Block 2 Lab 2 Report

Mim Kemal Tekin

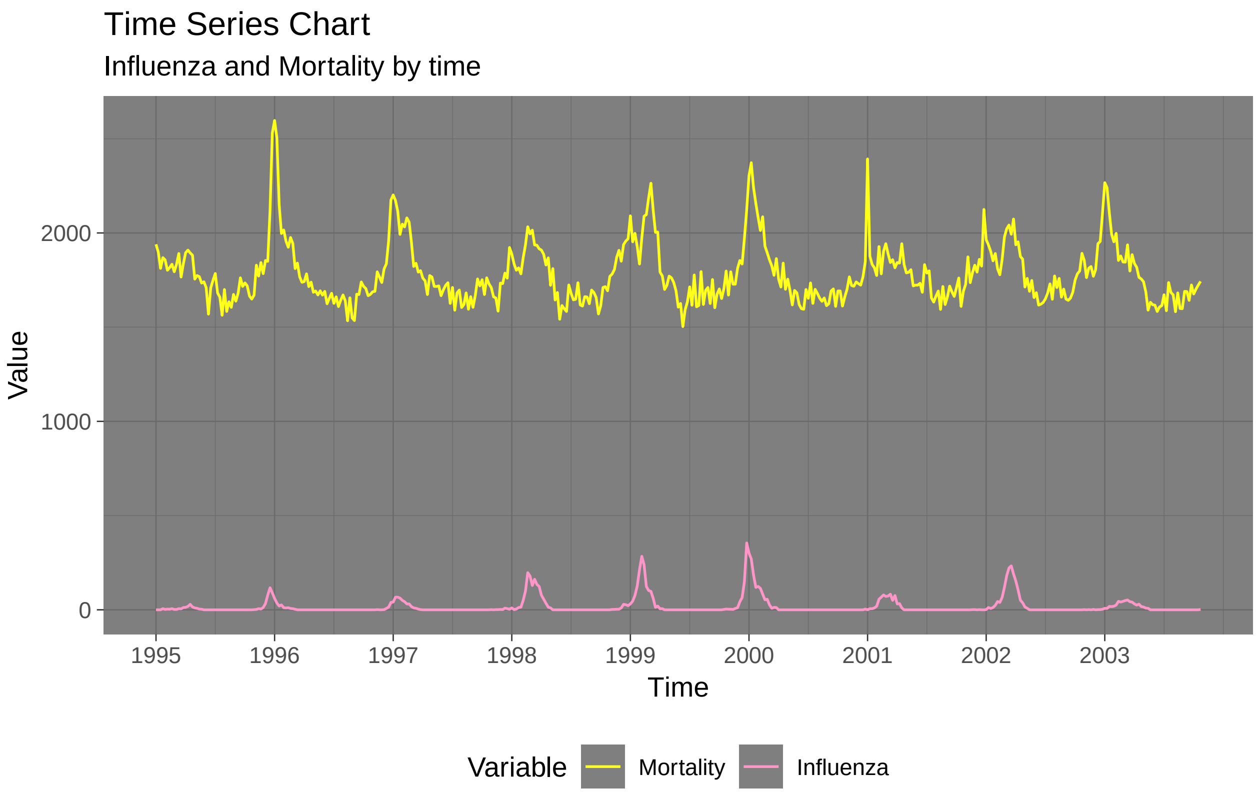
12/11/2018

Assignment 1: Using GAM and GLM to examine the 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).

Task 1.1

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.



We can see that there is a peak in the beginning of every year in both variable, so we can consider peak times are similar to each other. The influenza looks like appear in winter and the mortality increases with the influenza. First peak is in 1996 in influenza and even though it is a small peak the mortality rate peak too high. Following years although increasing value of influenza, mortality peaks are quite same. After 2001, looks like influenza is under control and peaks are not high as in 1998, 1999, 2000.

