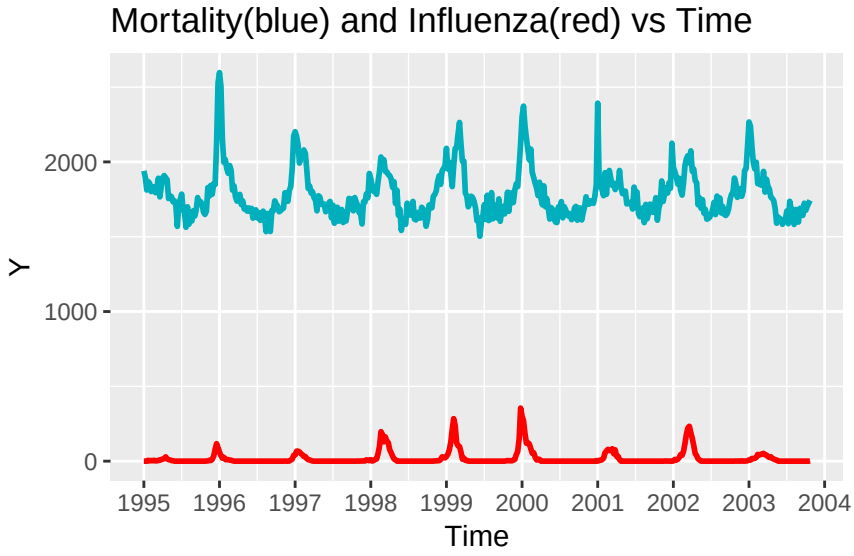
Lab2_Block2

Andreas Stasinakis

December 11, 2018

Assignment 1

Task 1.1



Analysis

From the plot, we can observe that that the amounts of influenza follow the same trend but with different values and lower peaks. We can also mention that most of the peaks for both mortality and influenza are in the beginning of the year and in order to be more precise during winter. This is logical because the probability of someone being sick is much higher in the winter than in other seasons. Moreover, for mortality after every peak (higher value 2597) there is an exponentially reduction, reaching close to 1500 during the spring, before increasing again until the end of each year. Although the amounts of influenza follows the same trend as mortality, the peaks are really low (highest 355 in 2000). Except for the lower peaks, another important difference is that influenza after reaching the lowest value, which is 0, stabilizes there for the rest of the year before starting rising again. Finally both of mortality and influenza values have only one peak every year.

Task 1.2

Analysis

In this task we fit a Generalized additive model in which the Mortality is the response variable and it is modeled as a linear function of Year and spline function of Week. We also assuming that the response variable Mortality is normally distributed and $y \sim N(\mu, \sigma^2)$ and also Y are i.i.d. The general model we fit the data is :

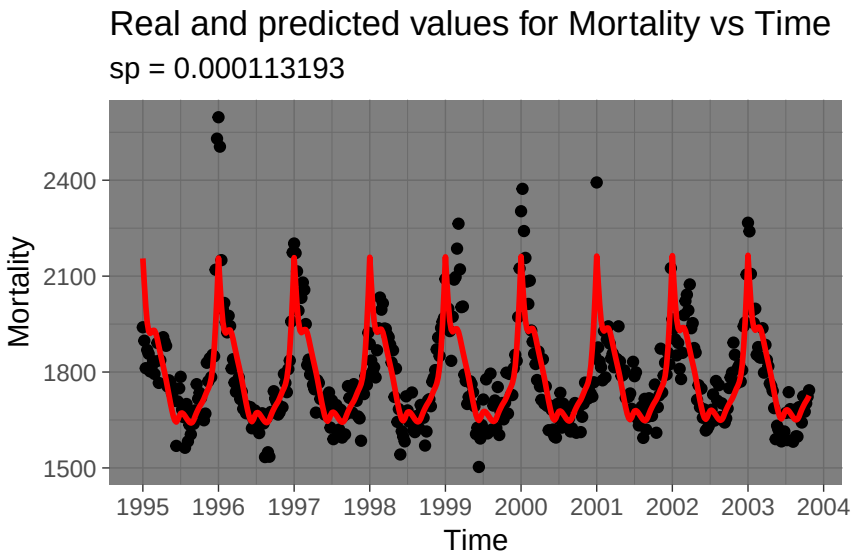
$$g(\mu) = \alpha + s_1(X_1) + ... + s_p(X_p) + \sum_{j=1}^{q} \beta_j X_{p+j}$$

where $g(\mu)$ is the link function and $s_i(X)$ the smoothers, normally splines.

