Graphical Model

Aman Kumar Nayak

9/5/2020

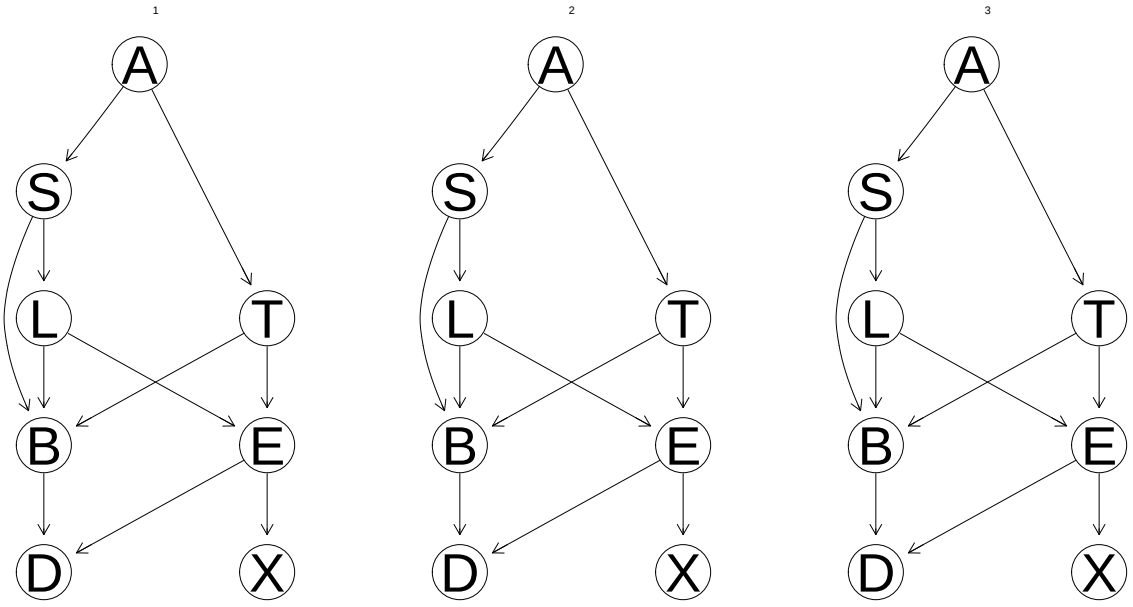
Question 1

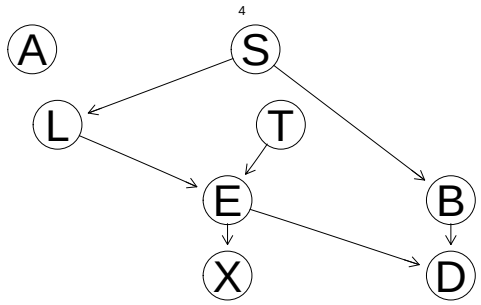
Data : The asia data set contains the following variables:

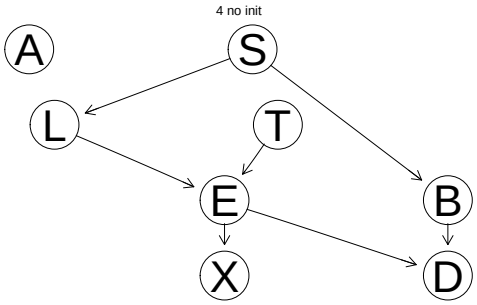
- D (dyspnoea), a two-level factor with levels yes and no.
- T (tuberculosis), a two-level factor with levels yes and no.
- L (lung cancer), a two-level factor with levels yes and no.
- B (bronchitis), a two-level factor with levels yes and no.
- A (visit to Asia), a two-level factor with levels yes and no.
- S (smoking), a two-level factor with levels yes and no.
- X (chest X-ray), a two-level factor with levels yes and no.
- E (tuberculosis versus lung cancer/bronchitis), a two-level factor with levels yes and no.

