
CaMML GUI Quick-Start Guide

BI-CaMML Version 1.4

10-11-2014

Contents

1	Introduction	1
1.1	Installing and Running CaMML	1
2	GUI Settings and Use	1
2.1	Setup	1
2.1.1	Parameterization Type	2
2.1.2	Random Number Generator / Seeds	2
2.1.3	Search Factor	3
2.1.4	Max. Number of SECs	3
2.1.5	Min. Total Posterior	4
2.2	Prior Arc Probability	4
2.2.1	Learn Dynamic Bayesian Network	4
2.3	Expert Priors	4
2.3.1	Arc Priors	5
2.3.2	Tier Priors	6
2.3.3	Edit Distance from Specified Network Structure	6
2.3.4	Kendall Tau Distance from Specified Ordering	6
2.3.5	Example Prior String	7
2.4	Run	7
2.5	Results	8
2.5.1	View Selected Network	8
2.5.2	Export Selected Network and Export All	9
2.5.3	Arc Probabilities	9
2.6	Status Bar	10
3	Command Line	11
4	Data and Data Formats	11
4.1	Importing Data	12
4.1.1	Supported Formats	12
4.2	Exporting Bayesian Networks	12
4.2.1	Bayesian Network Format Conversion	12
5	Improving Performance	13
5.1	Improving Model Quality	13
5.2	Reducing Computation Time	14
6	Technical Details	14
7	Common Issues and Solutions	14
7.1	Console output: Node costing warnings	14
7.2	Few (or very simple) models produced during search	15
7.3	Console output: java.lang.OutOfMemoryError: Java heap space	15
8	Usage Examples	16
8.1	“Asia Problem”	16
8.2	Letter Recognition	17

1 Introduction

CaMML is a program for learning causal Bayesian networks from data sets. CaMML, or *Causal MML* uses a ‘search-and-score’ approach to learning Bayesian networks (as opposed to using conditional independence testing), and uses the Minimum Message Length principle to trade off model complexity and fit-to-data.

CaMML also allows a user to specify structural prior information along side the data, in a variety of flexible formats. Such priors may be hard (i.e. deterministic) constraints or soft (i.e. probabilistic) constraints. These structural priors (“expert priors”) are discussed in Section subsection2.3.

1.1 Installing and Running CaMML

The version of CaMML described in this document is BI-CaMML 1.4, which is a packaged version provided by Bayesian Intelligence. BI-CaMML can be downloaded from the Bayesian Intelligence site at: <http://bayesian-intelligence.com/software/>.

Once downloaded, extract the zip file to somewhere on disk. On Windows, we recommend C:\BI-CaMML and /opt/bi-camml on Linux (and you may wish to add it to your system’s PATH). To start the GUI:

- On Windows: Run `camml_gui.bat`
- On Linux/Mac: Run `camml_gui.sh`

To start the command line version:

- On Windows: Run `camml.bat`
- On Linux/Mac: Run `camml.sh`

Command line usage of CaMML is discussed further in Section section3.

2 GUI Settings and Use

This section discusses how to use CaMML and the GUI. Information in this section is organized according to the tab to which it corresponds to in the GUI. For examples of how to use this GUI with sample models, see Section section8.

2.1 Setup

The **Setup** tab (Figure 1) allows you to load discrete data sets (see Section section4 for details). Once data has been loaded, it can be viewed (but not edited) using the built-in data viewer (Figure 2). You can also configure the parameters that will affect how CaMML learns Bayesian networks. Note that the default values (perhaps excluding parameterization type) are often suitable for most learning tasks. Each of the settings on the **Setup** tab is briefly discussed below.

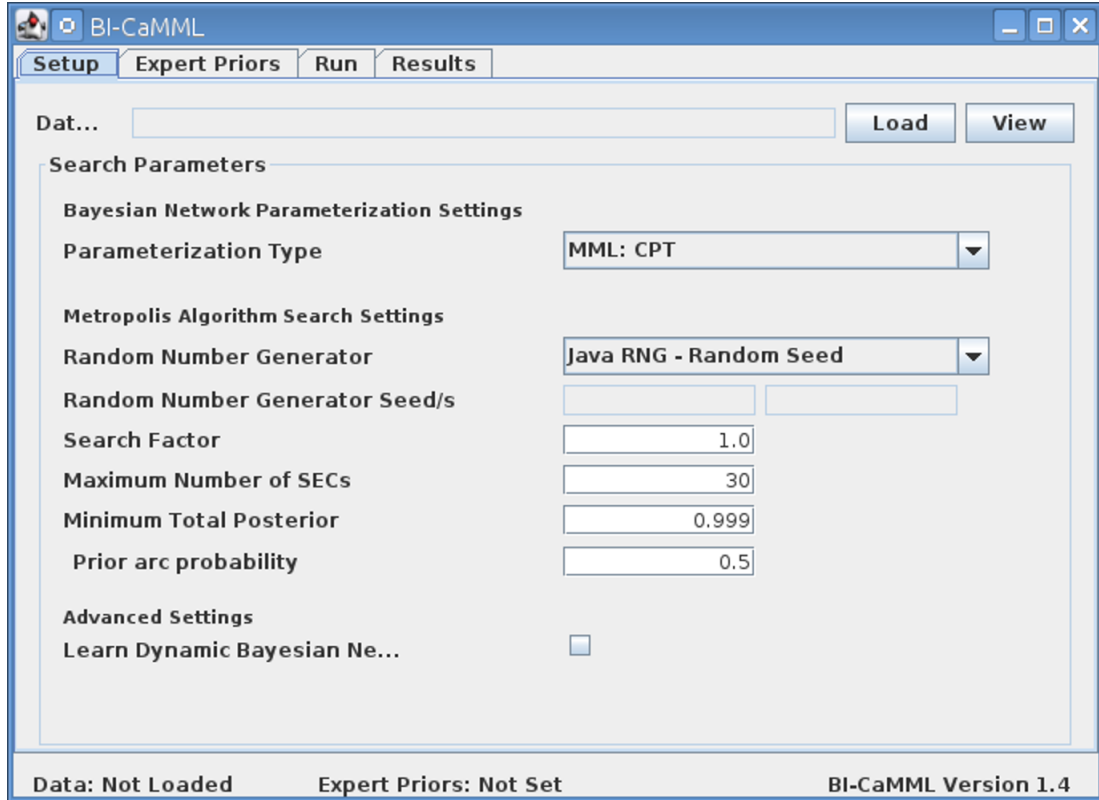


Figure 1: The Setup Tab, showing the default parameters used for learning Bayesian networks.

2.1.1 Parameterization Type

This setting controls the type of conditional probability distributions (CPDs) that CaMML will use when learning Bayesian Networks (BNs). Each node X in a Bayesian network has a probability distribution $P(X|Pa(X))$, where $Pa(X)$ is the set of parents for variable X ; the parameterization type setting controls the type(s) (or family) of conditional probability distributions that CaMML will learn.

