BN - Assignment 2

Thomas Rost

Saturday, December 13, 2014

Contents

1 comparison of two learning algos

1.1 1: data discretization

Investigate what is the effect of data discretisation on the structure of the learnt Bayesian network, e.g., in terms of number of arcs, changes in arc direction, etc., by considering three discretisations of the continuous features in the iris dataset with different number of bins (see Section B.3). Do this for both classes of learning algorithms and compare the results.

Discretizing the iris data set with the hartemink algorithm leads to the networks shown in the figures below. The first three depict a score-based algorithm, while the second triple shows the effect of discretization on constraint-based algorithms. ###General considerations: The best way to deal with data is case dependent. In the case of discretization, some Datasets might benefit from more bins while others benefit from less, depending on the distribution of the variables within the set. In the case of the iris data set, a bin number of three seems to be a sensible choice, given the three speicies. When looking at the plots, we can see that both petal width and petal length can be binned in three bins with only minor overlap between the species. However, sepal length and width have a big overlap and might benefit from higher bin numbers. We considered three, five and seven bins for both score and constraint based algorithms.

1.1.1 Score Based:

Three bins leads to four arcs with relatively sensible directions: species influences petal length which influences sepal length, seepal width influences petal width which in turn influences speices. The graph is well-connected, each variable will play a role in classifying the species, even if two of them are indirect. The causality seems a bit odd, but not completely against intuition. All arcs are directed.

five bins leads to four arcs with different causality compared to three bins: sepal length now influences petal length which influences petal width which influences species which in turn influences sepal width. On a first glance, the causality in the three bins case seems to make more sense, but again the logic does not ddefy intuition on a disturbing level. All arcs are directed.

seven bins also leads to four arcs. The causality in this case seems different to the others at least in two major ways: Species is a leaf and one node (petal width) has two children, compared to the one on one mapping of parents to children in the previous sections. The logic in this cacse seems weaker compared to the other two bin number cases, since species should have a more direct influence to sepal width.

```
plot(iris)
```

```
for( bins in c(3,5,7)){
tmp=discretize(iris[-5], method = 'hartemink',ibreaks=bins)
NewIris = cbind(tmp,iris[5])

IrisNetsb <- tabu(NewIris)

numNodes = length(IrisNetsb$nodes)
numArcs = length(IrisNetsb$arcs[,1])

plot(IrisNetsb, font.main = 1, main = paste("SB, ibreaks = ", bins, "number of nodes:" , numNodes, "bum




}
```

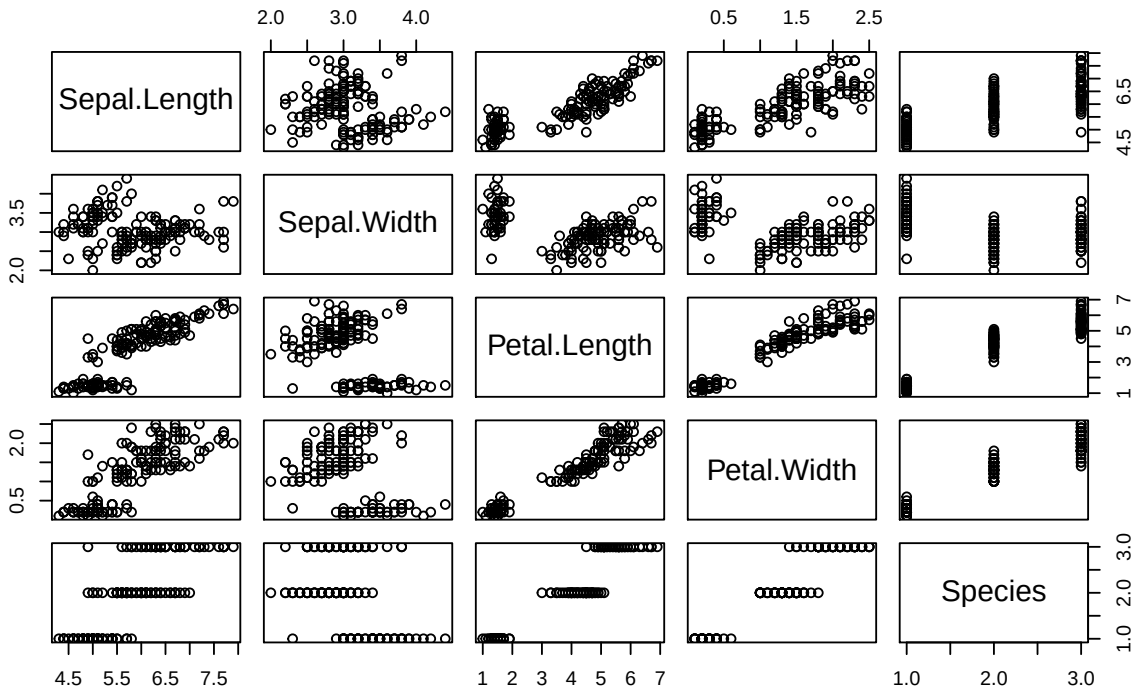
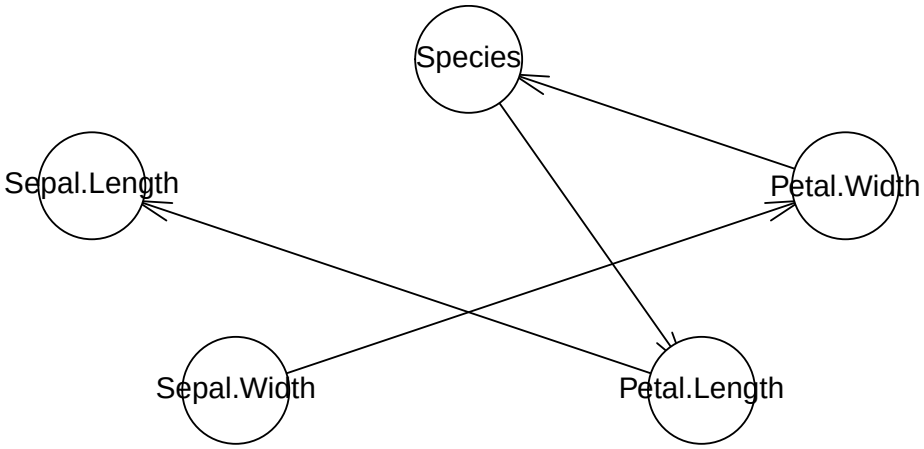
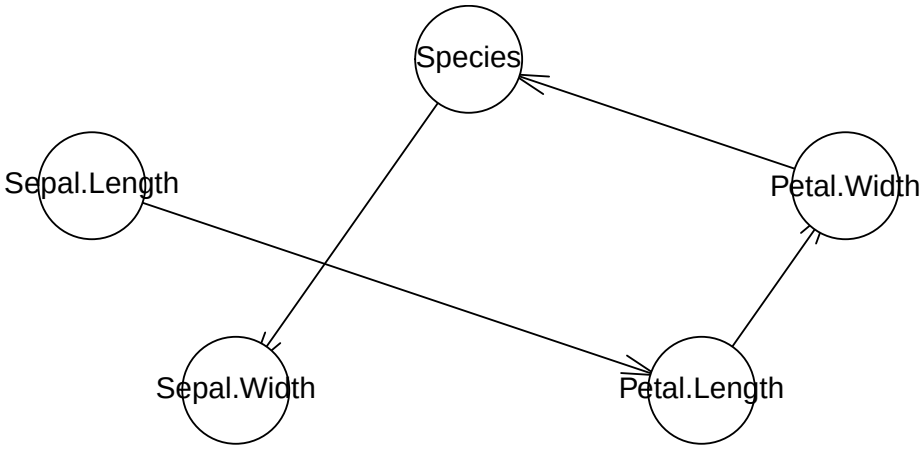


