enddocument/afteraux

package/fontawesome/before

enddocument/info

BayesNetty

Release 1.2

Richard Howey

Dec 11, 2024

CONTENTS:

!! UNDER CONSTRUCTION !! Use the old BayesNetty webpage for now.

Please use the menu to the left to navigate through the documentation for BayesNetty or use the PDF of these webpages, bayesnetty.pdf.

CHAPTER

ONE

CITATION

Please use the following papers [1] and [2] to reference the BayesNetty software as follows:

```
@article{howey:etal:21,
author={Howey, R AND Clark, A D AND Naamane, N AND Reynard, L N AND Pratt, A G AND␣
↪Cordell, H J},
title={{A Bayesian network approach incorporating imputation of missing data enables␣
↪exploratory analysis of complex causal biological relationships}},
journal={PLOS Genetics},
month = {September},
volume={17},
number={9},
doi = {10.1371/journal.pgen.1009811},
url = {https://doi.org/10.1371/journal.pgen.1009811},
pages={e1009811},
year={2021}}

@article{howey:etal:20,
author = {Howey, R AND Shin, S-Y AND Relton, C AND Davey Smith, G AND Cordell, H J},
journal = {PLOS Genetics},
title = {Bayesian network analysis incorporating genetic anchors complements␣
↪conventional Mendelian randomization approaches for exploratory analysis of causal␣
↪relationships in complex data},
year = {2020},
month = {March},
volume = {16},
url = {https://doi.org/10.1371/journal.pgen.1008198},
pages = {1-35},
number = {3},
doi = {10.1371/journal.pgen.1008198}}
```

1.1 Introduction

BayesNetty is a C++ program written to perform Bayesian network analyses using genetic and phenotypic data.

BayesNetty is copyright, 2015-present Richard Howey, GNU General Public License, version 3.

The **recommended use** of BayesNetty is to calculate the average network as described in the Average Network section, possibly additionally using imputation to fill in missing data as descibed in Impute Data section. However, for new users we reccomend you work your way through all sections in numerical order, in order to understand the functionality of the program.

1.2 Installation

Download an executable file from the GitHub download page for your system and off you go, or do the following:

1. Download the code from the download page.
2. Compile it by typing something like the following:

```
g++ -m64 -O3 *.cpp -o bayesnetty
```

3. Start using BayesNetty!

1.3 Using BayesNetty

BayesNetty is executed as follows:

```
./bayesnetty paras.txt
```

where `paras.txt` is a parameter file as described in the following sections.

1.3.1 Basic Options

The basic options for BayesNetty are as follows (typing `./bayesnetty` with no options will output the available options):

tabularytabulary

Option	Description
-log file.log	log file of screen output
-so	suppress output to screen
-seed number	random number generator seed

Random Seed

The random seed option `-seed` may be used to ensure exactly the same output for testing and reproducibility purposes. If you do this it is important to use the same number of processes if also using the parallel version of BayesNetty.

1.3.2 Parameter file

There are many different things that BayesNetty can do, each of these different things is referred to as a "task". The parameter file defines which tasks BayesNetty will perform and the order in which they are executed. With the exception of the *basic options*, all options in the parameter file define tasks. For example, a task to input some continuous data may be given as follows:

```
-input-data
-input-data-name myGreatData
-input-data-file example-cts.dat
-input-data-cts
-input-data-ids 1
```

There are a few basic rules for parameter files:

1. Each option must be written on a separate line.
2. Each line that does not start with a dash, "-", will be ignored, thus allowing comments to be written.
3. The task must first be declared and then followed by any options for the task.
4. An option for a task is always written by first writing the name of the task. For example, the task option to give the name of an input data file is given by `-input-data-file`, which begins with `-input-data`, the name of the task to input data.
5. A task, `XXX`, may always be given a task name with the option `XXX-name`. The task may then be referenced by other tasks (if permitted). This may be useful if there is more than one network.
6. Tasks are executed in order, so any tasks that depend on other tasks must be ordered accordingly.
7. Any tasks that require a network will be default use the previously defined network. Therefore, if there is only one network it is not necessary to name or reference it. By default tasks are name "Task-n", where n is the number of the task.

