Advanced Analytics 2
Project
Mahmoud Ghazi
110330165

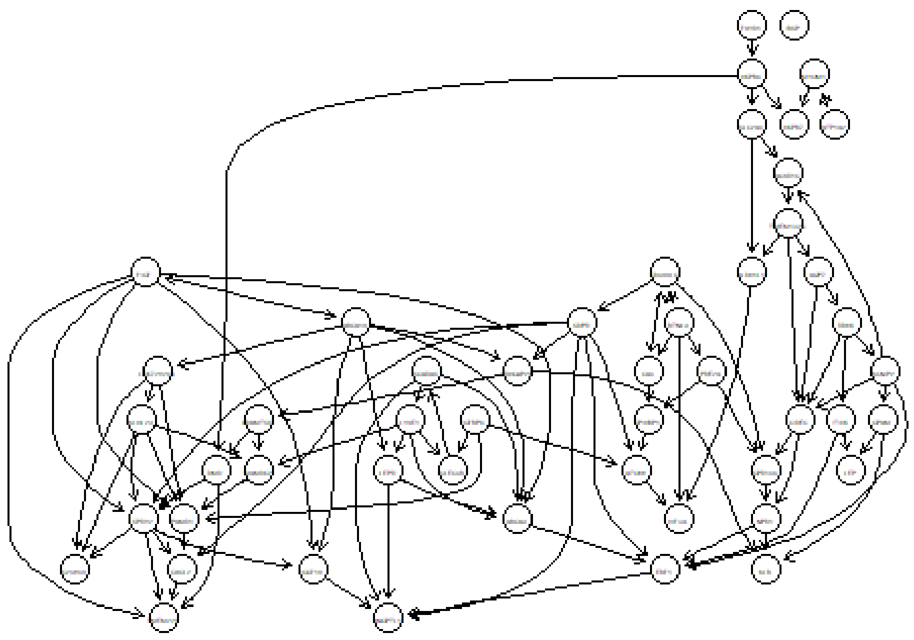
Q1- Gene regulatory network

To find the best causal structure, I apply the constraint-based approach name PC. After replacing the value of the Class variable with 0 and 1, we need to regulate PC arguments. In the following script, by **suffStat**, we specify a list of variables from data that correlates so that a conditional independence test will be applied to this list. Also, **indepTest** defines a function to test conditional independence in a significance level of **alpha**. Furthermore, by **labels**, we identify a vector of variable names for the algorithm. The output demonstrates the DAG for all variables.

```r
knitr::opts_chunk$set(echo = TRUE)
library(pcalg)

data <- read.csv("./BRCA_RNASeqv2_top50.csv")
data$class <- ifelse(data$class =="C", 1, 0)
data$class <- as.numeric(data$class)
genes <- subset(data, select=-(class))
n <- nrow (genes)
V <- colnames(genes)
pc.fit <- pc(suffStat = list(C = cor(genes), n = n),
             indepTest = gaussCItest, alpha=0.01, labels = V)
if (require(Rgraphviz)) {
  plot(pc.fit
       , main = "Graph 1 - Causal Structure")
}
```

Graph 1 - Causal Structure



How the PC (Peter & Clark) algorithm works?

PC algorithm (Neapolitan & Jiang 2009) starts with a fully connected graph shown in figure 1 obtained from the correlated variables. Then we apply a conditional independent test such as chi-square or partial correlation into levels to test the dependency. If the test shows that they are dependent, we try it next time in the next level; otherwise, we drop the link.

We must consider a set (Z) as a subset of the paired variable with a specific variable X. The math notation for Z comes as equation 1, which means all variables that can be paired with X in the graph exclude its pair Y. This set will be used in the following explanation of the algorithm.

$$Z \subseteq adj(X,G) \setminus \{Y\} \qquad \text{Equation 1}$$

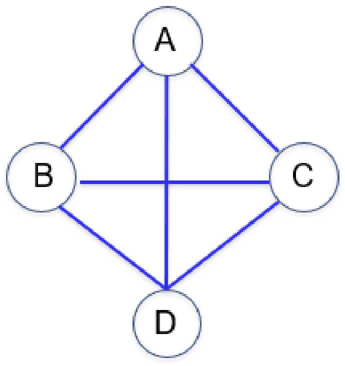


Figure 1

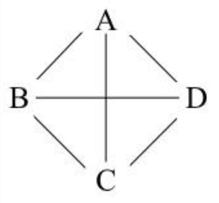
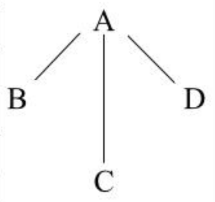
Explain PC algorithm by example:

PC algorithm helps to find the skeleton and orient the edges.