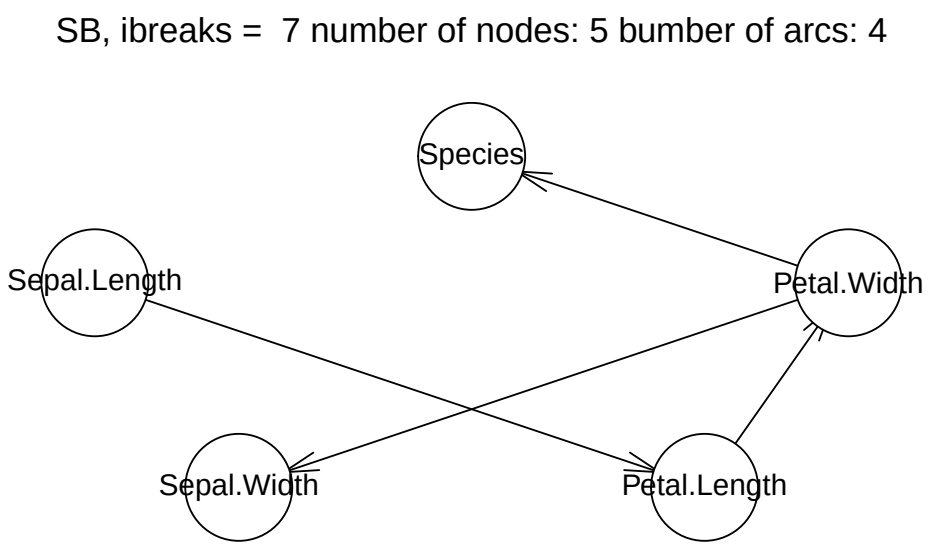
Figure 1: SB , ibreaks = 3

SB, ibreaks = 3 number of nodes: 5 bumber of arcs: 4



SB, ibreaks = 5 number of nodes: 5 bumber of arcs: 4





###Constraint based:

general statement. Constraint based algorithms (all of them were tried out, although only one is reported here) tend to lead to way less expressive networks. often arcs are undirected and the number of graphs is often significantly lower than with score based algorithms. However, this does not necessarily mean that constraint based algorithms are "worse". The constraint nature leads to more robust predictions. the strengths of the two algorithm classes should be considered for each task individually. (see Bayesian Probabilities for Constraint-based Causal Discovery Tom Claassen and Tom Heskes Radboud University Nijmegen Netherlands). in order to compare score based and constraint based algorithms, in some cases the cextend() function has been used.

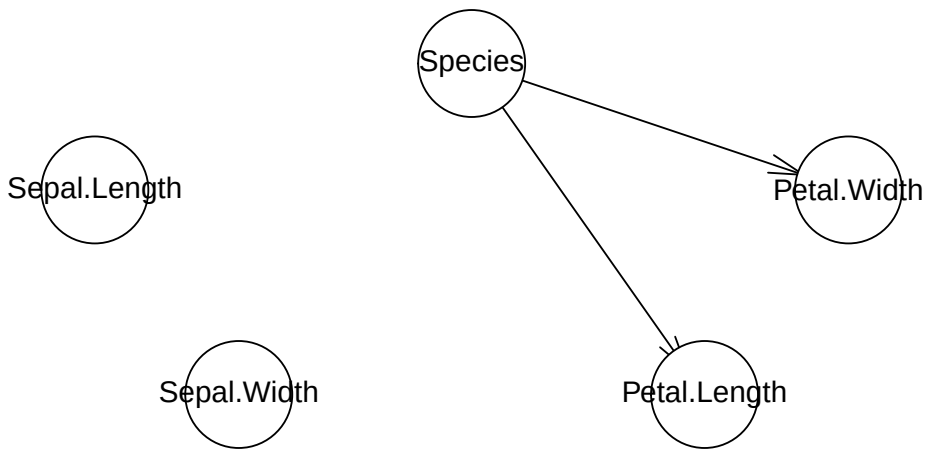
three bins: only two arcs, between petal width and petal length. No directions. sepal length and width are separate networks that aren't connected to species.

five bins: four arcs corresponding to the five bins score based algoriuthm's outcome.

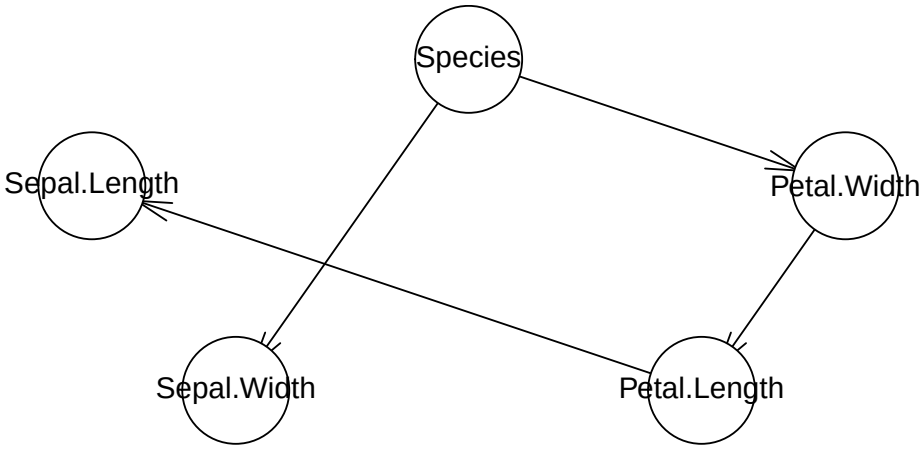
seven bins: three arcs, only connection between species and another node is petal width. The rest is a separate network.

```
for( bins in c(3,5,7)){
tmp=discretize(iris[-5], method = 'hartemink',ibreaks=bins)
NewIris = cbind(tmp,iris[5])

IrisNetcb <- iamb(NewIris)
IrisNetcb2 <- cextend(IrisNetcb)

numNodes = length(IrisNetcb$nodes)
numArcs = length(IrisNetcb$arcs[,1])

plot(IrisNetcb2, font.main = 1, main = paste("CB, ibreaks = ", bins,  "number of nodes:" , numNodes, "bu
```

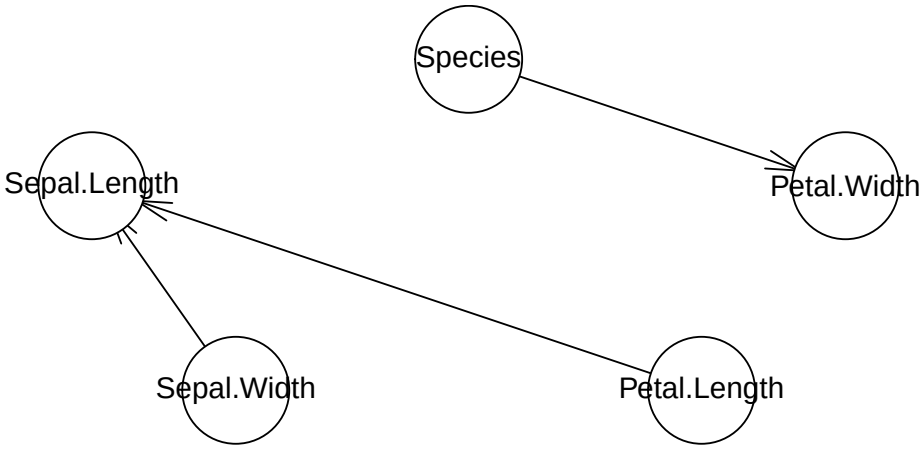
```
}
```


CB, ibreaks = 3 number of nodes: 5 bumber of arcs: 4
Species
Sepal.Length
Petal.Width
Sepal.Width
Petal.Length

