

CausalFS toolbox/package Manual

Public domain version 1.0 for C/C++(Ubuntu)

Release date: 10/30/2019

(Version 1.0)

Copyright © 2019, All rights reserved,

---

#### **References for citation:**

- Kui Yu, Xianjie Guo, Lin Liu, Jiuyong Li, Hao Wang, Zhaolong Ling, and Xindong Wu. Causality-based Feature Selection: Methods and Evaluations. arXiv: Artificial Intelligence, 2019.
- Kui Yu, Lin Liu, and Jiuyong Li. A Unified View of Causal and Non-causal Feature Selection. arXiv: Artificial Intelligence 2018.

#### **Authors:**

- Xianjie Guo ([Xianjie\\_reasonless@163.com](mailto:Xianjie_reasonless@163.com))
- Kui Yu ([ykui713@gmail.com](mailto:ykui713@gmail.com))

Hefei University of Technology  
485 Danxia Road, Shushan District, Hefei, 230601, China

## Table of Contents

<b>I. Toolbox overview .....</b>	<b>1</b>
I.1. Introduction .....	1
I.2. Architecture of toolbox .....	2
<b>II. Environment configuration.....</b>	<b>3</b>
II.1. Description .....	3
II.2. Configuring the boost library .....	3
II.3. Build executable file.....	3
<b>III. Constraint-based algorithms.....</b>	<b>6</b>
III.1. Description.....	6
III.2. Inputs and Outputs .....	6
A. MBtoPC algorithm.....	11
B. PC_simple, MMPC, GetPC, HITON_PC and Semi_HITON_PC algorithms .....	11
C. GSMB, IAMB, Inter_IAMB, LRH, FBED and MBOR algorithms.....	12
D. BAMB, EEMB, MMBB, PCMB, HITON_MB, Semi_HITON_MB, IPCMB, STMB and CCMB algorithms .....	13
E. KIAMB algorithm.....	15
F. Fast_IAMB algorithm .....	15
G. TIE* algorithm.....	16
H. MMBB-CSL, PCMB-CSL, HITON_MB-CSL, Semi_HITON_MB-CSL, IPC_MB-CSL, STMB-CSL, MMBB-SSL, PCMB-SSL, HITON_MB-SSL, Semi_HITON_MB-SSL, IPCMB-SSL and STMB-SSL algorithms .....	16
I. MBOR-CSL and MBOR-SSL algorithms .....	16
<b>IV. Score-based algorithms .....</b>	<b>17</b>
IV.1. Description .....	17
IV.2. Inputs and Outputs .....	17
A. SLL-PC and S2TMB-PC algorithms.....	20
B. SLL and S2TMB algorithms .....	21
C. SLL-CSL, S2TMB-CSL, SLL-SSL and S2TMB-SSL algorithms.....	21
<b>V. Script run experiment .....</b>	<b>21</b>
V.1. Description .....	21
V.2. “.sh” script basic syntax .....	22
A. The beginning .....	22
B. The notes .....	22
C. Common shell commands .....	22
V.3. Running experiment .....	23

## I. Toolbox overview

### I.1. Introduction

In the CausalFS toolbox, all algorithms and files are located at the “CausalFS” folder. The hierarchy of the CausalFS folder is as shown in Figure 1. The “CausalFS” folder consists of three subfolders: CCD [Constraint-based Markov Blanket or Parents and Children (MB/PC) learning algorithms for continuous data using the Fisher Z-test)], CDD [Constraint-based MB/PC learning algorithms for discrete data using the  $G^2$  test)] and SDD [Score-based MB/PC learning algorithms (only support discrete data)]. In addition, users can also do local-to-global Bayesian network structure learning tasks using these MB/PC algorithms.

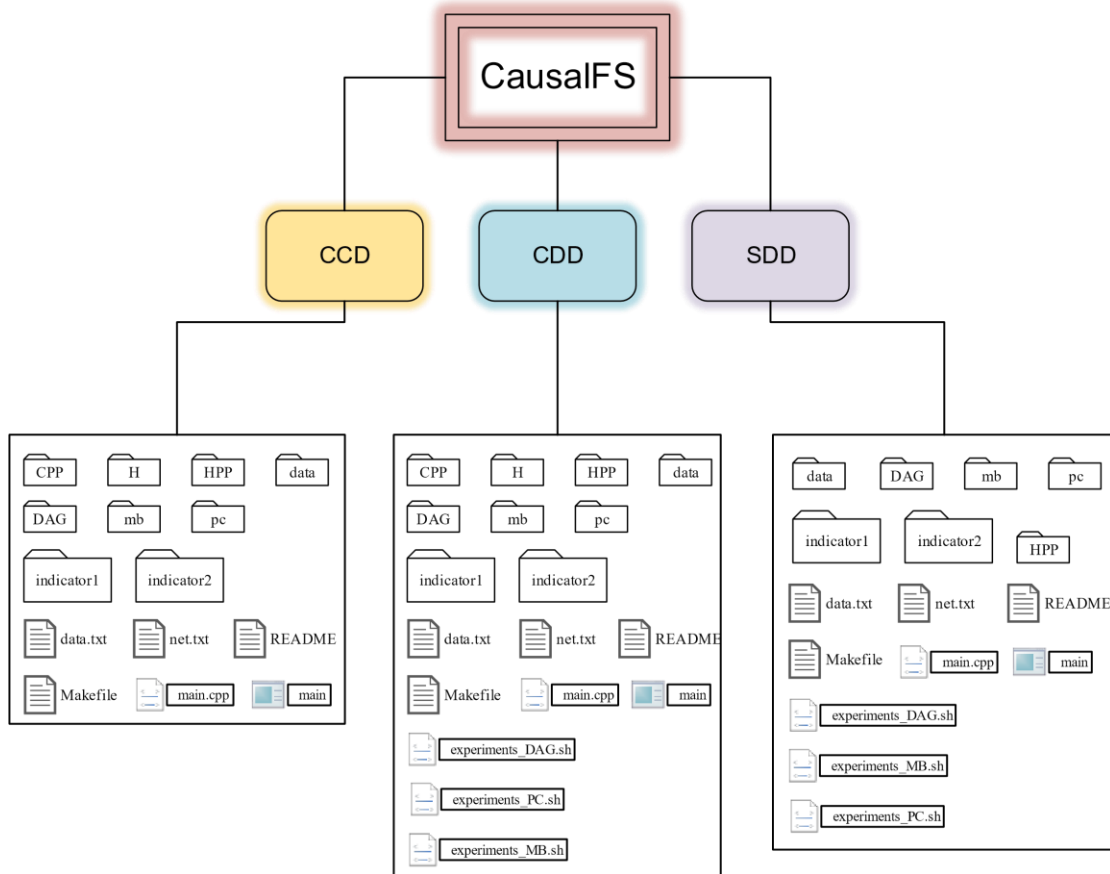


Figure 1 Hierarchy of the CausalFS folder in the CausalFS toolbox

In Figure 1, the \*.cpp source file, \*.h header file, and \*.hpp header file are kept in the *CPP*, *H*, and *HPP* folders respectively, and some benchmark BN data sets for example running are stored at the *data* folder. The *DAG* folder stores the DAG adjacency matrix learned by the structure learning algorithms. The *mb* folder keeps the MB of a target node learned by the MB learning algorithm, and the *pc* folder stores the

PC of the target node learned by the PC learning algorithms. The evaluation results of the PC or MB learning algorithms are kept in the *indicator1* folder, and the evaluation results of the structure learning algorithms are stored in the *indicator2* folder.

“data.txt” is the training data set for testing the MB, PC and structure learning algorithms, while “net.txt” is the corresponding true DAG adjacency matrix of a benchmark BN regarding the data set for validating the outputs returned by the MB, PC and structure learning algorithms implemented on the data set.

“main.cpp” is the main source file which includes all source codes of the MB and PC learning algorithms in the CausalFS toolbox, and “main” is the executable application. “\*.sh” is the script file used to run the experiment. The boost library environment **MUST** be configured before running the package.

