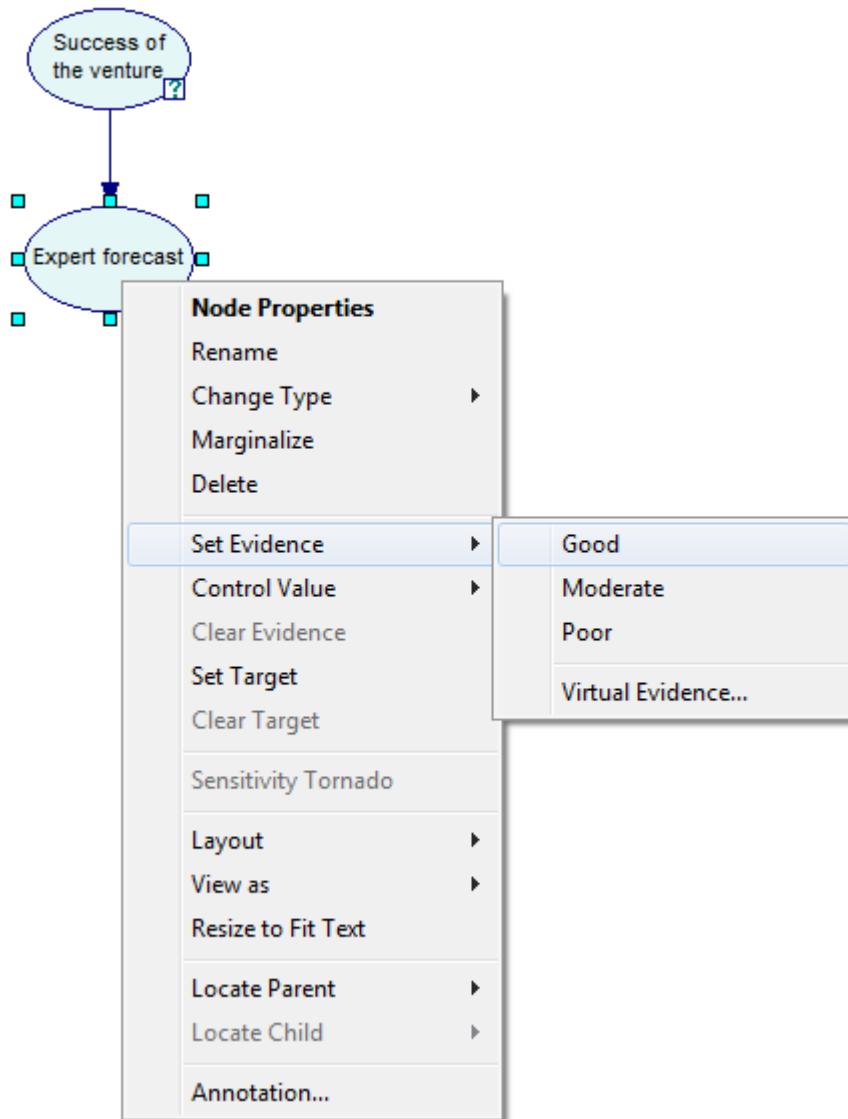
Please note the new arcs between *Income* and *Future Income* and *Assets* and *Future Income*. The operation of marginalization is powerful but it leads to loss of information, so we advise caution in using it.

6.2.3 Entering and retracting evidence

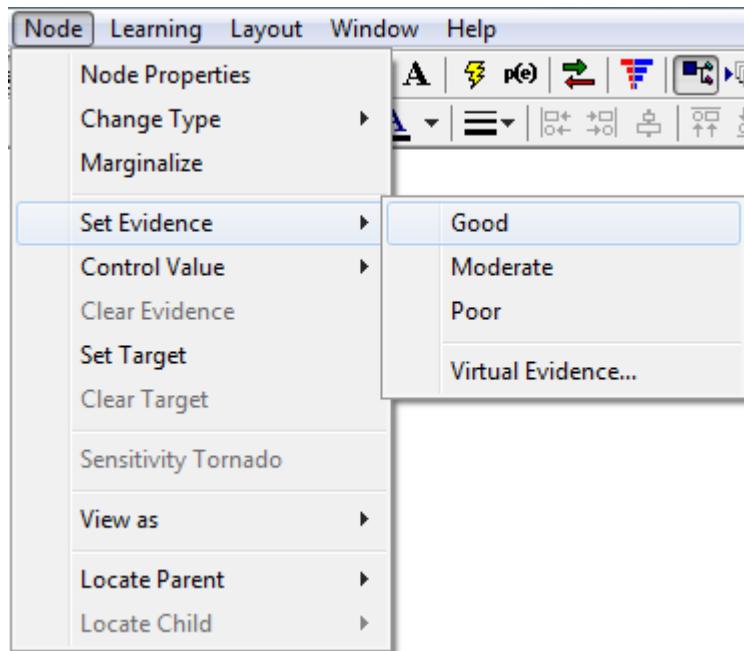
Entering observations (evidence) is one of the basic operations on a probabilistic model. It amounts to adjusting the model to a new situation, one in which more information is available. It allows to query the system subsequently about the new, posterior probability distributions. If you have gone through [Building a Bayesian network](#)¹², 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](#)¹² from the *Example Networks* folder.

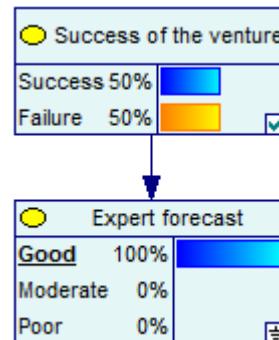
To enter evidence into your model, right-click on the node in question (in the picture below, node *Expert forecast*) and choose *Set Evidence*.



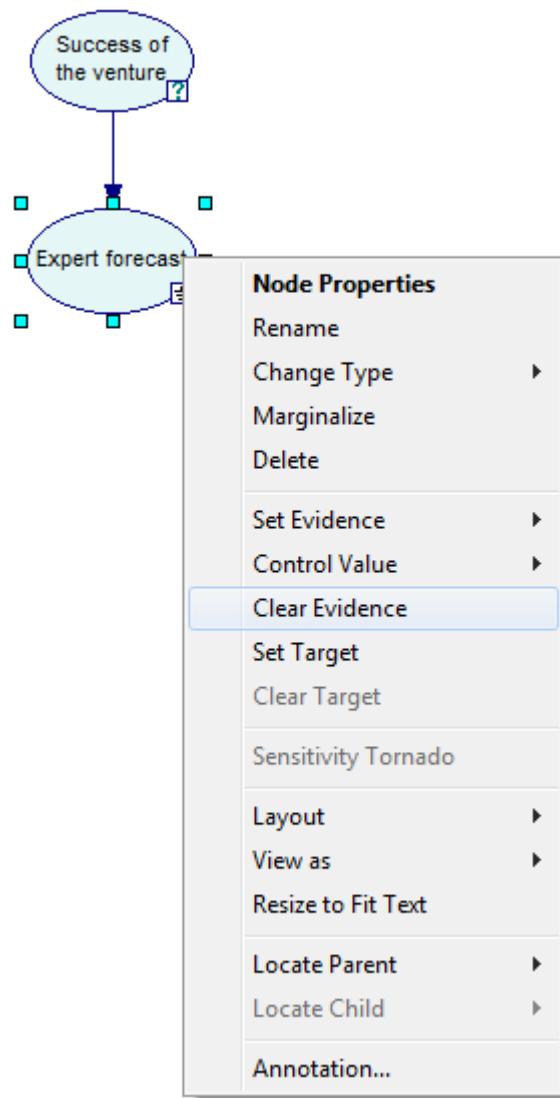
Alternatively, select the node in question and choose *Set Evidence* submenu from the [Node Menu](#)²⁰⁷.



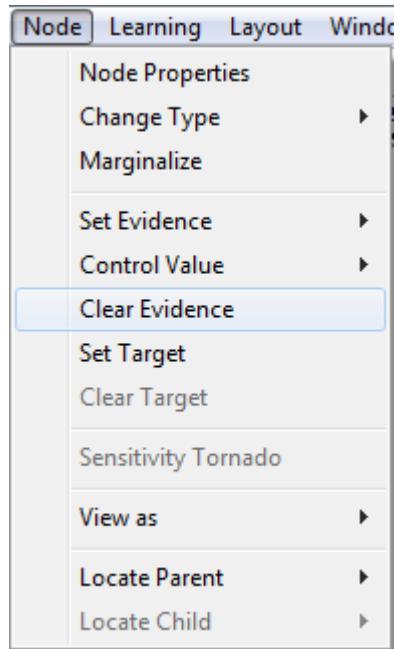
Yet another way is double-clicking on the observed state in the *Bar chart view* of the node



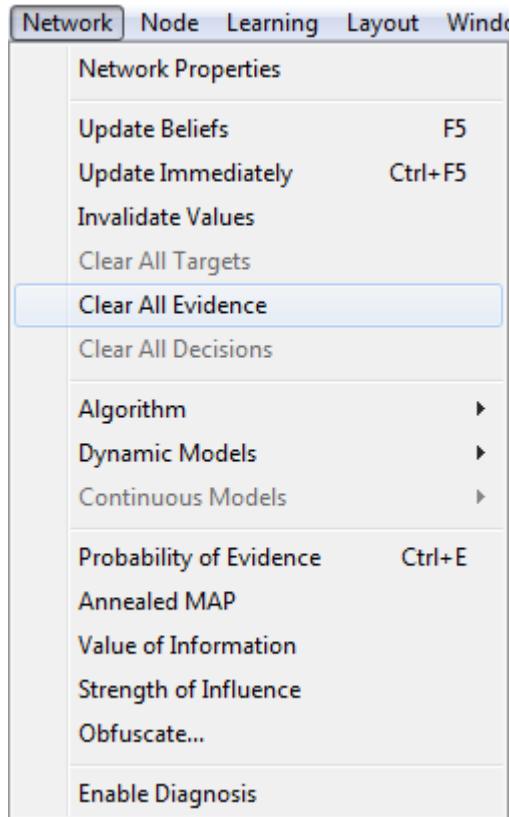
To retract evidence, right-click on the node with previously entered evidence (in the picture below, node *Forecast*) and choose *Clear evidence*.



Alternatively, select the node in question and choose *Clear evidence* from the [Node Menu](#)²⁰⁷.



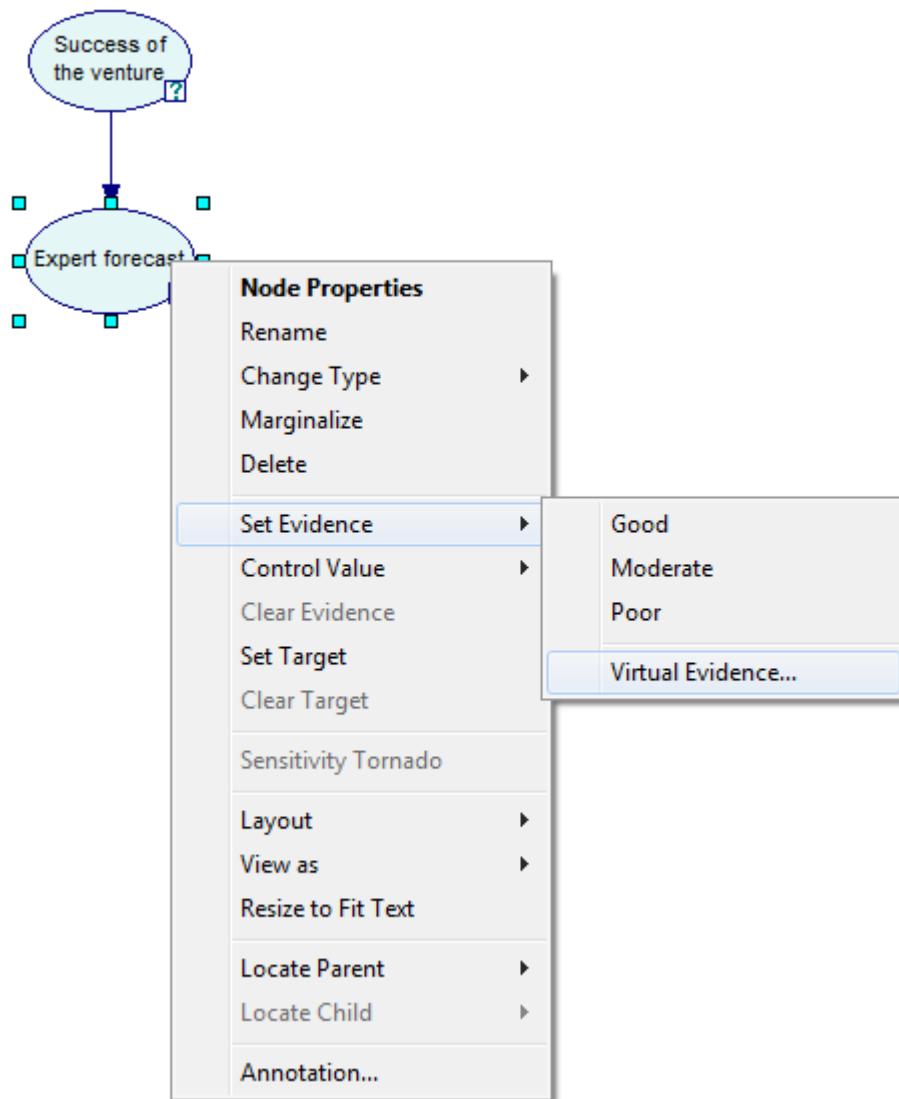
You can also retract all evidence by choosing *Clear all evidence* from the *Network* menu:



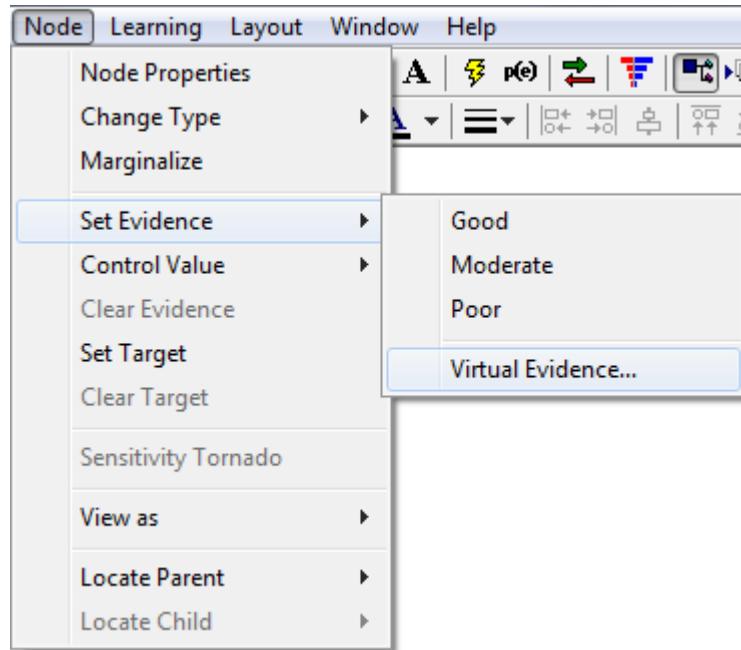
To enter different evidence instead of retracting evidence altogether, just set different evidence from the *Set Evidence* sub-menu.

6.2.4 Virtual evidence

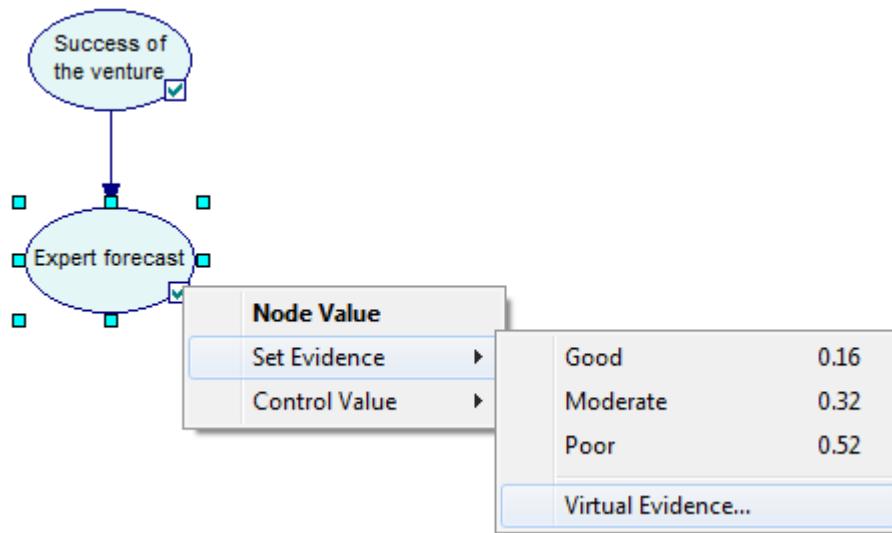
Entering virtual observations (evidence) is similar to entering evidence. The main difference is that instead of observing a state of a node, we enter a probability distribution over all states of the node. You may load the model *VentureBN.xdsl* created in [Building a Bayesian network](#)¹² from the *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*



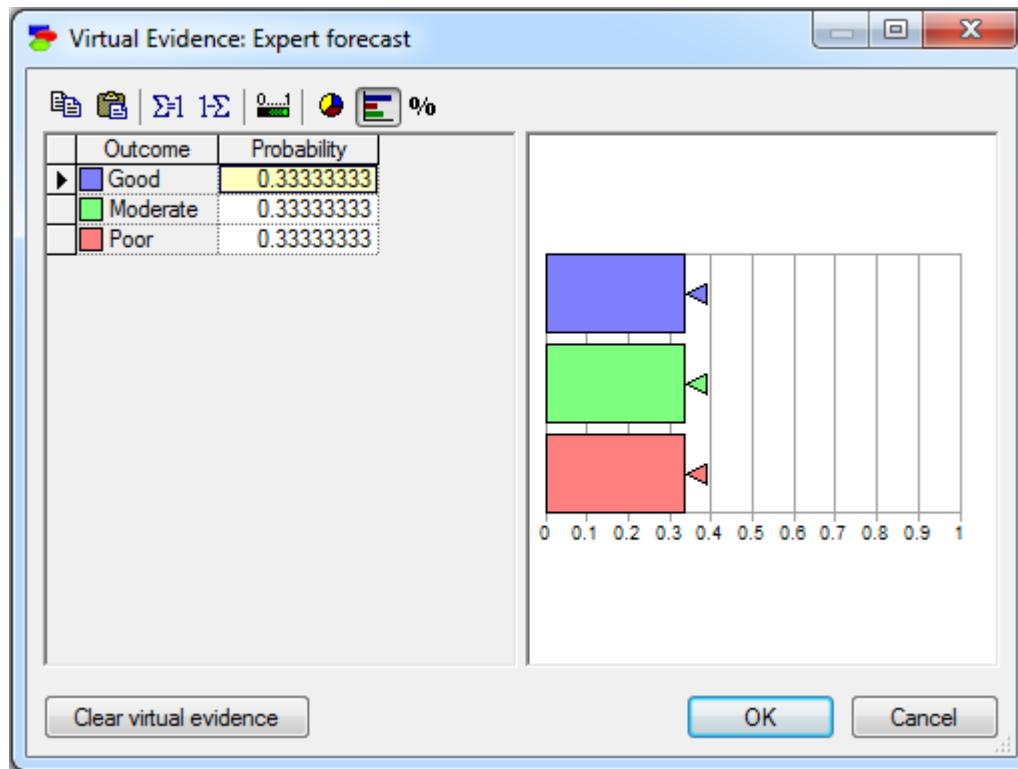
Alternatively, select the node in question and choose *Set Evidence-Virtual Evidence* sub-menu from the [Node Menu](#)²⁰⁷.



Yet another way is double-clicking on the observed state in the *Bar Chart View* of the node



In each of the 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. Individual probabilities in the distribution have to add up to 1.0.

The internal implementation of virtual evidence in GeNle 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.

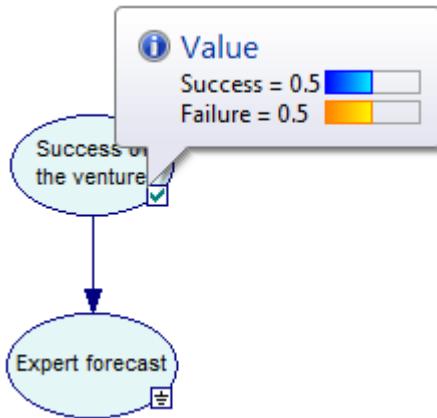
Retracting virtual evidence is identical to retracting evidence and has been described in section [Entering and retracting evidence](#).

To enter different evidence instead of retracting evidence altogether, invoke the *Virtual Evidence Dialog* and set different evidence.

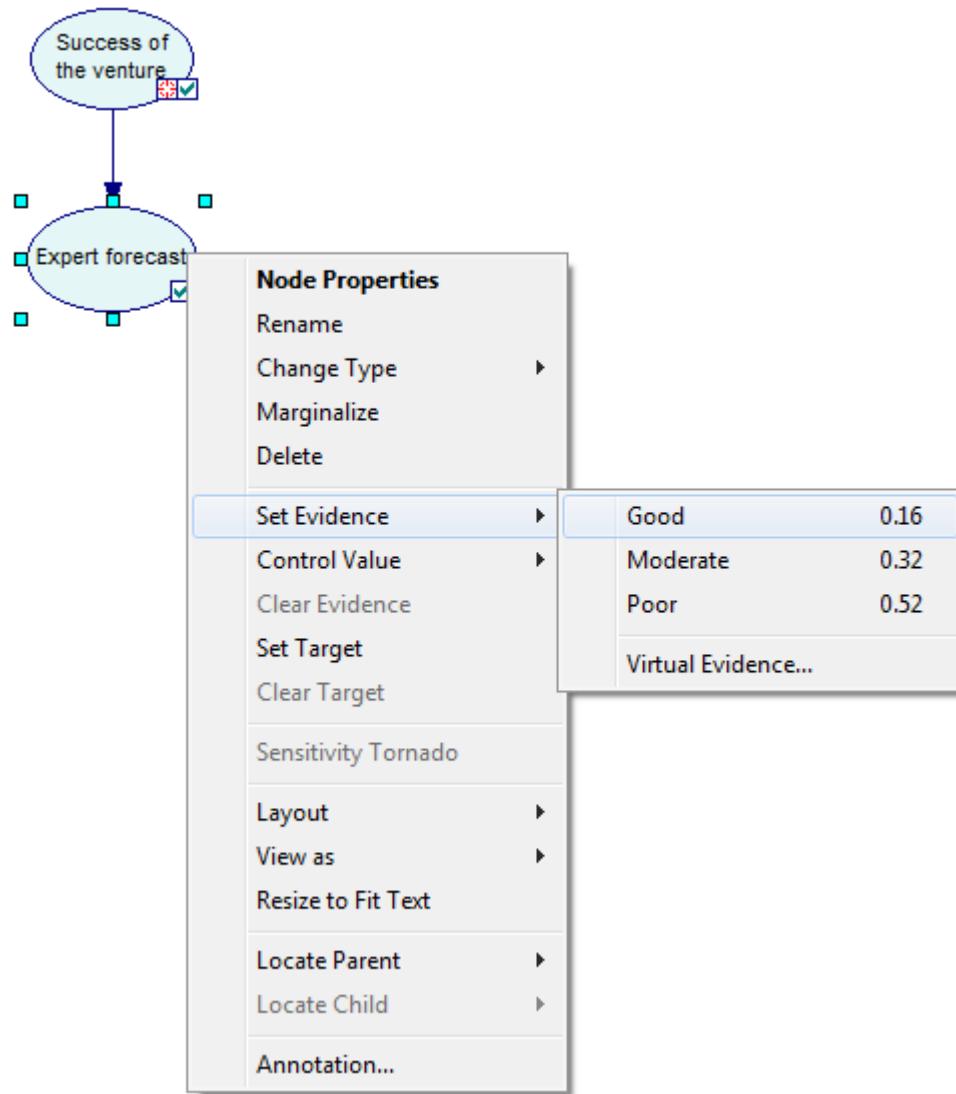
6.2.5 Viewing results

Marginal probability distributions

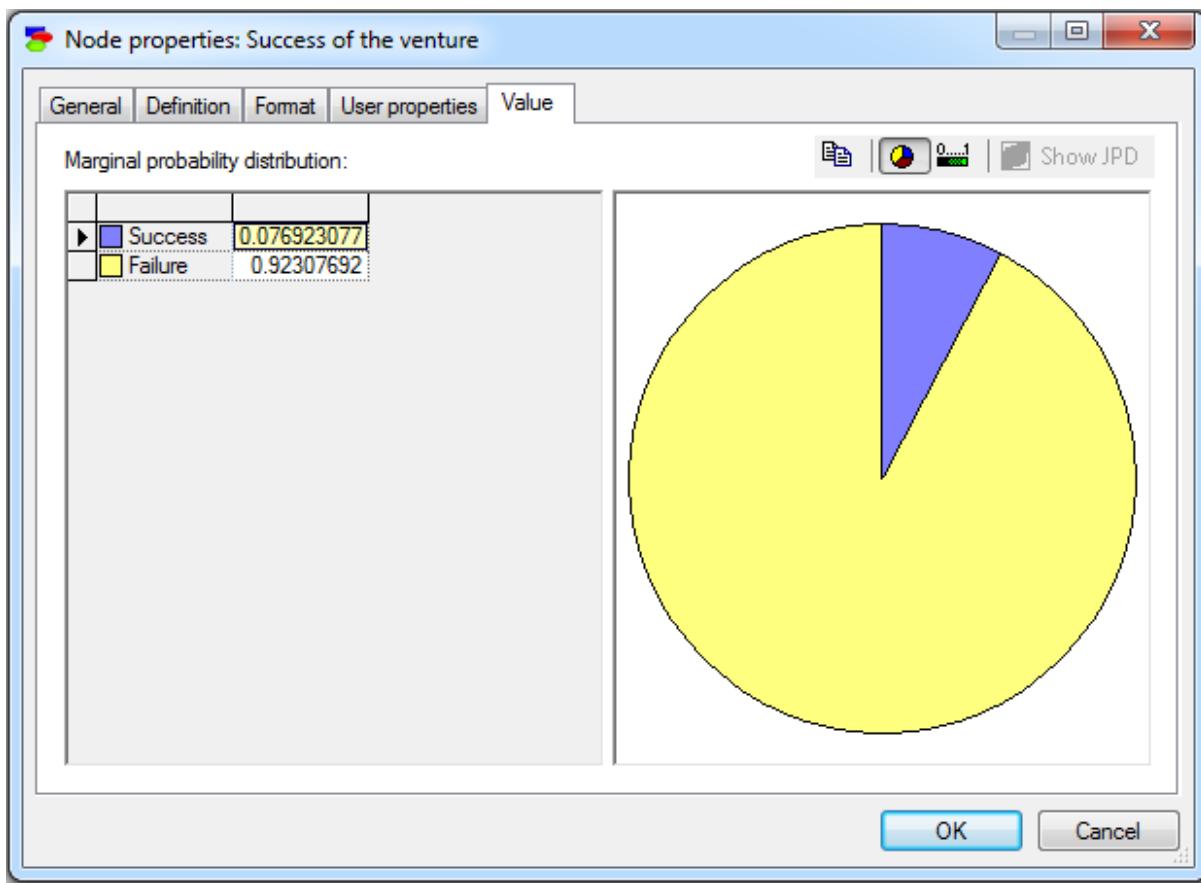
Once a model is evaluated, you are ready to view the results computed by GeNIE. This can be done in several ways. Posterior marginal probability distribution over any node in a [Bayesian network](#)⁴⁵ can be viewed by hovering the cursor over the status icon when it is *Updated* (). The probabilities of the various states of the node will be displayed as follows:



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](#)^[123]. Shown below is the *Value* tab when the Expert's prediction is *Poor*.

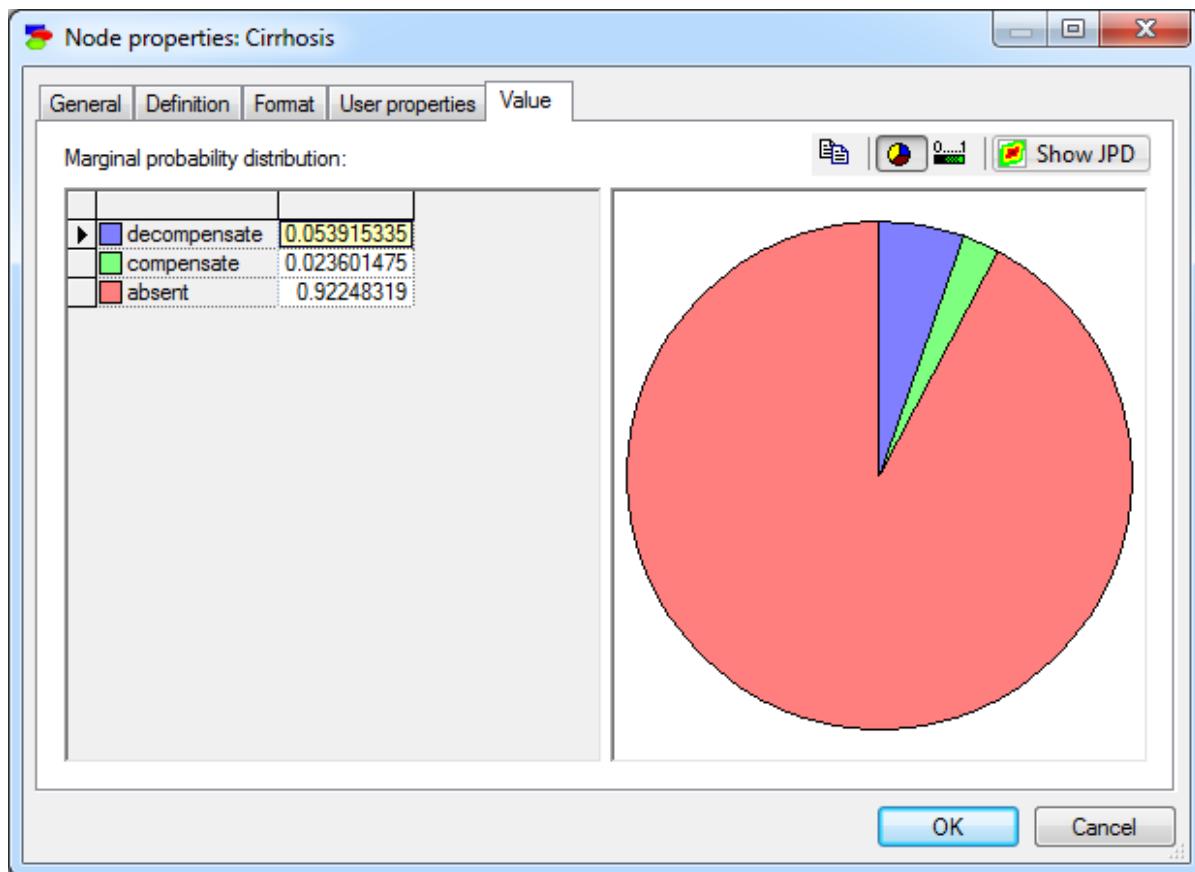


Posterior probability distribution in [influence diagrams](#)⁴⁷ are more detailed. In cases when a node's probability distribution is affected by a decision or by a node that precedes a decision node, the posterior probability distribution is indexed by the outcomes of these nodes. In such cases, right clicking on a node icon will display a message *The result is a multidimensional table, double click on the icon to examine it.*

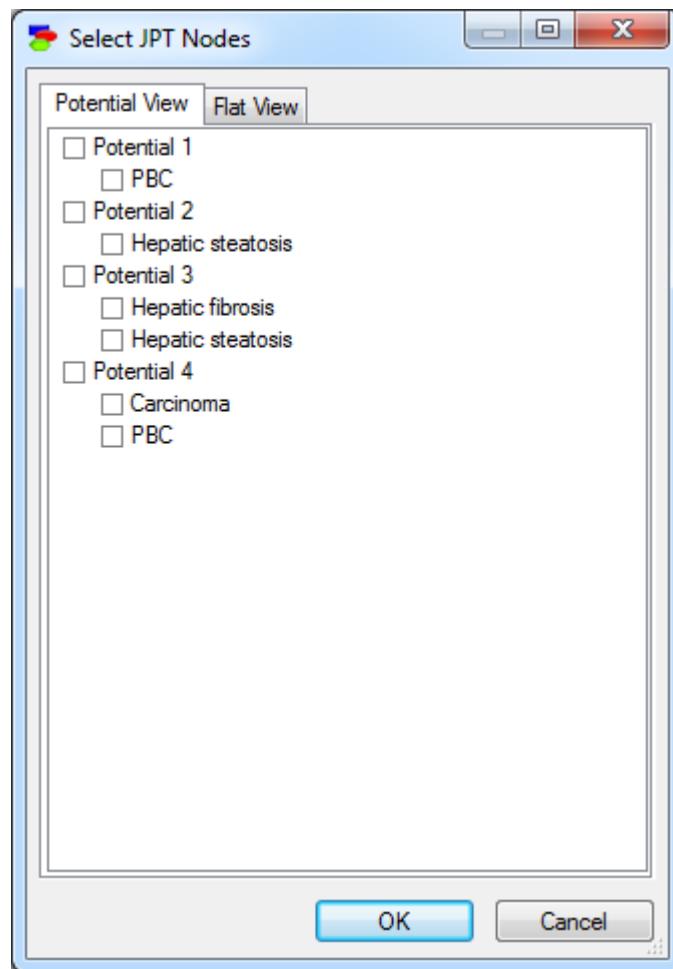
The only way to see the entire distribution, is by using the *Value Tab* of the decision node or the value node in question.

Joint probability distributions

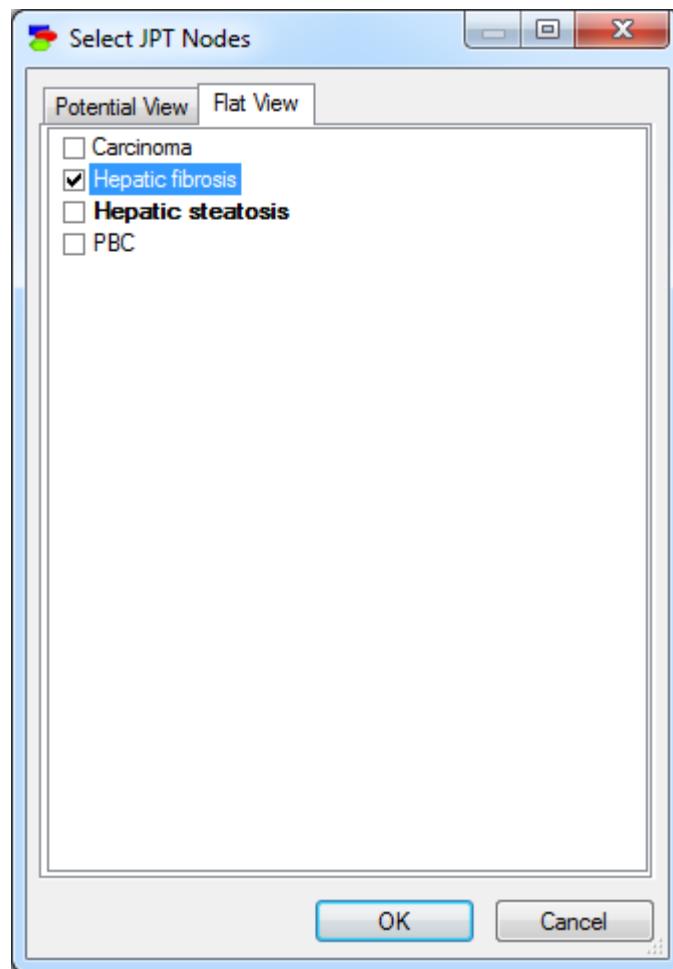
Preservation of clique potentials allows for viewing joint probability distribution over those variables that are located within the same clique. Should you wish to derive the joint probability distribution over any variable set, just make sure that they are in the same clique before running the clustering 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* ().



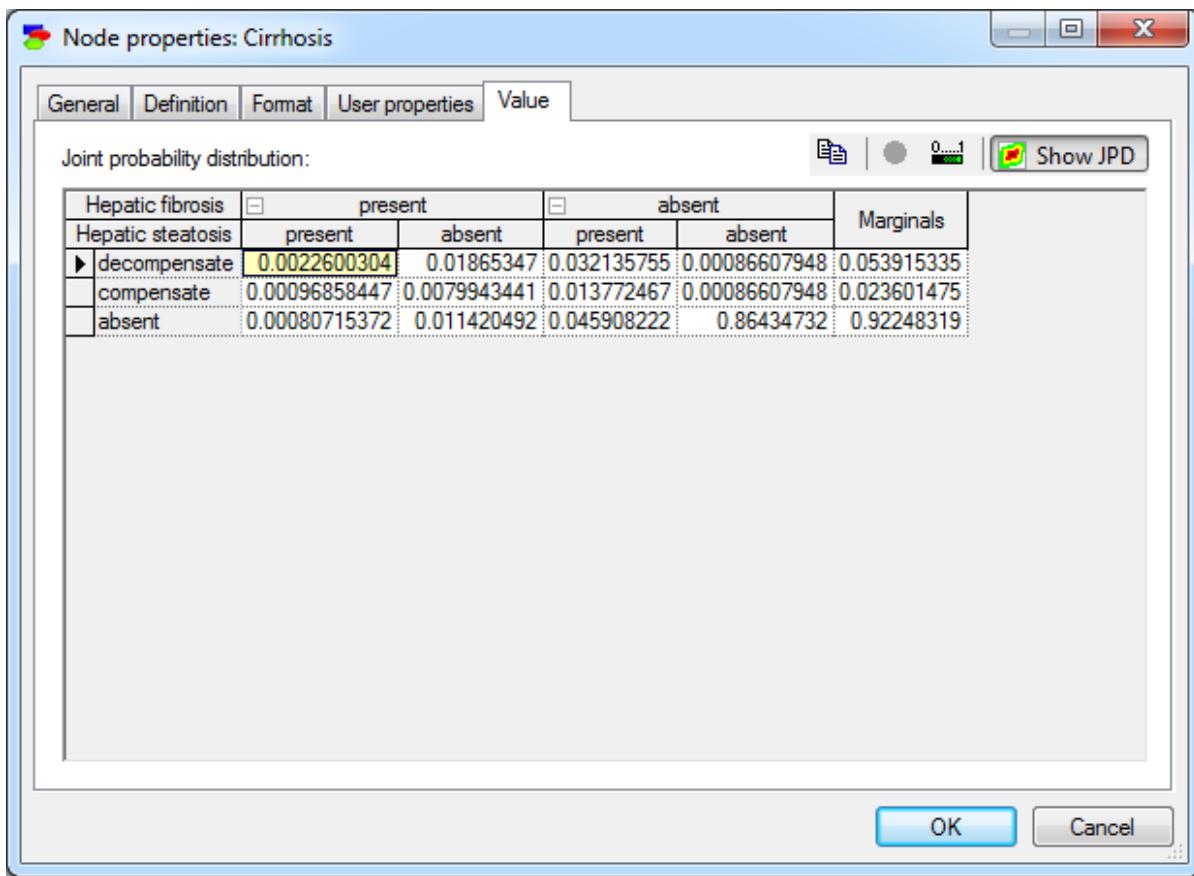
Pressing the *Show JPD* button invokes the following dialog:



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*.



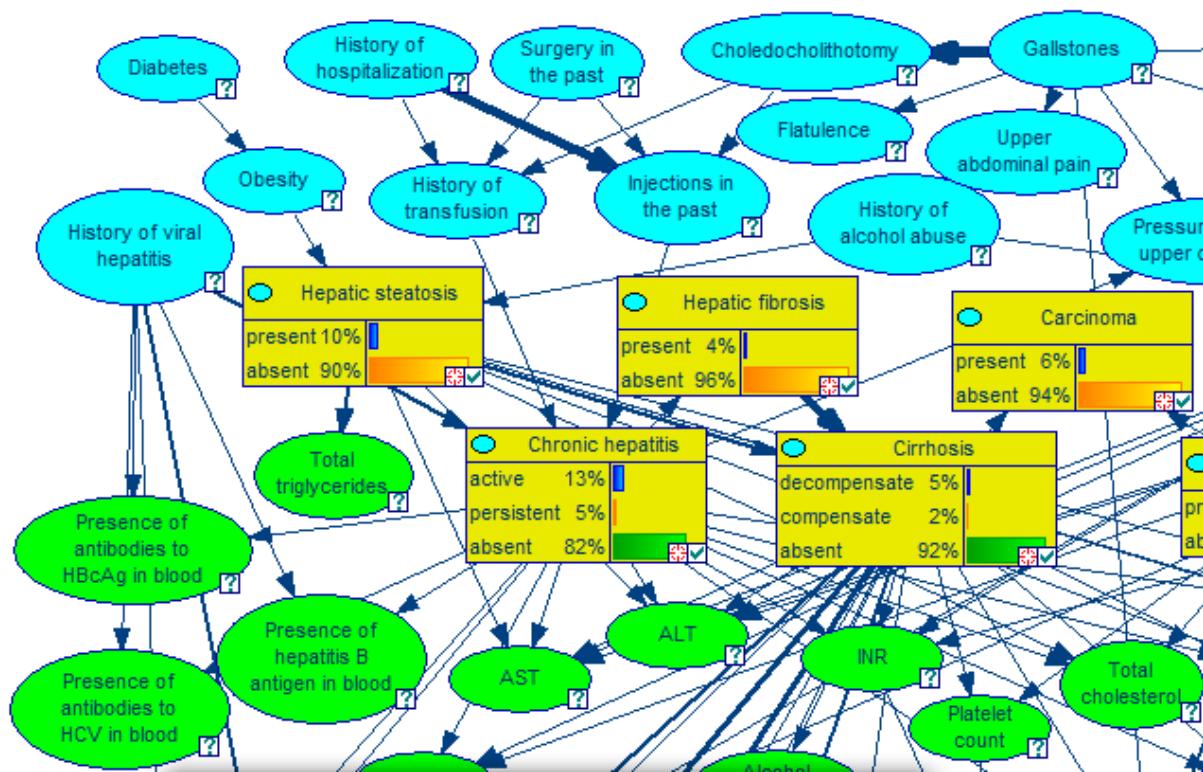
The clustering algorithm is GeNIE's default algorithm and should be sufficient for most applications. Only when networks become very large and complex, the clustering algorithm may not be fast enough. In that case, it is suggested that the user choose an approximate algorithm, such as one of the stochastic sampling algorithms. The best stochastic sampling algorithm available for discrete Bayesian networks is EPIS-BN (Yuan & Druzdzel, 2003).

6.2.6 Strength of influences

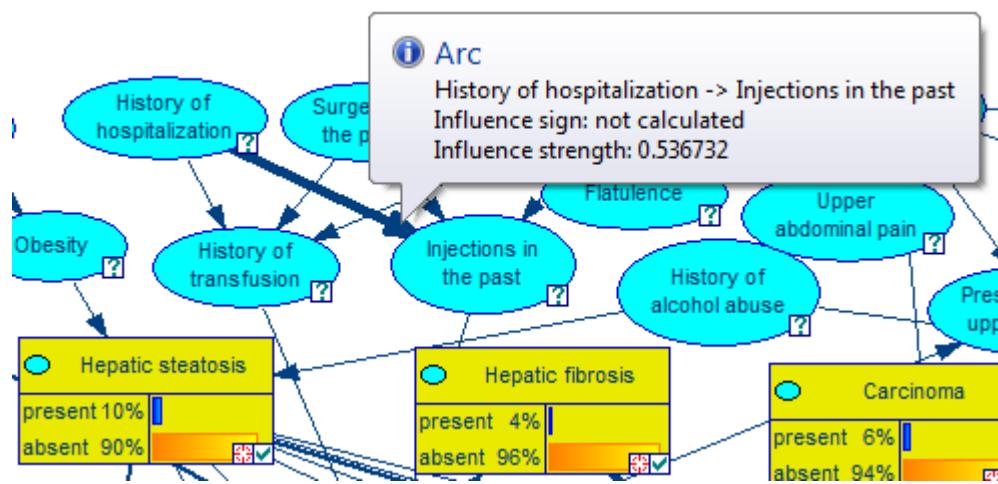
Clicking on the *Strength of Influence* (tool from the *Standard Toolbar*¹⁷⁶, brings up the *Influence Toolbar* and also changes the appearance of the arcs in the network. The arc have different thickness, dependent on the strength of influence between the nodes that they connect. Strength of influence is always calculated from the CPT of the child node and essentially expresses some form of distance between the probability distributions of the child node conditional on the state of the parent node.

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 toolbars 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*

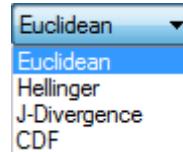


Maximum uses the largest distance between distributions, *Average* takes the plain average over distances, and *Weighted* weighs the distances by the marginal probability of the parent node.

The *Normalize* (**N**) 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* (**α**) button. The alpha parameter controls normalization of the J-Divergence.

Other settings

Influence shown can be *Static* (S) or *Dynamic* (D). 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 is context-specific and essentially shows the potential influence that two directly connected nodes can have on each other.

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.

When *Use arc thickness to show strength* () button is pressed, the thickness of the arcs is activated. The thickness of an arc indicates the strength of influence between the two nodes connected by that arc. The thickness in the static mode can be different from the thickness in the dynamic mode.

In the *Dynamic* mode, the strength of influence can be calculated from the parent to the child, from the child to the parent or in both directions. To visualize which option is used for a particular arc, icons are shown on the arcs to indicate the used direction. The *Show direction of influence* (PC) button can be used to toggle these icons on and off. This option is only valid in the dynamic mode. In the static mode the direction is always from parent to child, because the static mode relies on conditional probability tables only.

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.

6.2.7 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](#) ⁴⁵, 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.

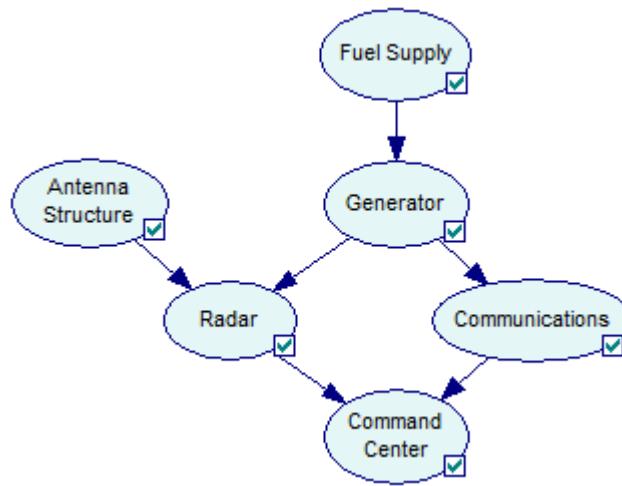
We will explain control values with the help of the following example:

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 saved as Command Center.xdsl in the *Example Networks* folder.

Once this network is loaded, you should see the following diagram in the [Graph View](#)⁶⁰.



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*.

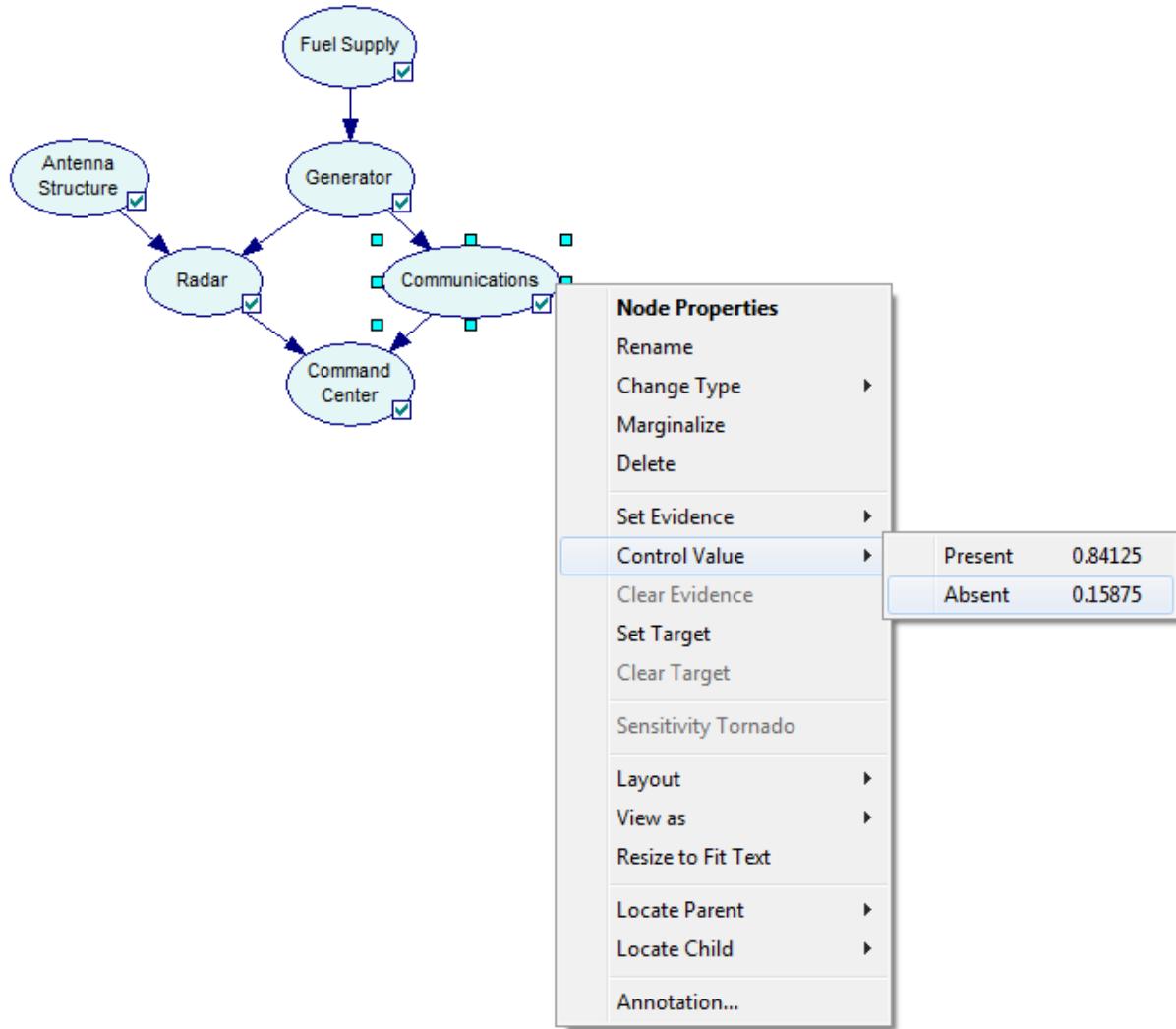
Communications will be *Absent* regardless of the values of other variables in the network (and, in particular, its parent variable, *Generator*).

Control Value is used to model such type of situations.

Let us control the value of the *Communications* node to state *Absent*.

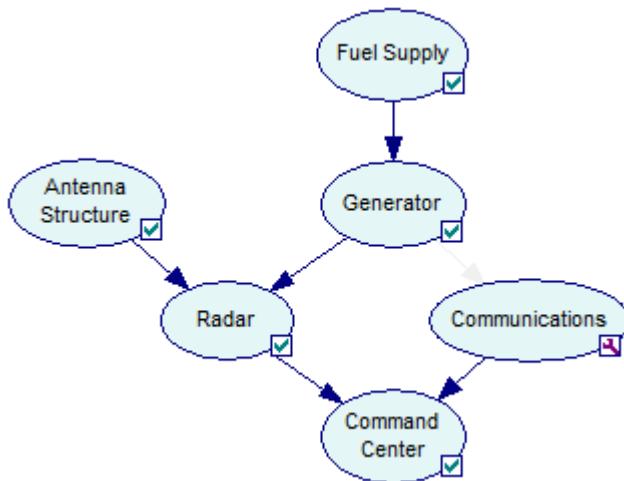
1. Right click on *Communications* node and select *Control Value* from the node's pop-up menu that appears.

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. GeNle 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.

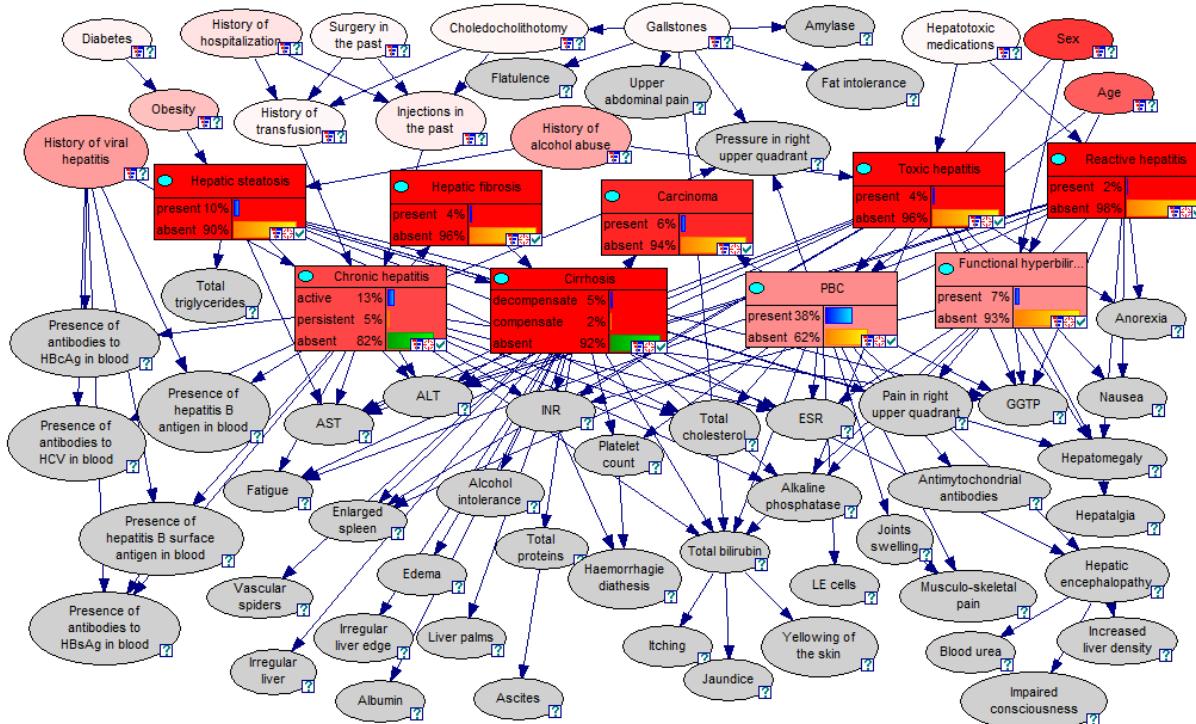
6.2.8 Sensitivity analysis in Bayesian networks

Sensitivity analysis (Castillo et al., 1997) is a technique that can help validate the probability parameters of a Bayesian network. This is done by investigating the effect of small changes in numerical parameters (i.e., 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.

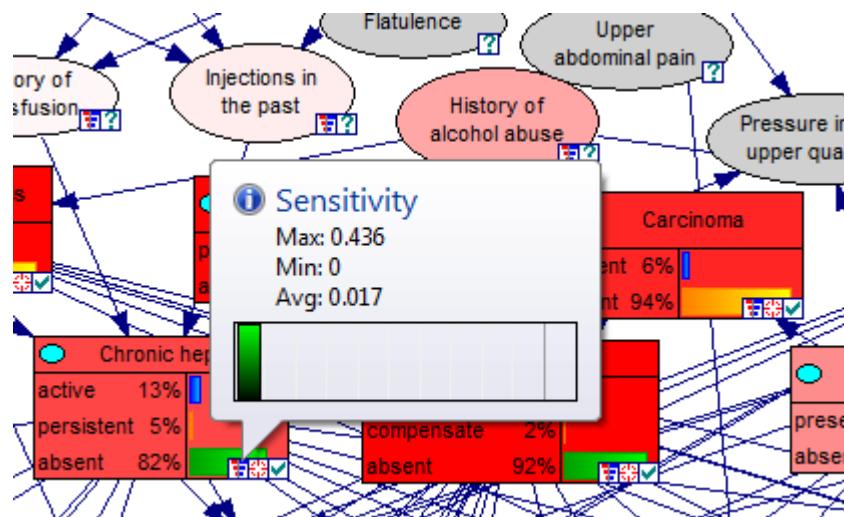
GeNIE implements an algorithm proposed by Kjaerulff and van der Gaag (2000) that performs simple sensitivity analysis in Bayesian networks. Roughly speaking, given a set of target nodes, the algorithm calculates efficiently a complete set of derivatives of the posterior probability distributions over 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 deviation in p may lead to a large difference in the posteriors of the targets. If the derivative is small, then even large deviations in the parameter make little difference in the posteriors.

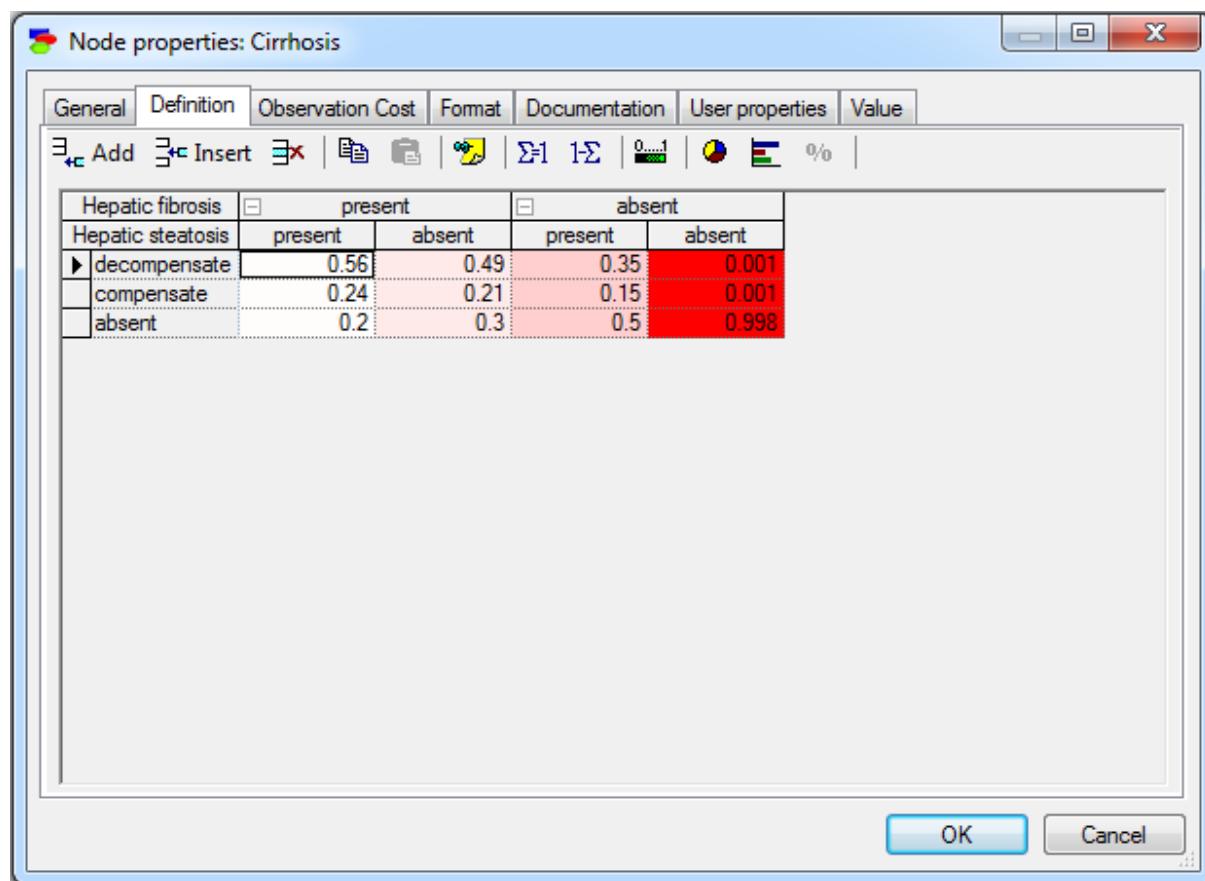
To invoke sensitivity analysis in a Bayesian network, press the *Sensitivity analysis* () tool on the [Standard Toolbar](#)¹⁷⁶. This leads to changing the coloring of the network to indicate where the sensitive parameters are located.



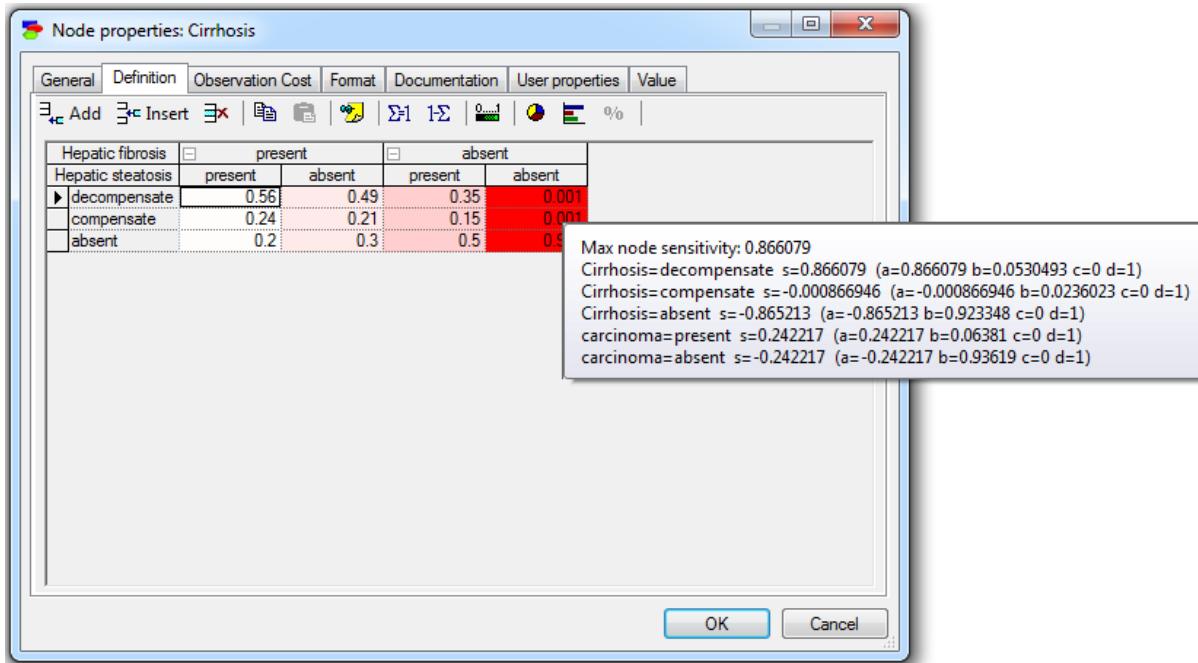
Nodes colored in red contain important parameters. Hovering over individual nodes shows summary information



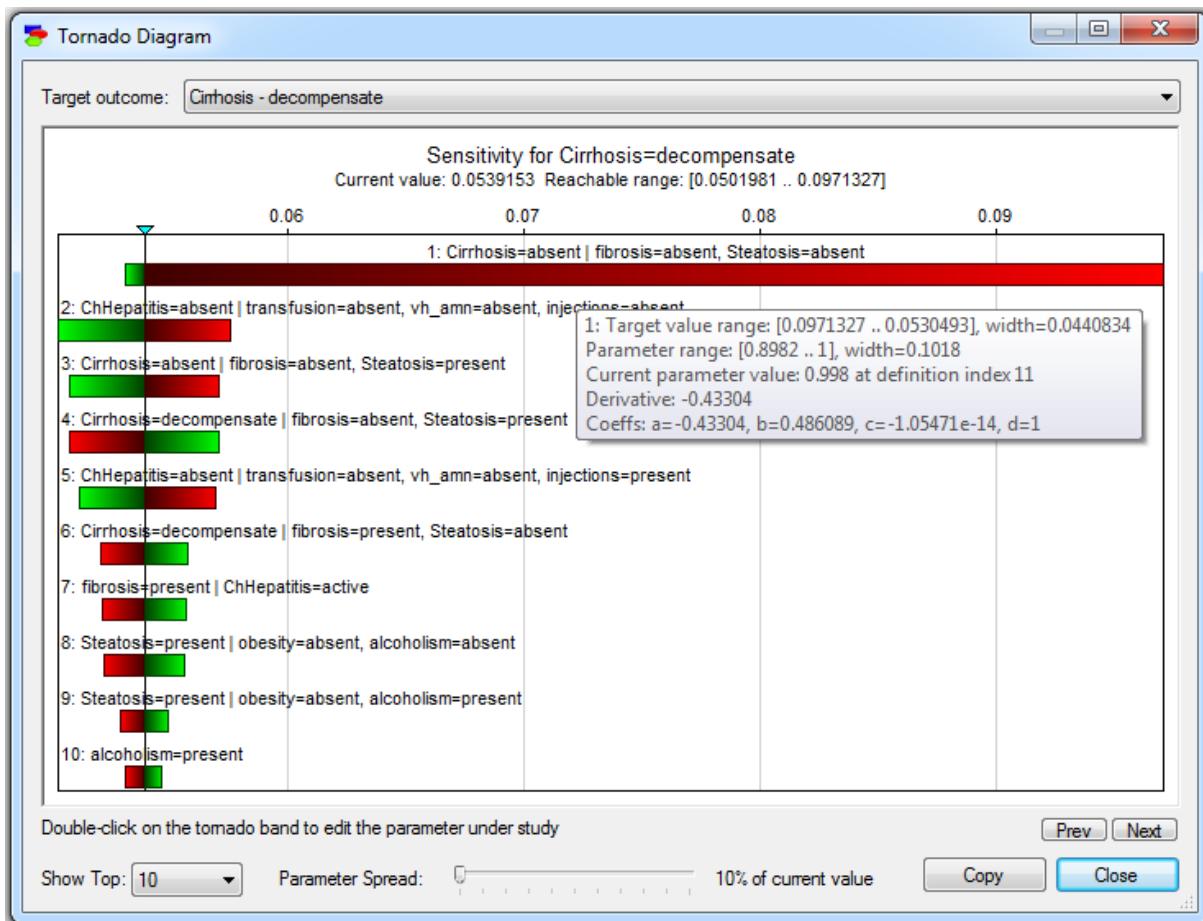
Double-clicking on the node *Cirrhosis* (one of the nodes in red) shows the following view of the definition tab



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



There is another, summary view of the most sensitive parameters. Double-clicking on the small *Tornado* (☒) icon on the node *Cirrhosis* invokes the following *Tornado Diagram* dialog



The diagram, which can be copied and later pasted into another program, shows the most sensitive parameters for a selected state of a target node (in this case, *decompensate*) sorted from the most to least sensitive. 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%). 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 over the parameter p in question. The posterior probability is represented by the following general 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.

Sensitivity analysis can be also run in influence diagrams. It is implemented in such a way that GeNle executes multiple sensitivity analyzes, 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 impossible to set 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 GeNle, is a good first step in this process.

6.3 Influence diagrams

6.3.1 Building an influence diagram

While [Bayesian networks](#)⁴⁵ allow us to quantify uncertain interactions among random variables and use this quantification to determine the impact of observations, [influence diagrams](#)⁴⁷ 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 be modeled. A model supports a decision by computing the expected value (or expected utility⁴⁴) of each decision alternative. The decision alternative with the highest expected gain is, by definition, optimal and should be chosen by the decision maker.

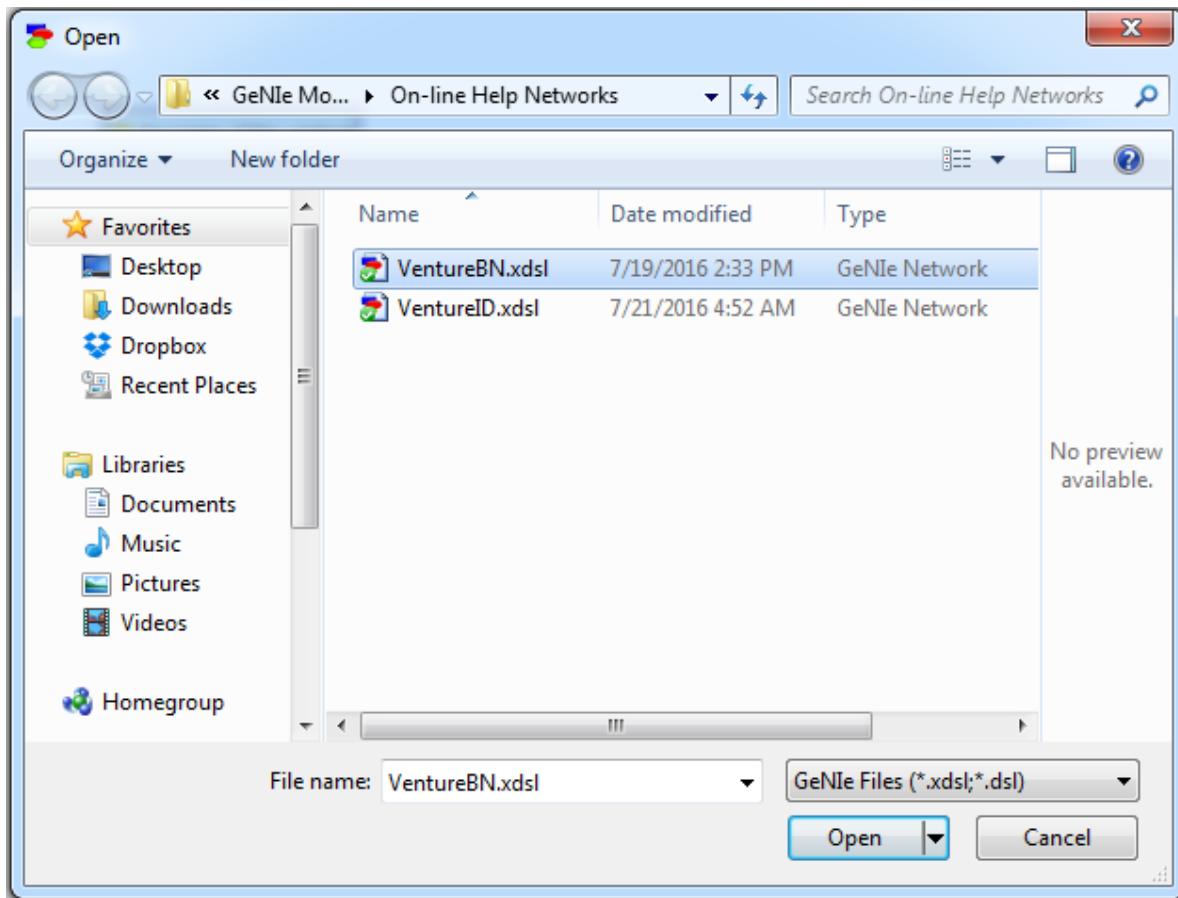
Consider the following scenario based on the example used in the [Hello GeNle!](#)¹² section. Let our venture capitalist consider investing \$5,000 in the venture. She knows that if the venture is successful, she will earn an additional \$10,000 on it. If it is not successful, she will lose her entire 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. GeNle 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 GeNle!](#)¹² 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 GeNle!](#)¹² section. You can find a copy of this Bayesian network in the *GeNle/Example Networks* folder. It is named *VentureBN.xdsl*.

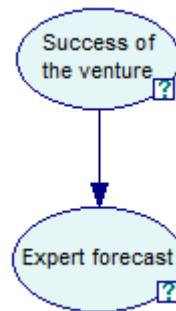
1. Click on the *Open network* () button on the [Standard Toolbar](#)¹⁷⁶.

GeNle will display the *Open* dialog as shown below:



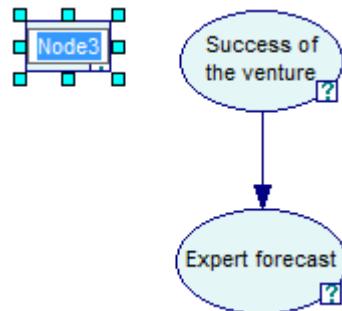
2. Click on *Example Networks* directory.
3. Select *VentureBN.xDSL* from the list and click on *Open*.

You should have the following network loaded in *Graph View*:



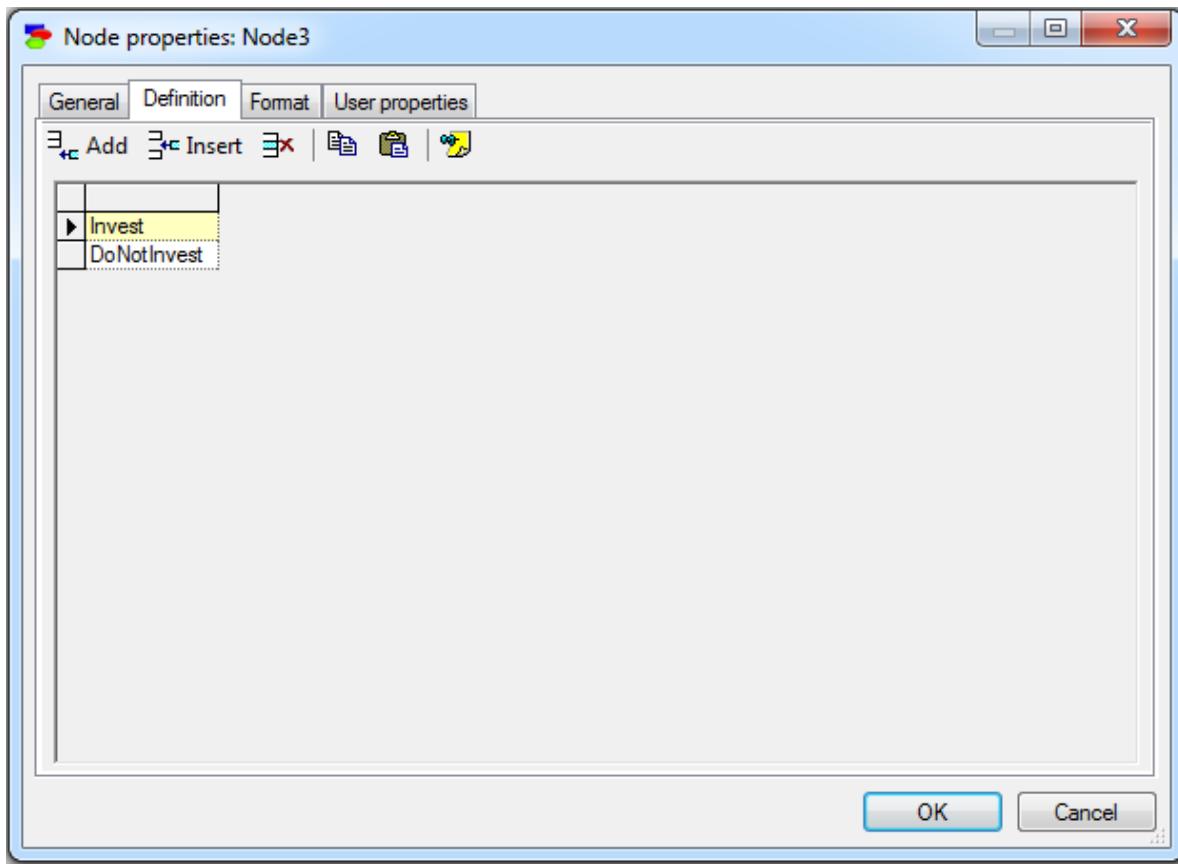
- B.** We will create a decision node and define its states. Start by selecting the *Decision* (□) tool from the *Tool menu* or the *Standard Toolbar*¹⁷⁶ and click on some empty space near the network.

