Bioinformatics Project

wangjun

2023-09-22

# Data import and preliminary analysis

Import data set and check whether successful or not:

# import data set  
data <- read.csv("BRCA\_RNASeqv2\_top50.csv")

Then confirm the distribution of missing values and process them if they exist:

# View the number of missing values in each column  
missing\_values\_per\_column <- sapply(data, function(x) sum(is.na(x)))  
  
# Output the number of missing values in each column  
missing\_values\_per\_column

## FIGF LYVE1 CD300LG SCARA5 PAMR1 SDPR MYOM1 BTNL9   
## 0 0 0 0 0 0 0 0   
## KCNIP2 SLC2A4 PDE2A LEP ACVR1C ABCA10 AQP7 GPR146   
## 0 0 0 0 0 0 0 0   
## ATP1A2 FXYD1 ARHGAP20 NPR1 ATOH8 ABCA9 ALDH1L1 ADAMTS5   
## 0 0 0 0 0 0 0 0   
## RDH5 GPAM CA4 KLHL29 GPIHBP1 LOC728264 MAMDC2 TMEM132C   
## 0 0 0 0 0 0 0 0   
## ITIH5 HSPB7 HSPB6 DMD SPRY2 IGFBP6 CXCL2 EBF1   
## 0 0 0 0 0 0 0 0   
## KLB CLEC3B TMEM220 IBSP HIF3A IGSF10 CIDEC C2orf40   
## 0 0 0 0 0 0 0 0   
## LEPR ANGPTL1 class   
## 0 0 0

Judging from the output results, there are no missing values in the data set, but here we need to consider the expression of missing values other than NA:

# Custom function to identify missing values  
is\_missing <- function(x) {  
 return(is.na(x) |   
 x == "None" |   
 is.nan(x) |   
 x == "" |   
 x == "N/A" |   
 x == "-" |   
 x == "Null")  
}  
  
# View the number of missing values in each column  
missing\_values\_per\_column <- sapply(data, function(x) sum(is\_missing(x)))  
  
# Output the number of missing values in each column  
missing\_values\_per\_column

## FIGF LYVE1 CD300LG SCARA5 PAMR1 SDPR MYOM1 BTNL9   
## 0 0 0 0 0 0 0 0   
## KCNIP2 SLC2A4 PDE2A LEP ACVR1C ABCA10 AQP7 GPR146   
## 0 0 0 0 0 0 0 0   
## ATP1A2 FXYD1 ARHGAP20 NPR1 ATOH8 ABCA9 ALDH1L1 ADAMTS5   
## 0 0 0 0 0 0 0 0   
## RDH5 GPAM CA4 KLHL29 GPIHBP1 LOC728264 MAMDC2 TMEM132C   
## 0 0 0 0 0 0 0 0   
## ITIH5 HSPB7 HSPB6 DMD SPRY2 IGFBP6 CXCL2 EBF1   
## 0 0 0 0 0 0 0 0   
## KLB CLEC3B TMEM220 IBSP HIF3A IGSF10 CIDEC C2orf40   
## 0 0 0 0 0 0 0 0   
## LEPR ANGPTL1 class   
## 0 0 0

Judging from the output results, there are indeed no missing values expressed in any form in the data set, so there is no need to process missing values and the data set can be used directly.

summary(data)

