Lab 02 Block 02 | Machine Learning - LiU

Annalena Erhard (anner218), Héctor Plata (hecpl268), Maximilian Pfundstein (maxpf364)

2018-12-17

Contents

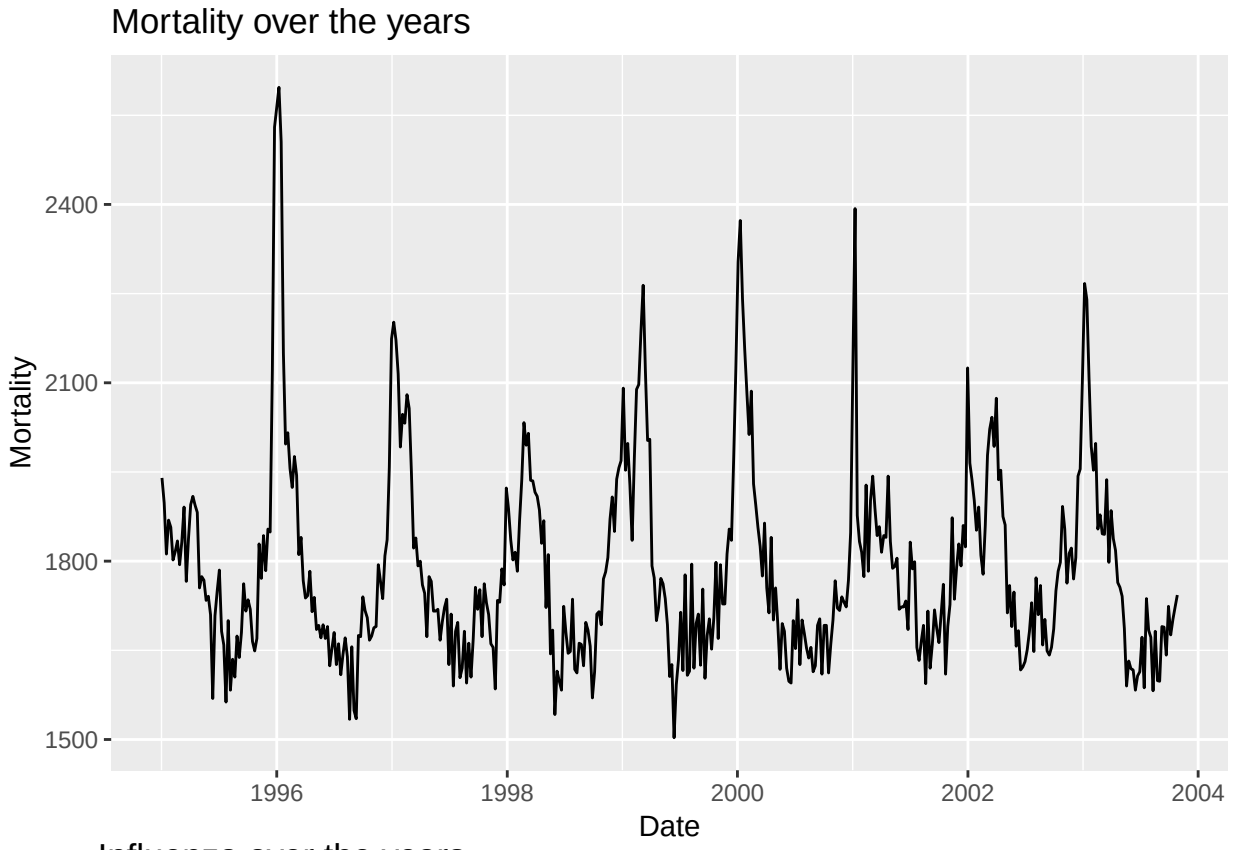
Assignment 1

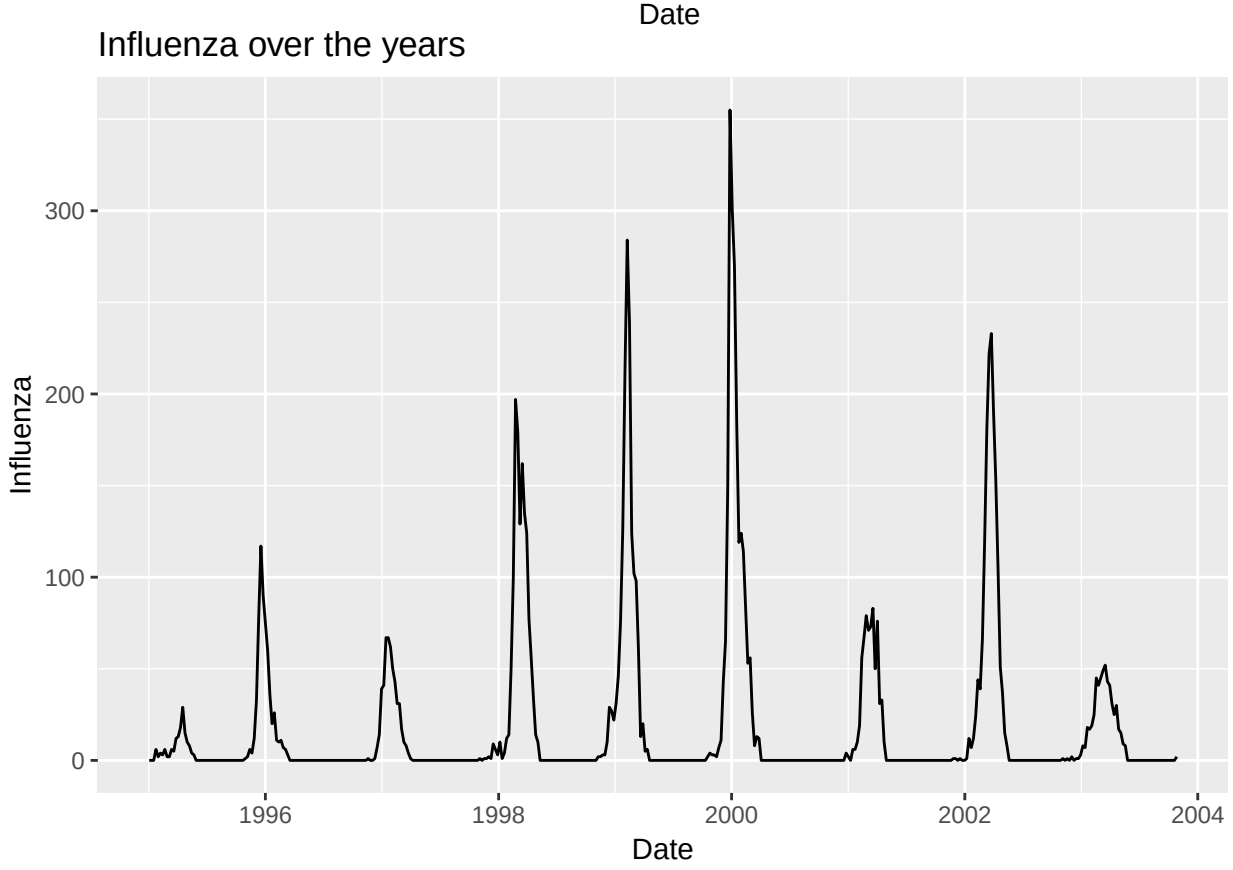
The excel document **influenza.xlsx** contians weekly data on the mortality and the number of laboratory-confirmed cases of influenza in Sweden. In addition, there is information about population-weighted temperature anomalies (temperature deficits).

