Graphical Model

Aman Kumar Nayak

9/5/2020

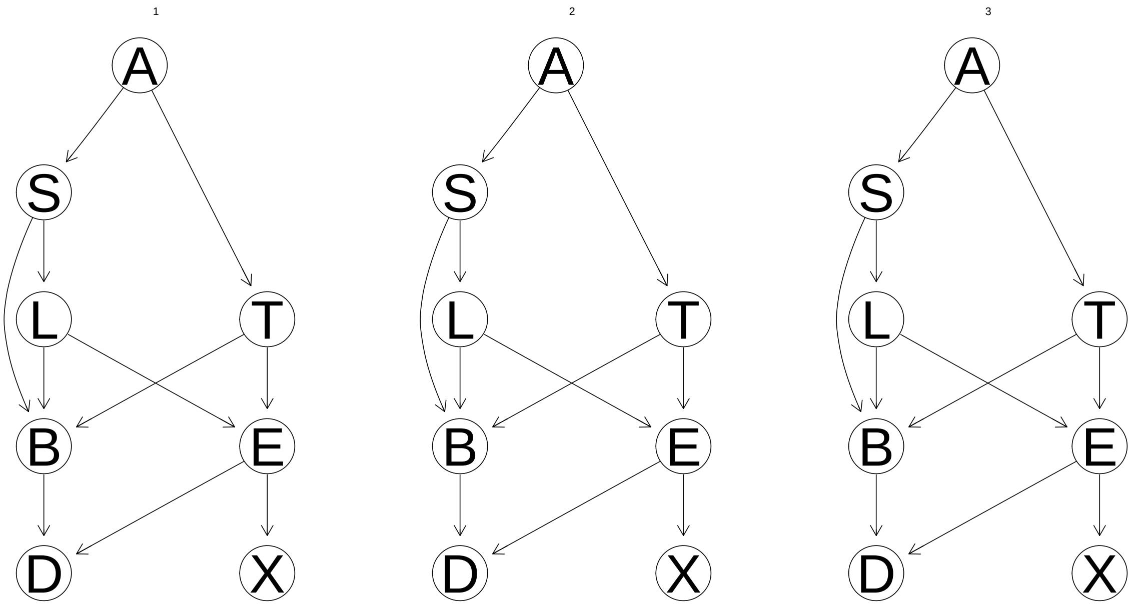
Question 1

Data : The asia data set contains the following variables:

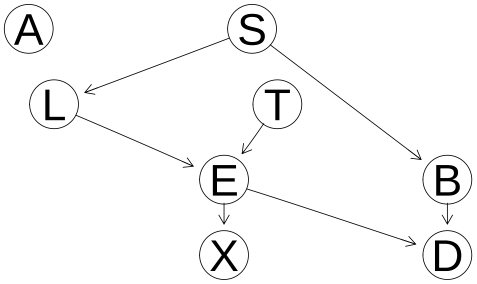
- D (dyspnoea), a two-level factor with levels yes and no.
- T (tuberculosis), a two-level factor with levels yes and no.
- L (lung cancer), a two-level factor with levels yes and no.
- B (bronchitis), a two-level factor with levels yes and no.
- A (visit to Asia), a two-level factor with levels yes 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