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



1. Double-click on the created rectangle to open its [Node Properties](#)¹³⁸ 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:



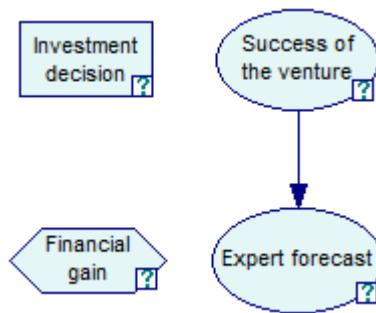
4. Click *OK* to return to *Graph View* ⁶⁰.

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 *Tool Menu* ¹⁷⁶ or the *Standard Toolbar* ¹⁷⁶ 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:

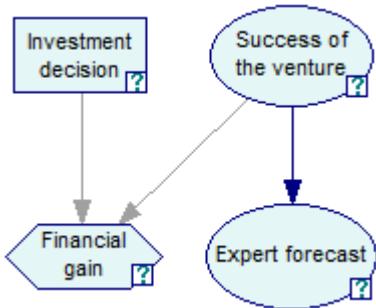


We need to tie the two new nodes (nodes *Investment decision* and *Financial gain*) with the original [Bayesian network](#)⁴⁵ (nodes *Success of the venture* and *Expert forecast*) using arcs.

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 as follows:

3. Draw an arc from node *Invest* to node *Gain* and from node *Success* to node *Gain*.

The resulting [influence diagram](#)⁴⁷ should look as follows:

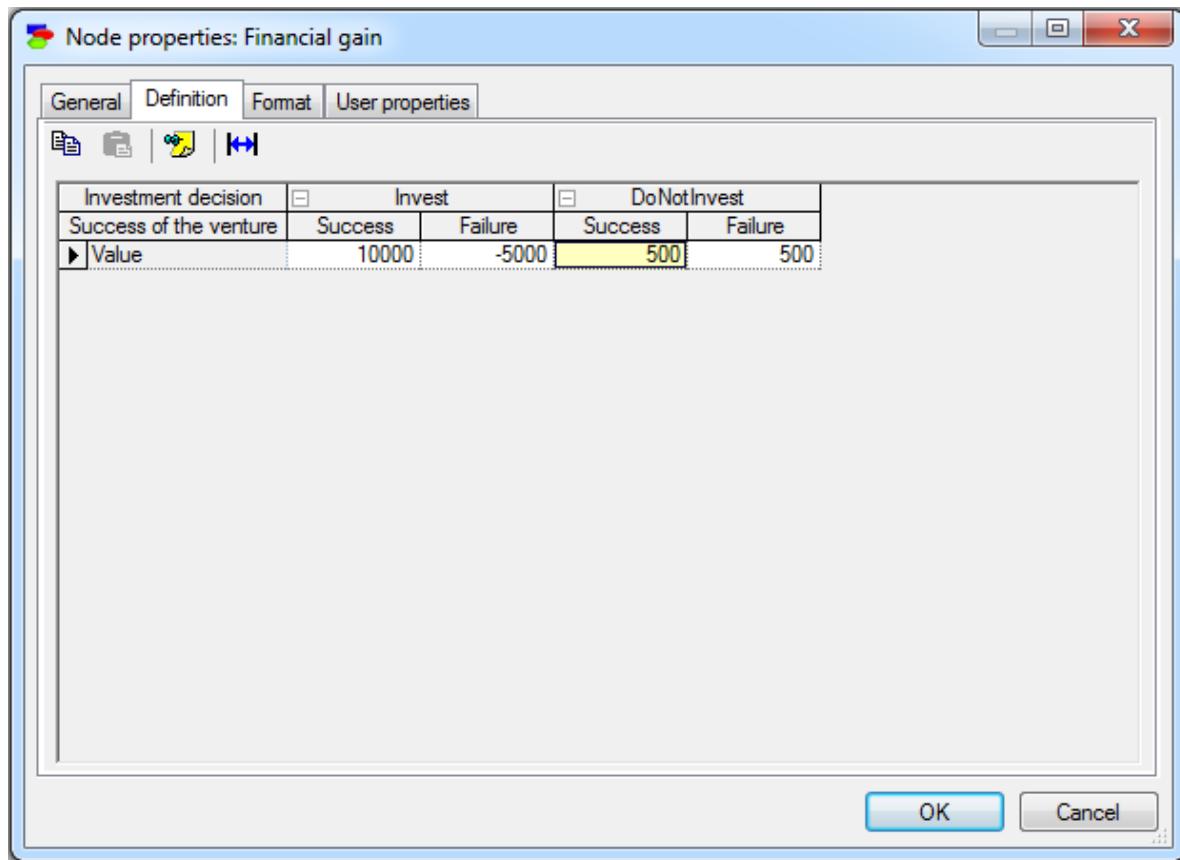


In as much as presence of an arc between two nodes denotes a direct dependence between them, absence of an arc represents independence. And so, the investment decision in this model has no impact on the success or failure of the venture. The only impact that the expert forecast has on the financial gain 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 the investment decision and the success of the venture: we have specified the monetary gain for each possible combination of values of the nodes *Investment decision* and *Success of the venture*. 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](#)⁴⁵, we will use the *Update* tool.

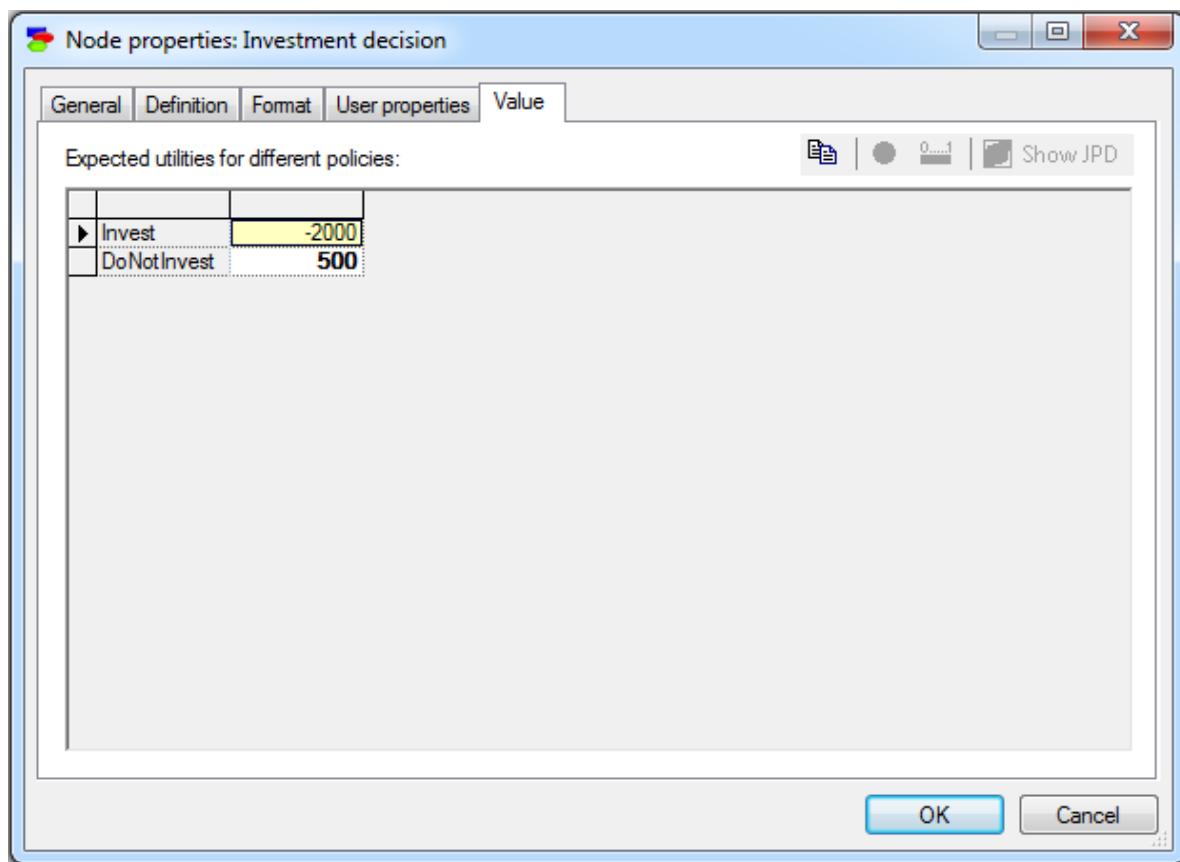
Click on the Update (⚡) tool from the [Standard Toolbar](#)¹⁷⁶.

This solves the [influence diagram](#)⁴⁷.

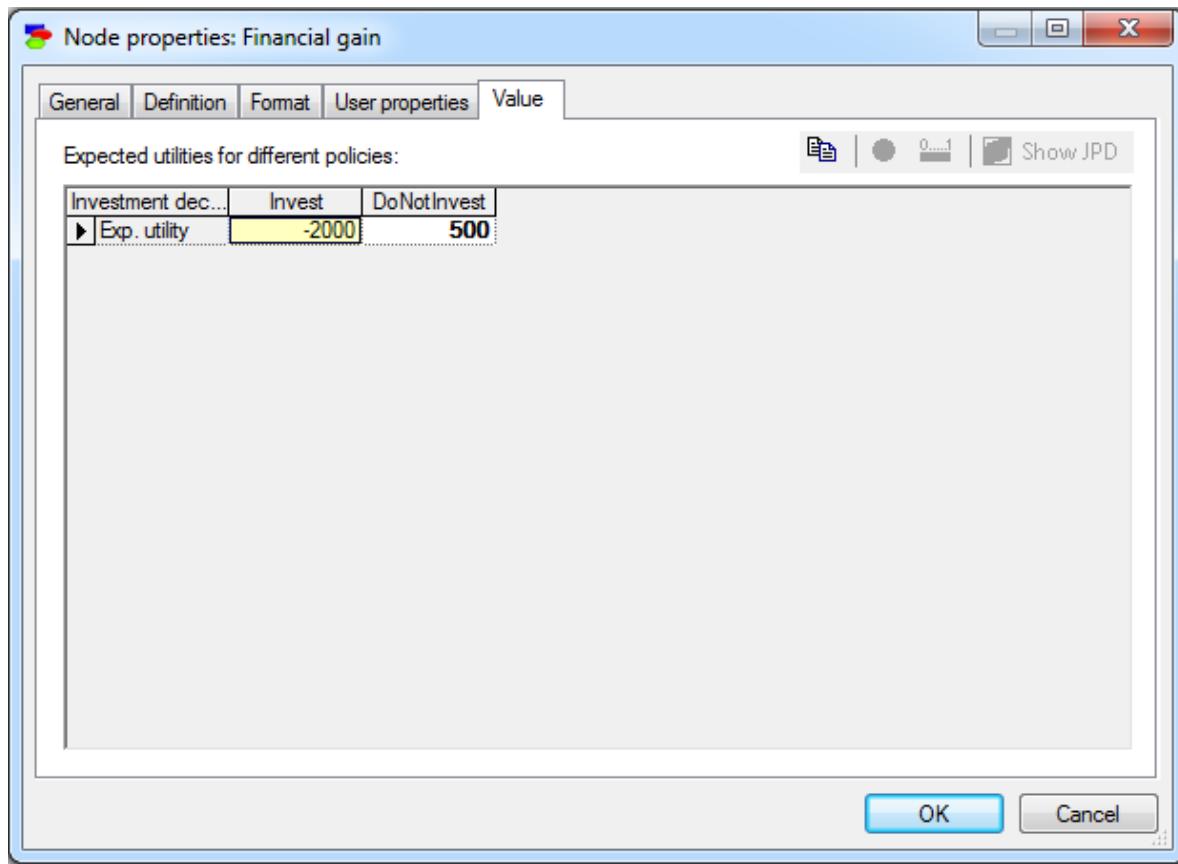
You can examine the solution by double-clicking on either the decision node (*Investment decision*) or the value node (*Financial gain*) and choosing the *Value* tab.

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](#)⁴⁷ 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.



GeNle offers two algorithms for solving influence diagrams. They are *Policy Evaluation* (default) and *Find Best Policy*. They are listed in the *Network* menu.

To choose an algorithm, click on the appropriate option in the *Network* menu. GeNle 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](#)⁴⁷ 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 rest of the choices will contain o. The algorithm is applicable only to those [influence diagrams](#)⁴⁷, 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 GeNle.

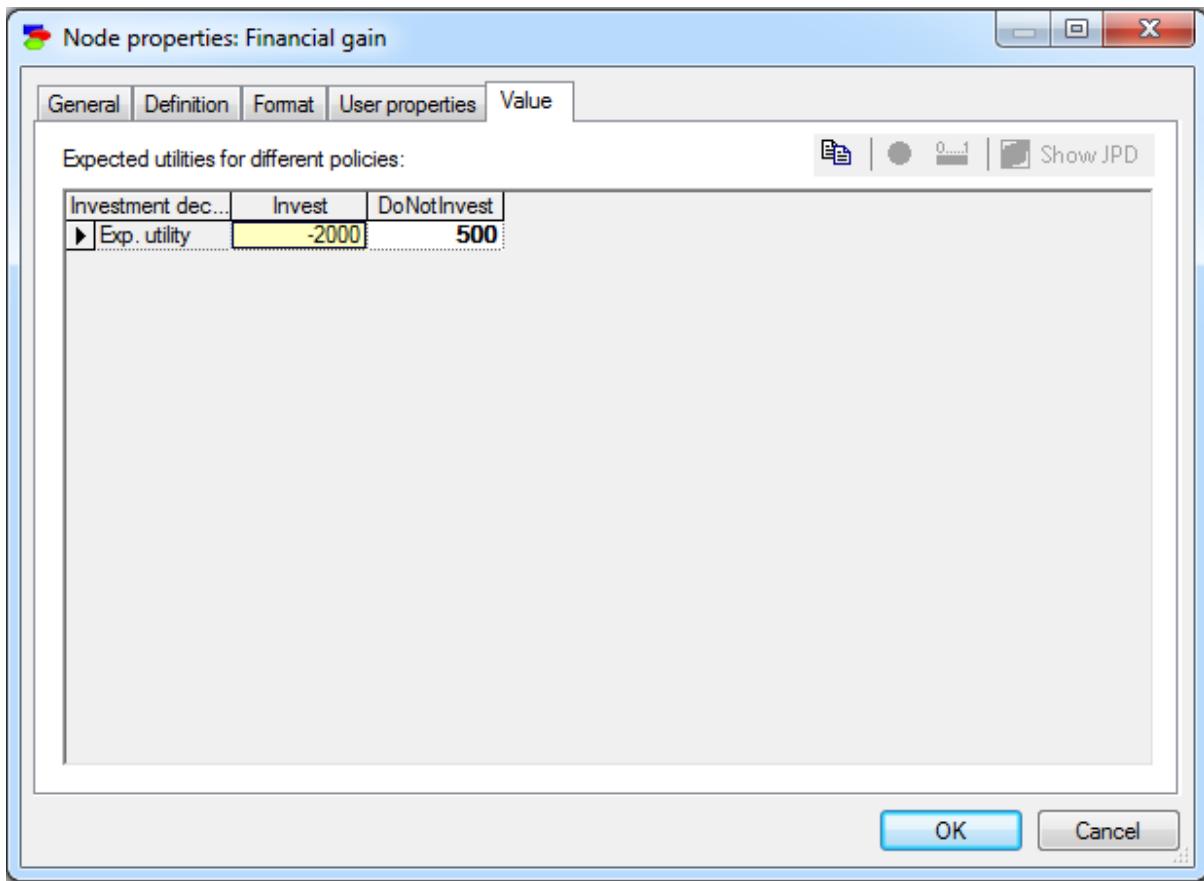
Be careful while saving the model that you have just worked on, because if you choose the *Save* option, then the *VentureBN* model will be overwritten. If you want to save the model that you have just created, choose *Save As* option from the [File Menu](#)¹⁹³.

6.3.2 Viewing results

Once a model is evaluated, you are ready to view the results computed by GeNle. 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](#)⁴⁷ 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:

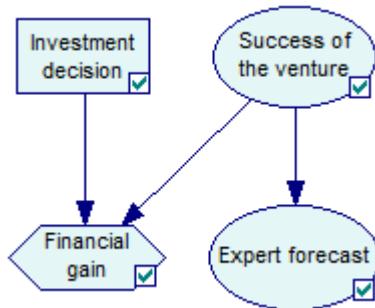


There is a difference in what you will see in terms of the result in each of these, although this difference materializes only in those influence diagrams that contain multiple decision nodes. The value node will show the expected utilities of all combinations of decision alternatives. The decision node 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

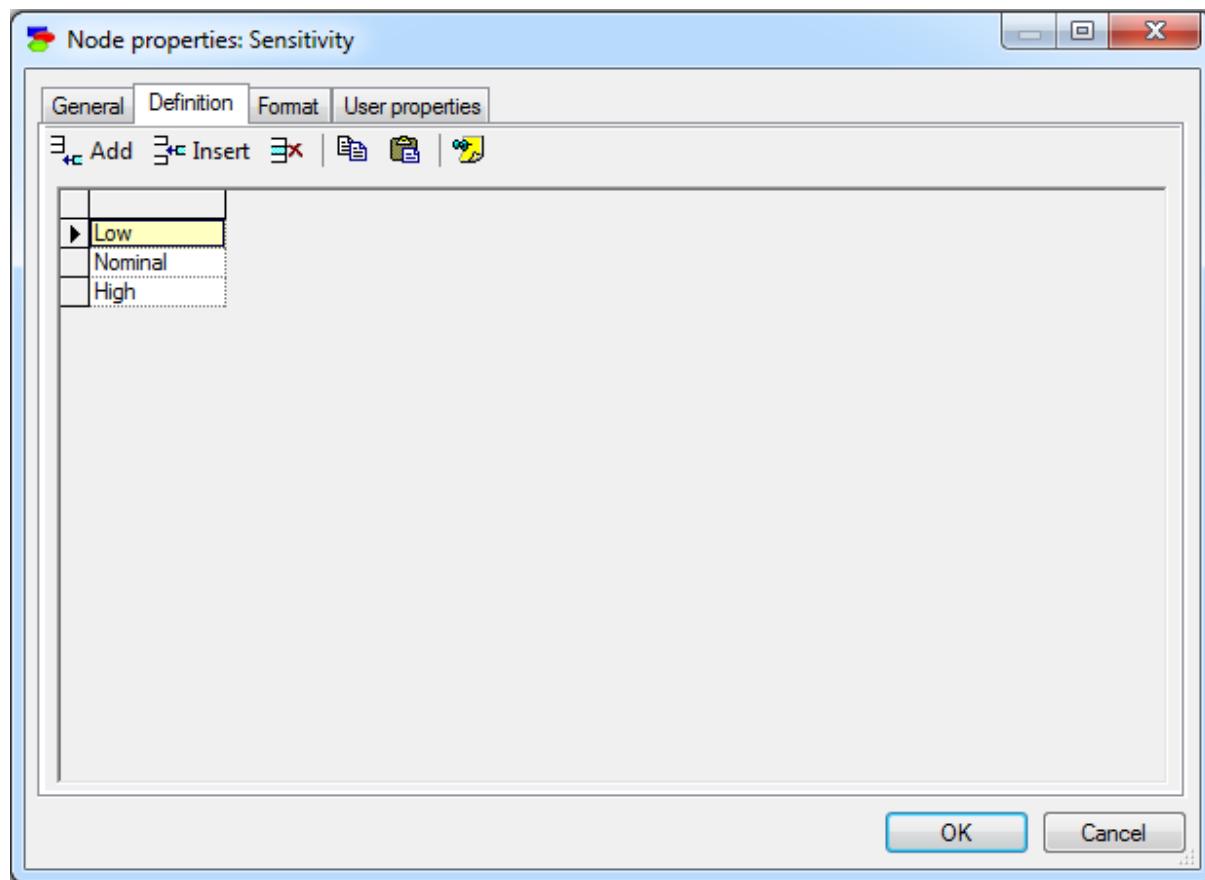
GeNle supports simple sensitivity analysis in graphical models. To perform sensitivity analysis, add an additional indexing variable that will index various values of parameters in question and have GeNle compute the impact of these values on the results. We will demonstrate this idea on the example diagram introduced in the [Building an influence diagram](#) 281 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](#)⁶⁰:

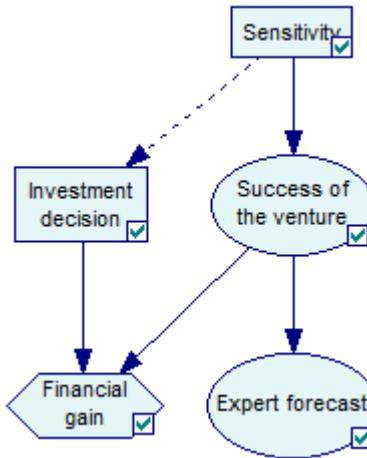


Suppose that we are uncertain as to the actual probability of the success of the venture. Believing that the nominal value of 0.2 is approximately right, we feel that it can be as low as 0.1 and as high as 0.35. To express this, we will add a *Decision* node called *Sensitivity* with three states: *Low*, *Nominal* and *High*.

Create a *Decision node* by selecting the *Decision* (¹⁷⁶) tool from the [Tool Menu](#) ¹⁷⁶ or the [Standard Toolbar](#) ¹⁷⁶ 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*.

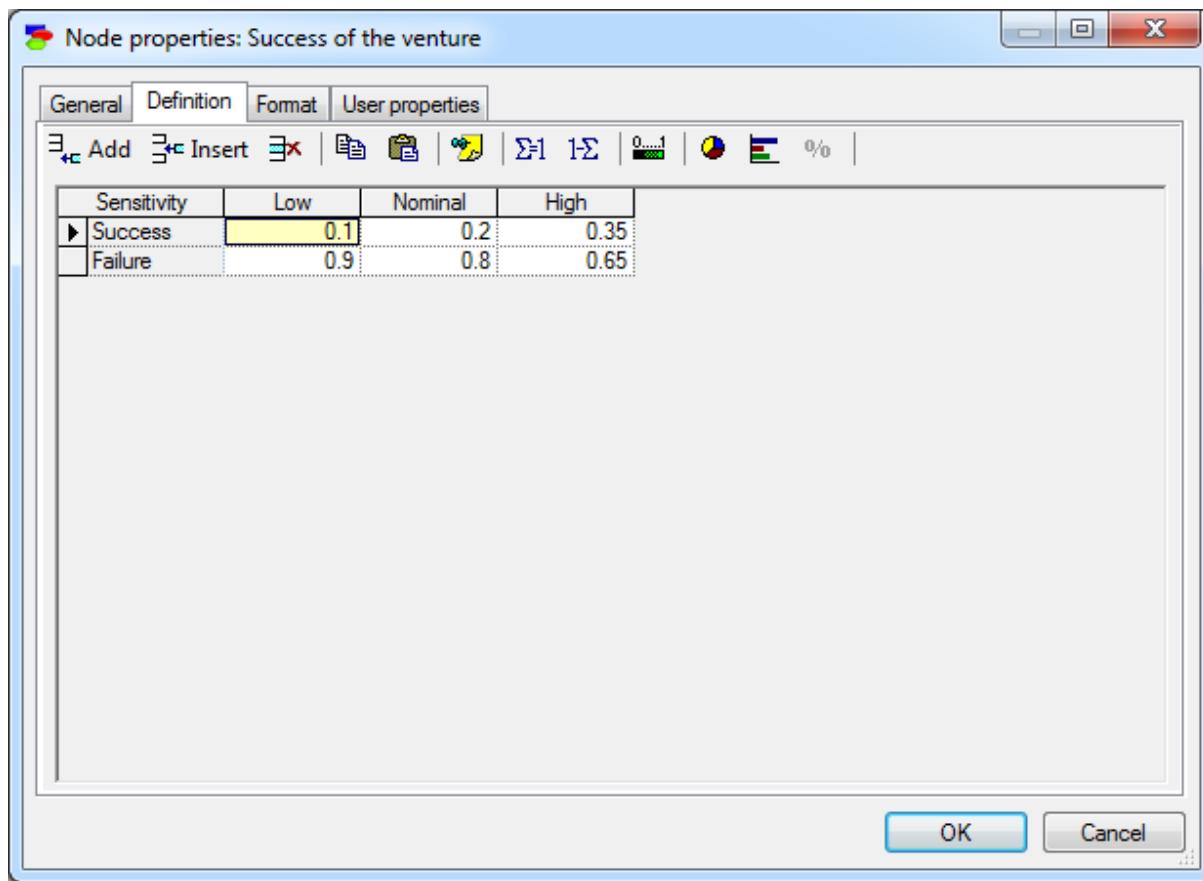


We will use this newly created decision node (*Sensitivity*) to index the *Success of the venture* node and, by this, define a range of values of probability of success. To this effect, we need to add a directed arc from *Sensitivity* to *Success of the venture*. We may add an arc from *Sensitivity* 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 will take the following form:

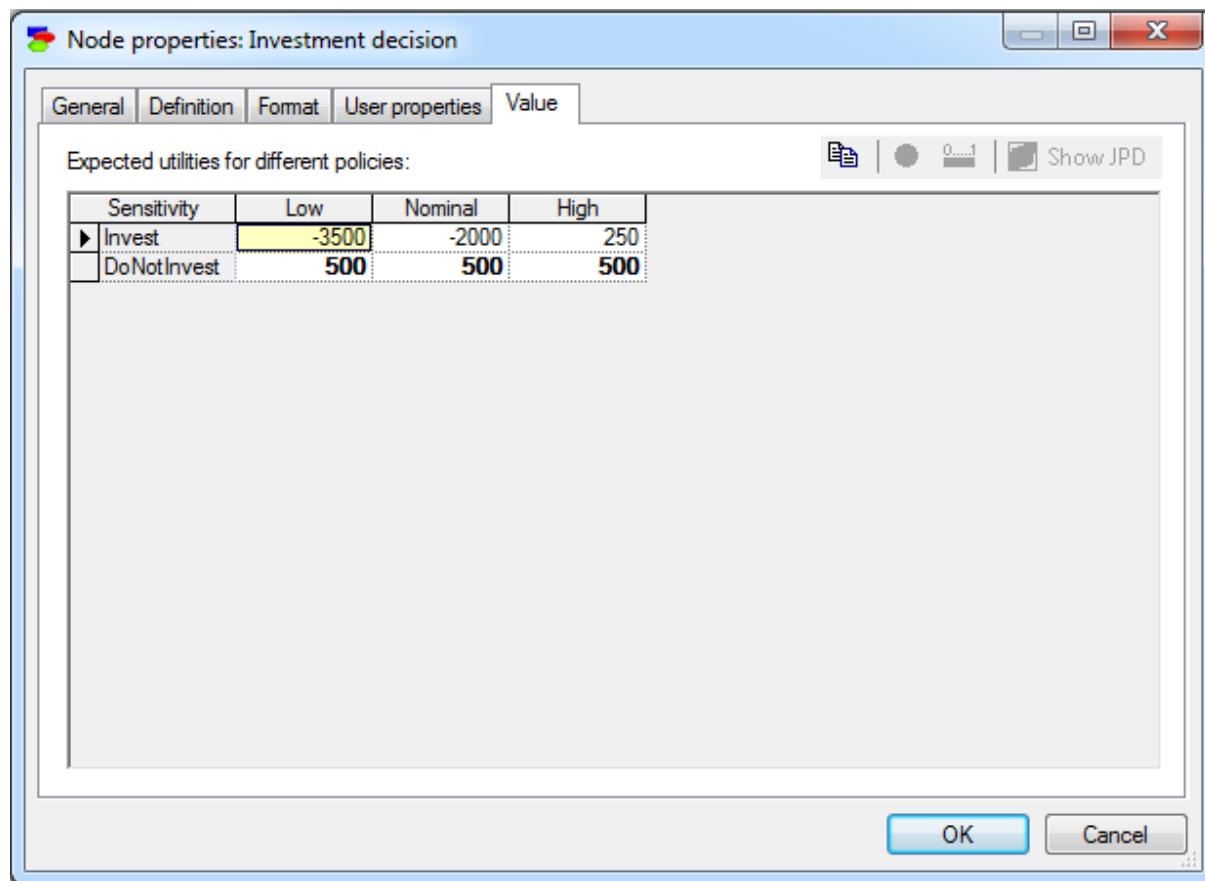


If you forget to add the arc between *Sensitivity* and *Investment decision*, do not worry, because GeNle will automatically assume the temporal order between the two and draw an arc for you between *Sensitivity* and *Investment decision*.

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:

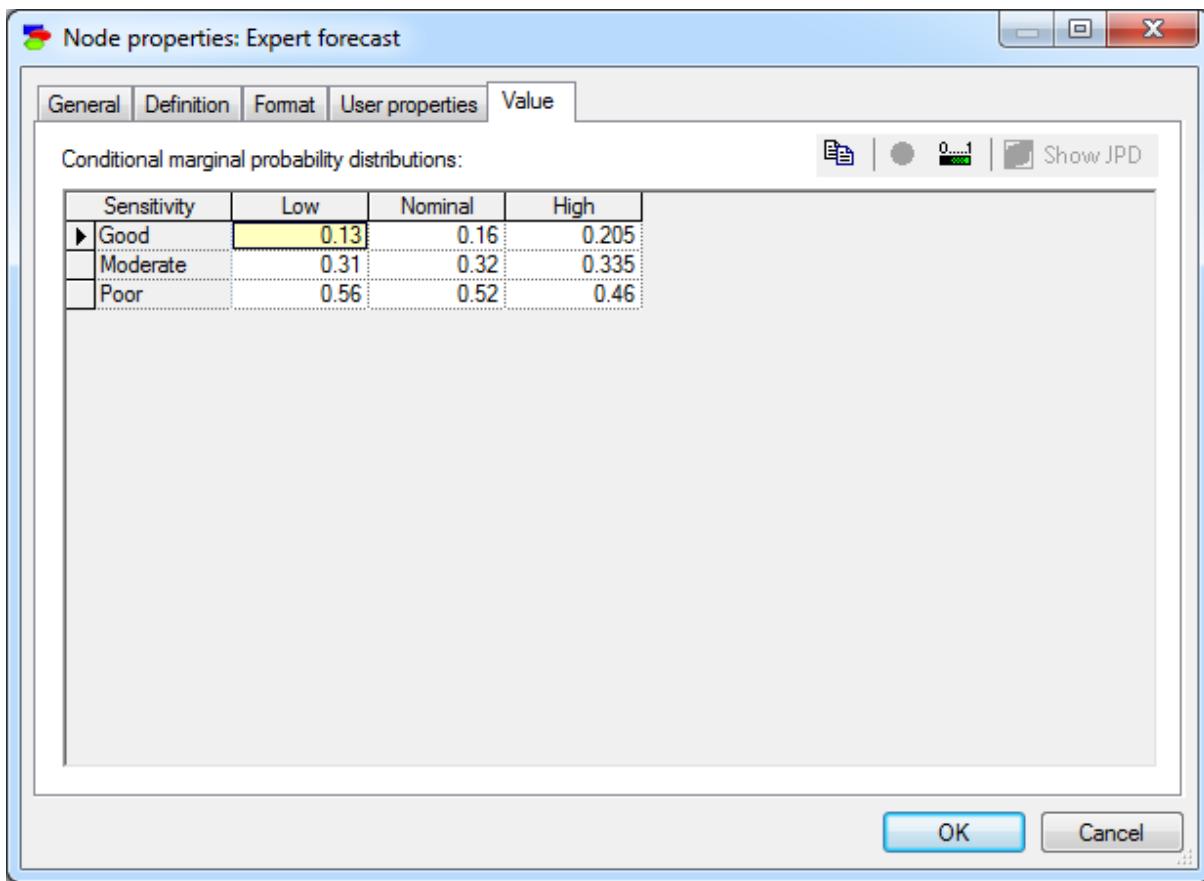


Now we are ready to update the model and observe the results. Update the model (e.g., by pressing the *Update* icon on the *Standard Toolbar*¹⁷⁶) and open the *Value* tab of *Node Properties Sheet*¹³⁸ of the *Investment decision* node. You should see the following:



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](#)²⁷⁶. 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](#)²⁷⁶ section. Captions over the tornado bars should help in identifying the individual scenario (combinations of values of the indexing nodes).

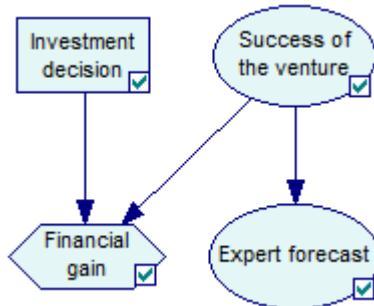
6.3.4 Value of information

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.

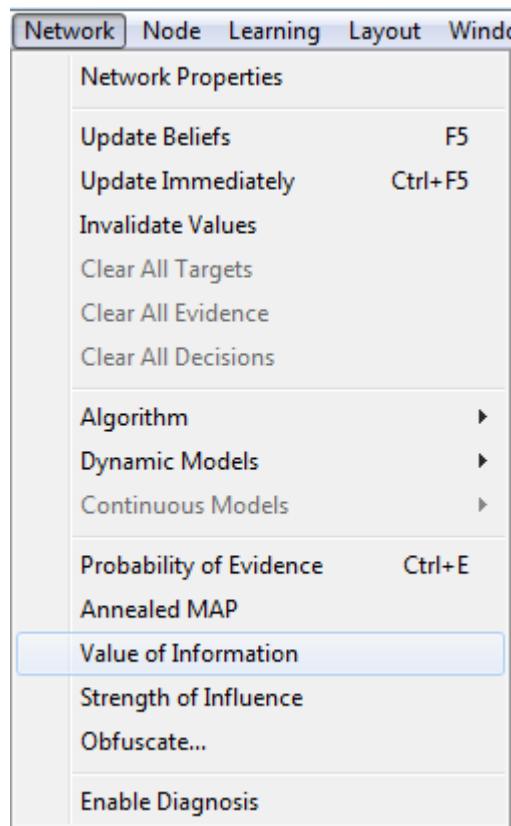
Definition of VOI and the formula for calculating it can be found in any decision-analysis textbook. An up to date list of textbooks covering the field of [decision analysis](#)^[42] can be found on [BayesFusion's web site](#). Plainly speaking, VOI for an uncertain node x is the expected difference 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 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](#)^[47] created in the [Building an influence diagram](#)^[281] section. If you do not have it saved, you can find a copy in the *Example Networks* folder (it is named *VentureID.xdsl*).

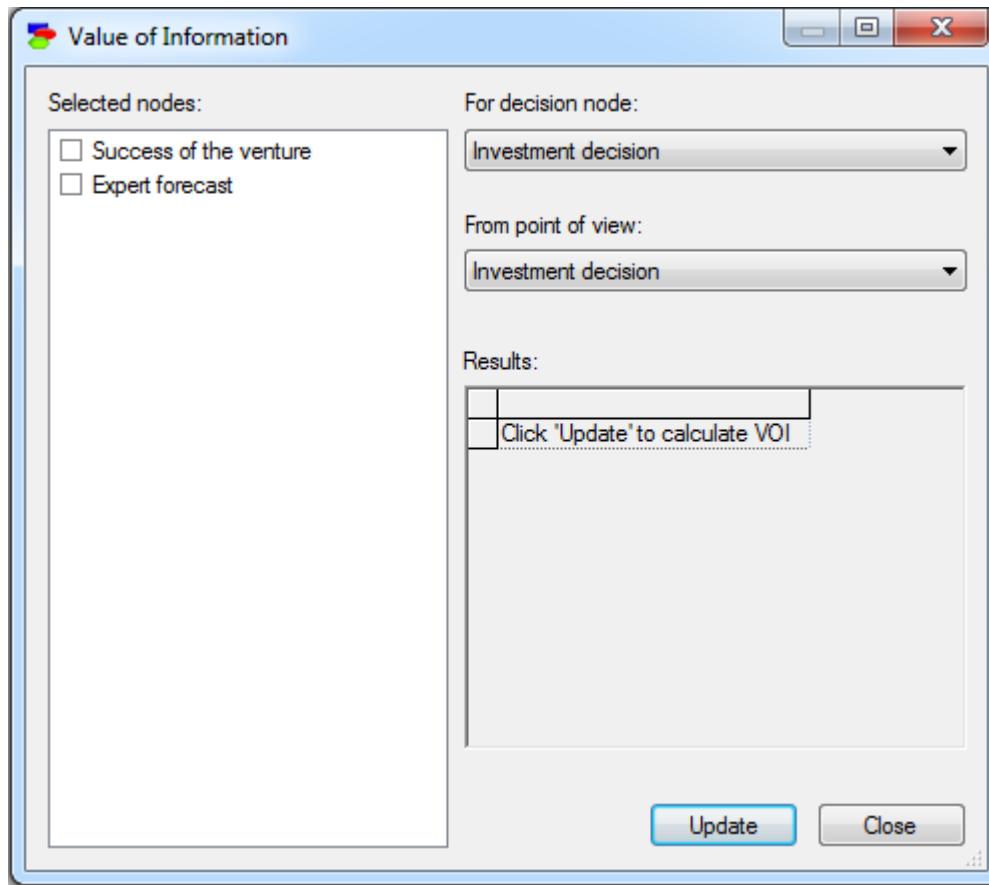
You should have the following network loaded in [Graph View](#)^[60]:



Select the *Value of Information* option from the *Network* menu as shown below:



GeNle 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.

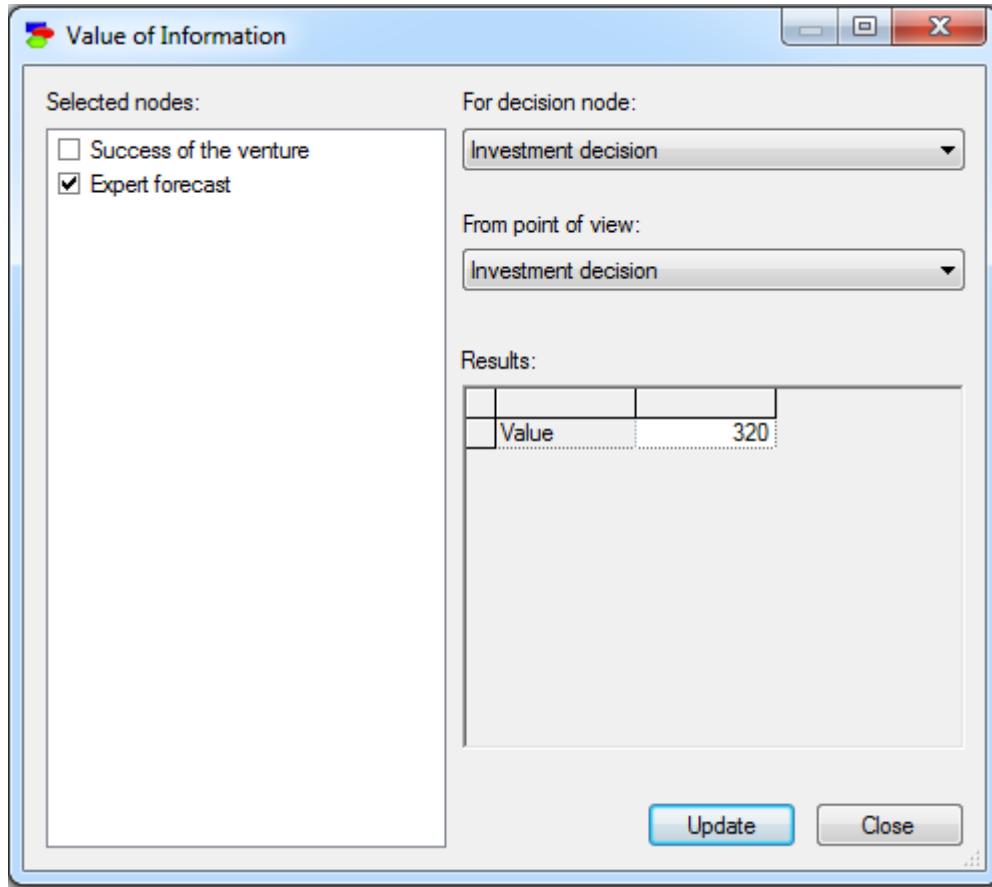
Suppose we want the value of information of the node *Expert forecast*. We proceed as follows:

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.

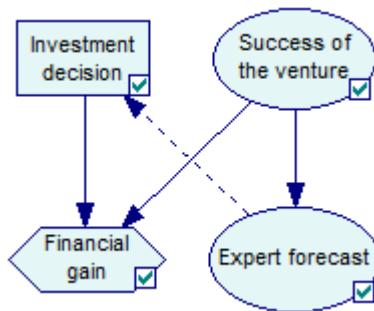
GeNle will display the calculated value of information for the node *Expert forecast*, i.e., value of observing it before making the decision *Investment decision*, in the lower-right pane, as shown below:



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

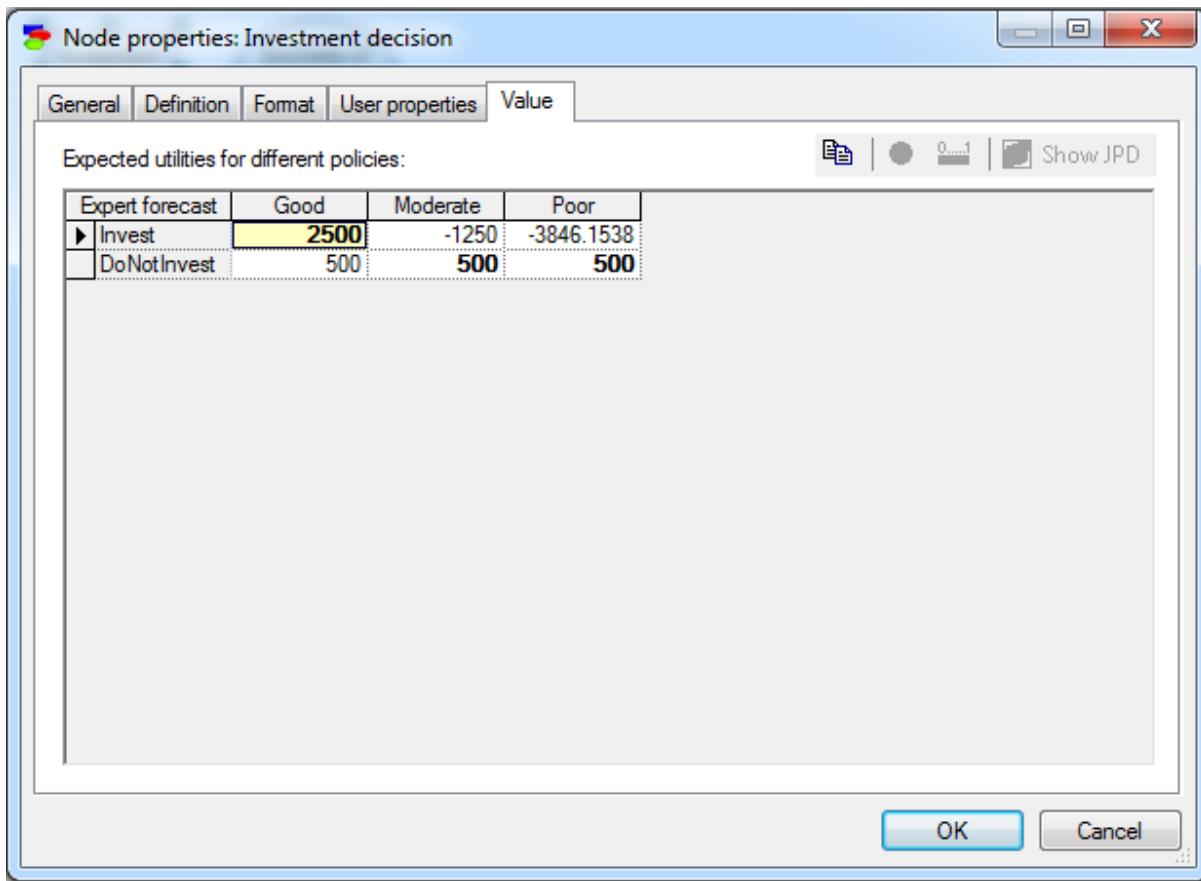
zero value would mean that learning the value of the node *Expert forecast* before making the decision has no impact on the decision. It can be proven that VOI calculation always yields a non-negative value - the intuition behind this is that we are never worse off by obtaining more information.

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](#)¹⁷⁶. After updating the values we observe the following result in the *Value* tab of the node *Investment decision*:



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 the [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](#)⁴⁵. 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](#)⁴⁵ can perform fusion of observations such as predispositions 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 system (symptoms), along with results of possible diagnostic tests. The model essentially captures how possible defects of a system (whether it is 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](#)³⁰⁸, 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](#)³¹⁷ 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.

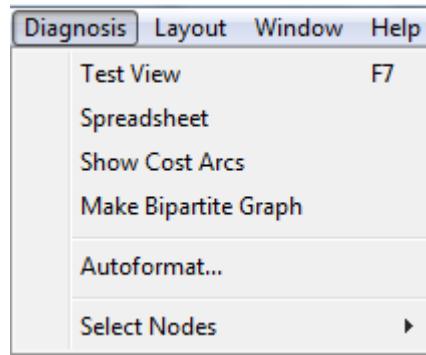
Section [Testing window](#)³²⁴ describes a special dialog window that allows the user to use a diagnostic model on real cases. The window allows for observing symptoms and signs, entering test results, and seeing GeNIE's suggestions as to what tests to perform next and what the probabilities of various faults are.

Section [Diagnostic case management](#)³³⁰ discusses how diagnostic cases can be saved to and retrieved from permanent storage (disk files).

Finally, section [Cost of observation](#)³³⁴ covers encoding and using costs of diagnosis as part of the model.

6.4.2 Diagnosis menu

When diagnostic extensions are enabled, an additional menu, called Diagnosis appears



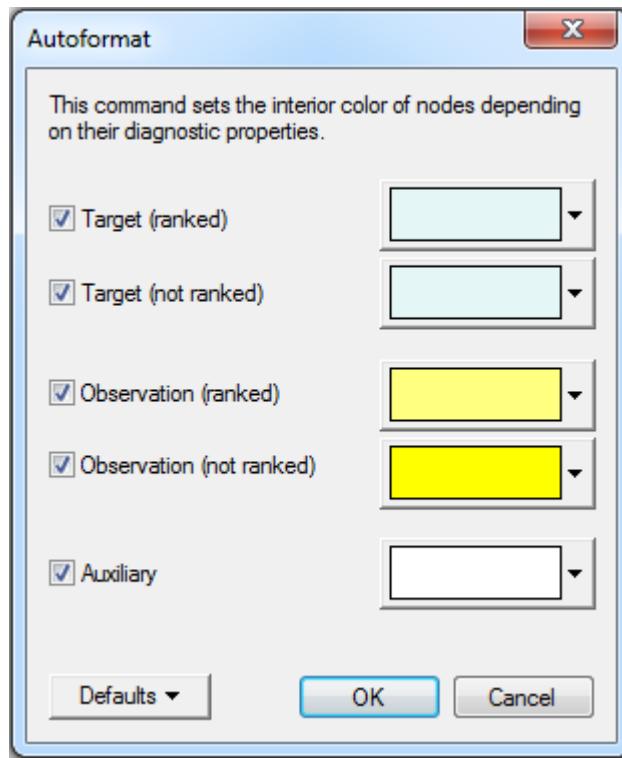
The *Diagnosis* menu offers the following commands.

Test View command (shortcut *F7*) opens the [Testing window](#)³²⁴.

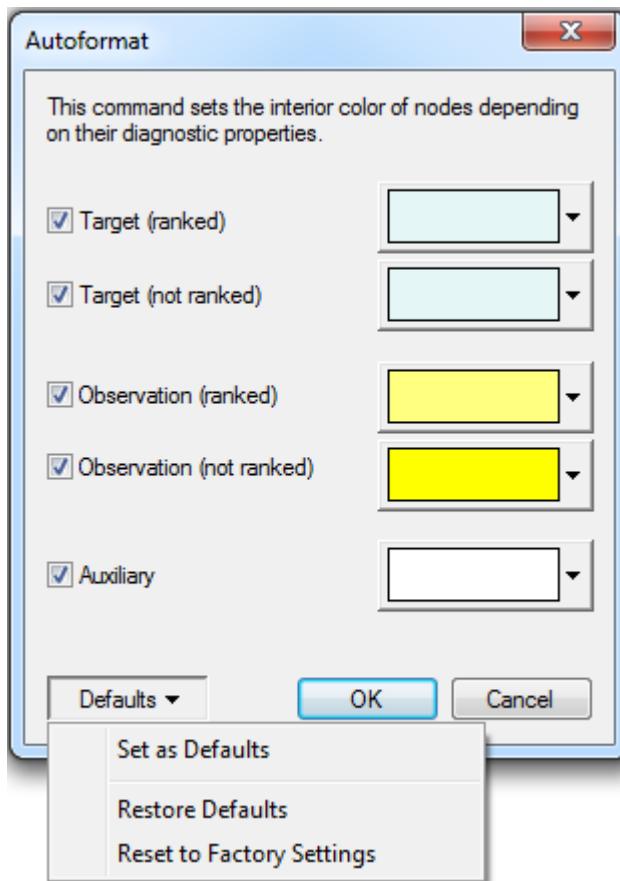
Spreadsheet command opens the [Spreadsheet View](#)³¹⁷ for the current network in a separate window.

Show Cost Arcs command toggles the *Graph View* to *Cost Graph View*. See [Cost of observation](#)³³⁴ section for more information.

The *Autoformat...* option invokes a dialog that allows to specify automatic coloring of nodes based on their diagnostic properties.



Clicking on the Defaults button displays the following pop-up menu:

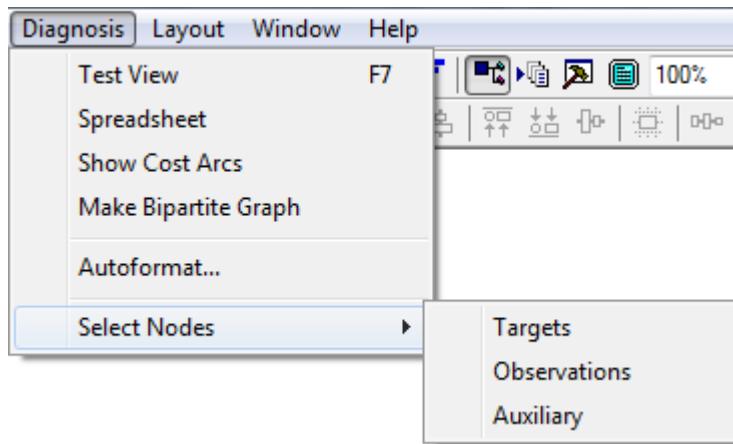


Set as Defaults sets the current selection of colors as the defaults colors for *Autoformat*.

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 GeNle.

Select Nodes submenu allows for selecting all nodes belonging to one of the three types of diagnostic nodes, *Targets*, *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 Diagnosis toolbar

The *Diagnosis* toolbar houses buttons that are used in the framework of GeNle diagnostic extensions. It can be made invisible using the toggle command *Toolbar / Diagnosis* on the *View Menu*. It can be also moved to any position within GeNle 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.

Note: The *Diagnosis Toolbar* is visible only when the *Network/Enable Diagnosis* option is checked.



Each of the commands of the *Diagnosis* toolbar is described in detail within the [Diagnosis](#)³⁰⁴ menu section.

The following buttons are available on the *Diagnosis Toolbar*.

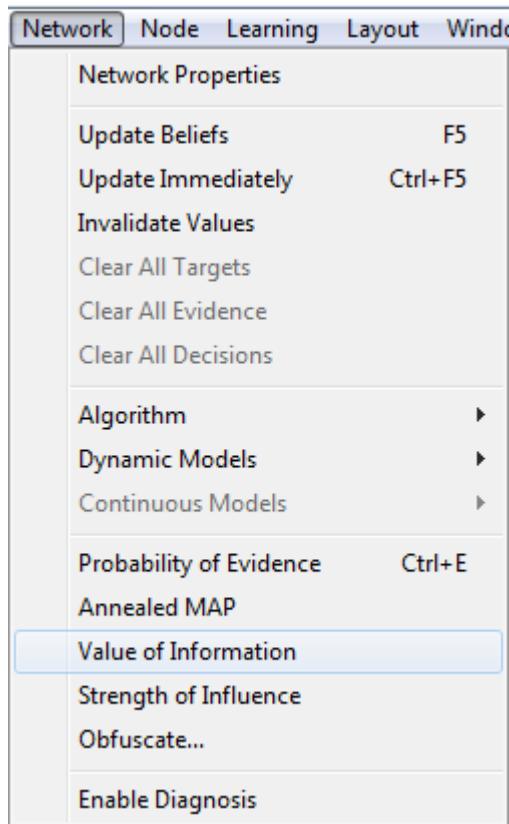
Testing Window (document icon) tool opens the [Testing Window](#)³²⁴.

Show Cost Arcs (dollar sign icon) tool toggles between the *Graph View* and *Cost Graph View*.

Spreadsheet View (spreadsheet icon) tool toggles between the *Graph View* and the *Spreadsheet View*.

6.4.4 Enabling diagnostic extensions

Diagnostic features of GeNIE are grouped into the [Diagnosis Menu](#)³⁰⁴ and [Diagnosis Toolbar](#)³⁰⁷. These features are visible only if the *Enable Diagnosis* option in *Network* menu is checked.



The user is prompted to *Enable Diagnosis* when loading a model with diagnostic information. The option is turned off by default in order to prevent unnecessary complexity of the user interface and stays on once turned on. When *Enable Diagnosis* option is on, [Diagnosis Menu](#)³⁰⁴ appears in the menu bar and [Diagnosis Toolbar](#)³⁰⁷ is added to the tool bars. Node property sheets are also changed.

Setting properties of the nodes for diagnostic applications

The critical element of diagnostic extensions is setting the roles that various model 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](#)⁴⁵ 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](#)⁴⁵ 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.

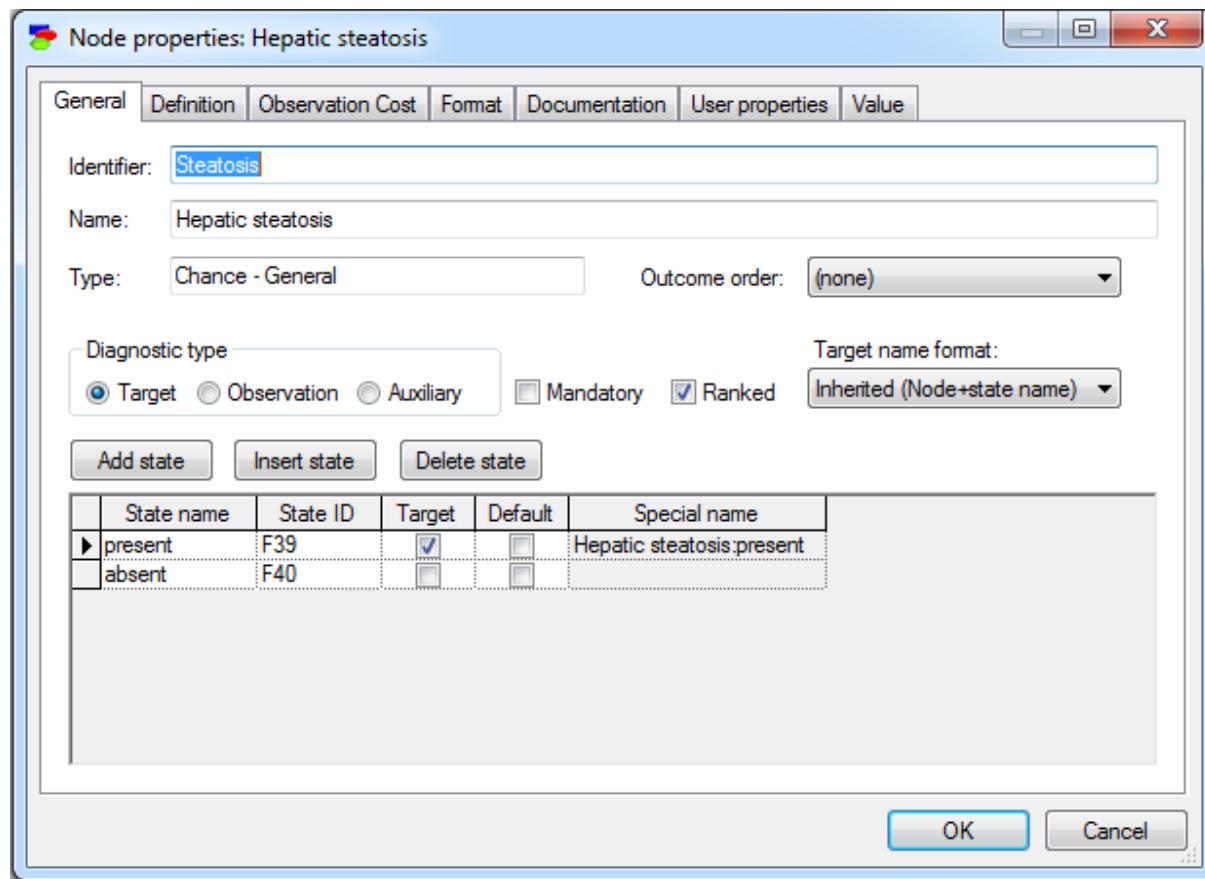
Nodes are divided into three classes: *Targets* (these are abnormalities such as machine failures or faults or diseases), *Observations* (these are observable symptoms or risk factors), and *Auxiliary* (these are additional variables, which are neither faults or observables but are useful in specifying the model). All nodes must be *Chance* type nodes for diagnosis.

The *Node properties* dialog tab can be used to enter new properties or alter current properties for a single node. Below is a list of descriptions of those tabs that are specific for diagnosis. Description of other tabs can be found in [Node Properties](#)¹³⁸ section of the reference manual.

The tabs that are specific for diagnosis are *General*, *Observation Cost*, and *Documentation*. Each of the tabs has three buttons on the bottom, *OK*, *Cancel*, and *Help*. After making changes to any of the tabs, click on *OK* to save the changes. To continue on with the model without saving your changes, simply click on *Cancel*. Please note that some changes cannot be undone, in which case the *Cancel* button is grayed out.

General tab

Shown below is the snapshot of the *General* tab of the *Node properties* sheet when the diagnostic extensions are enabled.



We can see additional properties specified on the *General* tab. The rows of the table represent different states of the variable. To rename a state, double-click on its name and edit the text. The name of a state should follow the syntax of identifiers in programming languages - it should start with a letter followed by letters, digits, and underscore characters. The reason is again due to various possible ways that a node state can be referred to.

In the diagnosis mode, every node has to be assigned a *Diagnostic type*, which must be chosen from among: *Target*, *Observation*, and *Auxiliary*.

Target nodes represent faults, hardware defects, disorders, etc. At least one state of a *Target* node must be designed as a *Target* state. A target state is the defective state, so all states that are not designated to be *Targets* are normal states by default.

Observation nodes represent 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: *Ranked* and *Mandatory*. Out of the six combinations of *Diagnostic types* and the two subtypes, only four are legal: *Target Ranked*, *Observation*, *Observation Ranked* and *Observation Mandatory*.

Target Ranked indicates that the target state of a node is to appear on the list of ranked targets in the diagnostic [Testing Window](#)³²⁴. *Target Ranked* components are ranked in terms of their probability. Each of the *Target* states of a *Target Ranked* fault will be listed along with its probability.

Observation nodes have several combinations of subtypes possible. However, only three will be discussed because they are most relevant to modeling using GeNle.

Observation can be set to a default state, which is assumed to be the initial state of the observation, in which case the *Observation* is not ranked.

Observation Ranked nodes are the most common observation nodes. They are ranked by the diagnostic [Testing Window](#)³²⁴ 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 [Testing Window](#)³²⁴.

Observation Mandatory represents information that needs to be provided before troubleshooting is to begin. *Observation Mandatory* may represent an action, a testing condition, or any other factor that needs to be performed or observed before the troubleshooting begins.

The user also has the option of selecting what state the node is in. For a target node, the target state can be selected. For a default node, the default state can be selected. The user must also specify the *Special Name* or *Format* that they have chosen for the node. All of the above can also be edited in the *Spreadsheet window*.

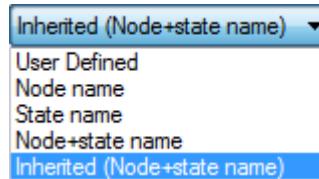
We need to set the distribution for the nodes appropriately. We can change the distribution by using the *Change Type* option from the [Node Menu](#)²⁰⁷.

All the above conditions are represented in a table below:

Diagnostic Type	Typical use	Ranked/Mandatory status
<i>Target</i>	To represent faulty and defective components	<i>Ranked</i>
<i>Observation</i>	Error messages, symptoms or tests	<i>None / Ranked / Mandatory</i>
<i>Auxiliary</i>	No specific use, only for convenience	<i>None</i>

State ID field allows to define an additional state ID, which can be referred to in embedded diagnostic applications. GeNIE and SMILE provide only means for editing this field.

Target name format (grayed out in the picture below, because *anorexia* is an *Observation* and not a *Target* node) specifies how diagnostic target nodes are referred to in the diagnostic [Testing window](#)³²⁴. There are five choices here: *User Defined*, *Node name*, *State name*, *Node+state name*, and *Inherited (Node+state name)*. The last option (chosen for the node *anorexia*) is inherited from the [Network properties](#)¹²³.



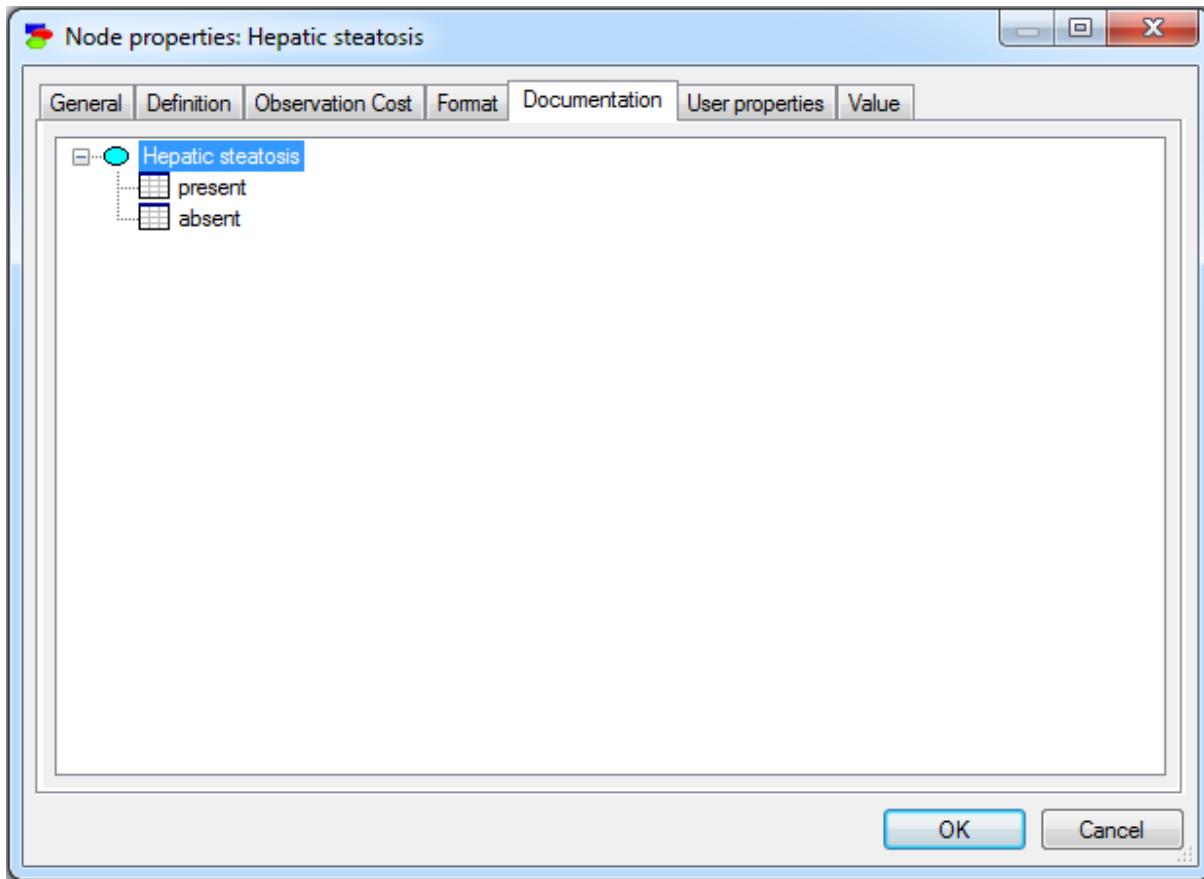
Special name contains a reference to a *Target* state of a *Target* node. Unless the *Target name format* is *User Defined*, this row is pre-filled by the program based on the choice in the *Target name format*.

It is possible to edit states of the node in the *General* tab when diagnostic extensions are enabled. Buttons *Add state*, *Insert state*, and *Delete state* work similarly to those on the *Definition* tab and allow to add a state after the selected state, add a state before the selected state, and delete the selected state, respectively.

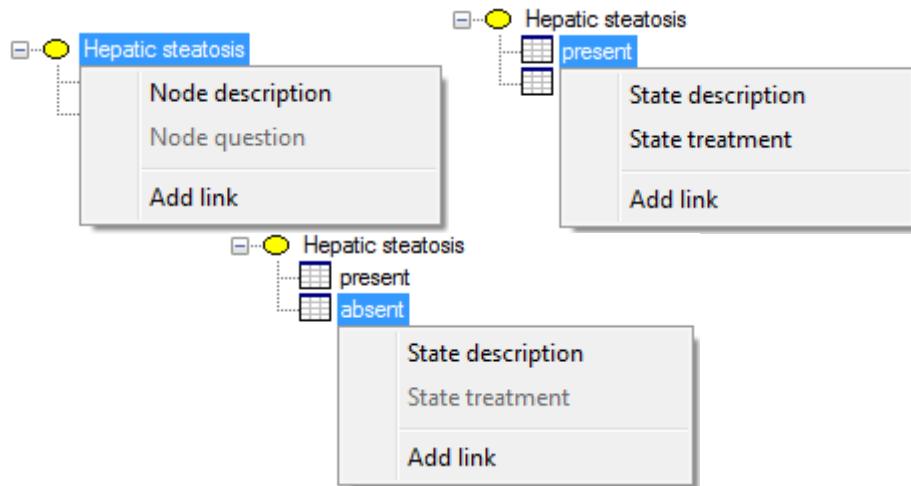
Note : Both state names and State IDs must be unique within the same node.

Documentation tab

The *Documentation* tab, an example of which is shown below, is used to enter information such as *Description*, *Fix*, *Question*, and *Link* for node and state documentation. Definitions of these terms are given in the [Spreadsheet View](#) 317 section.



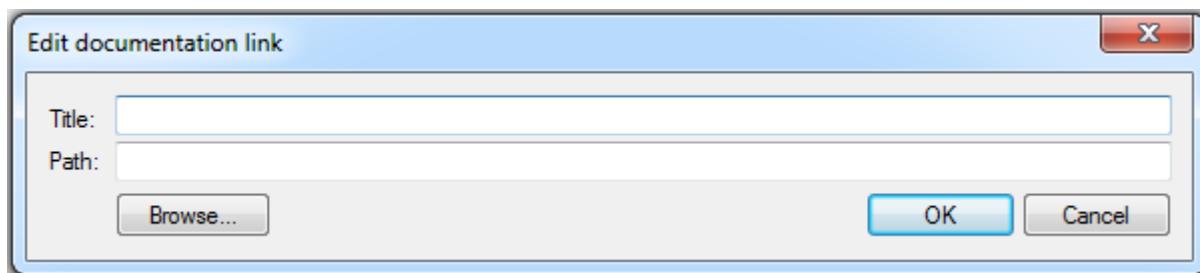
By right clicking on the node or state names in the *Documentation* tab, we will see a somewhat different menu for entering documentation information.



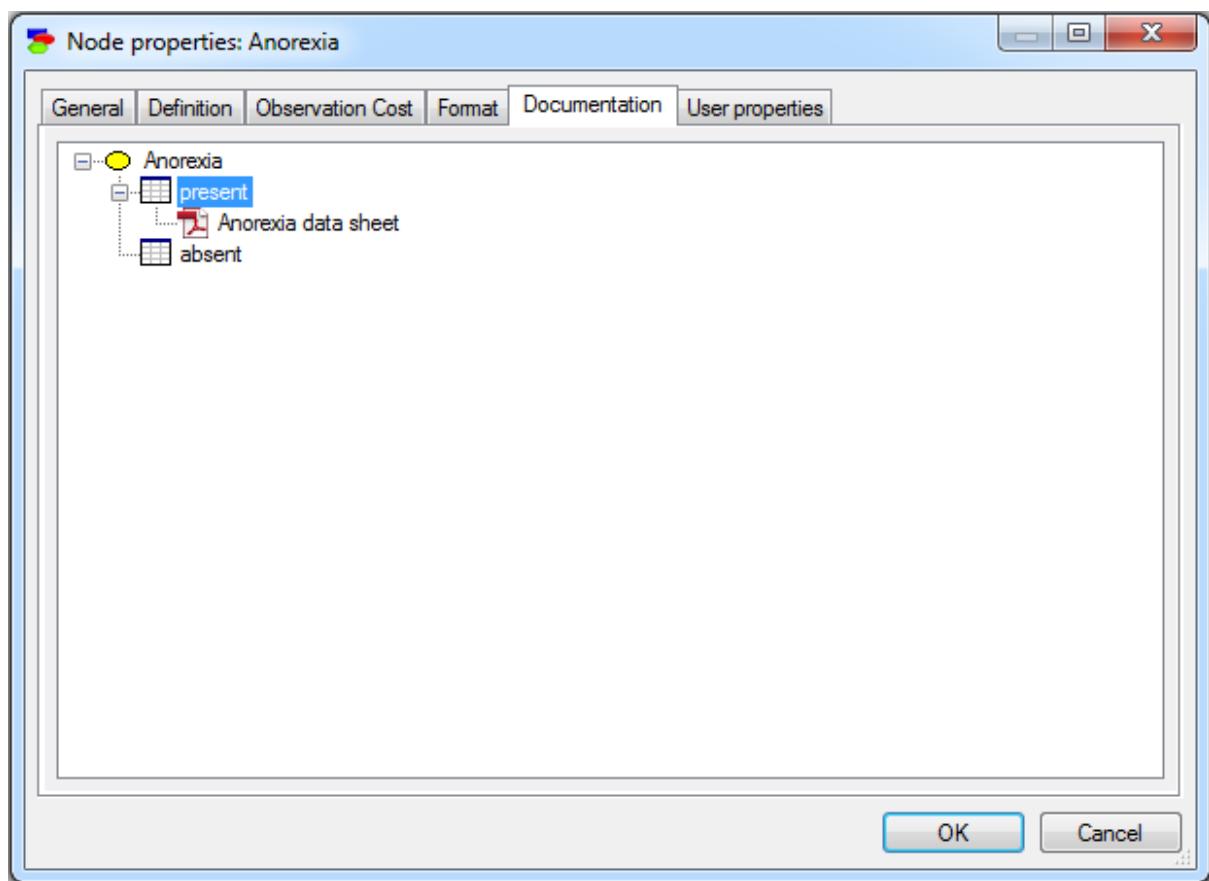
The *Node question* option is enabled only for *Observation* type nodes. *State treatment* is only enabled for *Target* states of *Target* nodes.

The user can select the type of information they would like to add or modify. The documentation-editing window will appear for *Description*, *Fix*, or *Question*, and then the user can enter their text into the window. For the links, a *Links* editing window will appear. In the box labeled *Title*, the user must enter the name of the link. Then click on the button next to *Link* to browse through the computer contents for the link file in a conventional window. To view the file for an existing link, simply click on the *Link* icon in the *Documentation* tab.