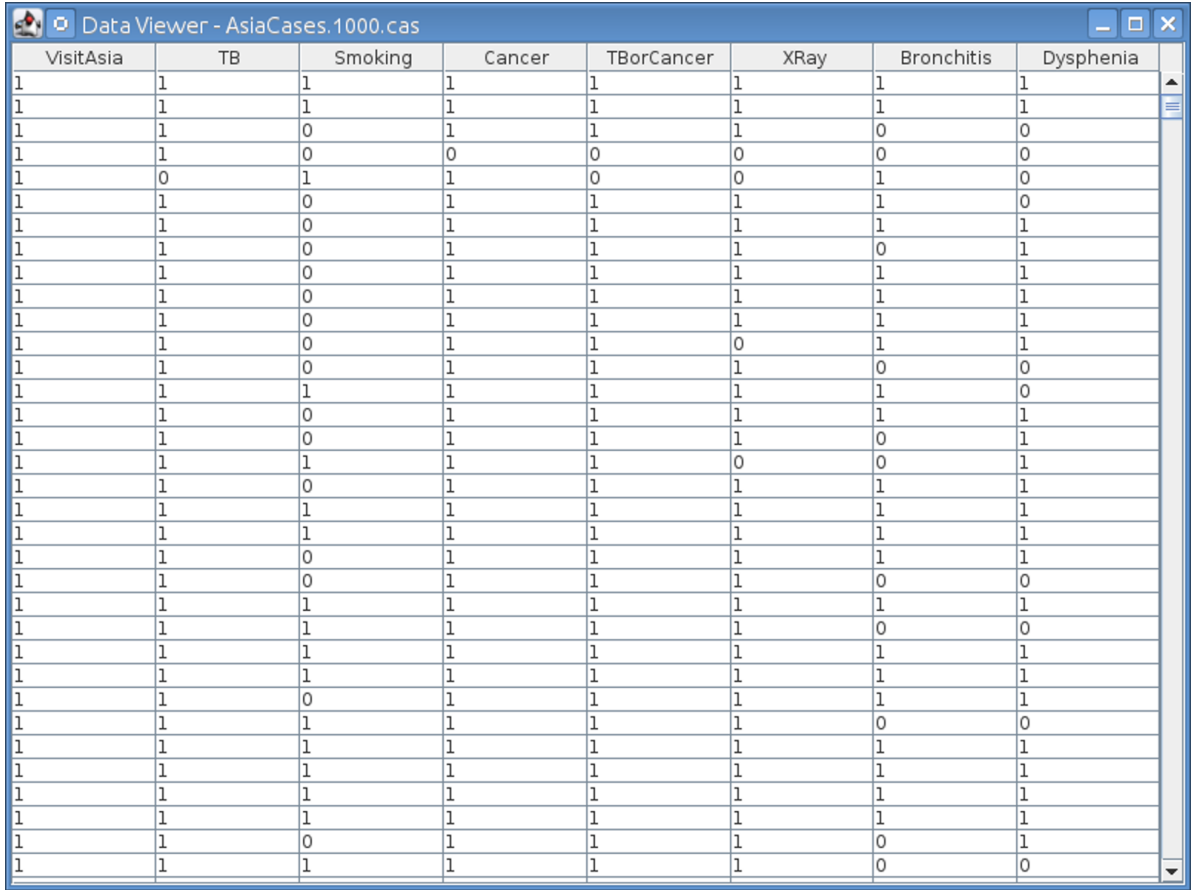
Options include Conditional Probability Tables (CPTs), Decision Trees (DTree) and Logit (Logit) models. CPTs are the most flexible, but have a larger number of parameters (than DTree and Logit models) that must be learned from the data.

When multiple options are selected (i.e. CPT + DTree), CaMML will select the parameterization type that results in the best simplicity/fit trade-off. Searches with multiple parameterization types will take longer than searches with a single parameterization type, but may result in more accurate networks being learned.

2.1.2 Random Number Generator / Seeds

Setting of the RNG seeds is reserved for specialized circumstances such as regression testing. The vast majority of users will leave this as the default value.

Setting RNG seeds is only enabled when using set seed(s) for one of the random number generators. Again, this option is not applicable for most users.



VisitAsia	TB	Smoking	Cancer	TBorCancer	XRay	Bronchitis	Dysphenia
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	0
1	1	0	0	0	0	0	0
1	0	1	1	0	0	1	0
1	1	0	1	1	1	1	0
1	1	0	1	1	1	1	1
1	1	0	1	1	1	0	1
1	1	0	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	0	1	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	1	1	1	1	0
1	1	0	1	1	1	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	0	1	1	1	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	0	0

Figure 2: The Data Viewer window, displaying the data contained within a loaded file.

2.1.3 Search Factor

CaMML uses a Markov Chain Monte Carlo (MCMC) sampling algorithm to search through the exponentially-large space of possible BN models. The **Search Factor** option is used to modify the number of model changes attempted by the learning algorithm. Increasing the search factor should increase the reliability of results (i.e., improving the accuracy of the posteriors for the models), by allowing CaMML more time to search through and sample the model space, at the expense of increased computation time. Decreasing the search factor reduces the reliability of the results and computation time.

Search Factor has a default value of 1.0. A search factor value of 2.0 means twice as many model changes are attempted; 0.5 means half as many model changes are attempted.

2.1.4 Max. Number of SECs

To obtain more reliable results, CaMML combines ‘equivalent’ models in a number of ways. TOMs (Totally Ordered Models) are combined to SECs (Structural Equivalence Classes), which are in turn combined to MMLECs (Minimum Message Length Equivalence Classes). SECs are combined to MMLECs where the available data is insufficient to differentiate the SECs. The **Max. Number of SECs** setting specifies how many SECs we should keep, and attempt to combine to MMLECs. A larger number may mean more reliable results, but will take longer to complete. The default setting of 30 is usually suitable, as the top 30 SECs usually contain the

majority of the posterior probability.

2.1.5 Min. Total Posterior

This setting specifies the minimum posterior value we want to keep. A higher value means keeping more SECs, at the expense of longer computation time at the end of the search (when SECs are combined into MMLECs). Note that the maximum number of SECs (discussed previously) setting takes precedence over the **Min Total Posterior** value.

The default value (0.999) is typically suitable for most tasks.

2.2 Prior Arc Probability

By default, CaMML assigns a global prior probability of 0.5 to an arc existing between any two nodes. If CaMML seems to be generating too many or too few arcs, this parameter can be used to bias CaMML as needed. Of course, the higher the value of this prior, the greater the arc density of the network that CaMML creates. This prior is overridden by data, to a degree that depends on the strength of the data. Therefore in some cases, if the data is sufficiently strong, the prior will have no effect. Note that arc priors can also be set on an arc by arc basis — see Section subsection2.3.

2.2.1 Learn Dynamic Bayesian Network

CaMML has the ability to learn Bayesian Networks and Dynamic Bayesian Networks. When the check-box is checked, CaMML will assume the data file represents a time series, and will attempt to learn a Dynamic Bayesian Network. At this stage, DBN learning has a few restrictions, such as the inability to specify custom expert priors (including the global arc prior).

2.3 Expert Priors

CaMML allows a user to (optionally) specify structural prior information, using a variety of flexible formats. Structure priors are used to represent various types of prior domain knowledge that an expert might have — information that is not contained in (or representable in terms of) the input data file. For example, a domain expert may have information about temporal relationships between observations, or information about causal dependence (or independence) between variables in the data. By incorporating structure priors, it is possible to improve the quality of the learned network.