CB, ibreaks = 5 number of nodes: 5 bumber of arcs: 8



CB, ibreaks = 7 number of nodes: 5 bumber of arcs: 4



1.2 2: size of dataset

Investigate what is the effect of the size of the dataset on the structure of the learnt Bayesian network by considering three subsets of the breast cancer dataset with different number of samples. Do this for both classes of learning algorithms and compare the results.

1.2.1 general disclaimer:

again, the constraint based algorithms do not always lead to directed nodes. the same function has been used as before. The effect of number of training examples on performance of the training algorithms has been examined. every tenth data point, half of the data and the full data set have been examined.

1.2.2 SB

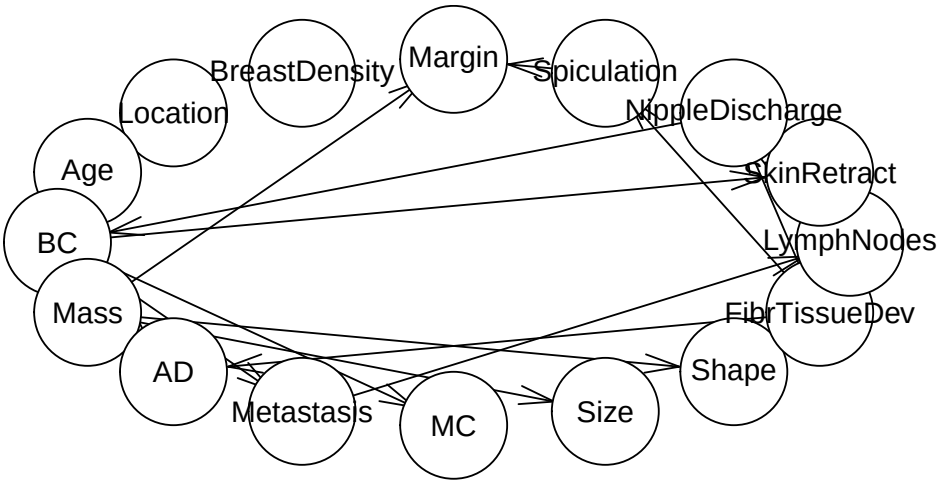
for score based algorithms, there is no difference between models whether the full or the half data set is used. The number of arcs and arc direction are the same. However, when only a tenth of the data set has been used, two arcs were not inferred. This seems to be in line with the idea that more training data generates more sophisticated models.

1.2.3 CB

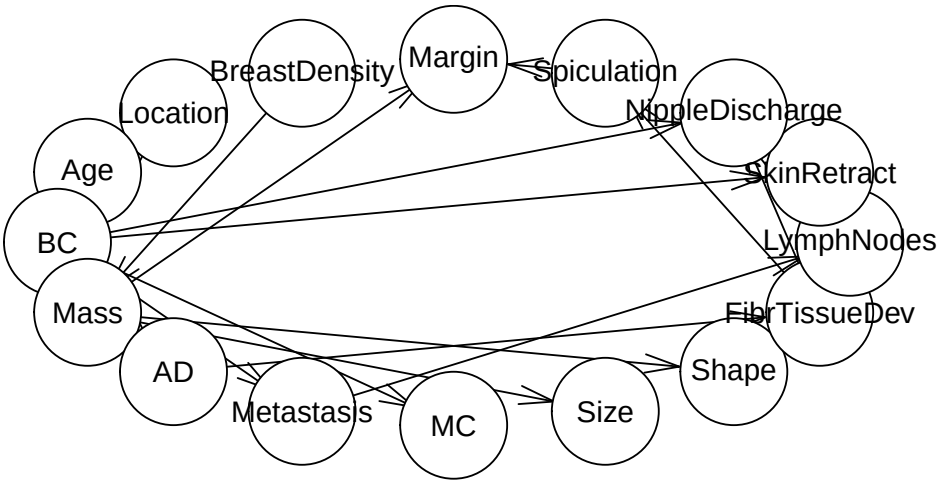
for constraint based algorithms, evaluation of node direction is complicated. Therefore the discussion will be only for arc number. Again, no difference in arc number for full or half data set, however, a node is missing in the tenth-dataset-model. Again, this seems to match intuition.

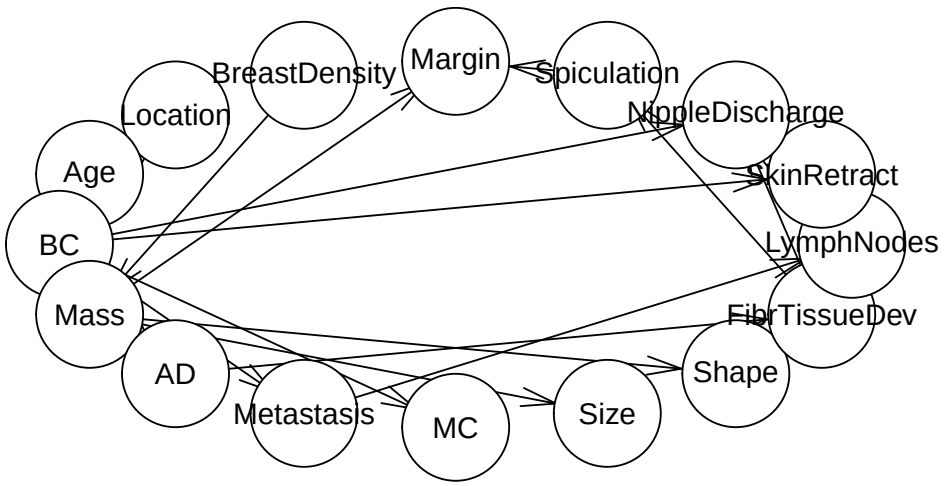
```
a <- read.csv('bc.csv')

#plot(a)

for(part in c(10,2,1)){
    ind = sample(1:(nrow(a)/part))
    BC <- a[ind,]

    BC_sbl <- tabu(BC)


numNodes = length(BC_sbl$nodes)
numArcs = length(BC_sbl$arcs[,1])


    plot(BC_sbl , font.main = 1, main = paste("SB, every ", part, "Datapoint", bins,  "number of nodes:"


}
```

