Lab 1

Simge Cinar, Simon Jorstedt, and Marijn Jaarsma

2024-09-10

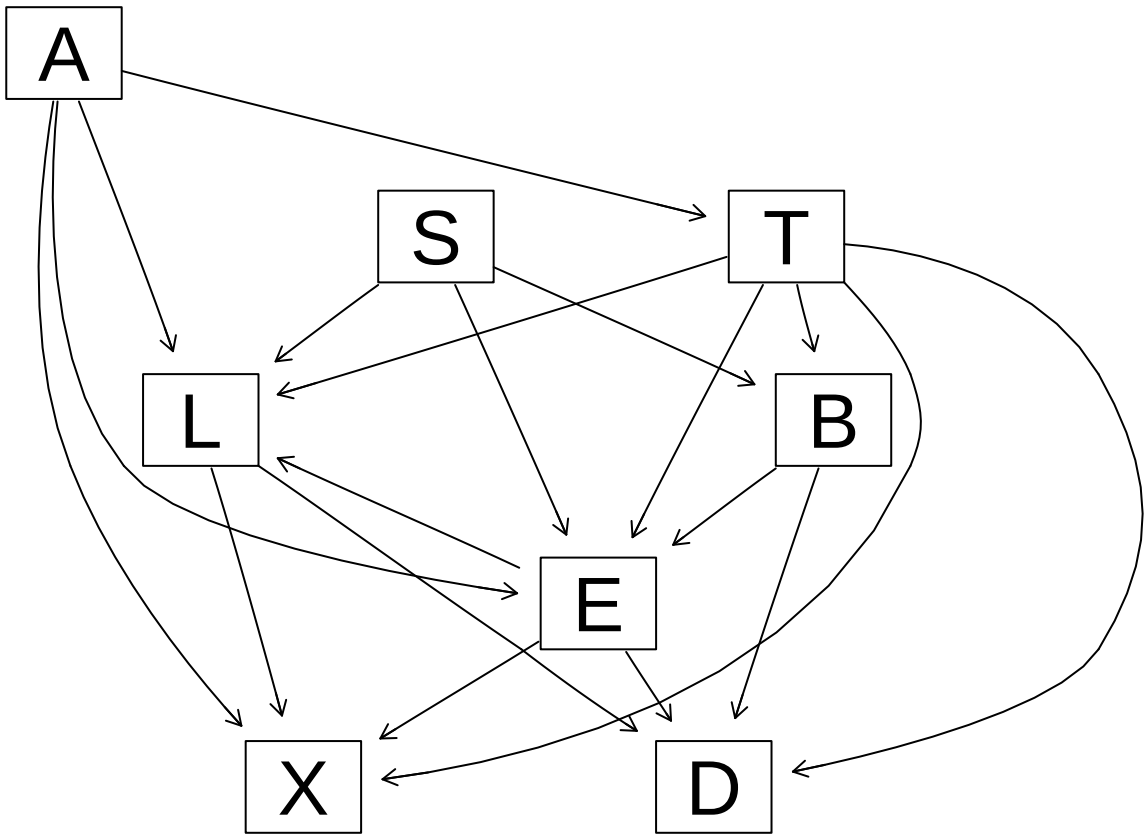
Statement of contribution

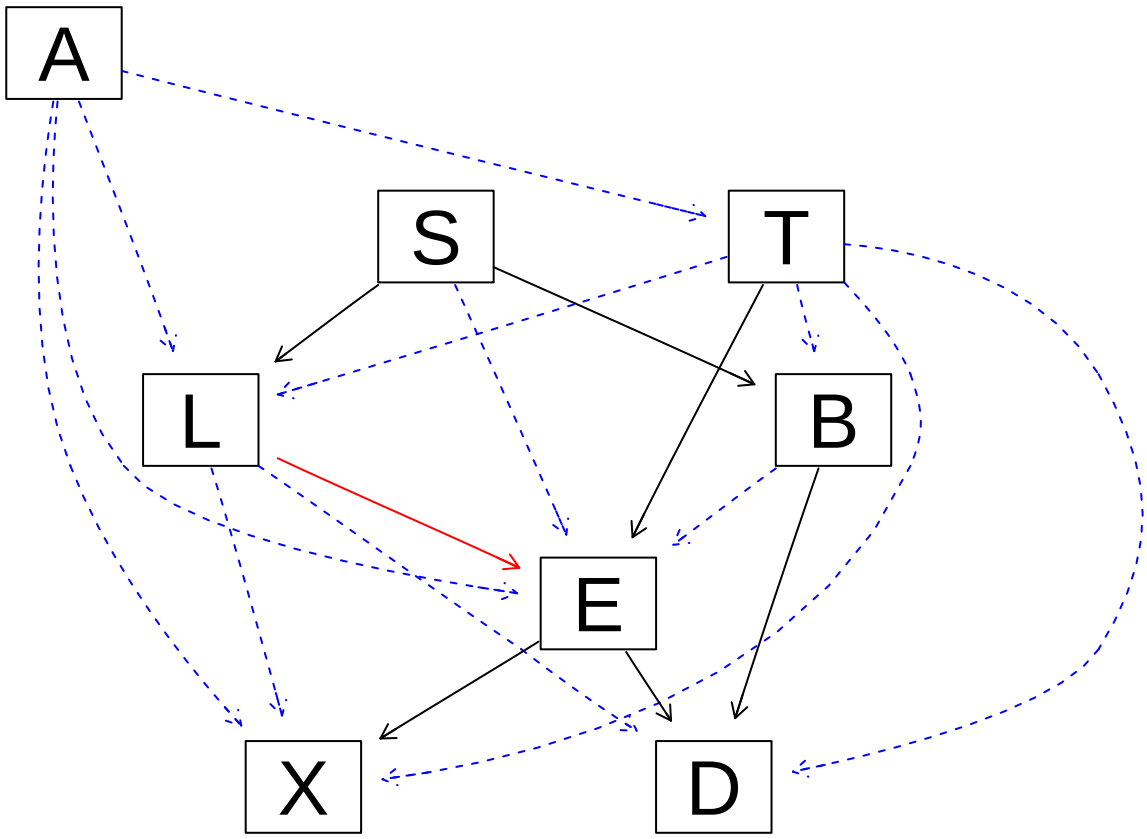
- Question 1 is a combination of our code with some additional plots and analyses from Simge.
- We had very similar solutions with similar results for questions 2-4.
- Question 5 was discussed all together as we combined our results.