Task 1.2

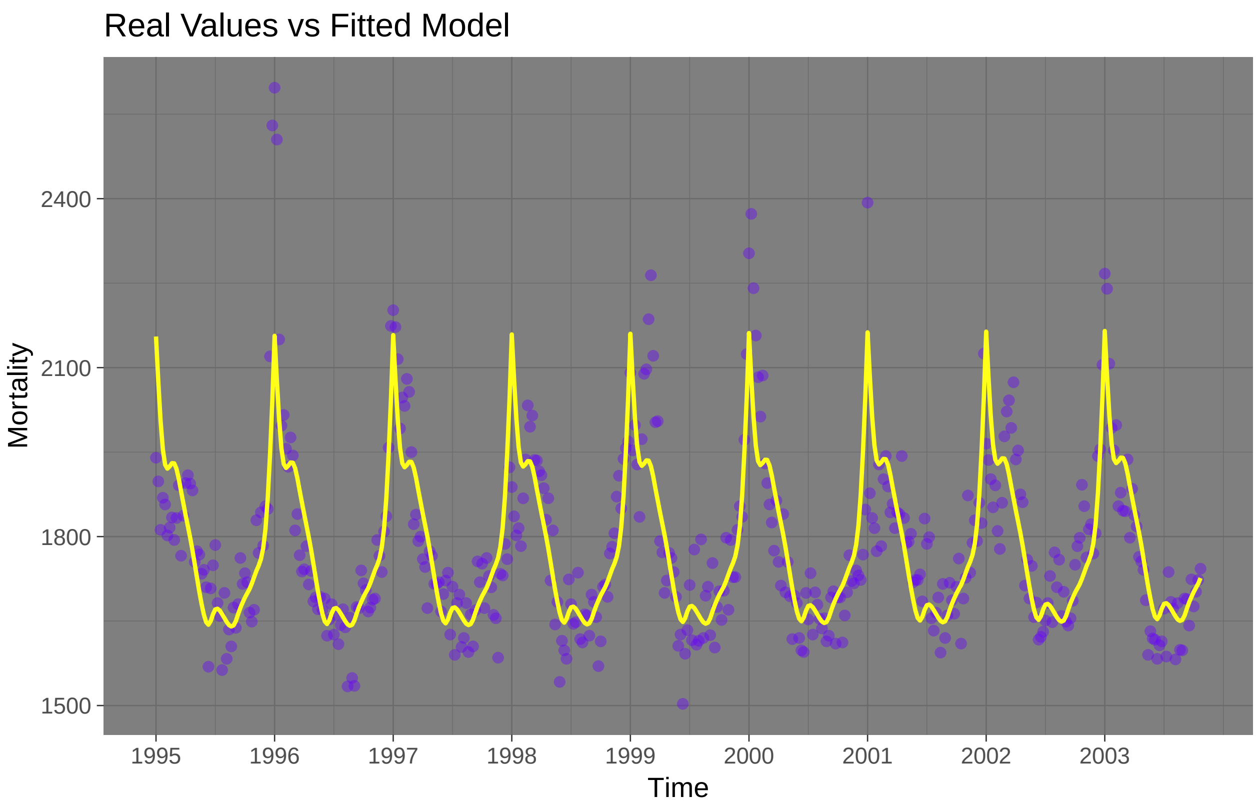
Use gam() function from mgcv package to fit a GAM model in which Mortality is normally distributed and modelled as a linear function of Year and spline function of Week, and make sure that the model parameters are selected by the generalized cross-validation. Report the underlying probabilistic model.

