Lab 1 Marijn

Marijn Jaarsma

2024-09-10

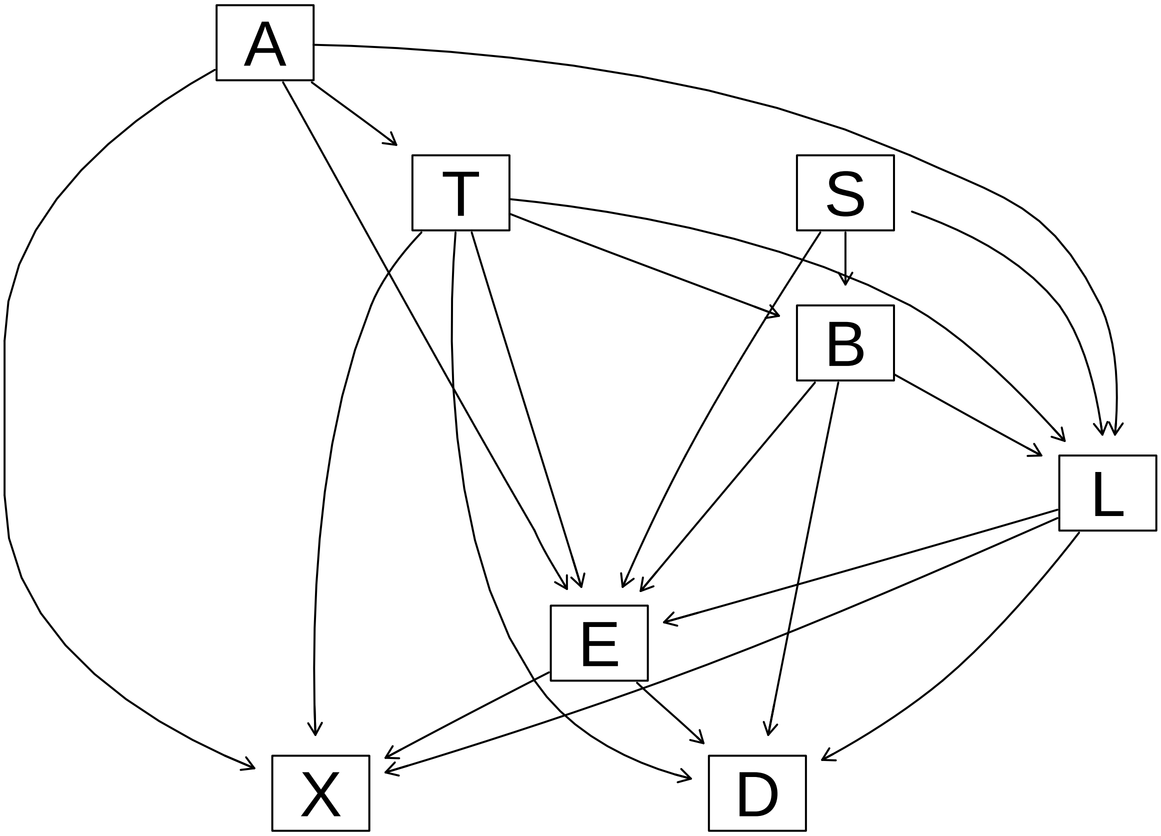
Question 1

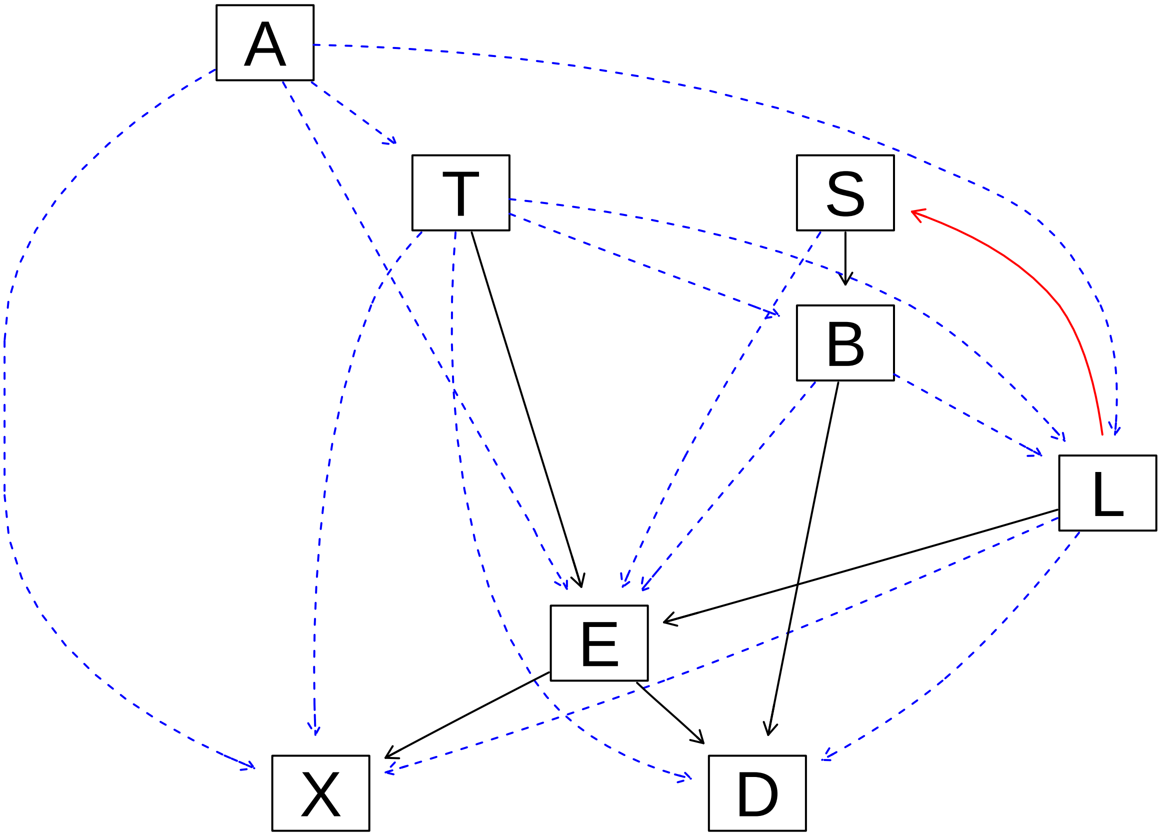
```r
# Load data
data("asia")

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

graphviz.compare(graph1, graph2)
```

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





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. 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

```r
# 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
# graphviz.plot(graph)
print(param)
```

