Arash

2023-09-05

# LAB 1

## Attaching The Packages and Loading the Data

First we attach the required packages for working with Baysian Networks. Also, the purr packages is used for vectorized version of some for loops. In the last line we load the data.

library(bnlearn)  
library(gRain)

## Loading required package: gRbase

##   
## Attaching package: 'gRbase'

## The following objects are masked from 'package:bnlearn':  
##   
## ancestors, children, nodes, parents

library(purrr)  
  
data("asia")

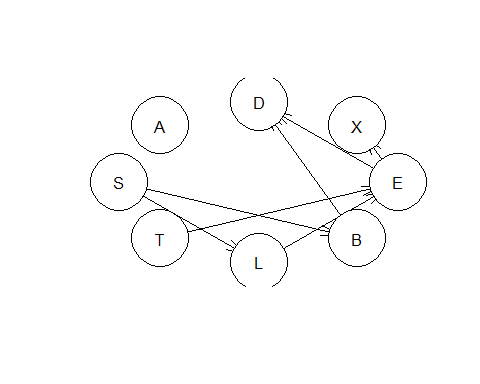
We can check the documentation of data using ?asia to get a better knowledge of the data at hand.

## Question 1

In order that we learn the structure from the data using the score based algorithms and an algorithm to maximize that score we need to use hc() which is the Hill Climbing algorithm.

First, the result of hc() can be checked using its default values

hc\_network <- hc(asia)  
plot(hc\_network)



Now, the arguments will be changed to see if we see any difference between different runs of Hill Climbing algorithm.

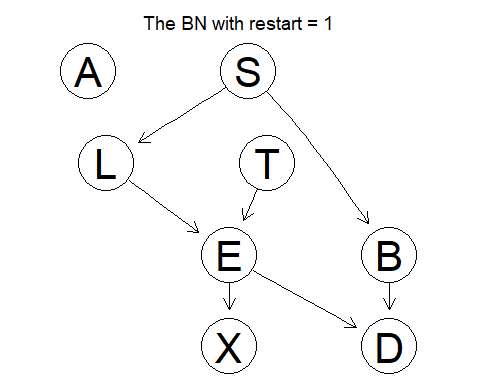
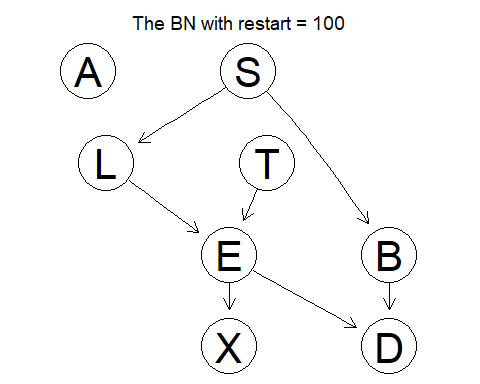
The first argument to change is restart:

hc\_network\_1 <- hc(asia, score = "bde", restart = 100)  
hc\_network\_2 <- hc(asia, score = "bde", restart = 1)  
  
graphviz.compare(hc\_network\_1, hc\_network\_2, main = c("The BN with restart = 100",   
 "The BN with restart = 1"))

## Loading required namespace: Rgraphviz

all.equal(hc\_network\_1, hc\_network\_2)

## [1] TRUE



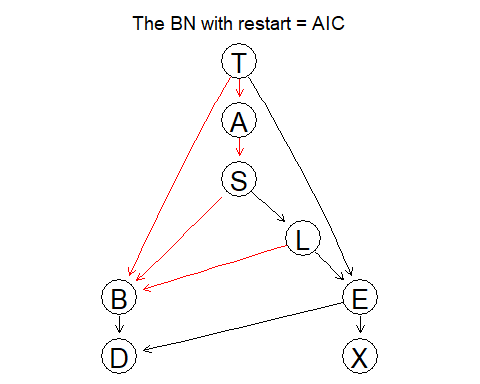
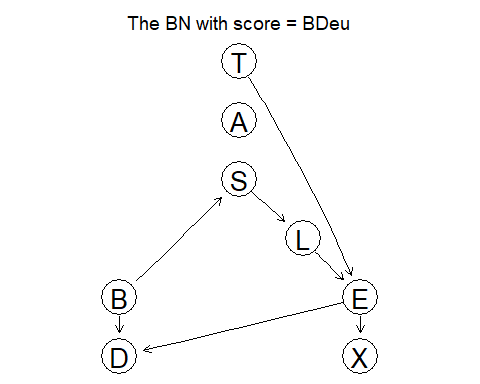
It is evidence that the DAGs are Markov equal (same dependency but different direction for connections).

