Decision Analysis - Bayesian Networks

Author: Can Aytöre

Last Update: 2022-02-20

Contents

1 Introduction

1.1 Aim

1.2 Motivation

- We have an understanding of BNs as graphical models representing probability distributions.
- What does that imply in terms of the underlying probability distribution?
- What happens if a probability distribution factorizes with respect to a graph?
- What kind of computations can we make on BNs?
- What kind of questions can we answer using (quantified) BNs?

2 Getting Started with ...

2.1 What is BN?

A Probabilistic Network (aka causal graph, Bayesian belief network, etc.) is a graphical representation of a **joint probability distribution**.

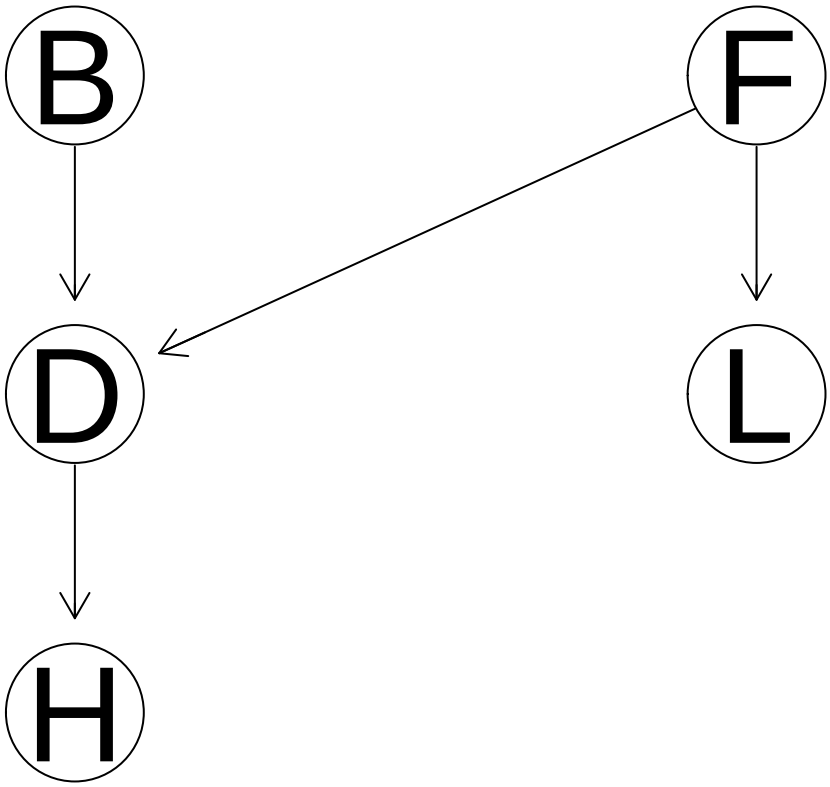
2.2 Running Example: "Is the Family Out?"

When Mr. West goes home at night, he wants to know if his family is home before trying the doors (maybe because the most convenient door to enter is double locked when nobody is home.) Often, when Mrs. West leaves the house, she turns on an outdoor light. However, she sometimes turns on this light if she is expecting a guest. Also (and of course!) the Wests have a dog. When nobody is home, the dog is put in the backyard. The same is true if the dog has bowel troubles. Finally, if the dog is in the backyard, Mr. West will probably hear her barking (or what he thinks is her barking), but sometimes he can be confused by other dogs barking.

```
fo.data <- readRDS(url("https://github.com/canaytore/bayesian-networks/raw/main/data/fo_data.rds")) #fo
head(fo.data)
```

```
##    B   D     F   H     L
## 1 NO  IN FALSE  NO FALSE
## 2 NO OUT  TRUE YES FALSE
## 3 NO  IN FALSE  NO FALSE
## 4 NO OUT  TRUE  NO FALSE
## 5 NO OUT FALSE YES FALSE
## 6 NO  IN  TRUE  NO  TRUE
```

```
library(bnlearn)
library(gRain)
library(ggplot2)

fo.dag <- model2network("[F][B][L|F][D|F:B][H|D]") #Family-out network is created
graphviz.plot(fo.dag)
```



Each iteration results will be collected in following arrays.

```
#First case: P(F=TRUE)
first.mle <- array(dim = 20)
first.bayes <- array(dim = 20)

#Second case: P(D=OUT | B=YES, F=TRUE)
second.mle <- array(dim = 20)
second.bayes <- array(dim = 20)
```

Now that we have a model (fo.dag) and data (fo.data) We can learn the conditional probability tables (parameters) using the bn.fit function which implements the maximum likelihood maximization and a Bayesian method to learn parameters.