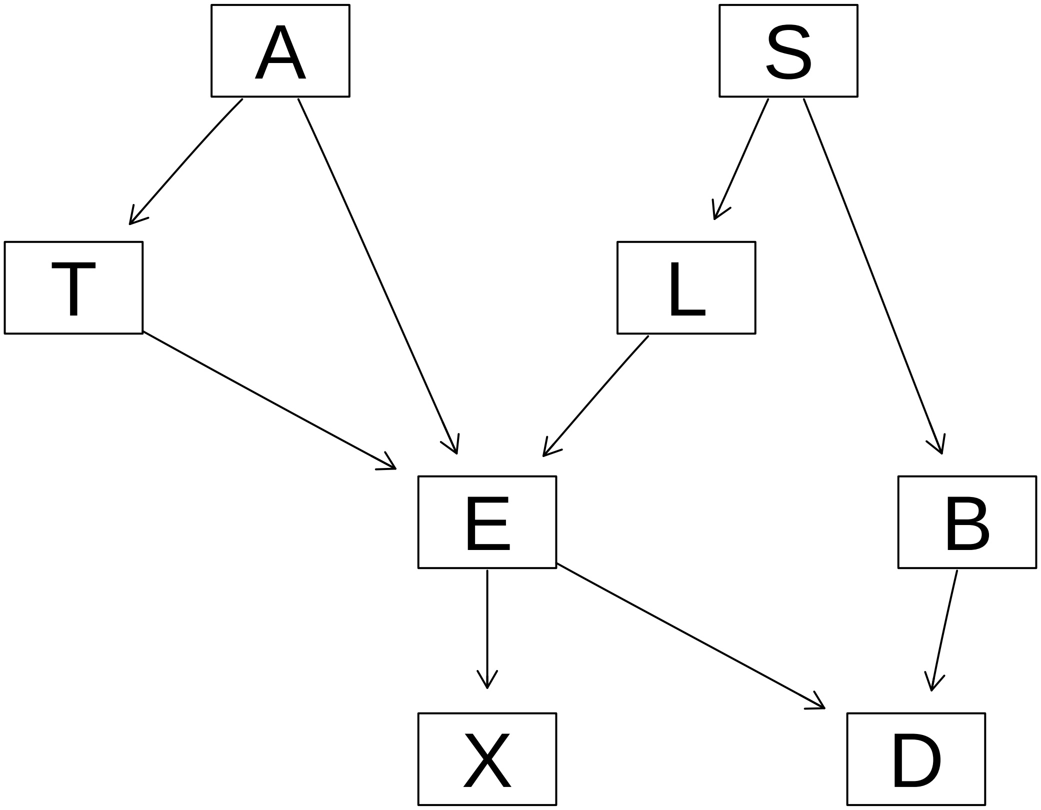
```
##
```

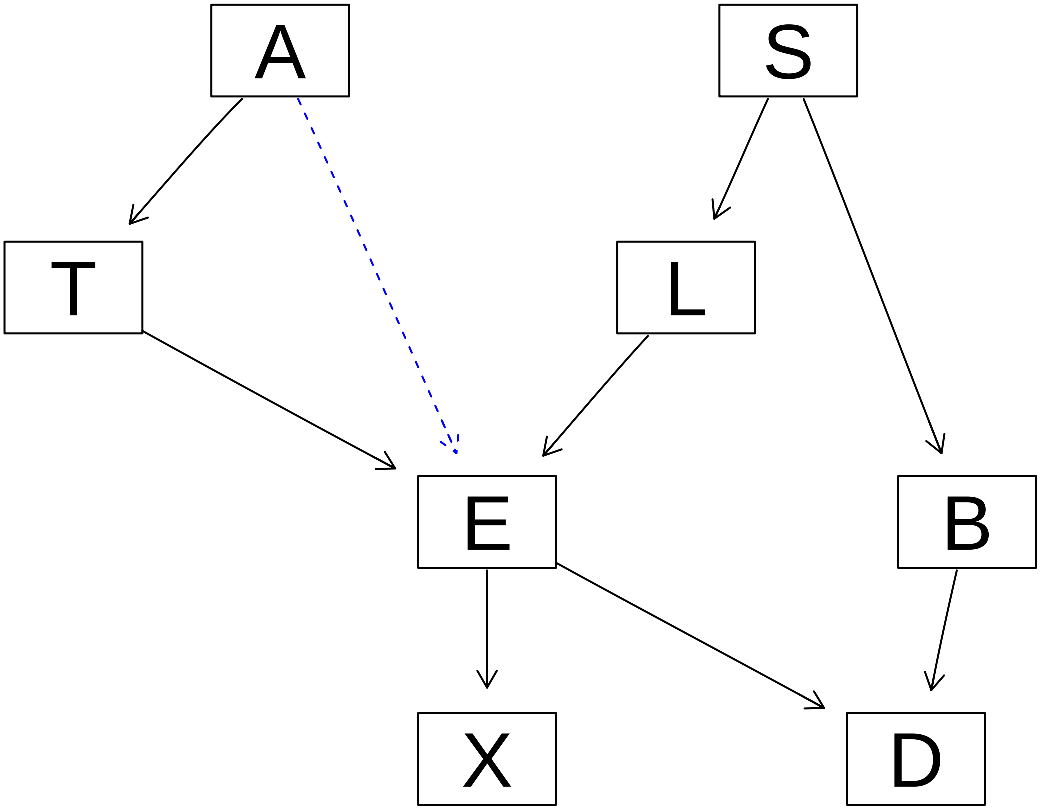
```
##   Bayesian network parameters
##
##   Parameters of node A (multinomial distribution)
##
## Conditional probability table:
##           no         yes
## 0.991381464 0.008618536
##
##   Parameters of node S (multinomial distribution)
##
## Conditional probability table:
##         no       yes
## 0.4970022 0.5029978
##
##   Parameters of node T (multinomial distribution)
##
## Conditional probability table:
##
##      A
## T              no         yes
##   no  0.990991559 0.920289855
##   yes 0.009008441 0.079710145
##
##   Parameters of node L (multinomial distribution)
##
## Conditional probability table:
##
##      S
## L             no        yes
##   no  0.98504649 0.87695555
##   yes 0.01495351 0.12304445
##
##   Parameters of node B (multinomial distribution)
##
## Conditional probability table:
##
##      S
## B            no       yes
##   no  0.6960292 0.2884281
##   yes 0.3039708 0.7115719
##
##   Parameters of node E (multinomial distribution)
##
## Conditional probability table:
##
## , , T = no, L = no
##
##      A
## E               no          yes
##   no  9.999488e-01 9.933921e-01
##   yes 5.118231e-05 6.607930e-03
##
## , , T = yes, L = no
##
```

```