The following is an extract from an example parameter file where a network is referenced by a task to calculate the network score:

```
...

# This is my comment
-input-network
-input-network-name myNetwork
-input-network-file network-model.dat

-calc-network-score
-calc-network-score-network-name myNetwork
```

The parameter file could be thought of as in an `R` programming style, such that the above would look as follows:

```
...

# This is my comment
```

(continues on next page)

(continued from previous page)

```
myNetwork<-input.network(file = "network-model.dat")

calc.network.score(network.name = myNetwork)
```

However, as BayesNetty is not an R package (or a programming language), the parameter file uses an unambiguous, longhand, and easy to parse style of syntax.

The options for all the different tasks may be found in the different task sections of the documentation.

1.3.3 Simple Example

Example data and parameter files can be found in the file example.zip. The example parameter file, `paras-example.txt`, can be used to perform a simple analysis by typing

```
./bayesnetty paras-example.txt
```

The following shows the `paras-example.txt` file

```
#input continuous data
-input-data
-input-data-file example-cts.dat
-input-data-cts

#input discrete data
-input-data
-input-data-file example-discrete.dat
-input-data-discrete

#input SNP data as discrete data
-input-data
-input-data-file example.bed
-input-data-discrete-snp

#search network models
-search-models
-search-models-file search-example.dat
```

The parameter file instructs BayesNetty to perform 4 tasks: (i) load continuous data from file `example-cts.dat`; (ii) load discrete data from file `example-discrete.dat`; (iii) load SNP data to be treated as discrete data from file `example.bed`; and finally (iv) search the network models. The screen output, which is also saved to a log file, will look something as follows:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551700145
--------------------------------------------------
Task name: Task-1
Loading data
Continuous data file: example-cts.dat
```

(continues on next page)

(continued from previous page)

```
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
Missing value: not set
--------------------------------------------------
--------------------------------------------------
Task name: Task-2
Loading data
Discrete data file: example-discrete.dat
Number of ID columns: 2
Including the 1 and only variable in analysis
Each variable has 1500 data entries
Missing value: NA
--------------------------------------------------
--------------------------------------------------
Task name: Task-3
Loading data
SNP binary data file: example.bed
SNP data treated as discrete data
Total number of SNPs: 2
Total number of subjects: 1500
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
--------------------------------------------------
--------------------------------------------------
Task name: Task-4
Searching network models
--------------------------------------------------
Loading defaultNetwork network
Network type: bnlearn
Network score type: BIC
Total number of nodes: 5 (Discrete: 3 | Factor: 0 | Continuous: 2)
Total number of edges: 0
Network Structure: [express][pheno][mood][rs1][rs2]
Total data at each node: 1495
Missing data at each node: 5
--------------------------------------------------
Network: defaultNetwork
Search: Greedy
Random restarts: 0
Random jitter restarts: 0
Network Structure: [mood][rs1][rs2][express|rs1:rs2][pheno|express:mood]
Network score type: BIC
Network score = -8213.45
Network search output to file: search-example.dat
--------------------------------------------------

Run time: less than one second
```

1.3.4 Command-line Options

It is also possible to add options on the command line to modify or add to the options in the parameter file. For example

```
./bayesnetty paras-example.txt -seed 1 -log seed-1-results.log
```

1.4 Parallel BayesNetty

It is possible to run BayesNetty using Open MPI (*cite* openmpi */cite*), which is an open source Message Passing Interface (MPI) (*cite* mpi */cite*) implementation designed for parallel programming.

The parallel version of Bayesnetty speeds up the search through network space for the best network by simultaneously evaluating different networks. This is particularly useful for large networks and any type of analysis that depends on network searches, such as network averaging and imputing data.

A much faster way to calculate an average network in parallel is given in the Average Network section.

A much faster way to impute network data in parallel is given in the Impute Data section.

After installing Open MPI on your system if it is not already installed, see *cite* openmpi */cite*, a parallel version of BayesNetty needs to be compiled. This can be done by firstly uncommenting a few lines in the `main.h` file. So that the following:

