Introduction to Machine Learning - Lab 2 Block 2 Group Report

Milda Poceviciute, Henrik Karlsson, Ugurcan Lacin

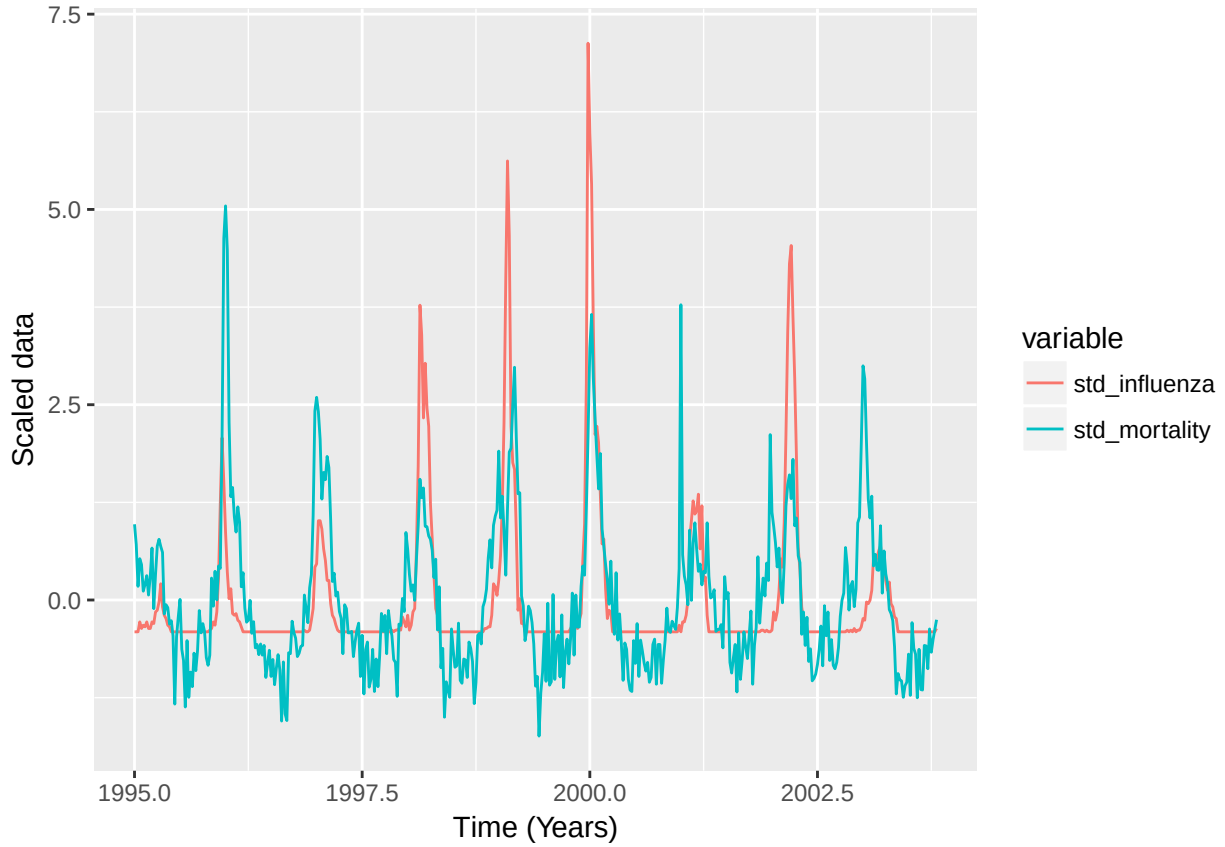
15 December 2017

```
library(readxl)
library(ggplot2)
library(tidyverse)
library(mgcv)
library(pamr)
library(glmnet)
library(kernlab)

set.seed(12345)
df <- read_excel("Influenza.xlsx")
```

Assignment 1

Part 1

```
# Assignment 1.1 ----------------------------------------------------------

std_df <- df %>%
    mutate(std_mortality = scale(.$Mortality),
           std_influenza = scale(.$Influenza)) %>%
    gather(variable, value, -Year, -Week, -Time, -`Temperature deficit`)

# Standardized data
ggplot(filter(std_df, variable == "std_influenza" |
                  variable == "std_mortality")) +
    geom_line(aes(x = Time, y = value, color = variable)) +
    labs(x = "Time (Years)", y = "Scaled data")
```



The plot above shows the time series for mortality and number of influenza outbreaks in Sweden from 1995 to 2003. The plot above shows the standardized values for both variables, in order to compare the variance within each variable. The standardization is computed by:

$$standardized = \frac{x_i - \bar{x_i}}{sd}$$

Part 2

Generalized Additative Model is a generalized linear model in which the linear predictor depend on unknown smoothing functions of predictor variables. GAM is used when predicting complex data, due to the possibility of adding splines and or smoothing functions.

A spline is a function that split up the data into partitions and fit a model to the partion and the model for each spline is linked over the knots. The variance of spline estimators have different boundary effects. Different smoothing functions can be applied to the GAM and the splines. The optimal smoothing is found by minimizing the RSS, residual Sum of Squares.

```
# Assignment 1.2 ------------------------------------------------------------

res <- gam(Mortality ~ Year + s(Week,k=52),data=df,family = gaussian(),method = "GCV.Cp")
summary(res)
```

```
##
## Family: gaussian
## Link function: identity
##
```

