The BayesNetExplorer package

Watkins W.S.[1], Hernandez, E.J.[1], and Tristani-Firouzi, M.[2]

[1]Department of Human Genetics, University of Utah, Salt Lake City, Utah, 84105, USA.
[2]Nora Eccles Harrison Cardiovascular Research and Training Institute, and Division of Pediatric Cardiology, University of Utah, Salt Lake City, UT 84112, USA

July 29, 2022

Overview

The BayesNetExplorer (BNE) package simplifies the creation, analysis, and interpretation of Bayesian networks. A user selects a target variable and calculates the probability of a target state outcome given one or more conditional variables. Absolute and relative risk ratios are obtained from any combination of conditional variables. Those variables with the greatest impact on the target are quickly identified, and the magnitude of their effect on the target, alone or in combination, are directly quantified.

A user simply provides a standard table of observations and variables. A Bayesian network is then created directly from the data using exact or approximate network learning methods. The relationships among all input variables is visualized using either a directed acyclic graph (DAG) or an undirected graph network.

A target variable is selected for analysis. That target is analyzed alone or conditionally with other variables in the network. If the variables are binary, all possible combinations of a target state with all combinations of conditional variable states can be analyzed. The absolute and relative risks for any target outcome with any combination of conditional variables can be calculated. Multistate discreet and discretized continuous variables can be analyzed, and user-selected conditional variable combinations can be fully traversed. Markov blankets for target variables are indicated graphically to allow the user to quickly identify those variables with the greatest effect on a specific target.

The current package utilizes the bnstruct and gRain packages from R [1, 2]. Although the BayesNets.jl and other julia packages for Bayesian networks are under development, bnstruct implements several network learning methods. Future versions of this package may also incorporate structure learning algorithms in julia as they become available.

Requirements

The user must have Julia (v1.7+) installed. Additionally, a fully working version of R (v3.6+) is also required. A GKS-term, such as XQuartz (MacOS) or XGKS (linux), should be installed.

This package has been developed and tested with Julia 1.7.3 and R 4.1.2 in an MacOS-Montery (Intel64-emulated) environment.

Installation

First, install R on your machine [https://www.r-project.org or https://www.rstudio.com/]. From within R, add the bnstruct and gRain packages with all of their dependencies. You will likely need to install BioConductor for some of the dependencies. Make sure that you can start the bnstruct and gRain packages on the R command line or within R-studio before installing the BayesNetExplorer package.

Verify that XQuartz for MacOS [https://www.xquartz.org] or GKS for linux is available. Install the Julia language package on your machine [https://julialang.org/downloads/].

Start the julia package. Enter the package manager, and add the registered julia packages listed below. Use the package manager to install the BayesNetExplorer (BNE) package from github. The BNE package is currently unregistered, therefore please use the developer's github account to obtain this package.

```
% julia
julia> ]
(@v1.7) add CSV, DelimitedFiles, DataFrames, StatsBase, Statistics, Random, Plots,
GraphRecipes, RCall, LinearAlgebra, Distributions, Impute, Combinatorics, Suppressor
(@v1.7) add https://github.com/ScottWatkins/BayesNetExplorer.jl
julia> using BayesNetExplorer
```

1 Preparing data for BayesNetExplorer

The following section presents the main functions of the BNE package. The functions are presented in the typical order of use. Each function and all options are described in the light blue boxes. Actual commands are shown in light gray. A discussion section provides additional details for the functions. Worked examples, using common data sets for 1) transportation preferences for 500 commuters, and 2) diabetes risk, are presented. The specific examples are minimal, and users are encouraged to try the additional options available for network analysis and risk prediction.

1.1 Convert the raw data to input files for analysis

format_file(data; datacols=[ ], delim=",", minfreq=0.0, recode_bool=false)

Options:

datacols	use a subset of columns e.g. [1,2,5,7]	[ ]
recode_bool	input is strictly 0/1 or true/false	[false]
delim	set the delimiter for input file	[","]
minfreq	min frequency to retained a variable	[0.0]

The format_file() function creates the BN.data and BN.header files on disk which will be used for input to create the network. The output is shown in the REPL. Each variable in the raw data file is recoded to numerical values, $\{1, 2, ..., , n\}$. The mapping between the numerical states and the original states can be found in the recode.map disk file. This mapping will be used throughout all additional functions.

Discussion: The raw input data must be in a standard fully-labeled table with observations in rows and variables in columns. The data can be binary, discreet, or continuous discretized.

The raw data should normally be completely text-based. For text data input, use Y/N or other string values for two-state variables instead of true/false or 0/1. If the data is completely two-state, it can be represented as true/false or 0/1. Boolean data (true/false, 0/1) should never be mixed with regular string variables.

The format_file() function creates two formatted input files called BN.data and BN.header from tab- or comma-delimited tables. These files are stored on disk. You can also use a data frame as input to this function. Variable states will be automatically mapped and converted to numerical values for computation. A short example of text-based input using the popular survey data set is shown below. There are six variables, two with three states and four with two states. All observations (instances) and variables (features) are labeled.