```
// Comment out if not using Open MPI for parallel processing
//#ifndef USING_OPEN_MPI
//#define USING_OPEN_MPI
//#endif //OPEN_MPI
```

becomes

```
// Comment out if not using Open MPI for parallel processing
#ifndef USING_OPEN_MPI
#define USING_OPEN_MPI
#endif //OPEN_MPI
```

then compile the parallel code as follows:

```
mpicxx -O3 -o pbayesnetty *.cpp
```

The code can then be ran using how many processes that you wish, for example to run with 12 processes use

```
mpirun -n 12 ./pbayesnetty paras-example.txt
```

For such a trivial example the code will not run any quicker, and in fact for very small networks one may find that analyses take longer. There is some overhead in using the MPI libraries, so if trivial networks are used there may be no speed up.

Even for large networks the optimal number of processes to perform the analysis as quick as possible may not be as many processes as you can use. As there is an overhead for processes the best amount to use may be a lot, but not too many… The best amount will vary depending on the analysis, the data and the computing system that you are using, so some trial and error may be needed.

The output will show the number of processes as well as the random seed. If you wish to reproduce exactly the same results both of these need to be set to the same value. The seed is set with the `-seed` option.

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Number of processes: 12
Random seed: 1541430503
--------------------------------------------------
Task name: Task-1
Loading data
Continuous data file: example-cts.dat
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
Missing value: not set
--------------------------------------------------
--------------------------------------------------


...
```

1.4.1 Compilation Scripts

Scripts to compile Bayesnetty as either parallel or non-parallel while automatically uncommenting or commenting the code as appropriate are given below.

Script to compile code in parallel:

```
sed -i s://#ifndef\ USING_OPEN_MPI:#ifndef\ USING_OPEN_MPI:g main.h
sed -i s://#define\ USING_OPEN_MPI:#define\ USING_OPEN_MPI:g main.h
sed -i s://#endif\ //:#endif\ //:g main.h

mpicxx -O3 -o pbayesnetty *.cpp
```

Script to compile code in non-parallel:

```
sed -i s://#ifndef\ USING_OPEN_MPI:#ifndef\ USING_OPEN_MPI:g main.h
sed -i s://#define\ USING_OPEN_MPI:#define\ USING_OPEN_MPI:g main.h
sed -i s://#endif\ //:#endif\ //:g main.h

sed -i s:#ifndef\ USING_OPEN_MPI://#ifndef\ USING_OPEN_MPI:g main.h
sed -i s:#define\ USING_OPEN_MPI://#define\ USING_OPEN_MPI:g main.h
sed -i s:#endif\ //://#endif\ //:g main.h

g++ -O3 *.cpp -o bayesnetty
```

1.5 Input data

All data must be input using the `-input-data` task.

1.5.1 Options

The options are as follows:

tabularytabulary

	Option		
	-input-data		
	-input-data-name name		
	-impute-network-data-network-name network		
	-input-data-include-file nodes.dat		a lis
	-input-data-exclude-file nodes.dat		
	-input-data-cts		
	-input-data-cts-snp		
	-input-data-cts-snp2		
	-input-data-cts-missing-value x		
	-input-data-discrete		
	-input-data-discrete-snp		
	-input-data-discrete-snp2		
	-input-data-discrete-missing-value x		
	-input-data-factor		
	-input-data-factor-snp		
	-input-data-factor-snp2		
	-input-data-factor-missing-value x		
	-input-data-ids n		
	-input-data-csv		

1.5.2 Discrete data

Discrete data is automatically constrained to have no parent nodes that are continuous.