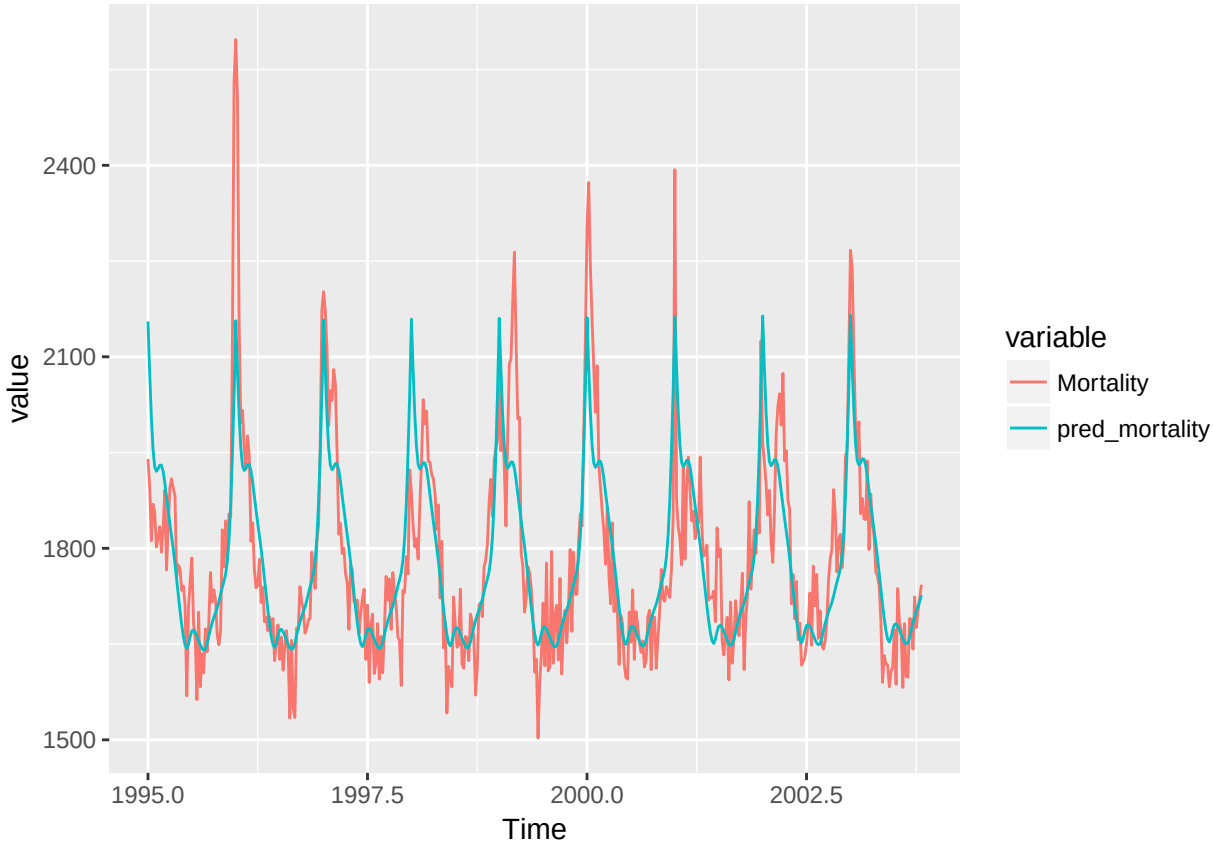
```
## Formula:
## Mortality ~ Year + s(Week, k = 52)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -680.598   3367.760  -0.202    0.840
## Year           1.233      1.685   0.732    0.465
##
## Approximate significance of smooth terms:
##           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 probablistic model is:

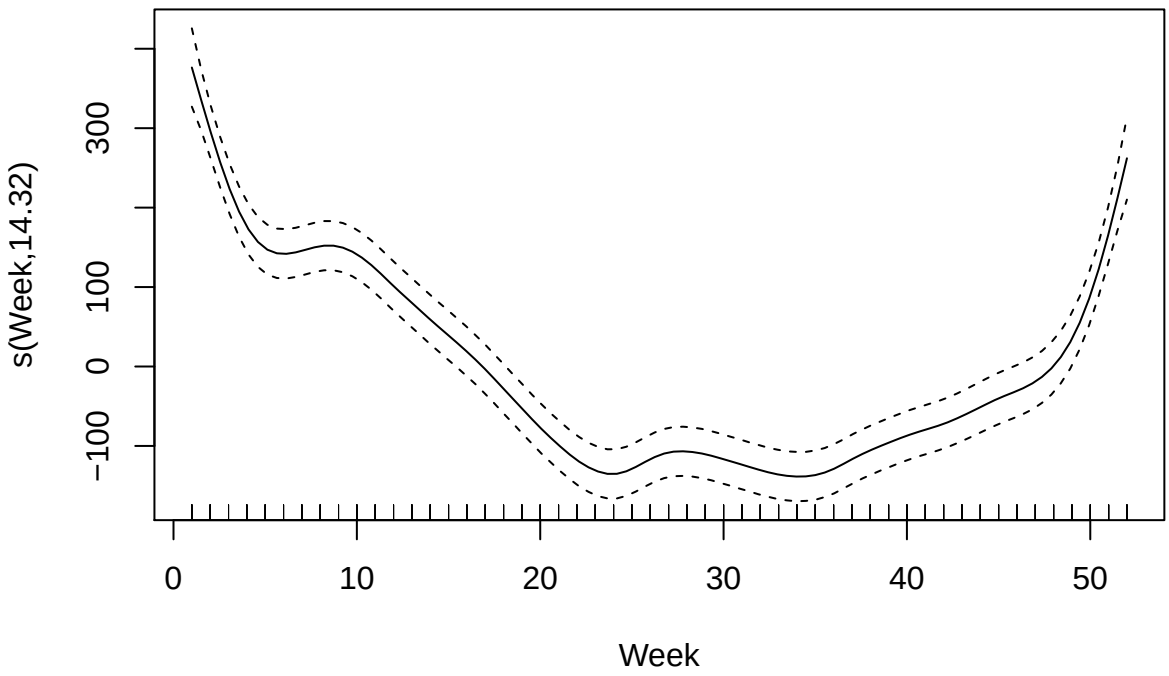
$$Mortality = \ N(Year + s(Week), \sigma^2)$$

Part 3

```
# Assignment 1.3 ------------------------------------------------------------

df$pred_mortality <- predict(res, newdata = df)



df_plot <- df %>%
    select(Time, Mortality, pred_mortality) %>%
    gather(variable, value, -Time)

ggplot(df_plot) +
    geom_line(aes(x = Time, y = value, color = variable))
```


2400
2100
1800
1500
value
1995.0
1997.5
2000.0
2002.5
Time
variable
Mortality
pred_mortality

```
plot(res)
```



When we look at the comparative chart for the predicted and observed data, we see that it catches the trend in general. But we can not say that it is successful for high values. We can come to the conclusion that we have a general pattern here.

When we examine the summary, we can see that the spline function of the week gives more effect on the result and is the significant (based on p-values comparison)

There is a strong seasonality in the mortality variable and our predicted mortality captures a yearly pattern that is applied to each year. The GAM model have an adjusted R-square value of 0.68.

The model is based on two variables, year and week, were week have a smoothing spline function applied to it. The Week variable with a smoother function is significant as a predictor with significance level towards 0.

From the second plot of the smoothing spline, the confidence interval is narrow. If it would be possible to fit a horizontal line within the confidence band, that variable would not be significant.

The smoothing spline function have a high positive value in the beginning and end of the year and a negative value during the summer weeks. The mortality rate follow the same pattern. Since the GAM-model is fitting linear combinations to explain the trend over the year, it seem intuitive that the smoothing spline function reduces the predicted effect during summer weeks and increases during winter weeks.

Mortality rates are changing over the years, however there seems to be a cyclical pattern - higher mortality rates occure in similar time intervals.

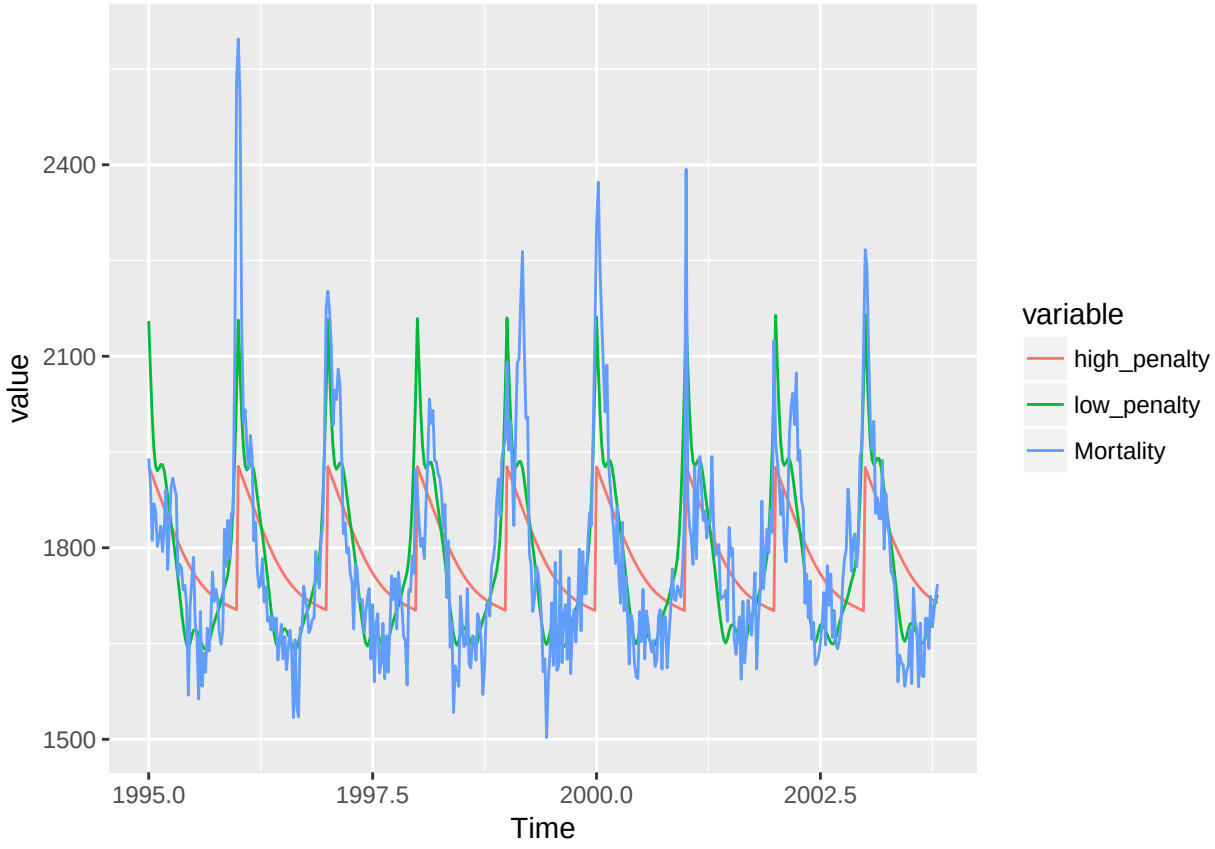
When we look at the graph of the spline component, we have proven that the supline function of the week is significant. The graph we obtained is quite similar to the graph of Mortality.

Part 4

