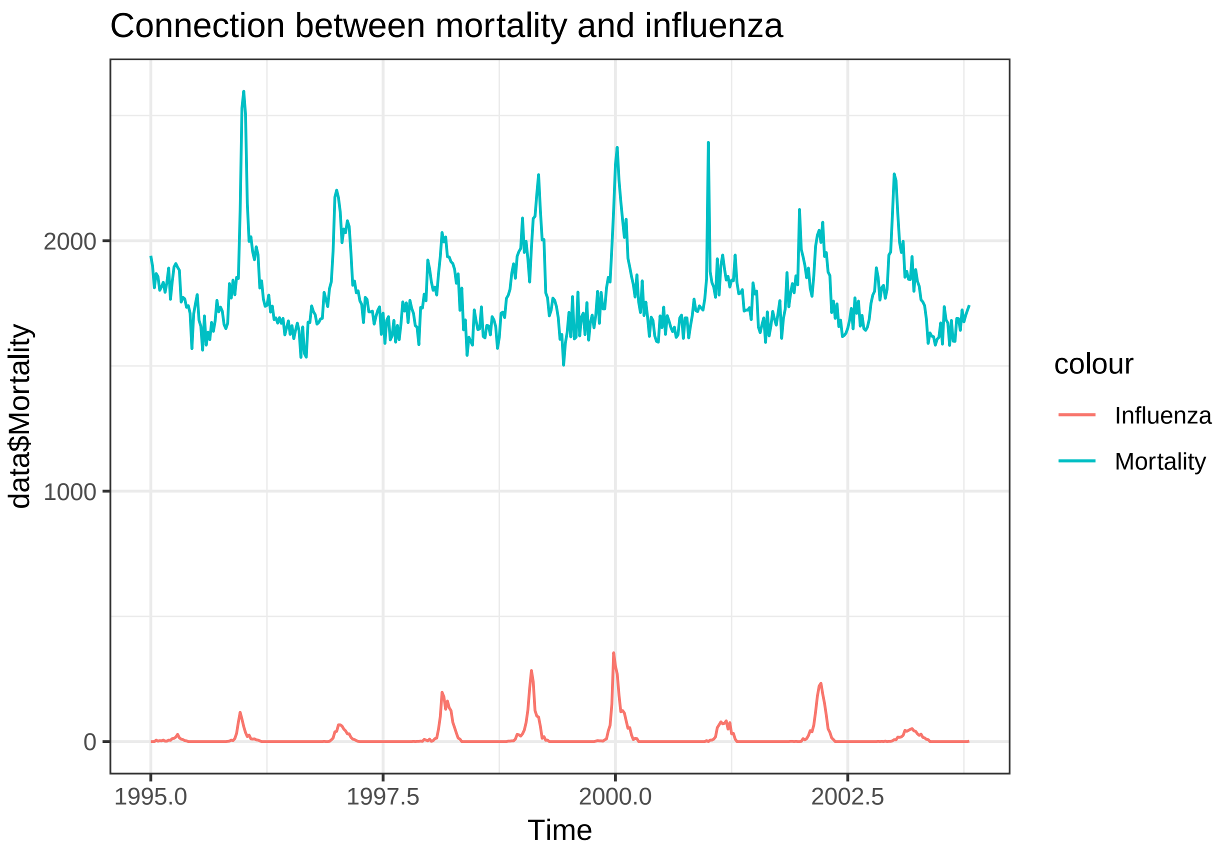
Block 2 Lab 2

Phillip Hölscher

17 12 2018

Assignment 1. Using GAM and GLM to examine the mortaily rates

1.1



In this visualization are we able to see a connection between mortality and influenza. The red line, which represents influenza, as time by the time a new up-station, at the same time does mortality has an up-station. It´s possible to see both have at the same time and the same amount of up-stations. But it´s difficult to say how the amount of influenza influence the mortality. When influenza has a small kick out, does that not mean the mortality also just has a small kick out.
For example, around 1996 is a small kick out of influenza to recognize, but it´s the biggest kick out of mortality in the whole time series. Between 1997.5 and 2000.0 are two big kick outs for influenza for this time series, which is not connected with a high mortality kick out.

1.2

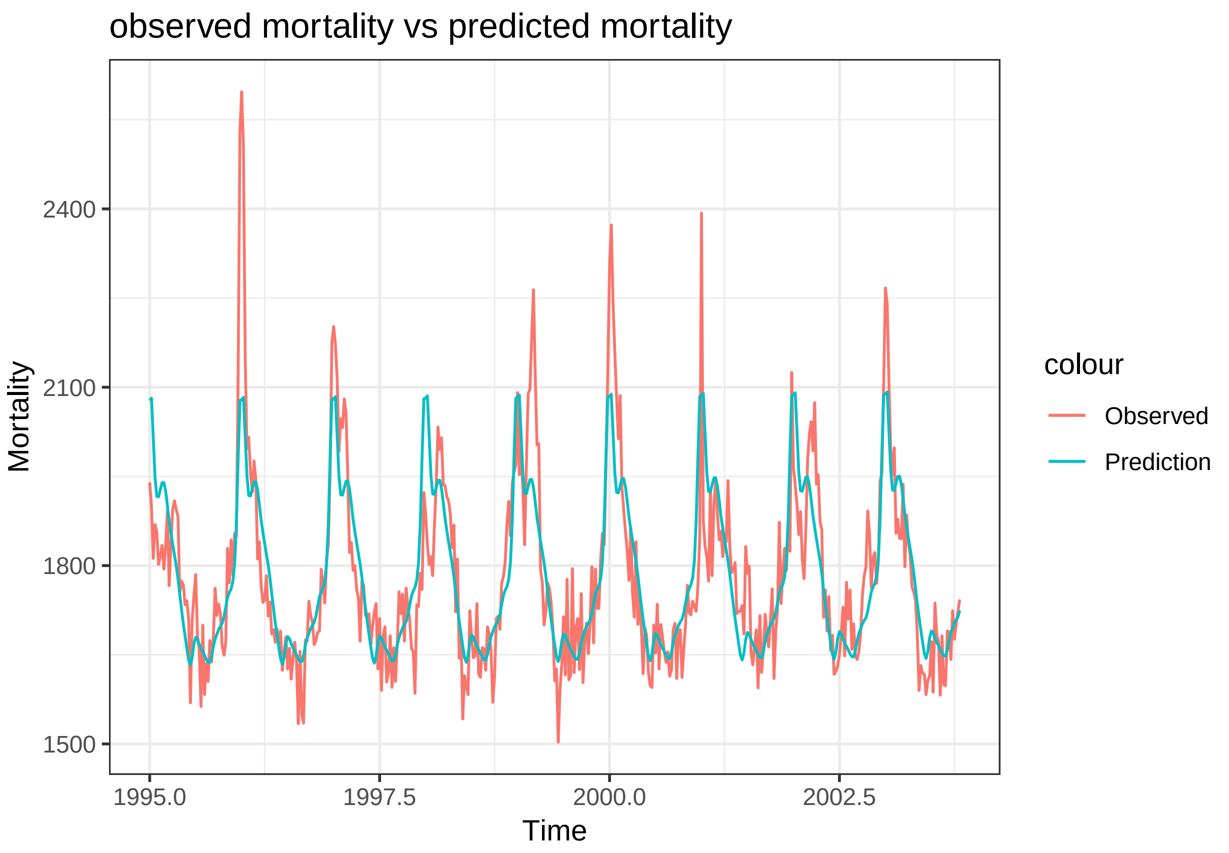
Probabilistic model:

