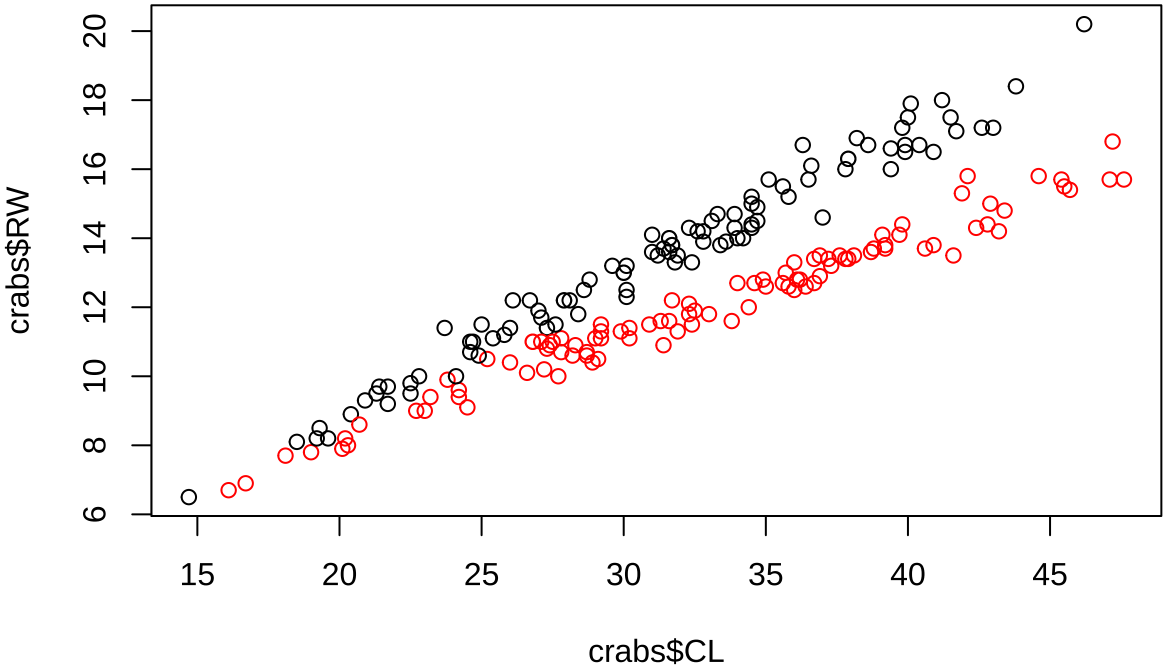
Lab2

Andreas C Charitos

3 Dec 2018

Contents

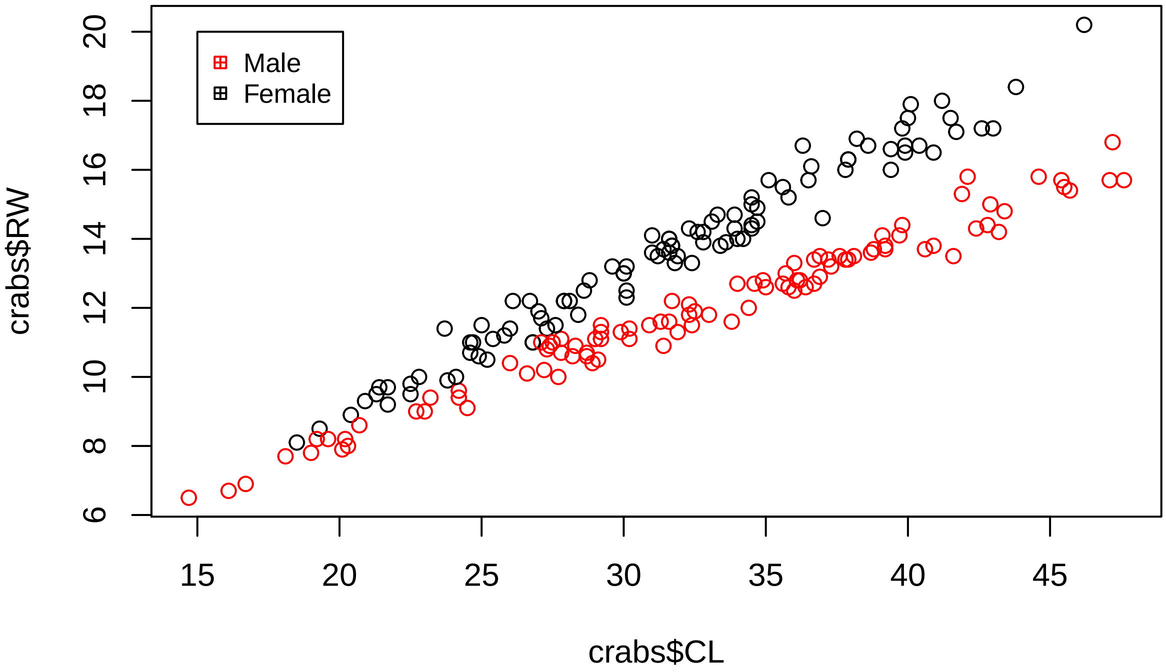
Assignment 1-LDA



2

```
##
## Attaching package: 'MASS'
```

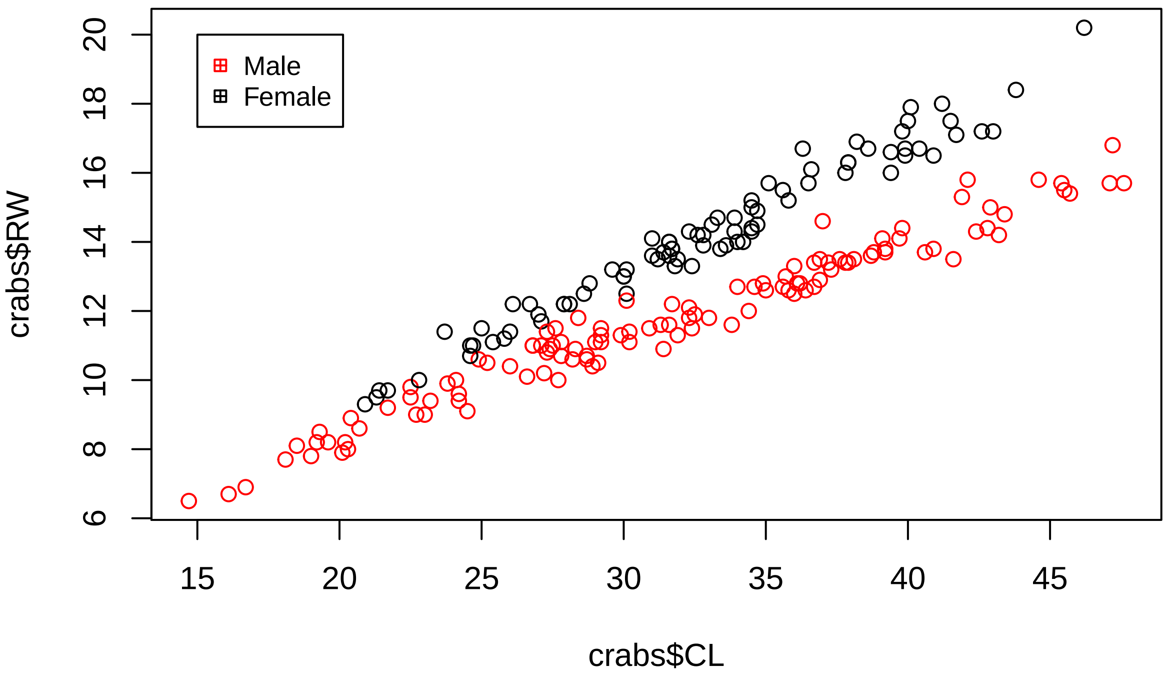
```
## The following object is masked _by_ '.GlobalEnv':
##
##     crabs
```


Male
Female
crabs$RW
crabs$CL
6
8
10
12
14
16
18
20
15
20
25
30
35
40
45

```
## [1] 0.035
```

```
## [1] 0.965
```

