LAB Assignment 5

Group: CKP

Sarang Nagar (202051168)

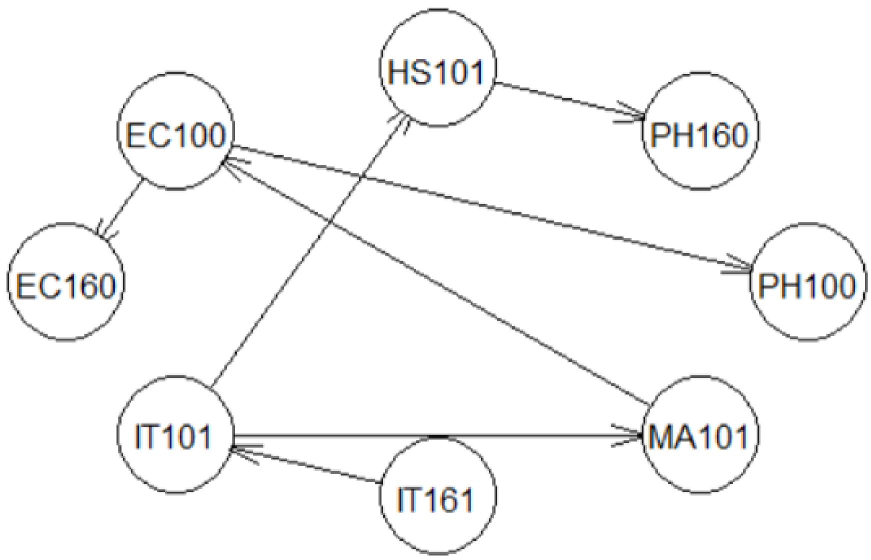
Sonu Raj (202051180)

Subodh Singh (202051184)

Kavish Shah (202051171)