## FIGF LYVE1 CD300LG SCARA5   
## Min. : 0.000 Min. : 0.00 Min. : 0.000 Min. : 0.000   
## 1st Qu.: 2.197 1st Qu.: 17.45 1st Qu.: 3.102 1st Qu.: 3.314   
## Median : 6.385 Median : 32.65 Median : 15.084 Median : 18.669   
## Mean : 84.983 Mean : 257.07 Mean : 211.590 Mean : 246.129   
## 3rd Qu.: 30.594 3rd Qu.: 89.26 3rd Qu.: 85.894 3rd Qu.: 92.496   
## Max. :2284.338 Max. :11111.93 Max. :6292.725 Max. :11533.028   
## PAMR1 SDPR MYOM1 BTNL9   
## Min. : 0.467 Min. : 2.806 Min. : 0.88 Min. : 0.833   
## 1st Qu.: 35.774 1st Qu.: 46.249 1st Qu.: 15.79 1st Qu.: 33.151   
## Median : 76.714 Median : 106.540 Median : 30.60 Median : 78.672   
## Mean : 243.117 Mean : 484.364 Mean : 163.42 Mean : 297.891   
## 3rd Qu.: 191.350 3rd Qu.: 348.140 3rd Qu.: 63.54 3rd Qu.: 207.829   
## Max. :4442.947 Max. :11292.082 Max. :76348.50 Max. :8577.470   
## KCNIP2 SLC2A4 PDE2A LEP   
## Min. : 0.00 Min. : 0.000 Min. : 8.834 Min. : 0.00   
## 1st Qu.: 13.08 1st Qu.: 7.591 1st Qu.: 65.868 1st Qu.: 1.73   
## Median : 31.46 Median : 17.038 Median : 121.833 Median : 14.51   
## Mean : 330.79 Mean : 96.034 Mean : 287.571 Mean : 786.65   
## 3rd Qu.: 101.47 3rd Qu.: 42.093 3rd Qu.: 274.830 3rd Qu.: 101.01   
## Max. :17898.33 Max. :10459.152 Max. :4116.836 Max. :47936.24   
## ACVR1C ABCA10 AQP7 GPR146   
## Min. : 0.000 Min. : 0.000 Min. : 0.00 Min. : 7.44   
## 1st Qu.: 8.844 1st Qu.: 4.824 1st Qu.: 2.85 1st Qu.: 53.51   
## Median : 21.187 Median : 12.093 Median : 15.29 Median : 83.88   
## Mean : 197.546 Mean : 55.358 Mean : 235.92 Mean : 160.22   
## 3rd Qu.: 59.182 3rd Qu.: 42.140 3rd Qu.: 87.05 3rd Qu.: 151.82   
## Max. :15144.774 Max. :861.533 Max. :11613.34 Max. :2849.61   
## ATP1A2 FXYD1 ARHGAP20 NPR1   
## Min. : 0.00 Min. : 0.00 Min. : 0.4965 Min. : 5.192   
## 1st Qu.: 7.53 1st Qu.: 4.24 1st Qu.: 20.5644 1st Qu.: 78.937   
## Median : 28.63 Median : 16.15 Median : 41.1099 Median : 145.770   
## Mean : 284.64 Mean : 85.89 Mean : 87.1257 Mean : 388.295   
## 3rd Qu.: 92.08 3rd Qu.: 64.75 3rd Qu.: 83.1719 3rd Qu.: 298.952   
## Max. :58904.34 Max. :4385.21 Max. :1705.8824 Max. :9271.230   
## ATOH8 ABCA9 ALDH1L1 ADAMTS5   
## Min. : 0.5924 Min. : 0.00 Min. : 0.000 Min. : 3.692   
## 1st Qu.: 13.8957 1st Qu.: 18.83 1st Qu.: 3.215 1st Qu.: 116.083   
## Median : 30.1735 Median : 53.81 Median : 12.918 Median : 231.296   
## Mean : 103.0018 Mean : 195.25 Mean : 168.590 Mean : 500.670   
## 3rd Qu.: 82.2975 3rd Qu.: 154.94 3rd Qu.: 64.642 3rd Qu.: 457.266   
## Max. :2357.1137 Max. :5814.01 Max. :18530.408 Max. :8410.825   
## RDH5 GPAM CA4 KLHL29   
## Min. : 1.609 Min. : 77.66 Min. : 0.0000 Min. : 3.671   
## 1st Qu.: 12.637 1st Qu.: 297.87 1st Qu.: 0.0000 1st Qu.: 32.181   
## Median : 26.371 Median : 448.50 Median : 0.8031 Median : 61.969   
## Mean : 141.050 Mean : 1629.26 Mean : 49.4441 Mean : 133.980   
## 3rd Qu.: 72.489 3rd Qu.: 694.04 3rd Qu.: 9.0870 3rd Qu.: 130.499   
## Max. :4722.437 Max. :86428.08 Max. :2345.7160 Max. :1682.311   
## GPIHBP1 LOC728264 MAMDC2 TMEM132C   
## Min. : 0.00 Min. : 4.409 Min. : 0.00 Min. : 0.000   
## 1st Qu.: 15.04 1st Qu.: 62.645 1st Qu.: 30.23 1st Qu.: 2.149   
## Median : 39.37 Median : 122.047 Median : 74.32 Median : 14.865   
## Mean : 133.88 Mean : 393.760 Mean : 198.83 Mean : 139.027   
## 3rd Qu.: 114.95 3rd Qu.: 274.893 3rd Qu.: 187.22 3rd Qu.: 70.364   
## Max. :3996.41 Max. :7221.175 Max. :4074.26 Max. :6837.168   
## ITIH5 HSPB7 HSPB6 DMD   
## Min. : 3.993 Min. : 0.00 Min. : 0.00 Min. : 1.538   
## 1st Qu.: 126.200 1st Qu.: 18.77 1st Qu.: 70.86 1st Qu.: 85.545   
## Median : 257.909 Median : 40.28 Median : 172.99 Median : 178.876   
## Mean : 994.826 Mean : 309.46 Mean : 1076.23 Mean : 527.454   
## 3rd Qu.: 670.645 3rd Qu.: 93.77 3rd Qu.: 539.07 3rd Qu.: 465.328   
## Max. :30476.583 Max. :44881.59 Max. :81041.37 Max. :15983.971   
## SPRY2 IGFBP6 CXCL2 EBF1   
## Min. : 13.63 Min. : 4.438 Min. : 0.000 Min. : 6.39   
## 1st Qu.: 111.79 1st Qu.: 75.485 1st Qu.: 5.498 1st Qu.: 100.18   
## Median : 198.26 Median : 147.285 Median : 17.584 Median : 173.87   
## Mean : 360.58 Mean : 351.459 Mean : 91.627 Mean : 358.76   
## 3rd Qu.: 387.96 3rd Qu.: 287.143 3rd Qu.: 61.339 3rd Qu.: 314.75   
## Max. :3602.64 Max. :12035.628 Max. :3290.953 Max. :6512.31   
## KLB CLEC3B TMEM220 IBSP   
## Min. : 0.00 Min. : 2.174 Min. : 3.618 Min. : 0.00   
## 1st Qu.: 11.49 1st Qu.: 76.331 1st Qu.: 26.863 1st Qu.: 3.99   
## Median : 22.29 Median : 146.489 Median : 49.115 Median : 18.06   
## Mean : 123.70 Mean : 463.011 Mean : 78.556 Mean : 126.85   
## 3rd Qu.: 47.91 3rd Qu.: 322.470 3rd Qu.: 89.731 3rd Qu.: 58.11   
## Max. :5352.45 Max. :14579.682 Max. :736.621 Max. :88358.72   
## HIF3A IGSF10 CIDEC C2orf40   
## Min. : 0.000 Min. : 0.00 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 1.998 1st Qu.: 24.45 1st Qu.: 3.34 1st Qu.: 3.93   
## Median : 5.845 Median : 67.64 Median : 33.22 Median : 19.62   
## Mean : 48.872 Mean : 197.19 Mean : 715.63 Mean : 142.42   
## 3rd Qu.: 24.567 3rd Qu.: 170.28 3rd Qu.: 221.10 3rd Qu.: 95.86   
## Max. :2282.147 Max. :6666.12 Max. :32690.62 Max. :6933.38   
## LEPR ANGPTL1 class   
## Min. : 8.036 Min. : 0.00 Length:1212   
## 1st Qu.: 95.140 1st Qu.: 16.13 Class :character   
## Median : 173.602 Median : 37.75 Mode :character   
## Mean : 391.079 Mean : 110.21   
## 3rd Qu.: 373.135 3rd Qu.: 89.81   
## Max. :8709.717 Max. :1919.26