## I.2. Architecture of toolbox

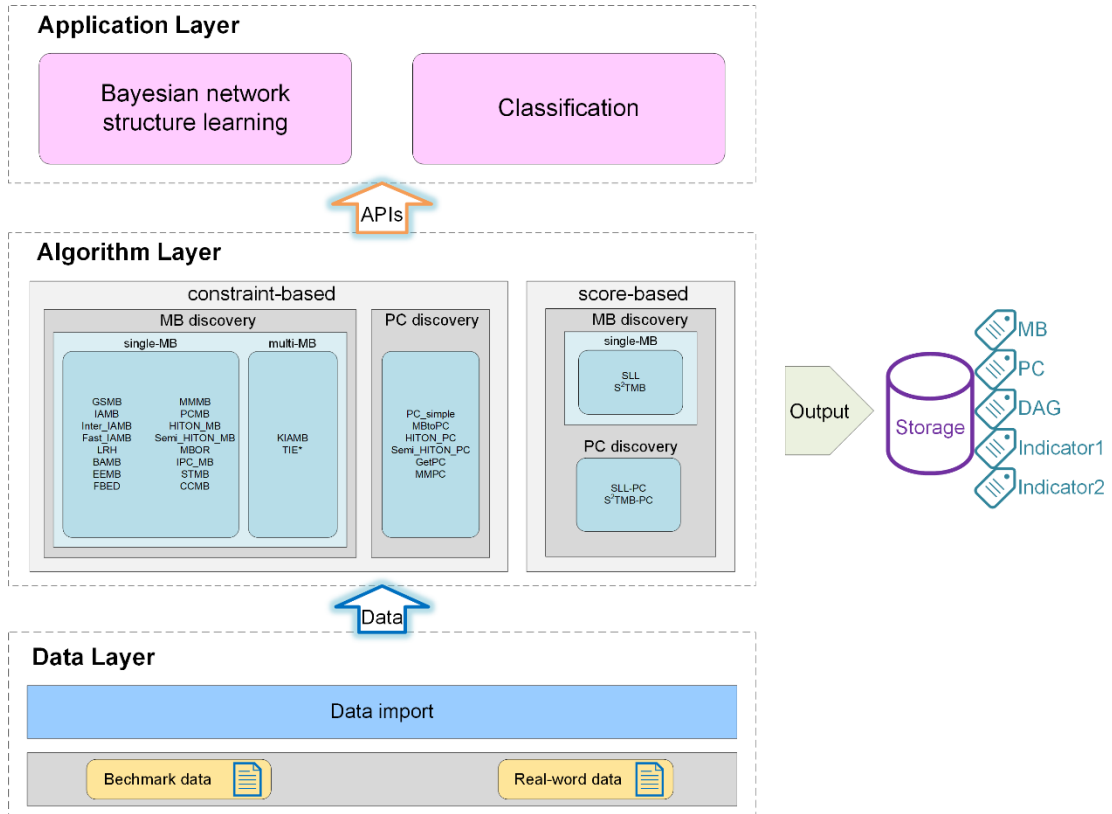


Figure 2 Architecture of the CausalFS toolbox/package

As shown in Figure 2, we can see that the CausalFS architecture is based on three modules, that is, data layer, algorithm layer, and application layer. The three modules are designed independently. This makes that CausalFS is simple, easy to implement, and extendable flexibly. One can easily add a new algorithm to the CausalFS toolbox and share it through the CausalFS framework without modifying the other modules.

In the algorithm layer, CausalFS implements 28 representative causality-based feature selection methods. Specifically, it consists of 24 methods using conditional independence tests (i.e., 16 single MB learning algorithms, 2 multiple MB learning

algorithms, and 6 PC learning algorithms), and 4 score-based approaches. Furthermore, by applying the MB and PC discovery algorithms in the algorithm layer, using the CausalFS toolbox, users can easily generate different local-to-global structure learning methods. Then the CausalFS toolbox not only supports feature selection for supervised classification, but also is able to do local-to-global BN structure learning in the application layer.

## II. Environment configuration

### II.1. Description

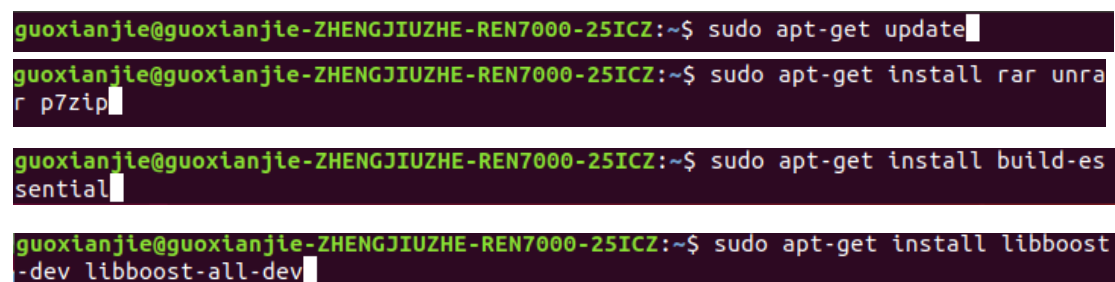
This package runs under the **Ubuntu** system and to configure the **boost library** environment. Without the boost library environment configured, none of the projects in the toolkit will be able to build executable files, and the executable files cannot work properly.

### II.2. Configuring the boost library

Press **CTRL+ALT+T** to open the terminal, then run the following four commands in sequence as shown as follows.

1. `sudo apt-get update`
2. `sudo apt-get install rar unrar p7zip`
3. `sudo apt-get install build-essential`
4. `sudo apt-get install libboost-dev libboost-all-dev`

The configuring examples are shown in Figure 3 as follows:



```
guoxianjie@guoxianjie-ZHENGJIUZHENG-REN7000-25ICZ:~$ sudo apt-get update
guoxianjie@guoxianjie-ZHENGJIUZHENG-REN7000-25ICZ:~$ sudo apt-get install rar unrar p7zip
guoxianjie@guoxianjie-ZHENGJIUZHENG-REN7000-25ICZ:~$ sudo apt-get install build-essential
guoxianjie@guoxianjie-ZHENGJIUZHENG-REN7000-25ICZ:~$ sudo apt-get install libboost-dev libboost-all-dev
```

Figure 3 Examples of configuring the boost library environment

### II.3. Build executable file

After all the environment configurations are completed, users click the right mouse

button in each project folder to open the terminal. Enter the "**make veryclean**" command to clear the original executable file, \*.o file and \*.d file in the project as shown in Figure 4 and Figure 5; Then enter the "**make**" command to build new executable file. Running examples are shown in Figure 6 and Figure 7 as follows.

