

# TDDE15 - Lab 1

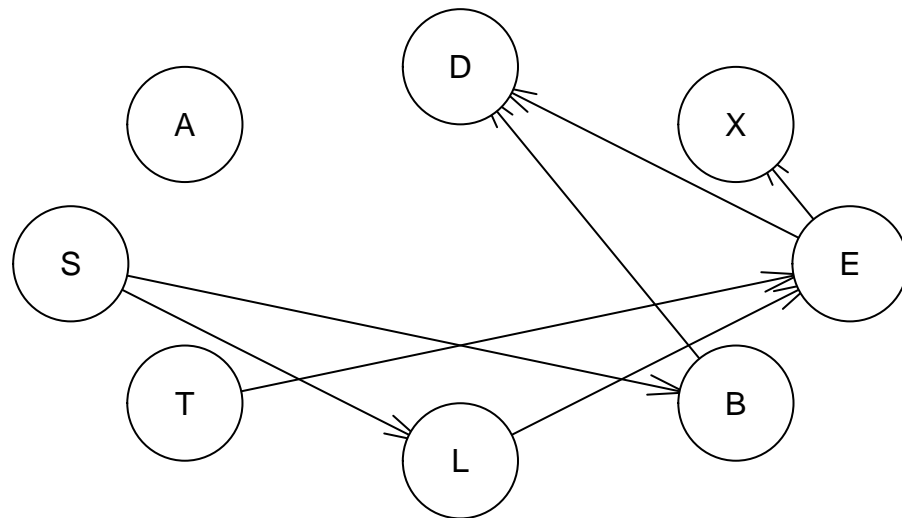
## Question 1

```
library(bnlearn)
data("asia")
structure = hc(asia, restart = 3) #start = initial structure, restart = random restarts, score = score,

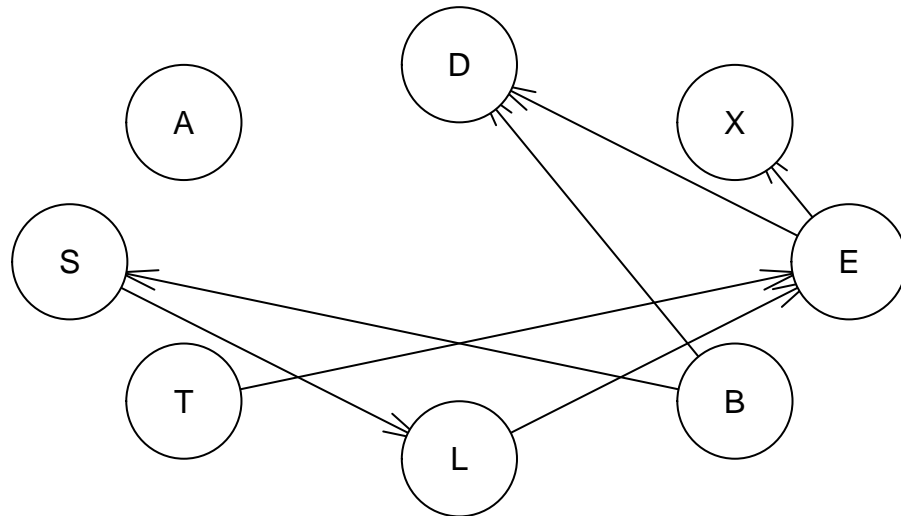
b=T
for (i in 1:100) {
  structure2 = hc(asia, restart = 1)
  b = all.equal(structure, structure2)
  if (b!=TRUE) {
    print("Different network found")
    break
  }
}
```

```
## [1] "Different network found"
```

```
plot(structure)
```



```
plot(structure2)
```