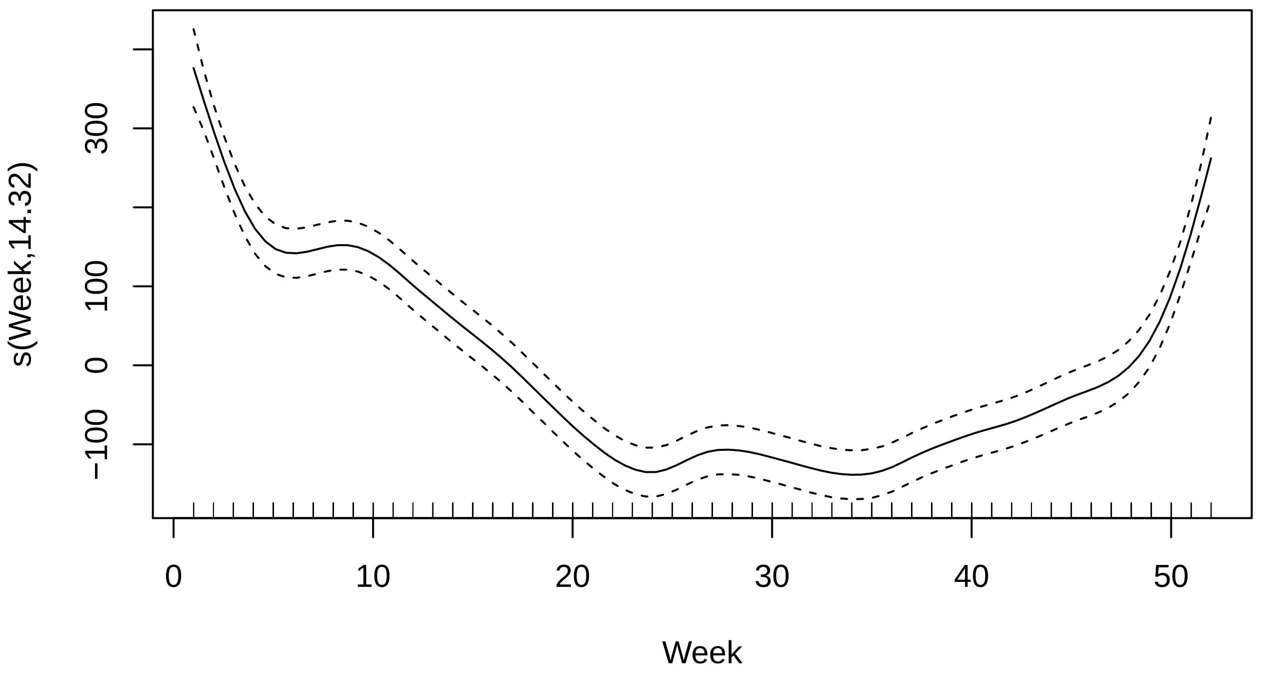
In this task, we fit a GAM model which contain 2 function. One is linear function which takes Year as parameter, and the other one is a spine function which takes Week as parameter. The probabilistic model is defined as following:

$$g(\mu) = \alpha + \beta_1 X_1 + s_1(X_2)$$

Task 1.3

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.