The estimated model we fit the model is the following :

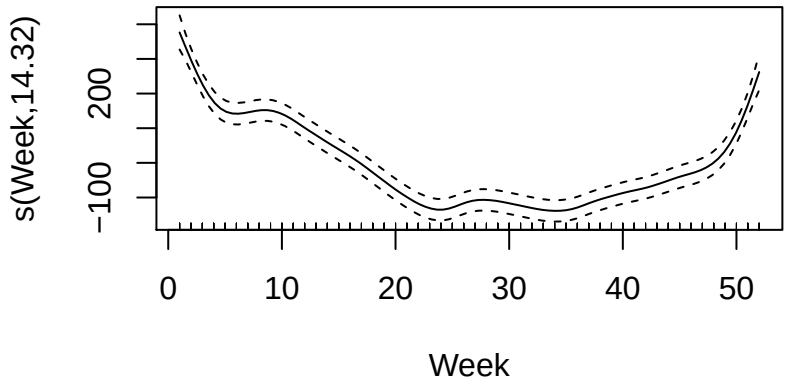
$$\hat{Y}_i = -680.598 + 1.232846 X_{i,1} + \hat{s_2}(X_{i,2})$$

Task 1.3




```
##
## 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
```

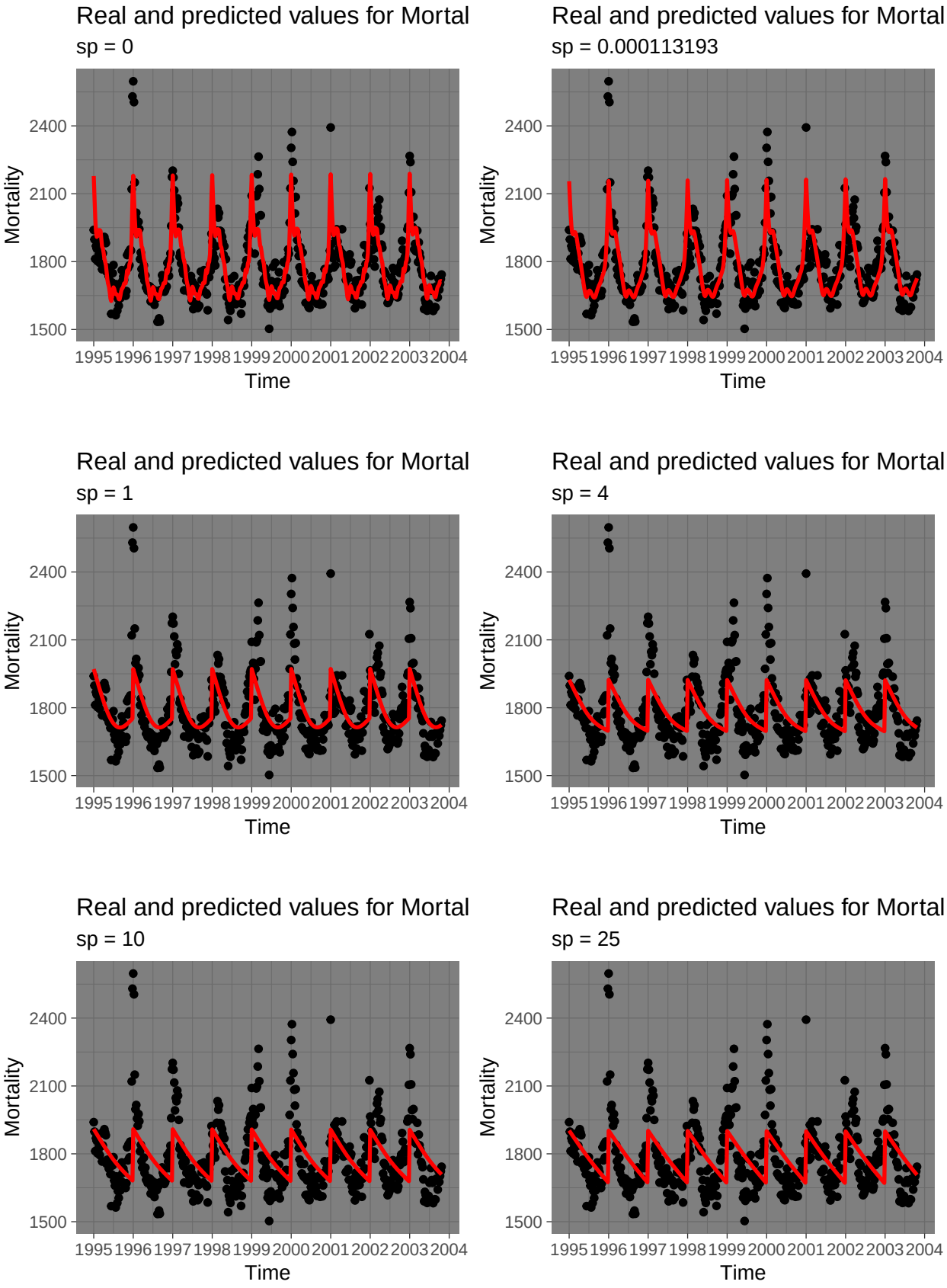


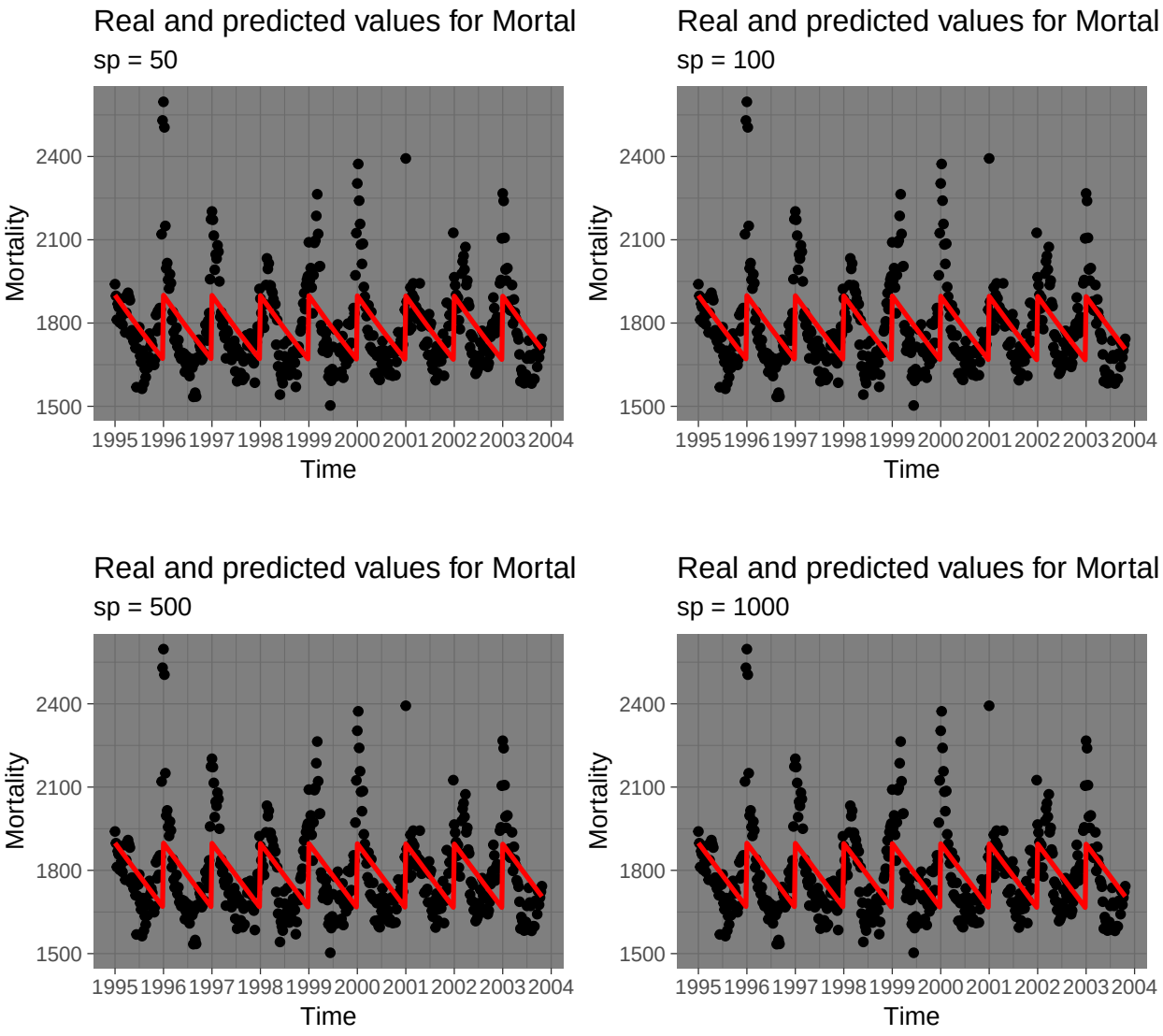
Analysis

From the first plot we can comment for the quality of the fit. It seems that the GAM model fits the data efficient enough. The only points that are not predicted correctly are the peaks points for the mortality and some of the lowest values, but in general the quality of the fit is really good. We also take many information from the summary of the model. It can be seen that the p - values of the spline have 3 starts which means that there are significant values as a result the term which appear to be significant in the model is the spline of the week. From the second plot, as mentioned before, there is a trend in mortality during one year and another. More specific, in most cases after reaching a peak, in the beginning of the year, reduces exponentially until the middle of the year and starts rising again until the end of the year.