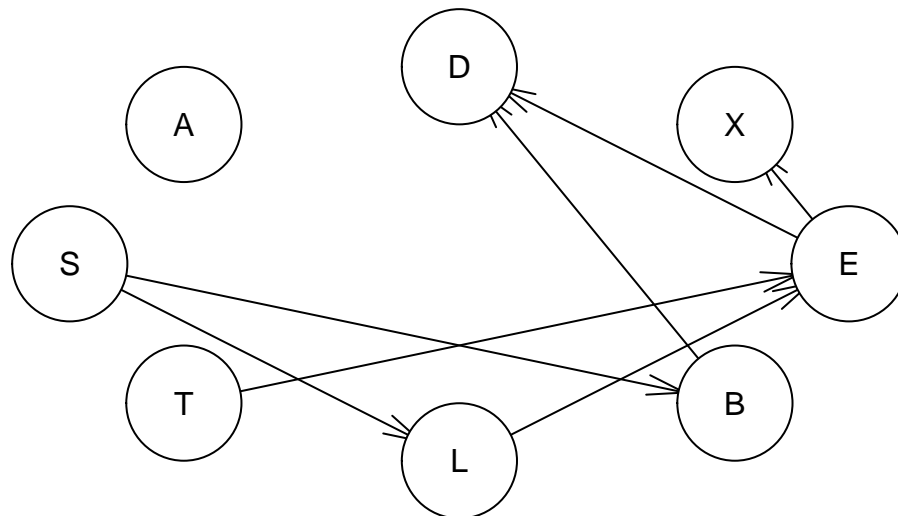
The HC return two different networks, because it can get trapped in a local optimum. HC gives you all independencies that exist in the true model. it does not give you any fals indenpendancies. In this case an edge is reversed, which would give the same score in the HC algorithm.

## Question 2

```
library(gRain)
N = dim(asia)[1]

#
train = asia[1:floor(N*0.8),]
test = asia[(floor(N*0.8)+1):N,]

structure = hc(train, restart = 0)
plot(structure)
```



```

#Learn conditional probabilities given the nodes parents
fit = bn.fit(structure, data=train)
#fit
#coefficients(fit)

# Create Graphical independence network ( grain object )
fit_grain = as.grain(fit)
#fit_grain

# create a junction tree and est. potential clique ( grain object )
#junc_tree = compile(fit_grain)
#print(junc_tree$cptlist)

#remove S from test-data
test_ans = test[, "S"]
test_evid = subset(test, select = -2)

#Predict S
pred_s = c()
for (j in 1:dim(test_evid[1])) {

  # finding/evidence or potentials
  #need to extract observed values correctly.....
  obs = c()
  for (i in 1:7) {
    obs = c(obs, as.character(test_evid[j,i]))
  }
}

```

```

}

nodes_ev = names(test_evid)
evid = setEvidence(fit_grain, nodes_ev, states = obs)
#pEvidence(evid)

# quergrain to get conditional distributon
node = c("S")
prob_s = querygrain(evid, nodes = node)

if (prob_s$S[1]>prob_s$S[2]) {
  pred_s=c(pred_s,"no")
}else{
  pred_s=c(pred_s,"yes")
}
}

table(pred_s,test_ans)

```

```

##      test_ans
## pred_s  no yes
##    no  358 120
##    yes 147 375

```

### Question 3

```

marc_blanc = mb(fit, node = c("S" ))
marc_blanc

```

```

## [1] "L" "B"

```

```

test_evid = subset(test_evid, select=marc_blanc)

pred_s =c()
for (j in 1:dim(test_evid[1])) {

# finding/evidence or potentials
#need to extract observed values correctly.....
  obs = c()
  for (i in 1:dim(test_evid[2])) {
    obs = c(obs, as.character(test_evid[j,i]))
  }

  nodes_ev = names(test_evid)
  evid = setEvidence(fit_grain, nodes_ev, states = obs)
  #pEvidence(evid)

# quergrain to get conditional distributon
  node = c("S")
  prob_s = querygrain(evid, nodes = node)

```

```

if (prob_s$S[1]>prob_s$S[2]) {
  pred_s=c(pred_s,"no")
}else{
  pred_s=c(pred_s,"yes")
}
}

table(pred_s,test_ans)

```

```

##      test_ans
## pred_s  no yes
##    no  358 120
##    yes 147 375

```

## Question 4

```

#Crating an empty network
library(bnlearn)
b_net = empty.graph(names(asia))
b_net

```

```

##
## Random/Generated Bayesian network
##
## model:
##   [A] [S] [T] [L] [B] [E] [X] [D]
## nodes:                                8
## arcs:                                0
##   undirected arcs:                    0
##   directed arcs:                      0
## average markov blanket size:          0.00
## average neighbourhood size:           0.00
## average branching factor:             0.00
##
## generation algorithm:                 Empty

```

```

class(b_net)

```

```

## [1] "bn"

```