LAB 2 BLOCK 2 - Group A15

*Zuxiang Li (**zuxli371**), Marcos F. Mourao (**marfr825**), Agustín Valencia (**aguva779**)*

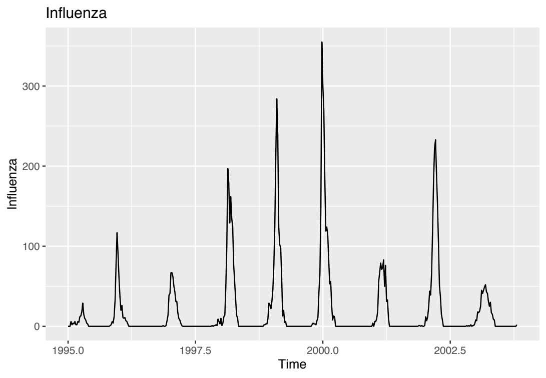
16 December 2019

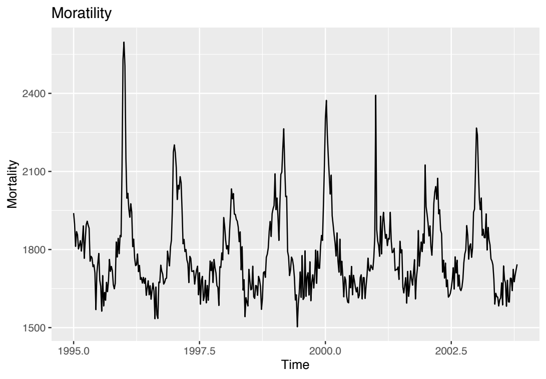
Assignment 1. Using GAM and GLM to examine mortality rates

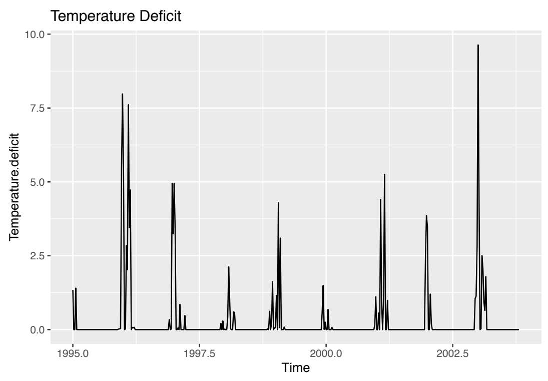
The Excel document influenza.xlsx contains weekly data on the mortality and the number of laboratory-confirmed cases of influenza in Sweden. In addition, there is information about population-weighted temperature anomalies (temperature deficits).

1. Use time series plots to visually inspect how the mortality and influenza number vary with time (use Time as X axis). By using this plot, comment how the amounts of influenza cases are related to mortality rates.

Plotting the data against the time:

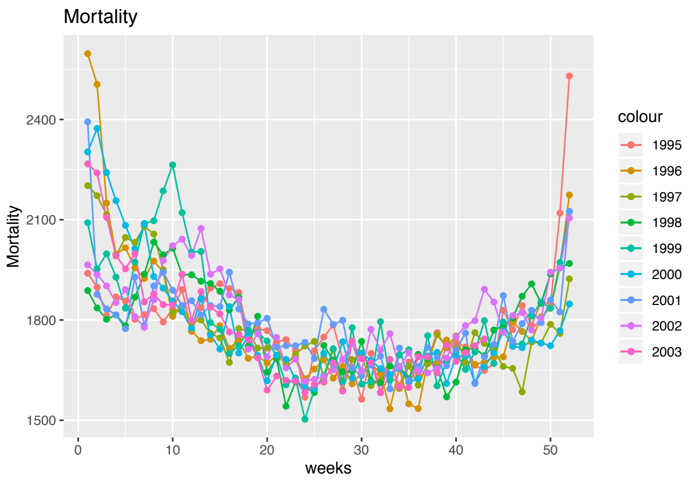


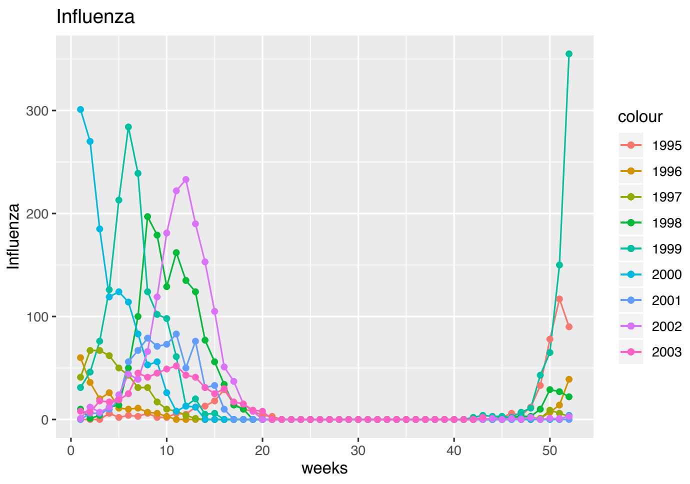


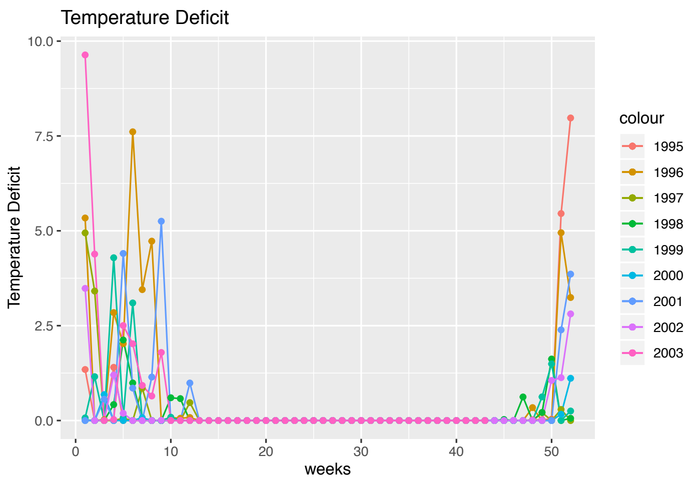


It seems to be a modality in the data. It can be seen that Influenza peaks seems to have a correlation with peaks in mortality and temperature deficit

As the model is meant to learn from weeks variations along the year, it is better to separate the data into several time series per year and see how the data looks now.







Now it is quite clear that the data shows that during winter weeks Influenza cases and Mortality related to increases, also the temperature deficit seems to follow that trend.

2. Use gam() function from mgcv package to fit a GAM model in which Mortality is normally distributed and modelled as a linear function of Year and spline function of Week, and make sure that the model parameters are selected by the generalized cross-validation. Report the underlying probabilistic model.

It is fitted one model with the following formula:

$$Mortality \approx Year + spline(Week)$$

The summary of the obtained probabilistic model is:

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = length(unique(data$Week)))
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -680.598   3367.760  -0.202    0.840
## Year           1.233      1.685   0.732    0.465
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Week) 14.32  17.87 53.86  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 52/53
## R-sq.(adj) =  0.677   Deviance explained = 68.8%
## GCV = 8708.6  Scale est. = 8398.9    n = 459
```

The model Mean and variance are :

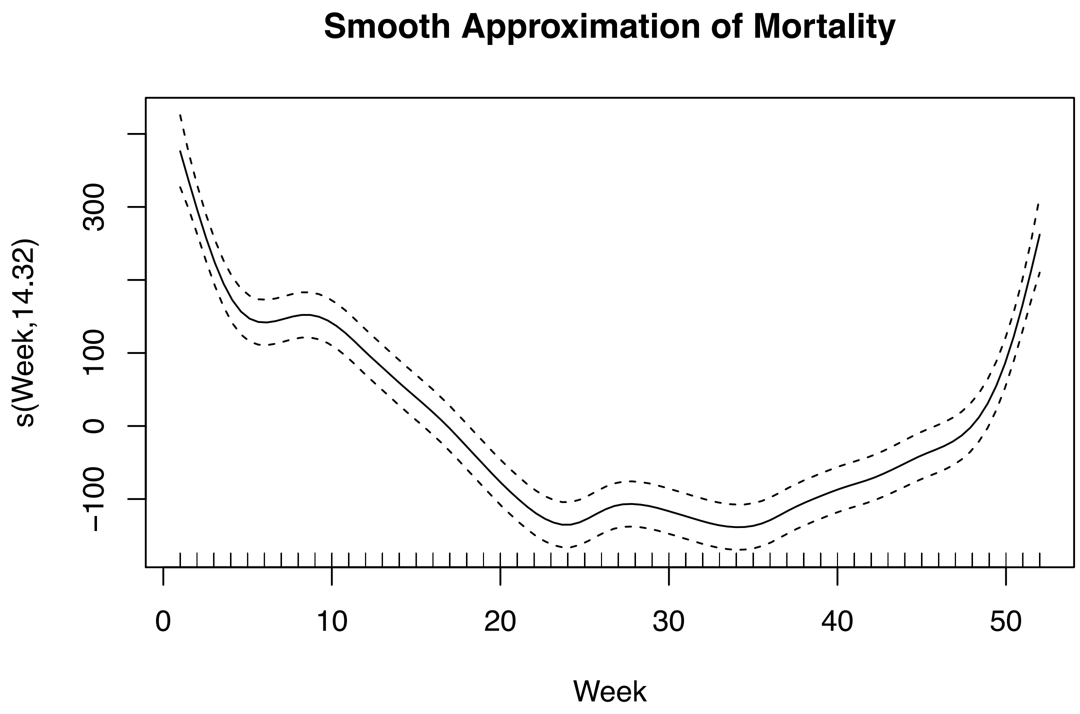
```
## [1] "Train predictions mean:"

## [1] 1783.765

## [1] "Train predictions variance:"

## [1] 25981.62
```

Thus as the summary states that the model family is Gaussian, it can be said that :

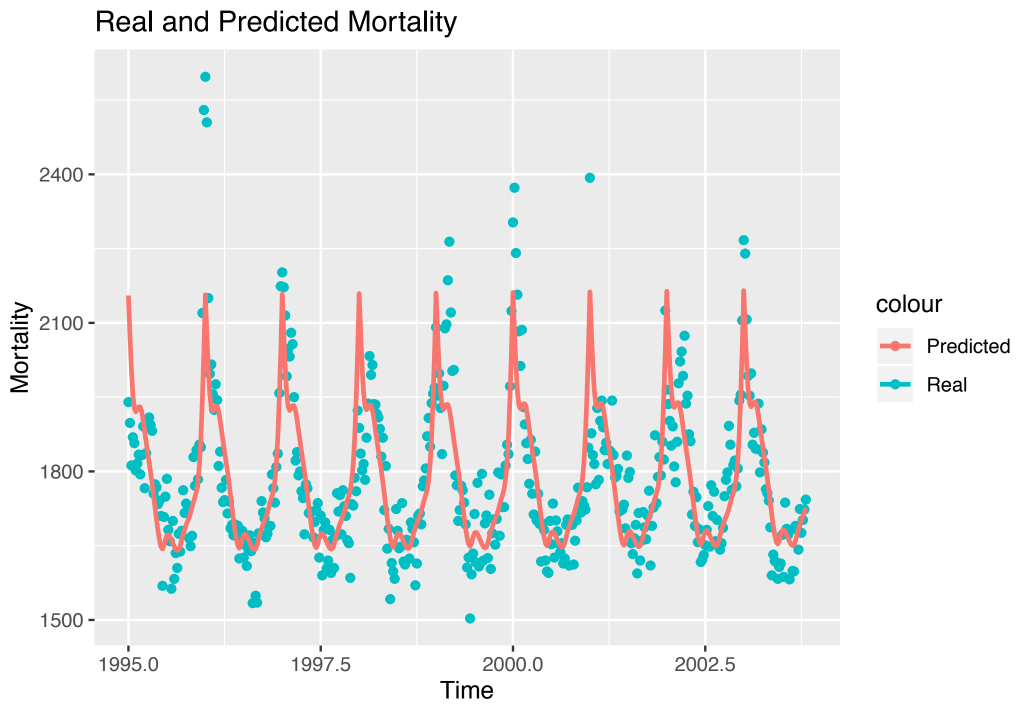
$$Mortality \sim Year + spline(Week)$$

$$Mortality \sim N(\mu = 1783.765, \sigma = 25981.62)$$

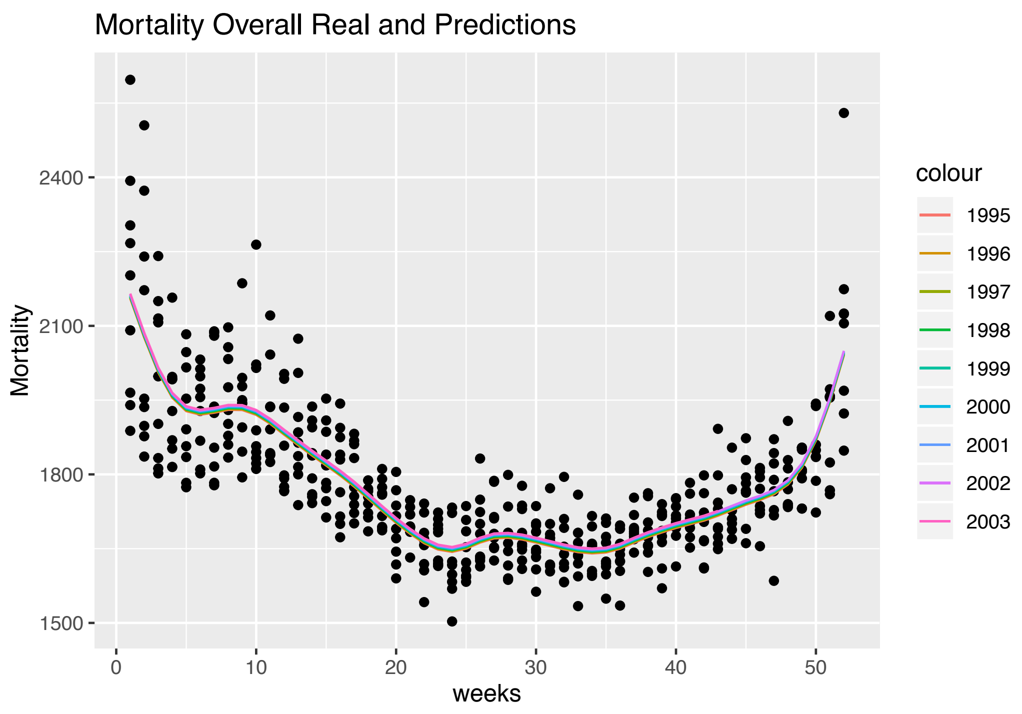
The smoother curve approximated by the GAM model is:



3. Plot predicted and observed mortality against time for the fitted model and comment on the quality of the fit. Investigate the output of the GAM model and report which terms appear to be significant in the model. Is there a trend in mortality change from one year to another? Plot the spline component and interpret the plot.



Although it seems to be a fair general approximation it seems to be missing some abnormal positive and negative peaks.



It can be seen that given the variability of mortality per week each year, the GAM fit acceptably well the data, though still too general, thus not as good as we would like to.

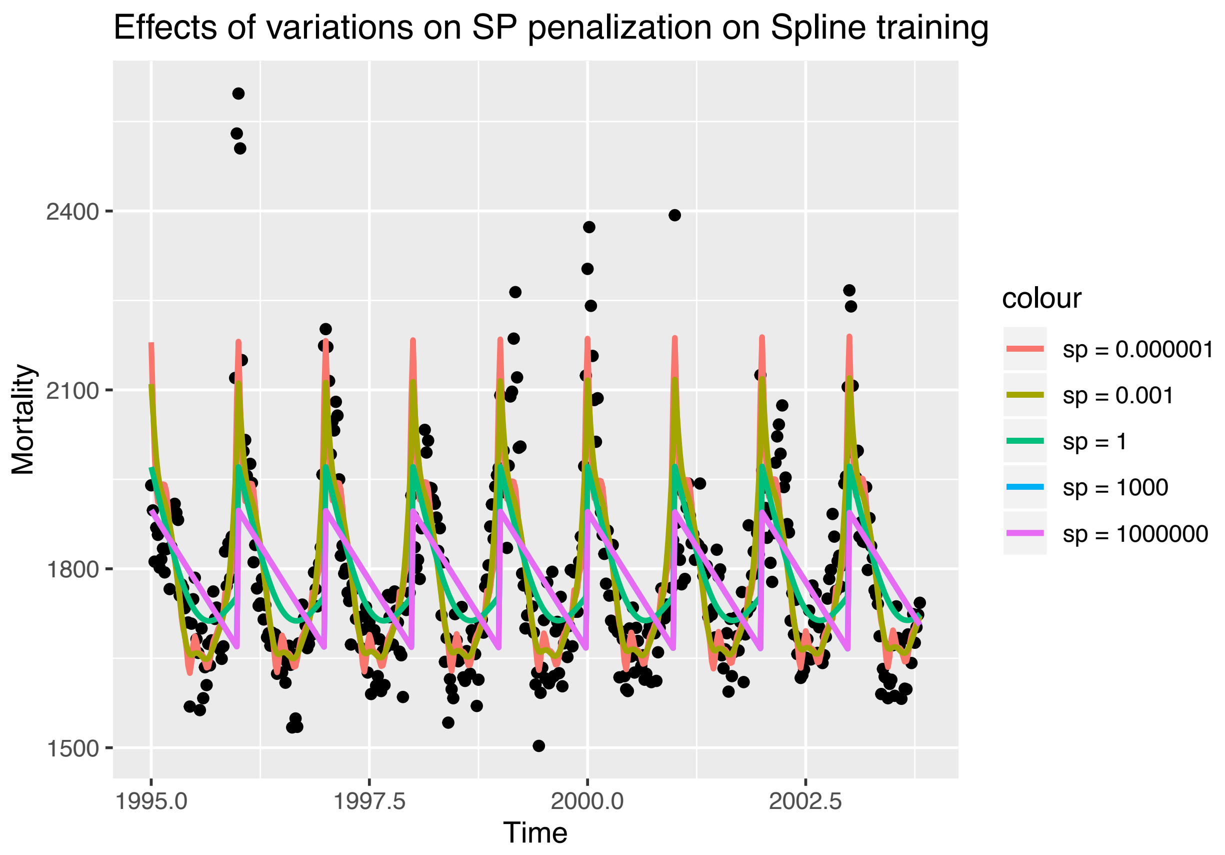
The 10 most significant terms in the model are:

