Lab2_Block2

Stefano Toffol (steto820), Andreas Stasinakis (andst745), Mim Kemal Tekin (mimte666)

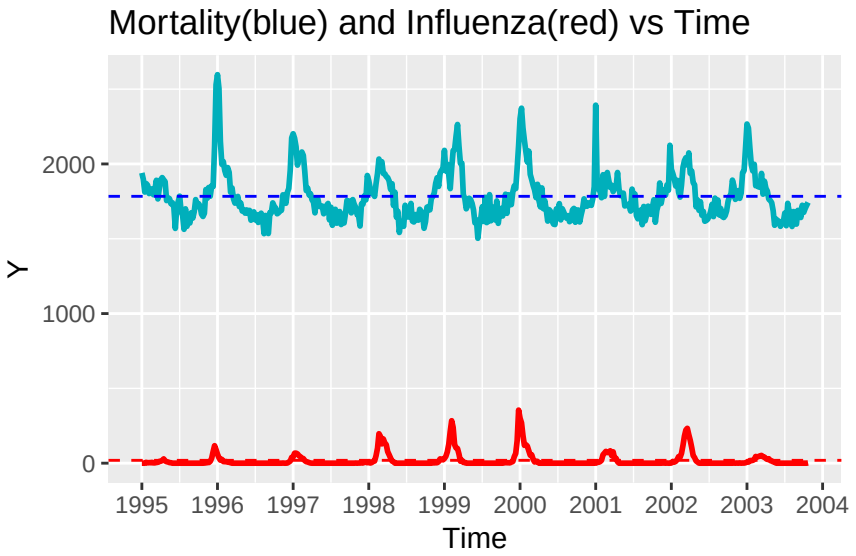
December 16, 2018

Contributions:

- Assignment 1 - Andreas Stasinakis(code), Stefano Toffol(Analysis);
- Assignment 2 - Mim Kemal Tekin;

Assignment 1

Task 1.1



The time series shows how the *Mortality* (blue) and the cases of *Influenza* (red) in the target population varied during the years. It is evident how the two variables appear correlated, following both a clear seasonal trend.
Mortality is subject to great fluctuations during the year, but it shows almost no change on average along the hole period considered, remaining practically stable at around 1800 (dashed line). The seasonal component brings each year the mortality at his minimum during summer, decreasing until approximately 1500, while almost always in January it reaches the maximum, going each time above 2000. The peaks in mortality happen fast, with a sharp exponential increase starting from generally two months before, in November, followed by a much slower decrease lasting the rest of the winter and the whole spring.
The cases of *Influenza* remain stationary around zero during most of the year but, as also the *Mortality* does, they suddenly increase during winter. There seems to be no pattern in the amount of cases from year to year and the relative increase of *Influence* does not strictly follow the one of *Mortality*. In fact the all-time high of the latter happens in 1996, when the former variable showed a regular increase. On the other hand the maximum cases of *Influenza* recorded happened in 2000, when the *Mortality* rate was only slightly above it's average (considering the month of the year).

Task 1.2

A *Generalized Additive Model (GAM)* has been run on the dataset, regressing *Mortality* (Y variable) over *Year* (X_1 variable) and a spline function of *Week* (X_2 variable). The parameters of the model were estimated through CV. The code can be found below:

```
fit_gam <- gam(Mortality ~ Year + s(Week, k = length(unique(data$Week))),
               familiy = gaussian(), data = data, method = "GCV.Cp")
s <- interp(data$Year, data$Week, fitted(fit_gam))
```

The model fitted relies on the hypothesis $Y_i \sim N(\mu, \sigma^2)$; Y *iid*.
The theoretical model is the following:

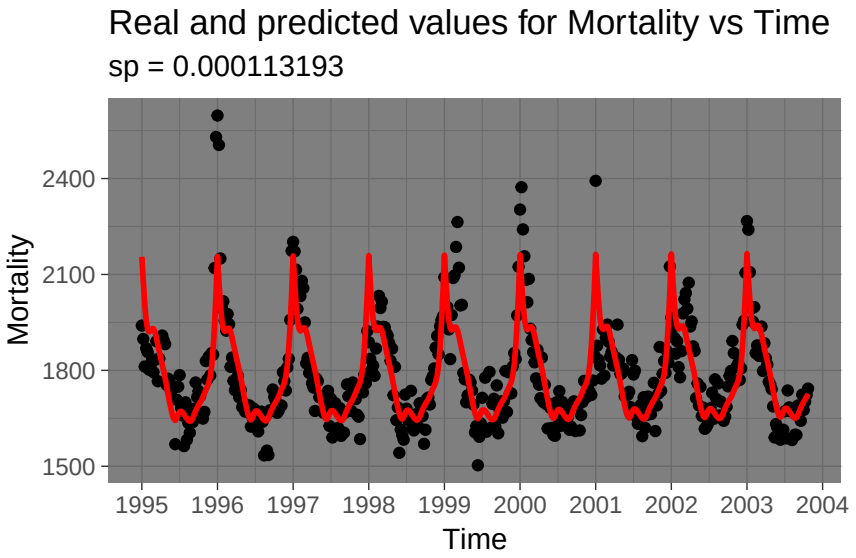
$$Y \sim N(\mu, \cdots) \qquad \text{where: } \begin{cases} g(\mu) = \alpha + \beta_1 \cdot X_{i,1} + s_2(X_{i,2}) & \\ X_{i,1} & \text{linear term} \\ s_2(X_{i,2}) & \text{smoothed term (spline)} \\ g(\cdot) & \text{link function (identity)} \end{cases}$$