3



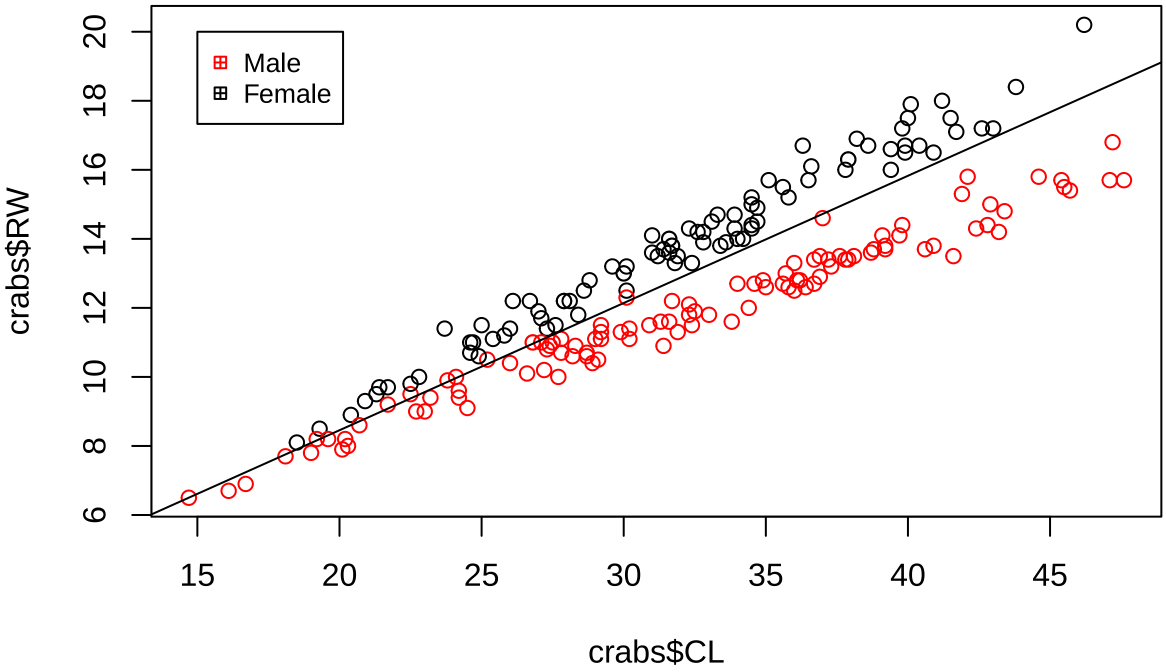
```
## [1] 0.08
```

```
## [1] 0.92
```

4

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

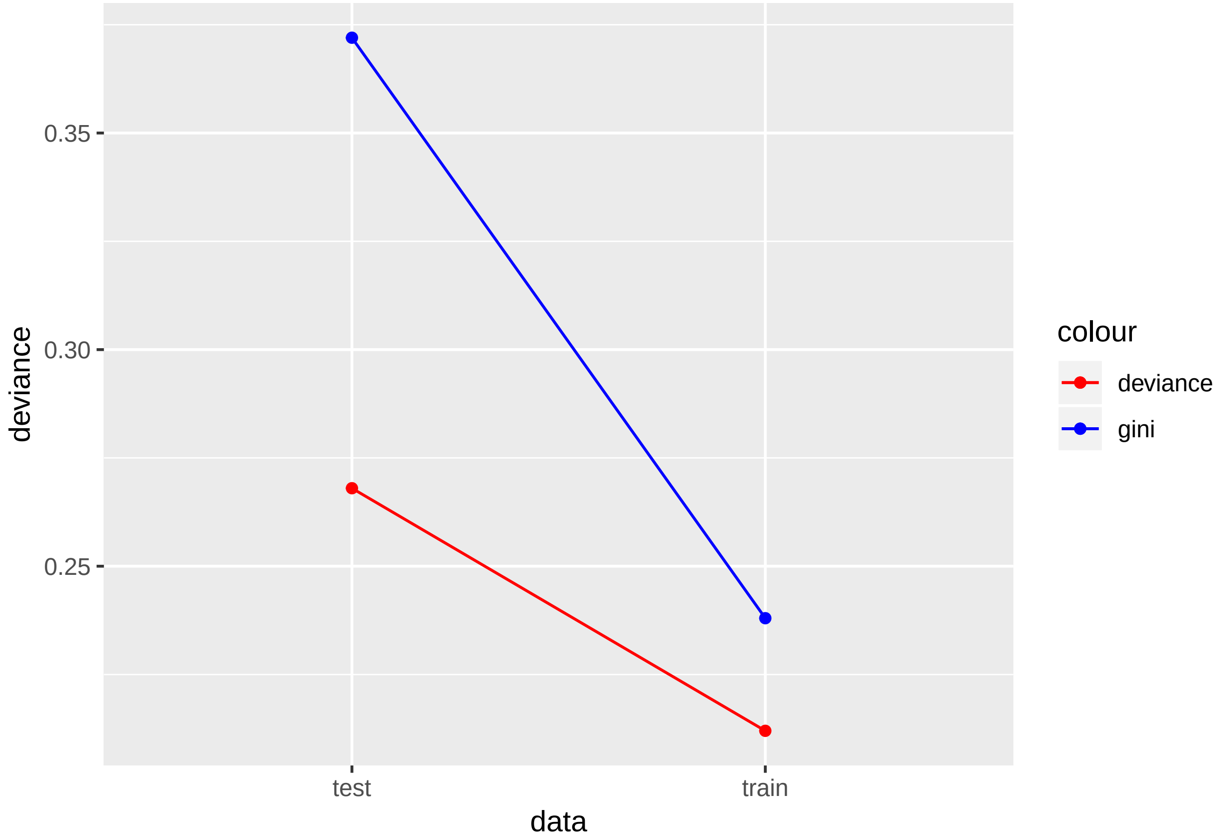
```
## [1] 0.04
```


Male
Female
crabs$RW
6
8
10
12
14
16
18
20
15
20
25
30
35
40
45
crabs$CL

Assignment 2-Analysis of credit scoring

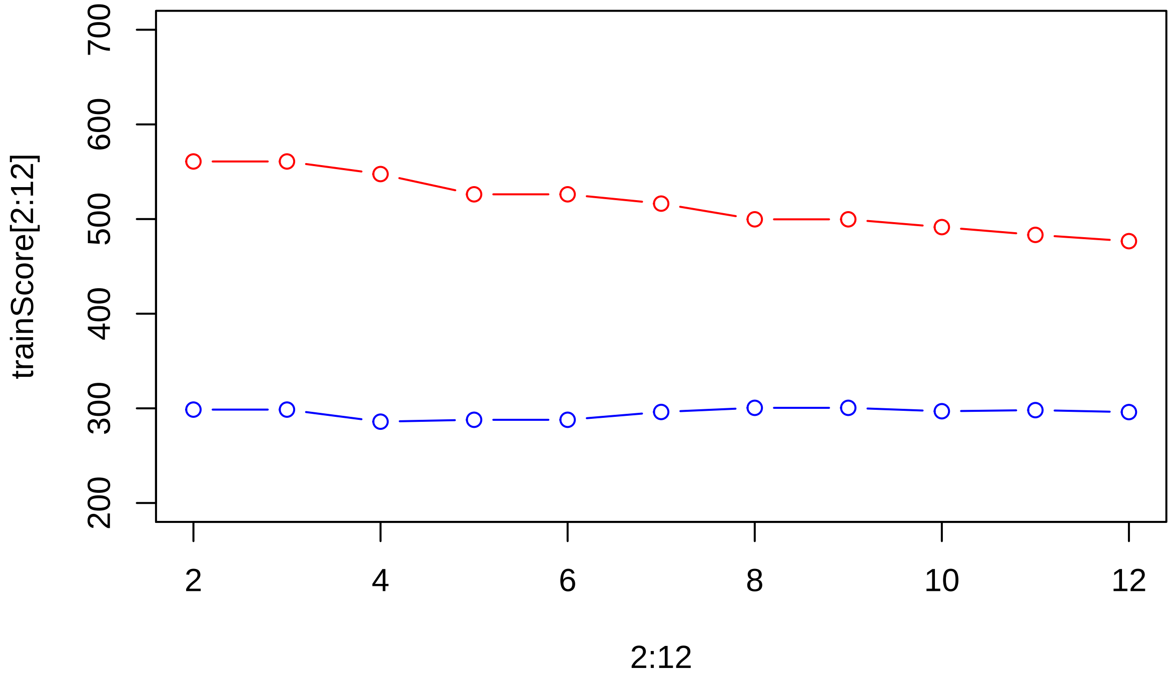
1

2



We can see from the plot that using deviance as impurity measure the misclassification errors for both train and test are lower compaired with the misclassification errors for gini index.We can conclude that using deviance as impurity measure we fit a better model tree.

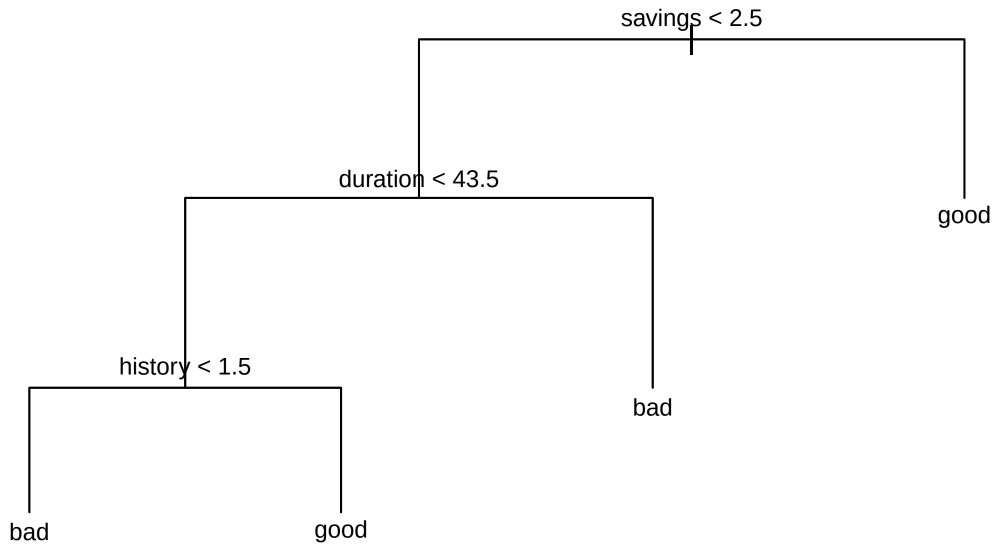
3



The above plot shows the deviance vs the number of trees for the train data(line red) and validation data(blue). As we can see the optimal tree depth is 4 in which we have the lowest validation error compared with the other depths.

```
##
## Classification tree:
## snip.tree(tree = fit, nodes = c(5L, 3L, 9L))
## Variables actually used in tree construction:
## [1] "savings"  "duration" "history"
## Number of terminal nodes:  4
## Residual mean deviance:  1.117 = 547.5 / 490
## Misclassification error rate: 0.251 = 124 / 494
```

Using summary we obtain information for the best tree.We can see that the variables used for the best tree are (“savings” ,“duration” ,“history”).



