Introduction to Machine Learning - Lab 2 Block 2

Ugurcan Lacin

December 4, 2017

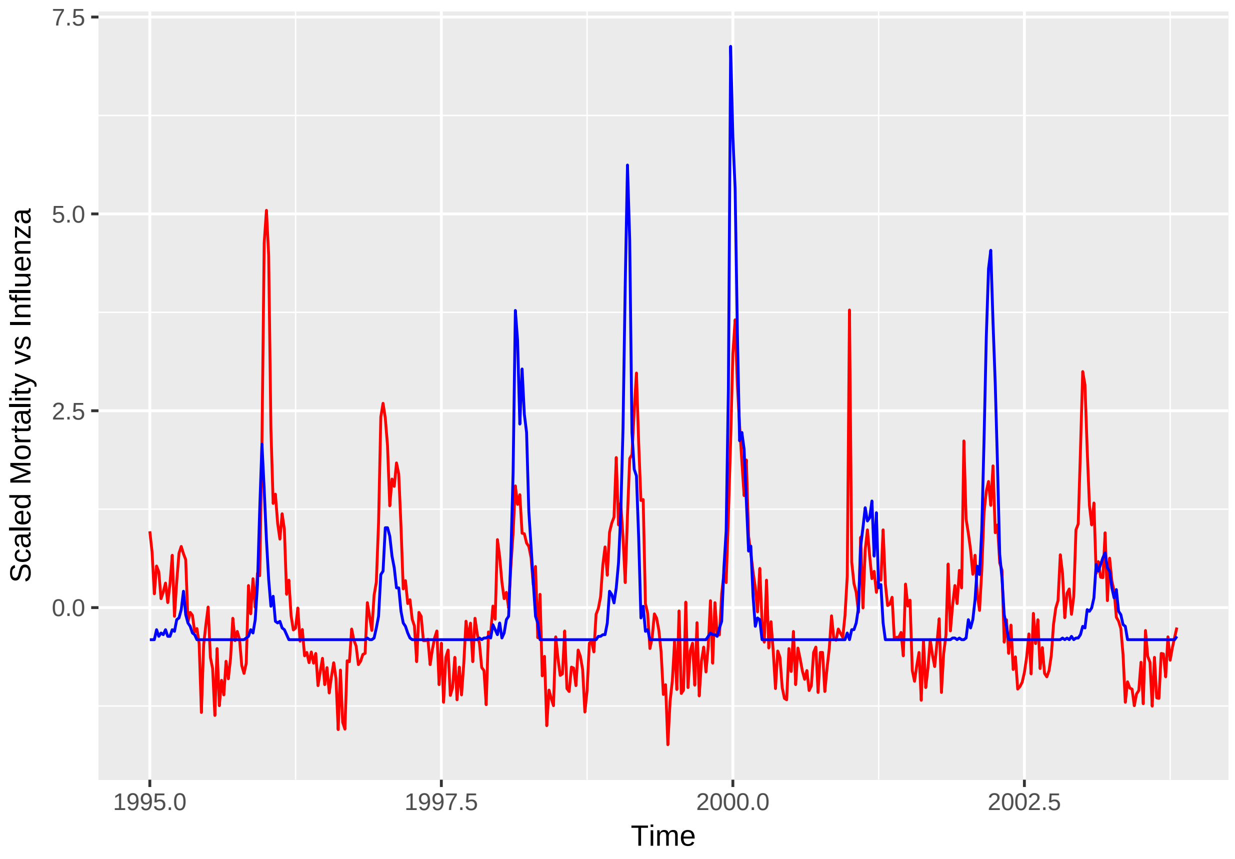
Assignment 1

In this assignment, we will try to find explict model for Influenza data set. The data set has some features such as Influenza,Year,Week. We will try to predict Mortality by applying those features. To understand which feature has more effect on our target variable which is Mortality, we will mostly investigate features.

Part 1

In the beginning, We applied scale function to examine the data. This is because Mortality and Influenza are not in same scale.

```r
library(readxl)
library(ggplot2)
library(mgcv)
library(pamr)

data <- as.data.frame(read_xlsx("Influenza.xlsx"))
scaled <- data.frame(scale(data))
scaled$Time <- data$Time
ggplot() + geom_line(data = scaled, aes(Time, Mortality), color="red") +
  geom_line(data = scaled, aes(Time, Influenza), color="blue") +
   xlab("Time") + ylab("Scaled Mortality vs Influenza")
```



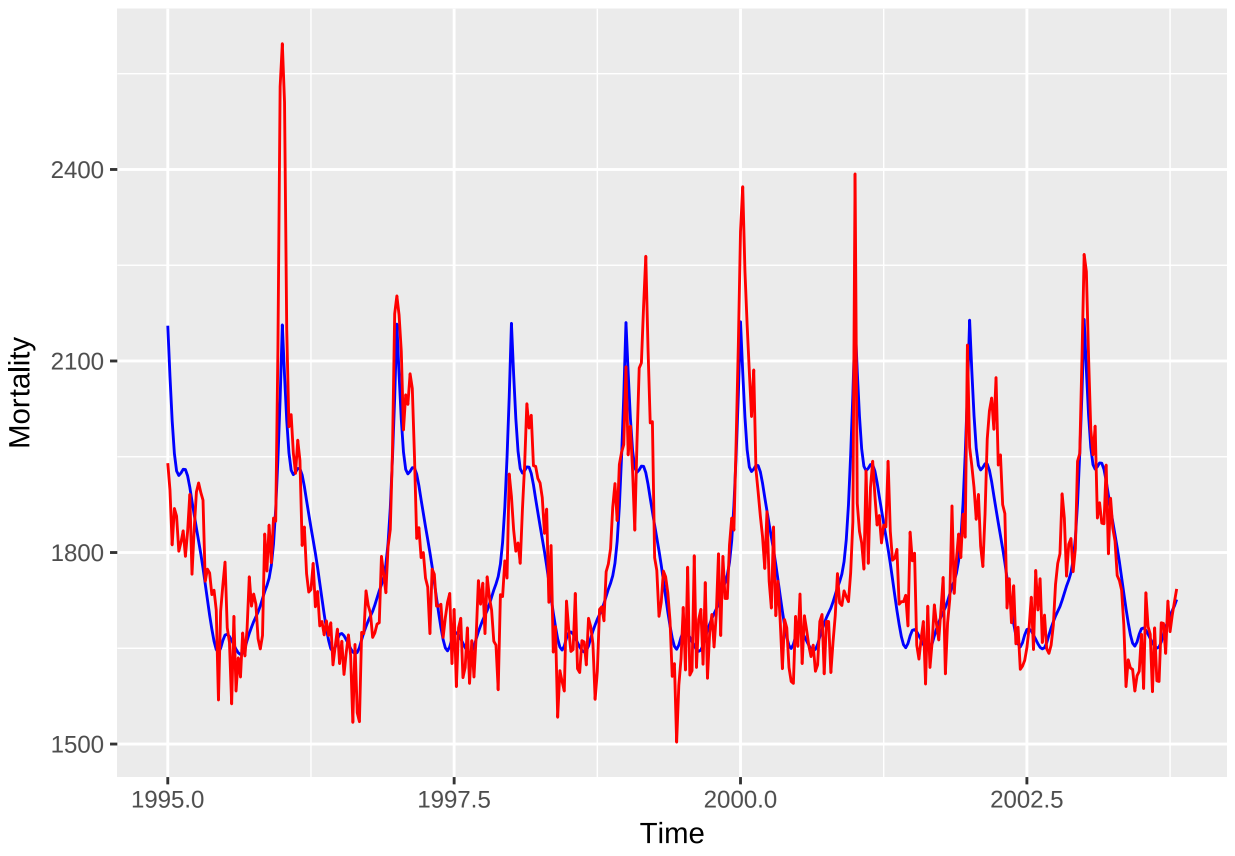
In the graph, We see the weekly data results of mortality and influenza in Sweden between the years 1995 and 2004. It is observed that there is a correlation between the influenza and the rising mortality rates until 2000, but after 2000 we do not see such a clear correlation.

Part 2 and 3

Now when we look at the comparative chart for the predicted and observed data, we see that it catches the trend in general. But we can not say that it is successful for high values. We can come to the conclusion that we have a general pattern here. When we examine the summary, we can see that the spline function of the week gives more effect on the result and is the significant comparing to p-values. The variable year has very high p value which is not significant.

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

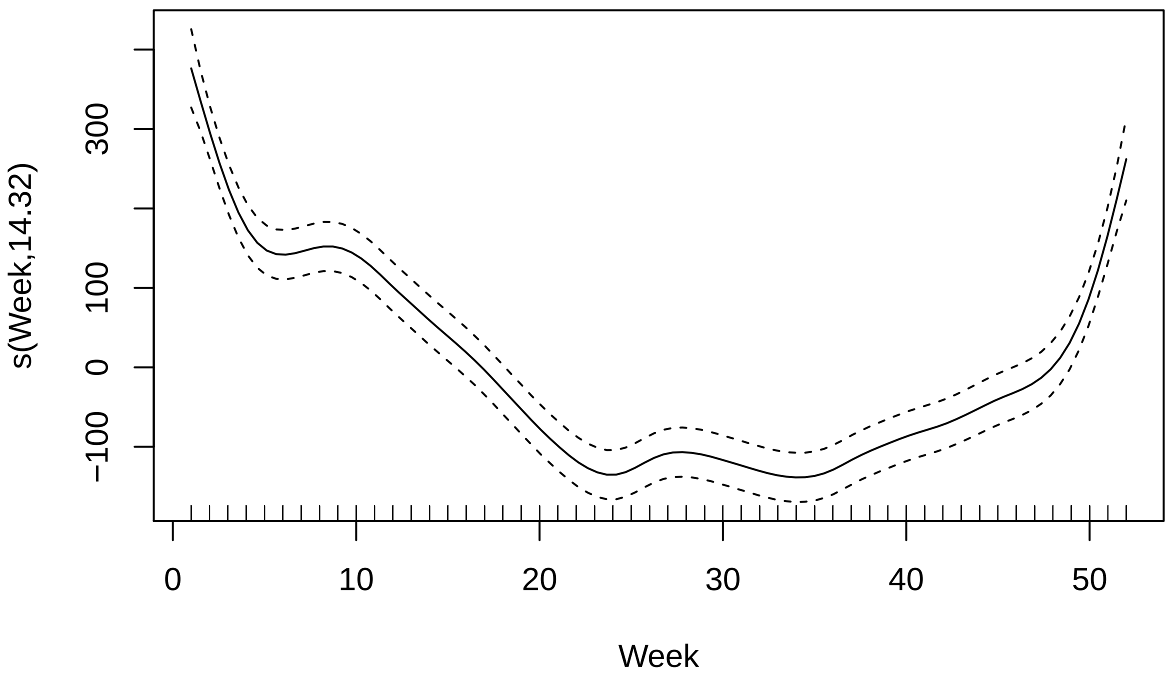
```
fit <- gam(Mortality ~ Year + s(Week,k=52),data=data,family = gaussian(),method = "GCV.Cp")
fittedMortality <- data
fittedMortality$Mortality <- fit$fitted.values
ggplot() + geom_line(data = fittedMortality, aes(Time, Mortality), color="blue") +
  geom_line(data = data, aes(Time, Mortality), color="red") +
   xlab("Time") + ylab("Mortality")
```



```r
summary(fit)
```