An example input file:

```
IID,AGE,CITY,EDU,OCC,SEX,TRANS
id1,adult,big,high,emp,F,car
id2,young,big,uni,emp,F,train
id3,adult,big,uni,self,M,car
id4,old,small,high,emp,F,other
```

Another typical input example might include several simple true/false or 0/1 variables that describe discretized continuous variables. It is recommended to conceptually align variable states with the hypothesis being tested. For instance, if the hypothesis is, "These factors predict late onset diabetes", all true (or 1) states should align with the expected prediction of increased diabetes risk. Variable alignment simplifies the downstream interpretation of multistate conditional probability queries.

Using Boolean data can be faster for larger data sets. Boolean data sets must contain only 0/1 or true/false variables.

ID	T2D	AGE70	BMI35	A1c65
1	true	true	true	true
2	false	false	true	false
3	false	true	false	false
4	true	false	true	true

1.2 Worked examples

The test_data directory contains some small data sets used in the examples. The location of the test data sets for the unregistered BNE package is typically in the /.julia/dev/BayesNetExplorer/test_-data/ directory.

Start the julia REPL and type ; in the REPL to get a shell prompt. Use the shell to find and navigate to the BayesNetExplorer/test_data/ directory. Check that you can see the survey_data.csv file, then exit the REPL shell using the delete key.

```
% julia
> ;
> cd /path/to/BayesNetExplorer/test_data/
> ls
> <delete>
```

Begin by formatting the survey_data.csv file for input using the format_file() function.

```
format_file("survey_data.csv")
```

The original data is converted to numerical data and returned in a data frame. Change to the shell mode (type ;) and look in your current directory. The BN.data and BN.header files contain the formatted data. The recode.map shows the mappings from the input text data to numerical data. The recode.out file contains a copy of the data.

Try formatting the tiny boolean data set T2D.tsv. Boolean data sets contain only 0/1 and true/false values. The recode_bool key word formats the two-state Boolean data (required). Remember to set the delimiter to the tab character for this data set.

```
format_file("T2D.tsv", delim = "\t", recode_bool = true)
```

Look at the disk files using your shell. The BN.data, BN.header, recode.map and recode.out now contain the data for T2D.tsv. The format_file function will create new input files every time the function is called. Note that the boolean data is now represented with 1 and 2 data states. All input data will always be recoded to $\{1, 2, ..., n\}$. Perhaps you decide you are not interested in the AGE70 variable. You can just select the columns of the raw data you want to include in the analysis.

```
df, ids, features = format_file("T2D.tsv", delim="\t", recode_bool=true, data-
cols=[1,2,4,5])
```

You can also filter variables by the minimum observed class frequency using the minfreq option. The return values include a data frame of the variables, a list of all ids, and a list of all features. These variables can be used in other functions.

1.3 Imputation

impute_dataframe(df; boolout=false, k=10)

Options:

boolout	write the imputed matrix to disk	[false]
k	number of nearest neighbors to use	[10]

The user input data must be complete and contain no missing values. Missing values must be removed or imputed. As long as observations or variables have only partial random missing data (up to 20 percent, recommended), they can be used by imputing the missing the values. Avoid imputing large blocks of data where many adjacent samples are all missing the data.

To impute data, first read the data into a data frame in the REPL using the CSV.read() function. Use the impute_dataframe() function to impute missing data. The data frame to be imputed must contain numerical values only. Missing data is imputed using a nearest-neighbors approach with a default of k=10 nearest-neighbors. Strictly two-state data can be output as a Boolean file on disk by setting boolout to true. Discreet and multistate text data can also be imputed by first recoding the data to ordered numerical values $\{1, 2, ..., n\}$.

To impute an input data set, first read the raw data into a data frame.

```
rawdata = CSV.read("impute_example.csv", DataFrame, delim=",")
```

Note that the rawdata dataframe contains missing values for ID3, ID10, and ID15.

```
impdata = impute_dataframe(rawdata)
```

The imputed data frame now has no missing data and is suitable for BNE analysis. You can use the data frame directly or save the imputed data by writing the data frame to disk.

```
CSV.write("imputed_data.csv", impdata, delim=",")
```

1.4 Feature Selection

When a data set consists of a large number of features (variables), it can be difficult to decide which features are most relevant to the target variable. Computationally, an exact Bayesian network learned directly from data is limited to about 20 variables. Expert knowledge can be very useful for variable selection but may not always be available. The BNE package includes the feature_selector() function to help identify the conditional variables that have the greatest impact on a target variable.

The feature_selector() function calculates the probability of the target given a single conditional variable X, that is the $P(Target)|X_i..n$, where $X_i..n$ represents each single variable in the data set, excluding the target variable itself. All probability states for the target, conditioned on each test variable state, are calculated and sorted.

The probabilities are network propagated. The default net includes only the target variable, the conditional variable, and a generated random variable. Because there is typically little interdependence between user variables and the random variable, the default conditional probability estimates are usually similar to an exact calculation based on simple counts. The user can also select multiple random variables or select variables from the data set itself. In the latter case, interdependence among conditional variables will be reflected in the final probability estimates.