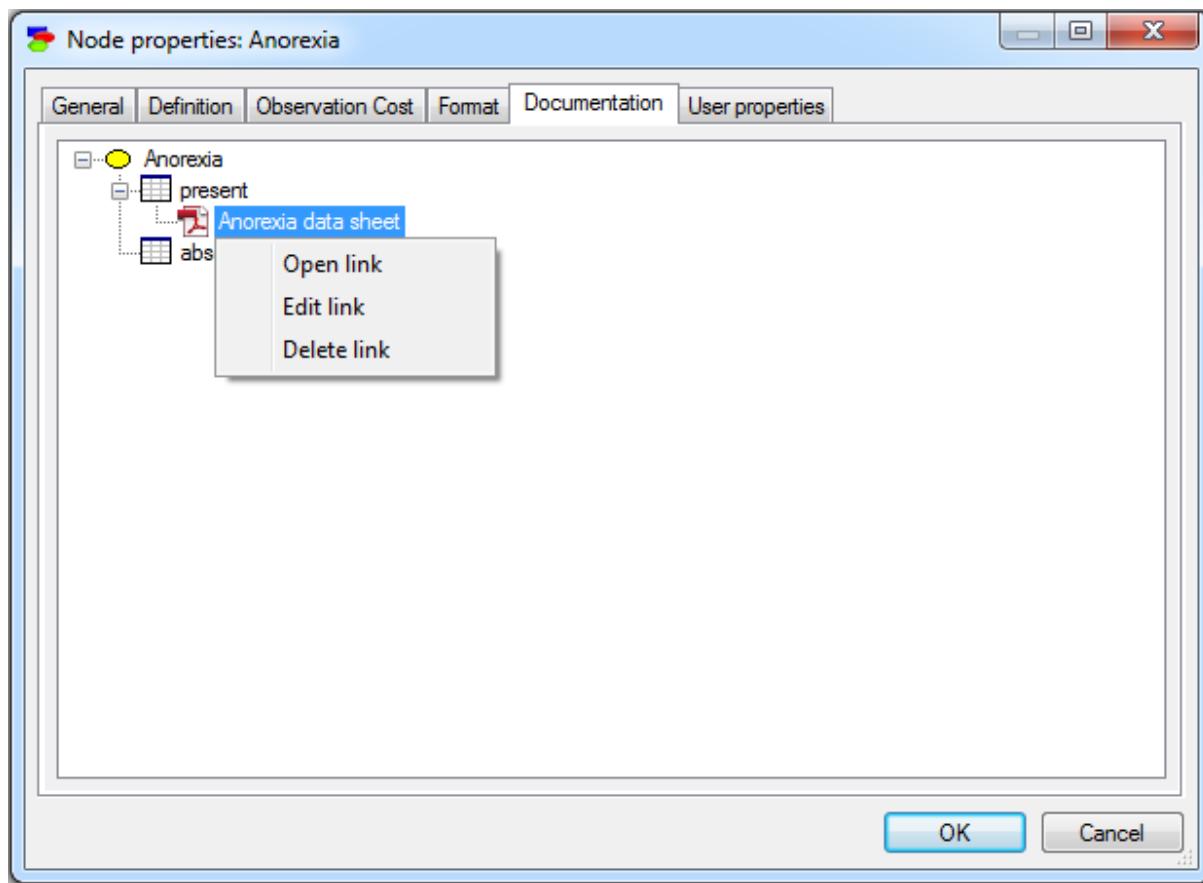
Add link in all variants of the pop-up menu opens an *Edit documentation link dialog*:



The dialog allows to add links to files on the local computer. Typically, these files will contain documentation for the state. A link added will be displayed in the *Documentation* window under the node or the state to which it was added.



You can right-click on the link name to open, edit, or delete the link.

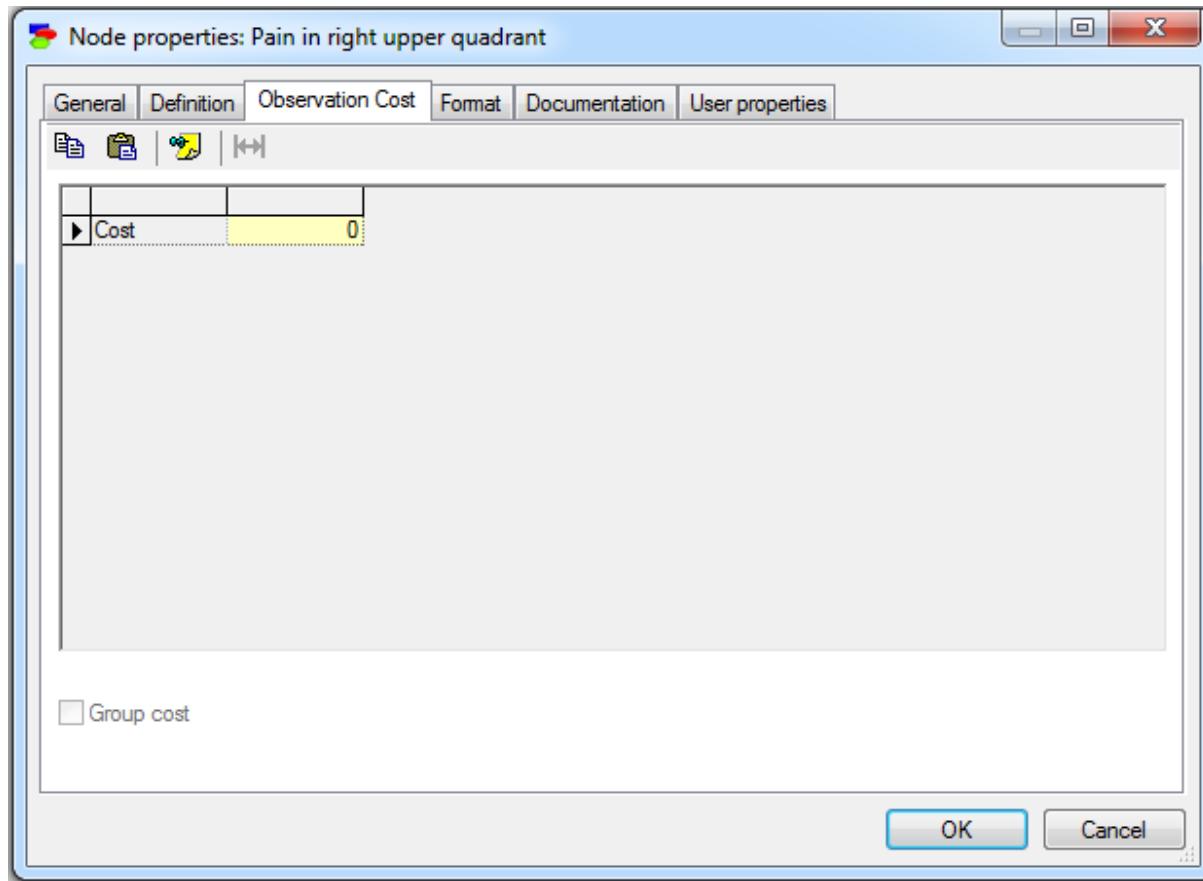


Open link opens the selected file or link. If it is an URL or a HTML file on disk, it will be displayed in a browser window. For other file types, it will try to open the default program processing that file type.

Note: If the file does not open as expected, then you probably do not have the program installed to open that type of file in Windows. Please install the respective program and try again.

Cost of observation

The *Observation Cost* tab 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., 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](#)³¹⁷.



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 costs close to nothing to determine the 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 [Testing Window](#)³²⁴ 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](#)³³⁴ section.

For more information on how to use cost of observation in your network see [Cost of Observation](#)³³⁴ section.

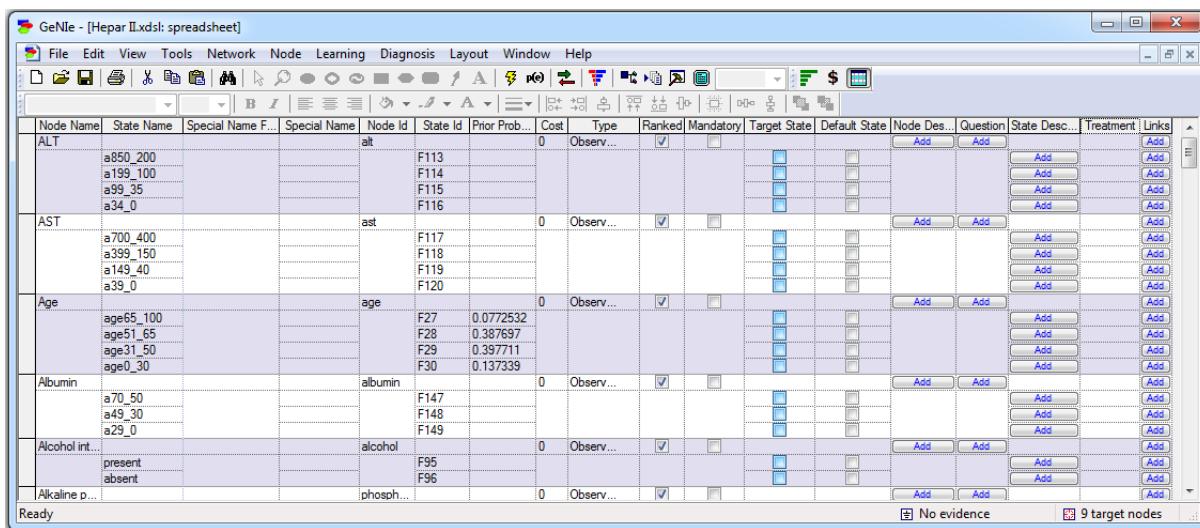
6.4.5 Spreadsheet view

The *Spreadsheet View* is a special extension of GeNle 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 click on the *Spreadsheet* () button located at the top right hand side of the task bar in GeNle.

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](#)^[138] sheets, although the *Node Properties Sheet* contains additional information that is not available in the *Spreadsheet View*, e.g., conditional probability tables.



The screenshot shows the GeNle application window titled "GeNle - [Hepar II.xls: spreadsheet]". The menu bar includes File, Edit, View, Tools, Network, Node, Learning, Diagnosis, Layout, Window, and Help. The toolbar below the menu has various icons for file operations like Open, Save, Print, and zoom. The main area is a spreadsheet grid with the following columns:

Node Name	State Name	Special Name F...	Special Name	Node Id	State Id	Prior Prob.	Cost	Type	Ranked	Mandatory	Target State	Default State	Node Des.	Question	State Desc...	Treatment	Links
ALT			alt		F113		0	Observ...	<input checked="" type="checkbox"/>	<input type="checkbox"/>			Add	Add		Add	Add
	a850_200				F114								Add	Add		Add	Add
	a199_100				F115								Add	Add		Add	Add
	a99_35				F116								Add	Add		Add	Add
AST			ast		F117		0	Observ...	<input checked="" type="checkbox"/>	<input type="checkbox"/>			Add	Add		Add	Add
	a700_400				F118								Add	Add		Add	Add
	a399_150				F119								Add	Add		Add	Add
	a149_40				F120								Add	Add		Add	Add
Age			age		F27	0.0772532		Observ...	<input checked="" type="checkbox"/>	<input type="checkbox"/>			Add	Add		Add	Add
	age65_100				F28	0.387697							Add	Add		Add	Add
	age51_65				F29	0.397711							Add	Add		Add	Add
	age31_50				F30	0.137339							Add	Add		Add	Add
Albumin			albumin		F147		0	Observ...	<input checked="" type="checkbox"/>	<input type="checkbox"/>			Add	Add		Add	Add
	a70_50				F148								Add	Add		Add	Add
	a49_30				F149								Add	Add		Add	Add
Alcohol int...			alcohol		F95		0	Observ...	<input checked="" type="checkbox"/>	<input type="checkbox"/>			Add	Add		Add	Add
	present				F96								Add	Add		Add	Add
Alkaline p...			phosph...				0	Observ...	<input checked="" type="checkbox"/>	<input type="checkbox"/>			Add	Add		Add	Add
	absent												Add	Add		Add	Add

At the bottom left is a status bar labeled "Ready". At the bottom right are two buttons: "No evidence" and "9 target nodes".

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*.

State Name

The *State Name* column lists the node's states. For example, node *ALT* has four states: *a850_200*, *a199_100*, *a99_35*, and *a34_0*. Each of these four states is described further in the spreadsheet. A state name can be changed manually by clicking on the box and by typing in a different name.

Special Name Format

Special Name Format is an additional name assigned only for *Target states* of *Target nodes* (see [Enabling diagnostic extensions](#)³⁰⁸ section). For *Special Name Format*, the user can choose from among *User Defined*, *Node Name*, *State Name*, and *Node+State Name* and *Inherit*. If *User Defined* is chosen, then the user can type in a name of their choice. If *Node Name* is selected, then the node name will appear. If *State Name* is selected, then the *Target state* will appear. If *Node+State Name* is selected, then the *Target node name* will appear followed by the *Target state name*. This name may be chosen to be the same as a *Target state name*, as a *Target node name*, or as a *Target node name* followed by *Target state name*.

Special Name Format can be defined at a network level. This name is used for the node if we choose the option *Inherit*. The user can specify one of the options above at the network level in the *General* tab of [Network property](#)¹²³ sheets in GeNle. The selected format become then the default selection for all new nodes of the network.

Special Name

Special Name column displays the special name for *Target states* of *Target nodes*. The name can be defined by using one of the *Special Name Formats* or can be specified by the user. When the user types anything in the box *Special Name*, the *Special Name Format* box for that node will be automatically changed to *User Defined*. If *State Name* were chosen, then only the state name would be used. Similarly, for *Node Name*, only the node name would be used. If the user were to choose *Node Name + State Name*, then the special name would be a concatenation of the two names. Special names can be changed manually by clicking on the box and by typing in a different name.

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).

State ID

The *State ID* is a unique identifier for each state of a particular node. The *State ID* for a particular state can be changed manually by clicking on the box and typing in a different name. *State IDs* consist of a string of alphanumerical characters with no spaces (underscore characters are allowed).

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](#)¹²³ 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. See the [Cost of observation](#)³³⁴ section in *Support for Diagnosis* section for more information. The *Observation* cost can also be specified from the *Node Property* sheets for the node. See the [Observation Cost](#)³³⁴ section for more information on how to do this.

Type

Type describes the type of node. By clicking on *Type*, the rows can be sorted according to type. The type can be changed manually by clicking on the box and by

typing in a different name. In the figure above, there is only one type of nodes listed, [Observation](#)³⁰⁸ nodes.

You can also change a node type after creation either by right clicking on the node in the [Graph View](#)⁶⁰ and selecting *Change Type* from the pop-up menu or selecting *Change Type* from the *Node Menu* in the *Menu Bar*. It will display a dialog box where you can choose the appropriate type for the node. See *Change Type* section of [Node Menu](#)²⁰⁷ for more information.

For more information on the functions of the various types of nodes, see [Enabling diagnostic extensions](#)³⁰⁸ section.

Ranked & Mandatory

The *Ranked* and *Mandatory* columns are used to represent subtypes of each node type. Checking and unchecking the boxes in corresponding columns specifies their subtype. More than one subtype may be selected for a given node type, i.e., a node can be both *Ranked* and *Mandatory*. Legal combinations of the two are described in the [Diagnostic Properties](#)³⁰⁸ section. If a box has been grayed out, then it does not pertain to the particular node type and cannot be changed.

The *Ranked* and *Mandatory* status can also be specified from the *General Tab* in the *Node Properties Sheet*. See the *General Tab* section of [Node Property](#)³⁰⁸ sheets for more information on how to do this.

Target State & Default State

The *Target 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 *Target* and *Default* status can also be specified from the *General Tab* in the *Node Properties Sheet*. See the *General Tab* section of [Node Property](#)³⁰⁸ 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](#)¹²³ sheets. See the *Documentation Tab* section of [Node Property](#)³⁰⁸ 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](#)³⁰⁸ sheets. See the *Documentation Tab* section of [Node Property](#)³⁰⁸ 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](#)³⁰⁸ sheets. See the *Documentation Tab* section of [Node Property](#)³⁰⁸ 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 *Target* 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](#)³⁰⁸ sheets. See the *Documentation Tab* section of [Node Property](#)³⁰⁸ sheets for more information on how to do this.

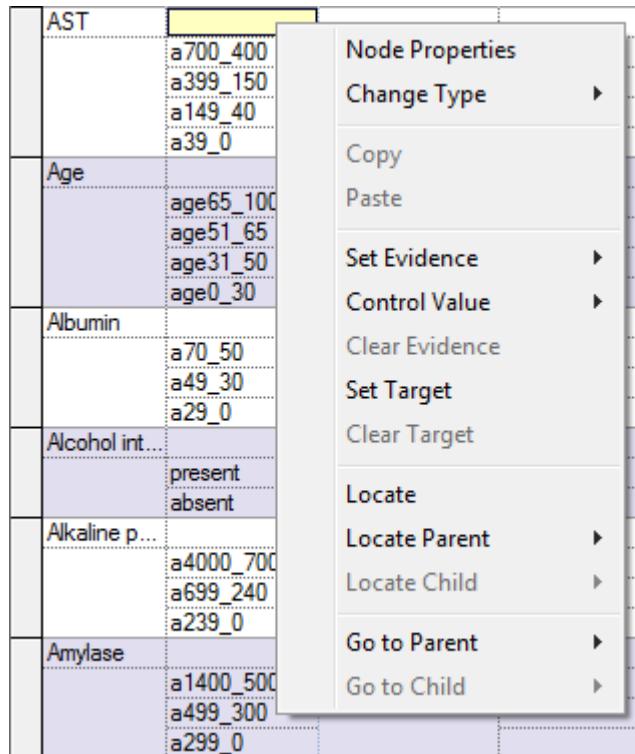
Links

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](#)³⁰⁸ sheets. See the *Documentation Tab* section of [Node Property](#)³⁰⁸ sheets for more information on how to do this.

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](#)⁶⁰.

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.

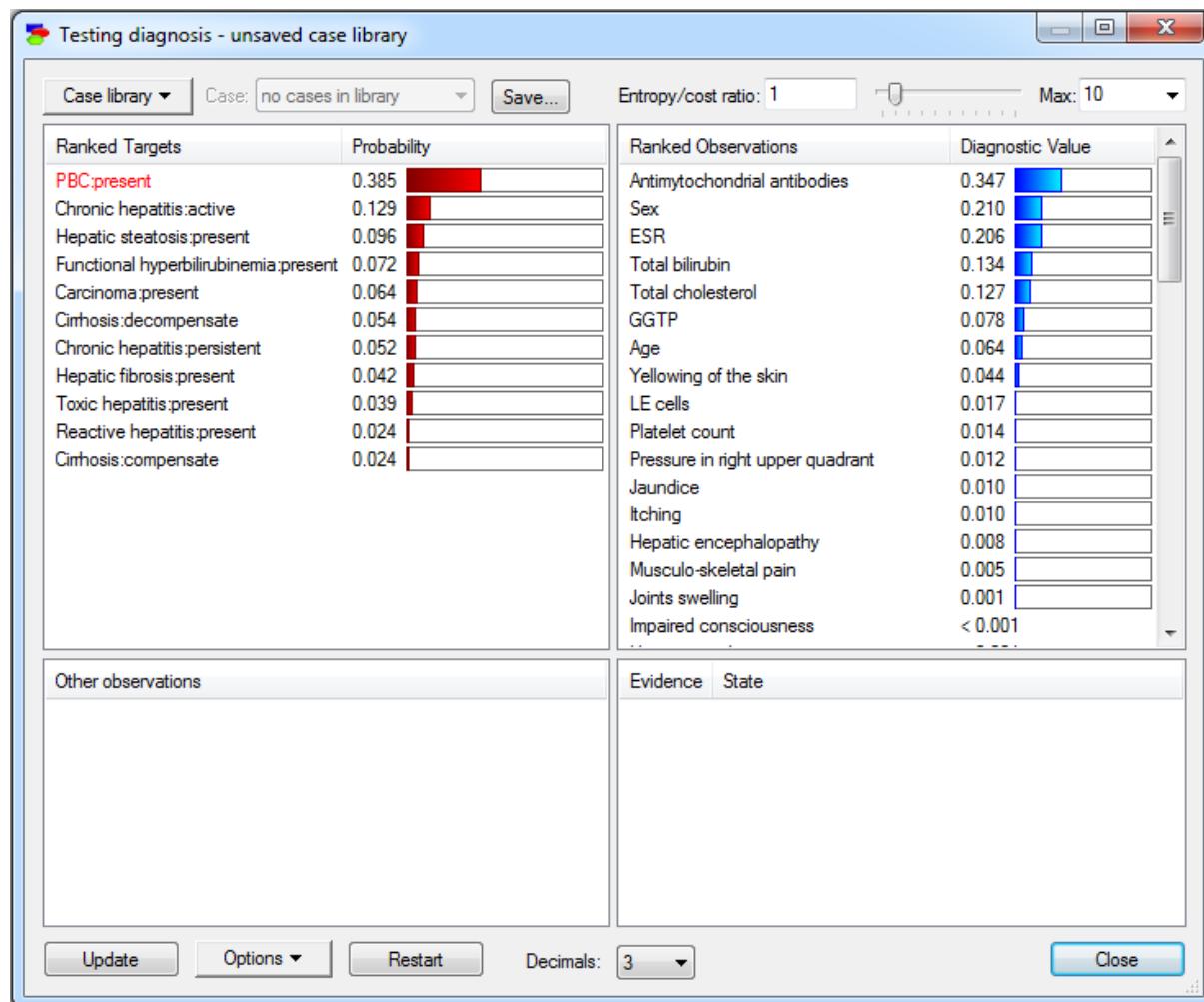
Go to Parent and *Go to Child* allow for moving around in the *Spreadsheet* view. They locate the selected parent or child and move the cursor to the corresponding line in the spreadsheet.

6.4.6 Testing window

The *Testing window* is a special GeNIE module that supports diagnostic applications.

Composition of the testing 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 *Test Diagnosis* () button on the toolbar. Once the *Testing window* has opened, it will appear as shown below:



The diagnostic window has four panes, which can be described as follows:

- The top-left pane is for the *Target* nodes, which are non-observable and ranked nodes, typically representing various faulty components.

- 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.
- 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 node from among both the *Ranked Observations* and the *Ranked Observations*, that have been observed.

To record the value of an observation node (both the *Ranked Observations* and the *Ranked Observations*), right-click the node name and select the observed state on the pop-up menu that show up.

Ranked Observations	Diagnostic Value
Antimytochondrial antibodies	0.347
Sex	0.210
ESR	0.206
Total bilirubin	0.134
Total cholesterol	0.127

Please note that the node will then show up in the bottom-right pane of the dialog window.

Evidence	State
Antimytochondrial antibodies	present

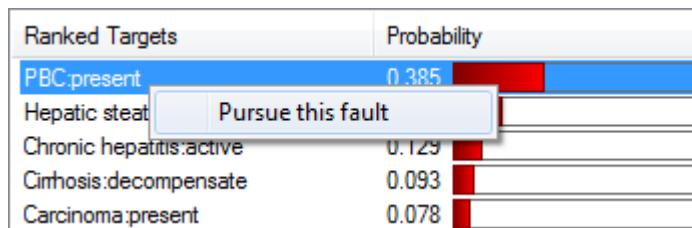
Diagnostic Value of Information

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 an information-theoretic measure known as cross-entropy and expresses, for each *Observation* node X individually, the expected reduction in entropy of the target nodes in red font in the top-left pane after observing X . Cross-entropy is a utility-free

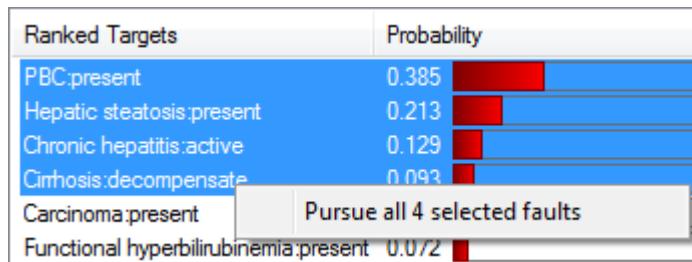
measure of value of information and it gives a good idea about the value of the observations for diagnosing the disorder in question.

Ranking the possible observations can be used as a guide for diagnostician what to do next. The cross-entropy is a dynamic measure and it depends strongly on the collection of target nodes and also on other observation. Calculation of cross-entropy is computationally complex and involves a series of runs of a belief updating algorithm. It is useful to view the calculation of cross-entropy in GeNle 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 to follow a standard "once-size-fits-all" set of questions that are typically irrelevant to the problem at hand. The ranked list calculated by GeNle is always up-to-date and tells us at each step what to do next and what question makes the most sense.

The first and foremost setting used in the calculation of cross-entropy is setting the focus of reasoning. We do this by selecting a target state (or a set of target states), right-clicking on the selection, and choosing pursue fault.



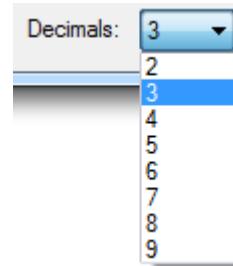
The selected fault will be displayed in red and the top-right pane will show a ranked list of observations that are relevant to this fault. We can select a group of target states and pursue them all, in which case, the top-right pane will show a ranked list of observations that help us in differentiating among the selected states.



Here is an example set of ranked observation nodes:

Ranked Observations	Diagnostic Value
Antimytochondrial antibodies	0.347
Sex	0.210
ESR	0.185
Total bilirubin	0.122
Total cholesterol	0.104
GGTP	0.078
Age	0.064
Yellowing of the skin	0.041
LE cells	0.017
Pressure in right upper quadrant	0.012
Platelet count	0.010
Jaundice	0.009
Itching	0.009
Hepatic encephalopathy	0.007
Musculo-skeletal pain	0.005
Joints swelling	0.001
Impaired consciousness	< 0.001
...	...

For each observation node, the window displays its diagnostic value numerically (precision of this number is controlled by the *Decimals* pop-up menu at the bottom of the dialog) and graphically.



While the graphical display is less precise, it is very convenient for quick judgment of relative value.

Entropy/Cost Ratio

Cross-entropy is a unit-less measure, while in most diagnostic applications diagnosticians may be more interested in monetary costs of the perform tests and benefits of observations. As you remember, one of the items specified for any observation node is the cost of observation. GeNle offers a simple formula for combining cross-entropy with costs, notably a weighted additive scheme. Total

benefits are $V = \text{cross-entropy} - wC/wE * \text{costs}$. The ratio $\alpha = wE/wC$ is called *entropy/cost ratio* and the formula for diagnostic value that includes α is as follows:

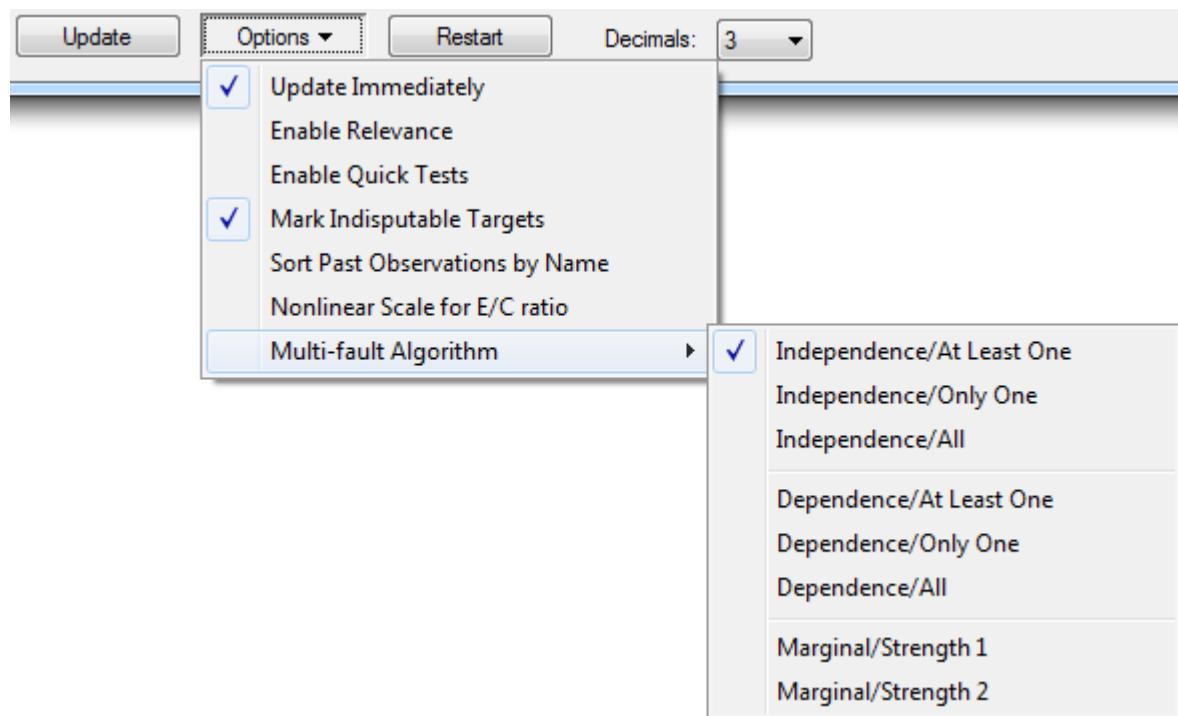
$$V = \text{cross-entropy} - \alpha * \text{costs}$$

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 at the top-right of the *Testing window*:



The *Entropy/cost ratio* is a real number ranging between 0 and 99999. The contents of the box next to the *Cost/Entropy Ratio* represents the highest value pictured on the slider 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.

Options Menu



Update Immediately, when set, makes GeNIE recalculate the ranking as soon as any change is made. If off, you will need to press the Update button to recalculate the ranking.

Enable Relevance 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.

Enable Quick Tests option, when enabled, moves those ranked observations that have negative costs (this is an indication of negligible observation costs) to the top of the ranked observations list.

Mark Indisputable Targets option, when enabled, makes any target with a probability value of 1 or 0 displayed in gray. Grayed targets cannot be pursued and do not take part in any calculations.

Sort Past Observations by Name option, when enabled, sorts the list of past observations (bottom-right window pane) alphabetically (they are otherwise sorted by the order of instantiation).

Nonlinear Scale for E/C ratio option, when enabled, switches the Entropy/cost ratio scale from linear to logarithmic. This is useful when the Entropy/cost ratio ratio is large.

Multi-fault Algorithm sub-menu

In case more than one target is pursued, calculation of cross-entropy between the selected targets and each of the observations would require deriving the joint probability distribution over the selected targets. This, for any sizable diagnostic model, would be computationally prohibitive. GeNIE offers the user in this case the choice of one of several approximate algorithms, which are divided into two groups:

Joint Probability Approach (first six options in the *Multi-fault Algorithm* sub-menu)

Marginal Probability Approach (last two options in the *Multi-fault Algorithm* sub-menu)

The *Marginal Probability Approach* is much faster but it is not as accurate as the *Joint Probability Approach*.

Joint Probability Approach

Basically this approach attempts to approximate the joint probability distribution over the targets by making assumptions about dependence among them. The two extremes are: (1) complete independence (this is taken by the first group of approaches) and (2) complete dependence (this is taken by the second approach). 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.

- *At Least One* means opposing the event that at least one of the targets is present against the event that none of the targets is present.
- *Only One* means opposing the event that exactly one target is present against all other possibilities (i.e., multiple targets or no targets present).
- *All* means opposing the event that all targets are present against the event that at least one target is absent.

In practice, each of them 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 targets.

Marginal Probability Approach

Here the approximation is quite strong and is based purely on the marginal probabilities of the targets. 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.

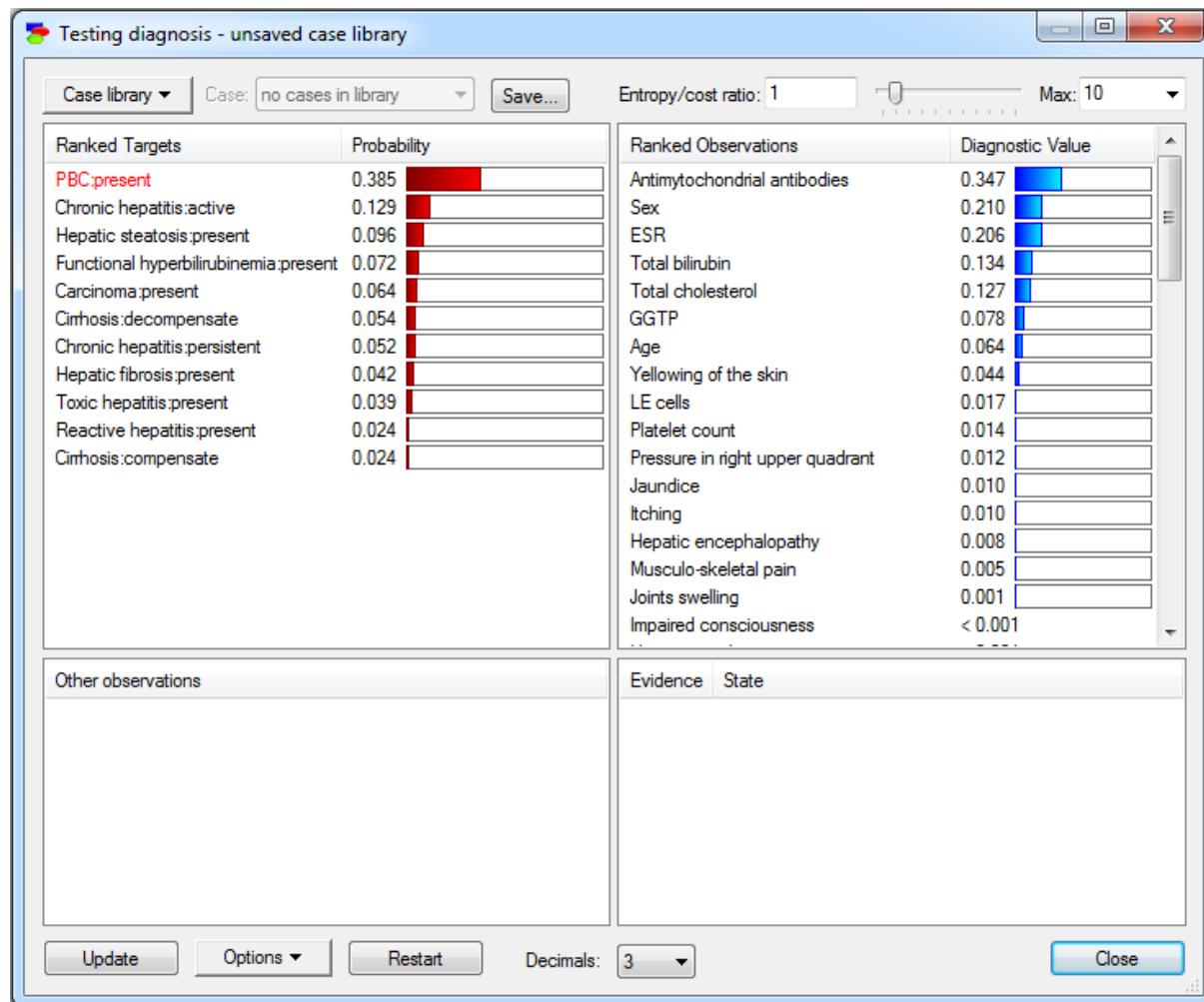
Marginal/Strength 1 uses a function without the support for maximum distance and its minimum is reached when all probabilities of the targets are equal to 0.5.

Marginal/Strength 2 uses a function that has support for maximum distance and is continuous in the domain [0,1].

6.4.7 Diagnostic case management

The *Diagnostic window* allows users to store and preserve diagnostic cases that they

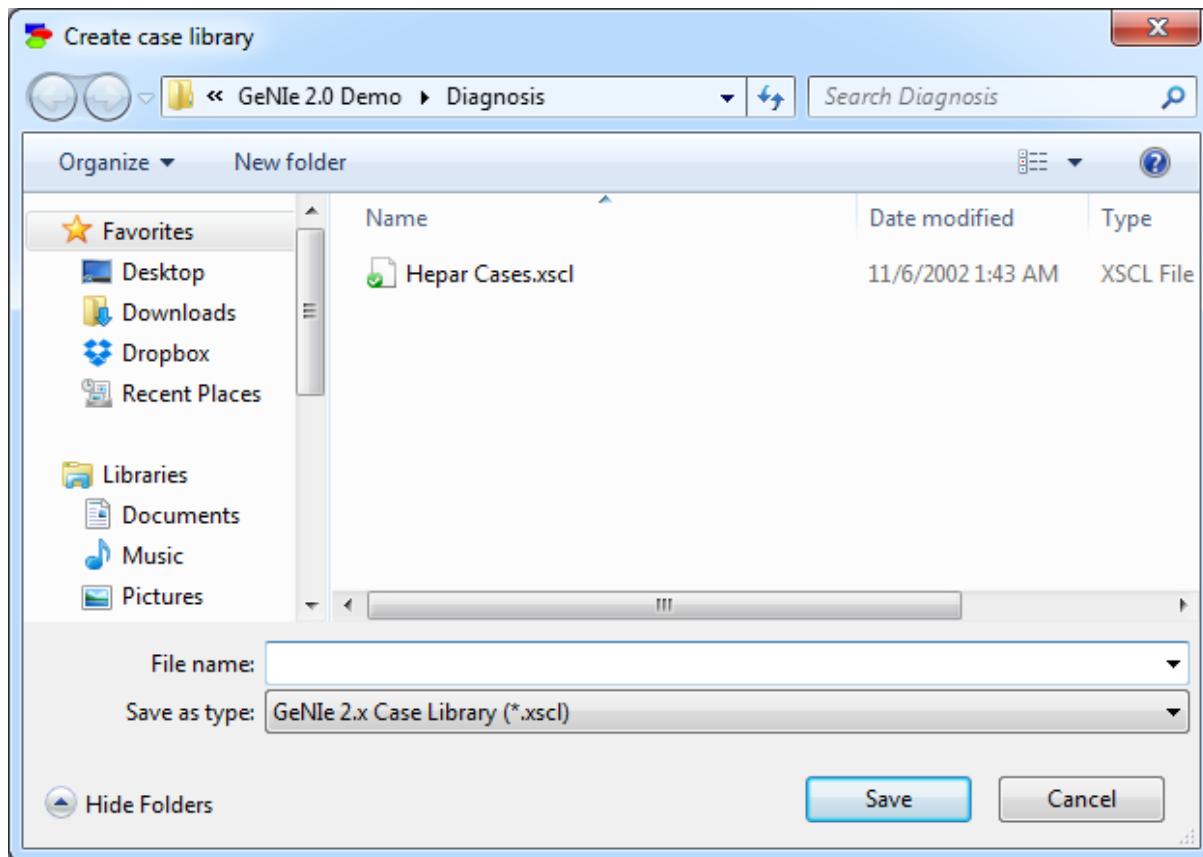
worked on using the model through the *Case library* pop-up menu in the upper-left corner.



This functionality is similar to the [Case manager](#)⁷⁸ view in the GeNIE workspace. The *Case library* pop-up menu can be used to create, open, or save a case library. Clicking on the button will display the following menu:



The *New Case Library* command is used to create a new case library. It will open the *Create case library* dialog, which is a standard *Savefile* dialog, as shown below:



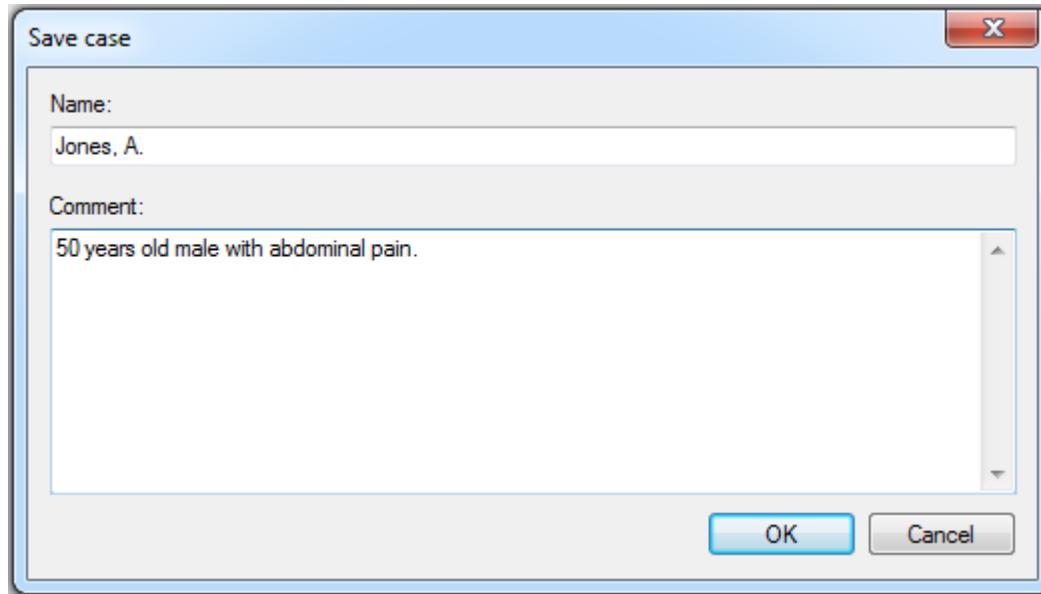
The *Open Case Library* command is used to open an existing case library. It will open a standard Open file dialog, which can be used to select the case library to open. Case libraries in GeNle are saved as files with the extension *.xscl*.

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 group your case files.

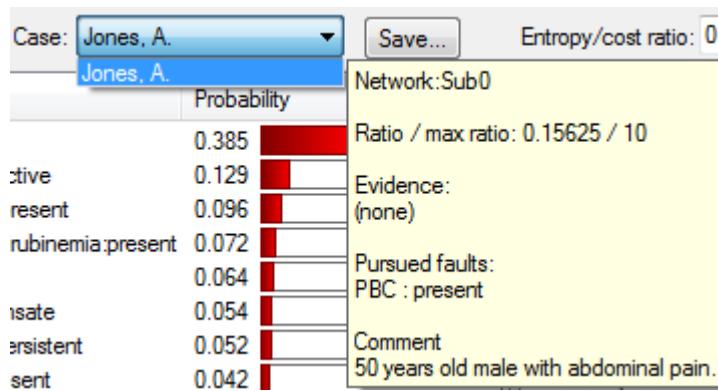
Saving a case

Diagnostic cases can be saved. A saved 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 *Save* (Save...) button becomes active. After pressing it, the following dialog appears:

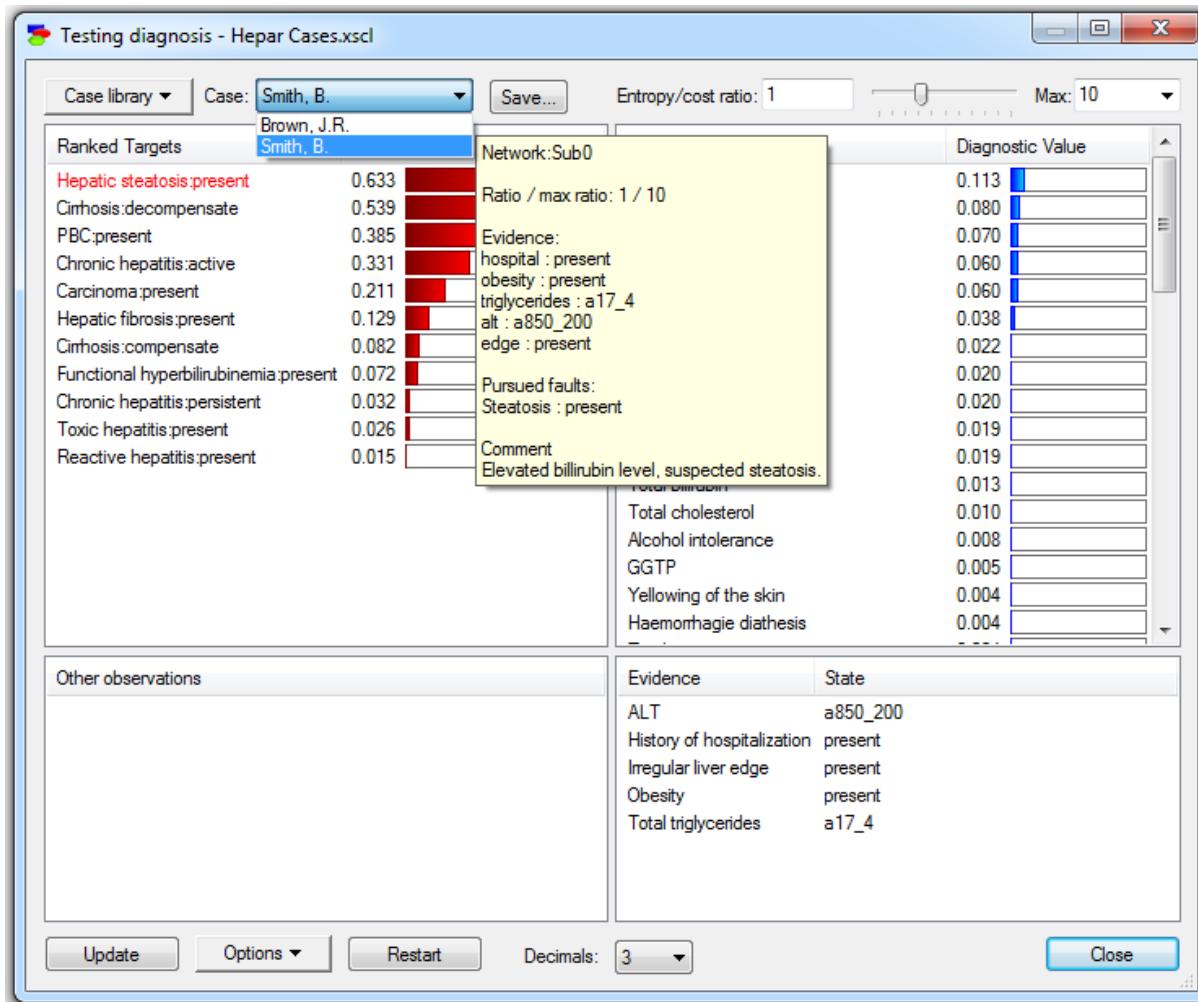


Enter the case *Name* and *Comment* and click *OK* to save the case within the currently open case library. When we click on the *Case* pop-up button, the list of cases in the library becomes visible, with the case that we hover over being summarized in a yellow preview window:



Loading a diagnostic case

To load a case from a currently open case library, use the *Case* pop-up button. 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 is instantiated.

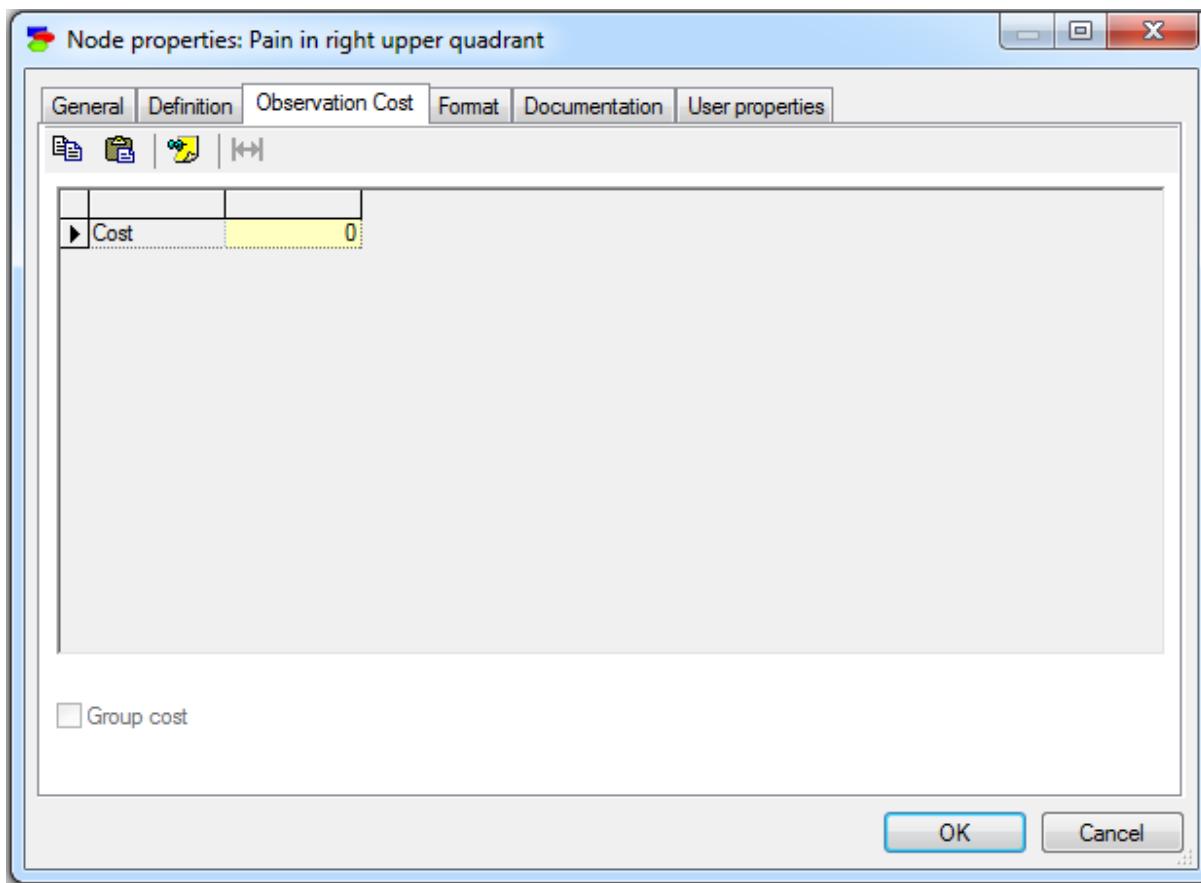


6.4.8 Cost of observation

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. GeNle allows for entering the cost of observing the value of a variable (node).

Simple costs

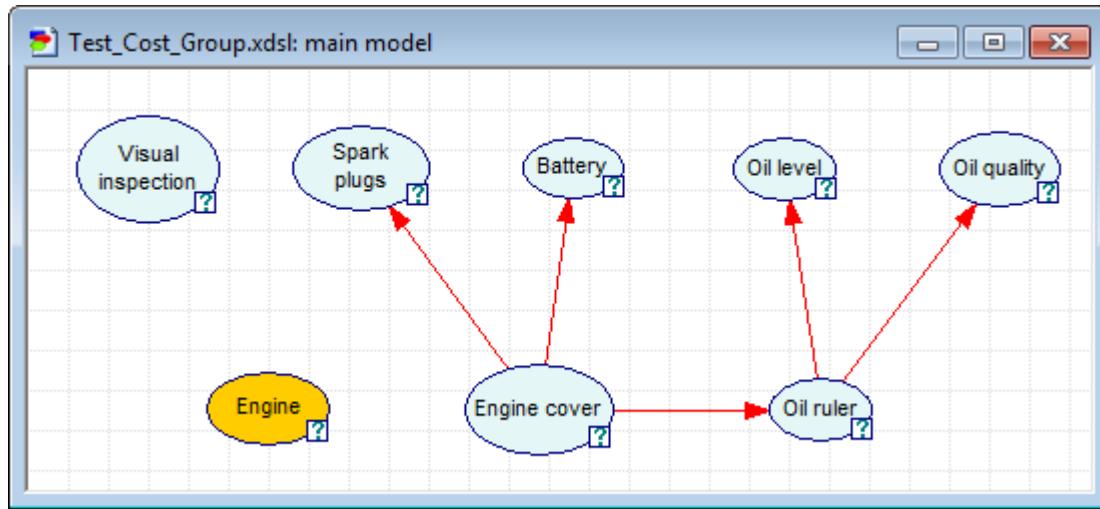
Simple cost is used when the cost of observing a node is independent of whether other nodes are observed or not. Simple cost could be the cost of performing a diagnostic test. This cost is set in the [Node Properties](#)¹³⁸ dialog under the *Observation Cost* tab. Simple costs can also be entered in 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 are 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](#)⁶⁰ 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 clicking on the cost () button on the *Diagnosis Toolbar*³⁰⁷. In the *Cost Graph View* mode, this button will remain pressed.

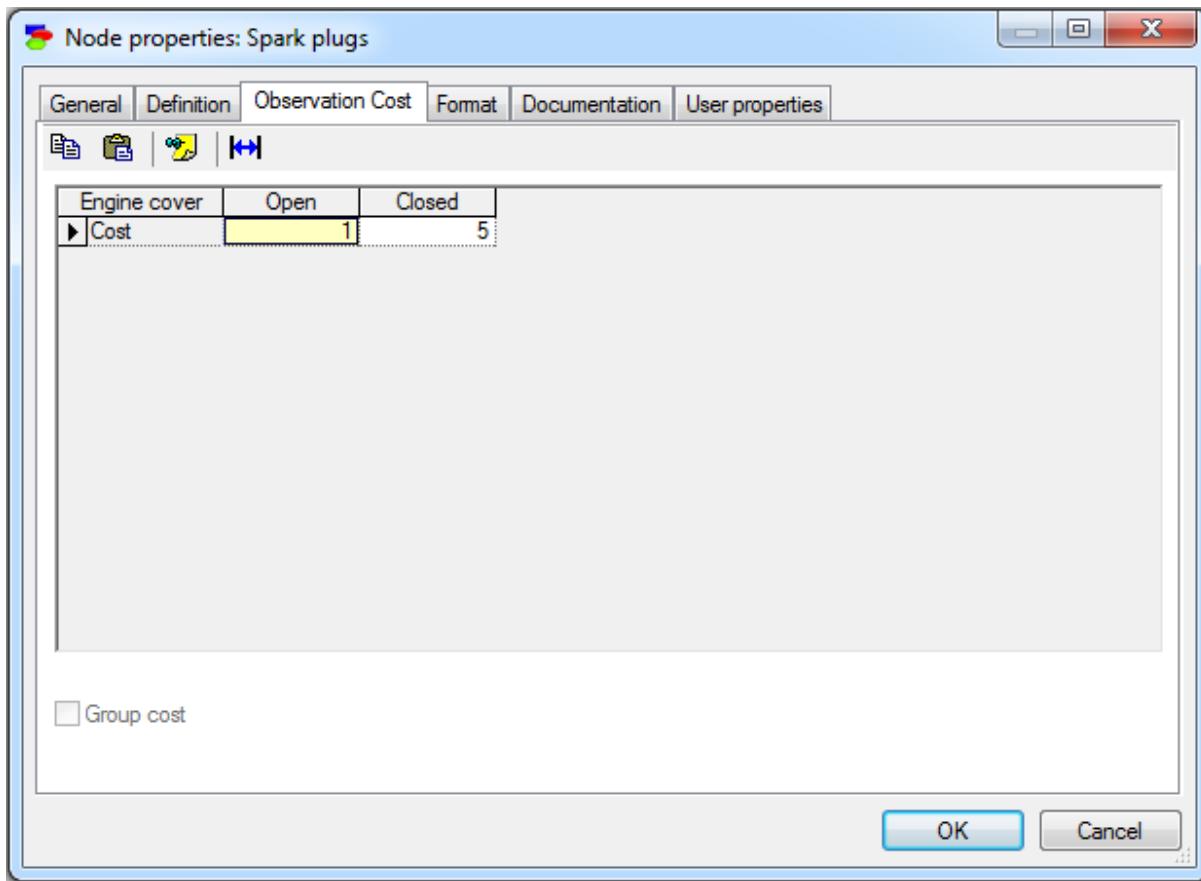
You can return to *Graph View*⁶⁰ by clicking on the cost () button again. The button returns to its normal position to indicate that you are in *Graph View*.

The following changes will occur when the *Cost Graph View* is invoked:

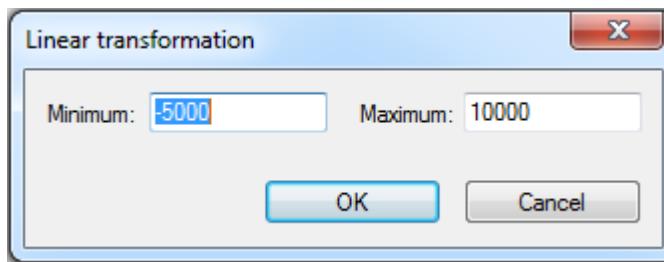
1. All arcs of your graphical model (dark navy colored by default) will disappear and only the cost arcs (red colored by default) will appear between the nodes. If the cost arcs are not visible, then probably none are yet defined (this is the initial state for any model).
2. The *arc* () button changes into a *cost arc* button ().
3. The *Cost Arc* menu item appears in the *Tool Menu*¹⁷⁶.

You can select the *cost arc* tool from either the *Tool Menu*¹⁷⁶ or by clicking on the *cost arc* button from the *Standard Toolbar*¹⁷⁶ 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



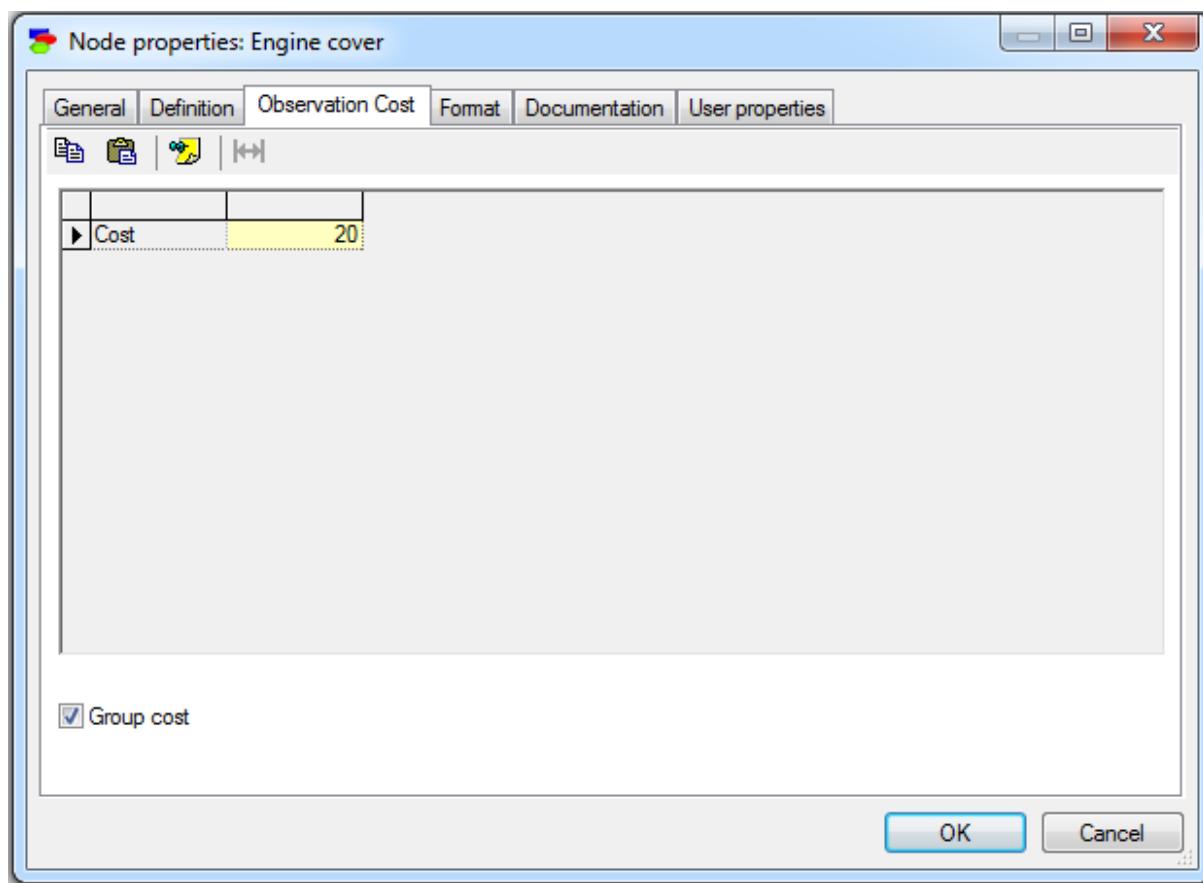
It performs a linear transformation on the costs, similarly to the transformation performed on utilities.

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

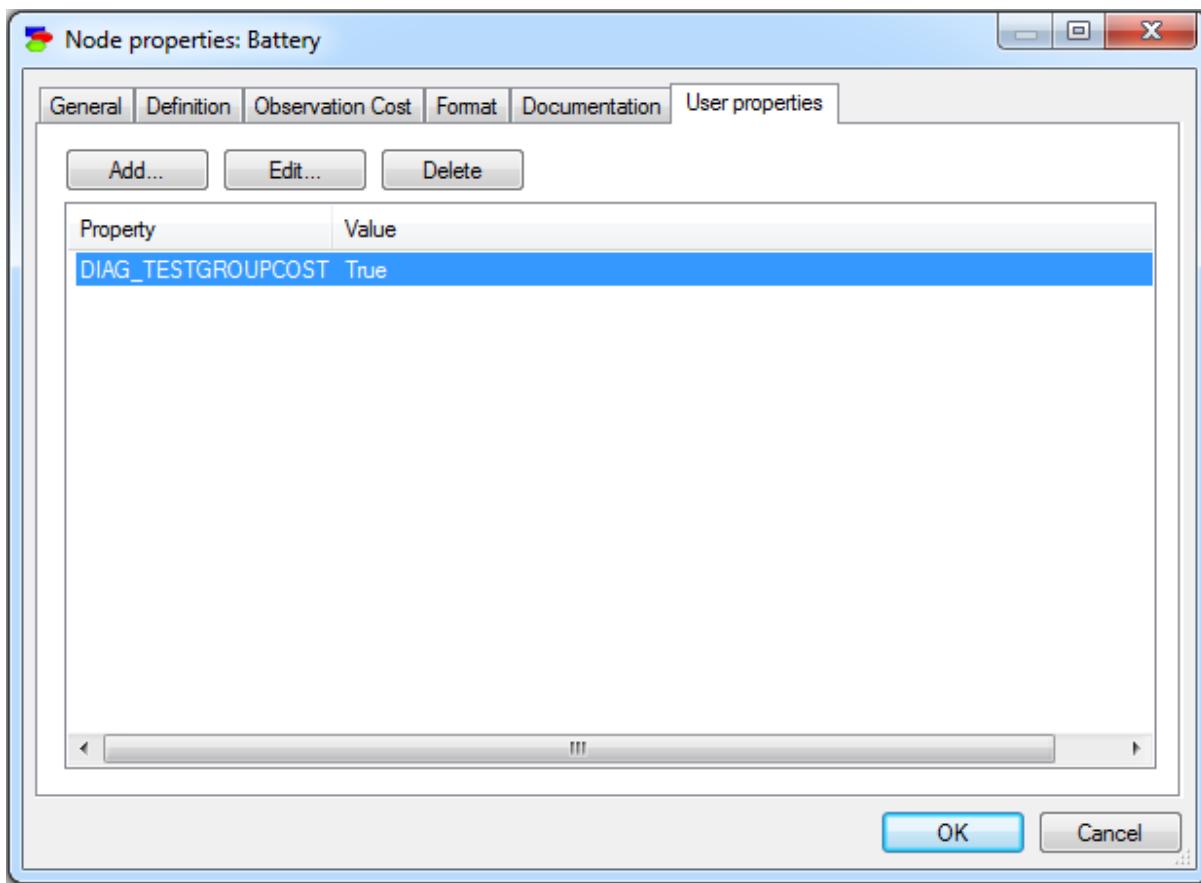
a group cost is taking a blood sample, in which the cost of the blood sedimentation rate test incurs the cost of first drawing a blood sample from a patient. Another example is performing a test of an internal part of a locomotive engine, which involves removing the engine cover of a locomotive to access 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 for 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.

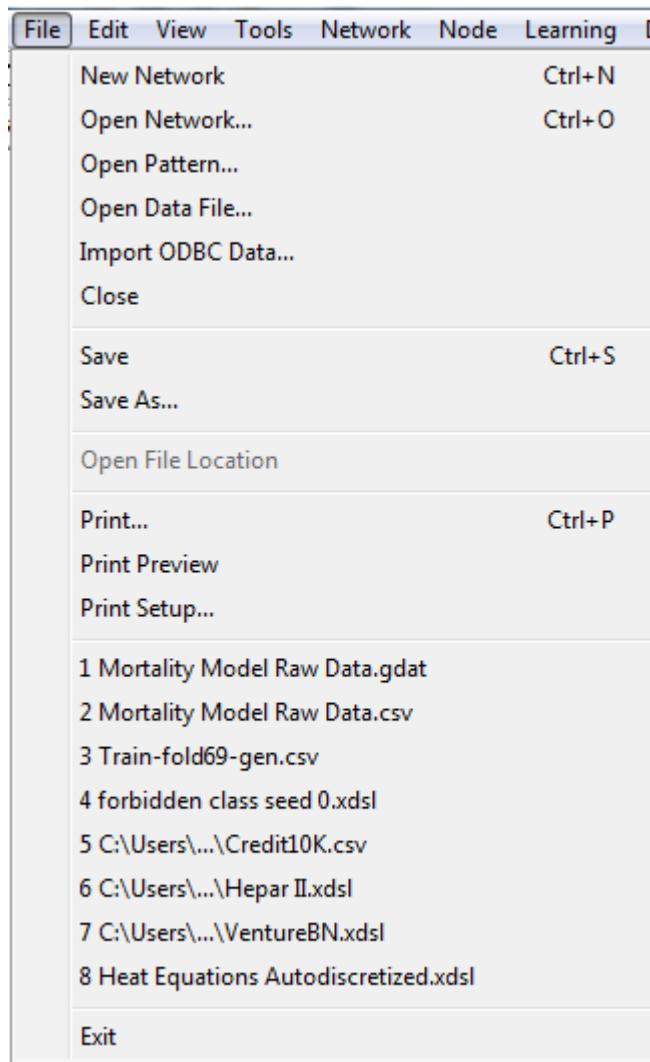
6.5 Learning

6.5.1 Accessing data

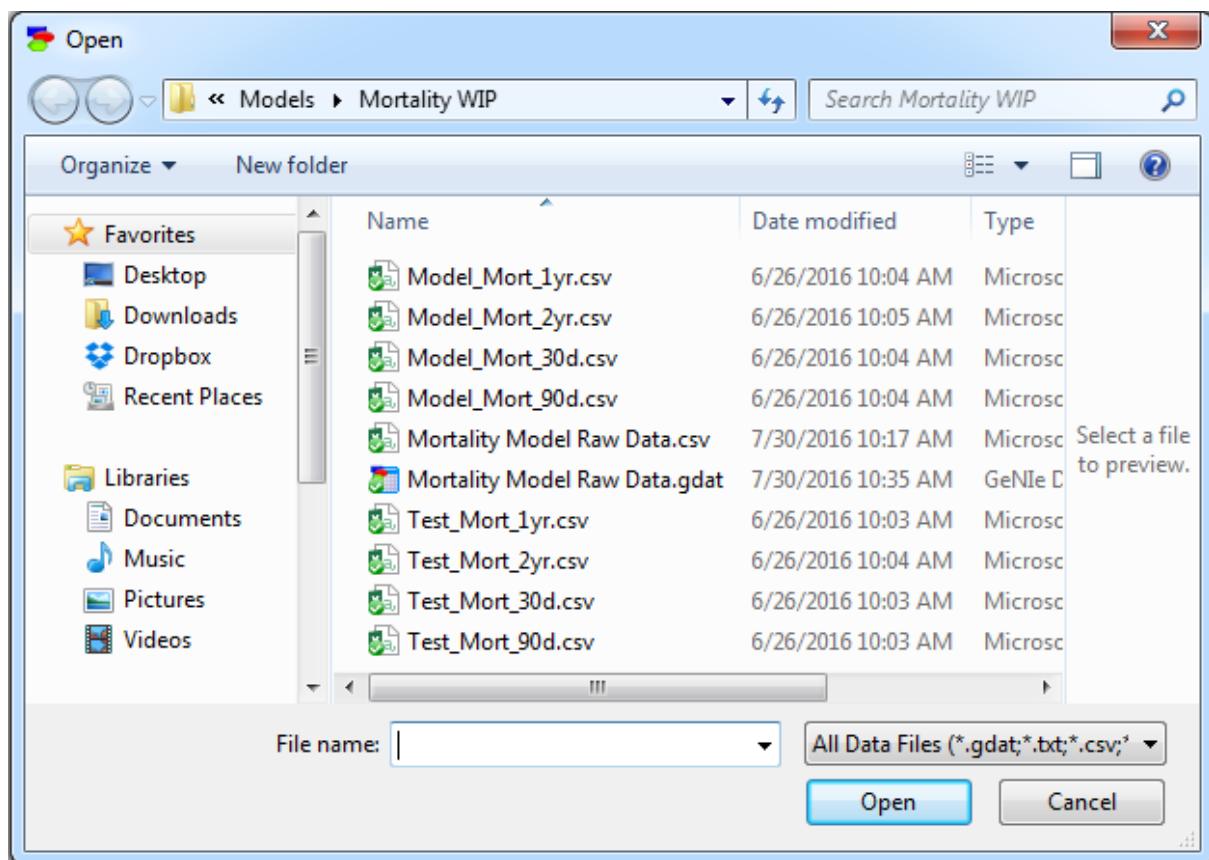
GeNIE can access data from three sources: text files, ODBC databases, and the native GeNIE data format. They will be subject of the following three sections.

Text Format (*.txt, *.dat, *.csv)

The simplest data format used by GeNle 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. The popular CSV format (used, among others, in Microsoft Excel), conforms to this standard. To access data stored in a text file select [File](#)¹⁹³-*Open Data File...*



Subsequently, select the data file that you wish to load.

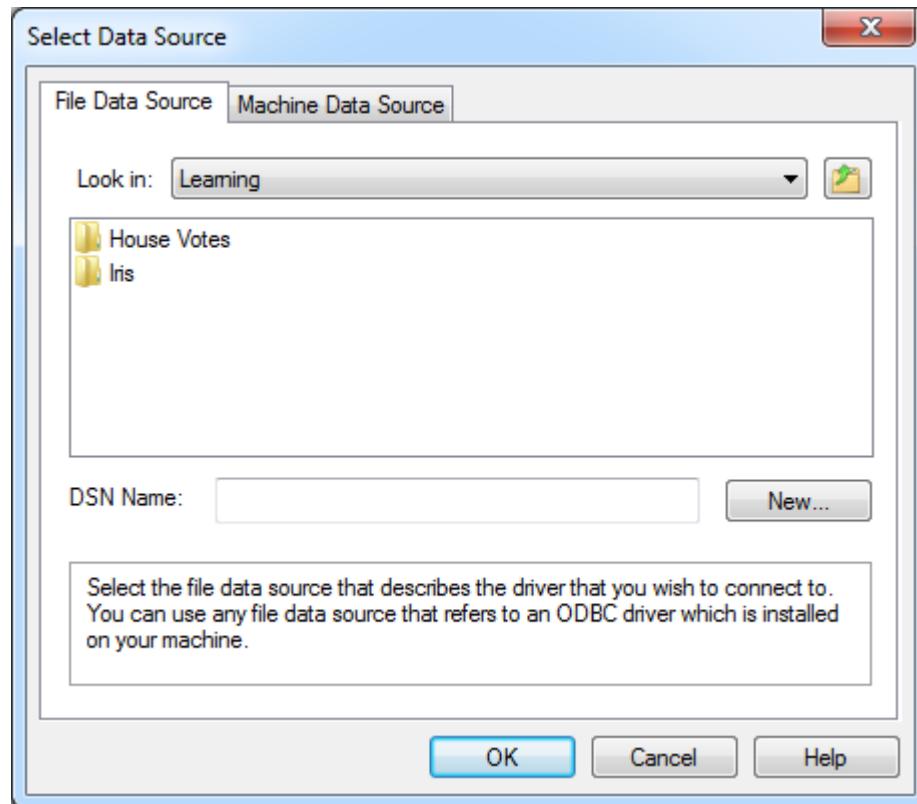


Data, once loaded, should look as follows:

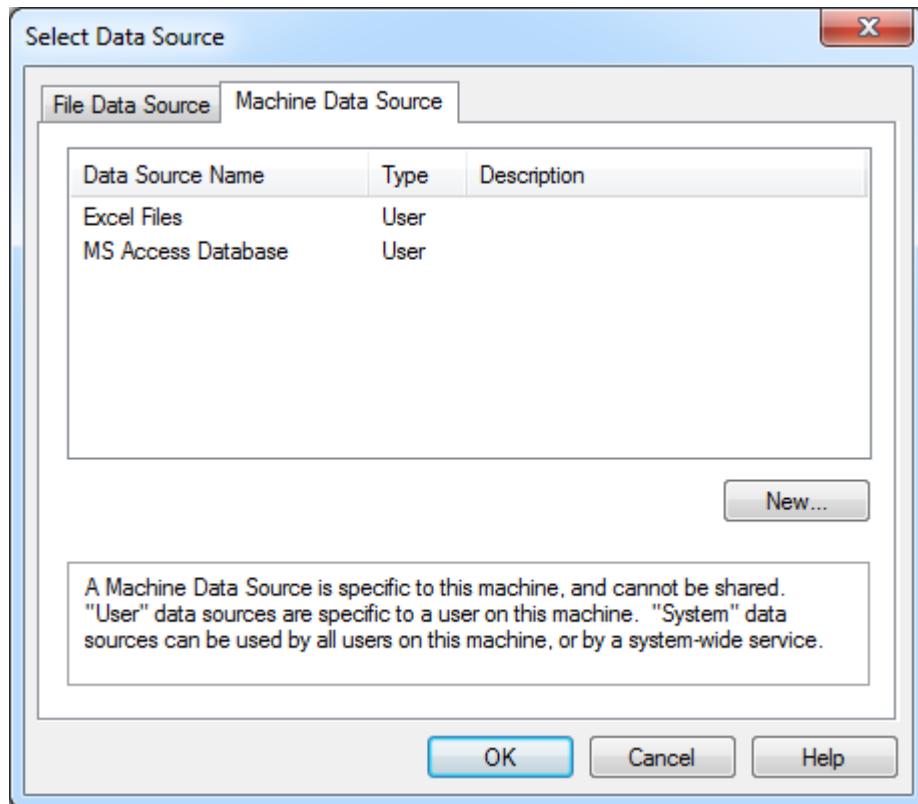
	spend	apret	top10	rejr	tstsc	pacc	strat	salar
7057	55.25	17	24.379	59.063	44.251	21.2	58200	
16848	77.75	48	26.69	75.938	27.187	9.2	63000	
18211	91	87	76.681	80.625	51.164	12.8	74400	
21561	69.25	58	44.702	76.25	26.689	9.2	75400	
20667	65	68	22.995	75.625	28.038	11	66200	
10684	61.75	26	8.774	66	33.99	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	

ODBC Data

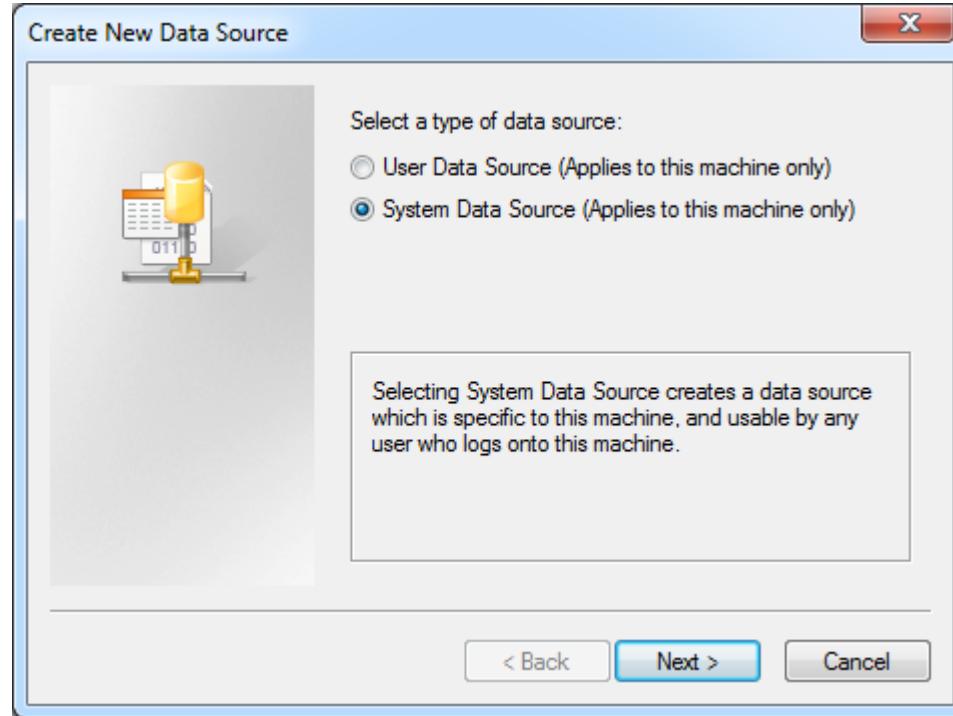
ODBC (Open Database Connectivity) is a standard application programming interface (API) for accessing database management systems (DBMS). ODBC is independent of the details of any concrete database system and operating systems. GeNle implements the ODBC standard, which allows it to connect to most DBMS. In this section, we will open a Microsoft Access database. To access the data from a database select [File](#)¹⁹³-Import ODBC Data..., which will open the *Select Data Source* dialog.



