AALTO UNIVERSITY, SCHOOL OF SCIENCE

Identification of gene regulatory network from gene expression time-course data

Alena Alferova, 996981

March 6, 2022

1 RESEARCH QUESTIONS

In this project the gene expression data was given, and the task is to identify regulation links between 5 genes using experimental time-course data. The goal of this project is to apply several computational algorithms in order to simulate regulatory network of these genes. It is assumed that these 5 genes operate in isolation and are not affected by other molecules. The data of gene expression is represented on the figure 1.1.

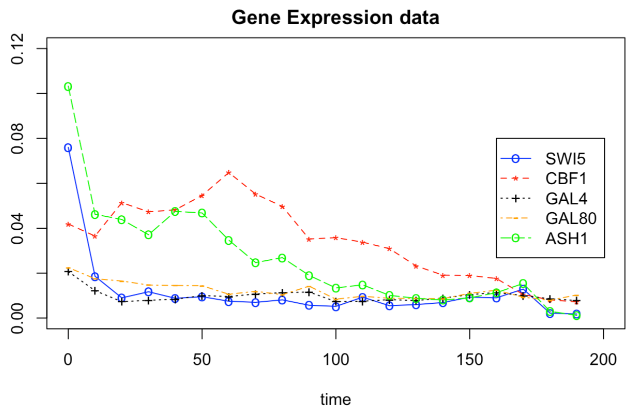


Figure 1.1: Gene expression time-course data [1]

2 MOTIVATION FOR CHOOSING ALGORITHMS

In order to solve this network inference problem, I have chosen several computational approaches of network reconstruction. The first one is *Information-Theoretic Approach*, which aimed at identifying statistical dependencies between variables. Such complex network requires more sophisticated approach than simple correlation measures, so I used mutual information-based algorithms.

I have chosen different algorithms within information-theoretic approach in order to compare different ways of dealing with statistical redundancy caused by indirect links.
The second approach is *Genetic Regulatory Networks* approach based on boolean logic. This approach stands for representation of each node (gene) as boolean function dependent on other genes as input variables.
In my opinion, these methods can be considered as a reasonable choice for gene expression data analysis, because it sets much less boundaries on the structure of regulatory network, as opposed to Bayesian Network methods, in which the network is assumed to be a DAG.

3 Information-Theoretic Approach

The information-theoretic method for network inference is primarily based on mutual information computation between pairs of genes X_i and X_j, that is considered to be more sophisticated alternative to correlation measure of statistical dependence.
Mutual information concept is a part of information theory, in which the information is defined as "the resolution of uncertainty", meaning that the most unlikely events convey the biggest amount of information. According to this notation, the information content of a message can be defined as $I(x) = -\log(p(x))$, where $p(x)$ is a probability distribution of a data item or a message.
Entropy is a quantitative measure of unpredictability of a random event within current information theory is defined as: $H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$.
However, the concept of mutual information is used, when the relationship between two variables is to be measured. In particular, mutual information defines how much information about one variable can be obtained form the observation of another.
The mutual information for all pairs of genes is stored in a form of *Mutual Information Matrix (MIM)*:

$$MIM_{ij} = I(X_i; X_j) = D_{KL}\big(p(X_i, X_j) || p(X_i)p(X_j)\big) = \sum_{i,j} p(x_i, x_j) \log\left(\frac{p(x_i, x_j)}{p(x_i)p(x_j)}\right)$$

There are several algorithms that use mutual information for likelihood estimate score of a regulatory link between genes in the network, and those algorithms will be explained in following sections.

3.1 CLR algorithm

CLR algorithm [2] operates similarly to the ordinary Relevance Network algorithm, which clams that two nodes (genes) are linked with an edge if the mutual information between these genes is more than a certain threshold I_0. However, in the CLR algorithms the likelihood estimate score of an edge that includes mutual information is calculated as following:

$$\text{score}_{ij} = \sqrt{(\text{score}_i^2 + \text{score}_j^2)},$$

where

$$\text{score}_i = \max\left(0, \frac{I(X_i; X_j) - \mu_i}{\sigma_i}\right), \text{score}_j = \max\left(0, \frac{I(X_i; X_j) - \mu_j}{\sigma_j}\right)$$

μ and σ are mean and standard deviation of empirical distribution of MI values.