Plotting the best tree and starting from first node savings,when savings are more than 2.5 then we classify the customer as good else we move to next node. Moving to next node which is duration ,when someone is having savings less that 2.5 and the duration is greater that 43.5 we classify them as bad or else we move to final node history.At the final mode history someone having savings less than 2.5,duration less than 43.5 and history less than 1.5 is classified as bad and good if history is over 1.5.

```
## ===========================================================================================
##  The misclassification error for train data is:  0.252 and for test data is:  0.256
```

4

The confusion matrices for the naive Bayes for the train and test data are :

```
## ================= Confusion matrix for naive train =====================================
```

```
##                   predictions train naive
## actual train naive bad good
##               bad   95   52
##               good  98  255
```

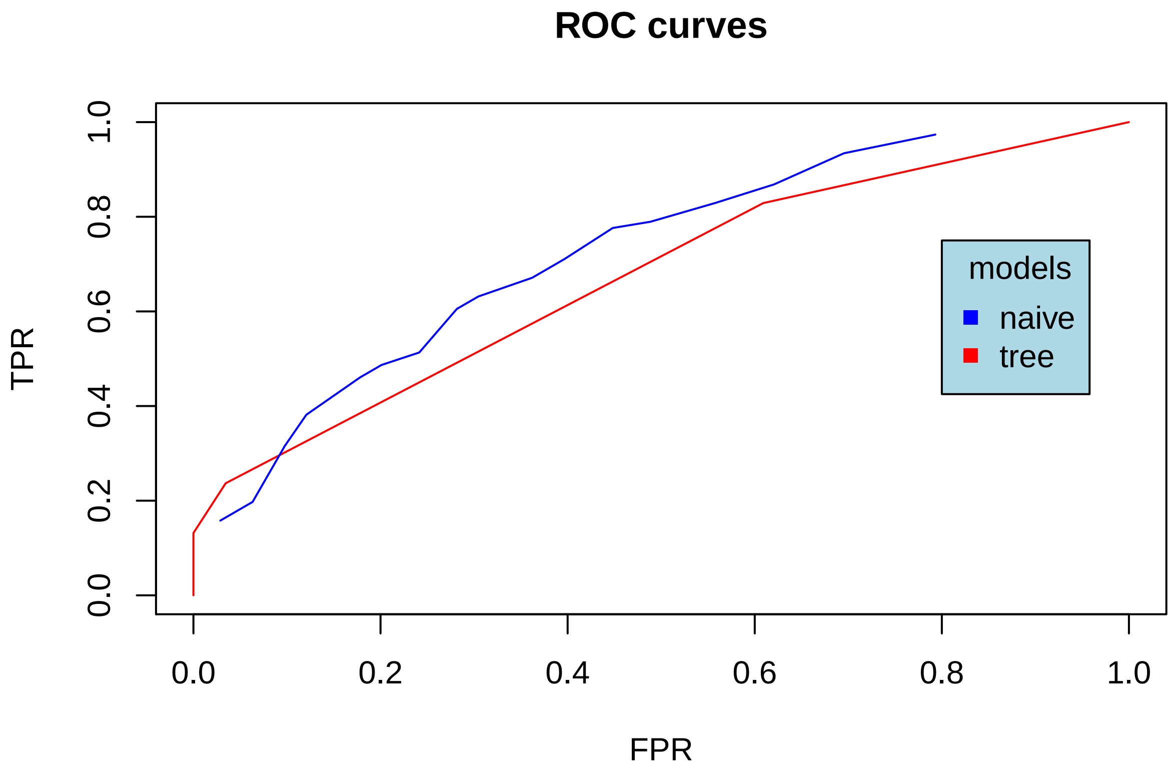
```
## ================= Confusion matrix for naive test ======================================
```

```
##                  predictions test naive
## actual test naive bad good
##              bad   46   30
##              good  49  125
```

```
## ===================================================================================
##  The misclassification error for train naive is : 0.3 and for test naive is : 0.316
```

Comparing the misclassification errors in naive Bayes model and the tree from previous step we can see that the naive Bayes model has bigger errors both in training and test data and the tree model from step 3 provides a better model.

5



The ROC curves plotted in the above plot for the 2 models we observe that the naive Bayes model has a bigger ROC curve from the tree model. We may conclude that using diffrent thesholds the naive bayes model achives a bigger ROC curve from tree model.

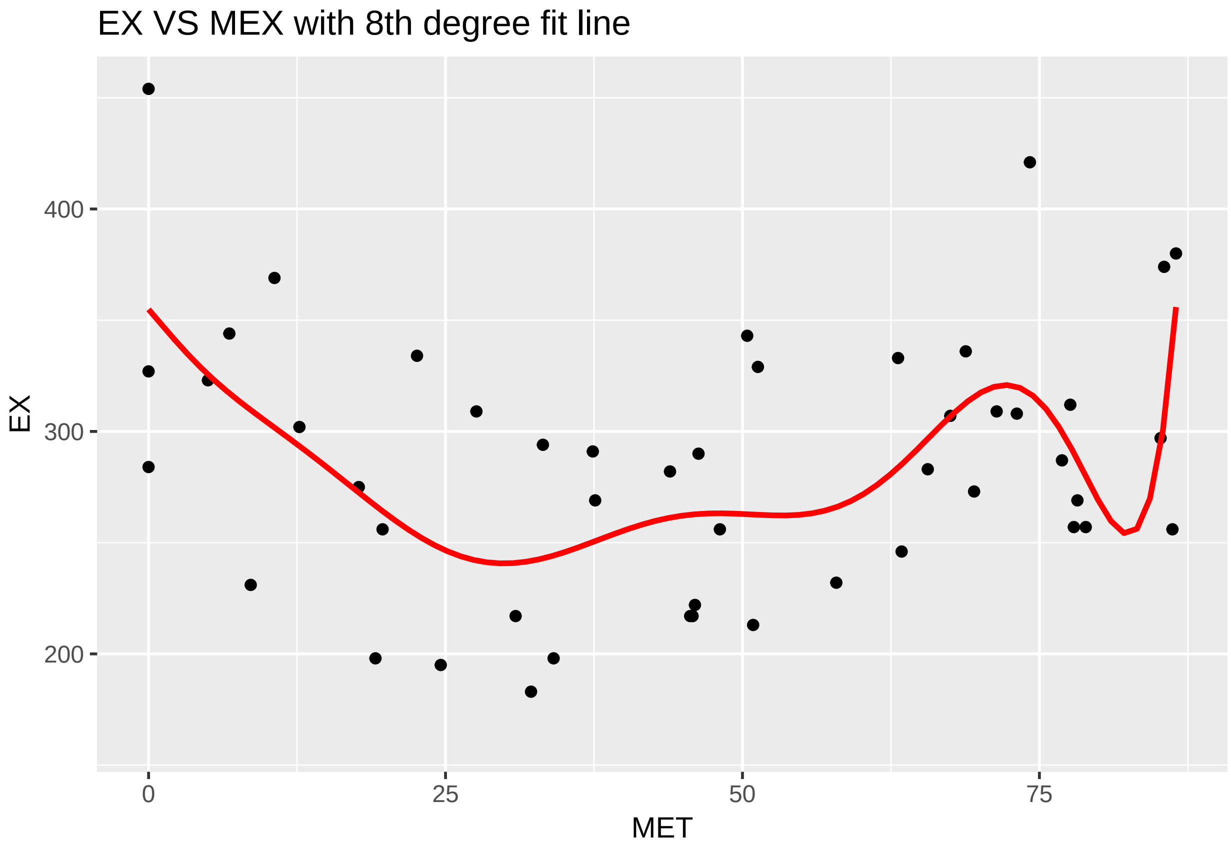
6

```
## ================= Confusion matrix for naive train =====================================

##       predictions
## actual bad good
##   bad  137   10
##   good 263   90

## ================= Confusion matrix for naive test ======================================

##       predictions
## actual bad good
##   bad   71    5
##   good 122   52

## ==========================================================================================
##  The misclassification error for train is : 0.546 and for test is : 0.508
```

We can see that applying the confusion matrix $\mathbf{L} = \begin{bmatrix} 0,1 \\ 100,0 \end{bmatrix}$ we put more weight on bad predicted good by applying a bigger loss of 10.Thus as we can see from the confusion matrices these leads to increasing the number of good that as predicted bad.Also from the misclassification errors we can see that both of the have increased using the loss matrix.