The above output gives the statistical results for each variable. It is not difficult to see that the values of these variables are continuous values rather than discrete values, which will affect the parameter selection during subsequent modeling.

# Using Causal Structure Learning Algorithm To Find The Gene Regulatory Network

Causal structure learning seeks to infer the causal relationships between variables based on observational data. Instead of just determining associations, it aims to discern the directionality of these associations, which is fundamental in fields like medicine, economics, and social sciences where understanding causality can lead to effective interventions

The PC algorithm, attributed to Peter Spirtes and Clark Glymour, stands as a cornerstone in the domain of causal structure learning. Initially conceptualized to decipher the causal relationships amidst a set of variables, it functions by leveraging statistical independence tests.

The PC algorithm initiates its process with a graph that is inherently undirected and fully connected. The primary objective during its early phase is the systematic removal of edges. This elimination is predicated upon the discernment of conditional independencies among the variables under consideration. Specifically, when two distinct variables exhibit conditional independence contingent upon a subset of other variables, the edge that interlinks them is expunged [1]. Upon the delineation of this skeletal structure, the algorithm subsequently ventures into the intricate task of determining the orientation of the residual edges. This determination is effectuated through the application of a compendium of rules, all of which are anchored in the observed conditional independencies. Illustratively, within a triadic structure such as A-B-C, if variable A manifests independence from variable C when conditioned on B, and simultaneously, B is not identified as a collider (a distinct nodal point where two directed edges intersect), the edge juxtaposing B and C is oriented as B->C [1].