Skeleton:

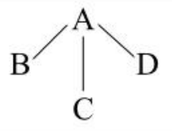
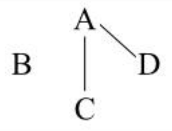
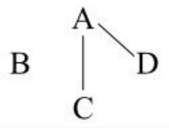
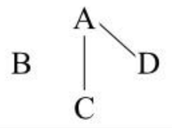
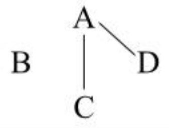
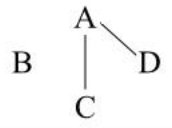
In the first level, we test the dependency of variables condition on empty set Z (|Z|=0). As shown in Table 1, we test dependency between all pairs in the graph and then update the graph. When the test indicates that two variables are independent, we remove the link in the graph. For example, test 5 shows that B and C are independent, so we remove the connection between these two variables.

Table 1

Level	Graph	#CI tests	Test	Result	Updated Graph
1		1	I(A, B)?	No	
		2	I(A, C)?	No	
		3	I(A, D)?	No	
		4	I(B, A)?	No	
		5	I(B, C)?	Yes	
		6	I(B, D)?	Yes	
		7	I(C, A)?	No	
		8	I(C, D)?	Yes	
		9	I(D, A)?	No	

In the second level, we use the updated graph from the last step and test the dependency of the variable condition on Z, which is a subset of variables with the size of one (|Z|=1) at a time. For example, to test the dependency of A and B, investigate their independency condition on Z with the size of one, which includes variables {C} and {D}. Once the result of one test comes as Yes, implying the independency between the two variables (test 10), we remove the link as shown in Table 2.

Table 2

2		10	I(A, B | C)?	Yes	
		11	I(A, C | D)?	No	
		12	I(A, D | C)?	No	

In the third level, we need to test the dependency of the variables given two variables at a time and the same for the upper levels. The algorithm stops once the size of Z is smaller than the level. For example, to test the independency between A and C conditions on two variables, the algorithm stops as we have only D as a member of Z (|Z|<level) at this level.

Orienting the edges:

After obtaining the skeleton graph shown in figure 2, the links remain undirected. We need to orient the undirected link into a directed link based on the following rule:

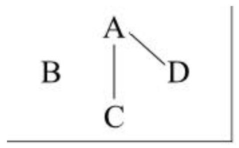


Figure 2

Replace all nonadjacent variables Xi, Xj with common neighbour Xk with a V shape of Xi → Xk ← Xj when k is not in the separation set of i and j.

The separation set includes variables that cause independence between two variables. For example, if we had a link between B and C, C was in the separation set of A and B, as test number 10 confirms the independency between A and B given C.

We have input as a skeleton with a link between C, A and D. Node A is not in a Separation set of C and D, so it satisfies the condition of this rule. So, we can change the undirected links into the V structure of C →A← D shown in figure 3.

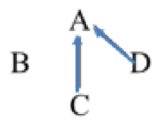


Figure 3

Q2- 10 genes that have strong causal effects on EBF1

To find parent and child nodes of the EBF1, we need a local structure learning algorithm like PC-select. The **response vector** of EBF1 as a first argument has been set. Then all variables except for the response variable were selected for the second argument, and a 0.05 significant level was identified for alpha. The output shows the sorted list of 10 genes that have a strong causal effect on EBF1.

```
pcS <- pcSelect(genes[c("EBF1")], genes[, !names(genes) %in% c("EBF1")] , alp
ha=0.05)
pcs_frame <- data.frame(pcS,stringsAsFactors = FALSE)
pcs_frame$gene <- rownames(pcs_frame)
rownames(pcs_frame) <- NULL
newdata <- pcs_frame[order(-pcs_frame$zMin),]
newdata[0:10,]
```

```
##         G     zMin     gene
## 22  TRUE 7.454331    ABCA9
## 9   TRUE 6.091818   KCNIP2
## 49  TRUE 4.458296  ANGPTL1
## 19  TRUE 3.377566 ARHGAP20
## 20  TRUE 2.867472     NPR1
## 33  TRUE 2.852502    ITIH5
## 6   TRUE 2.772298     SDPR
## 3  FALSE 1.955257  CD300LG
## 36 FALSE 1.945730      DMD
## 7  FALSE 1.943827    MYOM1
```