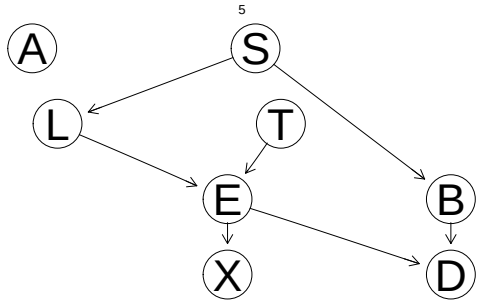
Hill Climbing Algorithm

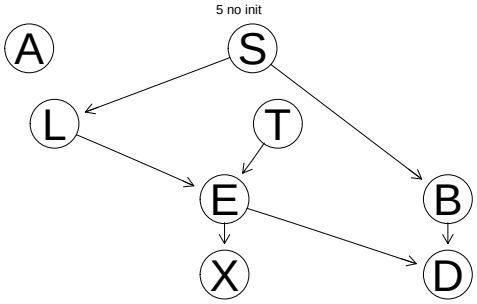
1st



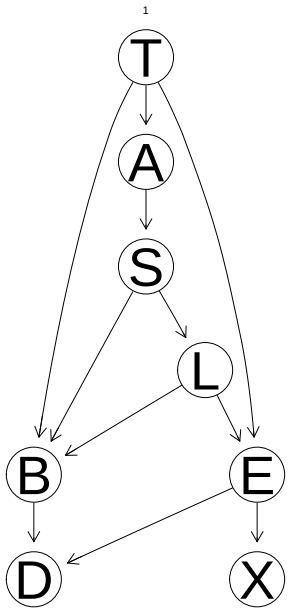


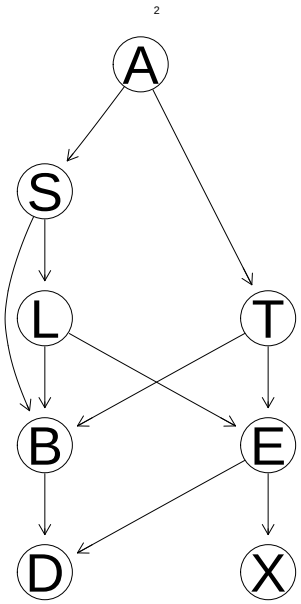


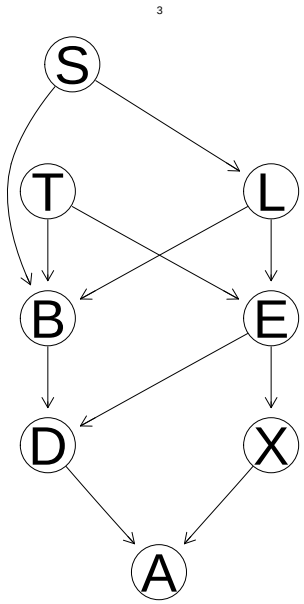


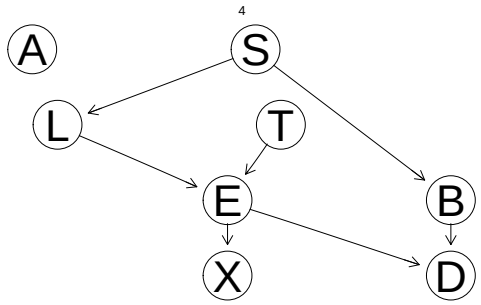


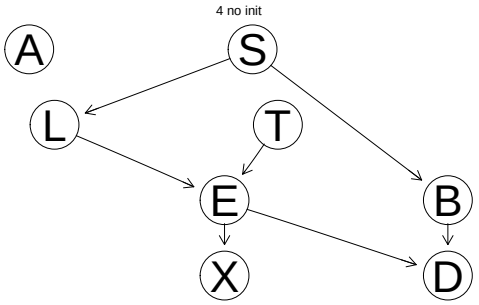
Set 2 just to check if re-run is changing graphical models

