Computer lab 2 block 2

Maximilian Pfundstein (maxpf364)

2018-12-16

Contents

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).

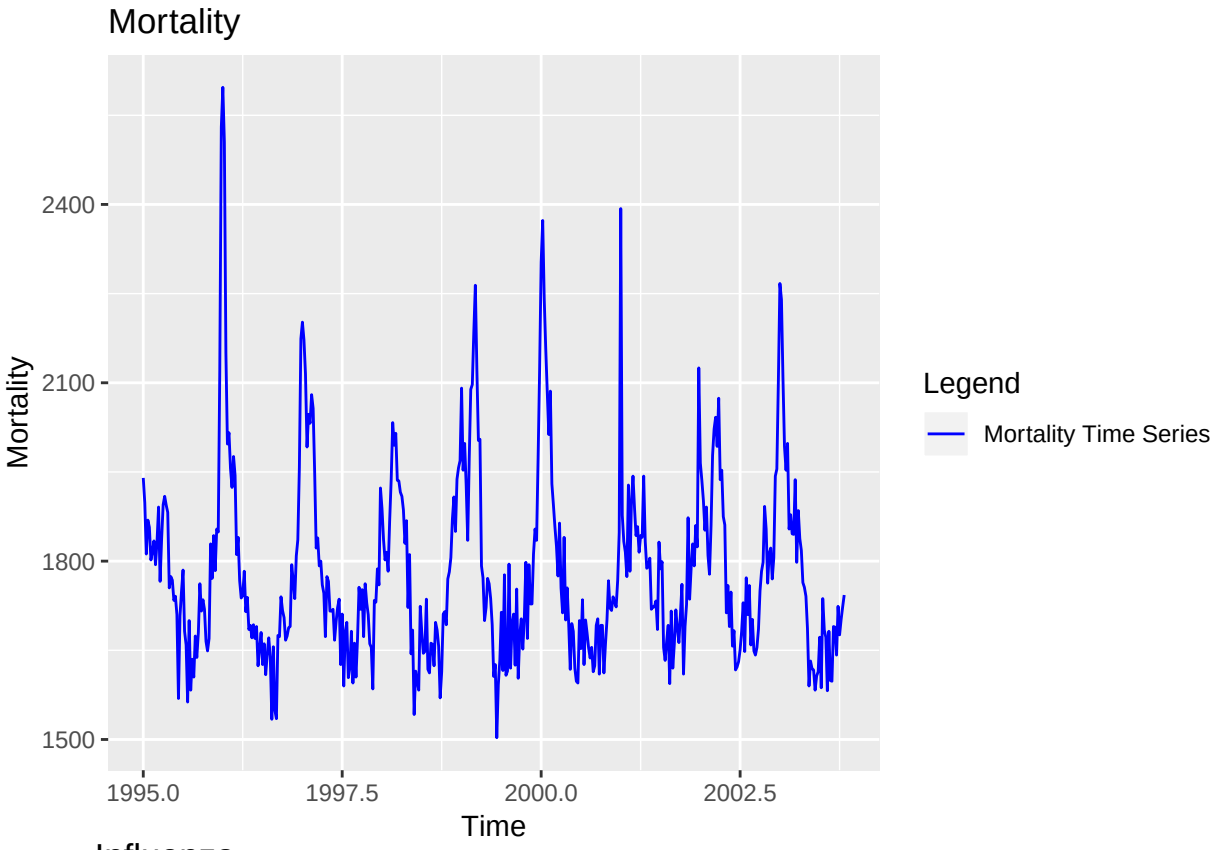
Table 1: influenza.xlsx

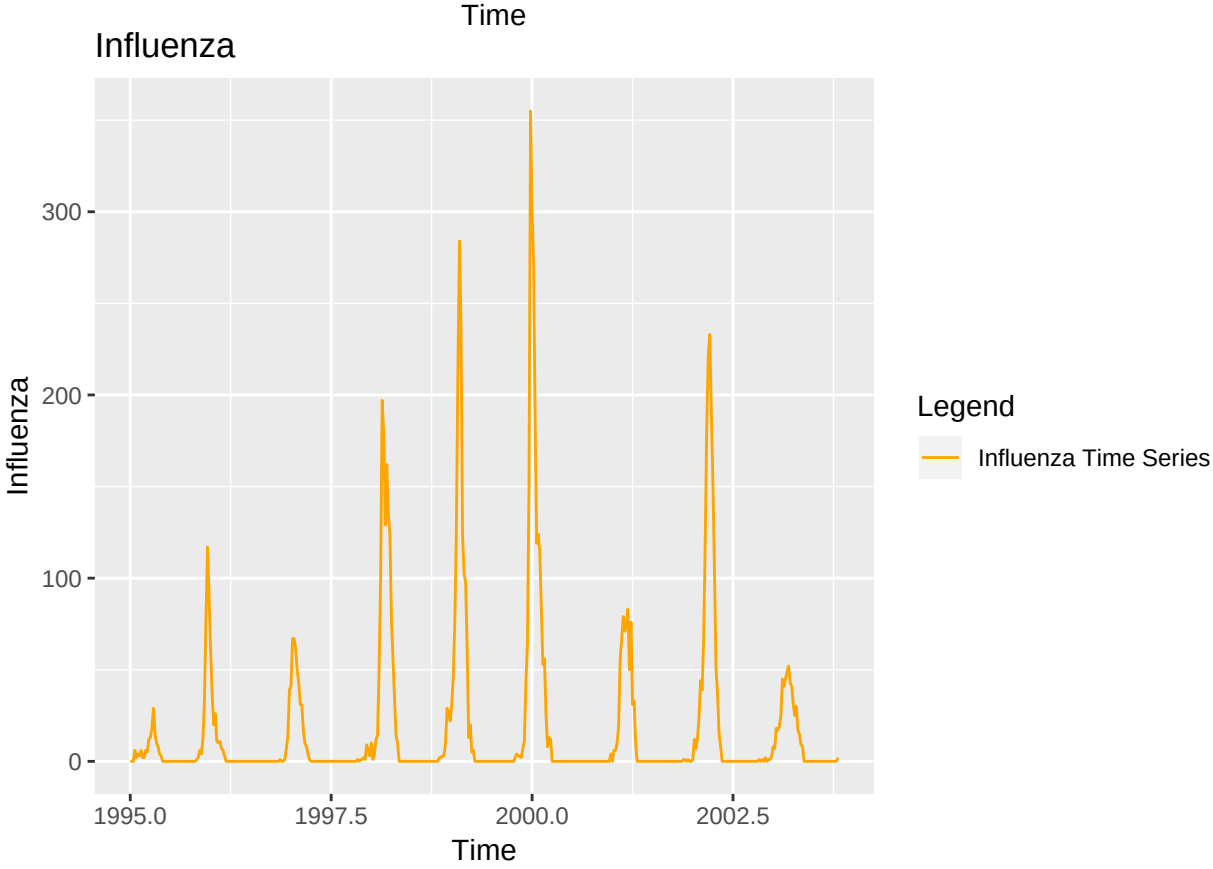
Year	Week	Time	Mortality	Influenza	Temperature deficit
1995	1	1995.000	1940	0	1.34518
1995	2	1995.019	1898	0	0.00000
1995	3	1995.038	1812	0	0.00000
1995	4	1995.058	1869	6	1.40262
1995	5	1995.077	1857	2	0.00000
1995	6	1995.096	1802	4	0.00000

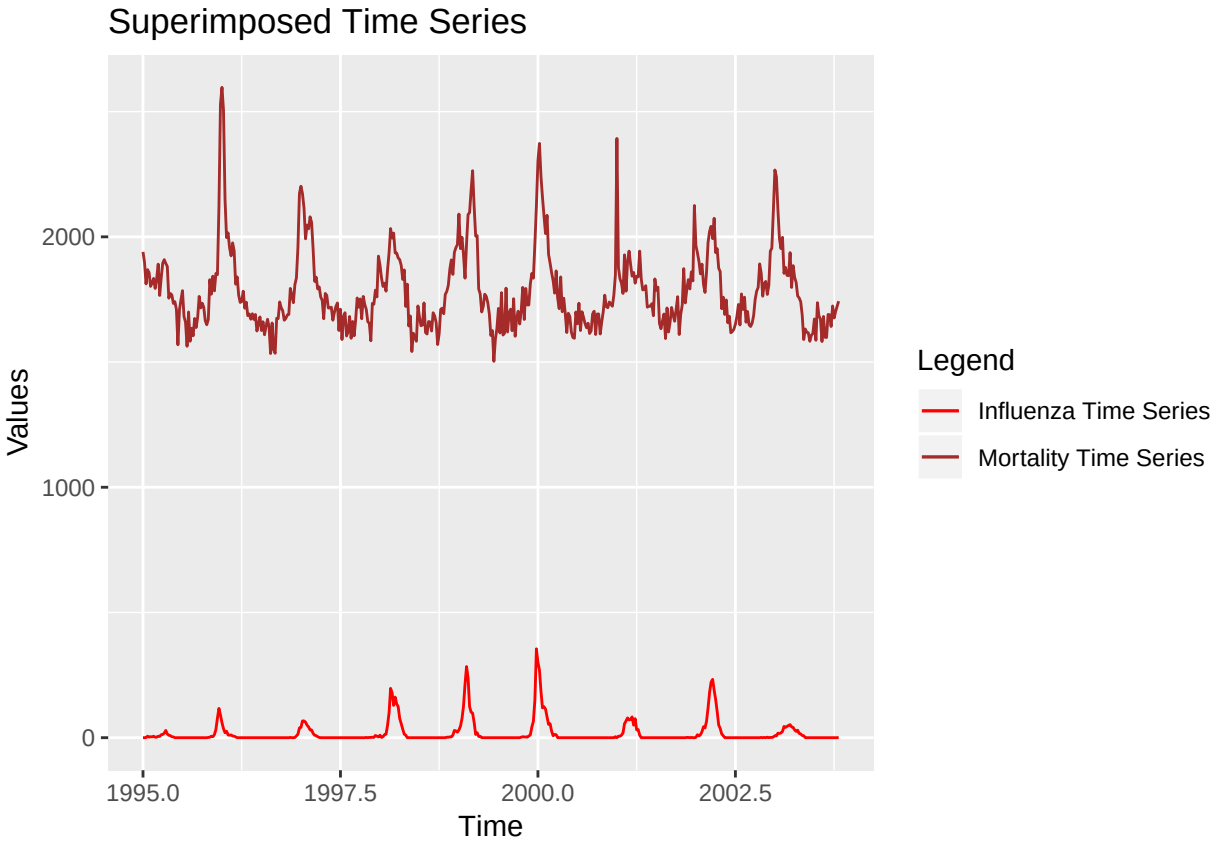
1.1)

Task: 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.

Answer: One plot for mortality and one for influenza. To better compare those two and show, if they are related or not, there is a third plot showing them in one plot.


Mortality
Mortality
2400
2100
1800
1500
1995.0
1997.5
2000.0
2002.5
Time
Legend
Mortality Time Series


Influenza
Influenza
300
200
100
0
1995.0
1997.5
2000.0
2002.5
Time
Legend
Influenza Time Series



If both plots are superimposed we can observe that it seems like that mortality in influenced by influenza.

1.2)

Task: 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.