Q3 - Genes in the Markov blanket

To find the MB set by **learn.mb**, the first argument is a data frame containing all variables. Then, we should specify the name of the node (ABCA9) whose local structure is being learned. **method** is a structure learning algorithm. In this work, I specify Incremental Association (iamb). Iamb works based on a two-phase selection scheme explained in the next part. The following output represents 22 genes in the Markov blanket set.

```
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
library(bnlearn)
MB.Z=learn.mb(genes, "ABCA9", method="iamb", alpha=0.01)
MB.Z
```

```
##  [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"
```

How the Markov blanket algorithm works

The Markov Blanket (MB) concept is associated with the conditional independence of nodes in a Bayesian network (BN). It indicates that for *each variable X, the set of all parents of X, children of X, and parents of children of X is considered a Markov blanket of X* (see figure 4). Thus, the specific node is conditionally independent of the other nodes that do not belong to its Markov blanket (Helldin & Riveiro 2009).

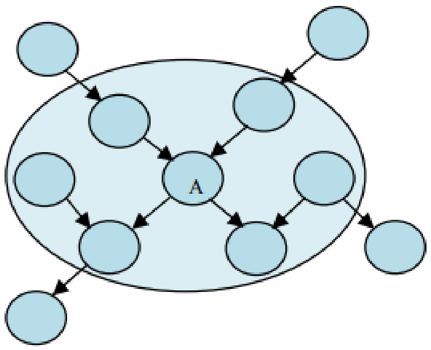


Figure 4- Markov blankets

Incremental Association Markov Blanket (IAMB) Algorithm:

IAMB is a novel algorithm (Tsamardinos et al. 2003) for discovering Markov MB(A), which can be performed well under the following assumptions:

1- BNs can represent data.

2- There is a reliable statistical test of conditional independency for the variable.

IAMB consists of two phases to identify MB(A). During the forward step, set CMB keeps all variables that is estimated as MB(A). It starts with an empty set of CMB, and by each iteration, the variable that maximizes a heuristic function f(X; A|CMB) get into the set and is assumed as a member of MB. The function returns a non-zero value which implies the relation between X and A given CMB. In this step, any feature that belongs to MB(A) will be admitted to CMB.

In the backward phase, the member of CMB that is independent of A given remaining CMB will be removed. In this step, any member that is independent of the A condition on MB(A) or any superset will be removed. We end up with a unique MB(T).