```
## [1] "The 10 most important coefficients:"
```

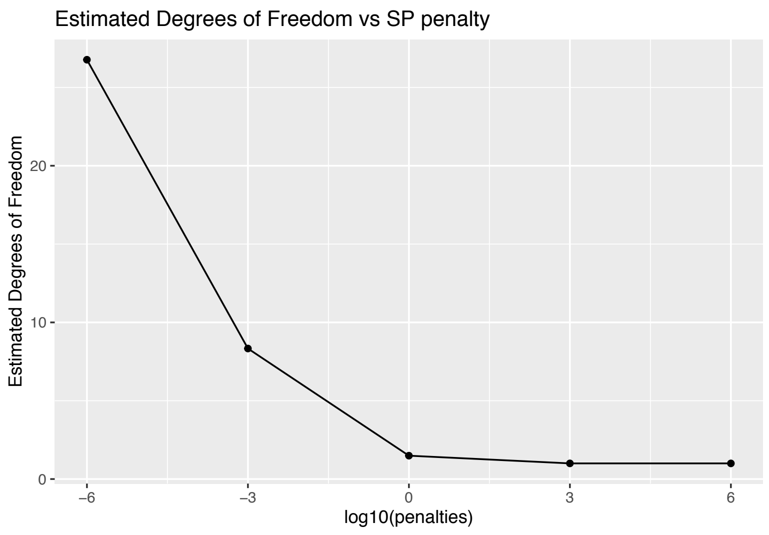
```
## s(Week).26 s(Week).36 s(Week).25 s(Week).20 s(Week).22 s(Week).39 s(Week).18
##   6962.767   6367.790   6224.860   3619.116   2193.093   2038.671   1588.407
## s(Week).16 s(Week).40 s(Week).12
##   1478.685   1468.449   1406.516
```

4. Examine how the penalty factor of the spline function in the GAM model from step 2 influences the estimated deviance of the model. Make plots of the predicted and observed mortality against time for cases of very high and very low penalty factors. What is the relation of the penalty factor to the degrees of freedom? Do your results confirm this relationship?

The smoothing penalty in `gam()` is given by the `sp` parameter sent to the constructor to be used in training. For this testing puropose it have been used the following penalties $\{10^{-6}, 10^{-3}, 1, 10^{3}, 10^{6}\}$



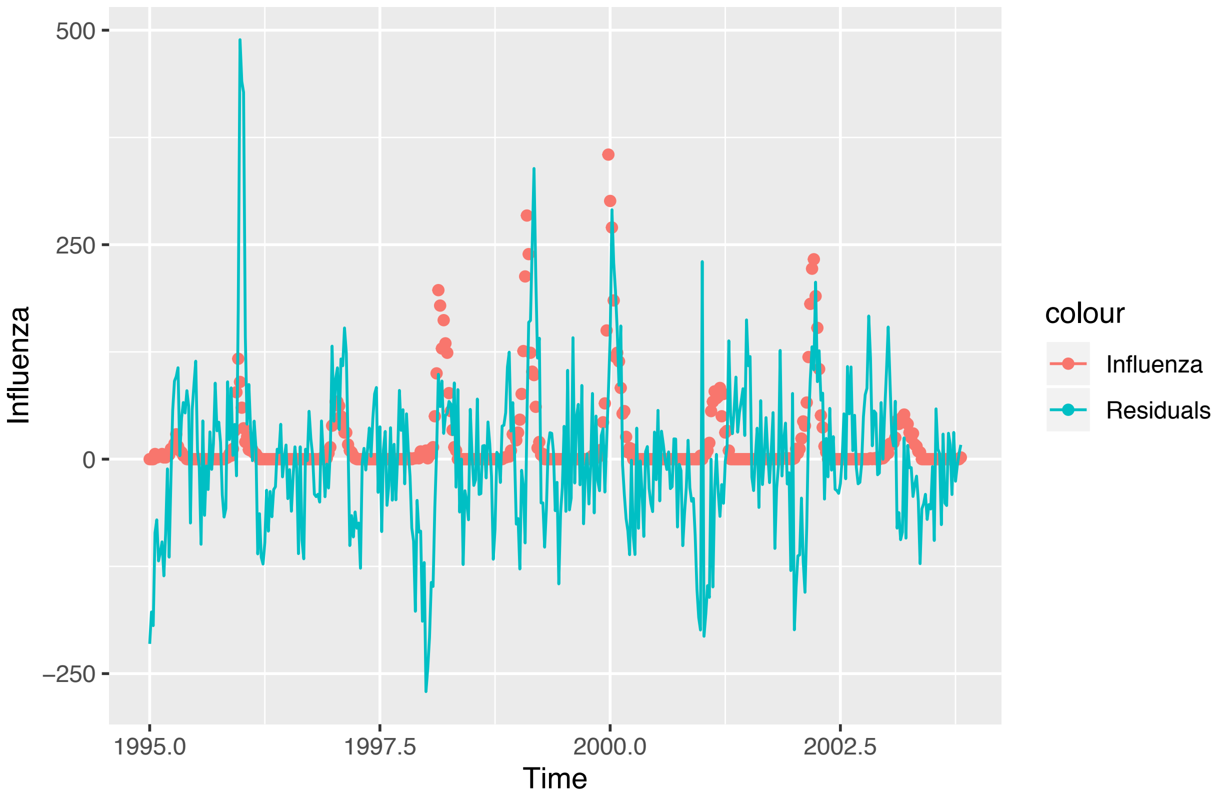
It can be seen that the lower the penalty the better the fit. In this case it looks good to have such a low penalty, though in other cases it might overfit the data.



Extracting the degrees of freedom from each penalization iteration it can be seen that the lower the penalization the more relaxed is the model, thus, the higher the degrees of freedom. The penalty factor is inversely proportional to the degrees of freedom. Indeed, by looking at the above plots, this relationship is confirmed. A high penalty essentialy drops a higher number of parameters in the linear smoother matrix S_λ (by approximating them to zero).

5. Use the model obtained in step 2 and plot the residuals and the influenza values against time (in one plot). Is the temporal pattern in the residuals correlated to the outbreaks of influenza?

At a first glance Residuals seem to be noise compared to Influenza behavior.



Nonetheless visual assumptions might be incorrect. Testing numerically by getting their correlation statistic is low, thus there is no evidence to say that the residuals from the model and the influenza measures are correlated.

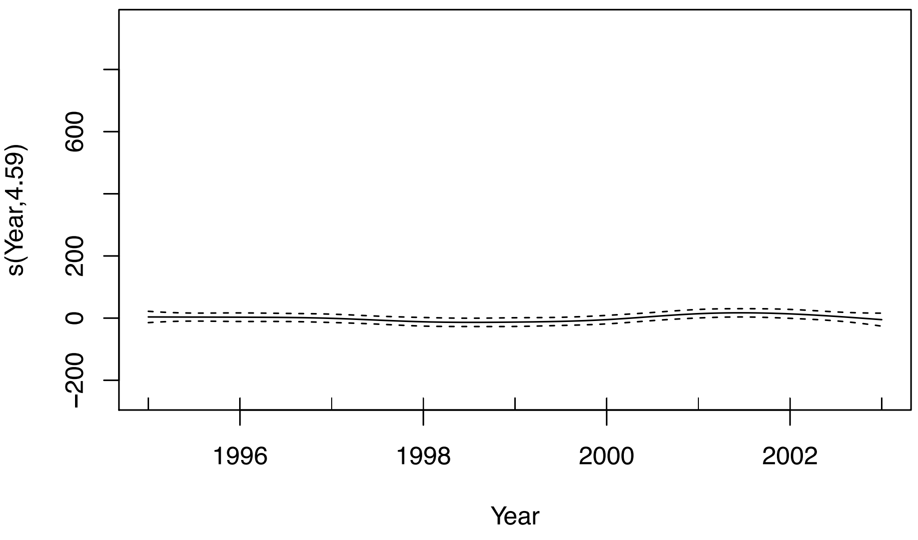
```
## [1] 0.3397395
```

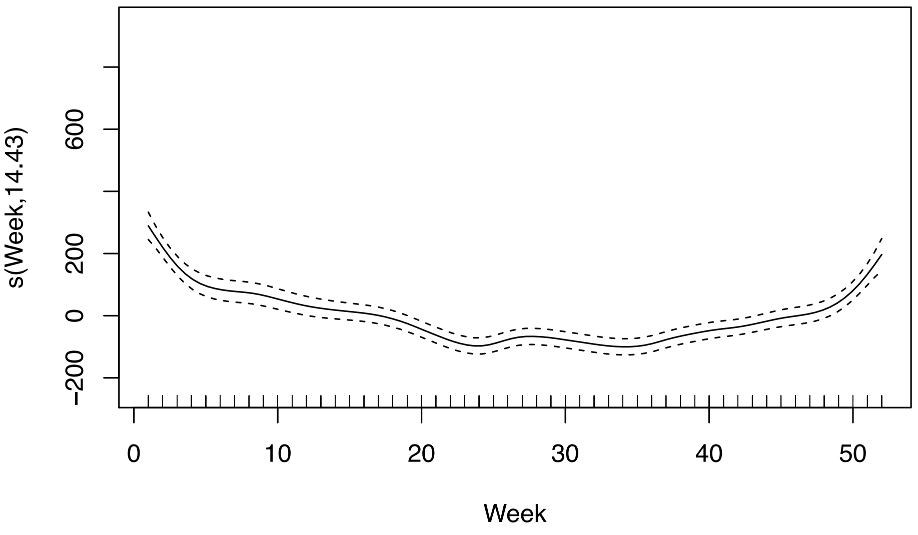
6. Fit a GAM model in R in which mortality is be modelled as an additive function of the spline functions of year, week, and the number of confirmed cases of influenza. Use the output of this GAM function to conclude whether or not the mortality is influenced by the outbreaks of influenza. Provide the plot of the original and fitted Mortality against Time and comment whether the model seems to be better than the previous GAM models.

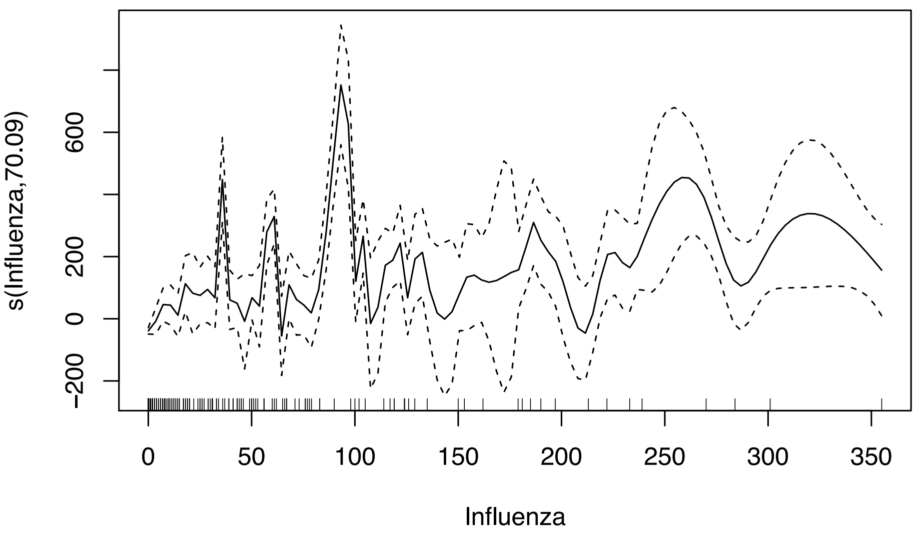
It has been trained a GAM using the following formula:

$$Mortality \approx spline(Year) + spline(Week) + spline(Influenza)$$

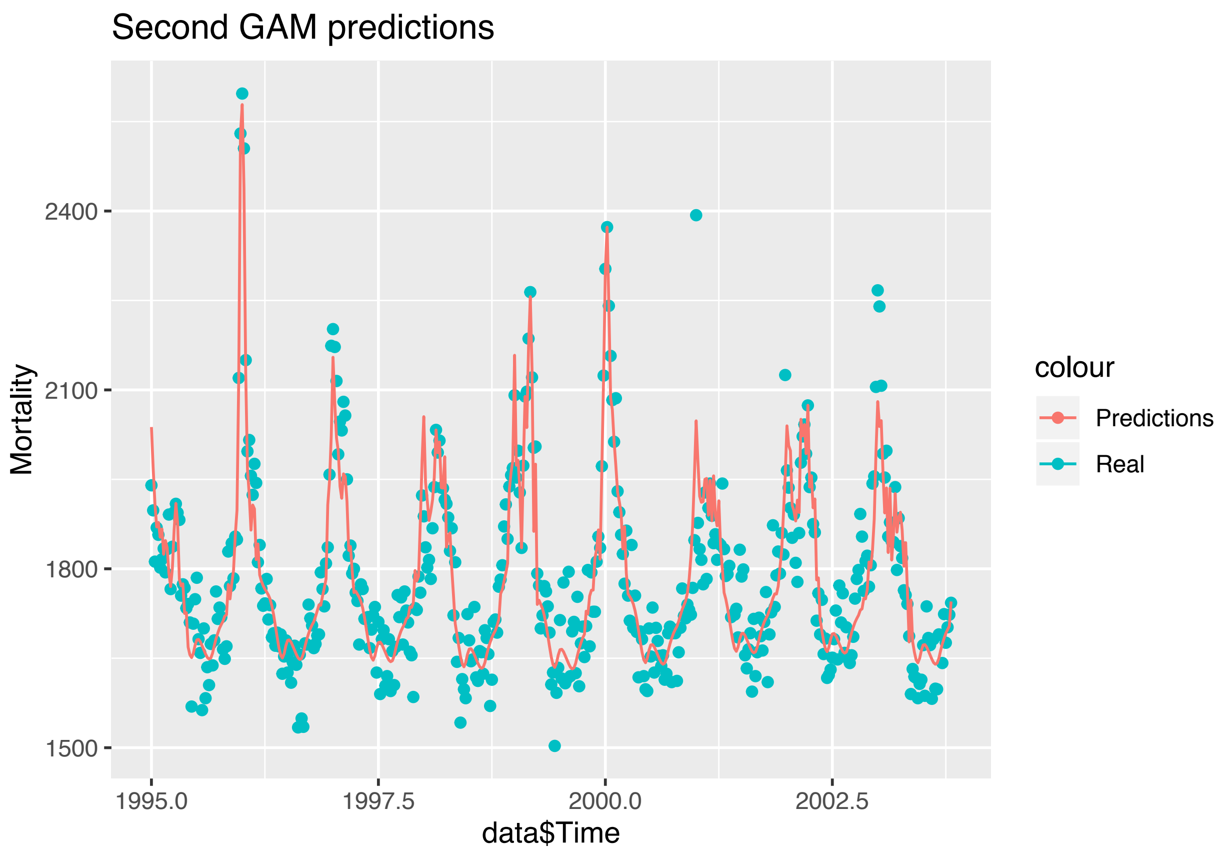
The obtained smoothers are the following.







Now, using the model to predict the Mortality :

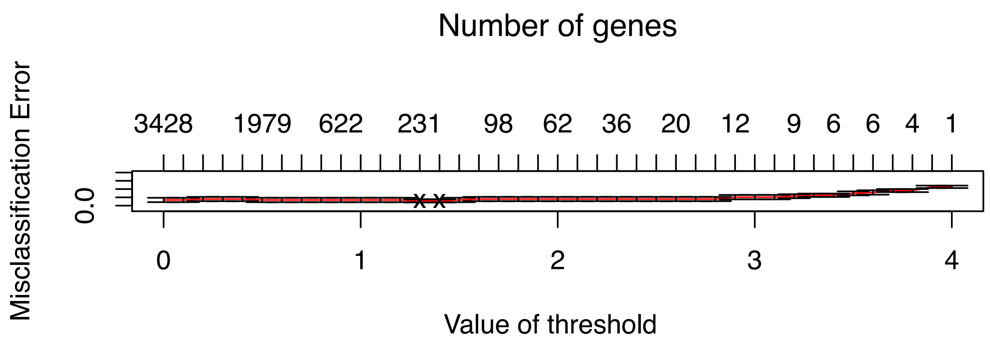


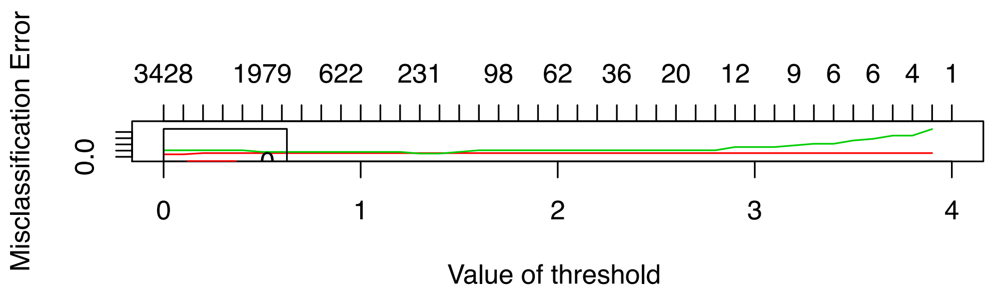
The model seems to fit better the data, though there are still some data points not being well approximatted. This is sign that the predictor is not overfitted, so compared against the previous predictors, this is the best one.

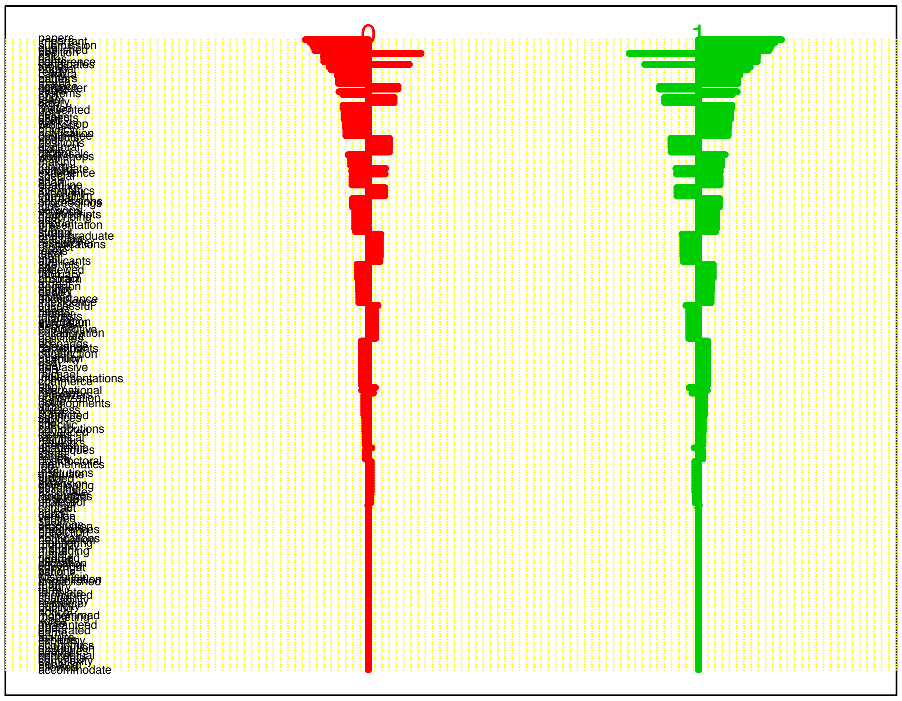
Assignment 2. High-dimensional methods

The data file data.csv contains information about 64 e-mails which were manually collected from DBWorld mailing list. They were classified as: 'announces of conferences' (1) and 'everything else' (0) (variable Conference)

1. Divide data into training and test sets (70/30) without scaling. Perform nearest shrunken centroid classification of training data in which the threshold is chosen by cross-validation. Provide a centroid plot and interpret it. How many features were selected by the method? List the names of the 10 most contributing features and comment whether it is reasonable that they have strong effect on the discrimination between the conference mails and other mails? Report the test error.



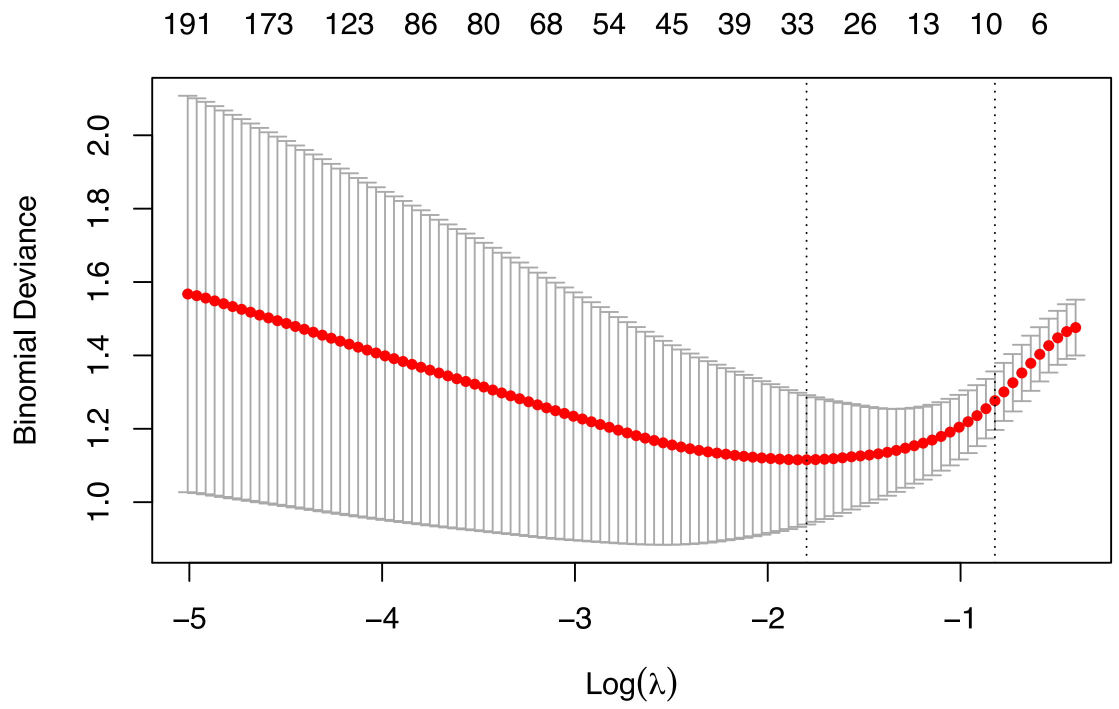




We get the threshold is 1.3 with using cross-validation. From the centroid plot we can see the contribution of each word made to the result(conference or not). There are 231 features selected in total. The 10 most contributing features are “papers”, “important”, “submission”, “due”, “published”, “position”, “call”, “conference”, “dates”, “candidates”. It’s clear that these word have a strong connection to conference. The test error is 5%.

2. Compute the test error and the number of the contributing features for the following methods fitted to the training data:

a. Elastic net with the binomial response and $\alpha = 0.5$ in which penalty is selected by the cross-validation



```
## Elasticnet model has  10  nonzero coefficients
```

```
## 10 %
```

b. Support vector machine with "vanilladot" kernel.

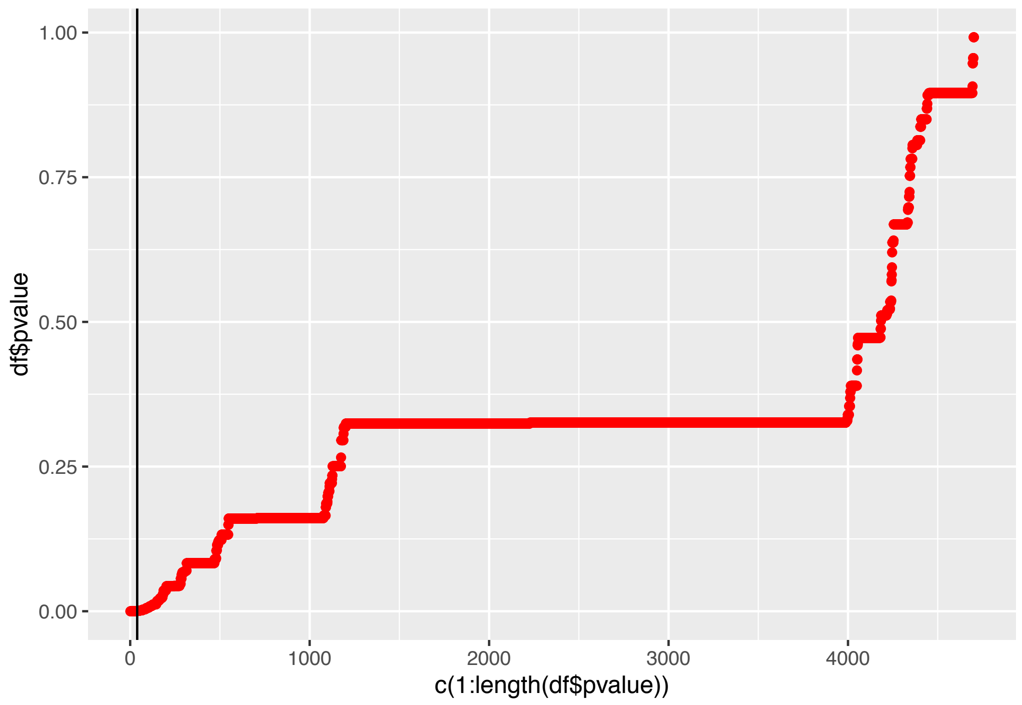
```
##  Setting default kernel parameters
```

```
## 5 %
```

Compare the results of these models with the results of the nearest shrunken centroids (make a comparative table). Which model would you prefer and why?

Error rate for Elastic net is 10% and for SVM is 5%. In this case we prefer to use SVM since it ignores the effect of high-dimensional data and it provides the lowest misclassfication rate.

3. Implement Benjamini-Hochberg method for the original data, and use t.test() for computing p-values. Which features correspond to the rejected hypotheses? Interpret the result.



```
## [1] 39
```

```
##               name       pvalue
## 3036        papers 1.116910e-10
## 4060    submission 7.949969e-10
## 3187      position 8.219362e-09
## 3364     published 1.835157e-07
## 2049     important 3.040833e-07
## 596           call 3.983540e-07
## 869     conference 5.091970e-07
## 607     candidates 8.612259e-07
## 1045         dates 1.398619e-06
## 3035         paper 1.398619e-06
## 4282        topics 5.068373e-06
## 2463       limited 7.907976e-06
## 606      candidate 1.190607e-05
## 599         camera 2.099119e-05
## 3433         ready 2.099119e-05
## 389        authors 2.154461e-05
## 3125           phd 3.382671e-05
## 3312      projects 3.499123e-05
## 2974           org 3.742010e-05
## 681         chairs 5.860175e-05
## 1262           due 6.488781e-05
## 2990      original 6.488781e-05
## 2889  notification 6.882210e-05
## 3671        salary 7.971981e-05
## 3458        record 9.090038e-05
## 3891        skills 9.090038e-05
## 1891          held 1.529174e-04
## 4177          team 1.757570e-04
## 3022         pages 2.007353e-04
## 4628      workshop 2.007353e-04
## 810      committee 2.117020e-04
## 3285   proceedings 2.117020e-04
## 272          apply 2.166414e-04
## 4039        strong 2.246309e-04
## 2175 international 2.295684e-04
## 1088        degree 3.762328e-04
## 1477     excellent 3.762328e-04
## 3191          post 3.762328e-04
## 3243     presented 3.765147e-04
```

There are 39 features correspond to the rejected hypotheses with $\alpha = 0.05$.

We are testing:

H_0, j: word j has no effect on classification.

H_1, j: word j has effect on classification.

The variables correspondent to the rejected NULL hypothesis, therefore, are the ones that are significant to the classification as conference mail.

The 39 words selected are significant and are ranked in order of importance, similarly to what can be seen in the centroid plot from item 1.

Appendix A : Code for Assignment 1

```r
###############################################################################
##                              Assignment 1
###############################################################################

# Import data
dataPath <- "data/influenza.xlsx"
data <- read.xlsx(dataPath)

# Time Series plotting
mortPlot <- ggplot(data) +
    geom_line(aes(x=Time, y=Mortality), color="black") + ggtitle("Moratility")
infPlot <- ggplot(data) +
    geom_line(aes(x=Time, y=Influenza), color="black") + ggtitle("Influenza")
tempPlot <- ggplot(data) +
    geom_line(aes(x=Time, y=Temperature.deficit), color="black") +
    ggtitle("Temperature Deficit")
infPlot
mortPlot
tempPlot

# Time series per year
years <- unique(data$Year)
weeks <- unique(data$Week)
mortData <- list()
infData  <- list()
tempData <- list()
for(i in 1:length(years)) {
    year <- years[i]
    mortData[[i]] <- data$Mortality[which(data$Year == year)]
    infData[[i]]  <- data$Influenza[which(data$Year == year)]
    tempData[[i]] <- data$Temperature.deficit[which(data$Year == year)]
}
names(mortData) <- years
names(infData)  <- years
names(tempData) <- years

# create data.frames for ggplot
plotData <- function(d, title) {
    shortWeeks <- 1:length(d$'2003')
    p <- ggplot() +
        geom_line(aes(x=weeks, y=d$'1995', color="1995")) +
        geom_line(aes(x=weeks, y=d$'1996', color="1996")) +
        geom_line(aes(x=weeks, y=d$'1997', color="1997")) +
        geom_line(aes(x=weeks, y=d$'1998', color="1998")) +
        geom_line(aes(x=weeks, y=d$'1999', color="1999")) +
        geom_line(aes(x=weeks, y=d$'2000', color="2000")) +
        geom_line(aes(x=weeks, y=d$'2001', color="2001")) +
        geom_line(aes(x=weeks, y=d$'2002', color="2002")) +
        geom_line(aes(x=shortWeeks, y=d$'2003', color="2003")) +
        geom_point(aes(x=weeks, y=d$'1995', color="1995")) +
        geom_point(aes(x=weeks, y=d$'1996', color="1996")) +
        geom_point(aes(x=weeks, y=d$'1997', color="1997")) +
        geom_point(aes(x=weeks, y=d$'1998', color="1998")) +
        geom_point(aes(x=weeks, y=d$'1999', color="1999")) +
        geom_point(aes(x=weeks, y=d$'2000', color="2000")) +
        geom_point(aes(x=weeks, y=d$'2001', color="2001")) +
        geom_point(aes(x=weeks, y=d$'2002', color="2002")) +
        geom_point(aes(x=shortWeeks, y=d$'2003', color="2003")) +
        ggtitle(title) + ylab(title)
    return(p)
}
mortPlot <- plotData(mortData, "Mortality")
infPlot  <- plotData(infData, "Influenza")
tempPlot <- plotData(tempData, "Temperature Deficit")
mortPlot
infPlot
tempPlot

# Training GAM
model <- gam (
            Mortality ~ Year +
            s(Week, k = length(unique(data$Week))),
            data=data,
            method = "GCV.Cp"
        )
summary(model)
modelMean <- mean(model$y)
modelVar <- var(model$y)
print("Train predictions mean:")
modelMean
print("Train predictions variance:")
modelVar
plot(model, main="Smooth Approximation of Mortality")

# Predictions
predictions <- predict(model, data)
ggplot(data) +
    geom_point(aes(x=Time, y=Mortality, color="Real")) +
    geom_line(aes(x=Time, y=predictions, color="Predicted"), size=1) +
    ggtitle("Real and Predicted Mortality")

# Plot GAM predictions per year
plotAll <- function(d, p, title) {
    shortWeeks <- 1:length(d$'2003')
    a <- ggplot() +
        geom_point(aes(x=weeks, y=d$'1995')) +
        geom_point(aes(x=weeks, y=d$'1996')) +
        geom_point(aes(x=weeks, y=d$'1997')) +
        geom_point(aes(x=weeks, y=d$'1998')) +
        geom_point(aes(x=weeks, y=d$'1999')) +
        geom_point(aes(x=weeks, y=d$'2000')) +
        geom_point(aes(x=weeks, y=d$'2001')) +
        geom_point(aes(x=weeks, y=d$'2002')) +
        geom_point(aes(x=shortWeeks, y=d$'2003')) +
```

```r
        geom_line(aes(x=weeks, y=p$'1995', color="1995")) +
        geom_line(aes(x=weeks, y=p$'1996', color="1996")) +
        geom_line(aes(x=weeks, y=p$'1997', color="1997")) +
        geom_line(aes(x=weeks, y=p$'1998', color="1998")) +
        geom_line(aes(x=weeks, y=p$'1999', color="1999")) +
        geom_line(aes(x=weeks, y=p$'2000', color="2000")) +
        geom_line(aes(x=weeks, y=p$'2001', color="2001")) +
        geom_line(aes(x=weeks, y=p$'2002', color="2002")) +
        geom_line(aes(x=shortWeeks, y=p$'2003', color="2003")) +
        ggtitle(title)
    return(a)
}

preds <- list()
for (i in 1:length(years)) {
    year <- years[i]
    d <- data[which(data$Year == year),]
    preds[[i]] <- predict(model, d)
}
names(preds) <- years
p <- plotAll(mortData, preds, "Mortality Overall Real and Predictions")
p <- p + ylab("Mortality")
p

# significant terms.
orderedCoefficients <- model$coefficients[order(model$coefficients, decreasing = TRUE)]
print("The 10 most important coefficients:")
orderedCoefficients[1:10]


# Penalty factor analysis
penalties <- c(1e-6, 1e-3, 1, 1e3, 1e6)
predictions <- list()
estDegFreedom <- vector(length = length(penalties))
for (i in 1:length(penalties)) {
    model <- gam (  Mortality ~ Year +
                    s(Week, k = length(unique(data$Week))),
                    data=data,
                    method = "GCV.Cp",
                    sp = penalties[i]
                 )
    predictions[[i]] <- predict(model, data)
    estDegFreedom[i] <- summary(model)$edf
}
df <- data.frame(
    time = data$Time,
    real = data$Mortality,
    sp_1u = predictions[[1]],
    sp_1m = predictions[[2]],
    sp_1  = predictions[[3]],
    sp_1k = predictions[[4]],
    sp_1M = predictions[[5]]
)
```

```r
ggplot(df) +
    geom_point(aes(x=time, y=real), color="black") +
    geom_line(aes(x=time, y=sp_1u,  color="sp = 0.000001"), size=1) +
    geom_line(aes(x=time, y=sp_1m,  color="sp = 0.001"), size=1) +
    geom_line(aes(x=time, y=sp_1 ,  color="sp = 1" ), size=1) +
    geom_line(aes(x=time, y=sp_1k,  color="sp = 1000"), size=1) +
    geom_line(aes(x=time, y=sp_1M,  color="sp = 1000000"), size=1) +
    ggtitle("Effects of variations on SP penalization on Spline training") +
    xlab("Time") + ylab("Mortality")


# Relation between penalty factors and degrees of freedom
ggplot() +
    geom_point(aes(x=log10(penalties), y=estDegFreedom)) +
    geom_line(aes(x=log10(penalties), y=estDegFreedom)) +
    ggtitle("Estimated Degrees of Freedom vs SP penalty") +
    ylab("Estimated Degrees of Freedom")