$$y = Mortality$$

$$y = N(\mu, \sigma^2)$$

$$\hat{y} = \beta_0 + \beta_1 Year_i + f(Week_i) + \epsilon_i$$

1.3



Comment the quality of the fit: It can be seen that the course of the prediction corresponds very much to that of the obversved values. Especially the respective fluctuations upwards and downwards at the right points in time reflect the prediction well. However, the exact height or depth of the fluctuation is usually not well reflected. The prediction does not exceed 2500 in any year, but the observed values are 7 times higher. The same can be observed with the depth, which is not quite as inaccurate as the altitude, but rarely reflects the true value.

Investivate the output of GAM:

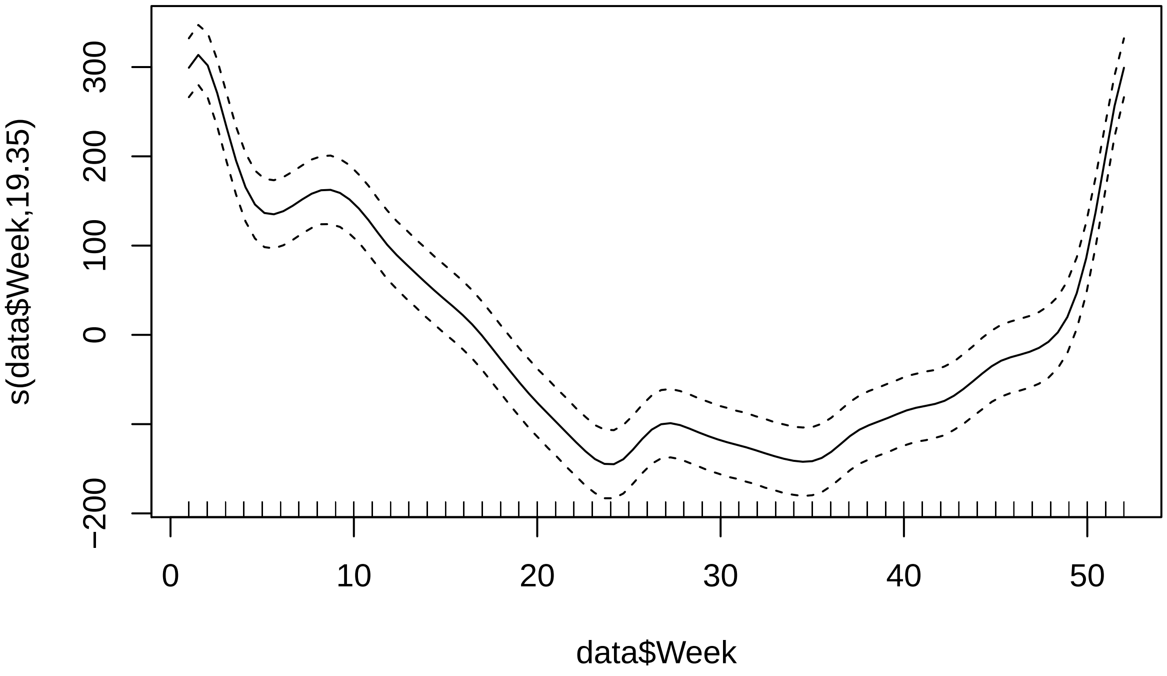
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## data$Mortality ~ data$Year + s(data$Week, k = length(unique(data$Week)),
##     bs = "cp")
##
## Estimated degrees of freedom:
## 19.3  total = 21.35
##
## GCV score: 9086.051
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## data$Mortality ~ data$Year + s(data$Week, k = length(unique(data$Week)),
##     bs = "cp")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -880.254   3419.848  -0.257    0.797
## data$Year      1.333      1.711   0.779    0.436
##
## Approximate significance of smooth terms:
##                edf Ref.df     F p-value
## s(data$Week) 19.35     50 18.45  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.667   Deviance explained = 68.1%
## GCV = 9086.1  Scale est. = 8663.5    n = 459
```