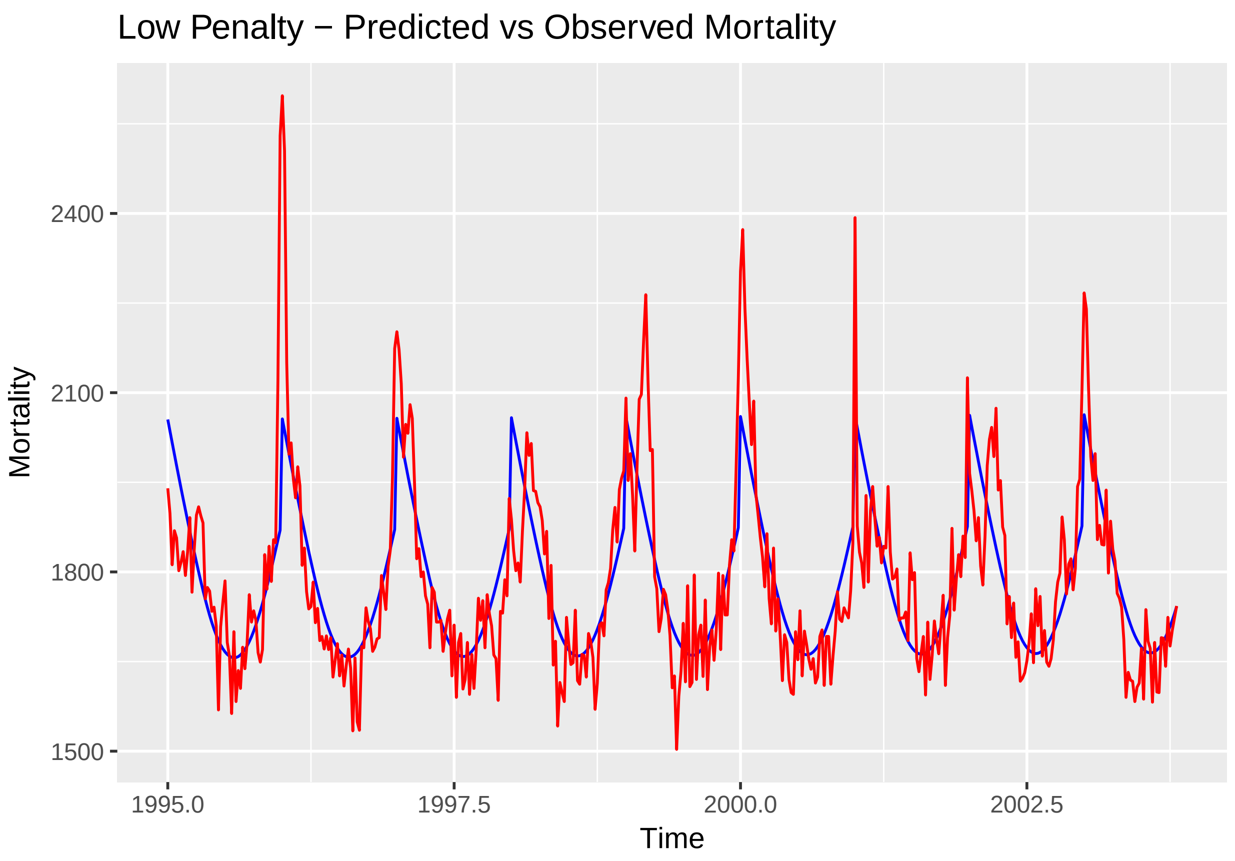
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = 52)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -680.598   3367.760  -0.202    0.840
## Year           1.233      1.685   0.732    0.465
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Week) 14.32  17.87 53.86  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 52/53
## R-sq.(adj) =  0.677   Deviance explained = 68.8%
## GCV = 8708.6  Scale est. = 8398.9    n = 459
```

```r
# Plot the spline component
plot(fit)
```

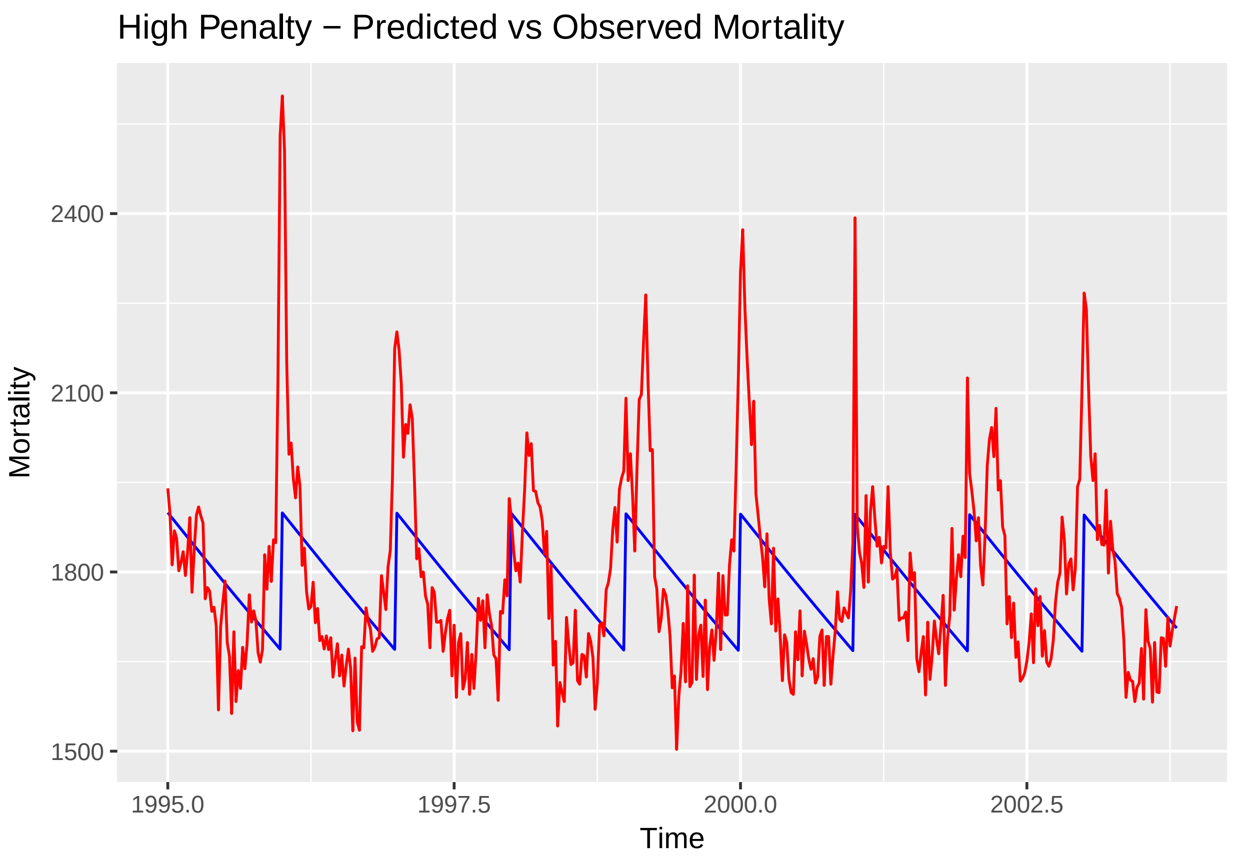


Part 4

```
low_penalty <- gam(Mortality ~ Year + s(Week,k=52,sp=0.1),data=data,
                   family = gaussian(),method = "GCV.Cp")
fittedMortality <- data
fittedMortality$Mortality <- low_penalty$fitted.values
ggplot() + geom_line(data = fittedMortality, aes(Time, Mortality), color="blue") +
  geom_line(data = data, aes(Time, Mortality), color="red") +
   xlab("Time") + ylab("Mortality") +
  ggtitle("Low Penalty - Predicted vs Observed Mortality")
```



```
high_penalty <- gam(Mortality ~ Year + s(Week,k=52,sp=100),data=data,
                    family = gaussian(),method = "GCV.Cp")
fittedMortality <- data
fittedMortality$Mortality <- high_penalty$fitted.values
ggplot() + geom_line(data = fittedMortality, aes(Time, Mortality), color="blue") +
  geom_line(data = data, aes(Time, Mortality), color="red") +
   xlab("Time") + ylab("Mortality")+
  ggtitle("High Penalty - Predicted vs Observed Mortality")
```



```
# Compare degrees of freedom
low_penalty
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = 52, sp = 0.1)
##
## Estimated degrees of freedom:
## 2.64  total = 4.64
##
## GCV score: 10127.58
```