Expert priors in CaMML are able to be specified probabilistically — that is, CaMML’s expert priors are ‘soft constraints’ (deterministic ‘hard’ constraints are also supported by specifying 0 or 1 for the prior probability). Essentially, it is possible to enter prior information with an specified degree of confidence; these soft expert prior constraints may be ‘overridden’ by the data if the data strongly suggests that the prior information is incorrect.

When using expert priors, refer to a variable by its name in the input data file. If the data file contains no names, use a number (in the order in which it appears in the data file) to refer to that variable. Note that the use of structure priors adds a small computational overhead, which can increase the amount of time taken for an instance of search to complete.

Currently, expert priors are not available when learning Dynamic Bayesian Networks (DBNs) and the **Expert Priors** tab is disabled if **Learn Dynamic Bayesian Network** was selected on the **Setup** tab.

The **Expert Priors** tab (Figure 3) allows users to describe various types of expert priors contained within a BN — some examples are provided within the GUI when expert priors are

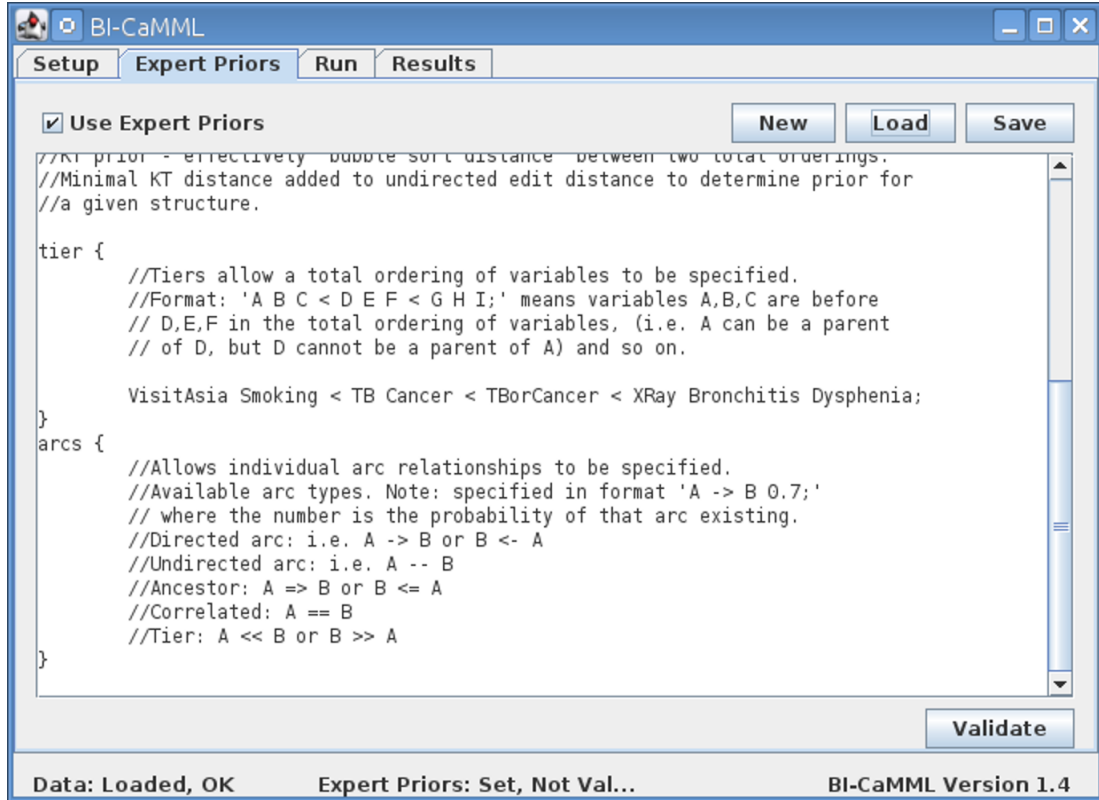


Figure 3: The Expert Priors Tab, with expert priors entered.

first enabled. Priors may be loaded, edited, saved or cleared. Clicking on the **Use Expert Priors** check-box enables the priors text window and fills it with blank default priors. The default text here contains comments that describe the format that the expert priors should take. The **New** button will replace any text entered into the priors text window with the default text comments. The **Save** button will save the text in the priors text window as a plain text file. The **Load** button will load expert priors from a file. Once the expert priors are finalised, clicking the **Validate** button will validate that the priors are formatted correctly. CaMML cannot be run on a model until the priors are valid.

The following sections (subsubsection2.3.1, subsubsection2.3.2, subsubsection2.3.3 and subsubsection2.3.4) describe in more detail the expert priors that are supported by CaMML and how they are formatted within the text window.

2.3.1 Arc Priors

Arc priors represent information that is known about the relationship between two variables in the data set. Prior information between any two pairs of variables can be specified in one or more of the following ways: Directed arc, Undirected arc, Ancestor relation, Correlation, Tier.

The general format for arc priors is *variable* **[operator]** *variable* **[prob]**; where *variables* are the names of a variables/columns in the input data set, **[operator]** specifies the type of relationship (i.e. a directed arc is **->**), and **[prob]** is a probability such as 0.70 or 1.00.

Directed arc: Allows us to specify the probability that a direct arc exists between two variables. For example: **Smoking -> Cancer 0.99;** or **Rainfall <- DayOfWeek 0.0;**.

Undirected arc: Probability of an arc existing in either direction (i.e. $A \rightarrow B$ or $B \rightarrow A$). For example, `Happiness -- Spending 0.8`; means we think either Happiness directly causes Spending or Spending directly causes Happiness with probability 0.8.

Ancestor: Used to specify that one variable is an ancestor of another (even if not directly connected). For example, `A => B 0.9`; means A is a parent of B, or A is a parent of one of B's parents, or so on.

Correlated: Two variables are correlated if A is an ancestor of B, B is an ancestor of A, or A and B have a common ancestral cause. Example: `A == B 0.95`;

Precedence: Used to represent the temporal relationship between two variables, even if they are not directly/indirectly connected. For example, `SchoolGrades >> RetirementWealth 1.0`.

Arcs priors are enclosed in the `arcs{}` section, as follows:

```
arcs {0 -> 1 0.9; 1 -- 2 0.4, 3 > 4 1.0;}
```

Though each arc can be placed on a separate line for clarity/ease of reading.

2.3.2 Tier Priors

Tier priors are (generally deterministic) constraints on the total ordering of some/all variables. Note that we could use arc priors (specifically, the precedence operator) instead of tiers for this purpose, but specifying all of the appropriate precedence relationships would be rather complex, compared to using a tier prior.

Example:

```
tier {1 2 3 < 4 5 6;}
```

Note that we can have multiple tier specifications in a single line:

```
tier {1 2 3 < 4 5 6 < 7 8 9;}
```

In this example, variables 1, 2 and 3 are before 4, 5 and 6 in the total ordering, and 4–6 are before 7–9. In the network, variables 1, 2 and 3 can be parents of variables 4, 5 and 6, but variables 4–6 cannot be parents of variables 1–3.

Tier priors are typically deterministic, but the prior probability can be set as follows:

```
set { tierPrior = 0.9; }
```

The default value for tierPrior is 1.0, and is used unless the tierPrior is explicitly set. This value of 1.0 means that all tier priors (if any) must be satisfied in all sampled networks.