Discrete data is input by using the `-input-data` task, setting the data file and setting the data file to discrete. For example, the following

```
-input-data
-input-data-file example-discrete.dat
-input-data-discrete
```

could be used and the output would be something as follows:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551695824
--------------------------------------------------
Task name: Task-1
```

(continues on next page)

(continued from previous page)

```
Loading data
Discrete data file: example-discrete.dat
Number of ID columns: 2
Including the 1 and only variable in analysis
Each variable has 1500 data entries
Missing value: NA
--------------------------------------------------

Run time: less than one second
```

This parameter file can be found `paras-input-discrete.txt` in the examples, example.zip.

1.5.3 Continuous data

Continuous data is input by using the `-input-data` task, setting the data file and setting the data file to continuous. For example, the following

```
-input-data
-input-data-file example-cts.dat
-input-data-cts
```

could be used and the output would be something as follows:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551695897
--------------------------------------------------
Task name: Task-1
Loading data
Continuous data file: example-cts.dat
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
Missing value: not set
--------------------------------------------------

Run time: less than one second
```

This parameter file can be found `paras-input-cts.txt` in the examples, example.zip.

1.5.4 Factor data

Another way of handling discrete data is with the use of *factors*. Indicator variables are created, one for each different discrete category minus one. These are treated as continuous explanatory variables in the linear regressions when they are parent nodes. A restriction of using discrete data with factors is that they cannot be child nodes of other nodes. Input by using the `-input-data` task, setting the data file and setting the data file to factor. For example, the following

```
-input-data
-input-data-file example-discrete.dat
-input-data-factor
```

could be used and the output would be something as follows:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551695930
--------------------------------------------------
Task name: Task-1
Loading data
Discrete data file: example-discrete.dat
Data treated as factors
Number of ID columns: 2
Including the 1 and only variable in analysis
Each variable has 1500 data entries
Missing value: NA
--------------------------------------------------

Run time: less than one second
```

This parameter file, `paras-input-factor.txt`, can be found in the examples, example.zip.

1.5.5 SNP data

SNP data is automatically constrained to have no parent nodes.

SNP data may be input as a binary PLINK format pedigree file, a `.bed` file, see [3] . This requires that the corresponding `.bim` and `.fam`, files are also available. A text PLINK pedigree file, `.ped`, with corresponding map file, `.map`, may be used to create a binary file using PLINK as follows:

```
plink --noweb --file mydata --make-bed --out myfile
```

This will create the binary pedigree file, `myfile.bed`, map file, `myfile.bim`, and family file, `myfile.fam` required.

The SNP data is input by using the `-input-data` task, setting the PLINK binary file and setting the data file to a SNP file in discrete mode or continuous mode. For example, in discrete mode, the following

```
-input-data
-input-data-file example.bed
-input-data-discrete-snp
```

could be used and the output would be something as follows:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551695984
--------------------------------------------------
Task name: Task-1
Loading data
SNP binary data file: example.bed
SNP data treated as discrete data
Total number of SNPs: 2
Total number of subjects: 1500
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
--------------------------------------------------

Run time: less than one second
```

This parameter file can be found `paras-input-snp.txt` in the examples, example.zip.

1.5.6 Missing data

Missing data is determined by any data matching the given missing value as defined by `-input-data-discrete-missing-value` and `-input-data-cts-missing-value` when inputting discrete and continuous data respectively (or `-input-data-factor-missing-value` when inputting factor data). When continuous data has an invalid entry this will also be set to missing, for example a value of "NaN" will be set to missing since a numerical value is required. Missing data for SNP data is given as defined by the PLINK binary pedigree format. When there is missing data for a node for a certain individual then data for this certain individual is considered as missing for *every* node in the network. Therefore the amount of missing data depends on which nodes are in the network.

Consider a network with 2 continuous nodes, with structure as given by network file `example-network-missing1.dat` and input using parameter file `paras-input-missing1.txt` as given in the file example.zip, then the output will be will look something as follows:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551694585
--------------------------------------------------
Task name: Task-1
Loading data
Continuous data file: example-cts.dat
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
Missing value: not set
--------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------
Task name: myNetwork
Loading network
Network file: example-network-missing1.dat
Network type: bnlearn
Network score type: BIC
Total number of nodes: 2 (Discrete: 0 | Factor: 0 | Continuous: 2)
Total number of edges: 1
Network Structure: [express][pheno|express]
Total data at each node: 1500
Missing data at each node: 0
--------------------------------------------------