```
# Assignment 1.4 ------------------------------------------------------------

res$sp
```

```
##     s(Week)
## 0.000113193
```

```
res_high_sp <- gam(formula = Mortality ~ Year + s(Week), data = df, sp = 100,family = gaussian(),method
df$pred_mortality_high_sp <- predict(res_high_sp, newdata = df)

df_plot2 <- df %>%
    select(Time,
           Mortality, low_penalty = pred_mortality,
           high_penalty = pred_mortality_high_sp) %>%
    gather(variable, value, -Time)

ggplot(df_plot2) +
    geom_line(aes(x = Time, y = value, color = variable))
```

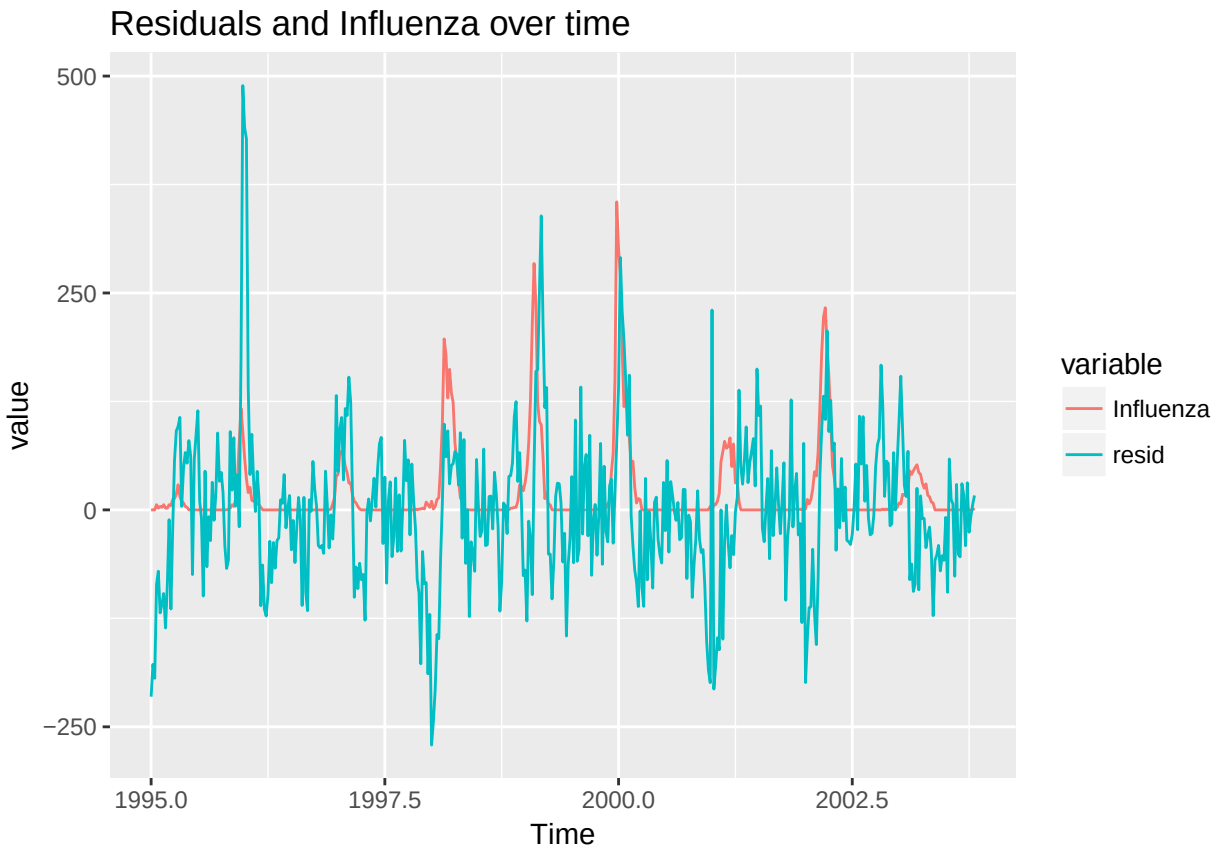


The above graph shows the mortality rate and two predicted mortality rates, one prediction with a high penalty factor and one with a low penalty factor which is the generated by cross validation. The high penalty value is an arbitrary high number, in this case 100 and the low value is approximatly 0.0038. As one can see from the graph, the low peanlty value have more variation within it's prediction but a high bias. The high penalty value have a low variance but a high variance, which can be seen since each year have more or less the same predicted results even though the true mortality rate differs inbetween years.

Degrees of freedom decrease as the penalty is increased. That is why the model with high penalty produces predictions that follow monotonic pattern - they do not have the freedom to move and adjust as much as the model with low penalty.

Part 5

```
# Assignment 1.5 ------------------------------------------------------------

df$resid <- res$residuals

df_plot3 <- df %>%
    select(Time, Influenza, resid) %>%
    gather(variable, value, -Time)

ggplot(df_plot3) +
    geom_line(aes(x = Time, y = value, color = variable)) +
    labs(title = "Residuals and Influenza over time")
```




```
cor(df$resid, df$Influenza)
```

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