A pivotal assumption underpinning the PC algorithm is the “faithfulness condition”. This posits that all observed statistical independencies in the dataset correspond unerringly to independencies within the true underlying causal structure, and vice-versa [1].

Understanding and interpreting the outcome of the PC algorithm mandates caution. While it proffers vital insights into potential causal relationships, it doesn’t serve as a definitive proof of causality, especially since the algorithm’s conclusions are contingent upon its foundational assumptions and the quality of the data at hand.

First, remove variables of type class from the data set.

data\_no\_class <- data[, !names(data) %in% "class"]

The class variable as the target has been removed from the data set. Now only features are left in the data set. Then the pc algorithm is used for modeling, and the potential causal relationship between variables can be observed in the data.

library(pcalg)  
  
# Calculate correlation matrix  
cor\_matrix <- cor(data\_no\_class)  
  
# Estimating structure using PC algorithm  
suffStat <- list(C = cor\_matrix, n = nrow(data\_no\_class))  
pc\_result <- pc(suffStat, indepTest = gaussCItest, alpha = 0.05,  
 labels = colnames(data\_no\_class))

The pcalg package is employed to discern potential causal relationships among variables in the data\_no\_class dataset using the PC algorithm. Initially, the Pearson correlation matrix of the dataset is computed with the cor function. To execute the PC algorithm, a requisite sufficiency statistic, comprising this correlation matrix and the total number of observations, is prepared. The pc function, subsequently invoked, leverages the gaussCItest for conditional independence tests at a significance level of and utilizes dataset column names as node labels in the resultant graph. The outcome is a partially directed acyclic graph representing inferred causal dependencies among the dataset’s variables.

# output the trained model  
pc\_result

## Object of class 'pcAlgo', from Call:  
## pc(suffStat = suffStat, indepTest = gaussCItest, alpha = 0.05,   
## labels = colnames(data\_no\_class))  
## Number of undirected edges: 1   
## Number of directed edges: 110   
## Total number of edges: 111

The output is an overview of the pc function in the pcalg package applied to the data\_no\_class data set. The output shows that based on the execution of the PC algorithm, the resulting partially directed acyclic graph (PDAG) has 1 undirected edge (that is, the direction of the relationship between the two variables has not yet been determined), and 110 directed edges. edges (i.e., the direction of the relationship between two variables is specified), so there are 111 edges in total. These edges represent potential causal relationships or correlations between variables in the dataset.

For more intuitive observation, a visualization of the results is provided.

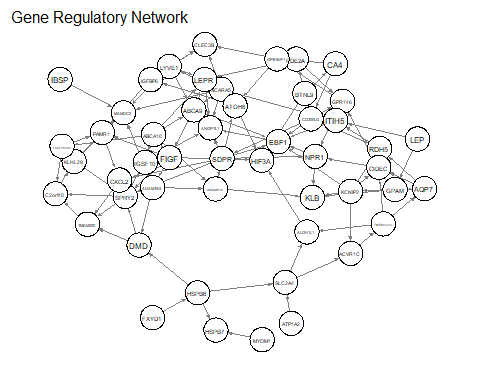


Figure 1: Gene Regulatory Network

From the visual examination, it can be inferred that the pc\_result largely embodies the characteristics of a directed acyclic graph. However, an ambiguity in directional precedence between the variables MYCM1 and ATP1A2 introduces a loop, indicating an indeterminacy in their interrelationship.

# Find The Top 10 Other Genes That Have Strong Causal Effects On EBF1

The primary objective of causal learning is to discern the entire causal structure or causal graph from data. In contrast, causal inference aims to estimate specific causal effects, given an established causal framework. The PC algorithm primarily functions as a causal learning tool, its principal goal being the elucidation of the causal architecture within the data.

Having previously ascertained this structure via the algorithm, one can now, based on the trained pc\_result, extract the top 10 genes that exert substantial causal effects on EBF1.