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = length(unique(influanza$Week)))
##
## Estimated degrees of freedom:
## 14.3  total = 16.32
##
## GCV score: 8708.581     rank: 52/53
```

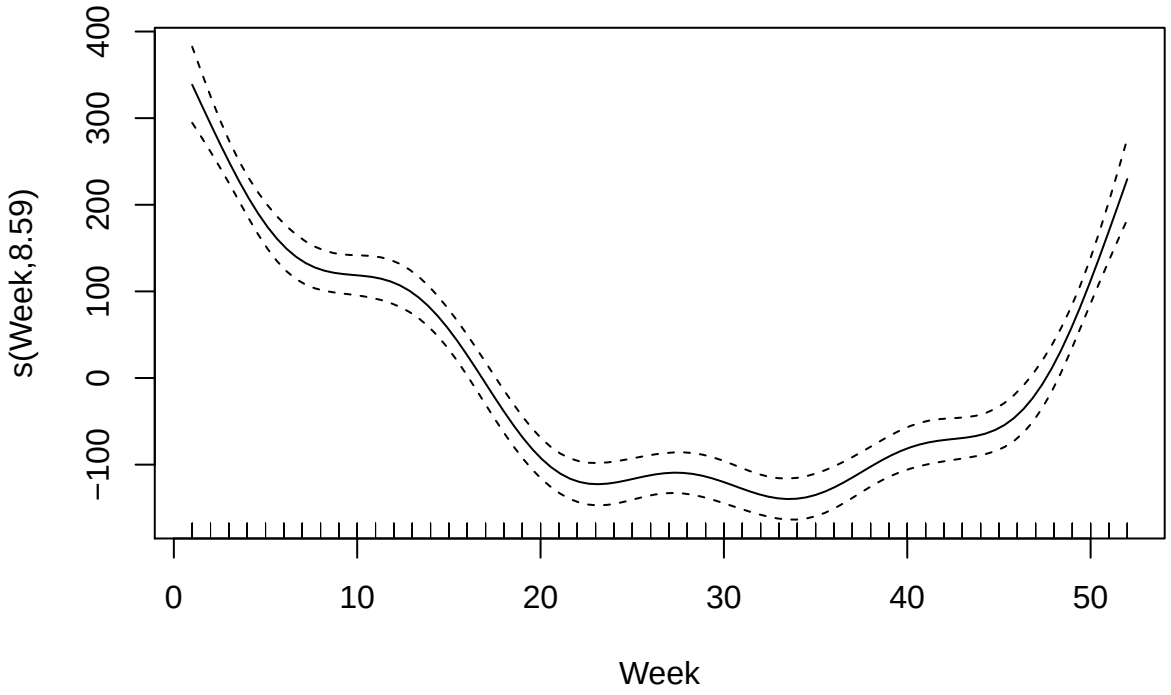
The probabilistic model looks as follows:

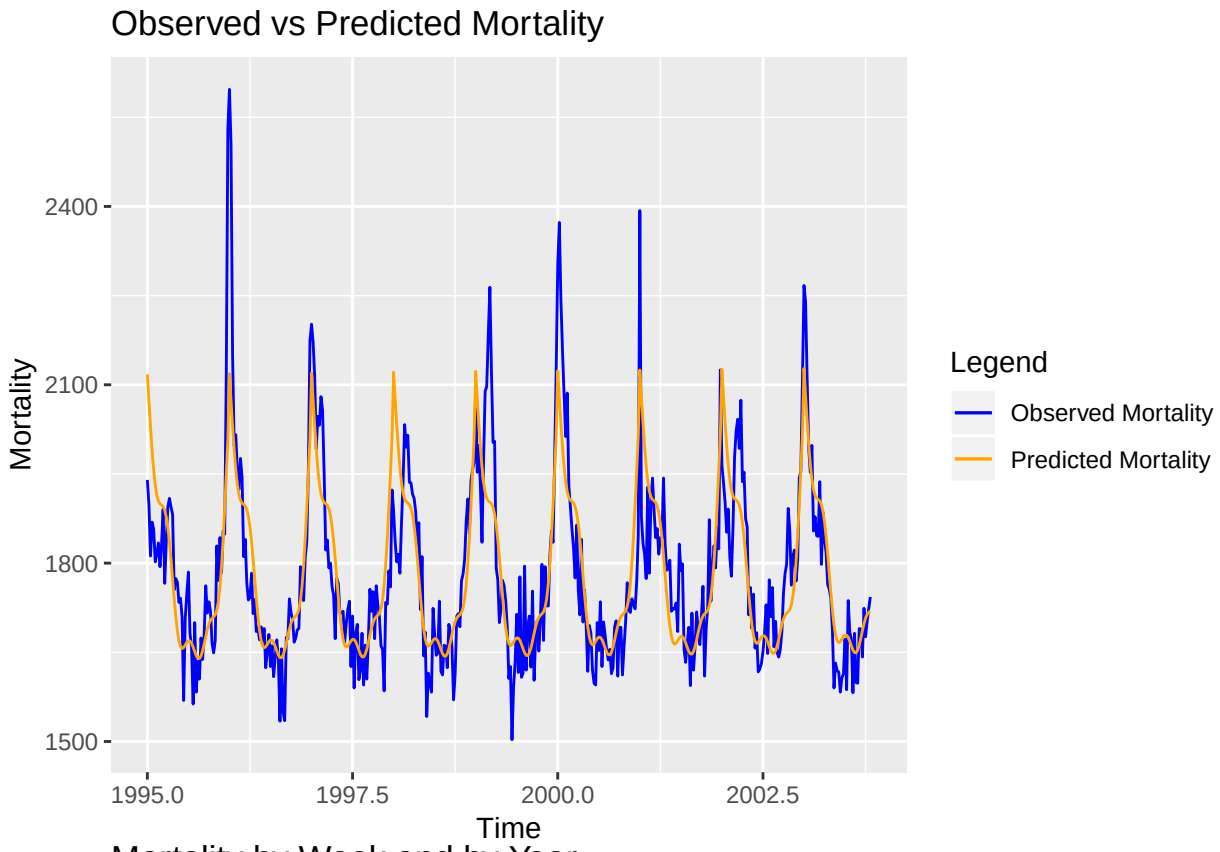
$$Model \sim \mathcal{N}(\beta_{year} * X_{Year} + S_{week} * X_{week} + \alpha, \sigma^2)$$

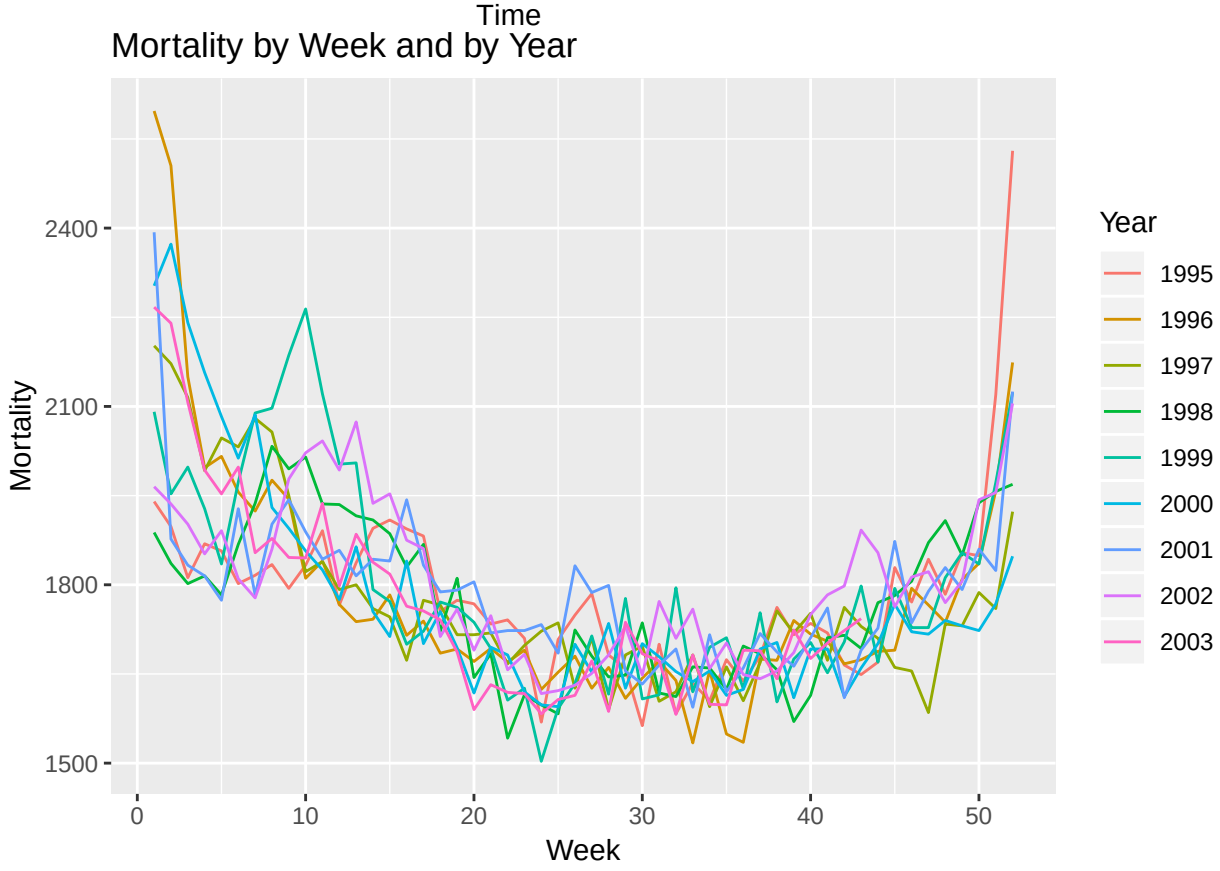
1.3)

Task: 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.

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -652.058   3448.379  -0.189     0.85
## Year           1.219      1.725   0.706     0.48
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Week) 8.587  8.951 100.6  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.661   Deviance explained = 66.8%
## GCV = 9014.6  Scale est. = 8806.7    n = 459
```