SB, every 10 Datapoint 7 number of nodes: 16 bumber of arcs: 16



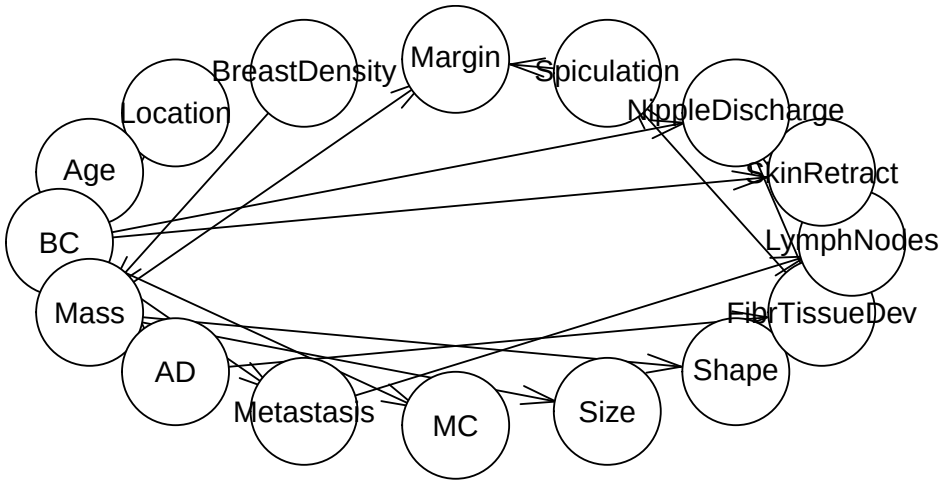
SB, every 2 Datapoint 7 number of nodes: 16 bumber of arcs: 18



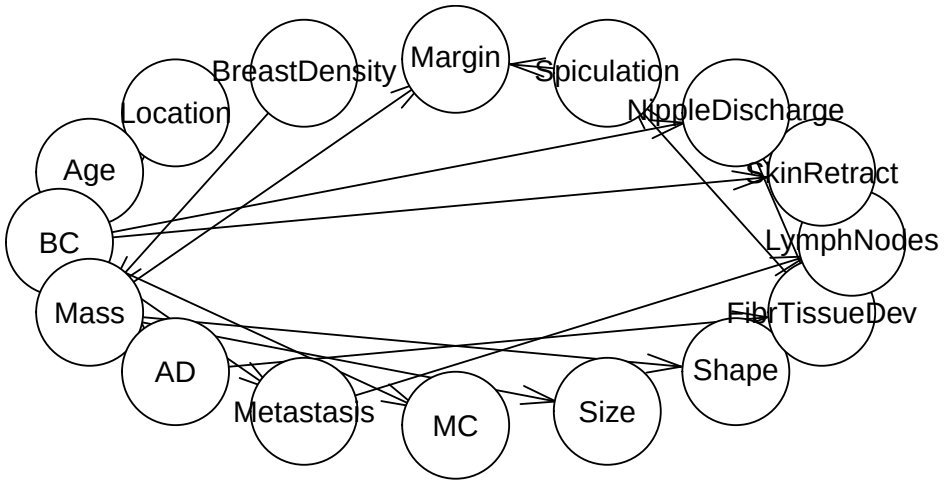



```
for(part in c(1,2,10)){
    ind = sample(1:(nrow(a)/part))
    BC <- a[ind,]

    BC_cbl <- iamb(BC)
    BC_cbl2 <- cextend(BC_sbl)
    numNodes = length(BC_cbl$nodes)
    numArcs = length(BC_cbl$arcs[,1])

    plot(BC_cbl2,  font.main = 1, main = paste("CB, every ", part, "Datapoint", bins,  "number of nodes



}
```

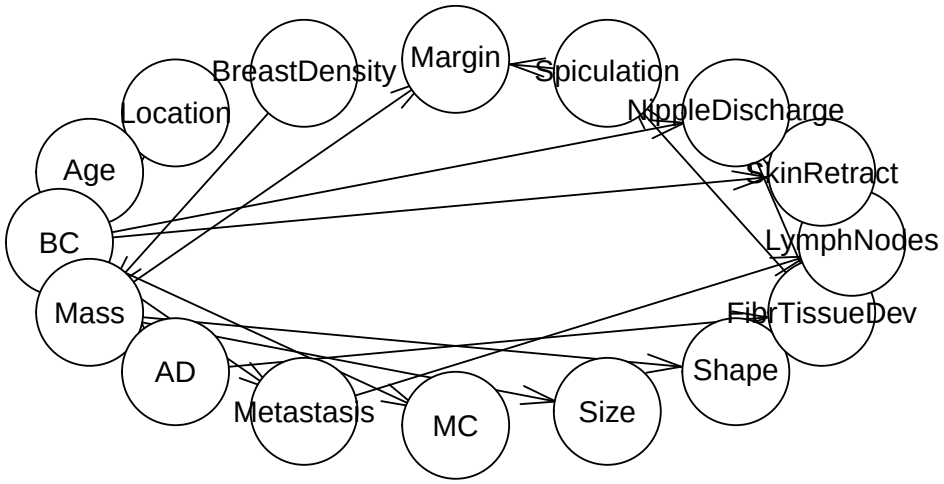
CB, every 1 Datapoint 7 number of nodes: 16 bumber of arcs: 20



CB, every 2 Datapoint 7 number of nodes: 16 bumber of arcs: 20



CB, every 10 Datapoint 7 number of nodes: 16 bumber of arcs: 19



2 comparison to manually constructed bayes network

3 3:define measures for quality of a learning algorithm in terms of a known bayes network structure, motivate definition

Define measures that can be used to determine the quality of a learning algorithm in terms of a known Bayesian network structure. Motivate the definition of these measures

I have to be honest: I do not really understand that Question. Do the authors suggest I find a measure that is original in describing the quality of a bayesian Network? I'm sure that can't be true.

My measure for quality of the learning algorithm is Aruhga, which is the Quotient between nodes and vertices.

$$Aruhga = \frac{Nodes}{Vertices}$$

Arhuga is a binary measure with the classes {useful, not useful}, depending on wether Aruhga falls inside the interval $[\alpha, \beta]$. α and β are depending heavily on the purpose and topic of the network and should normally be defined by logical reasoning before constructing the network. However, since the expert knowledge and necessary publications are not always readily available, the author provides some default values with $\alpha = 2.3$ and $\beta = 1.2/Nodes!$. Please note that Aruhga is a measure of the connectivity of the network. values above alpha may indicate that too many nodes are used, thus either weakening the correlation between the single nodes OR bringing in Nodes that are not relevant for each other. Networks with a value below beta indicate that the network is highly connected which might reduce functionality and also might point to a circular relationship between variables in the real world that cannot be represented by a DAG.

3.1 4:use measures to evaluate both classes of learning algs for NHL dataset

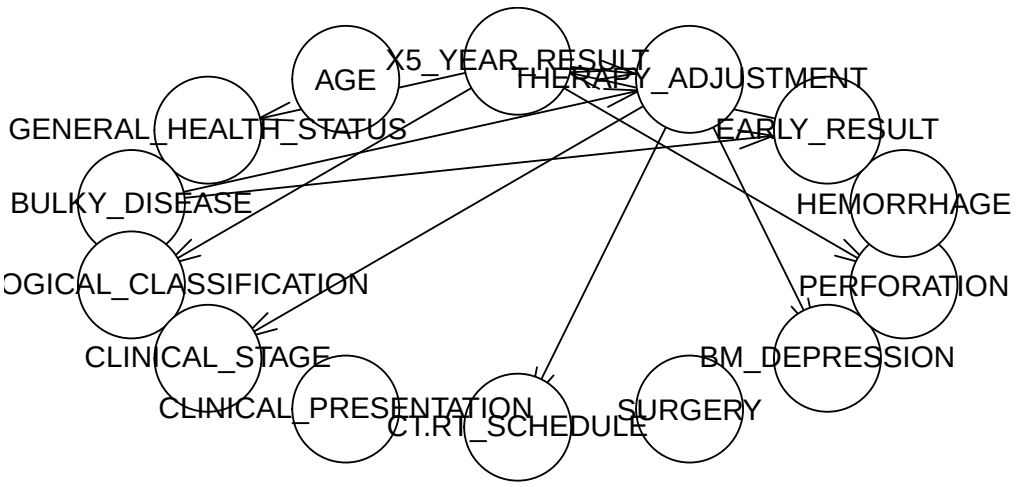
Use these measures in order to evaluate the quality of both classes of learning algorithms for the NHL dataset.