First convert pc\_result from the causal relationship structure to an adjacency matrix.

adjMatrix <- as(pc\_result, "amat")

Then find the position of EBF1 in the data and find the genes directly connected to EBF1.

ebf1\_index <- which(colnames(data\_no\_class) == "EBF1")  
ebf1\_neighbors <- which(adjMatrix[ebf1\_index, ] != 0)

The absolute value of the causal relationship between these genes and EBF1 is then calculated.

causal\_effects <- abs(cor(data\_no\_class)[ebf1\_neighbors, "EBF1"])

The correlation coefficient matrix cor(data\_no\_class) is used here to estimate the causal effect between each gene and EBF1, and its absolute value is taken.

Finally, obtain the gene name corresponding to each causal effect, merge the gene name and the corresponding causal effect into a data frame, then sort the causal effects and obtain the top 10, and then print the results.

# Get the gene name of the causal effect  
genes\_with\_effects <- rownames(cor(data\_no\_class)[ebf1\_neighbors, ])  
  
# Merge gene names and their corresponding effects into a data frame  
effects\_df <- data.frame(Genes = genes\_with\_effects, Effects = causal\_effects)  
  
# Rank causal effects and get the top 10  
top\_genes <- head(effects\_df[order(-effects\_df$Effects),], 10)  
  
# print result  
top\_genes

## Genes Effects  
## NPR1 NPR1 0.9189766  
## KCNIP2 KCNIP2 0.9187560  
## CD300LG CD300LG 0.9145221  
## SDPR SDPR 0.8822326  
## ITIH5 ITIH5 0.8733961  
## ABCA9 ABCA9 0.8459658

Judging from the output results, the number of genes that “have strong causal effects on EBF1” is only 6. Although these 6 genes have high values, they do not meet the demand in terms of quantity.

Therefore, in addition to considering directly connected genes, a second degree of connectivity needs to be considered, i.e., considering genes that are two edges away (i.e., genes that are connected to the genes which are directly connected to EBF1).

The first is to calculate directly connected genes.

# Extract the adjacency matrix  
adj\_matrix <- as(pc\_result, "amat")  
  
# Find directly connected genes to EBF1  
direct\_genes <- which(adj\_matrix["EBF1",] == 1)  
  
# Find the names of these genes  
direct\_gene\_names <- colnames(adj\_matrix)[direct\_genes]  
  
direct\_gene\_names

## [1] "CD300LG" "SDPR" "KCNIP2" "NPR1" "ABCA9" "ITIH5"

Judging from the output, this will produce the same results as direct processing.

The next additional calculation that needs to be done is to find second\_degree\_genes.

First, an empty vector is created to store other genes connected to the direct gene, that is, secondary genes.

second\_degree\_genes <- vector()

Then find the secondary genes connected to the direct genes.

for (gene in direct\_gene\_names) {  
 tmp\_genes <- which(adj\_matrix[gene,] == 1)  
 tmp\_gene\_names <- setdiff(colnames(adj\_matrix)[tmp\_genes], c(direct\_gene\_names, "EBF1"))  
 second\_degree\_genes <- unique(c(second\_degree\_genes, tmp\_gene\_names))  
}

For each gene directly linked to EBF1, the loop finds all genes linked to it and ensures that these do not include known direct genes or EBF1 itself.

Then merge direct genes and secondary genes.

all\_candidate\_genes <- c(direct\_gene\_names, second\_degree\_genes)

Finally, for each candidate gene (direct or secondary gene), use the sapply function to calculate its correlation with EBF1 and take its absolute value, sort the correlations and extract the top 10 genes.

correlations <- sapply(all\_candidate\_genes, function(gene) {  
 abs(cor(data\_no\_class[, gene], data\_no\_class[, "EBF1"]))  
})  
  
ranked\_genes <- names(sort(correlations, decreasing = TRUE))  
  
# Take the top 10 genes  
top\_10\_genes <- ranked\_genes[1:10]  
  
print(top\_10\_genes)

## [1] "NPR1" "KCNIP2" "CD300LG" "RDH5" "CIDEC" "GPR146" "BTNL9"   
## [8] "SDPR" "GPAM" "ITIH5"

From the output results, it can be found that in addition to the 6 genes with direct connections, 4 genes with a second degree of connectivity were found.