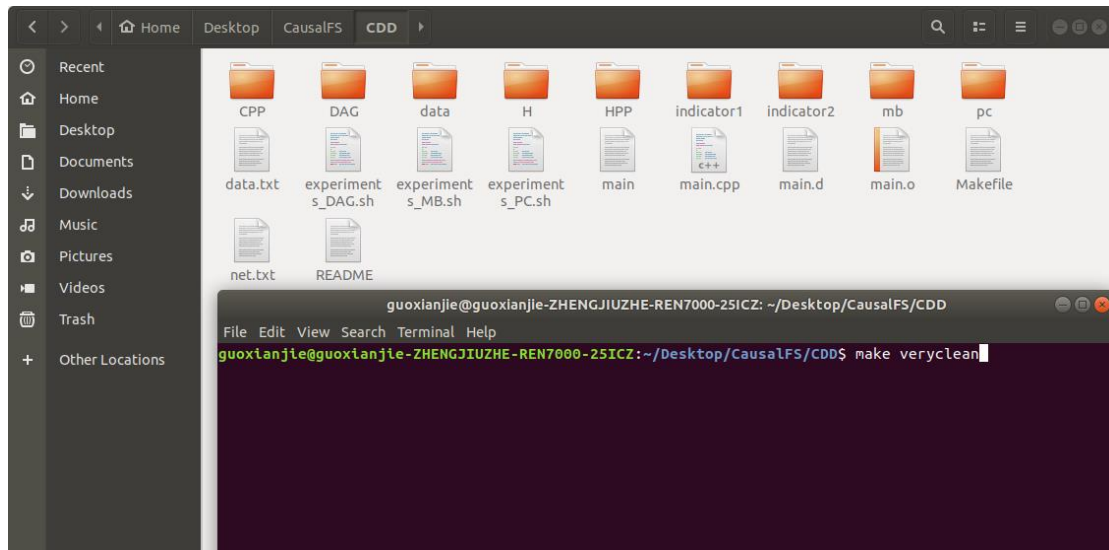


Figure 4 Before the "make veryclean" command running

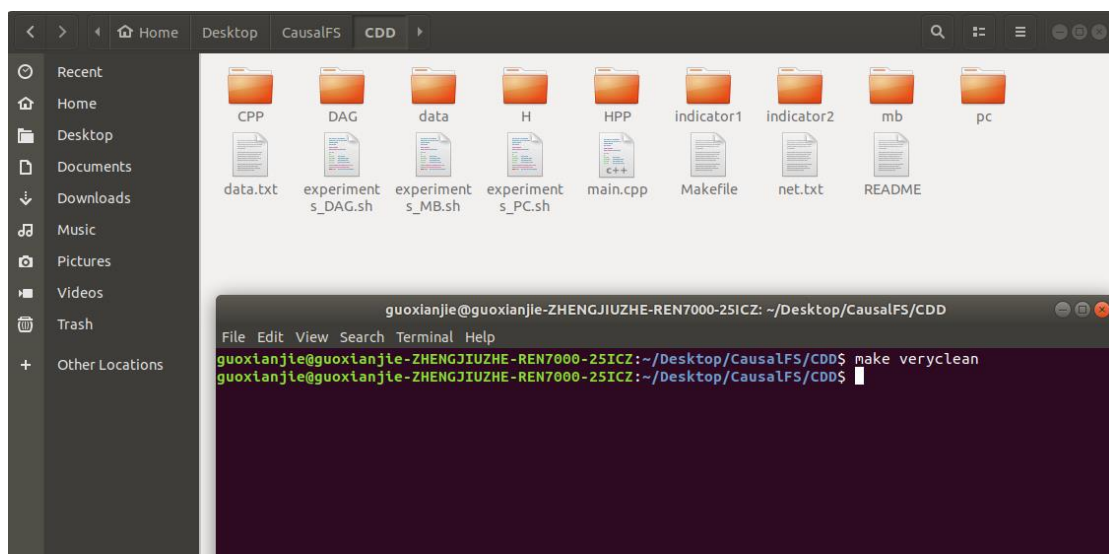


Figure 5 After the "make veryclean" command running

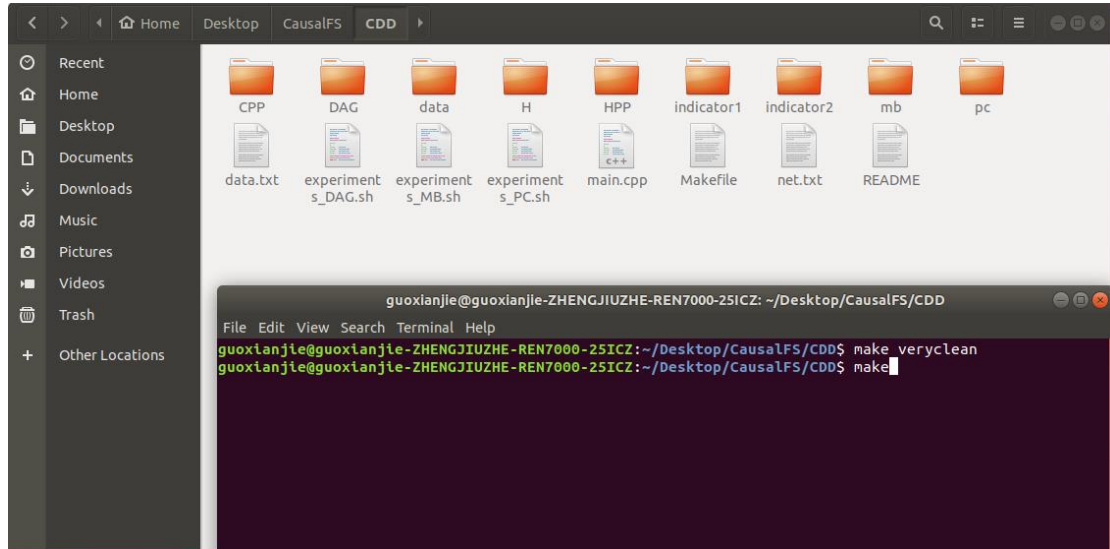


Figure 6 Before the "make" command running

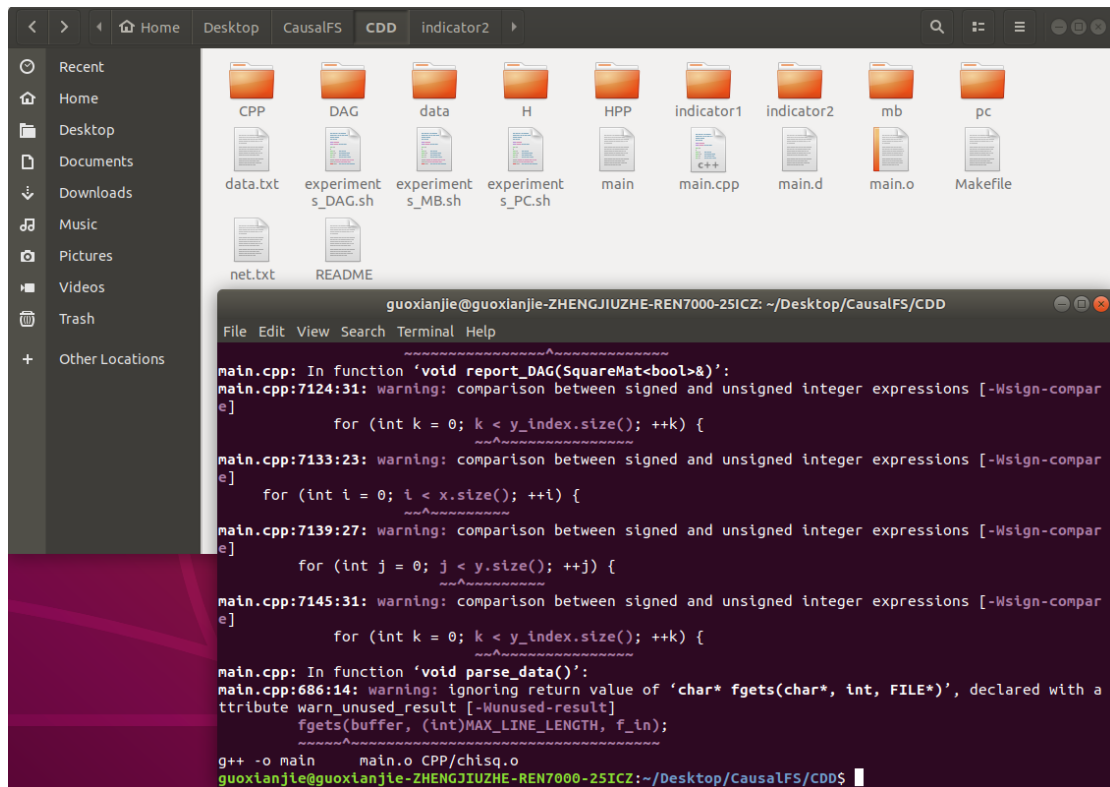


Figure 7 After the "make" command running

### III. Constraint-based algorithms

#### III.1. Description

For the constraint-based algorithms, there are two folds, namely "CCD" and "CDD". The algorithms in the "CCD" fold are used to deal with continuous data, while the algorithms in the "CDD" fold are used to handle discrete data. The input and output parameter types and quantities are the same for all algorithms in both projects as shown in Table 1.

#### III.2. Inputs and Outputs of Algorithms

Table 1 Inputs and outputs of constraint-based algorithms

Inputs (eight parameters):	1 <sup>st</sup> input = executable file name
	2 <sup>nd</sup> input = dataset
	3 <sup>rd</sup> input = BN (DAG) adjacency matrix
	4 <sup>th</sup> input = significance level
	5 <sup>th</sup> input = algorithm name
	6 <sup>th</sup> input = target variable index
	7 <sup>th</sup> input = K (only for KIAMB for the tradeoff between greediness and randomness)
	8 <sup>th</sup> input = the maximum size of the conditioning set allowed
Outputs (two outputs):	1 <sup>st</sup> output = Index of PC or MB of each target node or DAG adjacency matrix learnt
	2 <sup>nd</sup> output = evaluation metric

#### Inputs (eight parameters):

In the following input parameters, when an input parameter is "", it means that the algorithm does not need to specify any values for this parameter. For example, the seventh parameter is only used by KIAMB, while for the other algorithms, it is set "". When doing structure learning, all nodes should be taken as the target node by default, that is, the value of the sixth parameter need not be specified. For another example, IAMB and its variants do not need to specify the eighth parameter.