As we can see, the score based algorithm falls within a healthy 1.4 in scores of Arugha. Thus we can conclude that the model might be useful

The constraint based algoritm however does not lie within the limits of a "useful" Arugha, therefore we can conclude that it isn't very useful. This seems to go well with the intuition one gets when looking at the plot. Please note that some arcs are obstructed in the graph due to the size of the nodes.

```
Tmp = read.csv("nhl.csv",nrows = 5)


class = rep(list("factor"),ncol(Tmp))


NHL = read.csv("nhl.csv",colClasses = class)

Tmp2<- complete.cases(NHL) # only take complete cases
NHL <- NHL[Tmp2,]


NHLnet_sbl <- tabu(NHL)
NHLnet_cbl <- iamb(NHL)


numNodes = length(NHLnet_sbl$nodes)
```

```
numArcs = length(NHLnet_sbl$arcs[,1])

plot(NHLnet_sbl, font.main = 1, main = paste("NHL SB, ibreaks = ", bins,  "number of nodes:" , numNodes
```




```
print(length(NHLnet_sbl$nodes)/length(NHLnet_sbl$arcs[,1]))
```

```
## [1] 1.4
```

```
numNodes = length(NHLnet_cbl$nodes)
numArcs = length(NHLnet_cbl$arcs[,1])

plot(NHLnet_cbl, font.main = 1, main = paste("NHL CB, ibreaks = ", bins,  "number of nodes:" , numNodes
```

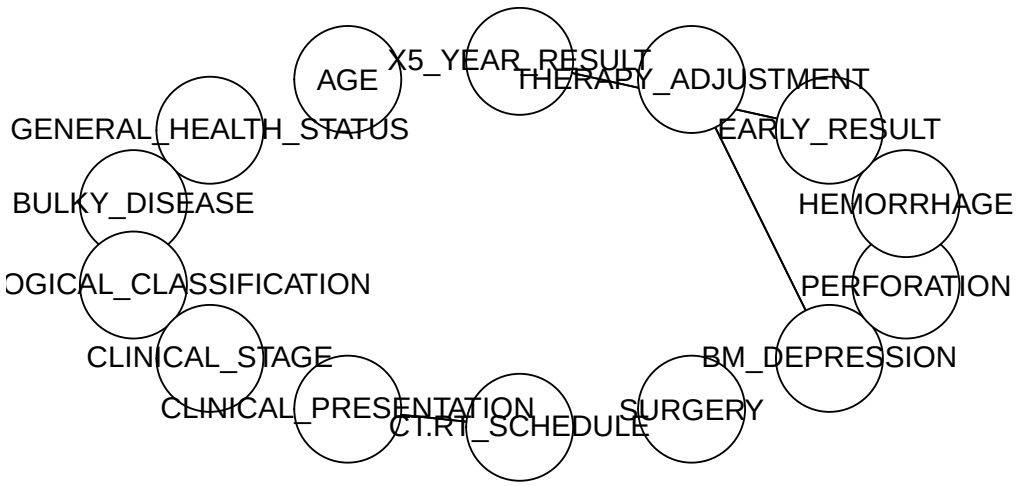
```
print(length(NHLnet_cbl$nodes)/length(NHLnet_cbl$arcs[,1]))
```

```
## [1] 2.333333
```

3.2 5: compare margial prob distributions of learnt NHL bN with those of the manually coinstructed ones

Compare the marginal probability distributions of the learnt NHL Bayesian network with those of the manually constructed network.

since constraint based aslgorithms do not necessarily produce directed graphs, we will only examine score based algorithms.




```
fittedNHL = bn.fit(NHLnet_sbl,NHL)
```

get the manually constructed network:

```
manual = read.net("nhl.net")


test <- bn.net(manual)
plot(test)
```

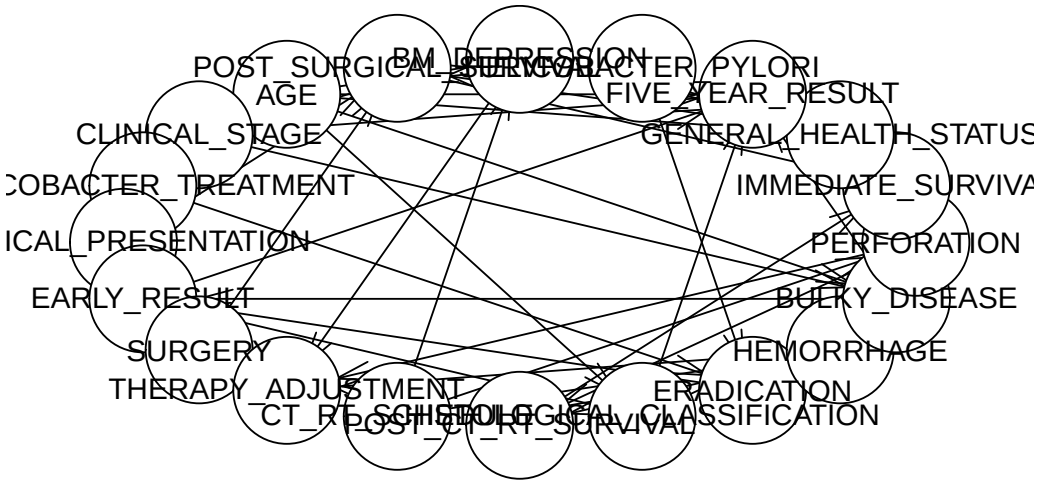
```
print(intersect(NHLnet_sbl$arcs,test$arcs))
```

```
## [1] "THERAPY_ADJUSTMENT"          "EARLY_RESULT"
## [3] "BULKY_DISEASE"               "HISTOLOGICAL_CLASSIFICATION"
## [5] "GENERAL_HEALTH_STATUS"       "CLINICAL_STAGE"
## [7] "PERFORATION"                 "BM_DEPRESSION"
```

```
print(setdiff(names(NHLnet_sbl$nodes),names(test$nodes)))
```

```
## [1] "CT.RT_SCHEDULE" "X5_YEAR_RESULT"
```

```
print(setdiff(names(test$nodes),names(NHLnet_sbl$nodes)))
```




```
## [1] "POST_SURGICAL_SURVIVAL" "HELICOBACTER_TREATMENT"
## [3] "CT_RT_SCHEDULE"         "POST_CT_RT_SURVIVAL"
## [5] "ERADICATION"            "IMMEDIATE_SURVIVAL"
## [7] "FIVE_YEAR_RESULT"       "HELICOBACTER_PYLORI"
```

```
print(setdiff(NHLnet_sbl$arcs,test$arcs))
```

```
## [1] "X5_YEAR_RESULT" "CT.RT_SCHEDULE"
```

```
print(setdiff(test$arcs,NHLnet_sbl$arcs))
```