Here, we check two different scores to evaluate the DAGs in HC algorithm:

hc\_network\_3 <- hc(asia, score = "bde", restart = 13)  
hc\_network\_4 <- hc(asia, score = "aic", restart = 13)  
  
  
  
all.equal(hc\_network\_3, hc\_network\_4)

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

graphviz.compare(hc\_network\_3, hc\_network\_4, main = c("The BN with score = BDeu",   
 "The BN with restart = AIC"))



The graphs will be different as we change the score method of HC algorithm. It is because the HC has to optimize different scores.

Other options to change the different numbers of arrow to be changed in each iteration of HC perturb, and the imaginary sample size iss:

hc\_network\_5 <- hc(asia, score = "bde", restart = 13, perturb = 3)  
hc\_network\_6 <- hc(asia, score = "bde", restart = 13, perturb = 7)  
all.equal(hc\_network\_5, hc\_network\_6)

## [1] "Different arc sets"

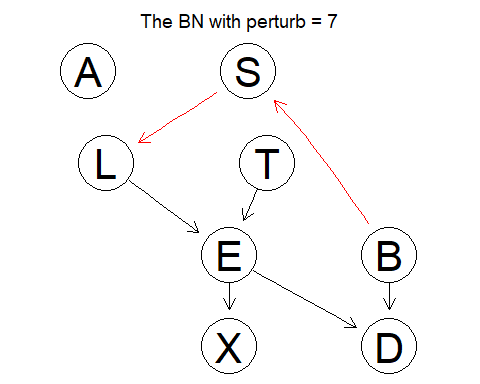
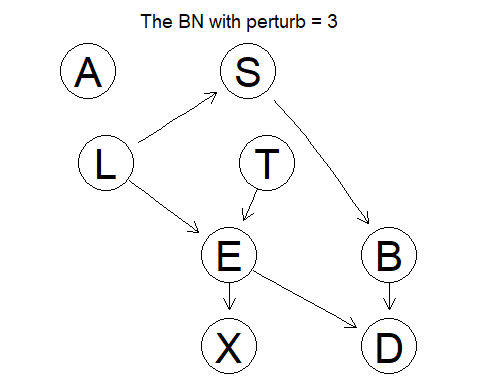
score(hc\_network\_5, asia, type = "bde")

## [1] -11095.79

score(hc\_network\_6, asia, type = "bde")

## [1] -11095.79

graphviz.compare(hc\_network\_5, hc\_network\_6, main = c("The BN with perturb = 3",   
 "The BN with perturb = 7"))



# different imaginary sample size for BDeu algorithm  
hc\_network\_7 <- hc(asia, score = "bde", restart = 13, iss = 2)  
hc\_network\_8 <- hc(asia, score = "bde", restart = 13, iss = 7)  
all.equal(hc\_network\_7, hc\_network\_8)

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

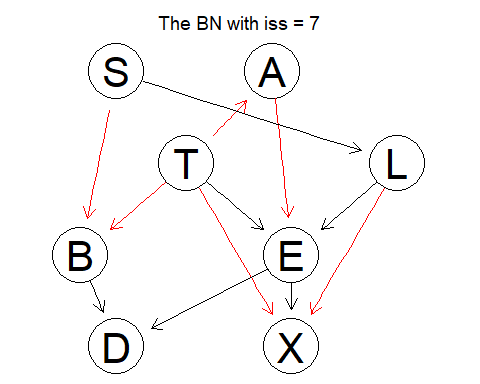
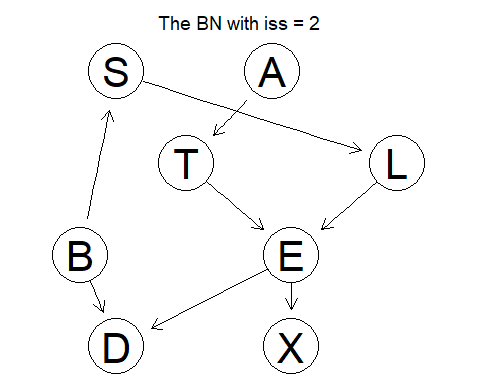
score(hc\_network\_7, asia, type = "bde")

## [1] -11096.64

score(hc\_network\_8, asia, type = "bde")

## [1] -11118.11

graphviz.compare(hc\_network\_7, hc\_network\_8, main = c("The BN with iss = 2",   
 "The BN with iss = 7"))



The ISS has to do with regularization of the graph. If iss is low, we have higher regularization hence the graph is more sparse.