Task 1.4

```
##             sp deviance degrees_of_freedom
## 1  0.00000e+00  3645526          27.000000
## 2  1.13193e-04  3718012          14.321891
## 3  1.00000e+00  6593860           1.491752
## 4  4.00000e+00  8553032           1.161915
## 5  1.00000e+01  9274216           1.069458
## 6  2.50000e+01  9618101           1.028626
## 7  5.00000e+01  9741090           1.014460
## 8  1.00000e+02  9804267           1.007268
## 9  5.00000e+02  9855642           1.001460
## 10 1.00000e+03  9862116           1.000730
```

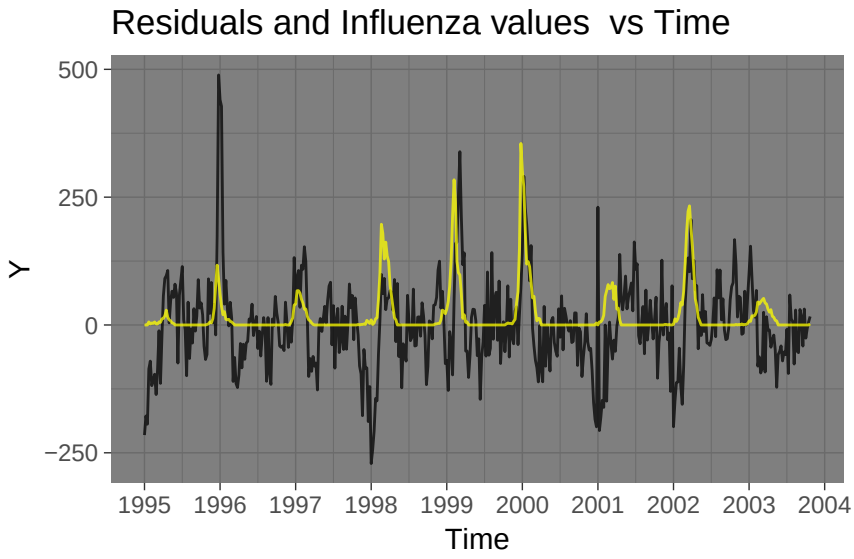

Real and predicted values for Mortal
sp = 0
Real and predicted values for Mortal
sp = 0.000113193
Real and predicted values for Mortal
sp = 1
Real and predicted values for Mortal
sp = 4
Real and predicted values for Mortal
sp = 10
Real and predicted values for Mortal
sp = 25
Mortality
2400
2100
1800
1500
1995 1996 1997 1998 1999 2000 2001 2002 2003 2004
Time



Analysis

From the printed data frame we can conclude that while the penalty factor of the spline increases, the deviance also increases exponentially, until a *sp* value after which the changes in the deviance are small. In this task we also plot the real and predicted values for many different penalty factors. It can be said that while the penalty factor is increasing, the quality of the fit is decreasing. We can also mention that for *sp* values close to 0 the fit is really good. For values close to $sp = 1$, despite the fact that the curve of the predictions is still smooth, the peaks of the model are smaller and the lowest predicted values substantially overestimate the real ones. For higher values for the penalty factor, the model is poorly fit to the data. The predictions' graph is a composition of linear functions. We can also mention that for really high values of the penalty factor, the plot does not change significantly. Finally, we know from the theory that the penalty factor is inversely proportional to the degrees of freedoms. From the data frame above we can verify this statement and we can observe that for values higher than one of the penalty factor, the degrees of freedom decrease but not that substantially.