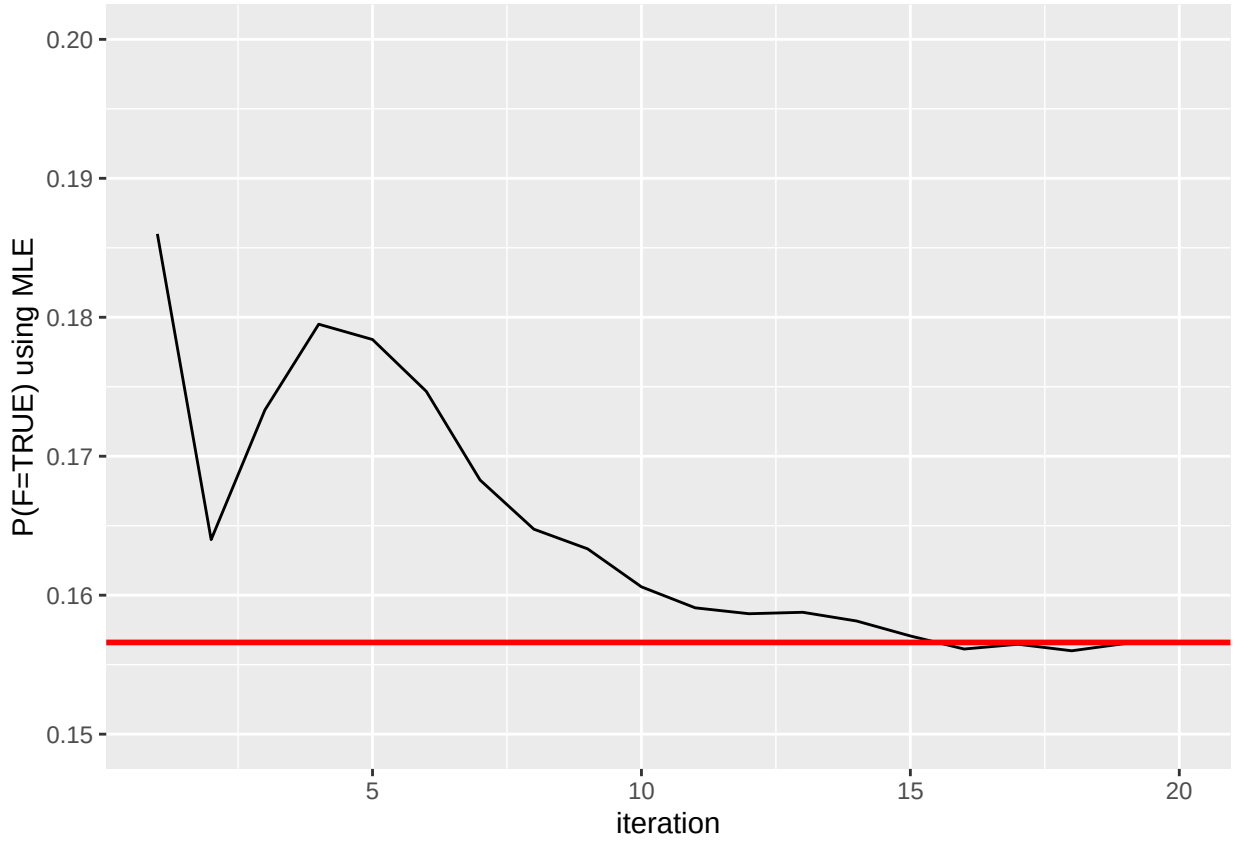
```
for(i in 1:20){
  #P(F=TRUE) using mle:
  first.mle[i] <- bn.fit(fo.dag, fo.data[1:(500*i),])$F$prob["TRUE"]
  #P(F=TRUE) using bayes:
  first.bayes[i] <- bn.fit(fo.dag, fo.data[1:(500*i),], method = "bayes", iss=10)$F$prob["TRUE"]

  #P(D=OUT | B=YES, F=TRUE) using mle:
  second.mle[i] <- bn.fit(fo.dag, fo.data[1:(500*i),])$D$prob["OUT","YES","TRUE"]
  #P(D=OUT | B=YES, F=TRUE) using bayes:
  second.bayes[i] <- bn.fit(fo.dag, fo.data[1:(500*i),], method = "bayes", iss=10)$D$prob["OUT","YES","
}
```