```
#Is the temporal pattern in the residuals correlated to the outbreaks of influenza?
```

When reviewing the graph, it seems like the there is a correlation between the residuals and outbreaks in influenza. When testing the correlation between the two variables, it can be seen that the correlation is

approximately 0.33. Since there seem to be a relationship between the residuals and the inflenza variable, it's an indication that influenza would improve the model.

The average residual in the model is very close to zero, an indication that the model isn't over- or underfitted.

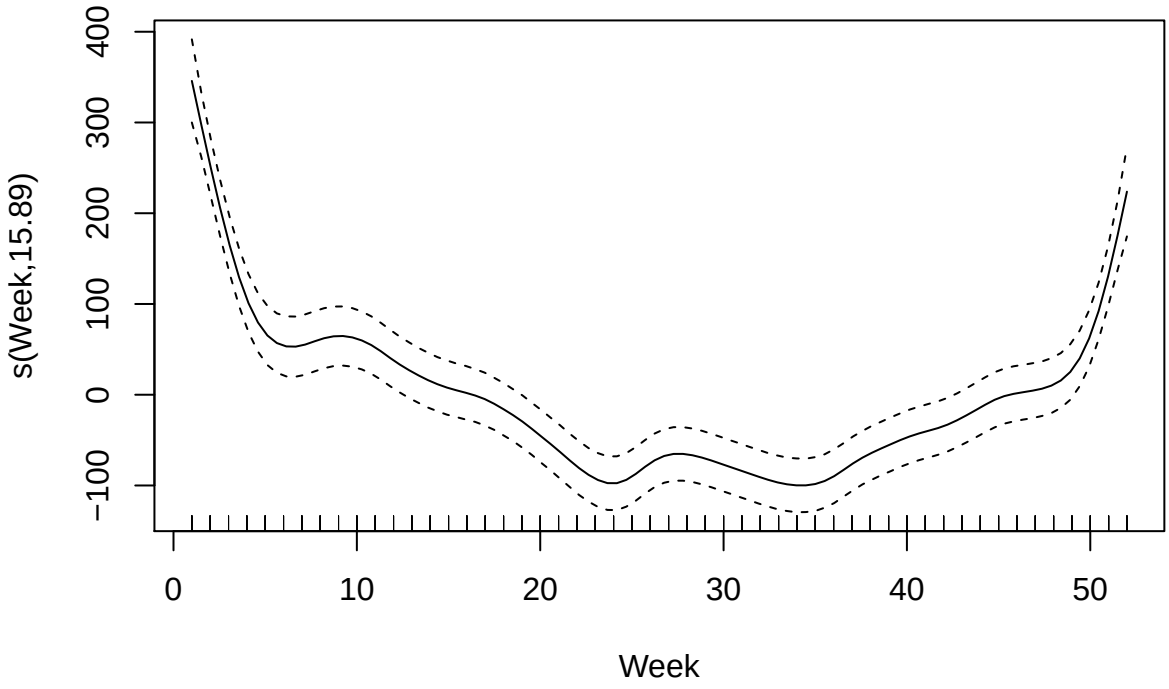
From the plot of the residuals against influenza cases seem that the high values of residuals tend to occur with high values of influenza cases. Hence, the errors in the prediction of the spline methods may be related to the influenza outbreaks.

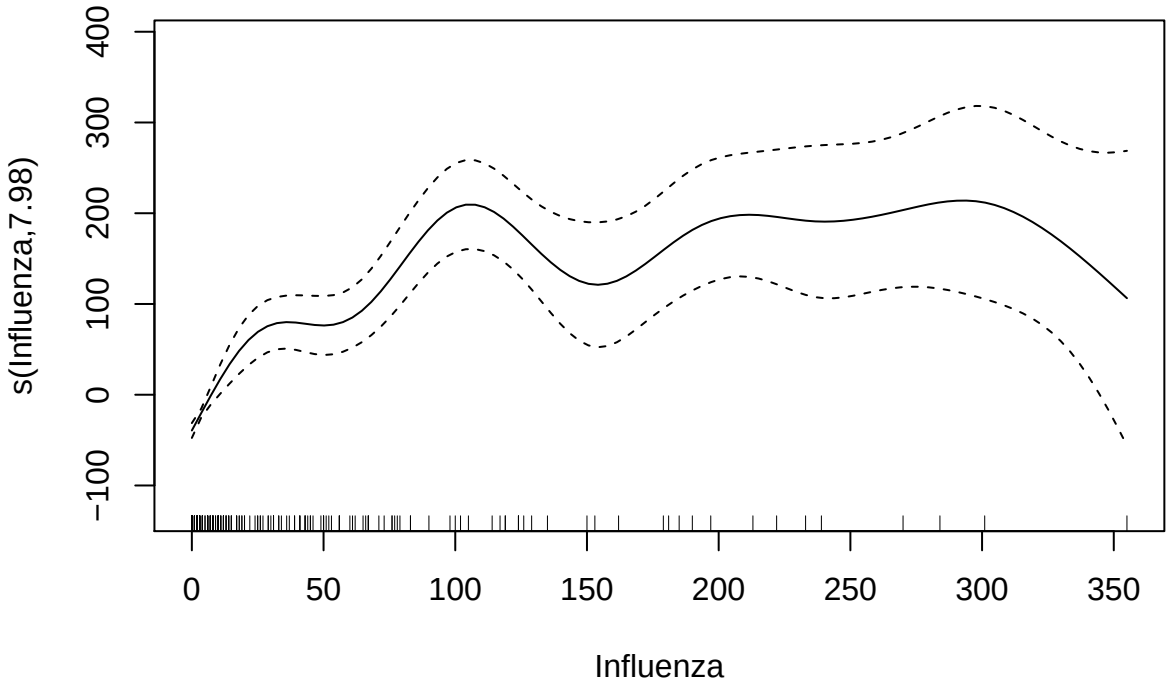
Part 6

```
# Assignment 1.6 ------------------------------------------------------------

unique_week <- length(unique(df$Week))