The estimated model is instead the following (where significant coefficients are marked with an asterisk):

$$\hat{Y}_i = -680.60 + 1.23 \cdot X_{i,1} + \hat{s}_2^*(X_{i,2})$$

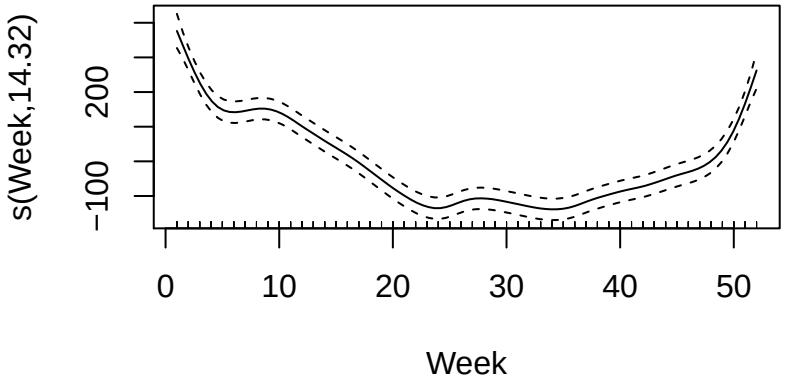
Therefore the only feature having a significant impact on the response variable is the spline function of the *Week*.