3.2 Aracne algorithm

Aracne algorithm is based on data processing inequality theorem, which states that if genes X_1 and X_3 communicate with each other through gene X_2, then:

$$I(X_1, X_2) \geq I(X_1, X_3), I(X_2, X_3) \geq I(X_1, X_3) \Rightarrow I(X_1, X_3) \leq \min(I(X_1, X_2), I(X_2, X_3))$$

This means that the mutual information of genes connected directly is bigger than those that are indirectly connected. Apart from thresholding each pair of genes by comparing the mutual information with a certain threshold, Aracne algorithm uses this theorem to reduce redundancy in statistical correlations, removing the indirect links by analyzing all triplets of genes and removing the weakest link among a triplet.

3.3 Minimum Redundancy networks (MRNET)

This algorithm is introduced in [3], and included in R software package MINET. This algorithm is based on maximum relevance/minimum redundancy concept, addressing the problem of correlation redundancy in links identification. The algorithms contains the following steps:

1. Let's denote X as a set of genes in the network, and let's choose the first target gene $Y \in X$, and then all possible input genes are in $V = X \backslash \{Y\}$.
2. Firstly, the input gene X_i with highest mutual information $I(X_i; Y)$ is chosen and put in the set S.
3. Secondly, the score function for the next gene X_j is calculated as the difference between mutual information $I(X_j; Y)$ and the redundancy measure for X_j, that is computed as follows: $r_j = \frac{1}{|S|} \sum_{X_i \in S} I(X_i, X_j)$. The gene X_j with the high mutual information with the target gene Y and low mutual information with previous $X_i \in S$ selected gene is put into S set:
$$X_j = \text{argmax}_{X_j \in V \backslash S}(I(X_j; Y) - r_j)$$
4. Repeat for all genes by denoting them as target genes.
5. After the loop is finished, two score for each pair of genes (X_i, X_j) is computed, and the maximum is chosen. As a result, weighted adjacency matrix is returned as output of the algorithm.

Although this algorithms help to solve the problem of redundancy in links identification, it is not able to identify the directions of the links due to the symmetry of mutual information measure. So, other algorithms should be added to the computational model in order to take into account directionality of the network.

4 Genetic Regulatory Networks

This class of methods is based on boolean logic. In current study, "GEne Network Inference with Ensemble of trees" (GENIE) [4] algorithm was used as a R software package *GENIE3*. In this algorithms, the gene expression data is represented as collection of N samples with the values for all p genes. For example, in our case $N = 20$, $p = 5$ and a sample vector can be written as: $x_k = (x_k^1, ..., x_k^5)^T, k = 1..20$.
It is assumed that the boolean function of expression for j-th gene is defined as follows:

$$x_k^j = f_j(x_k^{-j}), k = 1..20$$

where x_k^{-j} is the vector of expression values for all parental (or input) genes of the gene j.

Then, tree-based ensemble method is applied in order to find function f_j by minimizing square error loss: $\sum_{k=1}^{N}(x_k^j - f_j(x_k^{-j}))^2$.

So, the algorithm can be described using the following steps:

1. For each $j = 1,...5$, the learning sample is defined as $\{ (\mathbf{x}_k^{-j}, x_k^j), k = 1,...,20 \}$.
2. For each learning sample, the function of expression is calculated using Random Forest or Extra-trees tree-based ensemble methods, which allow to calculate the importance measure for each input gene.
3. Then, for each learning sample, the input (or parental) genes are ranked according to their importance in predicting the target gene x_k^j.

5 Performance Evaluation

In order to evaluate how well chosen algorithms were able to reconstruct given regulatory network, the results were compared with the reference network figure 5.1, provided in [1]. The output of all applied algorithms is the adjacency matrix, so I created the true adjacency matrix based on 5.1 and compared it with the results of modelling by using the function `minet::validate(model_matrix, true_matrix)`. This function returns the table with four metrics: different threshold for the score, TP rate, FP rate, and FN rate. Using this data, the ROC curve and optimal graph were plotted. The result is represented on the figures 5.2, 5.3, 5.4.
As can be seen from the figures, the best result was produced by MRNET algorithm.
The same comparison was conducted with the resulted weighted adjacency matrix of *GENIE3* algorithm, and the ROC curve was plotted 5.5.