```
high_penalty
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = 52, sp = 100)
##
## Estimated degrees of freedom:
## 1.01  total = 3.01
##
## GCV score: 21642.73
```

We see that the low penalty value catches the trend better. As we increase the penalty value, we see that the ups and downs have sharpened. This causes the model to express the trend more generally.

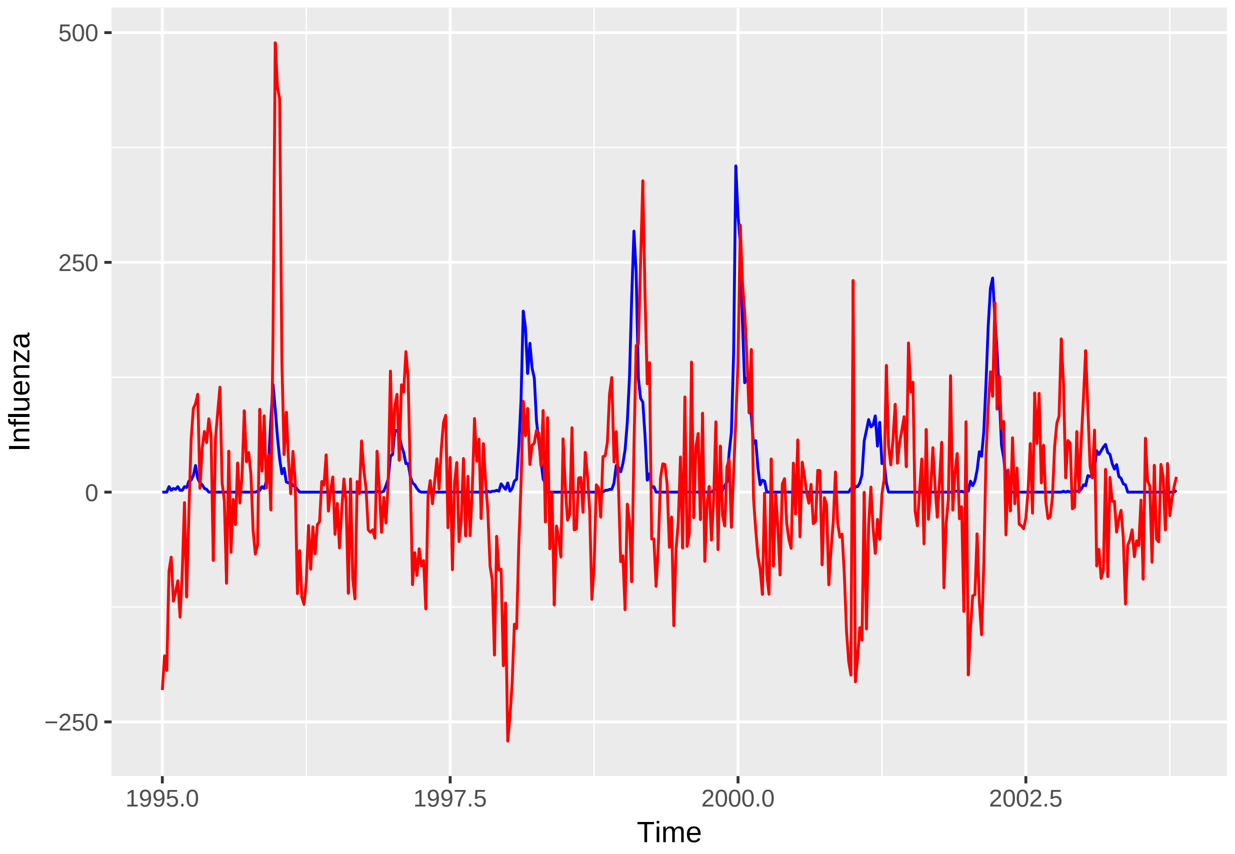
What is the relationship of the penalty factor to the degrees of freedom?

When we look at the low and high penalty printed results, we can say that low penalty has higher degrees of freedom than high penalty. So high penalty has less freedom to move.

Part 5

When we compare the residuals (red) and the influenza (blue) values, we mostly see that residuals catches peak values and correlates.

```r
fit <- gam(Mortality ~ Year + s(Week,k=52),data=data,
           family = gaussian(),method = "GCV.Cp")
customData <- data.frame(Time = data$Time, Residuals = fit$residuals)
ggplot() + geom_line(data = data, aes(Time, Influenza), color="blue") +
  geom_line(data = customData, aes(Time, Residuals), color="red") +
   xlab("Time") + ylab("Influenza")
```

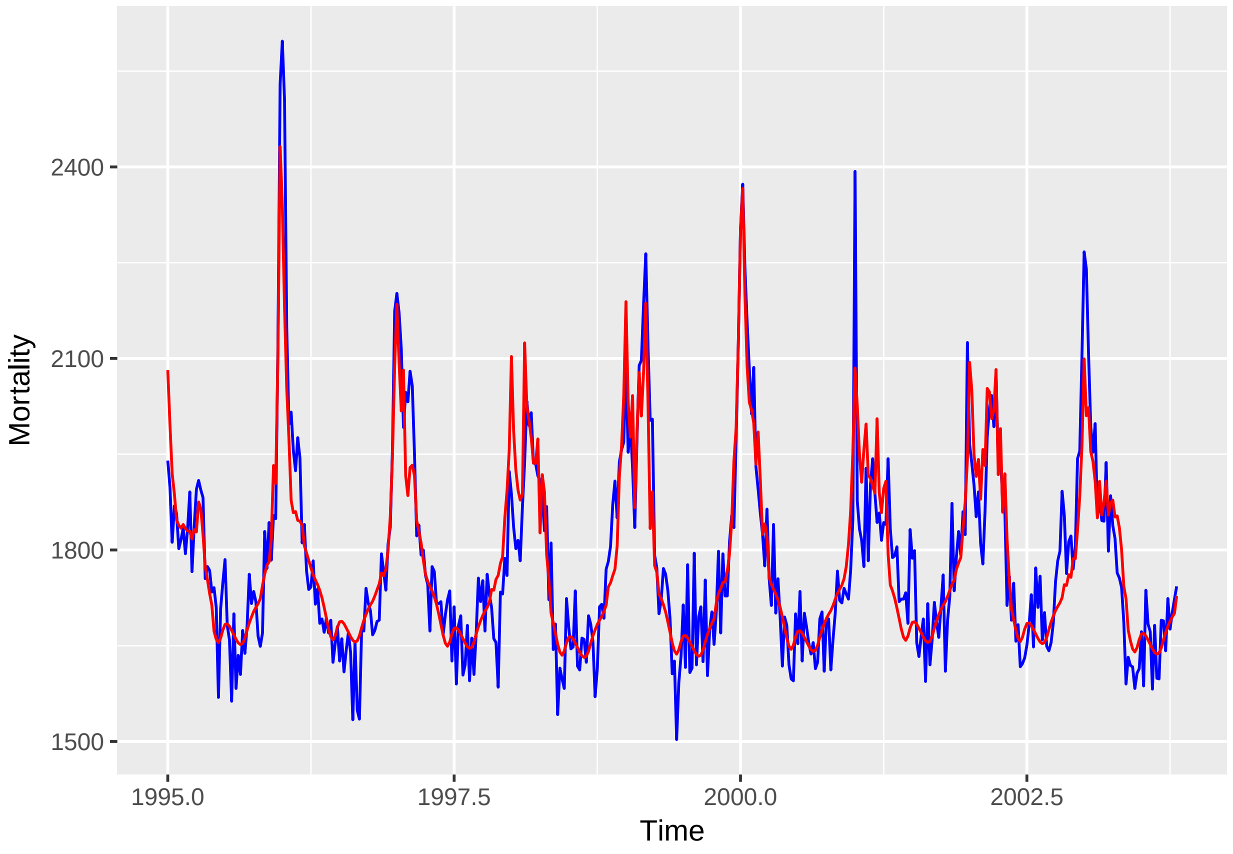


Part 6

Here we include the Influenza into model and the rate of catching up trend of the model has increased considerably. The model captures both trend and peak values. So the final model is better than previous ones.

By specifying k for spline component Influenza, we get less error result for prediction. Also, we plotted spline components.