In addition to the conditional probability estimates, the output table shows the probability change relative the target's baseline probability as a percent. Conditional variables with large relative effects on the target variable states are easily identified. Features with low impact on the target can be ignored. Features with a high-impact on the target can then be selected to build a final probabilistic graphical model.

dffs, baseline = feature_selector(df, f="target", ids=ids, features=features)

Test each variable in a data set, one by one, against the target variable. Measure the change in the probability of the target variable conditioned on the test variable. Probabilities are network propagated with BNE.

Options:

features	any variable names to analyze	[ ]
ids	input ids if using dataframe input	[ ]
net	random or select from data	"random"
netsize	the number of nodes in the network	3

Note: Use the command df,ids,features = format_file(...) to first create a data frame (df), the BN.data file, the BN.header file, the list of ids, and the list of features. The BN.data and BN.header files can also be used as input instead of the data frame and ids lists. You may select a subset list of features to include only those of interest.

Discussion: Running the feature_selector() function is straight-forward. First, run the format_file() function (e.g. df, ids, features = format_file(...)) to obtain the list of features. You can use this data frame with the ids keyword as input. Alternatively, you can put the ids back into the data frame returned by format_file() using the command df = insertcols!(df, 1, ids). This new data frame is also valid input. The feature_selector() returns propagated probabilities and requires additional nodes to build the network. These additional nodes are randomly generated or randomly selected from the rest of the data. The propagated network size can be changed with the netsize argument.

When using variable randomly selected from the data, low netsize values can sometimes prevent network creation which leads to an error. Although most random variable combinations demonstrate net stable probability estimates, it is worth running the program two or more times to verify any single probability estimate of interest. It is typically more useful to use the default setting that uses one generated random variable because the conditional probabilities do not include network propagated interdependencies. A table of probabilities and the baseline probabilities of the target are returned.

2 Running BayesNetExplore (BNE)

cpt, tt, tf, adjm = bne("BN.data", "BN.header"; algo="sm", scoring_-method="bic", f="", fs=0, g=[ ], gs=[ ], verbose=false, DAG=false)

Construct and explore a Bayesian network. Inputs are the formatted data and header files (see the format_file() function). Input data must be binary, discrete, or discretized continuous $\{1, 2, ..., n\}$. Return values are: 1) the conditional probabilities, 2) a dataframe of all input variables, mappings and state frequencies 3) target state frequencies and 4) the graph adjacency matrix.

Network learning algorithms

sm	Silander-Myllymaki exact search [default]
hc	Hill climbing method
mmhc	Max-Min hill climbing heuristic
sem	Structural expectation-maximization

Scoring methods

BDeu	Bayesian-Dirichlet equivalent uniform
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion [default]

Probability options (with input examples)

f	Target feature (required)	"F1"
g	Given (evidence) features	["F2", "F3"]
gs	Given (evidence) states	["2", "1"]
iterate	Iterate states	[nodes|all]
relrisk	Calc. rel risk ratio for target	[false]

Plot options

DAG	Plot is directed acyclic graph	[false]
plot	Network or Markov blanket "mb"	["net"]

Bootstrapping options

rr_bootstrap	number of bootstrap interations	[100]
bootstrap_method	resample or delete-half	[resample]
bootout	write and append results to file	""

Other options

minfreq	min frequency cutoff	[0.00]
verbose	verbose output	[false]

The bne() function automates the process of creating and analyzing Bayesian networks. You can easily obtain results for any conditional probability query over a medium sized network. It is important to remember that all of the conditional probability outputs are network propagated. When variables remain in the network but have little influence on the selected target, conditional probabilities are very similar to probabilities obtained using simple counts. Where there is high variable interdependence, the propagated probabilities will reflect this conditional dependence and may vary substantially from a non-propagated conditional probability estimate.

To get started, verify that you have created the two input files BN.data and BN.header with the format_file() function for the survey data set. This data set contains survey responses for transportation preferences of 500 commuters. It contains the transportation preference target variable, TRANS, and five additional variables: AGE, CITY, EDU (education), OCC (occupation), and SEX.

The first step in any BNE analysis is to specify a target feature (f) to analyze. Try the following simple example using the survey data set. Recreate the formatted input files (BN.data and BN.header) for the survey data set as before. If desired, add the semicolon at the end to suppress printing return values.

```
format_file("survey_data.csv");
```

Select the feature variable for analysis and assign it to “f” in the command. The feature/target variable should always be a string that matches one of the column names in the original input data set. We will focus the analysis on the three-state variable called “TRANS”.

```
bne("BN.data", "BN.header", f="TRANS");
```

The screen output shows the base probabilities for the feature “TRANS”. That is, the probabilities of the three states of the TRANS variable propagated through the Bayesian network without any conditional variables.

```
-----------------------------------------------------------
P(TRANS)|Any[]
Conditional state(s): Any[]

TRANS
1 2 3
0.5800927 0.1696161 0.2502911
-----------------------------------------------------------
```

Scrolling up, you can see a table showing the mapping of each analysis variable state to each original variable state. This table also includes the counts for each variable and each variable state. In our data set, 58 percent of people prefer to drive their car to work, 25 percent prefer to take the train, and almost 17 percent prefer to use other forms of transportation. Note that frequencies for TRANS in the table are a very close match to the unconditioned propagated probability estimates.