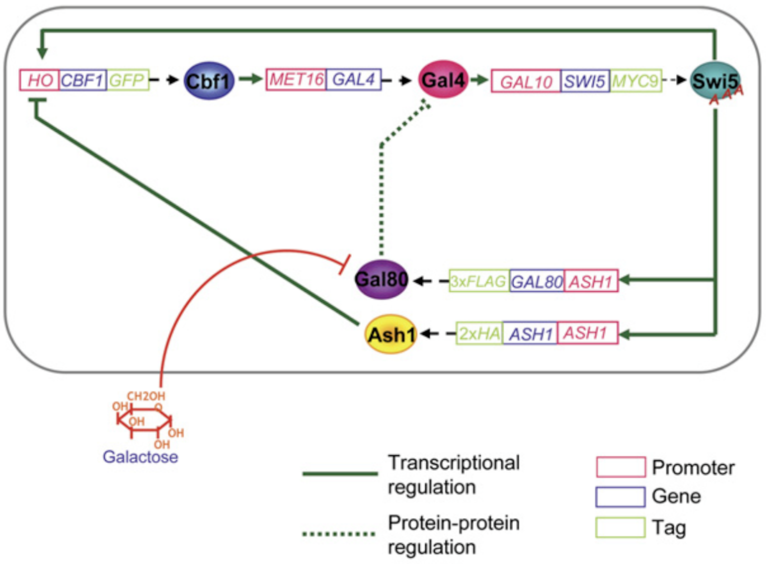
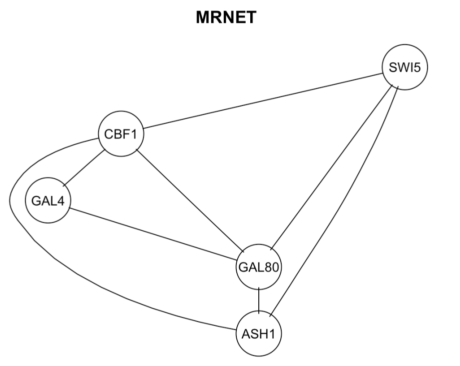


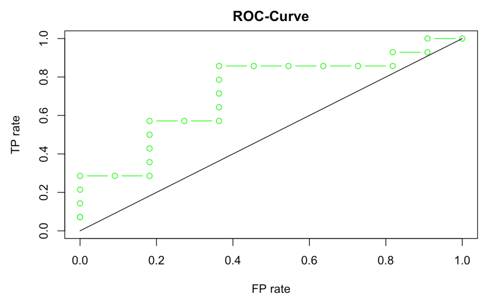
Figure 5.1: Reference 5-gene regulatory network [1]

6 Discussion

As a result, several approaches were applied to reconstruct the regulatory network for gene expression time-course data. Information-theoretic approach and genetic regulatory network

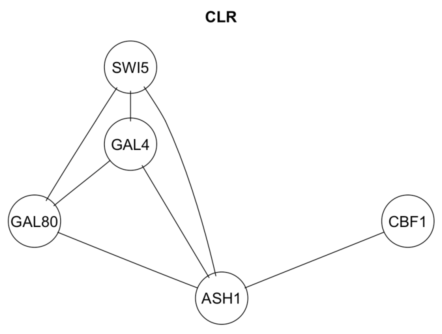


(a)

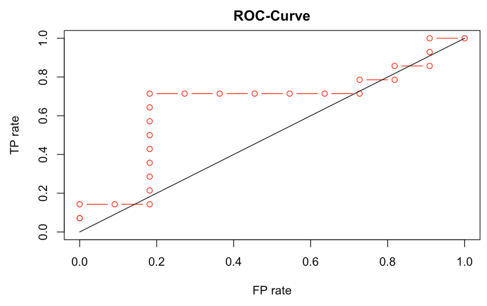


(b) ROC curve constructed by varying the threshold I_0

Figure 5.2: MRNET: Minimum Redundancy networks algorithm performance

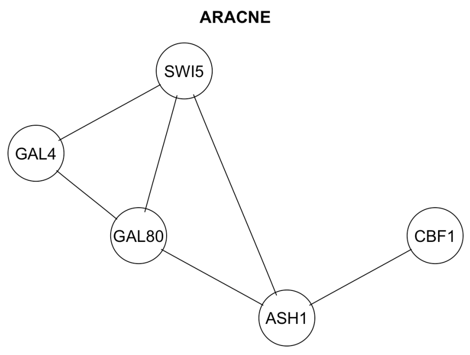


(a)