Assignment 3 - Uncertainty estimation

1



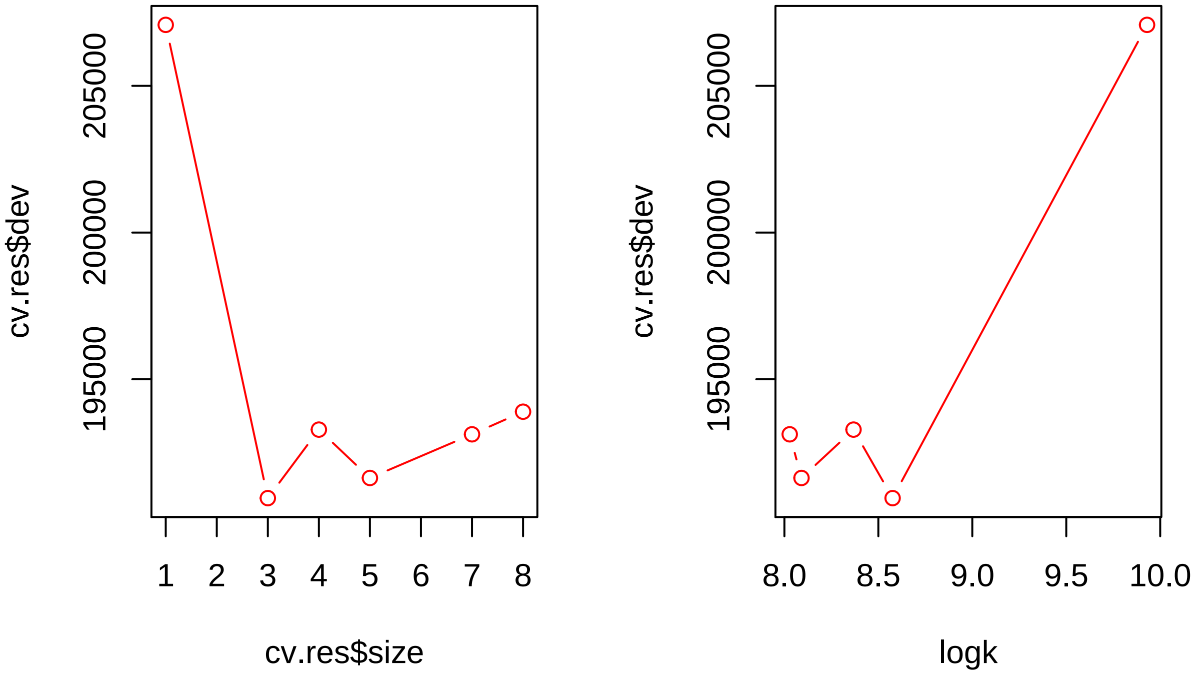
From the scatter plot we can distinguish that there's is no linear relationship in the points if we use a higher level of polynomial we might have to use a high power of polynomial that may lead to over fitting in the plot we see a 8th degree polynomial.Even with a 8th polynomial is hard to map the data into a model.A tree model would be more appropriate because it makes no assumption of linearity and might be a good fit.

2

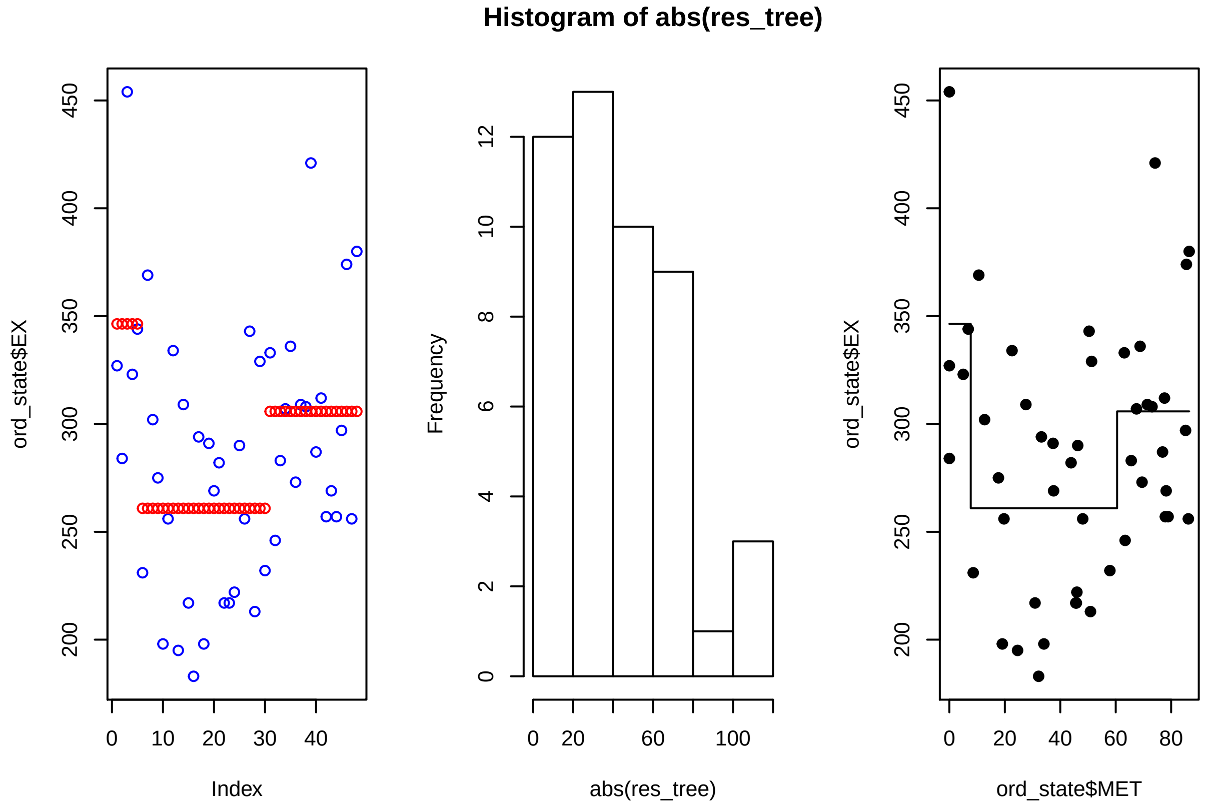
```
## The optimal tree leaves are: 3
```

The left plot is the dev vs the size and the right plot is the dev vs the log of k

```
## Warning in log(cv.res$k): NaNs produced
```



From the left plot where we plot the dev vs the size we confirm that the best tree in the one with 3 leaves.



The distribution of the residuals is quite skewed on the right and don't follow normal distribution maybe follows chi-square.Thus we can conclude that tree model doesn't make such a good fit on the data maybe maybe we need to apply a transformation on the data.

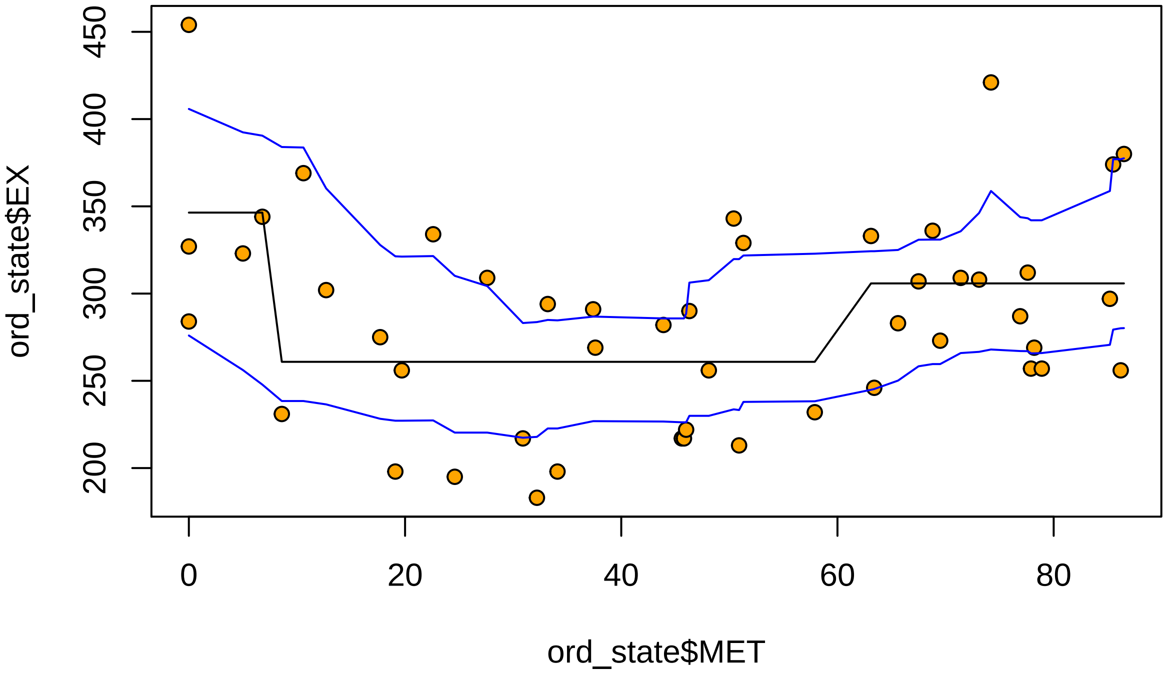
The summary of the model is :

```
summary(btree)
```

```
##
## Regression tree:
## snip.tree(tree = modtree, nodes = 7:6)
## Number of terminal nodes:  3
## Residual mean deviance:  2698 = 121400 / 45
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -77.88  -43.88   -4.88    0.00   30.13  115.20
```