A graphical representation of the Bayesian network is also created. The target feature (TRANS) is shown in orange in the graph. Nodes are connected by lines. By default, no direction in shown because causality is usually difficult to determine. Nevertheless, seeing the directed acyclic graph (DAG) provides a better feel for the relationships found by the structure-learning algorithm. The directed network can be seen using the DAG=true keyword.

Now include a conditional variable query to see its effect on the target variable probabilities. To query of the probability of A given B, $P(A|B)$, list each conditional variable in the input array, "g" (e.g. g=["EDU"]). The specific state of the conditional variable is specified in the input array, "gs" (e.g. gs=["2"]). The elements of g and gs must be quoted. When there are multiple conditional variable, the elements of "g" and the elements of"gs" must be in matching order.

Let's see what effect having a university education (EDU) and being young (AGE) has on the target variable transportation (TRANS). The following query explores the three possible probability states for the target TRANS (car, other, and train) given a university education and being young, $P(TRANS)|(EDU = 2, AGE = 3)$. Refer to the translation table that maps 2 to "uni" and 3 to "young", respectively by scrolling up in the REPL.

```
bne("BN.data", "BN.header", f="TRANS", g=["EDU", "AGE"], gs=["2","3"]);
```

```
P(TRANS)|["EDU", "AGE"]
Conditional state(s): ["2", "3"]

TRANS
1 2 3
0.5539740 0.2070197 0.2390063
```

The probabilities for the TRANS have changed. The TRANS target probabilities (1 for car, 2 for other, and 3 for train) are now conditionally dependent on the EDU and AGE variables. The results show that young (AGE=3), university-educated people (EDU=2) travel less by car and more by other means of transportation than the general public, as represented by our sample random sample of 500 people. The results provide new insight into the transportation preferences of one segment of the population.

2.1 Iteration

One important feature of the BNE package is the ability to traverse the probabilities of a target state given many conditional variables. The iterate keyword controls this function. We will now explore the target TRANS and iterate over all states of the two given variables EDU and AGE. It is no longer necessary to specify the conditional states (gs=["2", "3"]) because all states will be examined automatically. We will also examine the four objects returned from the bne() function.

```
cpt, tt, tf, adjm = bne("BN.data", "BN.header", f="TRANS", g=["EDU", "AGE"], iterate="nodes", DAG=true);
```

The cpt variable is a data frame table that contains all the major results. Type cpt to see the table. Each state for the target variable TRANS (TRANS_car, TRANS_other, TRANS_train) gets a column in the table. The columns are sorted from highest to lowest, sequentially. The target probabilities for each row sum to 1. Next is a column containing the conditional variable combinations followed by a column showing the corresponding conditional variable states for each combination. Each row in the table shows one or more conditional variables and the conditional variable state(s) alone or in combination with the other conditional variables and their states. In this case, there are 11 rows representing the 11 possibilities: three rows for AGE alone, two rows for EDU alone, and six rows for combinations of AGE and EDU states.

Target	TRANS_car	TRANS_other	TRANS_train	CondVars	CondStates
TRANS	0.593	0.166	0.240	AGE,EDU	3,1
TRANS	0.589	0.157	0.252	EDU	1
TRANS	0.589	0.156	0.254	AGE,EDU	1,1
TRANS	0.586	0.147	0.266	AGE,EDU	2,1
TRANS	0.582	0.153	0.263	AGE	2
TRANS	0.581	0.178	0.239	AGE	3
TRANS	0.578	0.170	0.251	AGE	1
TRANS	0.553	0.207	0.239	AGE,EDU	3,2
TRANS	0.553	0.203	0.243	EDU	2
TRANS	0.553	0.201	0.245	AGE,EDU	1,2
TRANS	0.553	0.200	0.246	AGE,EDU	2,2

Iterating over all states of the AGE and EDU variables provides insight into exactly how age in combination with education affects transportation preferences. Take for instance the preference to travel by car. We can see that young people with a high school education are most likely to travel by car, $P(TRANS = CAR|AGE = 3, EDU = 1) = 0.593$, while older university-educated people are least likely to travel by car, $P(TRANS = CAR|AGE = 2, EDU = 2) = 0.553$. The difference in car travel preference is about four percent.

Is AGE or EDU more important? Examine the table carefully. EDU=1 is in top four probabilities, and EDU=2 is in the bottom four probabilities. In contrast, all three AGE state are in both the top four and bottom four rows. The results also suggest that education not age has more impact on the preference to travel by car.