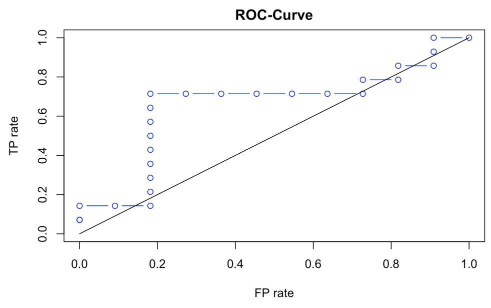


(b) ROC curve constructed by varying the threshold I_0

Figure 5.3: CLR algorithm performance



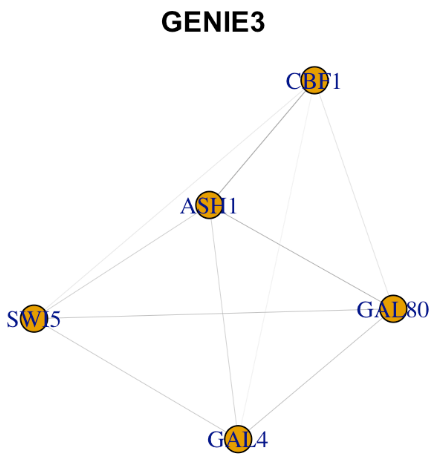
(a)



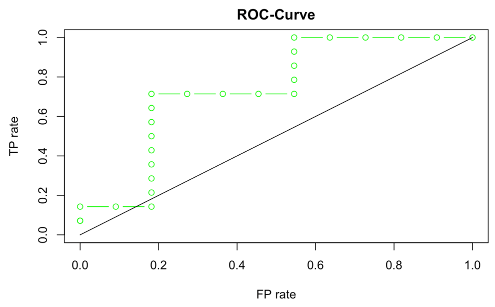
(b) ROC curve constructed by varying the threshold I_0

Figure 5.4: ARACNE algorithm performance

algorithms performed well, while Bayesian Networks methods (which I also tried to apply) performed poorly, probably because of the assumption of acyclicity in the network structure.
In my opinion, the most promising algorithms for network inference are genetic regulatory network class of methods, which are based on boolean logic. The reason behind that is the possibility to produce a directed graph using these methods, while information-theoretic approach itself does not allow to do that. However, I was not able to decently reconstruct the direction of edges using GRN algorithm in current study. So, I think there is a room for improvement.



(a) The graph with the width of edges corresponding to the weight



(b) ROC curve constructed by varying the threshold I_0

Figure 5.5: GENIE3: Genetic Regulatory Networks method performance

The performance of information-theoretic approach can be improved by adding some a priori or biological knowledge into the model.

References

[1] Cantone I et al. *"A yeast synthetic network for in vivo assessment of reverse- engineering and modeling approaches."* Cell, 137, 172–181. (2009)

[2] Jeremiah J Faith ,Boris Hayete ,Joshua T Thaden, at al *"Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles"*

[3] Patrick E. Meyer, Kevin Kontos, Frederic Lafitte, and Gianluca Bontempi *"Information-Theoretic Inference of Large Transcriptional Regulatory Networks"* Journal on Bioinformatics and Systems Biology

[4] Vân Anh Huynh-Thu, Alexandre Irrthum,cLouis Wehenkel, Pierre Geurts *"Inferring Regulatory Networks from Expression Data Using Tree-Based Methods"*