##      A
## E               no          yes
##   no  5.976096e-03 7.894737e-02
##   yes 9.940239e-01 9.210526e-01
##
## , , T = no, L = yes
##
##      A
## E               no          yes
##   no  6.960557e-04 5.555556e-02
##   yes 9.993039e-01 9.444444e-01
##
## , , T = yes, L = yes
##
##      A
## E               no          yes
##   no  4.285714e-02 5.000000e-01
##   yes 9.571429e-01 5.000000e-01
##
##
##   Parameters of node X (multinomial distribution)
##
## Conditional probability table:
##
##      E
## X              no         yes
##   no  0.955111713 0.008856683
##   yes 0.044888287 0.991143317
##
##   Parameters of node D (multinomial distribution)
##
## Conditional probability table:
##
## , , E = no
##
##      B
## D            no       yes
##   no  0.8986477 0.2171525
##   yes 0.1013523 0.7828475
##
## , , E = yes
##
##      B
## D            no       yes
##   no  0.2773019 0.1412903
##   yes 0.7226981 0.8587097
```

```
# bn.fit.barchart(fitted_graph$S)

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



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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  352 115
##        yes 145 388
```

```
##
##                Accuracy : 0.74
##                  95% CI : (0.7116, 0.7669)
##     No Information Rate : 0.503
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4798
##
##  Mcnemar’s Test P-Value : 0.0721
##
##             Sensitivity : 0.7082
##             Specificity : 0.7714
##          Pos Pred Value : 0.7537
##          Neg Pred Value : 0.7280
##              Prevalence : 0.4970
##          Detection Rate : 0.3520
##    Detection Prevalence : 0.4670
##       Balanced Accuracy : 0.7398
##
##        ’Positive’ Class : no
##
```