# Comparing model residuals against influenza cases
model <- gam (
                Mortality ~ Year +
                s(Week, k = length(unique(data$Week))),
                data=data,
                method = "GCV.Cp"
              )
ggplot(data) +
    geom_point(aes(x=Time, y=Influenza, color="Influenza")) +
    geom_line(aes(x=Time, y=model$residuals, color="Residuals")) +
    ggtitle("")

# Correlation measurement
corr <- cor(model$residuals, data$Influenza)
print(corr)

# GAM 2 training
model <- gam( formula = Mortality ~ s(Year, k = length(unique(data$Year))) +
                                    s(Week, k = length(unique(data$Week))) +
                                s(Influenza, k=length(unique(data$Influenza))),
              data = data,
              method = "GCV.Cp"
            )
plot(model)

# GAM 2 predictions
predictions <- predict(model, data)
ggplot() +
    geom_point(aes(x=data$Time, y=data$Mortality, color="Real")) +
    geom_line(aes(x=data$Time, y=predictions, color="Predictions")) +
    ggtitle("Second GAM predictions") + ylab("Mortality")
```

Appendix B : Code for Assignment 2

```
################################################################################
##                              Assignment 2
################################################################################
data<-read.csv2("data/data.csv",check.names = FALSE)
names(data)<-iconv(names(data),to="ASCII")
RNGversion("3.5.1")

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.7))
train=data[id,]
test=data[-id,]

x<-t(train[,-4703])

y<-train[[4703]]


x_test<-t(test[,-4703])

y_test<-test[[4703]]

my_data<-list(x=x,y=as.factor(y),geneid=as.character(1:nrow(x)),genenames=rownames(x))
my_data_test<-list(x=x_test,y=as.factor(y_test),geneid=as.character(1:nrow(x_test)),genenames=rownames(

mod<-pamr.train(my_data,threshold = seq(0,4,0.1))
cvmodel<-pamr.cv(mod,my_data)

thr<-cvmodel$threshold[which.min(cvmodel$error)]
pamr.plotcv(cvmodel)
pred<-pamr.predict(mod,my_data_test$x,threshold = thr,type="class")
pamr.plotcen(mod,my_data,thr)

#pamr.listgenes(mod,my_data,thr,genenames = TRUE)[1:10,]
#res<-as.data.frame(pamr.listgenes(mod,my_data,thr,genenames = TRUE))

#listgene
#my_data$genenames[as.numeric(listgene)]

library(glmnet)

x<-train[,-4703]
y<-train[[4703]]

x_test<-test[,-4703]
y_test<-test[[4703]]

mod<-cv.glmnet(as.matrix(x),y,alpha=0.5,family="binomial")
plot(mod)
penalty_min<-mod$lambda.min
real_mod<-glmnet(as.matrix(x),y,alpha=0.5,lambda = penalty_min,family="binomial")
```

```r
pred<-predict(real_mod,as.matrix(x_test),type="class")

coeffs <- coef(mod)
nonzeroCoeffs <- coeffs[which(coeffs != 0)]
selectedCoeffs <- length(nonzeroCoeffs)
cat("Elasticnet model has ", selectedCoeffs, " nonzero coefficients\n")

cft<-table(pred,y_test)
mis_rate<-1-(cft[1,1]+cft[2,2])/sum(cft)
cat(mis_rate*100,"%")
fit<-ksvm(as.matrix(x),y,data=train,kernel="vanilladot",type="C-svc",scale=FALSE)
pred<-predict(fit,x_test,type="response")

cft<-table(pred,y_test)

mis_rate<-1-(cft[1,1]+cft[2,2])/sum(cft)
cat(mis_rate*100,"%")
x=as.matrix(data[,-4703])
y=as.factor(data[[4703]])

df<-data.frame(name=c(),pvalue=c())

for(i in 1:ncol(x)){
  tmpv<-t.test(x[,i]~y,alternative="two.sided",conf.level=0.95)$p.value
  tdf<-data.frame(name=colnames(x)[i],pvalue=tmpv)
  df<-rbind(df,tdf)
}
df<-df[order(df$pvalue),]

a=0.05
max_i=1
for(i in 1:length(df$pvalue)){
  if(df$pvalue[i]<=a*i/length(df$pvalue)){
    max_i=i
  }
}
ggplot()+geom_point(aes(x=c(1:length(df$pvalue)),y=df$pvalue),col="red")+geom_vline(xintercept = 39)
print(max_i)

df[1:39,]
```

Appendix C : Environment setup Code

```r
knitr::opts_chunk$set(echo = FALSE)
library(openxlsx)
library(ggplot2)
library(mgcv)
library(pamr)
library(glmnet)
library(kernlab)
library(readr)
RNGversion('3.5.1')
set.seed(12345)
```

Machine Learning Assignment 01 - Group A15

Zuxiang Li (zuxli371), Marcos F. Mourao (marfr825), Agustín Valencia (aguva779)

23/11/2019

Good report overall, well done!

Acknowledgements

The assignments were solved mainly under Lab time and study meetings. Although we worked together, every team member has written his own individual report by himself and this document summarize the best of all three individual reports.

First question is based on Marcos' work, third one is based on Zuxiang's report, and fourth one is based on Agustín's solution. After compiling them the report was polished by all team members.

Assignment 1. Spam classification with nearest neighbors

2. Use logistic regression to classify the training and test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.5$, otherwise $\hat{Y} = 0$ and report the confusion matrices and the misclassification rates for train and test data. Analyze the obtained results.

Confusion matrix for training data:

```
##                 Rate    Value
## 1      True positive 73.15011
## 2      True negative 89.63211
## 3     False positive 26.84989
## 4     False negative 10.36789
## 5 Misclassification 16.05839
```

```
##     predicted
## true   0   1
##    0 804 127
##    1  93 346
```

Confusion matrix for testing data:

```
##                 Rate    Value
## 1      True positive 69.57447
## 2      True negative 89.77778
## 3     False positive 30.42553
## 4     False negative 10.22222
## 5 Misclassification 17.15328
```

```
##     predicted
## true   0   1
##    0 808 143
##    1  92 327
```

The confusion matrix measure how well our model is performing under its training. This model performed similarly for both training and testing data. Although having similar performance scores, the true positive rate drops around 4% from tranining to testing data. The misclassification rate is somewhat low (~17%), but the false positive rate is high (~30%). This means that real mail is often being flagged as spam. To reduce this, we can increase the classification principle, as we can see below.

3. Use logistic regression to classify the test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.8$, otherwise $\hat{Y} = 0$

Confusion matrix for training data:

```
##                Rate    Value
## 1     True positive 91.37931
## 2     True negative 73.44498
## 3    False positive  8.62069
## 4    False negative 26.55502
## 5 Misclassification 25.03650
```

```
##     predicted
## true   0   1
##    0 921  10
##    1 333 106
```

Confusion matrix for testing data:

```
##                Rate    Value
## 1     True positive 84.00000
## 2     True negative 74.77912
## 3    False positive 16.00000
## 4    False negative 25.22088
## 5 Misclassification 24.37956
```

```
##     predicted
## true   0   1
##    0 931  20
##    1 314 105
```

Very good analysis!

Even though the misclassification rate went up with the new criteria, it is still preferred over the previous criteria. In this particular classification scenario (spam or not spam), classifying "real mail" as spam is much worse than classifying spam as "real mail". From a user point of view, having some spam in one's inbox is better than missing "real mail" that went to the spam box. Here, the key is analysing the false positive rate, which is much smaller for the new criteria of $p(Y = 1|X) > 0.8$.

4. Use standard kknn() with K = 30 from package *kknn*, report the misclassification rates for the training and test data and compare the results with step 2.

Confusion matrix for training data:

```
##                Rate     Value
## 1     True positive 70.428016
## 2     True negative 91.004673
## 3    False positive 29.571984
## 4    False negative  8.995327
## 5 Misclassification 16.715328
```

```
##     predicted
## true   0   1
##    0 779 152
##    1  77 362
```

Confusion matrix for testing data:

```
##                Rate    Value
## 1     True positive 48.97541
## 2     True negative 79.59184
## 3    False positive 51.02459
## 4    False negative 20.40816
## 5 Misclassification 31.31387
```

```
##     predicted
## true   0   1
##    0 702 249
##    1 180 239
```

The new model has worse results than the logistic regression with significantly higher misclassification rates for both test and training data. This approach also results in a higher number of false positives and false negatives.

Misclassification rate for training data is not significantly high.

5. Repeat step 4 for K=1 and compare results with step 4. What effects does the decrease of K lead to and why?

Confusion matrix for training data:

```
##                  Rate Value
## 1     True positive   100
## 2     True negative   100
## 3    False positive     0
## 4    False negative     0
## 5 Misclassification     0
```

```
##     predicted
## true   0   1
##    0 931   0
##    1   0 439
```

Confusion matrix for testing data:

```
##                  Rate    Value
## 1     True positive 43.25323
## 2     True negative 77.68396
## 3    False positive 56.74677
## 4    False negative 22.31604
## 5 Misclassification 35.91241
```

```
##     predicted
## true   0   1
##    0 644 307
##    1 185 234
```

In this case, decreasing K to 1 in our model overfitted the data. That is, the predicted values according to the model are exaclty the training values. When predicting the training data itself, the misclassification rate was 0%, as expected.

Good!

While using the testing data, however, this leads to worse performance: higher misclassification, false positives, and false negatives rates.

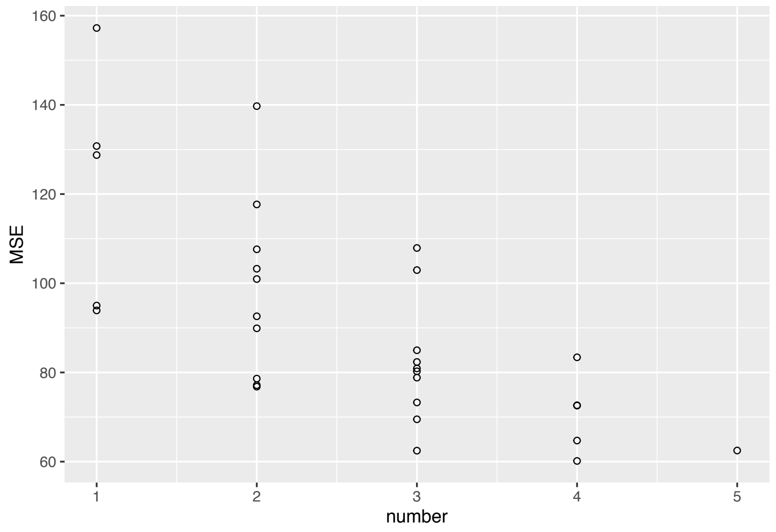
Assignment 3. Feature selection by cross-validation in a linear model.

2. Test your function on data set swiss available in the standard R repository:

- Fertility should be Y
- All other variables should be X
- Nfolds should be 5

Report the resulting plot and interpret it. Report the optimal subset of features and comment whether it is reasonable that these specific features have largest impact on the target.

```
## $CV
## [1] 60.15763
##
## $Features
## [1] 1 0 1 1 1
##
## $plot
```

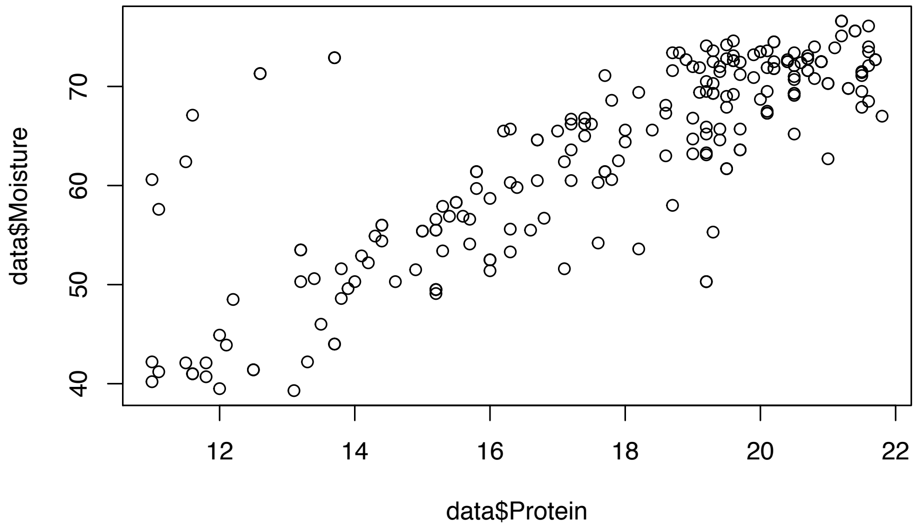


From the MSE plot it can be observed that when the number of features under consideration is increased, the MSE tends to decrease. This indicates that, in this case, using simple models underfits the data, though using too many can also overfit the data and make the MSE scores increase again.

The minimum MSE score is reached at 4 features and its value is 60.15763. The best features subset is $x = (1, 0, 1, 1, 1)$ which comprise `agriculture`, `education`, `religion` and `infant mortality` data. As `examination` feature is dropped it can be intepreted that marks on army examinations do not contribute to explain Fertility changes.

Assignment 4. Linear regression and regularization

1. Import data and create a plot of Moisture versus Protein. Do you think these data are described well by a linear model?



By the plot, although there are some outliers, it seems that the data could be approximated by a linear model.

2. Consider model M_i in which Moisture is normally distributed, and the expected Moisture is a polynomial function of Protein including the polynomial terms up to power i (i.e M_1 is a linear model, M_2 is a quadratic model and so on). Report a probabilistic model that describes M_i. Why is it appropriate to use MSE criterion when fitting this model to a training data?

$$M_i = \sum_{j=0}^{i} \beta_j x^j + \varepsilon$$

$$i = 1, \cdots, 6$$

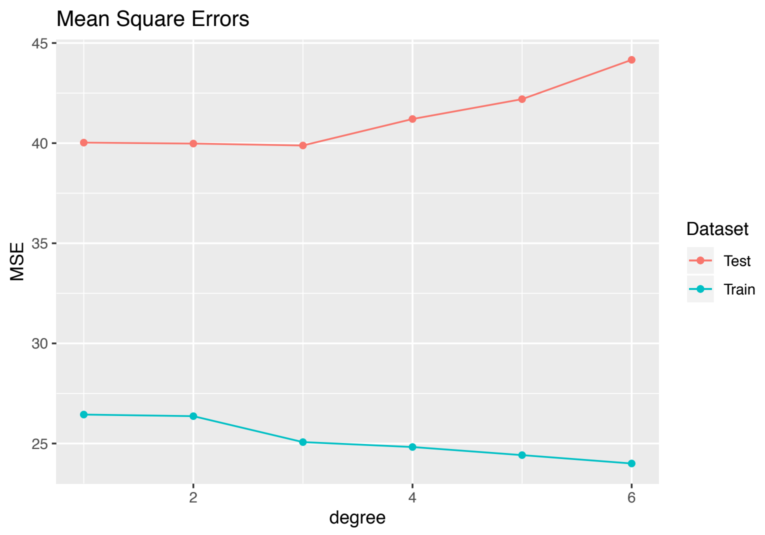
$$\varepsilon \sim N(\mu, \sigma^2)$$

\mu should be 0 here

With increasing model complexity (higher n polynomial degrees), the model tends to be overfitted, capturing noise from the training data. An overfitted model would produce inferior results (higher variance) when predicting for the test data. The MSE criterion therefore can tell how well fit (or under/overfit) the model is to the data.

But, so can mean absolute error or SSE. Why is MSE better?

3. Divide the data (50/50) and fit models $M_i, i = 1, \cdots, 6$. For each model, record the training and validation MSE and present a plot showing how training and validation MSE depend on i. Which model is best according to this plot? How do MSE values change and why? Interpret this picture in bias-variance tradeoff.



The plot shows a decreasing MSE score for the training data. This is expected because the more complex models fit the training data "better" by capturing its noise. By analysing the MSE score for the training data only, one could erroneously think that the more complex the model, the better. However, this is not always the case, as we can see in the graph.

High complexity models overfit the data and results in high variance and low bias. On the other hand, simplistic models have low variance and high bias. This MSE plot helps the choice of the most balanced model bias-variance wise. In this case, a 3rd degree polynomial model.

4. Perform variable selection of a linear model in which Fat is response and Channel1-Channel100 are predictors by using stepAIC. Comment on how many variables were selected.

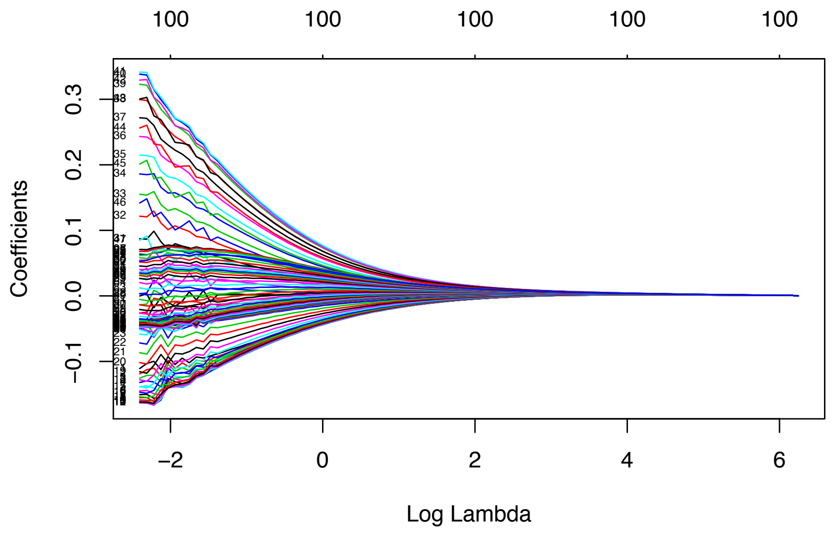
After running stepAIC we get that the amount of selected variables is :

```
## There were selected 64 variables
```

Thus, taking into account that one of them is the intercept, we have 63 selected variables out of 100.

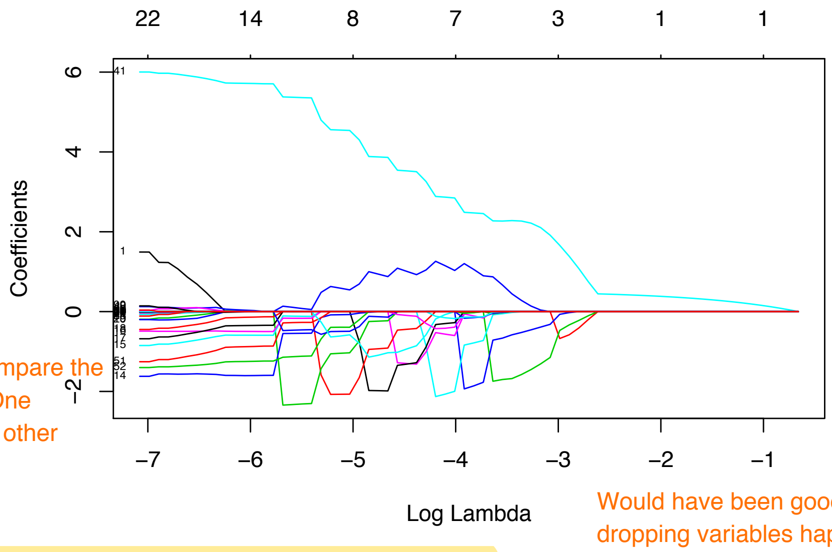
Good catch!

5. Fit a Ridge regression model with the same predictor and response variables. Present a plot showing how model coefficients depend on the log of the penalty factor λ and report how the coefficients change with λ



It can be seen that, when using Ridge regression, for increasing values of λ, the coefficients tend to zero, though the amount of parameters is still 100 since the penalty factor in this regression does not converge to zero, so no actual paramaters are being dropped. Convergence is not the same as equality.

6. Repeat step 5 but fit with LASSO instead of the Ridge regression and compare the plots from steps 5 and 6. Conclusions?



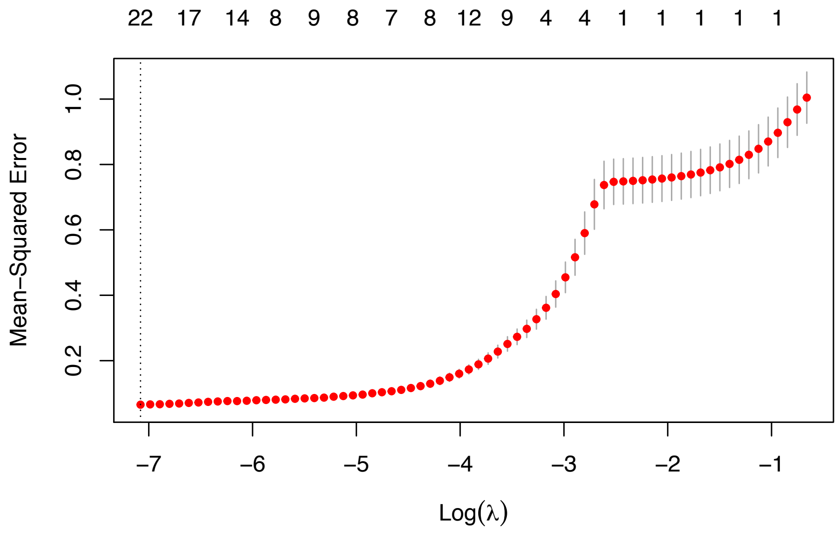
It doesn't make much sense to compare the lambda values for the 2 models. One penalizes the squared values and other penalizes the absolute values.

Would have been good if you explained why this kind of dropping variables happens.

LASSO converges to zero much faster than Ridge regression and it also drops variables.

7. Use cross-validation to find optimal LASSO model (make sure that case $\lambda = 0$ is also considered by the procedure), report the optimal λ and how many variables were chosen by the model and make conclusions. Present also a plot showing the dependence of the CV score and comment how the CV score changes λ

```
## The best performance was at lambda =  0

##
## Call:  cv.glmnet(x = as.matrix(covariates), y = response, lambda = lambdas,      alpha = 1, family =
##
## Measure: Mean-Squared Error
##
##         Lambda Measure       SE Nonzero
## min 0.0000000 0.05932 0.006345     100
## 1se 0.0008422 0.06538 0.006221      22
```



From the cross-validated model it is seen that the λ value for the minimum obtained MSE score is at $\lambda_{min} = 0$. The model has 100 non-zero parameters. An increase in $\log(\lambda)$ increases MSE.

You could also consider lambda.1se

8. Compare the results from steps 4 and 7.

For (4) after performing a stepAIC over the model, 36 variables were dropped, getting a final model with only 63 variables (plus the intercept). Nonetheless, for (7) after cross-validating the LASSO model it has been found that the best performance regarding MSE scores is at $\lambda = 0$ which implies no penalization, thus, no parameters will be dropped.

Appendix A - Code chunks

Code used for Assignment 1

1.1

```
#import and read data
data <- read.xlsx("./data/spambase.xlsx")
#dividing data
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
#train
train=data[id,]
#test
test=data[-id,]

confusion <- function(data, criteria, pred) {
    #criteria
    predicted <- as.integer(pred > criteria)

    #confusion matrix
    true <- data$Spam
    conf <- table(true, predicted)

    #misclassification rate
    total <- sum(conf)
    tp <- conf[2,2]
    tn <- conf[1,1]
    fp <- conf[1,2]
    fn <- conf[2,1]
    miss <- 100 * (fp + fn) / total

    #other rates
    true_positive <- 100 * tp/(tp + fp)
    true_negative <- 100 * tn/(tn + fn)
    false_positive <- 100 * fp/(tp + fp)
    false_negative <- 100 * fn/(tn + fn)

    df <- data.frame(Rate = c("True positive", "True negative", "False positive", "False negative", "Mi
                     Value = c(true_positive, true_negative, false_positive, false_negative, miss))

    print(df)
    return(conf)
}
```

1.2

```
#model
form <- Spam ~ .
model <- glm(formula = form, data = train, family = "binomial")
#train
pred <- predict(model, train, type = "response")
confusion(data = train, criteria = 0.5, pred = pred)
```

```
#test
pred <- predict(model, test, type = "response")
confusion(data = test, criteria = 0.5, pred = pred)
```

1.3

```
#train
pred <- predict(model, train, type = "response")
confusion(data = train, criteria = 0.8, pred = pred)
#test
pred <- predict(model, test, type = "response")
confusion(data = test, criteria = 0.8, pred = pred)
```

1.4

```
model <- train.kknn(formula = form, data = train, ks = 30)


#train
pred <- predict(model, train)
confusion(data = train, criteria = 0.5, pred = pred)
#test
pred <- predict(model, test)
confusion(data = test, criteria = 0.5, pred = pred)
```

1.5

```
model <- train.kknn(formula = form, data = train, ks = 1)

#train
pred <- predict(model, train)
confusion(data = train, criteria = 0.5, pred = pred)
#test
pred <- predict(model, test)
confusion(data = test, criteria = 0.5, pred = pred)
```

Code used for Assignment 3

```
#linear regression
mylin=function(X,Y, Xpred){
  X1=cbind(1,X)
  beta=solve(t(X1)%*%X1)%*%t(X1)%*%Y
  Xpred1=cbind(1,Xpred)
 #MISSING: check formulas for linear regression and compute beta
  Res=Xpred1%*%beta
  return(Res)
}

myCV=function(X,Y,Nfolds){
  n=length(Y)
  p=ncol(X)
  set.seed(12345)
```

```
ind=sample(n,n)
X1=X[ind,]
Y1=Y[ind]
sF=floor(n/Nfolds)
MSE=numeric(2^p-1)
Nfeat=numeric(2^p-1)
Features=list()
curr=0

#we assume 5 features.

for (f1 in 0:1)
  for (f2 in 0:1)
    for(f3 in 0:1)
      for(f4 in 0:1)
        for(f5 in 0:1){
          model= c(f1,f2,f3,f4,f5)
          if (sum(model)==0) next()
          SSE=0

          for (k in 1:Nfolds){
            #MISSING: compute which indices should belong to current fold
            current_feat=which(model==1)
            #MISSING: implement cross-validation for model with features in "model"
            #and iteration i.
            begin_pos=(k-1)*sF+1
            if(k==Nfolds){
              end_pos=length(Y1)
            }else{
              end_pos=k*sF
            }

            test_X=X1[begin_pos:end_pos,current_feat]

            train_X=X1[-begin_pos:-end_pos,current_feat]

            train_Y=Y1[-begin_pos:-end_pos]

            Ypred=mylin(train_X,train_Y,test_X)

