Lab1-Personal report

Arash

2023-09-05

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")
```

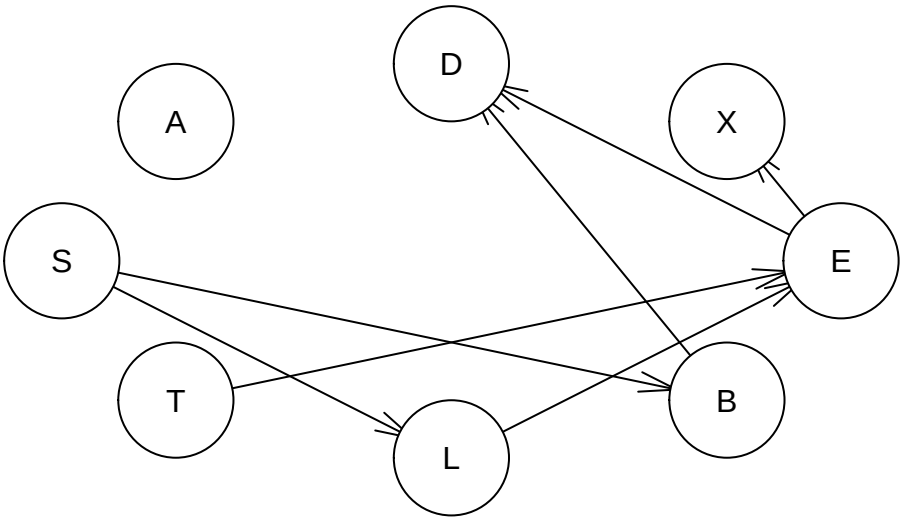
We can check the documentation of data using `?asia` to get a better knowledge of the data at hand.

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)
```



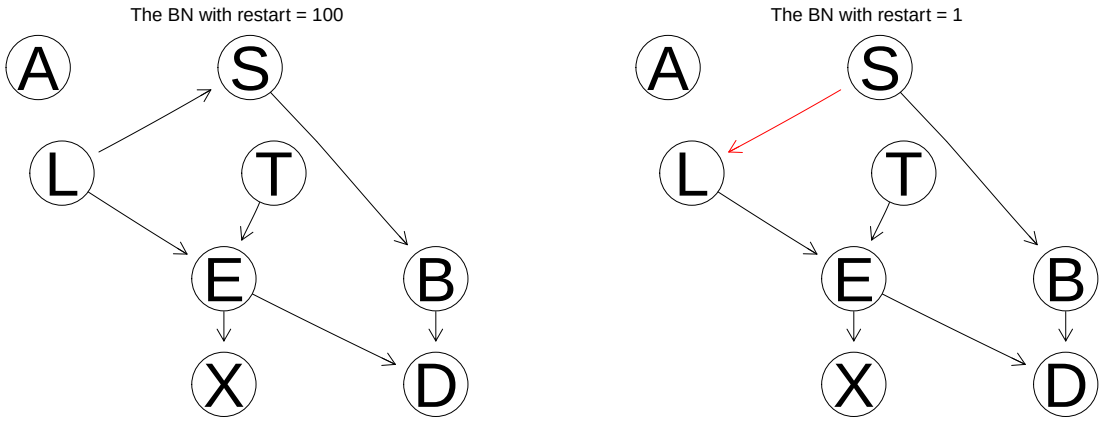
Now, the arguments will be changed to see if we see any difference between different runs of Hill Climbing algorithm.