Check how strongly the 10 selected genes are related to EBF1.

selected\_genes <- c("NPR1", "KCNIP2", "CD300LG", "RDH5", "CIDEC", "GPR146", "BTNL9", "SDPR", "GPAM", "ITIH5")  
  
# Calculate correlation with EBF1  
correlations <- sapply(selected\_genes, function(gene) {  
 cor(data\_no\_class[, gene], data\_no\_class[, "EBF1"])  
})  
  
# Create a data frame to display the results  
correlation\_df <- data.frame(Gene = selected\_genes, Correlation\_with\_EBF1 = correlations)  
  
correlation\_df

## Gene Correlation\_with\_EBF1  
## NPR1 NPR1 0.9189766  
## KCNIP2 KCNIP2 0.9187560  
## CD300LG CD300LG 0.9145221  
## RDH5 RDH5 0.8985167  
## CIDEC CIDEC 0.8973219  
## GPR146 GPR146 0.8956943  
## BTNL9 BTNL9 0.8943198  
## SDPR SDPR 0.8822326  
## GPAM GPAM 0.8802229  
## ITIH5 ITIH5 0.8733961

This result shows the correlation between 10 selected genes and the EBF1 gene. The value of correlation ranges from -1 to 1, where 1 means a perfect positive correlation, -1 means a perfect negative correlation, and 0 means no linear correlation.

From the results, the following observations can be seen: all these genes show a high positive correlation with EBF1. The correlation between NPR1 and EBF1 is the highest, about 0.919. This means that when the expression of NPR1 increases, the expression of EBF1 is likely to increase as well, and vice versa. The correlation between ITIH5 and EBF1 is the lowest among these 10 genes, but it is still relatively high, about 0.873.

The correlation values of all genes ranged from 0.87 to 0.92, indicating a strong linear relationship between the expression patterns of these genes and EBF1.

This observation is grounded in correlation and cannot be directly inferred as causation. Nevertheless, when one takes into account the application of the PC algorithm for the construction of a causal network and the subsequent selection of genes most closely associated with EBF1 based on this network, it indeed provides a foundational basis for causal analysis.

# Find Genes In The Markov Blanket of ABCA9 From Data

IAMB (Incremental Association Markov Blanket) is a Bayesian network structure learning algorithm. This algorithm aims to discover the Markov blanket for each node, which is the set of all nodes related to a given node, conditional on all other nodes. The IAMB algorithm incrementally adds or removes candidate parent nodes to determine the optimal Markov blanket for a node.

The IAMB (Incremental Association Markov Blanket) algorithm operates as follows: Initially, it initializes an empty Markov blanket for each node in the Bayesian network. Subsequently, the algorithm iteratively refines the Markov blanket for a target node. During the incremental addition of parent nodes, it evaluates the conditional independence of the target node with other nodes, selecting a candidate parent node that exhibits the highest conditional independence and incorporating it into the Markov blanket. Following each addition, the algorithm re-evaluates conditional independence, considering the possibility of further parent node additions. Concurrently, in the incremental removal of parent nodes, the algorithm assesses the conditional independence between the target node and its existing parent nodes. If removing a parent node does not diminish the conditional independence, it is removed from the Markov blanket. This addition and removal process is repeated iteratively until no more parent nodes can be added or removed, resulting in the determination of the optimal Markov blanket for the target node. In this manner, the IAMB algorithm efficiently constructs a compact Bayesian network structure by balancing conditional independence relationships[2].

The core idea of the IAMB algorithm is to determine the parent nodes of a node by calculating conditional independence, aiming to minimize the size of the Markov blanket while preserving conditional independence relationships, thus facilitating the construction of a concise Bayesian network structure for efficient probabilistic inference.

library(bnlearn)

##   
## Attaching package: 'bnlearn'

## The following objects are masked from 'package:pcalg':  
##   
## dsep, pdag2dag, shd, skeleton

nvar <- ncol(data\_no\_class)  
#learn the markov blanket  
MB.ABCA9=learn.mb(data\_no\_class, "ABCA9", method="iamb", alpha=0.01)  
MB.ABCA9

## [1] "EBF1" "ABCA10" "SCARA5" "ACVR1C" "CD300LG" "LYVE1"   
## [7] "GPAM" "FIGF" "LEPR" "LOC728264" "TMEM132C" "HIF3A"   
## [13] "LEP" "ANGPTL1" "PAMR1" "CLEC3B" "GPIHBP1" "KLB"   
## [19] "ATOH8" "RDH5" "NPR1" "CIDEC"

The above output results are determined to be markov blankets belonging to ABCA9, ABCA9 is related or potentially interacts with these genes in biological processes.

# Discretization Of Data

Discretize the dataset into a binary format using the mean expression level of all genes as the delineation threshold.