Question 1

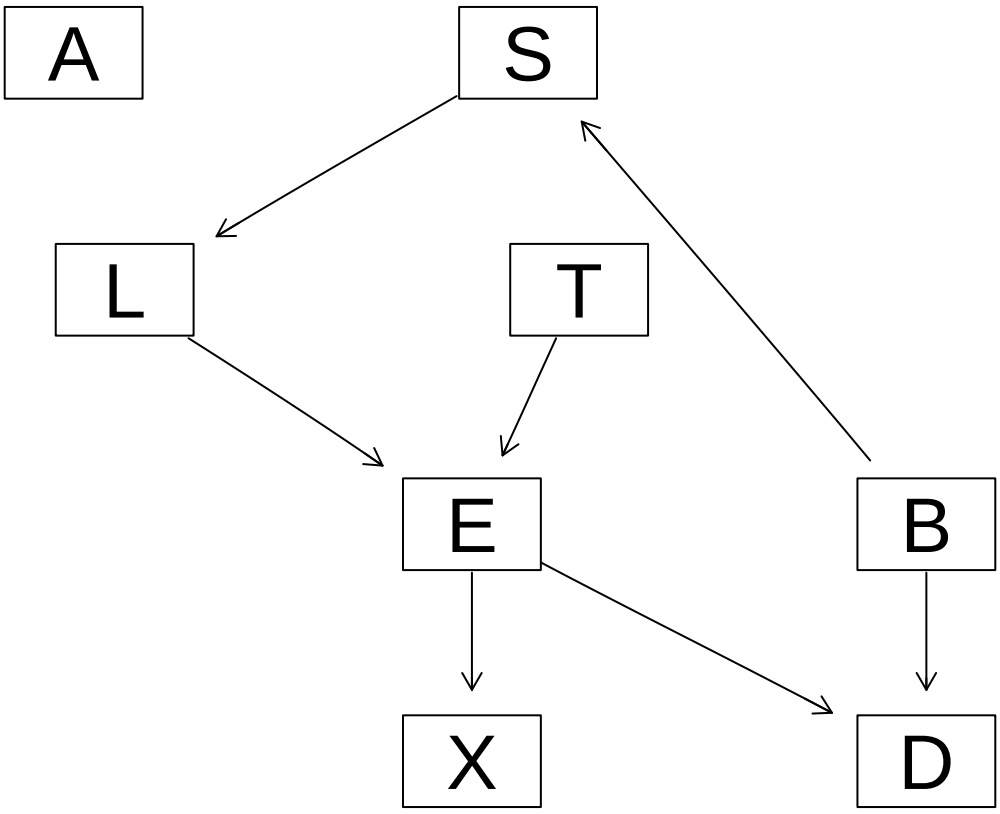
```
set.seed(12345)

# Load data
data("asia")

# Create and compare graphs with HC alg
graph1 <- hc(asia, score="bde", iss=100, restart=10)
graph2 <- hc(asia, score="bde", iss=1, restart=1)

graph3 <- hc(asia, restart=100)
graph4 <- hc(asia, restart=1)

graphviz.compare(graph1, graph2)
```