```
## s(data$Week)
##     61.64636
```

- significant is the term: s(data$Week)

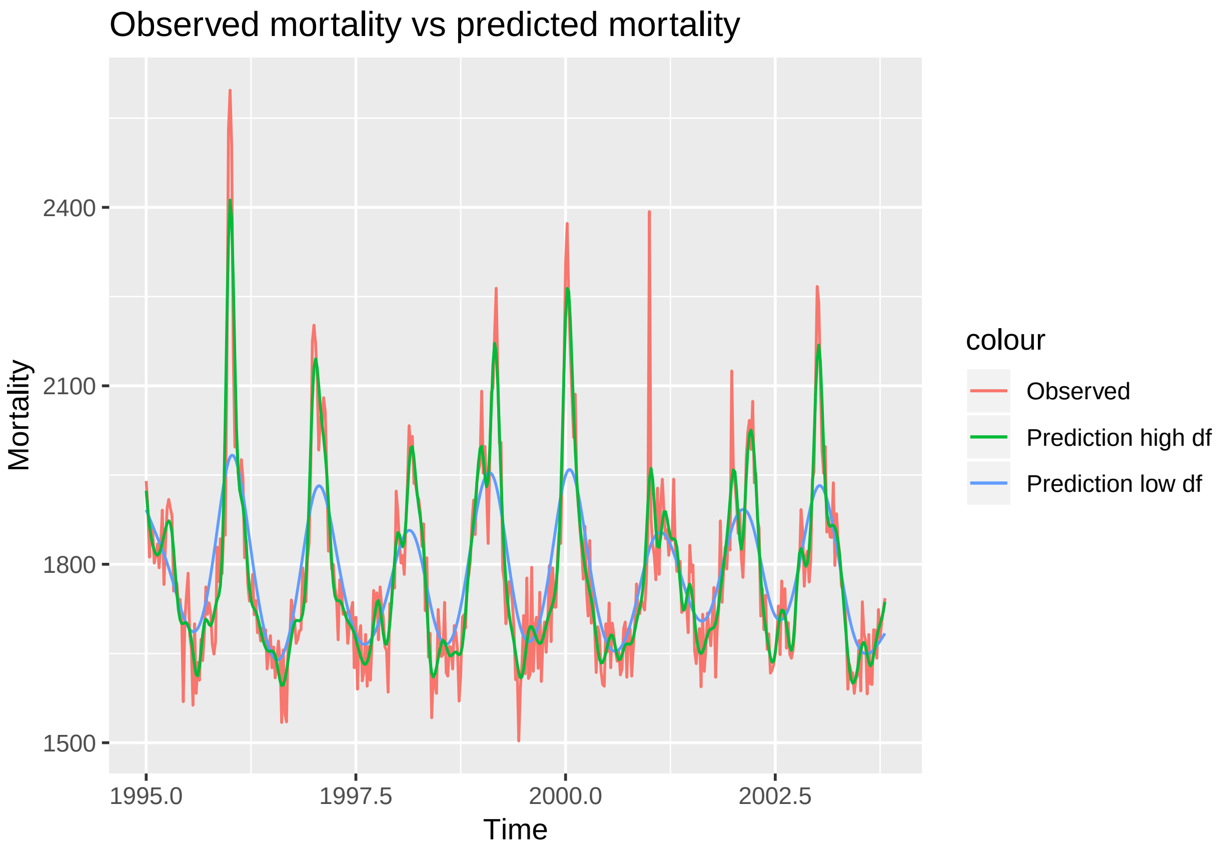
Plot spline component



Interpretation: In this plot can we see influenza over the whole year separated in weeks of the year. The highest values do we see at the beginning of the year and of the end. After week 10 we can see that influenza decreases strongly. Between week 20 and 40, the influenza is low. After week 40 an especially after week 45 does the influenza increase strongly. Which makes sense, at the cold time of the year in Europe, the beginning of the year and end of the year, does more people sicken on influenza than in the summertime.

1.4

The influance of penalty function



Degrees of freedom low:

```
## [1] 2.147448e-05
```

```
## [1] 25.00381
```

Degrees of freedom high:

```
## [1] 2.579748e-08
```

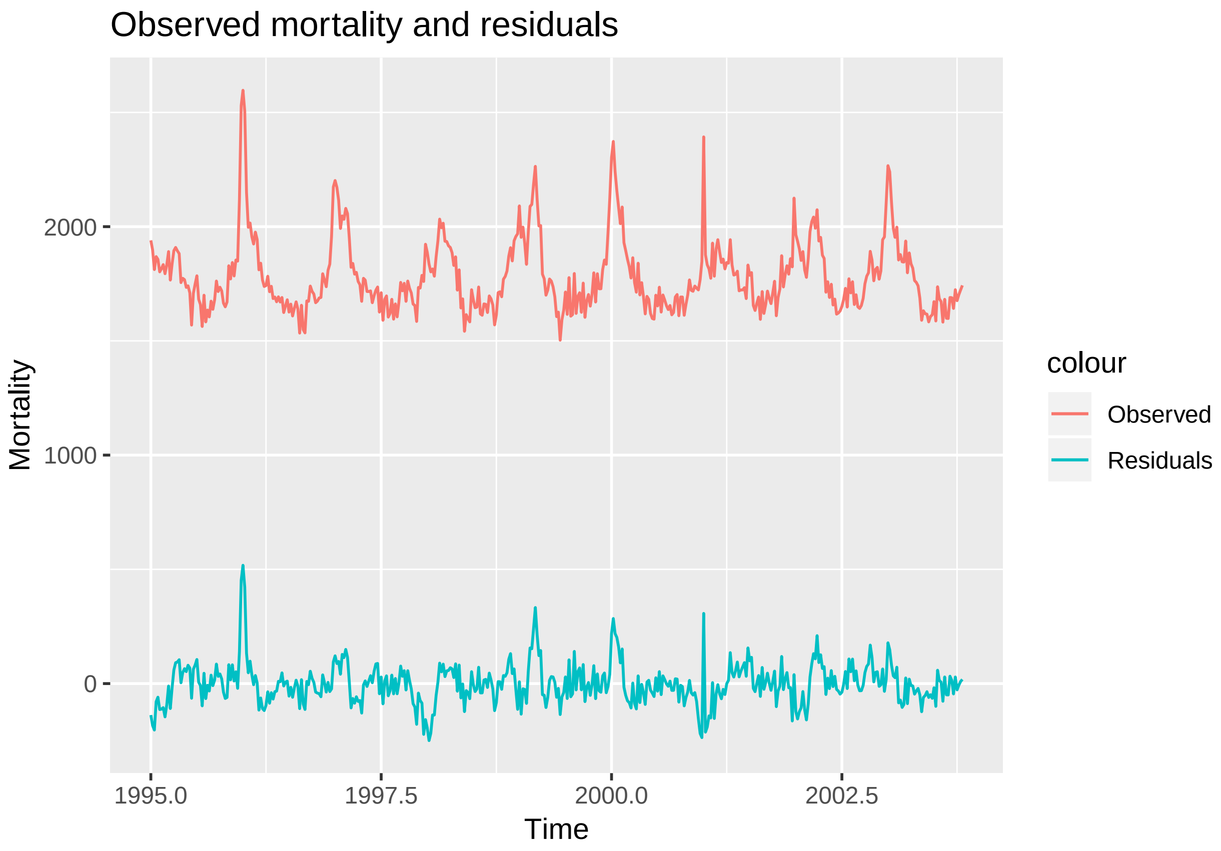
```
## [1] 100.0082
```

In this visualization can we see the prediction with a higher lambda fit better than with a low lambda. The high degree of freedom fits also good in the hights of the kick outs, the prediction with the low degree of freedom instead just follows the trend. We can also see the lambda for the “low df” is bigger than the lambda for the “high df”. Sinze lambda is a parameter of penalization does the prediction with a lower penalization (high df) fit the prediction better.

Estimated deviance:

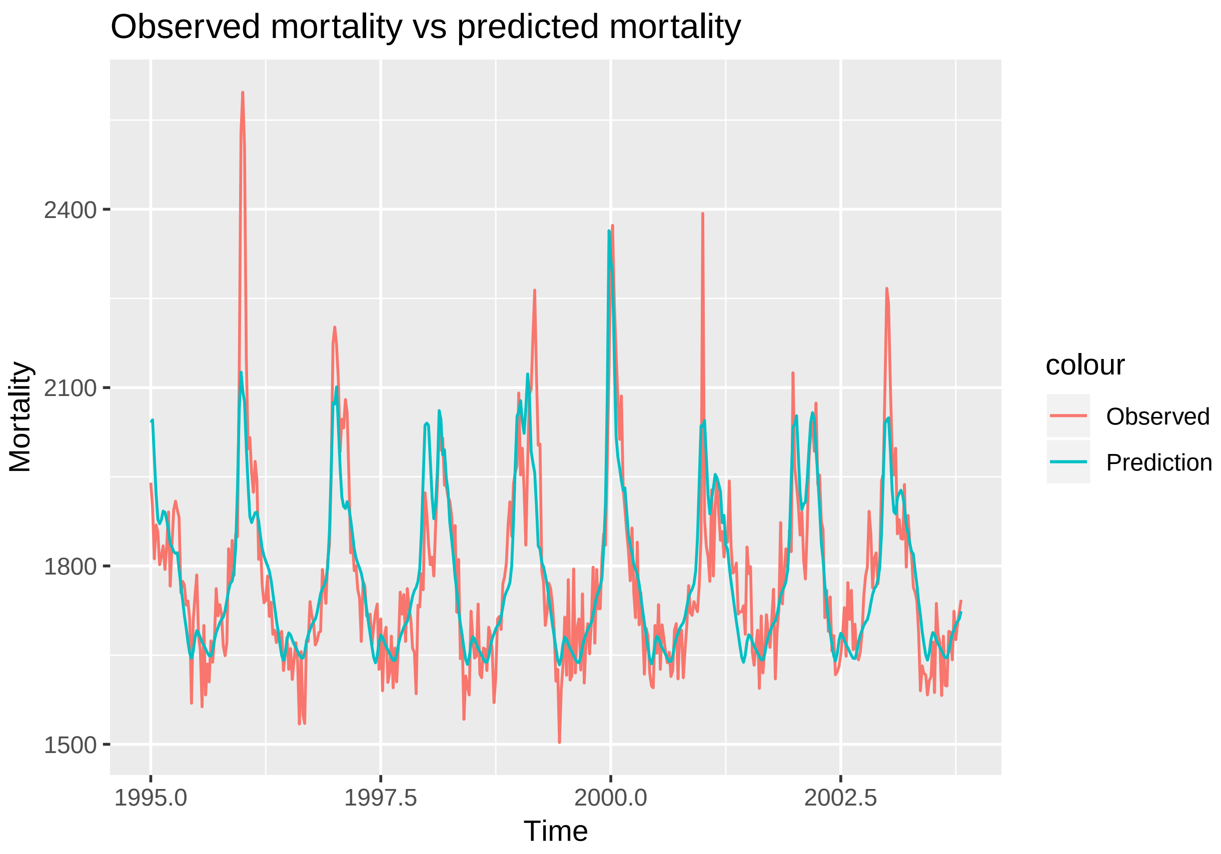
```
## Call:
## smooth.spline(x = data$Time, y = data$Mortality, df = 25)
##
## Smoothing Parameter  spar= 0.5618014  lambda= 2.147448e-05 (12 iterations)
## Equivalent Degrees of Freedom (Df): 25.00381
## Penalized Criterion (RSS): 4699898
## GCV: 11453.26
## Call:
## smooth.spline(x = data$Time, y = data$Mortality, df = 100)
##
## Smoothing Parameter  spar= 0.1575857  lambda= 2.579748e-08 (16 iterations)
## Equivalent Degrees of Freedom (Df): 100.0082
## Penalized Criterion (RSS): 1321590
## GCV: 4706.959
```

1.5



Analysis: In this visualization are we able to observe a relationship between the mortality and the residuals of the fit. In many cases does the time of the kick out in the residuals fit with the kick out of the mortality. But with hights of the kick out does not always fit together with the kick out of the residuals, also are negative kick outs around 1998 and 2002 in the residuals to observe, which are actually positive kick-outs in the mortality.

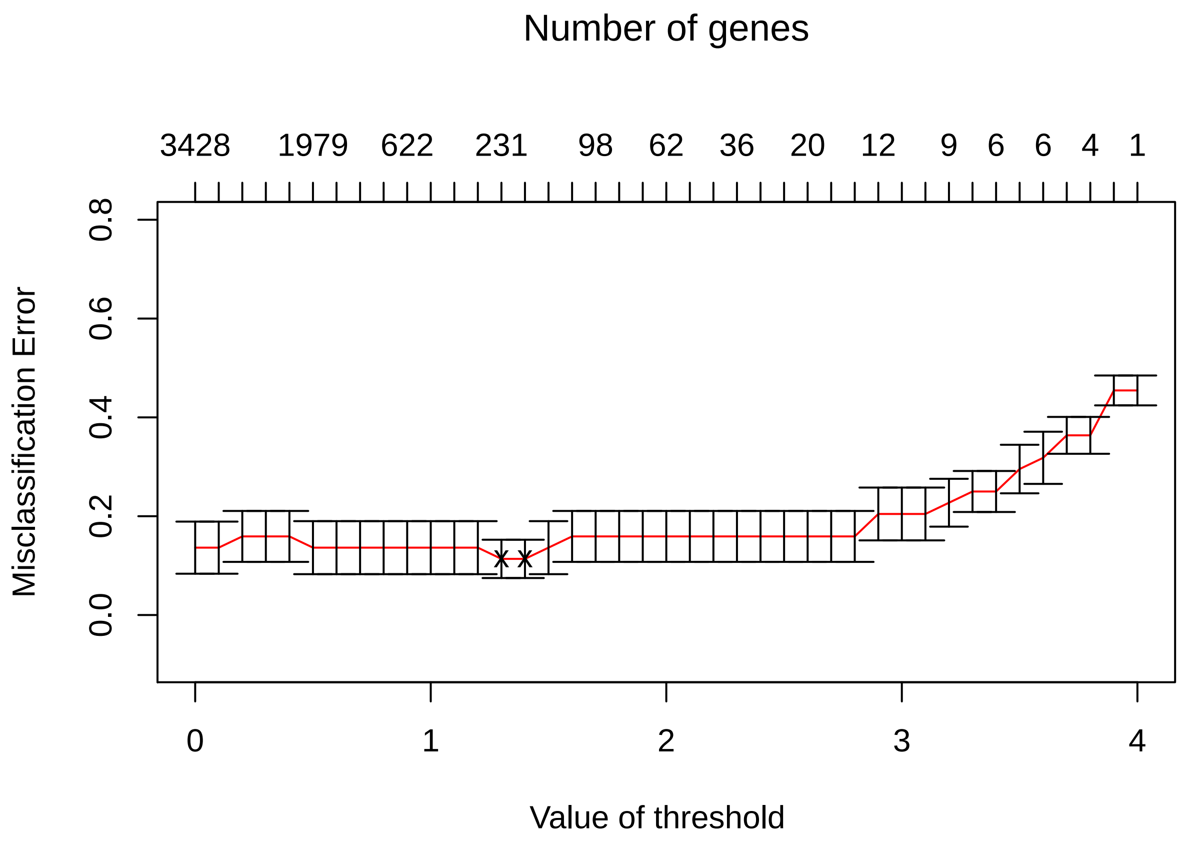
1.6

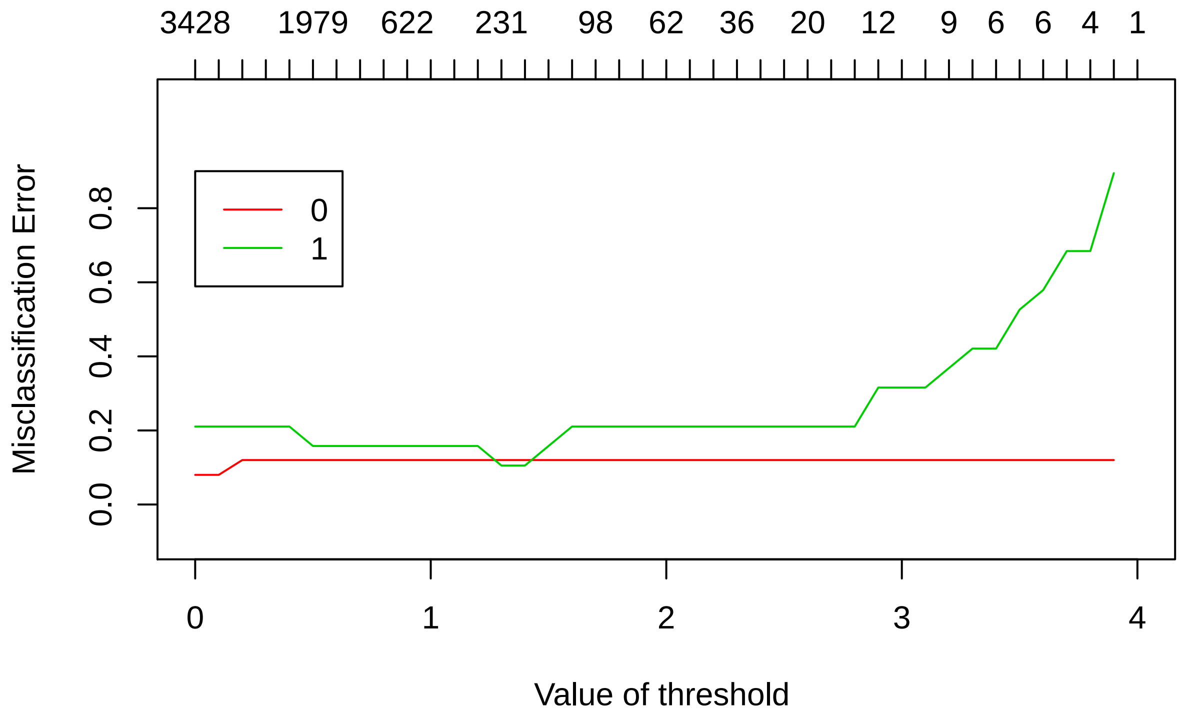


We can still see that the height and depth of the kick outs are still not optimally in line with the original values, but better than in Task 1.3.

Assignment 2. High-dimensional methods

Missclassification Error plot:


Number of genes
3428
1979
622
231
98
62
36
20
12
9
6
6
4
1
Misclassification Error
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
Value of threshold


3428
1979
622
231
98
62
36
20
12
9
6
6
4
1
Misclassification Error
0.0
0.2
0.4
0.6
0.8
0
1
0
1
2
3
4
Value of threshold

Print out of the cvmodel:

```
## Call:
## pamr.cv(fit = model, data = mydata_train)
##    threshold nonzero errors
## 1  0.0       3428    6
## 2  0.1       3409    6
## 3  0.2       3114    7
## 4  0.3       3024    7
## 5  0.4       3000    7
## 6  0.5       1979    6
## 7  0.6        852    6
## 8  0.7        841    6
## 9  0.8        673    6
## 10 0.9        622    6
## 11 1.0        297    6
## 12 1.1        293    6
## 13 1.2        272    6
## 14 1.3        231    5
## 15 1.4        170    5
## 16 1.5        138    6
## 17 1.6        129    7
## 18 1.7         98    7
## 19 1.8         88    7
## 20 1.9         71    7
## 21 2.0         62    7
## 22 2.1         47    7
## 23 2.2         43    7
## 24 2.3         36    7
## 25 2.4         30    7
## 26 2.5         20    7
## 27 2.6         20    7
## 28 2.7         14    7
## 29 2.8         12    7
## 30 2.9         12    9
## 31 3.0         12    9
## 32 3.1         11    9
## 33 3.2          9    10
## 34 3.3          9    11
## 35 3.4          6    11
## 36 3.5          6    13
## 37 3.6          6    14
## 38 3.7          6    16
## 39 3.8          4    16
## 40 3.9          2    20
## 41 4.0          1    20
```

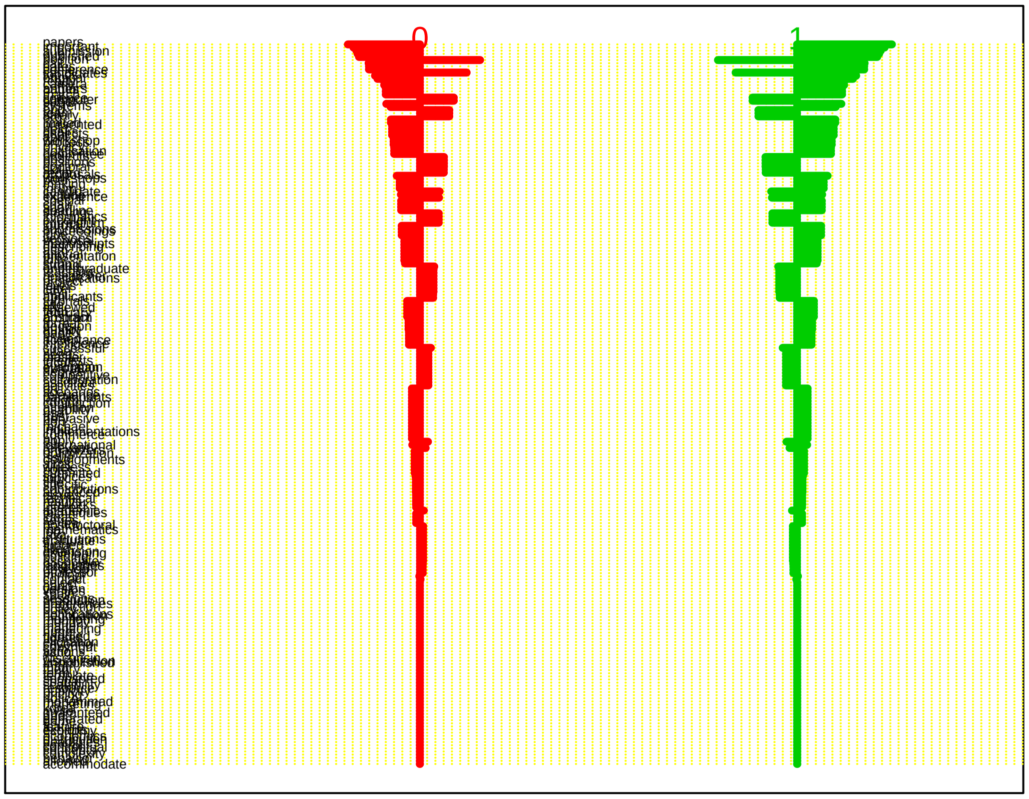
In the visualization we can see that the misclassification of between the beginning and middle of 1 and 2 lies. If we look at the print out we see that the threshold with the lowest error must be 1.3 or 1.4.

Min threshold:

```
## [1] 1.3
```

The threshold 1.3 has the lowest error of the cross validation model and is therefore used as a threshold value in the further course.

Centroid plot (threshold=1.3):



The selected features - Threshold = 1.3:

Number of features:

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

List of the 10 most contributing features:

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

2.a

Test error & number of contributing features

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

Table 1: Comparative Table

	Nearest.shrank	Elastic.net	SVM
Test error	0.1	0.1	0.05
Features	231.0	33.0	4702.00

In this table do we see the best error rate provides the model of support vector machine, as well it has the highest number of features. Since the number of features tells us something about the complexity of the model, do we not choose for support vector machine. The model of nearest shrank and elastic net does have the same value of test error, but the elastic net does just have 32 feature, do prefer this model.

2.3

List of rejected features:

```
##               name      p_value             L rejected
## 3036        papers 1.116910e-10 -1.063366e-05      yes
## 4060    submission 7.949969e-10 -2.126675e-05      yes
## 3187      position 8.219362e-09 -3.189310e-05      yes
## 3364     published 1.835157e-07 -4.235158e-05      yes
## 2049     important 3.040833e-07 -5.286478e-05      yes
## 596           call 3.983540e-07 -6.340428e-05      yes
## 869     conference 5.091970e-07 -7.392721e-05      yes
## 607     candidates 8.612259e-07 -8.420896e-05      yes
## 1045         dates 1.398619e-06 -9.430534e-05      yes
## 3035         paper 1.398619e-06 -1.049391e-04      yes
## 4282        topics 5.068373e-06 -1.119031e-04      yes
## 2463       limited 7.907976e-06 -1.196973e-04      yes
## 606      candidate 1.190607e-05 -1.263330e-04      yes
## 599         camera 2.099119e-05 -1.278816e-04      yes
## 3433         ready 2.099119e-05 -1.385154e-04      yes
## 389        authors 2.154461e-05 -1.485958e-04      yes
## 3125           phd 3.382671e-05 -1.469474e-04      yes
## 3312      projects 3.499123e-05 -1.564167e-04      yes
## 2974           org 3.742010e-05 -1.646216e-04      yes
## 681         chairs 5.860175e-05 -1.540737e-04      yes
## 1262           due 6.488781e-05 -1.584214e-04      yes
## 2990      original 6.488781e-05 -1.690552e-04      yes
## 2889  notification 6.882210e-05 -1.757547e-04      yes
## 3671        salary 7.971981e-05 -1.754907e-04      yes
## 3458        record 9.090038e-05 -1.749439e-04      yes
## 3891        skills 9.090038e-05 -1.855777e-04      yes
## 1891          held 1.529174e-04 -1.341945e-04      yes
## 4177          team 1.757570e-04 -1.219886e-04      yes
## 3022         pages 2.007353e-04 -1.076441e-04      yes
## 4628      workshop 2.007353e-04 -1.182779e-04      yes
## 810      committee 2.117020e-04 -1.179450e-04      yes
## 3285   proceedings 2.117020e-04 -1.285788e-04      yes
## 272          apply 2.166414e-04 -1.342731e-04      yes
## 4039        strong 2.246309e-04 -1.369174e-04      yes
## 2175 international 2.295684e-04 -1.426137e-04      yes
## 1088        degree 3.762328e-04 -6.582996e-06      yes
## 1477      excellent 3.762328e-04 -1.721677e-05       yes
## 3191           post 3.762328e-04 -2.785054e-05       yes
## 3243      presented 3.765147e-04 -3.820241e-05       yes
```

In this table are we able to see the 39 features which got rejected. This means, all the features have a clear relation to the conference text. As well do we see the p-value, the calculated L-value (Benjamini Hochberg) and if the feature to rejected or not.