Question 3

```r
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))
```

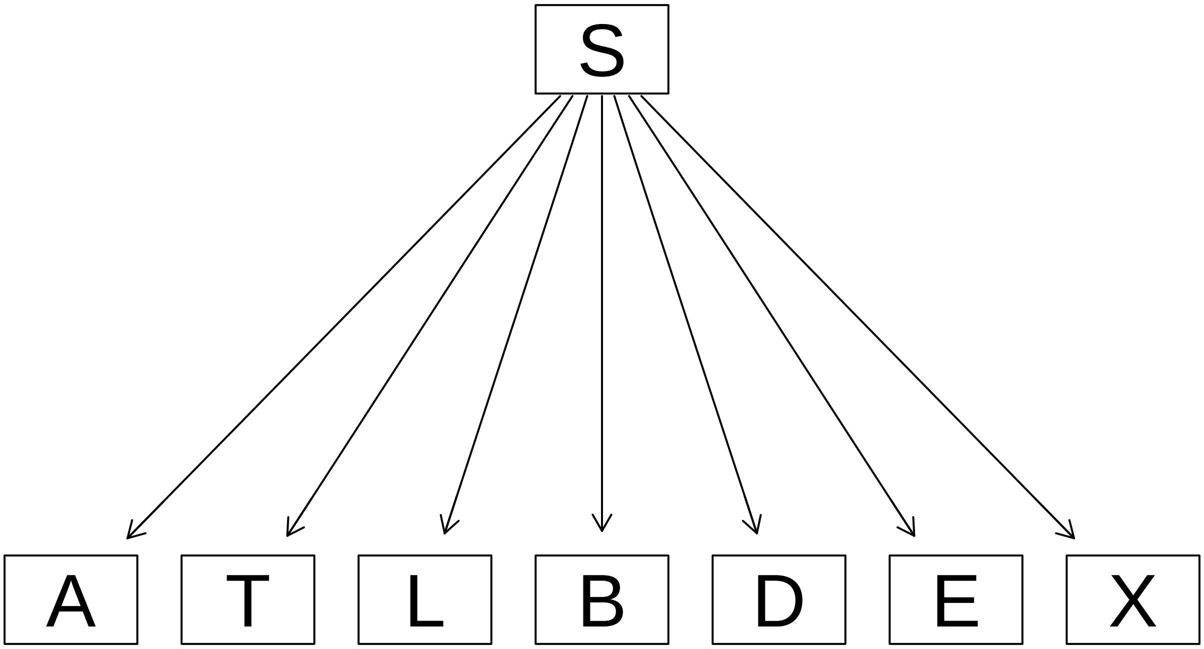
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  352 115
##        yes 145 388
##
##                Accuracy : 0.74
##                  95% CI : (0.7116, 0.7669)
##     No Information Rate : 0.503
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                   Kappa : 0.4798
##
##  Mcnemar's Test P-Value : 0.0721
##
##             Sensitivity : 0.7082
##             Specificity : 0.7714
##          Pos Pred Value : 0.7537
##          Neg Pred Value : 0.7280
##              Prevalence : 0.4970
##          Detection Rate : 0.3520
##    Detection Prevalence : 0.4670
##       Balanced Accuracy : 0.7398
##
##        '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  377 180
##        yes 120 323
##
##                Accuracy : 0.7
##                  95% CI : (0.6705, 0.7283)
##     No Information Rate : 0.503
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4004
##
##  Mcnemar’s Test P-Value : 0.0006583
##
##             Sensitivity : 0.7586
##             Specificity : 0.6421
##          Pos Pred Value : 0.6768
##          Neg Pred Value : 0.7291
##              Prevalence : 0.4970
##          Detection Rate : 0.3770
##    Detection Prevalence : 0.5570
##       Balanced Accuracy : 0.7003
##
##        ’Positive’ Class : no
##
```

Question 5

In this case, there was no difference in accuracy between using the full model as evidence and using the Markov blanket. We hypothesize that the Markov blanket would give slightly worse results in a more complex network, because then we may be ignoring dependencies that do have an impact on the investigated variable. 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.