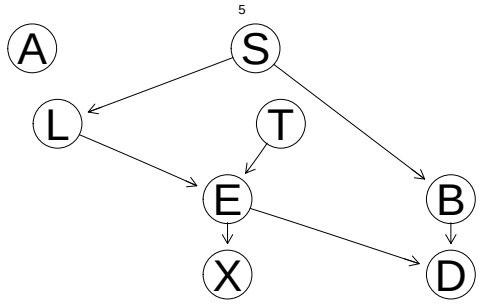


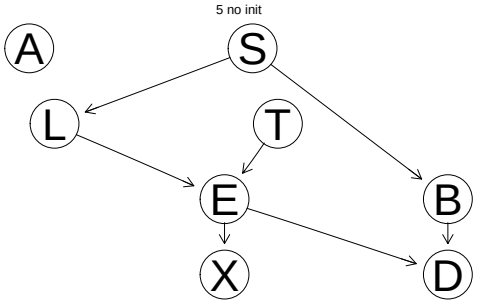












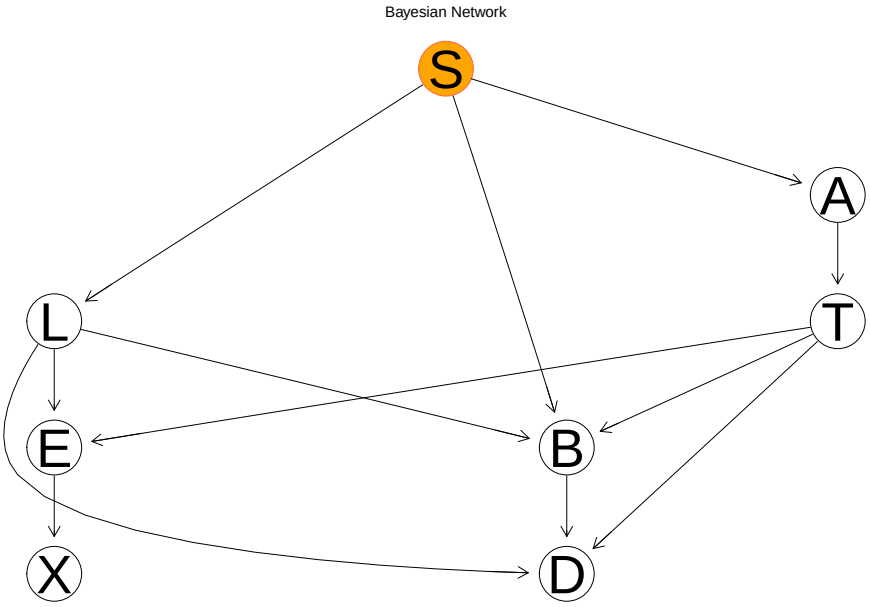
Question 2

Calculating inference for S (Smoking)

```
## Score Based Structure

##
##   Bayesian network learned via Score-based methods
##
##   model:
##    [S][A|S][L|S][T|A][B|S:T:L][E|T:L][X|E][D|T:L:B]
##   nodes:                                 8
##   arcs:                                  12
##     undirected arcs:                     0
##     directed arcs:                       12
##   average markov blanket size:           3.50
##   average neighbourhood size:            3.00
##   average branching factor:              1.50
##
##   learning algorithm:                    Hill-Climbing
##   score:                                 AIC (disc.)
##   penalization coefficient:              1
##   tests used in the learning procedure:  280
##   optimized:                             TRUE

## Graph associated with a Bayesian network
```




```
## Fitting Parameter of Bayesian Model

## Bayesian network as a list of conditional probability tables

## Under selected model Probability of observing this evidence for S = Yes is

## [1] 0.4957511

## Under selected model Probability of observing this evidence for S = No is

## [1] 0.5042489

## $A
## A
##         no        yes
## 0.98928661 0.01071339
##
## $L
## L
##        no       yes
## 0.8869423 0.1130577
##
## $D
## D
##        no       yes
## 0.3806469 0.6193531

## , , T = no, L = no, B = no, D = no, E = no, X = no
##
##      S
## A              no         yes
```

```
##   no  0.295847195 0.104997297
##   yes 0.002131739 0.001075716
##
## , , T = yes, L = no, B = no, D = no, E = no, X = no
##
##      S
## A              no          yes
##   no  1.153741e-06 6.423348e-07
##   yes 6.658985e-08 5.271260e-08
##
## , , T = no, L = yes, B = no, D = no, E = no, X = no
##
##      S
## A              no          yes
##   no  7.810737e-07 2.722758e-06
##   yes 5.628058e-09 2.789515e-08
##
## , , T = yes, L = yes, B = no, D = no, E = no, X = no
##
##      S
## A              no          yes
##   no  7.534379e-08 6.670155e-07
##   yes 4.348579e-09 5.473799e-08
##
## , , T = no, L = no, B = yes, D = no, E = no, X = no
##
##      S
## A              no          yes
##   no  0.0295079775 0.0620987796
##   yes 0.0002126209 0.0006362132
##
## , , T = yes, L = no, B = yes, D = no, E = no, X = no
##
##      S
## A              no          yes
##   no  1.575577e-06 1.959802e-06
##   yes 9.093679e-08 1.608293e-07
##
## , , T = no, L = yes, B = yes, D = no, E = no, X = no
##
##      S
## A              no          yes
##   no  8.241096e-08 2.913877e-06
##   yes 5.938155e-10 2.985320e-08
##
## , , T = yes, L = yes, B = yes, D = no, E = no, X = no
##
##      S
## A              no          yes
##   no  7.534379e-08 6.670155e-07
##   yes 4.348579e-09 5.473799e-08
##
## , , T = no, L = no, B = no, D = yes, E = no, X = no
##
```

```
##      S
## A              no          yes
##   no  0.0328037141 0.0116421631
##   yes 0.0002363685 0.0001192761
##
## , , T = yes, L = no, B = no, D = yes, E = no, X = no
##
##      S
## A              no          yes
##   no  6.747634e-06 3.756686e-06
##   yes 3.894498e-07 3.082888e-07
##
## , , T = no, L = yes, B = no, D = yes, E = no, X = no
##
##      S
## A              no          yes
##   no  1.721722e-06 6.001778e-06
##   yes 1.240593e-08 6.148930e-08
##
## , , T = yes, L = yes, B = no, D = yes, E = no, X = no
##
##      S
## A              no          yes
##   no  7.534379e-08 6.670155e-07
##   yes 4.348579e-09 5.473799e-08
##
## , , T = no, L = no, B = yes, D = yes, E = no, X = no
##
##      S
## A              no         yes
##   no  0.1115207602 0.234692571
##   yes 0.0008035673 0.002404468
##
## , , T = yes, L = no, B = yes, D = yes, E = no, X = no
##
##      S
## A              no          yes
##   no  5.453922e-06 6.783930e-06
##   yes 3.147812e-07 5.567168e-07
##
## , , T = no, L = yes, B = yes, D = yes, E = no, X = no
##
##      S
## A              no          yes
##   no  4.243856e-07 1.500538e-05
##   yes 3.057928e-09 1.537328e-07
##
## , , T = yes, L = yes, B = yes, D = yes, E = no, X = no
##
##      S
## A              no          yes
##   no  2.486345e-06 2.201151e-05
##   yes 1.435031e-07 1.806354e-06
##
```

```
## , , T = no, L = no, B = no, D = no, E = yes, X = no
##
##      S
## A               no          yes
##   no  8.293870e-08 2.943526e-08
##   yes 5.976181e-10 3.015696e-10
##
## , , T = yes, L = no, B = no, D = no, E = yes, X = no
##
##      S
## A               no          yes
##   no  2.473215e-06 1.376940e-06
##   yes 1.427453e-07 1.129973e-07
##
## , , T = no, L = yes, B = no, D = no, E = yes, X = no
##
##      S
## A               no          yes
##   no  1.288010e-05 4.489895e-05
##   yes 9.280807e-08 4.599979e-07
##
## , , T = yes, L = yes, B = no, D = no, E = yes, X = no
##
##      S
## A               no          yes
##   no  1.068358e-08 9.458132e-08
##   yes 6.166189e-10 7.761726e-09
##
## , , T = no, L = no, B = yes, D = no, E = yes, X = no
##
##      S
## A               no          yes
##   no  8.272356e-09 1.740896e-08
##   yes 5.960679e-11 1.783579e-10
##
## , , T = yes, L = no, B = yes, D = no, E = yes, X = no
##
##      S
## A               no          yes
##   no  3.377485e-06 4.201127e-06
##   yes 1.949366e-07 3.447615e-07
##
## , , T = no, L = yes, B = yes, D = no, E = yes, X = no
##
##      S
## A               no          yes
##   no  1.358977e-06 4.805056e-05
##   yes 9.792164e-09 4.922867e-07
##
## , , T = yes, L = yes, B = yes, D = no, E = yes, X = no
##
##      S
## A               no          yes
##   no  1.068358e-08 9.458132e-08
```

```
##   yes 6.166189e-10 7.761726e-09
##
## , , T = no, L = no, B = no, D = yes, E = yes, X = no
##
##      S
## A              no          yes
##   no  9.196293e-09 3.263799e-09
##   yes 6.626426e-11 3.343821e-11
##
## , , T = yes, L = no, B = no, D = yes, E = yes, X = no
##
##      S
## A              no          yes
##   no  1.446456e-05 8.053015e-06
##   yes 8.348435e-07 6.608630e-07
##
## , , T = no, L = yes, B = no, D = yes, E = yes, X = no
##
##      S
## A              no          yes
##   no  2.839162e-05 9.897081e-05
##   yes 2.045769e-07 1.013974e-06
##
## , , T = yes, L = yes, B = no, D = yes, E = yes, X = no
##
##      S
## A              no          yes
##   no  1.068358e-08 9.458132e-08
##   yes 6.166189e-10 7.761726e-09
##
## , , T = no, L = no, B = yes, D = yes, E = yes, X = no
##
##      S
## A              no          yes
##   no  3.126407e-08 6.579443e-08
##   yes 2.252745e-10 6.740758e-10
##
## , , T = yes, L = no, B = yes, D = yes, E = yes, X = no
##
##      S
## A              no          yes
##   no  1.169129e-05 1.454236e-05
##   yes 6.747804e-07 1.193405e-06
##
## , , T = no, L = yes, B = yes, D = yes, E = yes, X = no
##
##      S
## A              no          yes
##   no  6.998224e-06 2.474424e-04
##   yes 5.042598e-08 2.535092e-06
##
## , , T = yes, L = yes, B = yes, D = yes, E = yes, X = no
##
##      S
```