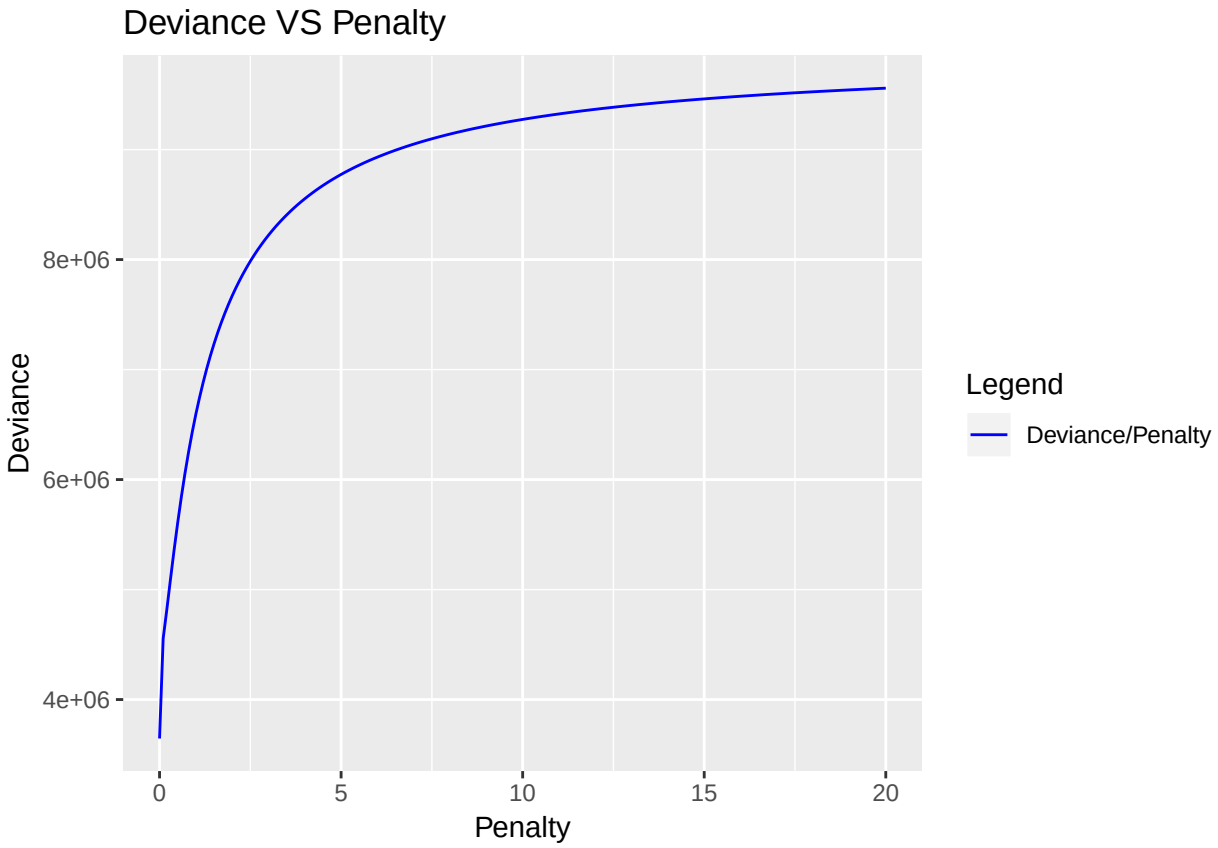
Comment: We can see that the fitted data is actually not performing that bad. It recognized the cyclical

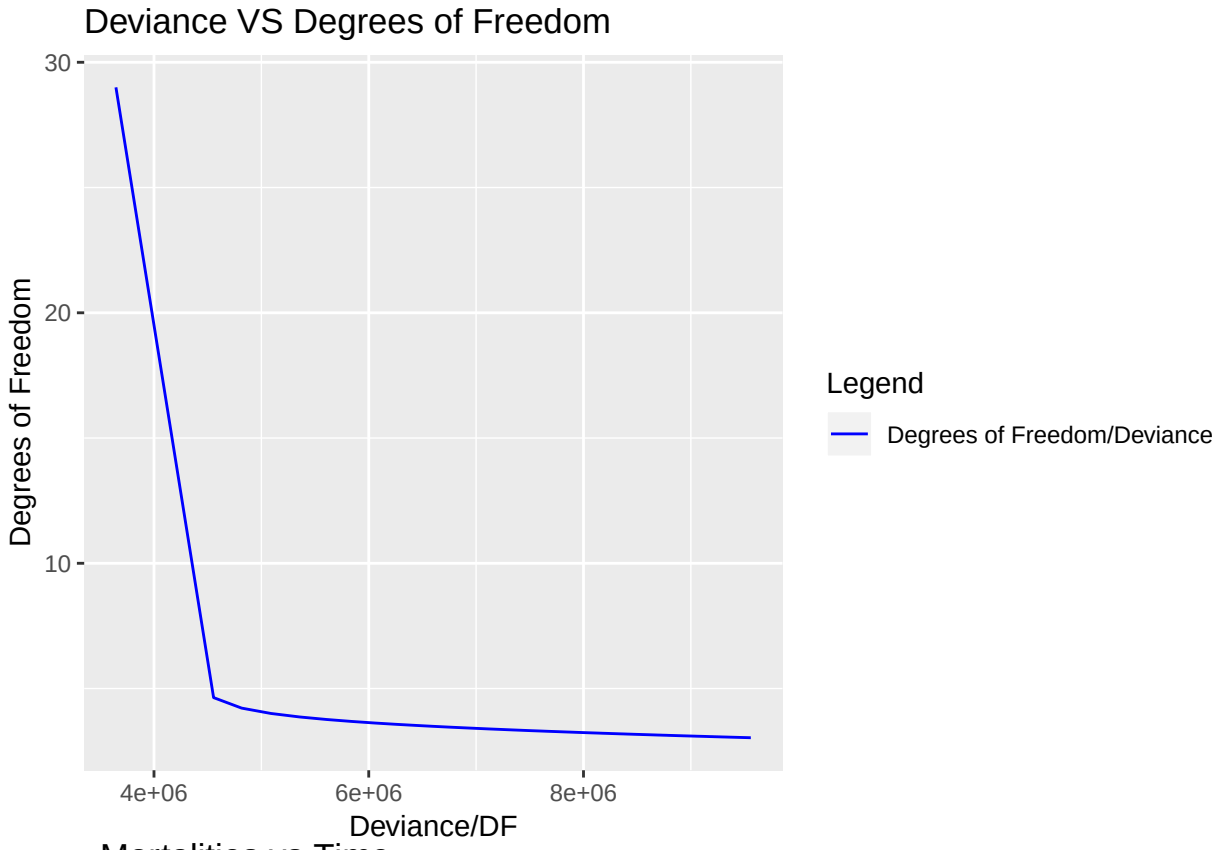
trend correctly. The drawback of the smoothing is that the extrem values are not correctly predicted, as those are smoothed out.

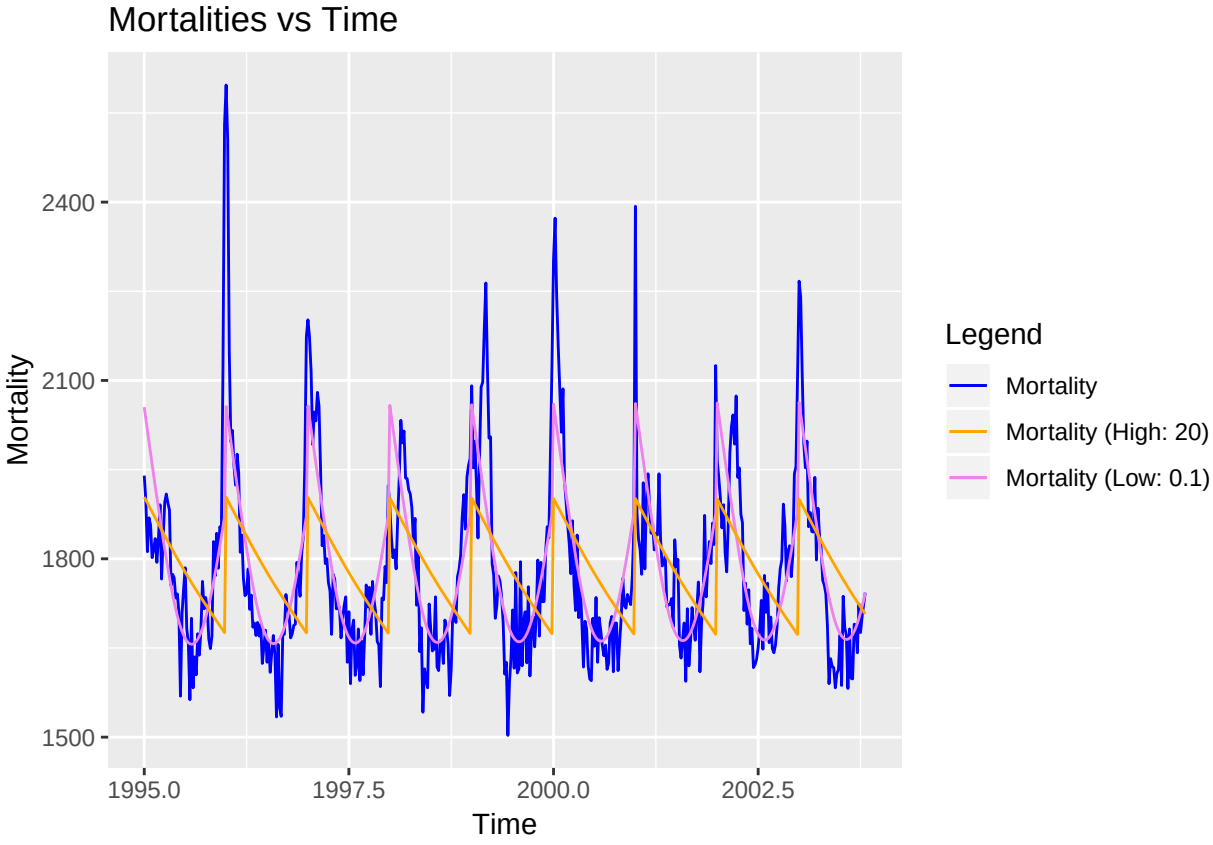
We observe that the spline curve is following the trend we can observe over the years, with a peak in the beginning/end of the year.

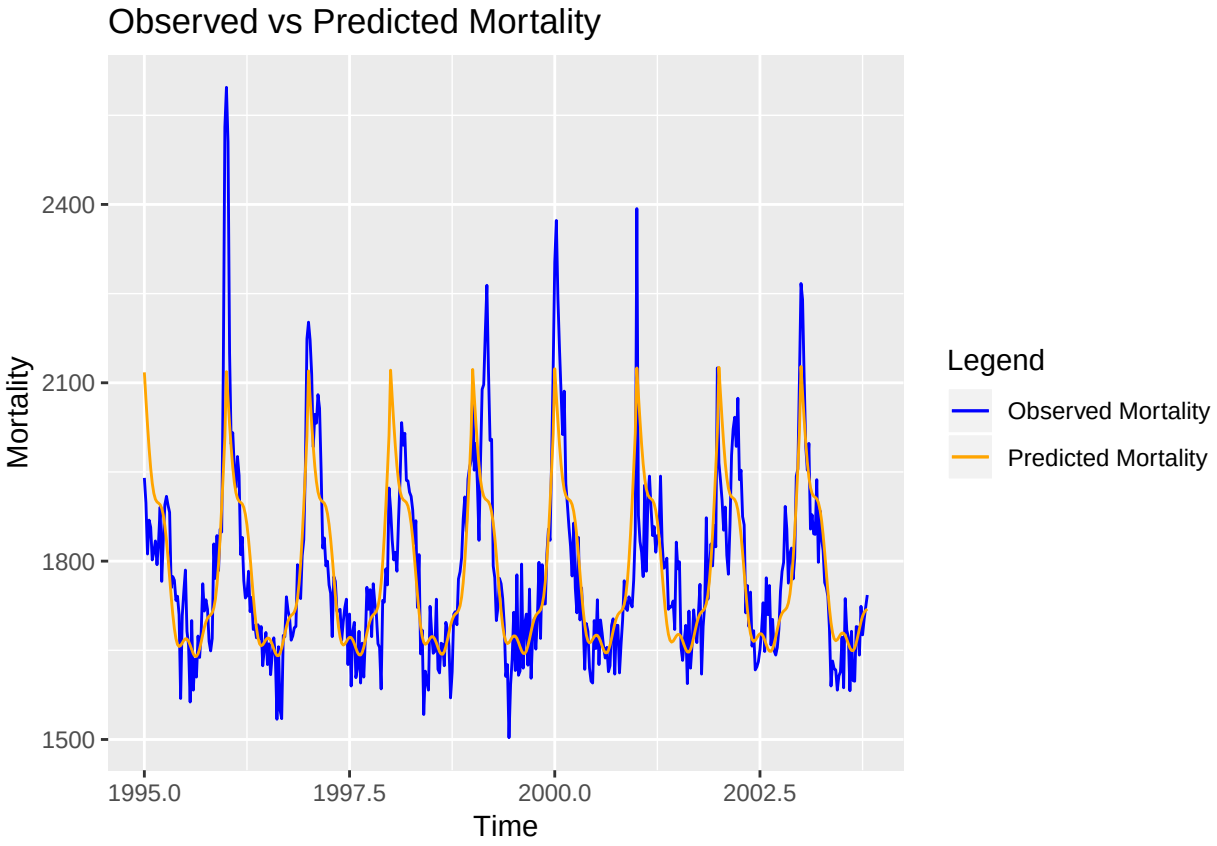
1.4)

Task: 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?