#A Bayesian network is a probability model defined over an acyclic directed graph. It is factored by using one conditional probability distribution for each variable in the model, whose distribution is given conditional on its parents in the graph.

```
# Installs pacman ("package manager") if needed
if (!require("pacman")) install.packages("pacman")

## Loading required package: pacman

# Use pacman to load add-on packages as desired
pacman::p_load(pacman, bnlearn, bnclassify)

#BiocManager package to install two additional packages: "graph" and "Rgra
phviz"
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("graph")

BiocManager::install("Rgraphviz")

# Read the data file using read.table function
data <- read.table("https://raw.githubusercontent.com/pratikiiitv/graphica
lmodels/main/2020_bn_nb_data.txt", header = TRUE, col.names = c("EC100", "
EC160", "IT101","IT161","MA101","PH100","PH160","HS101", "QP"))

# Convert character variables to factor variables
data[sapply(data, is.character)] <- lapply(data[sapply(data, is.character)
], as.factor)

# Convert the data frame into a Bayesian network object
bn<- hc(data[,-9],score = 'k2')

# Inspect the learned Bayesian network structure
plot(bn)
```




```
bn
##   Bayesian network learned via Score-based methods
##
##   model:
##    [IT161][IT101|IT161][MA101|IT101][HS101|IT101][EC100|MA101][PH160|HS
101]
##    [EC160|EC100][PH100|EC100]
##   nodes:                                 8
##   arcs:                                  7
##     undirected arcs:                     0
##     directed arcs:                       7
##   average markov blanket size:           1.75
##   average neighbourhood size:            1.75
##   average branching factor:              0.88
##
##   learning algorithm:                    Hill-Climbing
##   score:                                 Cooper & Herskovits' K2
##   tests used in the learning procedure:  105
##   optimized:                             TRUE

# fit the Bayesian network to the data
fitted_bn <- bn.fit(bn, data[,-9])
fitted_bn$EC100

##
##   Parameters of node EC100 (multinomial distribution)
```

```
##
## Conditional probability table:
##
##       MA101
## EC100          AA         AB         BB         BC         CC         CD
##    AA 0.75000000 0.07692308 0.03846154 0.01851852 0.00000000 0.00000000
##    AB 0.00000000 0.46153846 0.25000000 0.05555556 0.00000000 0.00000000
##    BB 0.25000000 0.23076923 0.32692308 0.22222222 0.04081633 0.00000000
##    BC 0.00000000 0.15384615 0.28846154 0.27777778 0.32653061 0.00000000
##    CC 0.00000000 0.07692308 0.09615385 0.24074074 0.32653061 0.04166667
##    CD 0.00000000 0.00000000 0.00000000 0.12962963 0.26530612 0.33333333
##    DD 0.00000000 0.00000000 0.00000000 0.03703704 0.04081633 0.50000000
##    F  0.00000000 0.00000000 0.00000000 0.01851852 0.00000000 0.12500000
##       MA101
## EC100          DD          F
##    AA 0.00000000 0.00000000
##    AB 0.00000000 0.00000000
##    BB 0.00000000 0.00000000
##    BC 0.00000000 0.00000000
##    CC 0.00000000 0.00000000
##    CD 0.04761905 0.00000000
##    DD 0.19047619 0.00000000
##    F  0.76190476 1.00000000
```

```
fitted_bn$EC160
```

```
##
##   Parameters of node EC160 (multinomial distribution)
##
## Conditional probability table:
##
##       EC100
## EC160          AA         AB         BB         BC         CC         CD
##    AA 0.42857143 0.22727273 0.05714286 0.04166667 0.00000000 0.00000000
##    AB 0.42857143 0.22727273 0.08571429 0.04166667 0.08333333 0.00000000
##    BB 0.14285714 0.31818182 0.20000000 0.22916667 0.08333333 0.03448276
##    BC 0.00000000 0.22727273 0.42857143 0.43750000 0.36111111 0.17241379
##    CC 0.00000000 0.00000000 0.22857143 0.25000000 0.30555556 0.34482759
##    CD 0.00000000 0.00000000 0.00000000 0.00000000 0.11111111 0.27586207
##    DD 0.00000000 0.00000000 0.00000000 0.00000000 0.05555556 0.17241379
##    F  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##       EC100
## EC160          DD          F
##    AA 0.00000000 0.00000000
##    AB 0.00000000 0.00000000
##    BB 0.05000000 0.00000000
##    BC 0.00000000 0.00000000
##    CC 0.25000000 0.02857143
##    CD 0.55000000 0.40000000
##    DD 0.15000000 0.34285714
##    F  0.00000000 0.22857143
```

```
fitted_bn$IT101
```

```
##
##   Parameters of node IT101 (multinomial distribution)
##
## Conditional probability table:
##
##      IT161
## IT101         AA         AB         BB         BC         CC         CD
##    AA 0.35000000 0.08000000 0.05714286 0.02040816 0.00000000 0.00000000
##    AB 0.30000000 0.40000000 0.17142857 0.02040816 0.02380952 0.02857143
##    BB 0.25000000 0.40000000 0.31428571 0.14285714 0.00000000 0.02857143
##    BC 0.10000000 0.04000000 0.28571429 0.36734694 0.28571429 0.14285714
##    CC 0.00000000 0.08000000 0.14285714 0.32653061 0.33333333 0.11428571
##    CD 0.00000000 0.00000000 0.02857143 0.12244898 0.26190476 0.31428571
##    DD 0.00000000 0.00000000 0.00000000 0.00000000 0.04761905 0.34285714
##    F  0.00000000 0.00000000 0.00000000 0.00000000 0.04761905 0.02857143
##      IT161
## IT101         DD          F
##    AA 0.00000000 0.00000000
##    AB 0.00000000 0.00000000
##    BB 0.00000000 0.00000000
##    BC 0.04347826 0.00000000
##    CC 0.04347826 0.00000000
##    CD 0.21739130 0.33333333
##    DD 0.39130435 0.00000000
##    F  0.30434783 0.66666667
```

```
fitted_bn$IT161
```

```
##
##   Parameters of node IT161 (multinomial distribution)
##
## Conditional probability table:
##         AA         AB         BB         BC         CC         CD
DD
## 0.08620690 0.10775862 0.15086207 0.21120690 0.18103448 0.15086207 0.099
13793
##          F
## 0.01293103
```

```
fitted_bn$MA101
```

```
##
##   Parameters of node MA101 (multinomial distribution)
##
## Conditional probability table:
##
##      IT101
## MA101         AA         AB         BB         BC         CC         CD
##    AA 0.16666667 0.04000000 0.00000000 0.00000000 0.02380952 0.00000000
##    AB 0.25000000 0.20000000 0.02941176 0.08163265 0.00000000 0.00000000
##    BB 0.33333333 0.56000000 0.38235294 0.22448980 0.19047619 0.05714286
##    BC 0.16666667 0.16000000 0.29411765 0.36734694 0.23809524 0.22857143
##    CC 0.08333333 0.00000000 0.20588235 0.28571429 0.35714286 0.31428571
##    CD 0.00000000 0.04000000 0.08823529 0.02040816 0.16666667 0.11428571
##    DD 0.00000000 0.00000000 0.00000000 0.02040816 0.02380952 0.22857143
```

```
##    F  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.05714286
##      IT101
## MA101         DD          F
##    AA 0.00000000 0.00000000
##    AB 0.00000000 0.00000000
##    BB 0.00000000 0.00000000
##    BC 0.08695652 0.00000000
##    CC 0.04347826 0.00000000
##    CD 0.30434783 0.08333333
##    DD 0.39130435 0.16666667
##    F  0.17391304 0.75000000
```

```
fitted_bn$PH100
```

```
##
##   Parameters of node PH100 (multinomial distribution)
##
## Conditional probability table:
##
##      EC100
## PH100         AA         AB         BB         BC         CC         CD
##    AA 0.71428571 0.40909091 0.22857143 0.08333333 0.00000000 0.00000000
##    AB 0.14285714 0.31818182 0.20000000 0.18750000 0.05555556 0.00000000
##    BB 0.00000000 0.18181818 0.31428571 0.29166667 0.13888889 0.03448276
##    BC 0.14285714 0.04545455 0.14285714 0.22916667 0.33333333 0.13793103
##    CC 0.00000000 0.04545455 0.11428571 0.18750000 0.25000000 0.41379310
##    CD 0.00000000 0.00000000 0.00000000 0.02083333 0.19444444 0.31034483
##    DD 0.00000000 0.00000000 0.00000000 0.00000000 0.02777778 0.10344828
##    F  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##      EC100
## PH100         DD          F
##    AA 0.00000000 0.00000000
##    AB 0.00000000 0.00000000
##    BB 0.05000000 0.00000000
##    BC 0.00000000 0.00000000
##    CC 0.20000000 0.02857143
##    CD 0.45000000 0.11428571
##    DD 0.20000000 0.45714286
##    F  0.10000000 0.40000000
```