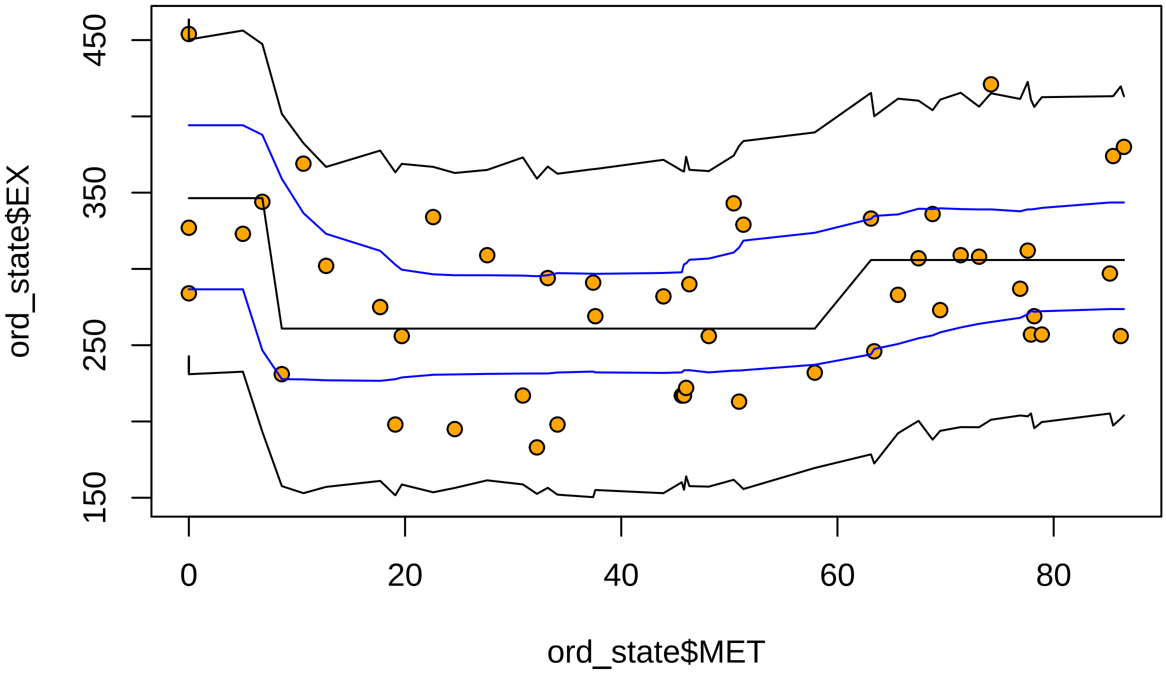
3

```
## Warning in prune.tree(mod, best = 3): best is bigger than tree size
```



The non parametric bootstrap makes no assumption of a distribution of the data thus every time we get a new sample which means a new distribution from the data and make predictions on that sample and calculate confidence interval every time.In the later step we combine all these confidence intervals in one line of confidence intervals and since the interval looks very bumpy. Considering the width of the confidence interval we can say that the results of step 2 seems to be reliable and the 95% confidence band has a 95% chance of containing the tree line we fit in step2.

```
## Warning in envelope(p_boot2): unable to achieve requested overall error
## rate
```



Using parametric bootstrap we can see that the confidence bands(blue lines) are more smother that the ones in non parametric the width seems to be the same and again the results of step 2 seems to be reliable with the tree we fitted. Considering the prediction bands(black lines) are wider than confidence bands because the 95% of all data points to fall showing the variation of the data.It seems less than 5% of the data are outside the predicted bands.

6

From the histogram of the residuals we say that the non parametric method might seems more appropriate because of normality assumption that is violated and it is better to make predictions using a new sample every time.

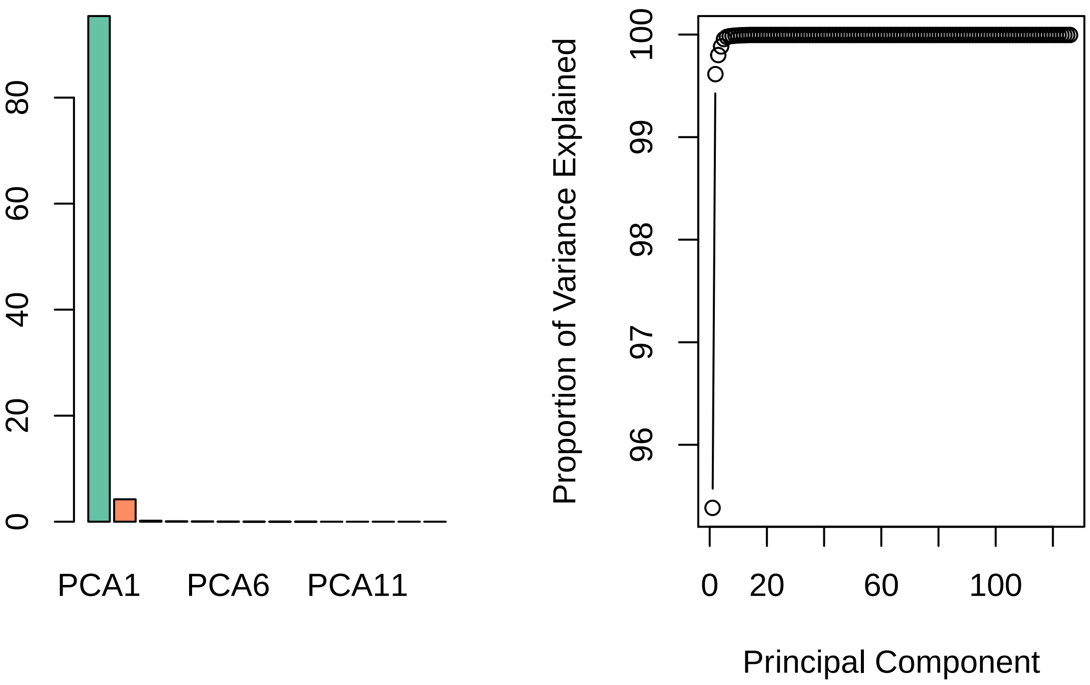
Assignment4-Principal Components

1

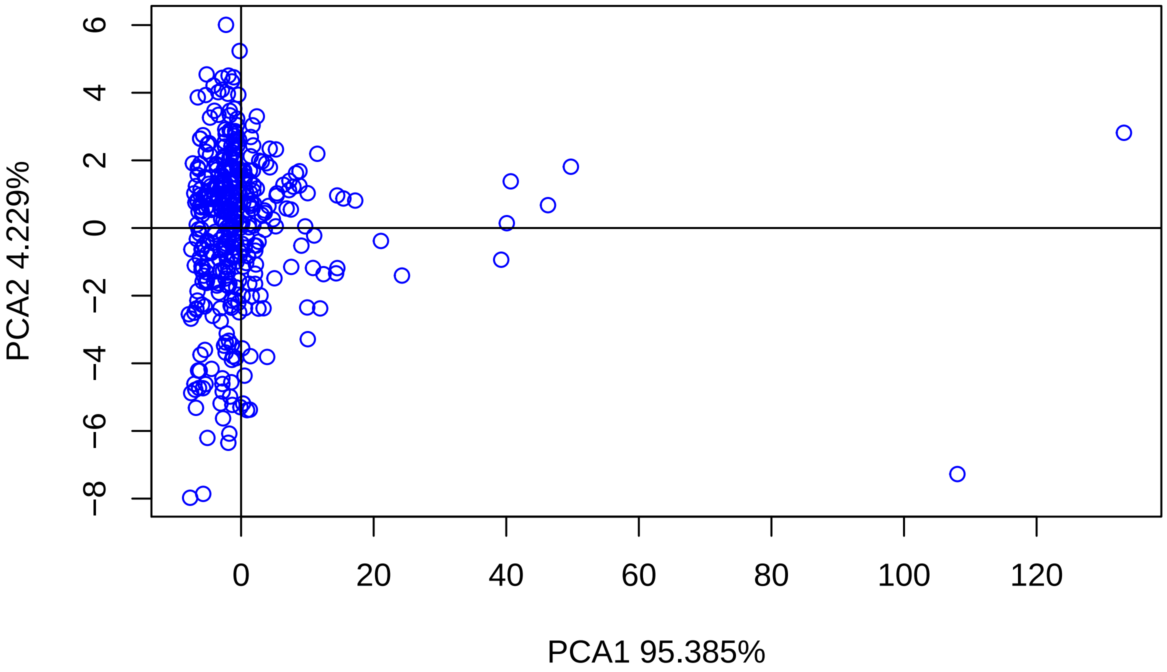
```
## [1] "The percentage of variance for feature space "
```

```
##   [1] 95.385  4.229  0.186  0.084  0.072  0.021  0.007  0.003  0.003  0.002
##  [11]  0.001  0.001  0.001  0.001  0.000  0.000  0.000  0.000  0.000  0.000
##  [21]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
##  [31]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
##  [41]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
##  [51]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
##  [61]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
##  [71]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
##  [81]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
##  [91]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
## [101]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
## [111]  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
## [121]  0.000  0.000  0.000  0.000  0.000  0.000
```

As we can see from the percentages of variance for all the features only 14 features provide information about the variance of the data.