Deviance VS Degrees of Freedom
Degrees of Freedom
30
20
10
4e+06
6e+06
8e+06
Deviance/DF
Legend
Degrees of Freedom/Deviance

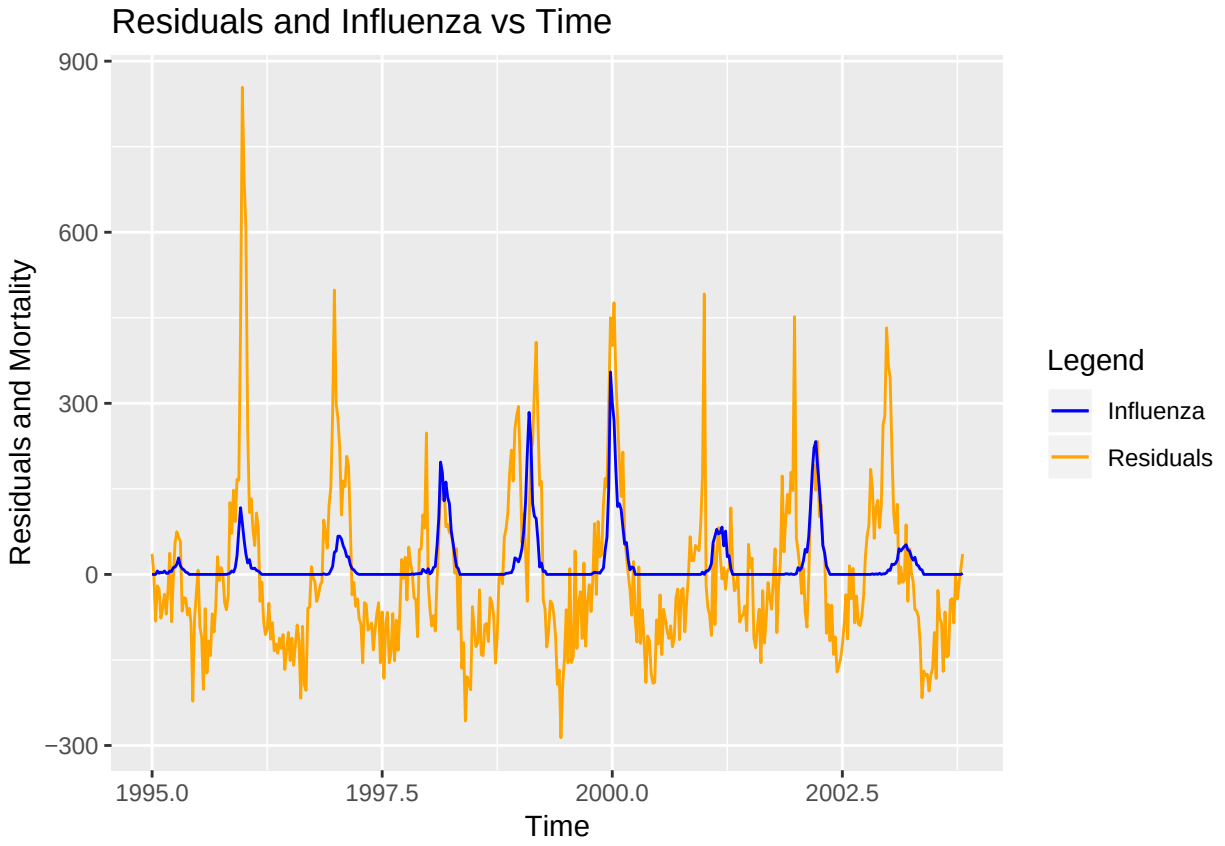

Mortalities vs Time
Mortality
2400
2100
1800
1500
1995.0
1997.5
2000.0
2002.5
Time
Legend
Mortality
Mortality (High: 20)
Mortality (Low: 0.1)



In the first plot we see, that with an increase in penalty the deviance increases as well. The greater the penalty the simpler is teh model, so this is kind of what we expect. For the plot with a low and high penalty we can see, that the model with a low penalty performs better in the way that it better follows the original trends while the model with the high penalty struggles to catch the individuals trends/spikes.

1.5)

Task: 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?



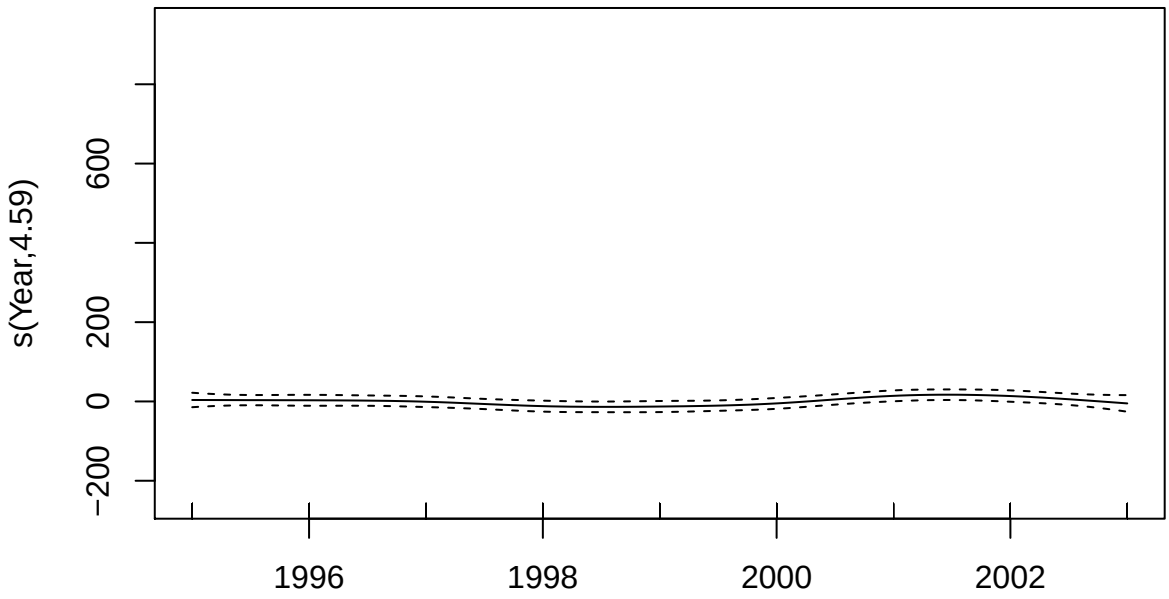
Comment: Yes it seems like that the residuals and the outbreaks are correlated. When there is an outbreak, the residuals also inscrease in that particular moment.

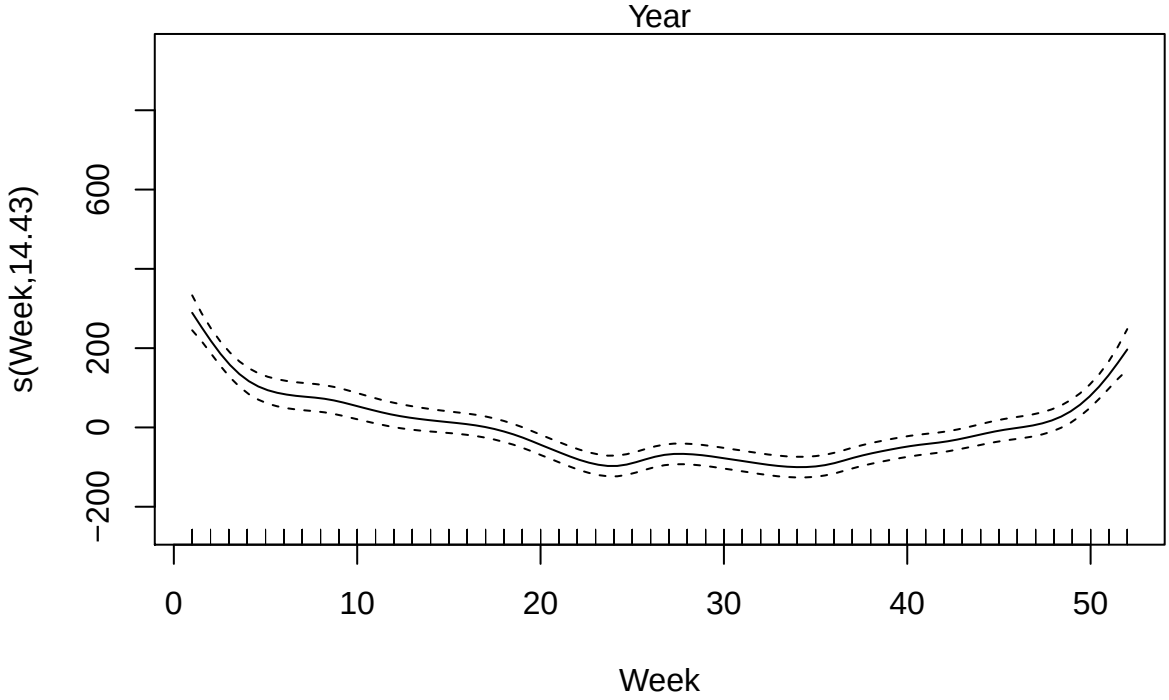
1.6)