Q4 - Parent and children set of the class variable by PC-simple

In order to discretize data, I calculated the mean for all variables except for Class variable and assigned 1 and 0 to those above and below the average, respectively. The following output shows the list of genes containing parents and children set regarding class variable obtained by the PC-simple algorithm on discretized data set.

```
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
library("caret")
library(e1071)
library("klaR")

mean_all <- mean(as.matrix(genes))
genes <- as.data.frame(ifelse(genes > mean_all, 1, 0))
data1 <- data
data1[, !names(data1) %in% c("class")]  <- as.data.frame(ifelse(data[,!names(
data) %in% c("class")]  > mean_all, 1, 0))

pcS4 <- pcSelect(data[c("class")], genes , alpha=0.05)
pcs4_frame <- data.frame(pcS4,stringsAsFactors = FALSE)
pcs4_frame$gene <- rownames(pcs4_frame)
rownames(pcs4_frame) <- NULL
newdata4 <- pcs4_frame[order(-pcs4_frame$zMin),]
pcs_genes <- newdata4[newdata4$G==TRUE,c("gene")]
pcs_genes

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

How PC-simple works

PC-simple (Neapolitan & Jiang 2009) is a simplified version of the PC algorithm and follows the same causal assumptions as those by the causal Bayesian network. This algorithm produces a set of variables that strongly influence target variables. If there are j different predicted variables in the data sets that are X(1), X(2) to X(j), and the target variable is Z, We would like to find the parental children set of target variable Z. First, it is assumed that all the predicted variables are in the parents and children set (PC). Then the conditional independence test is used to test the dependency between each predicted variable and variable target condition on S, that by each level, the size of S increases. As the notation of **S** in equation 2 shows, S includes all PC members in level k except for the pair(Y) of the target variable.

$S \in PC_K / \{Y\}$ Equation 2

Example:

PC-Simple by following graph and assume Z is a target variable.

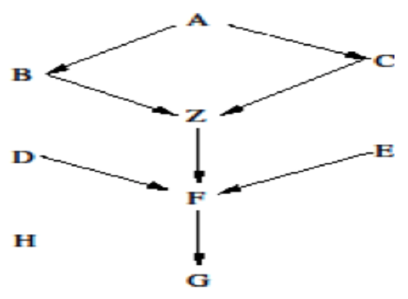

$$\begin{tikzcd}
 & A \arrow[dl] \arrow[dr] & \\
B \arrow[r] & Z \arrow[d] & C \arrow[l] \\
D \arrow[r] & F \arrow[d] & E \arrow[l] \\
H & G &
\end{tikzcd}$$

Figure 5

Level 1: In the first level, the independence condition is on the empty given set (|S|=0).

Initially all predicted Variables (PC0={A,B,C,D,E,F,G,H}) are considered as children and parent set. Examine the independence test of each variable with target variable Z given empty set S**.** At the end of this iteration, D, E and H are removed from PC1 since they are independent of Z, and PC updates to PC1 = {A,B,C,F,G}.

Level 2: In the second level, the independence condition is on the S with size one (|S|=1).

In this level, we examine the independency condition test of each variable in PC1 with the target variable condition on S. At this level, the size of S is 1, which contains all single variables of PC1 except for the target pair. We assume node G is independent of Z given F, so G is removed from PC1 and the set update to PC2={A,B,C,F}.

Level 3: In the third level, the independence condition is on the S with size two (|S|=2).

At this level, S implies all members of PC2 with size two except for the target pair. For example we check the dependency of A with Z given {B,C}, {B,F} and {C,F}. Assuming A is independent of Z given {B, C}, we remove A from PC2 and update the set to PC3={B, C, F}.

The algorithm will be stopped if the size of the last updated PC (PCk) is smaller or equal to the level. In this example size of PC3 is smaller than level 4 (3<4), so it stops, and PC3 is a final set for parents and children of Z.

Naïve Bayes classification with all features

I apply the train function from the caret library to classify data based on the Naive Bayes approach. We need to split data into train and test sets and assign the train set features to the first argument and the outcome for each sample to the second argument. I specify Naive Bayes as a string (nb), implying which classification model to use. to evaluate and compare the learning algorithm, I apply **trainControl** and set c*ross-validation (***cv)** as a method. *Cross-validation* is a resampling method in which the **number** of folds is the number of groups a given data sample will be split into. The following output shows a confusion matrix based on the Naïve Bayes classification of all genes in the data set by applying 10-fold cross-validation.

```
data$class <- as.factor(data$class)
train_index <- sample(1:nrow(genes), 0.8 * nrow(genes))
test_index <- setdiff(1:nrow(genes), train_index)
train <- genes[train_index,]
test <- genes[test_index,]
x <- train
y <- data[train_index,]$class

model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))
x <- test
y <- data[test_index,]$class
confusionMatrix <- prop.table(table(predict(model$finalModel,x)$class,y))
confusionMatrix

##    y
##              0          1
##   0 0.06995885 0.00000000
##   1 0.03292181 0.89711934
```

performance metrics

The following output represent performance metrics for the Naïve Bayes classification on all features.

```
accuracy <- (confusionMatrix[1,1]+confusionMatrix[2,2])/
  (confusionMatrix[1,1]+confusionMatrix[1,2]+
     confusionMatrix[2,1]+confusionMatrix[2,2])
recall <-  confusionMatrix[1,1]/
  (confusionMatrix[1,1]+confusionMatrix[1,2])
Precision <-  confusionMatrix[1,1]/
  (confusionMatrix[1,1]+confusionMatrix[2,1])
F1 <- 2*(Precision*recall)/(Precision+recall)
accuracy

## [1] 0.9670782

recall

## [1] 1

Precision

## [1] 0.68

F1

## [1] 0.8095238
```

Naïve Bayes classification with features obtain by PC-simple

The following confusion matrix based on the output of Naïve Bayes classification on genes obtained by PC-simple algorithm by applying 10-fold cross validation.