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"))
all.equal(hc_network_1, hc_network_2)
```

```
## [1] "Different arc sets"
```



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:

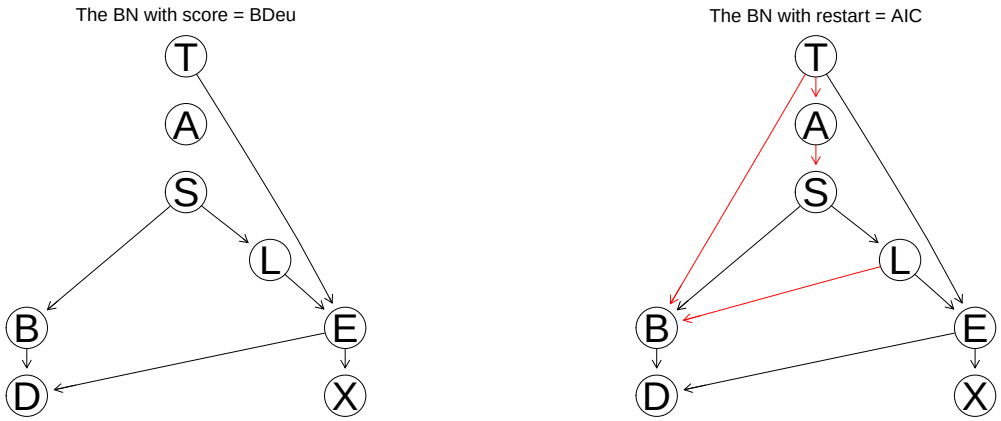
```
hc_network_3 <- hc(asia, score = "bde", restart = 13)
hc_network_4 <- hc(asia, score = "aic", restart = 13)



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] TRUE
```

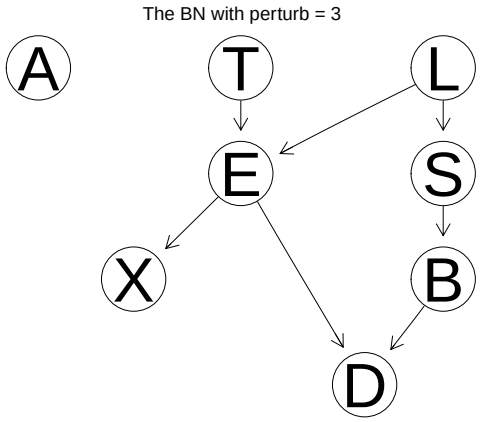
```
score(hc_network_5, asia, type = "bde")
```

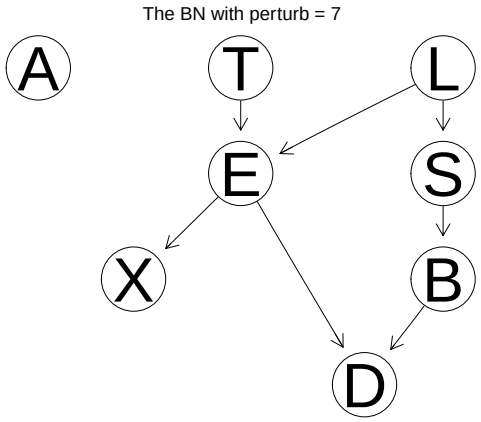
```
## [1] -11095.79
```

```
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"
```

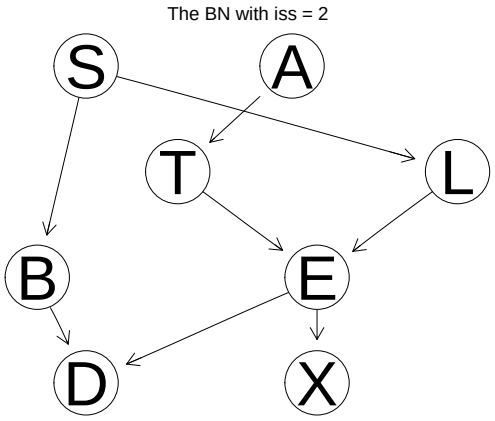
```
score(hc_network_7, asia, type = "bde")
```

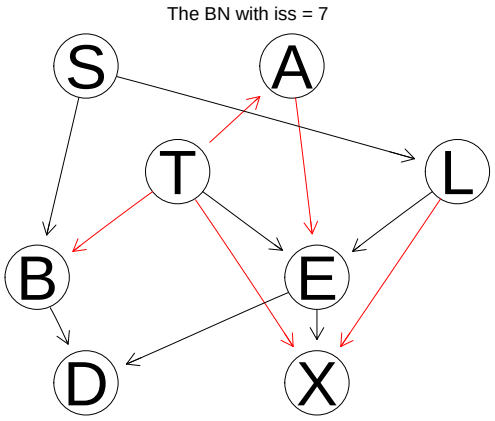
```
## [1] -11096.64
```

