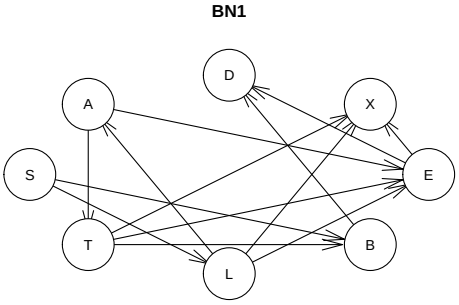
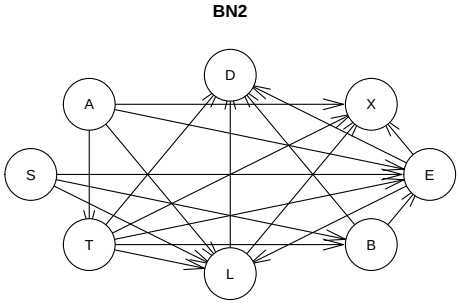
AML LAB 1

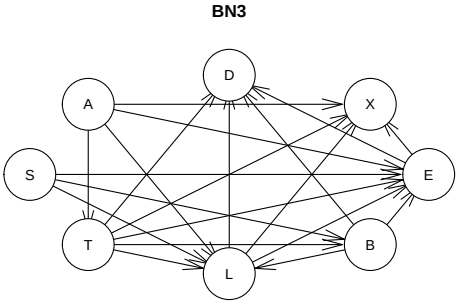
Harshavardhan Subramanian

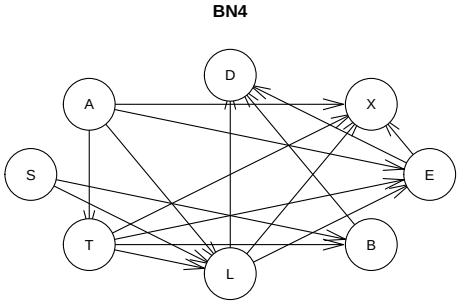
11/09/2020

Question 1










```
## Comparison of BN1 and BN2 :
## [1] "Different number of directed/undirected arcs"
## $tp
## [1] 11
##
## $fp
## [1] 8
##
## $fn
## [1] 2
## Comparison of BN2 and BN3 :
## [1] "Different number of directed/undirected arcs"
## $tp
## [1] 18
##
## $fp
```

```
## [1] 2
##
## $fn
## [1] 1
```

```
## Comparison of BN1 and BN3 :
```

```
## [1] "Different number of directed/undirected arcs"
```

```
## $tp
## [1] 12
##
## $fp
## [1] 8
##
## $fn
## [1] 1
```

```
## Comparison of BN1 and BN4 :
```

```
## [1] "Different number of directed/undirected arcs"
```

```
## $tp
## [1] 12
##
## $fp
## [1] 4
##
## $fn
## [1] 1
```

```
## Comparison of BN2 and BN4 :
```

```
## [1] "Different number of directed/undirected arcs"
```

```
## $tp
## [1] 15
##
## $fp
## [1] 1
##
## $fn
## [1] 4
```

```
## Comparison of BN3 and BN4 :
```

```
## [1] "Different number of directed/undirected arcs"
```

```
## $tp
## [1] 16
##
## $fp
## [1] 0
##
## $fn
## [1] 4
```

From the above comparison it is evident that when the Hill Climbing Algorithm is run multiple times can returns Non Equivalent Bayesian structures. To support the argument the all.equal() function provides

evidence saying whether the number of Directed and Undirected arcs are same or different and compare() function provides True and False Positives and False Negatives.

Hill climbing by nature is a heuristc approach which starts at a random point and find the optimal solution. It adds or removes edges based on Bayesian score and in this case, HC algorithm when run multiple times, can start at different random points and correspondingly end up in different optimal solution choosing a different path. Hence we can conclude it results in different Bayesian Networks.

Question 2

```
## The Confusion Matrix of learned Bayesian Network is:

##         S_pred
## S_actual  no yes
##      no  322 146
##      yes 120 412

## The Confusion Matrix of Original Bayesian Network is:

##         S_pred_org
## S_actual  no yes
##      no  322 146
##      yes 120 412
```

By comparing the Confusion matrix of both Learned Bayesian Network and Original Bayesian Network, it is found that the posterior probability distribution of S for each case is found to be same.