Run time: less than one second
```

As indicated in the network details there is no missing data. However, if the SNP node, `rs1`, is added (network file `example-network-missing2.dat`) then the following is given:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551696539
--------------------------------------------------
Task name: Task-1
Loading data
SNP binary data file: example.bed
SNP data treated as discrete data
Total number of SNPs: 2
Total number of subjects: 1500
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
--------------------------------------------------
--------------------------------------------------
Task name: Task-2
Loading data
Continuous data file: example-cts.dat
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
Missing value: not set
--------------------------------------------------
--------------------------------------------------
Task name: myNetwork
Loading network
Network file: example-network-missing2.dat
Network type: bnlearn
Network score type: BIC
Total number of nodes: 3 (Discrete: 1 | Factor: 0 | Continuous: 2)
Total number of edges: 2
```

(continues on next page)

(continued from previous page)

```
Network Structure: [rs1][express|rs1][pheno|express]
Total data at each node: 1497
Missing data at each node: 3
--------------------------------------------------

Run time: less than one second
```

This example is given in network file `example-network-missing2.dat``and parameter file ``paras-input-missing2.txt`. The amount of missing data for the network is now 3, indicating that 3 individuals have missing SNP data for `rs1`. Adding in another SNP node, `rs2` (network file `example-network-missing3.dat`), results in the following:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551696644
--------------------------------------------------
Task name: Task-1
Loading data
SNP binary data file: example.bed
SNP data treated as discrete data
Total number of SNPs: 2
Total number of subjects: 1500
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
--------------------------------------------------
--------------------------------------------------
Task name: Task-2
Loading data
Continuous data file: example-cts.dat
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
Missing value: not set
--------------------------------------------------
--------------------------------------------------
Task name: myNetwork
Loading network
Network file: example-network-missing3.dat
Network type: bnlearn
Network score type: BIC
Total number of nodes: 4 (Discrete: 2 | Factor: 0 | Continuous: 2)
Total number of edges: 3
Network Structure: [rs1][rs2][express|rs1:rs2][pheno|express]
Total data at each node: 1495
Missing data at each node: 5
--------------------------------------------------

Run time: less than one second
```

Similarly, this example is given in network file `example-network-missing3.dat` and parameter file `paras-input-missing3.txt`. Here we see that the amount of missing data in the network has increased due to missing data for SNP node `rs2`. This node also has missing data for 3 individuals, with the result that the total amount of missing data for each node is 5.

1.5.7 Data IDs

By default the first two columns of a data file should be IDs and match those in any other data files, and be in the same order (although the ID names in the header do not need to match). The number of ID columns can be changed using the `-input-data-ids` option, and may be set to zero. If the data contains SNP data in a PLINK binary pedigree file, `.bed`, then the number of ID columns must be set to 2. If the data is a binary pedigree file, `file.bed`, then the family and individual IDs in the file `file.fam` must match the IDs in any other data files, and all SNPs may be used as network nodes. The IDs in different files are checked to be the same including the order, if not BayesNetty will report an error. If there are zero IDs then the individuals are assumed to be in the same order in each file and are only checked to have the same number of individuals.

1.5.8 Example

See the above sections for examples of inputting data.

1.6 Input network

A network may be specified using the `-input-network` task. The network may be used as a starting point for analyses, such as searches, or to perform an analysis on this network. Only nodes in input files will be used in the network so that a subset of the data may be specified.

Any network **constraints** must be set using the `-input-network` option. These constraints then belong to the network and will be used in any subsquent analysis, including searches, calculating average networks etc.

If no network is specified then a network with no edges and a node for every data variable (as given by the input data) will be created and named “defaultNetwork”.

1.6.1 Options

The options are as follows:

tabularytabulary

Option
-input-network
-input-network-name name
-input-network-type t
-input-network-file network.dat
-input-network-file2 network2.dat
-input-network-igraph-file-prefix mygraph
-input-network-empty
-input-network-whitelist-file whitelist.dat
-input-network-blacklist-file blacklist.dat
-input-network-blacklist-edge-type dataName1 dataName2
-input-network-no-parents-node nodeX
-input-network-no-children-node nodeY
-input-network-prob-edge node1 node2 prob
-input-network-prob-edge-type nodeType1 nodeType2 prob
-input-network-imaginary-sample-size i
-input-network-score score

1.6.2 Black lists

A black list can be given using the `-input-network-blacklist-file` option to define a list of edges that must not be included in any network. The text file should be formatted as follows:

```
node1 node2
node2 node1
node1 node3
```

such that the two nodes of each blacklisted edge are on one line. The nodes are ordered so the first line states that the edge node1 to node2 is not permitted. The next line states that the edge in the reverse direction is also not permitted.

Any searches will ignore these blacklisted edges and attempting to use a network with a blacklisted edge will result in the edge being removed.

Edges between different types of nodes may also be blacklisted. This can be done using the `-input-network-blacklist-edge-type` option. It can be used as follows:

```
-input-data
-input-data-name horses
-input-data-file horses.dat
-input-data-cts

-input-data
-input-data-name whips
-input-data-file whips.dat
-input-data-cts

-input-network
-input-network-name race
```

(continues on next page)

(continued from previous page)

```
-input-network-file model.dat
-input-network-blacklist-edge-type horses whips
```

Firstly the different node types must be loaded separately and given names using the `-input-data-name` option. Then, when initially loading a network, the `-input-network-blacklist-edge-type` can be used to forbid any edge from one data set to another data set (or the same data if desired). In the above example the network may not have any edge that goes from a horse to a whip, that is, a whip node may not have a horse node as a parent. In any search that is performed these edges will not be considered.

1.6.3 White lists

A white list can be given using the `-input-network-whitelist-file` option to define a list of edges that must be included in any network. The text file should be formatted as follows:

```
node1 node3
node1 node2
node2 node1
```

such that the two nodes of each whitelisted edge are on one line. The nodes are ordered so the first line states that the edge node1 to node3 must be included. If both directions are included between two nodes then the edge must be included but may be in any direction.

If the whitelist and blacklist contradict one another then an error will be given.

1.6.4 Soft Constraints

Soft constraints provide a way that the direction of an edge may be influenced but not with certainty, unlike blacklisted edges or whitelisted edges as described above. An example parameter file setting a soft constraint, such that the prior probability of variable `express` to variable `pheno` is believed to be 0.8 is shown below.

```
#input continuous data
-input-data
-input-data-file example-cts.dat
-input-data-cts

#input discrete data
-input-data
-input-data-file example-discrete.dat
-input-data-discrete

#input SNP data as discrete data
-input-data
-input-data-file example.bed
-input-data-discrete-snp

#input the example network in format 1
-input-network
-input-network-name myNetwork
-input-network-file example-network-format1.dat
-input-network-prob-edge express pheno 0.8
```

(continues on next page)

(continued from previous page)

```
#search network models with the soft constraint
-search-models
```

This parameter file, `paras-soft-constraints.txt`, can be found in the file example.zip.

Any searches will use this prior probability.

If you wish to blacklist or whitelist an edge you should use those options rather than setting the prior probability to 0 or 1 for the sake of computational efficiency.

1.6.5 Network formats

The network may be defined using one of 3 different formats.

Network file format 1

The first format is given by using the `-input-network-file` option and the network text file should be formatted as follows:

```
node1
node2
node3
node2 node1
node3 node1
```

where the nodes are listed first followed by the directed edges. In the above example there are 3 nodes and 2 edges, which are node2 to node1 and node3 to node1.

Network file format 2

The second format is given by using the `-input-network-file2` option and the network text file should be formatted as follows:

```
[node2][node3][node1|node2:node3]
```

where the nodes are listed in order of dependency. The independent nodes node2 and node3 are list first followed by node1 which is a child node of both node2 and node3. This is the format that is typically output for searches and such like.

Network file format 3

The third format is given by using the `-input-network-igraph-file-prefix` option using the files that were output to draw the network in `R`, see igraph. There will be one file for the nodes and one for the edges, for example `myNetwork-nodes.dat` and `myNetwork-edges.dat` respectively. The node file will look something as follows:

```
id name type fileno
1 node1 c 1
2 node2 c 1
3 node3 c 1
```

and the edges file will look like something as follows:

```
from to chisq
2 1 6860.83
3 1 5709.51
```

1.6.6 Example

The following is an example parameter file to input a network.