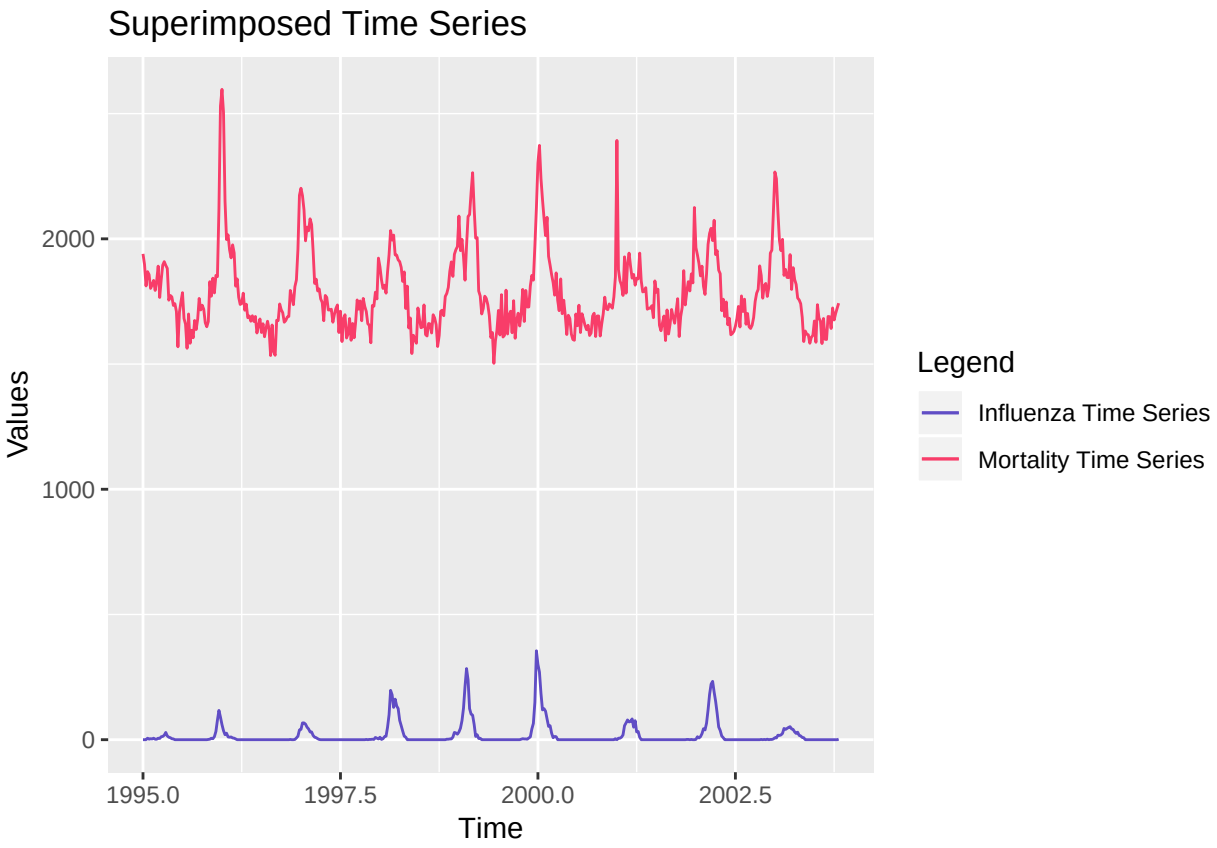
Task 1.1

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: From the plots shown below we see that the peaks of influenza cases are positioned at the same time as the peaks for moratlity rates. Which could possible mean in this case that the number of influenza cases drives the mortality for this periods.


Mortality over the years
Mortality
2400
2100
1800
1500
1996
1998
2000
2002
2004
Date


Influenza over the years
Influenza
300
200
100
0
1996
1998
2000
2002
2004
Date



Task 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.

Answer: The probabilistic model can be seen below with the code for fitting the model.

$$Mortality \sim \mathcal{N}\left(\alpha + X_{year}\beta_{year} + f_{week}\left(X_{week}\right), \sigma^2\right)$$