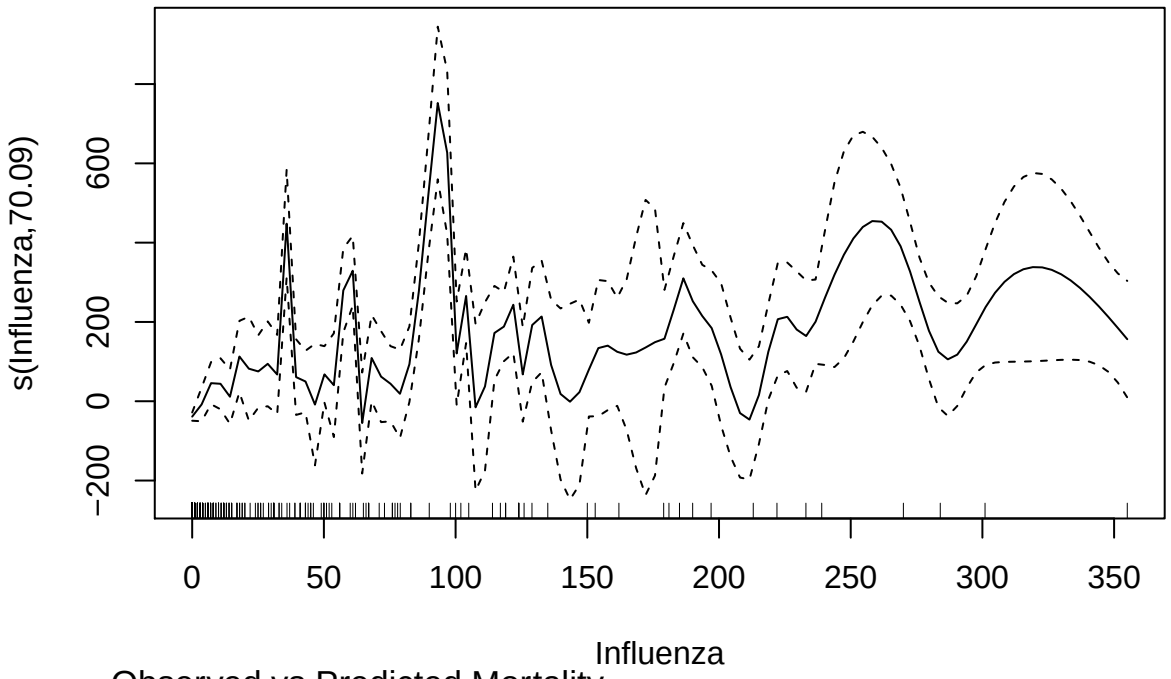
Task: Fit a GAM model in R in which mortality is be modelled as an additive function of the spline functions of year, week, and the number of confirmed cases of influenza. Use the output of this GAM function to conclude whether or not the mortality is influenced by the outbreaks of influenza. Provide the plot of the original and fitted Mortality against Time and comment whether the model seems to be better than the previous GAM models.

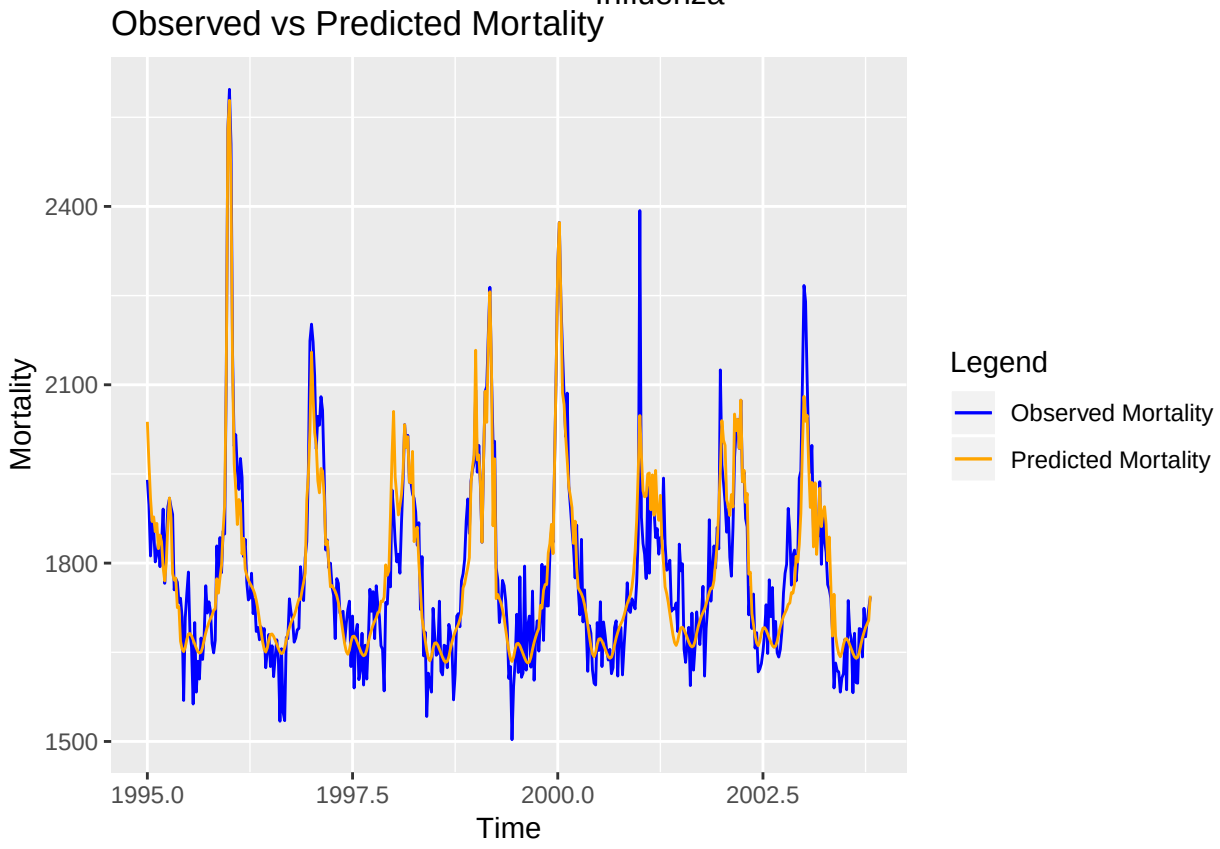
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ s(Year, k = length(unique(influanza$Year))) + s(Week,
##     k = length(unique(influanza$Week))) + s(Influenza, k = length(unique(influanza$Influenza)))
##
## Parametric coefficients:
##             Estimate Std. Error 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
```









Using the new splines in this model concludes in a better model than before. We see that the week and influence are statistically significant while the year is not. Comparing this statement with the provided plots we see, that the year is not capturing any trend, while the week and the influence are capturing the occuring trends.

2) High-dimensional method

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)

X000euro	X5102011	X10th	X11th	X12noon	X12th	X13th	X14th	X15th	X16th
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

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.

The model with the lowest error has a treshold of:

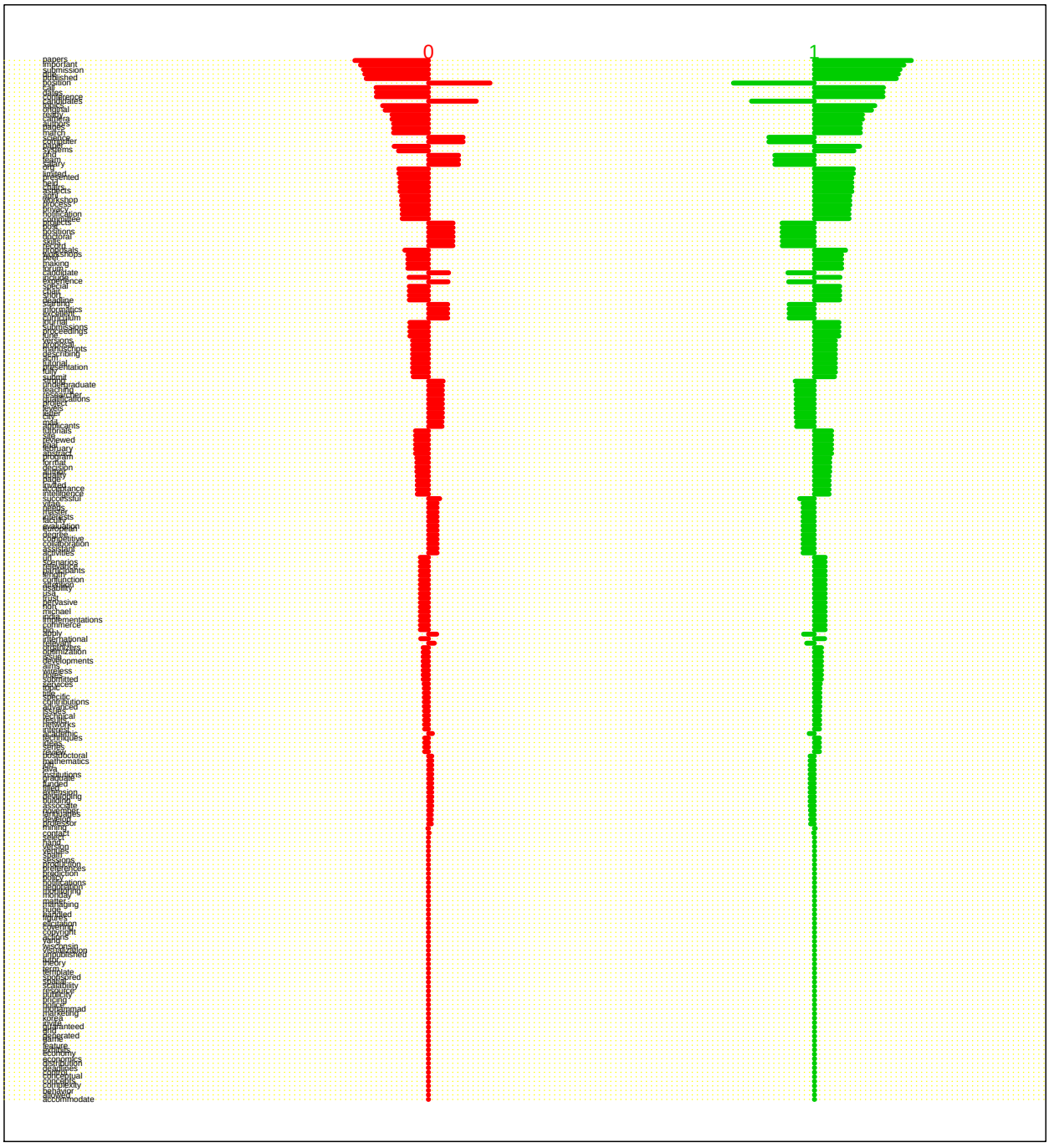
```
##        id   0-score 1-score
##   [1,] 3036 -0.3822 0.5029
##   [2,] 2049 -0.3527 0.4641
##   [3,] 4060 -0.3376 0.4441
##   [4,] 1262 -0.3309 0.4354
##   [5,] 3364 -0.3231 0.4252
##   [6,] 3187 0.3188  -0.4195
##   [7,] 596  -0.2725 0.3585
##   [8,] 869  -0.2706 0.356
##   [9,] 1045 -0.2706 0.356
##  [10,] 607  0.2476  -0.3258
##  [11,] 4282 -0.2383 0.3136
##  [12,] 2990 -0.2254 0.2966
##  [13,] 599  -0.1896 0.2495
##  [14,] 3433 -0.1896 0.2495
##  [15,] 389  -0.1815 0.2389
##  [16,] 2588 -0.1815 0.2389
##  [17,] 3022 -0.1815 0.2389
##  [18,] 850  0.1793  -0.2359
##  [19,] 3725 0.1793  -0.2359
##  [20,] 3035 -0.1785 0.2349
##  [21,] 4129 -0.1559 0.2051
##  [22,] 3125 0.1559  -0.2051
##  [23,] 4177 0.1556  -0.2047
##  [24,] 3671 0.1556  -0.2047
##  [25,] 2974 -0.1542 0.2029
##  [26,] 2463 -0.1542 0.2029
```