1<sup>st</sup> input = executable file name

Executable files are generated by building a project through a Makefile, any



algorithm in the running project needs to call the executable file in the project. The executable file name generated by the project defaults to **main** (executable file). The user can modify the name of the executable file in the Makefile, and modify the first line of the Makefile:

EXECUTABLE: = **main** # executable file name

**2<sup>nd</sup> input** = dataset

The data set is used for training. Columns are variables and rows are observations. Support both continuous and discrete sample data sets. There are some benchmark BN data sets in the data folder for using.

**3<sup>rd</sup> input** = DAG (of BN) adjacency matrix

A matrix of n rows and n columns consisting of 0 and 1. (i, j) = 1 indicates that a directed edge is from node i to node j, that is, node i is a parent of node j.

**4<sup>th</sup> input** = significance level

The level of significance for hypothesis testing (e.g.,  $G^2$  test and Fisher Z-test), and often is set 0.01 or 0.05.

**5<sup>th</sup> input** = algorithm name

Users can use one of the following algorithm names in Tables 2 to 4 as follows as the 5<sup>th</sup> input parameter, such as MMPC, MMB, HITON\_MB-CSL, etc.

Table 2 PC learning algorithm

- **MBtoPC** - MBtoPC algorithm
- **PC\_simple** - PC-simple algorithm
- **MMPC** - Min-Max Parents and Children (MMPC) algorithm
- **GetPC** - GetPC algorithm
- **HITON\_PC** - HITON\_PC algorithm
- **Semi\_HITON\_PC** – Semi\_HITON\_PC algorithm

Table 3 MB learning algorithm

- **GSMB** - Grow/Shrink MB algorithm
- **IAMB** - Incremental Association-Based Markov Blanket (IAMB)
- **KIAMB** – KAIMB algorithm for multiple MB learning
- **Inter\_IAMB** – Inter-IAMB algorithm
- **Fast\_IAMB** – Fast-IAMB algorithm
- **LRH** – Lessen swamping, resist masking and highlight the true positives
- **BAMB** - Balanced Markov blanket discovery (BAMB) algorithm
- **EEMB** - Efficient and Effective MB algorithm
- **FBED** - Forward-Backward selection with Early Dropping algorithm
- **MMMB** – Min-Max Markov Blanket (MMMB) algorithm
- **PCMB** - Parents and children based MB (PCMB) algorithm
- **HITON\_MB** - HITON\_MB algorithm
- **Semi\_HITON\_MB** - Semi\_HITON\_MB algorithm
- **MBOR** - MB search using the OR condition
- **IPCMB** - Iterative Parent-Child based search of MB (IPCMB) algorithm
- **STMB** - Simultaneous MB discovery (STMB) algorithm
- **CCMB** - Cross-check and complement MB discovery (CCMB) algorithm
- **TIE\*** - Target Information Equivalence algorithm for multiple MB discovery

Table 4 Local-to-global Structure learning algorithm

- **MMMB-CSL**
- **PCMB-CSL**
- **HITON\_MB-CSL**
- **Semi\_HITON\_MB-CSL**
- **MBOR-CSL**
- **IPC\_MB-CSL**
- **STMB-CSL**

- **MMMB-SSL**
- **PCMB-SSL**
- **HITON\_MB-SSL**
- **Semi\_HITON\_MB-SSL**
- **MBOR-SSL**
- **IPCMB-SSL**
- **STMB-SSL**

In Table 4, **-CSL** indicates that a MB (or PC) is learned by a constraint-based MB method while orienting edge directions uses constraint-based rules. For example, MMMB-CSL means that it uses MMMB to learn MBs to construct a DAG skeleton and then employs constraint-based rules to orient edge directions in the learnt skeleton. **-SSL** means that the MB (or PC) is learned by a constraint-based MB method while orient edge directions using score-based methods (e.g., score-based hill climbing methods). In addition, **-SSL only supports discrete data sets**.

6<sup>th</sup> input = target variable index

If the number of nodes in a BN network is  $n$ , the index value of a node is among 0 to  $n-1$ . When users need to learn the MBs of all nodes in the network, the parameter is set to "all". When users need to learn the MBs of several nodes in the network, the specified target nodes need to be separated by ",". For example, "0,5,9" means an algorithm will return the MBs of nodes 0, 5, and 9.

7<sup>th</sup> input =  $K$  (tradeoff between greediness and randomness only for KIAMB)

The difference between KIAMB and IAMB is that KIAMB allows the user to specify the trade-off between greediness and randomness in the MB search through a randomization parameter  $K \in [0,1]$ . IAMB greedily adds to  $CMB(T)$  the variable with the highest association with  $T$  among all variables excluding  $CMB(T)$  currently selected, while KIAMB adds to  $CMB(T)$  the variables with the highest associations with  $T$  in the  $CanMB$  set which is a random subset of  $CMB(T)$  with size  $\max(1, \lfloor |CMB(T)| \cdot K \rfloor)$ .  $K$  specifies the trade-off between greediness and randomness in the MB search: if setting  $K = 1$ , KIAMB reduces to IAMB, while if taking  $K = 0$ , KIAMB is a completely random approach expected to discover all the MBs of  $T$  with a nonzero probability if running repeatedly for enough times.

8<sup>th</sup> input = the maximum size of the conditioning set allowed

When a conditional independence test is performed, the size of the test set is limited by this parameter to improve the execution speed of the algorithm. Taking "-1" for this parameter means that there are no any restrictions on the size of the test set. In general, the parameter is often set 3 or 4.

Outputs (two outputs):

1<sup>st</sup> output = Index of the PC or MB of a target node or DAG adjacency matrix.

The index value of PC of the target node learnt will be written to the *pc.out* text file in the *PC* folder. And the MB of the target node learnt will be written to the *mb.out* text file in the *MB* folder. For structure learning algorithms, the DAG adjacency matrix of the learnt network structure will be written to the *DAG.out* text file in the *DAG* folder.

2<sup>nd</sup> output = evaluation metric

For PC or MB learning algorithms, the evaluation metrics include precision, recall, F1, distance, running time (in seconds), and number of independence tests (only for constraint-based MB/PC learning algorithms). The metric results will be written to the *indicator.out* file in the *indicator1* folder.

For structural learning algorithms, the evaluation metrics include SHD, undirected,

reverse, miss, extra, running time (in seconds), and number of independence tests (only for constraint-based MB/PC learning algorithms). The metric results will be written to the *indicator.out* file in the *indicator2* folder.

The example of running MMMB are shown in Figures 8 to 11 as follows.