Task 1.5



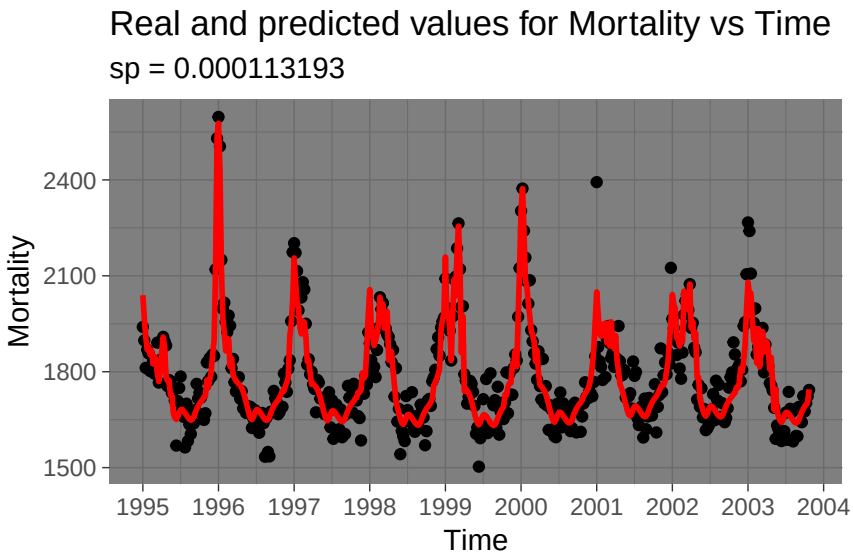
Analysis

In this task we plot both the residuals for the *GAM* model and the amounts of influenza vs Time. For the residuals we can observe that they have a great variate of different values between -250 and 500. For this kind of data and response variable, Y is normally distributed, the residuals should be normally distributed too, but in this case we can not assume that. Moreover, the residuals *should* not have a clear pattern, something which is happening in this task. More specific, the value of the residuals increases or decreases in the beginning of the year and after reaching a peak continue in the opposite direction. This means that we can improve our model in order to fix the problems above. As far as the influenza concerned, we mentioned before that they follow the same trend as the mortality. For residuals and influenza, we can observe that they have some peaks for the same year but we can not say that there is a clear correlation between each other.

Task 1.6

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ s(Year, k = length(unique(data$Year))) + s(Week,
##     k = length(unique(data$Week))) + s(Influenza, k = length(unique(data$Influenza)))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1783.8        3.2   557.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df      F p-value
## s(Year)      4.663  5.677  1.487   0.181
## s(Week)     14.641 18.248 18.533  <2e-16 ***
## s(Influenza) 69.740 72.833  5.600  <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 134/144
## R-sq.(adj) =  0.819   Deviance explained = 85.4%
## GCV = 5846.7  Scale est. = 4699.8    n = 459
```

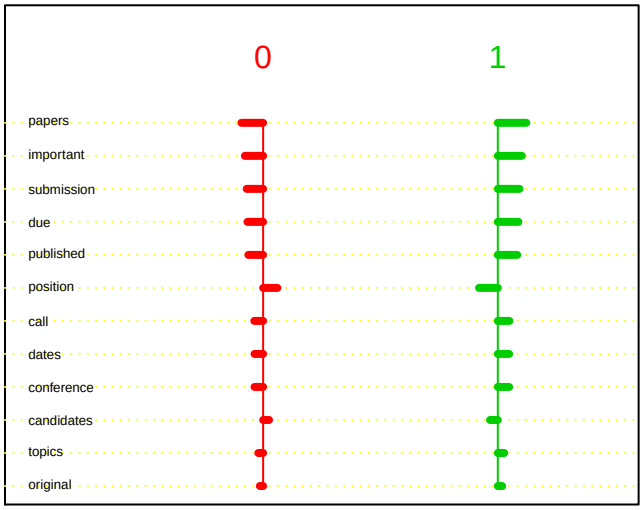


Analysis

In this task we fit again a *GAM* model but in this case the predictors are the spline functions of year, week, and the number of confirmed cases of influenza. From the plot above we can say that this model fits the data better than the one in task 1.2 because it can predict also most of the peaks correctly, while the model in 1.2 does not. Although this model will have a lower misclassification rate for this dataset, we can not really say which of them is the better one. The reason why is that we should test the model in unseen data in order to understand if it generalized the predictions and choose the one with the lowest generalization error. For example, the model in this task may have more accurate predictions for this dataset but if we use it for predictions in an unseen dataset and the misclassfication rate is high enough then this model is over fitting and we should choose the one in task 1.2. Finally, from the summary of the model, it can be said that the mortality is influenced by the outbreaks of influenza, because the p -value is really small and has *** which means that this variable is significant for the dependent variable.

Assignment 2

Task 2.1