Plotting P(F=TRUE) using mle:

```
first.mle #Each iteration results
```

```
##  [1] 0.1860000 0.1640000 0.1733333 0.1795000 0.1784000 0.1746667 0.1682857
##  [8] 0.1647500 0.1633333 0.1606000 0.1590909 0.1586667 0.1587692 0.1581429
## [15] 0.1570667 0.1561250 0.1564706 0.1560000 0.1565263 0.1566000
```

```
first.mle <- as.data.frame(first.mle)
first.mle$ssize <- 1:20 # add iteration column
ggplot(first.mle, aes(ssize, first.mle)) + geom_line() + geom_hline(yintercept = 0.1566, color="red", s
```

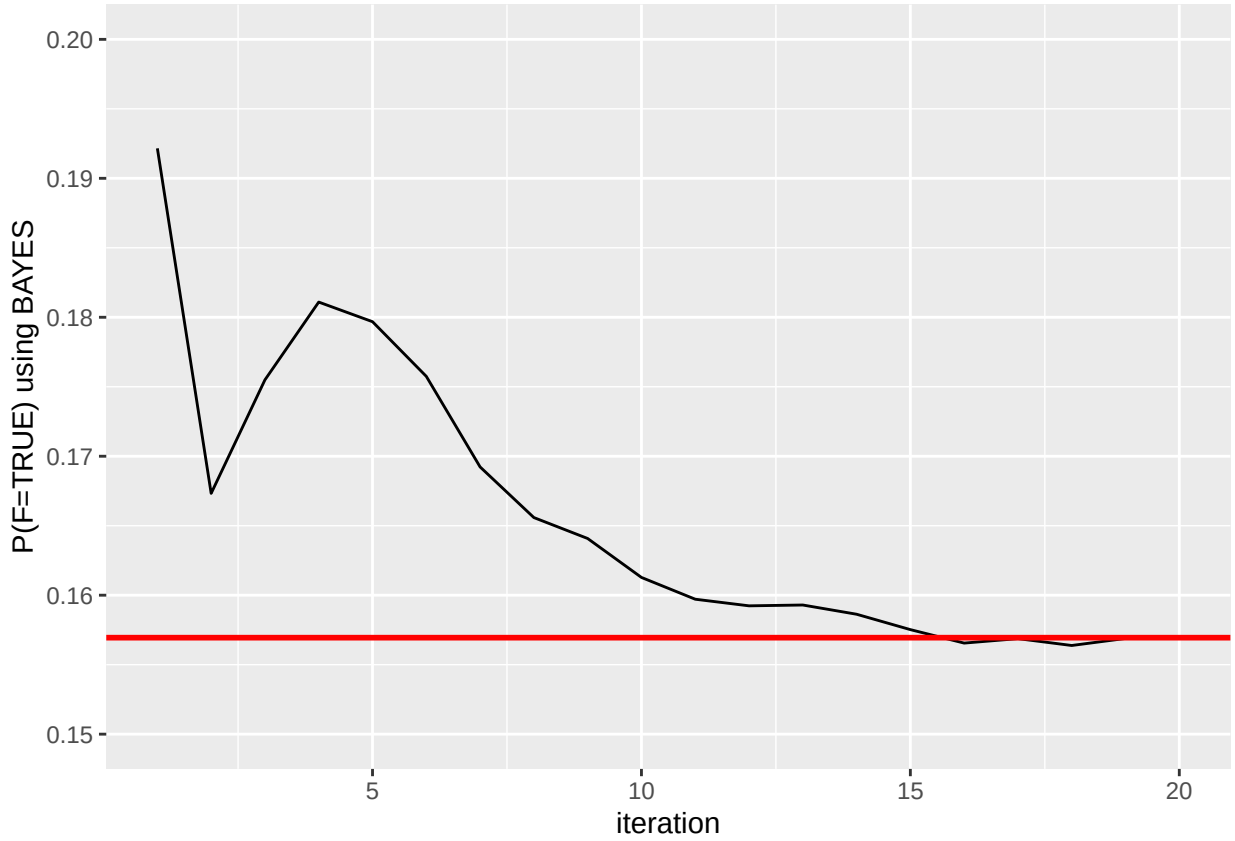


