Bioinformatics Project

Wangjun Shen 110248810 The University of South Australia

2023-09-19

Data Import And Preliminary Analysis

Import data set and check whether successful or not:

```r
# import data set
data <- read.csv("BRCA_RNASeqv2_top50.csv")
```

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

```r
# 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:

```r
# 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.

```r
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.

```r
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 $\alpha = 0.05$ 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.

```r
# 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.

For Figure 1, 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.

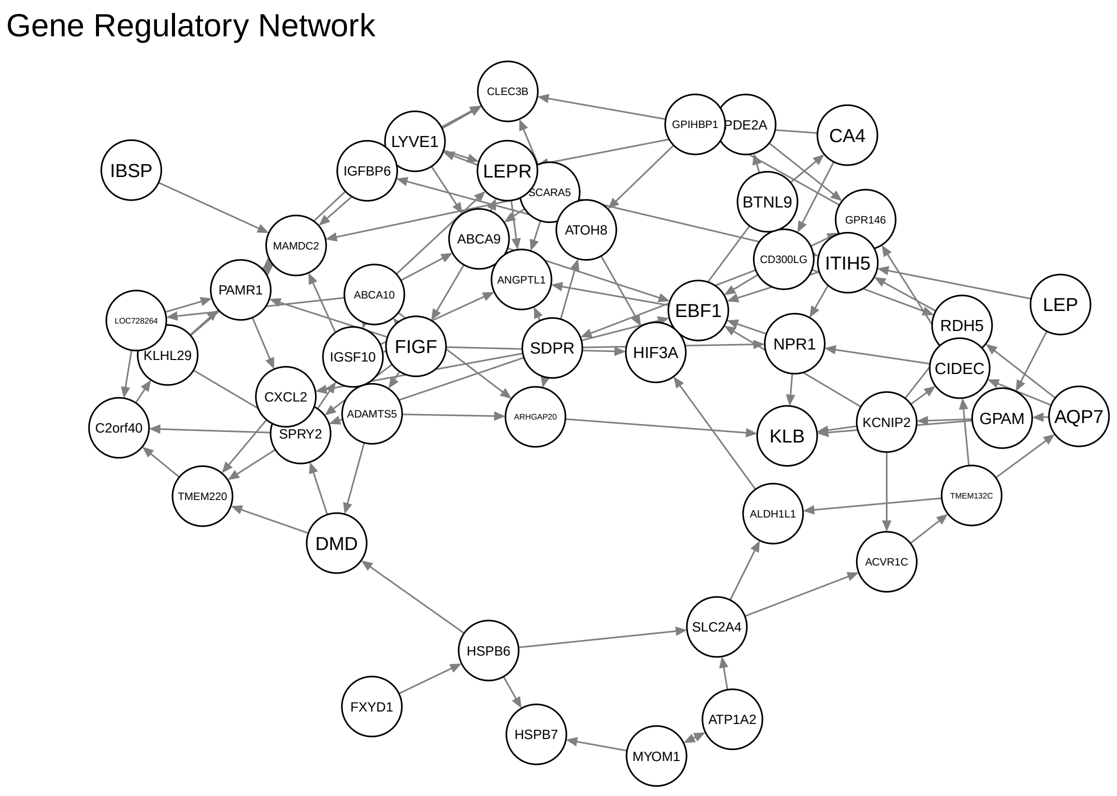

Figure 1: Gene Regulatory Network

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.

```r
# 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.

```r
second_degree_genes <- vector()
```

Then find the secondary genes connected to the direct genes.

```r
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.

```r
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.

```r
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.

```r
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.

```r
library(bnlearn)
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.

```r
# 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.

```r
# 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 For All Genes and Selected Genes

To make predictions, create a new data frame including binary_data and class columns.