```
kForYear <- length(unique(data$Year))
fit <- gam(Mortality ~ s(Year,k=kForYear) + s(Week,k=52) + s(Influenza,k=52),
           data=data,family = gaussian(),method = "GCV.Cp")
customData <- data.frame(Time = data$Time, FittedValues = fit$fitted.values)
ggplot() + geom_line(data = data, aes(Time, Mortality), color="blue") +
  geom_line(data = customData, aes(Time, FittedValues), color="red") +
   xlab("Time") + ylab("Mortality")
```

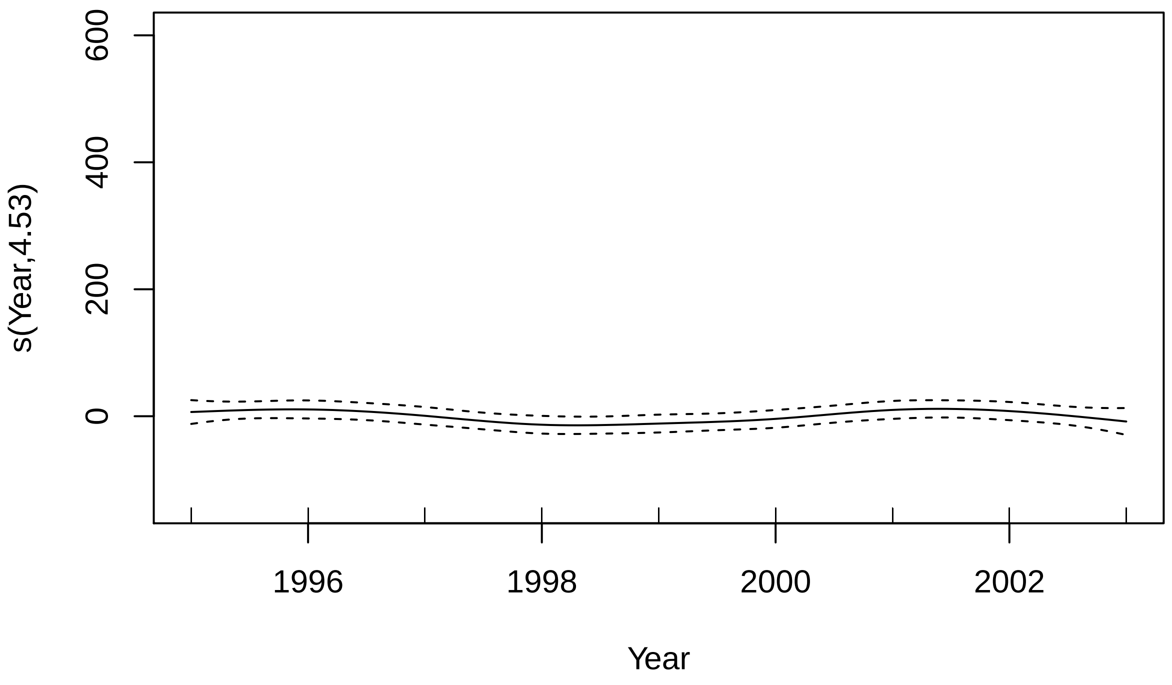


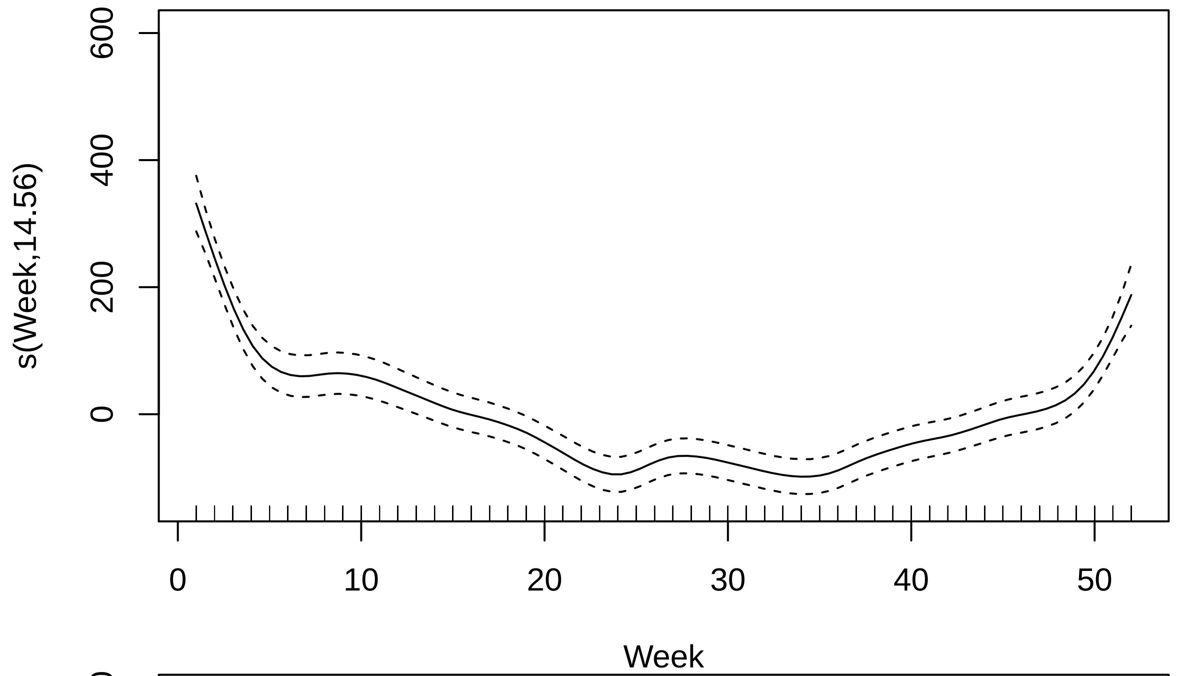
```
summary(fit)
```

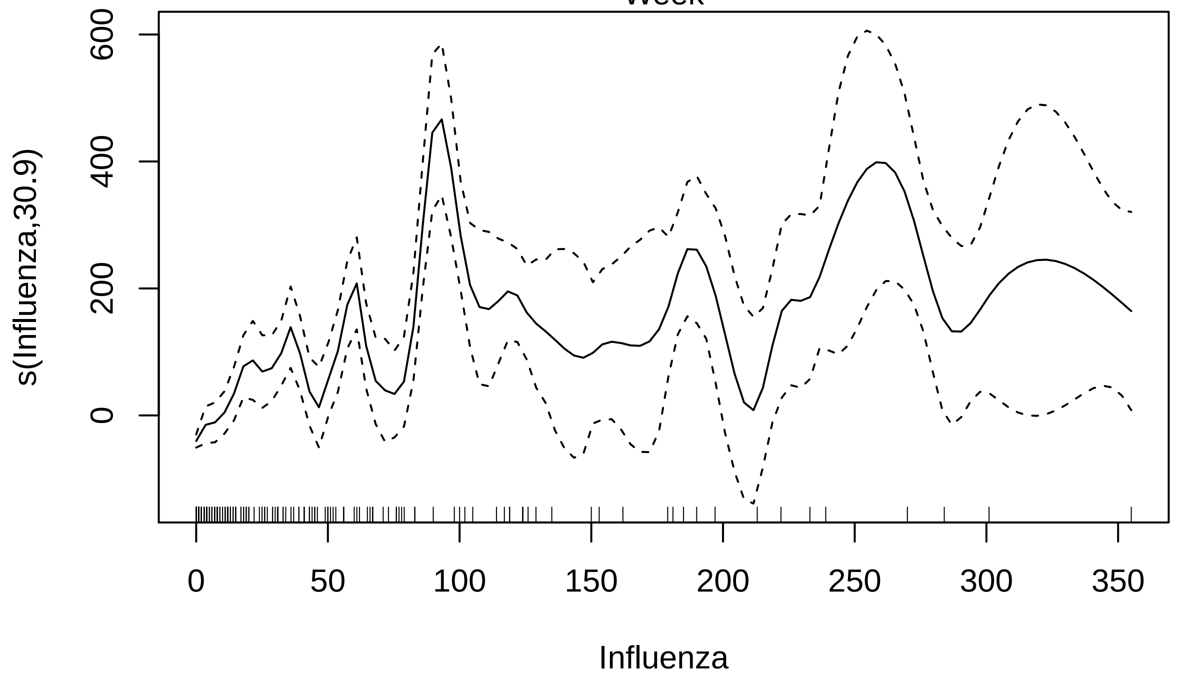
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ s(Year, k = kForYear) + s(Week, k = 52) + s(Influenza,
##     k = 52)
##
```

```
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1783.77       3.45     517   <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)       4.528  5.521  1.378   0.226
## s(Week)      14.560 18.126 20.651  <2e-16 ***
## s(Influenza) 30.903 36.301  6.981  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.79   Deviance explained = 81.3%
## GCV = 6147.7  Scale est. = 5464.8    n = 459
```

```
# Plot the spline component
plot(fit)
```







When visually looking at the graph, it seems the new prediction model outperforms the old one in peaks of the mortality rate and apart from the peaks, the two prediction models seem to follow a similar pattern.

Assignment 2

Part 1

```
set.seed(12345)
data0=read.csv2("data.csv", sep = ";",fileEncoding = "latin1")
```

```r
data=data0
confIndex <- which(colnames(data) == "Conference")
data$Conference=as.factor(data0$Conference)
n <- nrow(data)
rownames(data)=1:nrow(data)
index <- sample(1:n,size = floor(n*(7/10)))
train <- data[index,]
test <- data[-index,]

# Train X and Y
train_x=t(train[,-confIndex])
train_y=train[[confIndex]]

# Test X and Y
test_x=t(test[,-confIndex])
test_y=test[[confIndex]]

library(pamr)
mydata=list(x=train_x,y=train_y,geneid=as.character(1:nrow(train_x)),
            genenames=rownames(train_x))
model_pamr=pamr.train(mydata,threshold = seq(0,4,0.1))
```

```
## 1234567891011121314151617181920212223242526272829303132333435363738394041
```

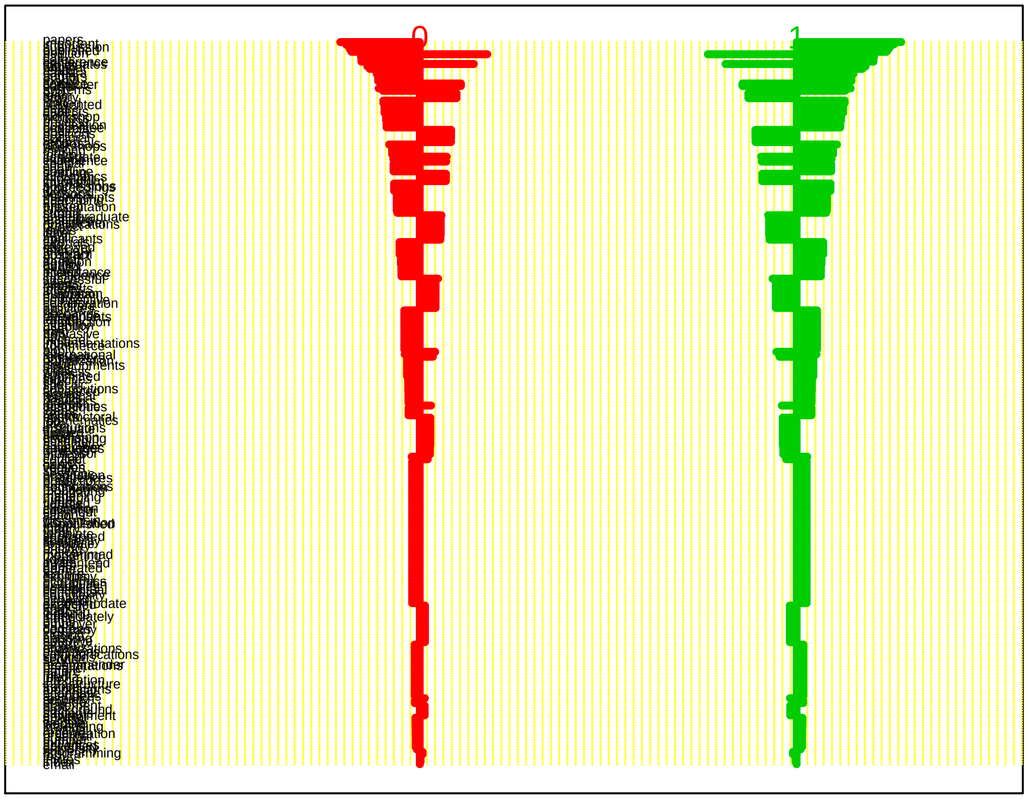
```r
cvmodel_pamr=pamr.cv(model_pamr,mydata)
```

```
## 12Fold 1 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 2 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 3 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 4 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 5 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 6 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 7 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 8 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 9 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 10 :1234567891011121314151617181920212223242526272829303132333435363738394041
```