```
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

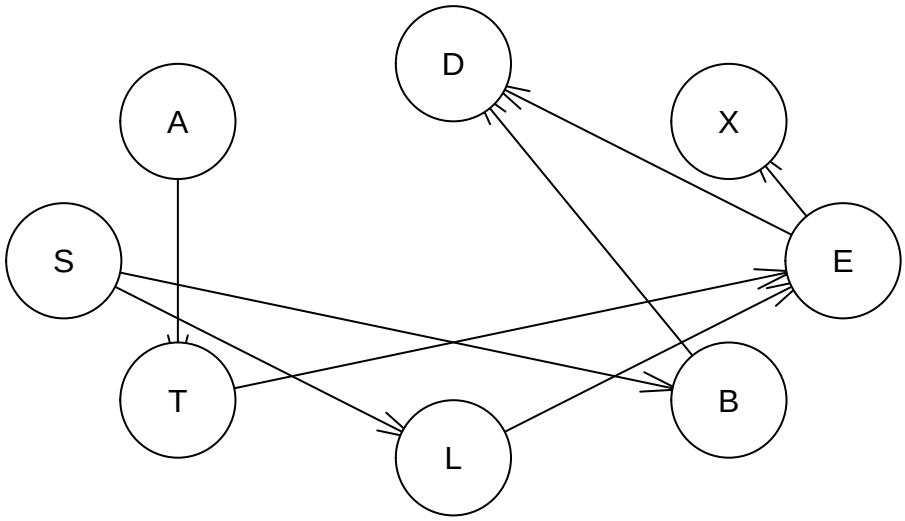
First, the data is split in two sets of Training and Testing where 80% of the total data is in the train data

```r
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

```r
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.

```r
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:

```r
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()
```

```
## Warning: Any reference to graphNEL objects will be removed in future versions of gRbase
## Please use igraphs instead
```

```
## Warning: Any reference to graphNEL objects will be removed in future versions of gRbase
## Please use igraphs instead
```

```
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()
```

```
## Warning: Any reference to graphNEL objects will be removed in future versions of gRbase
## Please use igraphs instead
```

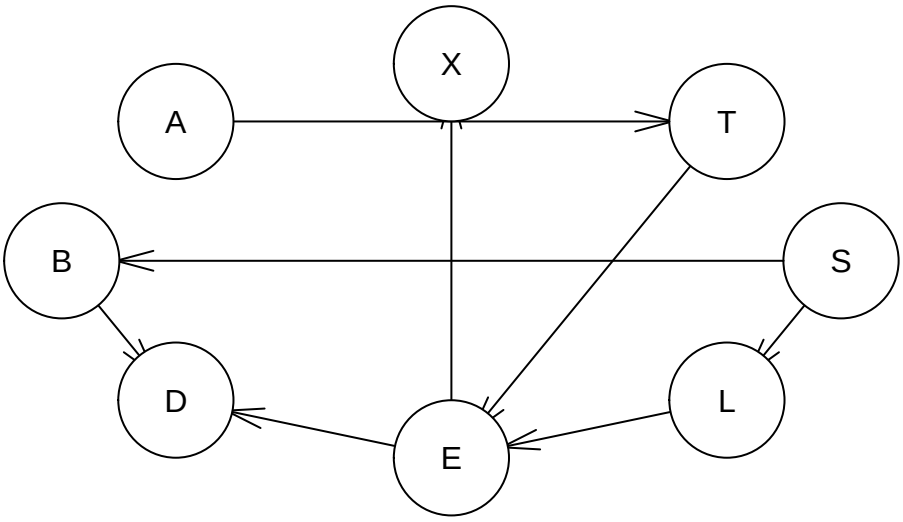
```
## Warning: Any reference to graphNEL objects will be removed in future versions of gRbase
## Please use igraphs instead
```