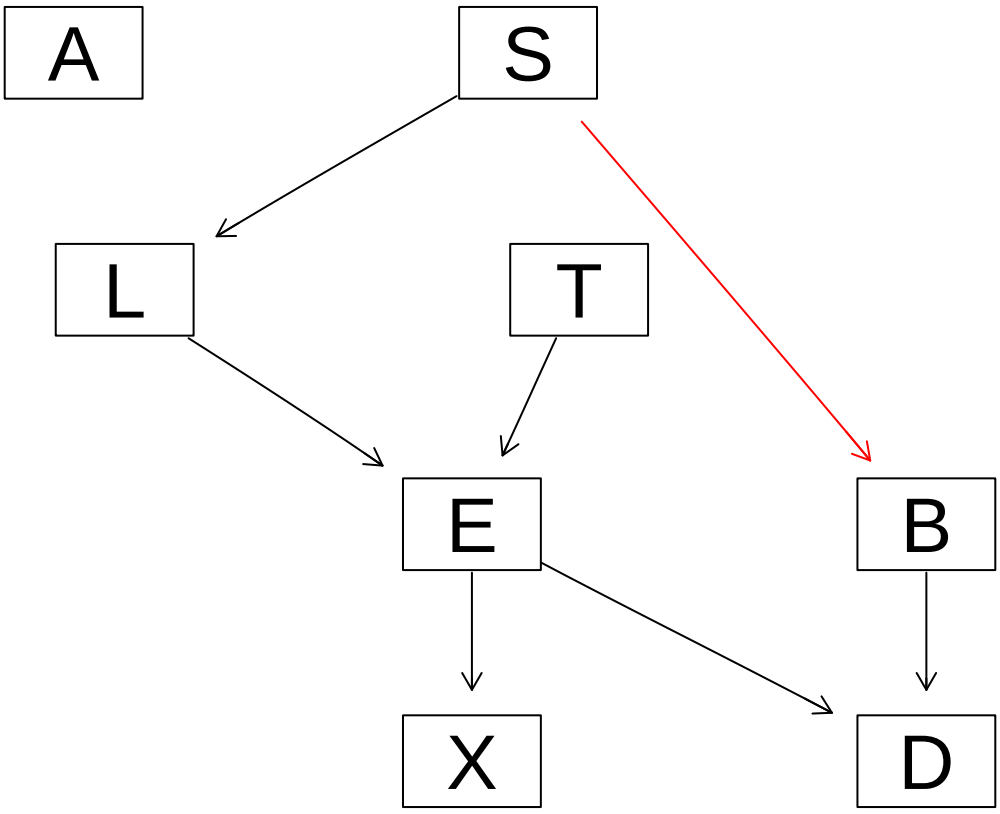
```
## Loading required namespace: Rgraphviz
```