```r
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, which makes the algorithm faster, but potentially also less accurate.

```r
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:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

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:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

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

$$P(x_1, x_2, ..., x_n|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$$

First, calculate the prior probability $P(C_k)$ for each category. Next, for a given text or feature, the likelihood $P(x|C_k)$ for each category is calculated. Then, calculate the posterior probability $P(C_k|x)$ 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:

```r
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:

```r
library(e1071)

# Use train_data for training
model <- naiveBayes(class ~ ., data=train_data)
```

Use the model to make predictions.

```r
predictions <- predict(model, newdata =
                         test_data[,-which(names(test_data) == "class")])
```

Evaluate model accuracy.

```r
conf_matrix <- table(predictions, test_data$class)
print(conf_matrix)
```

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

```r
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.

```r
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 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.

Bayesian Network Analysis of Gene Expression and Cancer Probability

The Bayesian network is given as figure 2.

The initial inquiry necessitates the formulation of conditional probability tables for the Bayesian network in accordance with the available dataset.

Constructing conditional probability tables (CPTs) for Bayesian networks involves calculating the conditional probability distribution of each node (variable). These distributions tell us what each node (variable) is, given its parent node (if any). The probability of possible values of a node.

First, determine the possible values of each node. For binary variables (such as 'class'), the possible values are 0 and 1. For other numeric nodes, the possible values can usually be determined based on unique values in the data. For each node, consider its parent node. If a node has no parent, then only its own marginal probability distribution is calculated. If a node has a parent node, then its own conditional probability distribution needs to be calculated given its parent node. When calculating conditional probability distributions, frequencies in the data to estimate probabilities can be used. For each possible value combination, calculate the frequency of a node taking a specific value under that combination and use it as a conditional probability.

First calculate the conditional Probability table of the BTNL9 node.

```
# Calculate the conditional probability table of BTNL9 node
# P(BTNL9 | parent nodes)

# Calculate P(BTNL9 = 0)
p_btnl9_0 <- sum(combined_data$BTNL9 == 0) / nrow(combined_data)

# Calculate P(BTNL9 = 1)
p_btnl9_1 <- sum(combined_data$BTNL9 == 1) / nrow(combined_data)
```

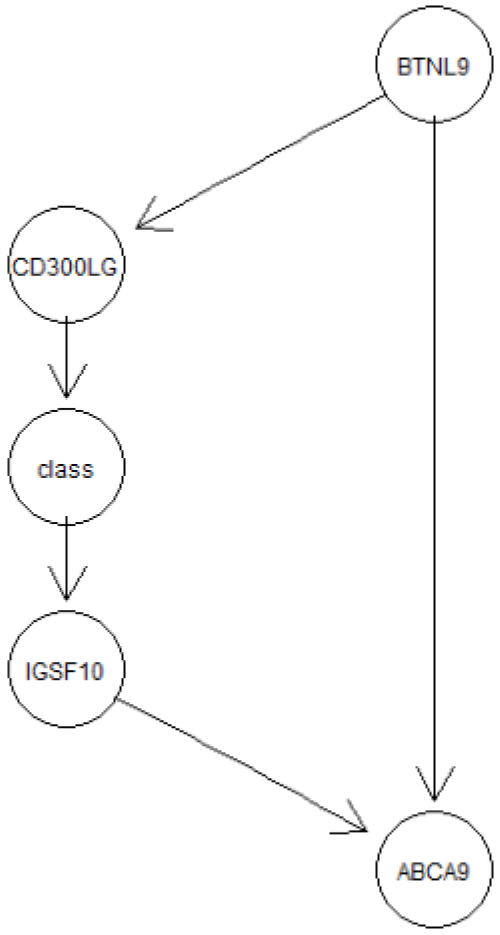

Figure 2: The Bayesian network

```r
# Create a data frame to store the probabilities
prob_table_btnl <- data.frame(
  BTNL9 = c(0, 1),
  Probability = c(p_btnl9_0, p_btnl9_1)
)

# Print the table
print(prob_table_btnl)
```

```
##   BTNL9 Probability
## 1     0   0.8209571
## 2     1   0.1790429
```