Where,

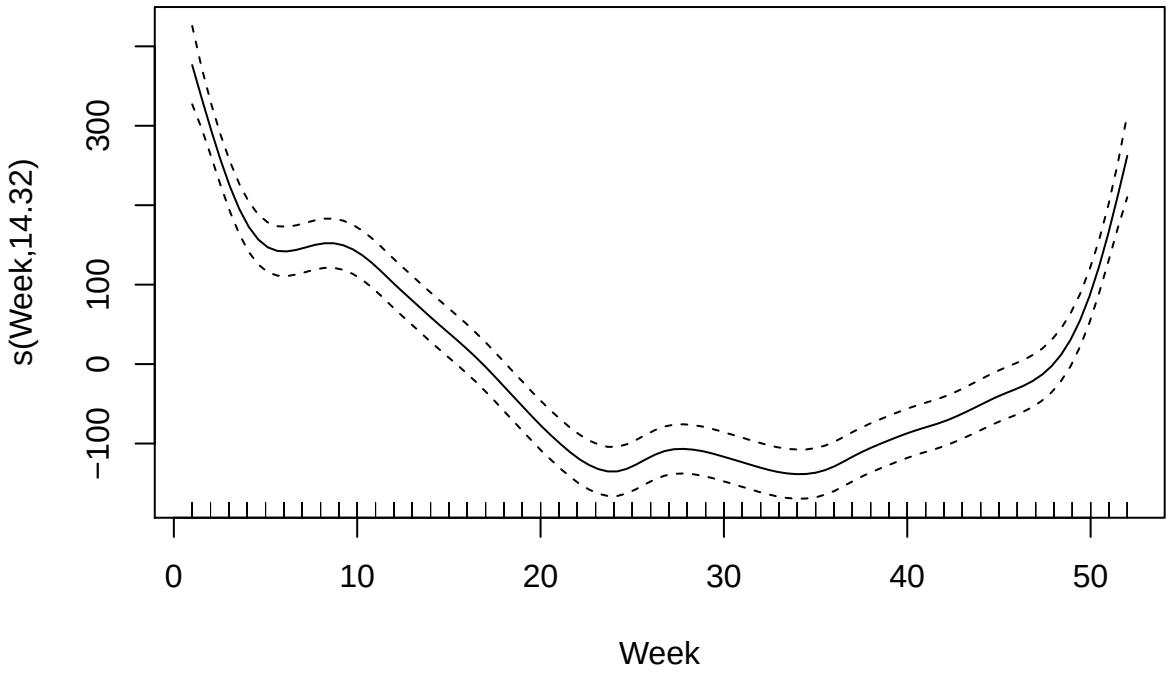
$$f_{week}\left(X_{week}\right) = \phi_{week}\left(X_{week}\right)\beta_{week}$$

```
############
# TASK 1.2 #
############

# Fitting the model.
reg = gam(Mortality ~ Year + s(Week, k=length(unique(data$Week))),
          data=data)
```

```
##
## Family: gaussian
## Link function: identity
##
```