2.3.3 Edit Distance from Specified Network Structure

We can specify prior information in terms of the edit distance from a specified network (or part network). For example, we could specify a diamond network as follows:

```
ed{a -> b; a -> c; b -> d; c -> d;}
```

Networks closer (in terms of edit distance) to this specified network will be scored higher than those further from the specified network. The strength in the prior belief can be set in the `set{}` section as follows:

```
set { edPrior = 0.99; }.
```

The default value for edPrior is approximately 0.73.

2.3.4 Kendall Tau Distance from Specified Ordering

Kendall Tau priors are similar to Edit Distance priors (in that an expert provides a network), except that KT priors also take into account the ordering of variables. Networks where the total

ordering of variables is close to the order of the specified network are scored higher than those that are further from the total ordering. KT priors take a specified network, and work out how many ‘swaps’ to the variable order we need to make on a given network consistent with the specified network. Each network is also penalized by the number of (undirected) arc differences there are between the specified and current network. Note that Kendall Tau and Edit Distance type priors cannot be used simultaneously.

KT priors are specified in the same manner as Edit Distance priors, except that `kt{...}` is used instead of `ed{...}`. Again, the strength of the prior belief can be set as per the edit distance prior:

```
set { ktPrior = 0.99; }.
```

The default value for `ktPrior` is approximately 0.73.

2.3.5 Example Prior String

Suppose our data set contains the variables “A”, “B”, ..., “G” and we have the following prior information about the domain:

1. Variables C and D are direct causes of F (with 0.75 probability for C->F and 0.95 probability for D->F)
2. Variables A, B and D occur before variables E and F with 0.9 probability (but we don’t know how they are related)
3. Variables A and C are statistically/causally independent (we want this to be enforced deterministically)

In this case, we might create a set of structure priors as follows:

```
set {
tierPrior = 0.9; //Sets probability for (2)
}
tier {
A B D < E F; //From (2). A, B & D occur before E & F in the
    // temporal ordering of variables
}
arc {
C -> F 0.75; //From (1), a directed arc from C to F
D -> F 0.95; //From (1), a directed arc from D to F
A == B 0.0; //From (3), A & B are causally independent (0.0
    // probability - deterministically uncorrelated)
}
```

Note that the `//` characters represent a comment; everything after these characters and is ignored, until the next line. Whitespace characters (i.e. tab, space etc) are also ignored. Also note that there can be a number of equivalent ways of specifying the same prior information.

2.4 Run

The **Run** tab (Figure 4) is relatively self explanatory. There are two buttons:

- **Run:** Starts the search
- **Clear Output:** Clears the console output window.

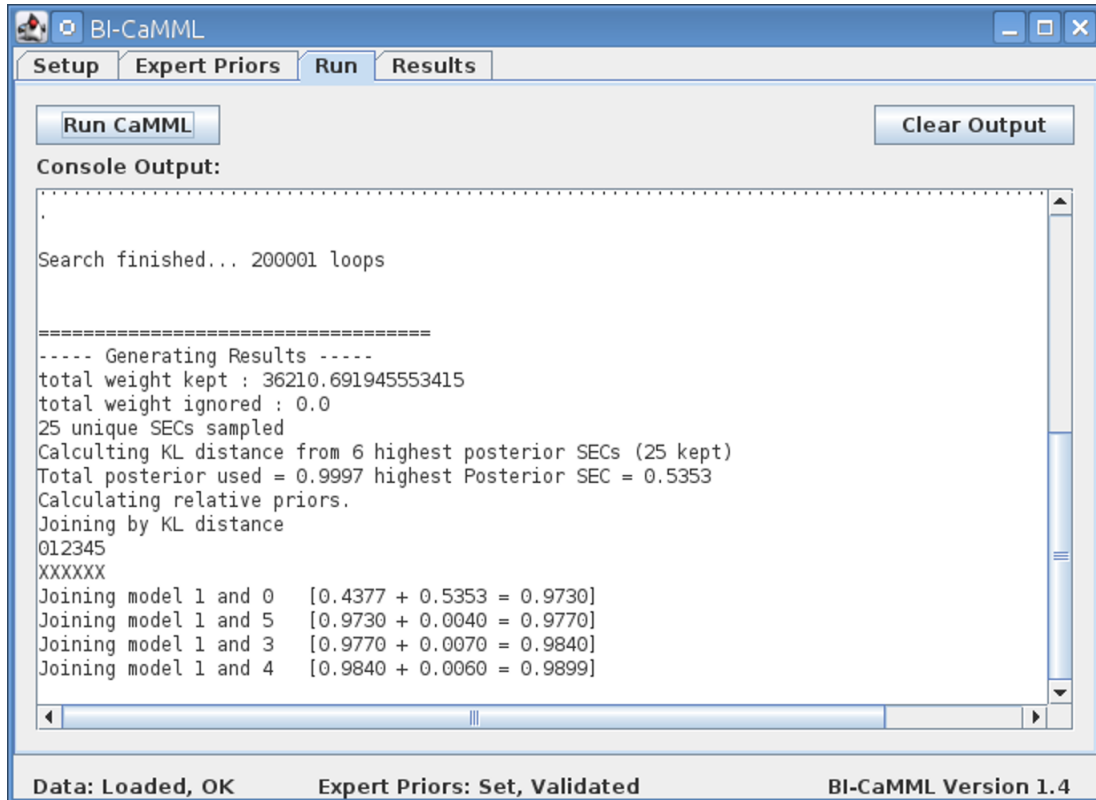


Figure 4: The Run Tab. Here the console output window shows CaMML’s progress, indicating that a search is complete.

Upon clicking the run button, users will be warned if the settings entered on the **Setup** tab (or expert priors on the **Expert Prior** tab) are invalid.

The console output window shows the progress of the search and any warning/error messages generated during the search process.

2.5 Results

Once the search is complete, results are displayed in the **Results** tab (Figure 5), sorted by posterior.

The results are MMLECs (Minimum Message Length Equivalence Classes) - each row in the table represents a single MMLEC. The top row represents the MMLEC with the highest posterior probability — the ‘best’ model essentially. If you are only using a single model, this is the one to export and use — though attention should certainly be paid to the posterior field in the table, as there may be no uniquely best model. That is, there can be multiple models with similar posterior probabilities.

2.5.1 View Selected Network

CaMML includes a very basic network viewer. It is possible to view the representative network of a MMLEC by selecting the appropriate row and then clicking **View Selected Network**. This network viewer includes basic drag and drop functionality (to move network nodes), although the layout set here will have no bearing on the layout of the exported network. Figure 6 shows