```
## A              no          yes
##   no  3.525582e-07 3.121184e-06
##   yes 2.034842e-08 2.561370e-07
##
## , , T = no, L = no, B = no, D = no, E = no, X = yes
##
##      S
## A              no          yes
##   no  1.288638e-02 4.573425e-03
##   yes 9.285333e-05 4.685556e-05
##
## , , T = yes, L = no, B = no, D = no, E = no, X = yes
##
##      S
## A              no          yes
##   no  5.025411e-08 2.797853e-08
##   yes 2.900491e-09 2.296032e-09
##
## , , T = no, L = yes, B = no, D = no, E = no, X = yes
##
##      S
## A              no          yes
##   no  3.402166e-08 1.185967e-07
##   yes 2.451444e-10 1.215044e-09
##
## , , T = yes, L = yes, B = no, D = no, E = no, X = yes
##
##      S
## A              no          yes
##   no  3.281791e-09 2.905356e-08
##   yes 1.894135e-10 2.384253e-09
##
## , , T = no, L = no, B = yes, D = no, E = no, X = yes
##
##      S
## A              no          yes
##   no  1.285295e-03 2.704871e-03
##   yes 9.261247e-06 2.771189e-05
##
## , , T = yes, L = no, B = yes, D = no, E = no, X = yes
##
##      S
## A              no          yes
##   no  6.862829e-08 8.536417e-08
##   yes 3.960984e-09 7.005330e-09
##
## , , T = no, L = yes, B = yes, D = no, E = no, X = yes
##
##      S
## A              no          yes
##   no  3.589619e-09 1.269214e-07
##   yes 2.586515e-11 1.300332e-09
##
## , , T = yes, L = yes, B = yes, D = no, E = no, X = yes
```

```
##
##      S
## A               no          yes
##   no  3.281791e-09 2.905356e-08
##   yes 1.894135e-10 2.384253e-09
##
## , , T = no, L = no, B = no, D = yes, E = no, X = yes
##
##      S
## A               no          yes
##   no  1.428849e-03 5.071041e-04
##   yes 1.029563e-05 5.195373e-06
##
## , , T = yes, L = no, B = no, D = yes, E = no, X = yes
##
##      S
## A               no          yes
##   no  2.939104e-07 1.636320e-07
##   yes 1.696348e-08 1.342831e-08
##
## , , T = no, L = yes, B = no, D = yes, E = no, X = yes
##
##      S
## A               no          yes
##   no  7.499397e-08 2.614228e-07
##   yes 5.403721e-10 2.678323e-09
##
## , , T = yes, L = yes, B = no, D = yes, E = no, X = yes
##
##      S
## A               no          yes
##   no  3.281791e-09 2.905356e-08
##   yes 1.894135e-10 2.384253e-09
##
## , , T = no, L = no, B = yes, D = yes, E = no, X = yes
##
##      S
## A               no          yes
##   no  4.857571e-03 0.0102226336
##   yes 3.500142e-05 0.0001047327
##
## , , T = yes, L = no, B = yes, D = yes, E = no, X = yes
##
##      S
## A               no          yes
##   no  2.375595e-07 2.954914e-07
##   yes 1.371110e-08 2.424922e-08
##
## , , T = no, L = yes, B = yes, D = yes, E = no, X = yes
##
##      S
## A               no          yes
##   no  1.848520e-08 6.535976e-07
##   yes 1.331958e-10 6.696225e-09
```

```
##
## , , T = yes, L = yes, B = yes, D = yes, E = no, X = yes
##
##      S
## A               no          yes
##   no  1.082991e-07 9.587675e-07
##   yes 6.250644e-09 7.868035e-08
##
## , , T = no, L = no, B = no, D = no, E = yes, X = yes
##
##      S
## A               no          yes
##   no  1.029361e-05 3.653243e-06
##   yes 7.417105e-08 3.742813e-08
##
## , , T = yes, L = no, B = no, D = no, E = yes, X = yes
##
##      S
## A               no          yes
##   no  3.069534e-04 1.708936e-04
##   yes 1.771627e-05 1.402422e-05
##
## , , T = no, L = yes, B = no, D = no, E = yes, X = yes
##
##      S
## A               no          yes
##   no  1.598563e-03 5.572459e-03
##   yes 1.151851e-05 5.709085e-05
##
## , , T = yes, L = yes, B = no, D = no, E = yes, X = yes
##
##      S
## A               no          yes
##   no  1.325951e-06 1.173859e-05
##   yes 7.652925e-08 9.633165e-07
##
## , , T = no, L = no, B = yes, D = no, E = yes, X = yes
##
##      S
## A               no          yes
##   no  1.026691e-06 2.160645e-06
##   yes 7.397865e-09 2.213620e-08
##
## , , T = yes, L = no, B = yes, D = no, E = yes, X = yes
##
##      S
## A               no          yes
##   no  4.191834e-04 5.214066e-04
##   yes 2.419379e-05 4.278874e-05
##
## , , T = no, L = yes, B = yes, D = no, E = yes, X = yes
##
##      S
## A               no          yes
```

```
##   no  1.686642e-04 5.963609e-03
##   yes 1.215316e-06 6.109825e-05
##
## , , T = yes, L = yes, B = yes, D = no, E = yes, X = yes
##
##      S
## A              no          yes
##   no  1.325951e-06 1.173859e-05
##   yes 7.652925e-08 9.633165e-07
##
## , , T = no, L = no, B = no, D = yes, E = yes, X = yes
##
##      S
## A              no          yes
##   no  1.141362e-06 4.050738e-07
##   yes 8.224131e-09 4.150054e-09
##
## , , T = yes, L = no, B = no, D = yes, E = yes, X = yes
##
##      S
## A              no          yes
##   no  0.0017952124 9.994686e-04
##   yes 0.0001036134 8.202044e-05
##
## , , T = no, L = yes, B = no, D = yes, E = yes, X = yes
##
##      S
## A              no          yes
##   no  3.523715e-03 0.0122833772
##   yes 2.539027e-05 0.0001258454
##
## , , T = yes, L = yes, B = no, D = yes, E = yes, X = yes
##
##      S
## A              no          yes
##   no  1.325951e-06 1.173859e-05
##   yes 7.652925e-08 9.633165e-07
##
## , , T = no, L = no, B = yes, D = yes, E = yes, X = yes
##
##      S
## A              no          yes
##   no  3.880218e-06 8.165819e-06
##   yes 2.795907e-08 8.366029e-08
##
## , , T = yes, L = no, B = yes, D = yes, E = yes, X = yes
##
##      S
## A              no          yes
##   no  1.451019e-03 0.0018048689
##   yes 8.374774e-05 0.0001481149
##
## , , T = no, L = yes, B = yes, D = yes, E = yes, X = yes
##
```

```
##      S
## A               no          yes
##   no  8.685573e-04 0.0307103541
##   yes 6.258425e-06 0.0003146331
##
## , , T = yes, L = yes, B = yes, D = yes, E = yes, X = yes
##
##      S
## A               no          yes
##   no  4.375639e-05 3.873736e-04
##   yes 2.525465e-06 3.178944e-05

## s
##   no  yes
## 2543 2457
```