Then calculate the conditional Probability table of CD300LG.

```r
# Calculate the conditional probability table from BTNL9 to CD300LG
# P(CD300LG | BTNL9)

# Calculate P(CD300LG = 0 | BTNL9 = 0)
p_cd300lg_0_given_btnl9_0 <- sum(combined_data$CD300LG == 0 &
                                   combined_data$BTNL9 == 0) /
  sum(combined_data$BTNL9 == 0)

# Calculate P(CD300LG = 1 | BTNL9 = 0)
p_cd300lg_1_given_btnl9_0 <- sum(combined_data$CD300LG == 1 &
                                   combined_data$BTNL9 == 0) /
  sum(combined_data$BTNL9 == 0)

# Calculate P(CD300LG = 0 | BTNL9 = 1)
p_cd300lg_0_given_btnl9_1 <- sum(combined_data$CD300LG == 0 &
                                   combined_data$BTNL9 == 1) /
  sum(combined_data$BTNL9 == 1)

# Calculate P(CD300LG = 1 | BTNL9 = 1)
p_cd300lg_1_given_btnl9_1 <- sum(combined_data$CD300LG == 1 &
                                   combined_data$BTNL9 == 1) /
  sum(combined_data$BTNL9 == 1)

# Create a data frame to store the conditional probabilities for CD300LG given BTNL9
prob_table_cd300lg <- data.frame(
  BTNL9 = c(0, 0, 1, 1),
  CD300LG = c(0, 1, 0, 1),
  Probability = c(p_cd300lg_0_given_btnl9_0,
                  p_cd300lg_1_given_btnl9_0,
                  p_cd300lg_0_given_btnl9_1,
                  p_cd300lg_1_given_btnl9_1)
)

# Print the table
print(prob_table_cd300lg)
```

```
##   BTNL9 CD300LG Probability
## 1     0       0 0.991959799
## 2     0       1 0.008040201
## 3     1       0 0.359447005
## 4     1       1 0.640552995
```

Then calculate the Probability conditional table of the class.

```
# Calculate the conditional probability table of class nodes
# P(class | CD300LG)

# Calculate P(class = 0 | CD300LG = 0)
p_class_0_given_cd300lg_0 <- sum(combined_data$class == "0" &
                                   combined_data$CD300LG == 0) /
  sum(combined_data$CD300LG == 0)

# Calculate P(class = 1 | CD300LG = 0)
p_class_1_given_cd300lg_0 <- sum(combined_data$class == "1" &
                                   combined_data$CD300LG == 0) /
  sum(combined_data$CD300LG == 0)

# Calculate P(class = 0 | CD300LG = 1)
p_class_0_given_cd300lg_1 <- sum(combined_data$class == "0" &
                                   combined_data$CD300LG == 1) /
  sum(combined_data$CD300LG == 1)

# Calculate P(class = 1 | CD300LG = 1)
p_class_1_given_cd300lg_1 <- sum(combined_data$class == "1" &
                                   combined_data$CD300LG == 1) /
  sum(combined_data$CD300LG == 1)

# Create a data frame for the calculated probabilities
conditional_probabilities <- data.frame(
  CD300LG = c(0, 0, 1, 1),
  class = c(0, 1, 0, 1),
  Probability = c(p_class_0_given_cd300lg_0,
                  p_class_1_given_cd300lg_0,
                  p_class_0_given_cd300lg_1,
                  p_class_1_given_cd300lg_1)
)

# Print the data frame
print(conditional_probabilities)
```

```
##   CD300LG class Probability
## 1       0     0 0.002816901
## 2       0     1 0.997183099
## 3       1     0 0.741496599
## 4       1     1 0.258503401
```