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|)
```

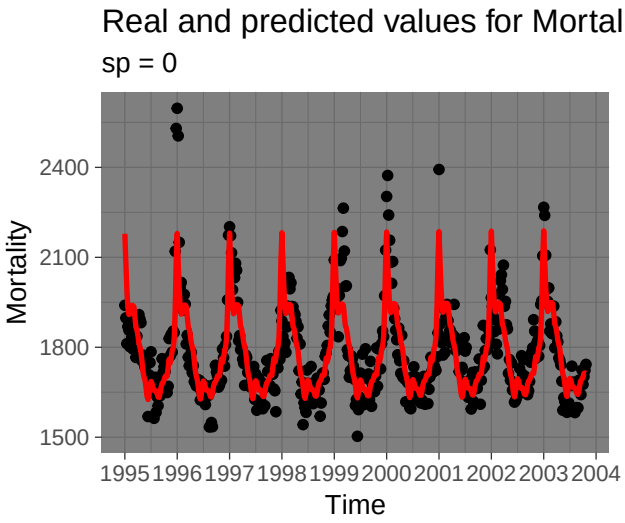
```
## (Intercept) -680.598   3367.760  -0.202    0.840
## Year           1.233      1.685   0.732    0.465
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Week) 14.32  17.87 53.86  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 52/53
## R-sq.(adj) =  0.677   Deviance explained = 68.8%
## GCV = 8708.6  Scale est. = 8398.9    n = 459
```

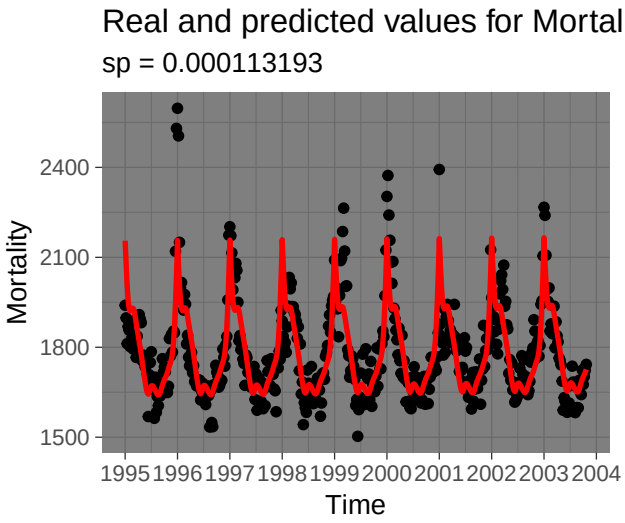


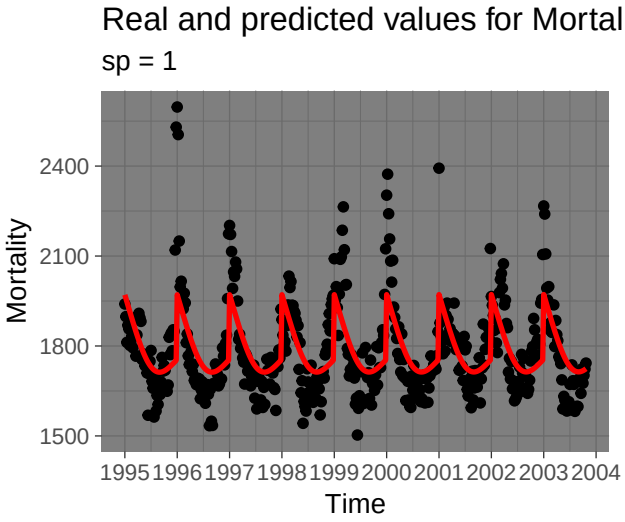
The results of the previously fitted *GAM* model are plotted in the first plot above (red line), together with the observed values (black points). The graph reveals remarkable fitting of the model to the data, being able to omit the random noise from its estimations and closely following the trend of the response. However, the different heights of the winter peaks are not cached and the corresponding predictions appear the same over the years.
As previously stated, the only significant parameter of the model was the spline function of the *Week*. This finding implies the absence of a significant linear trend of the response over the years. In fact, despite the variance in between different periods of time, the average of *Mortality* remains essentially constant.