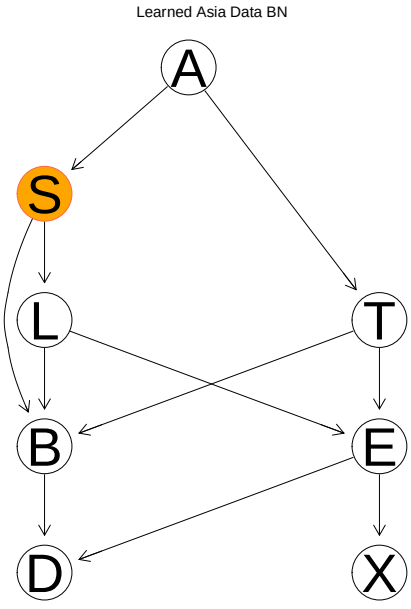
Question 2

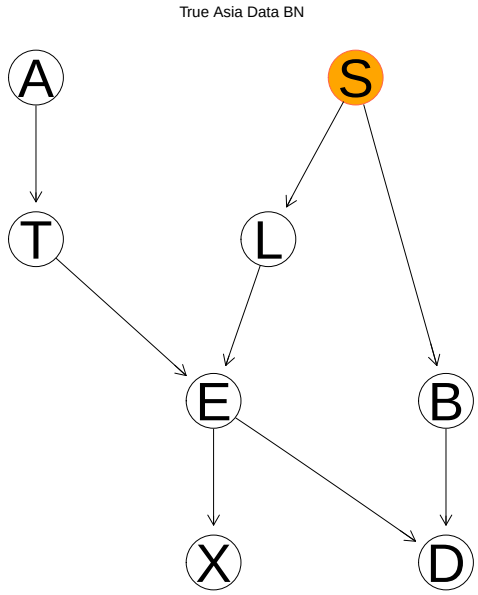
Prediction is completed

```
## confusion matrix

##      predForS
##          no   yes
##   no  0.322 0.146
##   yes 0.120 0.412
```

Writing common function for Fit and Predict






```
## Confusion Matrix for Learned

##      prediction
##          no   yes
##   no  0.322 0.146
##   yes 0.120 0.412
```

```
## Confusion Matrix for True DAG

##      prediction
##          no   yes
##   no  0.322 0.146
##   yes 0.120 0.412
```

Question 3

The Markov blanket of a node Xi, the set of nodes that completely separates Xi from the rest of the graph. Generally speaking, it is the set of nodes that includes all the knowledge needed to do inference on Xi, from estimation to hypothesis testing to prediction: the parents of Xi, the children of Xi, and those children's other parents.