res2 <- gam(formula = Mortality ~ Year + s(Week, k = unique_week) + s(Influenza), data = df)
summary(res2)
```

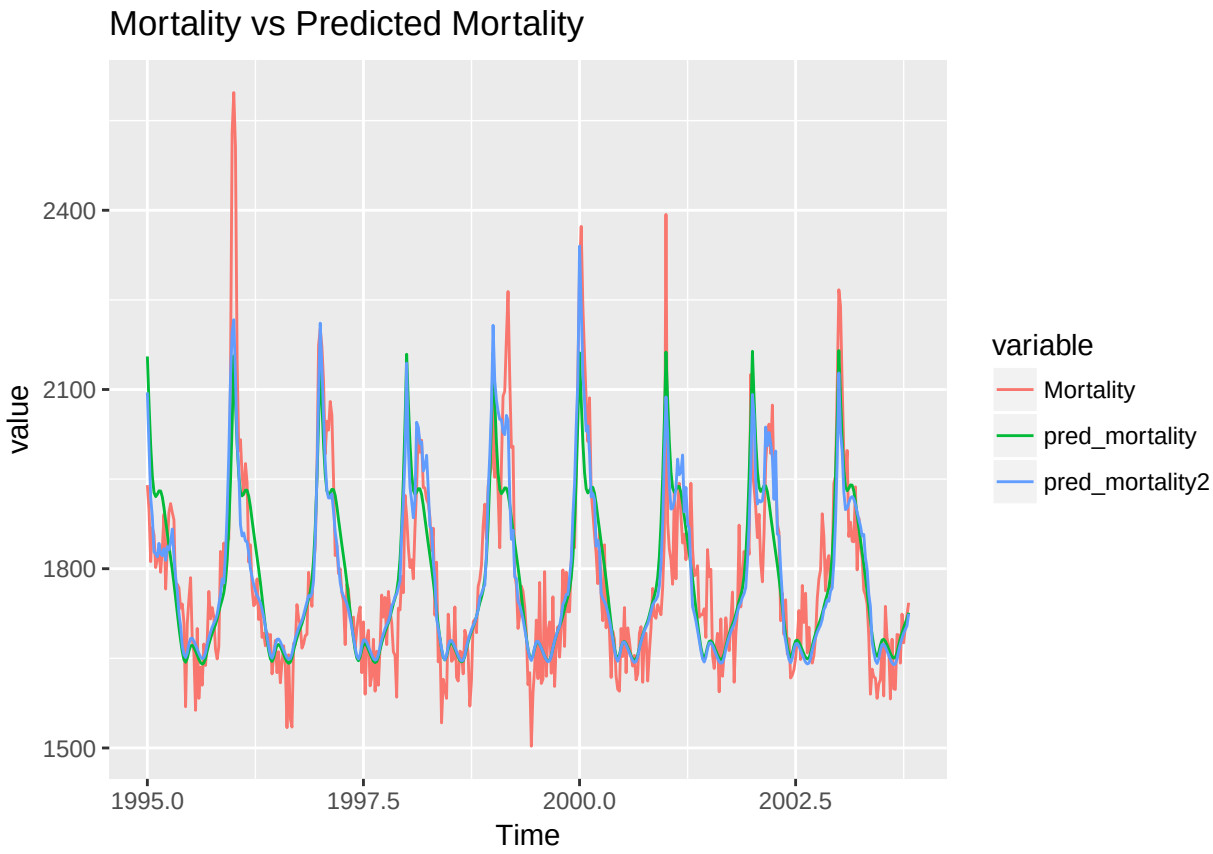
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = unique_week) + s(Influenza)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4314.061   2965.887   1.455    0.147
## Year          -1.266      1.484  -0.853    0.394
##
## Approximate significance of smooth terms:
##                 edf Ref.df     F p-value
## s(Week)      15.888 19.792 21.77  <2e-16 ***
## s(Influenza)  7.981  8.703 17.25  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 59/62
## R-sq.(adj) =  0.758   Deviance explained = 77.1%
## GCV = 6669.2  Scale est. = 6293.3    n = 459
```

```
plot(res2)
```


s(Week,15.89)
−100
0
100
200
300
400
0
10
20
30
40
50
Week




```
df$pred_mortality2 <- predict(res2, newdata = df)

df_plot4 <- df %>%
    select(Time, Mortality, pred_mortality, pred_mortality2) %>%
    gather(variable, value, -Time)

ggplot(df_plot4) +
    geom_line(aes(x = Time, y = value, color = variable)) +
    labs(title = "Mortality vs Predicted Mortality")
```



The summary shows that in this model the spline parameter of Week and Influenza are significant (p-value is close to 0), so the moratlity rate depends on week and influenza cases. The Year spline parameter is not significant in this case as p-valye is 0.181. Weeks and Years are very related and hence it makes sense that only one of these observations are significant in the model.

When the variable influenza is added to the second GAM-model, there is a increase in deviance explained by the model and the adjusted R-squared is higher compared to the first model without influenza. When visually looking at the graph, it seems the new prediction model outperforms the old one in peaks of the mortality rate and apart from the peaks, the two prediction models seem to follow a similar pattern.

In the spline graphs, it's impossible to fit a horizontal line within the confidence interval, which means that the variables is significant.

Assignment 2

Part 1

First a function for building a confusion matrix is built.

```
confusionmatrix <- function(pred, df_true){
    df <- data.frame(x = df_true[,confindex], y = pred)
    tab <- addmargins(table(df$x, df$y))
    tab2 <- table(df$x, df$y)
    error <- 1 - sum(diag(tab2))/sum(tab2)

    l <- list(confusion_matrix = tab,
              missclassification_rate = error)
    return(l)
}
```

Nearest Shrunken Centriod is used to estimate whether the email contains an invitation to a conference or not. The model is developed and trained by the pamr packaged and then cross validated. Out of the cross validated results the threshold for the minimum error is selected, here 1.306. This threshold is used when generating the graph of the results.

```
# Assignment 2.1 ------------------------------------------------------------
df <- as.data.frame(read_delim("data.csv", ";", escape_double = FALSE, trim_ws = TRUE, locale = locale(
df$Conference <- as.factor(df$Conference)

#colnames(df)[4703]
confindex <- 4703

set.seed(12345)

index <- sample(1:nrow(df), floor(nrow(df) * 0.7))
train <- df[index,]
test <- df[-index,]

model_df <- list(x = t(train[,-confindex]), y = train[,confindex],
                 geneid = as.character(1:ncol(train[,-confindex])),
                 genenames = rownames(t(train)))

model <- pamr.train(model_df)
```

```
## 1234567891011121314151617181920212223242526272829 30
```

```
cvmodel <- pamr.cv(model, model_df)
```

```
## 12Fold 1 :1234567891011121314151617181920212223242526272829 30
## Fold 2 :1234567891011121314151617181920212223242526272829 30
## Fold 3 :1234567891011121314151617181920212223242526272829 30
## Fold 4 :1234567891011121314151617181920212223242526272829 30
## Fold 5 :1234567891011121314151617181920212223242526272829 30
## Fold 6 :1234567891011121314151617181920212223242526272829 30
## Fold 7 :1234567891011121314151617181920212223242526272829 30
## Fold 8 :1234567891011121314151617181920212223242526272829 30
## Fold 9 :1234567891011121314151617181920212223242526272829 30
## Fold 10 :1234567891011121314151617181920212223242526272829 30
```

```
cvmodel
```

```
## Call:
## pamr.cv(fit = model, data = model_df)
##    threshold nonzero errors
## 1  0.000     3428    6
## 2  0.145     3162    7
## 3  0.290     3024    7
## 4  0.435     2021    7
## 5  0.580      862    6
## 6  0.726      840    6
## 7  0.871      658    6
## 8  1.016      296    6
## 9  1.161      290    6
## 10 1.306      231    5
## 11 1.451      154    6
## 12 1.596      129    7
## 13 1.741       97    7
## 14 1.886       71    7
## 15 2.031       62    7
## 16 2.177       43    7
## 17 2.322       36    7
## 18 2.467       26    7
## 19 2.612       20    7
## 20 2.757       12    7
## 21 2.902       12    9
## 22 3.047       11    9
## 23 3.192        9    10
## 24 3.337        9    11
## 25 3.482        6    12
## 26 3.628        6    14
## 27 3.773        4    16
## 28 3.918        2    20
## 29 4.063        1    19
## 30 4.208        0    19
```

```
length(cvlist[,1])
```