```
##  [27,] 329  -0.148  0.1947
##  [28,] 681  -0.148  0.1947
##  [29,] 1891 -0.148  0.1947
##  [30,] 3243 -0.148  0.1947
##  [31,] 283  -0.14   0.1842
##  [32,] 4628 -0.14   0.1842
##  [33,] 3286 -0.14   0.1842
##  [34,] 3274 -0.1368 0.18
##  [35,] 810  -0.1368 0.18
##  [36,] 2889 -0.1368 0.18
##  [37,] 1233 0.1272  -0.1674
##  [38,] 3188 0.1272  -0.1674
##  [39,] 3191 0.1272  -0.1674
##  [40,] 3312 0.1272  -0.1674
##  [41,] 3891 0.1265  -0.1664
##  [42,] 3458 0.1265  -0.1664
##  [43,] 3324 -0.1231 0.162
##  [44,] 1643 -0.1077 0.1417
##  [45,] 2561 -0.1077 0.1417
##  [46,] 3090 -0.1077 0.1417
##  [47,] 4629 -0.1077 0.1417
##  [48,] 606  0.1041  -0.137
##  [49,] 2058 -0.1012 0.1332
##  [50,] 1501 0.1012  -0.1332
##  [51,] 3952 -0.1    0.1316
##  [52,] 680  -0.1    0.1316
##  [53,] 3836 -0.1    0.1316
##  [54,] 1061 -0.0998 0.1314
##  [55,] 1007 0.0995  -0.1309
##  [56,] 1477 0.0995  -0.1309
##  [57,] 2103 0.0995  -0.1309
##  [58,] 3992 0.0995  -0.1309
##  [59,] 2295 -0.0971 0.1278
##  [60,] 4061 -0.0971 0.1278
##  [61,] 2305 -0.097  0.1276
##  [62,] 3285 -0.097  0.1276
##  [63,] 92   -0.0832 0.1094
##  [64,] 1127 -0.0832 0.1094
##  [65,] 2583 -0.0832 0.1094
##  [66,] 3323 -0.0832 0.1094
##  [67,] 4500 -0.0832 0.1094
##  [68,] 1698 -0.0832 0.1094
##  [69,] 3241 -0.0832 0.1094
##  [70,] 4364 -0.0832 0.1094
##  [71,] 4062 -0.0796 0.1048
##  [72,] 4039 0.0757  -0.0996
##  [73,] 740  0.0721  -0.0949
##  [74,] 2438 0.0721  -0.0949
##  [75,] 2442 0.0721  -0.0949
##  [76,] 3311 0.0721  -0.0949
##  [77,] 3383 0.0721  -0.0949
##  [78,] 3559 0.0721  -0.0949
##  [79,] 4176 0.0721  -0.0949
##  [80,] 4402 0.0721  -0.0949
```

```
##  [81,] 267  0.0701  -0.0923
##  [82,] 2553 0.0701  -0.0923
##  [83,] 63   -0.0681 0.0896
##  [84,] 1563 -0.0681 0.0896
##  [85,] 1594 -0.0681 0.0896
##  [86,] 3589 -0.0681 0.0896
##  [87,] 3882 -0.0681 0.0896
##  [88,] 4365 -0.0681 0.0896
##  [89,] 3301 -0.0612 0.0805
##  [90,] 1636 -0.061  0.0802
##  [91,] 1072 -0.061  0.0802
##  [92,] 386  -0.061  0.0802
##  [93,] 2198 -0.0586 0.0771
##  [94,] 3021 -0.0586 0.0771
##  [95,] 3386 -0.0586 0.0771
##  [96,] 76   -0.0583 0.0767
##  [97,] 2150 -0.0583 0.0767
##  [98,] 4075 0.0579  -0.0762
##  [99,] 107  0.0447  -0.0589
## [100,] 336  0.0447  -0.0589
## [101,] 776  0.0447  -0.0589
## [102,] 831  0.0447  -0.0589
## [103,] 1088 0.0447  -0.0589
## [104,] 1450 0.0447  -0.0589
## [105,] 1456 0.0447  -0.0589
## [106,] 1542 0.0447  -0.0589
## [107,] 2170 0.0447  -0.0589
## [108,] 2613 0.0447  -0.0589
## [109,] 2837 0.0447  -0.0589
## [110,] 4529 0.0447  -0.0589
## [111,] 363  -0.0429 0.0564
## [112,] 879  -0.0429 0.0564
## [113,] 2433 -0.0429 0.0564
## [114,] 3051 -0.0429 0.0564
## [115,] 3514 -0.0429 0.0564
## [116,] 3711 -0.0429 0.0564
## [117,] 4449 -0.0429 0.0564
## [118,] 501  -0.0429 0.0564
## [119,] 803  -0.0429 0.0564
## [120,] 2046 -0.0429 0.0564
## [121,] 2082 -0.0429 0.0564
## [122,] 2690 -0.0429 0.0564
## [123,] 2877 -0.0429 0.0564
## [124,] 3118 -0.0429 0.0564
## [125,] 4342 -0.0429 0.0564
## [126,] 4451 -0.0429 0.0564
## [127,] 4452 -0.0429 0.0564
## [128,] 272  0.0425  -0.0559
## [129,] 2175 -0.0408 0.0537
## [130,] 3515 0.0302  -0.0397
## [131,] 172  -0.0284 0.0373
## [132,] 1149 -0.0284 0.0373
## [133,] 2219 -0.0284 0.0373
## [134,] 2964 -0.0284 0.0373
```

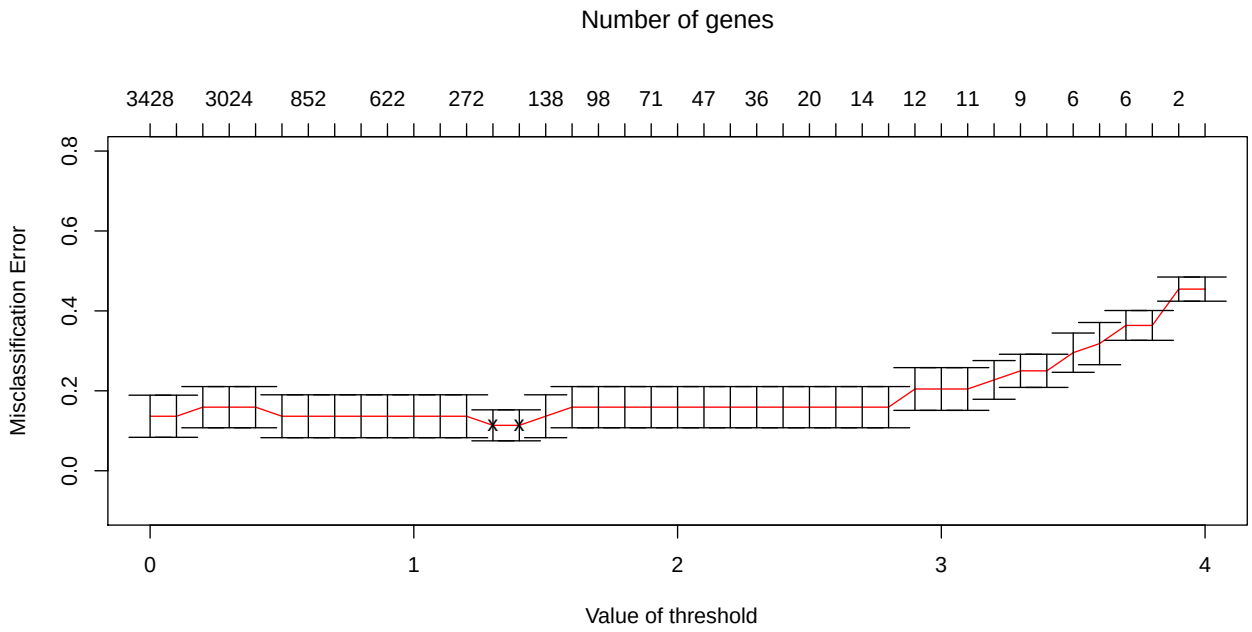
```
## [135,] 2984 -0.0284 0.0373
## [136,] 2887 -0.0284 0.0373
## [137,] 4605 -0.0284 0.0373
## [138,] 4064 -0.028  0.0369
## [139,] 3800 -0.0238 0.0313
## [140,] 134  -0.0222 0.0292
## [141,] 919  -0.0222 0.0292
## [142,] 3957 -0.0222 0.0292
## [143,] 4268 -0.0222 0.0292
## [144,] 4281 -0.0222 0.0292
## [145,] 2220 -0.0211 0.0277
## [146,] 2847 -0.0211 0.0277
## [147,] 3582 -0.0211 0.0277
## [148,] 4181 -0.0211 0.0277
## [149,] 2167 -0.0204 0.0268
## [150,] 67   0.0204  -0.0268
## [151,] 2005 -0.0203 0.0267
## [152,] 4185 -0.0203 0.0267
## [153,] 3588 -0.0203 0.0267
## [154,] 3794 -0.0203 0.0267
## [155,] 579  0.017   -0.0223
## [156,] 1147 0.017   -0.0223
## [157,] 1524 0.017   -0.0223
## [158,] 1591 0.017   -0.0223
## [159,] 1702 0.017   -0.0223
## [160,] 1797 0.017   -0.0223
## [161,] 2141 0.017   -0.0223
## [162,] 2251 0.017   -0.0223
## [163,] 2278 0.017   -0.0223
## [164,] 2619 0.017   -0.0223
## [165,] 3194 0.017   -0.0223
## [166,] 340  0.017   -0.0223
## [167,] 2894 0.0156  -0.0205
## [168,] 1144 0.0148  -0.0195
## [169,] 2392 0.0148  -0.0195
## [170,] 3295 0.0148  -0.0195
## [171,] 2713 -0.0036 0.0048
## [172,] 899  0.0032  -0.0042
## [173,] 1859 -0.0011 0.0015
## [174,] 3764 -0.0011 0.0015
## [175,] 104  -0.0011 0.0015
## [176,] 940  -0.0011 0.0015
## [177,] 967  -0.0011 0.0015
## [178,] 1343 -0.0011 0.0015
## [179,] 1587 -0.0011 0.0015
## [180,] 1861 -0.0011 0.0015
## [181,] 1965 -0.0011 0.0015
## [182,] 2574 -0.0011 0.0015
## [183,] 2623 -0.0011 0.0015
## [184,] 2754 -0.0011 0.0015
## [185,] 2757 -0.0011 0.0015
## [186,] 2839 -0.0011 0.0015
## [187,] 2890 -0.0011 0.0015
## [188,] 3169 -0.0011 0.0015
```

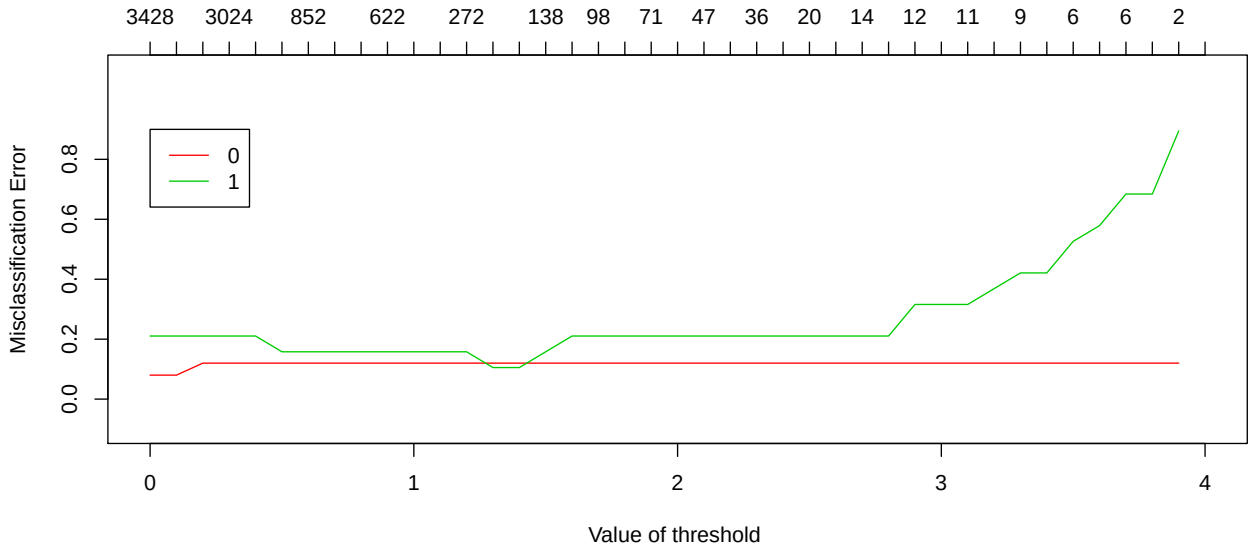
```
## [189,] 3224 -0.0011 0.0015
## [190,] 3231 -0.0011 0.0015
## [191,] 3289 -0.0011 0.0015
## [192,] 3802 -0.0011 0.0015
## [193,] 3943 -0.0011 0.0015
## [194,] 4490 -0.0011 0.0015
## [195,] 4499 -0.0011 0.0015
## [196,] 84   -0.0011 0.0015
## [197,] 196  -0.0011 0.0015
## [198,] 455  -0.0011 0.0015
## [199,] 837  -0.0011 0.0015
## [200,] 856  -0.0011 0.0015
## [201,] 857  -0.0011 0.0015
## [202,] 920  -0.0011 0.0015
## [203,] 1062 -0.0011 0.0015
## [204,] 1214 -0.0011 0.0015
## [205,] 1291 -0.0011 0.0015
## [206,] 1292 -0.0011 0.0015
## [207,] 1490 -0.0011 0.0015
## [208,] 1560 -0.0011 0.0015
## [209,] 1721 -0.0011 0.0015
## [210,] 1745 -0.0011 0.0015
## [211,] 1818 -0.0011 0.0015
## [212,] 1833 -0.0011 0.0015
## [213,] 2197 -0.0011 0.0015
## [214,] 2359 -0.0011 0.0015
## [215,] 2598 -0.0011 0.0015
## [216,] 2746 -0.0011 0.0015
## [217,] 2888 -0.0011 0.0015
## [218,] 3259 -0.0011 0.0015
## [219,] 3361 -0.0011 0.0015
## [220,] 3570 -0.0011 0.0015
## [221,] 3703 -0.0011 0.0015
## [222,] 3948 -0.0011 0.0015
## [223,] 3966 -0.0011 0.0015
## [224,] 4202 -0.0011 0.0015
## [225,] 4212 -0.0011 0.0015
## [226,] 4236 -0.0011 0.0015
## [227,] 4363 -0.0011 0.0015
## [228,] 4435 -0.0011 0.0015
## [229,] 4526 -0.0011 0.0015
## [230,] 4606 -0.0011 0.0015
## [231,] 4664 -0.0011 0.0015
```