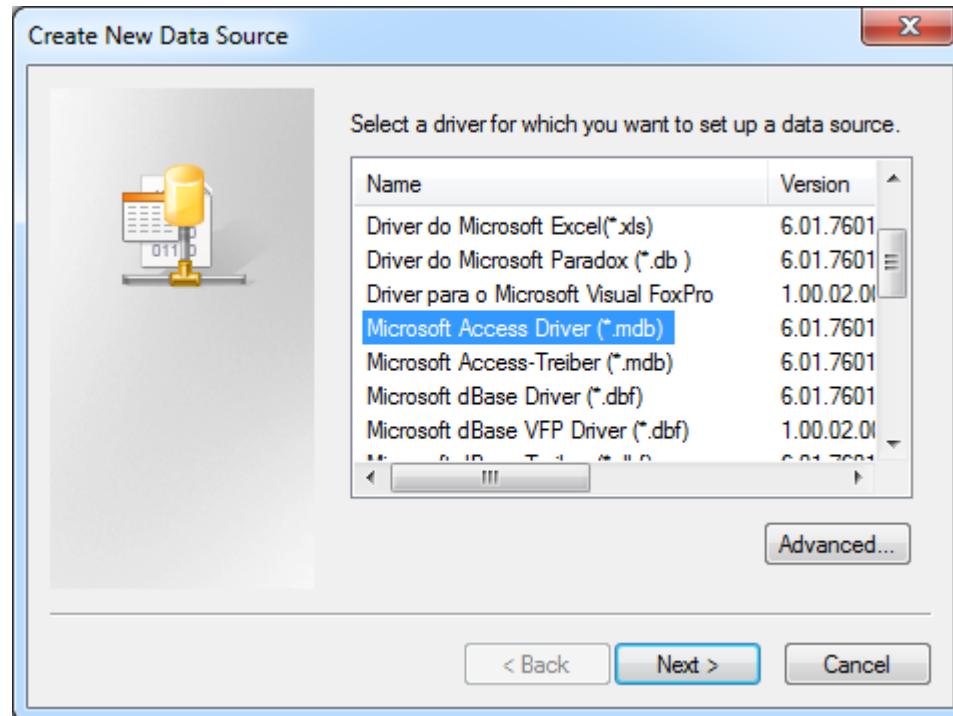
If you have never created a data source before, you will have to create a new one. It is most convenient to create a new data source that covers all files originating from a Windows application, which is a *Machine Data Source*. We will create a data source for Microsoft Access.



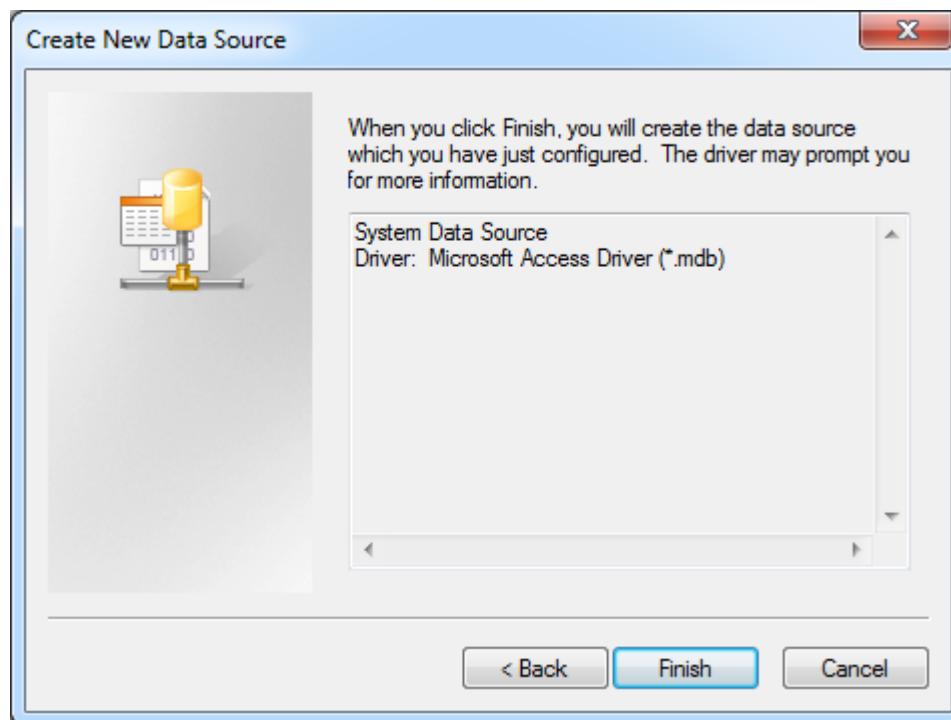
Pressing the *New...* button opens the following dialog:



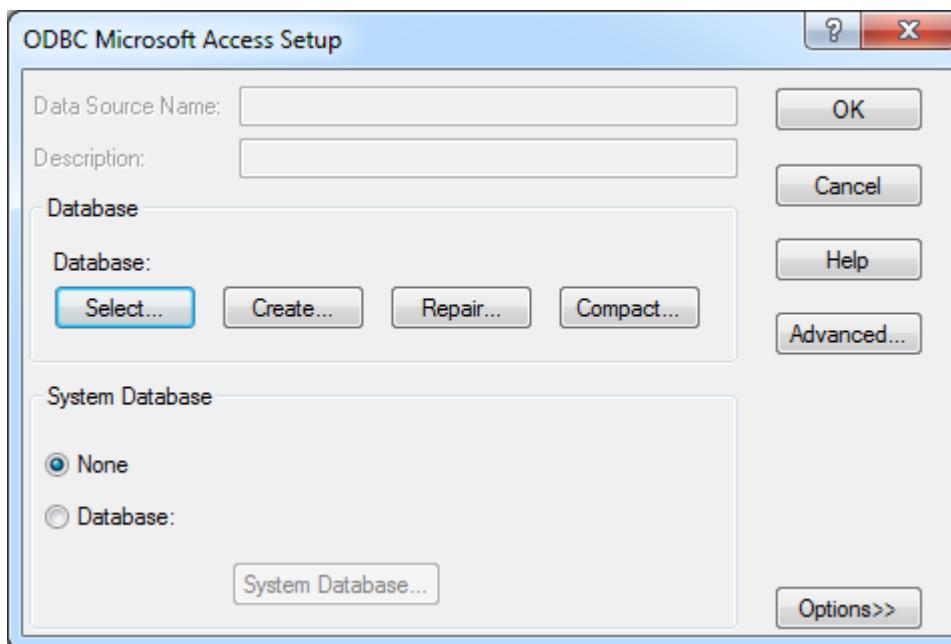
We select *Microsoft Access Driver (*.mdb)* and press *Next*:



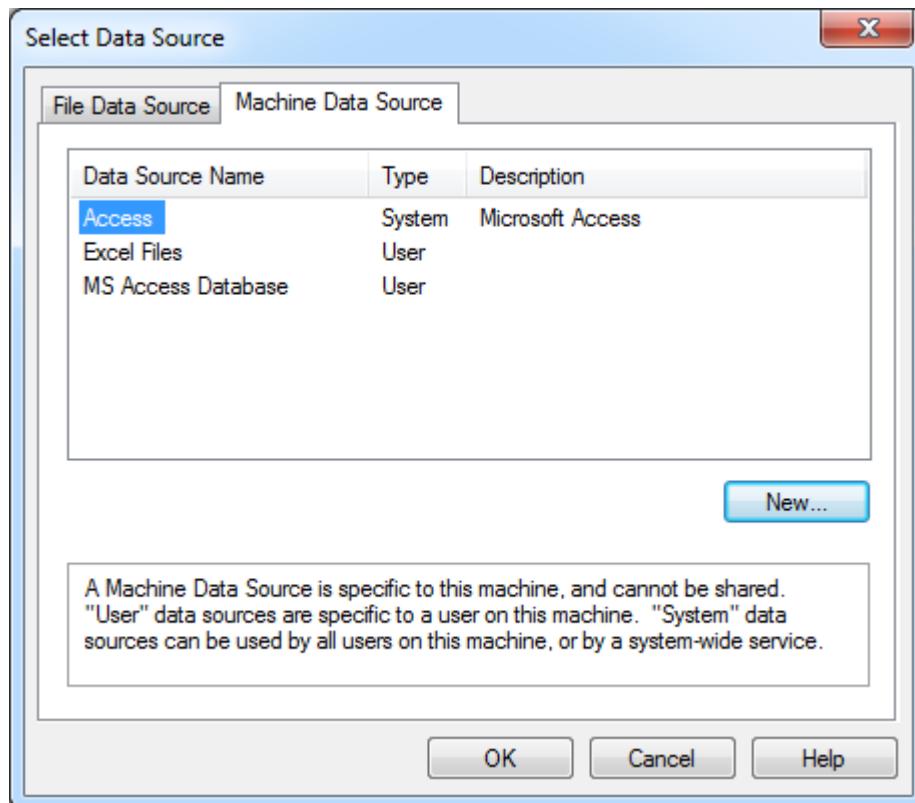
We continue by pressing *Finish*:



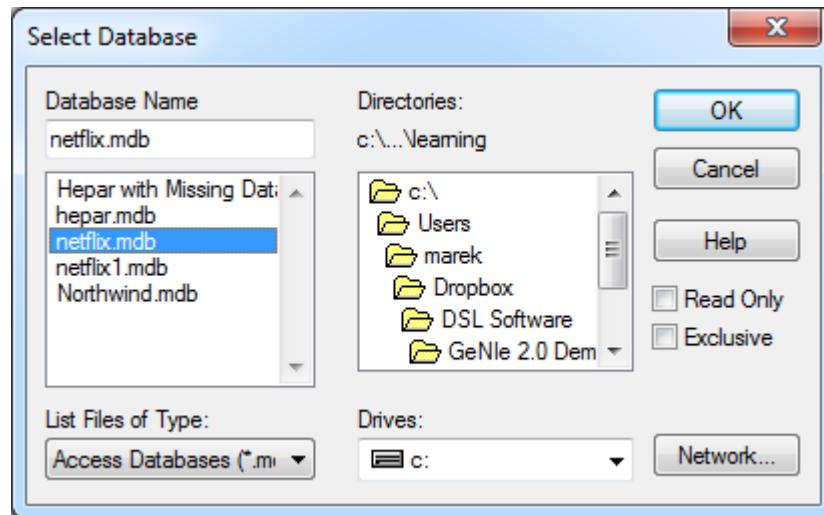
and *OK*



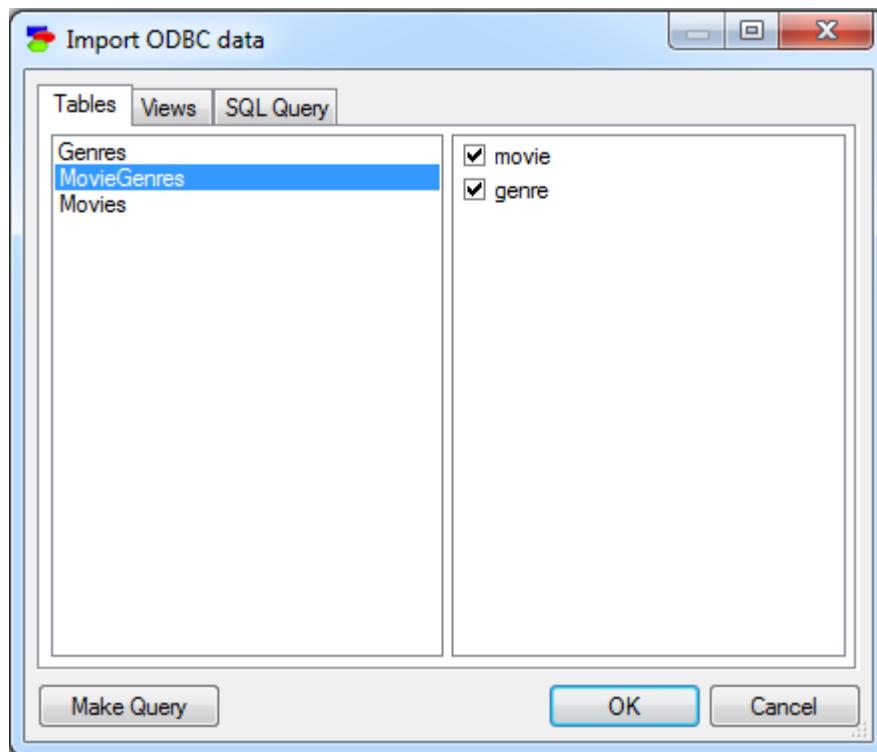
Which results in the following window



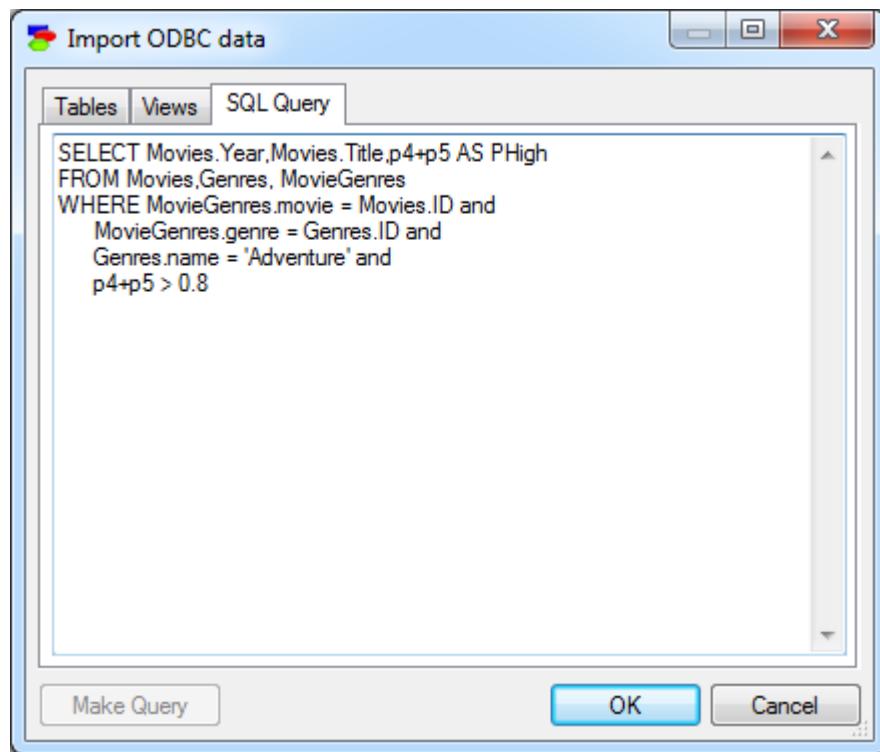
Once the data source has been created, select it and press *OK* or double click on it. GeNIE will display a dialog box that allows for selecting data, that should look as follows:



We will open the *netflix.mdb* database, visible in the previous dialog. 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.

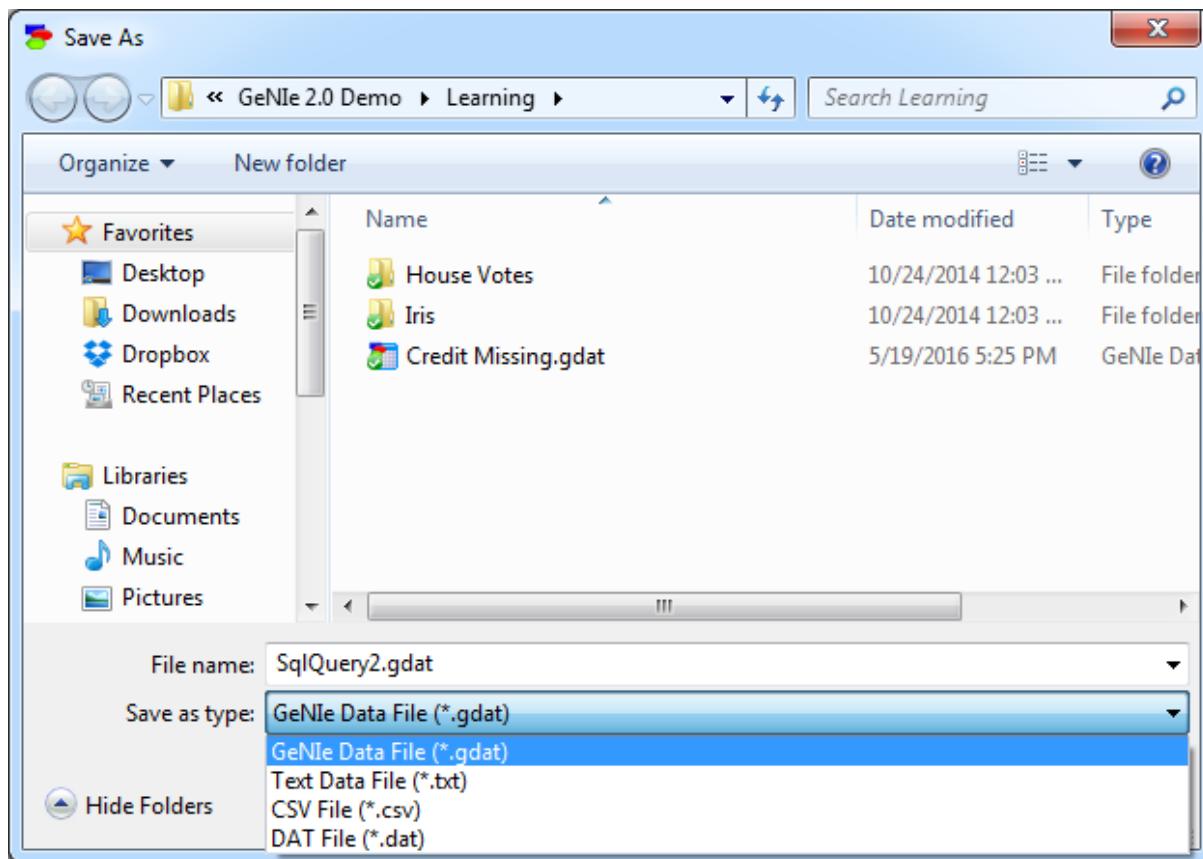


Pressing *OK* runs the query and opens the result in GeNIE:

	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

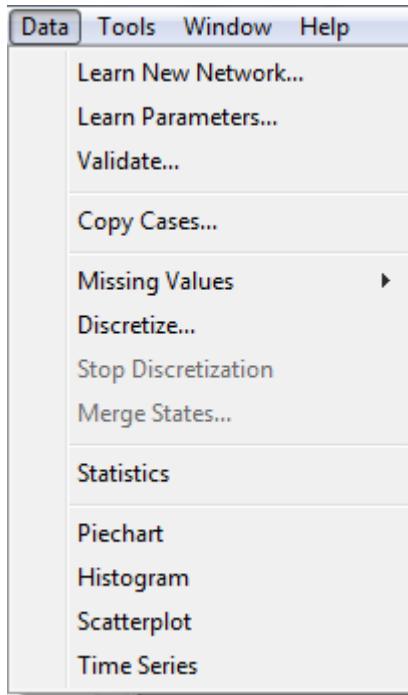
GeNle Data Format (*.gdat format)

GeNle allows to save data in a binary internal format that we call GeNle 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 preprocessing operations, such as discretization. To save your data in *GeNle Data Format*, select [File](#)¹⁹³-Save As...



6.5.2 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](#)⁴⁰⁰ section for more information.

Copy Cases... allows for importing cases from the data into a network. This function is described in the [Case manager](#)⁷⁸ 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](#)³⁵³ 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.

6.5.3 Cleaning data

Before the structure of a [Bayesian network](#)⁴⁵ (or the numerical parameters of an existing network) can be learned, it is a good idea to interactively examine the data. GeNle 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 GeNle installation folder.

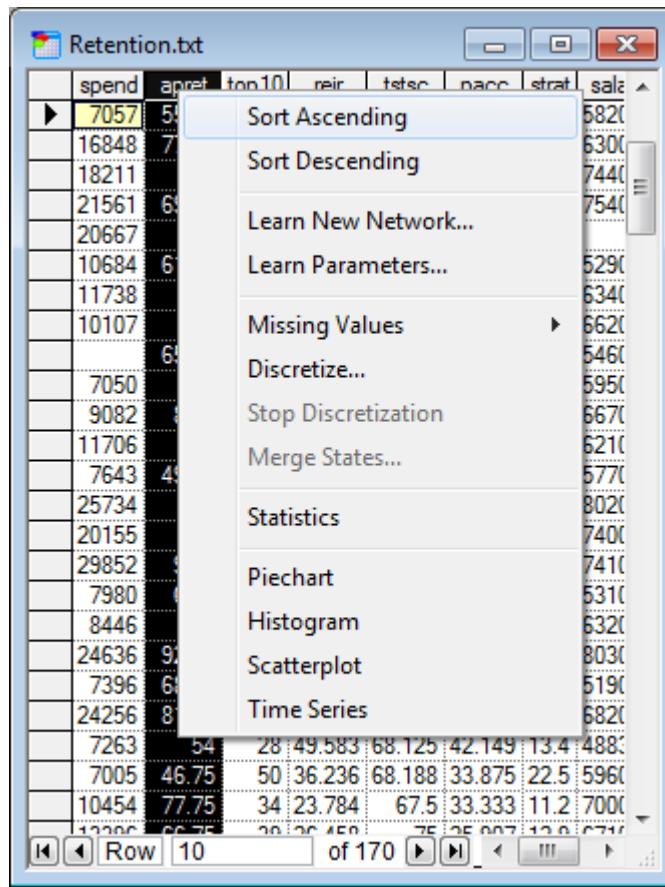
Data Grid view

GeNle 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	

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.



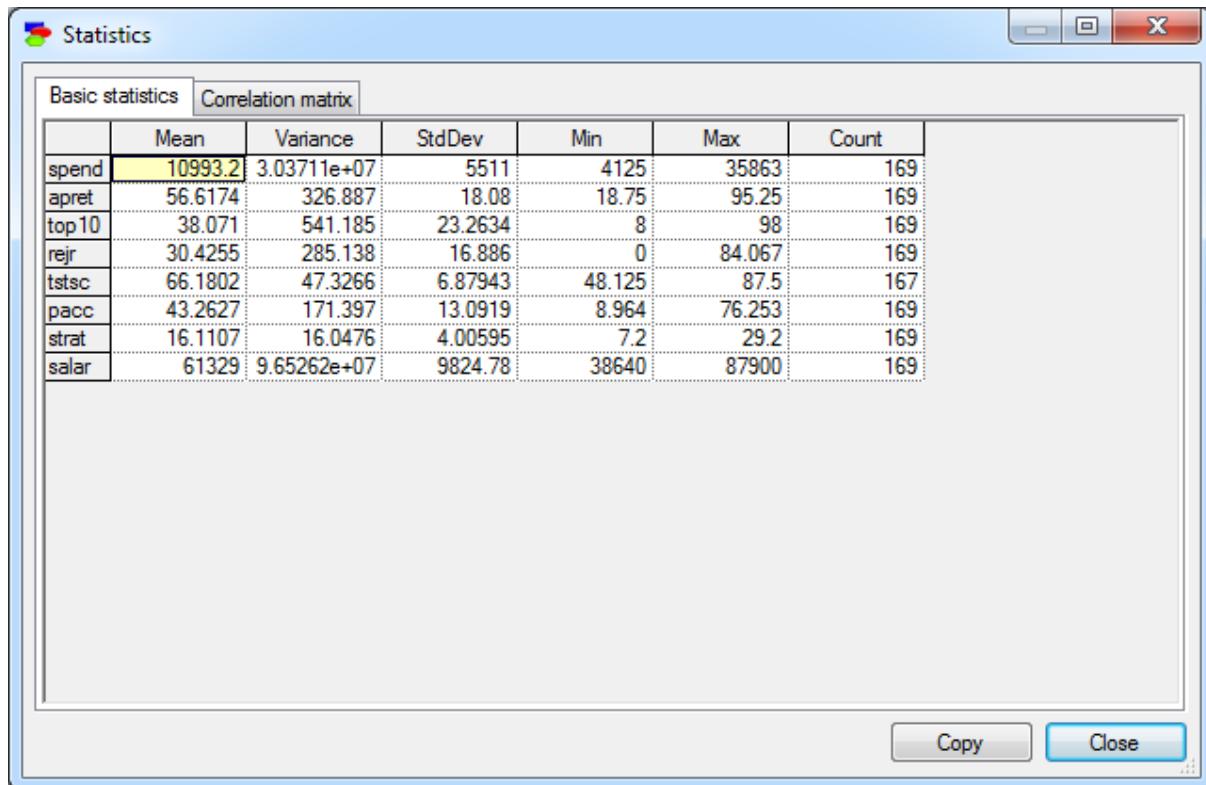
Similarly to Excel, *Data Grid* view allows for sorting data in ascending and in descending order. It is possible to sort using more than one column - just select them with *CTRL* or *SHIFT*. Please note that the order of selecting columns impacts the sort criteria. For example, with three columns (*A*, *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*.

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 an external application. 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 select 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

GeNle 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 [Data³⁵¹-Statistics](#).



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.

The screenshot shows the 'Statistics' dialog box with two tabs: 'Basic statistics' and 'Correlation matrix'. The 'Correlation matrix' tab is selected, displaying a 9x9 grid of correlation coefficients for variables: spend, apret, top10, rejr, tstsc, pacc, strat, and salar. The diagonal elements are all 1.0. The first row and column are also 1.0. The values are color-coded: green for positive correlations (e.g., 0.604935, 0.655846) and red for negative correlations (e.g., -0.225775, -0.1582147). The 'Copy' button is located at the bottom left, and the 'Close' button is at the bottom right.

	spend	apret	top10	rejr	tstsc	pacc	strat	salar
spend	-							
apret	0.604935	-						
top10	0.655846	0.614968	-					
rejr	0.627142	0.50814	0.583465	-				
tstsc	0.71076	0.782754	0.791333	0.597288	-			
pacc	-0.225775	-0.296254	-0.217051	-0.0819101	-0.166461	-		
strat	-0.1582147	-0.495107	-0.243254	-0.296102	-0.495816	0.117766	-	
salar	0.71316	0.636018	0.61544	0.603714	0.714854	-0.373601	-0.345698	-

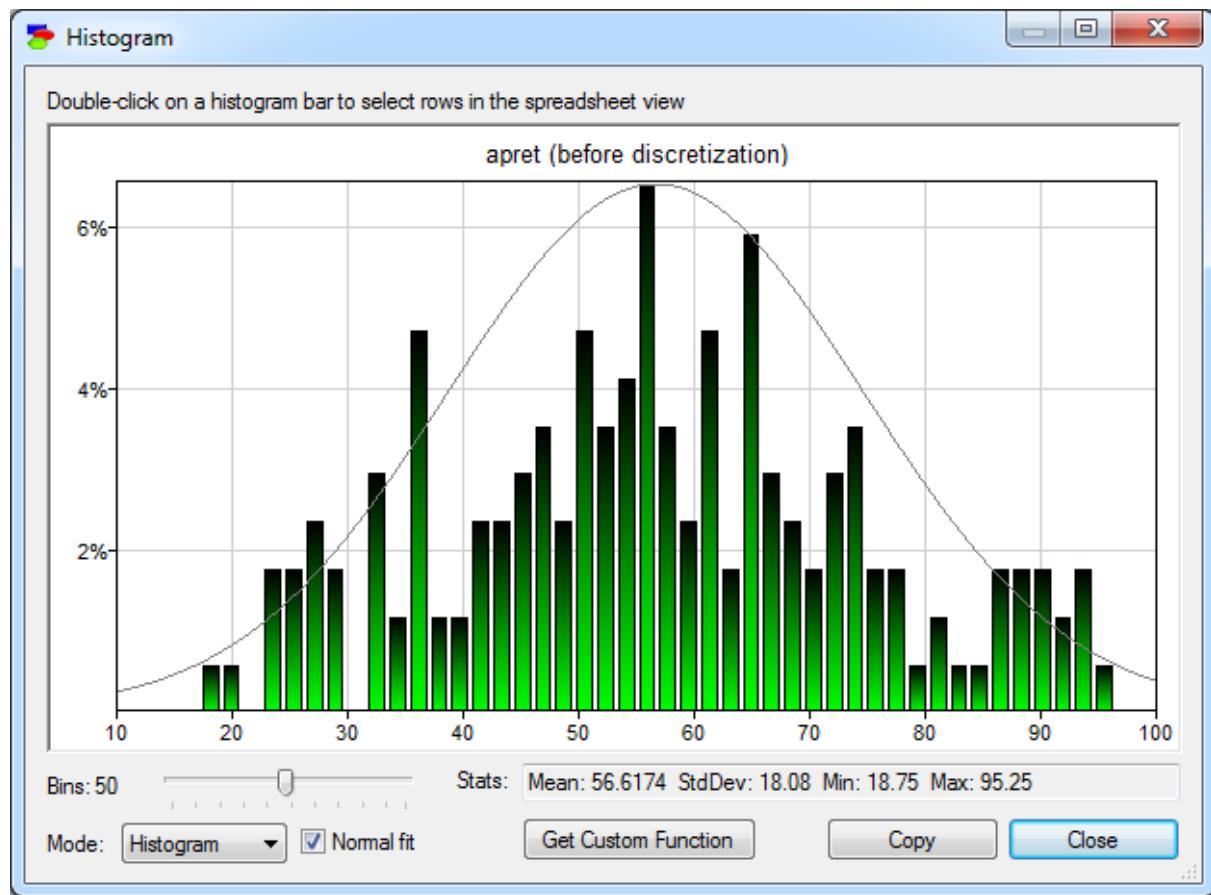
In both, the *Basic statistics* and *Correlation matrix* tabs, scrolling the mouse wheel forward and backward with *CTRL* key pressed zooms in and out, respectively. Also, in both tabs, the *Copy* button places the contents of the grid on the clipboard. It can be pasted (as text) to other text editors.

Histogram

To see the distribution of the values in a column on a histogram graph select [Data](#)³⁵¹ *Histogram* after selecting a column or, at least, placing the cursor inside one of the columns.



GeNle 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. GeNle 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:

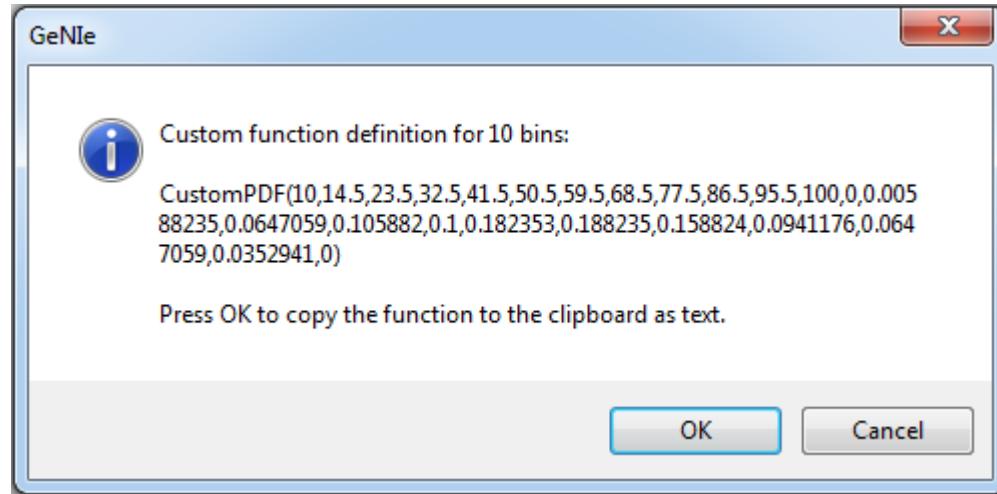


You can double-click a bar on the histogram to select all data rows that correspond to this bar. Histogram is always drawn of the selected rows (if no rows are selected, the histogram is of all rows). You can drill down the data by double-clicking on bars and then displaying histograms of the data that 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, 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

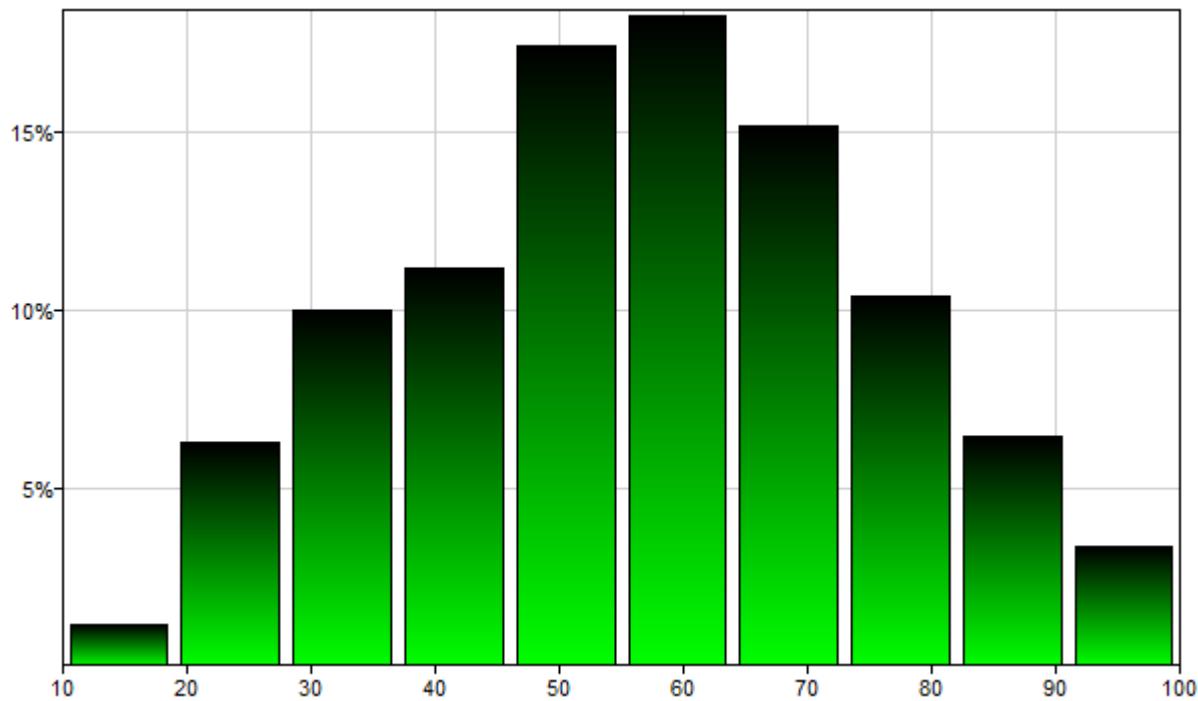
Get Custom Function button allows for learning a continuous probability distribution from the data describing the variable. For the first histogram above (with 10 bins), *Get Custom Function* opens the following dialog



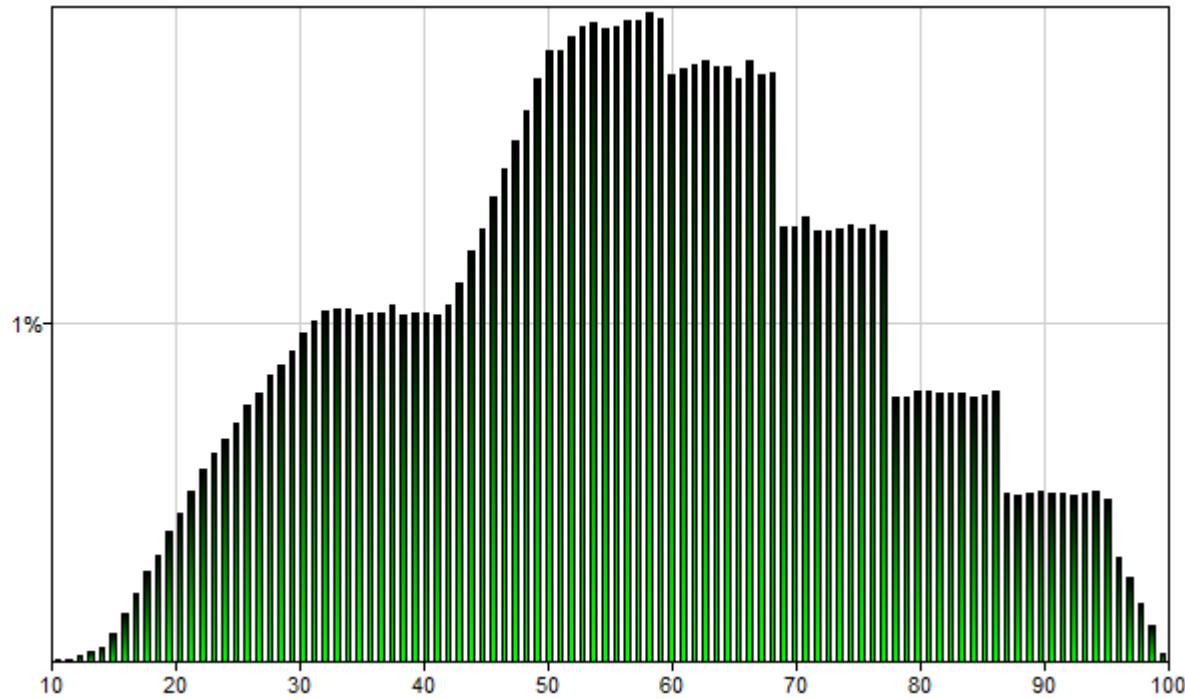
Pressing *OK* puts the following text on the Clipboard:

```
CustomPDF(10,14.5,23.5,32.5,41.5,50.5,59.5,68.5,77.5,86.5,95.5,100,  
0,0.00588235,0.0647059,0.105882,0.1,0.182353,0.188235,0.158824,0.0941176,0.064  
7059,0.0352941,0)
```

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.



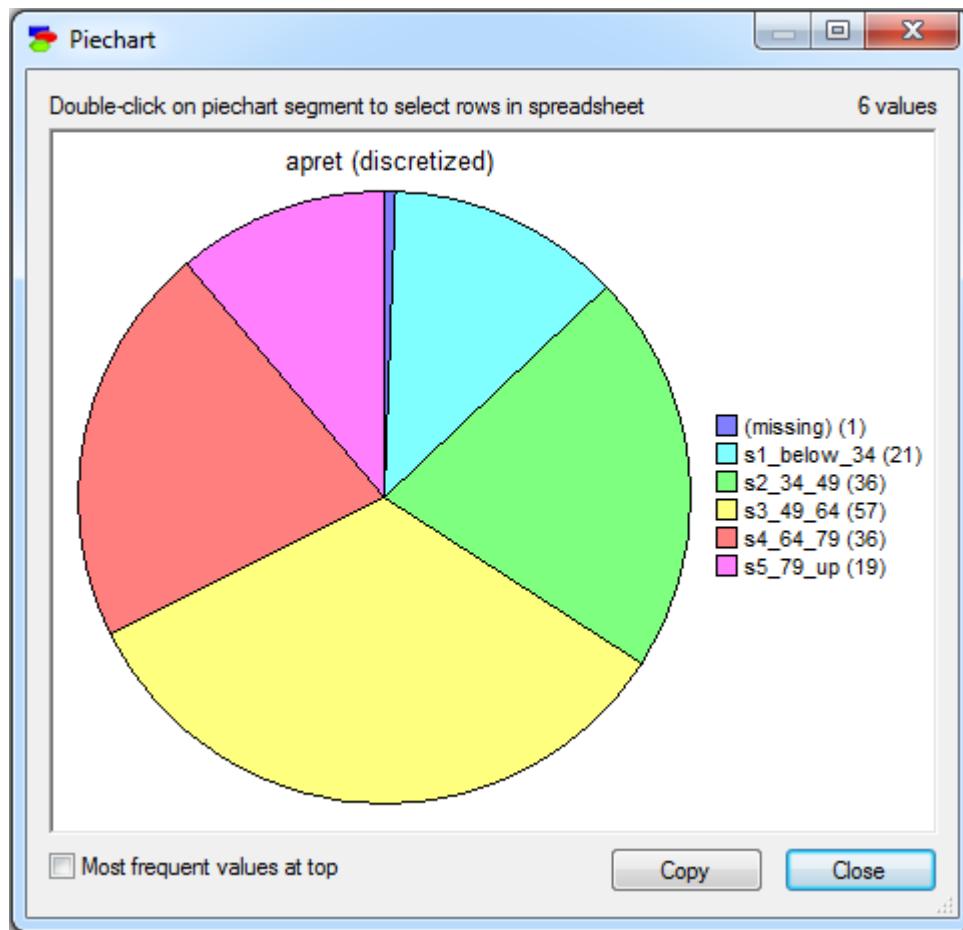
Please note that the shape of this histogram resembles to some degree the original histogram. The same histogram displayed with 100 bins looks as follows.



Please note that the shape of this histogram typically does not show steps but rather smoother transitions.

Pie chart

To see the distribution of the values in a discrete (or discretized) column on a piechart graph, select [Data](#)³⁵¹-Piechart. This will invoke the following dialog.



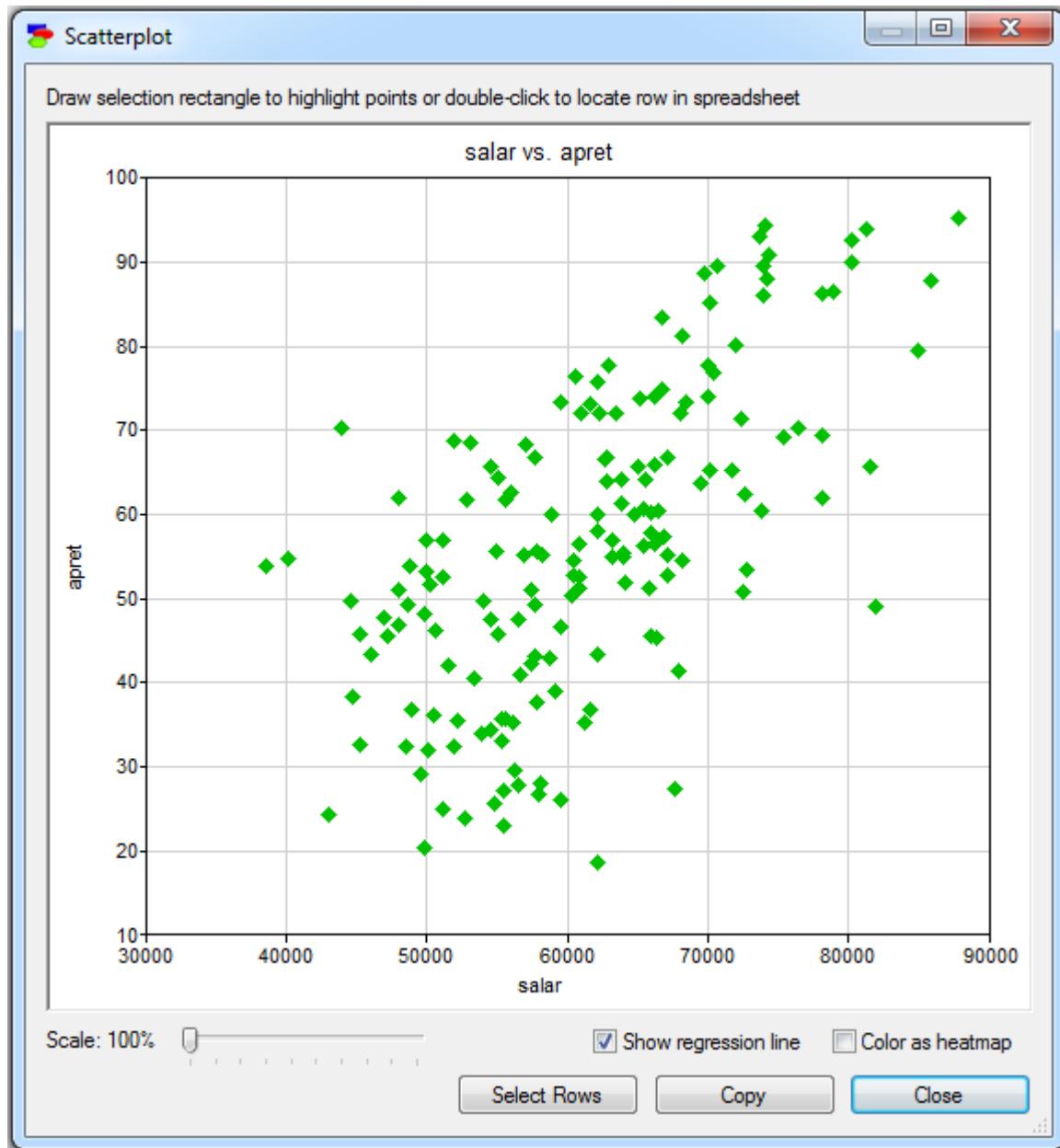
The *Most frequent values at top* check box sorts the states in the legend on the right-hand side from the most to the least frequent values. This is convenient when some of the states are very unlikely and hard to identify.

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, please use *Paste Special*.

Scatterplot

To see the joint distribution of two variables, select two (numerical) columns (by clicking on their IDs with *CTRL* key pressed) and select [Data³⁵¹-Scatterplot](#).



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. *Scale* allows for focusing on parts of the graph. Individual points on the graph can be selected using the mouse. Selected points can be selected in the *Data Grid View* by clicking on a corresponding button. The scatterplot can be copied (for a later paste into a program like Word) by clicking on the *Copy* button.

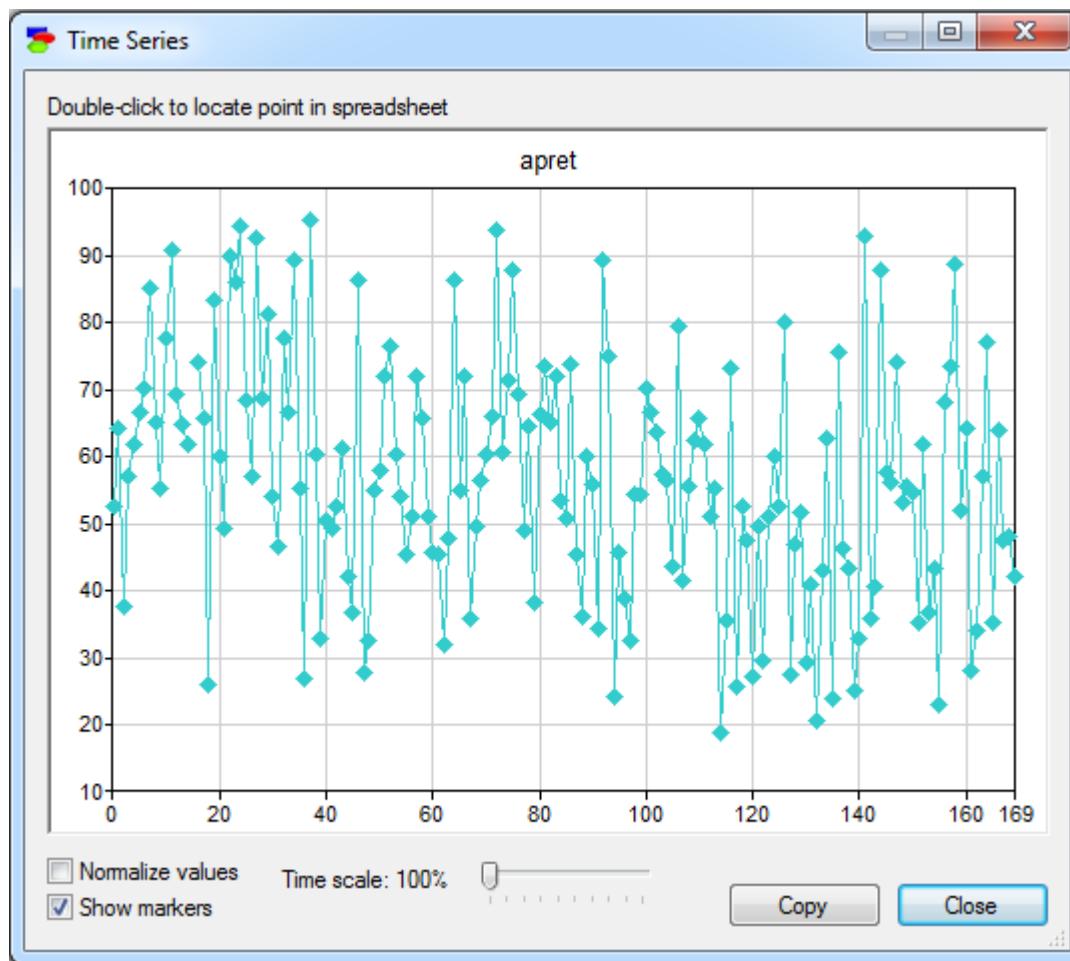
Scaling the *Scatterplot* (*Scale* slider) helps in distinguishing points that appear close on the plot but in reality are distinct.

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.

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 that shows. To paste it into an external program as an image, please use *Paste Special*.

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](#)³⁵¹-*Time Series*.

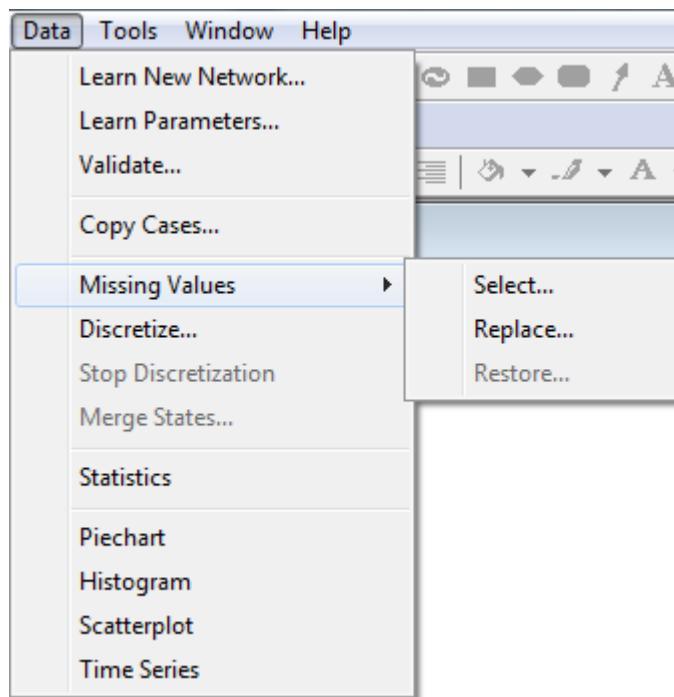


To show markers corresponding to the data points, select the *Show markers* check box. To normalize the values in the data so as the highest and lowest values take the highest and lowest points on the plot, select *Normalize values* check box. Time scale slider allows you to focus on parts of the plot in greater detail. Double click a point to locate the corresponding point in the data spreadsheet.

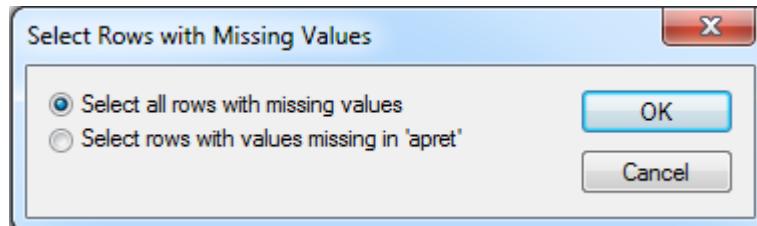
You can copy the image of the time series by clicking on the *Copy* button or right-clicking on it and selecting *Copy* from the pop-up menu that shows. To paste it into an external program as an image, please use *Paste Special*.

Missing Values

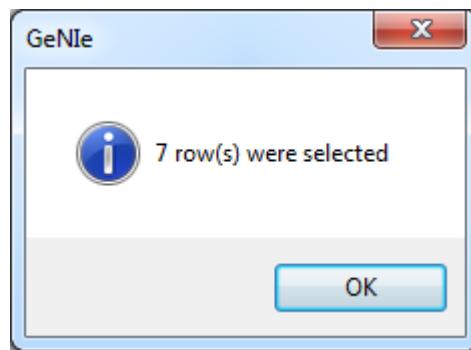
Missing values show as empty cells. To select rows that contain missing values (for instance, for deletion) select [Data](#)³⁵¹-*Missing Values-Select...*



You will be given two options: (1) *Select all rows containing missing values* and (2) *Select rows with values missing only in 'apret'* (the currently selected column):



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.

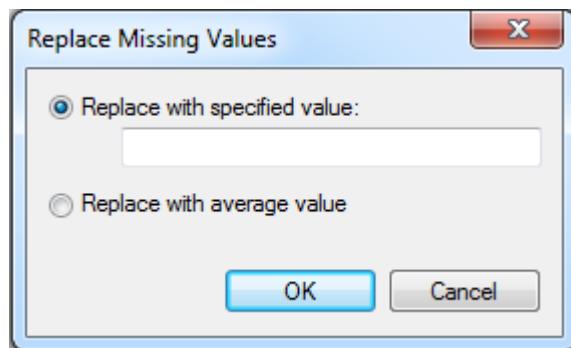


Selected rows will be highlighted in the *Data Grid View*:

	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	

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...](#) [351]. 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.



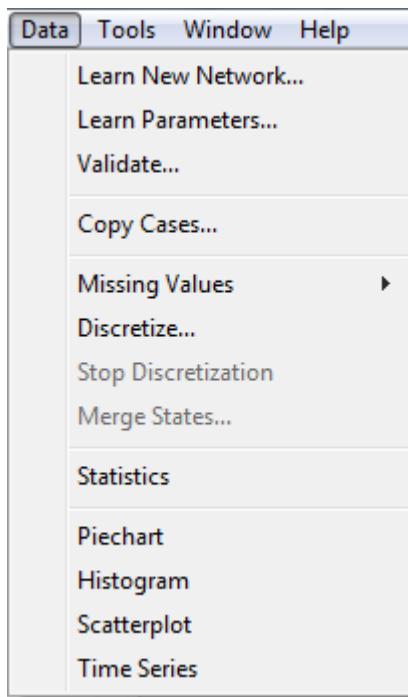
The replaced values will be distinguished with a red color, like on the screen shot below (the missing values were replaced in column *top10* with the value 9999):

	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

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](#)³⁵¹-Missing Values-Restore... The inserted values will be removed from the data grid.

Discretization

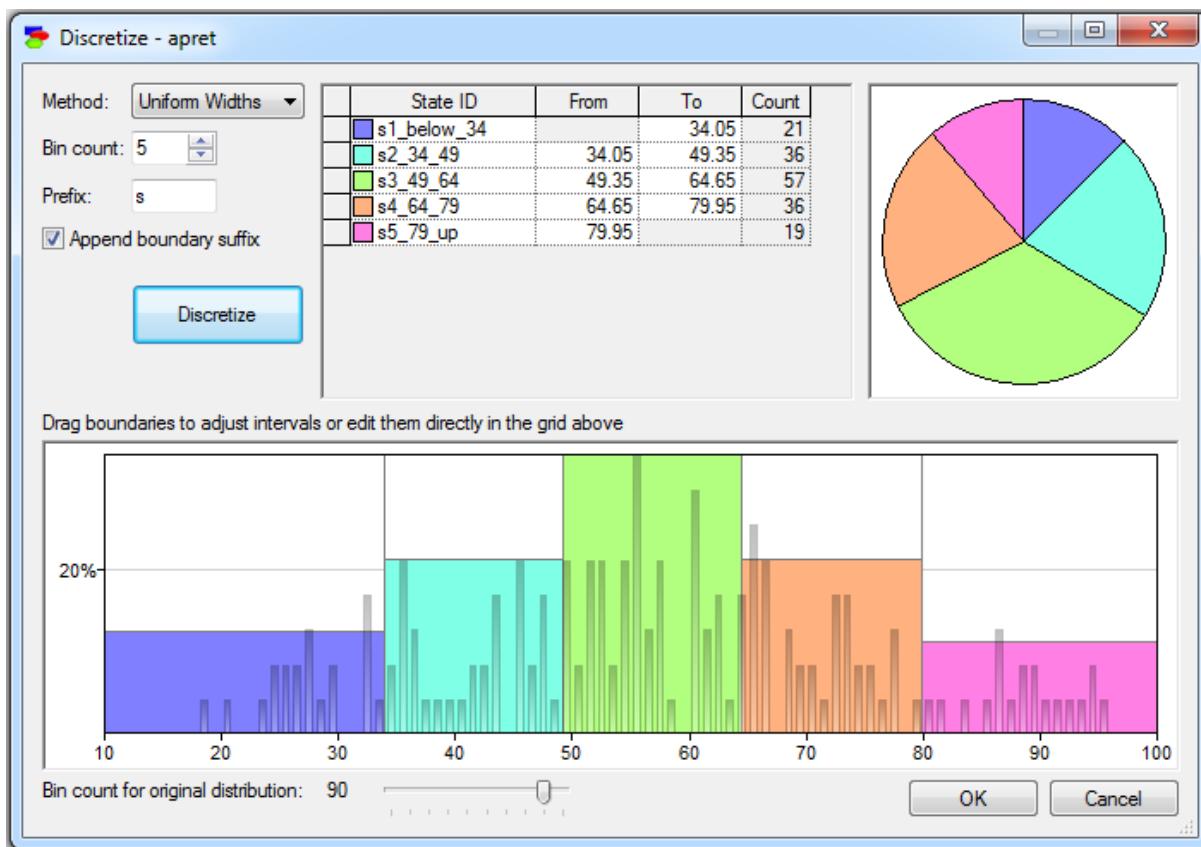
GeNle offers a powerful interface for interactive discretization of continuous variables. To invoke the discretization interface, select [Data³⁵¹-Discretize...](#)



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 GeNle 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 not have the literature reference handy but here is a sketch of the algorithm implemented in GeNle:

```
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
```

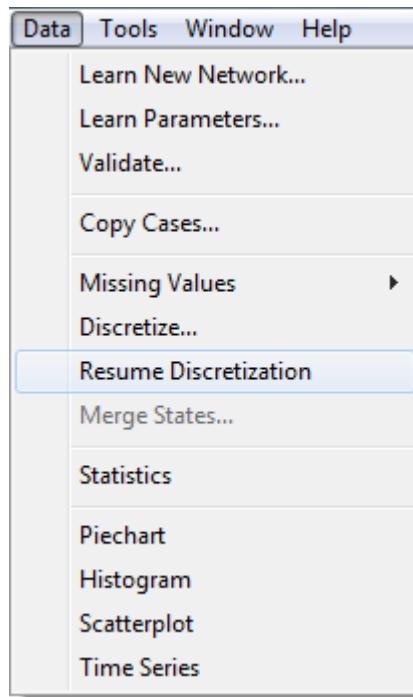
Discretization takes place once you press the *Discretize* button. The interface displays the distribution of records among the new intervals (as a pie chart) and a probability mass function with the histogram of the original continuous data in the background. The colors of the intervals correspond to 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.



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](#) ³⁵¹-*Stop Discretization* option from the main menu.

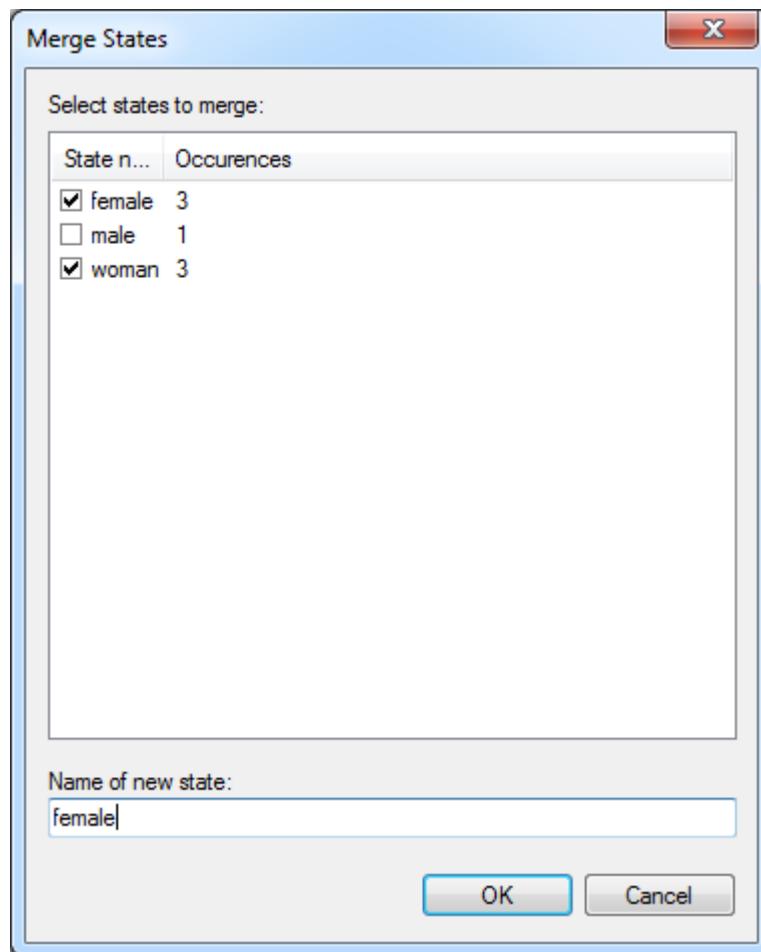
	spend	apret	top10	rejr	tstsc	pacc	strat	etc
▶	9855	s3_49_64	15	29.474	65.063	36.887	12.6	6
	10527	s3_49_64	36	22.309	71.063	30.97	12.8	6
	7904	s2_34_49	26	25.853	60.75	41.985	20.3	5
	6601	s3_49_64	23	11.296	67.188	40.289	17.5	5
	7251	s3_49_64	17	22.635	56.25	46.78	18.1	4
	6967	s4_64_79	40	9.718	65.625	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
	0110	10.04	00	00.00	01.005	00.101	11.00	0

You can resume a stopped discretization at any time by selecting [Data](#)³⁵¹-Resume Discretization. The values will be restored to the last successful discretization.



Merging States

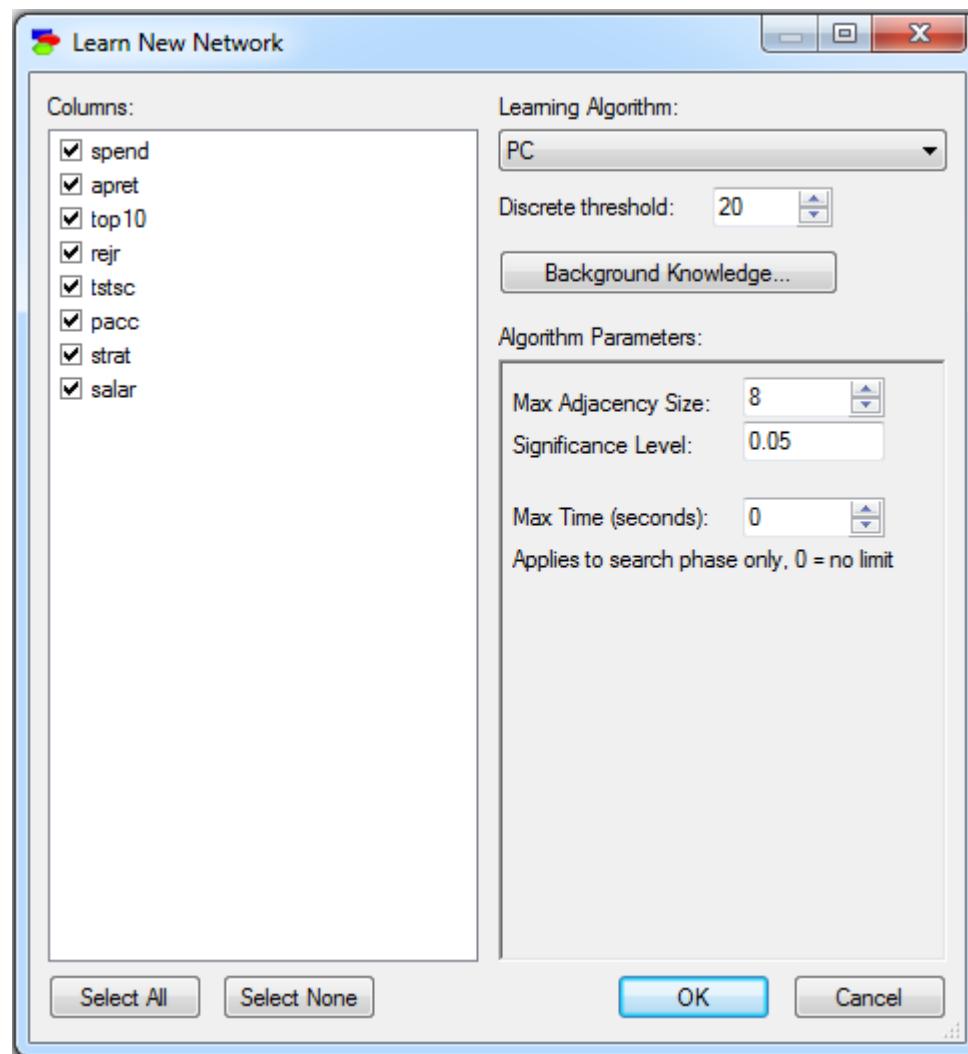
Sometime, through an error in the data collection or encoding, two or more states may denote the same value. For example, *female* and *woman* may all refer to the same value. GeNle allows to merge such states into one through the *Merge States...* functionality from the *Data Menu*. To merge states of a variable, select the column that represents it and select *Data*³⁵¹—*Merge States...*



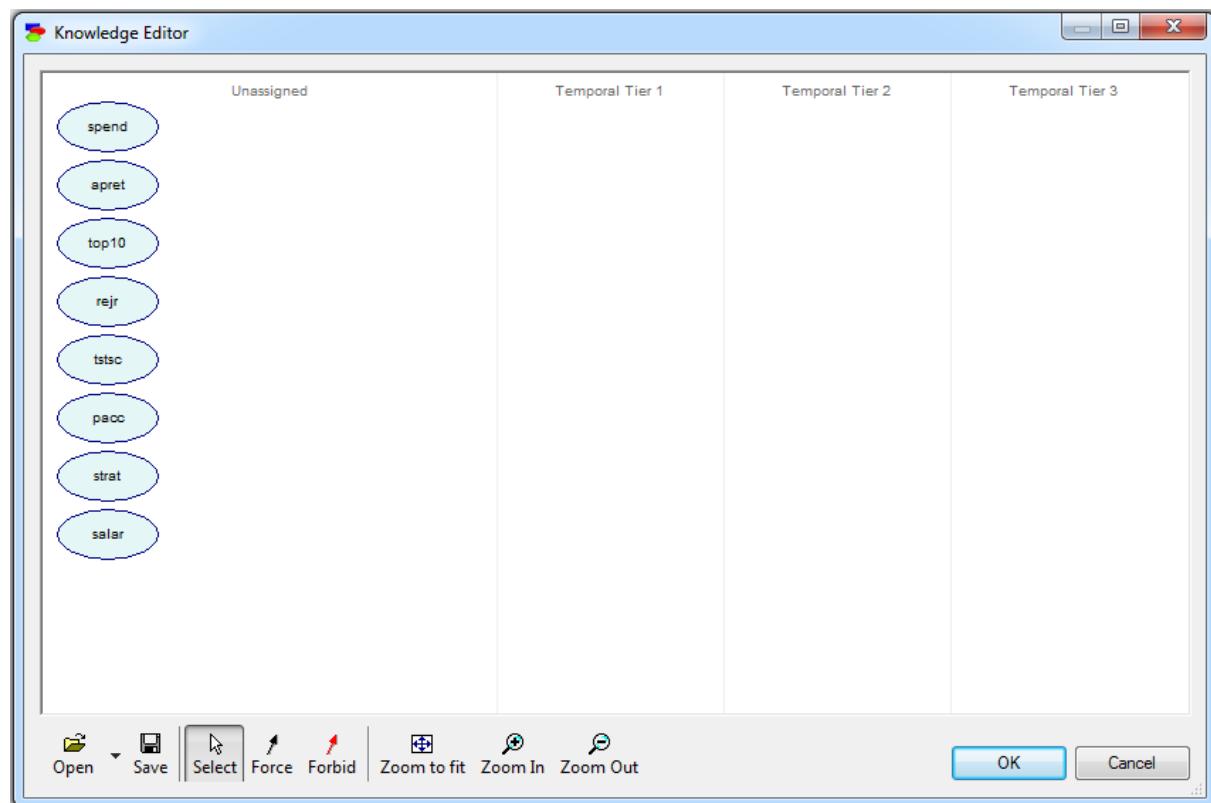
Select the states that you want to merge together and provide the name for the resulting state. You can see that GeNle gives you information about the number of occurrences of each of the states of the selected variable. The effect of this operation will be that all six states (*female* and *woman*) will be changed into *female*. The *Merge States* command can be viewed a convenient shortcut for a series of *Replace All* commands.

6.5.4 Knowledge editor

To invoke the *Knowledge Editor* dialog, click on the *Background Knowledge* button in the *Learn New Network* dialog.



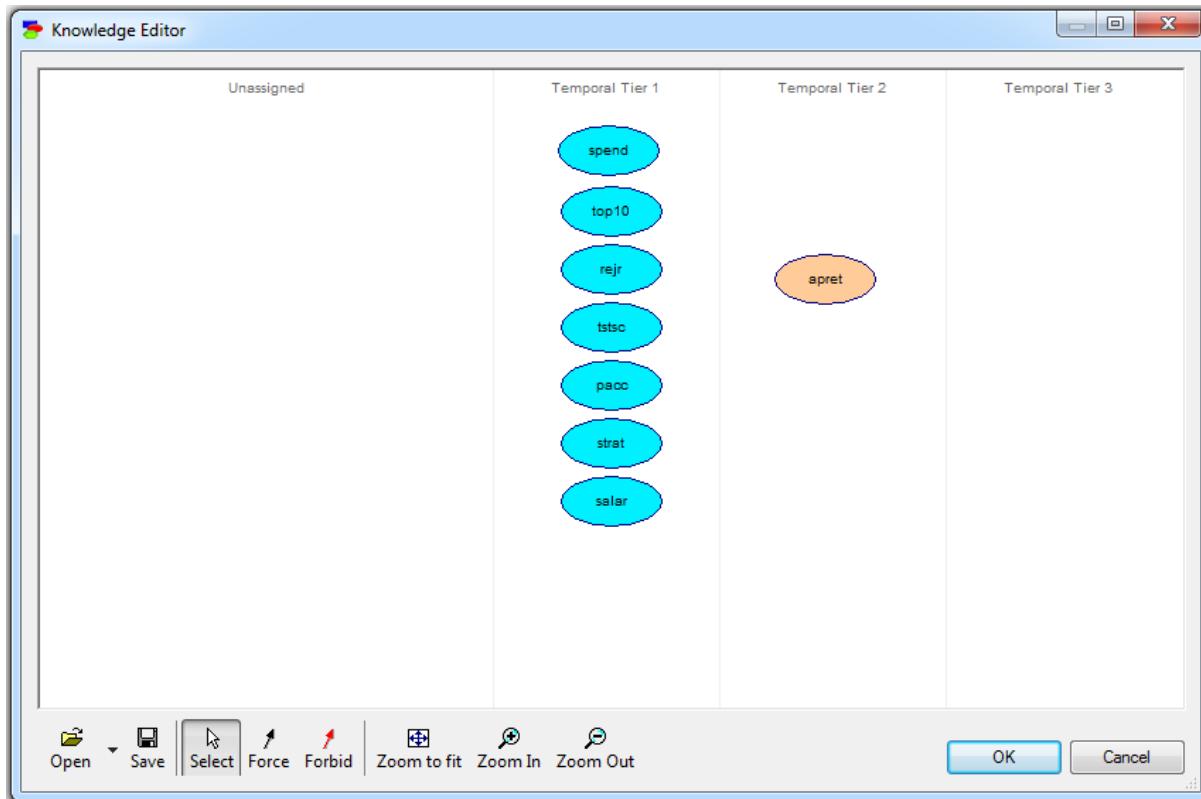
The *Knowledge Editor* dialog allows for entering domain knowledge that will aid in the structure learning.