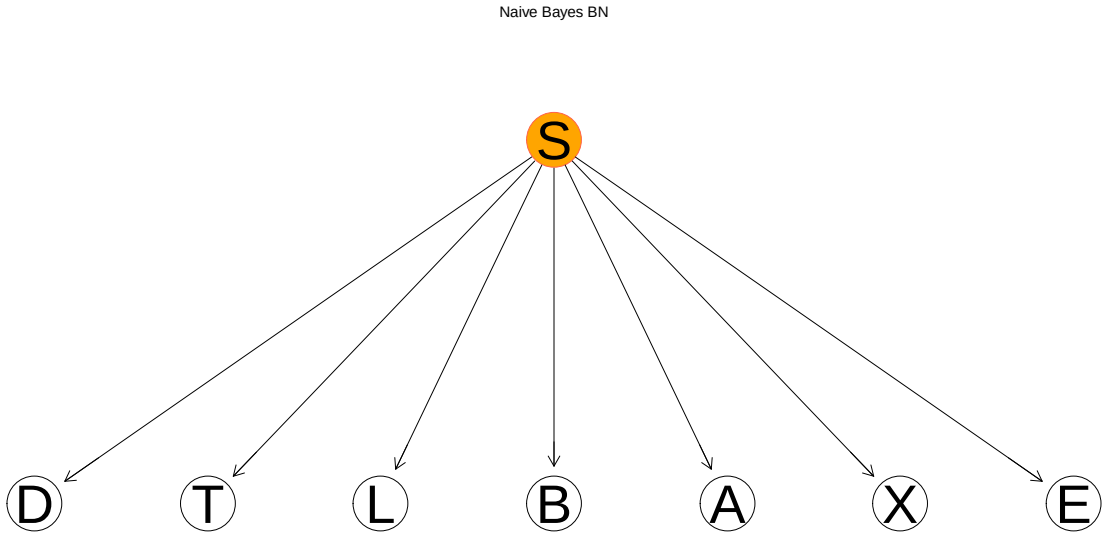
Ref : Doc

```
## Confusion Matrix for Markov Blanket of S

##      prediction
##          no   yes
##   no  0.322 0.146
##   yes 0.120 0.412
```

Question 4

Naive Bayes Assumption : In Naive Bayes, we use Bayes Theorem with strong Naive assumption we consider independence between features. Since here we have features namely "D" , "T", "L", "B", "A", "X", "E" while target is "S", we will assume all features are independent of each other.




```
## Confusion Matrix for Naive Bayes BN for S

##      prediction
##          no   yes
##   no  0.349 0.119
##   yes 0.188 0.344
```

Question 5

Written in FinalFile to be Submitted.