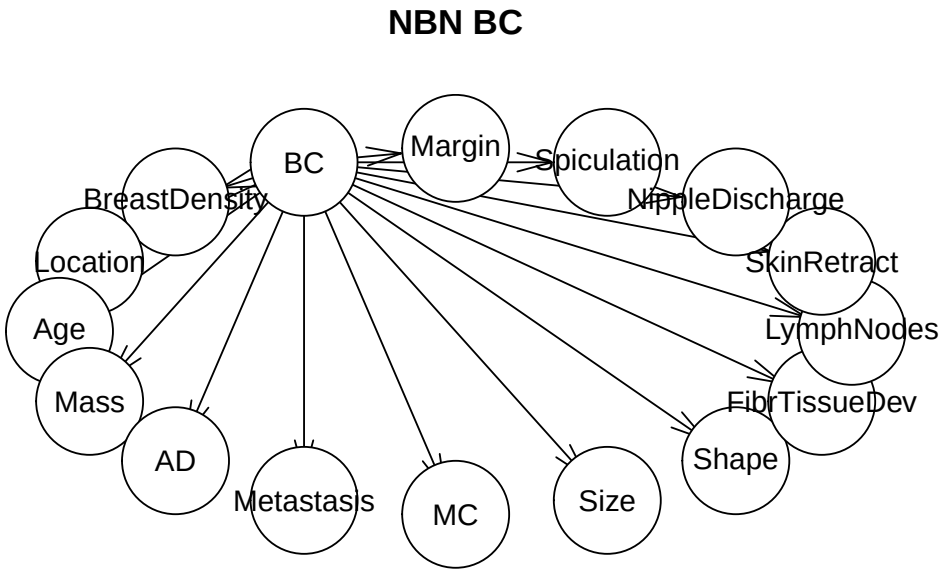
```
##  [1] "POST_SURGICAL_SURVIVAL" "AGE"
##  [3] "HELICOBACTER_TREATMENT" "CLINICAL_PRESENTATION"
##  [5] "SURGERY"                "CT_RT_SCHEDULE"
##  [7] "POST_CT_RT_SURVIVAL"    "ERADICATION"
##  [9] "HEMORRHAGE"             "IMMEDIATE_SURVIVAL"
## [11] "HELICOBACTER_PYLORI"    "FIVE_YEAR_RESULT"
```

3.3 6: learn bayesian network from breast cancer daataset with search and score or constraint based algorithm

see section 2. This question seems redundant since the net has already been trained in question 2 where it had to be compared in order of size of data set. The net from question 2 will also be used in subsequent sections.

3.4 7: develop special purpose BN (NBN or TAN, see .2)

```
netnb = naive.bayes(a,"BC")
plot(netnb, main = c("NBN BC"))
```




```
netnb2 <- bn.fit(netnb,a)
```

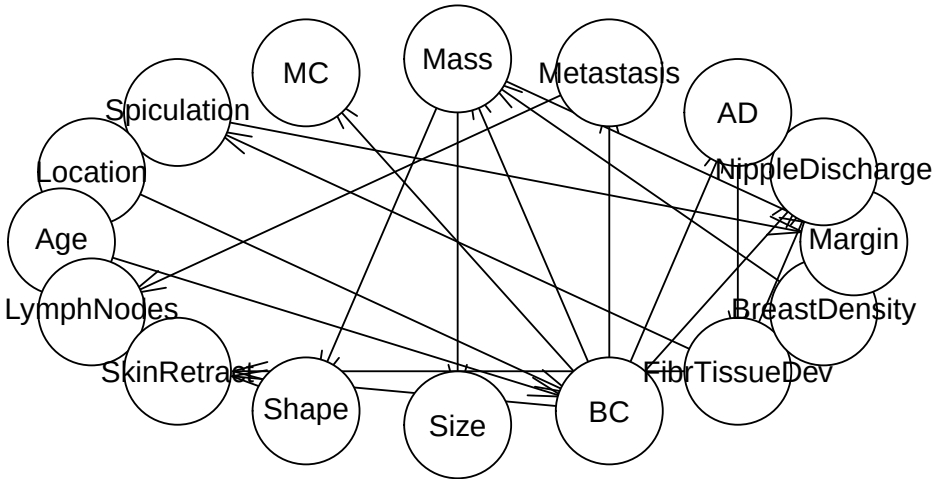
3.5 8:K compare BN and SPBN with manually constructed network in terms of network structure, goodnes of fit scores (lkikelohood,...) and accuracy measures such as misclassification error and ROC (B.2)

Compare the Bayesian network obtained by step (6) and the special purpose Bayesian network from (7) with the manually constructed network in Figure 2 in terms of network structure, goodness of fit scores (e.g., likelihood) and accuracy using measures such as misclassification error and area under the ROC curve from cross-validation

3.5.1 Read in manually constructed network:

```
manual = read.net("bc.net")
```

```
test <- bn.net(manual)
plot(test)
```




```
manual <- bn.fit(test,BC)
```

3.5.2 prediction error/cross validation

```
nbncv <- bn.cv(BC,netnb,loss="pred",loss.args = list(target="BC"),k=3)
```

```
## Warning in split.default(sample(n), seq_len(k)): Datenlänge ist kein
## Vielfaches der Split-Variablen
```

```
nbncv
```

```
##
##   k-fold cross-validation for Bayesian networks
##
##   target network structure:
##    [Naive Bayes Classifier]
##   number of subsets:                  3
##   loss function:                      Classification Error
##   training node:                      BC
##   expected loss:                      0.138
```

```
bncv <- bn.cv(BC,BC_sbl,loss="pred",loss.args = list(target="BC"),k=3)
```

```
## Warning in split.default(sample(n), seq_len(k)): Datenlänge ist kein
## Vielfaches der Split-Variablen
```

```
bncv
```

```
##
##   k-fold cross-validation for Bayesian networks
##
##   target network structure:
##    [BreastDensity][Age][BC|Age][Location|BC][Mass|BreastDensity:BC][AD|BC]
##    [Metastasis|BC][MC|BC][Size|Mass][Shape|Mass][FibrTissueDev|AD]
##    [LymphNodes|Metastasis][SkinRetract|BC:FibrTissueDev]
##    [NippleDischarge|BC:FibrTissueDev][Spiculation|FibrTissueDev]
##    [Margin|Mass:Spiculation]
##   number of subsets:                  3
##   loss function:                      Classification Error
##   training node:                      BC
##   expected loss:                      0.385
```

```
Mncv <- bn.cv(BC,test,loss="pred",loss.args = list(target="BC"),k=3)
```

```
## Warning in split.default(sample(n), seq_len(k)): Datenlänge ist kein
## Vielfaches der Split-Variablen
```

```
Mncv
```

```
##
##   k-fold cross-validation for Bayesian networks
##
##   target network structure:
##    [Location][Age][BreastDensity][BC|Location:Age][MC|BC][AD|BC]
##    [Metastasis|BC][Mass|BC:BreastDensity][LymphNodes|Metastasis]
##    [Shape|Mass][Size|Mass][FibrTissueDev|AD][Spiculation|FibrTissueDev]
##    [SkinRetract|BC:FibrTissueDev][NippleDischarge|BC:FibrTissueDev]
##    [Margin|Spiculation:Mass]
##   number of subsets:                  3
##   loss function:                      Classification Error
##   training node:                      BC
##   expected loss:                      0.385
```