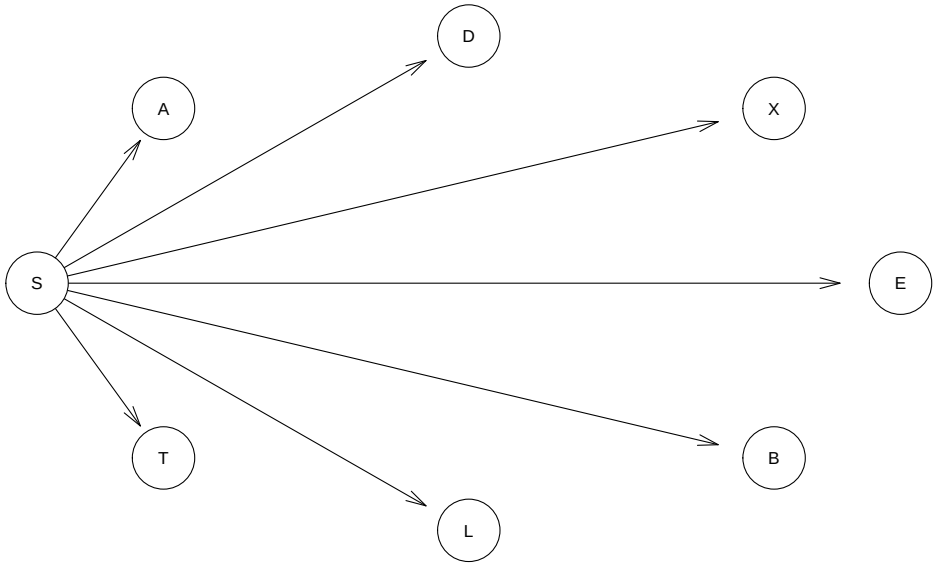
Question 3

```
## The Confusion Matrix of Markov Blanket of S is:

##         S_pred_mb
## S_actual  no yes
##      no  322 146
##      yes 120 412
```

The Confusion matrix considered only for Markov Blanket of S is found to be same as that of Original Network and Learned Network. Since the Markov Blanket gives the all nodes dependent on S or the nodes inline with S node, we can say that the confusion matrix in this question is the actual and when compared with the learned BN or Original BN, none of the nodes where actually dependent on S except for L and B. Hence the presence of other nodes apart from nodes dependent on S, did not affect the Confusion matrix. Therefore the Confusion matrix of all 3 networks are same.

Question 4




```
## The Confusion Matrix of Naive Bayes Model is:
```

```
##         S_pred_Bayes
## S_actual  no yes
##      no  349 119
##      yes 188 344
```

Question 5

Comparing the results of 2nd and 4th question, the assumption of Naive Bayes independency rule, the TRUE NEGATIVE prediction has significantly increased and the TRUE POSITIVE has decreased. The cause of this result is because of the assuming the false dependency of all other nodes on S. Where as in 2nd question only node "L" and "B" was dependent on S. Hence the results have varied when compared 2nd and 4th question.