# Calculate the mean of each column  
column\_means <- colMeans(data\_no\_class)  
  
# Calculate global mean  
global\_mean <- mean(column\_means)  
  
# Print the mean of each column  
print(global\_mean)

## [1] 304.7847

Then proceed to discretization.

# Binarization using global mean as threshold  
binary\_data <- data\_no\_class > global\_mean  
  
# Convert TRUE to 1 and FALSE to 0  
binary\_data <- ifelse(binary\_data, 1, 0)

# Naive Bayes Model

To make predictions, create a new data frame including binary\_data and class columns.

combined\_data <- cbind(binary\_data, class = ifelse(data$class == "C", 1, 0))

The PC-simple algorithm, often called pcSelect, is a constraint-based causal structure learning algorithm. It is a simplified version of the original PC (Peter-Clark) algorithm. The PC algorithm aims to start from an undirected graph and progressively remove edges until certain conditions are met to obtain a directed acyclic graph (DAG).

First, pcSelect uses a complete graph (with edges between every pair of nodes) as a starting point. Then, it checks the conditional independence between each pair of nodes. Delete the edge between two variables if they are conditionally independent given some other variables. Finally, it orients certain edges in the undirected graph, resulting in a DAG. This is done by considering the nodes’ neighbors on the remaining edges.

Unlike the standard PC algorithm, pcSelect does not increase the size of the condition set in step 2, which makes the algorithm faster, but potentially also less accurate.

pcS <- pcSelect(combined\_data[,"class"], binary\_data, alpha=0.05)  
pcS

## $G  
## FIGF LYVE1 CD300LG SCARA5 PAMR1 SDPR MYOM1 BTNL9   
## TRUE FALSE TRUE TRUE FALSE FALSE FALSE FALSE   
## KCNIP2 SLC2A4 PDE2A LEP ACVR1C ABCA10 AQP7 GPR146   
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE   
## ATP1A2 FXYD1 ARHGAP20 NPR1 ATOH8 ABCA9 ALDH1L1 ADAMTS5   
## TRUE FALSE TRUE FALSE TRUE FALSE FALSE FALSE   
## RDH5 GPAM CA4 KLHL29 GPIHBP1 LOC728264 MAMDC2 TMEM132C   
## FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE   
## ITIH5 HSPB7 HSPB6 DMD SPRY2 IGFBP6 CXCL2 EBF1   
## FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE   
## KLB CLEC3B TMEM220 IBSP HIF3A IGSF10 CIDEC C2orf40   
## FALSE FALSE TRUE FALSE FALSE FALSE FALSE TRUE   
## LEPR ANGPTL1   
## FALSE FALSE   
##   
## $zMin  
## [1] 11.84933190 1.75343173 7.98927318 2.78834345 0.84585960 1.26532234  
## [7] 1.56460098 0.66106380 0.17631320 1.90355728 0.77811144 1.39229163  
## [13] 0.09681083 1.18874762 1.28826218 1.19764236 4.96065926 1.84861317  
## [19] 10.72296828 1.33517180 2.66133750 1.84932203 1.92996580 1.75416443  
## [25] 0.85714220 0.68162567 1.34363062 6.71865988 1.80971737 1.94231385  
## [31] 3.99081592 1.39481527 0.72778271 1.92406445 0.48799378 1.95912820  
## [37] 1.91535988 0.50045810 7.44130670 1.30023202 1.90570040 0.01797574  
## [43] 4.35552257 1.70756766 1.36671392 1.54527631 1.26434812 2.20334898  
## [49] 1.20563823 1.12238863

In the output derived from the pcSelect algorithm, the $G component enumerates each feature or variable with a corresponding boolean value. A value of TRUE designates the selection of that particular feature, suggesting its significance in the model, while a FALSE value indicates its exclusion. Conversely, the $zMin component provides a numerical vector, representing the statistical scores or importance measures for each feature. For instance, a value of 11.84933190 for the feature FIGF suggests a potent association with the response variable, hence its inclusion in the model.

selected\_vars <- names(pcS$G)[pcS$G == TRUE]  
selected\_vars

## [1] "FIGF" "CD300LG" "SCARA5" "ATP1A2" "ARHGAP20" "ATOH8"   
## [7] "KLHL29" "MAMDC2" "CXCL2" "TMEM220" "C2orf40"

The Naive Bayes algorithm is a classification algorithm based on Bayes’ theorem and feature independence assumption [3]. It is often used for text classification, spam filtering, and other classification tasks.