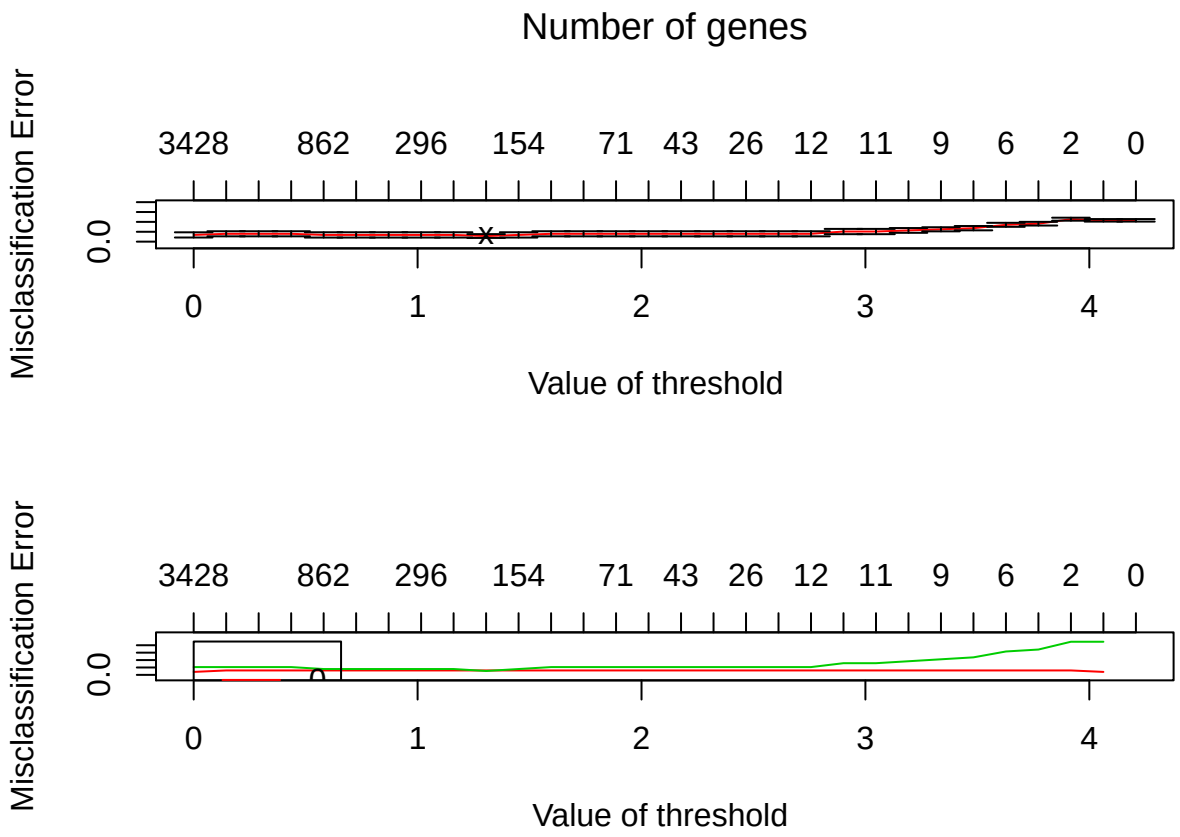
```
## [1] 231
```

```
cat(paste(colnames(df)[as.numeric(cvlist[1:10, 1])], collapse = "\n"))
```

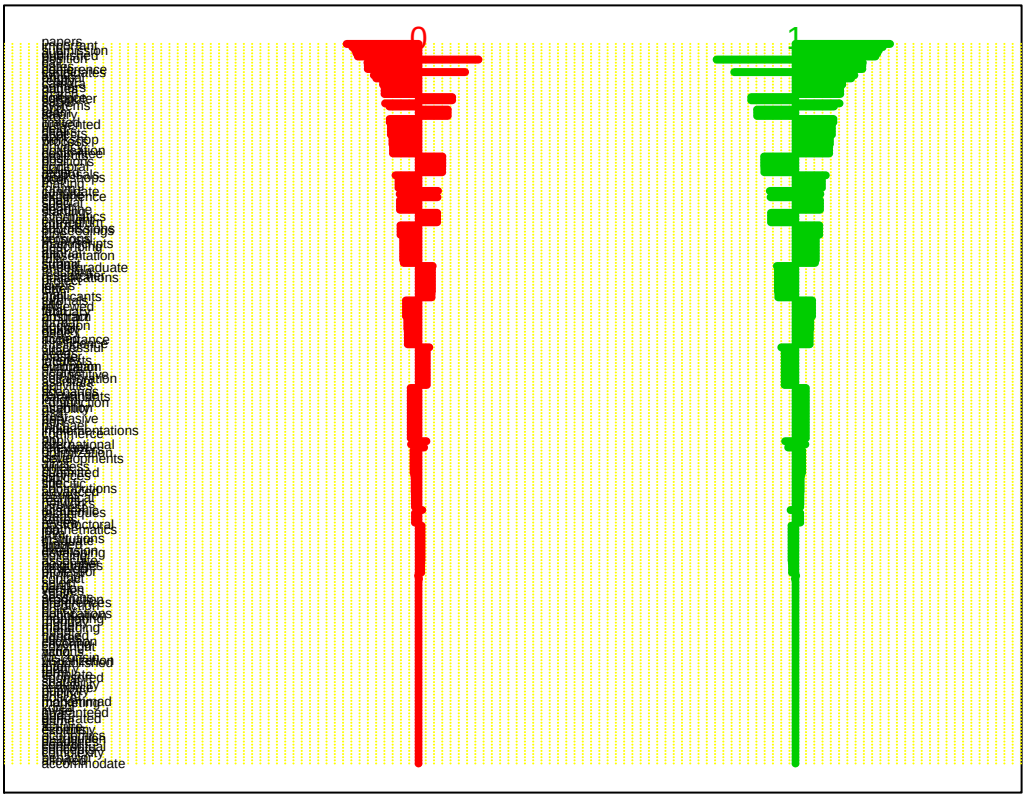
```
## papers
## important
## submission
## due
## published
## position
## call
## conference
## dates
## candidates
```

When the threshold from the cross validated model is used, the Nearest Shrunken Centroid uses 231 variables. The ten variables with most impact is listed above.

```
pamr.plotcv(cvmodel)
```




```
pamr.plotcen(model, model_df, threshold = cvmodel$threshold[10])
```




```
pred_NSC <- pamr.predict(model,
             newx = (t(test[,-confindex])),
             threshold = cvmodel$threshold[10],
             type = "class")

confusionmatrix(pred_NSC, test)
```

```
## $confusion_matrix
##
##        0  1 Sum
##   0   10  0  10
##   1    2  8  10
##   Sum 12  8  20
##
## $missclassification_rate
## [1] 0.1
```

The Nearest Shrunk Centroids model have a missclassification rate at 10%.