Figure 8 Before running the MMMB for learning MBs of all nodes

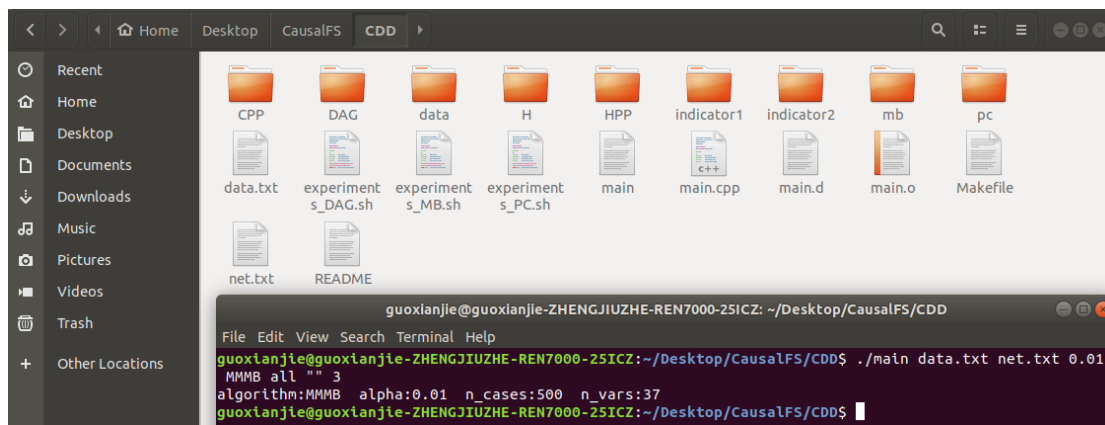


Figure 9 After running the MMMB for learning MBs of all nodes

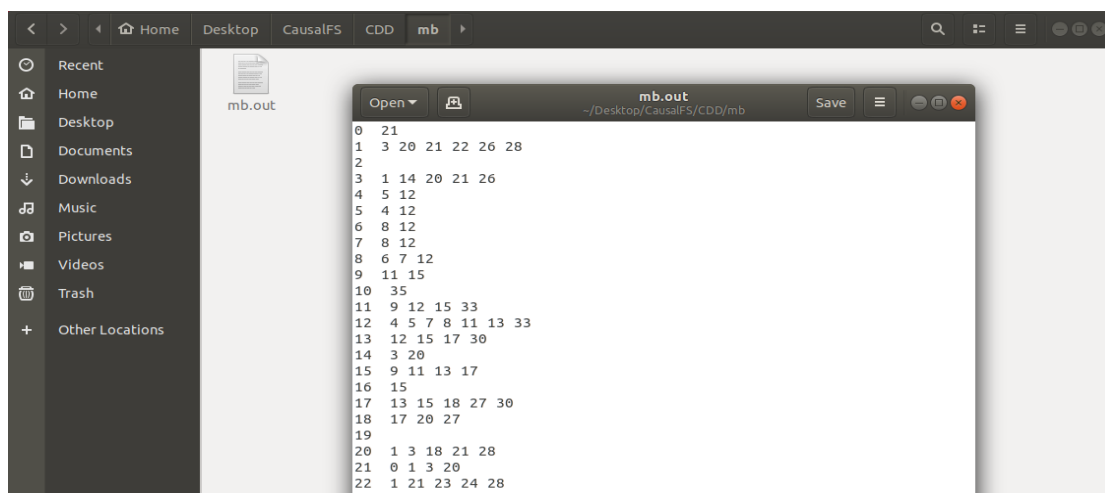


Figure 10 Example of the MBs learnt by MMMB

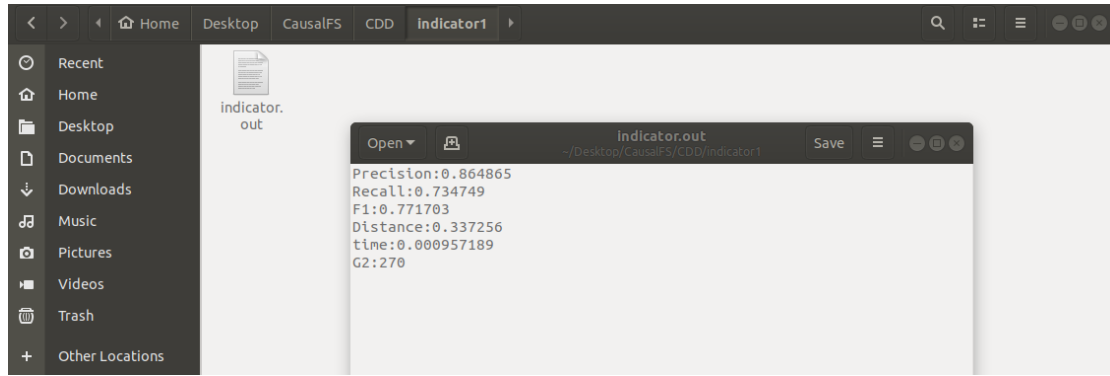


Figure 11 Example of evaluation results learnt by MMMB

Figure 8 and Figure 9 show how to run the MMMB algorithm on the test data set with a significant level set to 0.01 and a test set size limit of 3 for the conditional independence test, with all nodes in the network as the target nodes.

Figure 10 and Figure 11 demonstrate the MB of each target node at the *mb* folder and the evaluation metric results at the *indicator1* folder respectively.

### A. MBtoPC algorithm

```
./main data.txt net.txt 0.01 MBtoPC all "" ""
```

```
./main data.txt net.txt 0.05 MBtoPC 1,5,9 "" ""
```

### B. PC\_simple, MMPC, GetPC, HITON\_PC and Semi\_HITON\_PC algorithms

PC\_simple algorithm:

```
./main data.txt net.txt 0.01 PC_simple all "" -1
```

```
./main data.txt net.txt 0.05 PC_simple 0,4,12 "" 3
```

```
./main data.txt net.txt 0.01 PC_simple all "" 5
```

MMPC algorithm:

```
./main data.txt net.txt 0.01 MMPC all "" -1
```

```
./main data.txt net.txt 0.05 MMPC 0,5,9 "" 3
```

```
./main data.txt net.txt 0.01 MMPC all "" 5
```

GetPC algorithm:

```
./main data.txt net.txt 0.01 GetPC all "" -1
```

```
./main data.txt net.txt 0.05 GetPC 0,4,12 "" 3
```

```
./main data.txt net.txt 0.01 GetPC all "" 5
```

HITON\_PC algorithm:

```
./main data.txt net.txt 0.01 HITON_PC all "" -1
```

```
./main data.txt net.txt 0.05 HITON_PC 0,6,13 "" 3
```

```
./main data.txt net.txt 0.01 HITON_PC all "" 5
```

Semi-HITON\_PC algorithm:

```
./main data.txt net.txt 0.01 Semi_HITON_PC all "" -1
```

```
./main data.txt net.txt 0.05 Semi_HITON_PC 0,4,12 "" 3
```

```
./main data.txt net.txt 0.01 Semi_HITON_PC all "" 5
```

### *C. GSMB,IAMB,Inter\_IAMB, LRH, FBED and MBOR algorithms*

GSMB algorithm:

```
./main data.txt net.txt 0.01 GSMB all "" ""
```

```
./main data.txt net.txt 0.05 GSMB 1,5,9 "" ""
```

IAMB algorithm:

```
./main data.txt net.txt 0.01 IAMB all "" ""
```

```
./main data.txt net.txt 0.05 IAMB 1,5,9 "" ""
```

Inter\_IAMB algorithm:

```
./main data.txt net.txt 0.01 Inter_IAMB all "" ""  
./main data.txt net.txt 0.05 Inter_IAMB 1,5,9 "" ""
```

LRH algorithm:

```
./main data.txt net.txt 0.01 LRH all "" ""  
./main data.txt net.txt 0.05 LRH 1,5,9 "" ""
```

FBED algorithm:

```
./main data.txt net.txt 0.01 FBED all "" ""  
./main data.txt net.txt 0.05 FBED 1,5,9 "" ""
```

MBOR algorithm:

```
./main data.txt net.txt 0.01 MBOR all "" ""  
./main data.txt net.txt 0.05 MBOR 1,5,9 "" ""
```

*D. BAMB, EEMB, MMMB, PCMB, HITON\_MB, Semi\_HITON\_MB, IPC\_MB,*

*STMB and CCMB algorithms*

BAMB algorithm:

```
./main data.txt net.txt 0.01 BAMB all "" -1  
./main data.txt net.txt 0.05 BAMB 0,4,12 "" 3  
./main data.txt net.txt 0.01 BAMB all "" 5
```

EEMB algorithm:

```
./main data.txt net.txt 0.01 EEMB all "" -1  
  
./main data.txt net.txt 0.05 EEMB 0,4,12 "" 3  
  
./main data.txt net.txt 0.01 EEMB all "" 5
```

MMMB algorithm:

```
./main data.txt net.txt 0.01 MMMB all "" -1  
  
./main data.txt net.txt 0.05 MMMB 0,4,12 "" 3  
  
./main data.txt net.txt 0.01 MMMB all "" 5
```

PCMB algorithm:

```
./main data.txt net.txt 0.01 PCMB all "" -1  
  
./main data.txt net.txt 0.05 PCMB 0,4,12 "" 3  
  
./main data.txt net.txt 0.01 PCMB all "" 5
```

HITON\_MB algorithm:

```
./main data.txt net.txt 0.01 HITON_MB all "" -1  
  
./main data.txt net.txt 0.05 HITON_MB 0,4,12 "" 3  
  
./main data.txt net.txt 0.01 HITON_MB all "" 5
```

Semi\_HITON\_MB algorithm:

```
./main data.txt net.txt 0.01 Semi_HITON_MB all "" -1  
  
./main data.txt net.txt 0.05 Semi_HITON_MB 0,4,12 "" 3  
  
./main data.txt net.txt 0.01 Semi_HITON_MB all "" 5
```