Next is IGSF10.

```r
# Calculate the conditional probability table of IGSF10 node
# P(IGSF10 | class)

# Calculate P(IGSF10 = 0 | class = 0)
p_igsf10_0_given_class_0 <- sum(combined_data$IGSF10 == 0 &
                                   combined_data$class == "0") /
  sum(combined_data$class == "0")

# Calculate P(IGSF10 = 1 | class = 0)
p_igsf10_1_given_class_0 <- sum(combined_data$IGSF10 == 1 &
                                   combined_data$class == "0") /
  sum(combined_data$class == "0")

# Calculate P(IGSF10 = 0 | class = 1)
p_igsf10_0_given_class_1 <- sum(combined_data$IGSF10 == 0 &
                                   combined_data$class == "1") /
  sum(combined_data$class == "1")

# Calculate P(IGSF10 = 1 | class = 1)
p_igsf10_1_given_class_1 <- sum(combined_data$IGSF10 == 1 &
                                   combined_data$class == "1") /
  sum(combined_data$class == "1")

# Create a data frame to store the conditional probabilities for class given CD300LG
prob_table_class <- data.frame(
  CD300LG = c(0, 0, 1, 1),
  class = c(0, 1, 0, 1),
  Probability = c(p_class_0_given_cd300lg_0,
                  p_class_1_given_cd300lg_0,
                  p_class_0_given_cd300lg_1,
                  p_class_1_given_cd300lg_1)
)

# Print the table
print(prob_table_class)
```

```
##   CD300LG class Probability
## 1       0     0 0.002816901
## 2       0     1 0.997183099
## 3       1     0 0.741496599
## 4       1     1 0.258503401
```

The final one is ABCA9.

```r
# Calculate the conditional probability table of ABCA9 node
# P(ABCA9 | BTNL9, IGSF10)

# Calculate P(ABCA9 = 0 | BTNL9 = 0, IGSF10 = 0)
p_abca9_0_given_btnl9_0_igsf10_0 <- sum(combined_data$ABCA9 == 0 &
                                           combined_data$BTNL9 == 0 &
                                           combined_data$IGSF10 == 0) /
  sum(combined_data$BTNL9 == 0 & combined_data$IGSF10 == 0)
```

```r
# Calculate P(ABCA9 = 1 | BTNL9 = 0, IGSF10 = 0)
p_abca9_1_given_btnl9_0_igsf10_0 <- sum(combined_data$ABCA9 == 1 &
                                         combined_data$BTNL9 == 0 &
                                         combined_data$IGSF10 == 0) /
  sum(combined_data$BTNL9 == 0 & combined_data$IGSF10 == 0)

# Calculate P(ABCA9 = 0 | BTNL9 = 1, IGSF10 = 0)
p_abca9_0_given_btnl9_1_igsf10_0 <- sum(combined_data$ABCA9 == 0 &
                                         combined_data$BTNL9 == 1 &
                                         combined_data$IGSF10 == 0) /
  sum(combined_data$BTNL9 == 1 & combined_data$IGSF10 == 0)

# Calculate P(ABCA9 = 1 | BTNL9 = 1, IGSF10 = 0)
p_abca9_1_given_btnl9_1_igsf10_0 <- sum(combined_data$ABCA9 == 1 &
                                         combined_data$BTNL9 == 1 &
                                         combined_data$IGSF10 == 0) /
  sum(combined_data$BTNL9 == 1 & combined_data$IGSF10 == 0)

# Calculate P(ABCA9 = 0 | BTNL9 = 0, IGSF10 = 1)
p_abca9_0_given_btnl9_0_igsf10_1 <- sum(combined_data$ABCA9 == 0 &
                                         combined_data$BTNL9 == 0 &
                                         combined_data$IGSF10 == 1) /
  sum(combined_data$BTNL9 == 0 & combined_data$IGSF10 == 1)

# Calculate P(ABCA9 = 1 | BTNL9 = 0, IGSF10 = 1)
p_abca9_1_given_btnl9_0_igsf10_1 <- sum(combined_data$ABCA9 == 1 &
                                         combined_data$BTNL9 == 0 &
                                         combined_data$IGSF10 == 1) /
  sum(combined_data$BTNL9 == 0 & combined_data$IGSF10 == 1)

# Calculate P(ABCA9 = 0 | BTNL9 = 1, IGSF10 = 1)
p_abca9_0_given_btnl9_1_igsf10_1 <- sum(combined_data$ABCA9 == 0 &
                                         combined_data$BTNL9 == 1 &
                                         combined_data$IGSF10 == 1) /
  sum(combined_data$BTNL9 == 1 & combined_data$IGSF10 == 1)

# Calculate P(ABCA9 = 1 | BTNL9 = 1, IGSF10 = 1)
p_abca9_1_given_btnl9_1_igsf10_1 <- sum(combined_data$ABCA9 == 1 &
                                         combined_data$BTNL9 == 1 &
                                         combined_data$IGSF10 == 1) /
  sum(combined_data$BTNL9 == 1 & combined_data$IGSF10 == 1)

# Create a data frame to store the conditional probabilities for ABCA9 given BTNL9 and IGSF10
prob_table_abca9 <- data.frame(
  BTNL9 = c(0, 0, 1, 1, 0, 0, 1, 1),
  IGSF10 = c(0, 0, 0, 0, 1, 1, 1, 1),
  ABCA9 = c(0, 1, 0, 1, 0, 1, 0, 1),
  Probability = c(p_abca9_0_given_btnl9_0_igsf10_0,
                  p_abca9_1_given_btnl9_0_igsf10_0,
                  p_abca9_0_given_btnl9_1_igsf10_0,
                  p_abca9_1_given_btnl9_1_igsf10_0,
                  p_abca9_0_given_btnl9_0_igsf10_1,
                  p_abca9_1_given_btnl9_0_igsf10_1,
                  p_abca9_0_given_btnl9_1_igsf10_1,
                  p_abca9_1_given_btnl9_1_igsf10_1)
)

# Print the table
print(prob_table_abca9)
```

```
##   BTNL9 IGSF10 ABCA9 Probability
## 1     0      0     0  0.98305085
## 2     0      0     1  0.01694915
## 3     1      0     0  0.73737374
## 4     1      0     1  0.26262626
## 5     0      1     0  0.78431373
## 6     0      1     1  0.21568627
## 7     1      1     0  0.03389831
## 8     1      1     1  0.96610169
```