Regarding the peaks of the Y around the months of January, they appear to follow a cyclic trend, where years of high *Mortality* rate are followed by years of a relatively low number of deaths. Moreover in the time period considered the *Mortality* trend may be summarized by a quadratic trend, as also shown by the spline component of the *GAM* model.

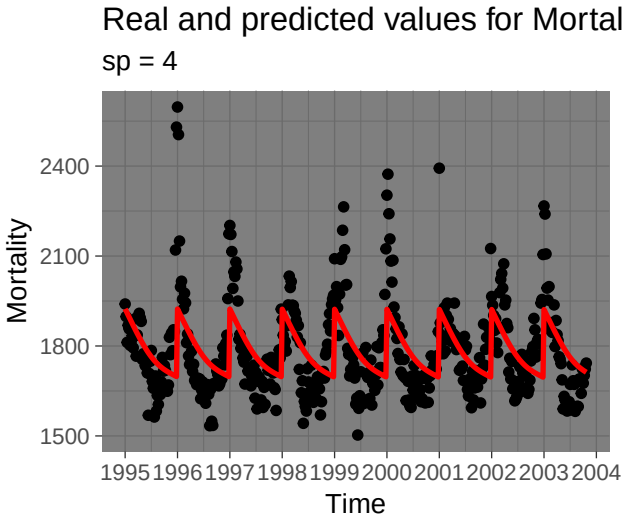
Task 1.4

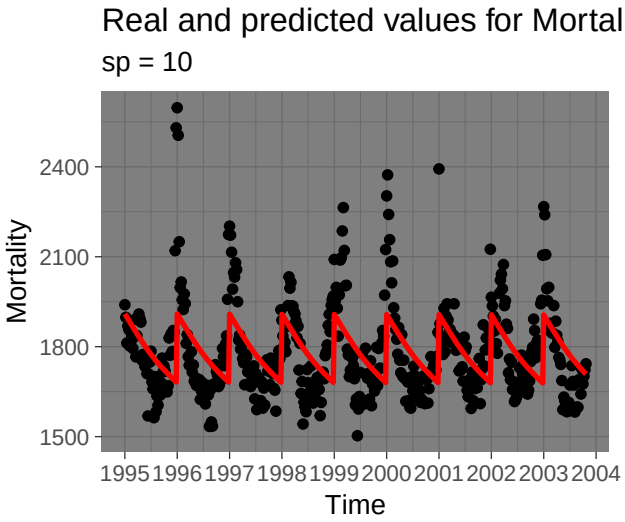
```
##             sp deviance degrees_of_freedom
## 1  0.00000e+00  3645526          27.000000
## 2  1.13193e-04  3718012          14.321891
## 3  1.00000e+00  6593860           1.491752
## 4  4.00000e+00  8553032           1.161915
## 5  1.00000e+01  9274216           1.069458
## 6  2.50000e+01  9618101           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
```

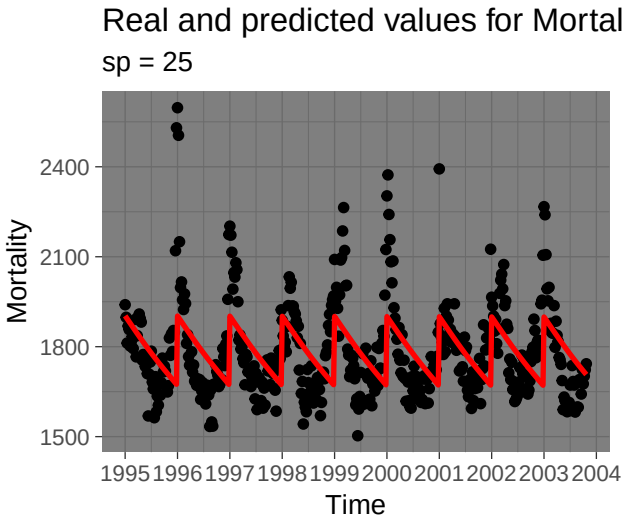

Real and predicted values for Mortal
sp = 0
Mortality
2400
2100
1800
1500
1995 1996 1997 1998 1999 2000 2001 2002 2003 2004
Time

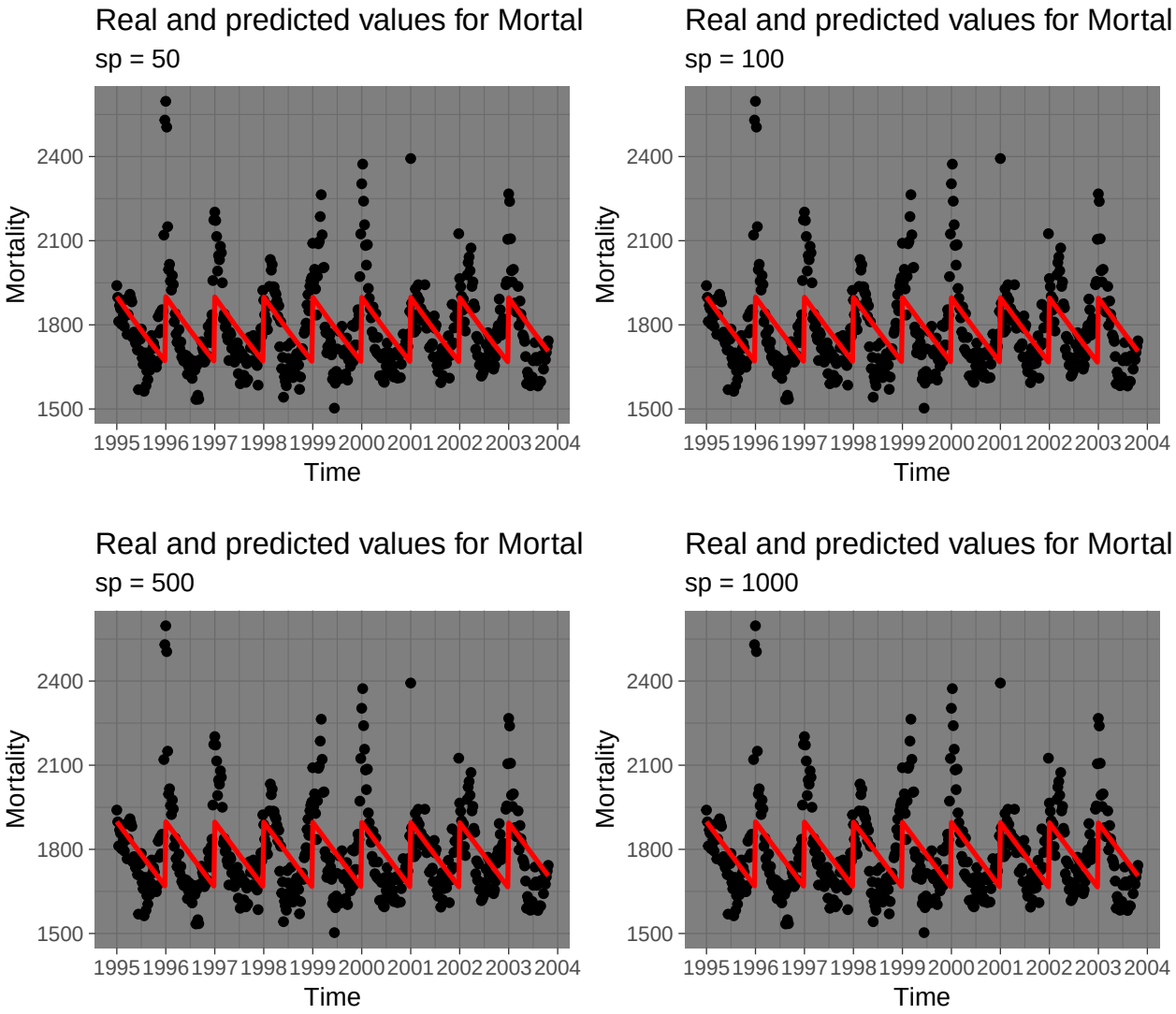

Real and predicted values for Mortal
sp = 0.000113193
Mortality
2400
2100
1800
1500
1995 1996 1997 1998 1999 2000 2001 2002 2003 2004
Time


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


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

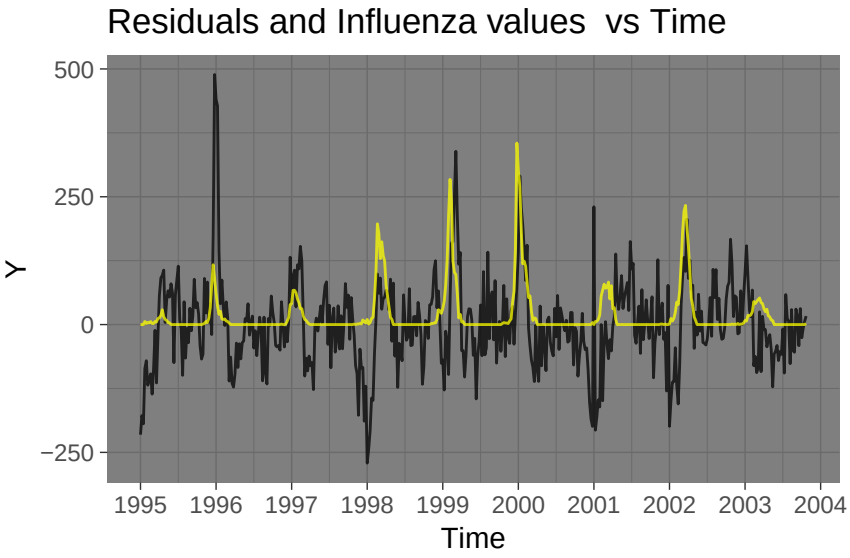

Real and predicted values for Mortal