Notice that HC with BDeu score and ISS of 2 manages to get the true Essential Graph.

Also, the initial graph to the HC may change the result of HC. The reason is that HC finds the local optima and different starting points may give different local optima.

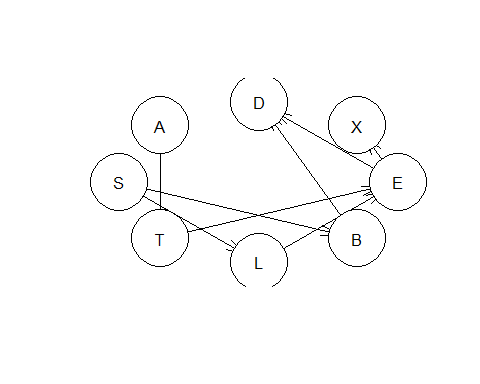
## Question 2

First, the data is split in two sets of Training and Testing where 80% of the total data is in the train data

set.seed(7)  
  
training\_idx <- sample(5000, nrow(asia) \* 0.8)  
asia\_training <- asia[training\_idx, ]  
asia\_test <- asia[-training\_idx, ]

Here, the HC with BDeu and ISS of 2 is used

bn\_dag <- hc(asia\_training, score = "bde", iss = 2, restart = 13)  
plot(bn\_dag)



Now the local table is estimated using Maximum Log-Likelihood Estimation which for the discrete case like asia data is just the relative frequency.

param\_tables <- bn.fit(bn\_dag, asia\_training, method = "mle")

### The Approx. Inference

First, the test data with S = 'NO' will be used to do approx. inference:

target\_node\_index <- 2  
evidence\_vector <-  
 map\_chr(  
 list\_transpose(as.list(asia\_test)),  
 ~ paste(  
 "(",  
 names(asia\_test)[-target\_node\_index],  
 " == '",  
 map\_chr(.x[-target\_node\_index], as.character),  
 "')",  
 sep = "",  
 collapse = " & "  
 )  
 )  
evnet <- "(S == 'yes')"  
  
cmds <- paste("cpquery(param\_tables, ", evnet, ", ", evidence\_vector, ")", sep = "")  
probs <- map\_dbl( cmds, ~eval(parse(text = .x)) )  
  
y\_hats <- ifelse(probs > 0.5, "yes", "no")  
  
  
table(y\_hats, asia\_test$S)

##   
## y\_hats no yes  
## no 339 128  
## yes 151 382

The accuracy is 0.721 .

### The Exact Inference

asia\_test\_evidence <- asia\_test[, -target\_node\_index]  
  
bn\_grain <- param\_tables %>% as.grain() %>% compile()  
  
bn\_grain\_evidences <- map\_dbl(1:nrow(asia\_test\_evidence),  
 ~ setEvidence(bn\_grain,  
 nodes = names(asia\_test\_evidence),  
 states = as.matrix(asia\_test\_evidence)[.x, ]) %>%  
 querygrain(nodes = "S") %>% pluck(1, "yes"))   
  
y\_hats <- ifelse(bn\_grain\_evidences > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)

##   
## y\_hats no yes  
## no 338 123  
## yes 152 387

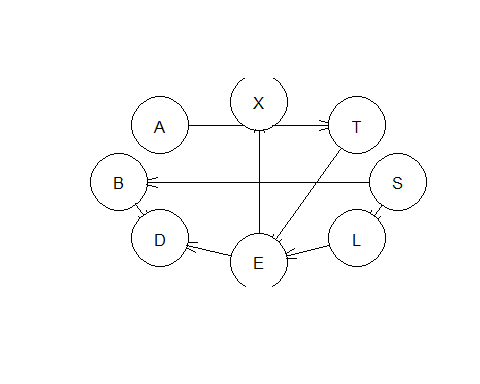
The accuracy is 0.725 .