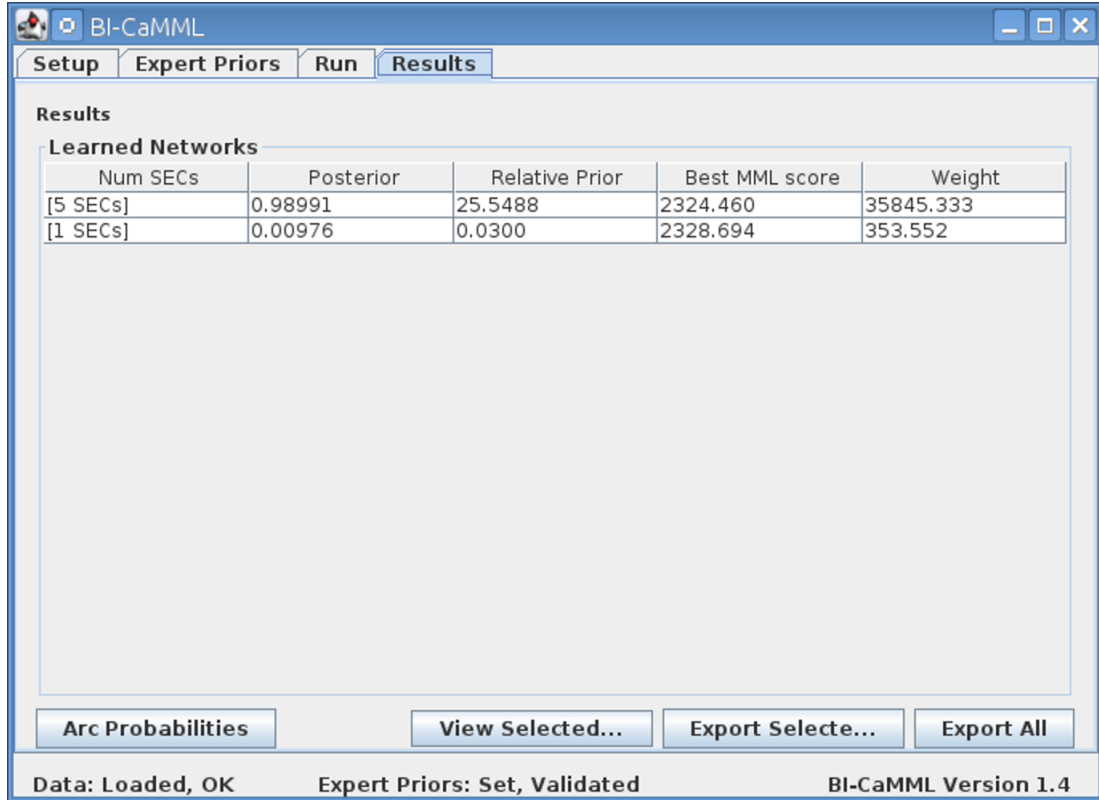


Figure 5: The Results Tab.

an example of the network viewer.

Basic inference can also be run on the network. A variable can be set to a specific state by clicking on the appropriate row.

2.5.2 Export Selected Network and Export All

A ‘representative network’ of an MMLEC can be exported to Netica (.dne) format only (see Section subsection4.2.1 for format conversion). Select the MMLEC (row) to export, and then click the **Export Selected Network** button. As noted, the single ‘best’ network (i.e. the network with the highest posterior probability) is contained in the top row.

It is also possible to export all networks. Clicking the **Export All** will open a file dialogue box. The name entered in the file dialogue box will be appended with a number for each consecutive network. Thus, if ‘MyNetwork.dne’ is entered as the name, the highest posterior network will be called ‘MyNetwork0.dne’, the second highest posterior network will be ‘MyNetwork1.dne’ and so on. Note that none of the information from the **Results** table will be saved — this should be copied (for example) to a spreadsheet. To do this, select the rows in the table and use the copy keyboard shortcut (i.e. Control+C or Command+C) to copy the data.

2.5.3 Arc Probabilities

CaMML also provides information regarding which arcs are present or absent. These ‘arc probabilities’ represent the probability that a directed arc between each of the variables in the dataset is present. Note that there is a single set of arc probabilities (i.e. they are calculated

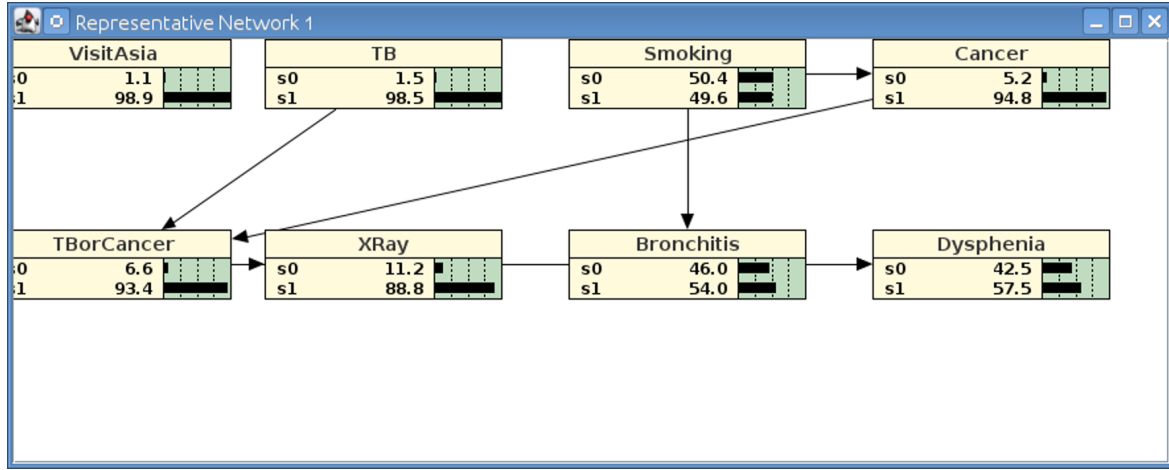


Figure 6: The Network Viewer, showing the Bayesian Network CaMML learned from the analysed data.

based on the entire search), and it does not matter which (if any) of the rows in the results table are selected.

To view the arc probabilities, click the **Arc Probabilities** button to open the viewer. The probability of an arc from A to B existing is presented in the cell at row A, column B. Figure 7 shows the arc probabilities for an example network.

	VisitAsia	TB	Smoking	Cancer	TBorCa...	XRay	Bronch...	Dysphe...
VisitAsia	0	0.15864	5.4985...	1.0885...	1.3164...	2.3074...	0.55109	2.1168...
TB	0	0	0	6.2223...	1	3.9403...	4.2514...	2.7331...
Smoking	2.1651...	7.5035...	0	0.99992	1.4895...	1.0092...	1	1.4663...
Cancer	0	8.8416...	0	0	1	4.2968...	5.6854...	2.7282...
TBorCancer	0	0	0	0	0	0.98556	3.0454...	0.99057
XRay	0	0	0	0	0	0	5.0414...	2.4928...
Bronchitis	0	0	0	0	0	9.5321...	0	0.99994
Dysphenia	0	0	0	0	0	2.4968...	5.5425...	0

Figure 7: The Arc Probabilities Window, showing the arc probabilities for the Bayesian Network CaMML learned from the analysed data.

These arc probabilities can be selected using the mouse, and copied using keyboard shortcuts (i.e. Control+C or Command+C), and then pasted into a spreadsheet as required.

2.6 Status Bar

The status bar is found at the bottom of the window and is visible from all tabs. It displays whether a data file has been loaded, the state of the expert priors (if any) and the current software version. The data file and expert priors messages will be updated as their status changes during the learning of Bayesian networks. When a data file has been loaded, if the file is valid then the Data status message changes to *Loaded, OK*. If the data file is invalid and

cannot be loaded, the status message is *Not Loaded*.

When the expert priors are set on the **Expert Priors** tab, then the status message is *Set, Not Validated*. Once the priors are validated, then the status message changes to *Set, Validated*.