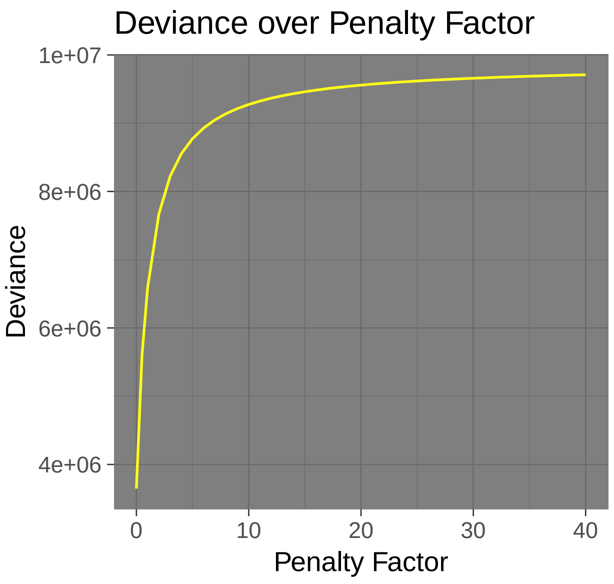


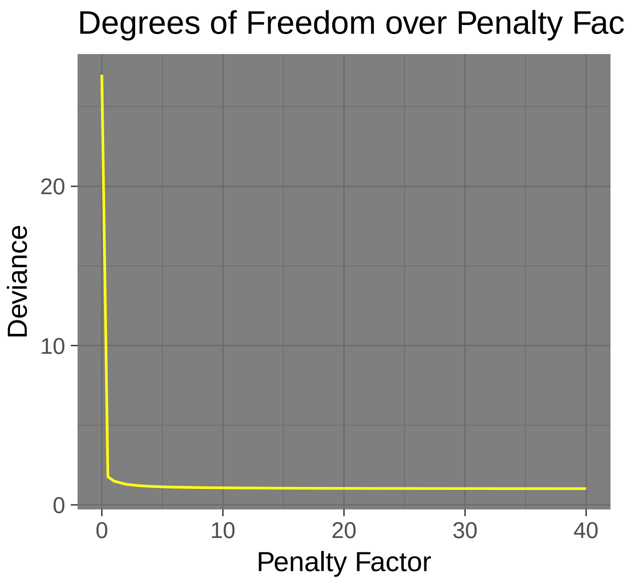


In this task, we can see fitted values as a line and real values as points in the plot above. The model looks like captured relations of trends of points, except outlier values which are peaks in this case. We can say that model is fitted good, but if we want to do more accurate predictions about our peaks we can say that this model is not fitted enough. But it gives enough information about trends. The significant term is Week in this model. If we summary our model in R, we can see p-value of spline function of week is between 0 and 0.001 and it has three star significancy.
Additionally, we can see contribution of spline function with Week feature to the model at 2nd plot above. It decreases until 24th week of year and after that it starts to increase.

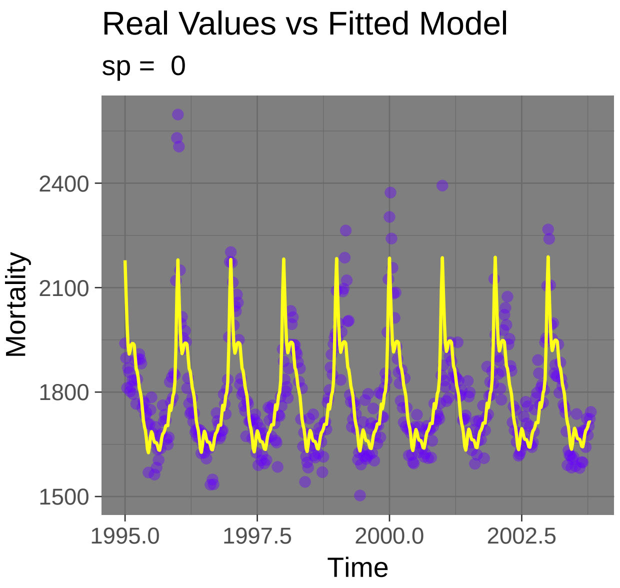
Task 1.4

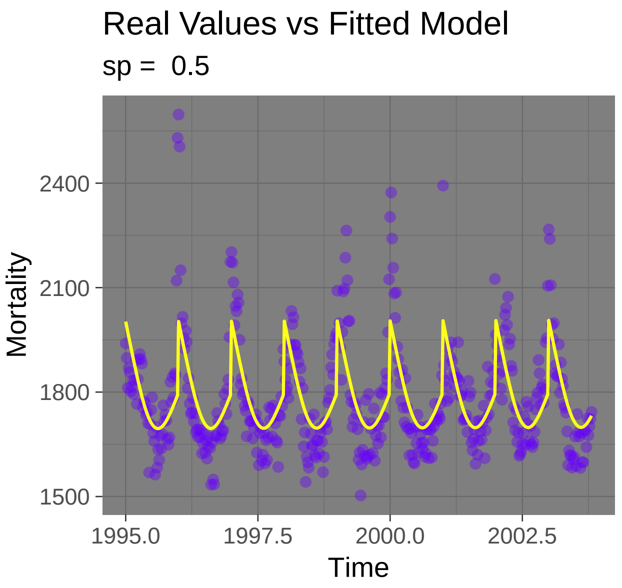
Examine how the penalty factor of the spline function in the GAM model from step 2 influences the estimated deviance of the model. Make plots of the predicted and observed mortality against time for cases of very high and very low penalty factors. What is the relation of the penalty factor to the degrees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