The *Knowledge Editor* dialog allows you to:

- force arcs (these arcs are guaranteed to be in the learned structure)
- forbid arcs (these arcs are guaranteed to be absent in the learned structure), and
- assign variables to temporal tiers

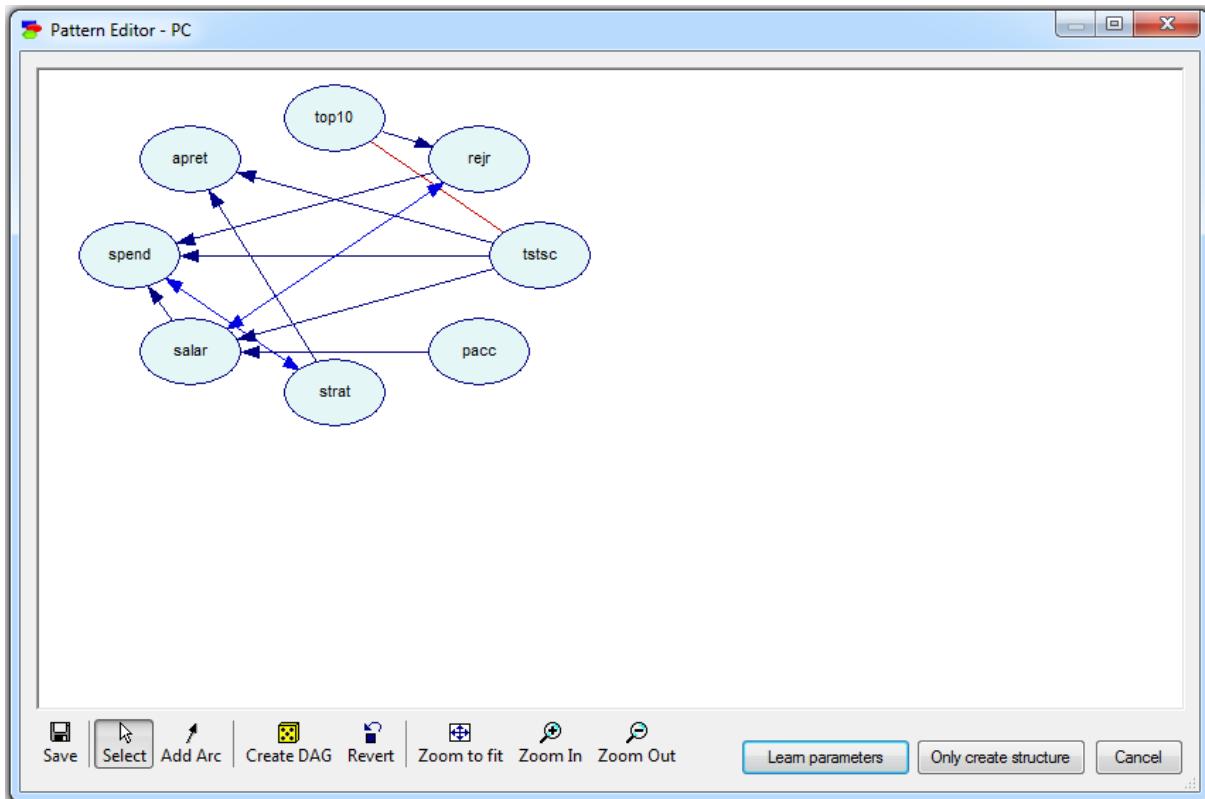
Forced and forbidden arcs can be drawn by clicking on the origin and dragging the arc to the destination node, similarly to the way we draw arcs in the *Graph View*. The rule about temporal tiers is simple: In the resulting network, there will be no arcs from nodes in higher tiers to nodes in lower tiers. 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:



It is possible to save and retrieve later knowledge entered by means of the *Knowledge Editor*. Knowledge files have suffix *.gkno for GeNIE KNOWledge.

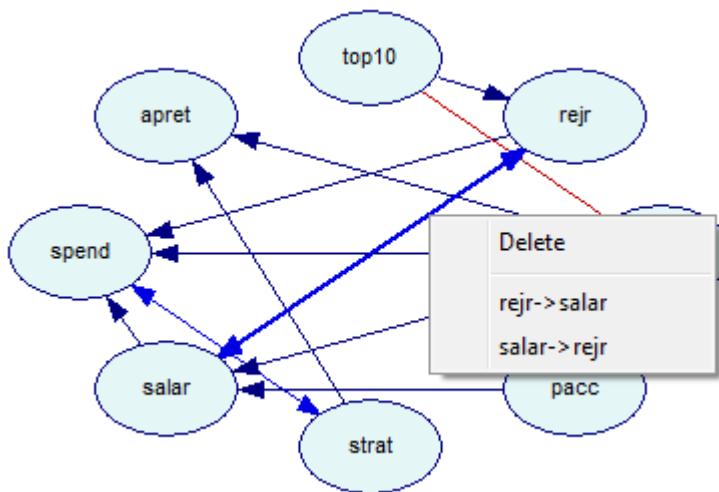
6.5.5 Pattern editor

The output of some of the structure learning algorithms, e.g., the *PC* algorithm, is the *Pattern Editor* dialog.



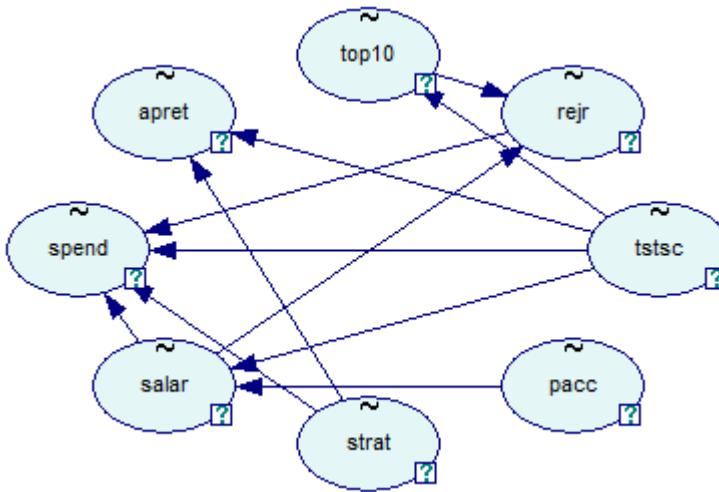
The *Pattern Editor* dialog displays the equivalence class of [Bayesian networks](#)⁴⁵ that has been learned from data. There are three types of arcs in the *Pattern Editor* dialog: undirected, directed, and doubled-headed arcs. Their meaning is: an arc that could not be oriented based purely on data, a directed arc, and an arc 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 causally related to *apret*: *strat* and *tstsc*.

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:



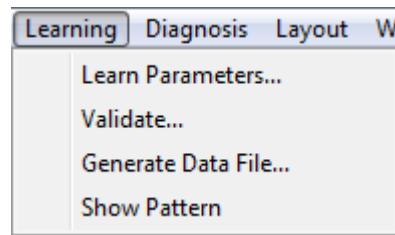
or by clicking on the *Create DAG* () button, which picks the direction of each undirected arc randomly to yield an acyclic directed graph.

Clicking on the *Learn parameters* () button yields a fully parametrized Bayesian network:



Which is typically the final product of any learning algorithm.

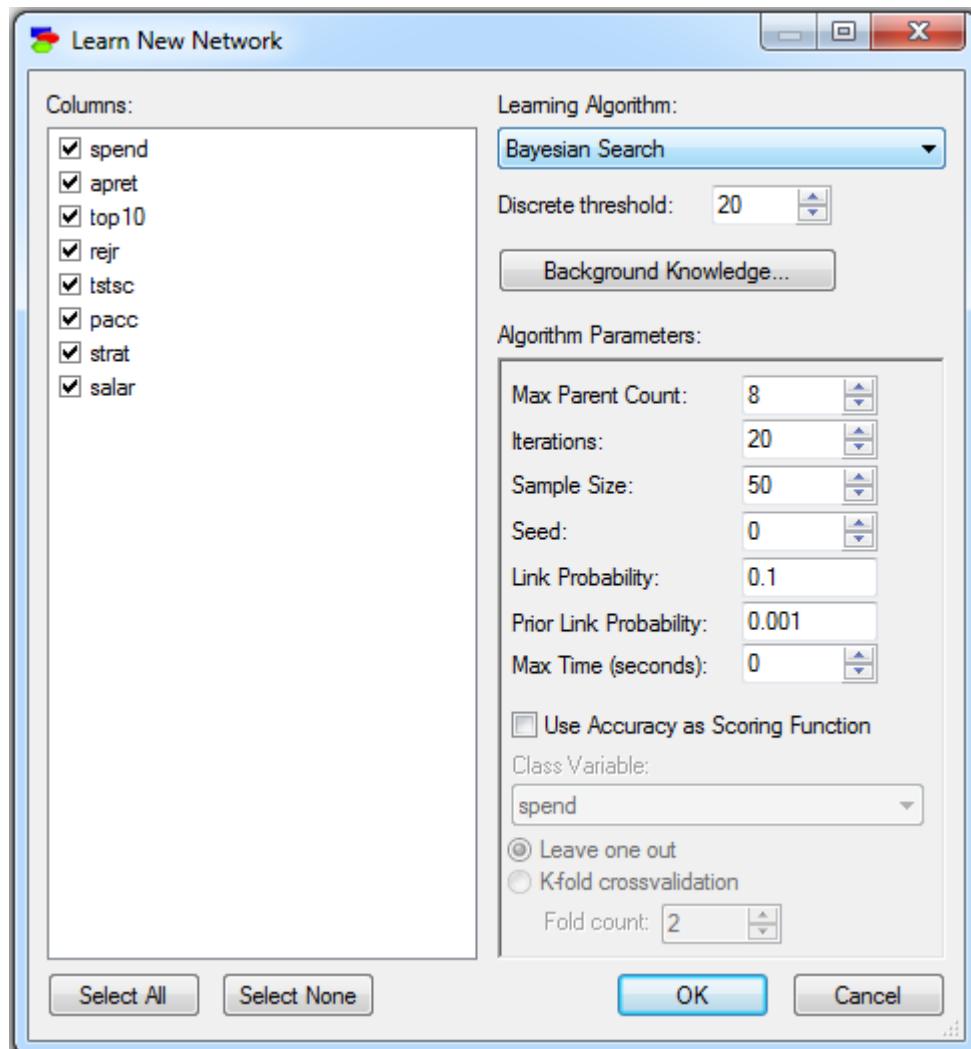
It is possible to come back from an acyclic directed graph of a Bayesian network to the *Pattern Editor* dialog by choosing *Show Pattern* from the *Learning* menu:



6.5.6 Structural learning

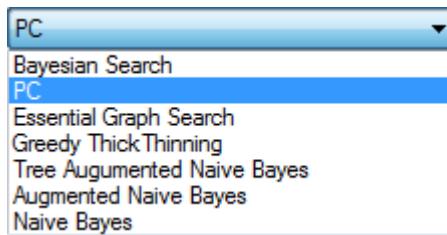
6.5.6.1 Introduction

To invoke the structure learning dialog, select [Data](#)³⁵¹-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.



Check boxes next to variable names on the left-hand side allow us for selecting those variables that will take part in learning. The dialog box above shows that all variables in the data file will be used in structure learning.

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.

The *Background knowledge* button invokes the *Knowledge Editor* dialog, which allows for entering domain knowledge that will aid in the 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 completely satisfactory, the user is encouraged to improve node layout manually.

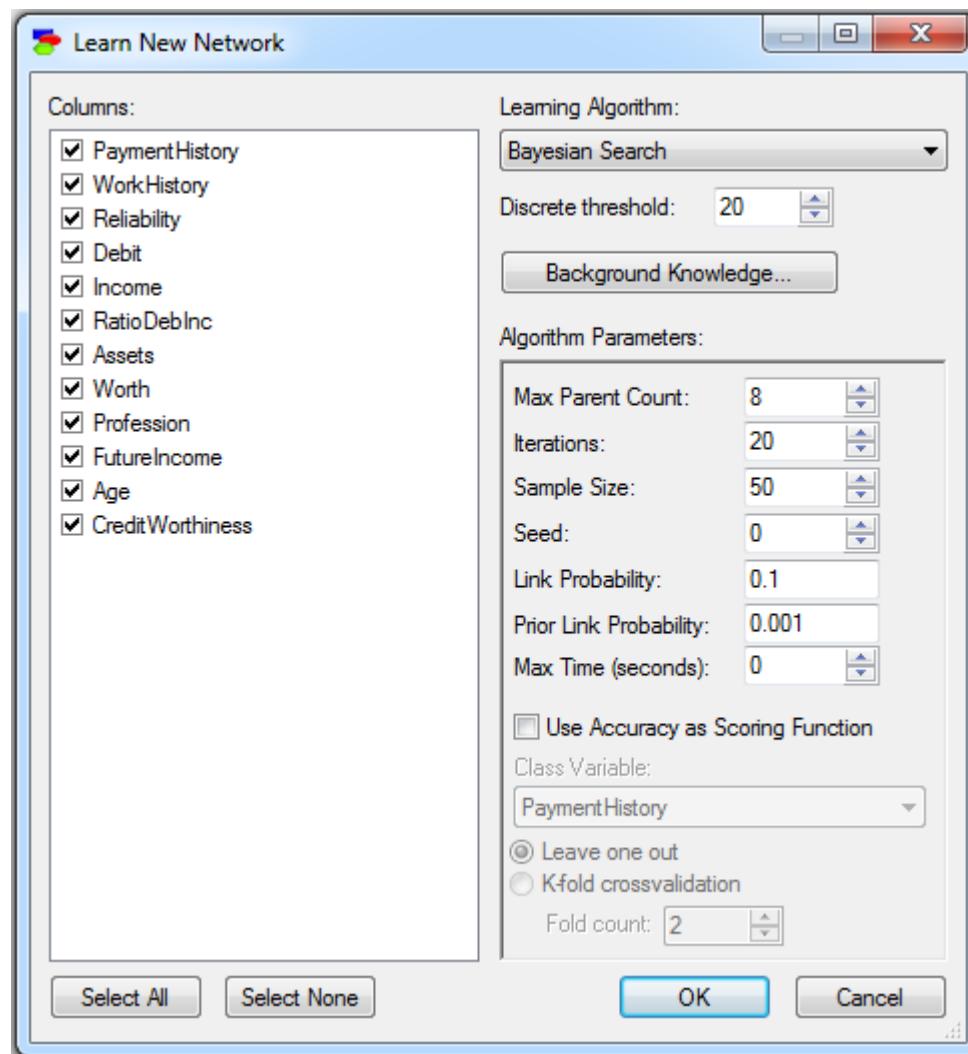
Each of the structure learning algorithms assumes that the variables are categorical. In addition, the PC algorithm allows for learning the structure with continuous variables, when the joint distribution over them is multivariate normal. None of the algorithms allows for learning from a mixture of discrete and continuous variables, so if there is a discrete variable in the learning set, it is necessary to discretize all continuous variables.

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 GeNle will complain if they are included in the learning set.

6.5.6.2 Bayesian Search

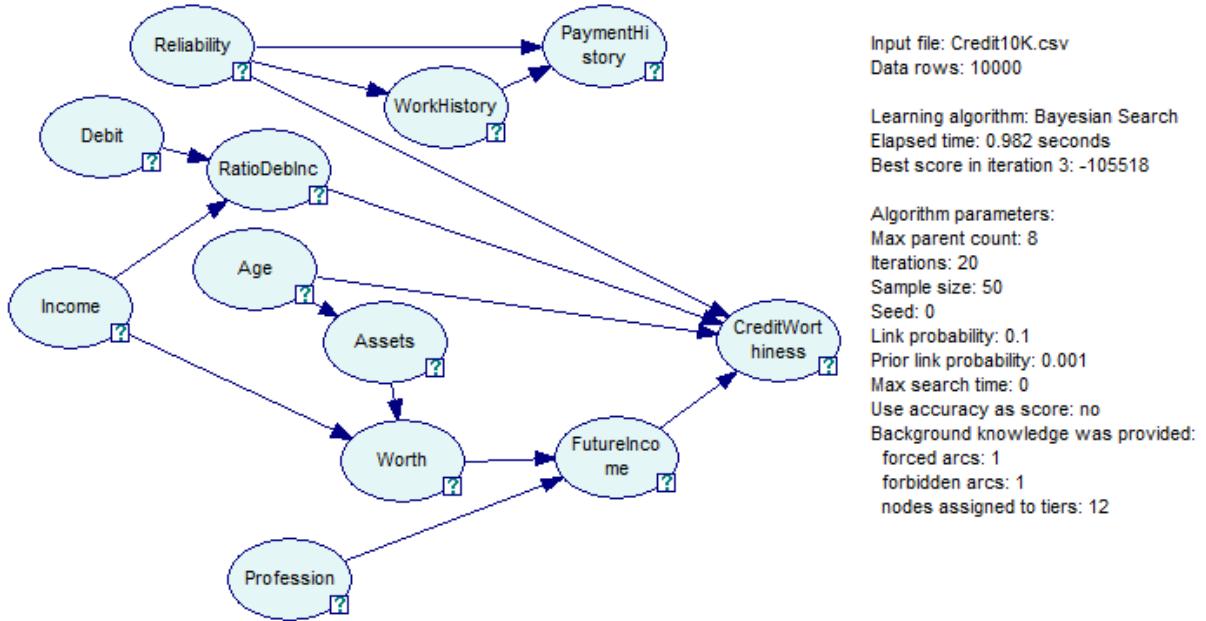
The *Bayesian Search* structure learning algorithm is one of the earliest and the most popular algorithms used. It was introduced by (Cooper & Herkovitz, 1992) and was refined somewhat by (Heckerman, 1995). It follows essentially a hill climbing procedure (guided by a scoring heuristic) with random restarts. 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.
- *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: $\log \text{Posterior score} = \log \text{marginal likelihood}$ (i.e., the *BDeu*) + $|\text{parents}| * \log(\text{pll}) + (|\text{nodes}| - |\text{parents}| - 1) * \log(1 - \text{pll})$.
- *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).

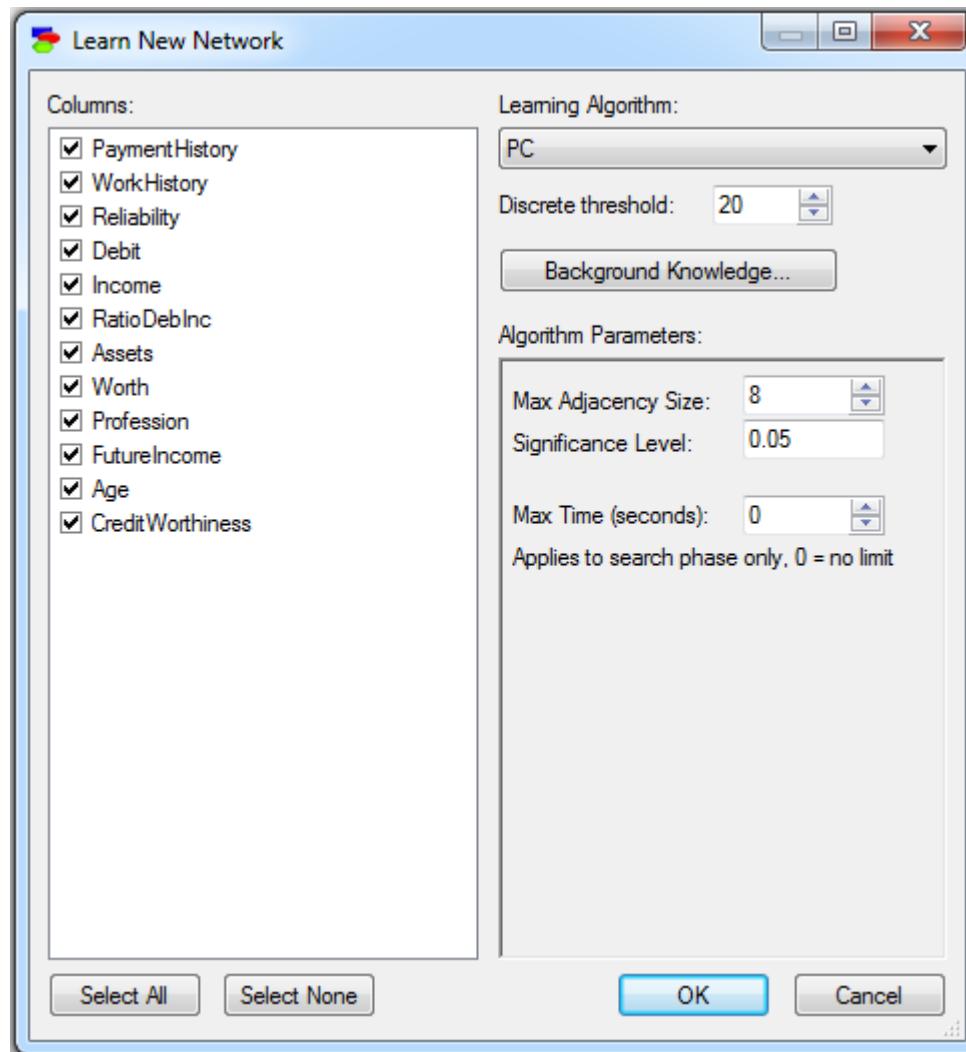
The algorithm produces an acyclic directed graph that gives the maximum score. The score is proportional to the probability of the data given the structure, which, assuming that we assign the same prior probability to any structure, is proportional to the probability of the structure given the data. 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.

6.5.6.3 PC

The *PC* structure learning algorithm is one of the earliest and the most popular algorithms, introduced by (Spirtes et al., 1993). Here is the *Learn New Network* dialog for the *PC* algorithm:

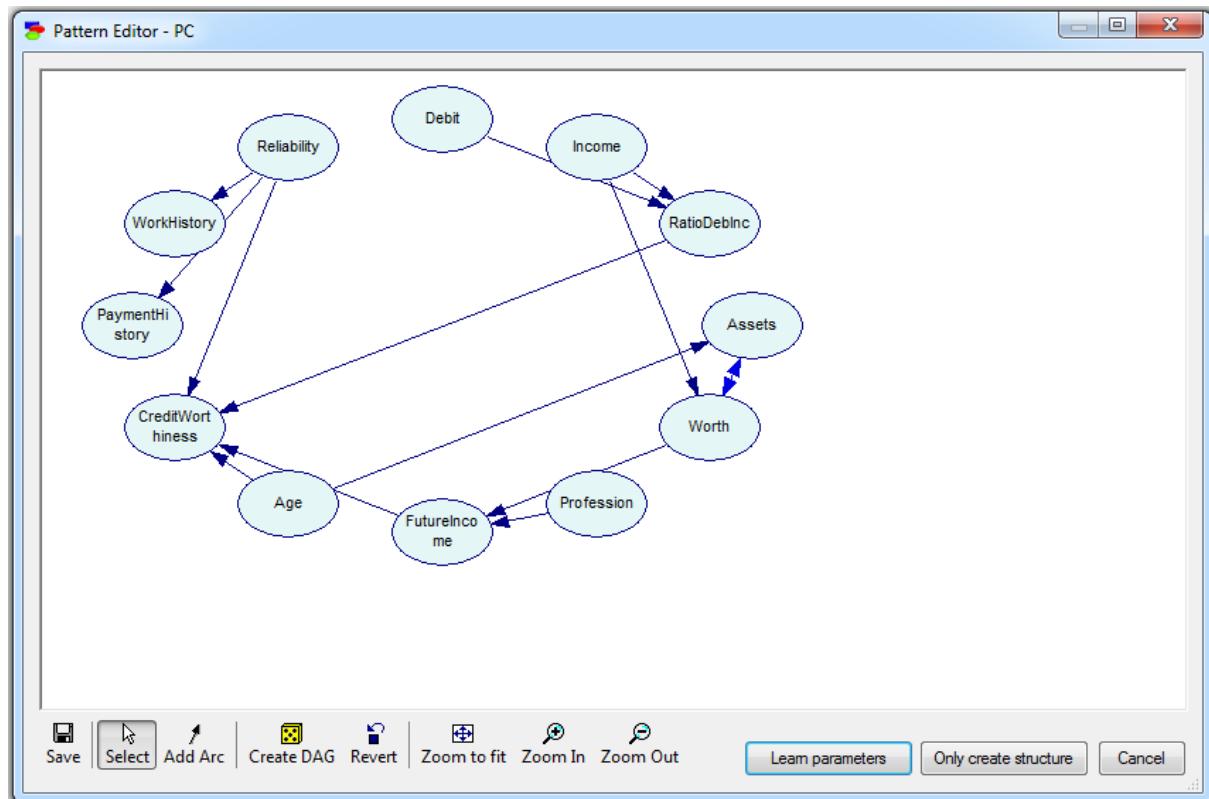


The PC algorithm has three parameters:

- *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.

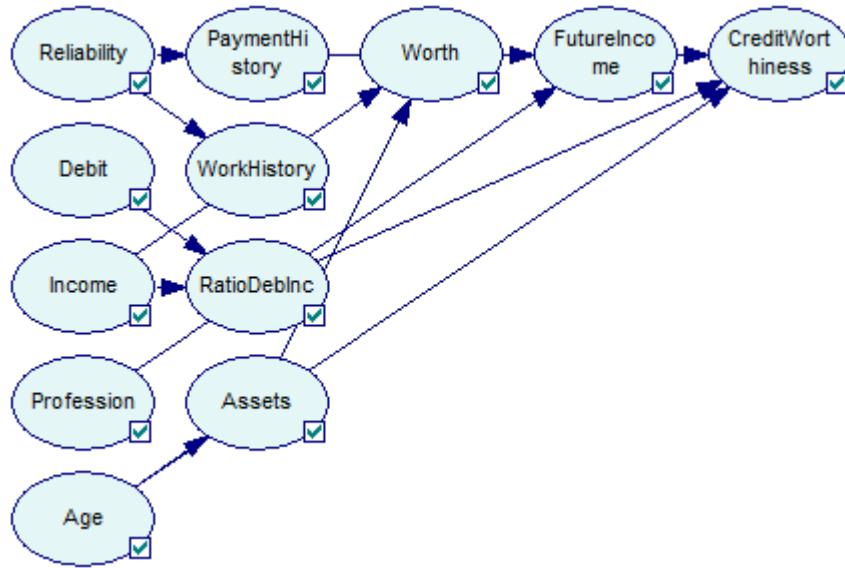
Pressing *OK* and then *OK* in the *Learn New Network* dialog starts the algorithm, which ends in the *Pattern Editor* dialog.



There is one double-edged arc in the pattern (between *Assets* and *Worth*). Clicking

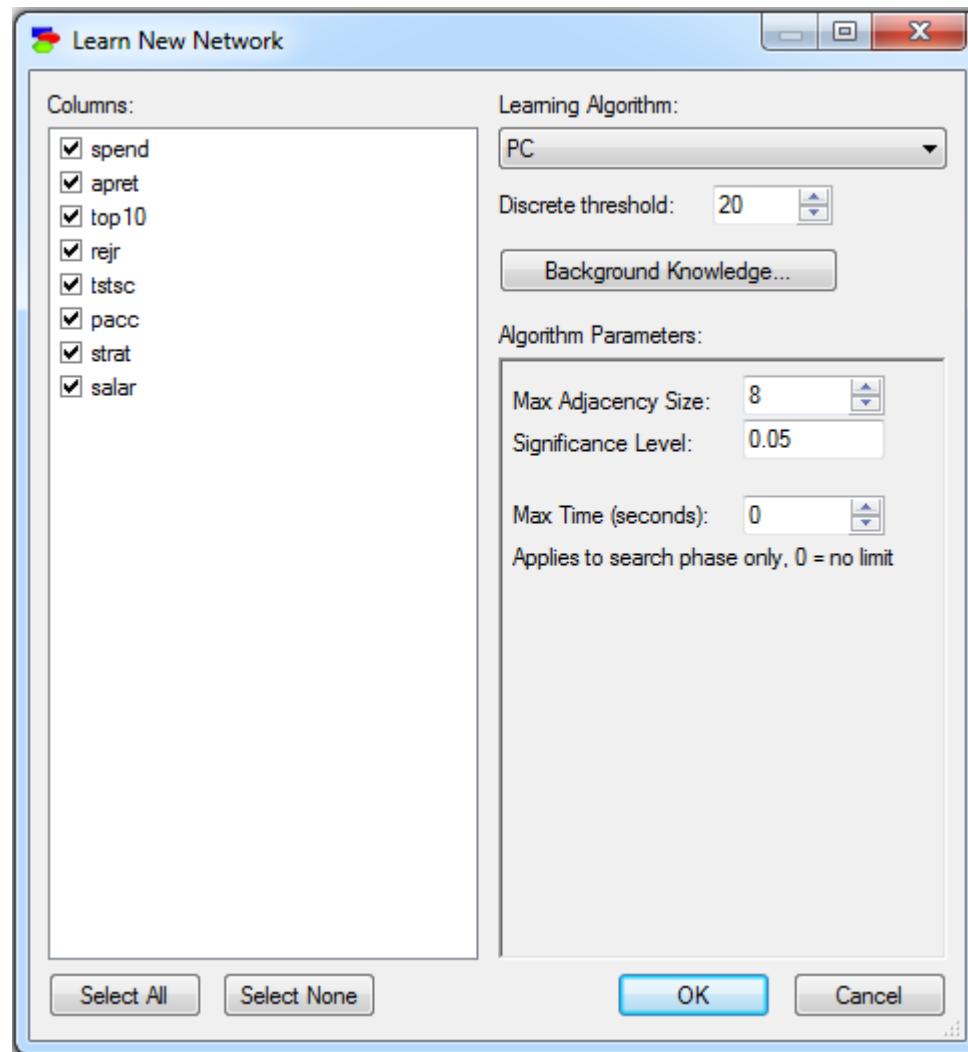


on the *Create DAG* (*Create DAG*) button, which picks the direction of each undirected arc randomly to yield an acyclic directed graph, clicking on the *Learn parameters* (*Learn parameters*) button, and then running a graph layout algorithm (*Parent Ordering* from left to right) results in the following Bayesian network:

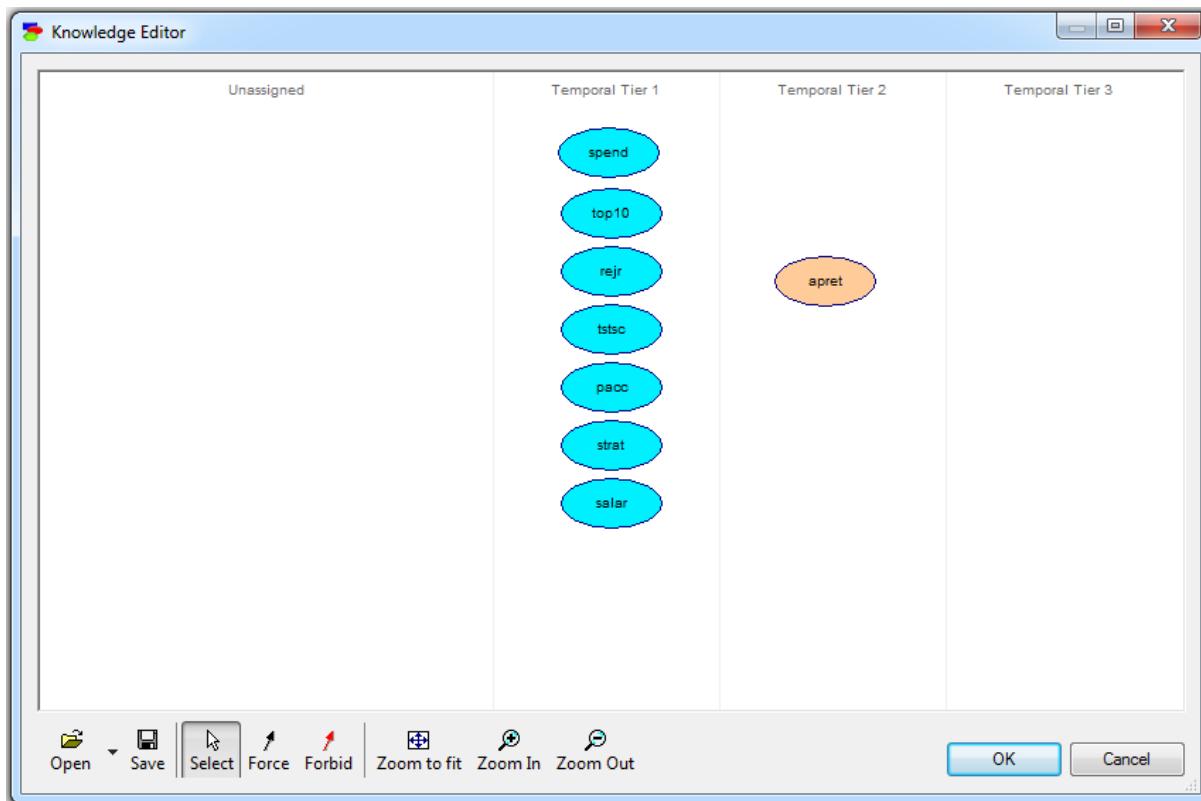


The PC algorithm is the only structure learning algorithm in GeNle that allows for continuous data. The data have to fulfill reasonably the assumption that they come from multivariate normal distribution. To verify this assumption, please check that histograms of every variable are close to the Normal 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.

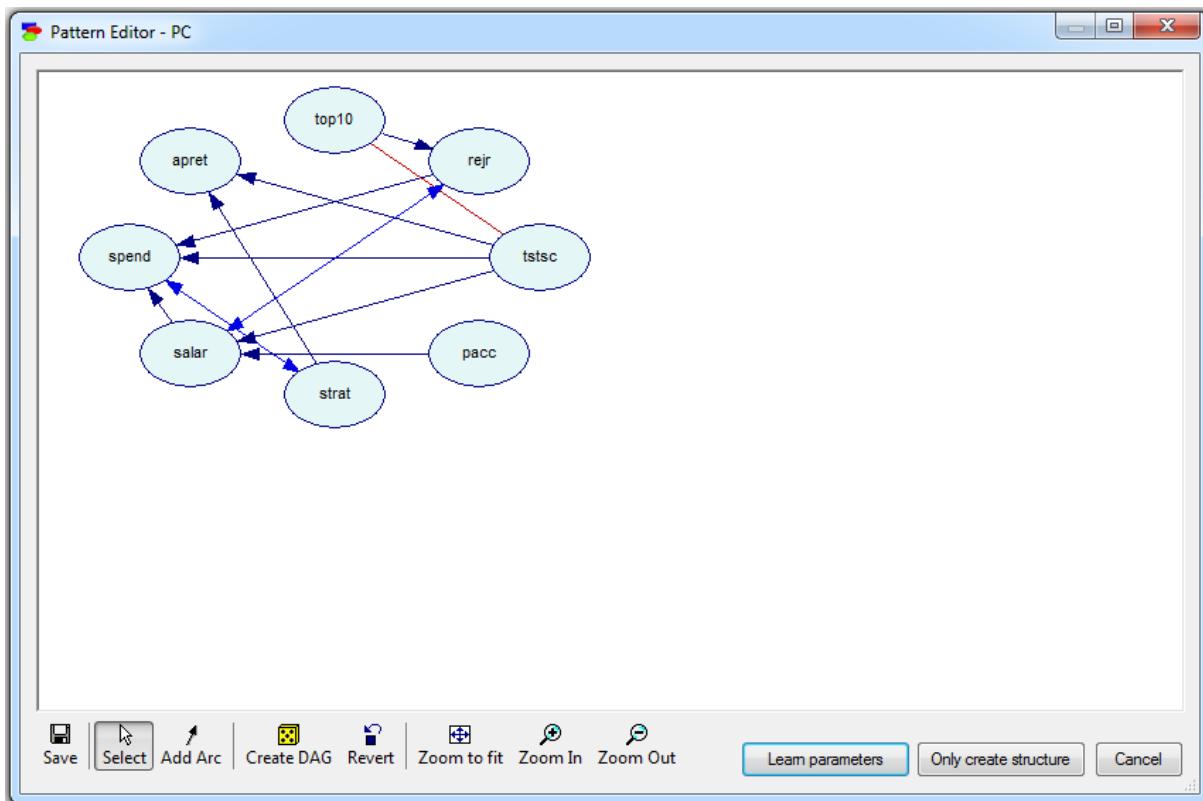
Let us demonstrate the working of the PC algorithm on a continuous data set retention.txt, included among the example data sets.



Here is the suggested specification of prior knowledge about the interactions among the variables in the data set:



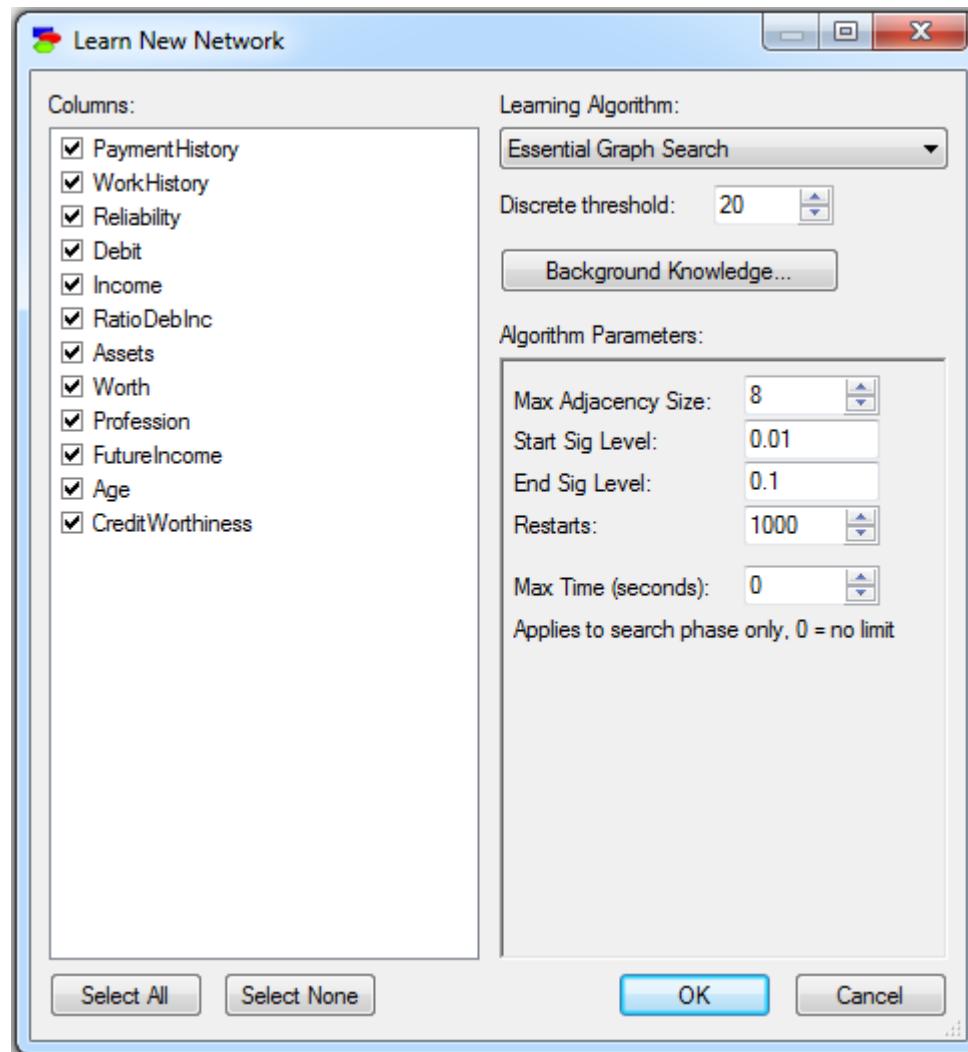
Pressing *OK* and then *OK* in the *Learn New Network* dialog starts the algorithm, which ends in the *Pattern Editor* dialog.



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.

6.5.6.4 Essential Graph Search

The *Essential Graph Search* structure learning algorithm is based on a combination of the constraint-based search (with its prominent representative being the *PC* algorithm) and *Bayesian Search* approach. The algorithm, proposed in (Dash & Druzdzel., 1999), performs a search for essential graphs using the PC algorithm and scores the various essential graphs using the Bayesian search approach. Here is the *Learn New Network* dialog for the *Essential Graph Search* algorithm:

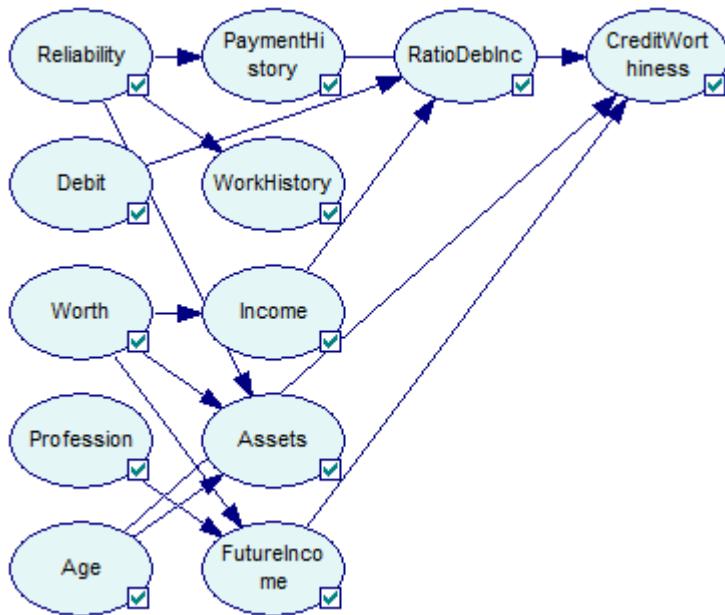


The *Essential Graph Search* algorithm has the following parameters:

- *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.
- *Start Sig Level* (default 0.01), which is the lower bound on the significance level for independence tests used in the first phase of the algorithm
- *End Sig Level* (default 0.1), which is the upper bound on the significance level for independence tests used in the first phase of the algorithm.

- *Restarts* (default 1000) 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 can be afforded it in terms of running time. Running the algorithm with default number of iterations first should give 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.
- *Max Time (seconds)* (default 0, which means no time limit) sets a limit on 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 terminate within a reasonable amount of time.

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.

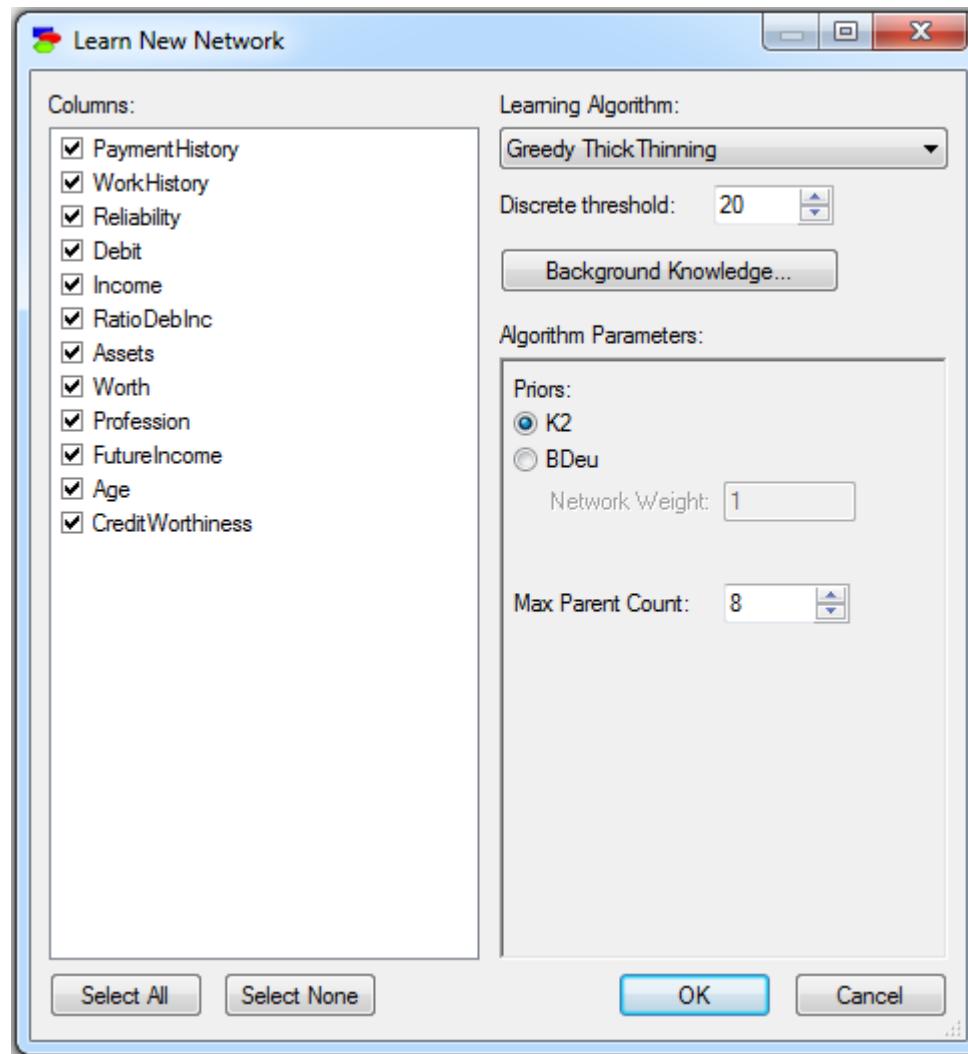


Because the *Essential Graph Search* algorithm produces an acyclic directed graph, it is a good idea to investigate the theoretical limits to what it can identify based on the data. To this effect, we advise the user to transform the acyclic directed graph of a Bayesian network to the *Pattern Editor* dialog.

6.5.6.5 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). Here is the *Learn New Network* dialog for the *Greedy Thick Thinning* algorithm:

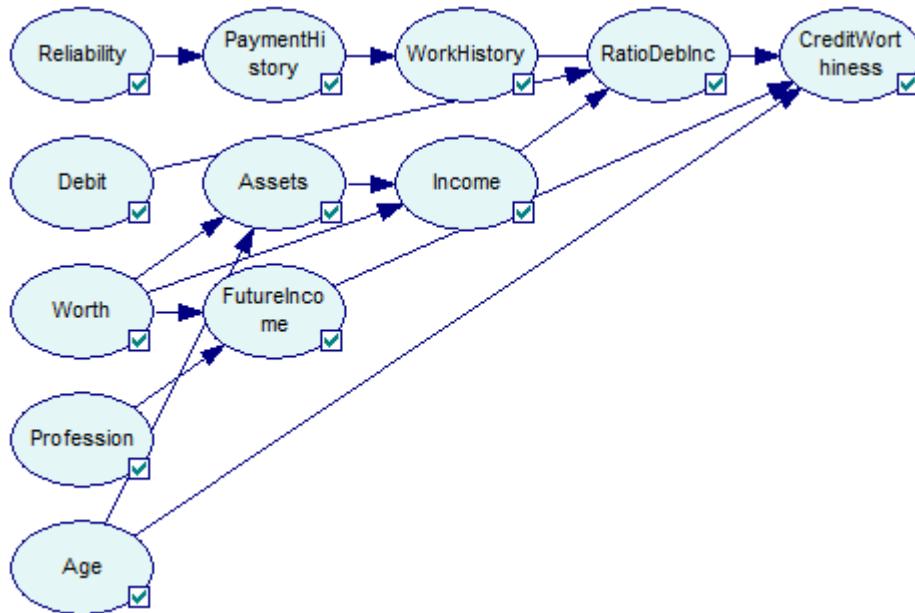


The *Greedy Thick Thinning* algorithm has the following parameters:

- *Priors* with two choices: *K2* (default) and *BDeu*. These are two popular priors used in Bayesian network scoring metrics introduced by (Cooper & Herskovitz, 1992) and (Buntine, 1991), respectively. A good comparison of these two priors can be found in (Kayaalp & Cooper, 2002). In case of the *BDeu* priors, an additional parameter is *Network Weight* (default 1).

- *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.

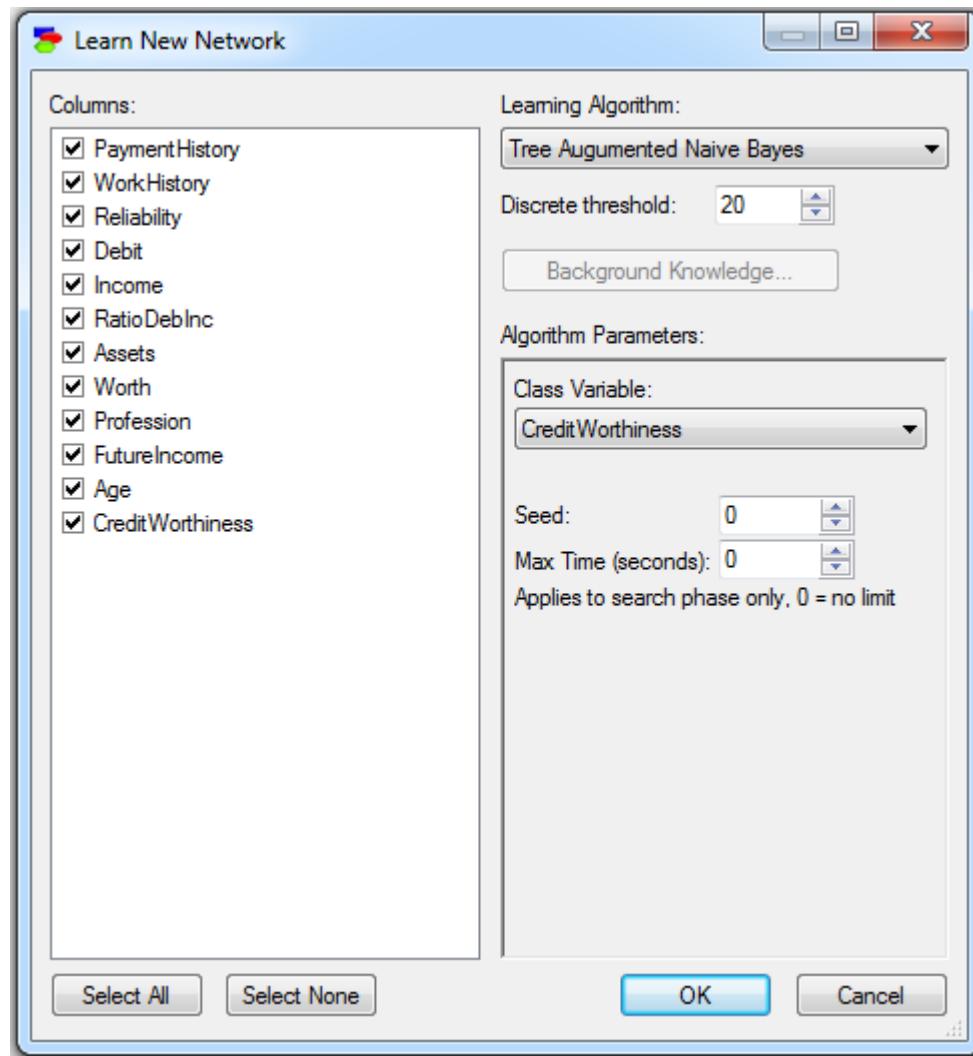


Because the *Essential Graph Search* algorithm produces an acyclic directed graph, it is a good idea to investigate the theoretical limits to what it can identify based on the data. To this effect, we advise the user to transform the acyclic directed graph of a Bayesian network to the *Pattern Editor* dialog.

6.5.6.6 Tree Augmented Naive Bayes

The *Tree Augmented Naive Bayes (TAN)* structure learning algorithm is a semi-naïve structure learning method based on the *Bayesian Search* approach, described and thoroughly evaluated in (Friedman et al., 1997). The *TAN* algorithm starts with a *Naïve 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 *Naïve 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:



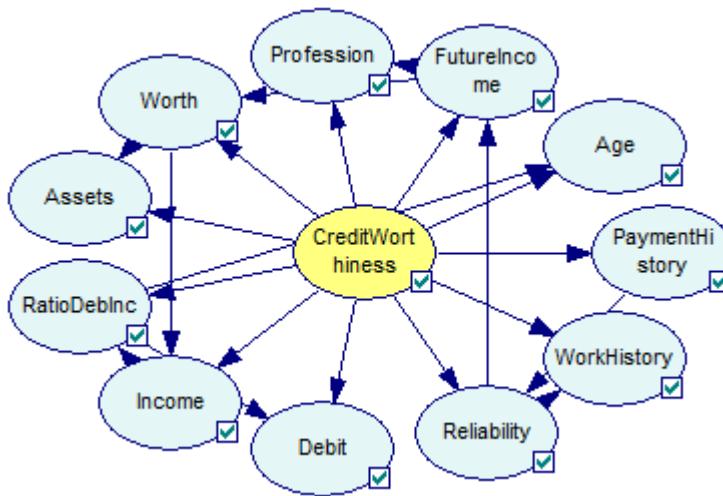
The *TAN* algorithm has the following parameters:

- *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.
- *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.

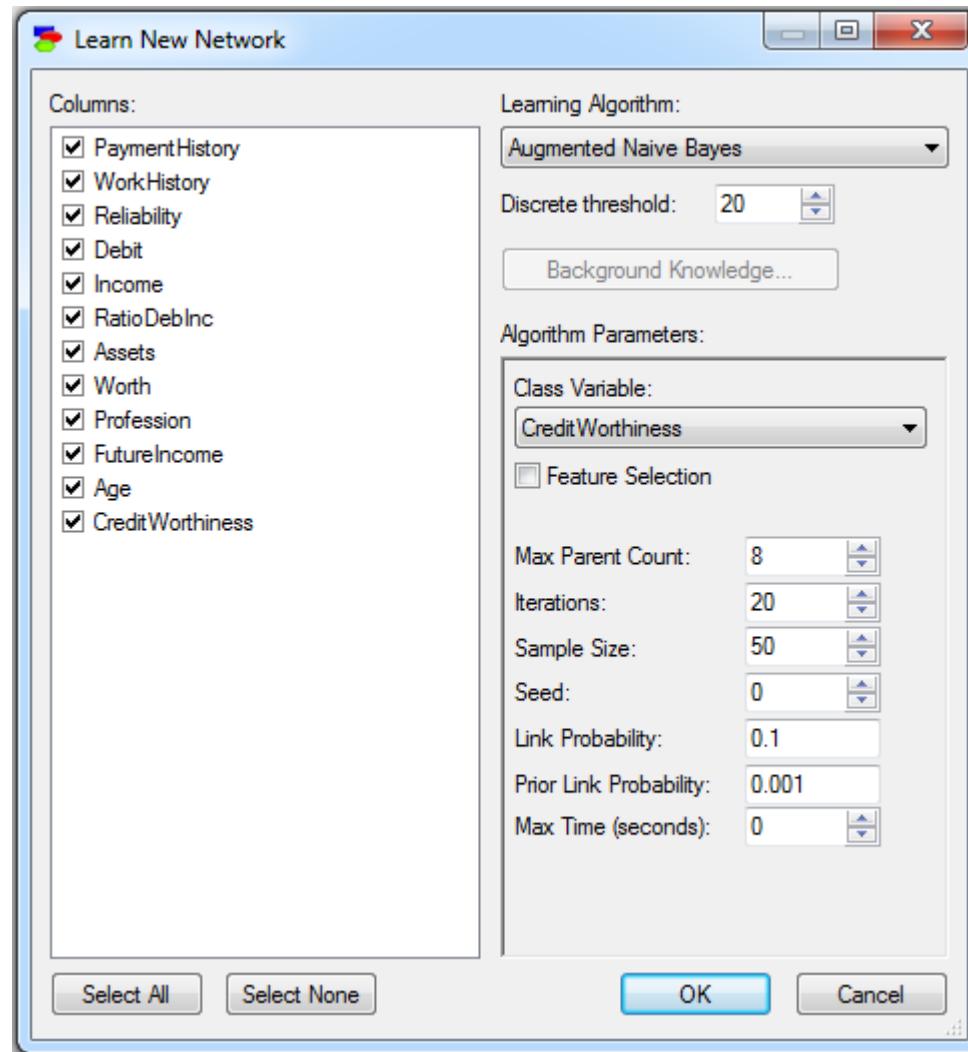
The algorithm produces an acyclic directed graph with the class variable being the parent of all the other (feature) variables and additional connections between the feature variables. The structure learned is one with the maximum score, similarly to other algorithms based on *Bayesian Search*. The score 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.



6.5.6.7 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 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 *Naïve Bayes*. Here is the *Learn New Network* dialog for the *ANB* algorithm:

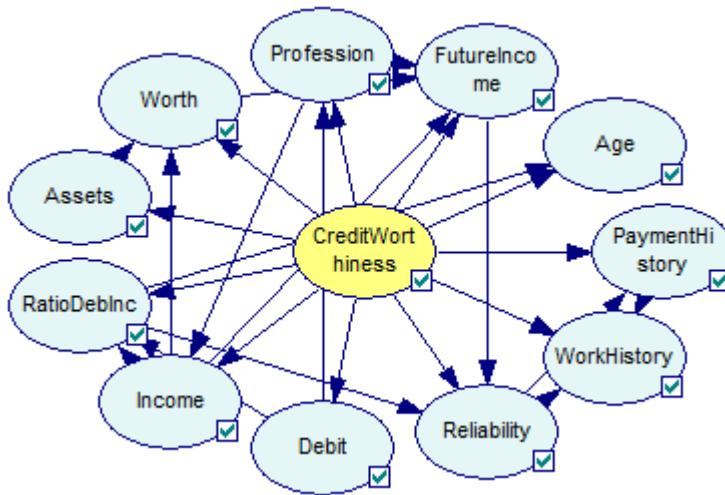


The *ANB* algorithm has a number of parameters, most of which mimic the parameters of the *Bayesian Search* algorithm that the *ANB* algorithm is based on:

- *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.
- *Feature Selection*, when checked, invokes an additional function that removes from the feature set those features that do not contribute enough to the classification.

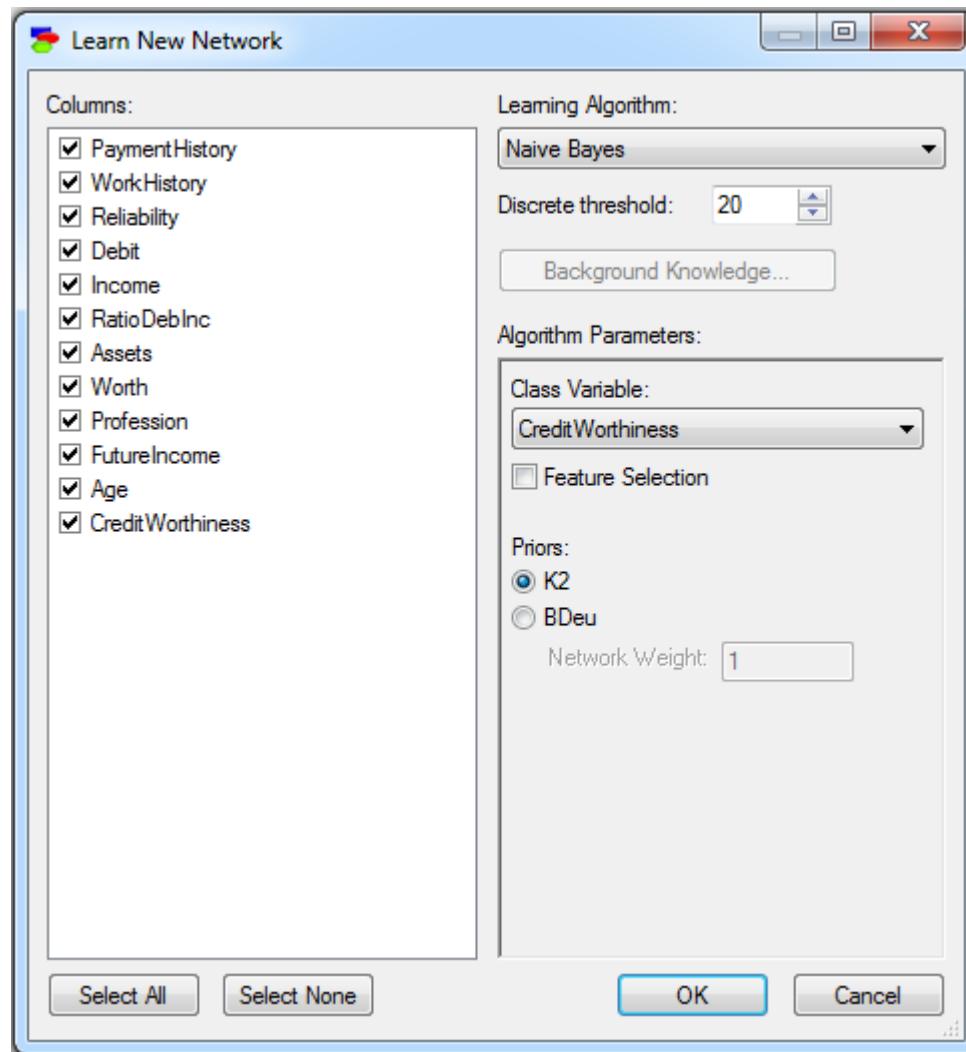
- *Max Parent Count* (default 8) limits the number of parents that a node can have. Because the size of conditional probability tables of a node grow exponentially in the number of the node's parents, it is a good idea to put a limit on the number of parents so that the construction of the network does not exhaust all available computer memory.
- *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)
- *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.



6.5.6.8 Naive Bayes

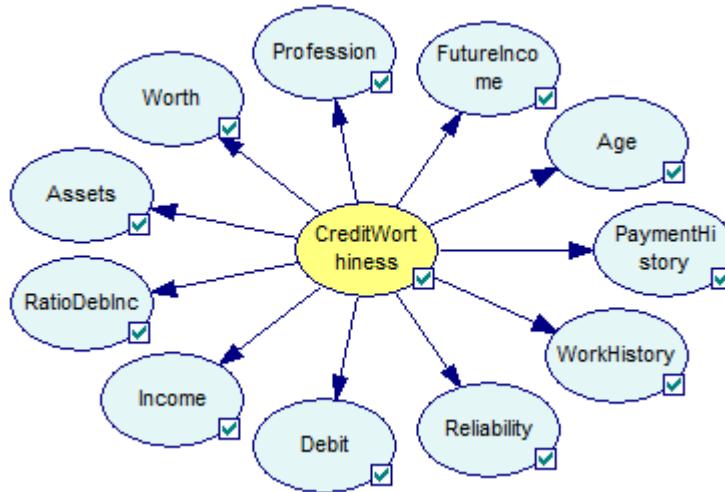
The *Naive Bayes* learning algorithm is a naive structure learning method that is included in the category of structure learning algorithms only because it creates a Bayesian network, including its structure and parameters directly from data. The structure of a Naive Bayes network is not learned but 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 or 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 the following parameters:

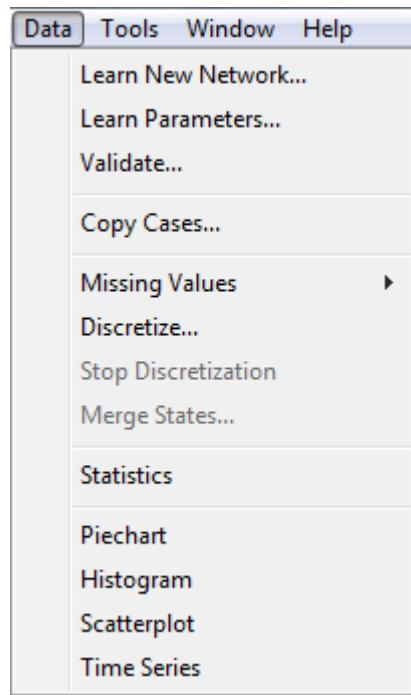
- *Class Variable*, which is a pop-up menu forcing the user to select one of the variables as the class variable. This is a crucial choice, as it determines the structure of the graph.
- *Feature Selection*, when checked, invokes an additional function that removes from the feature set those features that do not contribute enough to the classification.
- *Priors* with two choices: *K2* (default) and *BDeu*. These are two popular priors used in Bayesian network scoring metrics introduced by (Cooper & Herskovitz, 1992) and (Buntine, 1991), respectively. A good comparison of these two priors can be found in (Kayaalp & Cooper, 2002). In case of the *BDeu* priors, an additional parameter is *Network Weight* (default 1).

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.

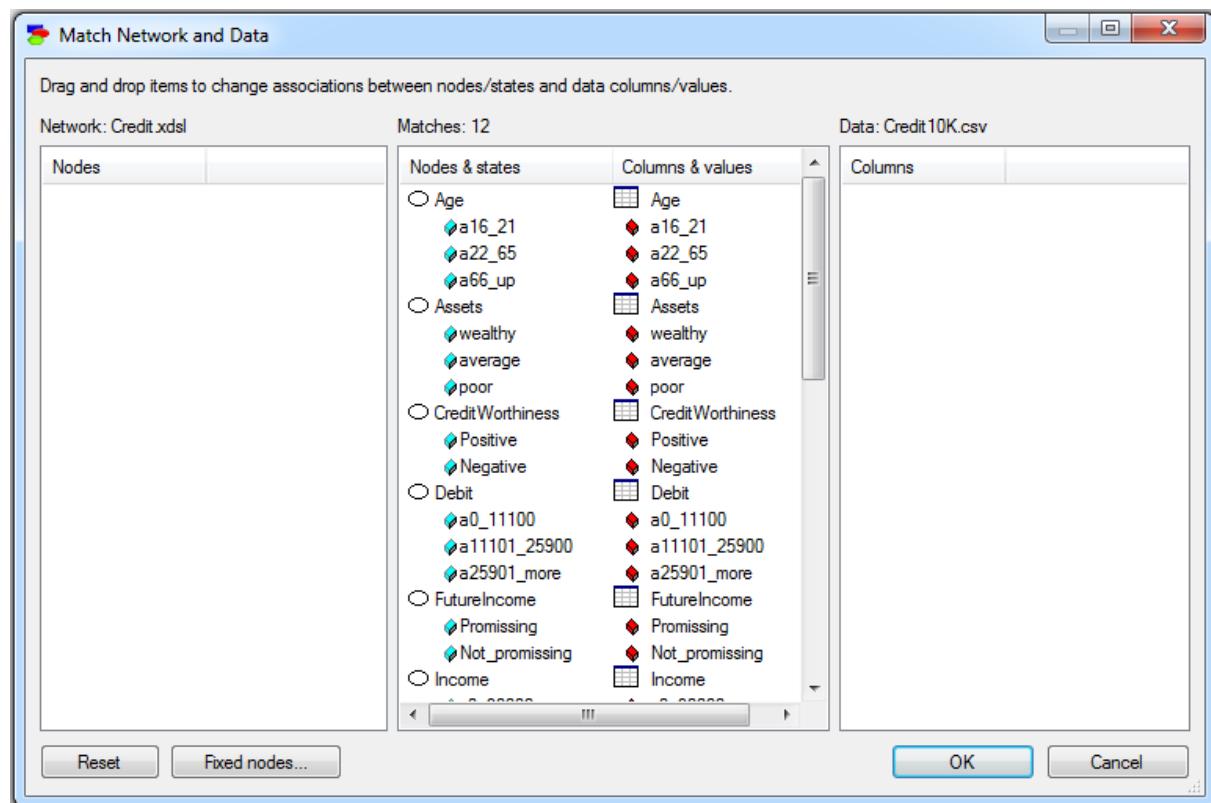


6.5.7 Learning parameters

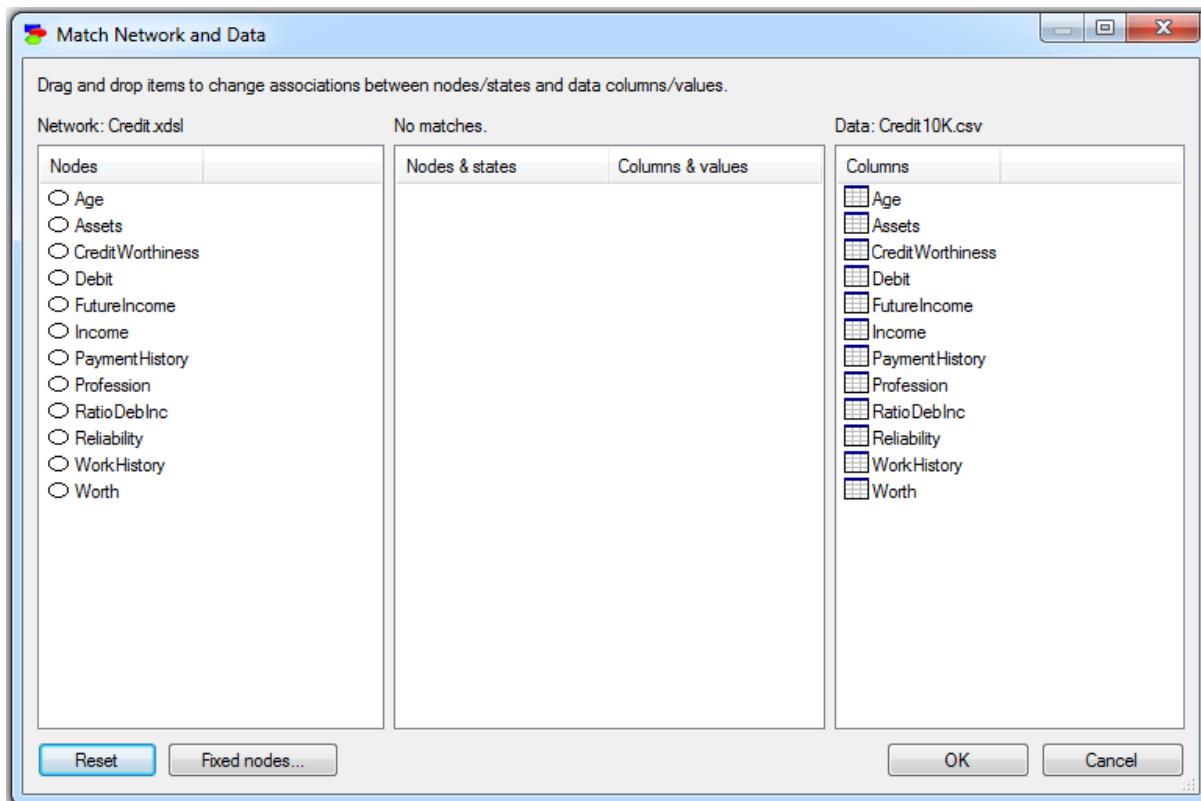
To learn parameters of an existing network (i.e., one for which the structure is already defined), you will need both, a data file and a network open. We will demonstrate the procedure of learning the parameters of a Bayesian network from data on the network *credit.xdsl* and the data file *credit10k.csv*, both available among the example files. Once you have opened both, select *Data-Learn Parameters...*



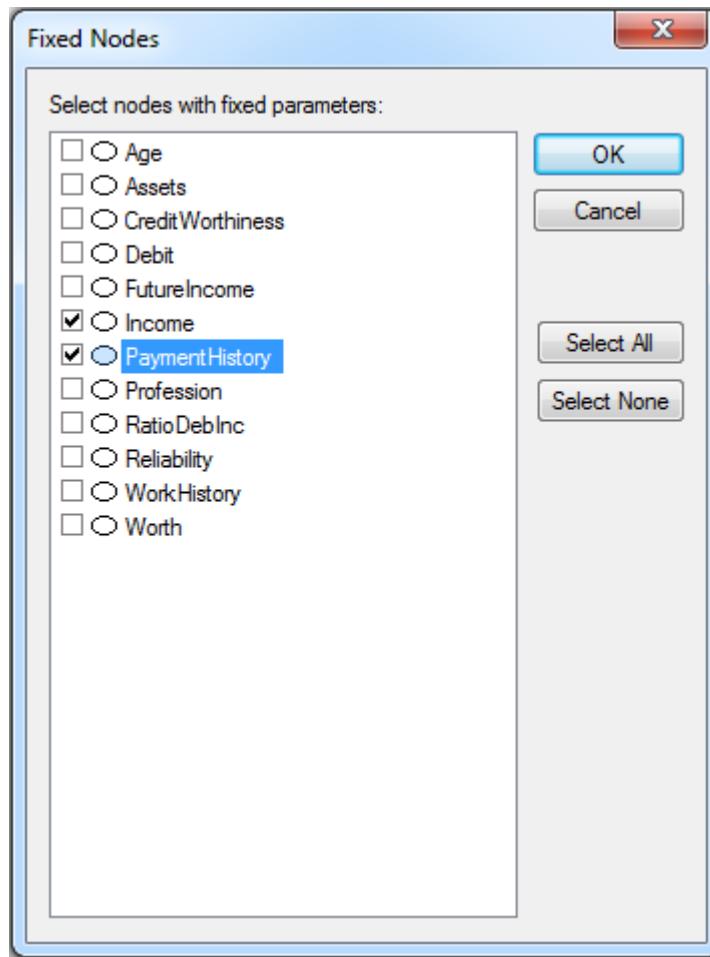
This will invoke the *Match Network and Data* dialog that serves to create a mapping between the variables defined in the network (left column) and the variables defined in the data set (right column).