A
S
T
L
B
E
X
D




```
graphviz.compare(graph3, graph4)
```


A
S
L
T
E
B
X
D




```
# Compute and print DBeu scores
score1 <- score(graph1, data=asia, type="bde")
score2 <- score(graph2, data=asia, type="bde")
score3 <- score(graph3, data=asia, type="bde")
score4 <- score(graph4, data=asia, type="bde")

cat("BDeu score for bn1:", score1, "\n")
```

```
## BDeu score for bn1: -11467.05
```

```
cat("BDeu score for bn2:", score2, "\n")
```

```
## BDeu score for bn2: -11095.79
```

```
cat("BDeu score for bn3:", score3, "\n")
```

```
## BDeu score for bn3: -11095.79
```

```
cat("BDeu score for bn4:", score4, "\n")
```

```
## BDeu score for bn4: -11095.79
```

```
# Print the best model composition
cat("Best model:\n")
```

```
## Best model:
```

```
modelstring(graph2)
```

```
## [1] "[A][S][T][L|S][B|S][E|T:L][X|E][D|B:E]"
```

```
# Check equality for some networks
cat("Equality check bn1 and bn4:\n")
```

```
## Equality check bn1 and bn4:
```

```
all.equal(graph1, graph4)
```

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

```
cat("Equality check bn3 and bn4:\n")
```

```
## Equality check bn3 and bn4:
```

```
all.equal(graph3, graph4)
```

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

The HC algorithm may find different network structures because it is not guaranteed to find a global optimum. In the comparison above, the imaginary sample size (ISS) was changed which means that one network regularizes less with the Bayesian score. This network is much bigger, with many more edges than the network with small ISS. We can also see that this larger network has a worse BDeu score than the smaller network. The last two networks only have a difference in the direction of the edge between S and B. These networks are Markov equivalent and have the same BDeu score, as the direction does not matter for the probability distribution. Increasing the number of random restarts may also result in different graphs as introducing more randomness may lead to separate runs of the algorithm to find a different local optimum.