            #MISSING: Get the predicted values for fold 'k', Ypred, and the original
            #values for folf 'k', Yp.
            Yp=Y1[begin_pos:end_pos]
            SSE=SSE+sum((Ypred-Yp)^2)
          }
          curr=curr+1
          MSE[curr]=SSE/n
          Nfeat[curr]=sum(model)
          Features[[curr]]=model
        }
#MISSING: plot MSE against number of features
library(ggplot2)
df<-data.frame(number=c(),MSE=c())
```

```r
  for(i in 1:length(Features)){
    tmp=data.frame(number=sum(Features[[i]]),MSE=MSE[i])
    df=rbind(df,tmp)
  }
  plot1<-ggplot(df,aes(x=number,y=MSE))+geom_point(shape=21)
  #return(plot1)
  i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]],plot=plot1))
}

myCV(as.matrix(swiss[,2:6]), swiss[[1]], 5)
```

Code used for Assignment 4

4.1

```r
## Import data and plot Moisture vs Protein.
data <- read.xlsx("data/tecator.xlsx")
plot(data$Protein, data$Moisture)
```

4.3

```r
# Creating data sets
p <- data$Protein
y <- data$Moisture
P <- data.frame(
    Y = y,
    P1 = p,
    P2 = p^2,
    P3 = p^3,
    P4 = p^4,
    P5 = p^5,
    P6 = p^6
)

n = dim(P)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
train = P[id,]
test = P[-id,]

# Training models
M1 <- lm(Y ~ ., data = train[,1:2])
M2 <- lm(Y ~ ., data = train[,1:3])
M3 <- lm(Y ~ ., data = train[,1:4])
M4 <- lm(Y ~ ., data = train[,1:5])
M5 <- lm(Y ~ ., data = train[,1:6])
M6 <- lm(Y ~ ., data = train[,1:7])

## Train Scores
eval_model <- function(model, data) {
  pred <- predict(model, data)
  errors <- pred - data$Y
  MSE <- mean(errors^2)
  return(MSE)
}

MSE_train <- c()
MSE_train[1] <- eval_model(M1, train)
MSE_train[2] <- eval_model(M2, train)
MSE_train[3] <- eval_model(M3, train)
MSE_train[4] <- eval_model(M4, train)
MSE_train[5] <- eval_model(M5, train)
MSE_train[6] <- eval_model(M6, train)

MSE_test <- c()
MSE_test[1] <- eval_model(M1, test)
MSE_test[2] <- eval_model(M2, test)
MSE_test[3] <- eval_model(M3, test)
MSE_test[4] <- eval_model(M4, test)
MSE_test[5] <- eval_model(M5, test)
MSE_test[6] <- eval_model(M6, test)

df <- data.frame(
    degree <- c(1:6),
    MSE_train,
    MSE_test
)
p <- ggplot()
p <- p +geom_point(data = df, aes( x = degree, y = MSE_train, color="Train")) +
    geom_line(data = df, aes( x = degree, y = MSE_train, color="Train"))
p <- p + geom_point(data = df, aes( x = degree, y = MSE_test, color="Test")) +
    geom_line(data = df, aes( x = degree, y = MSE_test, color="Test"))
p <- p + labs(y="MSE", colour="Dataset", title = "Mean Square Errors") +
      geom_line()
p
```

4.4

```
adhok_data <- data[,2:101]
Fat <- data$Fat
adhok_data <- cbind(adhok_data, Fat)
model <- lm(Fat ~ . , data=adhok_data)
step <- stepAIC(model, direction ="both")
selected_vars <- step$coefficients
cat("There were selected", length(selected_vars), "variables\n")
```

4.5

```
covariates <- scale(adhok_data[1:(length(adhok_data)-1)])
response <- scale(adhok_data$Fat)
model_ridge <- glmnet(covariates, response, alpha = 0, family = "gaussian")
plot(model_ridge, xvar="lambda", label=T)
```

4.6

```r
model_lasso <- glmnet(covariates, response, alpha = 1, family = "gaussian")
plot(model_lasso, xvar="lambda", label=T)
```

4.7

```r
lambdas <- append(model_lasso$lambda,0)
cv_model_lasso <- cv.glmnet(as.matrix(covariates), response, alpha=1,
                            family="gaussian", lambda=lambdas)
cat("The best performance was at lambda = ", cv_model_lasso$lambda.min, "\n")
cv_model_lasso
plot(cv_model_lasso)
```

Your report looks really good and nice explanations!

LAB 2 BLOCK 1 - Group A15

*Zuxiang Li (**zuxli371**), Marcos F. Mourao (**marfr825**), Agustín Valencia (**aguva779**)*

08 December 2019

Assignment 2: Analysis of credit scoring

1. Import the data to R and divide into training/validation/test as 50/25/25: use data partitioning code specified in Lecture 1e.

The data was divided as requested and all code used is in the appendix.

2. Fit a decision tree to the training data by using the following measures of impurity: *deviance and Gini index*, and report the misclassification rates for the training and test data. Choose the measure providing the better results for the following steps.

Misclassification rates for **deviance** impurity measure:

```
## [1] "Misclassification rate for TRAINING data:  21.2 %."
```

```
## [1] "Misclassification rate for TESTING data:  26.8 %."
```

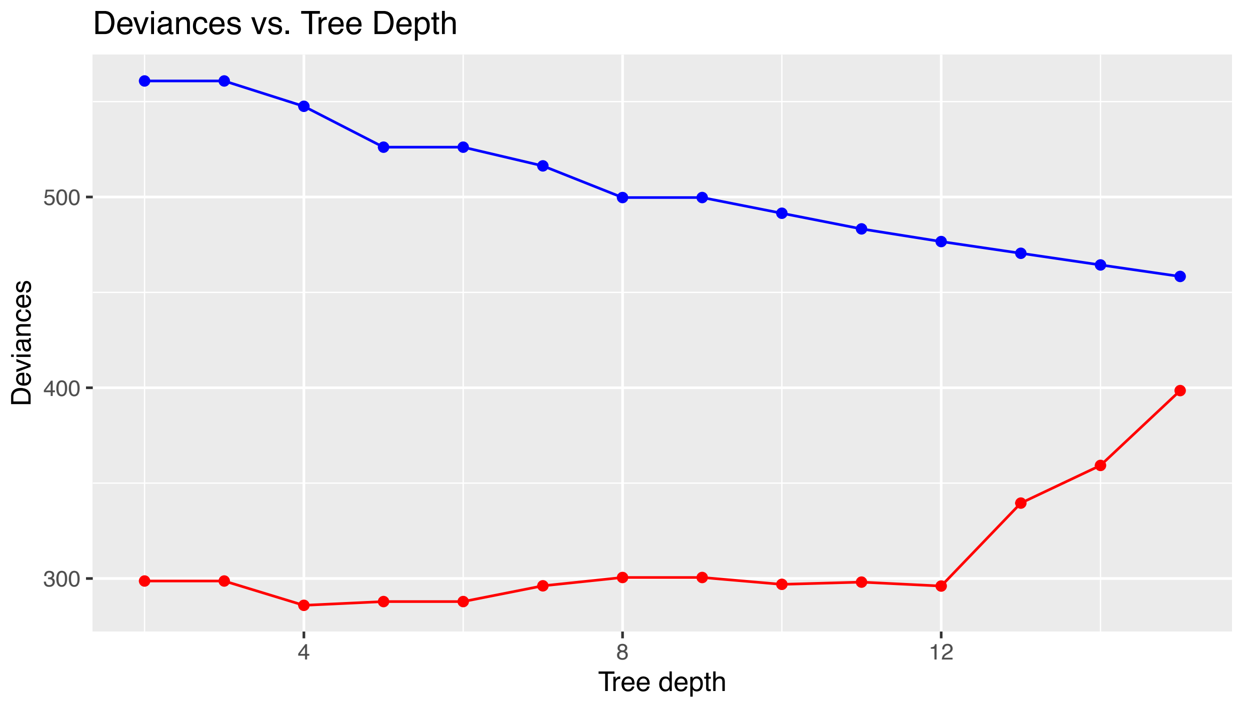
Misclassification rates for **Gini's index** impurity measure:

```
## [1] "Misclassification rate for TRAINING data:  23.8 %."
```

```
## [1] "Misclassification rate for TESTING data:  37.2 %."
```

Altough showing similar results, we choose the deviance measure as it has a lower misclassification rate for both training (21.2% against 23.8%) and testing data (26.8% against 37.2%).

3. Use training and validation sets to choose the optimal tree depth. Present the graphs of the dependence of deviances for the training and the validation data on the number of leaves. Report the optimal tree, report it's depth and the variables used by the tree. Interpret the information provided by the tree structure. Estimate the misclassification rate for the test data.

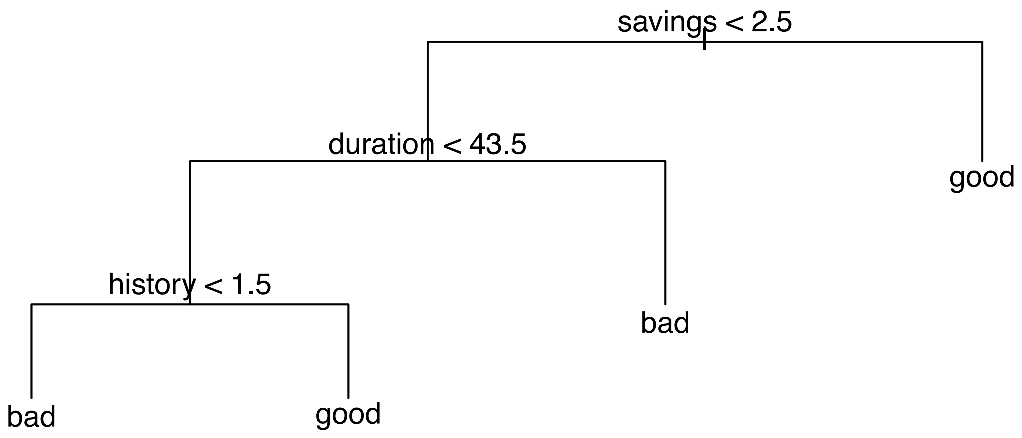


The deviance plot suggests that an optimal tree depth (minimal deviance in testing) of 4 should be used. Deeper trees perform better on training, but worse in testing due to overfitting. The optimal tree is shown below. The misclassification rate of the optimal tree on the testing data is 25.6%, a better result compared to item 2 (26.8%).

The optimal tree uses the variables: `savings`, `duration` and `history`. The root node, `savings`, indicates that costumers that save more are more likely to pay their loans. `duration` implies that loans that take too long to be paid are more likely to never be actually totally paid. Finally, `history` says that older costumers within the company are more likely to pay their loans.

It is important to notice that in the data, out of 1000 entries, 700 of them are classified as "good", and only 300 are classified as "bad". Such disparity will impact on the impurity of leafs, specially in a shallow tree.

Good explanations!



```
##       predicted
## true   bad good
##   bad   18   58
##   good   6  168
```

```
## [1] "Misclassification rate of the OPTIMAL TREE for TESTING data:  25.6 %."
```

4. Use training data to perform classification using Naïve Bayes and report the confusion matrices and misclassification rates for the training and for the test data. Compare the results with those from step 3.

```
##       predicted
## true   bad good
##   bad   95   52
##   good  98  255
```

```
## [1] "Naive Bayes misclassification rate for TRAINING data:  30 %."
```

```
##       predicted
## true   bad good
##   bad   46   30
##   good  49  125
```

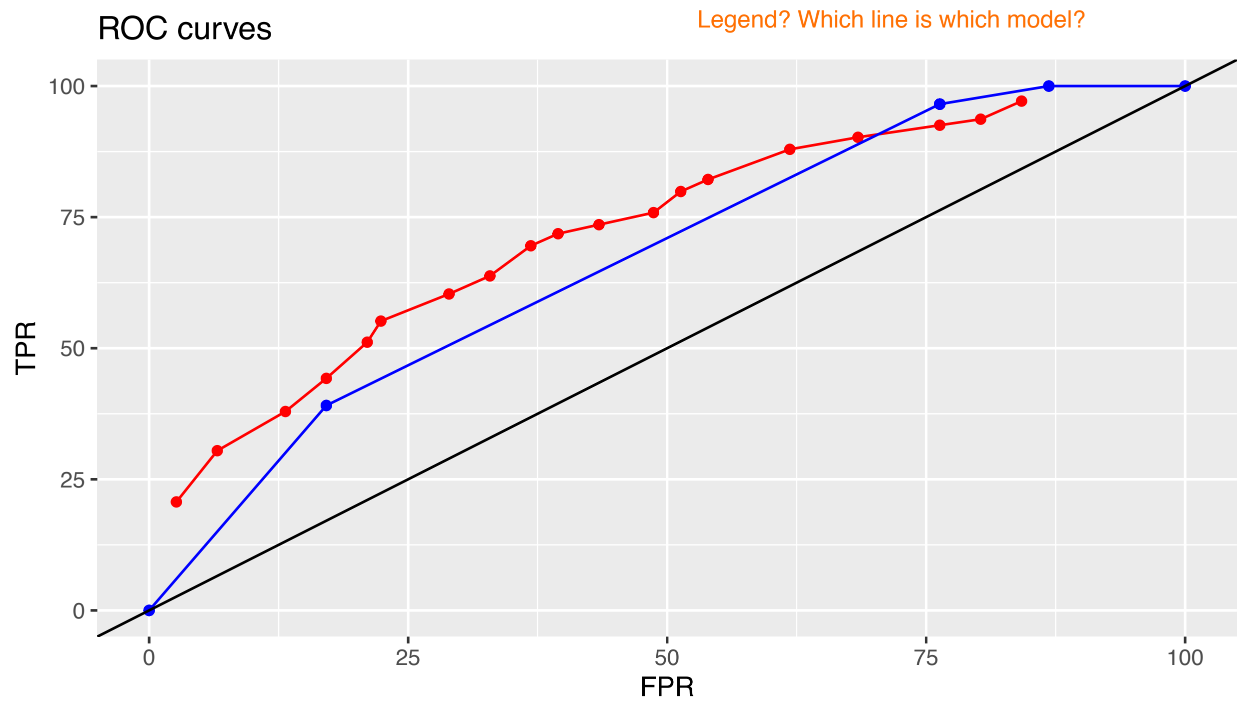
```
## [1] "Naive Bayes misclassification rate for TESTING data:  31.6 %."
```

The Naive Bayes classifier perfomerd worse than the optimal tree in both test and training, with higher misclassification rates.

5. Use the optimal tree and the Naïve Bayes model to classify the test data by using the following principle: $\hat{Y} = 1$ if $p(Y = good|X) > \pi$, otherwise $\hat{Y} = 0$ where $\pi = 0.05, 0.1, 0.15, ..., 0.9, 0.95$. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion?

The ROC curves for the optimal tree and Naive Bayes model are shown below:

Legend? Which line is which model?



The ROC curves indicate that the Naive Bayes classifier performs better (it is closer to the perfect classification point (100,0)). Note - higher area under the curve.

6. Repeat Naïve Bayes classification as it was in step 4 but use the following loss matrix:

$$L = \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix}$$

and report the confusion matrix for the training and test data. Compare the results with the results from step 4 and discuss how the rates has changed and why.

```
##       predicted
## true     0   1
##   bad   27 120
##   good  17 336
```

These results are wrong for the custom loss matrix.

```
## [1] "Naive Bayes misclassification rate for TRAINING data:  27.4 %."
```

```
##       predicted
## true     0   1
##   bad   14  62
##   good  10 164
```

```
## [1] "Naive Bayes misclassification rate for TESTING data:  28.8 %."
```

The given loss matrix penalizes false positives ten times more than false negatives. The confusion matrix produced by this new penalization presents, of course, less false positives. The misclassification rates also were reduced for both training and testing data.

In this particular setting, a false positive is indeed "worse" than a false negative. Classifying a costumer as "good" when he/she actually does not pay his/her loan ("bad") incurs in greater financial loss to the company than classifying a "good" costumer as "bad" and not lend him/her money. This model, therefore, is preffered over the one proposed in item 4.

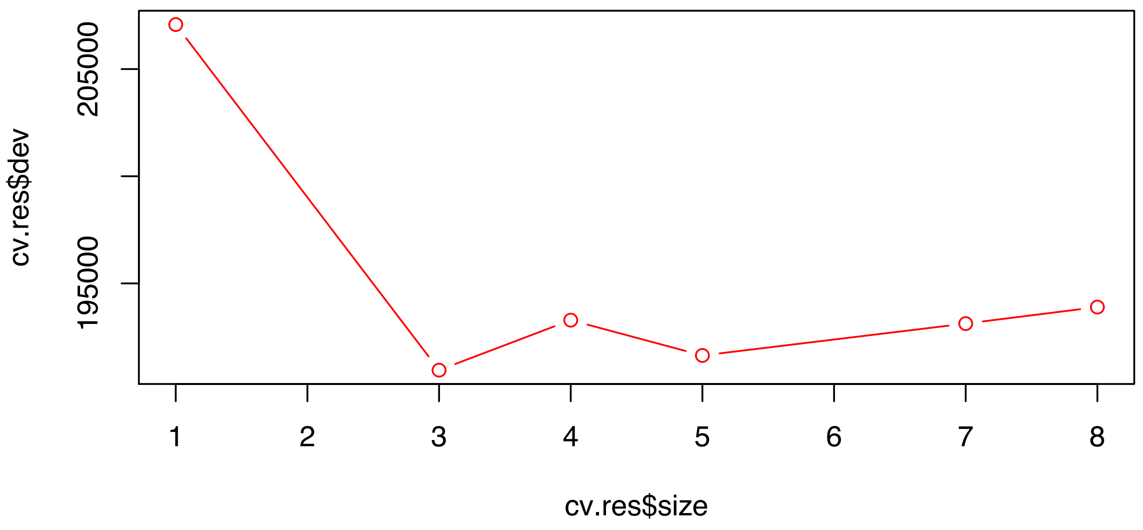
Assignment 3. Uncertainty estimation

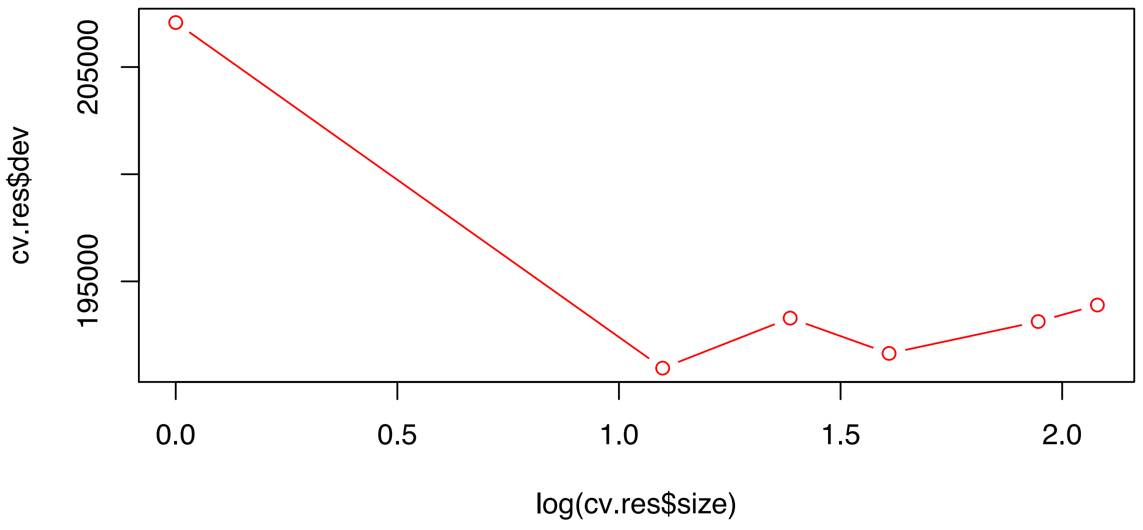
The data file State.csv contains per capita state and local public expenditures and associated state demographic and economic characteristics, 1960, and there are variables

- MET: Percentage of population living in standard metropolitan areas
- EX: Per capita state and local public expenditures ($)

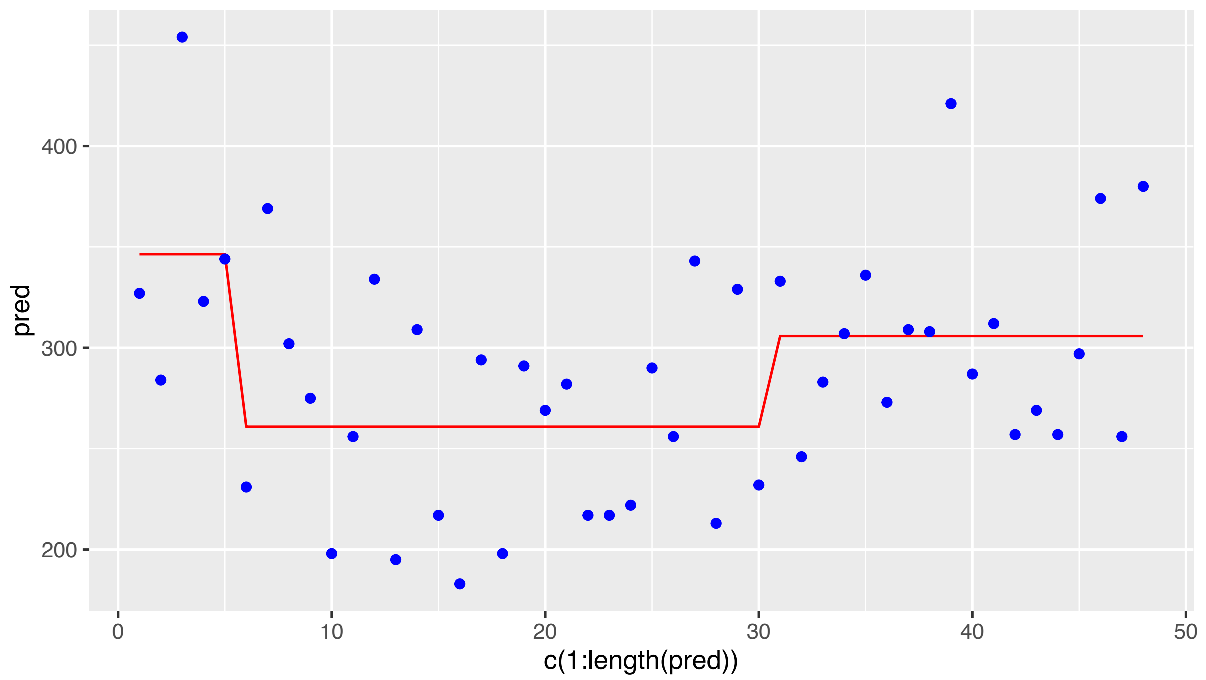
1. Reorder your data with respect to the increase of MET and plot EX versus MET. Discuss what kind of model can be appropriate here. Use the reordered data in steps 2-5.

2. Use package tree and fit a regression tree model with target EX and feature MET in which the number of the leaves is selected by cross-validation, use the entire data set and set minimum number of observations in a leaf equal to 8 (setting minsize in tree.control). Report the selected tree. Plot the original and the fitted data and histogram of residuals. Comment on the distribution of the residuals and the quality of the fit.

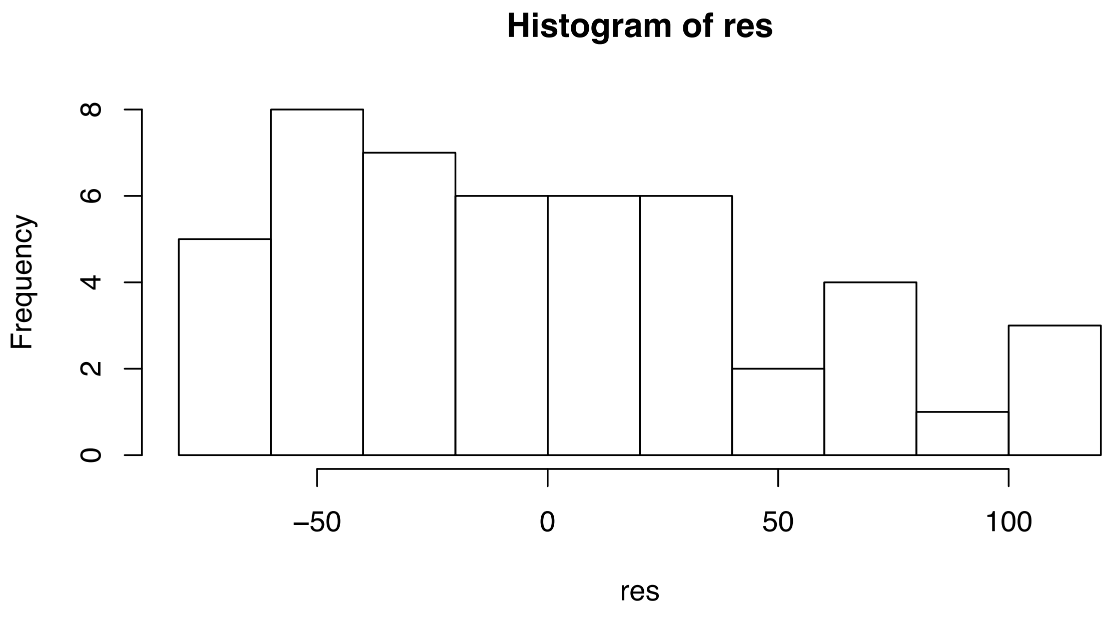




205000
195000
cv.res$dev
0.0
0.5
1.0
1.5
2.0
log(cv.res$size)



400
300
200
pred
0
10
20
30
40
50
c(1:length(pred))