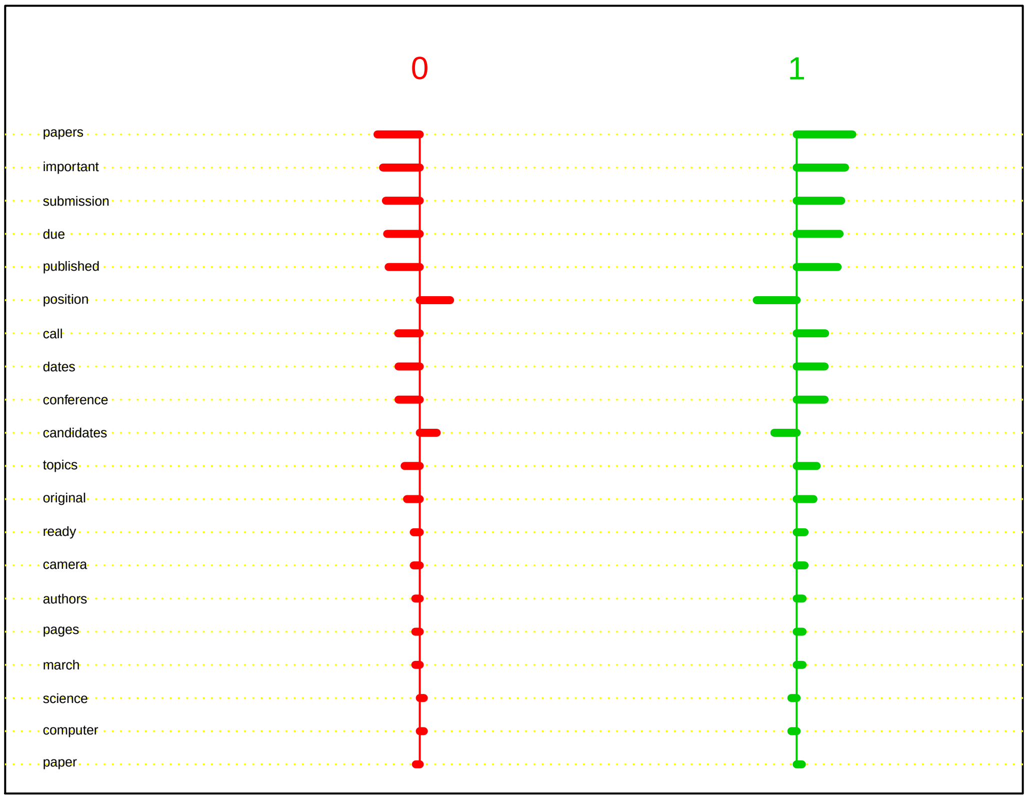
```r
print(cvmodel_pamr)
```

```
## Call:
## pamr.cv(fit = model_pamr, data = mydata)
##    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
```

```
# PLOT
pamr.plotcen(model_pamr, mydata, threshold=1)
```



```
pamr.plotcen(model_pamr, mydata, threshold=2.5)
```



```
a=pamr.listgenes(model_pamr,mydata,threshold=2.5)
```

```
##       id   0-score 1-score
##  [1,] 3036 -0.2245 0.2954
##  [2,] 2049 -0.195  0.2566
##  [3,] 4060 -0.1798 0.2366
##  [4,] 1262 -0.1732 0.2279
##  [5,] 3364 -0.1654 0.2176
##  [6,] 3187 0.1611  -0.2119
##  [7,] 596  -0.1147 0.151
##  [8,] 869  -0.1129 0.1485
##  [9,] 1045 -0.1129 0.1485
## [10,] 607  0.0899  -0.1182
## [11,] 4282 -0.0806 0.1061
## [12,] 2990 -0.0677 0.0891
## [13,] 599  -0.0319 0.042
## [14,] 3433 -0.0319 0.042
## [15,] 389  -0.0238 0.0314
## [16,] 2588 -0.0238 0.0314
## [17,] 3022 -0.0238 0.0314
## [18,] 850  0.0216  -0.0284
## [19,] 3725 0.0216  -0.0284
## [20,] 3035 -0.0208 0.0274
```

```
cat( paste( colnames(data)[as.numeric(a[,1])], collapse='\n' ) )
```

```
## papers
## important
## submission
## due
## published
## position
## call
## conference
## dates
## candidates
## topics
## original
## camera
## ready
## authors
## march
## pages
## computer
## science
## paper
```

```
cvmodel_pamr=pamr.cv(model_pamr,mydata)
```

```
## 12Fold 1 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 2 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 3 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 4 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 5 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 6 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 7 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 8 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 9 :1234567891011121314151617181920212223242526272829303132333435363738394041
## Fold 10 :1234567891011121314151617181920212223242526272829303132333435363738394041
```

```
print(cvmodel_pamr)
```

```
## Call:
## pamr.cv(fit = model_pamr, data = mydata)
##    threshold nonzero errors
## 1  0.0       3428    9
## 2  0.1       3409    8
## 3  0.2       3114    8
## 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    5
## 14 1.3        231    5
## 15 1.4        170    5
## 16 1.5        138    5
## 17 1.6        129    5
## 18 1.7         98    5
## 19 1.8         88    6
## 20 1.9         71    6
## 21 2.0         62    7
## 22 2.1         47    7
## 23 2.2         43    6
## 24 2.3         36    6
## 25 2.4         30    5
## 26 2.5         20    5
## 27 2.6         20    6
## 28 2.7         14    6
## 29 2.8         12    7
## 30 2.9         12    7
## 31 3.0         12    7
## 32 3.1         11    7
## 33 3.2          9    9
## 34 3.3          9    9
## 35 3.4          6    11
## 36 3.5          6    11
## 37 3.6          6    12
## 38 3.7          6    16
## 39 3.8          4    19
## 40 3.9          2    19
## 41 4.0          1    20
```

```
# pamr.plotcv(cvmodel_pamr)
```