Plotting P(F=TRUE) using bayes:

```
first.bayes #Each iteration results
```

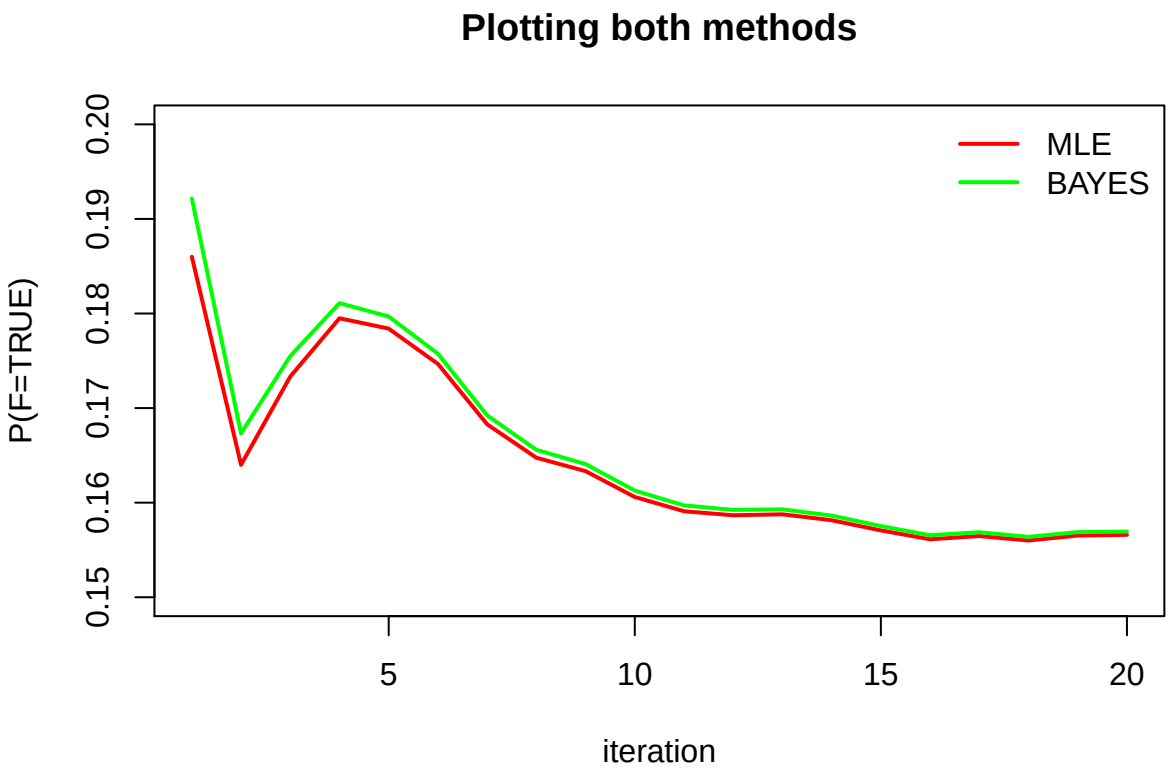
```
##  [1] 0.1921569 0.1673267 0.1754967 0.1810945 0.1796813 0.1757475 0.1692308
##  [8] 0.1655860 0.1640798 0.1612774 0.1597096 0.1592346 0.1592934 0.1586305
## [15] 0.1575233 0.1565543 0.1568743 0.1563818 0.1568875 0.1569431
```

```
first.bayes <- as.data.frame(first.bayes)
first.bayes$ssize <- 1:20 # add iteration column
ggplot(first.bayes, aes(ssize, first.bayes)) + geom_line() + geom_hline(yintercept = 0.1569431, color="
```