Notice that the accuracy of the two tables are the same. Now, the true DAG will be used to do the same inference.

### The Exact Inference with The True Network

dag <- model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")  
plot(dag)  
param\_tables\_true\_dag <- bn.fit(dag, asia\_training, method = "mle")  
  
  
asia\_test\_evidence <- asia\_test[, -target\_node\_index]  
  
bn\_true\_grain <- param\_tables\_true\_dag %>% as.grain() %>% compile()  
  
bn\_grain\_evidences <- map\_dbl(1:nrow(asia\_test\_evidence),  
 ~ setEvidence(bn\_true\_grain,  
 nodes = names(asia\_test\_evidence),  
 states = as.matrix(asia\_test\_evidence)[.x, ]) %>%  
 querygrain(nodes = "S") %>% pluck(1, "yes"))   
  
y\_hats <- ifelse(bn\_grain\_evidences > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)

##   
## y\_hats no yes  
## no 338 123  
## yes 152 387

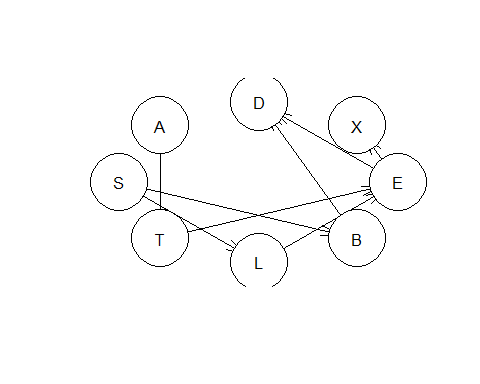


The accuracy is 0.725 .

## Question 3

set.seed(7)  
  
training\_idx <- sample(5000, nrow(asia) \* 0.8)  
asia\_training <- asia[training\_idx, ]  
asia\_test <- asia[-training\_idx, ]  
  
bn\_dag <- hc(asia\_training, score = "bde", iss = 2, restart = 13)  
plot(bn\_dag)  
  
  
mb\_s <- mb(bn\_dag, "S")  
  
  
param\_tables <- bn.fit(bn\_dag, asia\_training, method = "mle")  
  
  
asia\_test\_evidence <- asia\_test[, mb\_s]  
  
bn\_grain <- param\_tables %>% as.grain() %>% compile()  
  
bn\_grain\_evidences <- map\_dbl(1:nrow(asia\_test\_evidence),  
 ~ setEvidence(bn\_grain,  
 nodes = names(asia\_test\_evidence),  
 states = as.matrix(asia\_test\_evidence)[.x, ]) %>%  
 querygrain(nodes = "S") %>% pluck(1, "yes"))   
  
y\_hats <- ifelse(bn\_grain\_evidences > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)

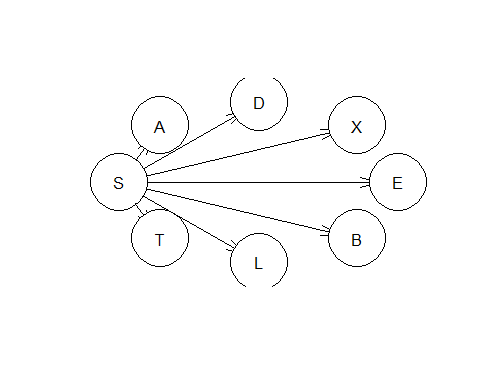
##   
## y\_hats no yes  
## no 338 123  
## yes 152 387



The accuracy is 0.725 .

## Question 4

target\_node\_index <- 2  
  
naive\_bn <- empty.graph(names(asia))  
  
naive\_arcs <- cbind("S", c(names(asia)[-target\_node\_index]))  
  
arcs(naive\_bn) <- naive\_arcs  
plot(naive\_bn)



set.seed(7)  
  
training\_idx <- sample(5000, nrow(asia) \* 0.8)  
asia\_training <- asia[training\_idx, ]  
asia\_test <- asia[-training\_idx, ]  
  
param\_tables\_naive <- bn.fit(naive\_bn, asia\_training, method = "mle")  
  
  
asia\_evidence <- asia\_test[, -target\_node\_index]  
  
bn\_naive\_grain <- param\_tables\_naive %>% as.grain() %>% compile()  
  
bn\_grain\_evidences <- map\_dbl(1:nrow(asia\_evidence),  
 ~ setEvidence(bn\_naive\_grain,  
 nodes = names(asia\_evidence),  
 states = as.matrix(asia\_evidence)[.x, ]) %>%  
 querygrain(nodes = "S") %>% pluck(1, "yes"))   
  
y\_hats <- ifelse(bn\_grain\_evidences > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)

##   
## y\_hats no yes  
## no 358 174  
## yes 132 336

The accuracy is 0.694 .