```
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()
```

```
## Warning: Any reference to graphNEL objects will be removed in future versions of gRbase
## Please use igraphs instead
```

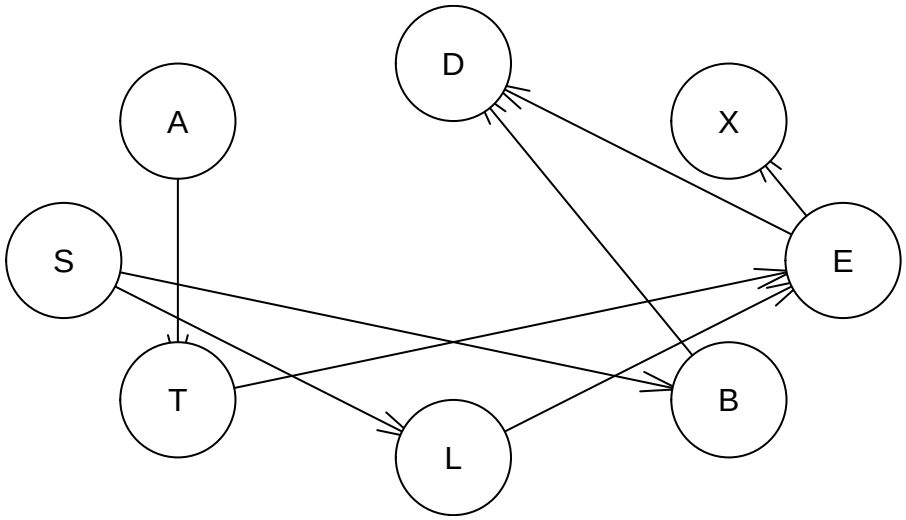
```
## Warning: Any reference to graphNEL objects will be removed in future versions of gRbase
## Please use igraphs instead
```

```
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 .

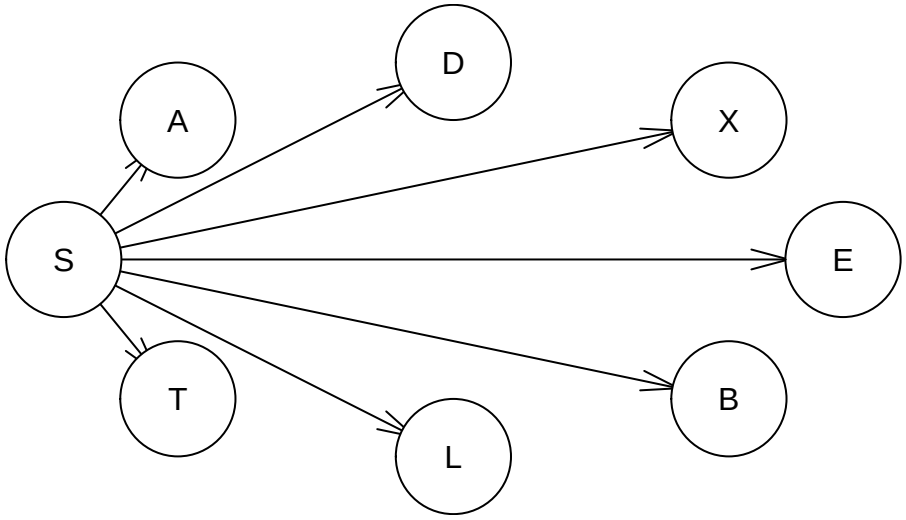
Question 4

```
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()
```

```
## Warning: Any reference to graphNEL objects will be removed in future versions of gRbase
## Please use igraphs instead
```

```
## Warning: Any reference to graphNEL objects will be removed in future versions of gRbase
## Please use igraphs instead
```

```
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"))
```

```r
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.