Plotting P(F=TRUE) comparing both mle and bayes methods:

```
plot(first.mle[,1], type="l", col="red", lwd = 2, xlab="iteration", ylab="P(F=TRUE)", ylim=range(0.15,0
lines(first.bayes[,1], type="l", col="green", lwd = 2)
legend("topright", legend = c("MLE", "BAYES"), col = c("red","green"), bty='n', lty=1, lwd=2)
```

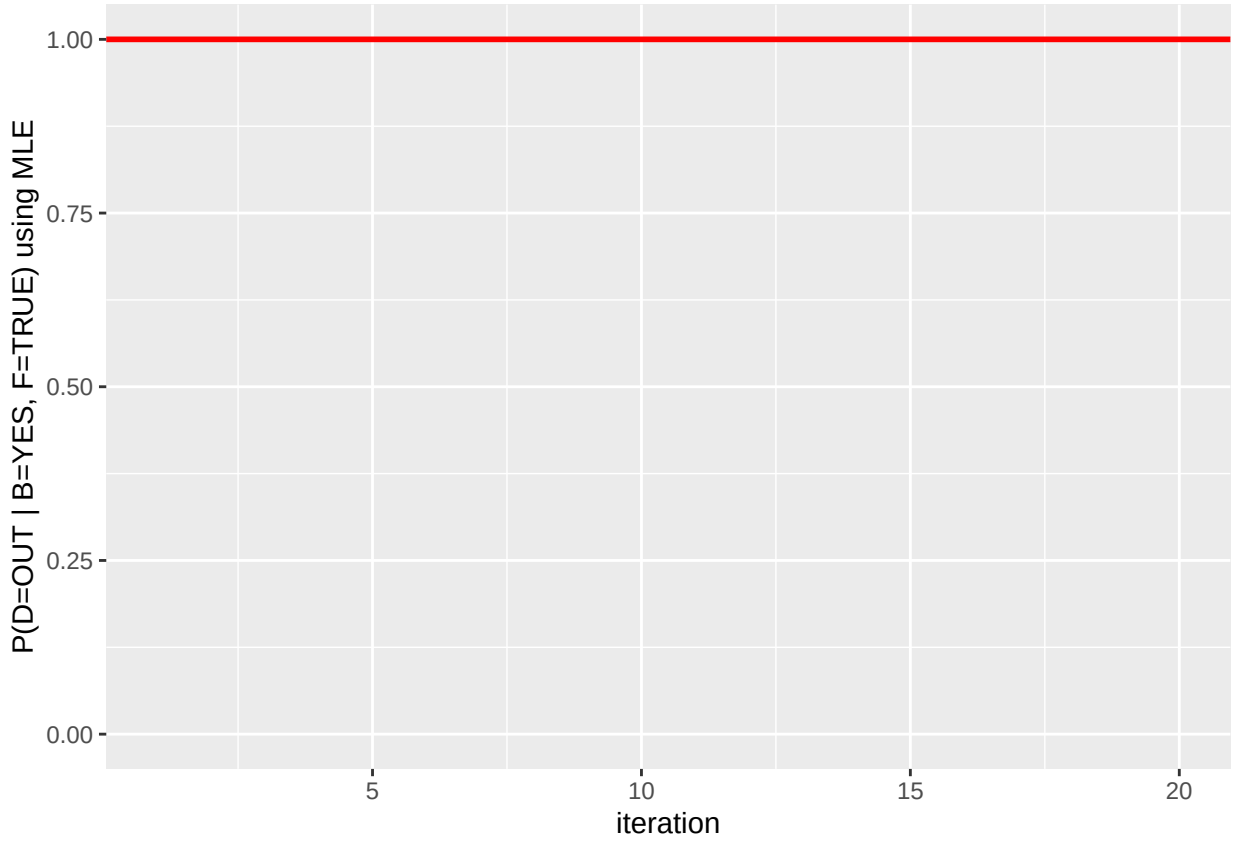


Plotting P(D=OUT | B=YES, F=TRUE) using mle:

```
second.mle #Each iteration results
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
second.mle <- as.data.frame(second.mle)
second.mle$ssize <- 1:20 # add iteration column
ggplot(second.mle, aes(ssize, second.mle)) + geom_line() + geom_hline(yintercept = 1, color="red", size=
```