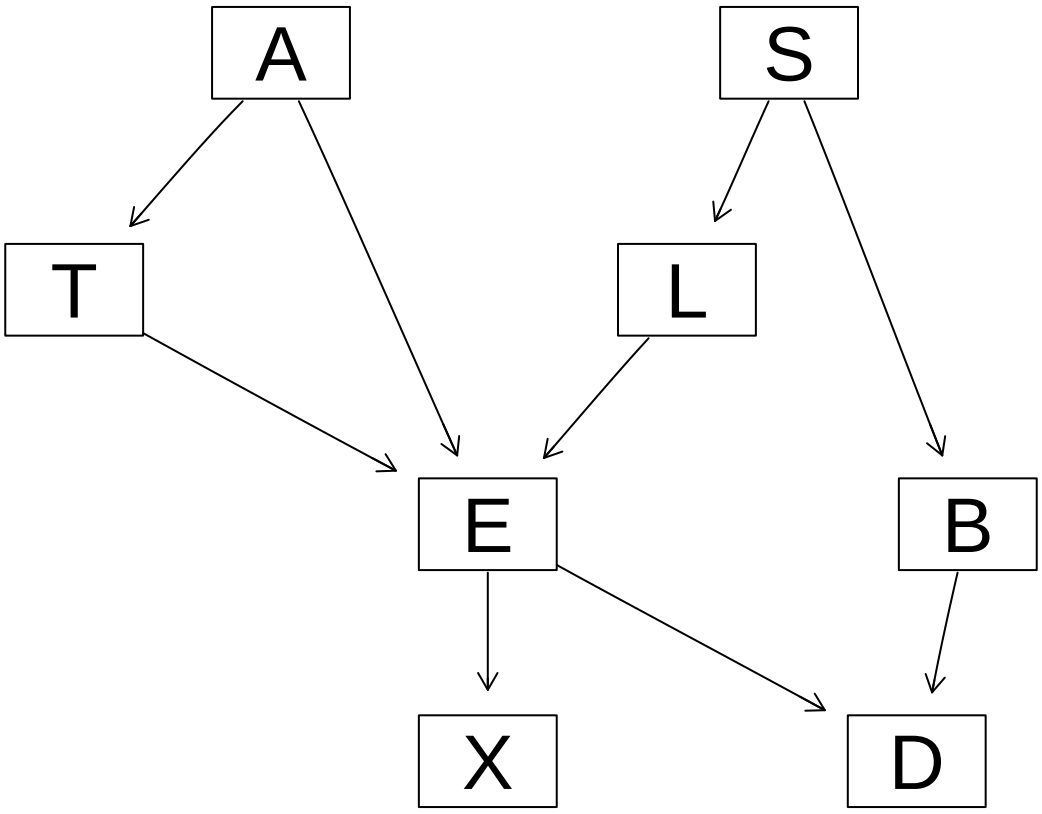
Question 2

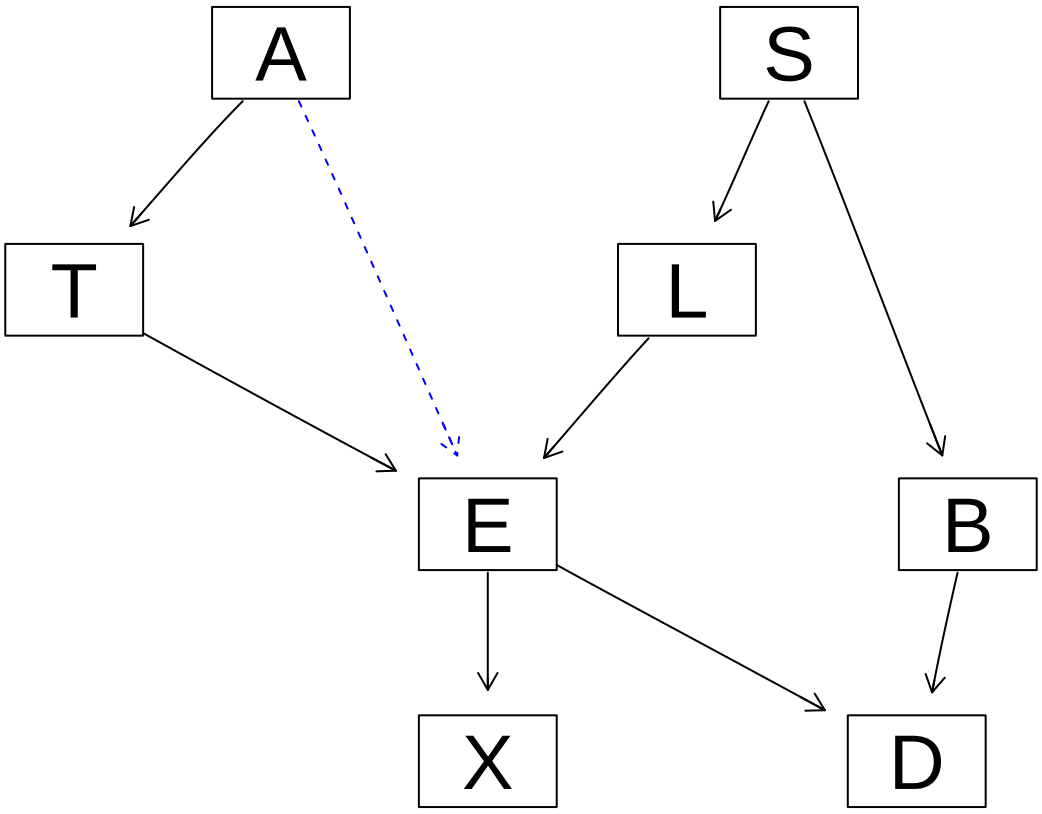
```
# Sample 80% of data for training
sample_ind <- sample(1:nrow(asia), 0.8 * nrow(asia))
df_train <- asia[sample_ind,]
df_test <- asia[-sample_ind,]

# Learn structure and parameters
# https://www.bnlearn.com/examples/fit/
graph <- hc(df_train, score="bde", iss=3, restart=20)
param <- bn.fit(graph, df_train, method="bayes")

# Visualize and compare to true model
print(param)
```

```
##
##   Bayesian network parameters
##
##   Parameters of node A (multinomial distribution)
##
## Conditional probability table:
##           no         yes
## 0.990632026 0.009367974
##
##   Parameters of node S (multinomial distribution)
##
## Conditional probability table:
##         no       yes
## 0.4950037 0.5049963
##
##   Parameters of node T (multinomial distribution)
##
## Conditional probability table:
##
##      A
## T              no         yes
##   no  0.991741268 0.926666667
##   yes 0.008258732 0.073333333
##
##   Parameters of node L (multinomial distribution)
##
## Conditional probability table:
##
##      S
## L             no        yes
##   no  0.98700479 0.88090527
##   yes 0.01299521 0.11909473
##
##   Parameters of node B (multinomial distribution)
##
## Conditional probability table:
##
##      S
## B            no       yes
##   no  0.7114560 0.2798664