3 Command Line

The command line version of CaMML is controlled by a number of command line arguments, which are discussed below. The basic form for running CaMML from the command line is `camml [options] <data-file> [<output-file>]`, where `[options]` is a list of the desired options (if any), `<data-file>` is the data file CaMML should open and `[<output-file>]` is the optional output file for the results.

- `-a, --all-bns` Output all representative networks retained. Without this, only one representative network is saved. Default: false
- `--arc-prob` Set the prior probability for arcs in the networks. Default: 0.5
- `--dbn` Learn a DBN network from the data (assumes data is in time series format with Row 1 = t , Row 2 = $t+1$, etc.) Default: false
- `--make-priors` Save a default priors file template to the file name given.
- `--max-secs` Maximum number of top SECs to retain. The more SECs retained, the slower the process to merge them to obtain a final BN. Default: 30
- `-p, --priors` File containing the expert priors.
- `-P, --priors-string` Expert priors as a string.
- `-q, --quiet` Only output the BN and errors; don't output status information. Default: false
- `--rand-seed` Set the random seed to use (for the Java random number generator). Default: 0
- `-s, --speed` Search speed. Lower is faster (and less extensive). Default: 1.0
- `--wallace-rand-seeds` Set the random seeds to use (for the Wallace random number generator — requires two integers, separated by a comma and no spaces. e.g. 2983749,-349879).

4 Data and Data Formats

This version of CaMML is able to learn causal Bayesian networks from data sets with discrete variables only, without any missing/unobserved values. An additional version of CaMML (“Linear CaMML”) which learns from continuous data sets (linear gaussian models) is available here: <http://bayesian-intelligence.com/software/>.

4.1 Importing Data

4.1.1 Supported Formats

CaMML currently supports four data formats. The main two formats are Weka's Attribute-Relation File Format (**.arff**) and comma separated values (**.csv**); **.data** and **.cas** ('Old CaMML' data format) are also supported.

Weka is a program for machine learning and data mining, and is available here (<http://www.cs.waikato.ac.nz/ml/weka/>). Converting data to **.arff** format can be done via Weka (discussion of which is beyond the scope of this document); it can also be done manually in some cases (the **.arff** format is human-readable and is quite simple). The **.arff** format is discussed here (<http://weka.wikispaces.com/ARFF>) and here (<http://www.cs.waikato.ac.nz/ml/weka/arff.html>).

Data in CSV (comma-separated values) format can also be used. The first row of the data contains the variable names, and subsequent rows contain the actual data. Variable names must conform to the following format: first letter is alphabetic, and remaining letters are alphanumeric or an underscore.

Unlike the other formats supported by CaMML, data in CSV format does not contain metadata about the variables. Thus, CaMML will scan the data and attempt to determine the type (i.e. continuous or discrete) and the states for each variable (i.e. the set of values that each variable may take). This allows data to be quickly and easily imported from spreadsheet software (such as Excel), with the down-side that any unobserved variable states will be ignored.

Examples of each data format can be found in the `\example-data\` directory within the main CaMML directory.

4.2 Exporting Bayesian Networks

CaMML exports Bayesian networks to the plain text (i.e. human-readable) Netica (**.dne**) file format. Netica is a commercial Bayesian network package (<http://www.norsys.com/netica.html>), though a free version (limited in network size) is available. The free version is adequate for opening (and running inference on) the networks learned in CaMML, but will be unable to save changes to networks with more than a small number of variables. Netica may require you to compile the network (*Network* → *Compile*) before you can run inference on it.

Note that Bayesian networks exported from CaMML are saved without any layout information.

4.2.1 Bayesian Network Format Conversion

Netica itself won't be much help with regard to getting the Bayesian networks learned with CaMML into other formats. However, one of the following packages may be of use:

- GeNIe (<http://genie.sis.pitt.edu/>) — Free, can do Netica (**.dne**) to GeNIe (**.dsl** and **.xdsl**), Hugin (**.net**), **.dsc** and Ergo (**.erg**) formats.
- SamIam (<http://reasoning.cs.ucla.edu/samiam/>) — Free, can do Netica (**.dne**) to GeNIe (**.dsl** and **.xdsl**), **.dsc** and Ergo (**.erg**) formats.

A list of software packages for Bayesian network (and other graphical models) is maintained by Kevin Murphy here: <http://people.cs.ubc.ca/~murphyk/Software/bnsoft.html>. You may find a package that can convert Netica's **.dne** format to the format you require.

5 Improving Performance

This section very briefly discusses some ways to improve the quality learned networks (essentially, so we are more likely to find/learn the true/best network in the model space). Some short tips on making CaMML complete searches faster are also discussed.

5.1 Improving Model Quality

The quality of the resulting model(s) depends on a number of factors.

Dataset size: To a first approximation, more data is always better. CaMML trades off complexity of models with how well they ‘fit’ the data. Increasing the amount of data can thus have two effects: more complex models can (in principle) be learned; also, more data allows different models to be more easily differentiated. The downside of more data is that learning will take longer with a large dataset than a small dataset.

Expert prior information: Expert priors can improve (sometimes significantly) the quality of the resulting networks generated by CaMML. In many domains, there is at least some prior information known about the domain; whenever possible, this should be included. Again, the trade-off is that learning with expert prior information takes longer than learning without expert prior information.

Discretization: This version of CaMML supports learning from discrete data sets only. If the data set includes continuous variables that have been discretized before being loaded into CaMML, the details of how the discretization was done can affect the resulting network. For example, if a continuous variable is discretized with too many or too few states, learning a network can be difficult. Higher variable arity/cardinality means more parameters must be learned, which means we need more data to support complex models.

Proportion of model space sampled: The number of possible networks is very large. CaMML attempts to sample this space using MCMC sampling (i.e. the Metropolis Algorithm). Briefly, the more of the model space that is sampled, the more reliable the results. Generally, CaMML will sample more of the model space if we run the algorithm for a longer period of time — this can be achieved by increasing the **Search Factor** value in the Setup tab.

Parameterization: Nodes in a Bayesian network can be parameterized using a number of different types of conditional probability distributions. CaMML supports the learning of CPTs, decision trees and logit models (see Section subsection2.1.1 for details) — depending on the source of the data, one of these distributions may be more suitable than the others. CaMML’s trade-off between model complexity and fit to data also extends to the parameters in a network (and data encoded using those parameters), and thus the choice of parameterization type can affect the presence or absence of arcs in the resulting network. Using multiple parameterization types (i.e. CPT + DTree + Logit) can often improve the quality of the resulting networks, compared to using one of these alone. For example, the smaller number of parameters in DTree and Logit models may allow for more densely connected networks (vs. using CPTs only), especially when data set is small (and/or variable arity is large).

5.2 Reducing Computation Time

CaMML can take a significant amount of time to run on large datasets. Specifically, the number of model changes attempted by CaMML (and hence, the search time) is cubic in the number of variables (though the actual number of possible models is exponential in the number of variables).

If the computation time is proving infeasible for a data set, the following aspects may help:

- Fewer variables in the data set
- Smaller number of observations (i.e. less data)
- Reducing number of model changes attempted (reduce **Search Factor** on Setup tab)
- Reducing the number of parameterization types (i.e. using CPT only instead of CPT + DTree + Logit)
- Running search without expert priors (as the inclusion of expert priors has a computational overhead)

Clearly, each of these options have trade-offs (in terms of model quality) that need to be considered.