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). If there is any disparity between them, GeNle 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. To start the matching process from scratch, use the *Reset* button, which will result in the following matching:

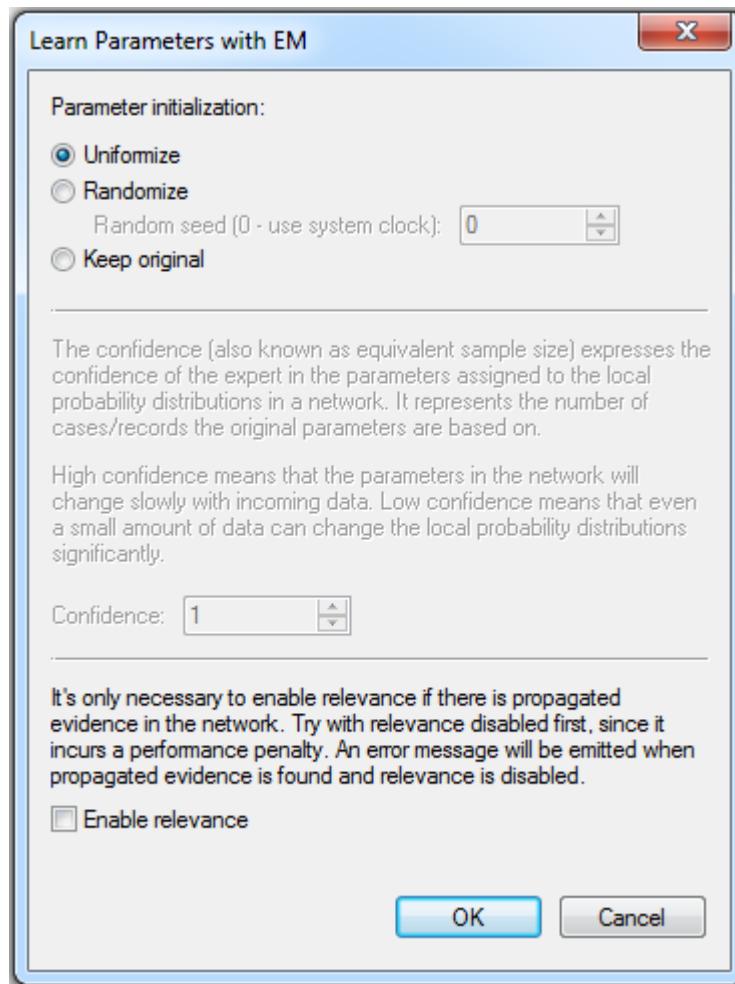


Fixed nodes... button invokes a dialog that allows for excluding nodes from the learning process:



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:



GeNle 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, typically the case). The algorithm has several parameters:

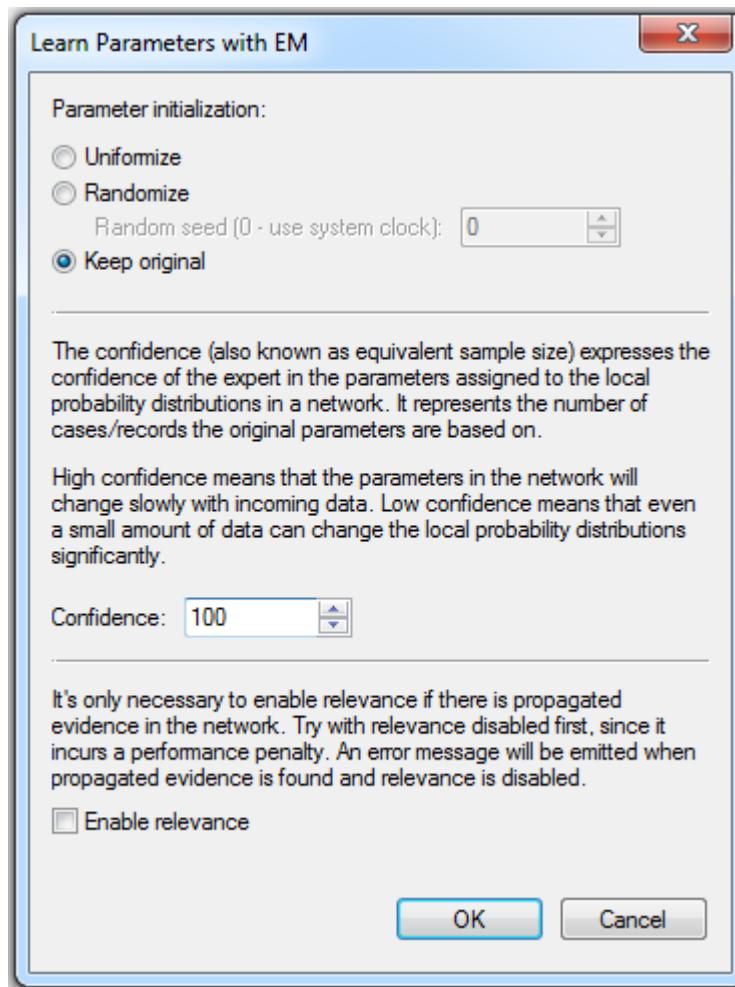
The *Parameter initialization* group allows for choosing one of the three possible starting points of the EM algorithm.

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. *Random number seed* is the seed passed to the random number generator. Using the same seed makes the results perfectly reproducible, unless the seed is equal to zero (the default value), in which case GeNle 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.

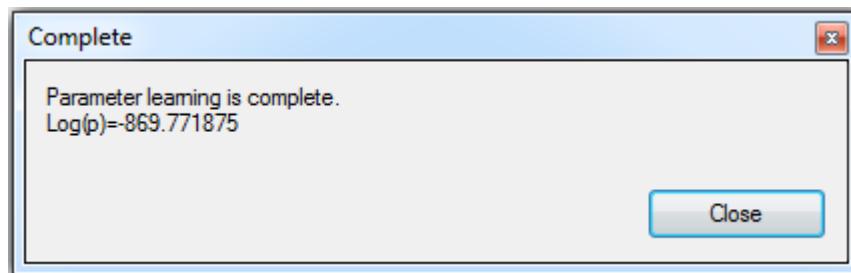


When the parameters in the network have been elicited from an expert, we can view them as the number of cases that the expert has seen before providing us with the current parameters. The larger the ESS, the less weight is assigned to the new cases, which gives a mechanism for gentle refinement of model 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 known in learning.

Enable relevance option makes the algorithm faster by speeding up the Bayesian inference part. We suggest that this be checked only if 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:



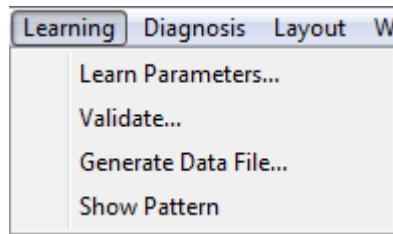
$\text{Log}(p)$, ranging from minus infinity to zero, is a measure of fit of the model to the data.

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.

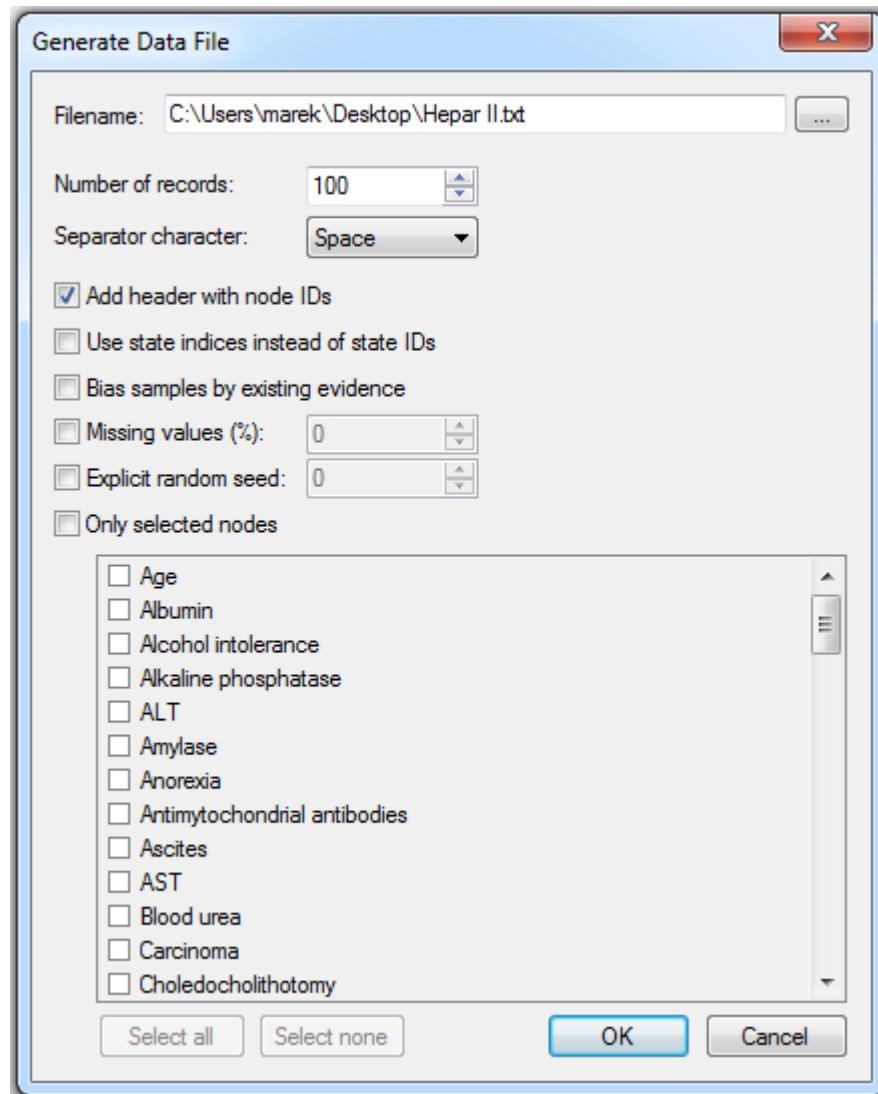
Finally, a remark on a limitation in learning parameters of continuous, multi-variate Gaussian models. Our implementation of the EM algorithm in this case does not allow for missing values. An extension of this implementation is on our development agenda, so please stay tuned!

6.5.8 Generating a data file

A [Bayesian network](#)⁴⁵ 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. GeNle allows for generating a data file from a mode through the *Generate Data File...* command from the *Learning* menu.



The following dialog appears



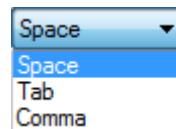
The *Generate Data File* command generates a text file containing records that are representative for the network in the sense of coming from a joint probability distribution modeled by the network. The individual records of the output file

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.) instead of state IDs or state names.

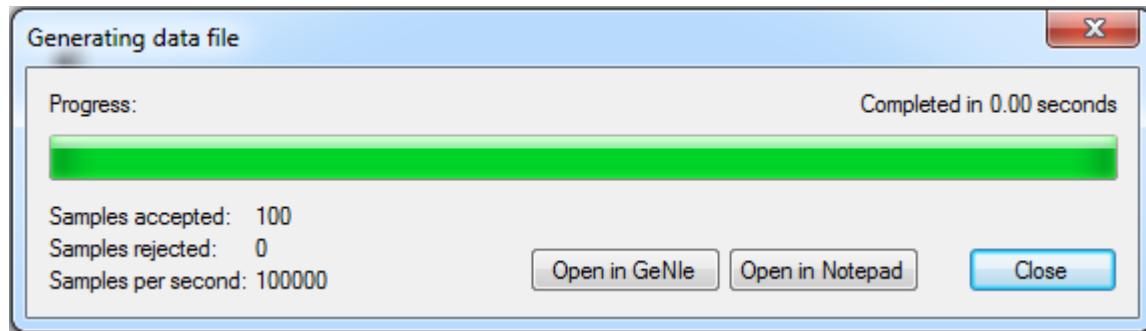
Bias samples by existing evidence, when checked, generates a data file from the posterior joint probability distribution (i.e., biased by the observations) rather than from the original joint probability distribution.

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.

Pressing the *OK* button starts the generation process, which ends with the following dialog

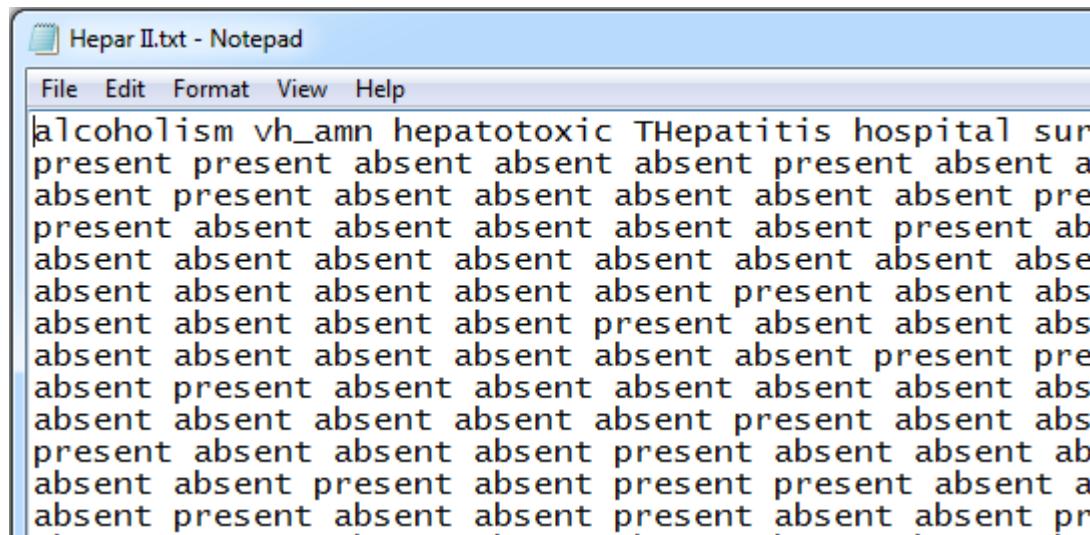


In addition to reporting the time taken by the generation process, the number of samples generated (these are divided into *Samples accepted* and *Samples rejected*, which becomes of essence only when generating a data file biased by observed evidence - in this case samples incompatible with the evidence 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.

	alcoholism	vh_amn	hepatotoxic	THepatitis	hospital	surv
▶	absent	absent	absent	absent	absent	pres
	present	absent	absent	present	absent	pres
	absent	present	absent	absent	present	absent
	absent	absent	absent	present	present	pres
	absent	absent	absent	absent	absent	absent
	absent	absent	absent	absent	absent	absent
	absent	absent	absent	absent	present	absent
	absent	absent	absent	absent	present	absent
	present	absent	absent	absent	absent	absent
	absent	absent	absent	absent	present	absent
	absent	absent	absent	absent	absent	absent

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



The screenshot shows a Notepad window titled "Hepar II.txt - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main content area contains a large amount of tab-separated text representing a dataset. The first few lines of the data are:

```

alcoholism vh_amn hepatotoxic THepatitis hospital sur
present present absent absent absent present absent al
absent present absent absent absent absent absent pres
present absent absent absent absent absent present abs
absent absent absent absent absent absent absent absen
absent absent absent absent absent present absent abs
absent absent absent absent absent absent present pres
absent present absent absent absent absent absent abs
absent absent absent absent absent present absent abs
absent absent absent absent absent present absent abs
absent absent absent absent absent present absent abs
absent absent present absent absent absent absent pres
absent absent present absent present present absent al
absent present absent absent present absent absent pre

```

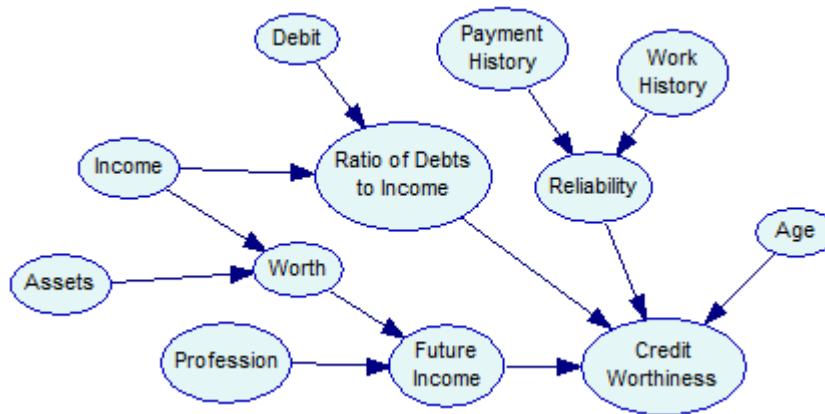
6.5.9 Validation

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. 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:

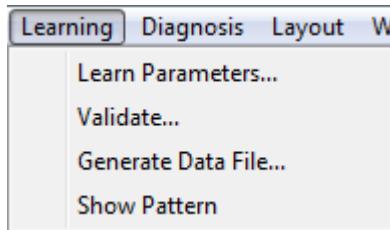
	Payment History	Work History	Reliability	Debit	Income	RatioDebInc	Assets	Worth	Profession	FutureIncome	Age	CreditWorthiness
► Without_Reference	Unstable	Unreliable	a0_11100	s70001_more	Favorable	wealthy	High	Medium_income_profession	Promissing	a16_21	Negative	
Aceptable	Unjustified_no_work	Unreliable	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	
NoAcceptable	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	
NoAcceptable	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	
NoAcceptable	Unstable	Unreliable	a25901_more	s30001_70000	Unfavorable	wealthy	High	Medium_income_profession	Promissing	a16_21	Negative	
NoAcceptable	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	Promissino	a16_21	Neative	

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:

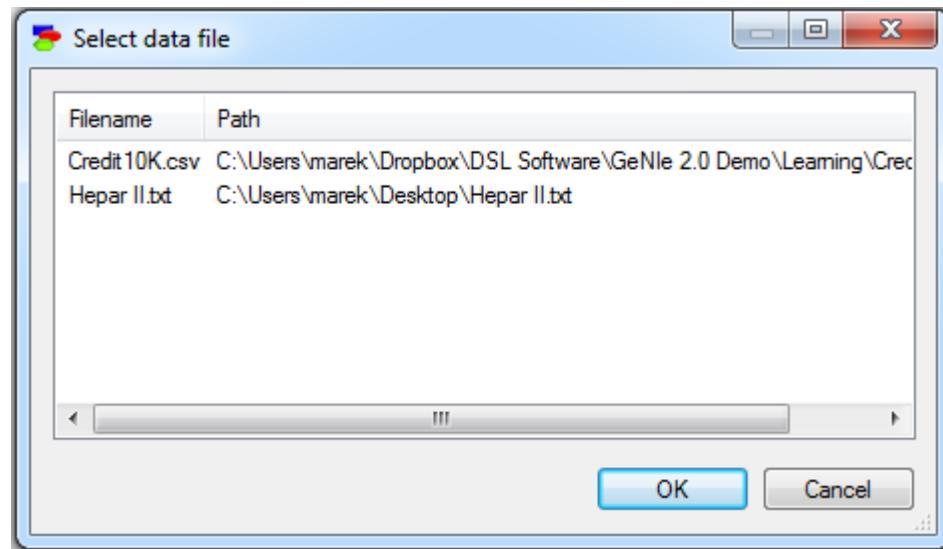


Running validation

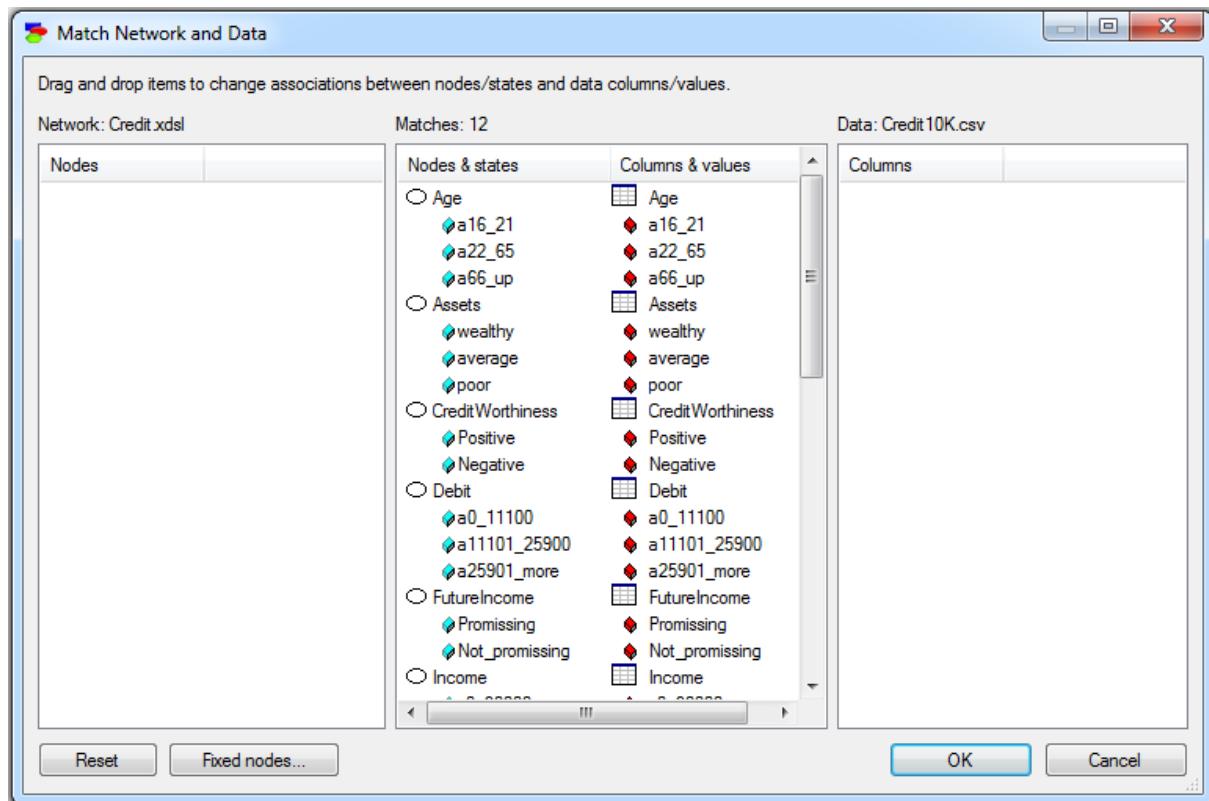
We can perform evaluation of the model by choosing *Validate...* from the *Learning* menu



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 GeNle, the following dialog pops up

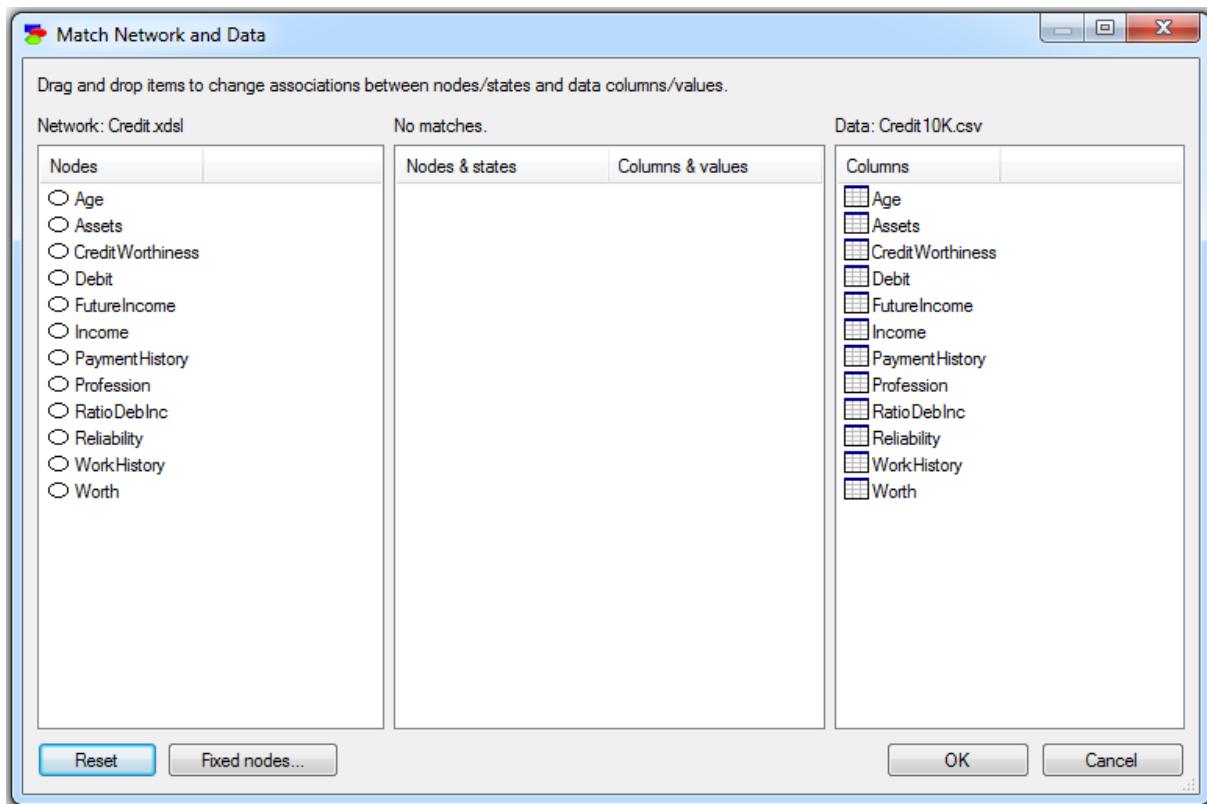


After selecting the file Credit10K.csv, we proceed with the following (*Match Network and Data*) dialog, whose only function is to make sure that the variables and states in the model (left column) are mapped precisely to the variables defined in the data set (right column). This dialog is identical to the dialog appearing when learning model parameters from data.

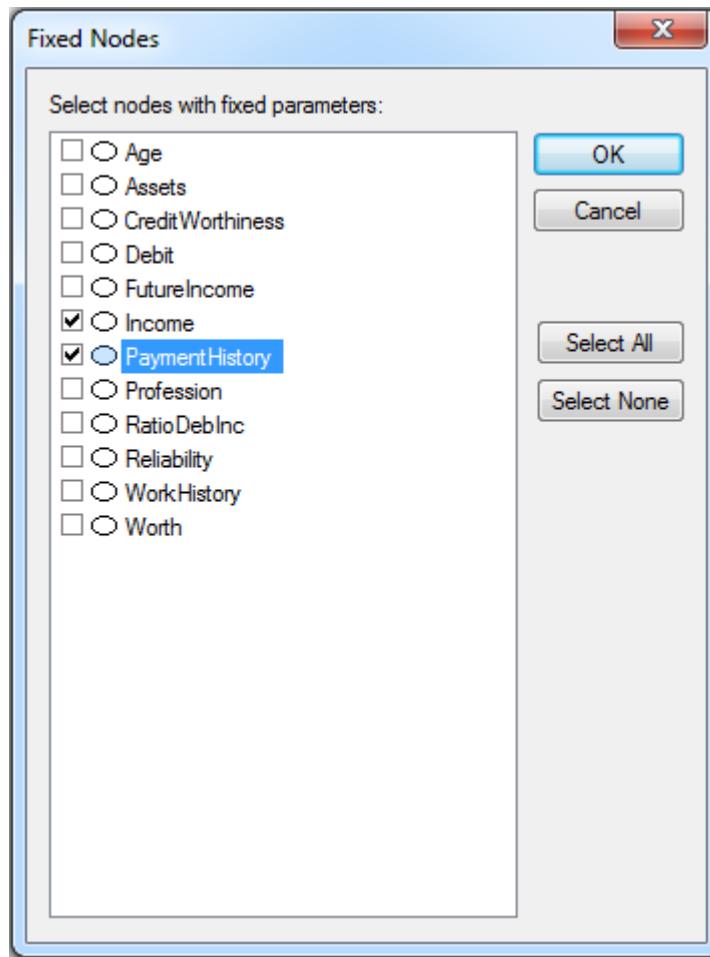


Both lists of variables are sorted alphabetically. The *Match Network and Data* dialog does text pre-matching and places in the central column all those variables and their states that match (have identical or close to identical names). If there is any disparity between them, GeNle highlights the differences 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:

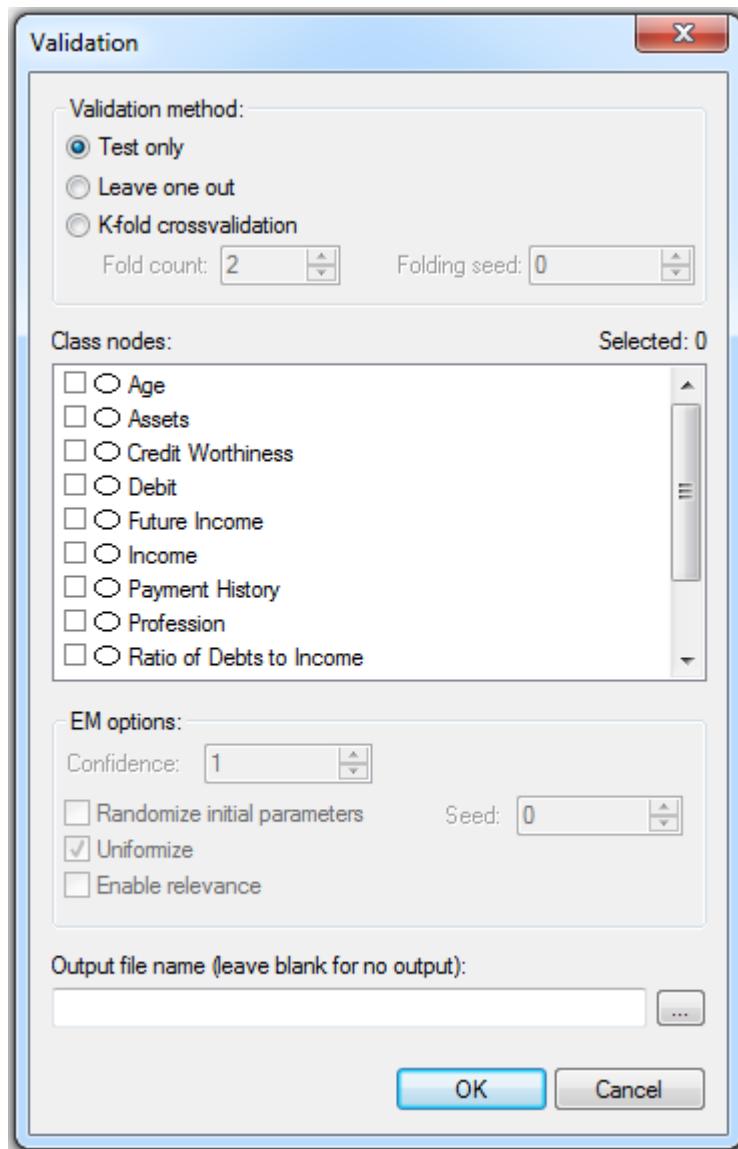


Fixed nodes... button invokes a dialog that allows for excluding nodes from the learning process in cross-validation



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 CPT 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.

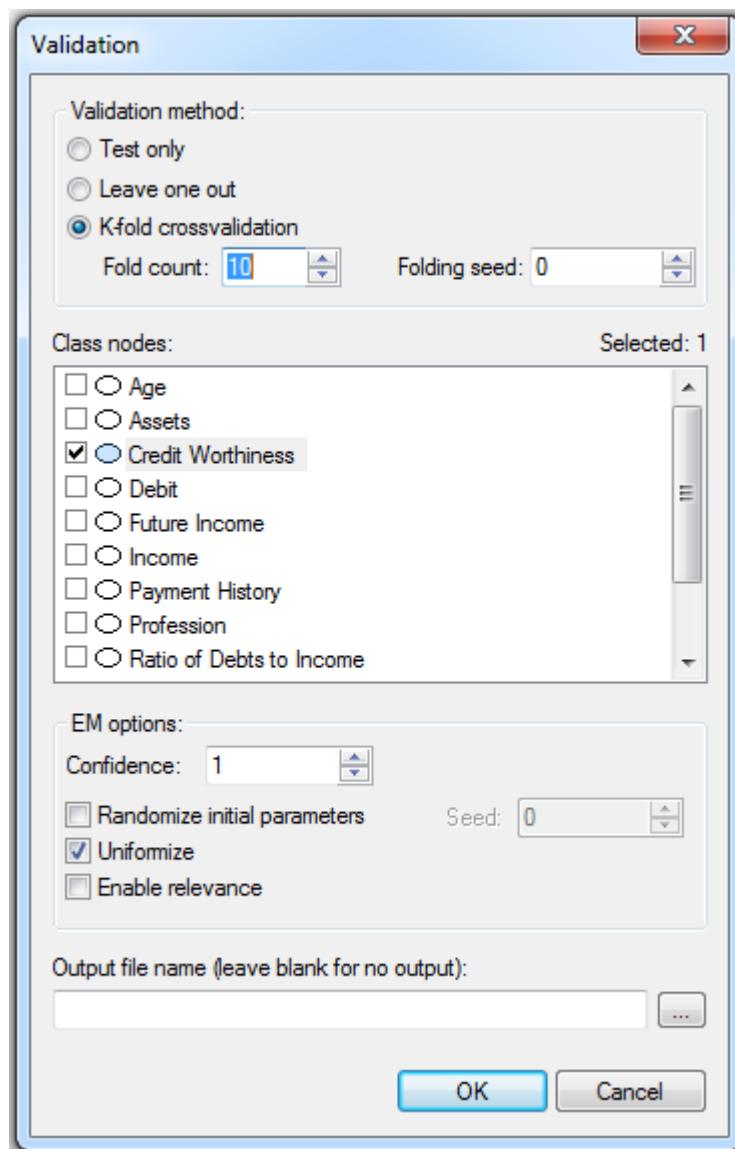
The first element is *Validation method*. The simplest evaluation is *Test only*, which amounts to testing the model on the data file. This is suitable to situations when the model has been developed based on expert knowledge or when the model was learned from a different data set and we want to test it on data that it has never seen. 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. GeNle 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 GeNle 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](#)⁴⁰⁰ section.

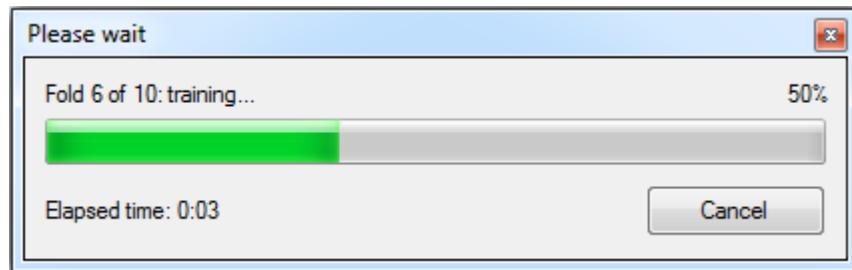
The second important element in the dialog is selection of the *Class nodes*, which are nodes that the model aims at predicting. At least one of the model variables has to be selected. In our example, we will select *Credit Worthiness*.

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 GeNle. 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:

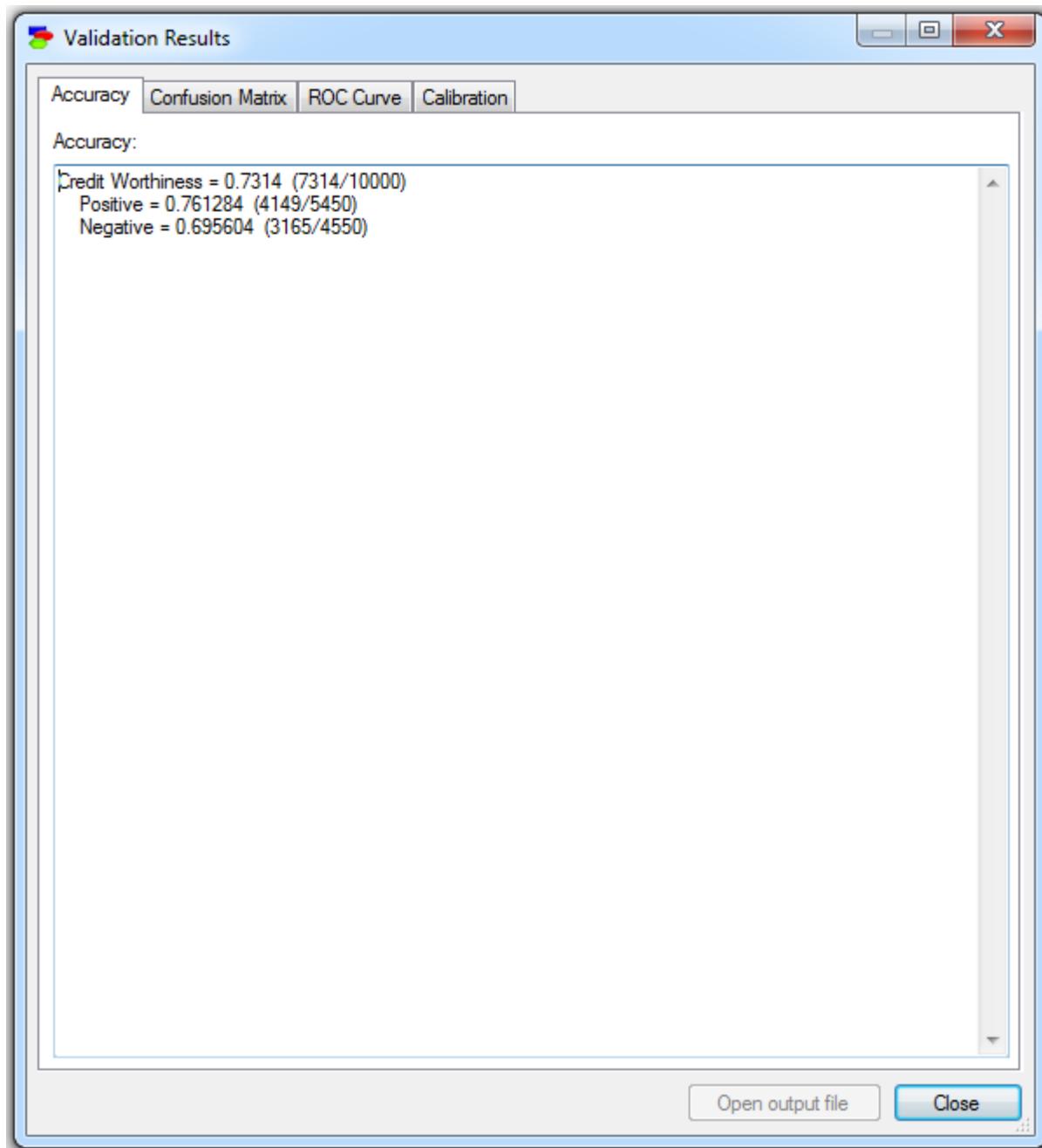


Should the validation take more time than planned for, you can always *Cancel* it and restart it with fewer folds.

Validation results for a single class node

Accuracy

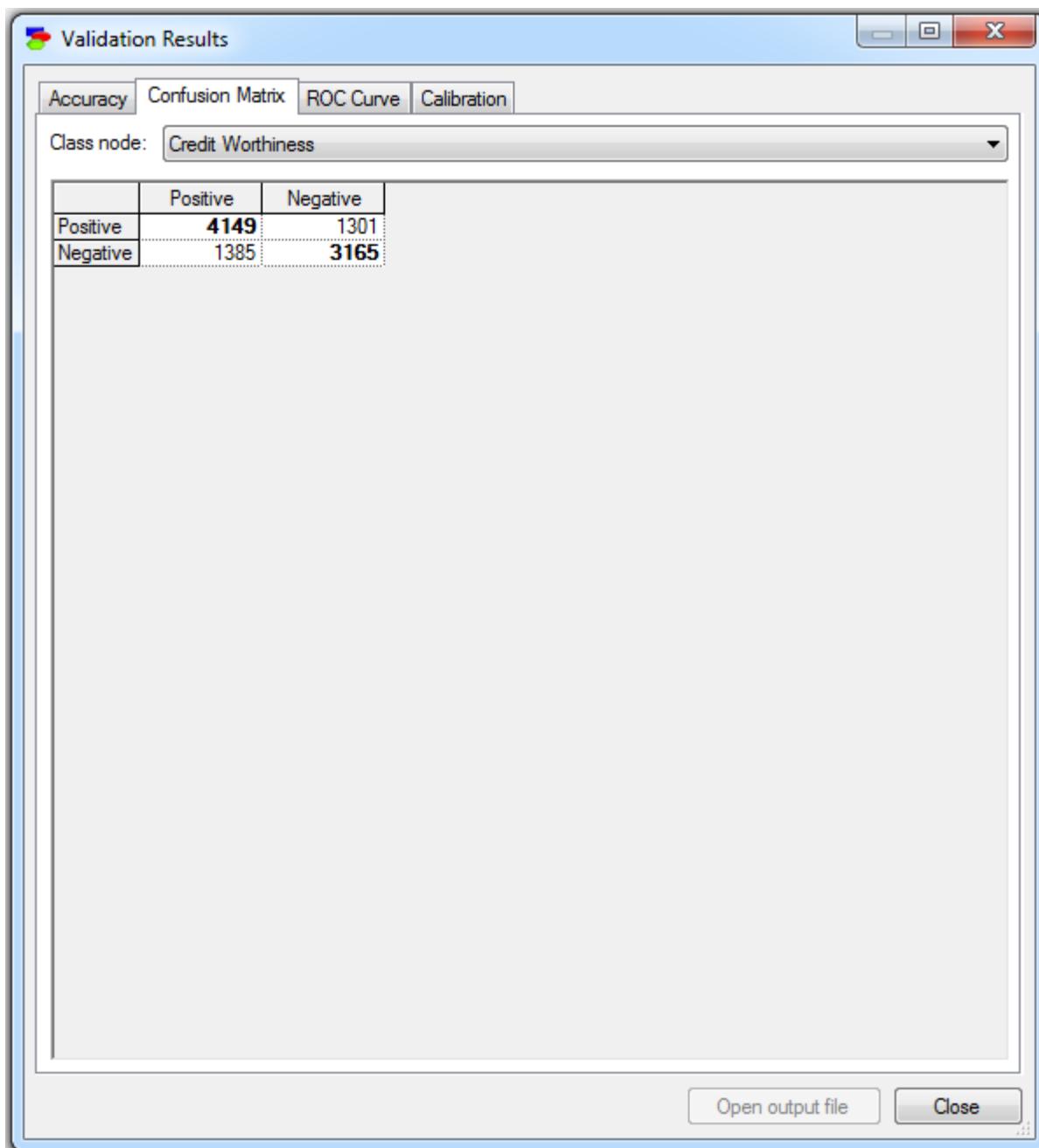
When finished has finished, the following dialog appears:



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*.

Confusion Matrix

The *Confusion Matrix* tab shows the same result in terms of the number of records correctly and incorrectly classified. Here the columns denote the actual state of affairs and the rows 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.



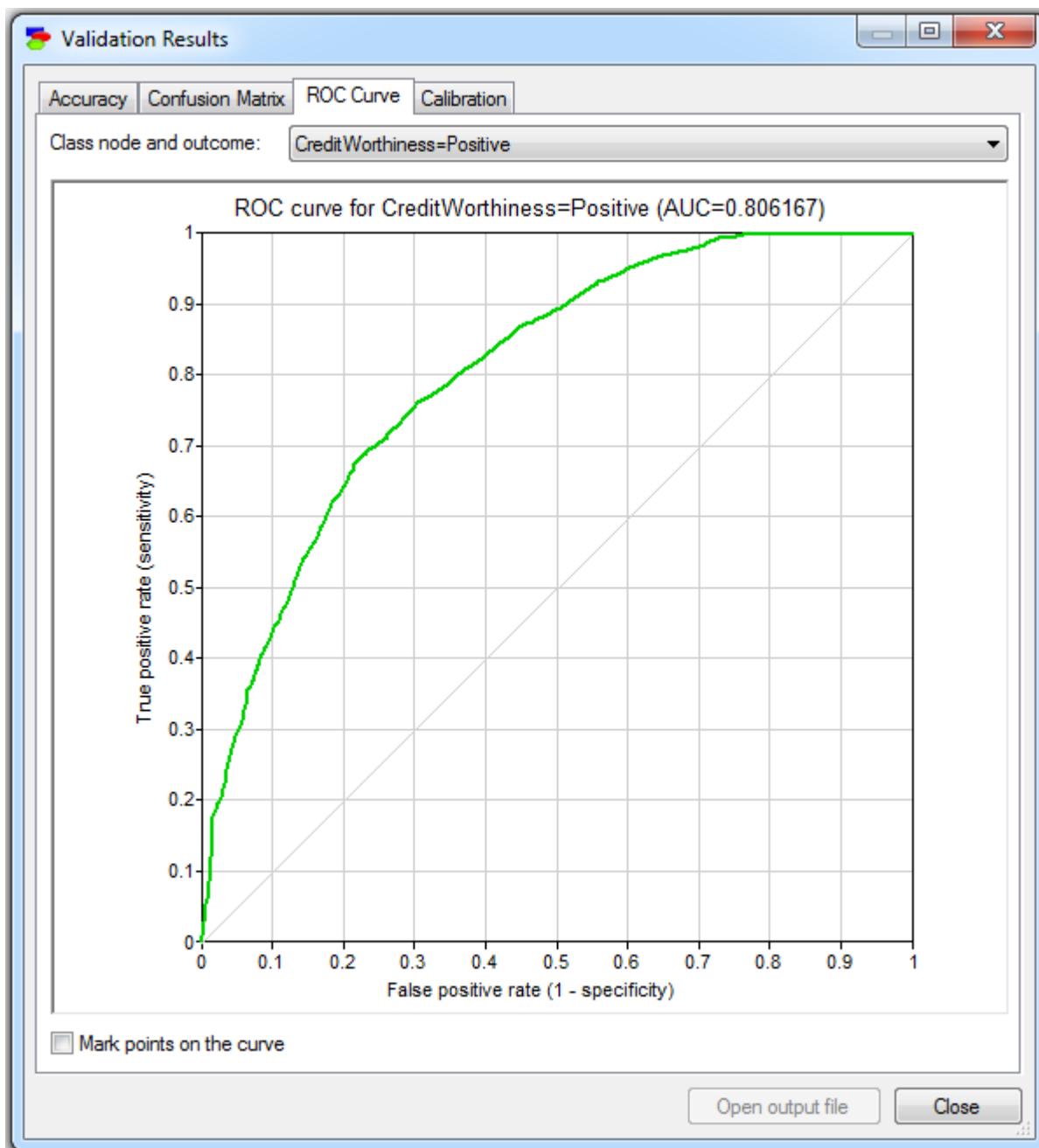
You can copy the contents of the confusion matrix by right-clicking *Copy* on the selected cells (or right-clicking and choosing *Select All*). You can paste the selected contents into other programs.

ROC Curve

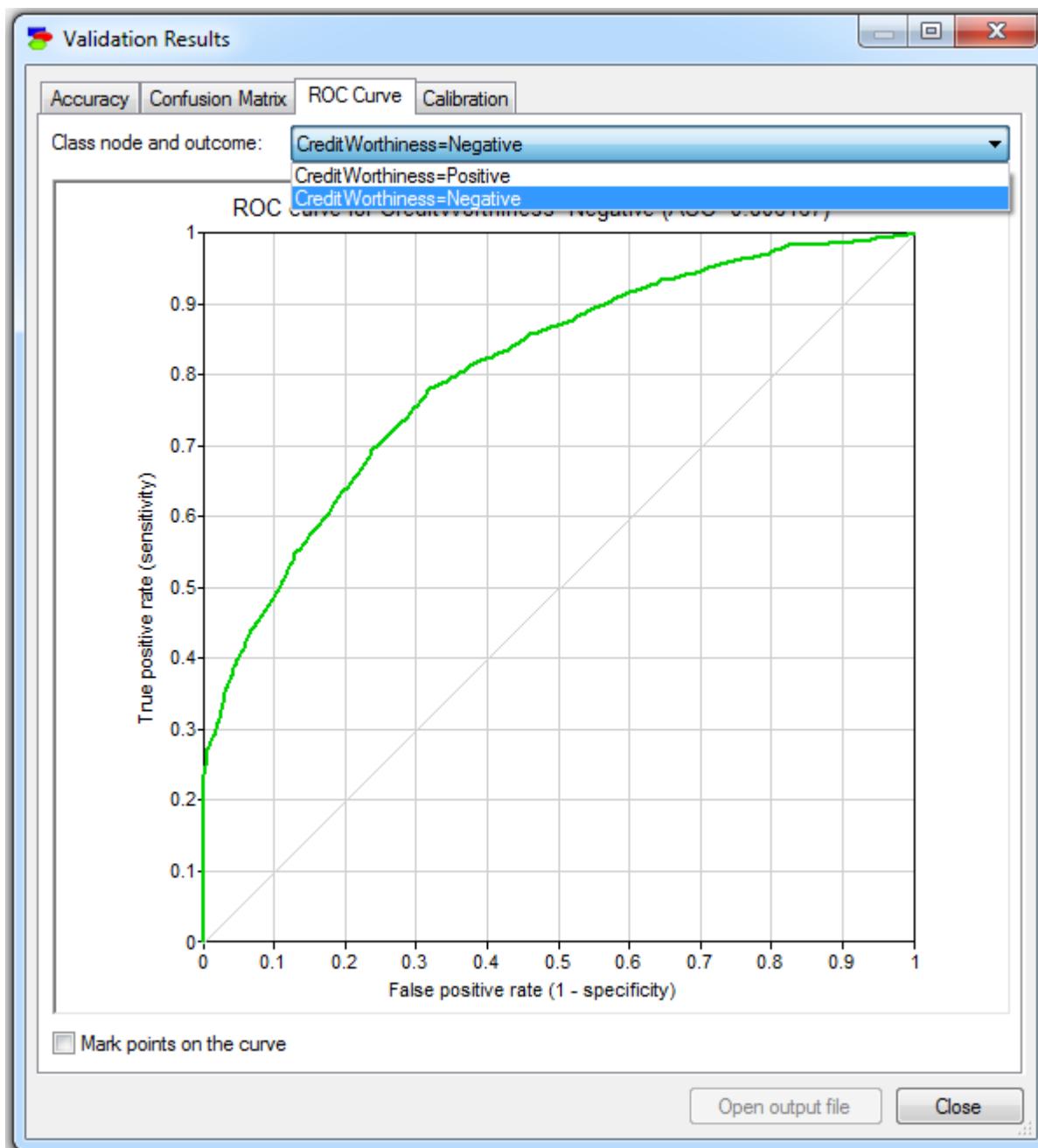
The *ROC Curve* tab shows the Receiver Operating Characteristic (ROC) curves for each of the states of each of the class variables. ROC curves originate from Information Theory and are an excellent way of expressing the quality of a model

independent of the classification decision (in case of GeNle 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 GeNle 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.

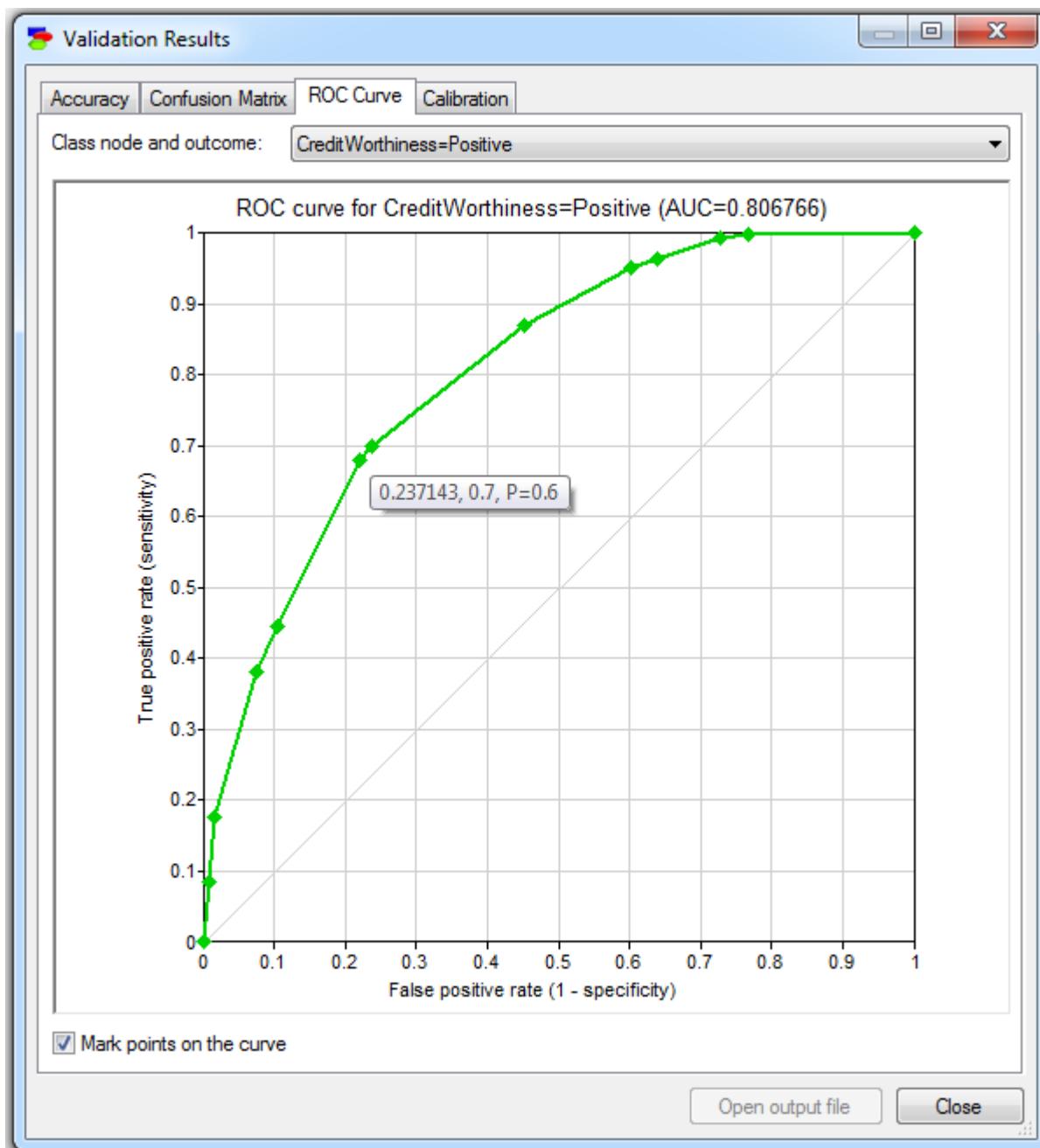
The following ROC curve is for the state *Positive* of the variable *Credit Worthiness*. The dim diagonal line shows a baseline ROC curve of a hypothetical classifier that is worthless. A classifier that does its job will have its ROC curve above this diagonal line. Above the curve, we see the Area Under 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, GeNle 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 or off. The following plot shows this idea.

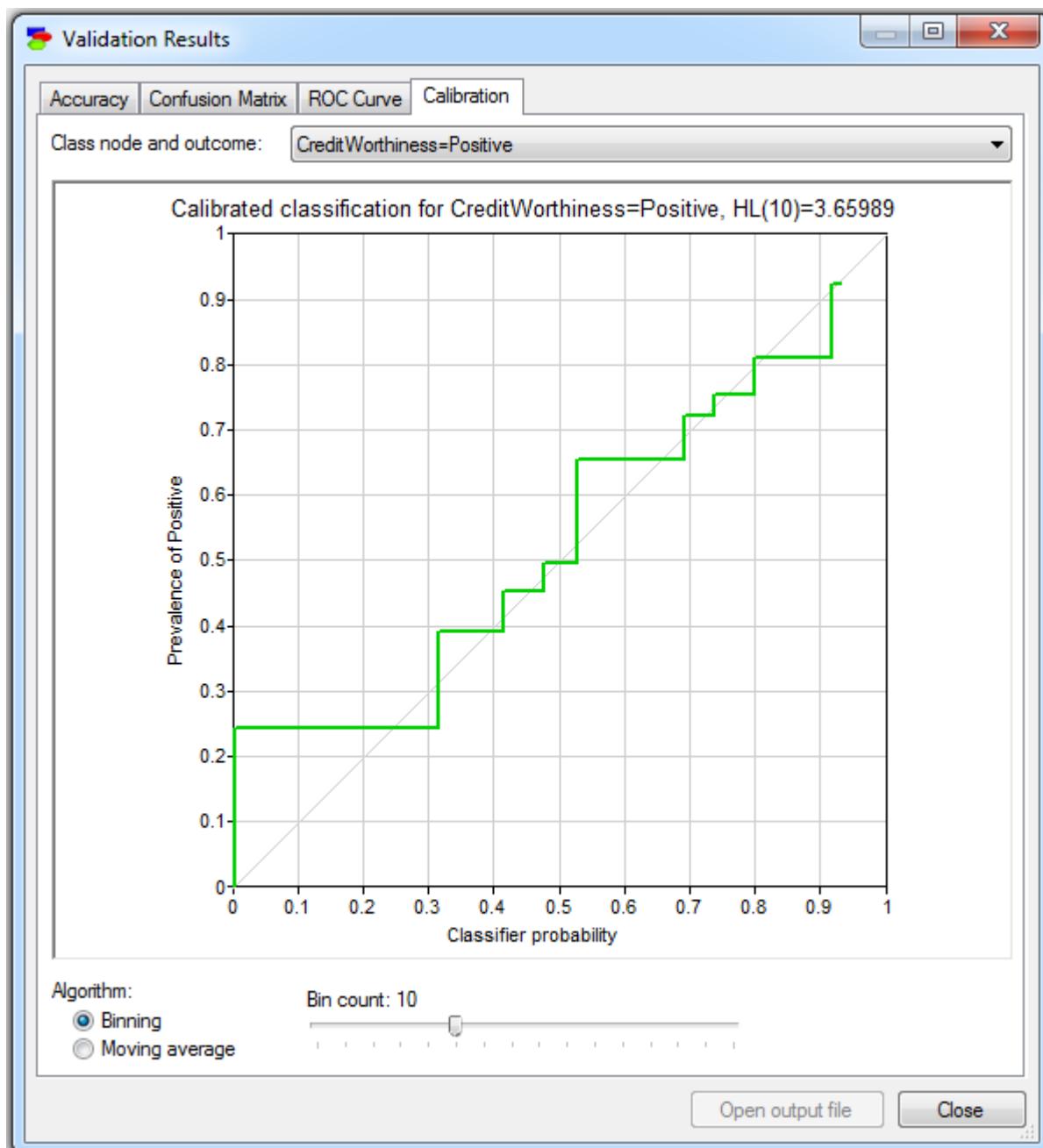


Hovering over any of the points shows the probability threshold value that needs to be used to achieve this point along with the resulting (1-specificity) and sensitivity. In the example above, when the probability threshold is $p=0.6$, the model achieves sensitivity of 0.7 and specificity of $1-0.237143=0.762857$. 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.

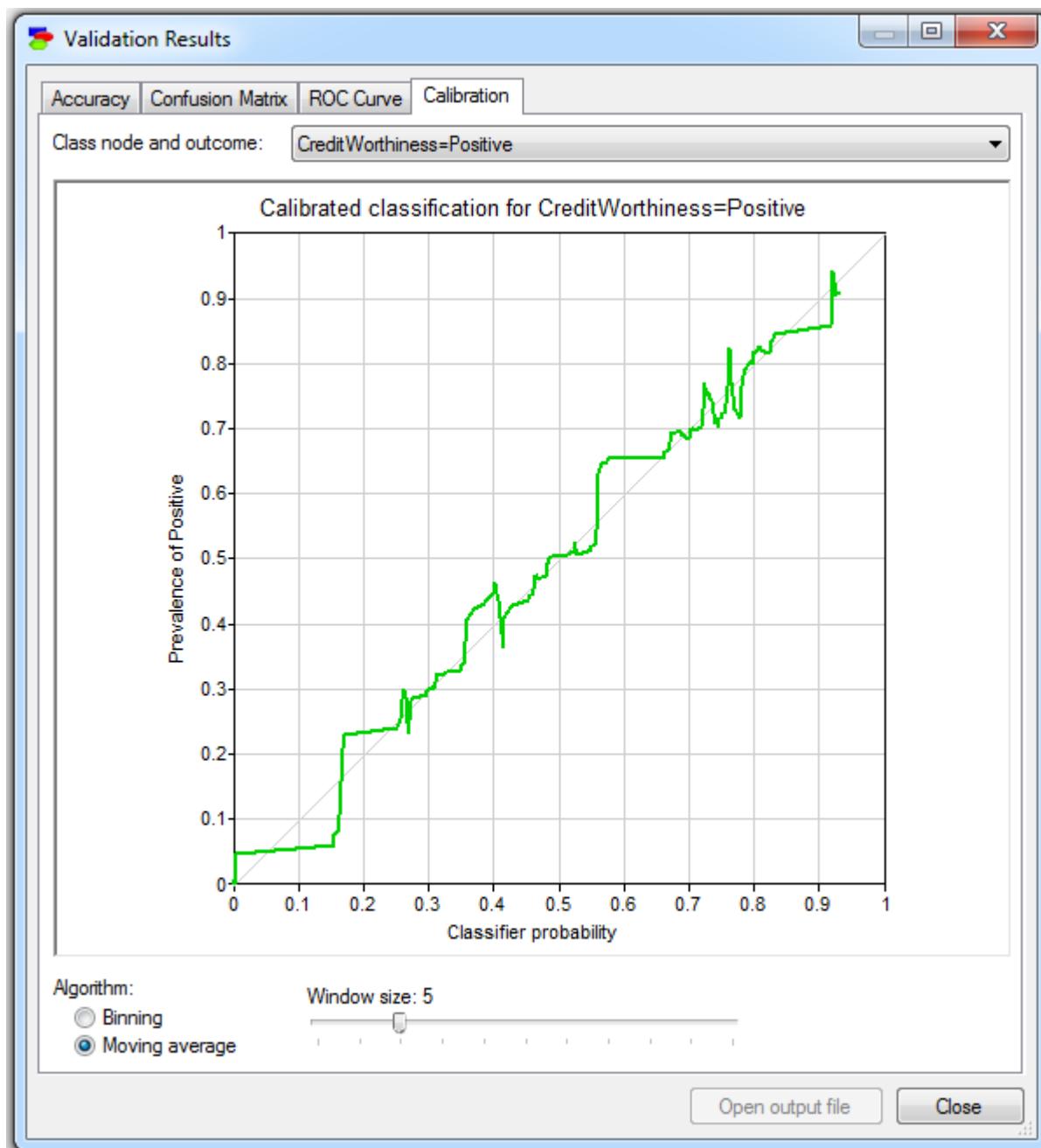
ROC curve is a fundamental and very useful measure of a model's performance. For those users, who are not familiar with the ROC curves and AUC measures, we recommend an excellent article on Wikipedia (https://en.wikipedia.org/wiki/Receiver_operating_characteristic).

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), 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.



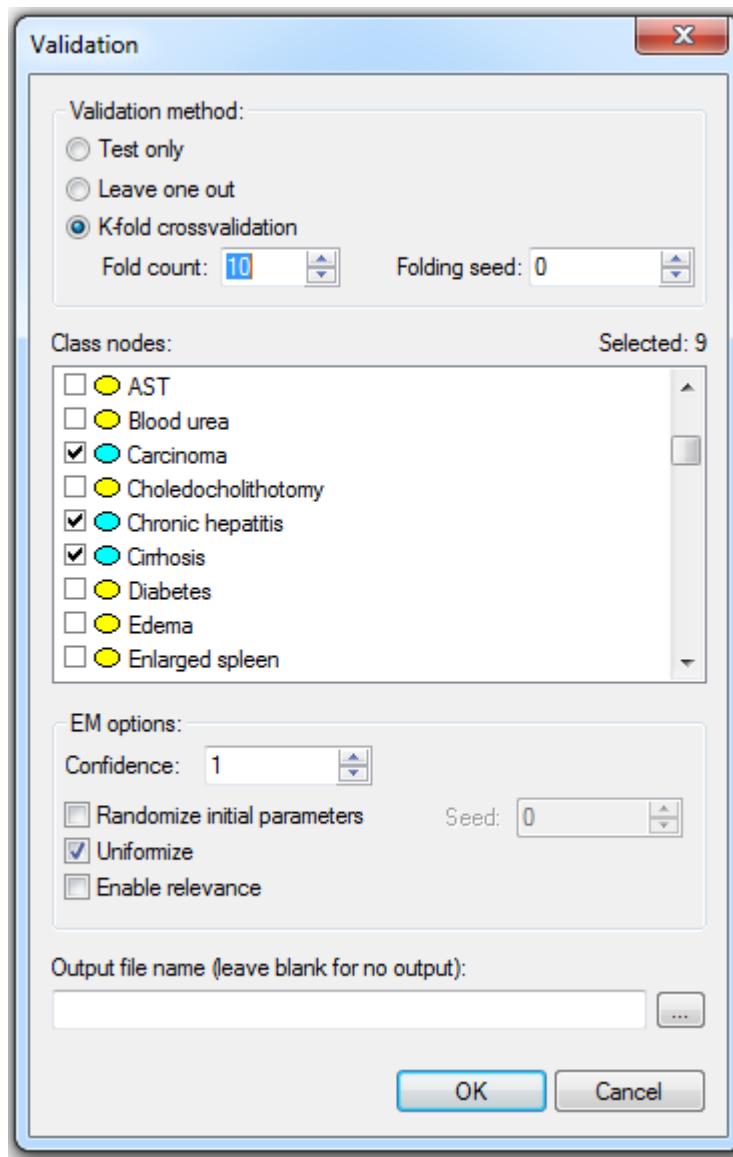
Moving average (see the screen shot below) works somewhat differently. We have a sliding window that takes always the neighboring k output probabilities on the horizontal axis and shows the class frequency among the records in this sliding window on the vertical axis. Here also, as we change the *Window size*, the plot changes as well. It is a good practice to explore several window sizes to get an idea of the model calibration.



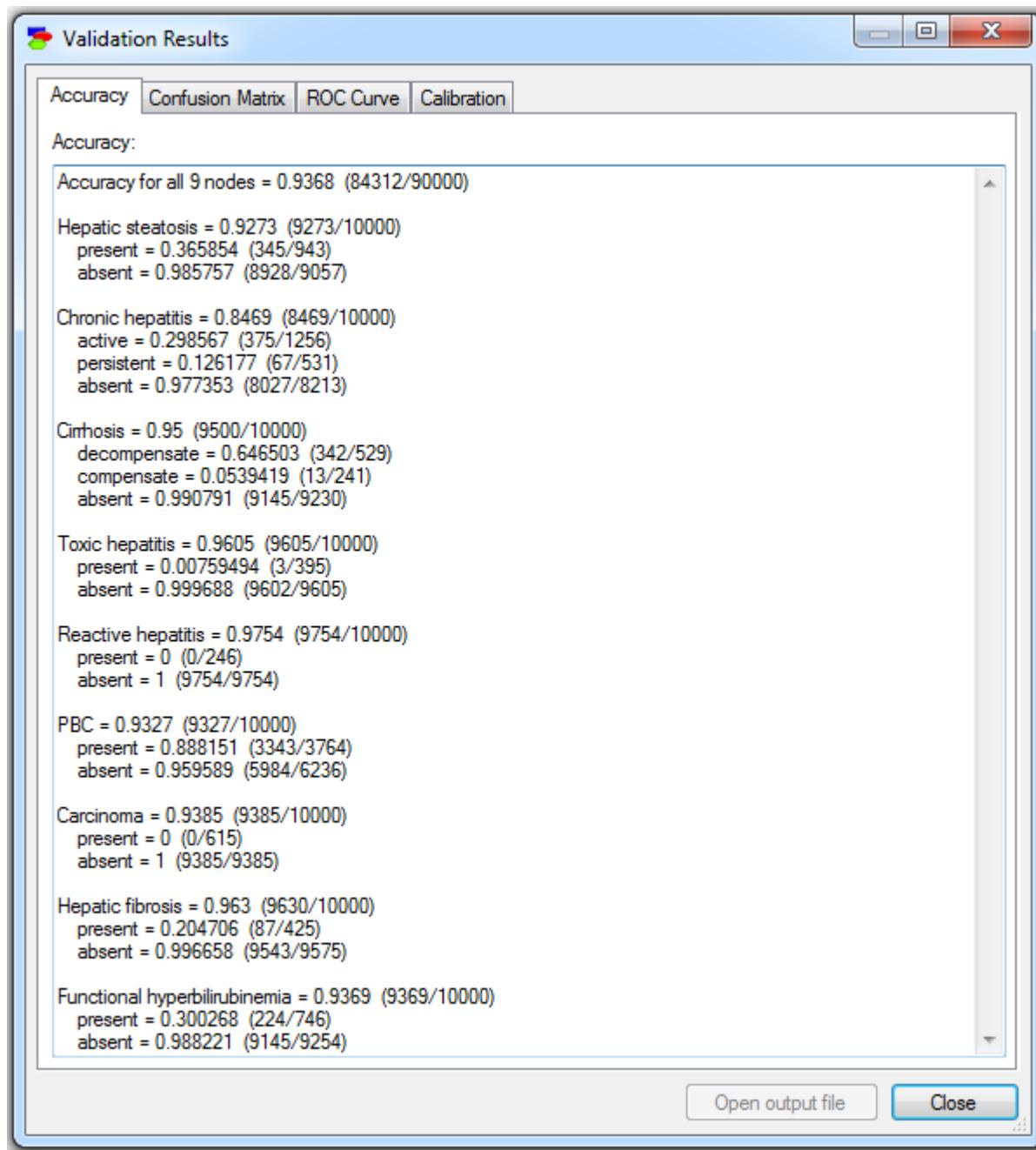
Similarly to the ROC curve, you can copy and paste the coordinates of points on the calibration curve by right-clicking anywhere on the chart, selecting *Copy* and then pasting the results as text into any text editor (such as Notepad or Word). Pasting into any image editor (or *pasting special* into a text editor such as Word) results in pasting the image of the calibration curve.

Validation for a 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



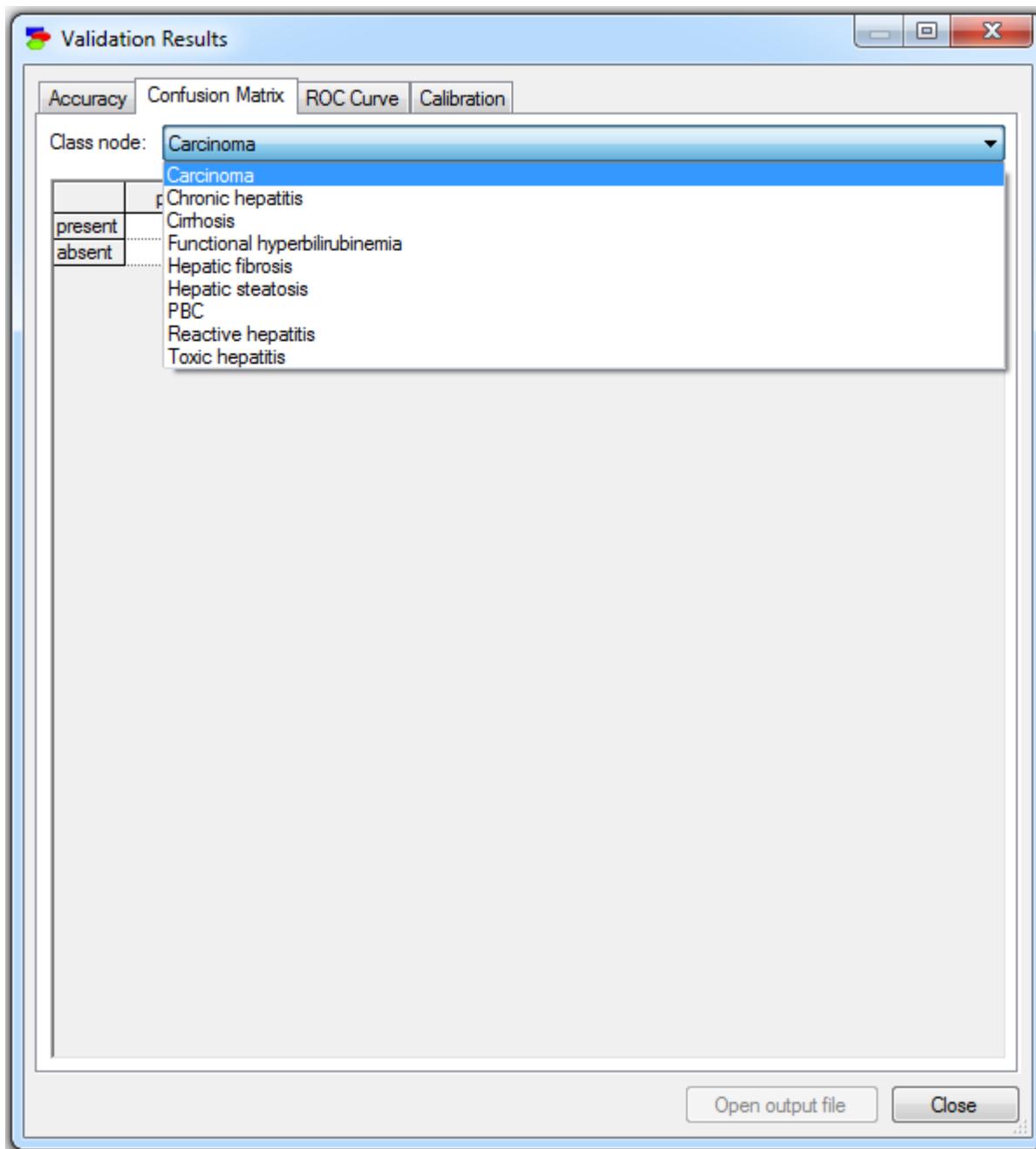
The results show accuracy for each of the class nodes in separation



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. This 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.

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

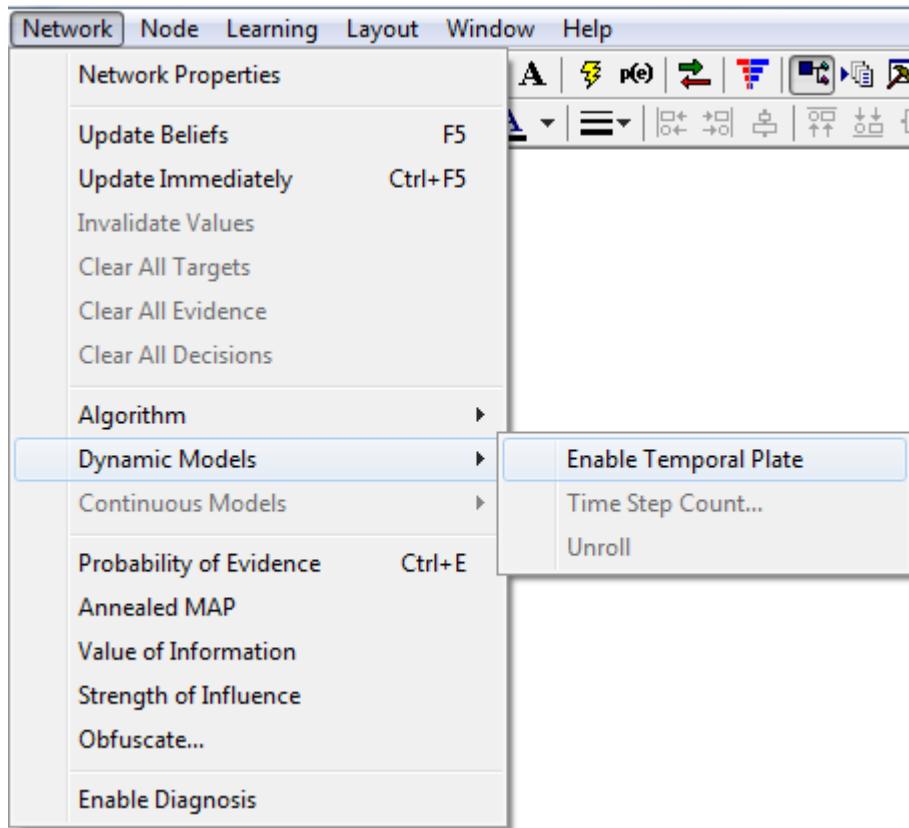
systems that are in some kind of equilibrium state. Unfortunately, most systems in the world change over time and sometimes we are interested in how these systems evolve over time more than we are interested in their equilibrium states. Whenever the focus of our reasoning is change of a system over time, 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 GeNle. 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 GeNle is based on a M.Sc. thesis by Joris Hulst (2006).