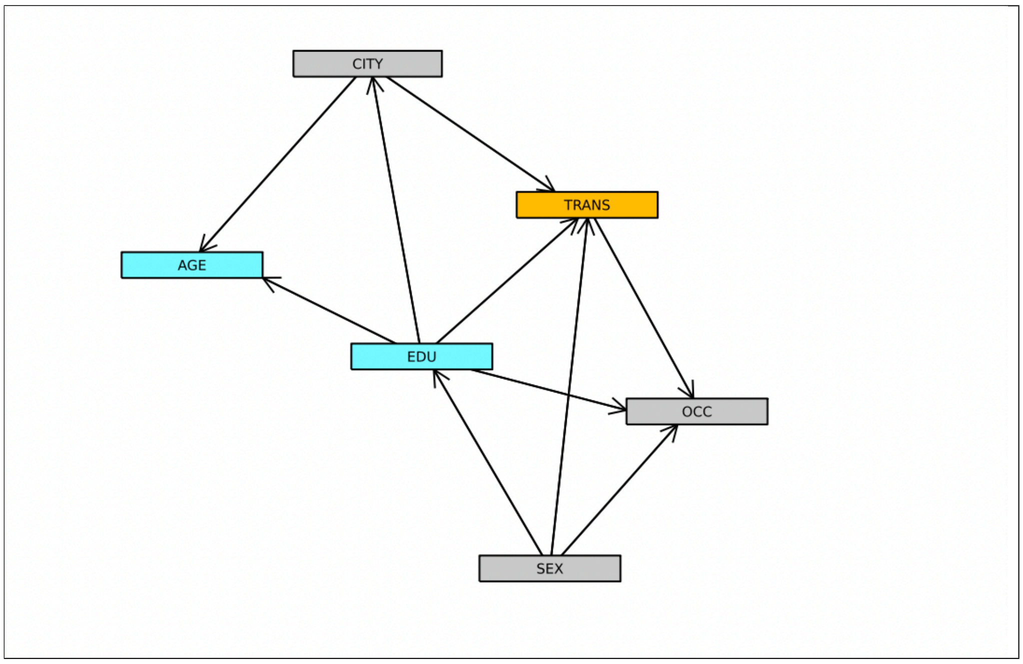
The tt variable contains the mapping, frequency, and counts for all variables and can be used to reference the variable states whenever necessary. The tf variable contains the baseline frequencies of the target variable states. The adjm variable contains the adjacency matrix for the graph. The adjm matrix can be used with the plot_network() function to customize the look and shape of the network. This is often necessary if the data set contains more than 8-10 variables.

Now set the iterate keyword to iterate="all" and examine the output again. The cpt table shows the target variable states with all conditional variables and all possible combinations of all conditional variables. In most cases it is necessary to filter the cpt table to remove conditional probability combinations that lead very small or zero samples in the conditional class.

2.2 DAGs, Networks, and Markov blankets

Each variable in the data set can be represented graphically as a node (vertex) in a graph. Learning algorithms attempt to find the best single graph structure that connects all nodes in a single directed acyclic graph (DAG). There are several learning methods, and different methods can produce different connections depending on the input data. Try running BNE with the DAG keyword set to true and examine the graphical output. The graph shows the target, TRANS, in orange and the selected conditional variables, EDU and AGE, in blue.

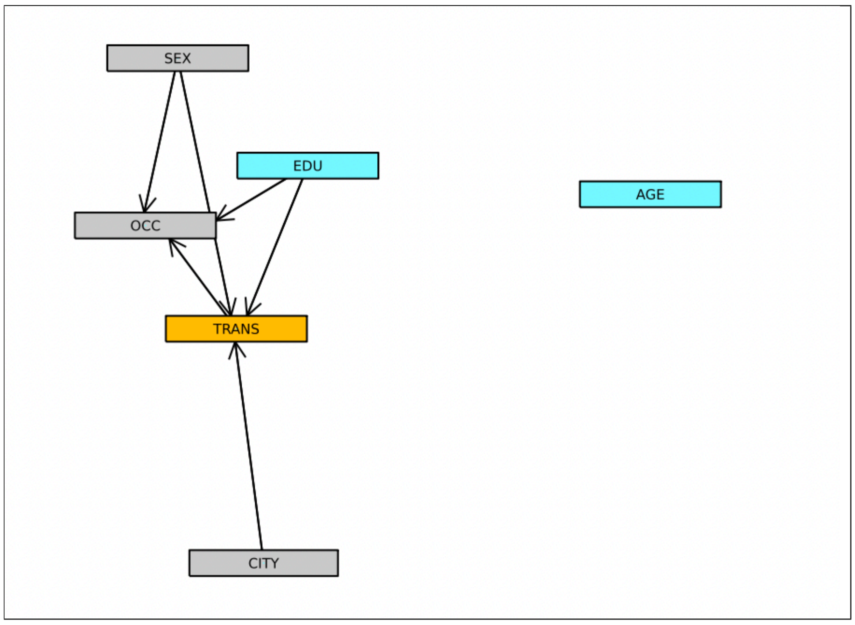
```
cpt, tt, tf, adjm = bne("BN.data", "BN.header", f="TRANS", g=["EDU", "AGE"], it-
erate="nodes", DAG=true);
```



The DAG also shows directional connections among the variables with arrows learned by the selected algorithm (default=sm). A DAG cannot have cycles. The arrows indicate the learned relationships among the variables but do not directly imply that one variable causes another. The nodes most closely connected to the target node will have the greatest influence on target's probabilities, especially those variables in the Markov blanket.