In this task we calculate deviances over the changes on penalty factor and we tried penalty factor which are between -1 and 100. The plot above shows while penalty factor increases the Deviance increases also. Until around 10 it increases exponentially and after that the increasement is getting slow. Additionally, we can see that degrees of freedom decreases over penalty factor. First values of penalty factor, it decreases exponentially.

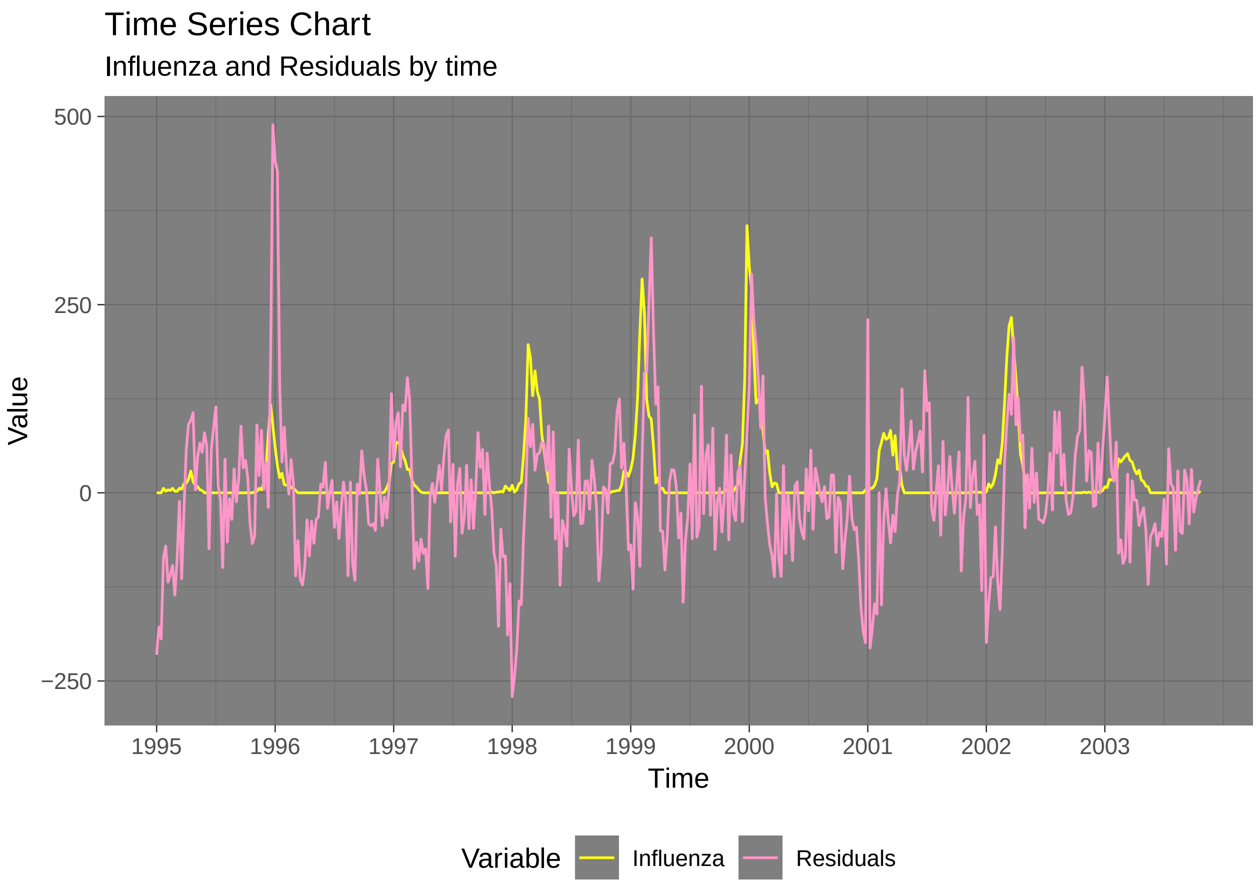




The plot above shows some models with different penalty factor values. We can see that when sp value increases peaks are shorter and model lost its shape, getting flatter and sharper.