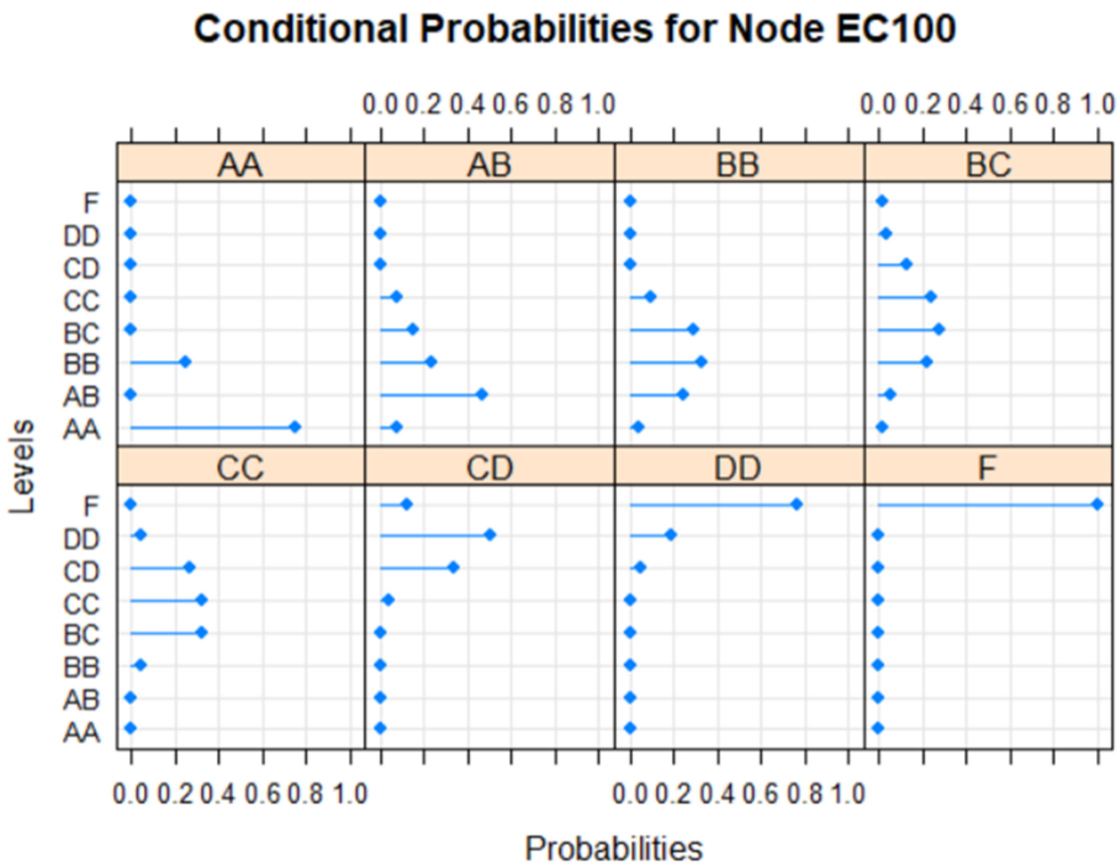
```
fitted_bn$HS101
```

```
##
##   Parameters of node HS101 (multinomial distribution)
##
## Conditional probability table:
##
##      IT101
## HS101         AA         AB         BB         BC         CC         CD
##    AA 0.58333333 0.56000000 0.32352941 0.10204082 0.07142857 0.05714286
##    AB 0.33333333 0.24000000 0.11764706 0.22448980 0.14285714 0.08571429
##    BB 0.00000000 0.12000000 0.26470588 0.26530612 0.26190476 0.11428571
##    BC 0.08333333 0.08000000 0.08823529 0.24489796 0.23809524 0.20000000
##    CC 0.00000000 0.00000000 0.11764706 0.12244898 0.14285714 0.11428571
##    CD 0.00000000 0.00000000 0.05882353 0.02040816 0.14285714 0.20000000
```

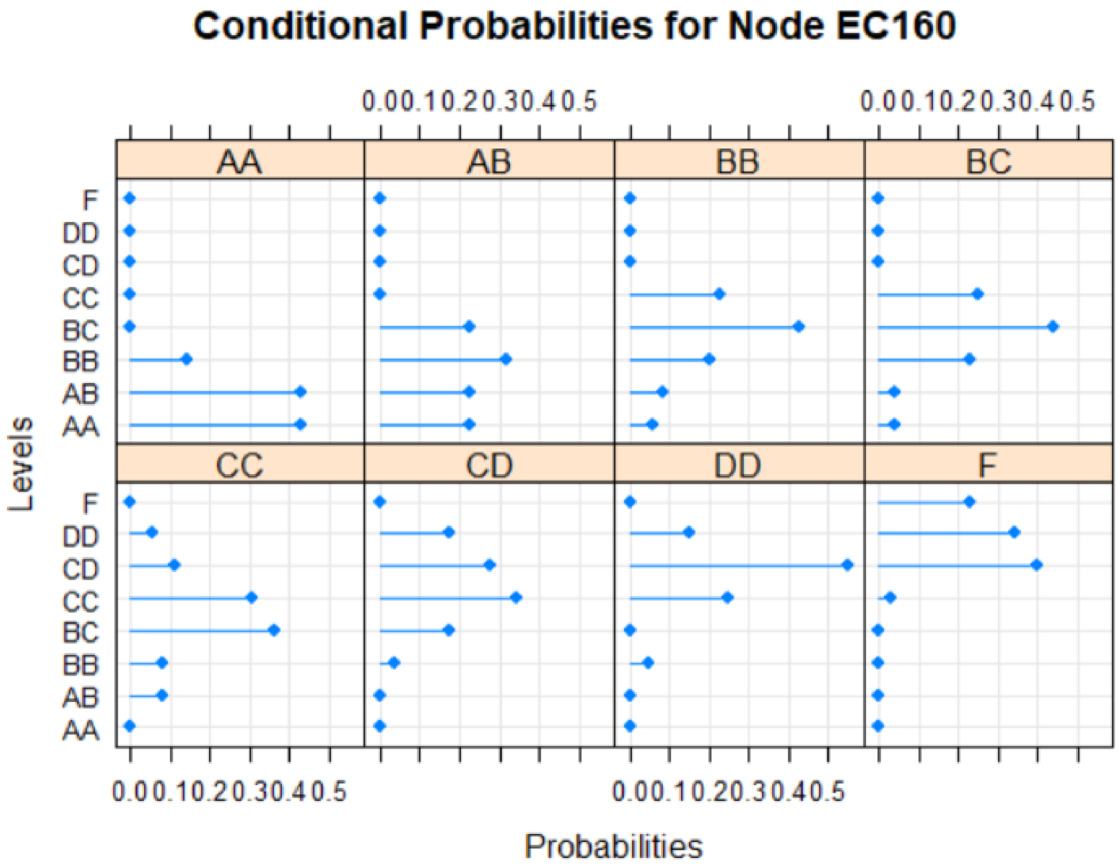
```
##     DD 0.00000000 0.00000000 0.02941176 0.02040816 0.00000000 0.22857143
##     F  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##       IT101
## HS101          DD           F
##     AA 0.00000000 0.00000000
##     AB 0.00000000 0.00000000
##     BB 0.00000000 0.00000000
##     BC 0.04347826 0.00000000
##     CC 0.26086957 0.00000000
##     CD 0.13043478 0.08333333
##     DD 0.52173913 0.58333333
##     F  0.04347826 0.33333333
```

```
# Plot the CPTs of each node as a dot plot (similar to bar chart) using bn
.fit.dotplot
bn.fit.dotplot(fitted_bn$EC100)

## Loading required namespace: lattice
```