```
genes_select <- genes[,c(pcs_genes)]
train_index <- sample(1:nrow(genes_select), 0.8 * nrow(genes_select))
test_index <- setdiff(1:nrow(genes_select), train_index)
train <- genes_select[train_index,]
test <- genes_select[test_index,]
x <- train
y <- data[train_index,]$class

model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

x <- test
y <- data[test_index,]$class
confusionMatrix_select <- prop.table(table(predict(model$finalModel,x)$class,
y))
confusionMatrix_select

##    y
##              0          1
##   0 0.07407407 0.00000000
##   1 0.02469136 0.90123457
```

performance metrics

The following output shows a confusion matrix based on the Naïve Bayes classification of genes obtained by the PC-simple algorithm by applying 10-fold cross-validation.

```
accuracy <- (confusionMatrix_select[1,1]+confusionMatrix_select[2,2])/
  (confusionMatrix_select[1,1]+confusionMatrix_select[1,2]+
     confusionMatrix_select[2,1]+confusionMatrix_select[2,2])
recall <-  confusionMatrix_select[1,1]/
  (confusionMatrix_select[1,1]+confusionMatrix_select[1,2])
Precision <-  confusionMatrix_select[1,1]/
  (confusionMatrix_select[1,1]+confusionMatrix_select[2,1])
F1 <- 2*(Precision*recall)/(Precision+recall)
accuracy

## [1] 0.9753086

recall

## [1] 1

Precision
```

```
## [1] 0.75

F1

## [1] 0.8571429
```

As Table 3 shows the classification model on features obtained by the PC algorithm shows better accuracy and performance metrics than the model on all features. It confirms how finding influential variables that is the aim of local causal structure learning algorithms can improve the result of classification.

Table 3

	Accuracy	precision	F1
All features	0.9670782	0.68	0.8095238
Selected features by PC-simple	0.9753086	0.75	0.8571429

Q5-a- Conditional probability tables

To create conditional probability tables, we need conditional probability tables (**cptable)** from Rgain. The first argument is the specifications of the names implying the form of probability. The second argument is a numeric vector indicating the frequency of names in the first argument. To calculate this vector, I applied piping from dplyr to group by data and counted the frequency for distinct rows. We have two levels of 0 and 1, which is set for the last argument. Figure 4 shows the DAG plot based on the conditional probability tables.

```r
library(gRain)

library(dplyr)

BTNL9_ <- table(data1[,c("BTNL9")])
CD300LG_BTNL9 <- data1[,c("CD300LG", "BTNL9")] %>% group_by_all %>% count
class_CD300LG <- data1[,c("class","CD300LG")] %>% group_by_all %>% count
IGSF10_class <- data1[,c("IGSF10","class")] %>% group_by_all %>% count
ABCA9_IGSF10_BTNL9 <- data1[,c("ABCA9", "IGSF10", "BTNL9")] %>% group_by_all
%>% count

lvl <- levels(as.factor(genes$BTNL9))
BTNL9 <- cptable(~BTNL9, values= as.vector(BTNL9_),levels=lvl)
CD300LG <- cptable(~CD300LG|BTNL9, values=genes$CD300LG,levels=lvl)
class_ <- cptable(~class_|CD300LG, values=data$class, levels=lvl)
IGSF10 <- cptable(~IGSF10|class_, values=genes$IGSF10, levels=lvl)
ABCA9 <- cptable(~ABSA9|IGSF10:BTNL9,values=genes$ABCA9,levels=lvl)
plist <- compileCPT(list(BTNL9,CD300LG,class_,IGSF10,ABCA9))
```

```
net1=grain(plist)
plot(net1$dag) # directed network
```

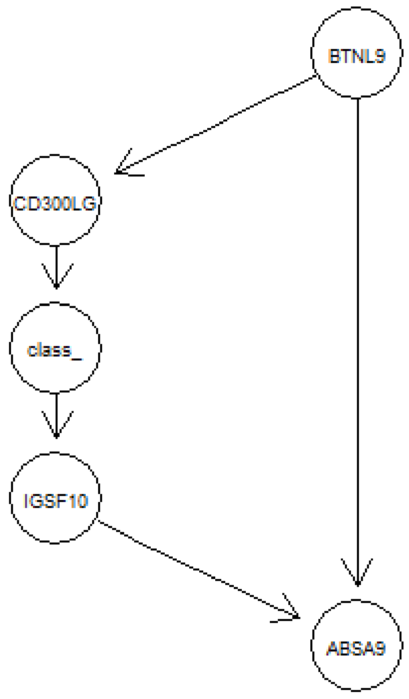


Figure 6- DAG plot

Q5-b - Probability of the four genes with high expression levels

To find the conditional distribution of a set of variables given other variables, I apply **querygrain**. The first argument is a network that was compiled over the nodes. The second argument is a vector of nodes. The last argument is a type of probability such as "marginal", "joint", or "conditional". The following outputs highlighted in yellow show the probability of the four genes in the network having high expression levels.