sp = 10
Mortality
2400
2100
1800
1500
1995 1996 1997 1998 1999 2000 2001 2002 2003 2004
Time


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



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

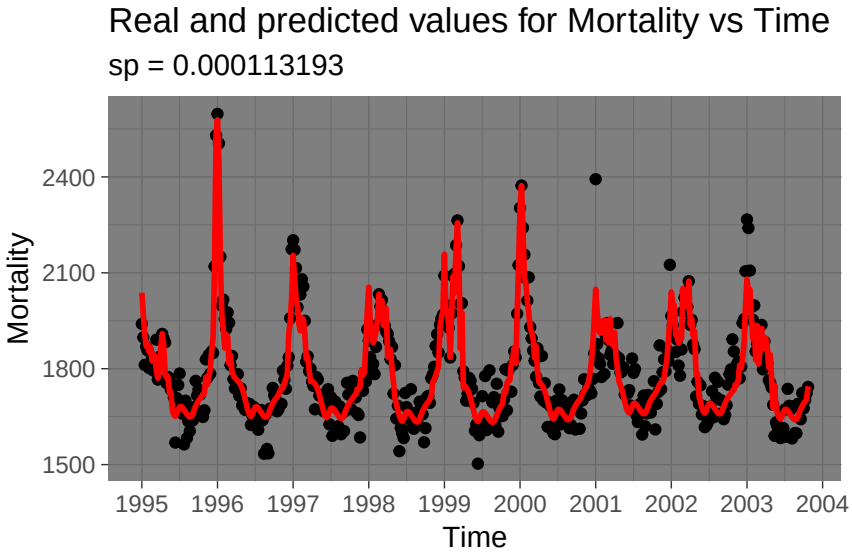


The plot of the residuals of the *GAM* model and of the variable *Influenza* is displayed above.
The residuals (black line) show high variability and are not uncorrelated nor normal as they should be, having a cyclical trend similar to the seasonal one of the *Mortality*. In particular the fluctuations increase substantially during winter, resulting in high/low peaks, whether the model had underestimated or overestimated the increase of deaths in that season.
The variable *Influence*, as previously observed, has peaks during the beginning of each year, that therefore correspond to the peaks of the residuals. However their intensity differs substantially and the relationship between the two lines is not that evident.

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) 1783.765      3.198   557.8   <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)      4.587  5.592  1.500   0.178
## s(Week)     14.431 17.990 18.763  <2e-16 ***
## s(Influenza) 70.094 72.998  5.622  <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 = 5840.5  Scale est. = 4693.7    n = 459
```