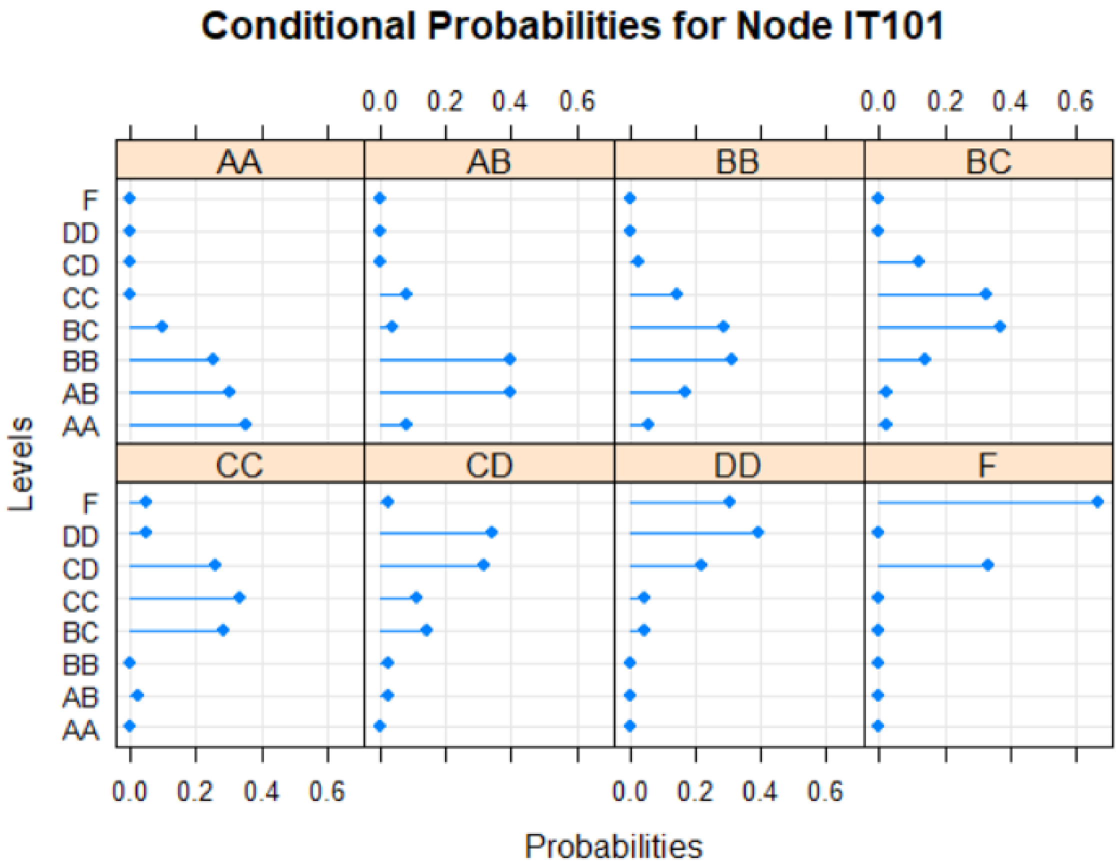
```
bn.fit.dotplot(fitted_bn$EC100)
```



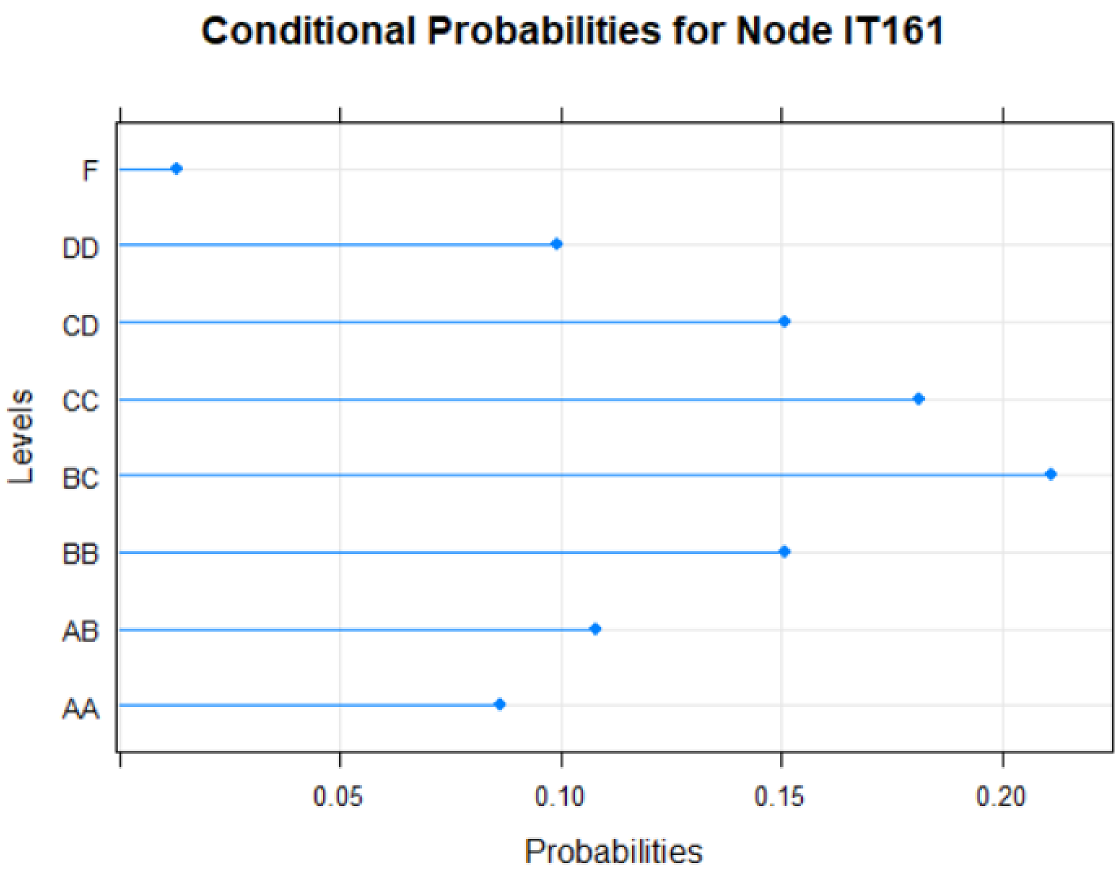