Task 1.5

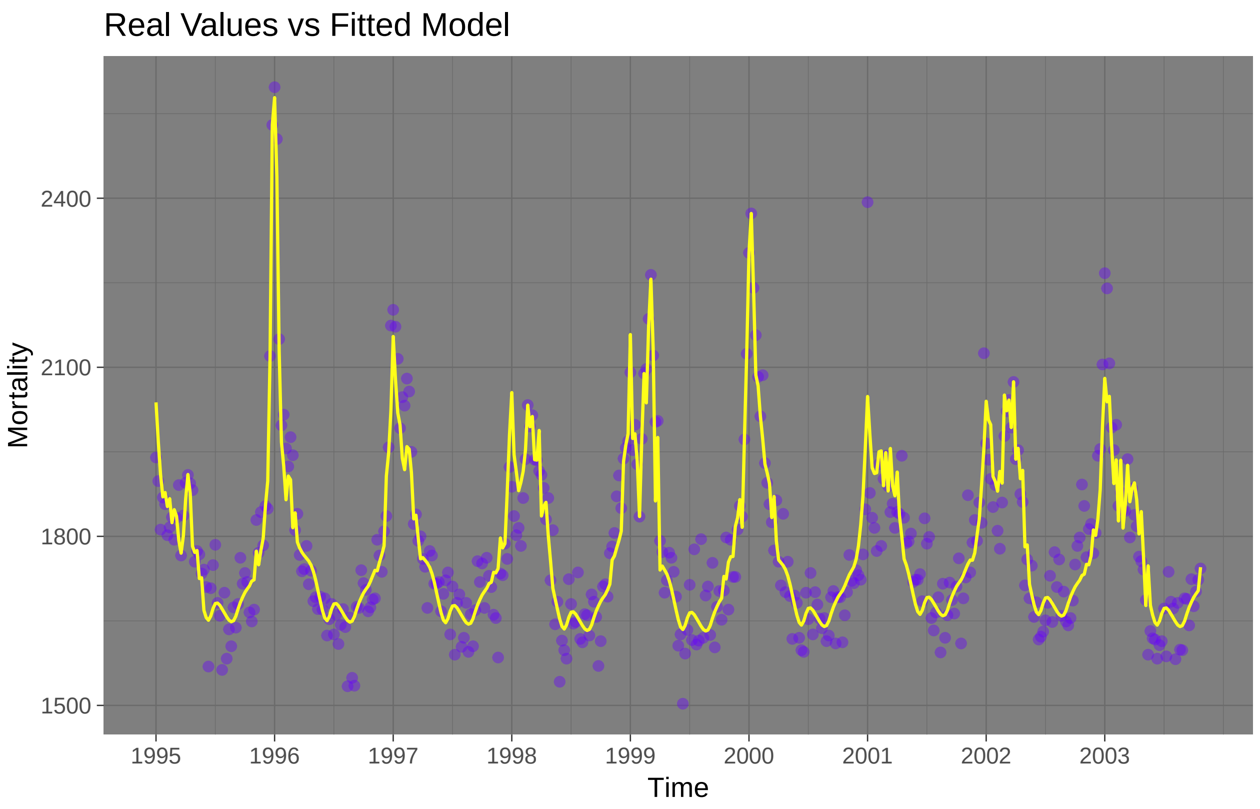
Use the model obtained in step 2 and plot the residuals and the influenza values against time (in one plot). Is the temporal pattern in the residuals correlated to the outbreaks of influenza?