A model including spline functions of the variables *Year*, *Week* and *Influenza* as regressors of *Mortality* has been fitted to the dataset. The results are plotted in the plot above.
The fitted values of the model (red line) now catch also the different heights of the seasonal peaks of the response variable. The predictions however appear to be subjected to the random noise of the phenomena. The addiction of *Influenza* to the model is likely to be the reason why this model is capable to correctly guess the outbreaks of the Y variable, however this flexibility comes to the price of a higher variance. In fact the two lines are often perfectly overlapped and the results of the model fluctuate excessively.
It is hard to say which one between the current model and the one trained in step 2 of our analysis is the best: the former is capable to correctly guess the trend of the response in almost all the cases but is subjected to several fluctuations; on the other hand the latter does not catch the different heights of the peaks of the Y variable, but correctly approximate its main trend ignoring the random noise.
Finally, looking at the summary , both the spline functions of *Week* and of *Influenza* are strongly significant, while the one of *Year* does not appear to have a significant relation with the response. The degrees of freedom of the latter are approximately 4.5, meaning that it interpolates the data in its dimensions almost perfectly. For this reason the variable should be excluded from the model (since the information it brings is probably summarized by *Week*), resulting in a probably smoother estimate of *Mortality*.

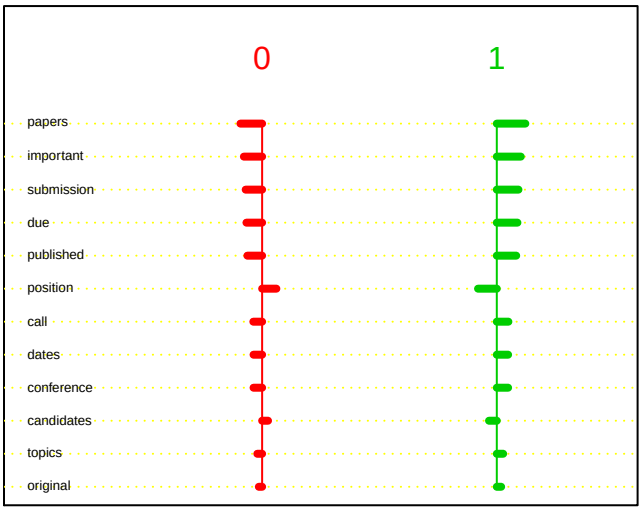
Assignment 2: High-Dimensional Methods

Task 2.1

In this task we use nearest shrunken centroid classification in order to reduce effect of noisy variables to our predictions. Additionally, we perform cross-validation to find the optimal threshold for the algorithm.
After cross validation we pick the threshold which satisfies minimum error and minimum complexity.

```
## [1] "Optimal Threshold: 2.8"
```

We can see the most contributing features to the model which has optimal threshold in following centroid plot:



Furthermore, name of the 10 most contributing features are listed below:

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

Table 1: Confusion Matrix of NSC with Test Data

	0	1	Frequencies
0	10	0	10
1	2	8	10
Frequencies	12	8	20

```
## [1] "Misclassification Rate:  0.1"
```

```
## [1] "Selected Feature Count:  12"
```

We can see that we have some close terms to announces of conferences in our top 10 important features, but also we have some different terms like “submission” and “position”. Additionally, our misclassification rate is 0.1 and selected feature count is 12. The centroid plot shows us the features which have at least one nonzero distance to the classes (“announces of conferences”, “everything else”). For this model if we examine being announces of conferences, “position” and “candidates” words are in negative relation which means if a document contains these words it is more likely to be in “everything else” class. Furthermore the word which has more impact in classifications is “papers”. Right behind it, “important”, “submission”, “due” and “published” words are following it.

Task 2.2

a. Elastic Net

```
## [1] "Misclassification Rate:  0.1"
## [1] "Selected Feature Count:  38"
```

b. Support Vector Machine

```
##  Setting default kernel parameters
## [1] "Misclassification Rate:  0.05"
## [1] "Selected Feature Count:  43"
```

Table 2: Comparison Between Methods and Real

	Misclassification	Feature_Count	Predictions
Real Values	-	4703	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0
NSC Classification	0.1	12	0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0
Elastic Net	0.1	38	0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0
SVM	0.05	43	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0

If we compare results we can see NSC is the one which reduces most features than the other ones. But in the other hand SVM has the lowest misclassification rate. Elastic Net has same misclassification rate with NSC, but higher feature count. Furthermore, Elastic Net and NSC misclassified exactly same observation. If we want to pick one of them, it should be NSC in this case and only testing with this test-data, because it is simpler than NSC. After this, if we have to pick one of SVM and NSC, it depends what we are looking for as model. If we want to create simpler model we can choose NSC because it has significantly lower feature count than SVM and real data, but if we want to more accurate classifications it is better to pick SVM, because it has the lowest misclassification rate. Moreover, it classified correctly the observation which NSC misclassified in the test data.