##   yes 0.2885440 0.7201336
##
##   Parameters of node E (multinomial distribution)
##
## Conditional probability table:
##
## , , T = no, L = no
##
##      A
## E               no          yes
##   no  9.999489e-01 9.942085e-01
##   yes 5.105688e-05 5.791506e-03
##
## , , T = yes, L = no
```

```
##
##      A
## E               no          yes
##   no  6.382979e-03 7.894737e-02
##   yes 9.936170e-01 9.210526e-01
##
## , , T = no, L = yes
##
##      A
## E               no          yes
##   no  7.201152e-04 7.894737e-02
##   yes 9.992799e-01 9.210526e-01
##
## , , T = yes, L = yes
##
##      A
## E               no          yes
##   no  5.555556e-02 5.000000e-01
##   yes 9.444444e-01 5.000000e-01
##
##
##   Parameters of node X (multinomial distribution)
##
## Conditional probability table:
##
##      E
## X              no         yes
##   no  0.955539064 0.009243697
##   yes 0.044460936 0.990756303
##
##   Parameters of node D (multinomial distribution)
##
## Conditional probability table:
##
## , , E = no
##
##      B
## D            no       yes
##   no  0.8960769 0.2126982
##   yes 0.1039231 0.7873018
##
## , , E = yes
##
##      B
## D            no       yes
##   no  0.2734205 0.1333789
##   yes 0.7265795 0.8666211
```

```
dag <- model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
graphviz.compare(graph, bn.fit(dag, asia))
```