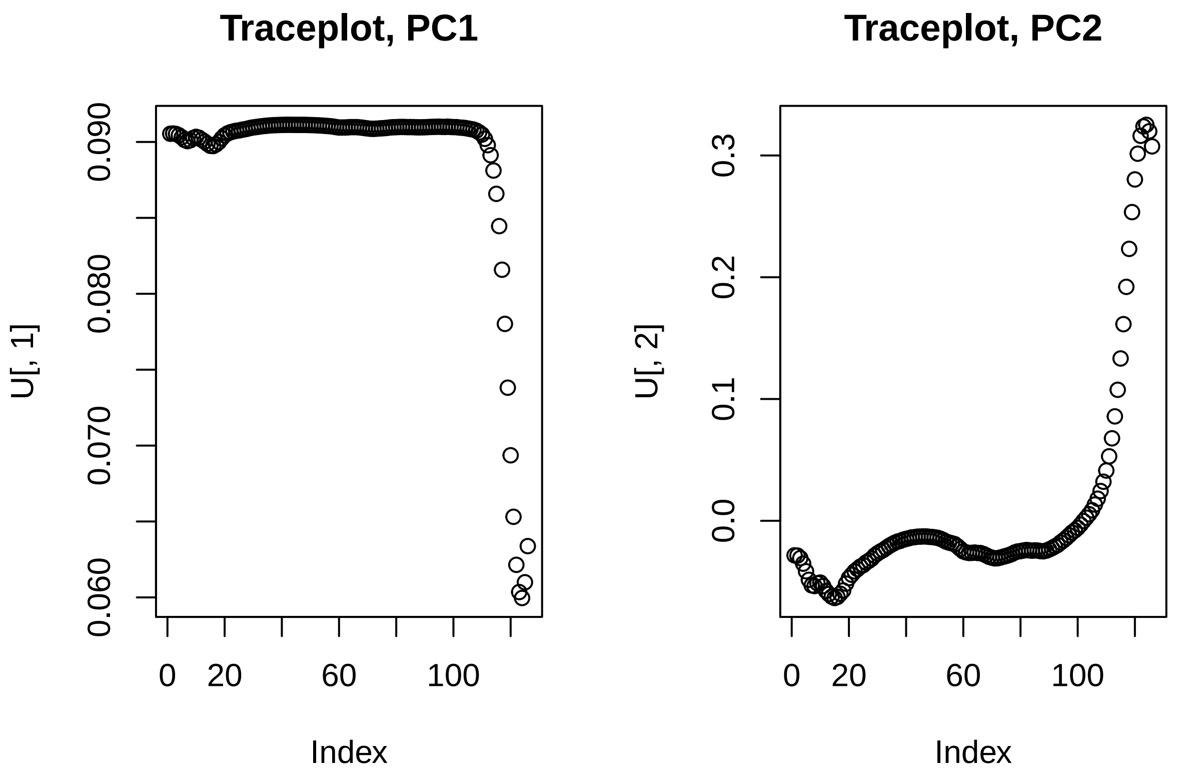


The left plot shows the percentages of variance for the 14 first components that describe the total variance of the data.The first principal component-PC1 describes the 95.83% of the variance of the data following the second principal component-PC2 with 4.23%.The rest of the RCA's have a very small percentage lower than 1%.We can see that only 2 principal components explain at least 99% of the total variance PC1 and PC2.The right plot shows the cumulative proportion of variance explained and we can see again that two component provide the 99% of the variance of the data and the contribution of only few improves the variance the rest components have no adding impact.



The above plot confirms the results from the previous plot that the first PC explains the 95.38% of the total variation in the data.The spred of the data in the first principal component is very small.We can also see from the plot that we have some points with unusual diesel fuels (outliers) and have a high variance even the PCaA1 was not able to capture.

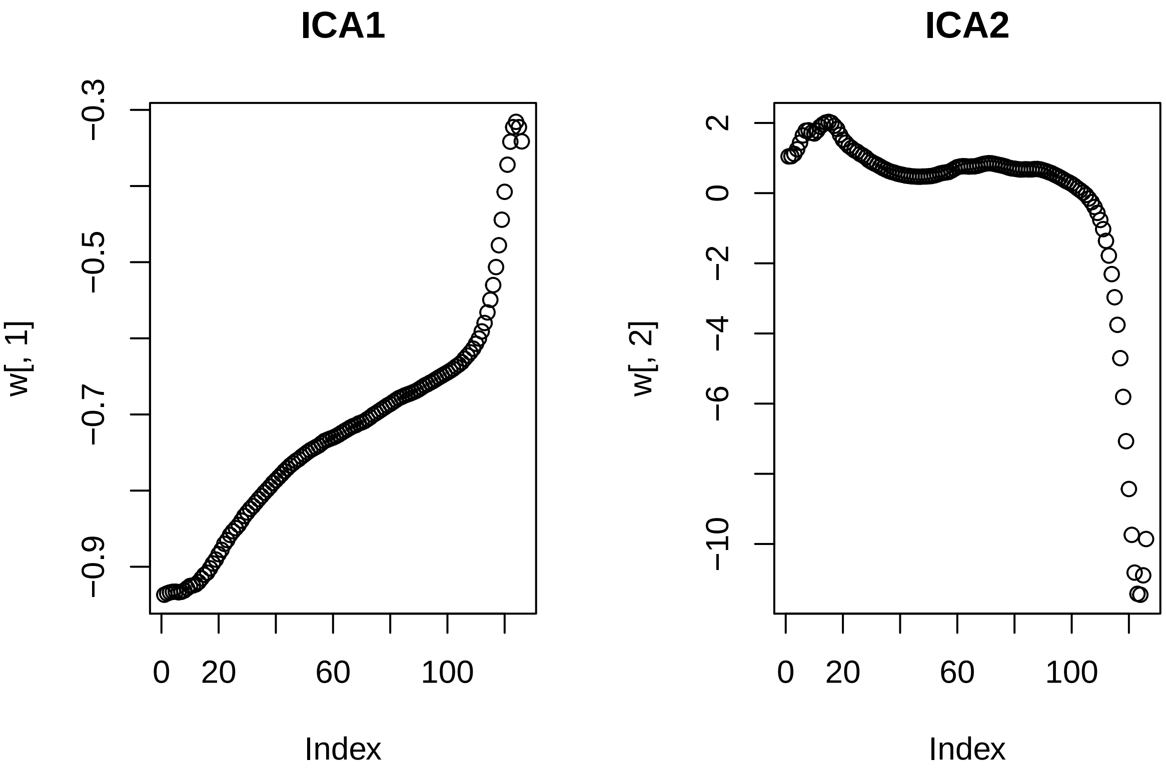
2



The above trace plots provide information about the 2 first principal components .As we can see approximately 100 features explaining the first PC1 having a value close to 0.0990.On the other hand the principle component 2 is explained by mainly by very few original features approximately 20 that have a possitive value and the rest are below 0.