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



Task 1.3

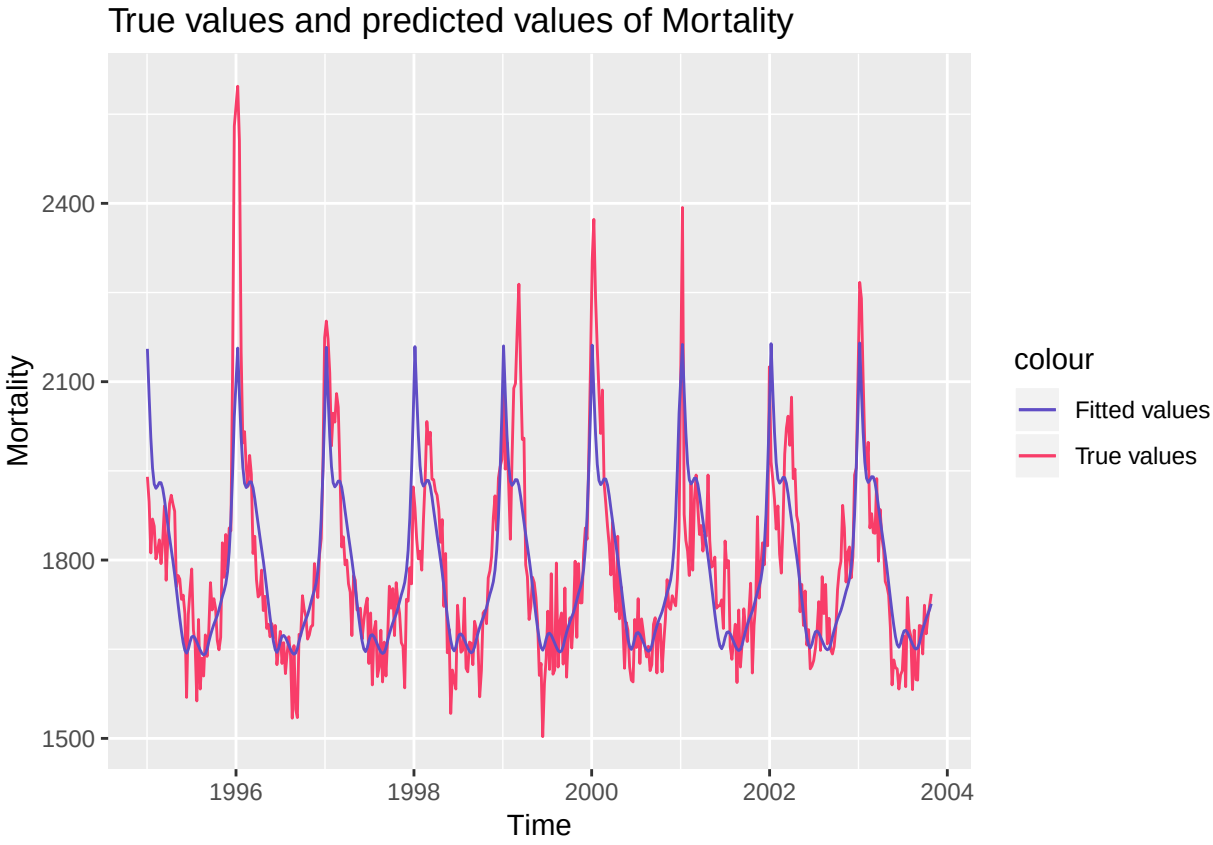
Task: Plot predicted and observed mortality against time for the fitted model and comment on the quality of the fit. Investigate the output of the `GAM` model and report which terms appear to be significant in the model. Is there a trend in mortality change from one year to another? Plot the spline component and interpret the plot.

Answer: It looks like the model is able to capture all the trends form the original data while avoiding the noise from the time series. From the looks of it, the model is a good fit for the data. The only term that is significant from the summary with a significance level of 1% is β_{week} which corresponds to the coefficient of the spline function. There doesn't seem to be a change in trend of mortality by year, since the plot doesn't show a change of behaviour and the linear coefficient for `Year` is not significant.

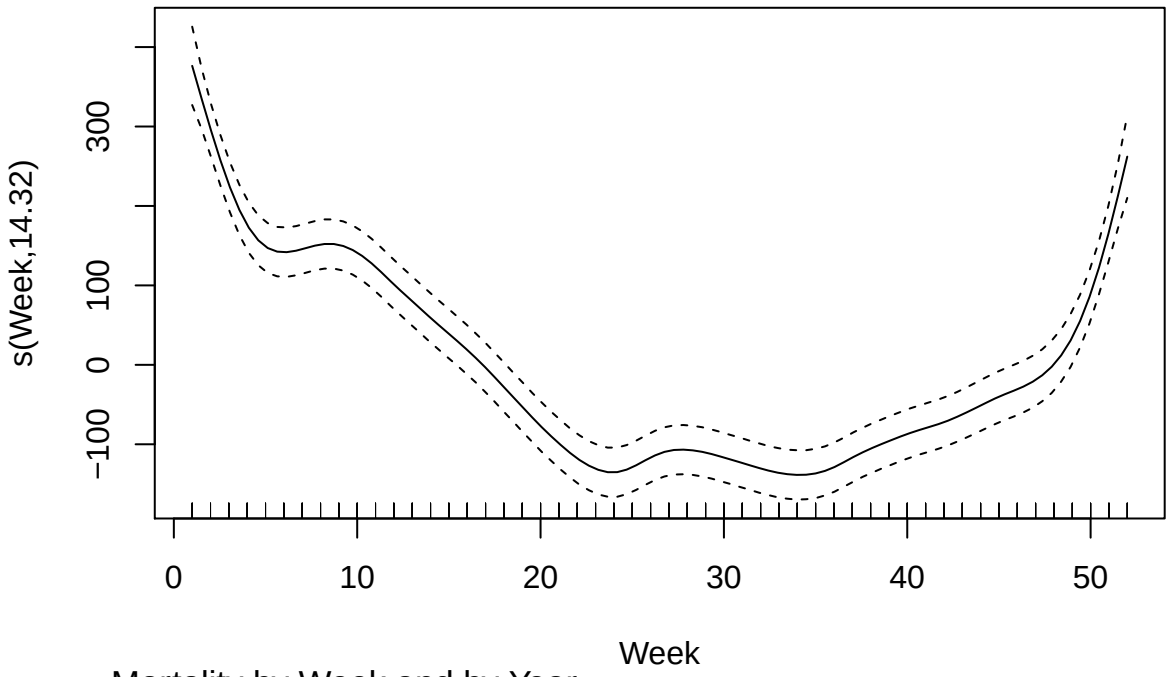
Below is presented the plot of the spline component and the mortality by `Week` for different values of `Year`. We see that the spline component mimics the average behaviour of the mortality given the `Week` by `Year`.