Appendix

```
knitr::opts_chunk$set(echo = TRUE,warning = FALSE, message = FALSE,fig.width=12, fig.height=8)

library(BiocGenerics)
library(BiocManager)
library(BiocVersion)
library(bnlearn)
library(gRain)
library(graph)
library(gRbase)
data("asia")
```

```
asia <- asia
BN1 <- hc(asia, score = "bde",max.iter = 100, restart = 10, iss = 10)
BN2 <- hc(asia, score = "bde",max.iter = 50, restart = 20, iss = 50)
BN3 <- hc(asia, score = "bde",max.iter = 20, restart = 10, iss = 100)
BN4 <- hc(asia, score = "bde",max.iter = 100, restart = 10, iss = 22)

par(mfrow = c(2,2))
plot(BN1, main = "BN1")
plot(BN2, main = "BN2")
plot(BN3, main = "BN3")
plot(BN4, main = "BN4")

cat("Comparison of BN1 and BN2 :")
all.equal(BN1,BN2)
compare(BN1,BN2)

cat("Comparison of BN2 and BN3 :")
all.equal(BN2,BN3)
compare(BN2,BN3)

cat("Comparison of BN1 and BN3 :")
all.equal(BN1,BN3)
compare(BN1,BN3)

cat("Comparison of BN1 and BN4 :")
all.equal(BN1,BN4)
compare(BN1,BN4)

cat("Comparison of BN2 and BN4 :")
all.equal(BN2,BN4)
compare(BN2,BN4)

cat("Comparison of BN3 and BN4 :")
all.equal(BN3,BN4)
compare(BN3,BN4)

suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
n <- dim(asia)[1]
id<-sample(1:n, floor(n*0.8))
train <- asia[id,]
test <- asia[-id,]

BN_Asia <- hc(train, score = "bde",max.iter = 100, restart = 10, iss = 100)
fit1 <- bn.fit(BN_Asia,train,method = "mle")
grain_fit1 <- as.grain(fit1)
grain_df_test <- test[,-2]
S_pred <- numeric()
for(i in seq(1:nrow(grain_df_test)))
{
  set <- setFinding(grain_fit1, nodes = c("A","T","L","B","E","X","D"), states = sapply(grain_df_test[i
  query <- querygrain(set,nodes = "S",type = "conditional")
  if(query[1] > query[2])
```

```
  {
    S_pred[i] <- "no"
  }
  else
  {
    S_pred[i] <- "yes"
  }
}

S_actual <- test[,"S"]
cm_bn <- table(S_actual,S_pred)
cat("The Confusion Matrix of learned Bayesian Network is: ")
cm_bn


dag <- model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
fit2 <- bn.fit(dag,train,method = "mle")
grain_fit2 <- as.grain(fit2)
grain_df_test <- test[,-2]
S_pred_org <- numeric()
for(i in seq(1:nrow(grain_df_test)))
{
  set <- setFinding(grain_fit2, nodes = c("A","T","L","B","E","X","D"), states = sapply(grain_df_test[i
  query <- querygrain(set,nodes = "S",type = "conditional")
  if(query[1] > query[2])
  {
    S_pred_org[i] <- "no"
  }
  else
  {
    S_pred_org[i] <- "yes"
  }
}

cm_org <- table(S_actual,S_pred_org)
cat("The Confusion Matrix of Original Bayesian Network is: ")
cm_org


mod <- hc(train)
fit_mod <- bn.fit(mod,train,method = "mle")
grain_fit_mod <- as.grain(fit_mod)
mb3 <- mb(fit_mod,"S")
grain_df_test_2 <- test[,c(-1,-2,-3,-6,-7,-8)]
S_pred_mb <- numeric()
for(i in seq(1:nrow(grain_df_test_2)))
{
  set <- setFinding(grain_fit_mod, nodes = c(mb3[1],mb3[2]), states = sapply(grain_df_test_2[i,],toStri
  query <- querygrain(set,nodes = "S",type = "conditional")
  if(query[1] > query[2])
  {
    S_pred_mb[i] <- "no"
  }
```

```
  else
  {
    S_pred_mb[i] <- "yes"
  }
}

S_actual <- test[,"S"]
cm_mb <- table(S_actual,S_pred_mb)
cat("The Confusion Matrix of Markov Blanket of S is: ")
cm_mb

par(mfrow = c(1,1))
baiyes <- empty.graph(c("A","S","T","L","B","E","X","D"))
arc.set <- matrix(c("S", "A", "S", "T", "S", "L","S","B","S","E","S","E","S","X","S","D"),ncol = 2, byr
arcs(baiyes) <- arc.set
plot(baiyes)
fit4 <- bn.fit(baiyes,train,method = "mle")
grain_fit4 <- as.grain(fit4)
grain_df_test <- test[,-2]
S_pred_Bayes <- numeric()
for(i in seq(1:nrow(grain_df_test)))
{
  set <- setFinding(grain_fit4, nodes = c("A","T","L","B","E","X","D"), states = sapply(grain_df_test[i
  query <- querygrain(set,nodes = "S",type = "conditional")
  if(query[1] > query[2])
  {
    S_pred_Bayes[i] <- "no"
  }
  else
  {
    S_pred_Bayes[i] <- "yes"
  }
}

S_actual <- test[,"S"]
cm_Bayes <- table(S_actual,S_pred_Bayes)
cat("The Confusion Matrix of Naive Bayes Model is: ")
cm_Bayes
```