```
## [1] 1.184223e-14
```

```
## [1] 50.82183
```

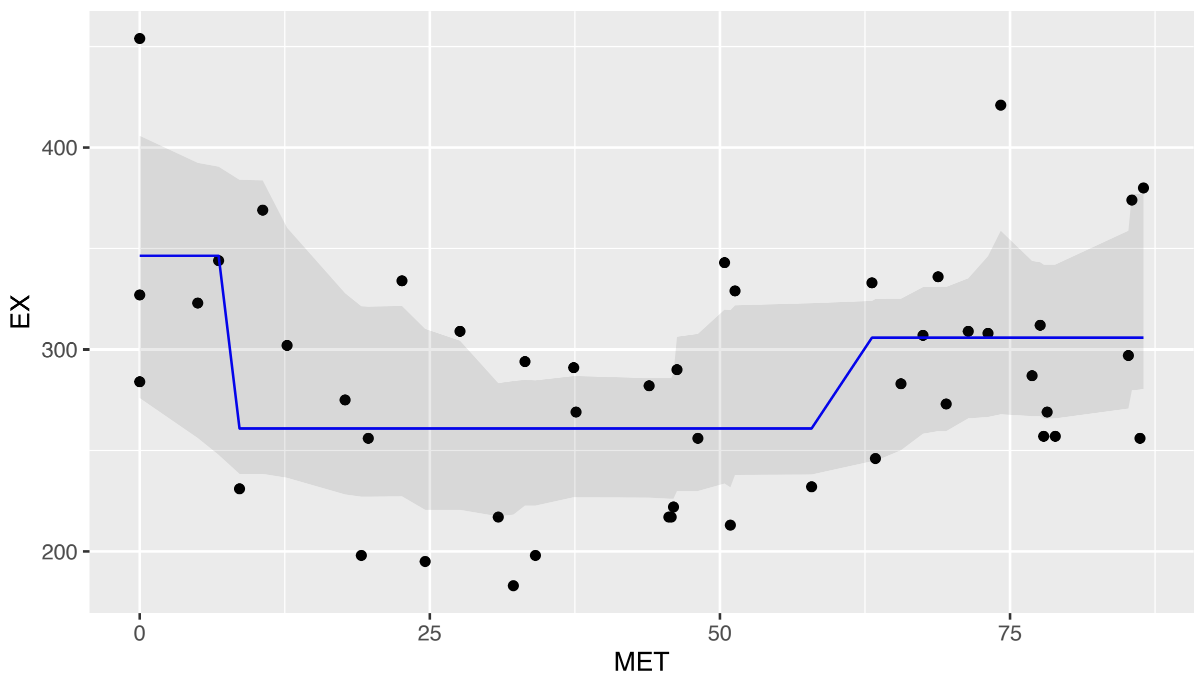
I don't think your residuals are normally distributed based on above plot.

The selected tree should be 3 in this case, the residuals distributed as normal distribution with mean 0 and standard deviance 50.8. The histogram of residuals suggest it is slightly skewed to the right, which means it's a good fit initially.

This is not correct. Negative error is also still error.

3. Compute and plot the 95% confidence bands for the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a non-parametric bootstrap. Comment whether the band is smooth or bumpy and try to explain why. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable.

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



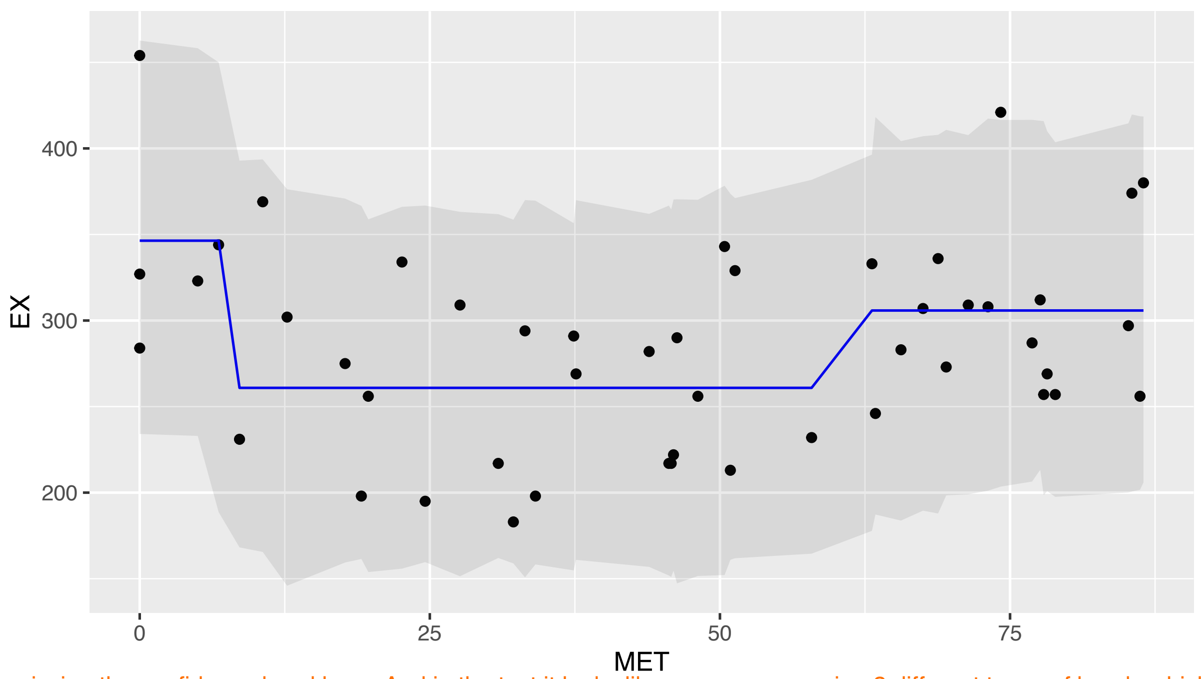
From the plot, we can see that the confidence band is quite bumpy and almost half of the dots are outside of the confidence band, which indicate that it's not a reliable model. The band is bumpy because nonparametric bootstrap does not work well for discrete estimators.

The confidence band is for the mean of your predictions - you need not expect most of your data points to lie within this band.

4. Compute and plot the 95% confidence and prediction bands the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a parametric bootstrap, assume are labels in the tree leaves and is the residual variance. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable. Does it look like only 5% of data are outside the prediction band? Should it be?

```
## Warning in envelope(boot_res1, level = 0.95): unable to achieve requested
## overall error rate
```

If your data has less noise, then the confidence band is going to be narrower and still a part of the data points are going to lie outside of this band.



You are missing the confidence band here. And in the text it looks like you are comparing 2 different types of bands which is wrong.

From the plot we can see that there are two points are outside the confidence band, it's more reliable than previous models.In addition, the sample with small size using nonparametric bootstrap can underestimate the amount of variation, but using parametric bootstrap which take values from known distribution covering a wider range.

Good that you thought about this!

5. Consider the histogram of residuals from step 2 and suggest what kind of bootstrap is actually more appropriate here.

Parametric bootstrap is more suitble here since the histrogram of residuals looks like normal distribution.

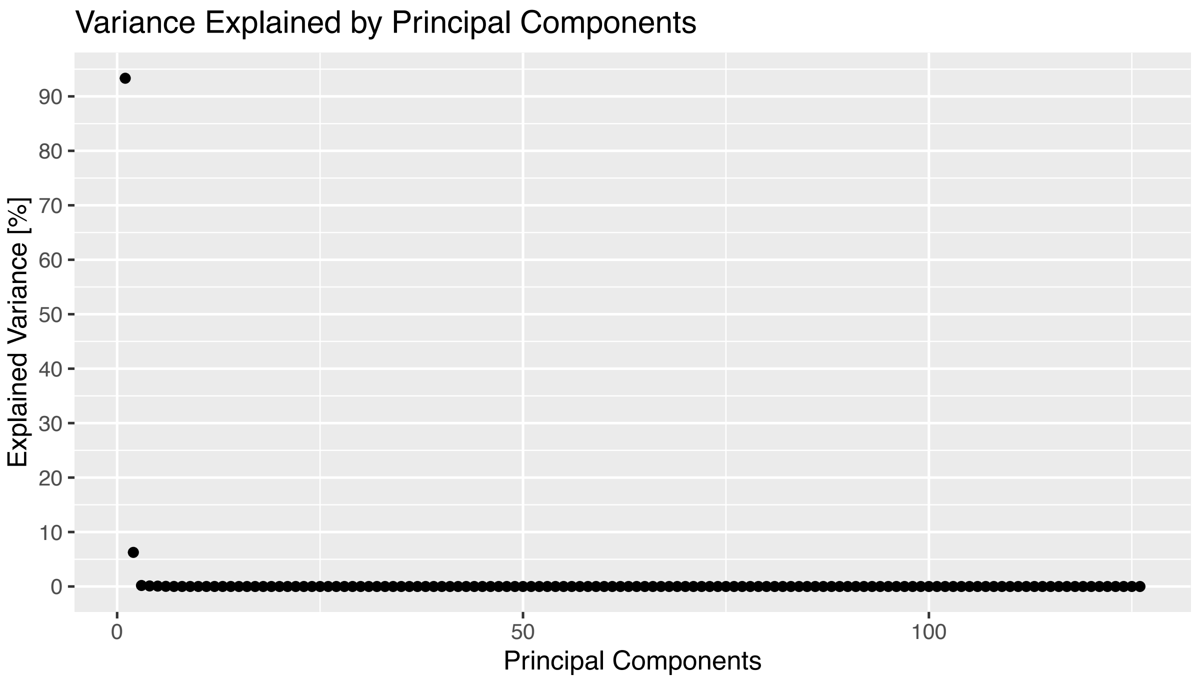
Really? :D You need to look up how normal distribution looks.

Assignment 4 - Principal components

The data file NIRspectra.csv contains near-infrared spectra and viscosity levels for a collection of diesel fuels. Your task is to investigate how the measured spectra can be used to predict the viscosity.

1. Conduct a standard PCA by using the feature space and provide a plot explaining how much variation is explained by each feature. Does the plot show how many PC should be extracted? Select the minimal number of components explaining at least 99% of the total variance. Provide also a plot of the scores in the coordinates (PC1, PC2). Are there unusual diesel fuels according to this plot?

According to the following plot very few Principal Components explain most of the variance, which implies that the data set dimension could be reduced from over a hundred to just a few variables.

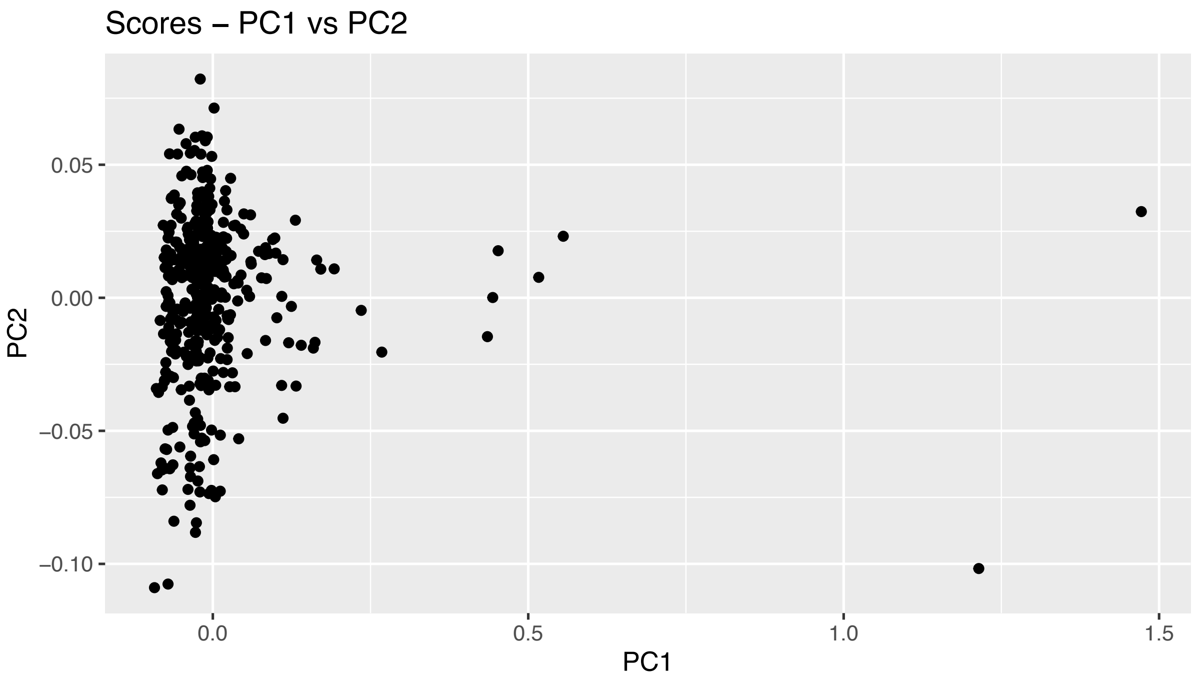


The printout below (close to zero values have been omitted) confirms that, after sorting their contributions, just the two first PCs represent 99.6% of the total variance. So the dimensionality could be reduced from 127 dimensions to just two.

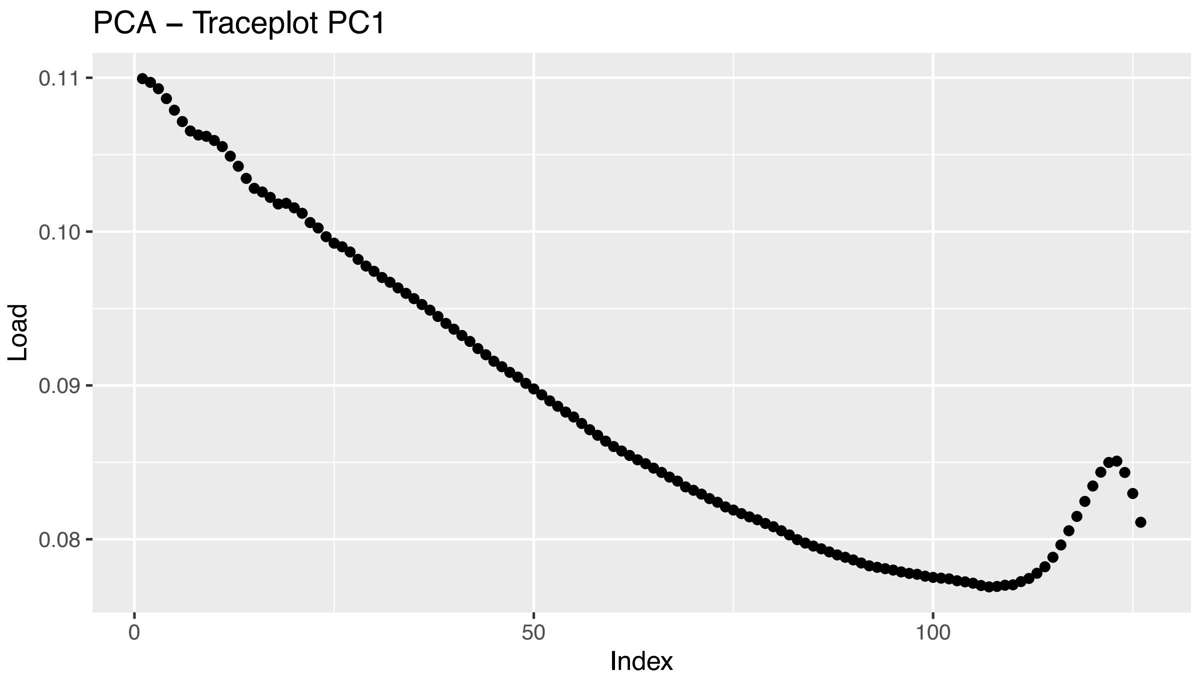
```
## [1] "Principal Components Contributions"
```

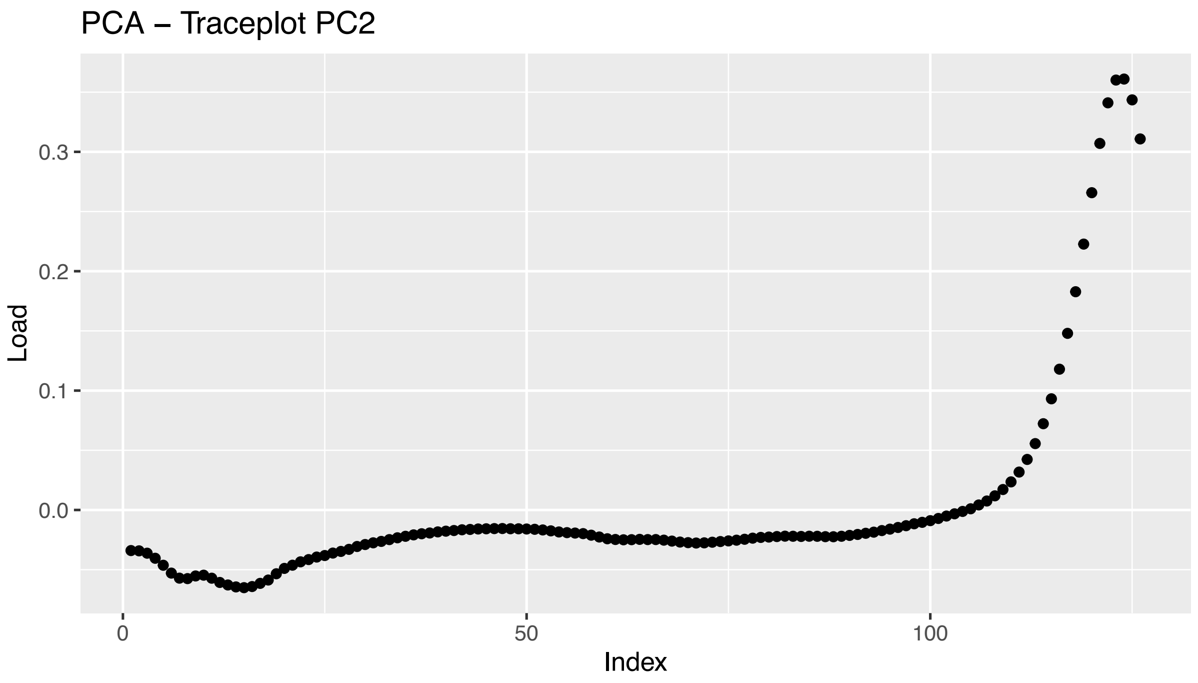
```
##  [1] "93.332" "6.263"  "0.185"  "0.101"  "0.068"  "0.025"  "0.009"  "0.003"
##  [9] "0.003"  "0.002"  "0.001"  "0.001"  "0.001"  "0.001"  "0.000"  "0.000"
```

The following plot show how the data is explained when the dimensionality is reduced to two principal components out of 127. Since these two new features are a linear combination of the 127 original characteristics, it looses interpretability, though for computational purposes it implies a good improvement. It can be seen some anormalities along the PC1



2. Make trace plots of the loadings of the components selected in step 1. Is there any principle component that is explained by mainly a few original features?





It can be seen that the two components are comprised by a linear combination of all the original variables. Although for the first component, the contributions come mainly for the first variables, and in the second the opposite, no weight represents more than a half of all the weights, thus it cannot be said that there is a principal component being represented mainly by one in the original space.

Maybe look closer at the weights for the 2nd component.

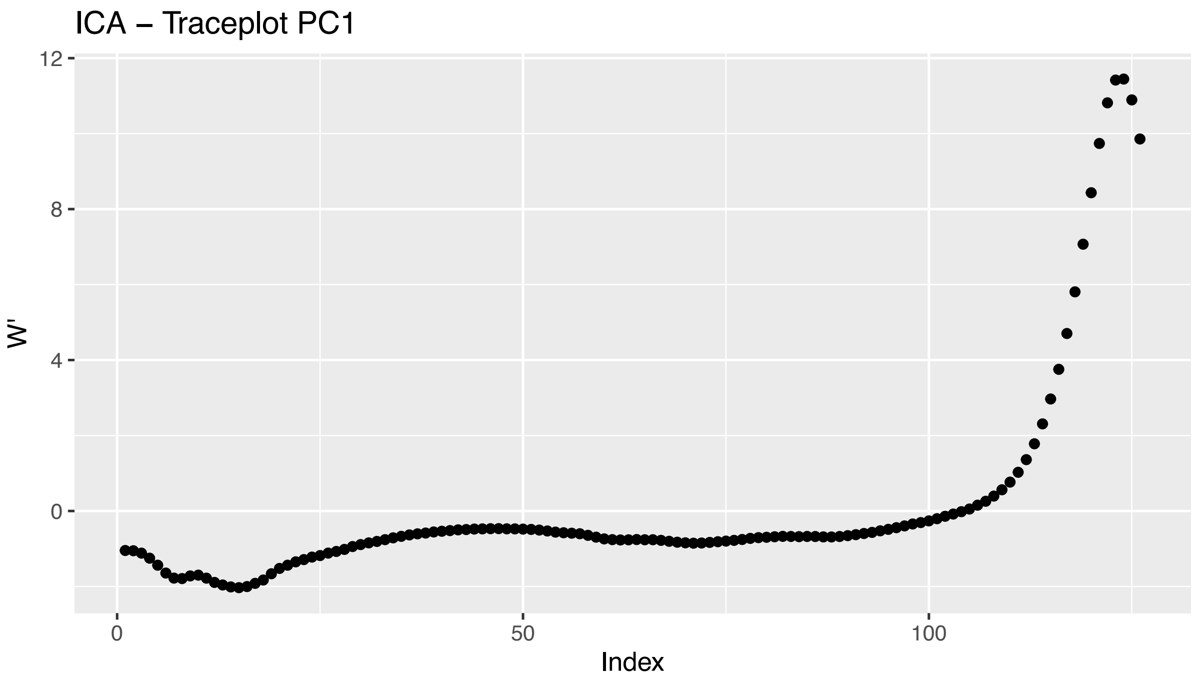
3. Perform Independent Component Analysis with the number of components selected in step 1 (set seed 12345). Check the documentation for the fastICA method in R and do the following:

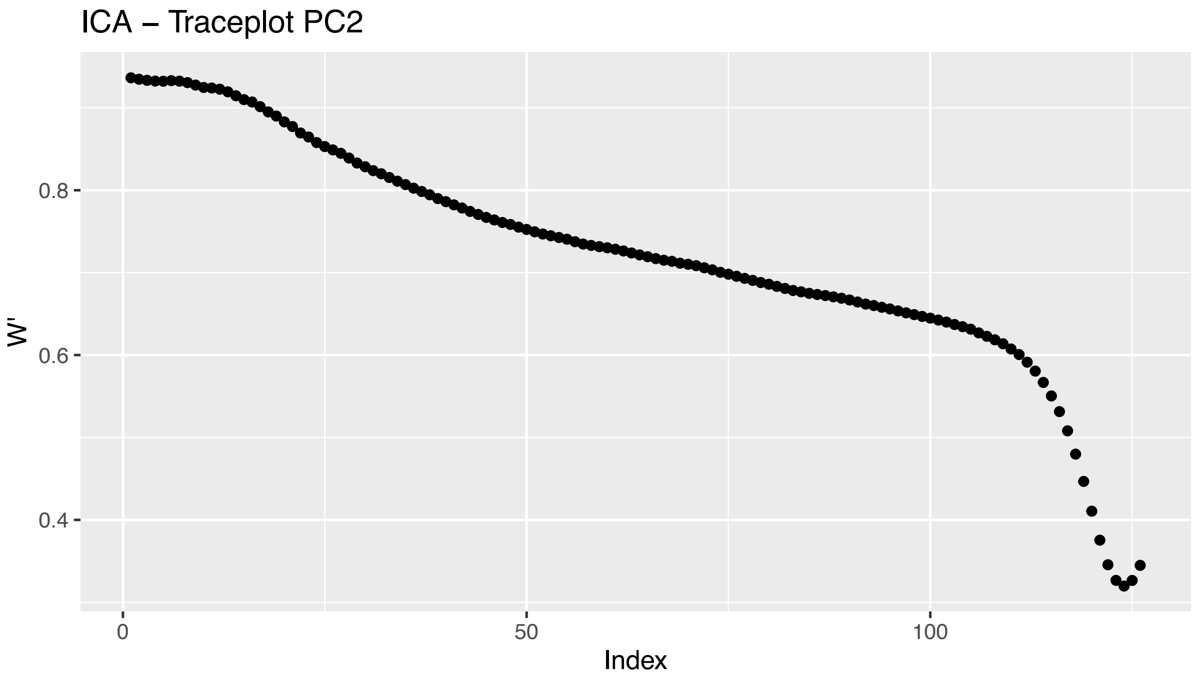
a. Compute $W' = K \cdot W$ and present the columns of W' in form of the trace plots. Compare with the trace plots in step 2 and make conclusions. What kind of measure is represented by the matrix W'?

W matrix contains the unmixing coefficients. It can be seen that the traceplots from it are somewhat similar but in a mirrored fashion. This will allow to *undo* the compression and get an approximation of the original variables before PCA.

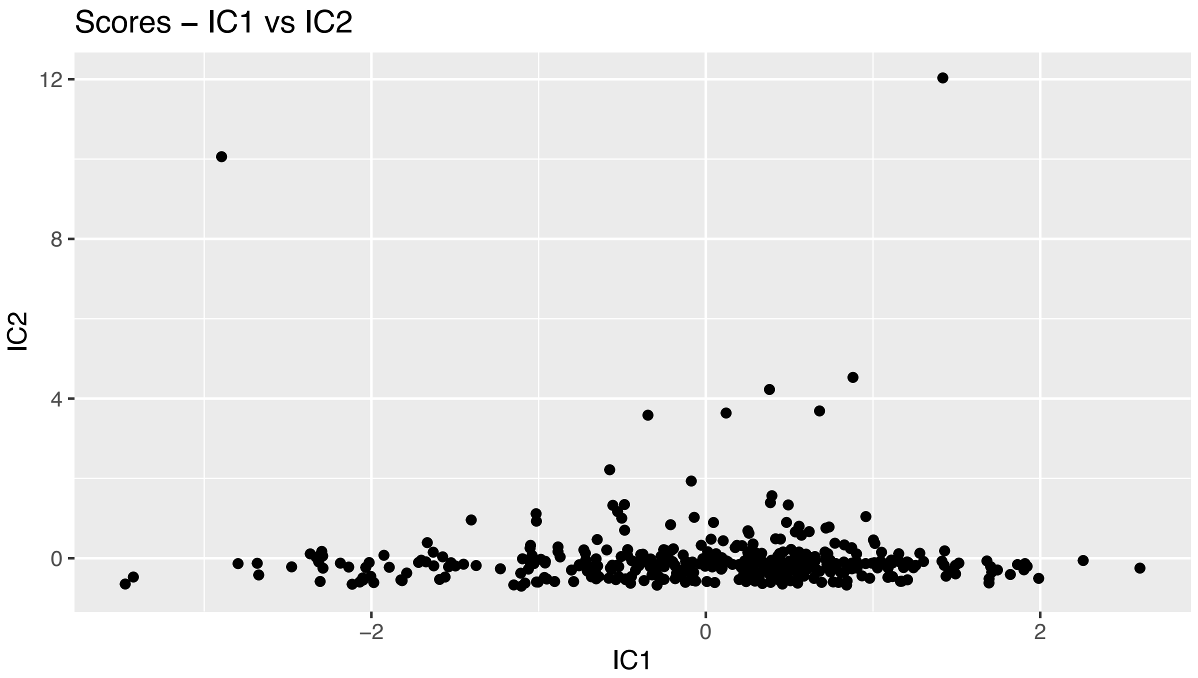
You are asked about W' and not W.

Unmixing is fine but the rest doesn't sound correct.
It just projects the values to the new components.


ICA – Traceplot PC1
W'
Index
12
8
4
0
0
50
100


ICA – Traceplot PC2
W'
Index
0.8
0.6
0.4
0
50
100

b. Make a plot of the scores of the first two latent features and compare it with the score plot from step 1.



It can be seen that the scores looks rotated when compared to the PCA scores.

Code Appendix

Assignment 2

2.1

```
#1. importing data
data_path <- "./Data/creditscoring.xls"
library(readxl)
data <- read_xls(data_path)
#changing to factor in order to trees to work
data$good_bad <- as.factor(data$good_bad)

#1. dividing into test, train and validation
n=dim(data)[1]
id=sample(1:n, floor(n*0.5))
train=data[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]

id3=setdiff(id1,id2)
test=data[id3,]
```

2.2

```
#2. Fit decision trees
form <- good_bad ~ .