6 Technical Details

A description of the technical details underlying CaMML are beyond the scope of this document; however, readers may want to refer to the following references in order to understand what CaMML is doing ‘under the hood’:

- Korb, Nicholson (2011). *Bayesian Artificial Intelligence (2nd ed.)*. Boca Raton: CRC Press
- O’Donnell, R (2010). *Flexible Causal Discovery with MML*. Doctoral dissertation, Monash University, Clayton, Australia.

Readers interested in understanding more about the Minimum Message Length principle are directed to the following:

- Wallace (2005). *Statistical and Inductive Inference by Minimum Message Length*. New York: Springer

7 Common Issues and Solutions

7.1 Console output: Node costing warnings

Issue: Console output gives a number of “WARNING: Costing node ...” messages, possibly followed by a “WARNING: Too many Node costing warnings, reporting disabled” message.

Discussion: This is a warning (not an error), and is relatively common. This warning occurs when CaMML attempts to create and score a model with a node that has too many parameters. For example, a CPT could potentially have many thousands of parameters, if the node and its parents have high arity (i.e. many possible states). In cases where the

parameters cannot be learned (for example, when the number of parameters is greater than the number of observations in the data set), this warning will be produced.

No action is required, and the search will continue as normal.

7.2 Few (or very simple) models produced during search

Issue: Search is producing very few models (i.e. only a single MMLEC in the results), and/or the few results are relatively simple (i.e. few arcs).

Discussion: This can be due to a number of reasons, including:

- There are few good models in the model space for the current data set
- CaMML is unable to search through much of the model space

In the former case, the variables in the data set may be (approximately) conditionally independent of all other variables, and hence the complexity/fit trade-off is far towards the end of low complexity. Alternatively (and perhaps less likely), if there is one (or a small number of) very good model(s) (compared to all other models) the search may not devote much time to visiting other (poorer) models.

In the latter case, one possible reason for this is the size of the data set. If the data set is small (and the number of states of each variable is large), then CaMML may not have enough data to learn the parameters for anything but relatively simple models.

Possible solutions: Depending upon the cause, the following steps may improve the situation:

- Use a larger dataset (if available)
- Use multiple parameterization types (i.e. CPT + DTree + Logit on the **Setup** tab) - Decision Trees and Logit models have fewer parameters than CPTs, and thus may allow for more complex network structures (especially with small data sets).
- Run the search for longer (i.e. increase the **Search Factor** on the **Setup** tab) — although this is unlikely to help for small datasets (i.e. ≤ 10 variables, or few observations)
- Adding expert prior information (if available) may help in some circumstances

7.3 Console output: `java.lang.OutOfMemoryError: Java heap space`

Issue: “`java.lang.OutOfMemoryError: Java heap space`” is displayed in console output window.

Discussion: When dealing with large datasets (that is, datasets with a large number of observations, a large number of variables, or very high cardinality variables), it is possible for CaMML to run out of memory.

Typically this is due to the amount of (heap) memory allocated to the Java virtual machine when CaMML is started, rather than a limit on the amount of physical memory available in the computer.

Possible Solution: To increase the amount of memory available to CaMML, edit the GUI start-up script (i.e. `camml_gui.bat` or `camml_gui.sh`) and edit the `Xmx` command line argument. By default, the `-Xmx512m` argument is passed, which specifies that 512MB is allocated to the JVM heap.

To assign 1GB of memory, change this argument to ‘-Xmx1024m’ or ‘-Xmx1g’; for 2GB of memory use ‘-Xmx2g’ and so on.

8 Usage Examples

This section will demonstrate how to use CaMML on the the examples found in the `/example-data/` directory of the CaMML installation.

8.1 “Asia Problem”

This example contains case data for the “Asia Problem”, described by Lauritzen and Spiegelhalter. The data consists of 1000 values for the variables of the model, which can be used to learn the underlying Bayesian Network. For further information describing the initial problem see S.L. Lauritzen and D. J. Spiegelhalter *Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems* (Paper available here: <http://intersci.ss.uci.edu/wiki/pdf/Lauritzen1988.pdf>).

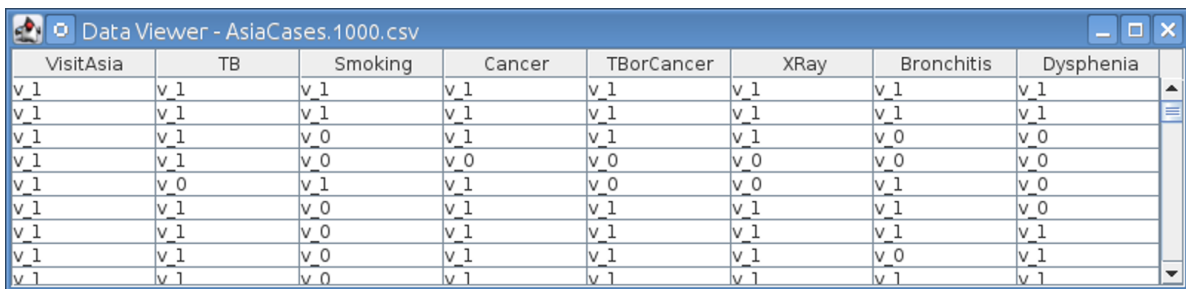
The example consists of three files:

- `AsiaCases.1000.cas`: data entries for the model in an old-format CaMML `.cas` file.
- `AsiaCases.1000.csv`: data entries for the model in a `.csv` file.
- `AsiaCases.priors`: the priors for the model.

Both the `.csv` and `.cas` files contain the same data, but in slightly different formats.

To load the Asia Cases data, select the **Setup** tab and click the **Load** button on the top right hand side. Use the file window to navigate to the `/example-data/` directory within your CaMML installation directory. Select either `AsiaCases.1000.cas` or `AsiaCases.1000.csv` and select open.

To view the data within the file, click the **View** button on the right-hand side of the **Setup** tab, next to the **Load** button. This will open the Data Viewer window, displaying the data within the file, as in Figure 8. If you wish, you may open the other data file and view its contents to verify that both files contain the same data. Note that if you do so, the data displayed in the **View** window may be prepended with `v_` in cases where the original format of the data is numerical. This will not affect the results of the calculation.



VisitAsia	TB	Smoking	Cancer	TBorCancer	XRay	Bronchitis	Dysphenia
v_1	v_1	v_1	v_1	v_1	v_1	v_1	v_1
v_1	v_1	v_1	v_1	v_1	v_1	v_1	v_1
v_1	v_1	v_0	v_1	v_1	v_1	v_0	v_0
v_1	v_1	v_0	v_0	v_0	v_0	v_0	v_0
v_1	v_0	v_1	v_1	v_0	v_0	v_1	v_0
v_1	v_1	v_0	v_1	v_1	v_1	v_1	v_0
v_1	v_1	v_0	v_1	v_1	v_1	v_1	v_1
v_1	v_1	v_0	v_1	v_1	v_1	v_0	v_1
v_1	v_1	v_0	v_1	v_1	v_1	v_1	v_1

Figure 8: The View Data window, displaying the data contained in `AsiaCases.1000.csv` file.