## Question 5

In the Questions 2 all the results are the same as the graphs are Markov equivalent. In Question 3 the result is the same as the results in Question 2 since the Markov Blanket of a variable makes the variable independent of the rest of the variables. In this case nodes “L” and “B” cause node “S” to be independent of the rest of the nodes. In other word:

$$ P(S|X \backslash S) = P(S|L, B) \\ \\ Where\ X\ is\ the\ set\ of\ all\ the\ Random\ Variables $$

Based on this knowledge, we should expect to get the same result.

However, in Question 4 we can see that the dependencies are much simpler (naive), therefore, the accuracy of the inference is lower than all the previous inferences.

## Appendix LAB1

library(bnlearn)  
library(gRain)  
library(purrr)  
  
data("asia")  
  
# 1  
  
hc\_network <- hc(asia)  
plot(hc\_network)  
  
  
hc\_network\_1 <- hc(asia, score = "bde", restart = 100)  
hc\_network\_2 <- hc(asia, score = "bde", restart = 1)  
all.equal(hc\_network\_1, hc\_network\_2)  
graphviz.compare(hc\_network\_1, hc\_network\_2, main = c("The BN with restart = 100",   
 "The BN with restart = 1"))  
  
  
  
hc\_network\_3 <- hc(asia, score = "bde", restart = 13)  
hc\_network\_4 <- hc(asia, score = "aic", restart = 13)  
all.equal(hc\_network\_3, hc\_network\_4)  
graphviz.compare(hc\_network\_3, hc\_network\_4, main = c("The BN with score = BDeu",   
 "The BN with restart = AIC"))  
  
  
hc\_network\_5 <- hc(asia, score = "bde", restart = 13, perturb = 3)  
hc\_network\_6 <- hc(asia, score = "bde", restart = 13, perturb = 7)  
all.equal(hc\_network\_5, hc\_network\_6)  
score(hc\_network\_5, asia, type = "bde")  
score(hc\_network\_6, asia, type = "bde")  
graphviz.compare(hc\_network\_5, hc\_network\_6, main = c("The BN with perturb = 3",   
 "The BN with perturb = 7"))  
  
  
# different imaginary sample size for BDeu algorithm  
hc\_network\_7 <- hc(asia, score = "bde", restart = 13, iss = 2)  
hc\_network\_8 <- hc(asia, score = "bde", restart = 13, iss = 7)  
all.equal(hc\_network\_7, hc\_network\_8)  
score(hc\_network\_7, asia, type = "bde")  
score(hc\_network\_8, asia, type = "bde")  
  
  
  
graphviz.compare(hc\_network\_7, hc\_network\_8, main = c("The BN with iss = 2",   
 "The BN with iss = 7"))  
  
# 2  
  
set.seed(7)  
  
training\_idx <- sample(5000, nrow(asia) \* 0.8)  
asia\_training <- asia[training\_idx, ]  
asia\_test <- asia[-training\_idx, ]  
  
  
bn\_dag <- hc(asia\_training, score = "bde", iss = 2, restart = 13)  
plot(bn\_dag)  
  
param\_tables <- bn.fit(bn\_dag, asia\_training, method = "mle")  
  
  
## Aprrox inference  
target\_node\_index <- 2  
evidence\_vector <-  
 map\_chr(  
 list\_transpose(as.list(asia\_test)),  
 ~ paste(  
 "(",  
 names(asia\_test)[-target\_node\_index],  
 " == '",  
 map\_chr(.x[-target\_node\_index], as.character),  
 "')",  
 sep = "",  
 collapse = " & "  
 )  
 )  
evnet <- "(S == 'yes')"  
  
cmds <- paste("cpquery(param\_tables, ", evnet, ", ", evidence\_vector, ")", sep = "")  
probs <- map\_dbl( cmds, ~eval(parse(text = .x)) )  
  
y\_hats <- ifelse(probs > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)  
sum( diag(table(y\_hats, asia\_test$S)) ) / nrow(asia\_test)  
  