A
S
T
L
E
B
X
D



Our graph is very close to the true graph, with only an extra edge between A and E.

NOTE THAT: Conditional probability tables in grain objects must be completely specified; on the other hand, bn.fit allows NaN values for unobserved parents' configurations. Such bn.fit objects will be converted to m grain objects by replacing the missing conditional probability distributions with uniform distributions.

Another solution to this problem is to fit another bn.fit with method="bayes" and a low iss value, using the same data and network structure. ?as.grain

```
# Convert to grain
grain <- compile(as.grain(param))

pred <- rep(0, nrow(df_test))
nodes <- names(df_test)[!names(df_test) %in% "S"]
for (i in 1:nrow(df_test)) {
  # Record evidence
  states <- as.vector(t(df_test[i, nodes]))

  # # https://www.rdocumentation.org/packages/gRain/versions/1.3-2/topics/grain-evidence
  evidence <- setEvidence(grain, nodes, states)

  # https://www.rdocumentation.org/packages/gRain/versions/1.4.1/topics/querygrain
  pred[i] <- names(which.max(querygrain(evidence, "S", evidence=evidence)$S))

}

# Compute confusion matrix
```

```r
confusionMatrix(factor(pred), factor(df_test$S))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  327 127
##        yes 178 368
##
##                Accuracy : 0.695
##                  95% CI : (0.6654, 0.7234)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3906
##
##  Mcnemar's Test P-Value : 0.004197
##
##             Sensitivity : 0.6475
##             Specificity : 0.7434
##          Pos Pred Value : 0.7203
##          Neg Pred Value : 0.6740
##              Prevalence : 0.5050
##          Detection Rate : 0.3270
##    Detection Prevalence : 0.4540
##       Balanced Accuracy : 0.6955
##
##        'Positive' Class : no
##
```

```r
param_best_dag <- bn.fit(dag, df_train, method="bayes")

# Convert to grain
grain_best_dag <- compile(as.grain(param_best_dag))

pred_best_dag <- rep(0, nrow(df_test))
nodes <- names(df_test)[!names(df_test) %in% "S"]
for (i in 1:nrow(df_test)) {
  # Record evidence
  states <- as.vector(t(df_test[i, nodes]))

  # # https://www.rdocumentation.org/packages/gRain/versions/1.3-2/topics/grain-evidence
  evidence <- setEvidence(grain_best_dag, nodes, states)

  # https://www.rdocumentation.org/packages/gRain/versions/1.4.1/topics/querygrain
  pred_best_dag[i] <- names(which.max(querygrain(evidence, "S", evidence=evidence)$S))

}

# Compute confusion matrix
confusionMatrix(factor(pred_best_dag), factor(df_test$S))
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  no yes
##        no  327 127
##        yes 178 368
##
##                Accuracy : 0.695
##                  95% CI : (0.6654, 0.7234)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3906
##
##  Mcnemar's Test P-Value : 0.004197
##
##             Sensitivity : 0.6475
##             Specificity : 0.7434
##          Pos Pred Value : 0.7203
##          Neg Pred Value : 0.6740
##              Prevalence : 0.5050
##          Detection Rate : 0.3270
##    Detection Prevalence : 0.4540
##       Balanced Accuracy : 0.6955
##
##        'Positive' Class : no
##
```

Our graphs and the true Asia BN you provided yielded the same confusion matrix. Note that those two graphs are equivalent.

Question 3

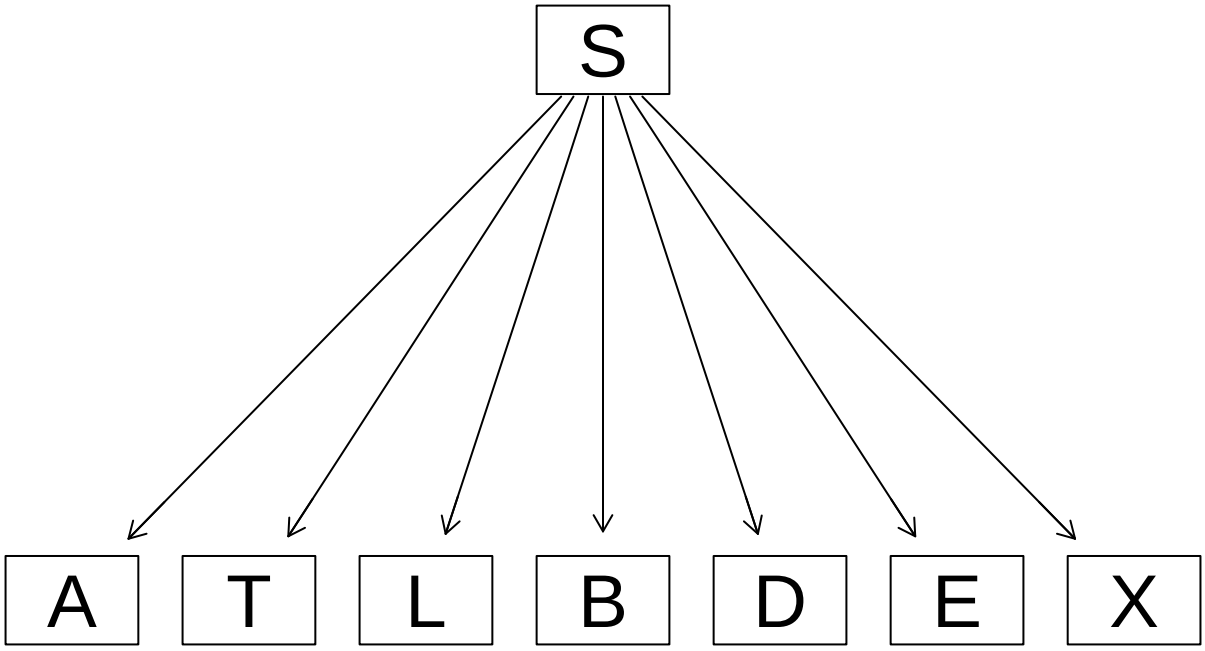
```
pred <- rep(0, nrow(df_test))
for (i in 1:nrow(df_test)) {
  # Record evidence
  nodes <- mb(param, "S")
  states <- as.vector(t(df_test[i, nodes]))

  # # https://www.rdocumentation.org/packages/gRain/versions/1.3-2/topics/grain-evidence
  evidence <- setEvidence(grain, nodes, states)

  # https://www.rdocumentation.org/packages/gRain/versions/1.4.1/topics/querygrain
  # pred[i] <- querygrain(grain, "S", evidence=evidence)
  pred[i] <- names(which.max(querygrain(evidence, "S")$S))
}

confusionMatrix(factor(pred), factor(df_test$S))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  no yes
##        no  327 127
##        yes 178 368
##
##                Accuracy : 0.695
##                  95% CI : (0.6654, 0.7234)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3906
##
##  Mcnemar's Test P-Value : 0.004197
##
##             Sensitivity : 0.6475
##             Specificity : 0.7434
##          Pos Pred Value : 0.7203
##          Neg Pred Value : 0.6740
##              Prevalence : 0.5050
##          Detection Rate : 0.3270
##    Detection Prevalence : 0.4540
##       Balanced Accuracy : 0.6955
##
##        'Positive' Class : no
##
```