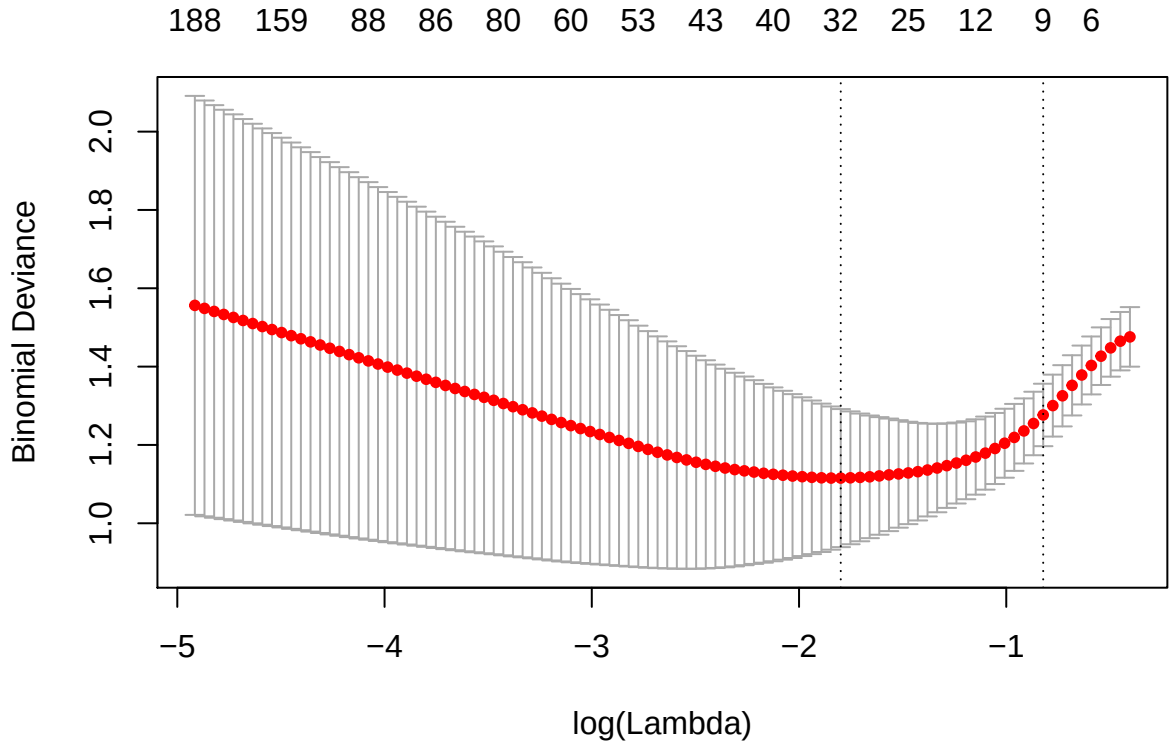
The centroid plot shows the centroid for each dimension (variable) in the data set. The red bars tells how the centroid cluster of zeros in variable is moved after the threshold have been removed. The same goes for the green bar with the cluster centroid for ones. If the threshold is bigger than the value, so the centroid is move to zero, the variable can be removed from the prediction model. The variables with biggest distances between the centroids is the ones that discriminate the best and is the best for the prediction. They are on top in descending order.

It's hard to tell whether these words is useful for discriminate whether the email contains an conference invitation. However the top ten words is usual words when applying for something, or deciding a time for a

meeting etc, so the selected variables seems reasonable.

Part 2a

```
# Assignment 2.2 a ------------------------------------------------------------
x <- model.matrix(Conference ~ ., data = train)
cv_glm <- cv.glmnet(x = x,
                    y = as.matrix(train[, confindex]),
                    family = "binomial", alpha = 0.5)
# cv_glm$lambda.min
plot(cv_glm)
```




```
glm_final <- glmnet(x = x,
                    y = as.matrix(train[, confindex]),
                    family = "binomial", alpha = 0.5,
                    lambda = cv_glm$lambda.min)

# pred_train <- as.numeric(predict(cv_glm, newx = as.matrix(train[,-confindex]), s = "lambda.min", type
pred_test <- as.numeric(predict(glm_final,
                                newx = model.matrix(Conference ~ ., data = test),
                                s = "lambda.min", type = "class"))

# confusionmatrix(pred_train, train)
confusionmatrix(pred_test, test)
```

```
## $confusion_matrix
##
##        0  1 Sum
##   0   10  0  10
##   1    2  8  10
##   Sum 12  8  20
##
## $missclassification_rate
## [1] 0.1
```

```
length(coef(glm_final, s = "lambda.min")@x)
```

```
## [1] 33
```

A elastic net built by cross validation. The confusion matrix above is reported. The missclassification rate is 10 % and the elastic net uses 33 variables in the prediction.

Here, the number of coeficients is computed but one can also extract the degrees of freedom + 1 (intercept).

Part 2b

```
# Assignment 2.2 b ------------------------------------------------------------
svm <- ksvm(x = as.matrix(train[, -confindex]),
            y = as.matrix(train[, confindex]),
            kernel = "vanilladot",scale=FALSE)
```

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

```
pred_svm <- predict(svm, as.matrix(test[, -confindex]), type = "response")

confusionmatrix(pred_svm, test)
```

```
## $confusion_matrix
##
##        0  1 Sum
##   0   10  0  10
##   1    1  9  10
##   Sum 11  9  20
##
## $missclassification_rate
## [1] 0.05
```

```
svm
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 43
##
## Objective Function Value : -2.0817
## Training error : 0.022727
```

A support vector machine model is built with a linear (vanilla dot) kernel. The SVM have a missclassification rate 5 % and uses 43 support vectors in the model.

The result for the Nearest Shrunken Centriod and the elasitic net is identical with a missclassification rate of 10 % and missclassifies in the same pattern. The Support Vector Machine is sligtly better, with a missclassification rate of 5 %. However the sample size is really small and the test data only contains 20 observations. If more data would have been available, there might have been bigger difference between the models. One should be careful when drawing conclusions / evaluating models with this small amount of data.

Model	Error_rate
NSC	0.10
Elastic Net	0.10
SVM	0.05

Part 3

First the Benjamini-Hochberg method is implemented, by computing the p-values for all variables. The BH-method adjusts the p-value and helps avoiding type 1 errors, rejecting the true null hypothesis.

The Benjamini-Hochberg method let you know which variables is significantly different from H_0. All variables were the p-value is lower than the computed Benjamini-Hochberg value is significant, even those variables that have a p-value higher than $1 - \alpha$.

```
set.seed(12345)
conf_col <- which(names(df) == "Conference")
xb=df[,-conf_col]
yb=df[,conf_col]
pval <- c()
for (i in 1:ncol(xb)){
    t <- t.test(xb[,i]~yb)
    pval[i] <- t$p.value
}

p_df <- data.frame(P_value = pval, Feature = colnames(xb))
# Order p-values ascending
p_df  <- p_df[order(p_df$P_value),]

# Let alpha be 0.05
M <- nrow(p_df)
L <-c() # initial value
alpha <- 0.05
j <- 1

for (i in 1:M){
    Q <- alpha * (i/M)
   if (p_df[i,1] < Q){
       L[j] <- i
       j <- j+1
   }

}

L <- max(L)
```

```
p_threshold <- p_df[L,1]
```

So, the threshold for rejecting H_0 is:

```
## [1] 0.0003765147
```

All p-values that are below this threshold, will cause the H_0 being reject - the corresponding features are significant by the Benjamini-Hochberg method. In this case, 39 variables are considered being significant and the rest are insignificant. This can be compared to the 281 variables which would be considered significant if the ordinary t-test would have been used.

```
non_signif_feat <- subset(p_df, p_df$P_value > p_threshold)
signif_feat <- subset(p_df, p_df$P_value <= p_threshold)
```

After doing Benjamini-Hochberg algorithm for testing, we conclude that 4663 features out of 4702 are insignificant, and 39 features are significant. The list of the significant features is below:

```
signif_feat
```