A minimal Markov blanket (or Markov boundary) for a target node is the collection of nodes that include 1) all parent nodes, 2) all offspring nodes, and 3) all offsprings' parent nodes. Because these nodes have the most influence on the target, it is helpful to first identify the minimal Markov blanket of a target variable given the data set. It is important to note that the minimal Markov blanket is not static. Adding or subtracting variables in the input data set can affect which variables are found in the blanket. The Markov blanket can be visualize graphically using the "plot" keyword. Plot the minimal Markov blanket for TRANS given AGE and EDU using all input variables.

```
cpt, tt, tf, adjm = bne("BN.data", "BN.header", f="TRANS", g=["EDU", "AGE"], it-
erate="nodes", DAG=true, plot="mb");
```



As before, the target variable, TRANS, is in orange and the user-selected conditional variables are in blue. The graph of the minimal Markov blanket shows that EDU falls within the blanket while AGE is outside the blanket. This graphical result agrees nicely with our analysis of the tabular results above. That is, EDU has a relatively large effect on TRANS compared to AGE. One method to quickly identify variables that have a high impact on a selected target is to simply plot the target variable with DAG=true and plot="mb".

```
cpt, tt, tf, adjm = bne("BN.data", "BN.header", f="TRANS", DAG=true, plot="mb");
```

3 Absolute and Relative Risk Calculations

One of the primary goals of a Bayesian network analysis is to understand how conditional variables change the risk of a target event relative to the baseline risk of that target event in a study population. For example, the baseline incidence of a middle aged male experiencing a heart attack may be about one percent. Individual risk factors such as fatty diet, smoking, vigorous exercise, etc. may increase or decrease that general risk. Bayesian networks are well-suited for discovering and quantifying the factors that increase the probability of an event or trait. The RRcalculator() function provides a straight-forward method to discover and calculate relative risks for one or more conditional variables that influence a target variable state such as disease risk or other event outcomes.

RRcalculator(cpt; target_state="", target_state_freq=0.0, translation_table=tt, mincondcount=10, minp=0.01, maxp=0.99)

Options

minp	minimum probability to report	[0.01]
maxp	maximum probability to report	[0.99]
filterstates	remove conditionals containing	[ ]
mincondcount	min conditional subset count	[20]

The RRcalculator() function calculates relative and absolute risks from a probability table generated with the bne() function. The results for a selected target and the target's probability states are processed, sorted, and returned in a table. A probability filter can also be applied to select any desired conditional states or to remove any conditional state combinations with very low probabilities. Relative risk calculations are currently limited to two-state systems. Continuous or multistate variables can be used if they are discretized to two-state systems, as is common in many data sets. For example, the continuous variable, systolic blood pressure, could be divided into "high" and "low" categories based on a cut-off value of 140 mmHg.

The RRcalculator() returns a relative risk table. The relative risk table has several columns. The most important columns for most runs are the absolute and relative risk ratios. To aid interpretation, risk ratios are defined follows:

Baseline absolute risk: the frequency of the trait in the sample population. The baseline absolute risk is number of samples with the target trait divided by the total number of samples in the data set. This is the standard definition of absolute risk.

Conditional risk: the probability of the trait given one or more conditional states (e.g. $P(T2D)|Age = old, Glucose = high)$).

Absolute risk ratio: the frequency of the target trait in the conditional subset compared to the baseline absolute risk. In other words, what is the specific risk of the trait given the conditional variable states V1=true, V2=true, V3=false ..., compared to the absolute risk of the target trait in the general population. Depending on whether the conditional variables are predisposing or protective, the absolute risk ratio may be higher or lower than the baseline absolute risk.

Relative risk ratio: the frequency of the target trait given a set of conditional variable states compared to the frequency of the target trait without the conditional variable states. This is sometimes referred to as the contrapositive risk. In other words, if the conditional variable states are V1=true, V2=false, and V3=true, what is the risk of the target feature compared to having conditional variable states V1=false, V2=true, V3=false.

The relative risk ratio is an important metric for identifying and assessing the effects of environmental exposure variables, patient behavioral traits, or even genetic mutations on outcomes. For example, a relative risk ratio for a heart disease trait when "high stress" = true and "aerobic exercise" = false and "vaping" = true is found to be ten. The relative risk ratio for "high stress" = true and "aerobic exercise" = false is only three. The result suggests a strong influence of the vaping variable on the target. Eliminating vaping should greatly improve patient outcomes.

4 Putting It All Together: Type-2 Diabetes Risk

We will now use an actual data set of individuals sampled from a population with a high risk of type-2 diabetes (T2D) to examine three factors that may affect diabetes risk. We will assess the relative risk of T2D given these risk factors. Each factor is a continuous variable that was discretized using expert knowledge gained from previously published studies. The two-state variables are aligned to be true for increased diabetes risk when: body mass index (BMI) > 30, age > 40 years, and fasting glucose > 130.