```
## Call:
## pamr.cv(fit = nsc_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
```





The amount of features selected is:

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

And the 10 most contributing features are:

x
papers
important
submission
due
published
position
call

x
conference
dates
candidates

We have the following confuses matrices.

	0	1
0	22	3
1	1	18

	0	1
0	10	0
1	2	8

And therefor the following error rates:

```
## [1] "Error Train: 0.0909090909090908"
```

```
## [1] "Error Validation: 0.1"
```

TODO:

- Threshold?
- Interpretation of the Centroid Plots!

2.2)

Compute the test error and the number of the contributing features for the following methods fitted to the training data:

- Elastic net with the binomial response and a = 0.5 in which penalty is selected by the cross-validation
- 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?

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

	0	1
0	10	0
1	4	6

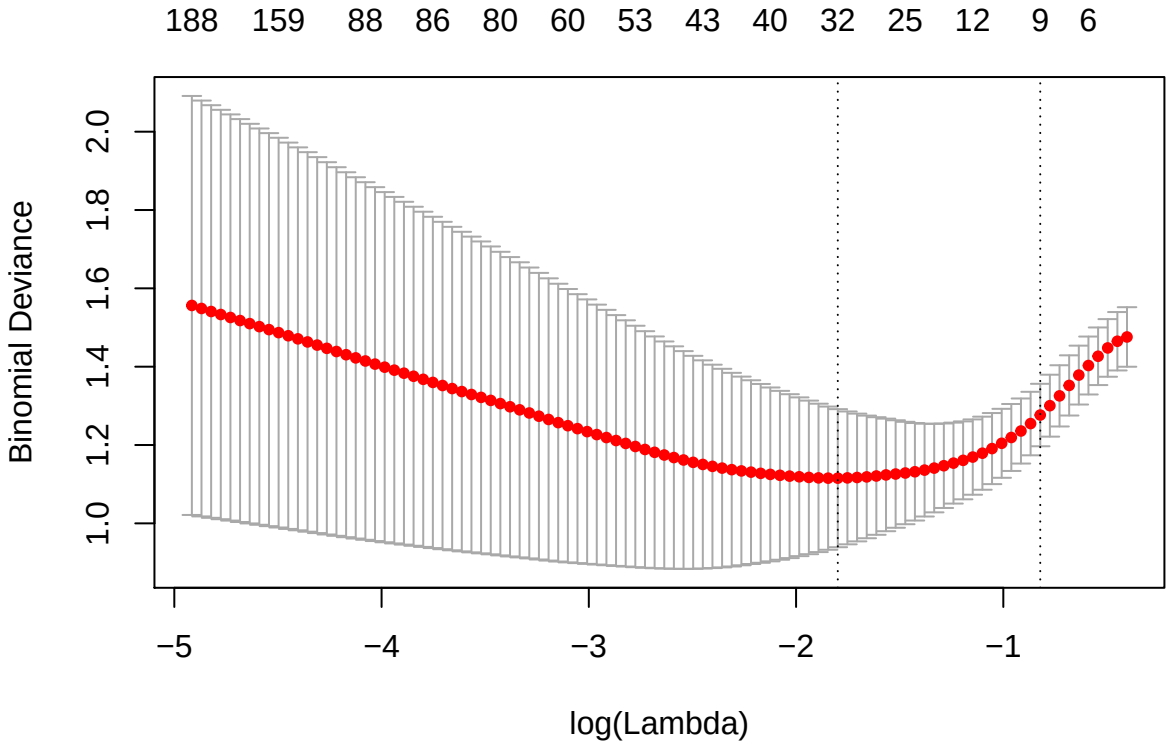
	0	1
0	10	0
1	1	9

```
## [1] "Error Elastic: 0.2"
```

```
## [1] "Error SVM: 0.05"
```

Data about the elastic model:

```
##             Length Class  Mode
## lambda      98     -none- numeric
## cvm         98     -none- numeric
## cvsd        98     -none- numeric
## cvup        98     -none- numeric
## cvlo        98     -none- numeric
## nzero       98     -none- numeric
## name         1     -none- character
## glmnet.fit 13     lognet list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
```



Data bout the svm model:

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 43
##
## Objective Function Value : -2.0817
## Training error : 0.022727
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

To get an overview over all trained model errors, let's have a look at the following table. The error rates are based on the test/validation data set, not on the train data set.

NSCM_error	Elastic_error	SVM_error
0.1	0.2	0.05

Therefore the best model is the SVM model as it has the smallest error. We shoudk keep in mind that if works with all features while NSCM uses less features. As the dataset is small and the computations are done quickly, the benefit of little features vanishes and we still take the SVM model. The elastic error is just to high to be considered.

2.3)

Task: 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.

Answer: As there is no α level given, we choose it as $\alpha = 0.05$. The implemented method can be found in the source code appendix.

After applying the BH-method, we have `39` features left, which are the following (inculdung their p_values and cv_scores).

feature_names	p_values	cv_scores
papers	0.0000000	0.0000106
submission	0.0000000	0.0000213
position	0.0000000	0.0000319
published	0.0000002	0.0000425
important	0.0000003	0.0000532
call	0.0000004	0.0000638
conference	0.0000005	0.0000744
candidates	0.0000009	0.0000851
dates	0.0000014	0.0000957
paper	0.0000014	0.0001063
topics	0.0000051	0.0001170
limited	0.0000079	0.0001276
candidate	0.0000119	0.0001382
camera	0.0000210	0.0001489
ready	0.0000210	0.0001595
authors	0.0000215	0.0001701
phd	0.0000338	0.0001808
projects	0.0000350	0.0001914
org	0.0000374	0.0002020
chairs	0.0000586	0.0002127
due	0.0000649	0.0002233
original	0.0000649	0.0002339
notification	0.0000688	0.0002446
salary	0.0000797	0.0002552
record	0.0000909	0.0002658
skills	0.0000909	0.0002765
held	0.0001529	0.0002871
team	0.0001758	0.0002977
pages	0.0002007	0.0003084

feature_names	p_values	cv_scores
workshop	0.0002007	0.0003190
committee	0.0002117	0.0003296
proceedings	0.0002117	0.0003403
apply	0.0002166	0.0003509
strong	0.0002246	0.0003615
international	0.0002296	0.0003722
degree	0.0003762	0.0003828
excellent	0.0003762	0.0003934
post	0.0003762	0.0004041
presented	0.0003765	0.0004147

The purpose of the BH-method is to decrease the FDR (False Discovery Rate). The last entry in the list has the highest p_value, which is smaller then the cirtical value. This means that all values after that (here truncated) are considered significant. This means, for those values H_0 is rejected.

Appendix

```
knitr::opts_chunk$set(echo = FALSE, cache = TRUE)
library(readxl)
library(ggplot2)
library(knitr)
library(mgcv)
library(pamr)
library(kernlab)
library(glmnet)
set.seed(12345)


################################################################################
# 1) Using GAM and GLM to examine the mortality rates
################################################################################

set.seed(12345)
influanza = read_excel("./influenza.xlsx")
#creditscoring$good_bad = as.factor(creditscoring$good_bad)
kable(head(influanza), caption = "influenza.xlsx")


################################################################################
# 1.1)
################################################################################

ggplot(influanza) +
  geom_line(aes(x = influanza$Time, y = influanza$Mortality,
                colour = "Mortality Time Series")) +
  labs(title = "Mortality", y = "Mortality",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("blue"))
```

```r
ggplot(influanza) +
  geom_line(aes(x = influanza$Time, y = influanza$Influenza,
                colour = "Influenza Time Series")) +
  labs(title = "Influenza", y = "Influenza",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("orange"))

ggplot(influanza) +
  geom_line(aes(x = influanza$Time, y = influanza$Mortality,
                colour = "Mortality Time Series")) +
  geom_line(aes(x = influanza$Time, y = influanza$Influenza,
                colour = "Influenza Time Series")) +
  labs(title = "Superimposed Time Series", y = "Values",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("red", "brown"))