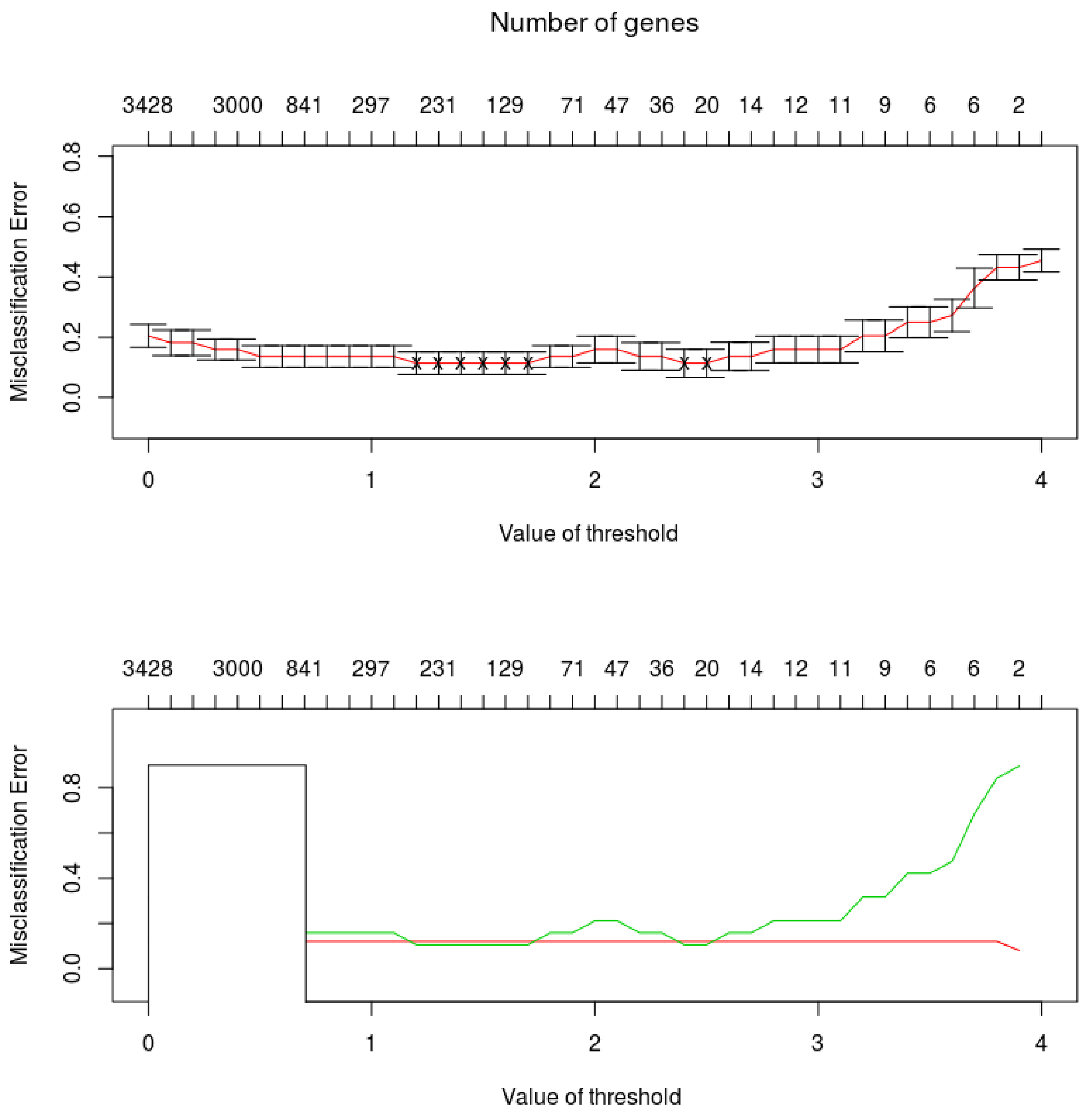


Figure 1: pamr.plotcv(cvmodel_pamr)

```
minimumErrorIndex <- which.min(cvmodel_pamr$error)
bestThreshold <- cvmodel_pamr$threshold[minimumErrorIndex]
numberOfSelectedFeatures_pamr <- cvmodel_pamr$size[minimumErrorIndex]

tenMostFeatures <- pamr.listgenes(model_pamr, mydata, threshold = bestThreshold)[1:10,]
```

```
##        id   0-score 1-score
##   [1,] 3036 -0.3953 0.5202
##   [2,] 2049 -0.3659 0.4814
##   [3,] 4060 -0.3507 0.4614
##   [4,] 1262 -0.344  0.4527
##   [5,] 3364 -0.3363 0.4424
##   [6,] 3187 0.3319  -0.4367
##   [7,] 596  -0.2856 0.3758
##   [8,] 869  -0.2837 0.3733
##   [9,] 1045 -0.2837 0.3733
##  [10,] 607  0.2607  -0.3431
##  [11,] 4282 -0.2515 0.3309
##  [12,] 2990 -0.2385 0.3139
##  [13,] 599  -0.2028 0.2668
##  [14,] 3433 -0.2028 0.2668
##  [15,] 389  -0.1947 0.2562
##  [16,] 2588 -0.1947 0.2562
##  [17,] 3022 -0.1947 0.2562
##  [18,] 850  0.1924  -0.2532
##  [19,] 3725 0.1924  -0.2532
##  [20,] 3035 -0.1916 0.2522
##  [21,] 4129 -0.169  0.2224
##  [22,] 3125 0.169   -0.2224
##  [23,] 4177 0.1687  -0.222
##  [24,] 3671 0.1687  -0.222
##  [25,] 2974 -0.1673 0.2201
##  [26,] 2463 -0.1673 0.2201
##  [27,] 329  -0.1611 0.212
##  [28,] 681  -0.1611 0.212
##  [29,] 1891 -0.1611 0.212
##  [30,] 3243 -0.1611 0.212
##  [31,] 283  -0.1531 0.2015
##  [32,] 4628 -0.1531 0.2015
##  [33,] 3286 -0.1531 0.2015
##  [34,] 3274 -0.15   0.1973
##  [35,] 810  -0.15   0.1973
##  [36,] 2889 -0.15   0.1973
##  [37,] 1233 0.1404  -0.1847
##  [38,] 3188 0.1404  -0.1847
##  [39,] 3191 0.1404  -0.1847
##  [40,] 3312 0.1404  -0.1847
##  [41,] 3891 0.1396  -0.1837
##  [42,] 3458 0.1396  -0.1837
##  [43,] 3324 -0.1363 0.1793
##  [44,] 1643 -0.1208 0.159
##  [45,] 2561 -0.1208 0.159
##  [46,] 3090 -0.1208 0.159
##  [47,] 4629 -0.1208 0.159
##  [48,] 606  0.1173  -0.1543
##  [49,] 2058 -0.1144 0.1505
##  [50,] 1501 0.1144  -0.1505
##  [51,] 3952 -0.1131 0.1489
##  [52,] 680  -0.1131 0.1489
##  [53,] 3836 -0.1131 0.1489
##  [54,] 1061 -0.113  0.1487
##  [55,] 1007 0.1127  -0.1482
##  [56,] 1477 0.1127  -0.1482
##  [57,] 2103 0.1127  -0.1482
##  [58,] 3992 0.1127  -0.1482
##  [59,] 2295 -0.1103 0.1451
##  [60,] 4061 -0.1103 0.1451
##  [61,] 2305 -0.1101 0.1449
##  [62,] 3285 -0.1101 0.1449
##  [63,] 92   -0.0963 0.1267
##  [64,] 1127 -0.0963 0.1267
##  [65,] 2583 -0.0963 0.1267
##  [66,] 3323 -0.0963 0.1267
##  [67,] 4500 -0.0963 0.1267
##  [68,] 1698 -0.0963 0.1267
##  [69,] 3241 -0.0963 0.1267
##  [70,] 4364 -0.0963 0.1267
##  [71,] 4062 -0.0928 0.1221
##  [72,] 4039 0.0889  -0.1169
##  [73,] 740  0.0853  -0.1122
##  [74,] 2438 0.0853  -0.1122
##  [75,] 2442 0.0853  -0.1122
##  [76,] 3311 0.0853  -0.1122
##  [77,] 3383 0.0853  -0.1122
##  [78,] 3559 0.0853  -0.1122
##  [79,] 4176 0.0853  -0.1122
##  [80,] 4402 0.0853  -0.1122
##  [81,] 267  0.0833  -0.1095
##  [82,] 2553 0.0833  -0.1095
##  [83,] 63   -0.0812 0.1068
##  [84,] 1563 -0.0812 0.1068
##  [85,] 1594 -0.0812 0.1068
##  [86,] 3589 -0.0812 0.1068
##  [87,] 3882 -0.0812 0.1068
##  [88,] 4365 -0.0812 0.1068
##  [89,] 3301 -0.0743 0.0978
##  [90,] 1636 -0.0741 0.0975
##  [91,] 1072 -0.0741 0.0975
##  [92,] 386  -0.0741 0.0975
##  [93,] 2198 -0.0718 0.0944
##  [94,] 3021 -0.0718 0.0944
##  [95,] 3386 -0.0718 0.0944
##  [96,] 76   -0.0715 0.094
##  [97,] 2150 -0.0715 0.094
##  [98,] 4075 0.071   -0.0935
##  [99,] 107  0.0579  -0.0762
## [100,] 336  0.0579  -0.0762
## [101,] 776  0.0579  -0.0762
## [102,] 831  0.0579  -0.0762
## [103,] 1088 0.0579  -0.0762
## [104,] 1450 0.0579  -0.0762
## [105,] 1456 0.0579  -0.0762
## [106,] 1542 0.0579  -0.0762
## [107,] 2170 0.0579  -0.0762
## [108,] 2613 0.0579  -0.0762
## [109,] 2837 0.0579  -0.0762
## [110,] 4529 0.0579  -0.0762
## [111,] 363  -0.056  0.0737
## [112,] 879  -0.056  0.0737
## [113,] 2433 -0.056  0.0737
## [114,] 3051 -0.056  0.0737
## [115,] 3514 -0.056  0.0737
## [116,] 3711 -0.056  0.0737
## [117,] 4449 -0.056  0.0737
## [118,] 501  -0.056  0.0737
## [119,] 803  -0.056  0.0737
## [120,] 2046 -0.056  0.0737
## [121,] 2082 -0.056  0.0737
## [122,] 2690 -0.056  0.0737
## [123,] 2877 -0.056  0.0737
## [124,] 3118 -0.056  0.0737
## [125,] 4342 -0.056  0.0737
## [126,] 4451 -0.056  0.0737
## [127,] 4452 -0.056  0.0737
## [128,] 272  0.0556  -0.0732
## [129,] 2175 -0.0539 0.071
## [130,] 3515 0.0433  -0.057
## [131,] 172  -0.0415 0.0546
## [132,] 1149 -0.0415 0.0546
## [133,] 2219 -0.0415 0.0546
## [134,] 2964 -0.0415 0.0546
## [135,] 2984 -0.0415 0.0546
## [136,] 2887 -0.0415 0.0546
## [137,] 4605 -0.0415 0.0546
## [138,] 4064 -0.0412 0.0542
## [139,] 3800 -0.0369 0.0486
## [140,] 134  -0.0353 0.0465
## [141,] 919  -0.0353 0.0465
## [142,] 3957 -0.0353 0.0465
## [143,] 4268 -0.0353 0.0465
## [144,] 4281 -0.0353 0.0465
## [145,] 2220 -0.0342 0.045
## [146,] 2847 -0.0342 0.045
## [147,] 3582 -0.0342 0.045
## [148,] 4181 -0.0342 0.045
## [149,] 2167 -0.0335 0.0441
## [150,] 67   0.0335  -0.0441
## [151,] 2005 -0.0334 0.044
## [152,] 4185 -0.0334 0.044
## [153,] 3588 -0.0334 0.044
## [154,] 3794 -0.0334 0.044
## [155,] 579  0.0301  -0.0396
## [156,] 1147 0.0301  -0.0396
## [157,] 1524 0.0301  -0.0396
## [158,] 1591 0.0301  -0.0396
## [159,] 1702 0.0301  -0.0396
## [160,] 1797 0.0301  -0.0396
## [161,] 2141 0.0301  -0.0396
## [162,] 2251 0.0301  -0.0396
## [163,] 2278 0.0301  -0.0396
## [164,] 2619 0.0301  -0.0396
## [165,] 3194 0.0301  -0.0396
## [166,] 340  0.0301  -0.0396
## [167,] 2894 0.0287  -0.0378
## [168,] 1144 0.028   -0.0368
## [169,] 2392 0.028   -0.0368
## [170,] 3295 0.028   -0.0368
## [171,] 2713 -0.0168 0.0221
## [172,] 899  0.0163  -0.0215
## [173,] 1859 -0.0143 0.0188
## [174,] 3764 -0.0143 0.0188
## [175,] 104  -0.0143 0.0188
## [176,] 940  -0.0143 0.0188
## [177,] 967  -0.0143 0.0188
## [178,] 1343 -0.0143 0.0188
## [179,] 1587 -0.0143 0.0188
## [180,] 1861 -0.0143 0.0188
## [181,] 1965 -0.0143 0.0188
## [182,] 2574 -0.0143 0.0188
## [183,] 2623 -0.0143 0.0188
## [184,] 2754 -0.0143 0.0188
## [185,] 2757 -0.0143 0.0188
## [186,] 2839 -0.0143 0.0188
## [187,] 2890 -0.0143 0.0188
## [188,] 3169 -0.0143 0.0188
## [189,] 3224 -0.0143 0.0188
## [190,] 3231 -0.0143 0.0188
## [191,] 3289 -0.0143 0.0188
## [192,] 3802 -0.0143 0.0188
## [193,] 3943 -0.0143 0.0188
## [194,] 4490 -0.0143 0.0188
## [195,] 4499 -0.0143 0.0188
## [196,] 84   -0.0143 0.0188
## [197,] 196  -0.0143 0.0188
## [198,] 455  -0.0143 0.0188
## [199,] 837  -0.0143 0.0188
## [200,] 856  -0.0143 0.0188
## [201,] 857  -0.0143 0.0188
## [202,] 920  -0.0143 0.0188
## [203,] 1062 -0.0143 0.0188
## [204,] 1214 -0.0143 0.0188
## [205,] 1291 -0.0143 0.0188
## [206,] 1292 -0.0143 0.0188
## [207,] 1490 -0.0143 0.0188
## [208,] 1560 -0.0143 0.0188
## [209,] 1721 -0.0143 0.0188
## [210,] 1745 -0.0143 0.0188
## [211,] 1818 -0.0143 0.0188
## [212,] 1833 -0.0143 0.0188
## [213,] 2197 -0.0143 0.0188
## [214,] 2359 -0.0143 0.0188
## [215,] 2598 -0.0143 0.0188
## [216,] 2746 -0.0143 0.0188
## [217,] 2888 -0.0143 0.0188
## [218,] 3259 -0.0143 0.0188
## [219,] 3361 -0.0143 0.0188
## [220,] 3570 -0.0143 0.0188
## [221,] 3703 -0.0143 0.0188
## [222,] 3948 -0.0143 0.0188
## [223,] 3966 -0.0143 0.0188
## [224,] 4202 -0.0143 0.0188
## [225,] 4212 -0.0143 0.0188
## [226,] 4236 -0.0143 0.0188
## [227,] 4363 -0.0143 0.0188
## [228,] 4435 -0.0143 0.0188
## [229,] 4526 -0.0143 0.0188
## [230,] 4606 -0.0143 0.0188
## [231,] 4664 -0.0143 0.0188
## [232,] 1395 0.0015  -0.002
## [233,] 1498 0.0015  -0.002
## [234,] 115  0.0015  -0.0019
## [235,] 274  0.0015  -0.0019
## [236,] 745  0.0015  -0.0019
## [237,] 756  0.0015  -0.0019
## [238,] 821  0.0015  -0.0019
## [239,] 1089 0.0015  -0.0019
## [240,] 1230 0.0015  -0.0019
## [241,] 1372 0.0015  -0.0019
## [242,] 1424 0.0015  -0.0019
## [243,] 1913 0.0015  -0.0019
## [244,] 2040 0.0015  -0.0019
## [245,] 2510 0.0015  -0.0019
## [246,] 2945 0.0015  -0.0019
## [247,] 3991 0.0015  -0.0019
## [248,] 362  -9e-04  0.0012
## [249,] 1290 -9e-04  0.0012
## [250,] 1368 -9e-04  0.0012
## [251,] 1653 -9e-04  0.0012
## [252,] 1662 -9e-04  0.0012
## [253,] 2109 -9e-04  0.0012
## [254,] 2147 -9e-04  0.0012
## [255,] 2303 -9e-04  0.0012
## [256,] 2643 -9e-04  0.0012
## [257,] 2787 -9e-04  0.0012
## [258,] 2819 -9e-04  0.0012
## [259,] 3040 -9e-04  0.0012
## [260,] 3242 -9e-04  0.0012
## [261,] 3454 -9e-04  0.0012
## [262,] 3732 -9e-04  0.0012
## [263,] 3799 -9e-04  0.0012
## [264,] 3923 -9e-04  0.0012
## [265,] 815  -9e-04  0.0012
## [266,] 1177 -9e-04  0.0012
## [267,] 2981 -9e-04  0.0012
## [268,] 3592 -9e-04  0.0012
## [269,] 3898 -9e-04  0.0012
## [270,] 3558 9e-04   -0.0012
## [271,] 3341 -2e-04  2e-04
## [272,] 3755 -2e-04  2e-04
```

```r
tenMostFeaturesIds <- as.numeric(tenMostFeatures[,1])
mostContributingFeatures <- colnames(data)[tenMostFeaturesIds]
```

```r
print(as.data.frame(mostContributingFeatures))
```