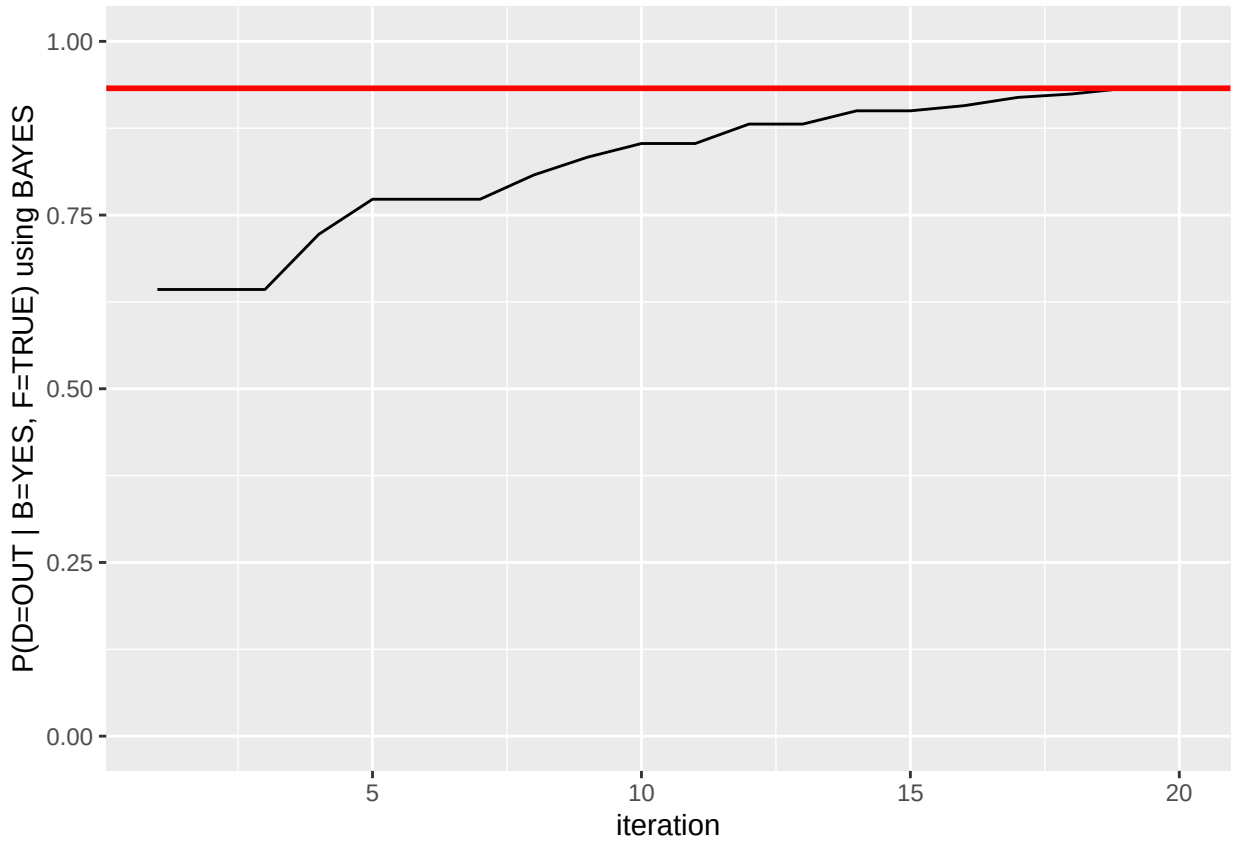


Plotting P(D=OUT | B=YES, F=TRUE) using bayes:

```
second.bayes #Each iteration results
```