```
bn.fit.dotplot(fitted_bn$EC160)
```



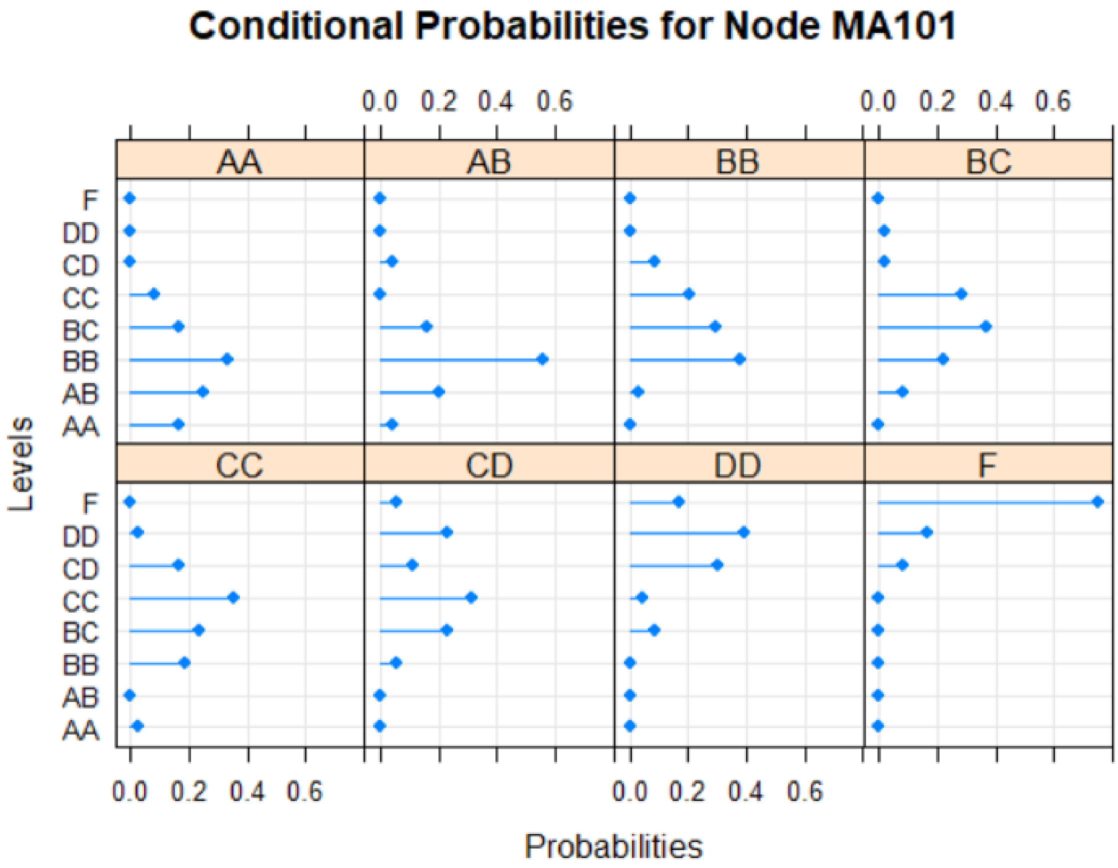

```
bn.fit.dotplot(fitted_bn$IT101)
```