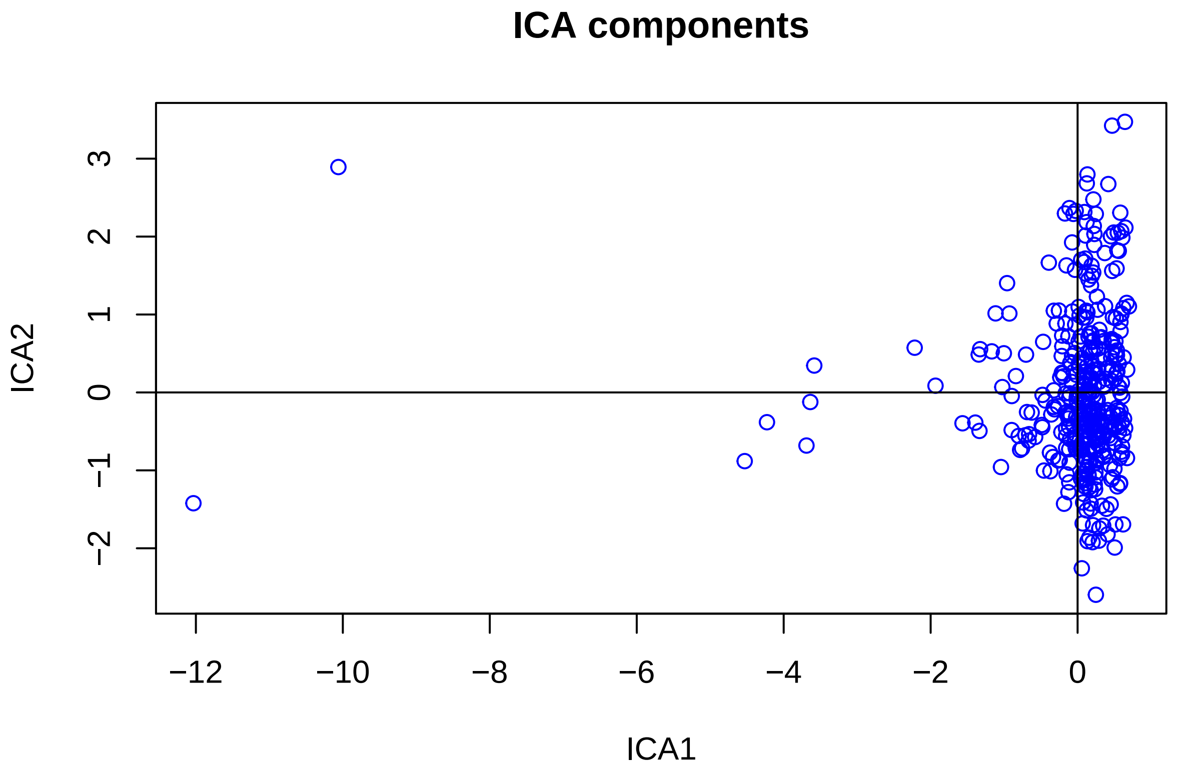
3

a



The above trace plots provide information about the 2 independent components.As we can see the independent component 2 is explained by approximately 100 features having positive value.On the other hand the independent component 1 has negative values for all the features. Comparing with trace plots from pca we can conclude that as in pca only one principal component that explains 95% of the variation of the data and on the other hand only one independent component provides the best separation of the data making them as independent (and non-Gaussian)as possible. The ICA algorithm estimates W tries to make components as uncorrelated as possible giving the constraint that W is orthogonal. The W' matrix is providing a measure of independence for the independent components.

b



The score plot for the 2 latent features of ica is very similar to the one obtained with pca.We can see that in the ICA1 the data are very close together in that axis and only few points are very far as we notice also in PCA.And regarding IC2 the data have similar spread as the ones in PC2.