In order to estimate the probability of the four genes in the network having high expressionlevels, it is necessary to calculate the probability of all four genes in the Bayesian network having high expression levels

That is, the following joint probability should be calculated.

$$P(BTNL9 = 1, CD300LG = 1, IGSF10 = 1, ABCA9 = 1)$$

Based on the previously calculated conditional probability table, the chain rule can be used to calculate the probability that all four genes in the Bayesian network are highly expressed.

$$\begin{gathered}P(BTNL9 = 1, CD300LG = 1, IGSF10 = 1, ABCA9 = 1) \\ = P(BTNL9 = 1) \times P(CD300LG = 1|BTNL9 = 1) \\ \times P(IGSF10 = 1|\text{class} = ?) \\ \times P(ABCA9 = 1|BTNL9 = 1, IGSF10 = 1)\end{gathered}$$

From the conditional probability table above, it is already known the values of all conditional probabilities except $P(IGSF10 = 1|class = ?)$. Since we don't know the value of *class* directly, we need to consider all possible values of *class* (i.e. 0 and 1) and weight these probabilities.

```r
# Given conditional probabilities
p_btnl9_1 <- 0.1790429
p_cd300lg_1_given_btnl9_1 <- 0.640553

# Calculate P(IGSF10 = 1 | class)
p_class_0_given_cd300lg_1 <- 0.7414966
p_class_1_given_cd300lg_1 <- 0.2585034
p_igsf10_1_given_class_0 <- 0.875
p_igsf10_1_given_class_1 <- 0.06454545

p_igsf10_1_given_cd300lg_1 <- p_class_0_given_cd300lg_1 *
  p_igsf10_1_given_class_0 + p_class_1_given_cd300lg_1 *
  p_igsf10_1_given_class_1

# P(ABCA9 = 1 | BTNL9 = 1, IGSF10 = 1)
```

```r
p_abca9_1_given_btnl9_1_igsf10_1 <- 0.9661017

# Calculate joint probability
p_all_high <- p_btnl9_1 * p_cd300lg_1_given_btnl9_1 *
  p_igsf10_1_given_cd300lg_1 *
  p_abca9_1_given_btnl9_1_igsf10_1

cat("P(BTNL9 = 1, CD300LG = 1, IGSF10 = 1, ABCA9 = 1):", p_all_high, "\n")
```

```
## P(BTNL9 = 1, CD300LG = 1, IGSF10 = 1, ABCA9 = 1): 0.07373601
```

Based on the Bayesian network and the conditional probability tables, the probability of all four genes $BTNL9$, $CD300LG$, $IGSF10$, and $ABCA9$ having high expression levels is approximately 7.37%.