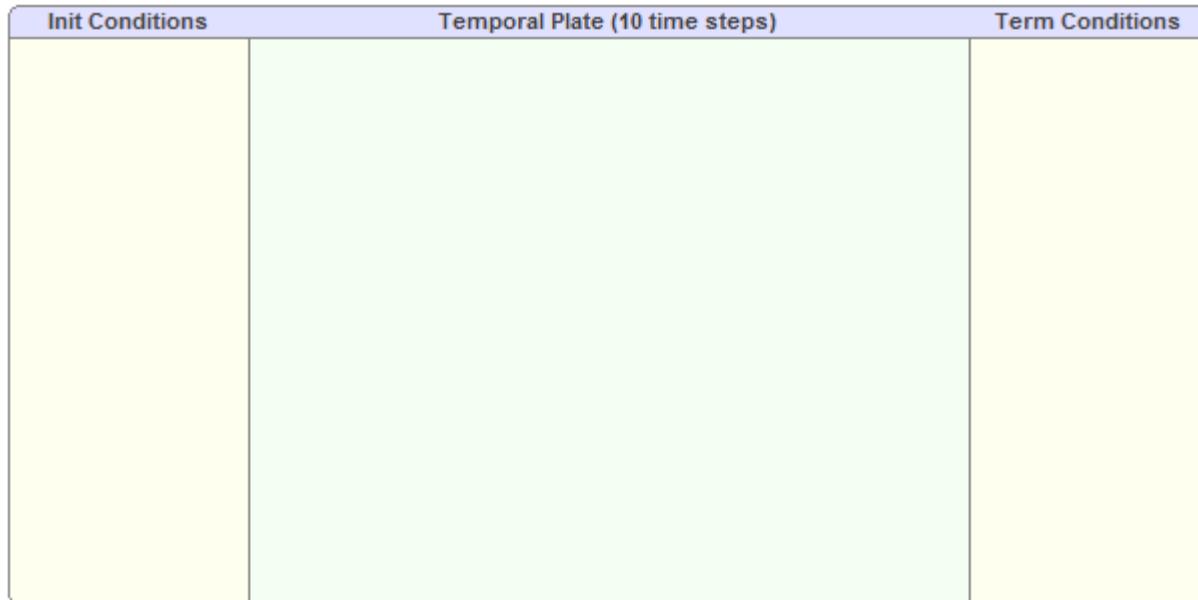
6.6.2 Creating DBN

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 GeNle 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

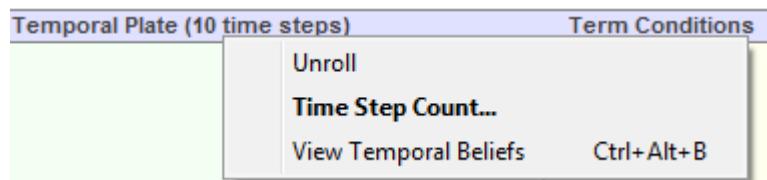


The *Temporal Plate* divides the *Graph View* into four areas:

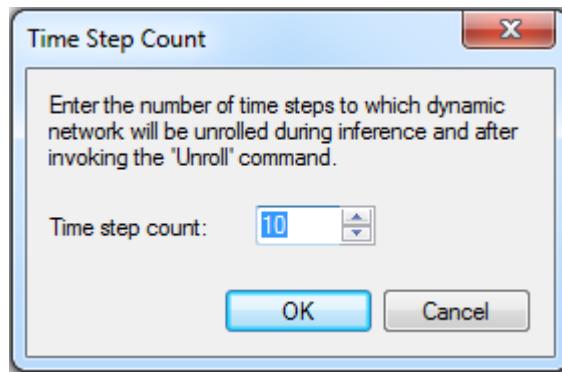
- *Contemporals*, which is the part of the *Network View* window that is outside of the temporal plate. All nodes in that are static.
- *Init Conditions*, which is the part of the network area where, so called, *anchor nodes* are stored. An *anchor node* is a node outside of the temporal plate that has 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 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 its 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.

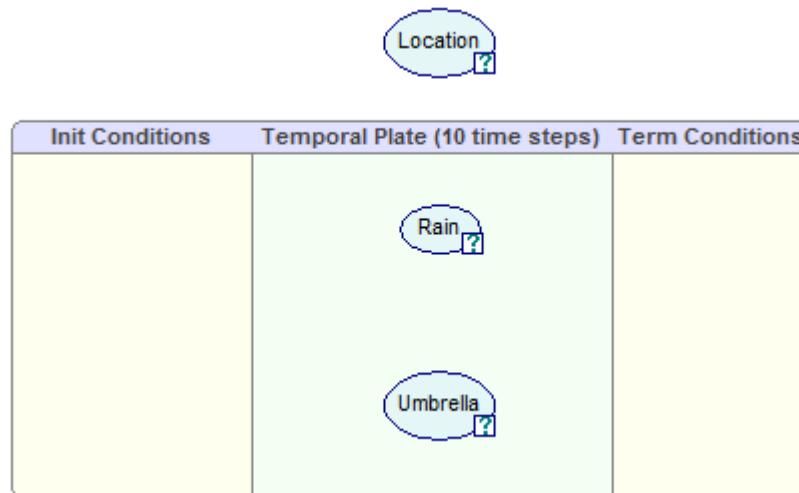
For our example, we set the number of steps to 8. We can either double-click or right-click on the header of the *Temporal Plate*



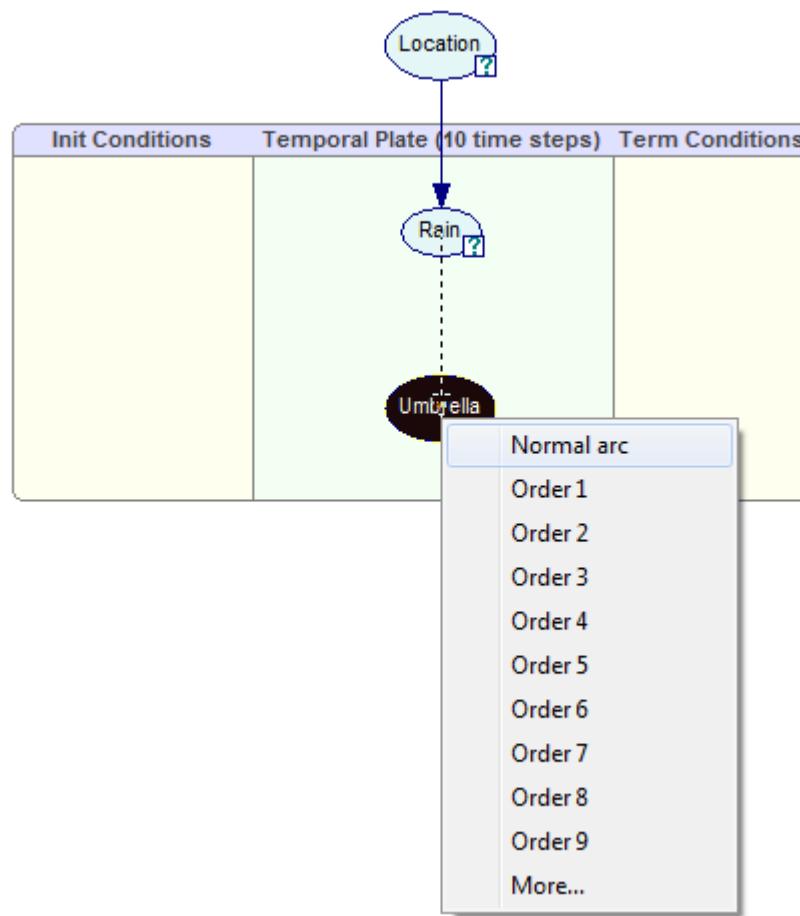
which will invoke the *Time Step Count* dialog that allows to change the *Time step count*



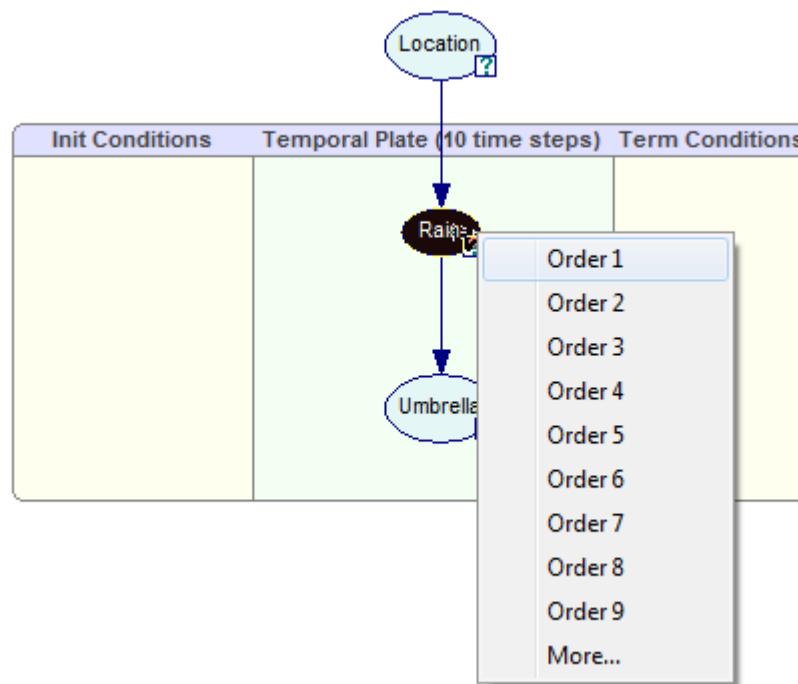
We create the following nodes: *Location* in the *Contemporals*, *Rain* and *Umbrella* in the *Temporal Plate* and set their states as described in the example above.



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*. The same happens when we draw an arc from *Rain* to itself, which will represent the impact on *Rain* on a given day that *Rain* on the prior day has. 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.

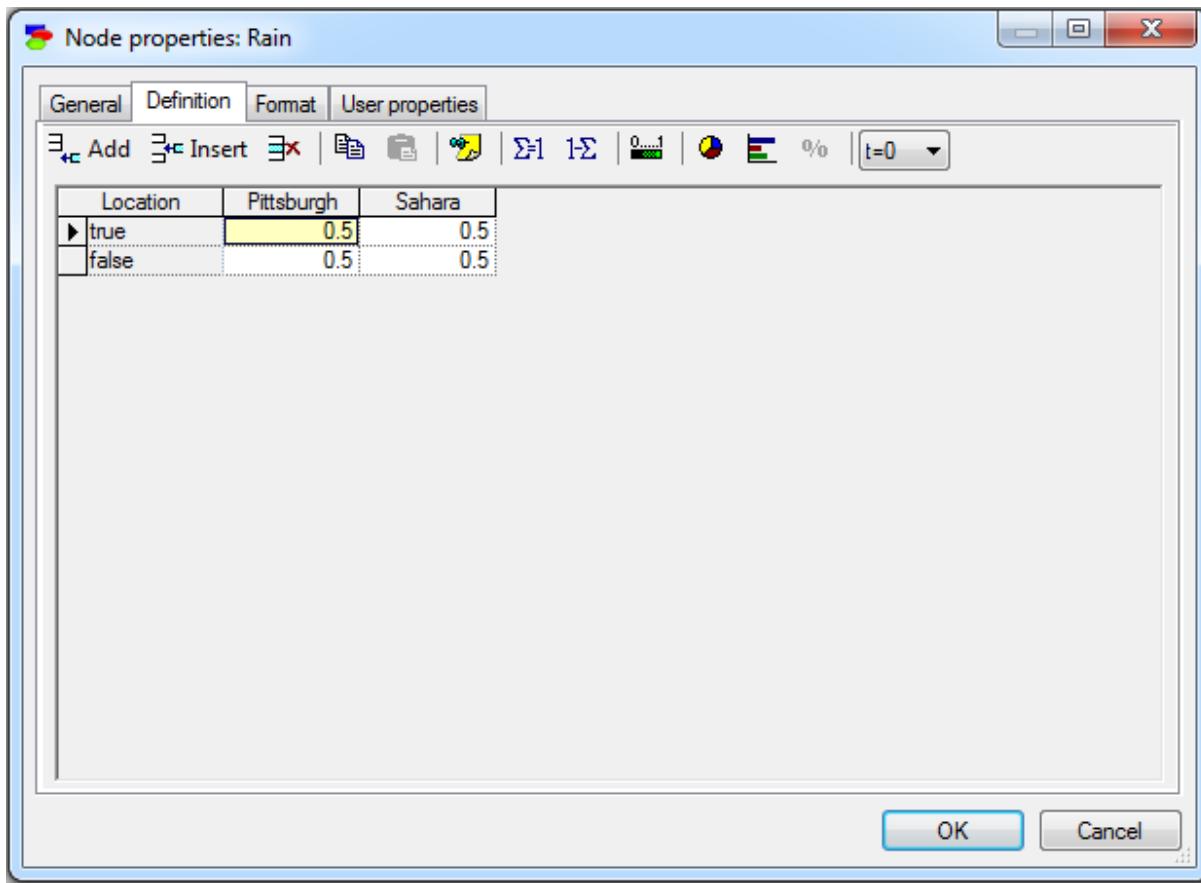
Please note that a DBN, as implemented in GeNIE, can have temporal arcs of any order, which means that DBNs in GeNIE can model dynamic processes of any order.

We define each of the nodes in terms of 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
true	0.7	0.3	0.001	0.01
false	0.3	0.7	0.999	0.99

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*, GeNle 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

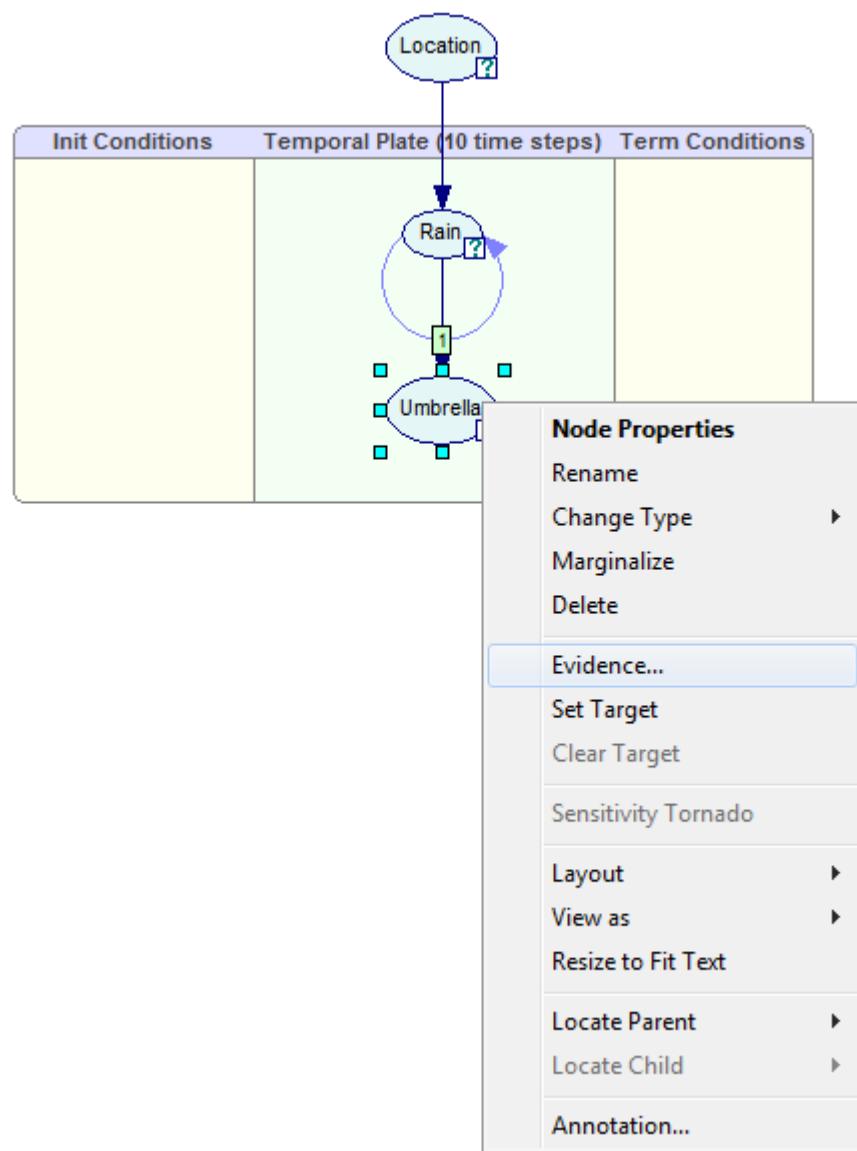
Inference in a DBN, similarly to inference in a BN, amounts to calculating the impact of observation of some of its variables on the probability distribution over other variables. The additional complication is that both evidence and the posterior probability distribution is indexed by time. We will go through the example used in the previous sections to demonstrate setting evidence, running an algorithm, and viewing the results.

Setting temporal evidence

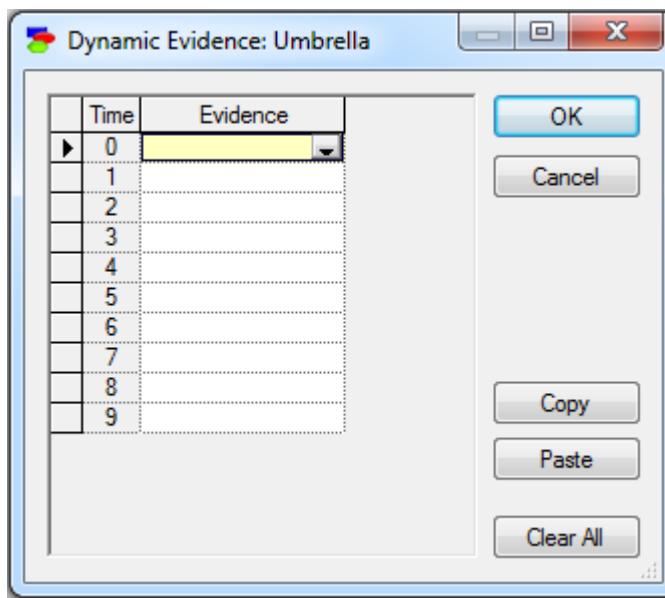
Suppose that during her week-long shift, the guard observes an umbrella on every day except for the second day, when she forgot whether she saw an umbrella, and the fourth day, when she was sure she did not see any umbrella. This means that the evidence vector for the node *Umbrella* is as follows:

$$\text{Umbrella}[0:6] = [\text{true}, \text{true}, \text{true}, \text{false}, \text{--}, \text{true}, \text{true}].$$

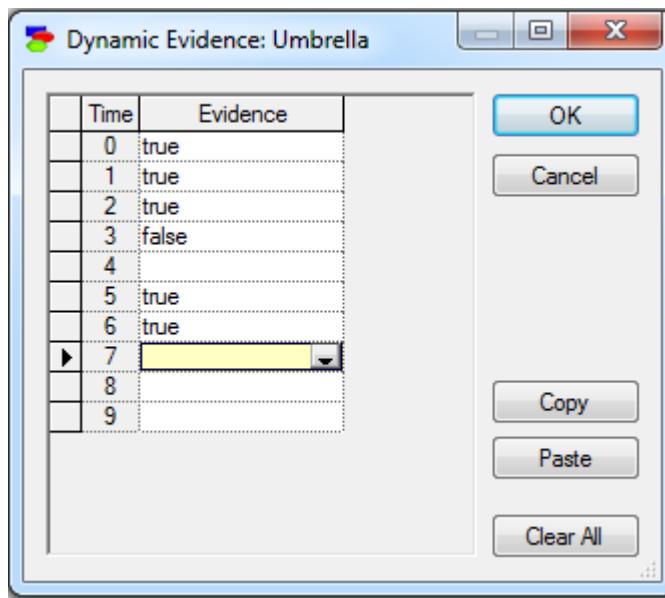
To enter this evidence, we right-click on the *Umbrella* node and select *Evidence...*



This invokes the *Dynamic Evidence Dialog*



We enter the evidence vector as specified above:



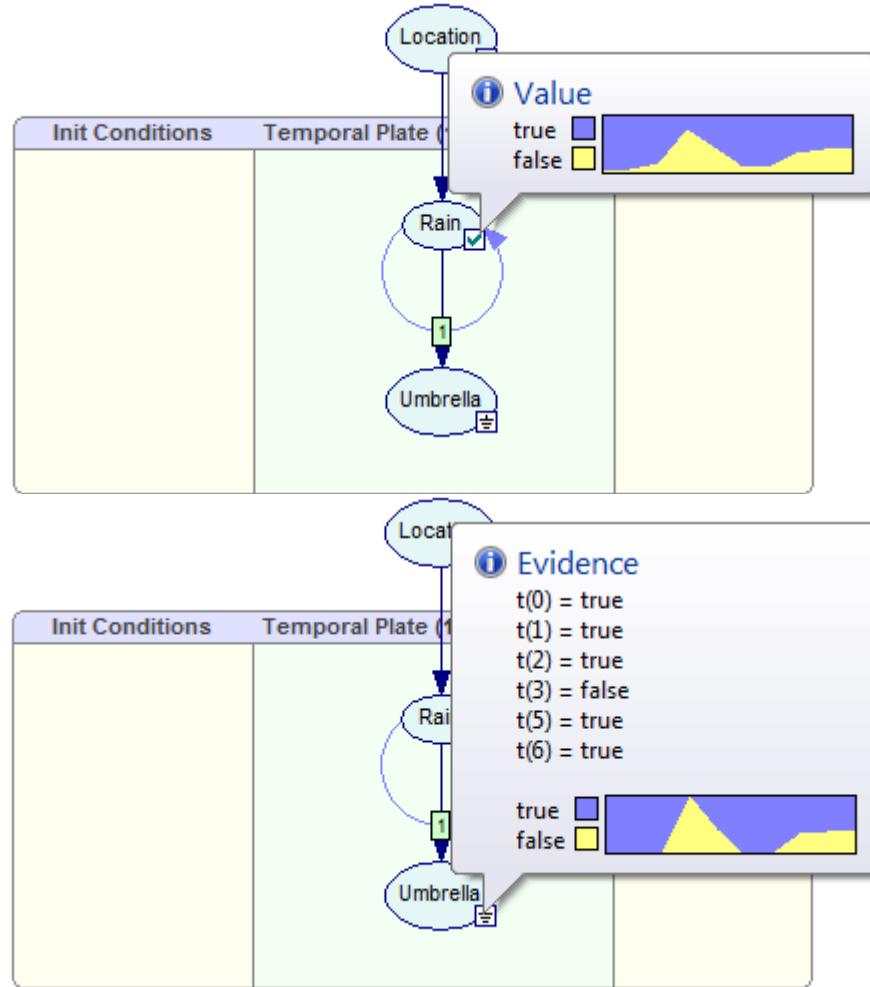
Running the belief updating algorithm

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. GeNle 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

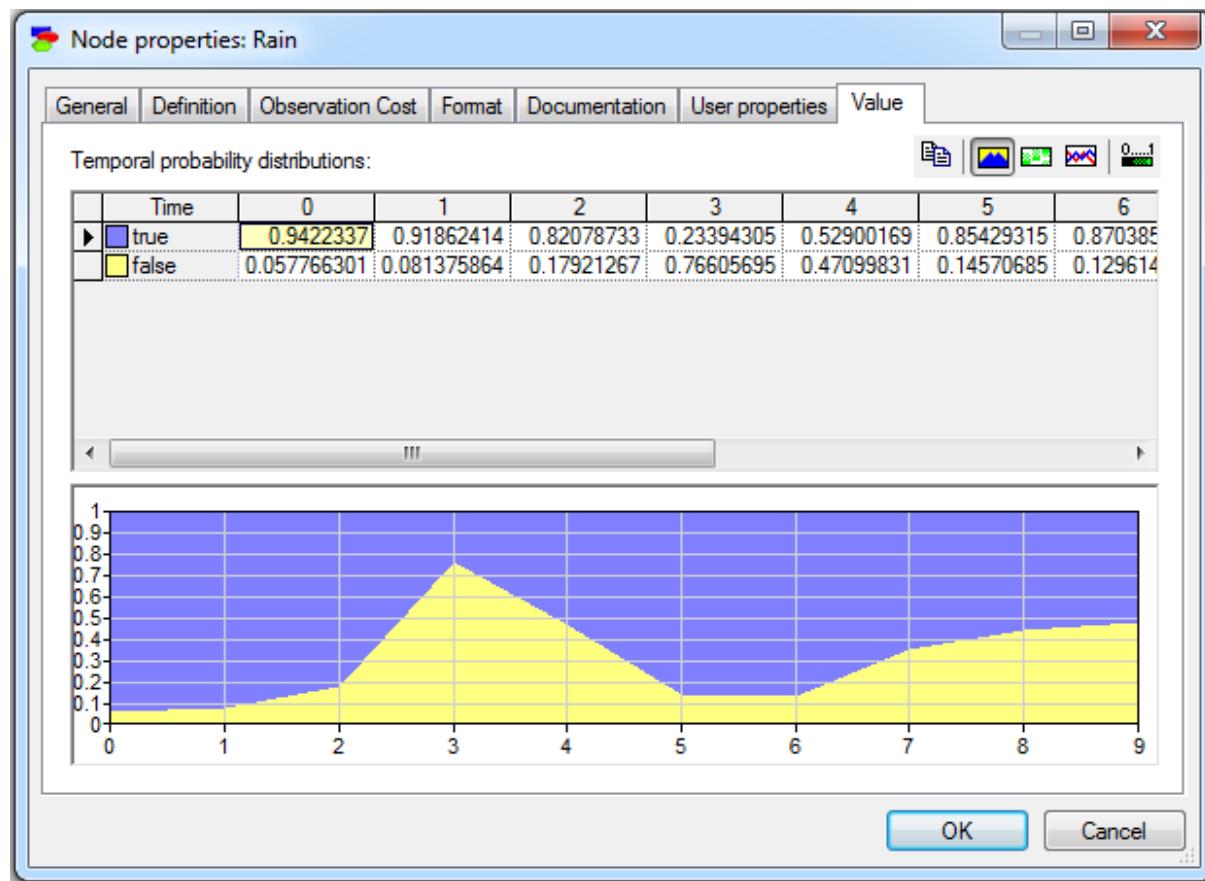
After the network has been updated, we can view its temporal beliefs, which are marginal posterior probability distributions as a function of time. Hovering the mouse over the status icon yields the following views:



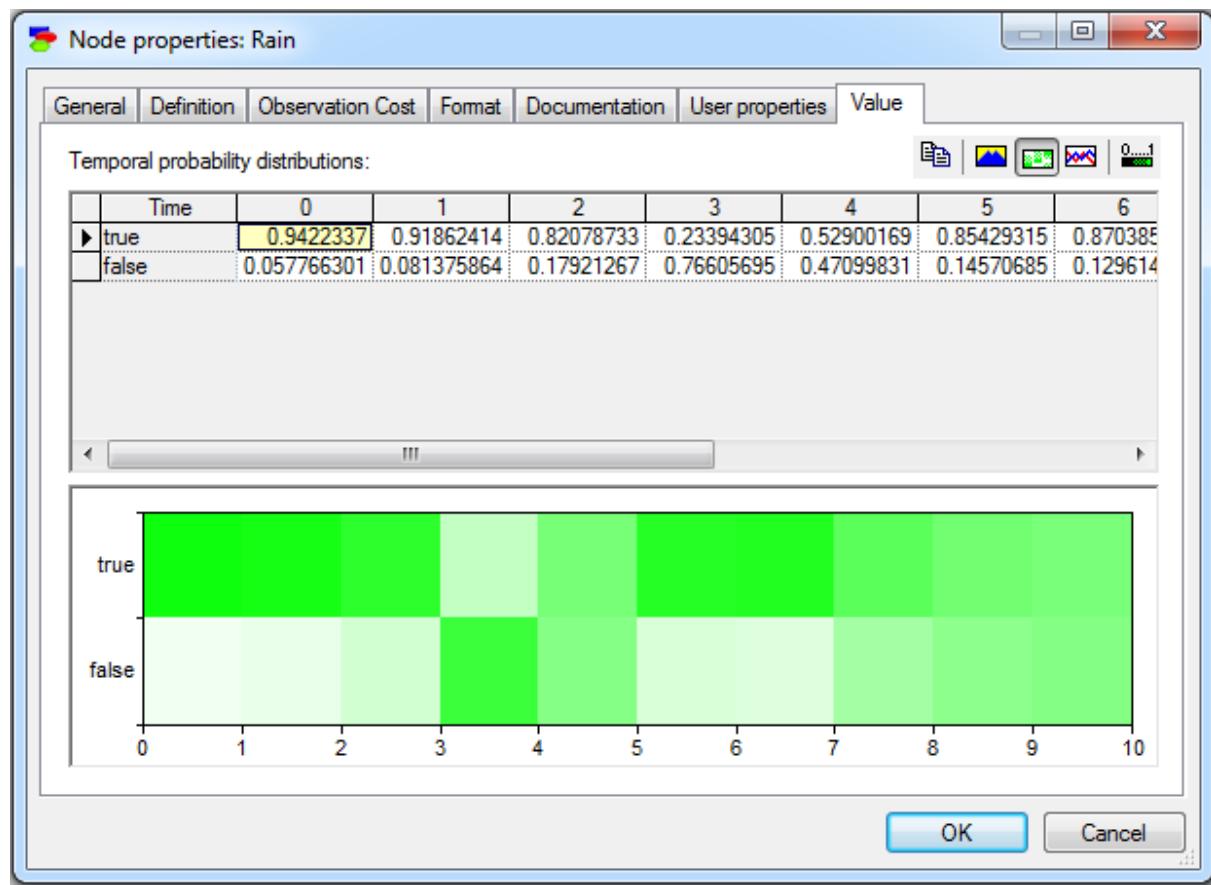
Please note that the *Umbrella* node is an evidence node. Its temporal beliefs are also well defined, albeit for those time slots for which there are observations, they are constant.

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.

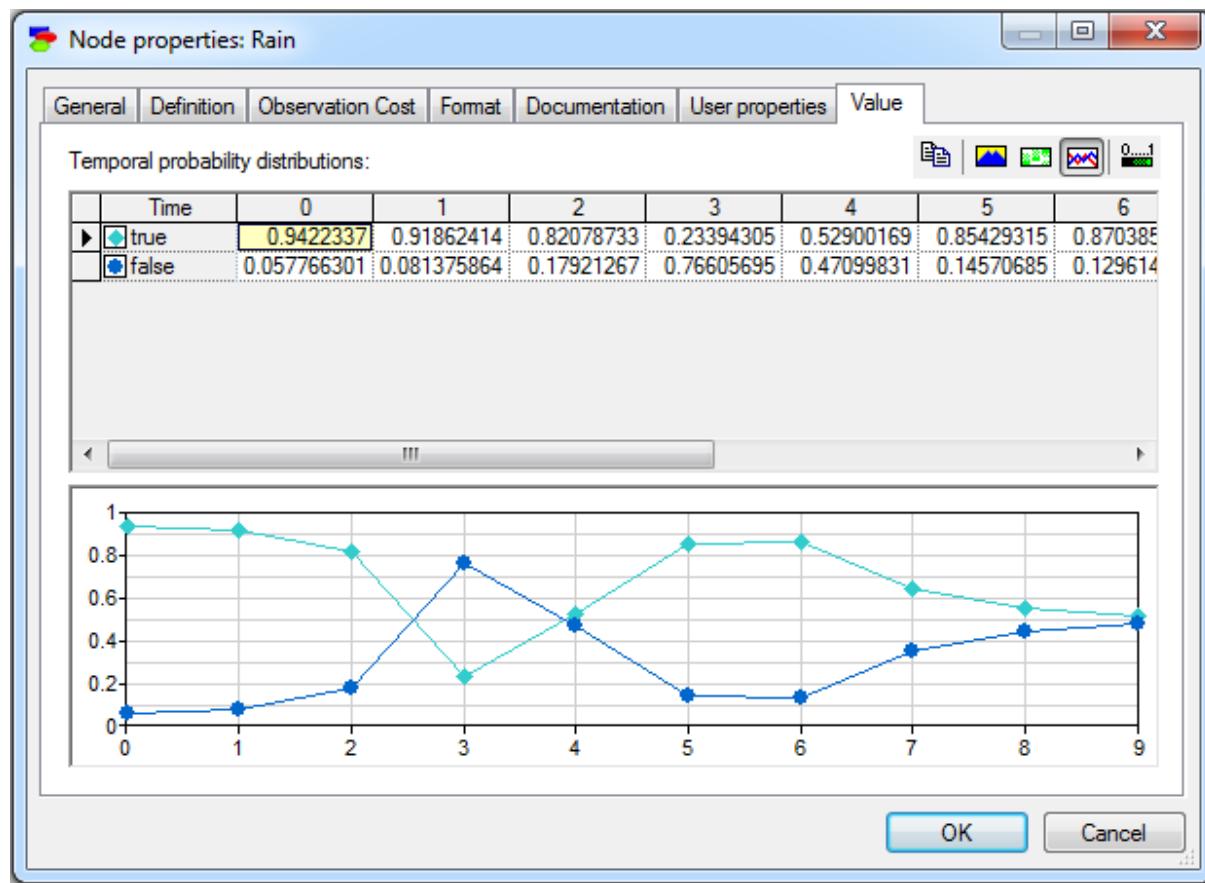
In addition to the numerical values of the marginal posterior probabilities over time, we can view the results graphically. Pressing the *Area chart* () button displays the posterior marginal probabilities graphically:



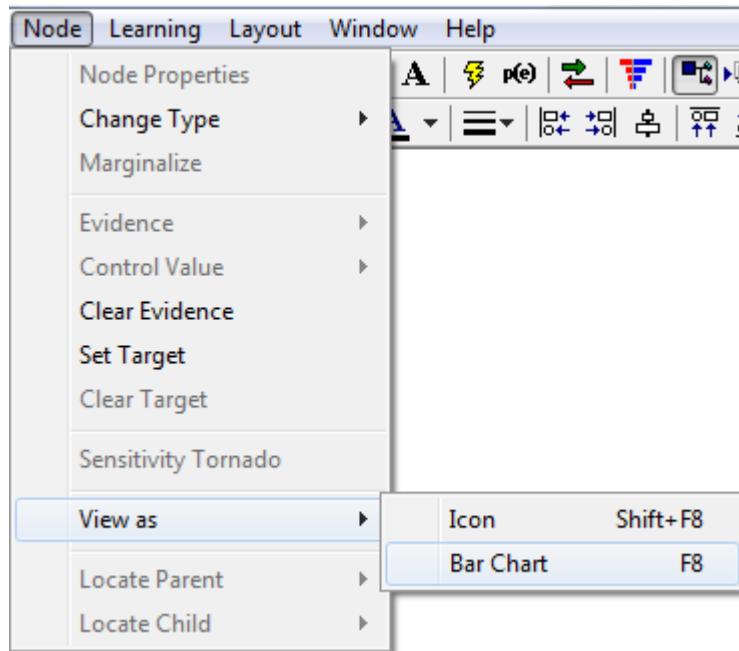
Pressing the *Contour plot* (button) displays the posterior marginal beliefs as a contour plot with probabilities displayed by colors. Hovering over individual areas shows the numerical probabilities corresponding to the areas/colors. The *Contour plot* is especially useful when the variable has many states and shows graphically the weight of probability mass.



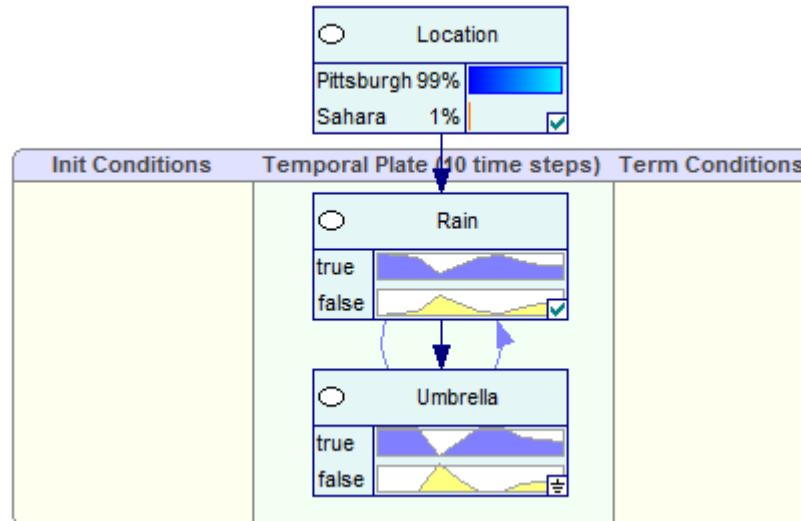
Finally, pressing the Time series () button shows the posteriors as a time series plot (a curve for every state of the variable):



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.

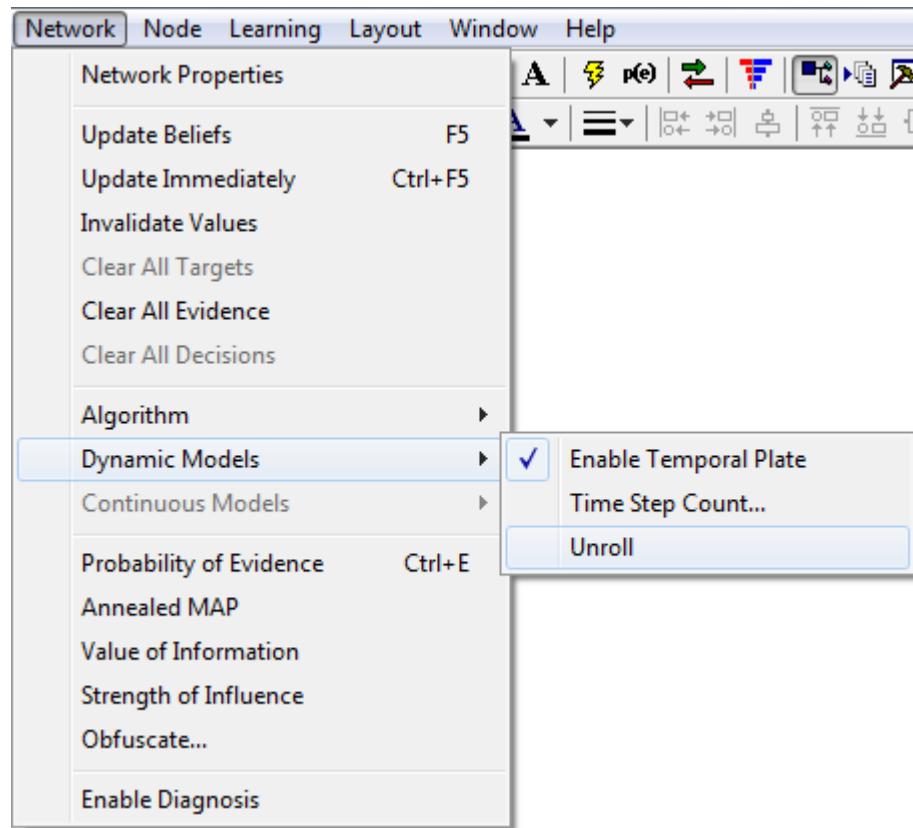


The Bar chart view allows for displaying the temporal posterior marginal probabilities on the screen permanently.

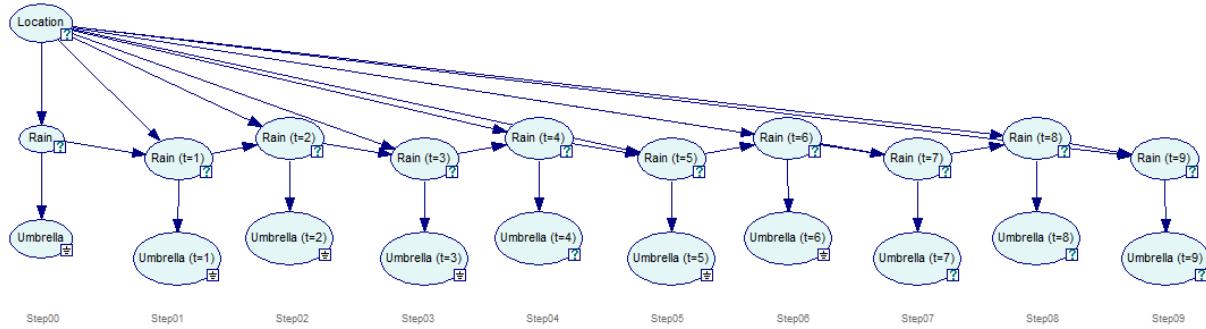


Unrolling the DBN

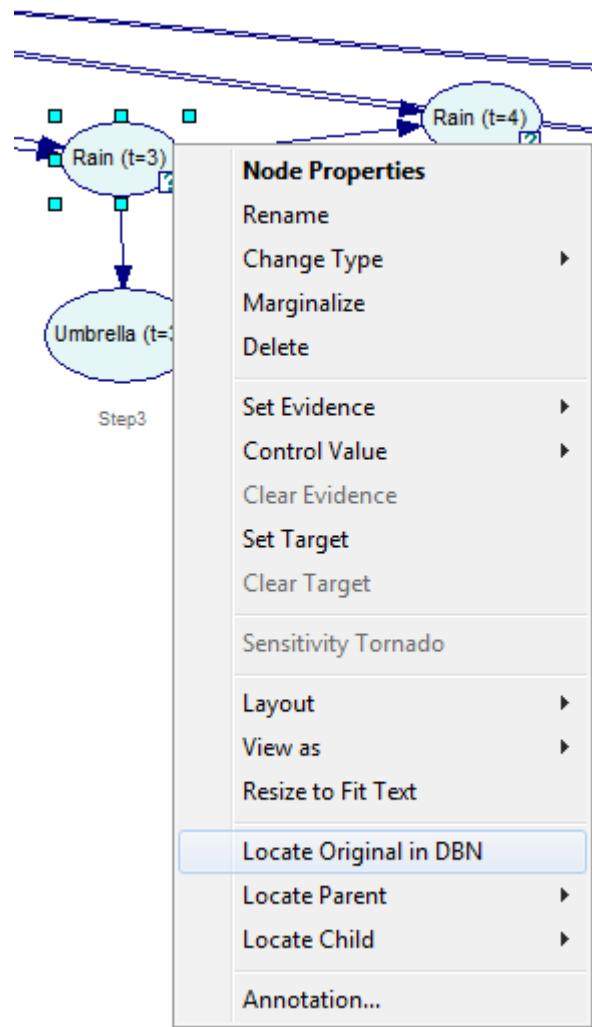
As we mentioned above, for the purpose of inference, GeNle converts the DBN into a Bayesian network and updates the beliefs using the selected belief updating algorithm. It can be useful, for example for model debugging purposes, to explicitly unroll a temporal network. GeNle provides this possibility through the *Network-Dynamic Models-Unroll* option.



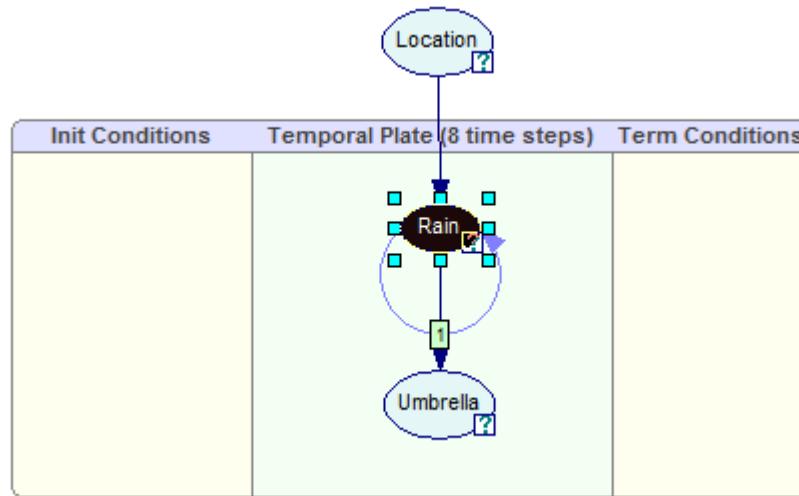
GeNle creates a new network that has the temporal network unrolled for the specified number of time-slices. It is possible to locate a node in the temporal network from the unrolled network by right-clicking on the node in the unrolled network and selecting -> Locate Original in DBN from the context-menu. The 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. Figure 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 find the corresponding DBN node for any node in the unrolled network.



The node will be identified in the original DBN:



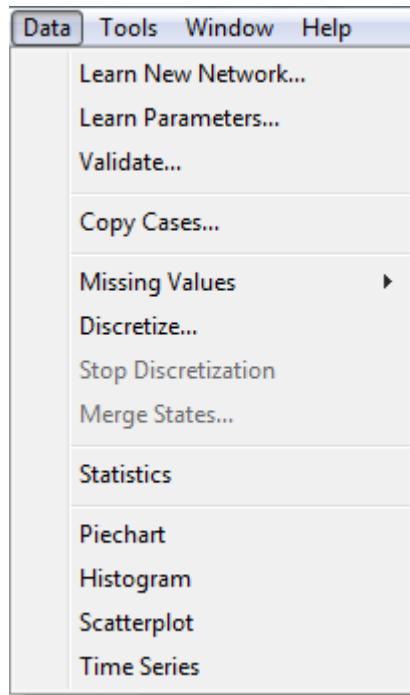
6.6.4 Learning DBN parameters

While GeNIE structure learning algorithms do not allow for learning the structure of dynamic models, it is possible to learn the parameters of DBNs from time series.

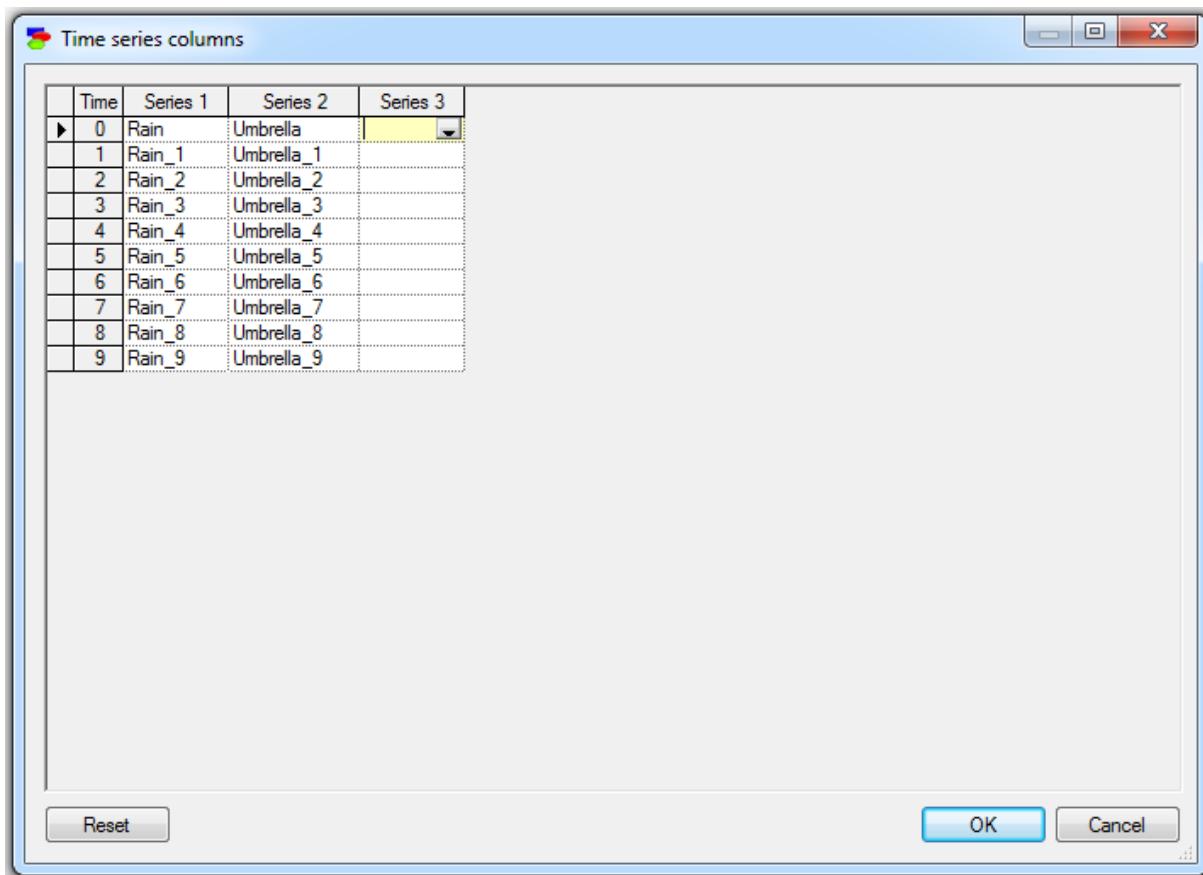
To learn parameters of an existing dynamic Bayesian network (i.e., one for which the structure is already defined), you will need both, a data file and a network open. We will demonstrate the procedure of learning the parameters of a dynamic Bayesian network from data on the network created in the section [Creating DBNs](#)⁴³³ and a corresponding data file, both available in the example models directory. Here is a screen shot of the data file opened in GeNIE:

Please note that 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).

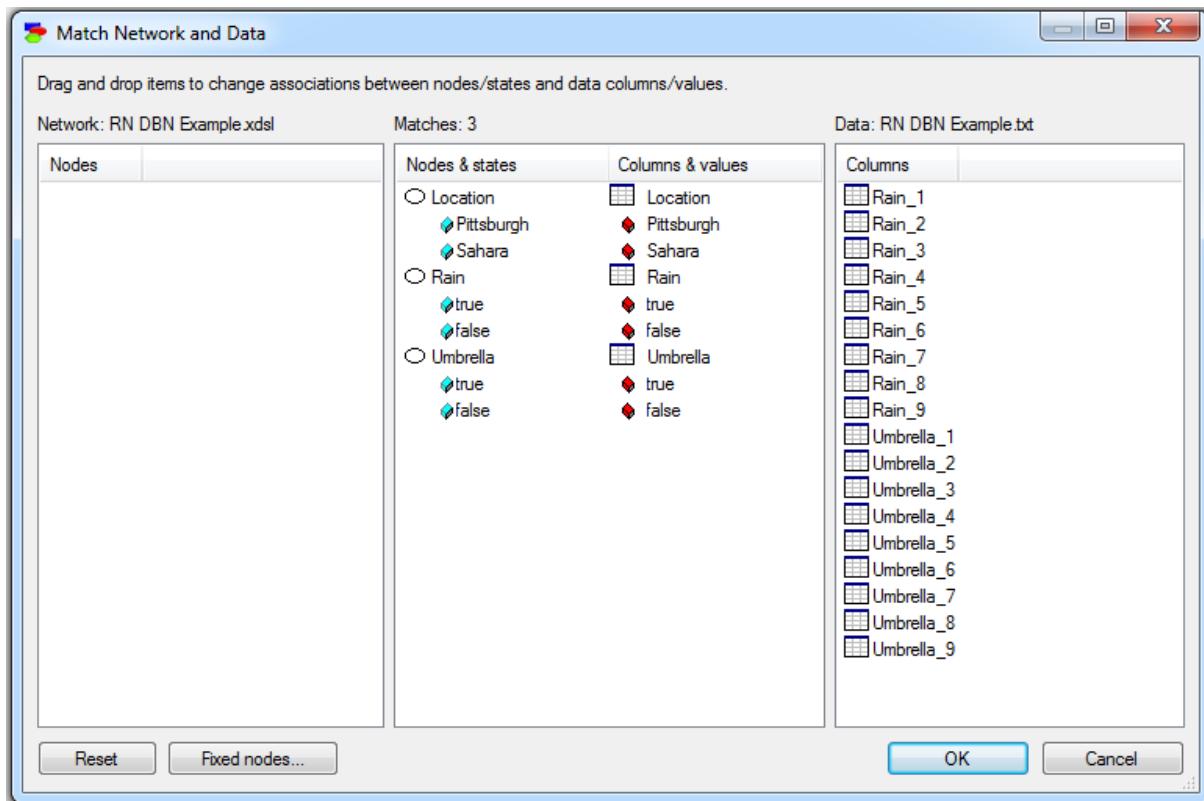
Once you have opened both the model and the data, select *Data-Learn Parameters...*



This will invoke the *Time series columns* dialog that allows for double-checking the matching between time steps and labels in the data set.

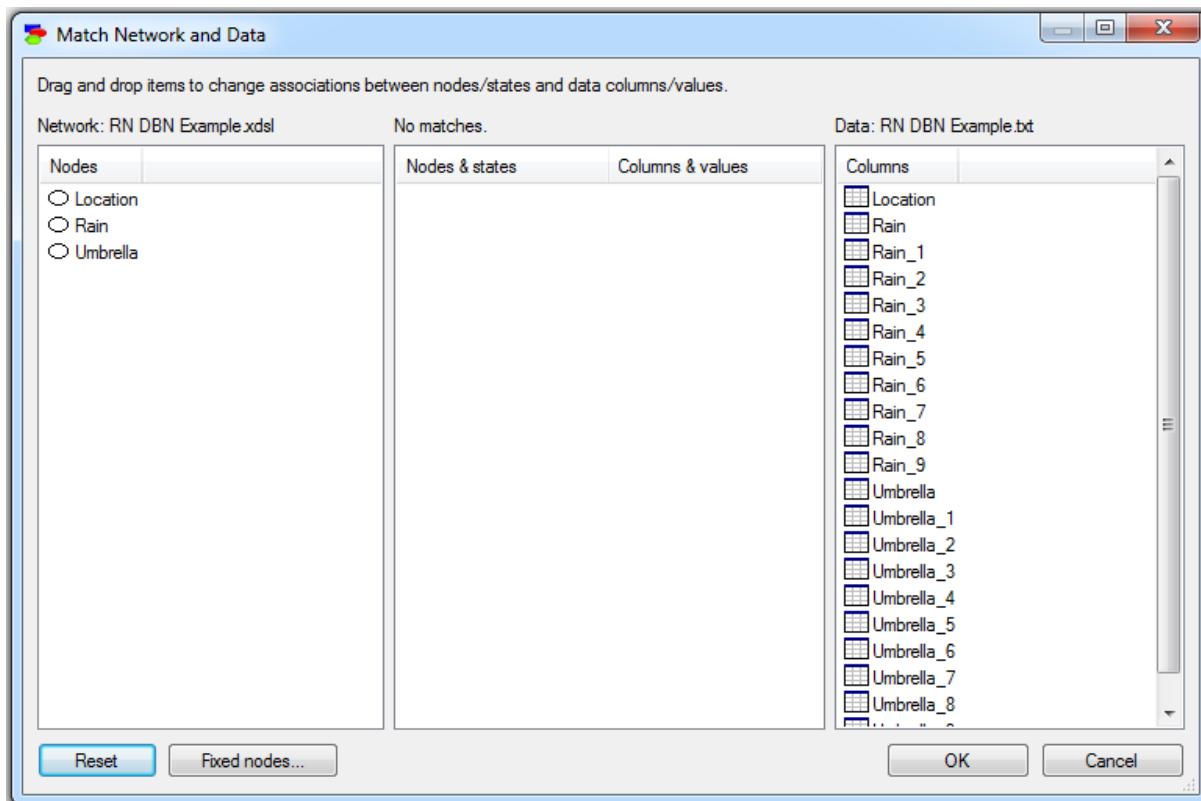


Closing this dialog invoke the *Match Network and Data* dialog that serves to create a mapping between the variables defined in the network (left column) and the variables defined in the data set (right column).



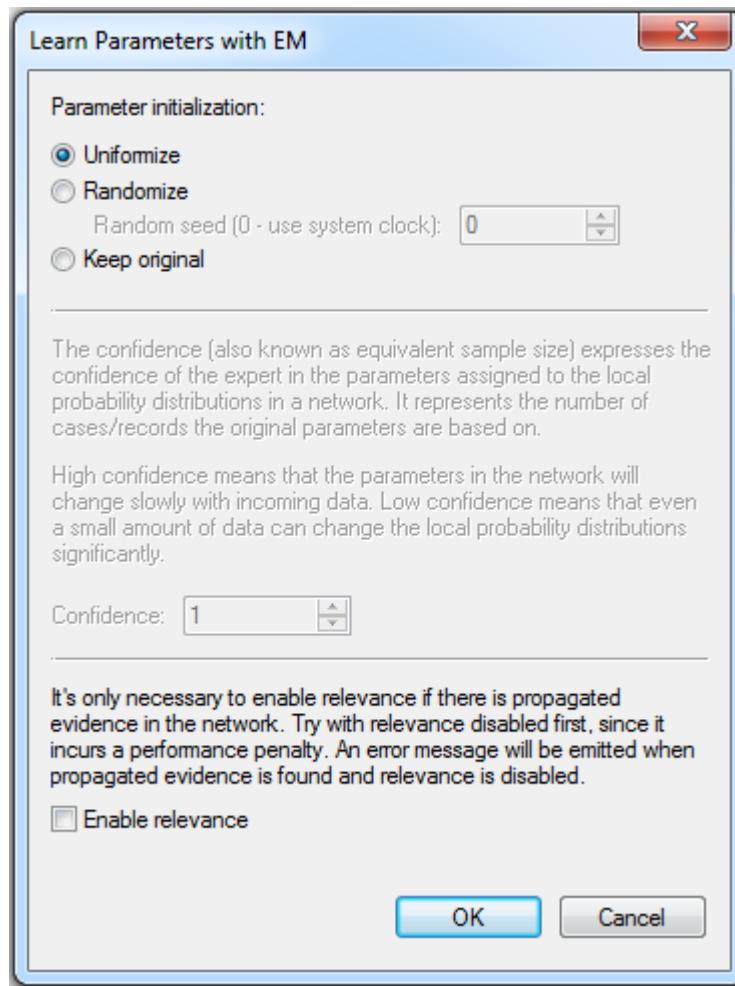
Both lists of variables are sorted alphabetically. The *Match Network and Data* dialog does text pre-matching and places in the central column all those variables and their states that match (have identical or close to identical names). If there is any disparity between them, GeNIE highlights the differences 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:



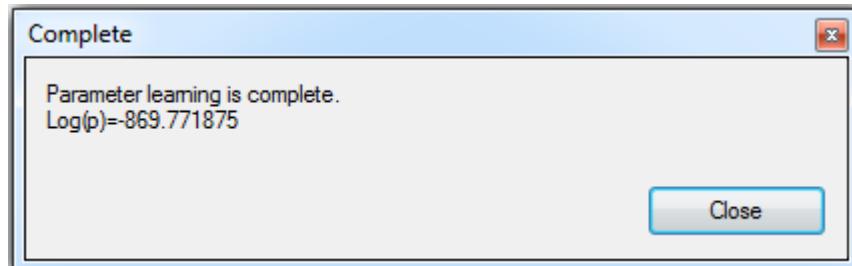
Fixed nodes... button, described in detail in section [Learning parameters](#)⁴⁰⁰, invokes a dialog that allows for excluding nodes from the learning process.

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](#)⁴⁰⁰.

Once we press *OK*, the EM algorithm updates the network parameters following the options chosen and comes back with the following dialog:



$\text{Log}(p)$, 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.

6.7 Equation-based models

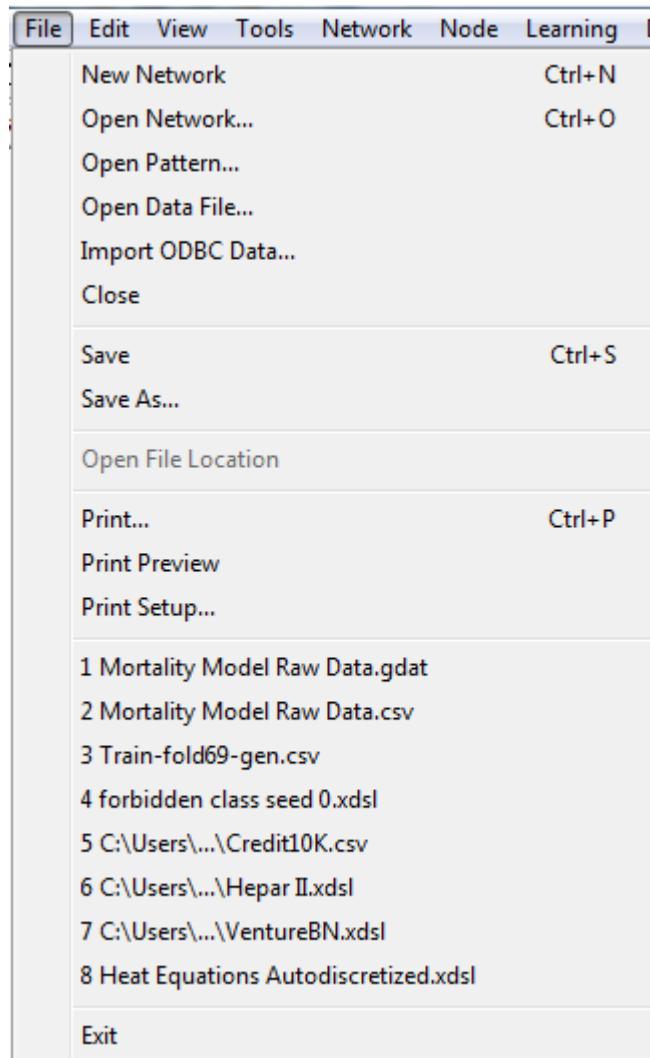
6.7.1 Introduction

It is often forgotten that graphical models, such as Bayesian networks, are not necessarily consisting of only discrete variables. They are, in fact, close relatives of systems of simultaneous structural equations. GeNle allows for constructing models consisting of equation nodes that are alternative, graphical representations of systems of simultaneous structural equations.

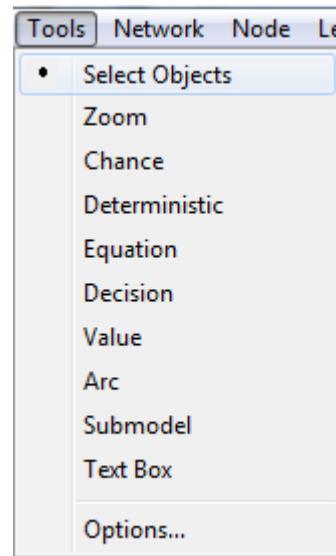
The following sections cover the process of constructing, inference, and viewing results in equation-based models.

6.7.2 Constructing equation-based models

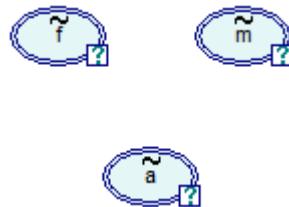
Equation-based models consist of *Equation* nodes. To construct an equation-based model, add *Equation* nodes to the *Graph View* and add connections between them. Let us create a simple model describing an object of mass m under influence of a force f , receiving acceleration a , which is governed by Newton's 2nd law of motion. We start by selecting *New Network* from the *File* menu



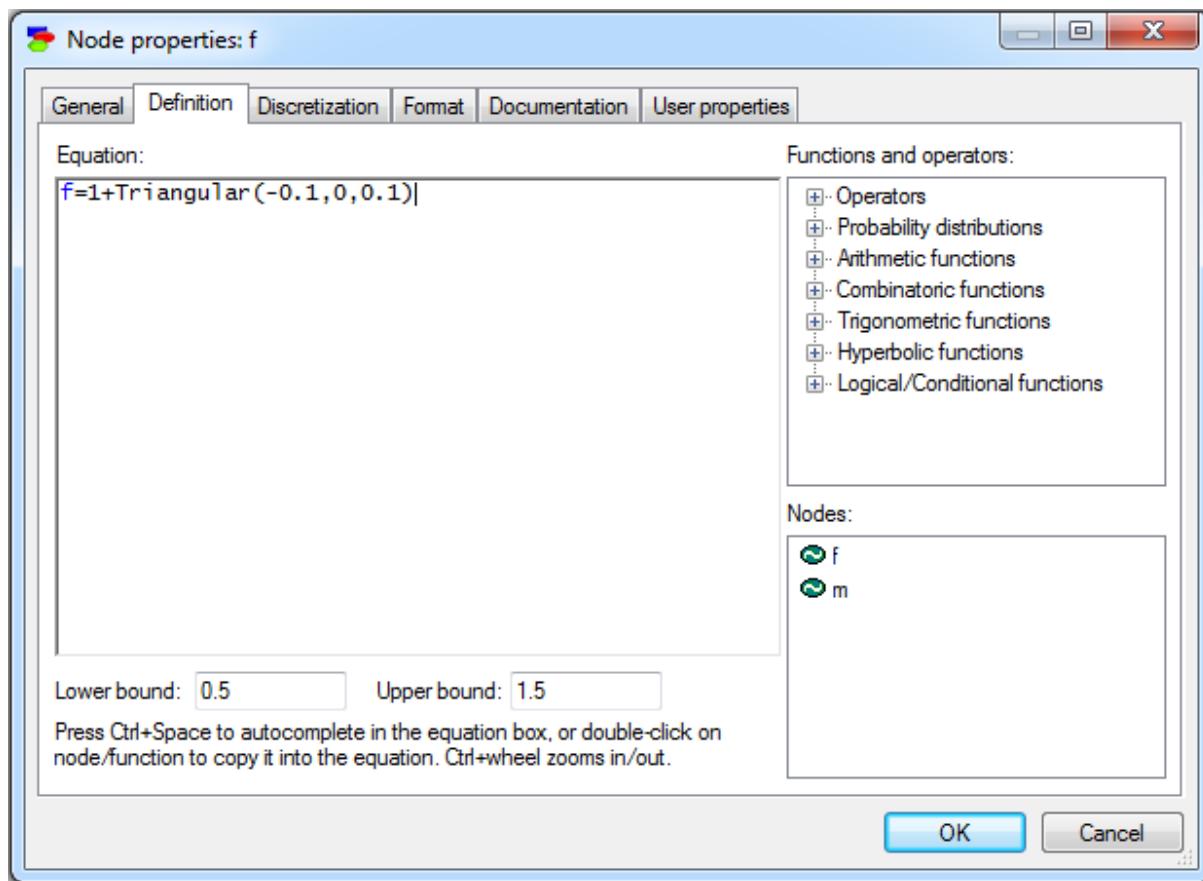
We proceed by dropping three *Equation* nodes in the *Graph View* window using the *Tools* menu



or the *Equation* () button from the [Standard Toolbar](#)¹⁷⁶. We name them f , m , and a .

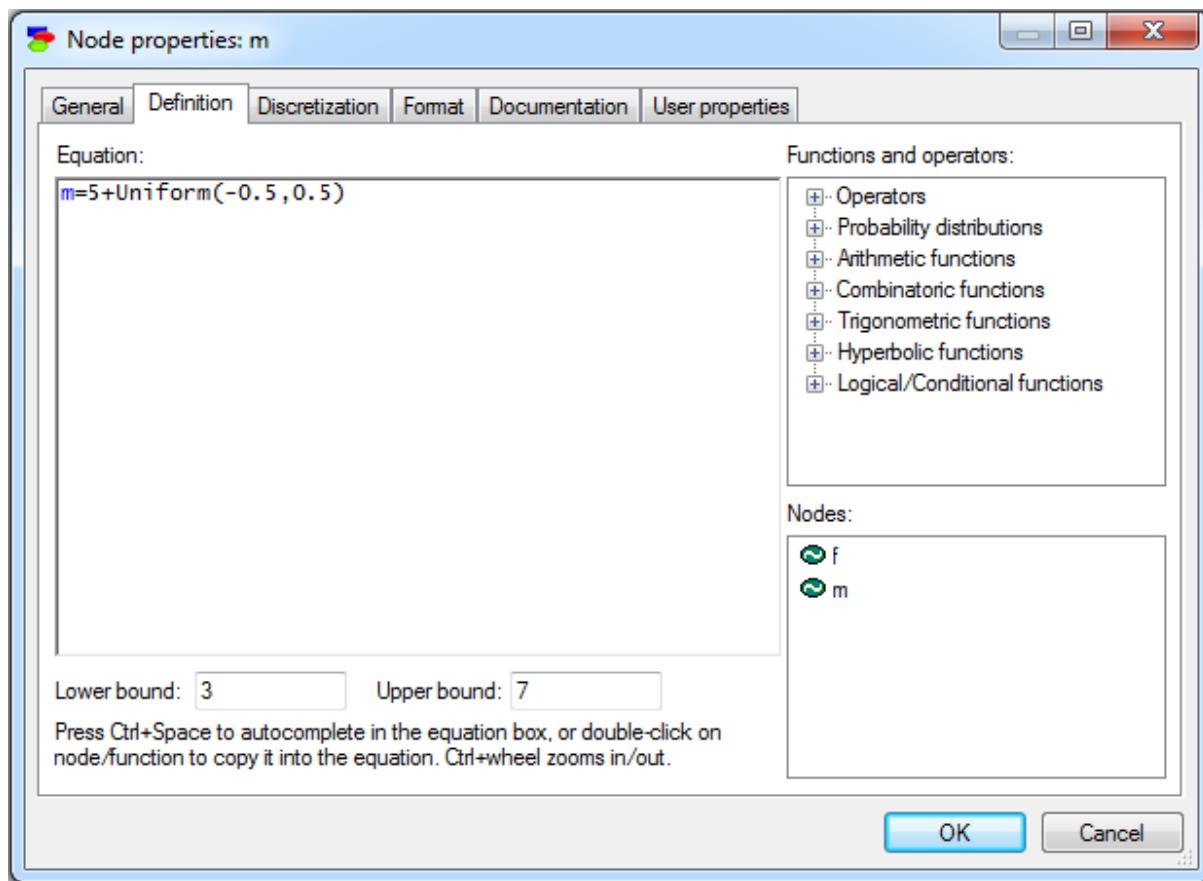


We define them as follows. Let the force be a constant ($f=1$) with some noise that we express by means of a *Triangular(-0.1,0,0.1)* distribution.



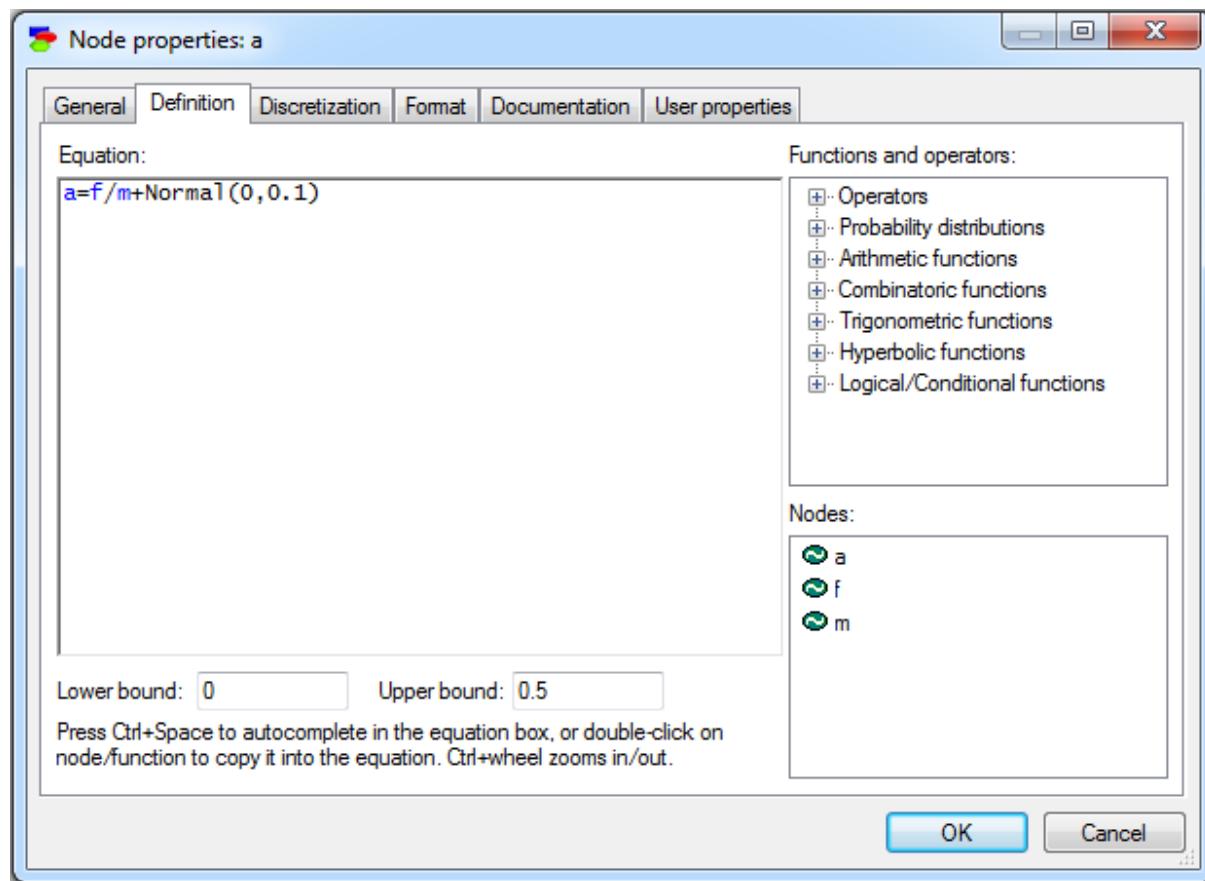
We specify the plausible domain of values of f to be between 0.5 and 1.5 Newtons.

Let the mass be a constant ($m=5$) with some noise that we express by means of a $Uniform(-0.5,0.5)$ distribution.

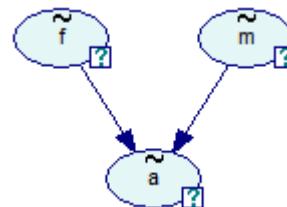


We specify the plausible domain of values of m to be between 3 and 7 kilograms.

We define the third variable, acceleration a , as a function of f and m using Newton's 2nd law of motion 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:

```

f=Triangular(-0.1,0,0.1)
m=Uniform(-0.5,0.5)
a=Normal(0,0.1)
f=1+ f
m=5+ m
a=f/m+
  
```

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. The distributions used (Triangular, Uniform, and Normal) may not be physically plausible in this example. We wanted to use them to

show that GeNle 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.