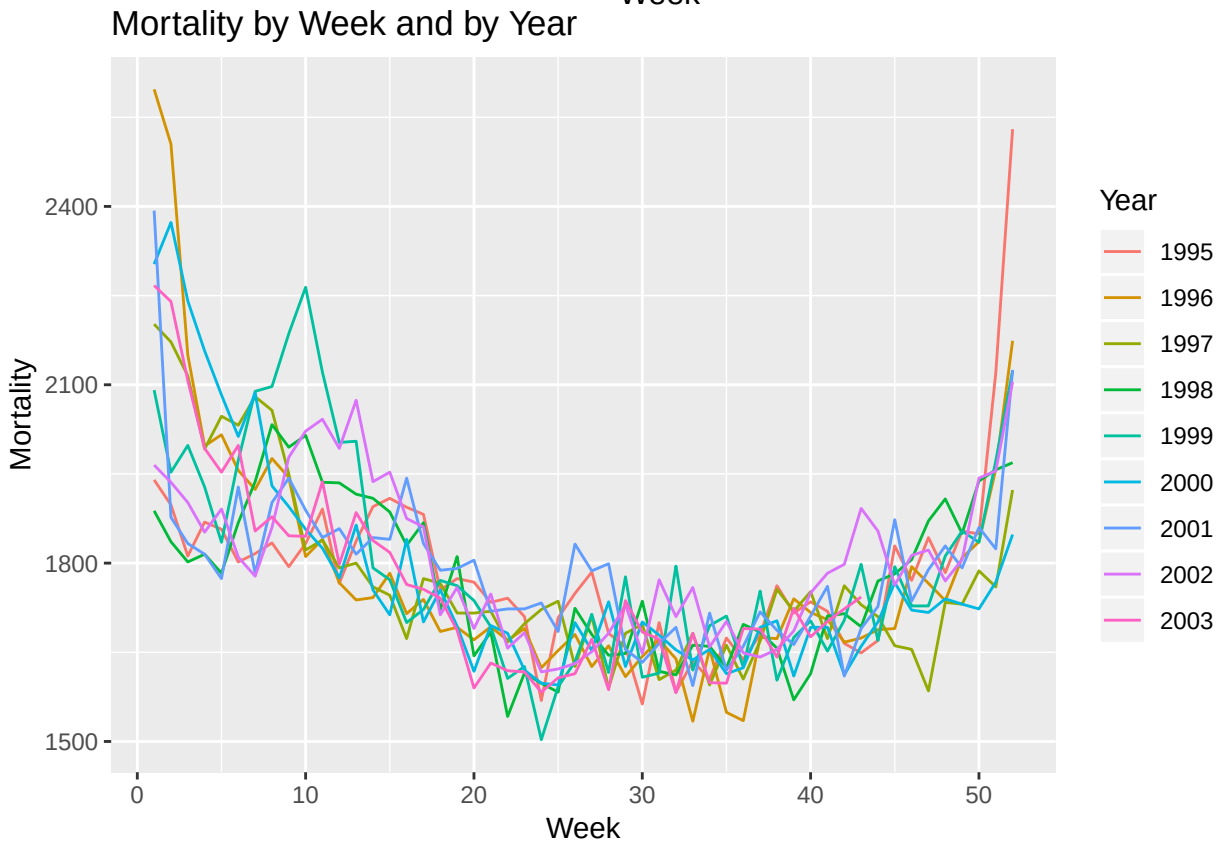
This shows that the mortality usually have a peak around the begining and end of the year while it decreases at the middle of it.