In task 1.1, we have seen there is similarities between peaks of mortality and influenza. If we compare influenza values and residuals of the model that we created with linear function of year and spline function of week, we can see that peak times are quite similar again. Other times residuals are not following a regular pattern. But as we can see we have more mispredictions around the beginning of the years and we have influenza data which also peaks early months in each year. This similarity makes reasonable to add influenza to our model in order to see the effect of the variable to our predictions. As we observed before, we could not capture exact peaks with only week and spline function.

Task 1.6

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.



In this task, we train GAM model with spline functions of year, weak and influenza variables. The results of this model shows us influenza has a strong effect on the mortality predictions. After drawing the plot of real values and fitted model above, we can see that this model captured peaks better than the other models. Models that we fitted before only predict with a regular same pattern for each year, but this model has different predictions. and different heights on peaks. We have predictions even the outliers in 1996.

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)

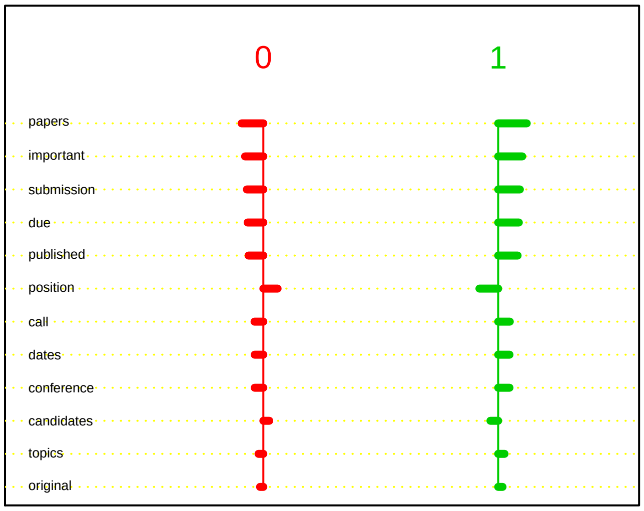
Task 2.1

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.

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"
```

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

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

Table 1: Confusion Matrix of NSC with Test Data

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

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

*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.*
b. Support vector machine with "vanilladot" kernel.
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?

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

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.

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

```r
knitr::opts_chunk$set(echo = FALSE, fig.width = 4.5, fig.height = 3,
                      fig.align = "center",
                      warning = F, error = F, message = F)

library(readxl)   # for reading excel files
library(ggplot2)  # for plots
library(gridExtra) # for plot grids
library(kableExtra)

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"))
}

################################### TASK 1.1 ###################################

df_mortal = read_xlsx("../dataset/Influenza.xlsx")


plot_df = rbind(data.frame(Time = df_mortal$Time,
                           Value = df_mortal$Mortality,
                           label = "Mortality"),
                data.frame(Time = df_mortal$Time,
                           Value = df_mortal$Influenza,
                           label = "Influenza"))
```

```r
plot_tseries = ggplot(plot_df, aes(x = Time)) +
  geom_line(aes(y = Value, col = label)) +
  labs(title="Time Series Chart",
       subtitle="Influenza and Mortality by time",
       col = "Variable") +
  theme_dark() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("#ffff19", "#ff96ca")) +
  scale_x_continuous(breaks = unique(df_mortal$Year))
plot_tseries

################################### TASK 1.2 ###################################

library(mgcv)

res = gam(Mortality ~ Year + s(Week, k=length(unique(df_mortal$Week))),
          family = gaussian,
          method = "GCV.Cp",
          data = df_mortal)
# library(rgl)
# library(akima)
# s = interp(df_mortal$Year,df_mortal$Week,
#   fitted(res))
# persp3d(s$x, s$y, s$z, col="red")