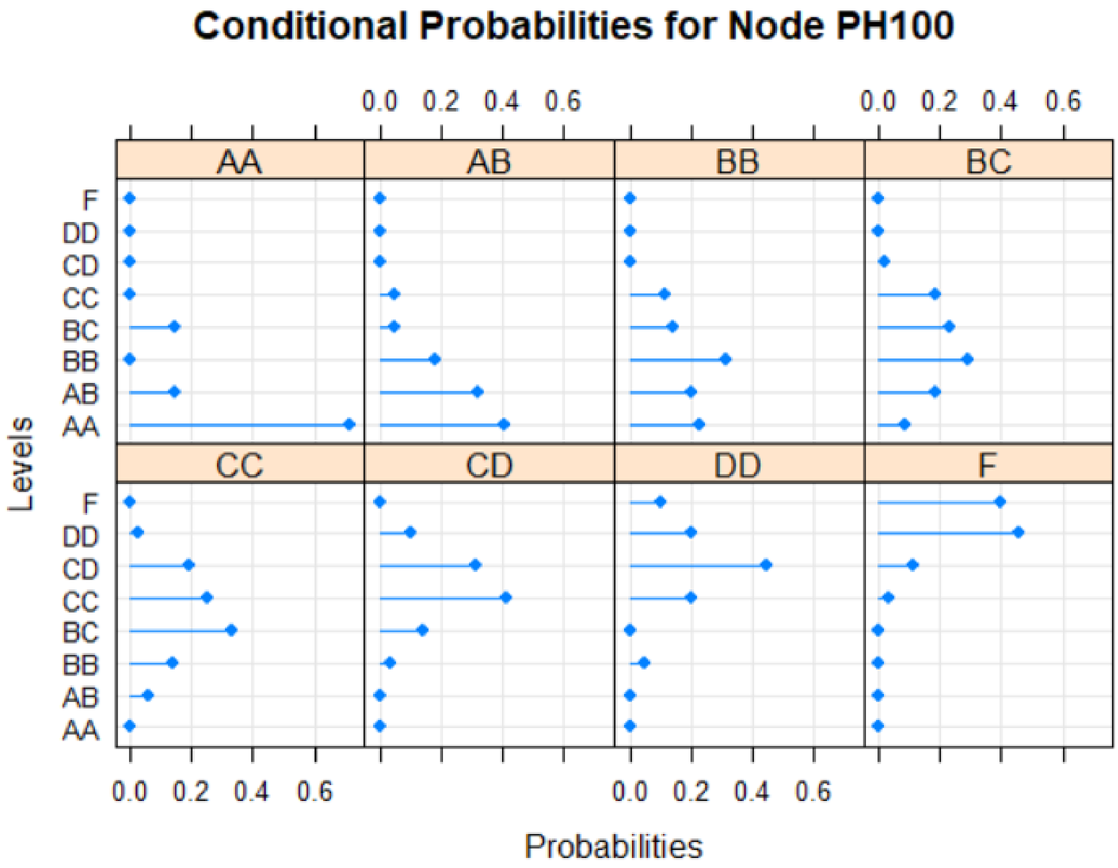

```
bn.fit.dotplot(fitted_bn$IT161)
```



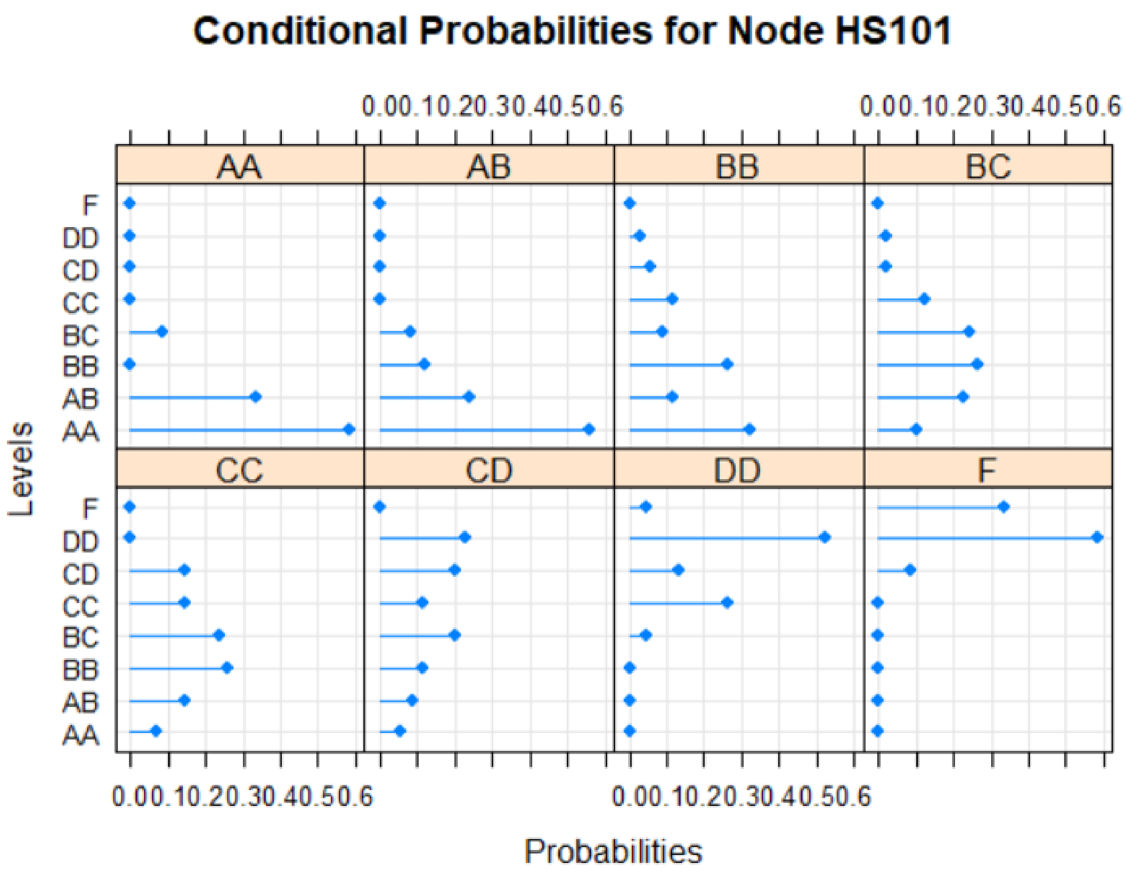

```
bn.fit.dotplot(fitted_bn$MA101)
```