IPCMB algorithm:

```
./main data.txt net.txt 0.01 IPC_MB all "" -1  
./main data.txt net.txt 0.05 IPC_MB 0,4,12 "" 3  
./main data.txt net.txt 0.01 IPC_MB all "" 5
```

STMB algorithm:

```
./main data.txt net.txt 0.01 STMB all "" -1  
./main data.txt net.txt 0.05 STMB 0,4,12 "" 3  
./main data.txt net.txt 0.01 STMB all "" 5
```

CCMB algorithm:

```
./main data.txt net.txt 0.01 CCMB all "" -1  
./main data.txt net.txt 0.05 CCMB 0,4,12 "" 3  
./main data.txt net.txt 0.01 CCMB all "" 5
```

### *E. KIAMB algorithm*

```
./main data.txt net.txt 0.01 KIAMB all 0.5 ""  
./main data.txt net.txt 0.05 KIAMB 0,4,12 1.0 ""  
./main data.txt net.txt 0.01 KIAMB all 0.7 ""
```

### *F. Fast\_IAMB algorithm*

This algorithm works only with discrete data.

```
./main data.txt net.txt 0.01 Fast_IAMB all "" ""  
./main data.txt net.txt 0.05 Fast_IAMB 1,5,9 "" ""
```

### *G. TIE\* algorithm*

In this package, the TIE\* algorithm use criterion independence (Z) to verify Markov blankets (boundaries).

```
./main data.txt net.txt 0.01 TIE* all "" ""  
./main data.txt net.txt 0.05 TIE* 1,5,9 "" ""
```

### *H. MMMB-CSL, PCMB-CSL, HITON\_MB-CSL, Semi\_HITON\_MB-CSL, IPC\_MB-CSL, STMB-CSL, MMMB-SSL, PCMB-SSL, HITON\_MB-SSL, Semi\_HITON\_MB-SSL, IPC\_MB-SSL and STMB-SSL algorithms*

XXXX-CSL algorithm:

```
./main data.txt net.txt 0.01 XXXX-CSL "" "" -1  
./main data.txt net.txt 0.05 XXXX-CSL "" "" 3  
./main data.txt net.txt 0.01 XXXX-CSL "" "" 5
```

XXXX-SSL algorithm:

```
./main data.txt net.txt 0.01 XXXX-SSL "" "" -1  
./main data.txt net.txt 0.05 XXXX-SSL "" "" 3  
./main data.txt net.txt 0.01 XXXX-SSL "" "" 5
```

The XXXX-SSL learning structure algorithm only supports discrete data sets.

### *I. MBOR-CSL and MBOR-SSL algorithms*

MBOR-CSL algorithm:

```
./main data.txt net.txt 0.01 MBOR-CSL "" "" ""  
./main data.txt net.txt 0.05 MBOR-CSL "" "" ""
```

MBOR-SSL algorithm:

```
./main data.txt net.txt 0.01 MBOR-SSL "" "" ""
```

```
./main data.txt net.txt 0.05 MBOR-SSL "" "" ""
```

## IV. Score-based algorithms

### IV.1. Description

The score-based project is located at the "SDD" folder, which can only be used to deal with discrete data sets.

### IV.2. Inputs and Outputs

Table 5 Inputs and outputs of score-based algorithms

Inputs (five parameters):	1 <sup>st</sup> input = executable file name
	2 <sup>nd</sup> input = dataset
	3 <sup>rd</sup> input = BN (DAG) adjacency matrix
	4 <sup>th</sup> input = algorithm name
	5 <sup>th</sup> input = target variable index
Outputs (two outputs):	1 <sup>st</sup> output = Index of the PC or MB node of each target node or DAG adjacency matrix
	2 <sup>nd</sup> output = indicator

#### Inputs (five parameters):

When the input parameter is "", it means that the operation of the algorithm does not need a specific value for this parameter. For example, when doing structure learning, all nodes should be taken as the target node by default, that is, the value of the fifth parameter need not be specified.

1<sup>st</sup> input = executable file name

Executable files are generated by building a project through a Makefile, any algorithm in the running project needs to call the executable file in the project. The executable file name generated by the project defaults to main (executable file). The user can modify the name of the executable file in the Makefile, and modify the first line of the Makefile:

EXECUTABLE: = main # executable file name

2<sup>nd</sup> input = dataset

The data set is used for training. Columns are variables and rows are observations. Support both continuous and discrete sample data sets. There are some benchmark BN data sets in the data folder for using.

3<sup>rd</sup> input = BN (DAG) adjacency matrix

A matrix of n rows and n columns consisting of 0 and 1. (i, j) = 1 indicates that a directed edge is from the i node to the j node.

4<sup>th</sup> input = algorithm name

Users can use one of the following algorithm names in Tables 6 to 8 as follows as the 4<sup>th</sup> input, such as SLL-PC, S2TMB, SLL-SSL, etc.

Table 6 PC learning algorithm

- SLL-PC - MBOR uses the MBtoPC algorithm to find PC
- S2TMB-PC - PC-simple algorithm

Table 7 MB learning algorithm:

- SLL - Grow/Shrink algorithm
- S2TMB - Incremental Association-Based Markov Blanket (IAMB)

Table 8 Structure learning algorithm:

- SLL-CSL
- S2TMB-CSL

- SLL-SSL
- S2TMB-SSL

In Table 8, -CSL indicates that the MB (or PC) is learned by a score-based MB method while orient edge directions using constraint-based rules. For example, SLL-CSL means that it uses SLL to learn MBs to construct a DAG skeleton and then employs constraint-based rules to orient edge directions in the learnt skeleton. -SSL means that the MB (or PC) is learned by a score-based MB method while orient edge directions using score-based methods (e.g., score-based hill climbing methods). In addition, -SSL only supports discrete data sets.

5<sup>th</sup> input = target variable index

If the number of nodes in a BN network is n, the index value of a node is among 0 to n-1. When users need to learn the MBs of all nodes in the network, the parameter is set to "all". When users need to learn the MBs of several node(s) in the network, the specified target nodes need to be separated by ",". For example, "0,5,9" means an algorithm will return the MBs of the nodes 0, 5, and 9.

Outputs (**two outputs**):

**1<sup>st</sup> output** = Index of the PC or MB of each target node or DAG adjacency matrix.

The index value of PC of the target node learnt will be written to the *pc.out* text file in the *PC* folder. And the MB of the target node learnt will be written to the *mb.out* text file in the *MB* folder. For structure learning algorithms, the adjacency matrix of the network structure learnt will be written to the *DAG.out* text file in the *DAG* folder.

**2<sup>nd</sup> output** = evaluation metric

For PC or MB learning algorithms, the evaluation metrics include **precision**, **recall**, **F1**, **distance** and **running time** (in seconds). The metric results will be written to the *indicator.out* file in the *indicator1* folder.

For structural learning algorithms, the evaluation metrics include **SHD**, **undirected**, **reverse**, **miss**, **extra** and **running time** (in seconds). The metric results will be written to the *indicator.out* file in the *indicator2* folder.

The example of running SLL are shown in Figures 12 to 15 as follows.