```
#input continuous data
-input-data
-input-data-file example-cts.dat
-input-data-cts

#input discrete data
-input-data
-input-data-file example-discrete.dat
-input-data-discrete

#input SNP data as discrete data
-input-data
-input-data-file example.bed
-input-data-discrete-snp

#input the example network in format 1
-input-network
-input-network-name myNetwork
-input-network-file example-network-format1.dat
```

This parameter file, `paras-input-network.txt`, can be found in the file example.zip and can be used as follows:

```
./bayesnetty paras-input-network.txt
```

Which should produce output that looks like something as follows:

```
BayesNetty: Bayesian Network software, v1.00
--------------------------------------------------
Copyright 2015-present Richard Howey, GNU General Public License, v3
Institute of Genetic Medicine, Newcastle University

Random seed: 1551697141
--------------------------------------------------
Task name: Task-1
Loading data
Continuous data file: example-cts.dat
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
Missing value: not set
--------------------------------------------------
--------------------------------------------------
Task name: Task-2
Loading data
```

(continues on next page)

(continued from previous page)

```
Discrete data file: example-discrete.dat
Number of ID columns: 2
Including the 1 and only variable in analysis
Each variable has 1500 data entries
Missing value: NA
--------------------------------------------------
--------------------------------------------------
Task name: Task-3
Loading data
SNP binary data file: example.bed
SNP data treated as discrete data
Total number of SNPs: 2
Total number of subjects: 1500
Number of ID columns: 2
Including (all) 2 variables in analysis
Each variable has 1500 data entries
--------------------------------------------------
--------------------------------------------------
Task name: myNetwork
Loading network
Network file: example-network-format1.dat
Network type: bnlearn
Network score type: BIC
Total number of nodes: 5 (Discrete: 3 | Factor: 0 | Continuous: 2)
Total number of edges: 4
Network Structure: [mood][rs1][rs2][pheno|rs1:rs2][express|pheno:mood]
Total data at each node: 1495
Missing data at each node: 5
--------------------------------------------------

Run time: 1 second
```

The data is loaded and then the network is loaded. The network has been named “myNetwork”, and basic information about the network is output.

Similarly, the network may be input using format 2 and 3 as given in parameter files `paras-input-network2.txt` and `paras-input-network3.txt` respectively.

1.7 References

CHAPTER

TWO

CONTACT

Please contact Richard Howey with any queries about the BayesNetty software.

BIBLIOGRAPHY

[1] R Howey, A D Clark, N Naamane, L N Reynard, A G Pratt, and H J Cordell. A Bayesian network approach incorporating imputation of missing data enables exploratory analysis of complex causal biological relationships. *PLOS Genetics*, 17(9):e1009811, September 2021. URL: https://doi.org/10.1371/journal.pgen.1009811, doi:10.1371/journal.pgen.1009811.

[2] R Howey, S-Y Shin, C Relton, G Davey Smith, and H J Cordell. Bayesian network analysis incorporating genetic anchors complements conventional mendelian randomization approaches for exploratory analysis of causal relationships in complex data. *PLOS Genetics*, 16(3):1–35, 03 2020. URL: https://doi.org/10.1371/journal.pgen.1008198, doi:10.1371/journal.pgen.1008198.

[3] S Purcell, B Neale, K Todd-Brown, L Thomas, M A Ferreira, D Bender, J Maller, P Sklar, P I de Bakker, M J Daly, and P C Sham. PLINK: a tool set for whole-genome association and population-based linkage analyses. *Am J Hum Genet*, 81:559–575, 2007.