```
##         words
## 1      papers
## 2   important
## 3  submission
## 4         due
## 5   published
## 6    position
## 7        call
## 8  conference
## 9       dates
## 10 candidates
```

Table 1: misclassification rate for test data and number of variables selected

misclassification_rate	features_selected
0.1	12

Analysis

In this task we perform *Nearest shrunken centroid classification* of training data in which the threshold is chosen by cross-validation. During the cross validation we choose the threshold with the minimum errors (6). There are many different thresholds with 6 errors so we choose the one with the less variables included. In the end the optimal threshold is 2.8 which uses 12 variables.

In general the *Nearest shrunken centroid classification* can improve the accuracy of the classifier and tries to eliminate the variables which are not that important for the prediction of the response variable. More specific if a predictor (in our case a specific "word") is shrunk to zero for all classes, then the classifier does not include it in the prediction procedure. As a result, in the centroid plot we can see the variables which are really important for the classifier. For the variable selected we can say that they are almost mutually exclusive between the two classes, except for maybe the variables in the first places of the plot.

From the list which include the 10 most contributing features, we can say that some of them, such as "conference" or "candidates" is logical to have a strong effect on the classification procedure, but most of the words are really common words in a lot of emails. For that reason we can not conclude that there is a reasonable connection between them and the classifier.

Task 2.2

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

Table 2: misclassification rates

	Misclassification_rates	Features_selected
NSC	0.10	12
Elastic_net	0.10	38
SVM	0.05	43

Analysis

In this task we fit two different models, an elastic net with the binomial response and $\alpha = 0.5$ in which penalty is selected by the cross-validation and one support vector machine with "vanilladot" kernel. We calculate their misclassification rates and we also make a comparative table including the misclassification rate for NSC too. First of all we have to mention that the model with the lower misclassification rate is the support vector machine (0.05) while the other two models have the same (0.1). Moreover, in order to decide which model is the best one we have to compare also the number of variables selected. For that reason, it is clear that *Elastic net* is the less efficient model because except for having the highest misclassification rate (same as NSC), the number of variables selected is 38 which makes the model more complex than *NSC*. Choosing between *NSC* and *SVM* is not that easy because despite the fact that *SVM* has lower misclassification rate, the *NSC* model is less complex. Therefore, if i had to choose one of them, i will probably choose the *NSC* because the difference in the misclassification rate is really small, only 0.05, but the *NSC* model is much less complex than the *SVM*, consists of only 12 variables in contrast to the 43 variables for *SVM*.

Task 2.3

Table 3: Features rejected from Benjamini-Hochberg method

features	p_values
papers	0.0000000
submission	0.0000000
position	0.0000000
published	0.0000002
important	0.0000003
call	0.0000004
conference	0.0000005
candidates	0.0000009
dates	0.0000014
paper	0.0000014
topics	0.0000051
limited	0.0000079
candidate	0.0000119
camera	0.0000210
ready	0.0000210

features	p_values
authors	0.0000215
phd	0.0000338
projects	0.0000350
org	0.0000374
chairs	0.0000586
due	0.0000649
original	0.0000649
notification	0.0000688
salary	0.0000797
record	0.0000909
skills	0.0000909
held	0.0001529
team	0.0001758
pages	0.0002007
workshop	0.0002007
committee	0.0002117
proceedings	0.0002117
apply	0.0002166
strong	0.0002246
international	0.0002296
degree	0.0003762
excellent	0.0003762
post	0.0003762
presented	0.0003765

Analysis

In this task we implement a Benjamini-Hochberg method for all the original data. Benjamini-Hochberg Procedure helps us reduce the false discovery rate. We assume the below hypothesis :

H_{0j} = feature has no effect on the classification of an email

H_{1j} = feature has an effect on the classification of an email

After calculating the p values, we do not reject the values which are less than 5 %. That is because some true null hypotheses are thrown out just because of the randomness of results. For that reason we calculate L from the formula below :

$$L = max\{j : p_(j) < a * j/M\}$$

This L will give us the BH threshold which means that we will reject the *Hypothesis test* only for the features which their p values is less than the p value of the threshold.

For this specific dataset, we rejected 39 features which means that those 39 are really significant for the classification of an email. The threshold in this case is the variable "presented", so we reject all the features with p value less than the the p value of "presented" (0.0003765). We can mention that the first 10 features are almost the same with the nearest shrunken centroid method. The only differences in this top10 list is the variable "paper" which is replaced by the variable "due" in NSC method and that some of the features have different order.

Appendix

```
knitr::opts_chunk$set(echo = FALSE, fig.width = 4.5, fig.height = 3,
                      fig.align = "center",
                      warning = F, error = F, message = F)
#packages we need
library(readxl)
library(ggplot2)
library(gamreg)

#import the data
data = read_excel("../dataset/influenza.xlsx")