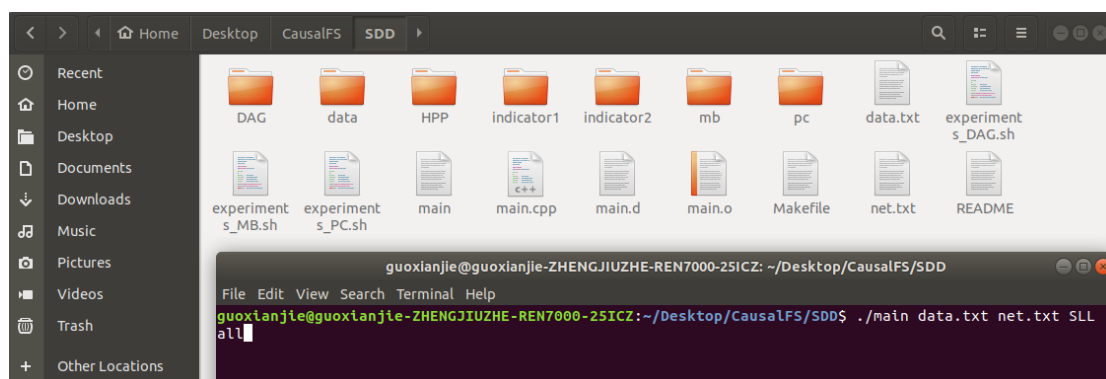


Figure 12 Before running the SLL for learning MBs of all nodes



Figure 13 After running the SLL for learning MBs of all nodes

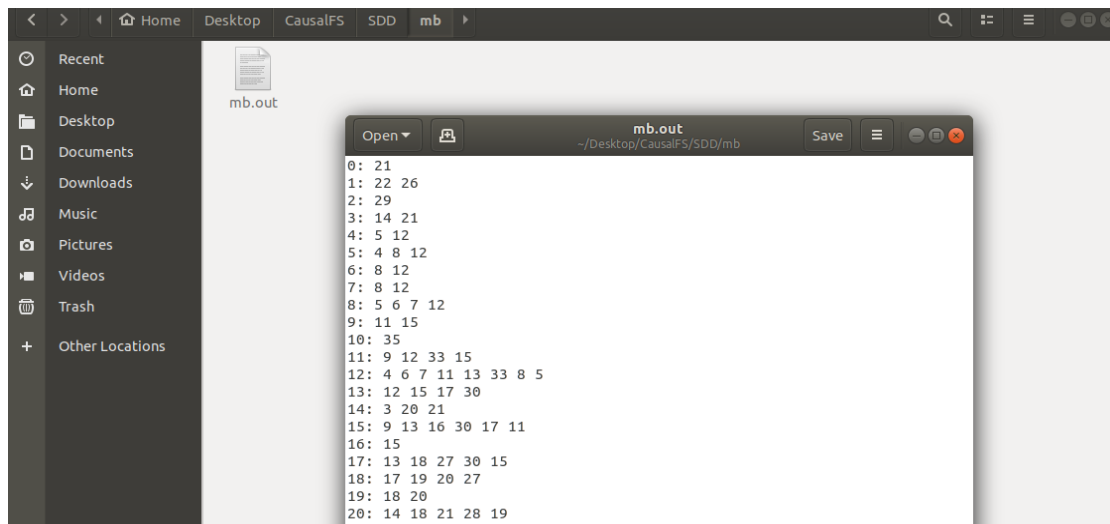


Figure 14 Example of MBs learnt by SLL



Figure 15 Example of evaluation results learnt by SLL

Figure 12 and Figure 13 show how to run the SLL algorithm on the test data set with all nodes in the network as the target nodes.

Figure 14 and Figure 15 demonstrate the MB of each target node at the *mb* folder and the evaluation metric results at the *indicator1* folder respectively.

### A. SLL-PC and S2TMB-PC algorithms

SLL-PC algorithm:

```
./main data.txt net.txt SLL-PC all
```

```
./main data.txt net.txt SLL-PC 0,8,13
```

S2TMB-PC algorithm:

```
./main data.txt net.txt S2TMB-PC all  
./main data.txt net.txt S2TMB-PC 0,8,13
```

### *B. SLL and S2TMB algorithms*

SLL algorithm:

```
./main data.txt net.txt SLL all  
./main data.txt net.txt SLL 0,8,13
```

S2TMB algorithm:

```
./main data.txt net.txt S2TMB all  
./main data.txt net.txt S2TMB 0,8,13
```

### *C. SLL-CSL, S2TMB-CSL, SLL-SSL and S2TMB-SSL algorithms*

XXXX-CSL algorithm:

```
./main data.txt net.txt XXXX-CSL ""
```

XXXX-SSL algorithm:

```
./main data.txt net.txt XXXX-SSL ""
```

## **V. Script run experiment**

### V.1. Description

If users run the experiments using large number of data sets, they can use the ".sh" script. Each project has a template for the ".sh" script to run experiments, such as: *experiments\_PC*, *experiments\_MB* and *experiments\_DAG*.

## V.2. “.sh” script basic syntax

### A. The beginning

The program must start with the following line (must be in the first line of the file):

```
#!/bin/sh
```

The symbol “#!” is used to tell the system that the parameter following it is the program used to execute the file.

### B. The notes

When doing shell programming, a sentence beginning with `#` indicates a comment until the end of the line.

### C. Common shell commands

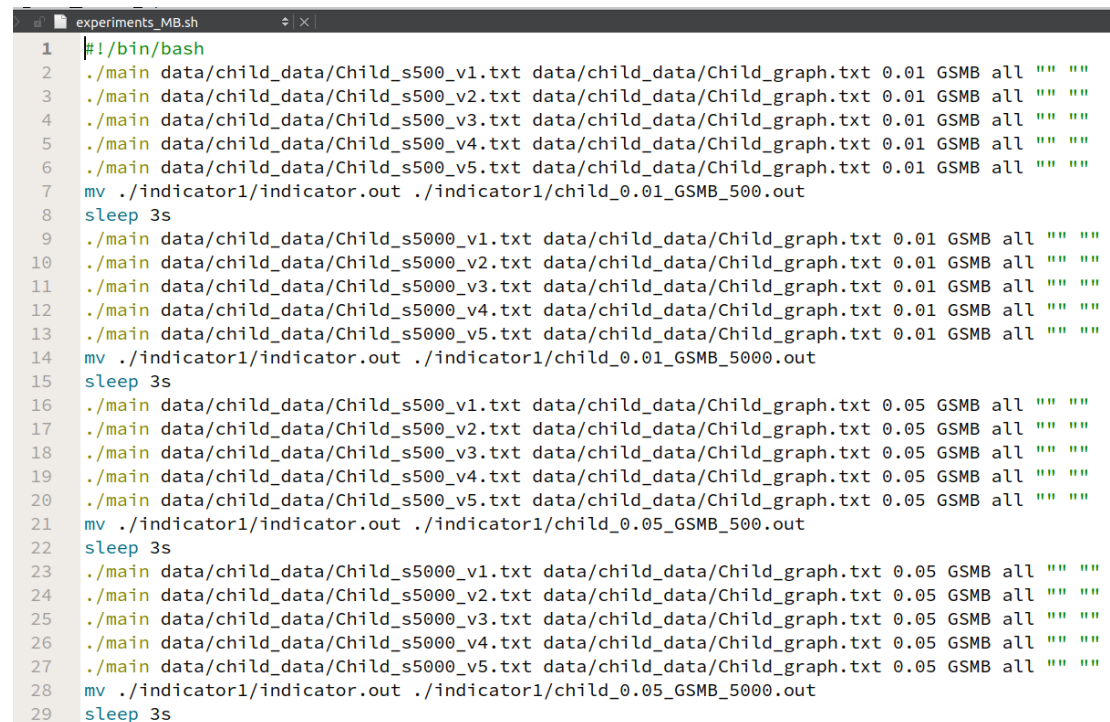
`echo` "some text": Print text content on the screen

`cp` sourcefile destfile: file copy

`mv` oldname newname: Rename file or move file

`rm` file: delete file

`sleep` time: Sleep for a while



```
1  #!/bin/bash
2  ./main data/child_data/Child_s500_v1.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
3  ./main data/child_data/Child_s500_v2.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
4  ./main data/child_data/Child_s500_v3.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
5  ./main data/child_data/Child_s500_v4.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
6  ./main data/child_data/Child_s500_v5.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
7  mv ./indicator1/indicator.out ./indicator1/child_0.01_GSMB_500.out
8  sleep 3s
9  ./main data/child_data/Child_s5000_v1.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
10 ./main data/child_data/Child_s5000_v2.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
11 ./main data/child_data/Child_s5000_v3.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
12 ./main data/child_data/Child_s5000_v4.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
13 ./main data/child_data/Child_s5000_v5.txt data/child_data/Child_graph.txt 0.01 GSMB all "" ""
14 mv ./indicator1/indicator.out ./indicator1/child_0.01_GSMB_5000.out
15 sleep 3s
16 ./main data/child_data/Child_s500_v1.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
17 ./main data/child_data/Child_s500_v2.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
18 ./main data/child_data/Child_s500_v3.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
19 ./main data/child_data/Child_s500_v4.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
20 ./main data/child_data/Child_s500_v5.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
21 mv ./indicator1/indicator.out ./indicator1/child_0.05_GSMB_500.out
22 sleep 3s
23 ./main data/child_data/Child_s5000_v1.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
24 ./main data/child_data/Child_s5000_v2.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
25 ./main data/child_data/Child_s5000_v3.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
26 ./main data/child_data/Child_s5000_v4.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
27 ./main data/child_data/Child_s5000_v5.txt data/child_data/Child_graph.txt 0.05 GSMB all "" ""
28 mv ./indicator1/indicator.out ./indicator1/child_0.05_GSMB_5000.out
29 sleep 3s
```

Figure 16 Example of the “.sh” script file (experiments\_MB.sh)



In Figure 16, we randomly generate two groups of data, one group including 5 data sets with 500 data instances, and the other group also containing 5 data sets with 5,000 data instances. Set the significant level of 0.01 and 0.05 to run the GSMB algorithm on the child Bayesian network, and writes the indicator results into the following files:

*child\_0.01\_GSMB\_500.out*、*child\_0.01\_GSMB\_5000.out*、*child\_0.05\_GSMB\_500.out*、*child\_0.05\_GSMB\_5000.out*.