The basis of the Naive Bayes classifier is Bayes’ theorem, which has the form:

. In text classification, A may represent a specific category, while B represents a given text or a specific word [4]. The core idea of the algorithm is to calculate the probability that the text belongs to each category given a text (or feature), and then select the category with the maximum probability.

To calculate these probabilities, the following formula is used:

. Among them, is the k-th category, and x is a feature vector. Due to the feature independence assumption, can be split into the product of the probabilities of individual features [3]:

First, calculate the prior probability for each category. Next, for a given text or feature, the likelihood for each category is calculated. Then, calculate the posterior probability for each category given the features. Finally, the category corresponding to the largest posterior probability is selected as the prediction result [4].

Since only 112 samples are of normal cases in the data set and the remaining 1100 samples are cancer patients, in order to avoid not including normal cases when creating the test data set, the two are extracted separately here.

combined\_data <- as.data.frame(combined\_data)  
combined\_data$class <- as.factor(combined\_data$class)  
  
data\_class\_0 <- combined\_data[combined\_data$class == 0, ]  
data\_class\_1 <- combined\_data[combined\_data$class == 1, ]

Then start using the split data to create a test set and a training set:

set.seed(123)  
  
train\_data\_class\_0 <- data\_class\_0[sample(1:nrow(data\_class\_0), 0.7 \* nrow(data\_class\_0)), ]  
train\_data\_class\_1 <- data\_class\_1[sample(1:nrow(data\_class\_1), 0.7 \* nrow(data\_class\_1)), ]  
  
train\_data <- rbind(train\_data\_class\_0, train\_data\_class\_1)  
  
test\_data <- combined\_data[!rownames(combined\_data) %in% rownames(train\_data), ]

Then start training the Naive Bayes model:

library(e1071)

##   
## Attaching package: 'e1071'

## The following object is masked from 'package:bnlearn':  
##   
## impute

# Use train\_data for training  
model <- naiveBayes(class ~ ., data=train\_data)

Use the model to make predictions.

predictions <- predict(model, newdata = test\_data[,-which(names(test\_data) == "class")])

Evaluate model accuracy.

conf\_matrix <- table(predictions, test\_data$class)  
print(conf\_matrix)

##   
## predictions 0 1  
## 0 34 12  
## 1 0 318

accuracy <- sum(diag(conf\_matrix)) / sum(conf\_matrix)  
print(paste("Accuracy: ", round(accuracy, 3)))

## [1] "Accuracy: 0.967"

This classifier performed very well on the test data set, with very few samples being misclassified, resulting in a high accuracy of 96.7%.

Now instead of using all the variables, we use the selected variables in pcS to build the naive Bayes model.

train\_data\_selected <- train\_data[, c("class", selected\_vars)]  
test\_data\_selected <- test\_data[, c("class", selected\_vars)]  
  
model\_pc <- naiveBayes(class ~ ., data=train\_data\_selected)  
predictions\_selected <- predict(model\_pc, newdata = test\_data\_selected[,-which(names(test\_data\_selected) == "class")])  
  
conf\_matrix\_selected <- table(predictions\_selected, test\_data\_selected$class)  
print(conf\_matrix\_selected)

##   
## predictions\_selected 0 1  
## 0 34 2  
## 1 0 328

accuracy\_selected <- sum(diag(conf\_matrix\_selected)) / sum(conf\_matrix\_selected)  
print(paste("Accuracy: ", round(accuracy\_selected, 3)))

## [1] "Accuracy: 0.995"

The accuracy is 0.995, or 99.5%. This means that on this test set, the classifier performed very well, with only a small proportion of samples being misclassified.

Compared with the previous full-feature classifier (accuracy rate 96.7%), the accuracy of the classifier after using selected\_vars for feature selection has improved. This also demonstrates the importance of feature selection to improve model performance.

# Question 5

# References

[1] Spirtes, P., Glymour, C. N., & Scheines, R. (2000). Causation, Prediction, and Search. MIT press.

[2]

[3] A. McCallum and K. Nigam, “A comparison of event models for Naive Bayes text classification”, AAAI-98 workshop on learning for text categorization, vol. 752, pp. 41-48, 1998.

[4] D. D. Lewis, “Naive (Bayes) at forty: The independence assumption in information retrieval”, European conference on machine learning, Springer, Berlin, Heidelberg, pp. 4-15, 1998