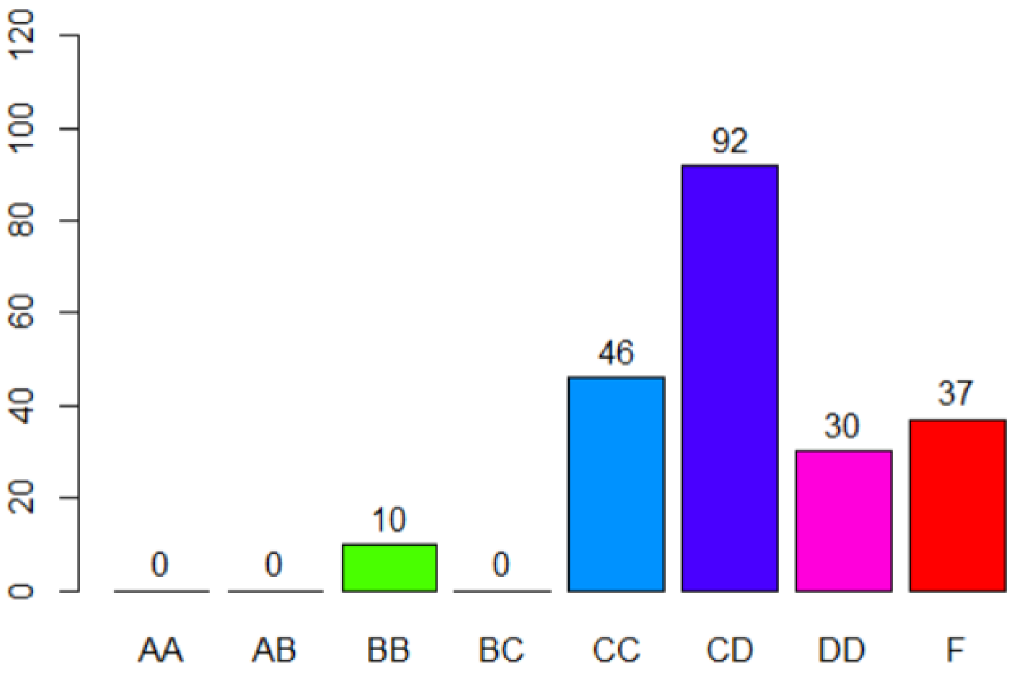

```
bn.fit.dotplot(fitted_bn$PH100)
```




```
bn.fit.dotplot(fitted_bn$HS101)
```

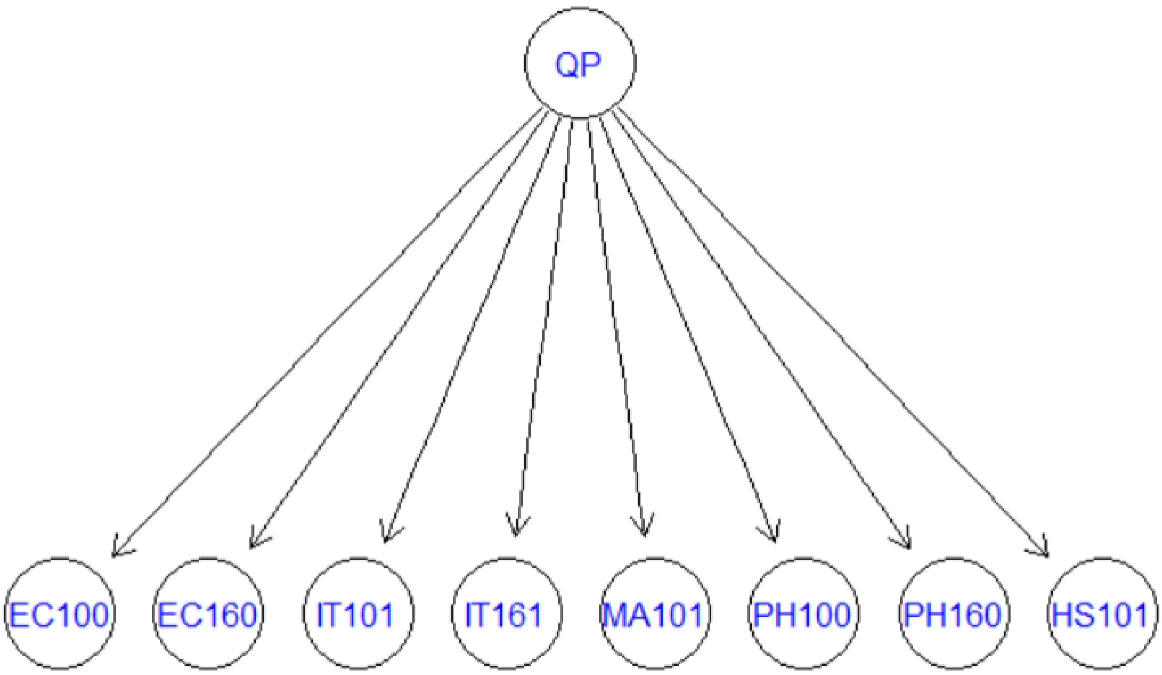



```
# What grade will a student get in PH100 if he earns DD in EC100, CC in IT
101 and CD in MA101:

# Predict the grade in PH100 based on evidence provided
prediction.PH100 <- data.frame(cpdist(fitted_bn, nodes = c("PH100"), evide
nce = (EC100 == "DD" & IT101 == "CC" & MA101 == "CD")))

# plot(prediction.PH100)
my_table <- table(prediction.PH100)
my_table

## PH100
##  AA  AB  BB  BC  CC  CD  DD   F
##   0   0  10   0  40 101  55  24

barp <- barplot(my_table, col = hsv(seq(0, 1, length.out = 8), 1, 1), ylim
= c(0, 120))
text(barp, my_table + 6, labels = my_table)
```