Appendix

```
knitr::opts_chunk$set(echo = TRUE,fig.height = 8,fig.width =16 )
#Requireinstall packages()
#install.packages("bnlearn")
#install.packages("BiocManager")
#library(BiocManager)
#BiocManager::install("RBGL")
#install.packages("gRain")
#install.packages("Rgraphviz")
#install.packages("qgraph")

library(gRain)
library(bnlearn)
library(Rgraphviz)


#graph without any inital structure is used here
#field start is set as null
bn.hc1 = hc(asia , score = "aic" , restart = 0)
# cat("Network with parameter Score = AIC and Restart = 5")
# cat("\n")
# print(bn.hc)

# cat("\n")
bn.hc2 = hc(asia , score = "aic" , restart = 1)
# cat("Network with parameter Score = AIC and Restart = 1")
# cat("\n")
# print(bn.hc2)
# cat("\n")


#intiate network
initiate = bnlearn::empty.graph(nodes = c("D" , "T", "L", "B", "A", "S", "X", "E"))

#initiate = bnlearn::compare(bn.hc, bn.hc2 , arcs = TRUE)

initiate = bnlearn::set.arc(initiate, from = "A" , to = "S" )


#using same config except different score methods

bn.hc3 = bnlearn::hc(asia , score = "aic" , restart = 1 , start = initiate)


# initiate2 = bnlearn::set.arc(initiateG, from = "A" , to = "X" )

#when score is set to Bayesian Dirichlet equivalent score (bde)
bn.hc4 = bnlearn::hc(asia , score = "bde" , restart = 1
                     , start = initiate
                     )
```

```
bn.hc4_noInit = bnlearn::hc(asia , score = "bde" , restart = 1

                      #, start = initiate
                      )


# initiate3 = bnlearn::set.arc(initiateG, from = "" , to = "X" )
#
# #when score is set to Bayesian Dirichlet equivalent score (bde)
# bn.hc4 = bnlearn::hc(asia , score = "bde" , restart = 5
#                       #, start = initiate2
#                       )


bn.hc5 = bnlearn::hc(asia, score = "bic" , restart = 1 , start = initiate)

bn.hc5_noInit = bnlearn::hc(asia, score = "bic" , restart = 1
                      #, start = initiate
                      )



#plot(bn.hc)
#library(Rgraphviz)
#graphviz.plot(bn.hc)
#library(qgraph)

par(mfrow = c(1, 3))
bnlearn::graphviz.plot(bn.hc1 , main = "1")
bnlearn::graphviz.plot(bn.hc2, main = "2")
bnlearn::graphviz.plot(bn.hc3, main = "3")


par(mfrow = c(2,2))
bnlearn::graphviz.plot(bn.hc4, main = "4")
bnlearn::graphviz.plot(bn.hc4_noInit, main = "4 no init")
bnlearn::graphviz.plot(bn.hc5, main = "5")
bnlearn::graphviz.plot(bn.hc5_noInit, main = "5 no init")

#check with arcs , vstructs, cpdag and all.equal


#graph without any inital structure is used here
#field start is set as null
bn.hc = hc(asia , score = "aic" , restart = 5)
# cat("Network with parameter Score = AIC and Restart = 5")
# cat("\n")
# print(bn.hc)

# cat("\n")
bn.hc2 = hc(asia , score = "aic" , restart = 1)
# cat("Network with parameter Score = AIC and Restart = 1")
# cat("\n")
```

```
# print(bn.hc2)
# cat("\n")


#intiate network
initiate = bnlearn::empty.graph(nodes = c("D" , "T", "L", "B", "A", "S", "X", "E"))

#initiate = bnlearn::compare(bn.hc, bn.hc2 , arcs = TRUE)

initiate = bnlearn::set.arc(initiate, from = "A" , to = "X" )


#using same config except different score methods

bn.hc3 = bnlearn::hc(asia , score = "aic" , restart = 1 , start = initiate)


# initiate2 = bnlearn::set.arc(initiateG, from = "A" , to = "X" )

#when score is set to Bayesian Dirichlet equivalent score (bde)
bn.hc4 = bnlearn::hc(asia , score = "bde" , restart = 1
                     , start = initiate
                     )

bn.hc4_noInit = bnlearn::hc(asia , score = "bde" , restart = 1

                     #, start = initiate
                     )


# initiate3 = bnlearn::set.arc(initiateG, from = "" , to = "X" )
#
# #when score is set to Bayesian Dirichlet equivalent score (bde)
# bn.hc4 = bnlearn::hc(asia , score = "bde" , restart = 5
#                       #, start = initiate2
#                       )


bn.hc5 = bnlearn::hc(asia, score = "bic" , restart = 1 , start = initiate)

bn.hc5_noInit = bnlearn::hc(asia, score = "bic" , restart = 1
                     #, start = initiate
                     )




#plot(bn.hc)
#library(Rgraphviz)
#graphviz.plot(bn.hc)
#library(qgraph)
```

```r
par(mfrow = c(1, 3))
bnlearn::graphviz.plot(bn.hc , main = "1")
bnlearn::graphviz.plot(bn.hc2, main = "2")
bnlearn::graphviz.plot(bn.hc3, main = "3")


par(mfrow = c(2,2))
bnlearn::graphviz.plot(bn.hc4, main = "4")
bnlearn::graphviz.plot(bn.hc4_noInit, main = "4 no init")
bnlearn::graphviz.plot(bn.hc5, main = "5")
bnlearn::graphviz.plot(bn.hc5_noInit, main = "5 no init")


asiaData = bnlearn::asia

#Fetching 80% of Data
n = dim(asiaData)[1]
suppressWarnings(RNGversion("3.5.9"))
set.seed(12345)
id = sample(1:n , floor(n*0.8))
train = asiaData[id,]
test = asiaData[-id , ]


#Inference Part

#creating object of class bn

#initiate network
initiate = bnlearn::empty.graph(nodes = c("D" , "T", "L", "B", "A",  "X", "E" , "S"))
initiate = bnlearn::set.arc(initiate, from = "T" , to = "D" )

bn.hc = bnlearn::hc(asia , score = "aic" , restart = 10 ,
                    start = initiate
                    )

cat("Score Based Structure")
bn.hc
cat("\n")
cat("Graph associated with a Bayesian network ")
cat("\n")
bnlearn::graphviz.plot(bn.hc, main = "Bayesian Network"
                       #,  layout = "neato",
                       ,highlight = list(nodes = "S" ,  col = "tomato", fill = "orange") )
cat("\n")

#fit
cat("Fitting Parameter of Bayesian Model ")

#*****

fitting = bn.fit(x = bn.hc , data =  train , method = "bayes"
                 #, debug = TRUE
```

```
                  )

#cat("Conditional Probability of node S")

#bn.fit.barchart(fitting$S)

#*****
cat("\n")

cat("Bayesian network as a list of conditional probability tables")

#grain object
cat("\n")
grainObject = bnlearn::as.grain(fitting)
cat("\n")
#grainObject
cat("\n")


#compile conditional probabilities
compiled = compile(object = grainObject)

#calling setfinding

findings_Yes = setFinding(grainObject , nodes = c("S") , states = c("yes" ))
findings_No = setFinding(grainObject , nodes = c("S") , states = c("no"))

#Stating Evidence

cat("Under selected model Probability of observing this evidence for S = Yes is ")
cat("\n")
pEvidence(findings_Yes)
cat("\n")
cat("Under selected model Probability of observing this evidence for S = No is ")
cat("\n")
pEvidence(findings_No)
cat("\n")


#Querying Network
querygrain(findings_Yes , nodes = c("L" , "A" , "D"))

#Furthur Exploration with querygrain

tt = querygrain(grainObject , type = "joint")
tt

prob_table = prop.table(tt)

tb = table(prob_table)


#querying on data
```

```
s = cpdist(fitting , nodes = "S" , evidence = TRUE , method = )

table(s)

#creating DAG
initiate = bnlearn::empty.graph(nodes = c("D" , "T", "L", "B", "A",  "X", "E" , "S"))
initiate = bnlearn::set.arc(initiate, from = "T" , to = "D" )
bn.hc = bnlearn::hc(asia , score = "aic" , restart = 10 , start = initiate )