### V.3. Running experiment

Set the permissions in the properties of the script file, make the executable option "Allow executing file as a program" (Shown in Figure 17), only then can you run with “./ scriptname”. Finally, run the script in the terminal.

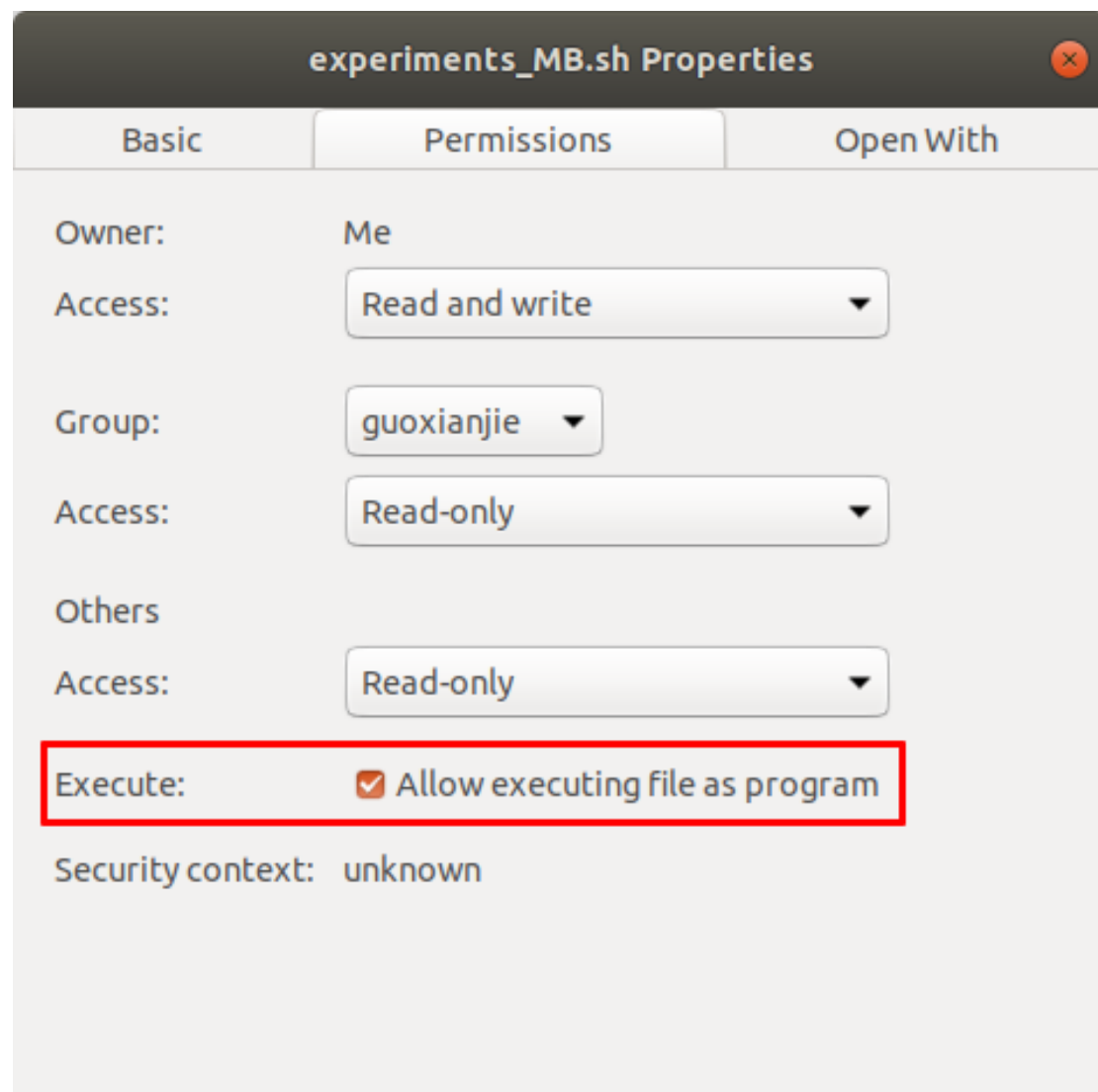


Figure 17 Example of setting file permissions

Figure 18 and Figure 19 show how to run the MB experiment in batches using the "experiments\_MB.sh" script file in the project package.

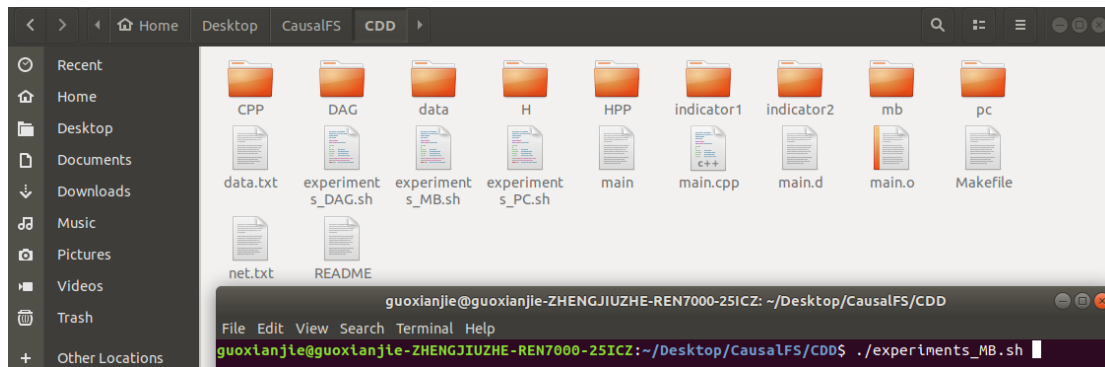


Figure 18 Before the script running

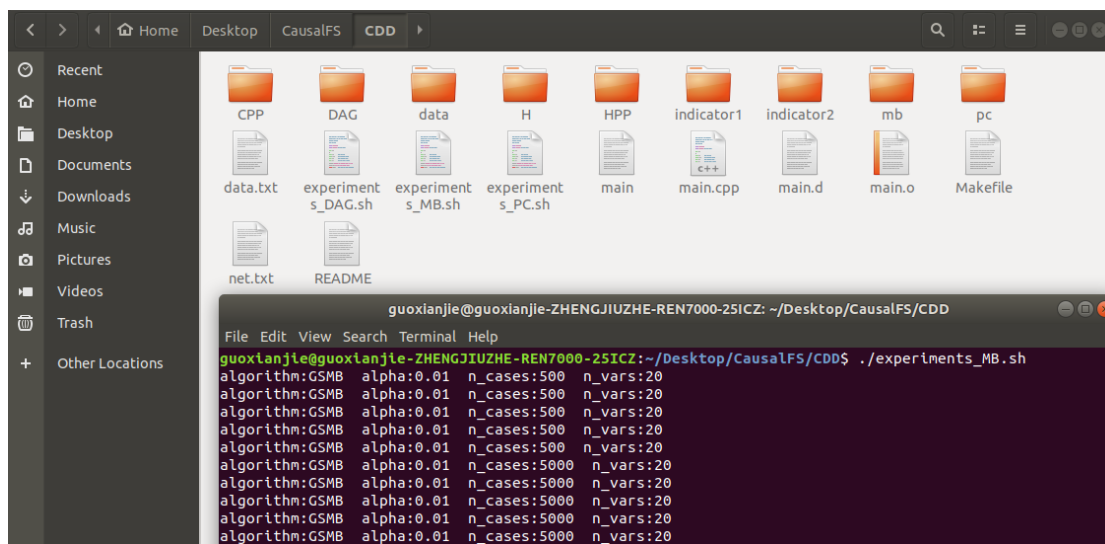


Figure 19 The script is running