# data frame for the time series plot
df_plot = data.frame(x = data$Time, y1 = data$Mortality, y2 = data$Influenza)

#plot the time series : Time vs Mortality
ggplot(data = df_plot) +
     geom_line( aes(x = df_plot$x, y = df_plot$y1),color = "#00AFBB", size = 1) +
     geom_line( aes(x = df_plot$x, y = df_plot$y2),color = "red", size = 1) +
      labs(title = "Mortality(blue) and Influenza(red) vs Time", x = "Time", y ="Y") +
      theme_grey() +
      scale_x_continuous(breaks  = c(1995:2004))


library("mgcv")
library("akima")
library("plotly")

# fit the model

gam_model = gam(formula = Mortality ~ Year +  s(Week,k=length(unique(data$Week))) ,data = data, method =


#function for the plot Real and predicted values for mortality vs time

plot_mor = function(model,data, sp){

  #predictions for the model
  predictions = fitted(model)

  #data frame for the plot consist of observations and predictions
  df_mortality = data.frame(x = data$Time, observations = data$Mortality, predictions = predictions)

  ggplot(df_mortality) +
     geom_point(aes(x = df_mortality$x, y = df_mortality$observations)) +
     geom_line(aes(x = df_mortality$x, y = df_mortality$predictions), col = "red",size = 1) +
      labs(title = "Real and predicted values for Mortality vs Time", subtitle = paste('sp =',sp )   , x
    scale_x_continuous(breaks  = c(1995:2004)) +
      theme_dark()

}
```

```r
# Use the above function to plot the real and the predicted values
plot_mor(gam_model,data, 0.000113193 )

#summary of the model

summary(gam_model)

# Plot for the spline component
plot(gam_model)



library("gridExtra")

# use the function above to plot
plots = list()
sp = c(0,0.000113193,1,4,10,25,50,100,500,1000)
df = c()
deviances = c()

# a for loop to test different values of penalty factor
for (i in 1:length(sp)) {
  gam_model = gam(formula = Mortality ~ Year +  s(Week,k=length(unique(data$Week)),sp = sp[i]),data = da

  deviances[i] = gam_model$deviance
  plots[[i]] = plot_mor(gam_model,data, sp[i])
  df[i] = mgcv::pen.edf(gam_model)

}

print(data.frame(sp = sp, deviance = deviances, degrees_of_freedom = df))

#print the plots
grid.arrange(plots[[1]], plots[[2]], ncol=2)
grid.arrange(plots[[3]], plots[[4]], ncol=2)
grid.arrange(plots[[5]], plots[[6]], ncol=2)
grid.arrange(plots[[7]], plots[[8]], ncol=2)
grid.arrange(plots[[9]], plots[[10]], ncol=2)




#  fit the model from the task 1.2
gam_model = gam(formula = Mortality ~ Year +  s(Week,k=length(unique(data$Week))) ,data = data)

#create the data frames for the plots
df_influenza = data.frame(x =data$Time, value = data$Influenza)
df_residuals = data.frame(x =data$Time, value = gam_model$residuals)

df = rbind(df_influenza,df_residuals)

#Plot for residuals and influenza vs Time
ggplot()  +
```

```
    geom_line(aes(x = df_residuals$x, y = df_residuals$value),alpha = 0.75,color = "black") +
    geom_line(aes(x = df_influenza$x, y = df_influenza$value),alpha = 0.75, color = "yellow") +
    labs(title = "Residuals and Influenza values  vs Time", x = "Time", y = "Y") +
    scale_x_continuous(breaks  = c(1995:2004)) +
    theme_dark()



gam_model  = gam(formula = Mortality ~ s(Year,k=length(unique(data$Year))) +  s(Week,k=length(unique(da

#summary of the model
summary(gam_model)

#Plot  for observed and predicted Mortality VS Time

plot_mor(model = gam_model,data = data, sp = 0.000113193)



#import the data
library("pamr")
data = read.csv2("../dataset/data.csv")

# split the data to train (70%) and test(30%)
n = dim(data)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.7))
x_train = as.matrix(data[id,-which(colnames(data) == "Conference")])
y_train = as.matrix(data[id,"Conference"])

x_test = as.matrix(data[-id,-which(colnames(data) == "Conference")])
y_test = as.matrix(data[-id,"Conference"])


# export the dependent variable

new_data = list(x = t(x_train) , y = as.factor(y_train), geneid = as.character(1:nrow(t(x_train))), gene

set.seed(12345)
invisible(capture.output(
NSC_model <<- pamr.train(new_data, threshold = seq(0,4,by = 0.1))))

set.seed(12345)
invisible(capture.output(
CV_NSC <<- pamr.cv(NSC_model,new_data)))