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

```r
pred_test <- pamr.predict(model_pamr,newx = test_x,threshold = bestThreshold,
                          type = "class")
# Function that builds the confusion matrix
confusion_matrix <- function(predicted, actual) {
    df <- data.frame("Pred" = predicted, "Real" = actual)
    confusion_mat <- with(df, table(Real, Pred))
    totals <- matrix(c(sum(confusion_mat[1, ]), sum(confusion_mat[2, ])), nrow = 2,
                     ncol = 1)
    confusion_mat <- cbind(confusion_mat, totals)
    colnames(confusion_mat) <- c("bad", "good", "Total")
    rownames(confusion_mat) <- c("bad", "good")
    return(confusion_mat)
}

cm <- confusion_matrix(pred_test,test_y)
error_rate_pamr <- (cm[1,2] + cm[2,1])/(cm[2,3] + cm[1,3])
print(error_rate_pamr)
```

```
## [1] 0.1
```

Part 2

We will try Elastic net and SVM algorithms to approach the problem from different ways. We set the Elastic net algorithm with $\alpha = 0.5$ and result as a binomial. Cross validation is also used to get best model.

```r
library(glmnet)
cvmodel_glmnet <- cv.glmnet(x = as.matrix(train[,-confIndex]),y = train_y,
                              family = "binomial",alpha = 0.5)
minLambdaIndex <- which(cvmodel_glmnet$lambda ==cvmodel_glmnet$lambda.min)
numberOfSelectedFeatures_glmnet <- cvmodel_glmnet$nzero[minLambdaIndex]
pred_test <- predict(cvmodel_glmnet,newx = as.matrix(test[,-confIndex]),
                      s = "lambda.min",type = "class")

cm <- confusion_matrix(as.factor(pred_test),test_y)
error_rate_glmnet <- (cm[1,2] + cm[2,1])/(cm[2,3] + cm[1,3])
```

We also use the SVM algorithm with vanilladot kernels.

```r
library(kernlab)
model_kernlab <- ksvm(as.factor(Conference) ~ .,data = train,kernel = "vanilladot",scale=FALSE)
```

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

```r
pred <- predict(model_kernlab, test, type = "response")

cm <- confusion_matrix(pred,test_y)
error_rate_kernlab <- (cm[1,2] + cm[2,1])/(cm[2,3] + cm[1,3])
```

We calculated the error rates for all the algorithms we used. In this step we will be able to evaluate the results with a comparative table. When we look at the graph, we see that the result of Nearest Shrinken Centroid and Elastic net is at the same rate But the SVM algorithm seems to have an error rate less than 0.05 comparing to others. In this case, we can use SVM algorithm for this problem.

```r
df <- data.frame(NSC = error_rate_pamr,
                 Elasticnet = error_rate_glmnet,
                 SVM = error_rate_kernlab)
rownames(df) <- "Error Rate"

library(pander)
pander(df, type = 'grid')
```

	NSC	Elasticnet	SVM
Error Rate	0.1	0.1	0.05

Part 3

To investigate Benjamini–Hochberg method, we need to implement some steps. First we need to put all p-values in order. After that we assign ranks for each p-value. Thus, we are ready to calculate Benjamini-Hochberg method. We divide each rank by total number of tests. Finally we multiply the result by alpha value which is $\alpha = 0.05$

Benjamini–Hochberg method allows us to evaluate p-value by approaching it from another way. Thus, we can avoid type 1 errors, rejecting the true null hypothesis.

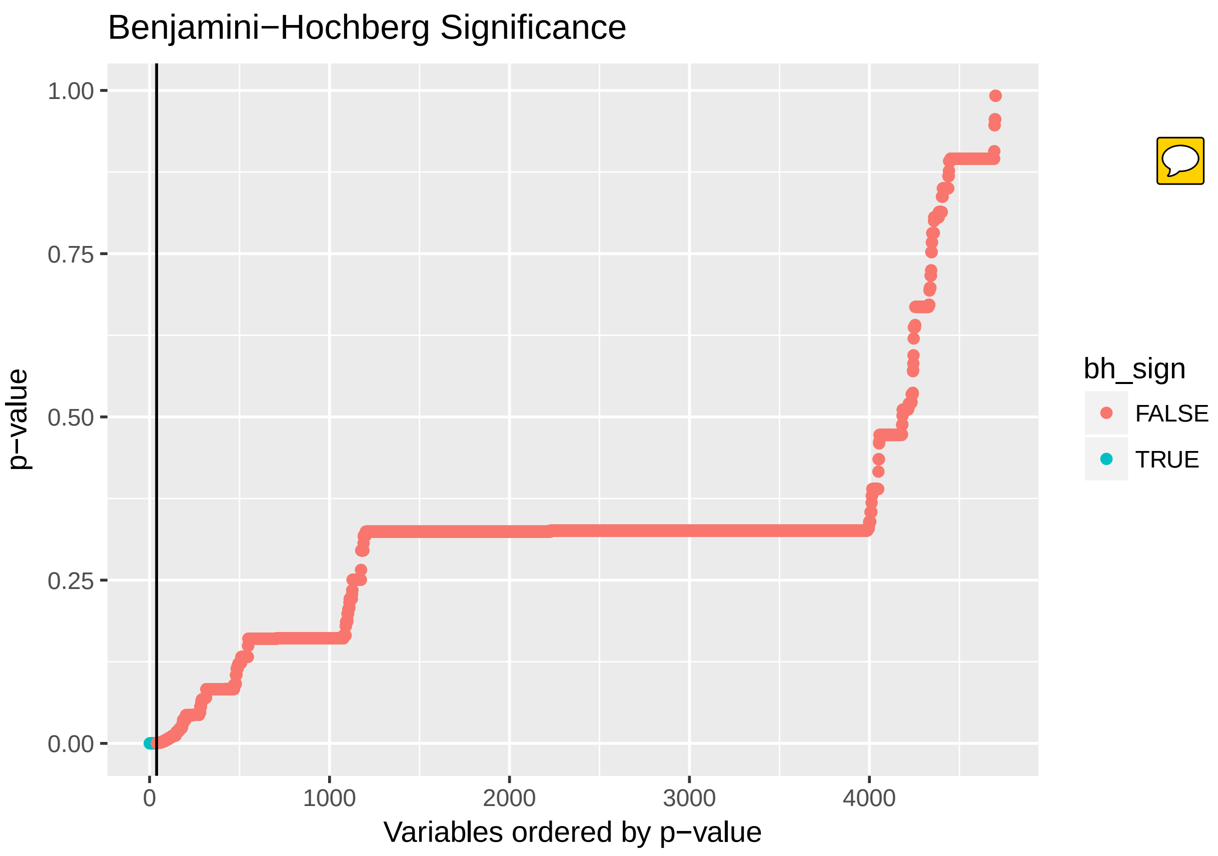
```r
p_value <- data.frame(iteration = numeric(), p_value = numeric())
maxIteration <- ncol(data)-1
```

```r
for(i in 1:maxIteration){
    p <-  t.test(data[,i] ~ data$Conference)$p.value
    p_value <- rbind(p_value,data.frame(i,p))
}

library(tidyverse)
p_value <- p_value %>%
    mutate(feature = colnames(data[1:4702])) %>%
    arrange(p) %>%
    mutate(rank = seq_along(.$i),
           BH = (rank/4702)*0.05,
           bh_sign = BH > p)

table(p_value$bh_sign)
```

```
##
## FALSE  TRUE
##  4663    39
```