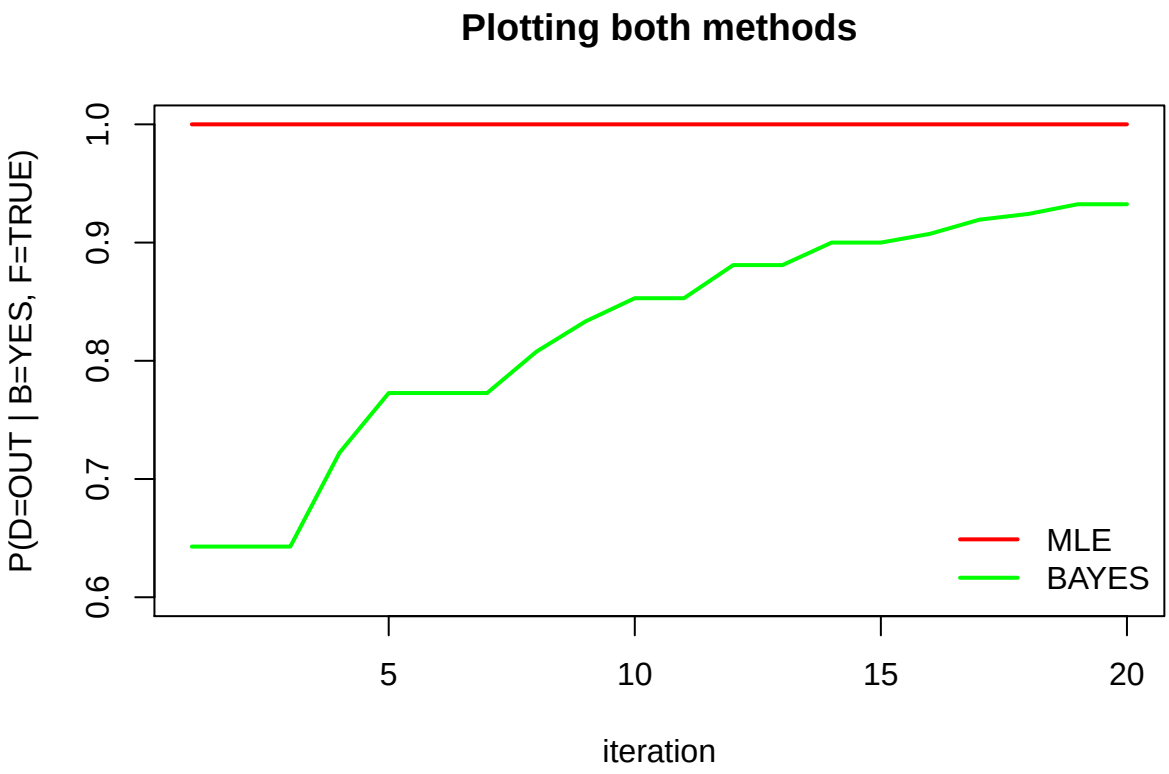
```
##  [1] 0.6428571 0.6428571 0.6428571 0.7222222 0.7727273 0.7727273 0.7727273
##  [8] 0.8076923 0.8333333 0.8529412 0.8529412 0.8809524 0.8809524 0.9000000
## [15] 0.9000000 0.9074074 0.9193548 0.9242424 0.9324324 0.9324324
```

```
second.bayes <- as.data.frame(second.bayes)
second.bayes$ssize <- 1:20 # add iteration column
ggplot(second.bayes, aes(ssize, second.bayes)) + geom_line() + geom_hline(yintercept = 0.9324324, color
```



Plotting P(D=OUT | B=YES, F=TRUE) comparing both mle and bayes methods:

```
plot(second.mle[,1], type="l", col="red", lwd = 2, xlab="iteration", ylab="P(D=OUT | B=YES, F=TRUE)", y
lines(second.bayes[,1], type="l", col="green", lwd = 2)
legend("bottomright", legend = c("MLE", "BAYES"), col = c("red","green"), bty='n', lty=1, lwd=2)
```



2.3 HydeNet

2.3.1 Extended Oil-Wildcatter Problem:

#Part 1.a) Define the extended network in HydeNet.

#Our first task will be to represent and solve the extended oil-wildcatter influence diagram using HydeNet. We generate the network in HydeNet as follows;

```
library(HydeNet)
```

```
## Warning: package 'HydeNet' was built under R version 4.0.5
```

```
## Loading required package: nnet
```

```
ow.net <- HydeNetwork(~ cost | test
                      + drill | test*seismic_results
                      + seismic_results | test*oil_content
                      + drill_reward | drill*oil_content
                      + recovery | drill*oil_content
                      + recovery_level | drill*oil_content
                      + recovery_reward | recovery*recovery_level)
```

For the chance node 'oil_content':

```
oil_content_prob <- c(0.5,0.3,0.2) #Prior probability for dry, wet and soaking.
ow.net <- setNode(ow.net, oil_content, nodeType="dcat", pi=vectorProbs(p=oil_content_prob, oil_content)
```