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

```
## 1477 3.762328e-04     excellent
## 3191 3.762328e-04          post
## 3243 3.765147e-04     presented
```

This result seems reasonable as most of these features were used in nearest shrunken centroid and elastic net classifications. Both of the models had low prediction errors, hence the keywords do seem to contain the information of the underlying true model of the data.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(readxl)
library(ggplot2)
library(tidyverse)
library(mgcv)
library(pamr)
library(glmnet)
library(kernlab)

set.seed(12345)
df <- read_excel("Influenza.xlsx")
# Assignment 1.1 ------------------------------------------------------------

std_df <- df %>%
    mutate(std_mortality = scale(.$Mortality),
           std_influenza = scale(.$Influenza)) %>%
    gather(variable, value, -Year, -Week, -Time, -`Temperature deficit`)

# Standardized data
ggplot(filter(std_df, variable == "std_influenza" |
                  variable == "std_mortality")) +
    geom_line(aes(x = Time, y = value, color = variable)) +
    labs(x = "Time (Years)", y = "Scaled data")
# Assignment 1.2 ------------------------------------------------------------

res <- gam(Mortality ~ Year + s(Week,k=52),data=df,family = gaussian(),method = "GCV.Cp")
summary(res)
# Assignment 1.3 ------------------------------------------------------------

df$pred_mortality <- predict(res, newdata = df)



df_plot <- df %>%
    select(Time, Mortality, pred_mortality) %>%
    gather(variable, value, -Time)

ggplot(df_plot) +
    geom_line(aes(x = Time, y = value, color = variable))

plot(res)
# Assignment 1.4 ------------------------------------------------------------
```

```
res$sp
res_high_sp <- gam(formula = Mortality ~ Year + s(Week), data = df, sp = 100,family = gaussian(),method
df$pred_mortality_high_sp <- predict(res_high_sp, newdata = df)

df_plot2 <- df %>%
    select(Time,
           Mortality, low_penalty = pred_mortality,
           high_penalty = pred_mortality_high_sp) %>%
    gather(variable, value, -Time)

ggplot(df_plot2) +
    geom_line(aes(x = Time, y = value, color = variable))

# Assignment 1.5 ------------------------------------------------------------

df$resid <- res$residuals

df_plot3 <- df %>%
    select(Time, Influenza, resid) %>%
    gather(variable, value, -Time)

ggplot(df_plot3) +
    geom_line(aes(x = Time, y = value, color = variable)) +
    labs(title = "Residuals and Influenza over time")

cor(df$resid, df$Influenza)
#Is the temporal pattern in the residuals correlated to the outbreaks of influenza?
# Assignment 1.6 ------------------------------------------------------------

unique_week <- length(unique(df$Week))

res2 <- gam(formula = Mortality ~ Year + s(Week, k = unique_week) + s(Influenza), data = df)
summary(res2)
plot(res2)

df$pred_mortality2 <- predict(res2, newdata = df)

df_plot4 <- df %>%
    select(Time, Mortality, pred_mortality, pred_mortality2) %>%
    gather(variable, value, -Time)

ggplot(df_plot4) +
    geom_line(aes(x = Time, y = value, color = variable)) +
    labs(title = "Mortality vs Predicted Mortality")
confusionmatrix <- function(pred, df_true){
    df <- data.frame(x = df_true[,confindex], y = pred)
    tab <- addmargins(table(df$x, df$y))
    tab2 <- table(df$x, df$y)
    error <- 1 - sum(diag(tab2))/sum(tab2)

    l <- list(confusion_matrix = tab,
              missclassification_rate = error)
    return(l)
```

```
}
# Assignment 2.1 ------------------------------------------------------------
df <- as.data.frame(read_delim("data.csv", ";", escape_double = FALSE, trim_ws = TRUE, locale = locale(
df$Conference <- as.factor(df$Conference)

#colnames(df)[4703]
confindex <- 4703

set.seed(12345)

index <- sample(1:nrow(df), floor(nrow(df) * 0.7))
train <- df[index,]
test <- df[-index,]

model_df <- list(x = t(train[,-confindex]), y = train[,confindex],
                 geneid = as.character(1:ncol(train[,-confindex])),
                 genenames = rownames(t(train)))

model <- pamr.train(model_df)
cvmodel <- pamr.cv(model, model_df)
cvmodel
cvlist <- pamr.listgenes(model, model_df, threshold = cvmodel$threshold[10])
length(cvlist[,1])

cat(paste(colnames(df)[as.numeric(cvlist[1:10, 1])], collapse = "\n"))
pamr.plotcv(cvmodel)

pamr.plotcen(model, model_df, threshold = cvmodel$threshold[10])

pred_NSC <- pamr.predict(model,
             newx = (t(test[,-confindex])),
             threshold = cvmodel$threshold[10],
             type = "class")

confusionmatrix(pred_NSC, test)
# Assignment 2.2 a ----------------------------------------------------------
x <- model.matrix(Conference ~ ., data = train)
cv_glm <- cv.glmnet(x = x,
                    y = as.matrix(train[, confindex]),
                    family = "binomial", alpha = 0.5)
# cv_glm$lambda.min
plot(cv_glm)

glm_final <- glmnet(x = x,
              y = as.matrix(train[, confindex]),
              family = "binomial", alpha = 0.5,
              lambda = cv_glm$lambda.min)

# pred_train <- as.numeric(predict(cv_glm, newx = as.matrix(train[,-confindex]), s = "lambda.min", type
pred_test <- as.numeric(predict(glm_final,
                                newx = model.matrix(Conference ~ ., data = test),
                                s = "lambda.min", type = "class"))
```

```r
# confusionmatrix(pred_train, train)
confusionmatrix(pred_test, test)
length(coef(glm_final, s = "lambda.min")@x)
# Assignment 2.2 b ------------------------------------------------------
svm <- ksvm(x = as.matrix(train[, -confindex]),
            y = as.matrix(train[, confindex]),
            kernel = "vanilladot",scale=FALSE)

pred_svm <- predict(svm, as.matrix(test[, -confindex]), type = "response")

confusionmatrix(pred_svm, test)
svm
tab <- data.frame(Model = c("NSC", "Elastic Net", "SVM"),
                  Error_rate = c(confusionmatrix(pred_NSC, test)[[2]],
                                 confusionmatrix(pred_test, test)[[2]],
                                 confusionmatrix(pred_svm, test)[[2]]))
knitr::kable(tab)
set.seed(12345)
conf_col <- which(names(df) == "Conference")
xb=df[,-conf_col]
yb=df[,conf_col]
pval <- c()
for (i in 1:ncol(xb)){
    t <- t.test(xb[,i]~yb)
    pval[i] <- t$p.value
}

p_df <- data.frame(P_value = pval, Feature = colnames(xb))
# Order p-values ascending
p_df  <- p_df[order(p_df$P_value),]

# Let alpha be 0.05
M <- nrow(p_df)
L <-c() # initial value
alpha <- 0.05
j <- 1

for (i in 1:M){
    Q <- alpha * (i/M)
   if (p_df[i,1] < Q){
       L[j] <- i
       j <- j+1
   }

}

L <- max(L)
p_threshold <- p_df[L,1]
p_threshold
non_signif_feat <- subset(p_df, p_df$P_value > p_threshold)
signif_feat <- subset(p_df, p_df$P_value <= p_threshold)

signif_feat
```