```r
knitr::opts_chunk$set(echo = FALSE)
# list of all libraries
#install.packages("readxl") # it´s an xlsx file - need the readxl package for this
library(readxl)
library(ggplot2)
#install.packages("mgcv")
library(mgcv)
# set working directory
#setwd("X")
data <- read_excel("Influenza.xlsx")
# time series plot
col <- c("Mortality" = "blue", "Influenza" = "green")
ggplot() +
  geom_line(aes(x = data$Time, y = data$Mortality, color = "Mortality")) +
  geom_line(aes(x = data$Time, y = data$Influenza, color = "Influenza")) +
  xlab("Time") +
  ggtitle("Connection between mortality and influenza") +
  theme_bw()

# fit GAM model
# https://www.rdocumentation.org/packages/mgcv/versions/1.8-26/topics/gam

gam_fit = gam(data$Mortality ~ data$Year +
                s(data$Week, k=length(unique(data$Week)),
                  bs = "cp"), # bs = "cp" ???
              data = data,
              family = gaussian, # gaussian is defult
              method = "GCV.Cp") # method generalized cross-validation
# method : The smoothing parameter estimation method
# "GCV.Cp" to use GCV for unknown scale parameter
# bs: B-Spline Basis for Polynomial Splines

# prediction for gam
gam_pred <- predict(gam_fit)

# plot - observed mortality vs predicted mortality
col <- c("Observed" == "#1abc9c", "Prediction" == "#e67e22")
ggplot() +
  geom_line(aes(x = data$Time, y = data$Mortality, color = "Observed")) +
  xlab("Time") +
  ylab("Mortality") +
  ggtitle("observed mortality vs predicted mortality") +
  geom_line(aes(x = data$Time, y = gam_pred ,color = "Prediction")) +
  theme_bw()
print(gam_fit)
summary(gam_fit)
gam_fit$sp
plot(gam_fit)
# increase lambda -> df_lambda decrease
# higher lambda higher penilize
gam_smooth_low = smooth.spline(data$Time, data$Mortality, df = 25)
gam_smooth_high = smooth.spline(data$Time, data$Mortality, df = 100)
```

```r
gam_smooth_low_pred = predict(gam_smooth_low)
gam_smooth_high_pred = predict(gam_smooth_high)


col <- c("Observed" == "#1abc9c", "Prediction high lmapda" == "#3498db" , "Prediction low lmapda" == "#3
ggplot() +
  geom_line(aes(x = data$Time, y = data$Mortality, color = "Observed")) +
  xlab("Time") +
  ylab("Mortality") +
  ggtitle("Observed mortality vs predicted mortality") +
  geom_line(aes(x = data$Time, y = gam_smooth_low_pred$y ,color = "Prediction low df")) +
  geom_line(aes(x = data$Time, y = gam_smooth_high_pred$y ,color = "Prediction high df"))
gam_smooth_low$lambda #2.147448e-05
gam_smooth_low$df #25.00381
gam_smooth_high$lambda #2.579748e-08
gam_smooth_high$df #100.0082
gam_smooth_low
gam_smooth_high

col <- c("Observed" = "blue", "Residuals" = "red")
ggplot() +
  geom_line(aes(x = data$Time, y = data$Mortality, color = "Observed")) +
  xlab("Time") +
  ylab("Mortality") +
  ggtitle("Observed mortality and residuals") +
  geom_line(aes(x = data$Time, y = gam_fit$residuals, color = "Residuals"))
  #geom_line(aes(x = data$Time, y = gam_smooth_low_pred$y ,color = "Prediction low lambda")) +
  #geom_line(aes(x = data$Time, y = gam_smooth_high_pred$y ,color = "Prediction high lambda"))


# gam_fit = gam(data$Mortality ~ data$Year +
#                 s(data$Week, k=length(unique(data$Week)),
#                   bs = "cp"),
#               data = data,
#               method = "GCV.Cp")

# create new fitting model
gam_fit6 = gam(data$Mortality ~ data$Influenza +
                s(Year,k=length(unique(data$Year)), bs="gp")+
                s(Week,k=length(unique(data$Week)), bs="cp"),
              family = gaussian,
              data=data,
              method = "GCV.Cp")

# create prediction
gam_pred6 = predict(gam_fit6)

# create visualization
col <- c("Observed" == "blue", "Prediction" == "green")
ggplot() +
  geom_line(aes(x = data$Time, y = data$Mortality, color = "Observed")) +
  xlab("Time") +
  ylab("Mortality") +
  ggtitle("Observed mortality vs predicted mortality") +
  geom_line(aes(x = data$Time, y = gam_pred6 ,color = "Prediction"))