For the decision nodes 'test', 'drill' and 'recovery':

```
dprob <- c(1,0) #Probability of decision nodes represent do and don't.
ow.net <- setNode(ow.net, test, nodeType="dcat", pi=vectorProbs(p=dprob, test), factorLevels = c("test"
ow.net <- setNode(ow.net, drill, nodeType="dcat", pi=vectorProbs(p=dprob, drill), factorLevels = c("dri
ow.net <- setNode(ow.net, recovery, nodeType="dcat", pi=vectorProbs(p=dprob, recovery), factorLevels =
```

For the chance node 'seismic_results':

```
cpt_seismic_results <- readRDS(url("https://github.com/canaytore/bayesian-networks/raw/main/data/cpt_se
#The file ?cpt_seismic_results.rds? was created using the inputCPT() function of HydeNet. It stands for
cpt_seismic_results
```

```
## , , seismic_results = ns
##
##            oil_content
## test         dry  wet soak
##   test      0.60 0.20 0.20
##   dont_test 0.33 0.33 0.33
##
## , , seismic_results = os
##
##            oil_content
## test         dry  wet soak
##   test      0.30 0.40 0.30
##   dont_test 0.33 0.33 0.33
##
## , , seismic_results = cs
##
##            oil_content
## test         dry  wet soak
##   test      0.10 0.40 0.50
##   dont_test 0.34 0.34 0.34
```

```
ow.net <- setNodeModels(ow.net, cpt_seismic_results)
```

For the chance node 'recovery_level':

```
cpt_recovery_level <- readRDS(url("https://github.com/canaytore/bayesian-networks/raw/main/data/cpt_rec
#The file ?cpt_recovery_level.rds? was created using the inputCPT() function of HydeNet. It stands for
#The trick here is that 'recovery_level' should exactly be nr (no recovery) when the drill is not done
cpt_recovery_level
```

```
## , , recovery_level = nr
##
##             oil_content
## drill        dry wet soak
##   drill        1 0.5  0.3
```

```
##   dont_drill   1 1.0  1.0
##
## , , recovery_level = lr
##
##             oil_content
## drill        dry wet soak
##   drill        0 0.4  0.5
##   dont_drill   0 0.0  0.0
##
## , , recovery_level = hr
##
##             oil_content
## drill        dry wet soak
##   drill        0 0.1  0.2
##   dont_drill   0 0.0  0.0
```

```
ow.net <- setNodeModels(ow.net, cpt_recovery_level)
```

For the utility node 'cost':

```
ow.net <- setNode(ow.net, cost, "determ", define=fromFormula(),
                  nodeFormula = cost ~ ifelse(test == "test", -10, 0))
```

For the utility node 'drill_reward':

```
ow.net <- setNode(ow.net, drill_reward, "determ", define=fromFormula(),
                  nodeFormula = drill_reward ~ ifelse(oil_content == "dry",
                                                      ifelse(drill == "drill", -70, 0),
                                                      ifelse(oil_content == "wet",
                                                             ifelse(drill == "drill", 50, 0),
                                                             ifelse(drill == "drill", 200, 0))))
```

For the utility node 'recovery_reward':

```
ow.net <- setNode(ow.net, recovery_reward, "determ", define=fromFormula(),
                  nodeFormula = recovery_reward ~ ifelse(recovery_level == "nr",
                                                         ifelse(recovery == "recovery", -20, 0),
                                                         ifelse(recovery_level == "lr",
                                                                ifelse(recovery == "recovery", 10, 0),
                                                                ifelse(recovery == "recovery", 30, 0))))

ow.net <- setDecisionNodes(ow.net, test, drill, recovery) #Setting the decision nodes
ow.net <- setUtilityNodes(ow.net, cost, drill_reward, recovery_reward) #Setting the utility nodes
plot(ow.net) #Plotting the influence diagram of extended version of oil-wildcatter problem
```

The trick is here that we will have precise information about ‘oil_content’ after the decision ‘drill’. Besides, there must be an arc from ?drill? to ?recovery_level? since ‘recovery_level’ should exactly be nr (no recovery) when the drill is not done since it is not sensible to be able to do secondary ‘recovery’ without the primary ‘drill’.

References

Appendices

About Author