The influence of the smoothed variable week behaves almost like a parabola and mimics the course of the data over one year.




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





Task 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?

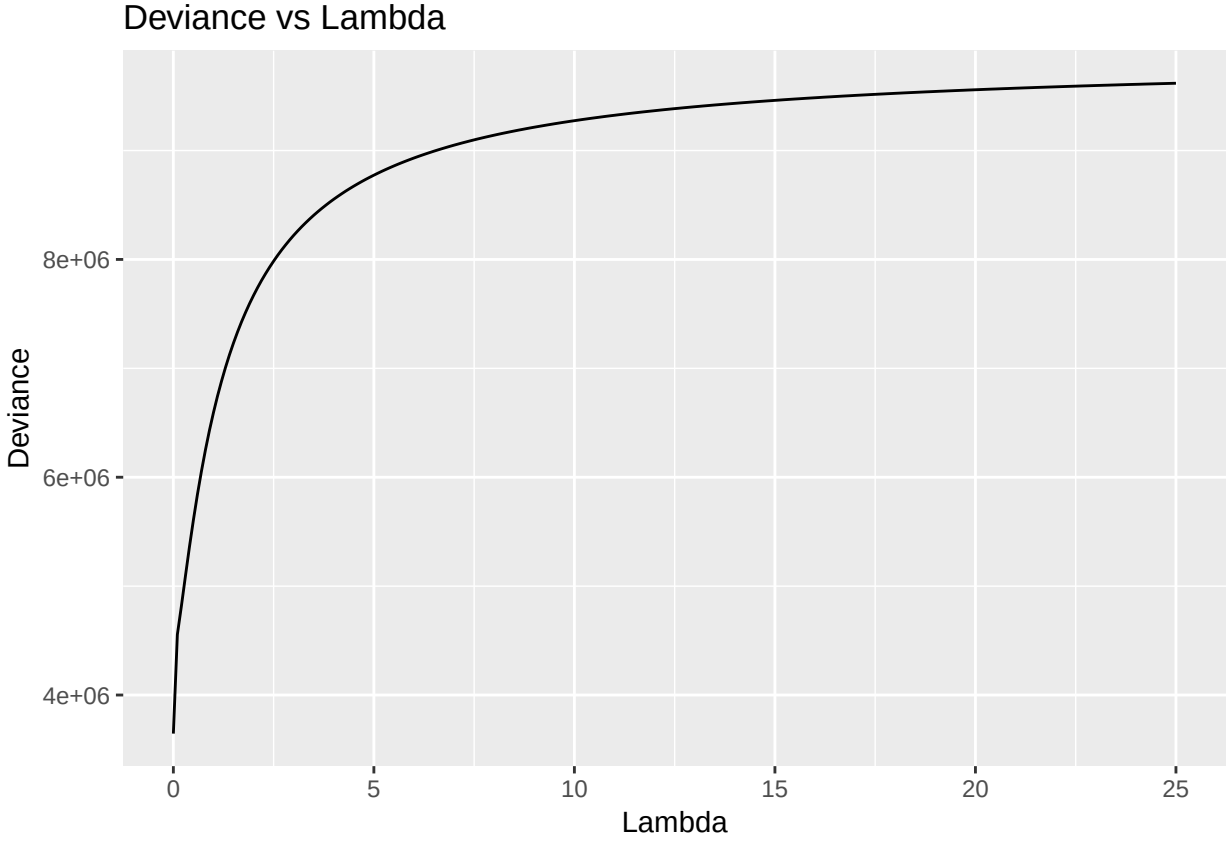
Do your results confirm this relationship?