The top row of the viewer lists the names of the variables across the top of each column, with their corresponding data entries listed below. The eight variables are *VisitAsia*, *TB*, *Smoking*, *Cancer*, *TBorCancer*, *XRay*, *Bronchitis*, and *Dysphenia*. Under each variable name is a column

of data, which for this model contains binary values. The order of the columns in the viewer can be altered by clicking on a variable name and dragging that column to a new location. However, note that such changes will not be saved upon closing the view window. You may also copy the data from the viewer window by highlighting the relevant data fields and pressing the keyboard shortcut for Copy (typically Control+C or Command+C). You may then paste this data into external applications.

For this model the default settings on the **Setup** tab are acceptable, so no further changes are necessary.

Now change to the **Expert Priors** tab. To load the priors for this model, check the **Use Expert Priors** checkbox. Doing so will enable the various interface elements on this tab. The editor window has filled with a default priors file, containing the necessary structures and comments. To load the priors for this model click the **Load** button and select the `AsiaCases.1000.priors` file. If this file is not visible in the file window, you may need to change to the `/example-data/` directory within your CaMML installation directory and change the **Files of type** option from 'Text Files' to 'All Files'. Once the priors are loaded the contents of the window will change from the default comments to those in the file. To validate the loaded priors, click the **Validate** button. Now note the description at the bottom of the window — 'Expert Priors: Set, Validated', indicating that they are set correctly. Once the priors are validated the data is ready to be analysed.

The **Run** tab allows you to run CaMML on your model. Check the console here; if you clicked **Validate** on the **Expert Priors** tab, then there will be a corresponding line of output here listing the labels of the eight model variables. If you did not validate the priors previously, CaMML will do it for you once it starts. To run CaMML on the model, simply click **Run CaMML**.

As CaMML searches the included data output will be displayed in the **Console Output** text window. Once CaMML has finished the console output will display a 'Search finished' message and a summary of the search, as in Figure 9. The results can be viewed on the **Results** tab.

Changing to the **Results** tab to view the results shows the networks that CaMML has learned from this data. Clicking the **Arc Probabilities** button opens a table of the computed arc probabilities in a separate window. Clicking the **View Selected** button displays a visual representation of the model in a separate window. The nodes in this view may be moved by clicking and dragging, although any changes will not be saved. To load one of the networks in another program, click the **Export Selected** button to export only the selected model or click **Export All** to export all the models. The network will be exported into a Netica Bayesian Network file (`.dne`). Tools for opening this network are briefly discussed in Section `subsubsection4.2.1`.

8.2 Letter Recognition

The letter recognition example (contained within `letter.symbolic.arff`) contains data that describes a set of 20,000 letters, which was generated by randomly distorting pixel images of uppercase letters in 20 different fonts. The letters are summarised by 16 attributes that each partially describe the the letter contained in the image. For further information about this model and its source data, see the comments within the file itself, or see P. W. Frey and D. J. Slate *Letter Recognition Using Holland-style Adaptive Classifiers* (Machine Learning Vol 6 #2 March 91) (Paper available here: <http://cns-classes.bu.edu/cn550/Readings/frey-slate-91.pdf>).

The data within this model can be searched to produce a network that details which of the 16 attributes occur within each of the 26 letters. Such a network could be used to classify new

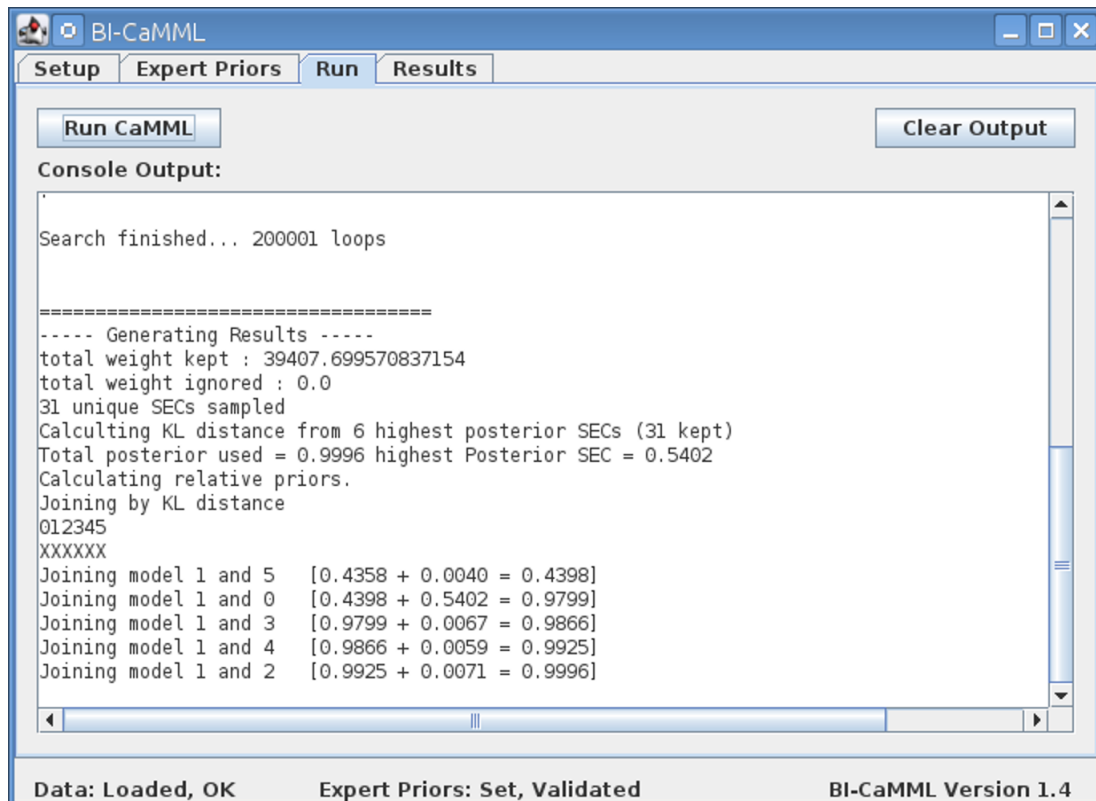


Figure 9: A Search Complete message and brief Search results in the console window.

letter inputs into their correct category, although it is included here merely as an example for CaMML.

To start, open this model from the **Setup** tab. Viewing this model with the **View** button shows that it has 17 variables that describe the various properties of a letter and the actual letter represented. The default parameterization type is 'MML: CPT', which is suitable for this model. The default values will be suitable for this model and can remain unchanged. Since the data is not a time series, the **Learn Dynamic Bayesian Network** checkbox should remain unchecked. Since this model does not have any expert priors to provide, the **Expert Prior**'s tab remains unchanged.

The **Run** tab allows you to run CaMML on this model by clicking **Run CaMML**. The output produced by CaMML will appear in the console window. In this example note the repeated warnings raised, which can be ignored in this instance (see Section subsection7.1 for more details). With MML: CPT as the parameterization type this model will be quickly searched. Scrolling down in the console window displays a completion message and a summary of the search process.

The **Results** tab shows that one model has been learned. Clicking the **View Selected** button displays a visual representation of the model in a separate window. Arc probabilities can be viewed by clicking the **Arc Probabilities** button. To see a visual representation of the Bayesian Network created from the data, click the **View Selected** button. If desired, this network can be exported by clicking on the **Export Selected** button, while the model is selected, or **Export All** button. The network will be exported into a Netica Bayesian Network file (.dne). Tools for opening this network are briefly discussed in Section subsubsection4.2.1.