project_code

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
data <- read.table("data.txt", header = T, row.names = 1)
(data)
```

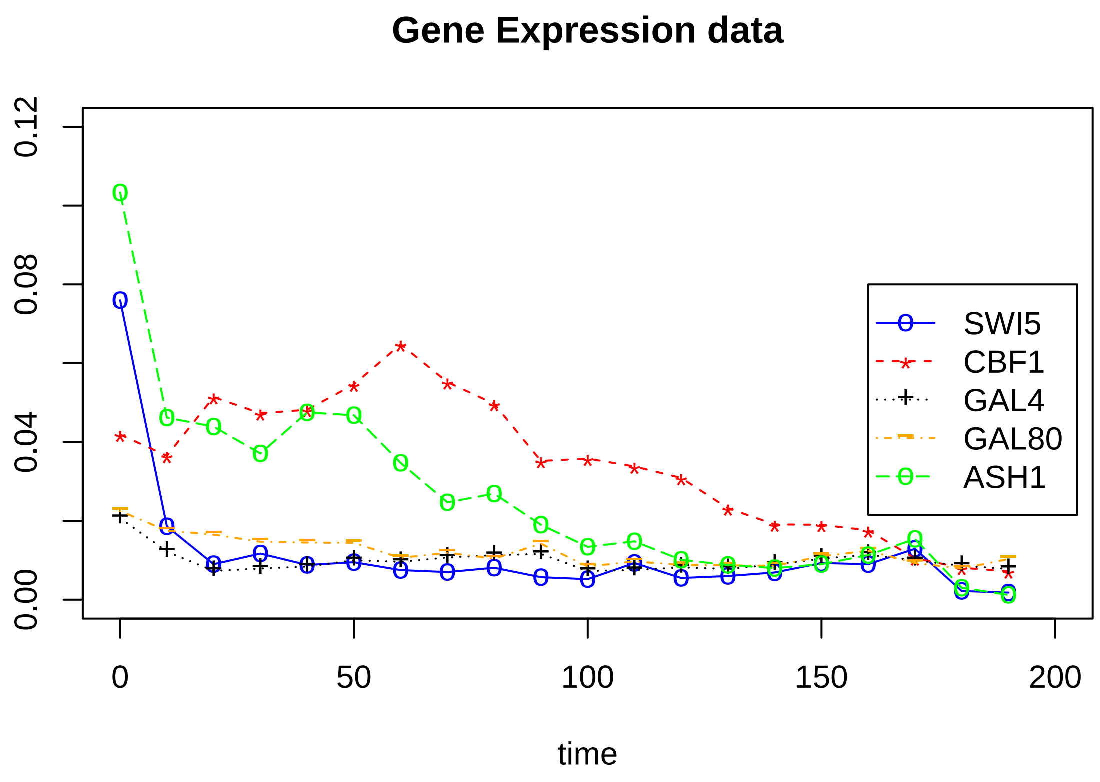
```
##       SWI5   CBF1   GAL4  GAL80   ASH1
## 0   0.0760 0.0419 0.0207 0.0225 0.1033
## 10  0.0186 0.0365 0.0122 0.0175 0.0462
## 20  0.0090 0.0514 0.0073 0.0165 0.0439
## 30  0.0117 0.0473 0.0079 0.0147 0.0371
## 40  0.0088 0.0482 0.0084 0.0145 0.0475
## 50  0.0095 0.0546 0.0100 0.0144 0.0468
## 60  0.0075 0.0648 0.0096 0.0106 0.0347
## 70  0.0070 0.0552 0.0107 0.0119 0.0247
## 80  0.0081 0.0497 0.0113 0.0104 0.0269
## 90  0.0057 0.0352 0.0116 0.0142 0.0190
## 100 0.0052 0.0358 0.0073 0.0084 0.0134
## 110 0.0093 0.0338 0.0075 0.0097 0.0148
## 120 0.0055 0.0309 0.0082 0.0088 0.0101
## 130 0.0060 0.0232 0.0078 0.0087 0.0088
## 140 0.0069 0.0191 0.0089 0.0086 0.0080
## 150 0.0093 0.0190 0.0104 0.0110 0.0090
## 160 0.0090 0.0176 0.0114 0.0124 0.0113
## 170 0.0129 0.0105 0.0100 0.0093 0.0154
## 180 0.0022 0.0081 0.0086 0.0079 0.0030
## 190 0.0018 0.0072 0.0078 0.0103 0.0012
```

```
(time <- seq(from = 0, to = 190, by = 10))
```

```
##  [1]   0  10  20  30  40  50  60  70  80  90 100 110 120 130 140 150 160 170 180
## [20] 190
```

```
plot(time, data[,1], type="o", col="blue", pch="o", lty=1, ylim=c(0, 0.12), xlim=c(0,200), ylab="", mai

points(time, data[,2], col="red", pch="*")
lines(time, data[,2], col="red",lty=2)

points(time, data[,3], col="black",pch="+")
lines(time, data[,3], col="black", lty=3)

points(time, data[,4], col="orange",pch="-")
lines(time, data[,4], col="orange", lty=4)
```