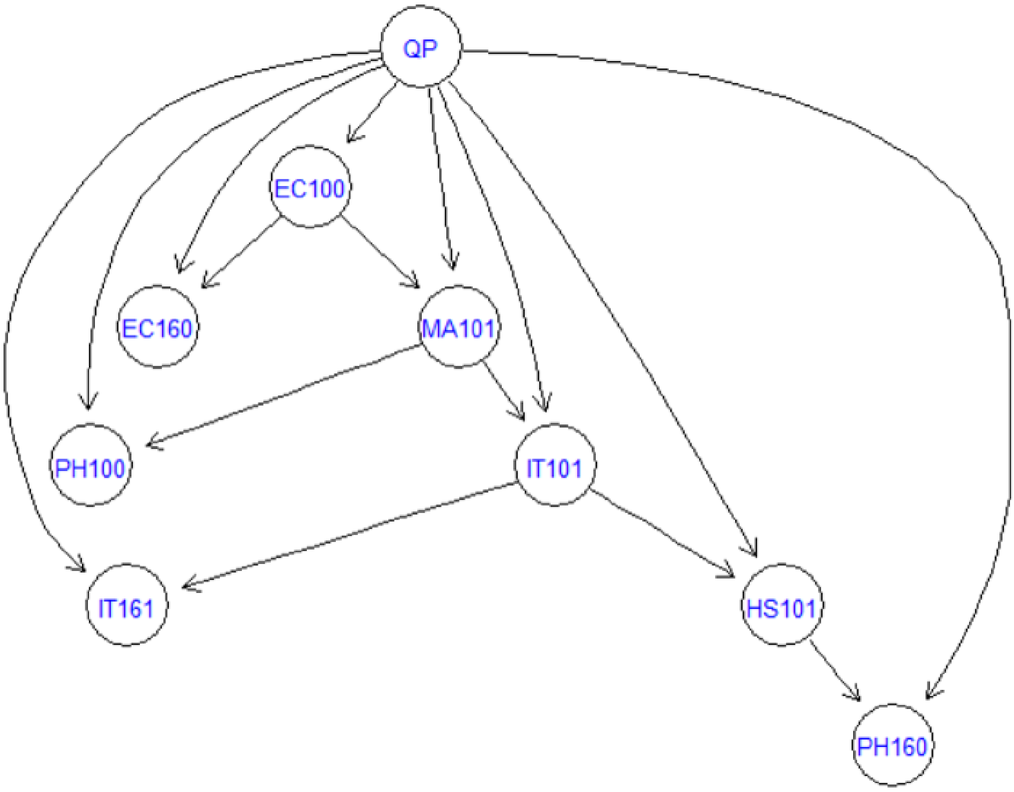

```
# Set the seed for reproducibility
set.seed(101)

# Initialize an empty vector to store accuracy results
accuracy_results <- c()

# Loop 20 times
for (i in 1:20) {

  # Split the data into training and testing sets using the sample functio
n
  sample <- sample.int(n = nrow(data), size = floor(.7*nrow(data)), replac
e = F)
  data.train <-data[sample,]
  data.test<- data[-sample,]

  # Build the naive Bayes classifier on the training data using the nb fun
ction from the bnlearn package.
  nb.grades <- nb(class = "QP",dataset= data.train)

  #Fit the naive Bayes classifier to the training data using the lp functi
on
  nb.grades<-lp(nb.grades, data.train, smooth=0)
  #nb.grades$.params

  #Use the predict function to predict the grades of the test data
```

```
  p<-predict(nb.grades, data.test)
  #Compute the confusion matrix using the table function
  cm<-table(predicted=p, true=data.test$QP)
  cm

  #Compute the accuracy of the prediction using the accuracy function from
the bnclassify package
  accuracy <- bnclassify:::accuracy(p, data.test$QP)

  # Store the accuracy in the vector
  accuracy_results <- c(accuracy_results, accuracy)

}

plot(nb.grades)
```




```
# Report the mean accuracy of the classifier (when courses are independent
of each other)
mean(accuracy_results)

## [1] 0.9528571

accuracy_results2 <- c()

for (i in 1:20) {

  #Build the TAN classifier on the training data using the tan_cl function
```

```
from the bnlearn package.
  tn <- tan_cl("QP", data.train)
  #Fit the TAN classifier to the training data using the lp function
  tn <- lp(tn, data.train, smooth = 1)


  #Use the predict function to predict the grades of the test data.
  p <- predict(tn, data.test)

  #Compute the confusion matrix using the table function
  cm1<-table(predicted=p, true=data.test$QP)
  cm1

  #Compute the accuracy of the prediction using the accuracy function from
the bnclassify package
  accuracy2 <- bnclassify:::accuracy(p, data.test$QP)

  # Store the accuracy in the vector
  accuracy_results2 <- c(accuracy_results, accuracy2)
}

plot(tn)
```




```
# Report the mean accuracy of the classifier (considering that the grades
earned in different courses may be dependent)
mean(accuracy_results2)

## [1] 0.952381
```

References

1. Bayesian Network without Tears by Eugene Charniak

2. Bayesian Networks with R by Bojan Mihaljevic

3. bnstruct: an R package for Bayesian Network Structure Learning with missing data by Francesco Sambo and Alberto Franzin

4. Introduction to Artificial Intelligence by Stuart Russell and Peter Norvig

5. https://github.com/TanmayAmbadkar/CS302-AI/tree/master/Lab5

6. https://github.com/pratikiiitv/graphicalmodels