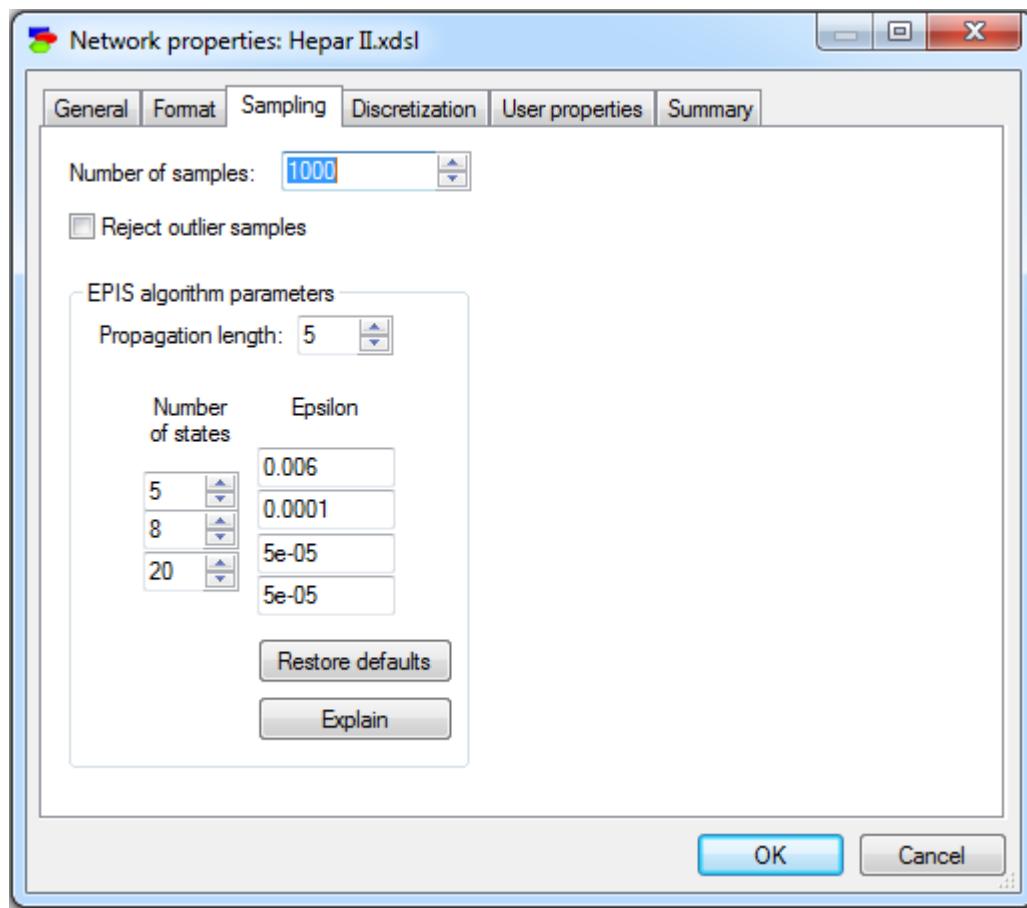
6.7.3 Inference

The fundamental class of algorithms used for equation-based 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.

Stochastic sampling algorithms

The reason why stochastic sampling algorithms are fundamental for equation-based models is that GeNle 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 GeNle 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.

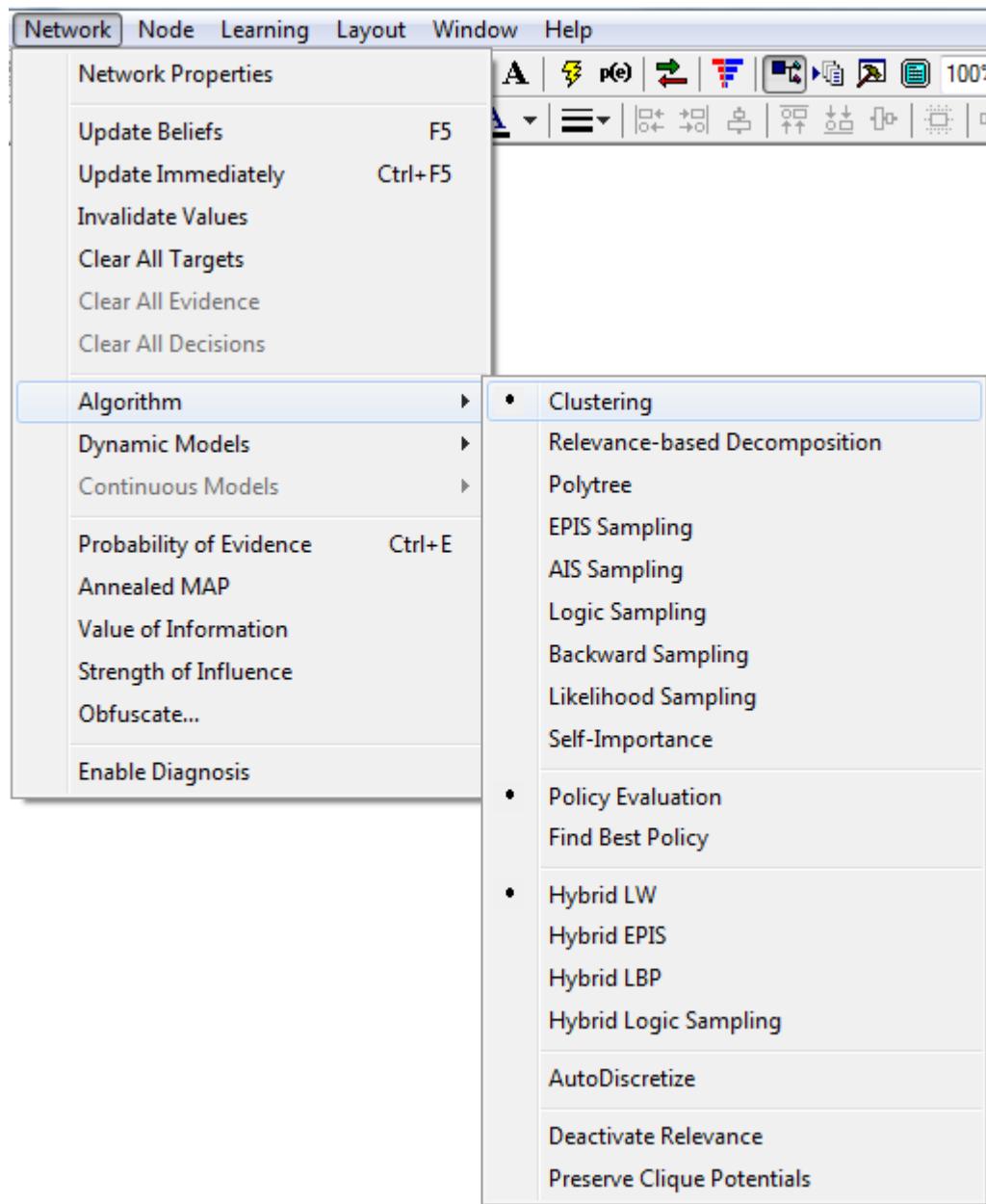
Let us start with setting the number of samples to 10,000 from the default 1,000,000. We can do this on the *Sampling* tab of the [Network properties](#)¹²³. We should generally choose the number of samples to be as large as possible given the constraints on the execution time. Execution time is pretty 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](#)¹⁷⁶. Section [Viewing results](#)⁴⁷⁰ discusses how to view and interpret the results of the algorithm.

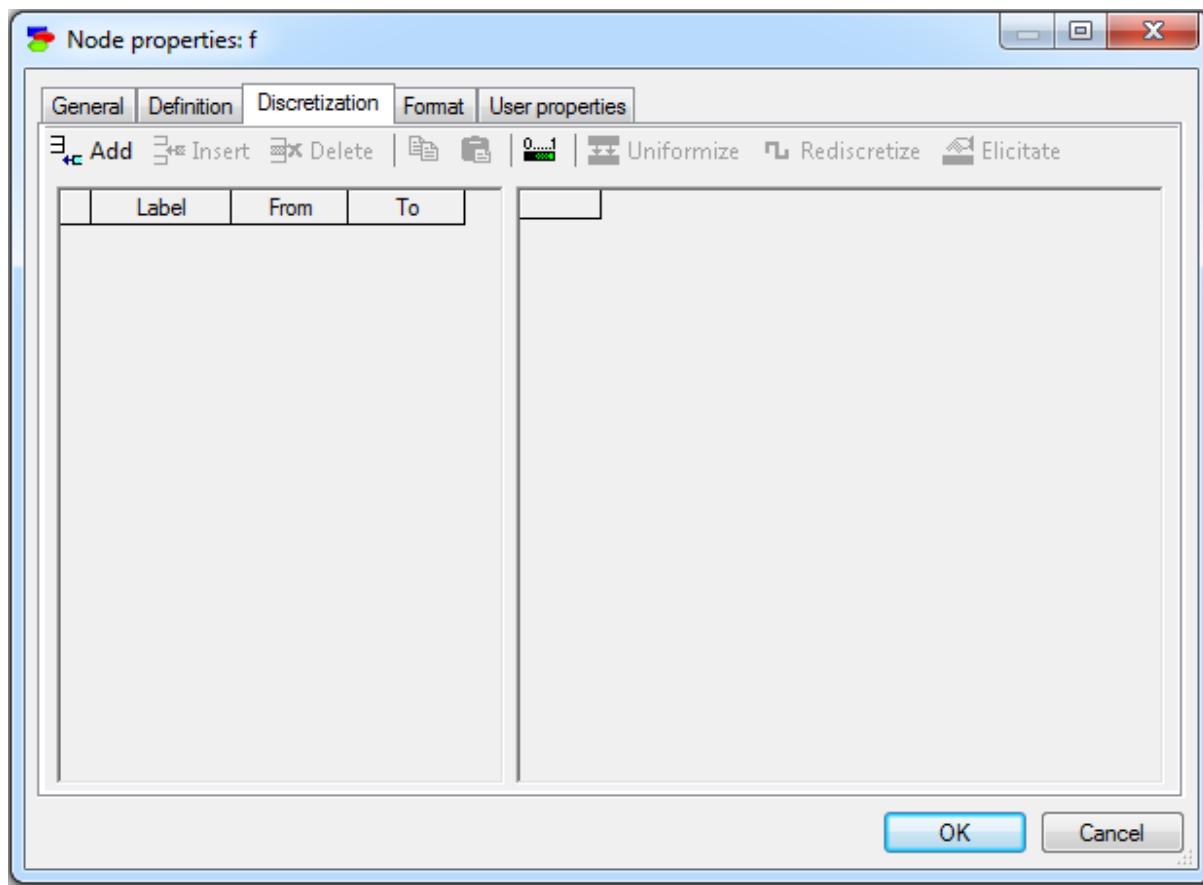
Auto-discretization

When an equation-based network contains evidence, there are no universally reliable stochastic sampling algorithms. We advise to rely on an algorithm that we call Auto-discretization. To choose this algorithm, select *AutoDiscretize* from the *Network-Algorithm* menu.

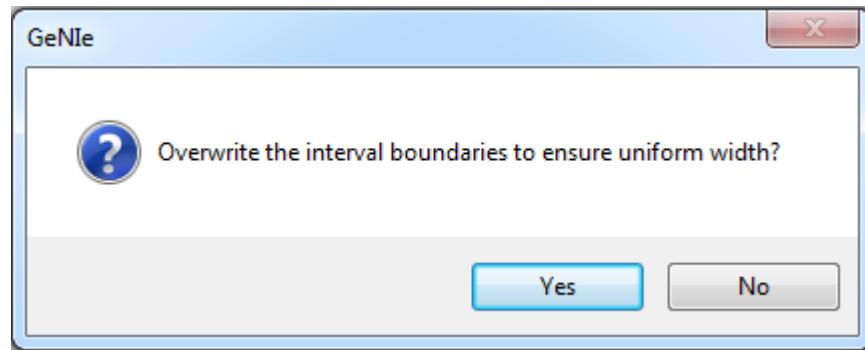


The algorithm translates the original continuous, equation-based network into a discrete Bayesian network. No changes are made to the original network definition but inference is performed in a temporary discrete Bayesian network, created solely for the purpose of inference. In order to use this algorithm, we need to enhance the definitions of the nodes in the network with a specification of the on-demand discretization.

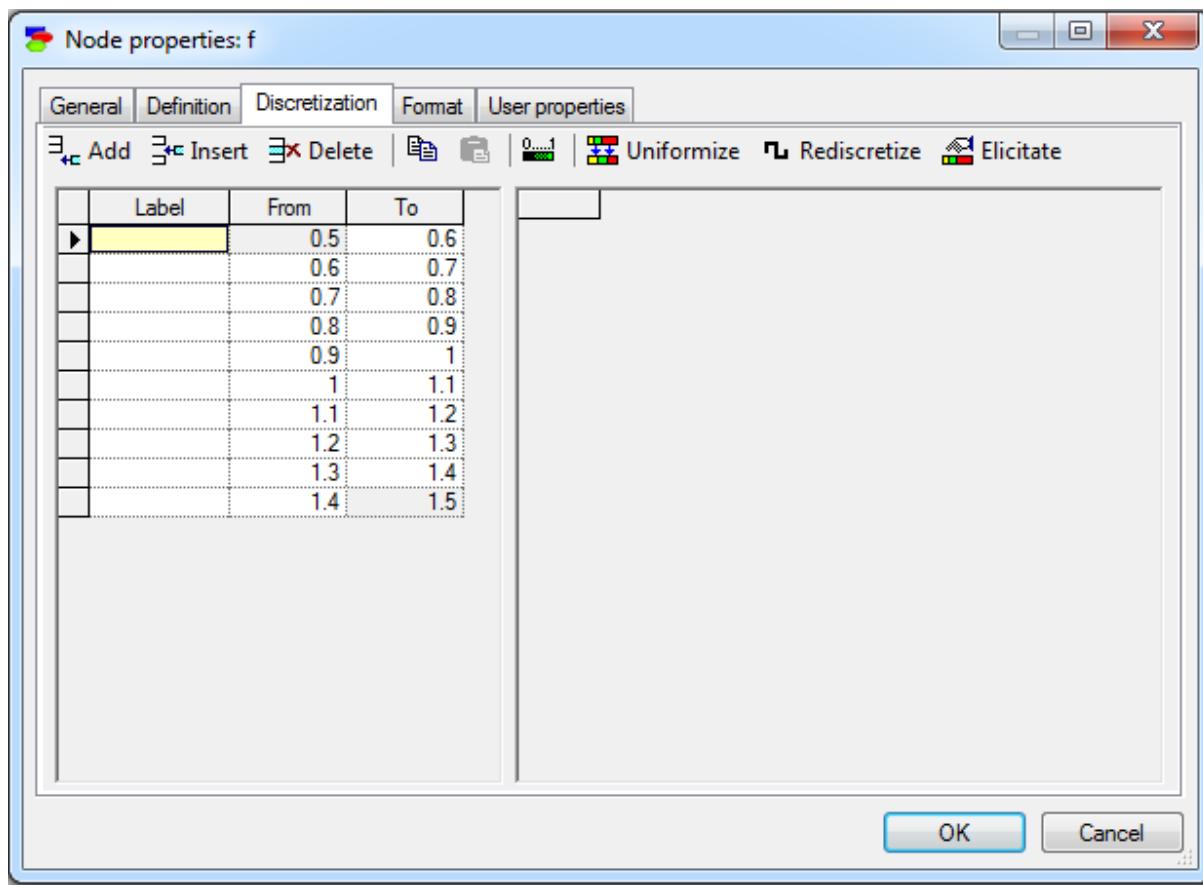
We start with node f



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* (Add) button and press *Uniformize* (Uniformize) button.

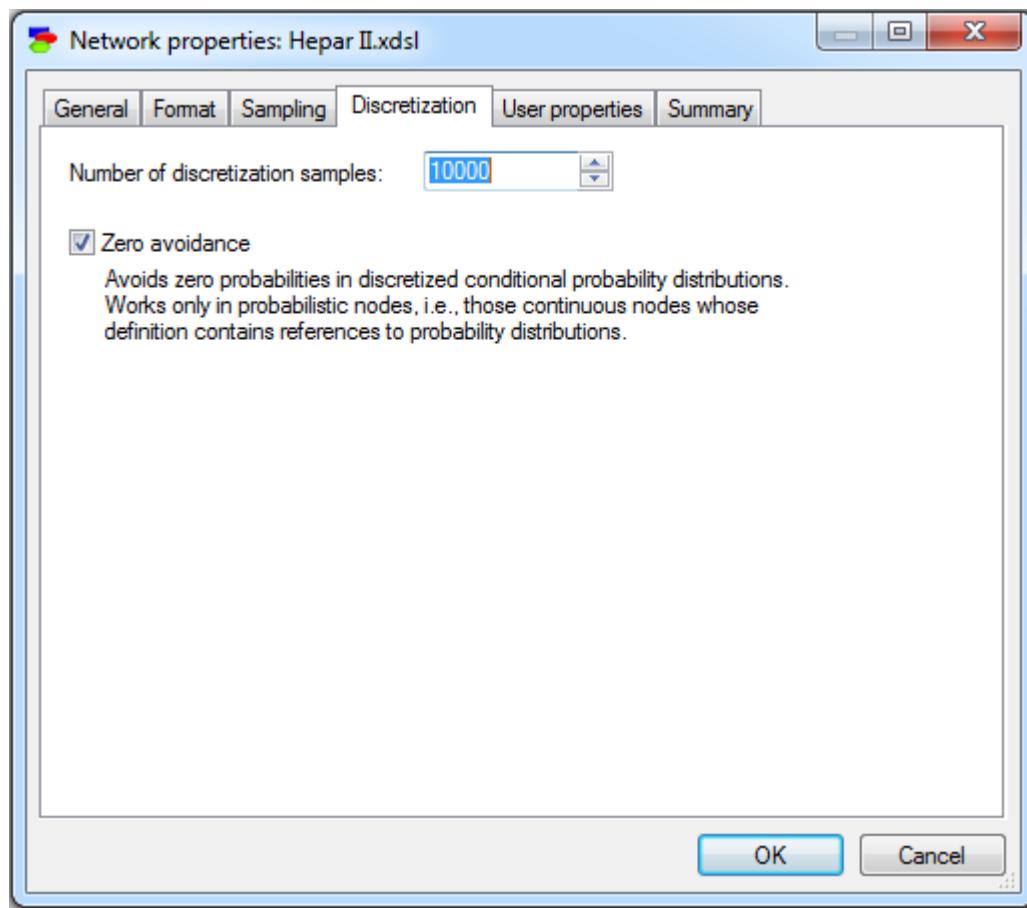


Pressing *Yes* creates new boundaries for the intervals and results in the following discretization



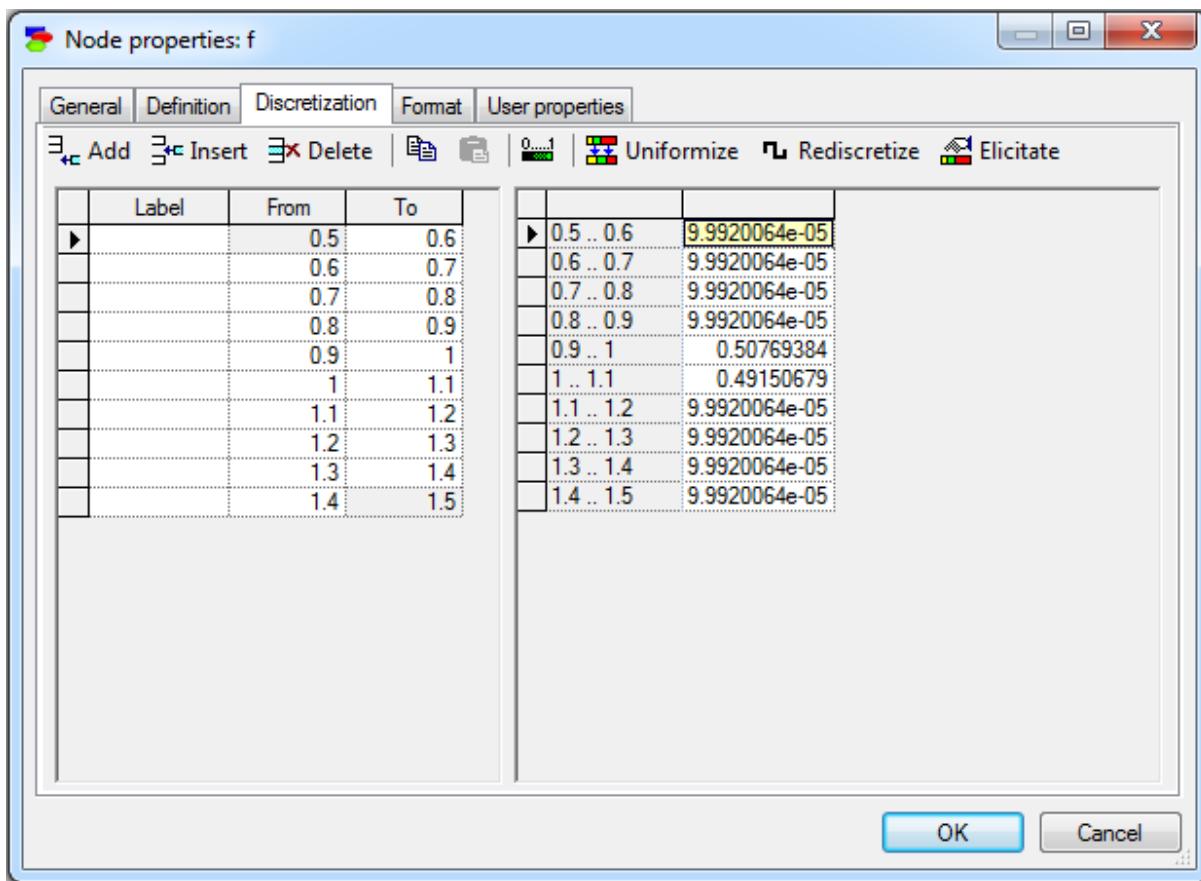
At this point, we have two choices: Elicitation and automatic derivation of the discrete distribution over the intervals from the continuous definition of the node.

Pressing *Elicitate* (**Elicitate**) shows the node's CPT and allows us to enter the probabilities manually. Pressing the *Rediscretize* (**Rediscretize**) button derives the probabilities from the definition of the node using stochastic sampling. The number of samples generated is specified in the *Network* properties, *Discretization* tab.

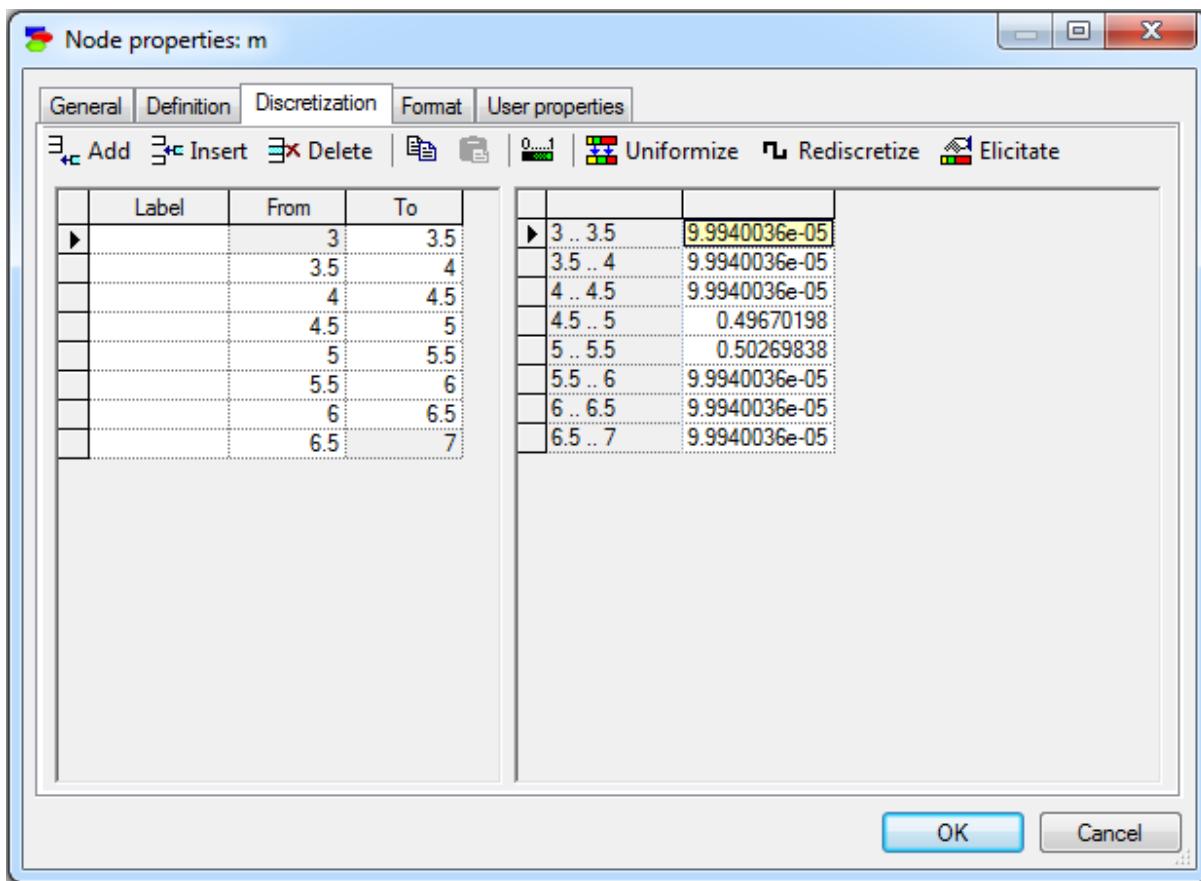


In addition to the *Number of discretization samples*, which determines the number of samples used in deriving the conditional probability tables in auto-discretized models, the tab allows the user to avoid zero probabilities in discretized conditional probability distributions. This option works only 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.

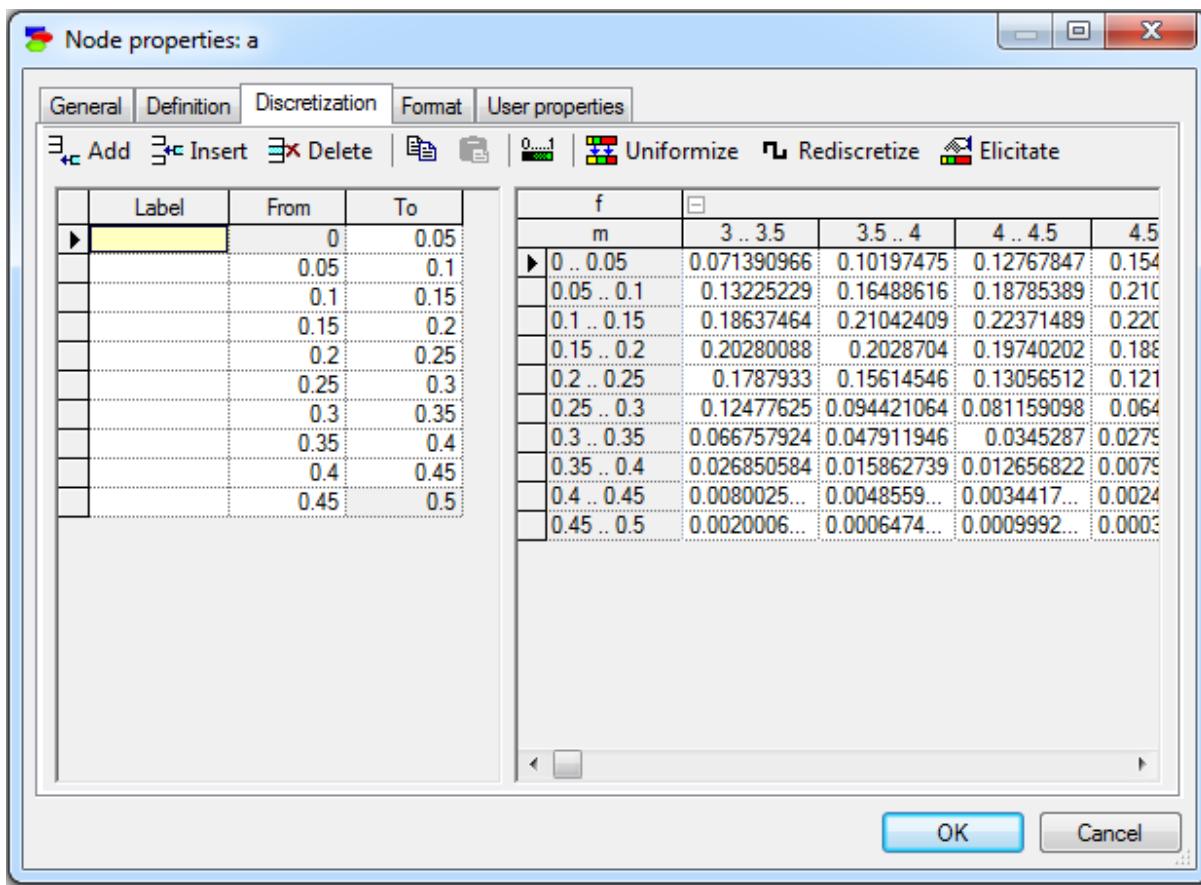
Rediscretization of the node f with 10,000 samples and zero avoidance should lead to the following discretized definition.



We repeat the same process for the variable *m* with eight intervals.



And for the variable a with 10 intervals.

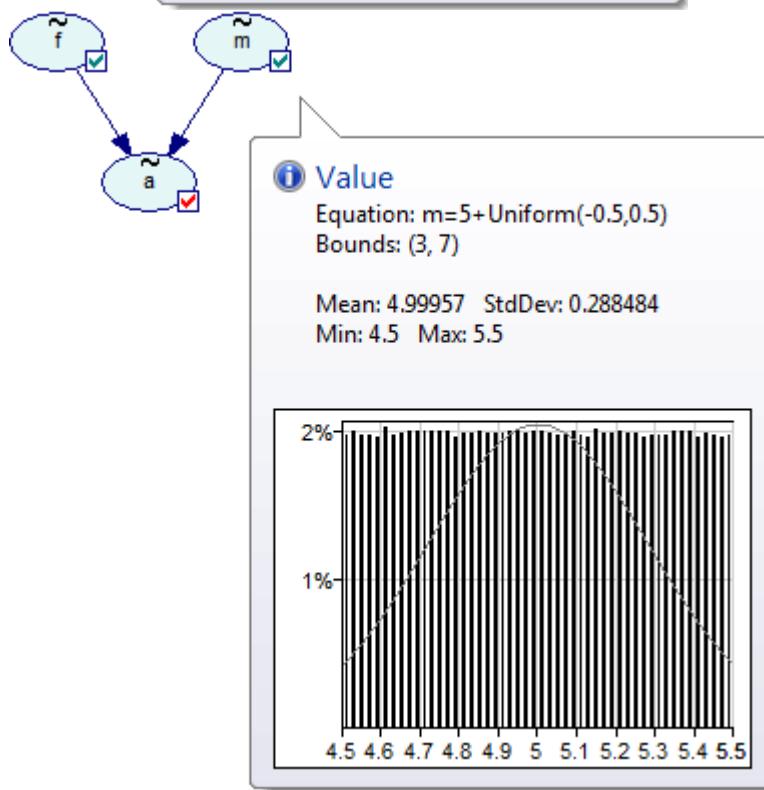
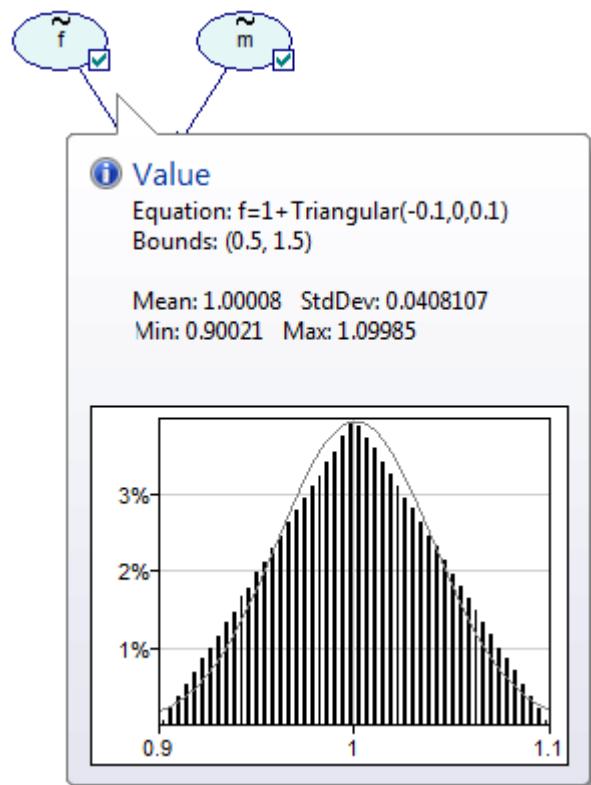


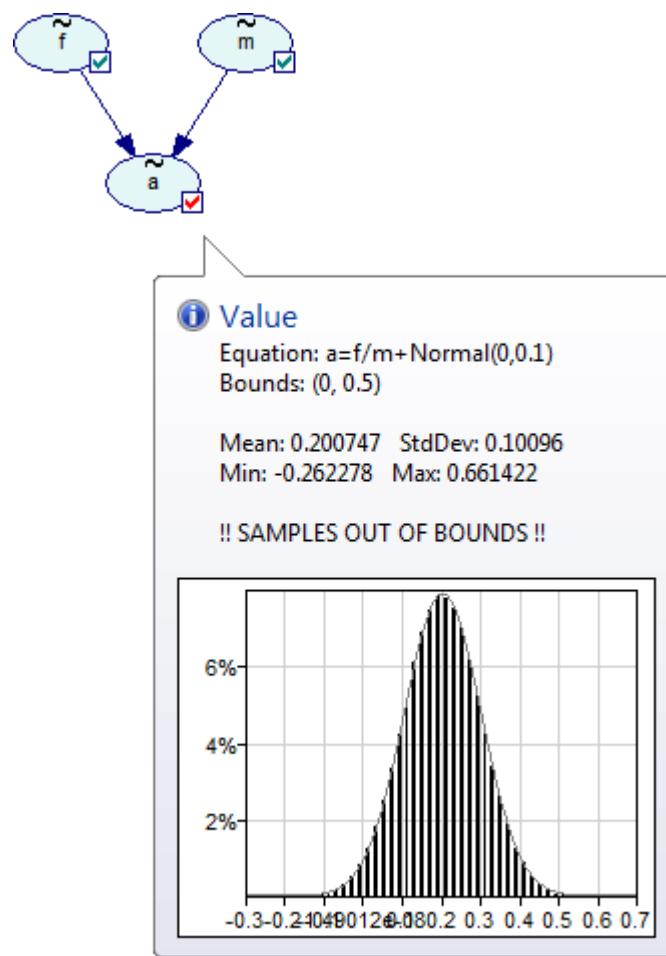
Please note that the CPT for node a is very large and contains $10 \times 8 = 80$ distributions, one for each combination of discretized values of the variables f and m . The screen shot above shows only its fragment.

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* ¹⁷⁶.

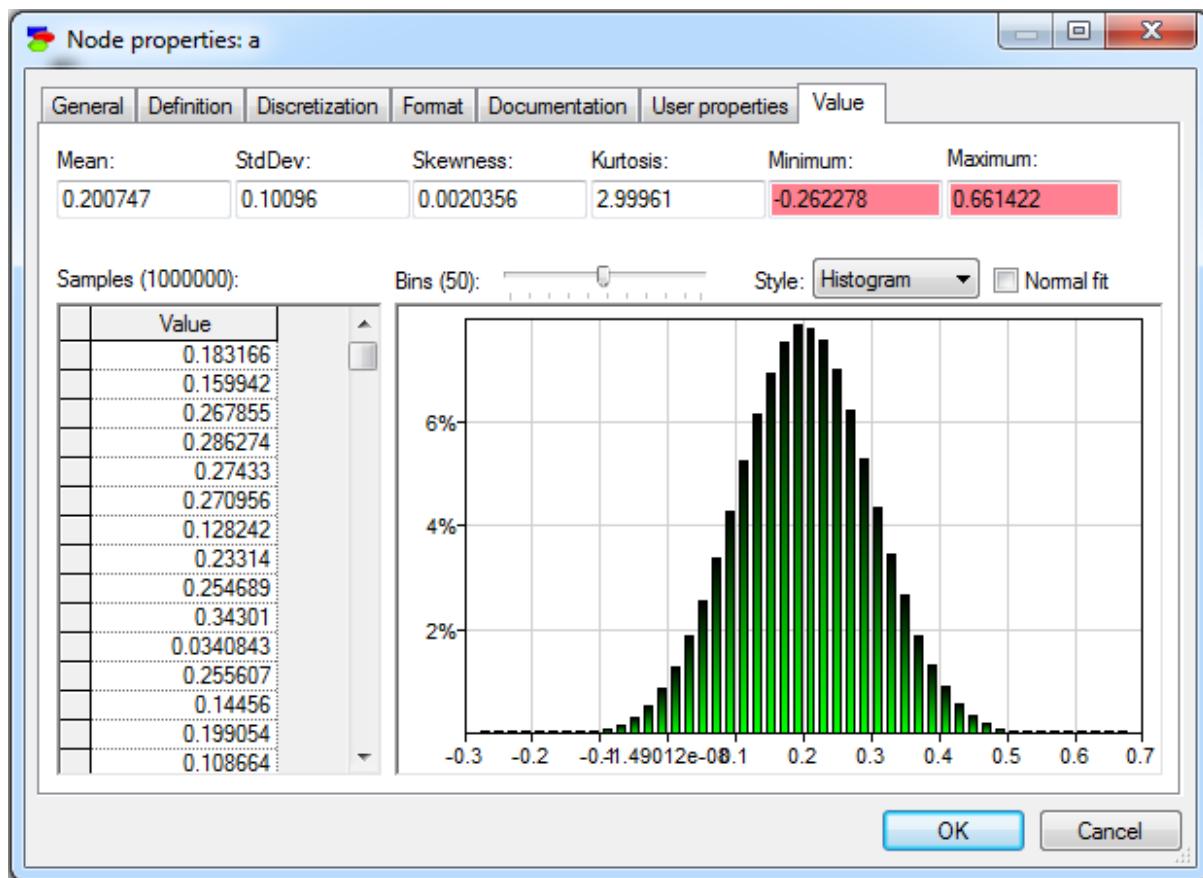
6.7.4 Viewing results

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:

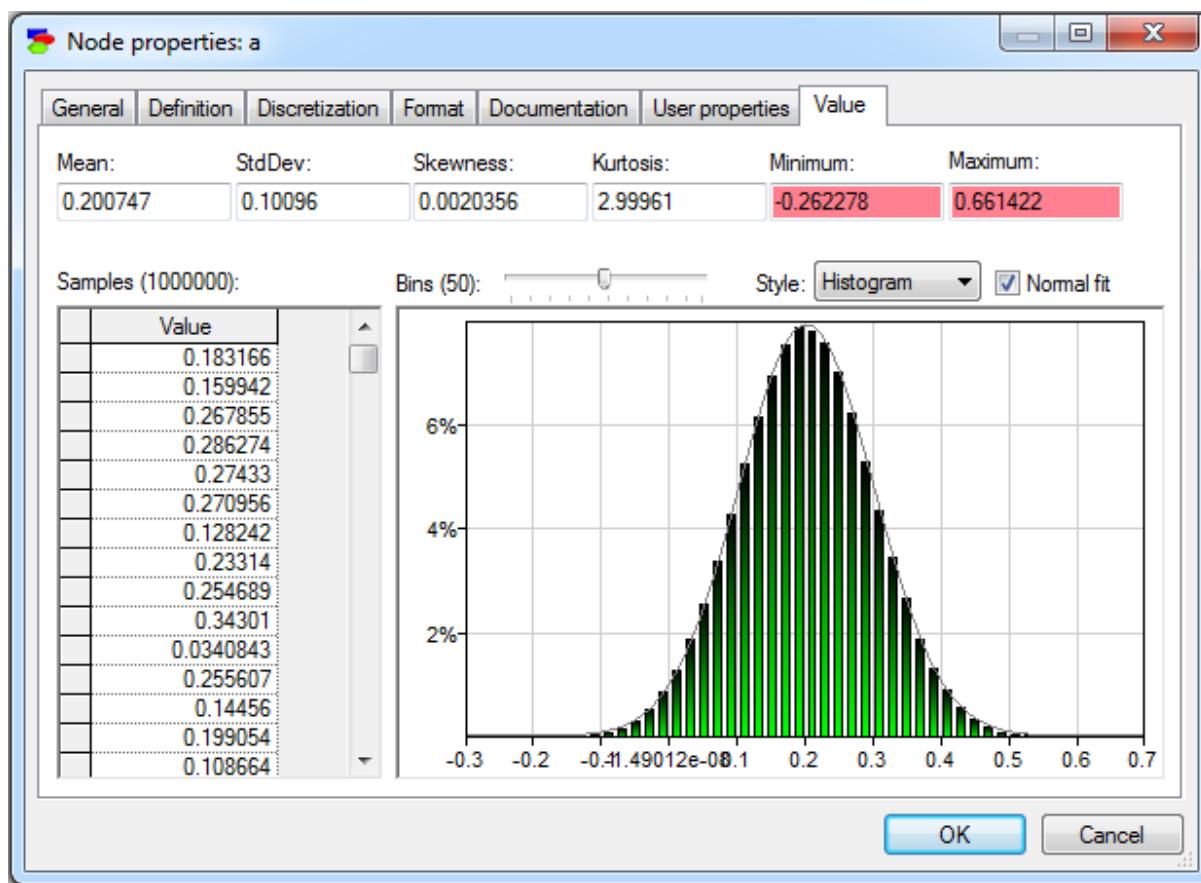




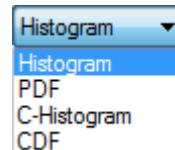
We can examine these distributions in more detail on the *Value* tab of the *Node* properties. Node a is most interesting from the point of view.



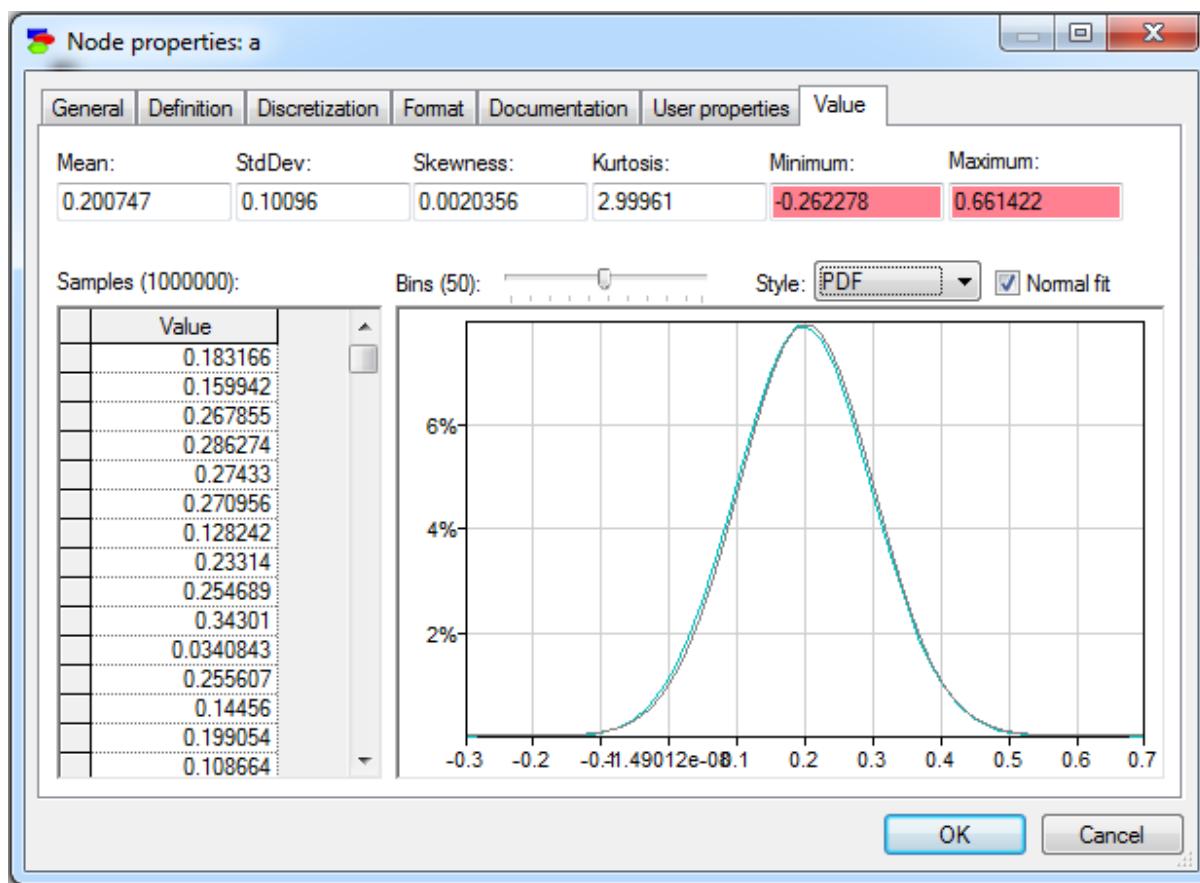
Equation nodes are continuous and show the results in form of a plot of the samples obtained during the most recent run of the sampling algorithm. The samples themselves are preserved and displayed in the vector on the left-hand side. The tab shows the first four moments of the marginal distribution over *a*: *Mean*, *StdDev*, *Skewness* and *Kurtosis* along with the *Minimum* and the *Maximum*. The last two are shown in red because they fall outside of the range designated on the definition tab ([0..0.5]). Similarly to the histogram interface in the data pre-processing module, the user can change the number of bins in the histogram. *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.



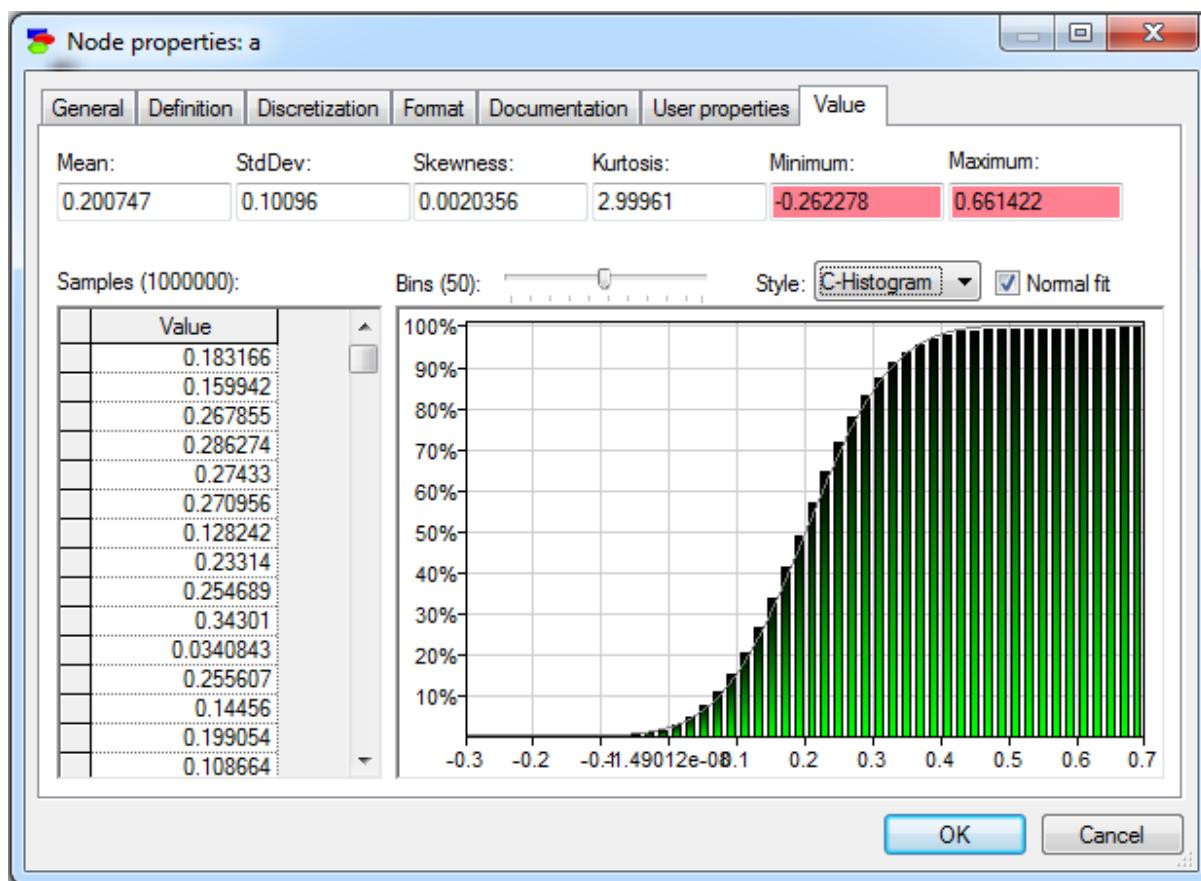
Style pop-up menu allows for choosing a different plot:



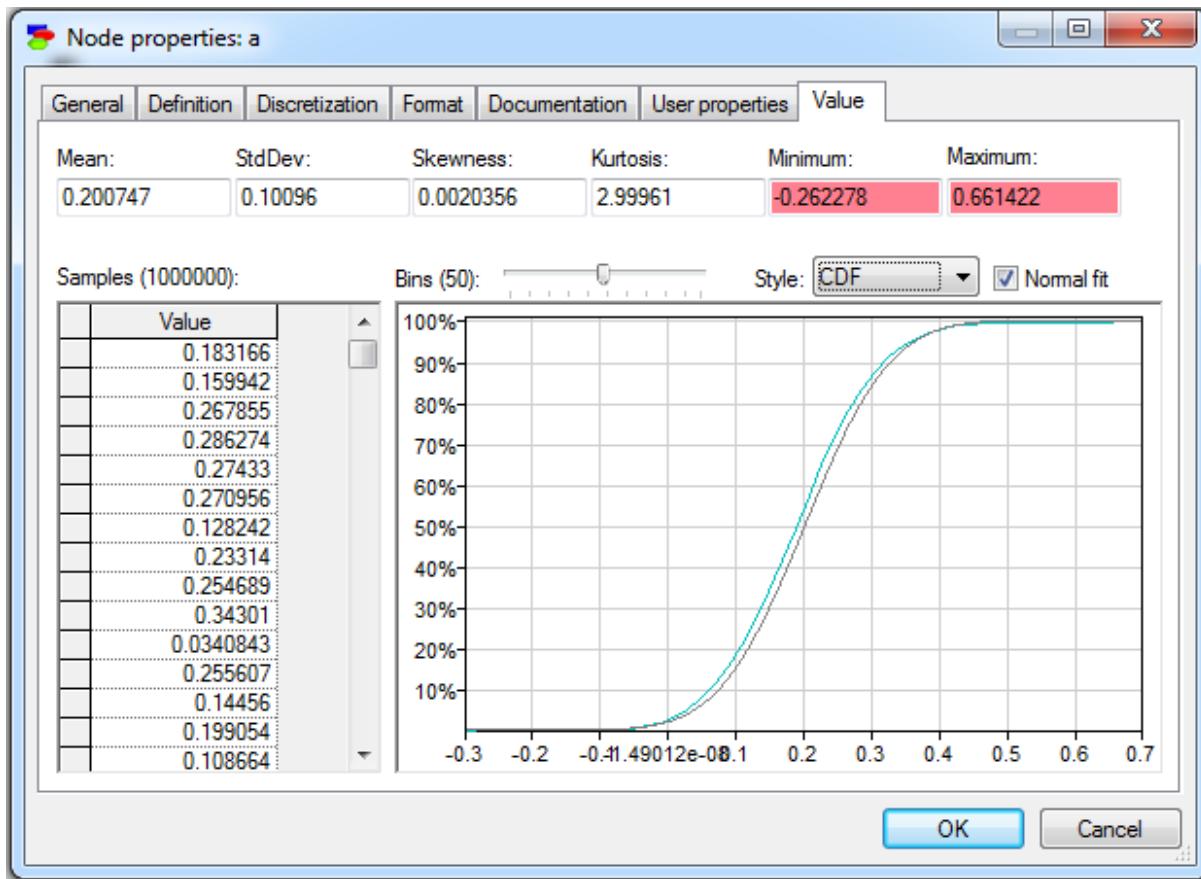
PDF is an abstraction of the histogram plot:



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



CDF is an abstraction of the cumulative histogram plot:

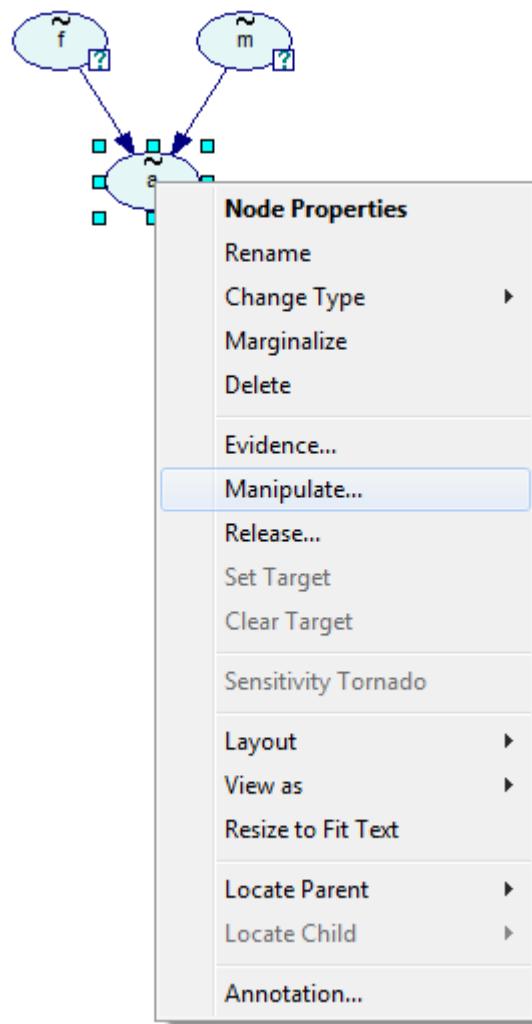


It is useful to note that the marginal probability distribution of a does not follow the Normal distribution - while its probability density function (PDF) follows a unimodal bell-shaped curve, there is a visible difference between the curve and the Normal distribution plotted alongside. This is not surprising given the definition of variables a , f , and m .

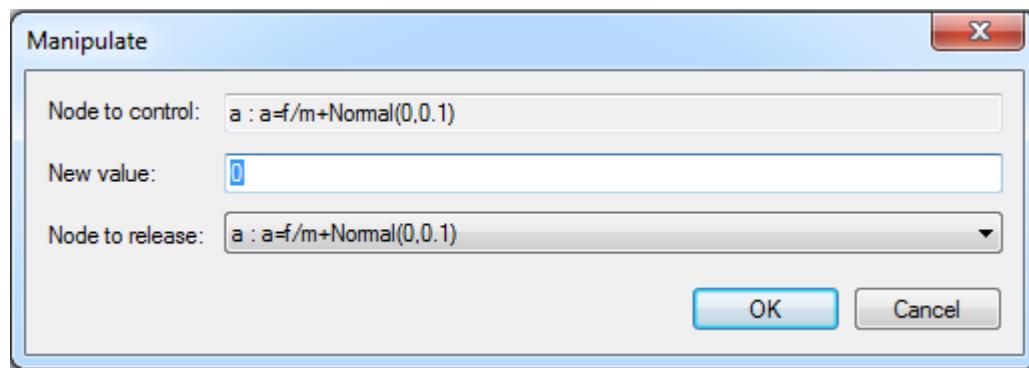
With the *AutoDiscretize* algorithm used for inference in continuous models, the *Value* tab of *Equation* nodes is identical to those of *Chance* nodes in Bayesian networks.

6.7.5 Structural changes

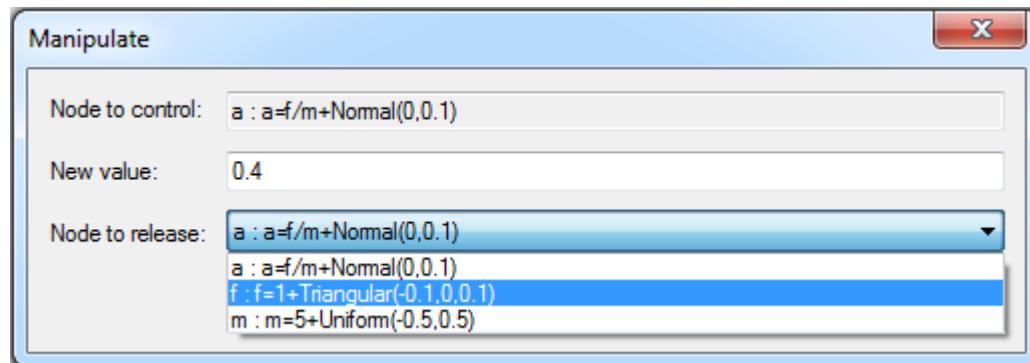
Equation-based models support true changes in structure, which may lead to a structural change to the model. Consider the example model of Newton's 2nd law, used throughout this section. Imagine that we want to determine the force necessary to act on our mass in order to achieve acceleration $a=0.4$. To achieve this, we select *Manipulate...* from the context menu of variable a .



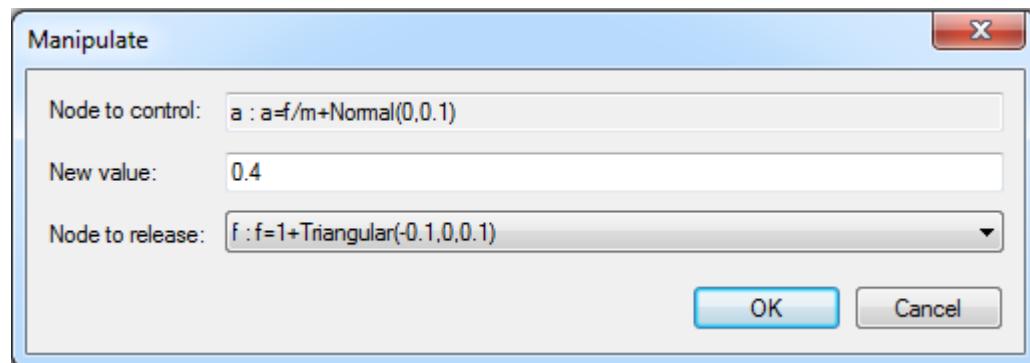
This invokes the following dialog:



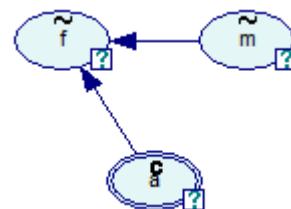
We enter the value *New value* 5 and release node f



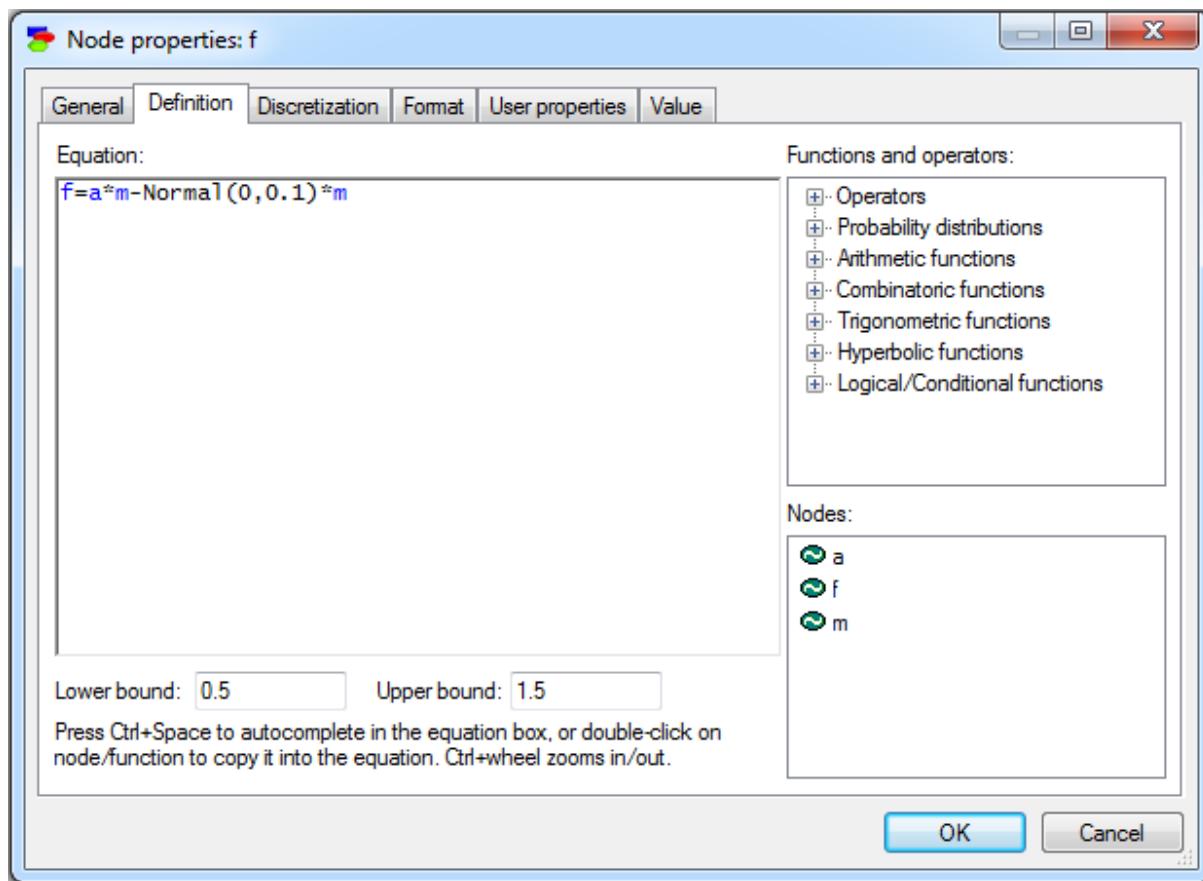
yielding



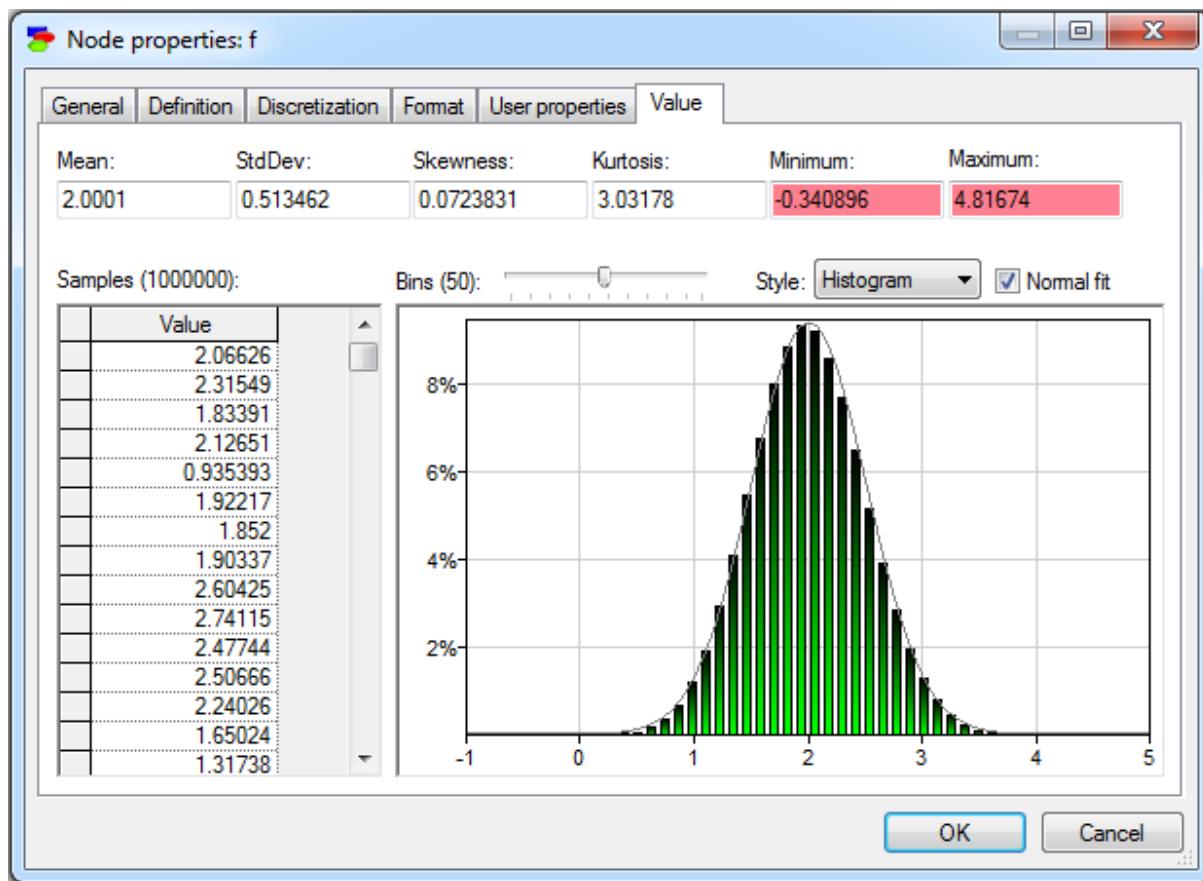
Pressing OK performs a structural manipulation on the network, yielding



The structure of the network has changed, making node f a child of nodes m and a . The definition of node f has changed to the following



Updating the network yields the following result at node *f*



Please note that we have achieved this result by a structural manipulation that changed the structure of the network, modified the definitions of variables a and f , and derived the numerical results from the new network.

This page is intentionally left blank.
Remove this text from the manual
template if you want it completely blank.

Resources

7 Resources

7.1 Books

A good source of elementary knowledge of [Bayesian networks](#)⁴⁵ is the path-breaking book by Pearl (1988). A good, thorough synthesis of the theoretical aspects of Bayesian networks and [probabilistic decision support systems](#)⁵⁰, useful for readers interested in building a reasoning system from the ground up, is the book by Neapolitan (1990), regrettably 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](#)⁴³ 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](#)⁴² 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](#)⁴⁵, 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 electronically from Decision System Laboratory's web site located at

<https://dslt.org/uai/>

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 2016 conference, listing web sites for all previous PGM conference and their on-line proceedings, is at the following location:

<http://www2.idsia.ch/cms/pgm/>

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 are model repositories. They are a great inspiration to model builders and source of models for the purpose of testing algorithms. Here are two most popular model repositories:

[GeNIE Network Repository](#)

[Norsys, Inc.'s library of networks](#)

You should be able to read all these models using GeNIE.

7.5 Social Media

Please visit [BayeFusion'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.

Finally, [BayesFusion's LinkedIn account](#) contains basic information about BayesFusion.

7.6 References

Abramson, B., J. Brown, W. Edwards, A. Murphy & R. Winkler (1996). *Hailfinder: A Bayesian system for forecasting severe weather*. International Journal of Forecasting 12(1):57-72.

Bayes, Rev. Thomas (1702-1761). *An essay toward solving a problem in the doctrine of chances* (reprint). Biometrika, 45(3-4):293-315.

Beinlich, I.A., H.J. Suermondt, R.M. Chavez & G.F. Cooper (1989). *The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks*, Proceedings of the Second European Conference on Artificial Intelligence in Medical Care, pages 247-256, Springer-Verlag, Berlin.

Buntine, Wray L. (1991). *Theory Refinement on Bayesian Networks*. in Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence (UAI-1991), pages 52-60.

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.

Cooper, Gregory F. (1984). *NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge*. PhD thesis, Medical Information Sciences, Stanford University, Stanford, CA.

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*. Artificial 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.

Dash, Denver H. & Marek J. Druzdzel (1999). *A hybrid anytime algorithm for the construction of causal models from sparse data*. In Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99), pages 142-149, Morgan Kaufmann Publishers, Inc., San Francisco, CA.

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.

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, 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 Statistical 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 (Methodological), 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*.)

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. Kłopotek, 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.

Spirites, 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.

building influence diagrams 281

- . -

.gdat 339

- A -

about GeNle 30
accuracy 411
acknowledgments 39
acknowledgment 38
acyclic directed graph 45, 47
AIS 215
AIS Sampling 215
ALU 138
ANB 395
Annealed MAP 220
annotation 120
approximate belief updating 48
arc 115, 270
arc reversal 250
arc thickness 270
AUC 411
Augmented Naive Bayes 395
autodiscretization 230

- B -

backward sampling 48, 214
Bayesian approach 45
Bayesian belief network 45
Bayesian inference 48
Bayesian network 45, 216, 220, 256, 261, 263, 270, 273, 276
Bayesian networks 250
Bayesian Networks Interchange Format 200
Bayesian networks tutorial 12
Bayesian Search 381
Bayesian updating 48
belief network 45
belief updating 48
BNIF 200
books 484
BS 381
bug reporting 39
building a Bayesian network 12
building blocks of GeNle 56

- C -

calibration curve 411
Canonical Nodes 87
case management 330
Case Manager 78
cases 78
causal probabilistic network 45
CDF 470
chance node 85, 138
Chang 48
changes in structure 50, 477
clustering algorithm 211
conferences 485
confusion matrix 411
console 82
continuous models 229, 230, 235, 236
continuous variables 42
controlling values 273
Controlled status icon 116
Cooper 49
copyright notice 38
costs of observation 334
create a node 12
creating Bayesian network 12

- D -

data 339
Data Grid View 353
data menu 351
database 339
DBN 432, 433, 440, 450
decision analysis 42
decision node 85, 138, 281
decision support system 50
decision-analytic decision support 50
define properties 12
definition tab 138
del Favero 48
deterministic equation node 85
deterministic node 85, 138
diagnosis 303, 308, 317, 324, 330, 334
Diagnosis Menu 304
Diagnosis Toolbar 307

diagnostic extensions 308
differential diagnosis 324
disclaimer 39
discrete variables 42
discretization 230, 353
DSL file format 197
DSS 50
dynamic Bayesian network 432, 433, 440, 450

- E -

edge 115
effects of changes 50
effects of changes in structure 50
EGS 389
EM algorithm 400, 450
entering evidence 256
entropy/cost ratio 324
EPIS 215
EPIS Sampling 215
equation 138
equation node 85
equation-based models 229, 230, 235, 236, 456, 462, 470, 477
Ergo 198
Ergo file format 198
error messages 82
Essential Graph Search 389
evidence 256
example networks 32

- F -

Facebook 486
File Menu 193
find best policy 228
flat file 339
forbid arc 373
force arc 373
format 241
format tab 138
Format Toolbar 177
frequentist interpretation 43
full-screen mode 176
Fung 48

- G -

general tab 138
generating data file 407
GeNle data format 339
GeNle options 241
GeNle version 32
GeNle workspace 57
graph layout 173, 182
Graph View 60
Greedy Thick Thinning 391
GTT 391

- H -

Hardware and software requirements 32
Help Menu 83
Henrion 48
histogram 353, 470
Hugin 200
Hugin file format 200
Hybrid EPIS 236
Hybrid LBP 235
Hybrid Likelihood Weighting 235
Hybrid Logic Sampling 235
Hybrid LW 235
hybrid models 235, 236

- I -

impact of observing values 48
Implied status icon 116
inference 281, 440, 462
inference algorithms 200
inference in influence diagrams 49
influence diagram 47, 227, 228
influence diagrams 290, 291, 296
introduction 250
Invalid status icon 116

- J -

joint probability distribution 263
joint-tree algorithm 211

- K -

keyboard shortcuts 246
KI file format 200
Knowledge Editor 373
knowledge engineering 433, 456
Knowledge Industries 200

- L -

layout 173
Layout Menu 173, 182
lazy evaluation 203
LBP 235
learning 339
learning parameters 400, 450
likelihood sampling 48, 214
LinkedIn 486
literature list 487
loopy belief propagation 235

- M -

manipulation 50, 273
MAP 220
marginal probability distribution 263
marginalization 250
MAU 100, 138
Menu Bar 59
merging states 353
missing value 353
model repositories 486
movies 486
multi-attribute utility 100

- N -

Naive Bayes 398
navigation 186
NB 398
Netica 199
Netica file format 199
Network Menu 209
Network Property Sheets 123
Node Menu 207
Node Property Sheets 138

node status icons 116
node type 85
Noisy-Adder 87
Noisy-AND 87
Noisy-MAX 87
Noisy-MIN 87
Noisy-OR 87

- O -

obfuscation 236
objectivist interpretation 43
observe 12
Observed status icon 116
observing 48
ODBC 339
Olmsted 49
options 241
Output Window 82

- P -

parent ordering 173, 182
Pattern Editor 376
PC 383
PDF 470
Peot 48
pie chart 353
policy evaluation 227
polytree algorithm 213
precision 241
probabilistic logic sampling 48, 214
probability 43
probability of evidence 216
program options 241
propensity view 43

- R -

read me first 10
references 487
relevance diagram 47
relevance reasoning 204
relevance-based decomposition 213
research papers 484
resources 484, 485, 486
results 470

retracting evidence 256

ROC curve 411

- S -

save model 12

saving and loading models 188

scatterplot 353

selection 184

self-importance sampling 214

sensitivity 411

sensitivity analysis 49, 276, 291

Shachter 48, 49

size 176

SMILE 31

specificity 411

Spreadsheet View 317

spring embedder 173, 182

Standard Toolbar 136

start here 10

statistics 353

Status Bar 76

stochastic sampling 48

strength of influence 270

structural learning 379

structural transformations 250

subjectivist view 43

submodel 105

submodel node 85

Submodel Property Sheets 133

- T -

TAN 393

target 204

Target status icon 116

temporal tier 373

Testing Window 324

text box 119

time series 353

Tools Menu 136

transparent mode 138

Tree Augmented Naive Bayes 393

Tree View 73

Twitter 486

- U -

update immediately 203

update the model 12

user properties 138

using GeNle 250

utility 44, 138

- V -

validation 411

value node 85, 138, 281

value of information 49, 296

value tab 138

view results 12

viewing results 263, 290

virtual evidence 261

visual appearance 173

VOI 49, 296

- W -

warnings 82

- X -

XDSL file format 197

- Y -

YouTube 486

- Z -

zooming 176