Task 2.3

In this task, we implement Benjamini-Hochberh method. Benjamini-Hochberg method helps to reduce false positive and we have hypothesis to test as following:

H_{0j} = feature does NOT contributes to being announcement of conference

H_{0j} = feature DOES contributes to being announcement of conference

In hypothesis testing, as a procedure we calculate p-values between each variables and the target variable. We cannot take 5% of this result directly, because of randomness of results. There can be same Type-I errors and to reduce this errors we apply BH method which is defined as following:

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

L will be the threshold value which we reject values less than L and we do not reject greater than L. For this data we have 39 rejected variables and these variables contribute to classification of text (email). We can see the words are rejected below. If we compare this result and the result from task 2.1, the only difference in top 10 words is "paper" instead of "due" which makes more sense, and some order differences.

```
## [1] "Threshold Value: 0.000376514731179718"
```

```
## [1] "Count of Rejected Features: 39"
```

Table 3: Rejected Features

name	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
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

Appendix

```
knitr::opts_chunk$set(echo = FALSE, fig.width = 4.5, fig.height = 3,
                      fig.align = "center",
                      warning = F, error = F, message = F,
                      fig.pos = 'h')
options(kableExtra.latex.load_packages = FALSE)



library(readxl)
library(kableExtra)
library(ggplot2)
library(viridis)
library(dplyr)
library(mgcv)
library(akima)
library(pamr)
library(glmnet)
library(kernlab)
library(grid)
library(gridExtra)

conf_matrix = function(real_data, predicted_data, levels){
  # make the values factor
  real_data = factor(real_data, levels=levels)
  predicted_data = factor(predicted_data, levels = levels(real_data))
  # we can have "not predicted" values in predicted data
  # make equal the levels of predicted values with real data
  levels(predicted_data) = levels(real_data)

  ct = table(real_data, predicted_data)
  df = data.frame(c(as.vector(ct[,1]), sum(ct[,1])),
                  c(as.vector(ct[,2]), sum(ct[,2])),
                  c(sum(ct[1,]), sum(ct[2,]), sum(ct)))
  rownames(df) = c(levels, "Frequencies")
  colnames(df) = c(levels, "Frequencies")
  return(df)
}

calculate_rate = function(conf_matrix){
  conf_matrix = as.matrix(conf_matrix)
  return(1 - sum(diag(conf_matrix[1:2,1:2]))/sum(conf_matrix[1:2,1:2]))
}

kable_cm = function(cm, capture){
  return(kableExtra::kable(cm, "latex", booktabs = T, align = "c",
            caption = capture) %>%
            row_spec(2, hline_after = T) %>%
            # column_spec(c(1,3), border_right = T) %>%
```

```
            kable_styling(latex_options = "hold_position"))
}

#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_hline(yintercept = mean(data$Mortality),
             size = 0.5, col = "blue", lty = 2) +
     geom_line( aes(x = df_plot$x, y = df_plot$y2),color = "red", size = 1) +
  geom_hline(yintercept = mean(data$Influenza),
             size = 0.5, col = "red", lty = 2) +
      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 =



# ------------------------------------------------------------------------------
# A1 - Q2
# ------------------------------------------------------------------------------


fit_gam <- gam(Mortality ~ Year + s(Week, k = length(unique(data$Week))),
               familiy = gaussian(), data = data, method = "GCV.Cp")
s <- interp(data$Year, data$Week, fitted(fit_gam))


alpha <- format(round(fit_gam$coefficients[1], 2), nsmall = 2)
beta <- round(fit_gam$coefficients[2], 2)

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

plot_mor = function(model,data, sp){
```

```r
  #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()

}


# 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)
```

```r
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)



################################### TASK 2.1 ###################################

# import data
df_mail = read.csv2("../dataset/data.csv", fileEncoding="ISO-8859-1")
rownames(df_mail)=1:nrow(df_mail)
# split data 70/30
n = dim(df_mail)[1]
p = dim(df_mail)[2]
set.seed(12345)
id = sample(1:n, size=floor(n*0.7))
test = df_mail[-id,]
train = df_mail[id,]

target_index = which(colnames(df_mail)=="Conference")
x = t(train[, -target_index])
y = train[, target_index]
```

```
x_test = t(test[, -target_index])
y_test = test[,target_index]

mydata = list(x = x, y = as.factor(y), geneid = as.character(1:nrow(x)),
              genenames=rownames(x))

# fit model
set.seed(12345)
model = NA
invisible(capture.output(
  model <<- pamr.train(mydata,threshold=seq(0,4, 0.1))
))

# cross-validation to find optimal threshold
set.seed(12345)
cvmodel = NA
invisible(capture.output(
  cvmodel <<- pamr.cv(model,mydata)