According to result, we have 39 variables which are considered as significant by the Benjamini-Hochberg method.

```r
 ggplot(data = p_value) +
     geom_point(aes(x = rank, y = p, color = bh_sign)) +
     geom_vline(xintercept = 39) +
     labs(title = "Benjamini-Hochberg Significance", x = "Variables ordered by p-value", y = "p-value")
```



Appendix

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

data <- as.data.frame(read_xlsx("Influenza.xlsx"))
scaled <- data.frame(scale(data))
scaled$Time <- data$Time
ggplot() + geom_line(data = scaled, aes(Time, Mortality), color="red") +
  geom_line(data = scaled, aes(Time, Influenza), color="blue") +
   xlab("Time") + ylab("Scaled Mortality vs Influenza")

fit <- gam(Mortality ~ Year + s(Week,k=52),data=data,family = gaussian(),method = "GCV.Cp")
fittedMortality <- data
fittedMortality$Mortality <- fit$fitted.values
ggplot() + geom_line(data = fittedMortality, aes(Time, Mortality), color="blue") +
  geom_line(data = data, aes(Time, Mortality), color="red") +
   xlab("Time") + ylab("Mortality")

summary(fit)

# Plot the spline component
plot(fit)
low_penalty <- gam(Mortality ~ Year + s(Week,k=52,sp=0.1),data=data,
                   family = gaussian(),method = "GCV.Cp")
fittedMortality <- data
fittedMortality$Mortality <- low_penalty$fitted.values
ggplot() + geom_line(data = fittedMortality, aes(Time, Mortality), color="blue") +
  geom_line(data = data, aes(Time, Mortality), color="red") +
   xlab("Time") + ylab("Mortality") +
  ggtitle("Low Penalty - Predicted vs Observed Mortality")

high_penalty <- gam(Mortality ~ Year + s(Week,k=52,sp=100),data=data,
                    family = gaussian(),method = "GCV.Cp")
fittedMortality <- data
fittedMortality$Mortality <- high_penalty$fitted.values
ggplot() + geom_line(data = fittedMortality, aes(Time, Mortality), color="blue") +
  geom_line(data = data, aes(Time, Mortality), color="red") +
   xlab("Time") + ylab("Mortality")+
  ggtitle("High Penalty - Predicted vs Observed Mortality")

# Compare degrees of freedom
low_penalty
high_penalty
fit <- gam(Mortality ~ Year + s(Week,k=52),data=data,
           family = gaussian(),method = "GCV.Cp")
customData <- data.frame(Time = data$Time, Residuals = fit$residuals)
ggplot() + geom_line(data = data, aes(Time, Influenza), color="blue") +
  geom_line(data = customData, aes(Time, Residuals), color="red") +
   xlab("Time") + ylab("Influenza")
```

```r
kForYear <- length(unique(data$Year))
fit <- gam(Mortality ~ s(Year,k=kForYear) + s(Week,k=52) + s(Influenza,k=52),
           data=data,family = gaussian(),method = "GCV.Cp")
customData <- data.frame(Time = data$Time, FittedValues = fit$fitted.values)
ggplot() + geom_line(data = data, aes(Time, Mortality), color="blue") +
  geom_line(data = customData, aes(Time, FittedValues), color="red") +
   xlab("Time") + ylab("Mortality")

summary(fit)
# Plot the spline component
plot(fit)
set.seed(12345)
data0=read.csv2("data.csv", sep = ";",fileEncoding = "latin1")
data=data0
confIndex <- which(colnames(data) == "Conference")
data$Conference=as.factor(data0$Conference)
n <- nrow(data)
rownames(data)=1:nrow(data)
index <- sample(1:n,size = floor(n*(7/10)))
train <- data[index,]
test <- data[-index,]

# Train X and Y
train_x=t(train[,-confIndex])
train_y=train[[confIndex]]

# Test X and Y
test_x=t(test[,-confIndex])
test_y=test[[confIndex]]

library(pamr)
mydata=list(x=train_x,y=train_y,geneid=as.character(1:nrow(train_x)),
            genenames=rownames(train_x))
model_pamr=pamr.train(mydata,threshold = seq(0,4,0.1))
cvmodel_pamr=pamr.cv(model_pamr,mydata)
print(cvmodel_pamr)

# PLOT
pamr.plotcen(model_pamr, mydata, threshold=1)
pamr.plotcen(model_pamr, mydata, threshold=2.5)
a=pamr.listgenes(model_pamr,mydata,threshold=2.5)
cat( paste( colnames(data)[as.numeric(a[,1])], collapse='\n' ) )
cvmodel_pamr=pamr.cv(model_pamr,mydata)
print(cvmodel_pamr)

# pamr.plotcv(cvmodel_pamr)
minimumErrorIndex <- which.min(cvmodel_pamr$error)
bestThreshold <- cvmodel_pamr$threshold[minimumErrorIndex]
numberOfSelectedFeatures_pamr <- cvmodel_pamr$size[minimumErrorIndex]

tenMostFeatures <- pamr.listgenes(model_pamr, mydata, threshold = bestThreshold)[1:10,]
tenMostFeaturesIds <- as.numeric(tenMostFeatures[,1])
mostContributingFeatures <- colnames(data)[tenMostFeaturesIds]
```

```r
print(as.data.frame(mostContributingFeatures))
pred_test <- pamr.predict(model_pamr,newx = test_x,threshold = bestThreshold,
                          type = "class")
# Function that builds the confusion matrix
confusion_matrix <- function(predicted, actual) {
    df <- data.frame("Pred" = predicted, "Real" = actual)
    confusion_mat <- with(df, table(Real, Pred))
    totals <- matrix(c(sum(confusion_mat[1, ]), sum(confusion_mat[2, ])), nrow = 2,
                     ncol = 1)
    confusion_mat <- cbind(confusion_mat, totals)
    colnames(confusion_mat) <- c("bad", "good", "Total")
    rownames(confusion_mat) <- c("bad", "good")
    return(confusion_mat)
}

cm <- confusion_matrix(pred_test,test_y)
error_rate_pamr <- (cm[1,2] + cm[2,1])/(cm[2,3] + cm[1,3])
print(error_rate_pamr)
library(glmnet)
cvmodel_glmnet <- cv.glmnet(x = as.matrix(train[,-confIndex]),y = train_y,
                            family = "binomial",alpha = 0.5)
minLambdaIndex <- which(cvmodel_glmnet$lambda ==cvmodel_glmnet$lambda.min)
numberOfSelectedFeatures_glmnet <- cvmodel_glmnet$nzero[minLambdaIndex]
pred_test <- predict(cvmodel_glmnet,newx = as.matrix(test[,-confIndex]),
                     s = "lambda.min",type = "class")

cm <- confusion_matrix(as.factor(pred_test),test_y)
error_rate_glmnet <- (cm[1,2] + cm[2,1])/(cm[2,3] + cm[1,3])
library(kernlab)
model_kernlab <- ksvm(as.factor(Conference) ~ .,data = train,kernel = "vanilladot",scale=FALSE)
pred <- predict(model_kernlab, test, type = "response")

cm <- confusion_matrix(pred,test_y)
error_rate_kernlab <- (cm[1,2] + cm[2,1])/(cm[2,3] + cm[1,3])
df <- data.frame(NSC = error_rate_pamr,
                 Elasticnet = error_rate_glmnet,
                 SVM = error_rate_kernlab)
rownames(df) <- "Error Rate"

library(pander)
pander(df, type = 'grid')


p_value <- data.frame(iteration = numeric(), p_value = numeric())
maxIteration <- ncol(data)-1
for(i in 1:maxIteration){
    p <-  t.test(data[,i] ~ data$Conference)$p.value
    p_value <- rbind(p_value,data.frame(i,p))
}

library(tidyverse)
p_value <- p_value %>%
    mutate(feature = colnames(data[1:4702])) %>%
    arrange(p) %>%
    mutate(rank = seq_along(.$i),
           BH = (rank/4702)*0.05,
           bh_sign = BH > p)

table(p_value$bh_sign)

 ggplot(data = p_value) +
     geom_point(aes(x = rank, y = p, color = bh_sign)) +
     geom_vline(xintercept = 39) +
     labs(title = "Benjamini-Hochberg Significance", x = "Variables ordered by p-value", y = "p-value")
```