```
querygrain(net1, nodes=c("BTNL9","CD300LG","IGSF10","ABCA9"), type="marginal"
)

## $BTNL9
## BTNL9
##         0         1
## 0.8209571 0.1790429
##
## $CD300LG
## CD300LG
##         0         1
## 0.7705745 0.2294255
##
```

```
## $IGSF10
## IGSF10
##         0         1
## 0.4427192 0.5572808
##
## $ABCA9
## ABCA9
##         0         1
## 0.7730902 0.2269098
```

Q5-c - probability of having cancer when the expression level of CD300LG is high and BTNL9 is low

The following output highlighted with yellow shows the probability of having cancer when the expression level of CD300LG is high, and the expression level of BTNL9 is low.

```
querygrain(net1, nodes=c("class_","CD300LG"," BTNL9"), type="conditional")
```

```
## , , BTNL9= 0
##
##       CD300LG
## class_          0          1
##      0 0.02678571 0.96545455
##      1 0.97321429 0.03454545
##
## , , BTNL9= 1
##
##       CD300LG
## class_          0          1
##      0 0.02678571 0.96545455
##      1 0.97321429 0.03454545
```

Q5-d - prove the result

Using the Bayesian network, we can calculate inference. As the problem is from cause to effect so we encounter a causal Bayesian network inference that can be written as follow:

`P(class_=1| CD300LG=1 ,BTNL9=0) ?`

Assign all names with letters:

C = Class

D = CD300LG

T = BTNL9

According to bayes rule we have (Neapolitan & Jiang 2009):

P(X|E) = P(E|X)P(X)/P(E) $\rightarrow$ P(X|E) = P(E,X)/P(E)

$$\mathrm{P(C|D,-T)} = \frac{\mathrm{P(C,D,-T)}}{\mathrm{P(D,-T)}}$$ >> Bayes Rule

$P(C,D,-T) = P(C|D,-T)P(D,-T)$

According to Markov condition (Neapolitan & Jiang 2009): given its parents (P_1, P_2), a node (X) is conditionally independent of its non-descendants (ND_1, ND_2).

$P(C|D,-T)P(D,-T)$

= $P(C|D)P(D,-T)$ >> Markov condition: I(X, Non-Descendants|Parent(X)) -> I(C, T|D)

So, we have: $\mathrm{P(C|D,-T)} = \frac{\mathrm{P}(C|D)\mathrm{P(D,-T)}}{\mathrm{P(D,-T)}}$

= $\mathrm{P(C|D,-T)} = \mathrm{P(C|D)}$

According to the frequency of Class and CD300LG, to get the probability of cancer give CD300LG with a high expression level, we can calculate as follows:

Probability = 38/38+1062 = 0.03454

```
class_CD300LG <- data1[,c("class","CD300LG")] %>% group_by_all %>% count

## class  CD300LG   n
## <dbl>  <dbl>    <int>
##1  0      0        3
##2  0      1        109
##3  1      0        1062
##4  1      1        38
```

It confirms with the following query that shows the probability of Cancer given CD300LG with high expression level.

```
#P(class_=yes|CD300LG=yes)
querygrain(net1, nodes=c("class_","CD300LG"), type="conditional")

##       CD300LG
## class_          0          1
##      0 0.02678571 0.96545455
##      1 0.97321429 0.03454545
```

Q5-e - Given CD300LG, “class” conditionally independent of ABCA9

Yes, it is conditionally independent. According to the Markov condition, given we know the value of Class’s parent CD300LG, the node Class is conditionally independent of its non-descendants, and ABCA9 is non-descendent.

Reference

Neapolitan, RE & Jiang, X 2009, *Probabilistic Methods for Bioinformatics: With an Introduction to Bayesian Networks*, San Francisco: Elsevier Science & Technology, San Francisco.

Helldin, T & Riveiro, M 2009, 'Explanation Methods for Bayesian Networks: review and application to a maritime scenario', *Proceedings of the 3rd Annual Skövde Workshop on Information Fusion Topics (SWIFT 2009)*, pp. 11-16.

Tsamardinos, I, Aliferis, CF, Statnikov, AR & Statnikov, E 2003, 'Algorithms for large scale Markov blanket discovery', *FLAIRS conference*, St. Augustine, FL, pp. 376-380.