As clearly visible, the NB network has the lowest prediction loss, whilöe the manually and algorithmically created networks have similar values around ~3.85 (probably light differences due to rounding issues)

3.5.3 ROC

ROC fpr the BN

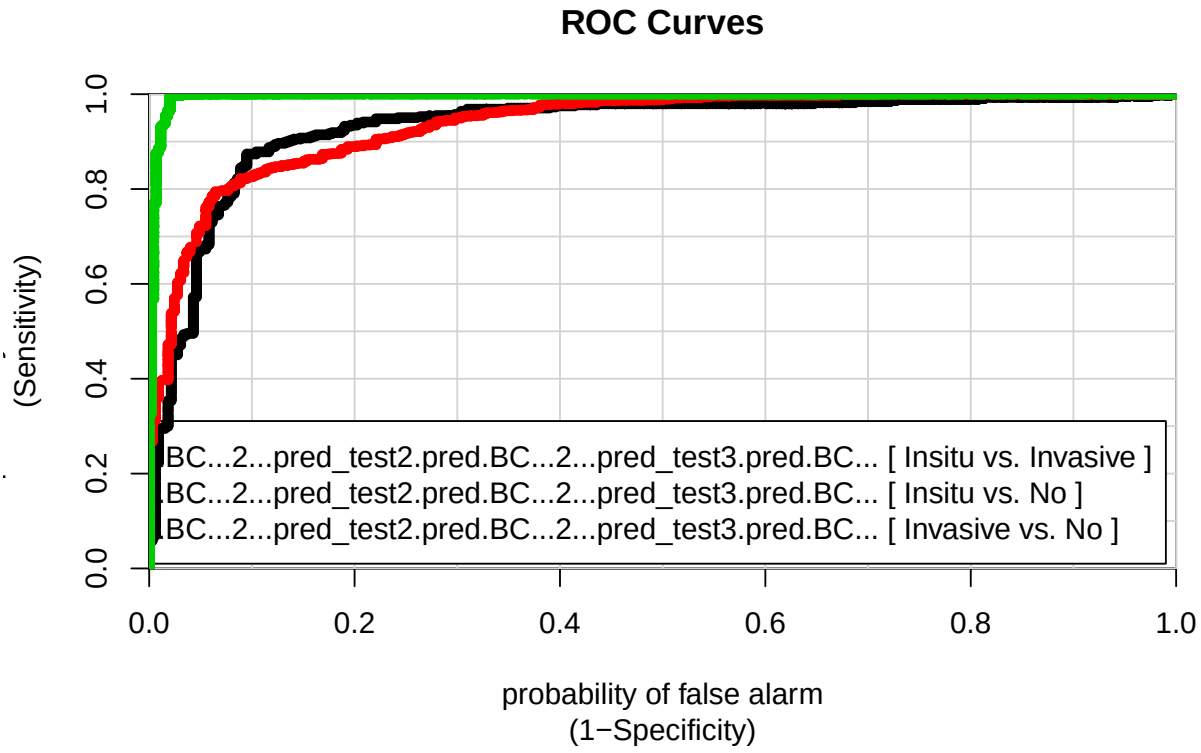
```
netcvfit1 = as.grain(bncv[[1]]$fitted)
netcvfit2 = as.grain(bncv[[2]]$fitted)
netcvfit3 = as.grain(bncv[[3]]$fitted)
#head(netcvfit1)

nbntest1 = BC[bncv[[1]]$test, ]
nbntest2 = BC[bncv[[2]]$test, ]
nbntest3 = BC[bncv[[3]]$test, ]
#head(nbntest1)


pred_test1 = predict(netcvfit1, response = c("BC"), newdata = nbntest1,
predictors = names(nbntest1)[-4], type = "distribution")
pred_test2 = predict(netcvfit2, response = c("BC"), newdata = nbntest2,
predictors = names(nbntest2)[-4], type = "distribution")
pred_test3 = predict(netcvfit3, response = c("BC"), newdata = nbntest3,
predictors = names(nbntest3)[-4], type = "distribution")

nbntest = rbind(nbntest1,nbntest2,nbntest3)

#head(nbntest)



pred_test = data.frame(c(pred_test1$pred$BC[ ,2], pred_test2$pred$BC[ ,2],
pred_test3$pred$BC[ ,2]))

#head(pred_test)
#names(pred_test)


colAUC(pred_test, nbntest[ ,4], plotROC = TRUE)
```

```
##                     c.pred_test1.pred.BC...2...pred_test2.pred.BC...2...pred_test3.pred.BC...
## Insitu vs. Invasive                                                                 0.9322321
## Insitu vs. No                                                                       0.9377949
## Invasive vs. No                                                                     0.9959340
```

```
#title("test")
```

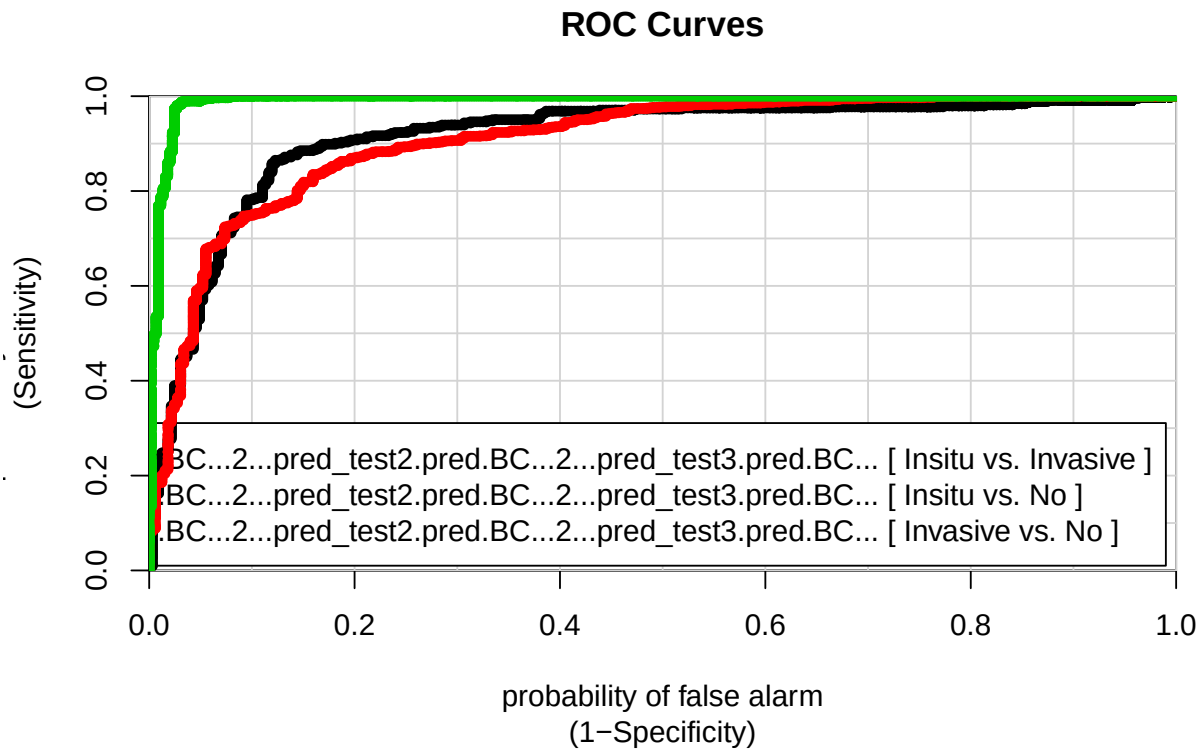
ROC for the NBN

```
netcvfit1 = as.grain(nbncv[[1]]$fitted)
netcvfit2 = as.grain(nbncv[[2]]$fitted)
netcvfit3 = as.grain(nbncv[[3]]$fitted)
#head(netcvfit1)

nbntest1 = BC[nbncv[[1]]$test, ]
nbntest2 = BC[nbncv[[2]]$test, ]
nbntest3 = BC[nbncv[[3]]$test, ]
#head(nbntest1)
```




```
pred_test1 = predict(netcvfit1, response = c("BC"), newdata = nbntest1,
predictors = names(nbntest1)[-4], type = "distribution")
pred_test2 = predict(netcvfit2, response = c("BC"), newdata = nbntest2,
predictors = names(nbntest2)[-4], type = "distribution")
pred_test3 = predict(netcvfit3, response = c("BC"), newdata = nbntest3,
predictors = names(nbntest3)[-4], type = "distribution")

nbntest = rbind(nbntest1,nbntest2,nbntest3)

#head(nbntest)



pred_test = data.frame(c(pred_test1$pred$BC[ ,2], pred_test2$pred$BC[ ,2],
pred_test3$pred$BC[ ,2]))

#head(pred_test)
#names(pred_test)


colAUC(pred_test, nbntest[ ,4], plotROC = TRUE)
```