```
points(time, data[,5], col="green",pch="o")
lines(time, data[,5], col="green", lty=5)

legend(160,0.08,legend=c("SWI5","CBF1","GAL4", "GAL80", "ASH1"), col=c("blue","red","black"   ,"orange"
```



```
library(BiocManager)
library(minet)
library(infotheo)
library("graph")
```

```
## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

```
library("bnlearn")
```

```
##
## Attaching package: 'bnlearn'

## The following object is masked from 'package:infotheo':
##
##     discretize

## The following object is masked from 'package:minet':
##
##     aracne
```

```
# Inference
(mr <- minet(data, method="mrnet", estimator="mi.mm", disc="equalfreq"))
```

```
##            SWI5      CBF1      GAL4     GAL80      ASH1
## SWI5  0.0000000 0.1548494 0.0000000 0.1032329 0.7455555
## CBF1  0.1548494 0.0000000 0.2544445 0.1907316 1.0000000
## GAL4  0.0000000 0.2544445 0.0000000 0.9447457 0.0000000
## GAL80 0.1032329 0.1907316 0.9447457 0.0000000 0.7992156
## ASH1  0.7455555 1.0000000 0.0000000 0.7992156 0.0000000
```

```
(ar <- minet( data, method="aracne", estimator="mi.mm", disc="equalwidth"))
```

```
##            SWI5 CBF1      GAL4     GAL80      ASH1
## SWI5  0.0000000    0 0.3813264 0.3813264 0.3813264
## CBF1  0.0000000    0 0.0000000 0.0000000 1.0000000
## GAL4  0.3813264    0 0.0000000 0.4712496 0.0000000
## GAL80 0.3813264    0 0.4712496 0.0000000 0.6056431
## ASH1  0.3813264    1 0.0000000 0.6056431 0.0000000
```

```
(clr<- minet( data, method="clr", estimator="mi.mm", disc="equalwidth"))
```

```
##            SWI5 CBF1       GAL4     GAL80       ASH1
## SWI5  0.0000000    0 0.31339532 0.2483612 0.24836120
## CBF1  0.0000000    0 0.00000000 0.0000000 1.00000000
## GAL4  0.3133953    0 0.00000000 0.4601762 0.07550061
## GAL80 0.2483612    0 0.46017617 0.0000000 0.62201085
## ASH1  0.2483612    1 0.07550061 0.6220108 0.00000000
```

```
gr_mr <- as(mr, "graphNEL")
gr_clr <- as(clr, "graphNEL")
gr_ar <- as(ar, "graphNEL")

dim_names <- c("SWI5","CBF1","GAL4", "GAL80", "ASH1")
mat_data <- c(0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0)
(true_adj_mat <- matrix(mat_data, nrow=5, ncol=5, dimnames = list(dim_names, dim_names)))
```

```
##       SWI5 CBF1 GAL4 GAL80 ASH1
## SWI5     0    1    1     1    1
## CBF1     1    0    1     0    1
## GAL4     1    1    0     1    0
## GAL80    1    0    1     0    0
## ASH1     1    1    0     0    0
```

```
mr.tbl <- minet::validate(mr, true_adj_mat)
ar.tbl <- minet::validate(ar, true_adj_mat)
```

```r
clr.tbl <- minet::validate(clr, true_adj_mat)

show.roc(mr.tbl, col="green", type="b")
```