  
  
## The Exact Inference  
  
asia\_test\_evidence <- asia\_test[, -target\_node\_index]  
  
bn\_grain <- param\_tables %>% as.grain() %>% compile()  
  
bn\_grain\_evidences <- map\_dbl(1:nrow(asia\_test\_evidence),  
 ~ setEvidence(bn\_grain,  
 nodes = names(asia\_test\_evidence),  
 states = as.matrix(asia\_test\_evidence)[.x, ]) %>%  
 querygrain(nodes = "S") %>% pluck(1, "yes"))   
  
y\_hats <- ifelse(bn\_grain\_evidences > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)  
sum( diag(table(y\_hats, asia\_test$S)) ) / nrow(asia\_test)  
  
  
  
## The exact inference with true graph  
  
dag <- model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")  
plot(dag)  
param\_tables\_true\_dag <- bn.fit(dag, asia\_training, method = "mle")  
  
  
asia\_test\_evidence <- asia\_test[, -target\_node\_index]  
  
bn\_true\_grain <- param\_tables\_true\_dag %>% as.grain() %>% compile()  
  
bn\_grain\_evidences <- map\_dbl(1:nrow(asia\_test\_evidence),  
 ~ setEvidence(bn\_true\_grain,  
 nodes = names(asia\_test\_evidence),  
 states = as.matrix(asia\_test\_evidence)[.x, ]) %>%  
 querygrain(nodes = "S") %>% pluck(1, "yes"))   
  
y\_hats <- ifelse(bn\_grain\_evidences > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)  
sum( diag(table(y\_hats, asia\_test$S)) ) / nrow(asia\_test)  
  
  
# 3  
  
set.seed(7)  
  
training\_idx <- sample(5000, nrow(asia) \* 0.8)  
asia\_training <- asia[training\_idx, ]  
asia\_test <- asia[-training\_idx, ]  
  
bn\_dag <- hc(asia\_training, score = "bde", iss = 2, restart = 13)  
plot(bn\_dag)  
  
  
mb\_s <- mb(bn\_dag, "S")  
  
  
param\_tables <- bn.fit(bn\_dag, asia\_training, method = "mle")  
  
  
asia\_test\_evidence <- asia\_test[, mb\_s]  
  
bn\_grain <- param\_tables %>% as.grain() %>% compile()  
  
bn\_grain\_evidences <- map\_dbl(1:nrow(asia\_test\_evidence),  
 ~ setEvidence(bn\_grain,  
 nodes = names(asia\_test\_evidence),  
 states = as.matrix(asia\_test\_evidence)[.x, ]) %>%  
 querygrain(nodes = "S") %>% pluck(1, "yes"))   
  
y\_hats <- ifelse(bn\_grain\_evidences > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)  
  
  
# 4  
  
target\_node\_index <- 2  
  
naive\_bn <- empty.graph(names(asia))  
  
naive\_arcs <- cbind("S", c(names(asia)[-target\_node\_index]))  
  
arcs(naive\_bn) <- naive\_arcs  
plot(naive\_bn)  
  
  
set.seed(7)  
  
training\_idx <- sample(5000, nrow(asia) \* 0.8)  
asia\_training <- asia[training\_idx, ]  
asia\_test <- asia[-training\_idx, ]  
  
param\_tables\_naive <- bn.fit(naive\_bn, asia\_training, method = "mle")  
  
  
asia\_evidence <- asia\_test[, -target\_node\_index]  
  
bn\_naive\_grain <- param\_tables\_naive %>% as.grain() %>% compile()  
  
bn\_grain\_evidences <- map\_dbl(1:nrow(asia\_evidence),  
 ~ setEvidence(bn\_naive\_grain,  
 nodes = names(asia\_evidence),  
 states = as.matrix(asia\_evidence)[.x, ]) %>%  
 querygrain(nodes = "S") %>% pluck(1, "yes"))   
  
y\_hats <- ifelse(bn\_grain\_evidences > 0.5, "yes", "no")  
  
table(y\_hats, asia\_test$S)