First, prepare the input data from the T2D.csv file found in the test_data directory.

```
format_file("T2D.csv", recode_bool=true)
```

Note in the formatted data file that 0 is now 1, and 1 is now 2. The true state (2) is aligned with our expert knowledge that 2 indicates higher diabetes risk.

Now, make the conditional probability table for all nodes using type-2 diabetes as the target variable. For convenience, use the iterate="all"setting to test all combinations of the three conditional variables and their effect on the risk of diabetes.

```
cpt, ctt, tf, adjM = bne("BN.data", "BN.header", f="T2D", iterate="all", DAG=true);
```

The output indicates that 35 percent of the population has T2D. If our population sampling is random, this is the *baseline* absolute risk for T2D in this population, This estimate agrees reasonably well with other studies that show a prevalence of T2D in the range of 34 - 42 percent in this population.

Examine the first and last lines of the conditional probability table (cpt). We can see that when our (aligned) risk factors are all false, the probability of type-2 diabetes (T2D_true) is very low (0.09), and when our risk factors are all true, the probability of type-2 diabetes (T2D_true) is very high (0.82).

Target	T2D_false	T2D_true	CondVars	CondStates
T2D	0.91	0.09	BMI30,AGE40,GLUC130	1,1,1
T2D	0.88	0.12	BMI30,GLUC130	1,1
T2D	0.88	0.12	BMI30,AGE40	1,1
(additional lines ...)				
T2D	0.31	0.69	AGE40,GLUC130	2,2
T2D	0.27	0.73	BMI30,GLUC130	2,2
T2D	0.18	0.82	BMI30,AGE40,GLUC130	2,2,2

The relative risk calculator function, RRcalculator(), can now be used to calculate the absolute and relative risk ratios for each risk trait and for all possible combinations of the risk traits. All the necessary input data for the RRcalculator() has already been produced by the bne() run.

The main input to the RRcalculator is the cpt table from the BayesNetExplorer run. The target name_state (T2D_true), the baseline frequency of the target state (T2D_true) in the tf array, (0.348958), and the translation table (tt) are also needed. The minimum number of samples in the conditional subset (mincondcount) will be set to the default value of 20 (see important guidelines below).

```
RRtable = RRcalculator(cpt, target_state = "T2D_true", target_state_freq = 0.348958, translation_table = ctt, mincondcount = 20)
```

The relative risk table shows the detail risk assessment for the target T2D. The three main risk estimates of interest are: 1) the conditional probability, P(T2D_true)|Y, 2) the absolute risk ratio (AbsRiskRatio), and 3) the relative risk ratio (RelRiskRatio). Since all of our suspected risk factors are aligned and are known risk factors for T2D, interpretation is straight-forward. Look at rows 5, 6, and 8 to find the risk estimates of T2D conditioned on each of the risk variables alone. AGE40, BMI30, and GLUC130 each increase the risk of diabetes.

Of great interest is the combinatorial effect of these risk variables. The top four rows are particularly interesting. We can see that combining any two risk factors increases the probability of T2D substantially over the population baseline absolute risk (0.348958). Combining all three risk factors elevates the risk of T2D further. Individuals with BMI $>$ 30, AGE $>$ 40, and GLUC $>$ 130 have a conditional probability of 0.82 for T2D, a 2.3 fold greater risk of T2D that the study population in general, and a 9.2 fold greater risk of T2D than individuals with BMI $\leq$ 30, AGE $\leq$ 40, and GLUC $\leq$ 130. As a next step, we might gather data on exercise or the use of a new drug that affect glucose metabolism to determine their effects on T2D risk in this population.

4.1 Bootstrapping the Results

For reporting results, it is highly desirable to provide an estimate of the confidence in the risk estimate. BNE provides 95 percent confidence intervals for the absolute and relative risk ratio estimates by resampling with replacement or jackknife methods. Because resampling is computationally expensive, CI95 intervals are estimated for a single target and a specific set of conditions. Typically, the user will use the RRcalculator() function to first identify the most interesting results. Once one or a few results are selected, each can be bootstrapped using BNE. To generate the 95 percent CI for the top absolute and relative risk ratios found in the T2D data set, use the bne() function and add the relrisk=true keyword. Additionally, you must specify the target/feature-state (T2D_true) as a numeric string in the "fs" variable (fs="2").

```
bne("BN.data", "BN.header", f="T2D", fs="2", g=["BMI30", "AGE40", "GLUC130"],
gs=["2", "2", "2"], relrisk=true);
```

```
Relative risk:
P(T2D=2)|(["BMI30", "AGE40", "GLUC130"]); states=["2", "2", "2"]; P = 0.815385
vs.
P(T2D=2)|(["BMI30", "AGE40", "GLUC130"]); states=["1", "1", "1"]; P = 0.089109
Relative Risk: 9.1504 CI95: (7.8197, 11.8375)


Absolute risk:
P(T2D=2)|(["BMI30", "AGE40", "GLUC130"]); states=["2", "2", "2"]
vs.
P(T2D=2), all samples; P = 0.348958
Absolute risk: 2.3366 CI95: (2.2384, 2.4019)
```

The output shows probabilities and risk ratio calculated as before along with the 95 percent confidence intervals. Since the bootstrap procedure randomly resamples the data, the CI95 values will vary slightly. Variance should decrease as the number of bootstrap replicates increase, provided that the conditional sample size is adequate. The number of bootstraps can be increased using the rr_bootstrap keyword.