#deviance TRAINING
fit_dev <- tree(form, train, split = "deviance")
# plot(fit_dev)
# text(fit_dev, pretty = 0)
# fit_dev
# summary(fit_dev)
Ypred <- predict(fit_dev, newdata = train, type = "class")
confusion <- table(true = train$good_bad, predicted = Ypred )
#confusion
miss <- round((100 * (confusion[1,2] +
                          confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate for TRAINING data: ", miss , "%.")

#deviance TESTING
Ypred <- predict(fit_dev, newdata = test, type = "class")
confusion <- table(true = test$good_bad, predicted = Ypred )
#confusion
miss <- round((100 * (confusion[1,2] +
                          confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate for TESTING data: ", miss , "%.")

#gini TRAINING
fit_gini <- tree(form, train, split = "gini")
```

```r
# plot(fit_gini)
# text(fit_gini, pretty = 0)
# fit_gini
# summary(fit_gini)
Ypred <- predict(fit_gini, newdata = train, type = "class")
confusion <- table(true = train$good_bad, predicted = Ypred )
#confusion
miss <- round((100 * (confusion[1,2] +
                          confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate for TRAINING data: ", miss , "%.")

#gini TESTING
Ypred <- predict(fit_gini, newdata = test, type = "class")
confusion <- table(true = test$good_bad, predicted = Ypred )
#confusion
miss <- round((100 * (confusion[1,2] +
                          confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate for TESTING data: ", miss , "%.")
```

2.3

```r
fit <- tree(form, data = train, split = "deviance")
trainScore <- rep(0,15)
validScore <- rep(0,15)

#get deviances for different number of leafs
for (i in 2:15) {
    prunedTree <- prune.tree(fit, best = i)
    pred <- predict(prunedTree, newdata = valid, type = "tree")
    trainScore[i] <- deviance(prunedTree)
    validScore[i] <- deviance(pred)
}

#deviance plot
ggplot()+
    geom_line(aes(x = 2:15, y = trainScore[2:15]), col = "blue") +
    geom_point(aes(x = 2:15, y = trainScore[2:15]), col = "blue") +
    geom_line(aes(x = 2:15, y = validScore[2:15]), col = "red") +
    geom_point(aes(x = 2:15, y = validScore[2:15]), col = "red") +
    ggtitle(label = "Deviances vs. Tree Depth" ) +
    ylab("Deviances") +
    xlab("Tree depth")


# plot(x = 2:12, y = trainScore[2:12], type = "b",
#main = "Deviances vs. Leaf count", xlab = "leaf count", ylab = "Deviance", col = "red")
# plot(x = 2:12, y = validScore[2:12], type = "b",
#main = "Deviances vs. Leaf count", xlab = "leaf count", ylab = "Deviance", col = "blue")

#optimal tree
optimalTree <- prune.tree(fit, best = 4)
#summary(optimalTree)
plot(optimalTree)
```

```r
text(optimalTree, pretty = 0)
Ypred_opt <- predict(optimalTree, newdata = test, type = "class")
confusion <- table(true = test$good_bad, predicted = Ypred_opt)
confusion
miss <- round((100 * (confusion[1,2] + confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate of the OPTIMAL TREE for TESTING data: ", miss , "%.")
```

2.4

```r
naive_fit <- naiveBayes(form, data = train)
#training data
Ypred_naive_train <- predict(naive_fit, newdata = train)
confusion_naive_train <- table(true = train$good_bad, predicted = Ypred_naive_train)
confusion_naive_train
miss_naive_train <- round((100 * (confusion_naive_train[1,2] +
                                      confusion_naive_train[2,1]) /
                               sum(confusion_naive_train)), digits = 2)
paste("Naive Bayes misclassification rate for TRAINING data: ", miss_naive_train, "%.")

#test data
Ypred_naive_test <- predict(naive_fit, newdata = test)
confusion_naive_test <- table(true = test$good_bad, predicted = Ypred_naive_test)
confusion_naive_test
miss_naive_test <- round((100 * (confusion_naive_test[1,2] +
                                     confusion_naive_test[2,1]) /
                              sum(confusion_naive_test)), digits = 2)
paste("Naive Bayes misclassification rate for TESTING data: ", miss_naive_test , "%.")
```

2.5

```r
prob_naive <- predict(naive_fit, newdata = test, type = "raw")
prob_opt <- predict(optimalTree, newdata = test)
pi <- seq(0.05, 0.95, 0.05)
#initializing
TPR_naive <- vector(length = length(pi))
TPR_opt <- vector(length = length(pi))
FPR_naive <- vector(length = length(pi))
FPR_opt <- vector(length = length(pi))
Yhat_naive <- data.frame(matrix(nrow=length(test$good_bad), ncol= length(pi)))
Yhat_opt <- data.frame(matrix(nrow=length(test$good_bad), ncol= length(pi)))

for (i in 1:length(pi)){
    for(j in 1:length(test$good_bad)){
        Yhat_naive[j,i] <- as.integer(prob_naive[j,2] > pi[i])
        Yhat_opt[j,i] <- as.integer(prob_opt[j,2] > pi[i])
    }
}


for(k in 1:length(pi)){
    #naive bayes
    conf1 <- table(test$good_bad, factor(Yhat_naive[,k], levels = c(0,1)))
    FPR_naive[k] <- 100*conf1[1,2] / (conf1[1,2] + conf1[1,1])
    TPR_naive[k] <- 100*conf1[2,2] / (conf1[2,1] + conf1[2,2])

    #optimal tree
    conf2<- table(test$good_bad, factor(Yhat_opt[,k], levels = c(0,1)))
    FPR_opt[k] <- 100*conf2[1,2] / (conf2[1,2] + conf2[1,1])
    TPR_opt[k] <- 100*conf2[2,2] / (conf2[2,1] + conf2[2,2])
}

#ROC curves
#naive bayes
ggplot() +
    geom_line(aes(x = FPR_naive, y = TPR_naive), col = "red") +
    geom_point(aes(x = FPR_naive, y = TPR_naive), col = "red") +
    geom_line(aes(x = FPR_opt, y = TPR_opt), col = "blue") +
    geom_point(aes(x = FPR_opt, y = TPR_opt), col = "blue") +
    geom_abline(slope = 1, intercept = 0, col = "black") +
    ggtitle(label = "ROC curves" ) +
    ylab("TPR") +
    xlab("FPR")



# plot(x = FPR_naive, y = TPR_naive, type = "b",
#main = "ROC - Naive Bayes", col = "red")
# #optimal tree
# plot(x = FPR_opt, y = TPR_opt, type = "b",
#main = "ROC - Optimal tree", col = "blue")
```

2.6

```
#formula
form <- good_bad ~ .
#model
naive_fit <- naiveBayes(form, data = train)
#loss matrix
L <- matrix(c(0,10,1,0), nrow = 2)


#TRAINING
prob_naive_train <- predict(naive_fit, newdata = train, type = "raw")
Yhat_train <- vector(length = dim(prob_naive_train)[1])
for (i in 1:dim(prob_naive_train)[1]){
    if(prob_naive_train[i,1]/prob_naive_train[i,2] > L[2,1]/L[1,2]){
        Yhat_train[i] <- 0
    } else {
        Yhat_train[i] <- 1
    }
}

confusion_naive_train <- table(true = train$good_bad, predicted = Yhat_train)
confusion_naive_train
miss_naive_train <- round((100 * (confusion_naive_train[1,2] +
```

```r
                                       confusion_naive_train[2,1]) /
                              sum(confusion_naive_train)), digits = 2)
paste("Naive Bayes misclassification rate for TRAINING data: ", miss_naive_train, "%.")


#TESTING
prob_naive_test <- predict(naive_fit, newdata = test, type = "raw")
Yhat_test <- vector(length = dim(prob_naive_test)[1])
for (i in 1:dim(prob_naive_test)[1]){
    if(prob_naive_test[i,1]/prob_naive_test[i,2] > L[2,1]/L[1,2]){
        Yhat_test[i] <- 0
    } else {
        Yhat_test[i] <- 1
    }
}

confusion_naive_test <- table(true = test$good_bad, predicted = Yhat_test)
confusion_naive_test
miss_naive_test <- round((100 * (confusion_naive_test[1,2] +
                                     confusion_naive_test[2,1]) /
                             sum(confusion_naive_test)), digits = 2)
paste("Naive Bayes misclassification rate for TESTING data: ", miss_naive_test, "%.")
```

Assignment 3

```r
RNGversion("3.5.1")
data<-read.csv("data/State.csv",header = TRUE,sep=";",dec = ",")
set.seed(12345)
data<-data[order(data$MET),]
```

3.1.

```r
tree_mod<-tree(EX~MET,data=data,control = tree.control(minsize = 8,
                                                      nobs = nrow(data)))
cv.res=cv.tree(tree_mod)
plot(cv.res$size, cv.res$dev, type="b", col="red")
plot(log(cv.res$size), cv.res$dev, type="b", col="red")

prunedTree=prune.tree(tree_mod,best=3)
pred=predict(prunedTree,data=data)

ggplot()+
  geom_line(aes(x=c(1:length(pred)),y=pred),col="red")+
  geom_point(aes(x=c(1:length(data$EX)),y=data$EX),col="blue")
res=data$EX-pred
hist(res)

mean(res)
sd(res)
```

3.2.

```
f=function(data,ind){
  data1=data[ind,]
  tree_mod<-tree(EX~MET,data=data1,control = tree.control(minsize = 8,
                                                          nobs = nrow(data)))
  prunedTree=prune.tree(tree_mod,best=3)
  pred=predict(prunedTree,data)
  return(pred)
}

boot_res=boot(data,f,R=1000)
CE<-envelope(boot_res,level = 0.95)


tree_mod<-tree(EX~MET,data=data,control = tree.control(minsize = 8,
                                                       nobs = nrow(data)))
prunedTree=prune.tree(tree_mod,best=3)
pred=predict(prunedTree,data)

CI_band=data.frame(upper=CE$point[1,],
                   lower=CE$point[2,],
                   EX=data$EX,MET=data$MET,pred=pred)
ggplot(data=CI_band)+
  geom_point(aes(x=MET,y=EX))+
  geom_line(aes(x=MET,y=pred),col="blue")+
  geom_ribbon(aes(x=MET,ymin=lower,ymax=upper),alpha=0.1)
```

3.3.

```
regression_tree=tree(EX~MET,data=data,control = tree.control(minsize = 8,
                                                             nobs = nrow(data)))
mle=prune.tree(regression_tree,best=3)

rng<-function(data,mle){
  data1=data.frame(EX=data$EX,MET=data$MET)
  data1$EX=rnorm(length(data1$EX),predict(mle,data1),sd=sd(residuals(mle)))
  return(data1)
}
f1<-function(data1){
  new_tree<-tree(EX~MET,data=data1,control = tree.control(minsize = 8,
                                                          nobs = nrow(data)))
  new_prunedTree=prune.tree(new_tree,best=3)
  new_pred<-predict(new_prunedTree,data)
  new_pred<-rnorm(length(new_pred),mean=new_pred,sd=sd(residuals(mle)))
  return(new_pred)
}

boot_res1=boot(data,statistic = f1,R=1000,mle = prunedTree,ran.gen = rng,sim="parametric")

CE<-envelope(boot_res1,level = 0.95)

tree_mod<-tree(EX~MET,data=data,control = tree.control(minsize = 8,nobs = nrow(data)))
prunedTree=prune.tree(tree_mod,best=3)
```

```
pred=predict(prunedTree,data)

CI_band=data.frame(upper=CE$point[1,],
                   lower=CE$point[2,],
                   EX=data$EX,MET=data$MET,pred=pred)
ggplot(data=CI_band)+
  geom_point(aes(x=MET,y=EX))+
  geom_line(aes(x=MET,y=pred),col="blue")+
  geom_ribbon(aes(x=MET,ymin=lower,ymax=upper),alpha=0.1)
```

3.4.

Assignment 4

```
data <- read.csv2("data/NIRSpectra.csv")
# removing viscosity since it is the variable to be predicted
data <- data[,-ncol(data)]
# 4.1 - PCA
res <- prcomp(data)
lambda <- res$sdev^2
prop <- lambda/sum(lambda) * 100
prop <- prop[order(prop, decreasing = TRUE)]
ggplot() + geom_point(aes(x=1:length(prop), y=prop)) +
    ggtitle("Variance Explained by Principal Components") +
    xlab("Principal Components") + ylab("Explained Variance [%]") +
    scale_y_continuous(breaks = seq(0, 110, by=10))
print("Principal Components Contributions")
sprintf("%2.3f", prop[1:16])
ggplot() + geom_point( aes(x=res$x[,1], y=res$x[,2]) ) +
    ggtitle("Scores - PC1 vs PC2") +
    xlab("PC1") + ylab("PC2")
```

4.1.

```
# 4.2 - PCA Trace plots
U <- res$rotation
x <- 1:nrow(res$rotation)
ggplot() + geom_point(aes(x=x, y=U[,1])) + ggtitle("PCA - Traceplot PC1") +
    xlab("Index") + ylab("Load")
ggplot() + geom_point(aes(x=x, y=U[,2])) + ggtitle("PCA - Traceplot PC2") +
    xlab("Index") + ylab("Load")
```

4.2.

```
# 4.3.a - Fast ICA
ica <- fastICA(X = data, n.comp = 2, alg.typ = "parallel")
W_ <- ica$K %*% ica$W
ggplot() + geom_point(aes(x=x, y=W_[,1])) + ggtitle("ICA - Traceplot PC1") +
    xlab("Index") + ylab("W'")
ggplot() + geom_point(aes(x=x, y=W_[,2])) + ggtitle("ICA - Traceplot PC2") +
    xlab("Index") + ylab("W'")
```

```r
# 4.3.b - Scores
ggplot() + geom_point( aes(x=ica$S[,1], y=ica$S[,2]) ) +
    ggtitle("Scores - IC1 vs IC2") +
    xlab("IC1") + ylab("IC2")
```

4.3.

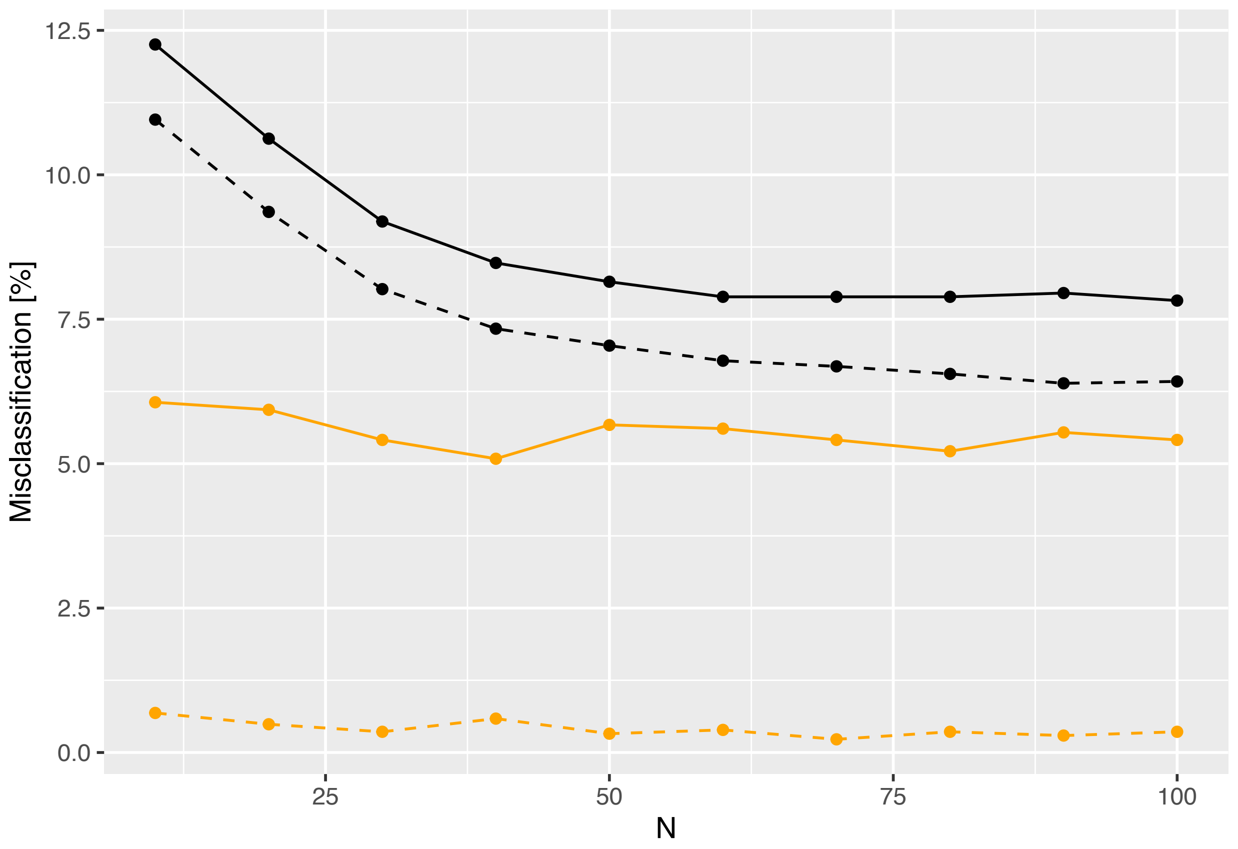
Machine Learning Assignment 02 - Group A15

Zuxiang Li (zuxli371), Marcos F. Mourao (marfr825), Agustín Valencia (aguva779)

12/04/2019

Task 01 - Ensemble Methods

Your task is to evaluate the performance of Adaboost classification trees and random forests on the spam data. Specifically, provide a plot showing the error rates when the number of trees considered are 10, 20, . . . , 100. To estimate the error rates, use 2/3 of the data for training and 1/3 as hold-out test data.



The misclassification rate for Adaboost decreases as the model's complexity (tree depth) grows in both train an test data. This is expected from a boosting algorithm, since the current classifier is based on how the previous classifier performed, weighting the misclassifications (with an exponential loss function) while maintaining the weight on correct classifications.

Random forest's presented a lower misclassification rate overall, but the results are somewhat indifferent regarding tree depth. The test error for training data was approximately zero, while for test data was higher (~5%). Since this model "grows" decision trees in parallel and later make the final classification by majority voting, the overall error is lower given that the error gets averaged down by all models.

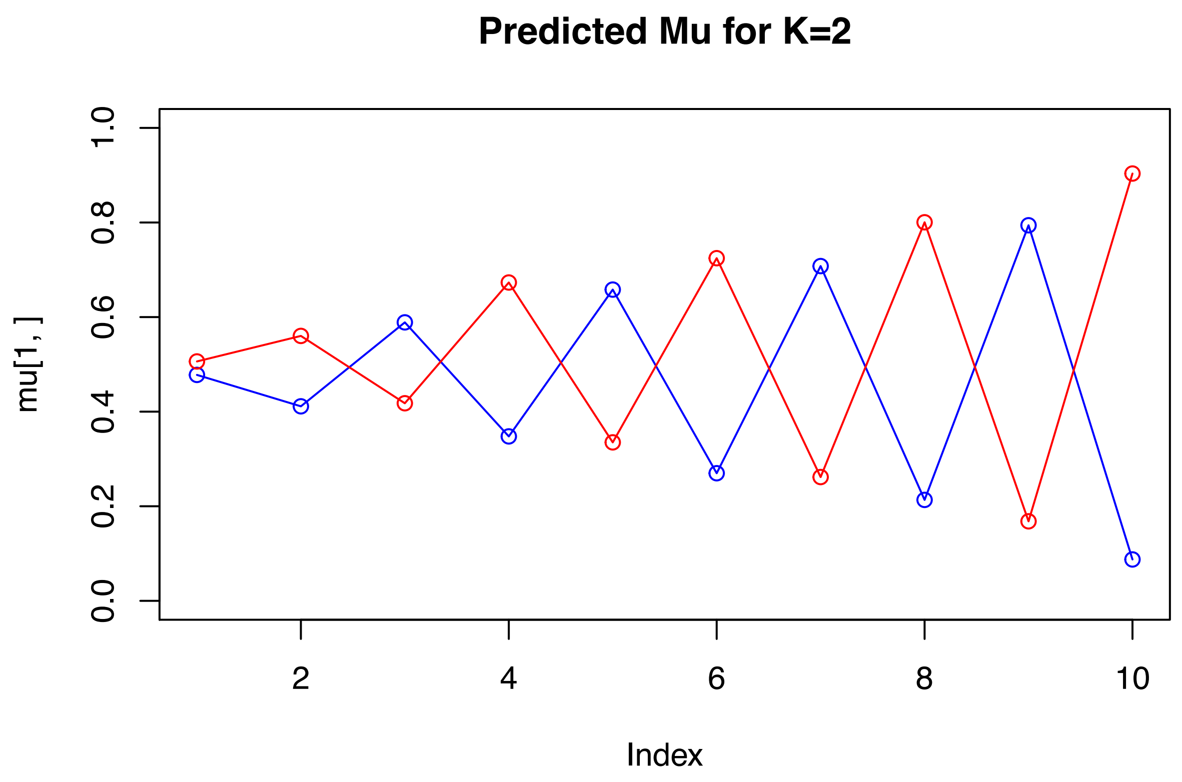
For both test and training data, random forest performed better than AdaBoost (i.e. lower misclassification rates).

Note - We are varying the number of trees and not the depth of each of these trees.
Good explanation otherwise.
You have it correct in your code though.

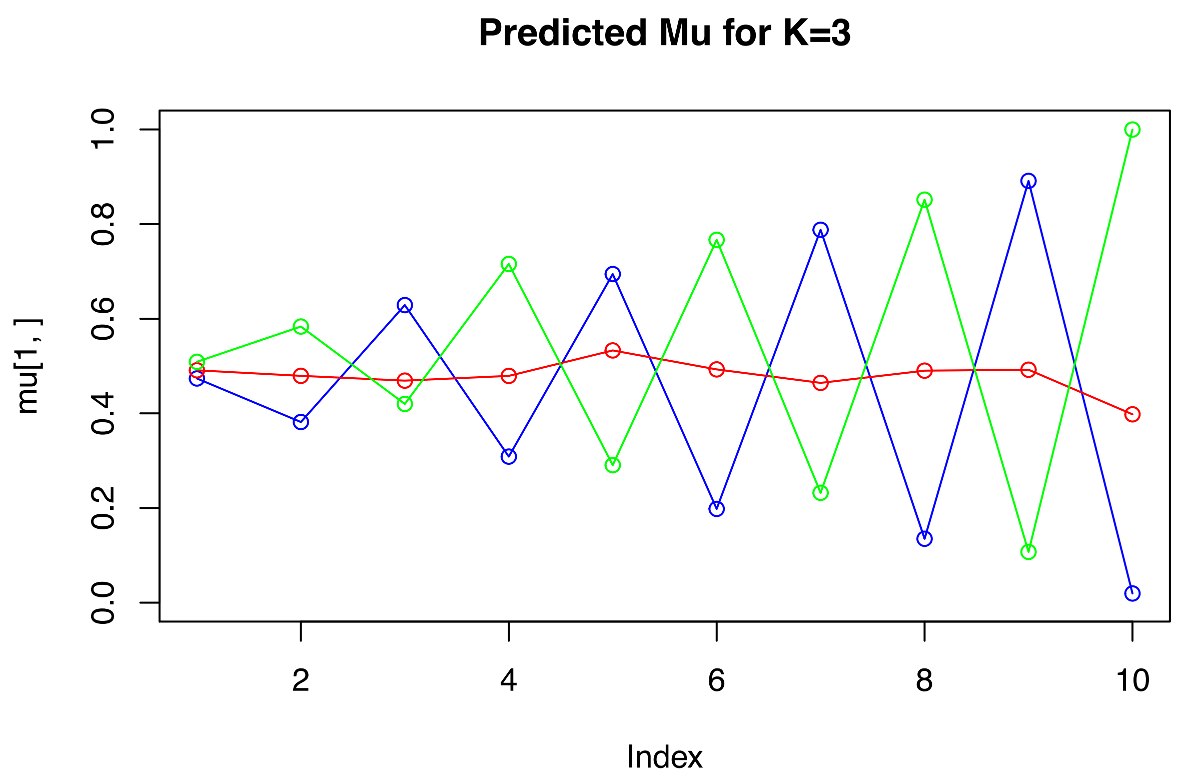
Task 02 - Mixture Models

Your task is to implement the EM algorithm for mixtures of multivariate Benoulli distributions. Then, use your implementation to show what happens when your mixture models has too few and too many components, i.e. set K = 2, 3, 4 and compare results. Please provide a short explanation as well.

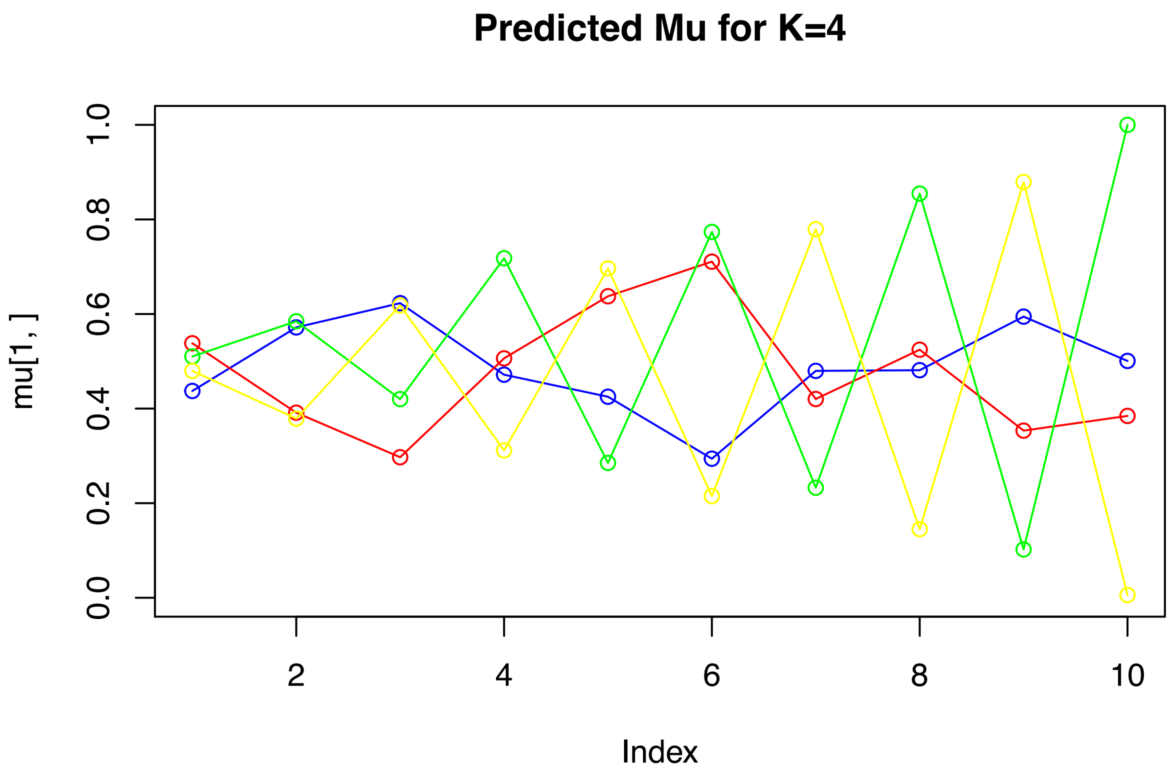
Solution