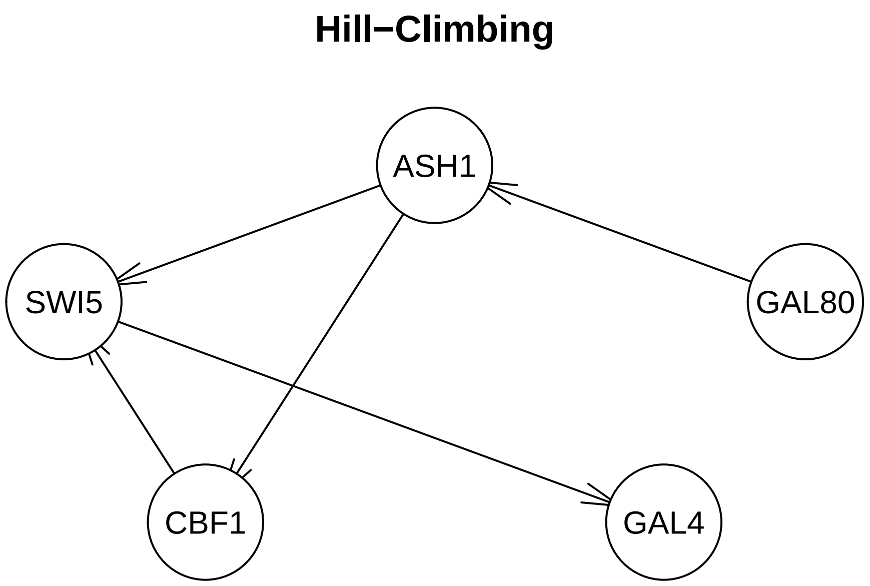
```
## pdf
##   3
```

```r
#show.roc(ar.tbl, col="blue", type="b")
#show.roc(clr.tbl, col="red", type="b")

plot(gr_mr, main = "MRNET")
#plot(gr_ar, main = "ARACNE")
#plot(gr_clr, main = "CLR")
```