Since gene expression in the data is binary (0 and 1), consider using a threshold of 1 to represent high expression.

To determine the probability of having cancer given the expression level of $CD300LG = 1$ (high) and $BTNL9 = 0$ (low), the following formula can be used:

$$P(\text{cancer} \mid CD300LG = 1, BTNL9 = 0) = P(\text{class} = 1 \mid CD300LG = 1, BTNL9 = 0)$$

Given that it can be gained:

$$P(CD300LG = k|BTNL9 = j)$$
$$P(\text{class} = \text{i} \mid CD300LG = k)$$

Using the law of total probability to compute $P(\text{class} = 1 \mid CD300LG = 1, BTNL9 = 0)$.

$$P(\text{class} = 1 \mid CD300LG = 1, BTNL9 = 0)$$
$$= \sum_k P(\text{class} = 1 \mid CD300LG = k) \times P(CD300LG = k|BTNL9 = 0)$$

However, since it is given $CD300LG = 1$, the only term in the summation we're interested in is when $k = 1$.

Therefore:

$$P(\text{class} = 1 \mid CD300LG = 1, BTNL9 = 0) = P(\text{class} = 1 \mid CD300LG = 1) \times P(CD300LG = 1|BTNL9 = 0)$$

Given the conditional probability tables are calculated before:

$$P(CD300LG = 1|BTNL9 = 0) = 0.008040201$$
$$P(class = 1|CD300LG = 1) = 0.2585034$$

Now compute the desired probability:

$$P(\text{class} = 1 \mid CD300LG = 1, BTNL9 = 0) = 0.2585034 \times 0.008040201 = 0.002078419$$

Use R language code to implement.

```r
p_cd300lg_1_given_btnl9_0 <- 0.008040201
p_class_1_given_cd300lg_1 <- 0.2585034

p_class_1_given_cd300lg_1_and_btnl9_0 <-
  p_class_1_given_cd300lg_1 * p_cd300lg_1_given_btnl9_0

print(p_class_1_given_cd300lg_1_and_btnl9_0)
```

```
## [1] 0.002078419
```

When the expression level of $CD300LG$ is high (i.e., 1) and the expression level of $BTNL9$ is low (i.e., 0), the probability of having cancer (represented by "class" being 1) is approximately 0.2078%.

Based on the Bayesian network given in Figure 2, the network structure can be described as:

1. BTNL9 → CD300LG
2. CD300LG → class
3. class → IGSF10
4. BTNL9, IGSF10 → ABCA9

The conditional independence between "class" and ABCA9 given CD300LG can be ascertained by examining all undirected paths between "class" and ABCA9. Conditional independence would imply that, given the observed value of CD300LG, any path between "class" and ABCA9 is blocked.

In the Bayes structure, there exists an active path from "class" to ABCA9 through IGSF10. Specifically, "class" influences IGSF10, which in conjunction with BTNL9, influences ABCA9. Knowledge of CD300LG does not deactivate this path; thus, the influence of "class" on ABCA9 remains, even when CD300LG is known.

Therefore, given the value of CD300LG, "class" is not conditionally independent of ABCA9 due to the active path from "class" to ABCA9 via IGSF10.

References

[1] P. Spirtes, C. Glymour, and R. Scheines, Causation, Prediction, and Search. Springer Science & Business Media, 2012.

[2] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," Machine Learning, vol. 65, no. 1, pp. 31–78, Mar. 2006, doi: https://doi.org/10.1007/s10994-006-6889-7.

[3] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," National Conference on Artificial Intelligence, pp. 41–48, Jan. 1998.

[4] Lewis, D.D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In: Nédellec, C., Rouveirol, C. (eds) Machine Learning: ECML-98. ECML 1998. Lecture Notes in Computer Science, vol 1398. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0026666