5 Optimization and Interpretation

Bayesian networks provide an excellent way to analyze the relationships among all variables in a data set. The networks and the relative probabilities provide an explainable artificial intelligence (AI) approach to investigate many questions. The key concepts for producing a reproducible analysis include 1) a large sample size and 2) an independent replication data set.

Sample size is very important. As the number of conditional nodes in a specific query or global analysis increases, the number of samples in the conditional subset may become very small. This is often the case when low frequency variables are combined in a conditional query. Consider a small data set with 100 samples. Variables V1=true and V2=true both have baseline frequencies in the population of about ten percent. The sample size is likely to be inadequate to accurately assess the conditional query $P(A)|(V1 = true, V2 = true)$ because of the low number of samples in the conditional subset. Increasing the sample size to about 1000 - 2000 would be needed to generate a highly stable estimate of the conditional probability of A and both low frequency variables. This does not imply that smaller data set are not useful, only that larger data sets produce more accurate and stable risk estimates.

Relative risk ratios are also strongly affected by small sample sizes in the conditional query. A conditional query combination of several variables may identify a few target samples with that combination (numerator). If the net-propagated contrapositive conditional combination (denominator) is close to zero, the relative risk ratio may be very large. This inflated relative risk ratio may not reflect the true relative risk. Consideration of the conditional sample size is always important to assess if a result is likely to be accurate. The analyst must always consider if the sample size is sufficiently large to accurately estimate the conditional (in)dependence of the target state and the conditional variables states.

As one additional example, in the tutorial results above, the first row of the cpt table shows that self-employed university-educated males living in a big city prefer to travel by car. This seems like a reasonable result. But, consider that there are only six samples that meet these inclusion criteria in the data set. As expected, based on the probability estimates, all six prefer car travel. Now consider sampling one additional individual that prefers train travel. The one additional sample shifts the probability for car preference by 14 percent. The effect could be much greater if the conditional subset included only three individuals. Estimating the 95 percent confidence interval for any final conditional probability is highly recommended.

6 Getting reproducible results

Please consider the following techniques to obtain high-quality, reproducible results.

1. Use the largest data set available and continue to add more data as new samples become available. Having enough data points in all states of all variables is critical for testing a target with even a single conditional feature and becomes increasingly important as the number of conditionals features increase.

2. Check the initial frequencies of all states for each variable. Consider omitting variables with a minimum initial frequency of less than five percent. Alternatively, merge low-frequency classes, if appropriate. Use results with at least 20 or more samples in the conditional subset (mincondcount=20). If 20 is too high, set mincondcount lower, but consider whether the overall size of the data set is adequately powered for that specific query.

3. Understand each feature you put into the final network. Mutually exclusive variables or one-hot encoded variable sets can distort networks and their propagated probabilities. Consider combining mutually exclusive variables into a single feature if appropriate. Be aware of features that are highly overlapping. For instance, two gene pathways may appear to increase the risk of some target trait, but if these pathways contain mostly the same genes, it may be better to use the intersect and the symmetric difference of the two gene lists.

4. Bootstrap the results. Use the bne() function with the rr_bootstrap option to perform bootstrap resampling of the data to generate 95 percent confidence intervals around the relative risk ratio estimates. Large confidence intervals often indicate low numbers in the conditional sample set.

5. Be critical. Does it make sense that you just identified some new factor (Q) that leads to a 450-fold increased risk in the $P(X)|(Q)$? Perhaps, but there may also be another explanation for the result. Always check the actual numbers in the input data set that produce the conditional and final probability estimates before reporting the final risk ratios.

With accurate input data and an adequate sample size, Bayesian networks provide a powerful method to assess the relationships among all variables in a data set. The network defines the relationships among all variables and allows any set of conditional probabilities to be propagated through the network. The networks yield highly accurate probability estimates for any target variable. All network relationship and probability estimates are transparent, leading to a fully explainable machine-learning framework for data analysis, hypothesis generation, and the interpretation of new results.

Acknowledgments

This package was inspired by many discussions of the utility and effectiveness of Bayesian network and explainable AI systems in the laboratories of Drs. Tristani and Yandell at the University of Utah. Many thanks to E.J. Hernandez and S. Wesolowski for their expertise in Bayesian and graphical systems analyses.

Please send comments, bug reports, and feature requests to scott.watkins@utah.edu.

References

[1] A. Franzin, F. Sambo, and B. Di Camillo. bnstruct: an r package for bayesian network structure learning in the presence of missing data. *Bioinformatics*, 33(8):1250–1252, 2017. Franzin, Alberto Sambo, Francesco Di Camillo, Barbara eng England Bioinformatics. 2017 Apr 15;33(8):1250-1252. doi: 10.1093/bioinformatics/btw807.

[2] S. Hojsgaard. Graphical independence networks with the grain package for r. *Journal of Statistical Software*, 46(10):1–26, 2012.