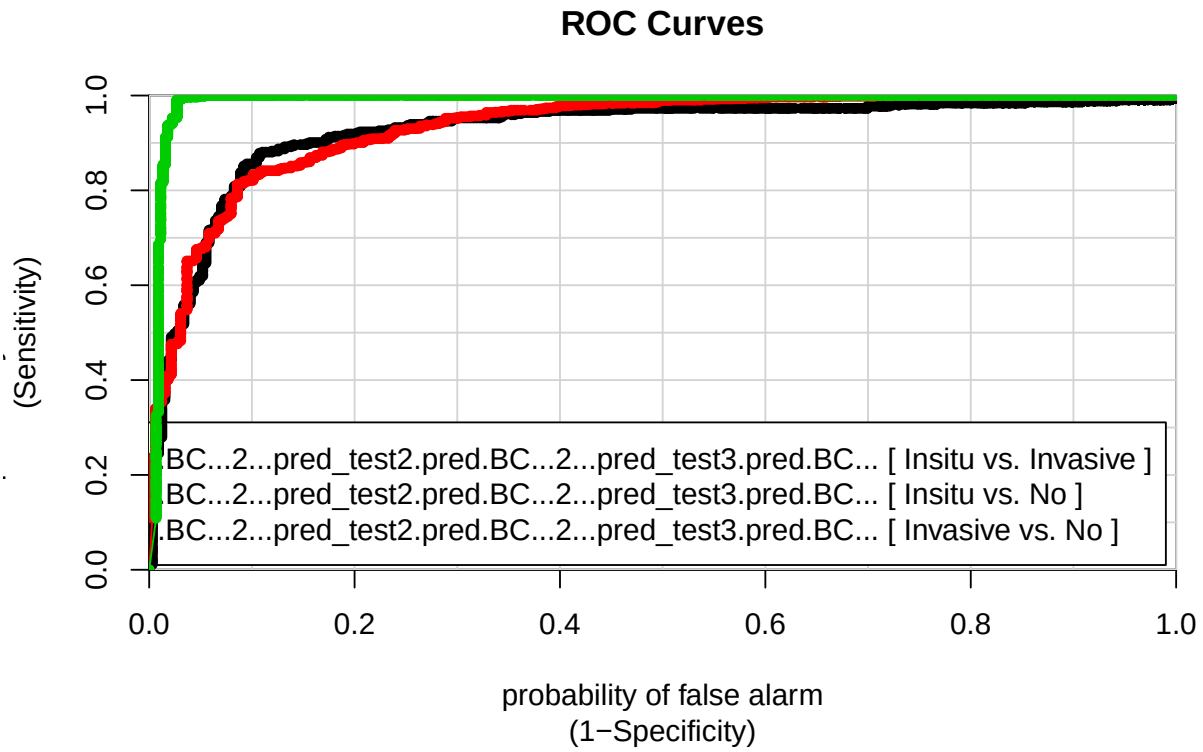
```
##                     c.pred_test1.pred.BC...2...pred_test2.pred.BC...2...pred_test3.pred.BC...
## Insitu vs. Invasive                                                                 0.9139066
## Insitu vs. No                                                                       0.9077385
## Invasive vs. No                                                                     0.9916419
```



ROC fpor the manually created network

```
netcvfit1 = as.grain(Mncv[[1]]$fitted)
netcvfit2 = as.grain(Mncv[[2]]$fitted)
netcvfit3 = as.grain(Mncv[[3]]$fitted)
#head(netcvfit1)

nbntest1 = BC[Mncv[[1]]$test, ]
nbntest2 = BC[Mncv[[2]]$test, ]
nbntest3 = BC[Mncv[[3]]$test, ]
#head(nbntest1)


pred_test1 = predict(netcvfit1, response = c("BC"), newdata = nbntest1,
predictors = names(nbntest1)[-4], type = "distribution")
pred_test2 = predict(netcvfit2, response = c("BC"), newdata = nbntest2,
predictors = names(nbntest2)[-4], type = "distribution")
pred_test3 = predict(netcvfit3, response = c("BC"), newdata = nbntest3,
predictors = names(nbntest3)[-4], type = "distribution")

nbntest = rbind(nbntest1,nbntest2,nbntest3)

#head(nbntest)



pred_test = data.frame(c(pred_test1$pred$BC[ ,2], pred_test2$pred$BC[ ,2],
```

```
pred_test3$pred$BC[ ,2]))

#head(pred_test)
#names(pred_test)


colAUC(pred_test, nbntest[ ,4], plotROC = TRUE)
```




```
##                     c.pred_test1.pred.BC...2...pred_test2.pred.BC...2...pred_test3.pred.BC...
## Insitu vs. Invasive                                                                 0.9261455
## Insitu vs. No                                                                       0.9333495
## Invasive vs. No                                                                     0.9898786
```

All the networks seem to perform pretty good, with scores around or above .9. Best performance happens between invasive vs no, which seems to match intuition since they are the very different.Best performance overall seems to be held by the automatically created network with .93, .937 and .995, while nbn and manally created network seem to have different strengths, with nbn performing slightly better in invasive vs no and the manually created slightly better in the others.

[1]

[1] Tobias Sing, Oliver Sander, Niko Beerenwinkel, Thomas Lengauer. ROCR: visualizing classifier performance in R. Bioinformatics 21(20):3940-3941 (2005).

Sing et al. (2005)

4 list of figures

List of Figures

5 references

Sing, Tobias, Oliver Sander, Niko Beerenwinkel, and Thomas Lengauer. 2005. "ROCR: visualizing Classifier Performance in R." *Bioinformatics (Oxford, England)* 21 (20): 3940–1. doi:10.1093/bioinformatics/bti623. http://bioinformatics.oxfordjournals.org/content/21/20/3940.abstract.