R package for Bayesian Network Structure Learning

Francesco Sambo, Alberto Franzin

August 12, 2014

1 Introduction

Bayesian Networks are a powerful tool for probabilistic inference among a set of variables, modeled using a directed acyclic graph. However, one often does not have the network, but only a set of observations, and wants ro reconstruct the network that generated the data. The `bnstruct` package provides objects and methods for learning the structure and parameters of the network, also in presence of missing data, and a set of additional tools to use Bayesian Networks, such as methods to perform belief propagation.

This document is intended to show some examples of how `bnstruct` can be used to learn and use Bayesian Networks. First we describe how to manage datasets, how to use them to discover a Bayesian Network, and finally how to perform some operations on a network.

Complete reference for classes and methods can be found in the package documentation.

2 Installation

The latest version of `bnstruct` can be found at `http://github.com/sambofra/bnstruct`.

In order to install the package, it suffices to launch
`R CMD INSTALL path/to/bnstruct`
from a terminal, or to use R command `install_packages`. `bnstruct` requires R $\geq$ 2.10, and depends on `bitops`, `igraph`, `Matrix` and `methods`. Package `Rgraphviz` is requested in order to plot graphs, but it is not mandatory.

3 Datasets

The class that `bnstruct` provides to manage datasets is the *BNDataset* object. It contains all of the data and the informations related to it: raw and imputed data, raw and imputed bootstrap samples, and variable names and cardinality.

```
> dataset <- BNDataset(name="Example")
```

3.1 Data format

`bnstruct` requires the data files to be in a format we describe in this section. The actual data has to be in (a text file containing data in) tabular format, one tuple per row, with the values for each variable separated by a space or a tab. Values for each variable have to be numbers, starting from `0`. Data files can have a first row containing the names of the corresponding variables.

In addition to the data file, a header file containing additional informations can also be provided. An header file has to be composed by three rows:

1. list of names of the variables, in the same order of the data file;
2. a list of integers representing the cardinality of the variables, in case of discrete variables, or the number of levels each variable has to be quantized in, in case of continuous variables;
3. a list that indicates, for each variable, if the variable is continuous (`c`) or discrete (`d`).

We provide two sample datasets, one with complete data (the `Asia` network) and one with missing values (the `Child` network), in the `extdata` subfolder; the user can refer to them as an example.

3.2 Importing a dataset

The preferred way to create a *BNDataset* object is by reading a dataset from a file. In order to accomplish this, we provide the `read.dataset` method.

```
> dataset <- BNDataset(name="Example")
> dataset <- read.dataset(dataset,
+                         header.file = "path/to/file.header",
+                         data.file   = "path/to/file.data")
```

The sample datasets we provide come with two custom loaders:

```
> asia.data  <- asia()
> child.data <- child()
```

3.3 Imputation

A dataset may contain various kinds of missing data, namely unobserved variables, and unobserved values for otherwise observed variables. We currently deal only with this second kind of missing data. The process of guessing the missing values is called *imputation*.

We provide the `impute` to perform imputation.

```
> dataset <- BNDataset(name="Example")
> dataset <- read.dataset(dataset,
+                         header.file = "path/to/file.header",
+                         data.file   = "path/to/file.data")
> dataset <- impute(dataset)
```

Imputation can also be performed during the loading of a dataset, as shown in the following example.

```
> dataset <- BNDataset(name="Example")
> dataset <- read.dataset(dataset,
+                         header.file = "path/to/file.header",
+                         data.file   = "path/to/file.data",
+                         imputation  = TRUE)
```

Note that, when imputed data is present, it has higher priority over raw data when using a dataset (see section 3.5).

The sample dataset available using the `child()` method contains raw and imputed data.

3.4 Bootstrap

`BNDataset` objects have also room for bootstrap samples, both for raw and imputed data. We provide the `bootstrap` method for this.

```
> dataset <- BNDataset(name="Example")
> dataset <- read.dataset(dataset,
+                         header.file = "path/to/file.header",
+                         data.file   = "path/to/file.data")
> dataset <- bootstrap(dataset, num.boots = 100)
> dataset.with.imputed.samples <- bootstrap(dataset,
+                             num.boots = 100, imputation = TRUE)
```

Again, the generation of bootstrap samples can be performed during the loading of a dataset.

```
> dataset <- BNDataset(name="Example")
> dataset <- read.dataset(dataset,
+                         header.file = "path/to/file.header",
+                         data.file   = "path/to/file.data",
+                         bootstrap   = TRUE,
+                         num.boots   = 100,
+                         imputation  = TRUE)
```

The sample datasets provided have no bootstrap samples in them.

3.5 Using data

After a `BNDataset` has been created, it is ready to be used. The complete list of methods available for a `BNDataset` object is available oin the package documentation; we are not going to cover all of the methods in this brief series of examples, but we just show how to retrieve data.

The main operation that can be done with a `BNDataset` is to get the data it contains. The main methods we provide are `get.raw.data`, `get.imputed.data` and `get.data`. `get.data` is just a proxy for one of the other two methods. As previously mentioned, imputed data (if present) has higher priority over raw data, since it is supposed to be more useful. Therefore, if imputed data is present, `get.data` will behave as `get.imputed.data`; otherwise, it will return the raw dataset just like `get.raw.data`.

```
> dataset.1 <- child()
> # if we want raw data
> get.raw.data(dataset.1)
> # if we want imputed dataset, the following are equivalent
> get.imputed.data(dataset.1)
> get.data(dataset.1)

> dataset.2 <- asia()
> # we only can get raw data, the following are equivalent
> get.raw.data(dataset.2)
> get.data(dataset.2)
```

We can check if a dataset has imputed data or not with the `has.imputed.data` method.

```
> dataset.1 <- child()
> has.imputed.data(dataset.1) # TRUE

> dataset.2 <- asia()
> has.imputed.data(dataset.2) # FALSE
```

In order to retrieve bootstrap samples, one can use the `boots` and `imp.boots` methods for samples made of raw and imputed data. The presence of imputed samples can be tested using `has.imp.boots`. We also provide the `get.boot` method to directly access a single sample. Again, imputed samples have higher priority.

```
> # get imputed samples
> for (i in 1:num.boots(dataset))
+   print( get.boot(dataset, i) )

> # get raw samples
> for (i in 1:num.boots(dataset))
+   print( get.boot(dataset, i, imputed = FALSE) )
```

4 Bayesian Networks