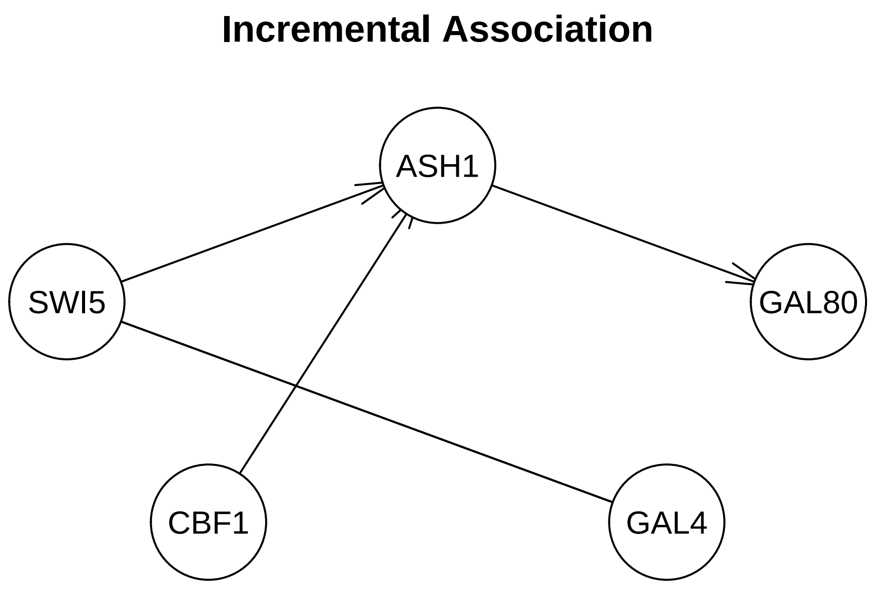
```r
disc_data <- discretize(data)
bn.hc <- bnlearn::hc(data)
(bn.hc)
```

```
##
##   Bayesian network learned via Score-based methods
##
##   model:
##    [GAL80][ASH1|GAL80][CBF1|ASH1][SWI5|CBF1:ASH1][GAL4|SWI5]
##   nodes:                                 5
##   arcs:                                  5
##     undirected arcs:                     0
##     directed arcs:                       5
##   average markov blanket size:           2.00
##   average neighbourhood size:            2.00
##   average branching factor:              1.00
##
##   learning algorithm:                    Hill-Climbing
##   score:                                 BIC (Gauss.)
##   penalization coefficient:              1.497866
##   tests used in the learning procedure:  30
##   optimized:                             TRUE
```

```r
plot(bn.hc, main = "Hill-Climbing")
```



```r
bn.iamb <- bnlearn::iamb(data)
plot(bn.iamb, main = "Incremental Association")
```



```
library("GeneNet")
```

```
## Loading required package: corpcor

## Loading required package: longitudinal

## Loading required package: fdrtool
```

```
data_long <- as.longitudinal(as.matrix(data))
(data_long)
```

```
## Longitudinal data:
##  5 variables measured at 20 different time points
##  Total number of measurements per variable: 20
##  Repeated measurements: none
##
##  To obtain the measurement design call 'get.time.repeats()'.
##
##         SWI5   CBF1   GAL4  GAL80   ASH1
## 1-1  0.0760 0.0419 0.0207 0.0225 0.1033
## 2-1  0.0186 0.0365 0.0122 0.0175 0.0462
## 3-1  0.0090 0.0514 0.0073 0.0165 0.0439
## 4-1  0.0117 0.0473 0.0079 0.0147 0.0371
## 5-1  0.0088 0.0482 0.0084 0.0145 0.0475
## 6-1  0.0095 0.0546 0.0100 0.0144 0.0468
## 7-1  0.0075 0.0648 0.0096 0.0106 0.0347
## 8-1  0.0070 0.0552 0.0107 0.0119 0.0247
## 9-1  0.0081 0.0497 0.0113 0.0104 0.0269
## 10-1 0.0057 0.0352 0.0116 0.0142 0.0190
## 11-1 0.0052 0.0358 0.0073 0.0084 0.0134
## 12-1 0.0093 0.0338 0.0075 0.0097 0.0148
## 13-1 0.0055 0.0309 0.0082 0.0088 0.0101
## 14-1 0.0060 0.0232 0.0078 0.0087 0.0088
## 15-1 0.0069 0.0191 0.0089 0.0086 0.0080
## 16-1 0.0093 0.0190 0.0104 0.0110 0.0090
## 17-1 0.0090 0.0176 0.0114 0.0124 0.0113
## 18-1 0.0129 0.0105 0.0100 0.0093 0.0154
## 19-1 0.0022 0.0081 0.0086 0.0079 0.0030
## 20-1 0.0018 0.0072 0.0078 0.0103 0.0012
```

```
library("GENIE3")
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:bnlearn':
##
##     as.igraph, compare, degree, subgraph

## The following objects are masked from 'package:graph':
##
##     degree, edges, intersection, union

## The following objects are masked from 'package:BiocGenerics':
##
##     normalize, path, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
(weightMatrix <- GENIE3(t(as.matrix(data)), returnMatrix=TRUE))
```

```
##             SWI5       CBF1       GAL4     GAL80      ASH1
## SWI5  0.00000000 0.13896289 0.32351607 0.2152248 0.2008716
## CBF1  0.01880103 0.00000000 0.04707072 0.1077469 0.2704956
## GAL4  0.34860985 0.08007972 0.00000000 0.1968220 0.1515865
## GAL80 0.34614179 0.17708869 0.35965278 0.0000000 0.3770463
## ASH1  0.28644733 0.60386870 0.26976043 0.4802063 0.0000000
```

```
linkList <- getLinkList(weightMatrix)
(linkList)
```

```
##   regulatoryGene targetGene     weight
## 1           ASH1       CBF1 0.60386870
## 2           ASH1      GAL80 0.48020631
## 3          GAL80       ASH1 0.37704634
## 4          GAL80       GAL4 0.35965278
## 5           GAL4       SWI5 0.34860985
## 6          GAL80       SWI5 0.34614179
## 7           SWI5       GAL4 0.32351607
## 8           ASH1       SWI5 0.28644733
## 9            CBF1       ASH1 0.27049556
## 10           ASH1       GAL4 0.26976043
## 11           SWI5      GAL80 0.21522482
## 12           SWI5       ASH1 0.20087164
## 13           GAL4      GAL80 0.19682196
## 14          GAL80       CBF1 0.17708869
## 15           GAL4       ASH1 0.15158646
## 16           SWI5       CBF1 0.13896289
## 17           CBF1      GAL80 0.10774691
## 18           GAL4       CBF1 0.08007972
## 19           CBF1       GAL4 0.04707072
## 20           CBF1       SWI5 0.01880103
```

```
(gr_mr <- as(weightMatrix, "graphNEL"))
```

```
## A graphNEL graph with directed edges
## Number of Nodes = 5
## Number of Edges = 20
```

```
eg <- graph_from_data_frame(linkList, directed=TRUE)
g <- graph_from_adjacency_matrix(weightMatrix, mode="max", weighted=T)
genie <- minet::validate(weightMatrix, true_adj_mat)
```

```
## Warning in minet::validate(weightMatrix, true_adj_mat): infered network arcs
## will be consider as undirected edges
```

```
minet::show.roc(genie, col="green", type="b")
```

```
## pdf
##   4
```

```
plot(g, edge.width=E(g)$weight, main="GENIE3")
```