```
## [1] 0.4981919 0.5018081
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4777136 0.4113065 0.5888317 0.3477062 0.6580979 0.2698303 0.7078467
## [2,] 0.5061835 0.5601552 0.4177868 0.6731171 0.3350702 0.7245255 0.2617664
##           [,8]      [,9]     [,10]
## [1,] 0.2134061 0.7941168 0.0875152
## [2,] 0.8004711 0.1681467 0.9035340
```

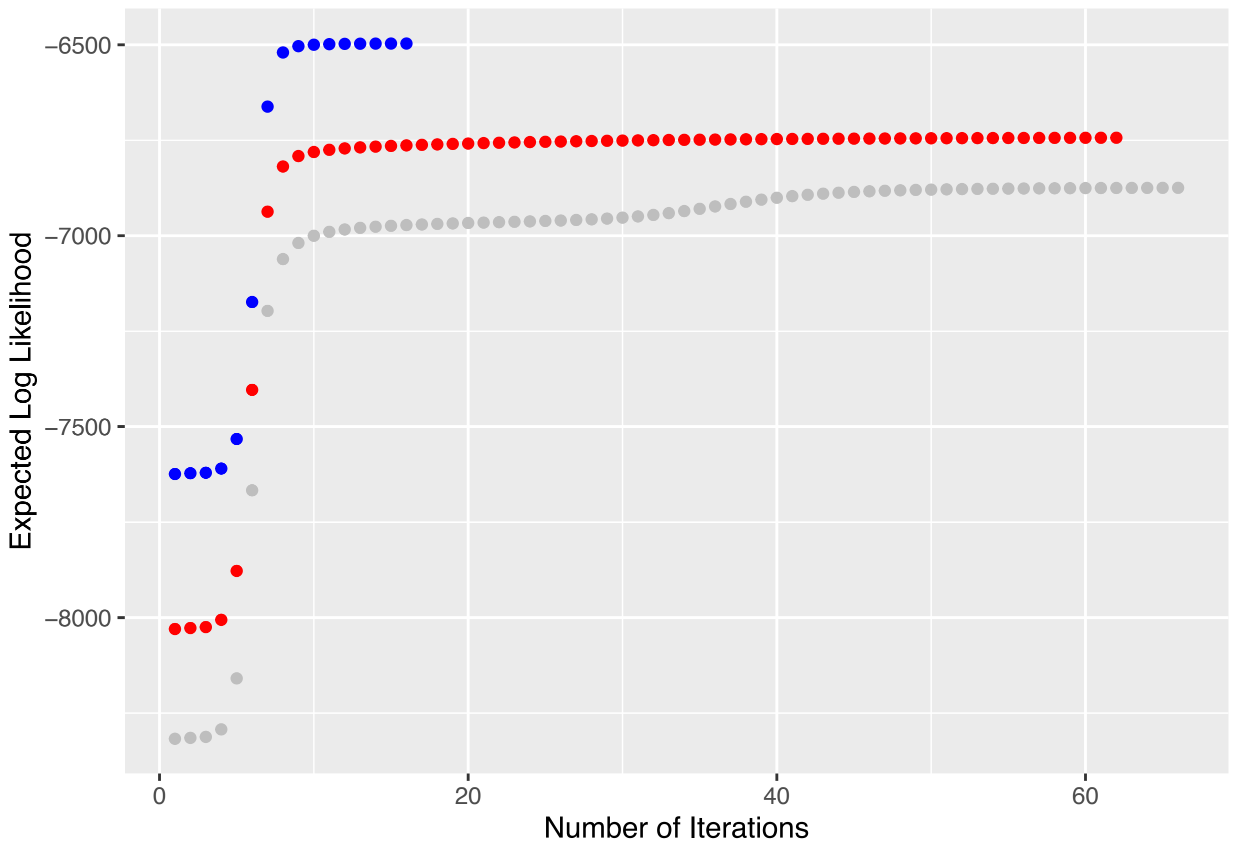


```
## [1] 0.3259592 0.3044579 0.3695828
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4737193 0.3817120 0.6288021 0.3086143 0.6943731 0.1980896 0.7879447
## [2,] 0.4909874 0.4793213 0.4691560 0.4791793 0.5329895 0.4928830 0.4643990
## [3,] 0.5089571 0.5834802 0.4199272 0.7157107 0.2905703 0.7667258 0.2320784
##           [,8]      [,9]      [,10]
## [1,] 0.1349651 0.8912534 0.01937869
## [2,] 0.4902682 0.4922194 0.39798407
## [3,] 0.8516111 0.1072226 0.99981353
```



```
## [1] 0.1614155 0.1383613 0.3609912 0.3392319
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4372908 0.5716691 0.6230114 0.4717152 0.4251232 0.2940734 0.4797605
## [2,] 0.5381955 0.3913346 0.2971686 0.5062848 0.6375272 0.7107583 0.4202372
## [3,] 0.5102441 0.5846281 0.4200464 0.7178717 0.2850900 0.7735833 0.2327656
## [4,] 0.4797762 0.3788928 0.6181216 0.3114748 0.6964392 0.2149967 0.7793732
##           [,8]      [,9]       [,10]
## [1,] 0.4812185 0.5945364 0.500815206
## [2,] 0.5246082 0.3534161 0.384513179
## [3,] 0.8546627 0.1022323 0.999999734
## [4,] 0.1450708 0.8791286 0.005800712
```

Your log-likelihood computation is not correct. You are calculating the joint log-likelihood of observed data and latent variables. You are supposed to calculate the log-likelihood of the observed data only.



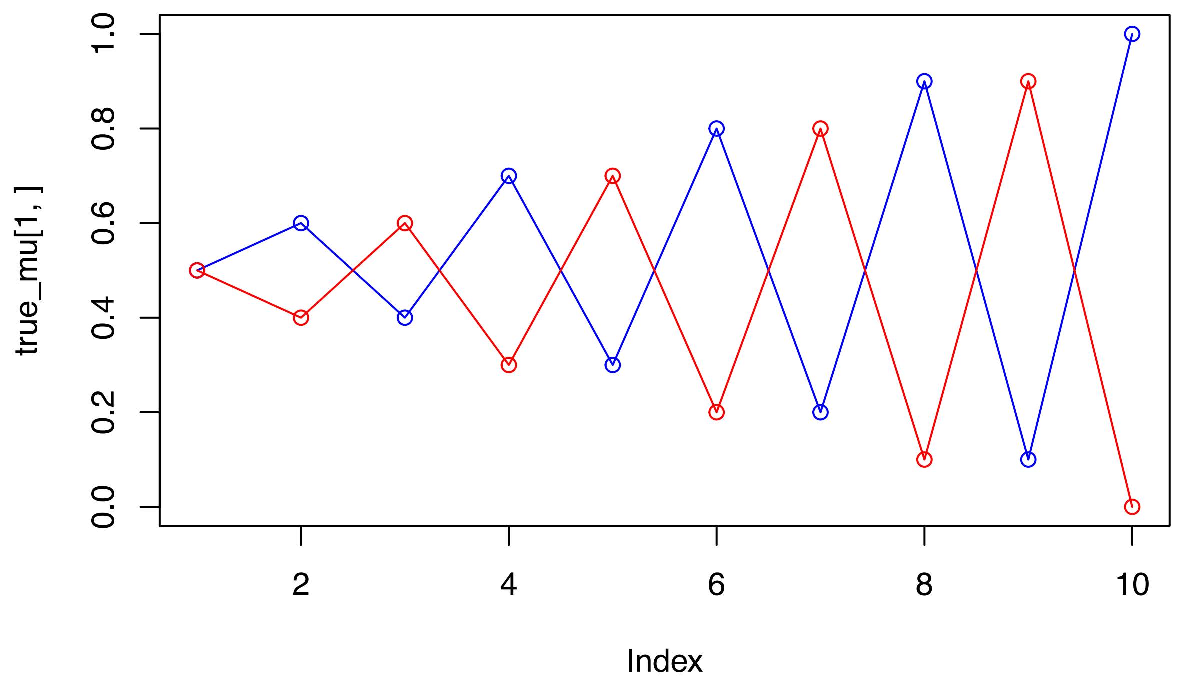
In this case, we generate data using true_mu and true_pi. For K=2, the blue dots, which means too few components, the blue points in the plot, only 16 iterations were made, this model is underfitted. For K=4, the grey dots, too many components, grey dots in the plot, 66 iterations were made, then this model is overfitted. Only when K=3, the red dots in plot, we got similar mu and pi compare to the true values.

The values of `true_mu` and predicted `mu` are shown below:

What is your motivation behind saying that the model if underfitting/overfitting? We cannot conclude this based on the plots of mu alone.

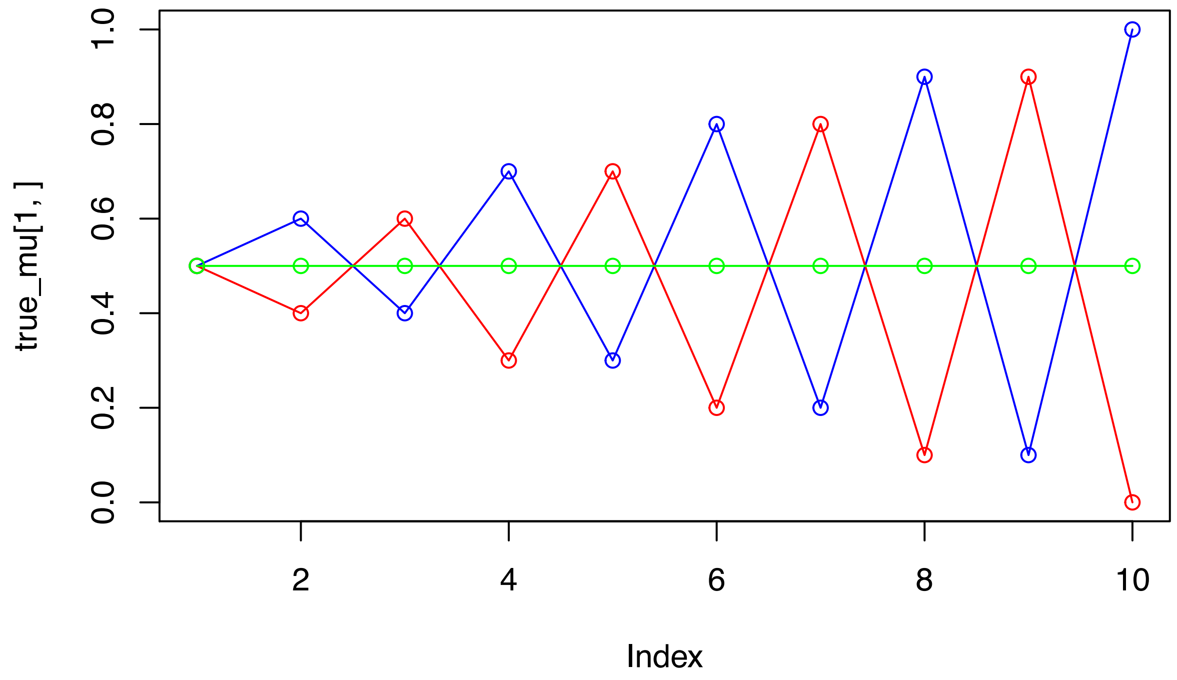
K=2

`true_mus` :



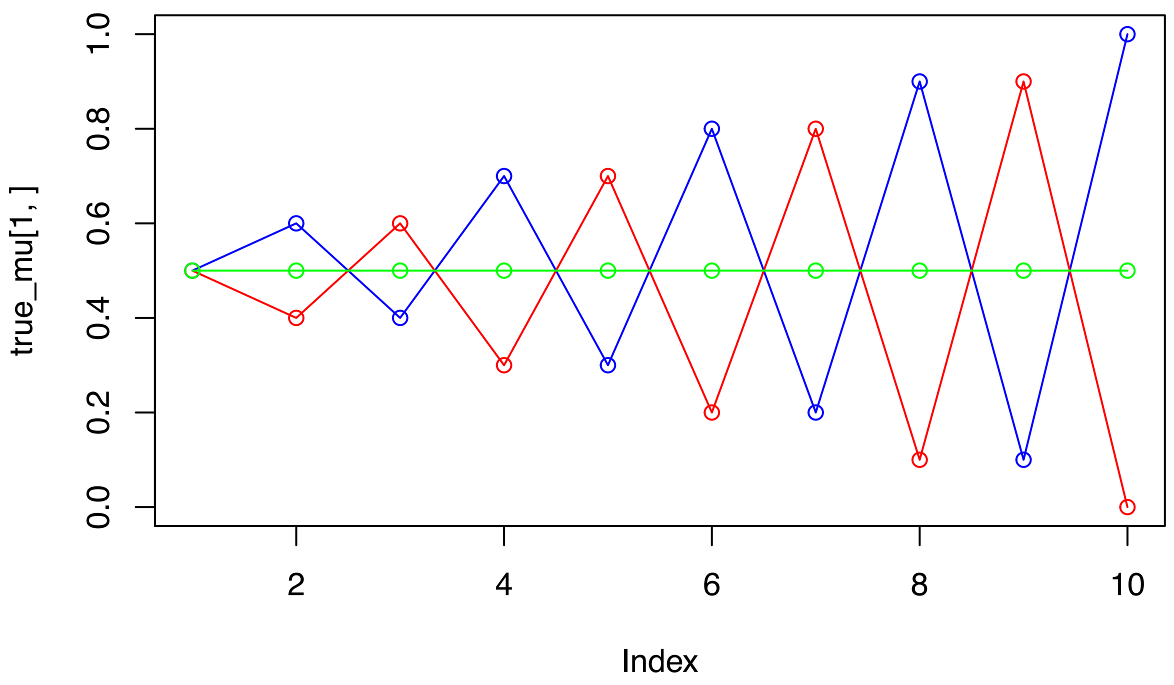
K=3

`true_mus` :



K=4

`true_mus`:



true_mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Index

Code Appendix

```r
knitr::opts_chunk$set(fig.pos = "!h")
library(mboost)
library(randomForest)
library(ggplot2)
library(partykit)
library(GLDEX)
set.seed(1234567890)
sp <- read.csv2("data/spambase.csv")
sp$Spam <- as.factor(sp$Spam)
## Splitting data
n <- dim(sp)[1]
idxs <- sample(1:n, floor(2*n/3))
train <- sp[idxs,]
test <- sp[-idxs,]
get_missed <- function (true, predicted) {
    confussion <- table(true, predicted)
    fn <- confussion[1,2]
    fp <- confussion[2,1]
    total <- sum(confussion)
    miss <- (fp + fn) / total * 100
    return(miss)
}
# Training
nums <- seq(10,100,10)
formula <- Spam ~ .
error_rates_train_bb <- c()
error_rates_test_bb <- c()
error_rates_train_rf <- c()
error_rates_test_rf <- c()
depths <- c()
for (i in nums) {
    bb <- blackboost (
            Spam ~ .,
            data = train,
            family = AdaExp(),
            control = boost_control(mstop = i)
     )

    rf <- randomForest(
            Spam ~ .,
            data = train,
            ntree = i
    )

    predicted <- predict(bb, train, type = "class")
    miss <- get_missed(train$Spam, predicted)
    error_rates_train_bb <- append(error_rates_train_bb, miss)

    predicted <- predict(bb, test, type = "class")
    miss <- get_missed(test$Spam, predicted)
    error_rates_test_bb <- append(error_rates_test_bb, miss)
    predicted <- predict(rf, train, type = "class")
    miss <- get_missed(train$Spam, predicted)
    error_rates_train_rf <- append(error_rates_train_rf, miss)

    predicted <- predict(rf, test, type = "class")
    miss <- get_missed(test$Spam, predicted)
    error_rates_test_rf <- append(error_rates_test_rf, miss)

    depths <- append(depths, i)
}
df <- data.frame(
    nums = nums,
    error_rates_train_bb = error_rates_train_bb,
    error_rates_test_bb = error_rates_test_bb,
    error_rates_train_rf = error_rates_train_rf,
    error_rates_test_rf = error_rates_test_rf
)
p <- ggplot() +
    geom_line(aes(x = depths, y = error_rates_train_bb), color = "black", linetype = "dashed") +
    geom_line(aes(x = depths, y = error_rates_test_bb), color = "black") +
    geom_line(aes(x = depths, y = error_rates_train_rf), color = "orange", linetype = "dashed") +
    geom_line(aes(x = depths, y = error_rates_test_rf), color = "orange") +
    geom_point(aes(x = depths, y = error_rates_train_bb), color = "black") +
    geom_point(aes(x = depths, y = error_rates_test_bb), color = "black") +
    geom_point(aes(x = depths, y = error_rates_train_rf), color = "orange") +
    geom_point(aes(x = depths, y = error_rates_test_rf), color = "orange") +
    xlab("N") + ylab("Misclassification [%]")
p
RNGversion('3.5.1')
EM_algorithm<-function(K){
  set.seed(1234567890)
  max_it <- 100 # max number of EM iterations
  min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
  N=1000 # number of training points
  D=10 # number of dimensions
  x <- matrix(nrow=N, ncol=D) # training data
  true_pi <- vector(length = 3) # true mixing coefficients
  true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
  true_pi=c(1/3, 1/3, 1/3)
  true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
  true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
  true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
  #plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
  #points(true_mu[2,], type="o", col="red")
  #points(true_mu[3,], type="o", col="green")
  # Producing the training data
  for(n in 1:N) {
    k <- sample(1:3,1,prob=true_pi)
    for(d in 1:D) {
      x[n,d] <- rbinom(1,1,true_mu[k,d])
    }
  }
  #K=3 # number of guessed components
```

```r
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
for(k in 1:K) {
  mu[k,] <- runif(D,0.49,0.51)
}

for(it in 1:max_it) {

  #plot(mu[1,], type="o", col="blue", ylim=c(0,1))
  #points(mu[2,], type="o", col="red")
  #points(mu[3,], type="o", col="green")
  #points(mu[4,], type="o", col="yellow")
  Sys.sleep(0.5)
  # E-step: Computation of the fractional component assignments
  # Your code here
  for(n in 1:N){
    denominator<-c()
    numerator<-c()
    for(k in 1:K){
      new_numerator<-pi[k]*prod(mu[k,]^x[n,]*(1-mu[k,])^(1-x[n,]))
      numerator<-c(numerator,new_numerator)
      denominator<-sum(numerator)
    }
    z[n,]<-numerator/denominator
  }
  #Log likelihood computation.
  sum=0
  for(n in 1:N){
    for(k in 1:K){
      sum=sum+z[n,k]*(log(pi[k])+sum(x[n,]*log(mu[k,])+(1-x[n,])*log(1-mu[k,])))
    }
  }
  llik[it]<-sum
  # Your code here
  #cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  # Stop if the lok likelihood has not changed significantly
  if(it>1){
    if(abs(llik[it]-llik[it-1])<min_change){
      titStr <- paste("Predicted Mu for K=", K, sep = "")
      plot(mu[1,], type="o", col="blue", ylim=c(0,1),main = titStr)
      if(K>=2)
        points(mu[2,], type="o", col="red")
      if(K>=3)
        points(mu[3,], type="o", col="green")
      if(K>=4)
        points(mu[4,], type="o", col="yellow")
        print(pi)
        print(mu)
        break
      }

    }
    # Your code here
    #M-step: ML parameter estimation from the data and fractional component assignments
    # Your code here
    pi=apply(z,2,mean)
    for(k in 1:K){
      mu[k,]=0
      for (n in 1:N) {
        mu[k,] = mu[k,] + x[n,] * z[n,k]
      }
      mu[k,] = mu[k,] / sum(z[,k])
    }
  }
  pi
  mu
  #plot(llik[1:it], type="o")
  return(llik[1:it])
  #return(list(pi=pi,mu=mu,res=,llik=llik[1:it])
}

em1<-EM_algorithm(K=2)
em2<-EM_algorithm(K=3)
em3<-EM_algorithm(K=4)

ggplot()+geom_point(aes(x=c(1:length(em1)),y=em1),color="blue")+
  geom_point(aes(x=c(1:length(em2)),y=em2),color="red")+
  geom_point(aes(x=c(1:length(em3)),y=em3),color="grey")+
  xlab("Number of Iterations")+ylab("Expected Log Likelihood")
true_mu <- matrix(nrow=3, ncol=10) # true conditional distributions
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
    true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
    true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
    plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
    points(true_mu[2,], type="o", col="red")
true_mu <- matrix(nrow=3, ncol=10) # true conditional distributions
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
    true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
    true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
    plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
    points(true_mu[2,], type="o", col="red")
    points(true_mu[3,], type="o", col="green")
   true_mu <- matrix(nrow=3, ncol=10) # true conditional distributions
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
    true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
    true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
    plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
    points(true_mu[2,], type="o", col="red")
    points(true_mu[3,], type="o", col="green")
```

LAB 3 BLOCK 1 - Group A15

*Zuxiang Li (**zuxli371**), Marcos F. Mourao (**marfr825**), Agustín Valencia (**aguva779**)*

17 December 2019

There are some small mistakes but really good report overall and nice explanations and analysis!

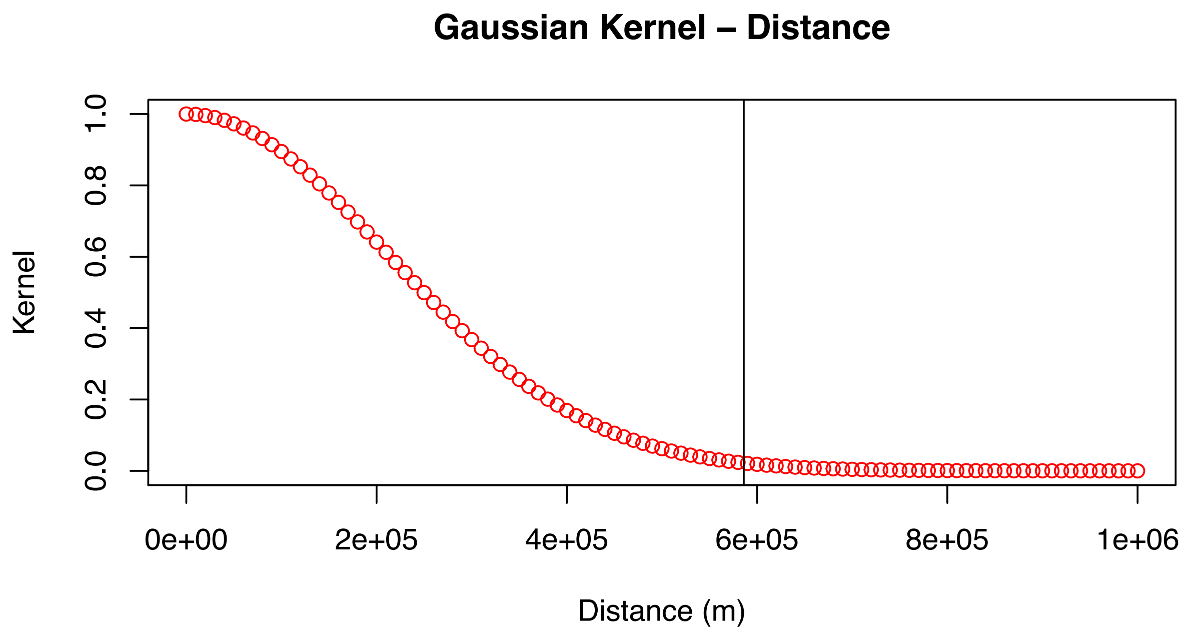
Assignment 1: Kernel Methods

For illustration purposes, a point in Linköping (latitude: 58.41086, longitude: 15.62157) is chosen. We want to predict temperatures in said point at July 7th, 2014.