Question 4

```r
# Create naive Bayesian graph
# https://www.bnlearn.com/examples/dag/
graph <- empty.graph(c("A", "S", "T", "L", "B", "D", "E", "X"))
arc_set <- matrix(c("S", "A", "S", "T", "S", "L", "S", "B", "S", "D", "S", "E", "S", "X"),
                  ncol=2, byrow=TRUE, dimnames=list(NULL, c("from", "to")))
arcs(graph) <- arc_set

graphviz.plot(graph)
```




```
# Train and convert to grain
param <- bn.fit(graph, df_train, method="bayes")
grain <- compile(as.grain(param))

pred <- rep(0, nrow(df_test))
nodes <- names(df_test)[!names(df_test) %in% "S"]
for (i in 1:nrow(df_test)) {
  # Record evidence
  states <- as.vector(t(df_test[i, nodes]))

  # # https://www.rdocumentation.org/packages/gRain/versions/1.3-2/topics/grain-evidence
  evidence <- setEvidence(grain, nodes, states)

  # https://www.rdocumentation.org/packages/gRain/versions/1.4.1/topics/querygrain
  pred[i] <- names(which.max(querygrain(evidence, "S", evidence=evidence)$S))

}

# Compute confusion matrix
confusionMatrix(factor(pred), factor(df_test$S))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  360 190
```

```
##        yes 145 305
##
##                Accuracy : 0.665
##                  95% CI : (0.6348, 0.6942)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.3293
##
##  Mcnemar's Test P-Value : 0.01622
##
##             Sensitivity : 0.7129
##             Specificity : 0.6162
##          Pos Pred Value : 0.6545
##          Neg Pred Value : 0.6778
##              Prevalence : 0.5050
##          Detection Rate : 0.3600
##    Detection Prevalence : 0.5500
##       Balanced Accuracy : 0.6645
##
##        'Positive' Class : no
##
```

Question 5

Markov blanket is the minimal set of nodes that separates a given node from the rest. In question 2 and question 3, there was no difference in accuracy between using the full model as evidence and using the Markov blanket, as the Markov blanket retains all necessary information to do inference on S. The naive classifier performs a little bit worse than the other two, likely because the model is modeling dependencies incorrectly. This may lead to other variables impacting S within this graph, while that is not true in reality.