#install.packages("pamr")
library(pamr)
#install.packages("kernlab")
library(kernlab)
#install.packages("glmnet")
library(glmnet)
#install.packages("kernlab")
library(kernlab)
#install.packages("sgof")
library(sgof)
# load data
# set woring directory
#setwd("X")
# read data
data = read.csv2("data.csv",
                 fileEncoding = "ISO-8859-1")

# devide data into train (70%) and test (30%) set - without scaling
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.7))
train=data[id,]
test=data[-id,]

#train
rownames(train) = 1:nrow(train)
x_train = t(train[,-4703]) # remove dependent variable
y_train = train[[4703]] # vector of the dependent variable
mydata_train = list(x = x_train,y=as.factor(y_train),geneid=as.character(1:nrow(x_train)), genenames=ro

#test
rownames(test) = 1:nrow(test)
x_test = t(test[,-4703])
y_test = test[[4703]]
mydata_test = list(x = x_test,y=as.factor(y_test),geneid=as.character(1:nrow(x_test)), genenames=rownam

# create the model
model = pamr.train(mydata_train,threshold=seq(0,4, 0.1))

# choice threshold by cv
cvmodel = pamr.cv(model,mydata_train)
pamr.plotcv(cvmodel)
print(cvmodel)
# the threshold for the min error of cvmodel
best_thresbold = cvmodel$threshold[which.min(cvmodel$error)]
best_thresbold
pamr.plotcen(model, mydata_train, threshold=best_thresbold)
a = pamr.listgenes(model,mydata_train,threshold=best_thresbold)
nrow(a)
#cat(paste(colnames(data)[as.numeric(a[,1])], collapse='\n'))
top10 = as.matrix(colnames(data)[as.numeric(a[,1])][1:10])
top10
pred_model = pamr.predict(model,
                          newx = x_test,
                          threshold = 1) # also for the fit threshold of 1
cm_pred_model = table(y_test, pred_model)
test_error_nearestshrank = (cm_pred_model[1,2] + cm_pred_model[2,1]) / sum(cm_pred_model)
# Elastic net
x_train = t(x_train) # transpose x_train back to normal
x_test = t(x_test)

# fit the elastic net
elastic_net = cv.glmnet(x = x_train,
                        y = y_train,
                        family = "binomial",
                        alpha = 0.5)

# create prediction
elastic_net_pred = predict.cv.glmnet(elastic_net,
                                     newx = x_test,
                                     type = "class",
                                     s="lambda.min")
# s penalty parameter
cm_elastic_net  = table(y_test, elastic_net_pred)
test_error_elastic_net = (cm_elastic_net[1,2] + cm_elastic_net[2,1]) / sum(cm_elastic_net)

# SVM
# svm() function does not support vanilladot
# svm_fit = svm(x = x_train,
#                y = y_train,
#                kernel = "vanilladot")

# https://www.rdocumentation.org/packages/kernlab/versions/0.9-27/topics/ksvm
# used function - ksvm(), package
svm_fit  = ksvm(x = x_train,
                y = y_train,
                kernel = "vanilladot")
                #,scale = FALSE, # Variable(s) `' constant. Cannot scale data.
                #type = "C-svc") # C-svc C classification

svm_fit_pred = predict(svm_fit,
                       newdata = x_test)

cm_svm = table(y_test, svm_fit_pred)
test_error_svm = (cm_svm[1,2] + cm_svm[2,1]) / sum(cm_svm)
# comparing the results
# create df with the three values of the test error
test_error_df = data.frame(
  "Nearest shrank" = test_error_nearestshrank,
  "Elastic net" = test_error_elastic_net,
  "SVM" = test_error_svm
)
```

```r
cf<-as.matrix(coef(elastic_net, elastic_net$lambda.min))
features_nearest_shrank = nrow(a)
features_svm_fit  = dim(data)[2] -1
features_elasticnet = length(names(cf[cf!=0,]))

features_lengt = c(features_nearest_shrank,features_elasticnet, features_svm_fit)

test_error_df = rbind(test_error_df, features_lengt )

rownames(test_error_df)[1] = "Test error"
rownames(test_error_df)[2] = "Features"


# summary(model)
# svm_fit
# summary(elastic_net)
knitr::kable(test_error_df, caption = "Comparative Table")
# Benjamin Hochberg
# laod the data
data = read.csv2("data.csv",
                 fileEncoding = "ISO-8859-1")
# create a data frame to save p-value & name
name_p_value = data.frame(name = character(),
                          p_value = numeric(),
                          stringsAsFactors = FALSE)
# for loop to fill data frame with wanted values
for (i in 1:4702) {
    x <- data[,i]
  p <- t.test( x ~ Conference,
               data = data,
               alternative = "two.sided" # default - don´t neet this
               )[["p.value"]]
  colname = colnames(data)[i]
 name_p_value = rbind(name_p_value, data.frame(colname,p))
}
colnames(name_p_value) = c("name", "p_value")

# -----
# TEST
# bla_test = BH(p_value, alpha = 0.05)
# bla_test_class = ifelse(bla_test$Adjusted.pvalues > 0.05, "Don´t reject", "Reject")
# which(bla_test_class == "Don´t reject")
# ----
# get the same results in the following calculation


# values to use the benjamin hochberg algorithm
alpha = 0.05
M = ncol(data)-1

# bring the p-values in order
# copy of the original data
name_p_value_order = name_p_value
name_p_value_order = name_p_value_order[order(name_p_value_order$p_value),]

# calculate the prob for L
L <- c()
for (i in 1:M) {
  L[i] = name_p_value_order$p_value[i] - ((alpha *i)/ M)
}
# include L in the data frame name_p_value_order
name_p_value_order$L = L

# save rejecteion region
rejection_switch_vector <- which(name_p_value_order$L < 0)
rejection_switch_max <- max(which(L < 0))

reject_yes_no <- c()
for (i in 1:length(L)) {
  if( i <= rejection_switch_max){
    reject_yes_no[i] = "yes"
  } else{
    reject_yes_no[i] = "no"
  }
}

# add vector rejection_yes_no to data frame name_p_value_order
name_p_value_order$rejected = reject_yes_no

# create data frame just with rejected variables
name_p_value_order_rejected = name_p_value_order[1:rejection_switch_max,]
name_p_value_order_rejected
```