```r
#please input the coordinates and date of your desired prediction:
date <- "2014-7-11" #The date to predict
lat <- 58.41086 #Latitude of the point to predict
lon <- 15.62157 #Longitude of the point to predict

X <- c(lat,lon)
```

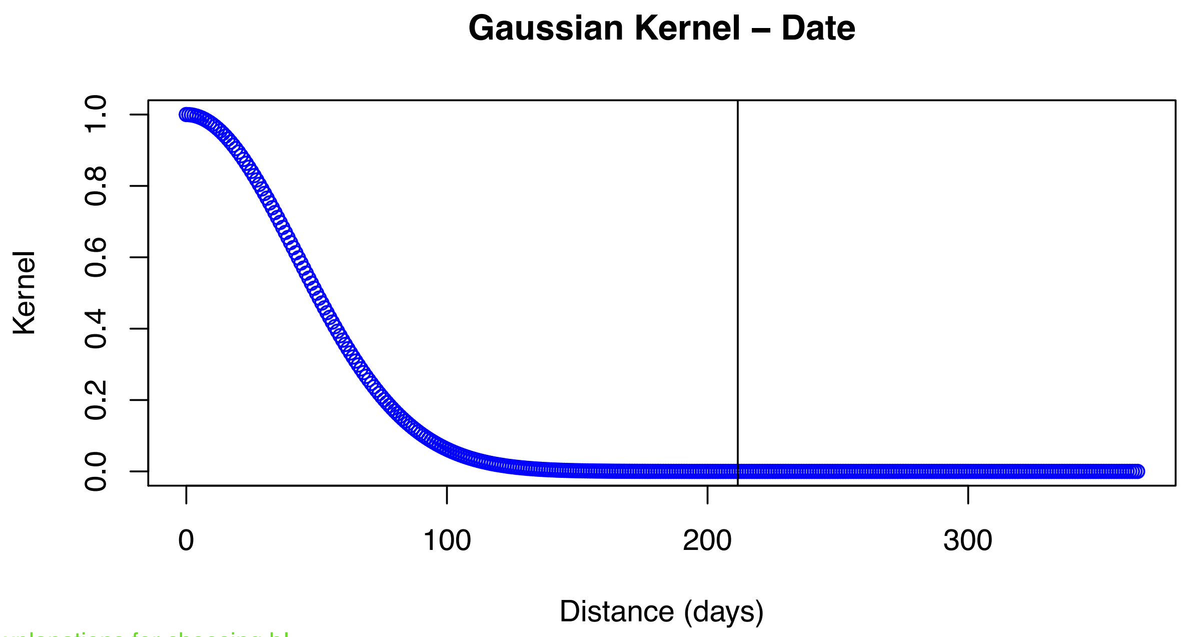
The smoothing coefficients of the kernels were chosen based on the Gaussian kernel behaviour. By using the empirical rule, it is possible to have an idea of the interval on which 95% of our data is (be it distances, days or hours).

The smoothing factor for distance, `h_distance` was set to 300000m (or 300km). It seems reasonable to think that points within an 300km radius should have high impact on the weather prediction. The black line in the plot below shows the band where 95% of distance measures lie. Distances beyond that point (~600km) have almost no impact on the predictions. Again, that seems reasonable.



As for dates (measured in days), a similar thought process was applied: the smoothing coefficient `h_date` was set to 60 days. The plot below shows that it is reasonable to think measurements taken ~100 days apart from the desired prediction have approximately zero weight in said prediction.

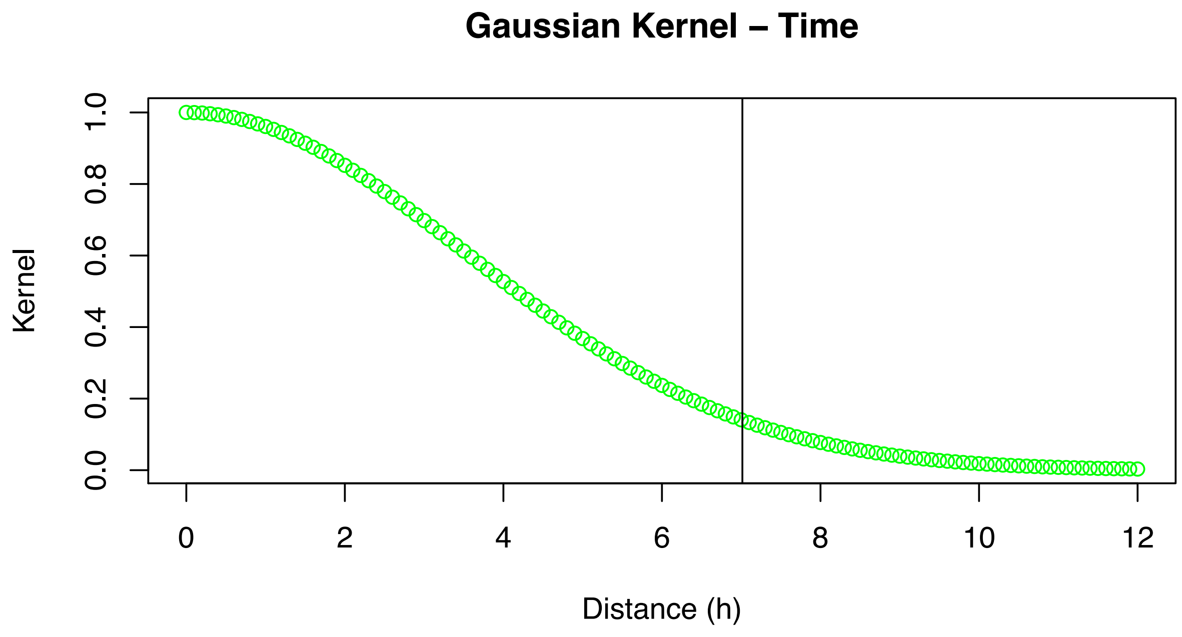
Your code does not handle the case that the distance between Dec 31 and Jan 1 is 1 day.
I think you give low weights only for days that are after the date for which you predict.



Good explanations for choosing h!

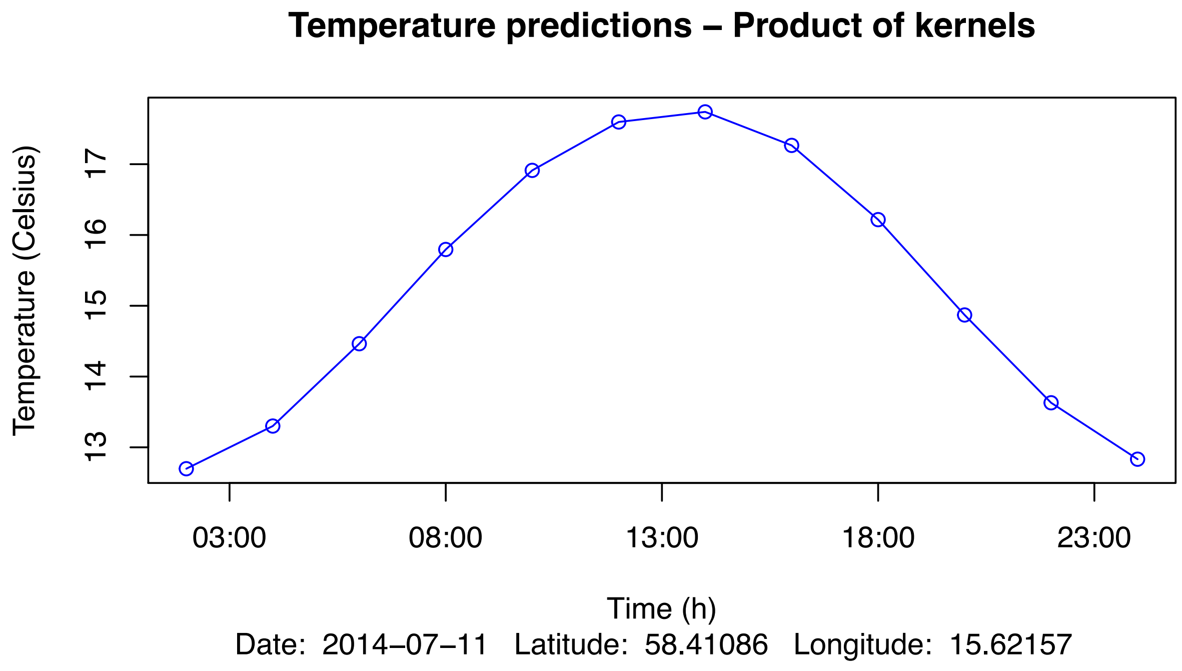
Finally, for time distances (measured in hours), the smoothing coefficient `h_time` was set to 5 hours. Similarly to what has been argued for the above cases, it seems reasonable by looking at the plot that measurements made ~7 hours apart from the desired prediction have lesser influence in said prediction.

Based on below plot, I would maybe reduce h a bit more if you want to give very low weights beyond 7 hrs.

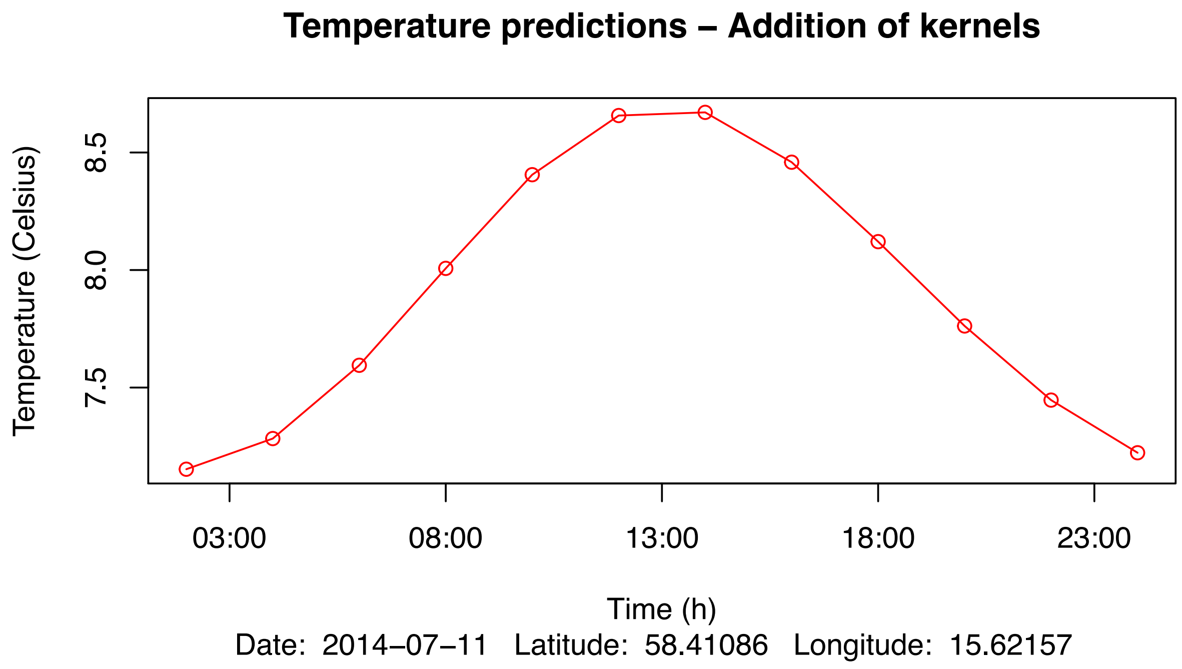


With all distances and kernels (weights) computed, it is possible to make predictions. The predictions were made using two methods: product and summation of all three kernels.

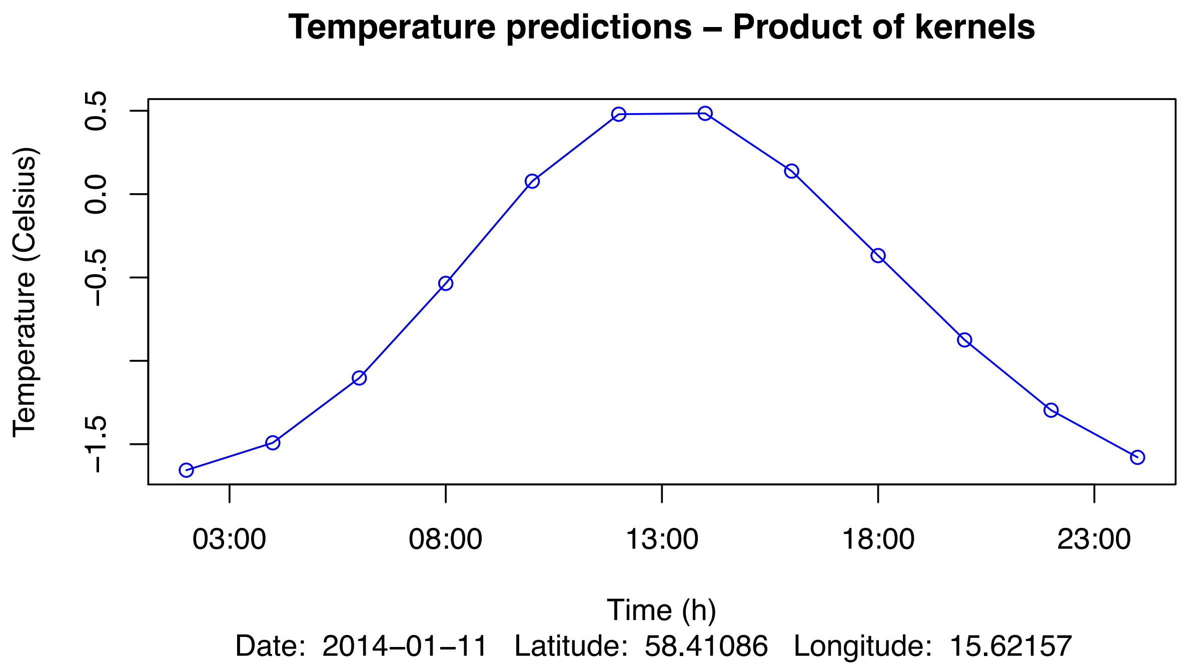
The product of kernels produce reasonable temperature predictions, as it can be seen below. The behaviour of the curve is consisten with an "usual" temperature fluctuation within a day, i.e. warmer temperatures during daytime and coler temperatures at night.

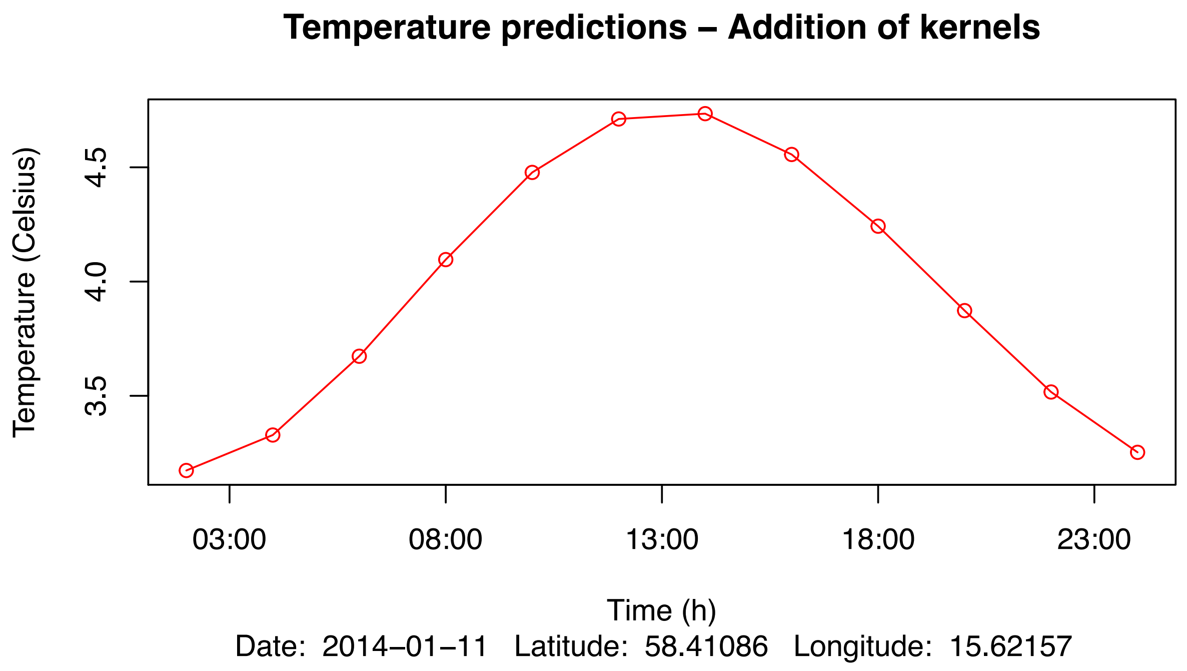


The addition of kernels produced a similar shaped curve. This curve, on the other hand is not reasonable due to the magnitude of these temperatues. Our prediction day is set in summer and intuitevely the predicted temperatures for this time of the year are too low.



The temperatures at the same location are also predicted for January 7th 2014, winter. Again, the behaviour of the predictions is reasonable: warmer temperatures during daytime than night. Also the differences in magnitude between the product and summation of kernels is seen: the product of kernels generate a more reasonable result: i.e. reasonably colder temperatures (negative) during winter.





The conclusion is that while giving some insight in temperature behaviour throught the day, the models proposed are not good: they are too simple. Location, day and hour of measurement is simply not enough to model this type of data. Temperatures throughout the years are impacted by many other factors. For example, global warming.

The different results for product and summation of kernels is due to the independency of kernels: the temperature of a given place is independent from the temperature of a given hour of the day, which is independent from the temperature in a given day of the year. It seems reasonable then to multiply each kernel's contribution to the prediction. This is analogous to the Multiplicative Law of Probability.

Yeah, that can be one way to think about it. But consider when you give reasonable weights in both the kernels

Assignment 2: Support Vector Machines

The spam dataset has been split in a ratio of 70%/30% for training and testing purposes respectively.

In order to compare the predictors performance, it have been ran experiments to get the training error, number of support vectors, true positives , false negative , false positive, false negative rates and overall misclassification from all the predictors

The code of the experiments is detailed in Appendix B and the results are summarized in the following table (rates are given in % format).

C	trainingError	numSV	TPR	TNR	FPR	FNR	Misclassification
0.5	4.41	1365	93.09	91.11	6.91	8.89	8.18
1.0	3.76	1278	91.76	92.08	8.24	7.92	8.04
5.0	1.96	1187	92.32	92.21	7.68	7.79	7.75

Taking into account that our classifier is meant to detect spam emails, it has to be noticed that, from the users perspective, loosing true email in the spambox is way worse than getting spam email in the inbox. That is measured by the False Positive Rate, for which $C = 0.5$ get the best score.

Analyzing the overall misclassification it is seen that $C = 5.0$ gets the better score, though, it also gets a higher False Positive rate than $C = 0.5$, which has been said is the most important score for this application. Thus, we drop it anyway. Nice analysis!

Regarding the amount of support vectors chosen by the model training, one might tend to think that the highest the amount of support vectors the overfitted the model, which is not false at all, though it is not always true. Having a wider soft margin might also produce more support vectors in the sense of the dynamics of *pushing* (support vectors in the right side of the hyperplane) and *pulling* (support vectors in the wrong side of the hyperplane). This can be interpreted as the highest training error for model $C = 0.5$

Therefore the chosen model to be deployed into the customer's system would be the following:

```r
svmToDeploy <- ksvm(
    type ~ .,
    data=spam,
    type="C-svc",
    kernel="rbfdot",
    C = 0.5,
    kpar = list(sigma = 0.05)
)
```

Model should be trained on train+validation data only after model selection.

Parameter C denotes the inverse of the regularization penalization. The bigger the C the smaller the soft margin between the classes to learn, thus, the more strict the learning process which could lead to overfit the data. Fortunately that is not the case for this classifier, since it gets even better scores under testing data than the more relaxed models.

Cross-validation still recommended in order to get the best results possible for the classifier, though for this assignment we are not allowed to perform that type of analisys.

Generalization error is missing!
If you are using the holdout method, then you need to split the data into train-validation-test and the test error is is a kind of generalization error as it is calculated on unseen data.

Appendix A - Assignment 1 Code

```r
##############################-ASSIGNMENT 1-################################
#Initial configs.
suppressWarnings(RNGversion('3.5.1'))
set.seed(1234567890)
library(geosphere)
library(readr)
library(ggplot2)
library(kernlab)
library(knitr)
#please input the coordinates and date of your desired prediction:
date <- "2014-7-11" #The date to predict
lat <- 58.41086 #Latitude of the point to predict
lon <- 15.62157 #Longitude of the point to predict

X <- c(lat,lon)
# reading data
stations <- read_csv("Data/stations.csv",
                     locale = locale(encoding = "ISO-8859-1"))
temps <- read.csv("./Data/temps50k.csv")
st <- merge(stations,temps,by="station_number")
#cleaning the data
st <- st[,c(-1,-2,-3,-6,-7,-8, -12)]
st$date <- as.Date(st$date)
st$time <- strptime(st$time, format = "%H:%M:%S")

date <- as.Date(date)
times <- c("02:00:00",
           "04:00:00",
           "06:00:00",
           "08:00:00",
           "10:00:00",
           "12:00:00",
           "14:00:00",
           "16:00:00",
           "18:00:00",
           "20:00:00",
           "22:00:00",
           "24:00:00")
times <- strptime(times, format = "%H:%M:%S")


#filtering out posterior dates
ind <- which(st$date > date)
st <- st[ind, ]

#-------------------------------DISTANCES-------------------------

#getting distances (in meters)
get_distance <- function() {
  distance <- distHaversine(st[,1:2], X)
  return(distance)
}
```

```
#getting date distances (in days)
get_datedist <- function() {
  date.distance <- as.numeric(abs(difftime(time1 = st$date,
                                time2 = date,
                                units = "days")))
  date.distance <- date.distance %% 365
  return(date.distance)
}

#getting time distances (in hours)

get_timedist <- function(idx) {
  time.distance <- as.numeric(abs((difftime(time1 = st$time,
                                 time2 = times[idx],
                                 units = "hours"))))

  time.distance <- ifelse(time.distance<12, time.distance, 24-time.distance)

  return(time.distance)
}
#-------------SMOOTHING COEFFICIENTS--------------------------
#Distance smoothing
h_distance <- 300000
x <- seq(0,1000000, 10000)
line <- 2 * sd(x)
y <- exp(-(x/h_distance)^2)
plot(x, y, main = "Gaussian Kernel - Distance",
     xlab = "Distance (m)",
     ylab = "Kernel",
     col = "red")
abline(v = line)
#Day smoothing
h_date <- 60
x <- seq(0,365, 1)
line <- 2 * sd(x)
y <- exp(-(x/h_date)^2)
plot(x, y, main = "Gaussian Kernel - Date",
     xlab = "Distance (days)",
     ylab = "Kernel",
     col = "blue")
abline(v = line)
#Hour smoothing
h_time <- 5
x <- seq(0,12, 0.1)
line <- 2 * sd(x)
y <- exp(-(x/h_time)^2)
plot(x, y, main = "Gaussian Kernel - Time",
     xlab = "Distance (h)",
     ylab = "Kernel",
     col = "green")
abline(v = line)
#---------------KERNEL COMPUTATIONS------------------------------------
#distance kernel
```

```r
distanceKernel <- function(){
  dist <- get_distance()
  kern <- exp(-(dist/h_distance)^2)
  return(kern)
}

#day kernel
dateKernel <- function(){
  dist <- get_datedist()
  kern <- exp(-(dist/h_date)^2)
  return(kern)
}

#hour kernel
timeKernel <- function(index){
  dist <- get_timedist(index)
  kern <- exp(-(dist/h_time)^2)
  return(kern)
}

#-------------------PREDICTIONS-----------------------
#addition of kernels
predict_add <- function(index) {
  temp <- sum(st$air_temperature * (distanceKernel() + dateKernel() + timeKernel(index)))/
      (sum(distanceKernel() + dateKernel() + timeKernel(index)))
  return(temp)
}

add.temp <- vector(length = length(times))
for (i in 1:length(times)){
  add.temp[i] <- predict_add(i)
}

#product of kernels
predict_mult <- function(index) {
  temp <- sum(st$air_temperature * (distanceKernel() * dateKernel() * timeKernel(index)))/
      (sum(distanceKernel() * dateKernel() * timeKernel(index)))
  return(temp)
}

mult.temp <- vector(length = length(times))
for (i in 1:length(times)){
  mult.temp[i] <- predict_mult(i)
}
#------------------------PLOTTING-----------------------------
hour.names <- format(times, "%H:%M:%S")
plot.subtitle <- paste("Date: ", date , " ",
                       "Latitude: ", lat, " ",
                       "Longitude: ", lon, " ")

#COMBINED PLOT
# ggplot() +
#   geom_line(aes(x = hour.names, y = mult.temp, colour = "Product of kernels", group = 1)) +
#   geom_line(aes(x = hour.names, y = add.temp, colour = "Addition of kernels", group = 2)) +
#   geom_point(aes(x = hour.names, y = mult.temp, colour = "Product of kernels", group = 1)) +
#   geom_point(aes(x = hour.names, y = add.temp, colour = "Addition of kernels", group = 2)) +
#   ggtitle(label = "Temperature prediction", subtitle = plot.subtitle) +
#   xlab("Time of day (HH:MM:SS)" ) +
#   ylab ("Temperature (Celsius)") +
#   theme(axis.text.x = element_text(angle = 45))
#ORIGINAL PLOT
#mult
plot(x = times, y = mult.temp, type="o", col = "blue",
     main ="Temperature predictions - Product of kernels",
     xlab = "Time (h)", ylab = "Temperature (Celsius)",
     sub = plot.subtitle)
#add
plot(x = times, y = add.temp, type = "o", col = "red",
     main = "Temperature predictions - Addition of kernels",
     xlab = "Time (h)",ylab = "Temperature (Celsius)",
     sub = plot.subtitle)
```

Appendix B - Assignment 2 Code

```
## Utils
splitData <- function(data, trainRate) {
    n <- dim(data)[1]
    idxs <- sample(1:n, floor(trainRate*n))
    train <- data[idxs,]
    test <- data[-idxs,]
    return (list(train = train, test = test))
}

get_performance <- function(targets, predictions, text) {
    t <- table(targets, predictions)
    tn <- t[1,1]
    tp <- t[2,2]
    fp <- t[1,2]
    fn <- t[2,1]
    total <- sum(t)
    tpr <- tp/(tp+fp) * 100
    tnr <- tn/(tn+fn) * 100
    fpr <- fp/(tp+fp) * 100
    fnr <- fn/(tn+fn) * 100

    return (
        list(
            tpr = tpr,
            tnr = tnr,
            fpr = fpr,
            fnr = fnr,
            misclass = (fp+fn)/total * 100
        )
    )
}

# Data split
data(spam)
split <- splitData(spam, .7)
train <- split$train
test  <- split$test

train.x <- train[,-ncol(train)]
train.y <- train[,ncol(train)]
test.x  <- test[,-ncol(test)]
test.y  <- test[,ncol(test)]

Cs <- c(.5, 1, 5)
kWidth <- 0.05
svmModels <- list()
svmScores <- data.frame (
    C = vector(length = 3),
    trainingError = vector(length = 3),
    numSV = vector(length = 3),
    TPR = vector(length = 3),
    TNR = vector(length = 3),
    FPR = vector(length = 3),
    FNR = vector(length = 3),
    Misclassification = vector(length = 3)
)
for (i in 1:length(Cs)) {
    svmModel <- ksvm(
                    type ~ .,
                    data=train,
                    type="C-svc",
                    kernel="rbfdot",
                    C = Cs[i],
                    kpar = list(sigma = kWidth)
              )
    predictions <- predict(svmModel, test)
    performance <- get_performance(test.y, predictions)
    svmScore <- c(
                    Cs[i],
                    error(svmModel) * 100,
                    nSV(svmModel),
                    performance$tpr,
                    performance$tnr,
                    performance$fpr,
                    performance$fnr,
                    performance$misclass
                  )
    svmScores[i,] <- svmScore
    svmModels[[i]] <- svmModel
}
kable(svmScores, digits = 2)
```