))
# print(cvmodel)
# This chunk did not run when knit because the plot could not render...
# plot exported as png, and added after this chunk via latex
# get plot of cv results
# plot_cv = pamr.plotcv(cvmodel)
# plot_cv

# find min error and min size index
min_err_size_index = max(which(cvmodel$error==min(cvmodel$error)))
# find optimal threshold which satisfies
optimal_threshold = cvmodel$threshold[min_err_size_index]

# store feature count
fc_nsc = cvmodel$size[min_err_size_index]

print(paste("Optimal Threshold:", optimal_threshold))
# plot the most contributing features
plot_best_features = pamr.plotcen(model, mydata, threshold=optimal_threshold)
# get best feature list
best_f_scores = NA
invisible(capture.output(
  best_f_scores <<- pamr.listgenes(model, mydata, threshold=optimal_threshold)
))

# first 10 most contributing feature names
features_10 = colnames(df_mail)[as.numeric(best_f_scores[1:10,"id"])]
print(as.matrix(features_10))
# get posterior
posterior = pamr.predict(model, x_test, threshold = optimal_threshold,
                         type="posterior")
# get predictions from posterior
pred_nsc = as.vector(ifelse(posterior[,1]>posterior[,2], 0, 1))

# get confusion matrix with test data
```

```r
cm = conf_matrix(real_data = y_test,
                 predicted_data = pred_nsc, levels=c(0,1))

# calculate misclassification rate
mis_rate_nsc = calculate_rate(cm)

kable_cm(cm, "Confusion Matrix of NSC with Test Data")

print(paste("Misclassification Rate: ", mis_rate_nsc))
print(paste("Selected Feature Count: ", fc_nsc))

################################### TASK 2.2 ###################################

# a. Elastic Net implementation

set.seed(12345)

library(glmnet)
# fit and do cross-validation to find optimal lambda
cv_elastic = cv.glmnet(t(x), y, alpha=0.5, family="binomial",
                       lambda=seq(0,1,0.001))
# get the optimal values
min_lambda = cv_elastic$lambda.min
fc_el = as.numeric(cv_elastic$nzero[which(cv_elastic$lambda==min_lambda)])

# plot lambda
# plot(cv_elastic)

# predict
pred_el = as.numeric(predict(cv_elastic, t(x_test), s=min_lambda, type="class"))
# confusion matrix
cm = conf_matrix(y_test, pred_el, c(0,1))
# kable_cm(cm, "Confusion Matrix of Elastic Net with Test Data")
# misclassification rate
mis_rate_el = calculate_rate(cm)

print(paste("Misclassification Rate: ", mis_rate_el))
print(paste("Selected Feature Count: ", fc_el))

# b. Support Vector Machine

set.seed(12345)

library(kernlab)
# fit model
svm = ksvm(t(x), y, type="C-svc", kernel = "vanilladot")

fc_svm = svm@nSV

# predictions
pred_svm = predict(svm, t(x_test), type = "response")
# confusion matrix
cm = conf_matrix(y_test, pred_svm, c(0,1))
```

```r
# kable_cm(cm, "Confusion Matrix of svm with Test Data")

# calculate misclassification rate
mis_rate_svm = calculate_rate(cm)
print(paste("Misclassification Rate: ", mis_rate_svm))
print(paste("Selected Feature Count: ", fc_svm))

# comparative table
df = data.frame(Misclassification = c("-",
                                      mis_rate_nsc,
                                      mis_rate_el,
                                      mis_rate_svm),
                Feature_Count = c(p, fc_nsc, fc_el, fc_svm),
                Predictions = c(paste0(y_test, collapse = " "),
                                paste0(pred_nsc, collapse = " "),
                                paste0(pred_el, collapse = " "),
                                paste0(pred_svm, collapse = " ")),
                row.names = c("Real Values",
                              "NSC Classification",
                              "Elastic Net",
                              "SVM"))

kableExtra::kable(df, "latex", booktabs = T, align = "c",
            caption = "Comparison Between Methods and Real") %>%
            # row_spec(2, hline_after = T) %>%
            # column_spec(c(1,3), border_right = T) %>%
            kable_styling(latex_options = "hold_position")


################################### TASK 2.3 ###################################


target_index = which(colnames(df_mail)=="Conference")
c_names = colnames(df_mail[, -target_index])
p_values = c()
for(f_name in c_names){
  f = formula(paste(f_name,"~","Conference"))
  t = t.test(formula=f, data=df_mail, alternative="two.sided")
  p_values = c(p_values, t$p.value)
}

n = length(p_values)
o = order(p_values)
o_pvalues = p_values[o]

# finding max rejected feature by using fdr method
max_i = 0
alpha = 0.05
for(i in 1:n){
  if(o_pvalues[i] < alpha*i/n)
    max_i = i
  else
    break
```

```
}

rejected_indexes = o[1:max_i]

rejected = data.frame(name = colnames(df_mail)[rejected_indexes],
                      p_values = p_values[rejected_indexes])

print(paste("Threshold Value:", o_pvalues[max_i]))
print(paste("Count of Rejected Features:", max_i))
kableExtra::kable(rejected, "latex", booktabs = T, align = "c",
            caption = "Rejected Features") %>%
            kable_styling(latex_options = "hold_position")
```