#Centroid plot for optimal model

pamr.plotcen(NSC_model, new_data,threshold = 2.8)

# a list with all the scores

set.seed(12345)
```

```r
features = NA
invisible(capture.output(
  features <<- pamr.listgenes(NSC_model,new_data, threshold = 2.8)
))

# features which contributed the most
best_features = features[1:10,1]

# take the "real" words which contributed the most

words = rownames(t(x_train))[as.numeric(best_features)]
data.frame(words)

#calculate the error for task
# create a list for the test data
test_data = list(x = t(x_test) , y = as.factor(y_test), geneid = as.character(1:nrow(t(x_test))), genena

#predictions for test data
pred_nsc = pamr.predict(NSC_model,newx = test_data$x,threshold = 2.8, type = "class")

#mislassification rate for test data
misrate_nsc = length(which(pred_nsc != y_test))/ nrow(y_test)

df = data.frame(misclassification_rate = misrate_nsc, features_selected = length(features[,1]))
#print the misclasification rate
knitr::kable(x = df, caption = "misclassification rate for test data and number of variables selected")


library("glmnet")
library("kernlab")

#elastic net
#fit the elastic net model using cross validation
set.seed(12345)
elastic = glmnet(x = x_train, y=  y_train, family = "binomial", alpha = 0.5)

set.seed(12345)
elastic_cv = cv.glmnet(x = x_train, y=  y_train, family = "binomial", alpha = 0.5, lambda = elastic$lam

# predictions for the elastic net
predictions_net = as.numeric(predict(object = elastic_cv,newx = x_test,s = elastic_cv$lambda.min,type =

#misclassification rate for elastic net
misrate_net = length(which(predictions_net != y_test))/nrow(y_test)

# number of variables selected from CV
optimal_features = elastic_cv$nzero[which(elastic_cv$lambda==elastic_cv$lambda.min)]

# SVM
# predictions for SVM
# fit the model using svm

set.seed(12345)
```

```r
SVM = ksvm(x = x_train,y_train, kernel = "vanilladot", type = "C-svc" )

#make predictions for test data
set.seed(12345)
pred_svm = predict(object = SVM, newdata = x_test, type = "response")

# misclassification rate for svm
misrate_svm = length(which(pred_svm != y_test))/nrow(y_test)

# number of variables selected

svm_features = SVM@nSV

# data frame for all misclassification rates
df = data.frame(c(misrate_nsc,misrate_net,misrate_svm), c(length(features[,1]), as.vector(optimal_featu
colnames(df) = c("Misclassification_rates","Features_selected")
rownames(df) = c("NSC","Elastic_net","SVM")

# print the misclassification rates for all models
knitr::kable(x = df, caption = "misclassification rates")


#function calculate BH treshold

a = 0.05
#df = df_p_sorted

L = function(a,df){
  n = nrow(df)
  L_vec = c()

  # loop for calculate all tresholds
  for (i in 1:n) {
    if(df$p_values[i] < a*(i/n)){

      # all the indexes
      L_vec = c(L_vec, i)
    }

  }
  return(L_vec)
}


library("sgof")

# split the data to predictors and response
x = data[,-which(colnames(data) == "Conference")]
y = data[,"Conference"]

features = c()
p_values = c()
```

```r
# for loop to calcultate all p - values for each colum
for (i in 1:ncol(x)) {

  # create the formula for the i-th colum
  formula = as.formula(paste(colnames(x)[i],"Conference",sep = "~"))

  #t - test for the i-th comuln
  temp = t.test(formula,data)

  #p value for each colum
  p_values = c(p_values,temp$p.value)
  features = c(features, colnames(x)[i])

}

# another aproach
#calculate the adjusted p values
#final_p = p.adjust(p_values, method = "fdr")
# Use BH hypothesis for false discovery rate
#n_features = length(which(final_p<= 0.05))
#features_rejected = features[which(final_p<= 0.05)]

# sort p_values and features
df_p = data.frame(features = features, p_values = p_values)

df_p_sorted = df_p[order(df_p$p_values),]

# calculate the L -index for the treshold with function L for a = 0.05

a = 0.05
L = max(L(a,df_p_sorted))

#features rejected

rejected = data.frame(features = df_p_sorted$features[1:L], p_values = df_p_sorted$p_values[1:L])
knitr::kable(x =rejected, caption = "Features rejected from Benjamini-Hochberg method")
```