Apdendix

```
crabs<-read.csv("australian-crabs.csv")

plot(crabs$CL,crabs$RW,col=crabs$sex,pch=21)

library(MASS)
lda.fit<-lda(sex~CL+RW,data=crabs)

preds.lda<-predict(lda.fit,crabs)


plot(crabs$CL,crabs$RW,col=preds.lda$class)
legend(15, 20, pch=12, col=c("red", "black"), c("Male", "Female"), cex=.8)

mean(crabs$sex!=preds.lda$class)
1-mean(crabs$sex!=preds.lda$class)


lda.fit2<-lda(sex ~ CL+RW, crabs, prior = c(0.1,0.9))

preds.lda2 = predict(lda.fit2, data = crabs, type="class")

plot(crabs$CL,crabs$RW,col=preds.lda2$class)
legend(15, 20, pch=12, col=c("red", "black"), c("Male", "Female"), cex=.8)

mean(crabs$sex!=preds.lda2$class)
1-mean(crabs$sex!=preds.lda2$class)


logistic<-glm(sex~CL+RW,data=crabs,family = binomial("logit"))

log_preds<-predict(logistic,newdata = crabs[,-2],type="response")

log_preds<-ifelse(log_preds>0.1,"Male","Female")

mean(crabs$sex!=log_preds)



slope <- coef(logistic)[2]/(-coef(logistic)[3])
intercept <- coef(logistic)[1]/(-coef(logistic)[3])

plot(crabs$CL,crabs$RW,col=as.factor(log_preds))
abline(intercept , slope)
legend(15, 20, pch=12, col=c("red", "black"), c("Male", "Female"), cex=.8)

# read data and make scoring data frame
scoring<-readxl::read_excel("creditscoring.xls")
scoring<-as.data.frame(scoring)
scoring$good_bad<-as.factor(scoring$good_bad)
# split data to train/valid/test-50/25/25
n=dim(scoring)[1]
set.seed(12345)
```

```
id=sample(1:n, floor(n*0.5))
train=scoring[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=scoring[id2,]
id3=setdiff(id1,id2)
test=scoring[id3,]


library(tree) #import tree

n=dim(train)[1]
train$good_bad<-as.factor(train$good_bad) #make good_bad as factor
#fit trees with deviance,gini impurity
fit_dev<-tree(good_bad~., data=train,split="deviance")
fit_gini<-tree(good_bad~.,data=train,split="gini")
#misclassification error function
mis_error<-function(X,X1){
  n<-length(X)
  return(1-sum(diag(table(X,X1)))/n)

}
#make predictions,calculate miserrors for deviance
pred_dev_train<-predict(fit_dev, newdata=train, type="class")
     #table(train$good_bad,pred_dev_train)

mis.error1<-mis_error(pred_dev_train,train$good_bad)
#mis.error1

pred_dev_test<-predict(fit_dev,newdata = test,type="class")
     #table(test$good_bad,pred_dev_test)

mis.error2<-mis_error(pred_dev_test,test$good_bad)
#mis.error2

#make predictions,calculate miserrors for gini

pred_gini_train<-predict(fit_gini, newdata=train, type="class")
     #table(train$good_bad,pred_gini_train)

mis.error3<-mis_error(pred_gini_train,train$good_bad)


pred_gini_test<-predict(fit_gini,newdata = test,type="class")
     #table(test$good_bad,pred_gini_test)

mis.error4<-mis_error(test$good_bad,pred_gini_test)

#combine the misclassifications errors
df<-cbind(c(mis.error1,mis.error2),c(mis.error3,mis.error4))
df<-as.data.frame(df)
colnames(df)<-c("deviance","gini")
```

```
df$data<-c("train","test")

#plot the data
library(ggplot2)
ggplot(df,aes(x=data,y=deviance,group=1))+geom_point(aes(color="deviance"))+
  geom_point(aes(y=gini,color="gini"))+geom_line(aes(y=gini,color="gini"))+
  geom_line(aes(y=deviance,color="deviance"))+
  scale_color_manual(labels = c("deviance", "gini"), values = c("red", "blue"))


#fit the tree
fit=tree(good_bad~., data=train)
trainScore=rep(0,9)
testScore=rep(0,9)
for(i in 2:12) {
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid,type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}

plot(2:12, trainScore[2:12], type="b", col="red",ylim=c(200,700))
points(2:12, testScore[2:12], type="b", col="blue")                #optimal tree depth is 4

#prune the tree to best depth
finalTree<-prune.tree(fit, best=4)
#summary of the tree
summary(finalTree)

#plot the tree
plot(finalTree)
text(finalTree,cex=.75)


#make predicitons using best tree
pred_best_train<-predict(finalTree,newdata = train,type="class")
pred_best_test<-predict(finalTree, newdata=test,type="class")
#table(test$good_bad,pred_best)
mis.error_b1<-mis_error(train$good_bad,pred_best_train)
mis.error_b2<-mis_error(test$good_bad,pred_best_test)
#misclassification for test,train data with the best tree with leaves
cat("==========================================================================================\n",
    "The misclassification error for train data is: ",mis.error_b1,"and for test data is: ",mis.error_b

library(MASS)
library(e1071)
#make naive bayes model
naive=naiveBayes(good_bad~., data=train)

cat("================= Confusion matrix for naive train ====================================\n")
pred_naive_train<-predict(naive,newdata=train)#predictions train naive
table(train$good_bad,pred_naive_train,dnn = c("actual train naive","predictions train naive"))
cat("================= Confusion matrix for naive test =====================================\n")
```

```r
pred_naive_test<-predict(naive, newdata=test)#predicitons test naive
table(test$good_bad,pred_naive_test,dnn=c("actual test naive","predictions test naive"))

#misclassification train 0.30 misclassification 0.32

#misclassification error for train naive
enaive_train<-mis_error(train$good_bad,pred_naive_train)
#misclassification error for test naive
enaive_test<-mis_error(test$good_bad,pred_naive_test)

cat("=============================================================================\n",
    "The misclassification error for train naive is :",enaive_train,"and for test naive is :" ,enaive_te

#misclassification rates for naive bayes are more than the ones from tree model 3


l<-seq(0.05,0.95,0.05)
#extracting the probs from predict naive test and take the good
p_naive<-as.data.frame(predict(naive,test,type="raw"))$good
mat<-vector()
for (i in l){
  pred_naive<-ifelse(p_naive>i,"good","bad")
  t<-table(test$good_bad,pred_naive)
  tpr<-t[1]/(t[1]+t[3])
  fpr<-t[2]/(t[2]+t[4])
  mat<-rbind(mat,c(tpr,fpr))

}

dmat<-as.data.frame(mat)
colnames(dmat)<-c("tpr","fpr")



#extracting the probs of predict tree test and take the good
p_tree<-as.data.frame(predict(finalTree,test,type="vector"))$good
vec<-vector()
for (i in l){
  pred_tree<-ifelse(p_tree>i,"good","bad")
  t1<-table(test$good_bad,factor(pred_tree, labels = c("bad","good"),levels = c("bad","good")))
  tpr1<-t1[1]/(t1[1]+t1[3])
  fpr1<-t1[2]/(t1[2]+t1[4])
  vec<-rbind(vec,c(tpr1,fpr1))

}

dmat1<-as.data.frame(vec)
colnames(dmat1)<-c("tpr","fpr")
#ROC plot for tree
plot(dmat1$tpr~dmat1$fpr,dmat1,col="red",type="l",xlab = "FPR",ylab="TPR",
     main = "ROC curves")
# ROC plot for naive Bayes
lines(dmat$tpr~dmat$fpr,dmat,col="blue")
```

```r
#add legend
legend(0.8,0.75,legend = c("naive","tree"),col=c("blue","red"),pch=15,
       title=" models", bg="lightblue")

#probs for good,bad on train
pntr<-predict(naive,newdata=train,type="raw")
#probs for good,bad on test
pntes<-predict(naive,newdata=test,type="raw")

my_fun<-function(X,Y){
  X<-as.data.frame(X)
  X[,1]<-X[,1]*10
  X$naive<-ifelse(X[,1]>X[,2],"bad","good")
  table<-table(Y,X$naive,dnn = c("actual","predictions"))
  preds<-X$naive
  list("table"=table,"preds"=preds)
}


cnf_train<-my_fun(pntr,train$good_bad)  #confusion matrix for train
cnf_test<-my_fun(pntes,test$good_bad)  #confusion matrix for test

cat("================= Confusion matrix for naive train =====================================\n")
cnf_train$table
cat("================= Confusion matrix for naive test ======================================\n")
cnf_test$table

me1<-mis_error(cnf_train$preds,train$good_bad)

me2<-mis_error(cnf_test$preds,test$good_bad)

cat("===================================================================================\n",
    "The misclassification error for train is :",me1,"and for test is :" ,me2)

#import library,data
library(ggplot2)
state<-read.csv("State.csv",header = T,sep=";")
#change the dot with comma
for (i in 2:6){
  state[,i]<-as.numeric(gsub(",",".",state[,i]))
}

#order data according to MET
ord_state<-state[order(state[,"MET"]),]

#plot the data
pl<-ggplot(ord_state,aes(y=EX,x=MET))+geom_point()+stat_smooth(method="lm",
                se=TRUE, fill=NA,formula=y ~ poly(x, 9, raw=TRUE),colour="red")+
  ggtitle("EX VS MEX with 8th degree fit line")
pl

library(tree)
#fit tree
```

```
modtree<-tree(EX~MET,data=ord_state,minsize=8)
set.seed(12345)
#apply cross validation
cv.res=cv.tree(modtree)
#find best leave number
best_leaf <- cv.res$size[which(cv.res$dev==min(cv.res$dev))]

cat("The optimal tree leaves are:",best_leaf)



par(mfrow=c(1,2))
plot(cv.res$size, cv.res$dev, type="b",col="red")
logk<-log(cv.res$k)
plot(logk, cv.res$dev,type="b", col="red")

#prune tree to best number of leaves
btree<-prune.tree(modtree, best = best_leaf)
#btree
#make predicitons
pred_btree<-predict(btree,ord_state)
#calculate errors
res_tree<-pred_btree-ord_state$EX
par(mfrow=c(1,3))
#plot of original and fitted
plot(ord_state$EX,col="blue")   #plot of the original data
points(pred_btree,col="red")    #ading point of predictions
#hostogram of residuals
hist(abs(res_tree))      #histogram of residuals
#original and fitted line
plot(ord_state$MET,ord_state$EX, pch=19, col="black")
partition.tree(btree, label="tree", add=TRUE)
summary(btree)


library(boot)
library(parallel)
set.seed(12345)
# computing bootstrap samples

f=function(data, ind){
  data1=data[ind,]# extract bootstrap sample
  mod<-tree(EX~MET,data=data1,control=tree.control(nobs=nrow(data1),minsize=8))
  modd<-prune.tree(mod,best=3)  #fit tree model
  #predict values
  pEX=predict(modd,newdata=data)

  return(pEX)
}


np_boot<-boot(ord_state, f, R=1000,parallel="multicore",ncpus=4) #make bootstrap
```

```
e=envelope(np_boot) #compute confidence bands

plot(ord_state$MET, ord_state$EX, pch=21, bg="orange")
points(ord_state$MET,pred_btree,type="l") #plot fitted line
#plot cofidence bands
points(ord_state$MET,e$point[2,], type="l", col="blue")
points(ord_state$MET,e$point[1,], type="l", col="blue")
#lines(ord_state$MET, e$overall[1, ], lty = 1)
#lines(ord_state$MET, e$overall[2, ], lty = 1)




set.seed(12345)

rng=function(data, mle) {
  data1=data.frame(EX=data$EX, MET=data$MET)
  n=length(data$EX)
  #generate new EX
  m<-predict(mle,data=data1)
  #
  data1$EX<-rnorm(n,m,sd(residuals(mle)))
  return(data1)
}

f1=function(data1){
  mod<-tree(EX~MET,data=data1,minsize=8)
  modd<-prune.tree(mod,best=3)                #fit tree
    #predict values from the original data
  predictedP<-predict(modd,ord_state)
  return(predictedP)

}


f2=function(data1){
  mod<-tree(EX~MET,data=data1,minsize=8)
  modd<-prune.tree(mod,best=3)                #fit tree
    #predict values from the original data
  pEX<-predict(modd,ord_state)
  n=length(ord_state$EX)
  predictedP=rnorm(n,pEX,sd(residuals(btree)))
  return(predictedP)
}


p_boot1=boot(ord_state, statistic=f1, R=1000,
         mle=btree,ran.gen=rng, sim="parametric",parallel = "multicore",ncpus=4)


e1=envelope(p_boot1) #compute confidence bands
set.seed(12345)
```

```r
p_boot2=boot(ord_state,statistic=f2,R=1000,
             mle=btree,ran.gen = rng,sim="parametric",parallel = "multicore",ncpus = 4)

e2=envelope(p_boot2)

plot(ord_state$MET, ord_state$EX, pch=21, bg="orange",ylim=c(150,460))
points(ord_state$MET,pred_btree,type="l") #plot fitted line
#plot cofidence bands
points(ord_state$MET,e1$point[2,], type="l", col="blue")
points(ord_state$MET,e1$point[1,], type="l", col="blue")
#plot prediction bands
lines(ord_state$MET, e2$point[1, ], lty = 1)
lines(ord_state$MET, e2$point[2, ], lty = 1)




#read data
nir<-read.csv2("NIRspectra.csv",sep=";")
#make pca
pca<-prcomp(nir[,1:126],scale=T)   # we scale because features have diffrent variance

# calculate variance of each PCA
pca.var<-pca$sdev^2
# calculate percentage of variance of PCAs
pca.var.per<-round(pca.var/sum(pca.var)*100,3)

print("The percentage of variance for feature space ")
pca.var.per

#alternative use summary(pca)
library(ggplot2)
library(RColorBrewer)
#make a df for the 14 pca.var.per
#D_pca<-as.data.frame(pca.var.per[1:14])
#D_pca$PCAs<-c("PCA1","PCA2","PCA3","PCA4","PCA5","PCA6","PCA7")
par(mfrow=c(1,2))
#ploting the percentage variances
# ggplot(D_pca,aes(y=D_pca[,1],x=D_pca[,2],group=1,fill=D_pca[,2]))+
#   geom_bar(stat = "identity")+labs(title="Percentages of PCAs",
#                                    y="percentage",x="PCAs")
barplot(pca.var.per[1:14],names.arg=c("PCA1","PCA2","PCA3","PCA4","PCA5","PCA6","PCA7","PCA8","PCA9","P
        col=brewer.pal(8, "Set2"))
#cumulative plot of percentages of pca's
plot(cumsum(pca.var.per), xlab = "Principal Component",
             ylab = "Proportion of Variance Explained",
             type = "b")

#plot of the 2 first PCA's
plot(pca$x[,1:2], col = "blue",xlab = paste0("PCA1 ",pca.var.per[1],"%"),
     ylab=paste0("PCA2 ",pca.var.per[2],"%"))
abline(v=0)#vertical line
abline(h=0)#horizontal line
```

```r
U=pca$rotation
par(mfrow=c(1,2))
plot(U[,1], main="Traceplot, PC1")   #trace plot PCA1
plot(U[,2],main="Traceplot, PC2")    #trace plots PCA2


library(fastICA)

set.seed(12345)
a <- fastICA(nir[,c(1:126)], 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "R", row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = F)

w<-a$K%*%a$W

par(mfrow=c(1,2))
#trace plots for ica 2
plot(w[,1],main="ICA1")
#trace plots for ica 1
plot(w[,2],main="ICA2")

plot(a$S, main = "ICA components",xlab="ICA1",ylab="ICA2",col="blue")
abline(v=0)#vertical line
abline(h=0)#horizontal line
```