# store optimal sp
optimal_sp = res$sp

################################### TASK 1.3 ###################################
# "#ffff19", "#ff96ca"
# blue line is the model
p_t_1_3 = ggplot(data = df_mortal, aes(x = Time)) +
  geom_point(aes(y = Mortality), alpha=0.4, col = "#6600FF") +
  geom_line(aes(y = res$fitted.values), col="#ffff19", size=0.8) +
  labs(title = "Real Values vs Fitted Model") +
  theme_dark() +
  scale_x_continuous(breaks = unique(df_mortal$Year))
p_t_1_3
plot(res)

################################### TASK 1.4 ###################################

sp_values = c( 0, 0.5, 1:40)
plots = list()
devs = c()
dof = c()
for(i in 1:length(sp_values)){
  fit = gam(Mortality ~ Year + s(Week, k=length(unique(df_mortal$Week)),
                                 sp = sp_values[i]),
        family = gaussian,
        method = "GCV.Cp",
        data = df_mortal)
  # degrees of freedom
  dof =  c(dof, mgcv::pen.edf(fit))
  # store degrees of freedom
  devs = c(devs, fit$deviance)

  if(sp_values[i] %in% c(-1, 0, 0.5, 100)){
    pred = fitted(fit)

    plot_df = data.frame(Time = df_mortal$Time,
                         Mortality = df_mortal$Mortality,
                         Predictions = pred)

    pl = ggplot(data = plot_df, aes(x = Time)) +
      geom_point(aes(y = Mortality), alpha=0.4, col="#6600FF") +
      geom_line(aes(y = Predictions), col="#ffff19", size=0.6) +
      labs(title = "Real Values vs Fitted Model",
           subtitle = paste("sp = ", sp_values[i])) +
      theme_dark()

    plots[[paste("sp", i, sep = "_")]] = pl
  }
}



df_dev = data.frame(sp = sp_values,
                      deviance = devs,
                    dof = dof)

plot_dev = ggplot(df_dev, aes(x=sp, y=deviance)) +
  geom_line(col="#ffff19") +
  labs(title = "Deviance over Penalty Factor",
       x = "Penalty Factor", y = "Deviance") +
  theme_dark()

plot_dof = ggplot(df_dev, aes(x=sp, y=dof)) +
  geom_line(col="#ffff19") +
  labs(title = "Degrees of Freedom over Penalty Factor",
       x = "Penalty Factor", y = "Deviance") +
  theme_dark()

grid.arrange(grobs=list(plot_dev, plot_dof), ncol=2)

grid.arrange(grobs = plots, ncol=2)

#################################### TASK 1.5 ####################################

df_plot = rbind(data.frame(Time = df_mortal$Time,
                           Value = df_mortal$Influenza,
                           label = "Influenza"),
                data.frame(Time = df_mortal$Time,
                           Value = res$residuals,
                           label = "Residuals"))
```

```r
plot_res = ggplot(df_plot, aes(x = Time)) +
  geom_line(aes(y = Value, col = label)) +
  labs(title="Time Series Chart",
       subtitle="Influenza and Residuals by time",
       col = "Variable") +
  theme_dark() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("#ffff19", "#ff96ca")) +
  scale_x_continuous(breaks = unique(df_mortal$Year))
plot_res


#################################### TASK 1.6 ####################################

gam_1_6 = gam(Mortality ~ s(Year, k=length(unique(df_mortal$Year))) +
                            s(Week, k=length(unique(df_mortal$Week))) +
                            s(Influenza, k=length(unique(df_mortal$Influenza))),
          family = gaussian,
          method = "GCV.Cp",
          data = df_mortal)

df_plot = data.frame(Time = df_mortal$Time,
                       Mortality = df_mortal$Mortality,
                       Predictions = fitted(gam_1_6))

plot_mort = ggplot(data = df_plot, aes(x = Time)) +
    geom_point(aes(y = Mortality), alpha=0.4, col="#6600FF") +
    geom_line(aes(y = Predictions), col="#ffff19", size=0.6) +
    labs(title = "Real Values vs Fitted Model") +
    theme_dark() +
    scale_x_continuous(breaks = unique(df_mortal$Year))

plot_mort

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

library(pamr)

# 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]
```

```r
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")
```