################################################################################
# 1.2)
################################################################################

# - Mortality is normally distributed and modelled as...
# - ...a linear function of Year...
# - ...and spline function of Week
# Don't forget to select the model bei generalized cross-validation

gam_model = gam(formula = Mortality ~ Year + s(Week, k=length(unique(influanza$Week))), family = gaussi
                data = influanza, method="GCV.Cp")

print(gam_model)


################################################################################
# 1.3)
################################################################################

summary(gam_model)
plot(gam_model)

time = influanza$Time
observed = influanza$Mortality
predicted = gam_model$fitted.values

mortality_values = data.frame(time, observed, predicted)

ggplot(mortality_values) +
  geom_line(aes(x = time, y = observed,
                colour = "Observed Mortality")) +
  geom_line(aes(x = time, y = predicted,
                colour = "Predicted Mortality")) +
  labs(title = "Observed vs Predicted Mortality", y = "Mortality",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("blue", "orange"))
```

```r
ggplot() +
  geom_line(aes(x=influanza$Week,y=influanza$Mortality,
                        colour=as.factor(influanza$Year))) +
  labs(x='Week', y='Mortality', colour='Year',
       title='Mortality by Week and by Year')


################################################################################
# 1.4)
################################################################################

penalties = seq(from = 0 , to = 20, by = 0.1)
index = 1

fitted_gam_with_penalties = data.frame()

for (penalty in penalties) {
  gam_model = gam(formula = Mortality ~ Year +
                    s(Week, sp = penalty, k = length(unique(influanza$Week))),
                  family = gaussian(), data = influanza, method="GCV.Cp")

  fitted_gam_with_penalties = rbind(fitted_gam_with_penalties,
        data.frame(list(penalty = penalty,
                        deviance = gam_model$deviance,
                        df = sum(influence(gam_model)))))
}

## Deviance VS Penalty
ggplot(fitted_gam_with_penalties) +
  geom_line(aes(x = fitted_gam_with_penalties$penalty,
                y = fitted_gam_with_penalties$deviance,
                colour = "Deviance/Penalty")) +
  labs(title = "Deviance VS Penalty", y = "Deviance",
       x = "Penalty", color = "Legend") +
  scale_color_manual(values = c("blue", "orange"))

## Penalty Factors vs Degrees of Freedom
ggplot(fitted_gam_with_penalties) +
  geom_line(aes(x = fitted_gam_with_penalties$deviance,
                y = fitted_gam_with_penalties$df,
                colour = "Degrees of Freedom/Deviance")) +
  labs(title = "Deviance VS Degrees of Freedom", y = "Degrees of Freedom",
       x = "Deviance/DF", color = "Legend") +
  scale_color_manual(values = c("blue", "orange"))

## Create Models for High and Low penalties
gam_model_low_pen = gam(formula = Mortality ~ Year + s(Week, k=length(unique(influanza$Week)), sp = 0.1
                  family = gaussian(), data = influanza, method="GCV.Cp")
gam_model_high_pen =
  gam(formula = Mortality ~ Year + s(Week, k=length(unique(influanza$Week)),
                                     sp = 20),
                  family = gaussian(), data = influanza, method="GCV.Cp")
```

```r
high_low_df = data.frame()
high_low_df = rbind(high_low_df,
                    list(mortality = influanza$Mortality,
                         mortality_low = fitted(gam_model_low_pen),
                         mortality_high = fitted(gam_model_high_pen),
                         date = influanza$Time))

ggplot(high_low_df) +
  geom_line(aes(x = date, y = mortality, colour = "Mortality")) +
  geom_line(aes(x = date, y = mortality_low, colour = "Mortality (Low: 0.1)")) +
  geom_line(aes(x = date, y = mortality_high, colour = "Mortality (High: 20)")) +
  labs(title = "Mortalities vs Time", y = "Mortality",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("blue", "orange", "violet"))

## Observed vs Predicted Mortality
ggplot(mortality_values) +
  geom_line(aes(x = time, y = observed,
                colour = "Observed Mortality")) +
  geom_line(aes(x = time, y = predicted,
                colour = "Predicted Mortality")) +
  labs(title = "Observed vs Predicted Mortality", y = "Mortality",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("blue", "orange"))


################################################################################
# 1.5)
################################################################################

time = influanza$Time
resid = gam_model$residuals
influenza_data = influanza$Influenza

print_data = data.frame(time, resid, influenza_data)

ggplot(print_data) +
  geom_line(aes(x = time, y = resid,
                colour = "Residuals")) +
  geom_line(aes(x = time, y = influenza_data,
                colour = "Influenza")) +
  labs(title = "Residuals and Influenza vs Time", y = "Residuals and Mortality",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("blue", "orange"))


################################################################################
# 1.6)
################################################################################

gam_model_additive = gam(formula =
                         Mortality ~ s(Year, k=length(unique(influanza$Year))) +
                         s(Week, k=length(unique(influanza$Week))) +
```

```
                        s(Influenza, k=length(unique(influanza$Influenza))),
                        family = gaussian(), data = influanza, method="GCV.Cp")

summary(gam_model_additive)
plot(gam_model_additive)

time = influanza$Time
observed = influanza$Mortality
predicted = fitted(gam_model_additive)

mortality_values = data.frame(time, observed, predicted)

ggplot(mortality_values) +
  geom_line(aes(x = time, y = observed,
                colour = "Observed Mortality")) +
  geom_line(aes(x = time, y = predicted,
                colour = "Predicted Mortality")) +
  labs(title = "Observed vs Predicted Mortality", y = "Mortality",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("blue", "orange"))


##############################################################################
# 2) High-dimensional method
##############################################################################

emails = read.csv2("data.csv", fileEncoding = "ISO-8859-1", sep = ";")
emails$Conference = as.factor(emails$Conference)
kable(head(emails[,1:10]))

set.seed(12345)

n = dim(emails)[1]
id = sample(1:n, floor(n*0.70))
train_emails = emails[id,]
val_emails = emails[-id,]


##############################################################################
# 2.1)
##############################################################################

rownames(train_emails) = 1:nrow(train_emails)
x_train = t(train_emails[,-ncol(train_emails)])
y_train = train_emails[[ncol(train_emails)]]
mydata_train = list(x=x_train ,y=as.factor(y_train), geneid =
                      as.character(1:nrow(x_train)),
              genenames = rownames(x_train))

nsc_model = pamr.train(mydata_train, threshold=seq(0, 4, 0.1))
nsc_model_cv = pamr.cv(nsc_model, mydata_train)

nsc_model_cv$threshold[which.min(nsc_model_cv$error)]
```

```r
genes = pamr.listgenes(nsc_model, mydata_train, threshold = 1.3)

pamr.plotcen(nsc_model, mydata_train, threshold = 1.3)

print(nsc_model_cv)
pamr.plotcv(nsc_model_cv)


nrow(genes)


kable(colnames(emails)[as.numeric(genes[1:10, 1])])


rownames(val_emails) = 1:nrow(val_emails)
x_val = t(val_emails[,-ncol(val_emails)])
y_val = val_emails[[ncol(val_emails)]]
#mydata_val = list(x=x_val ,y=as.factor(y_val),
# geneid = as.character(1:nrow(x_val)),
#              genenames = rownames(x_val))

pred_train = pamr.predict(nsc_model, newx = x_train, threshold = 2.5)
pred_val = pamr.predict(nsc_model, newx = x_val, threshold = 2.5)

matrix_train = table(train_emails$Conference, pred_train)
matrix_val = table(val_emails$Conference, pred_val)

kable(matrix_train)
kable(matrix_val)


error_train_nsc = 1 - sum(diag(matrix_train)/sum(matrix_train))
error_val_scn = 1 - sum(diag(matrix_val)/sum(matrix_val))

print(paste("Error Train:", error_train_nsc))
print(paste("Error Validation:", error_val_scn))


##############################################################################
# 2.3)
##############################################################################

# Elastic Net

# Predictor Variables. The -1 removes the intercept component
x_test = as.matrix(train_emails[, -ncol(train_emails)])
x_val = as.matrix(val_emails[, -ncol(val_emails)])

# Outcome variable
y_test = train_emails$Conference
y_val = val_emails$Conference
```

```r
model_elastic_net = cv.glmnet(x = x_test, y = y_test, alpha = 0.5,
                              family = "binomial")

# Support vector machine with "vanilladot" kernel.
# Vanilladot means "Linear Kernel Function"
# set scale = FALSE to prevent "variable(s) `' constant. Cannot scale data.""
model_svm = ksvm(x = x_test, y = y_test, kernel = "vanilladot", scale = FALSE)

# Predictions
pred_val_elastic = predict(model_elastic_net, newx = x_val, type = "class")
pred_val_svm = predict(model_svm, x_val, type = "response")

matrix_elastic = table(y_val, t(pred_val_elastic))
matrix_svm = table(y_val, pred_val_svm)

kable(matrix_elastic)
kable(matrix_svm)


error_elastic = 1 - sum(diag(matrix_elastic)/sum(matrix_elastic))
error_svm = 1 - sum(diag(matrix_svm)/sum(matrix_svm))

print(paste("Error Elastic:", error_elastic))
print(paste("Error SVM:", error_svm))


#print(model_elastic_net)
summary(model_elastic_net)
plot(model_elastic_net)


print(model_svm)
summary(model_svm)
#plot(model_svm)


plot_table = data.frame(
  NSCM_error = error_val_scn,
  Elastic_error = error_elastic,
  SVM_error = error_svm
)

kable(plot_table)


##############################################################################
# 2.3)
##############################################################################

# Orignal Data Set: emails
# H0: Feature has effect on Conference
# Ha: Feature has no effect on Conference
```

```r
# 1) Calculate p-values
# 2) Assign ranks and sort
# 3) BH Critical Value
# 4) Find critical value and select all p < score

q_value = 0.05

## 1)

feature_names = c()
p_values = c()

# For each data point calculate the p_value

for (i in 1:(ncol(emails)-1)) {
  colname = colnames(emails)[i]
  c_formula = paste(sep = "", colname, " ~ Conference")
  p_value = t.test(as.formula(c_formula), data = emails,
                   alternative="two.sided")

  feature_names = c(feature_names, colname)
  p_values = c(p_values, p_value$p.value)
}

## 2)

# Sorting
bh_entries = data.frame(feature_names, p_values)
bh_entries = bh_entries[order(bh_entries$p_values, decreasing = FALSE),]
rownames(bh_entries) = NULL

## 3)

# Define the function for the CV-Score
getCV_BH = function(i, m, Q) {
  return(i/m*Q)
}

cv_scores = c()

for (j in 1:nrow(bh_entries)) {
  current_val = getCV_BH(as.numeric(rownames(bh_entries)[j]),
                         nrow(bh_entries), q_value)
  cv_scores = c(cv_scores, current_val)
}

#cv_col = apply(bh_entries, 1, function(x)
#getCV_BH(as.numeric(rownames(bh_entries)), nrow(bh_entries), q_value))
bh_entries = data.frame(bh_entries, cv_scores)

## 4)
## Find all rows where p_values < cv_scores
bh_entries = bh_entries[bh_entries$p_values < bh_entries$cv_scores,]
```

```r
kable(bh_entries)
```

Bibliography