#fitting data to DAC
fitData = bn.fit(x = bn.hc , data =  train , method = "bayes")
#converting to grain object and compiling
compiled_Grain = compile(object = bnlearn::as.grain(fitData))

#Prediction on Test Data except S
nodesName = colnames(test[-2])

#testRowSet = as.vector(unname(unlist(test[1, ])))

predForS = numeric(nrow(test))

for(i in 1:nrow(test)){

  #nodes : A vector of nodes; those nodes for which the (conditional) distribution is requested.
  #states : A vector of states (of the nodes given by 'nodes')

  States = as.vector(unname(unlist(test[i, -2])))
  Evidence_Grain = gRain::setEvidence(object = compiled_Grain ,
                                      nodes = nodesName ,
                                      states = States
                                      )

  queryGrain = querygrain(object = Evidence_Grain , nodes = "S" ,
                          #type = "joint"
                          type = "marginal" )


  predForS[i] = ifelse( queryGrain$S[1] < queryGrain$S[2] ,  "yes" , "no")


}


confusionMatrix = prop.table(table(test$S , predForS))

#resProb = sum(diag(propTable))
cat("\n")
cat("confusion matrix")

cat("\n")
confusionMatrix
cat("\n")

fnPredict = function(trainData , testData , DAGraph.bnObject , predNode = "S" ,
```

```
                    Method = "bayes" , markovB = FALSE , markovObj = NULL){

#fit bn model to training Data
modelFit = bnlearn::bn.fit(x = DAGraph.bnObject , data = trainData ,
                           method = Method)

# The main data structure in gRain is the grain class, which stores a fitted Bayesian network as a li
# conditional probability tables (much like bnlearn's bn.fit objects) and makes it possible for setEv
# and querygrain() to perform posterior inference via belief propagation.  #via docs

# #converting to grain object and compiling
compiled_Grain = gRbase::compile(object = bnlearn::as.grain(modelFit))
#compiled_Grain = bnlearn::as.grain(modelFit)


#fecth index of predNode in order to remove that while making predictions


if(markovB == FALSE)
  {
    indx = which(colnames(testData) == predNode)
    nodesName = colnames(testData[-indx])
  }else{
    nodesName = as.vector(markovObj)
  }

prediction = numeric(nrow(testData))

#Added to see changing probabilites such that we can validate confusion matrix
#dfPredTest = data.frame(matrix(data = NA , ncol = 2 , nrow = nrow(testData)))


for(i in 1:nrow(testData))
{

  #In setEvidence function
  #nodes : A vector of nodes; those nodes for which the (conditional) distribution is requested.
  #states : A vector of states (of the nodes given by 'nodes')

  #fetching ith row of data set to predicted without target
  #extending logic for markov blanket case model

  if(markovB == FALSE){
    testDataRow.State = as.vector(unname(unlist(testData[i, -indx])))

  }else{
    testDataRow.State = as.vector(unname(unlist(testData[i, nodesName])))
  }

  # if(i == 1)
  #   print(testDataRow.State)

  Evidence_Grain = gRain::setEvidence(object = compiled_Grain ,
```

```
                                        nodes = nodesName ,
                                        states = testDataRow.State
                                        )

    queryGrain = querygrain(object = Evidence_Grain,
                            #object = compiled_Grain , evidence = Evidence_Grain ,
                             nodes = predNode ,
                            #type = "joint"
                            type = "marginal"
                            )

    qG = as.vector(unname(unlist(queryGrain)))

    #qG[1] = No and qG[2] = Yes
    #Added for visual validation of result @removed from final result
    #dfPredTest[i,] = qG

    prediction[i] = ifelse(qG[2] > qG[1]   ,  "yes" , "no")

  }#for

  #confusion matrix
  confusionMatrix = prop.table(table(test$S , prediction))

  #return(list("Cmat" = confusionMatrix , "df" = dfPredTest))
  return(confusionMatrix)

}#fnPredict

#plot both Learned and True Graph
#creating DAG
initiate = bnlearn::empty.graph(nodes = c("D" , "T", "L", "B", "A",  "X", "E" , "S"))
initiate = bnlearn::set.arc(initiate, from = "T" , to = "D" )
bn.hc = bnlearn::hc(asia , score = "aic" , restart = 10 , start = initiate )

#True
dag.True  = bnlearn::model2network(string = "[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")


par(mfrow = c(1,2))
bnlearn::graphviz.plot(bn.hc2, main = "Learned Asia Data BN"
                       #,  layout = "neato",
                       ,highlight = list(nodes = "S" ,  col = "tomato",
                                         fill = "orange") )
bnlearn::graphviz.plot(dag.True, main = "True Asia Data BN"
                       #,  layout = "neato",
                       ,highlight = list(nodes = "S" ,  col = "tomato",
                                         fill = "orange") )



asiaData = bnlearn::asia
```

```r
#Fetching 80% of Data
n = dim(asiaData)[1]
suppressWarnings(RNGversion("3.5.9"))
set.seed(12345)
id = sample(1:n , floor(n*0.8))
#id = seq(1 , 4000)
train = asiaData[id,]
test = asiaData[-id , ]


customModel = fnPredict(trainData = train , testData = test , DAGraph.bnObject = bn.hc2 , predNode = "S
cat("\n")
cat("Confusion Matrix for Learned")
cat("\n")
customModel
cat("\n")


trueModel = fnPredict(trainData = train , testData = test , DAGraph.bnObject = dag.True , predNode = "S
cat("\n")
cat("Confusion Matrix for True DAG")
cat("\n")
trueModel
cat("\n")
#calculating MarkovBlanket of S
bn.mb = mb(bn.hc , node = "S")

MarkovBNPred = fnPredict(trainData = train , testData = test , DAGraph.bnObject = bn.hc
                 , predNode = "S" , Method = "mle" , markovB = TRUE ,
                 markovObj = bn.mb)

cat("Confusion Matrix for Markov Blanket of S")
cat("\n")
MarkovBNPred
cat("\n")



NaiveBayesBN = bnlearn::empty.graph(nodes = c("D" , "T", "L", "B", "A",  "X", "E" , "S"))
NaiveBayesBN = bnlearn::set.arc(NaiveBayesBN, from = "S" , to = "D" )
NaiveBayesBN = bnlearn::set.arc(NaiveBayesBN, from = "S" , to = "T" )
NaiveBayesBN = bnlearn::set.arc(NaiveBayesBN, from = "S" , to = "L" )
NaiveBayesBN = bnlearn::set.arc(NaiveBayesBN, from = "S" , to = "B" )
NaiveBayesBN = bnlearn::set.arc(NaiveBayesBN, from = "S" , to = "A" )
NaiveBayesBN = bnlearn::set.arc(NaiveBayesBN, from = "S" , to = "X" )
NaiveBayesBN = bnlearn::set.arc(NaiveBayesBN, from = "S" , to = "E" )

bnlearn::graphviz.plot(NaiveBayesBN, main = "Naive Bayes BN"
                       ,highlight = list(nodes = "S" ,  col = "tomato",
                                         fill = "orange") )


NaiveBayesBNPred = fnPredict(trainData = train , testData = test , DAGraph.bnObject = NaiveBayesBN
```

```
                  , predNode = "S" , Method = "mle" , markovB = FALSE  )

cat("Confusion Matrix for Naive Bayes BN for S")
cat("\n")
NaiveBayesBNPred
cat("\n")
```