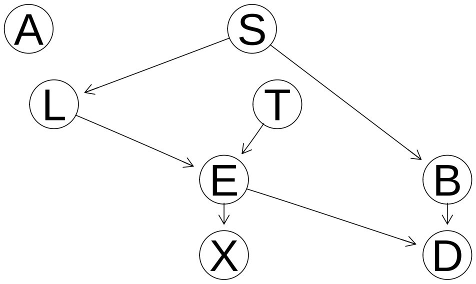
1st



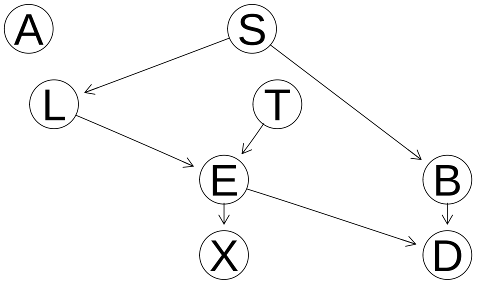
4



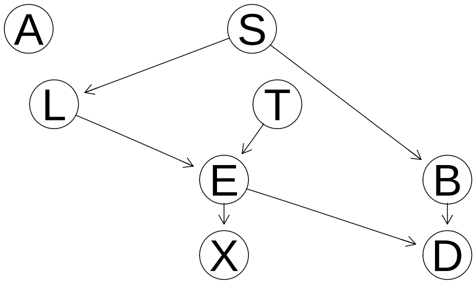
4 no init



5

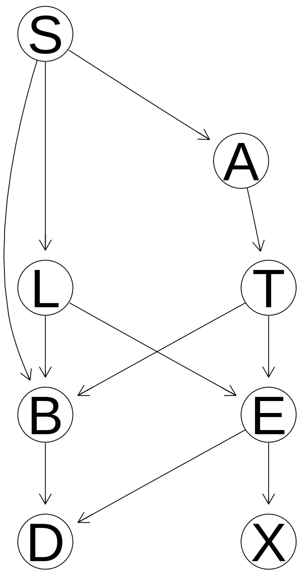


5 no init

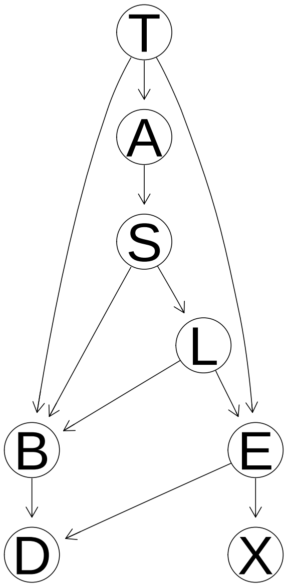


Set 2 just to check if re-run is changing graphical models

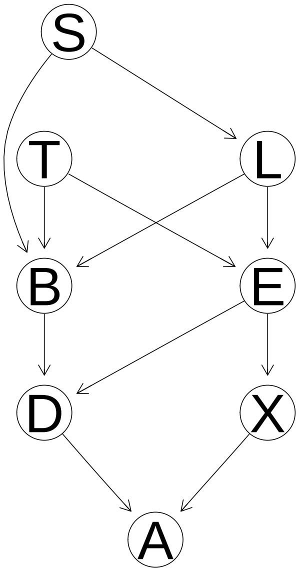
1

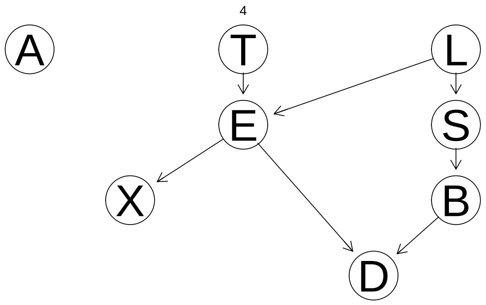


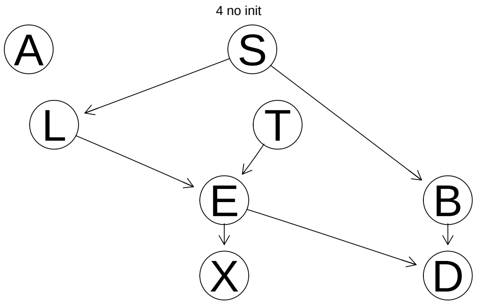
2

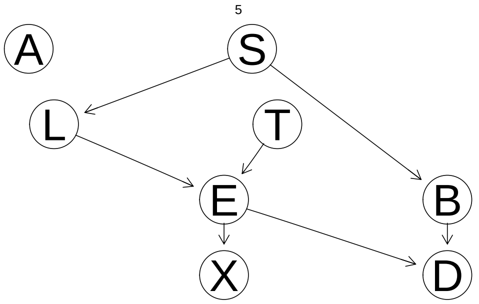


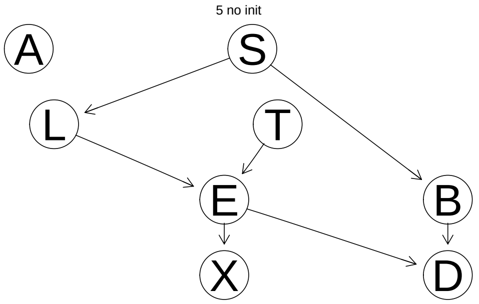
3




4


4 no init


5


5 no init

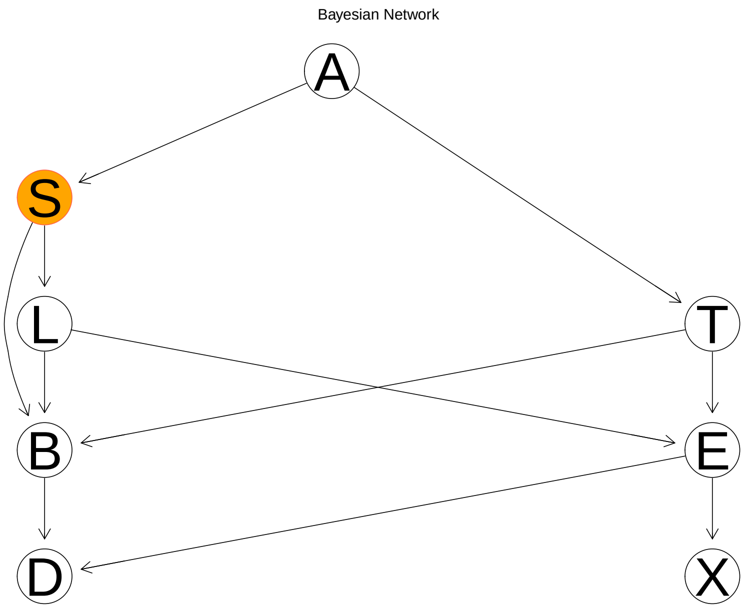
Question 2

Calculating inference for S (Smoking)

```
## Score Based Structure
```

```
##
##   Bayesian network learned via Score-based methods
##
##   model:
##    [A][S|A][T|A][L|S][B|S:T:L][E|T:L][X|E][D|B:E]
##   nodes:                                 8
##   arcs:                                  11
##     undirected arcs:                     0
##     directed arcs:                       11
##   average markov blanket size:           3.50
##   average neighbourhood size:            2.75
##   average branching factor:              1.38
##
##   learning algorithm:                    Hill-Climbing
##   score:                                 AIC (disc.)
##   penalization coefficient:              1
##   tests used in the learning procedure:  301
##   optimized:                             TRUE
```

```
## Graph associated with a Bayesian network
```



```
## Fitting Parameter of Bayesian Model
```

```
## Conditional Probability of node S
```

```
## Bayesian network as a list of conditional probability tables
```

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)
```

```r
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)

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
```

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

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