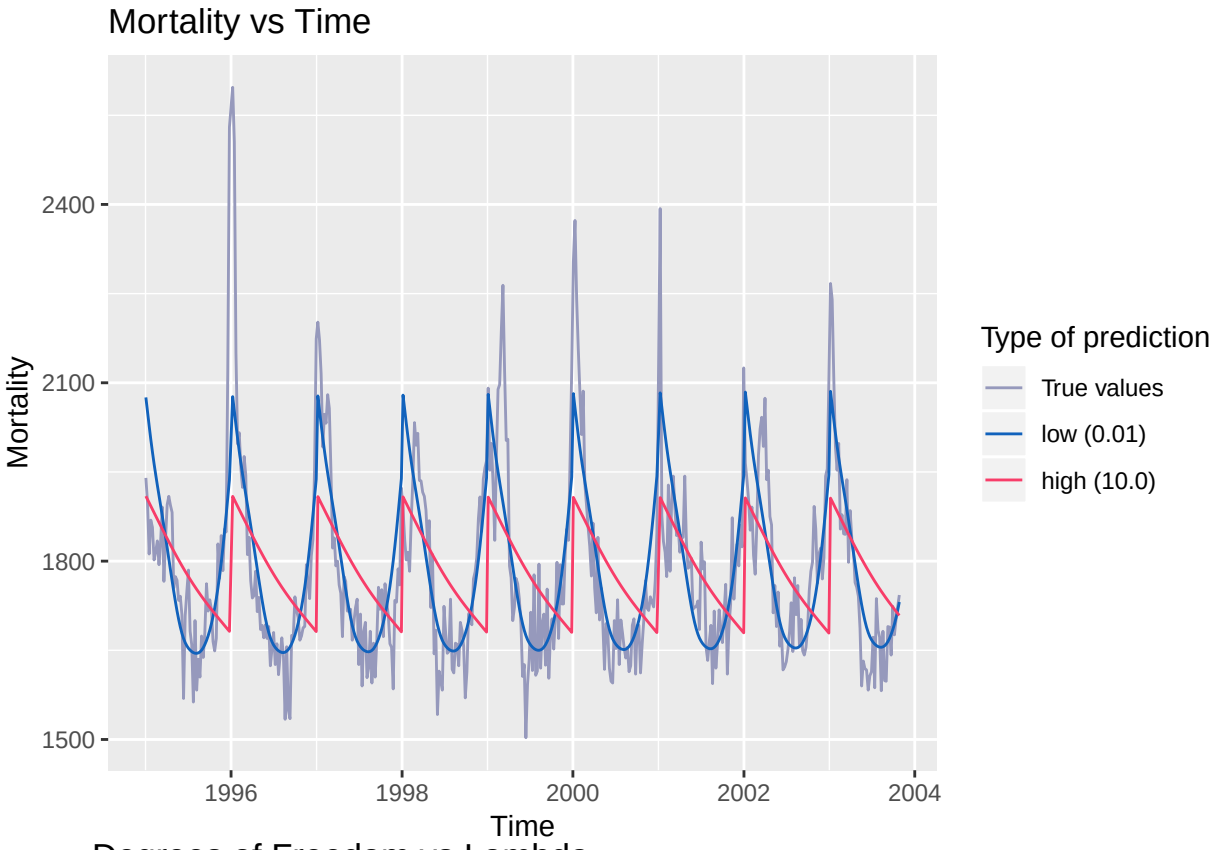
Answer: We see that the higher the λ factor the bigger the deviance of the model, which is expected as it's making the model less complex and it increases the error (deviance). As for the two different models with different penalty factors, it's visible from the second plot that λ plays an important role on the flexibility of the model in general. A high lambda will penalize too hard the curve, while a low lambda will let the model follow more closely the trends and movement of the curve.

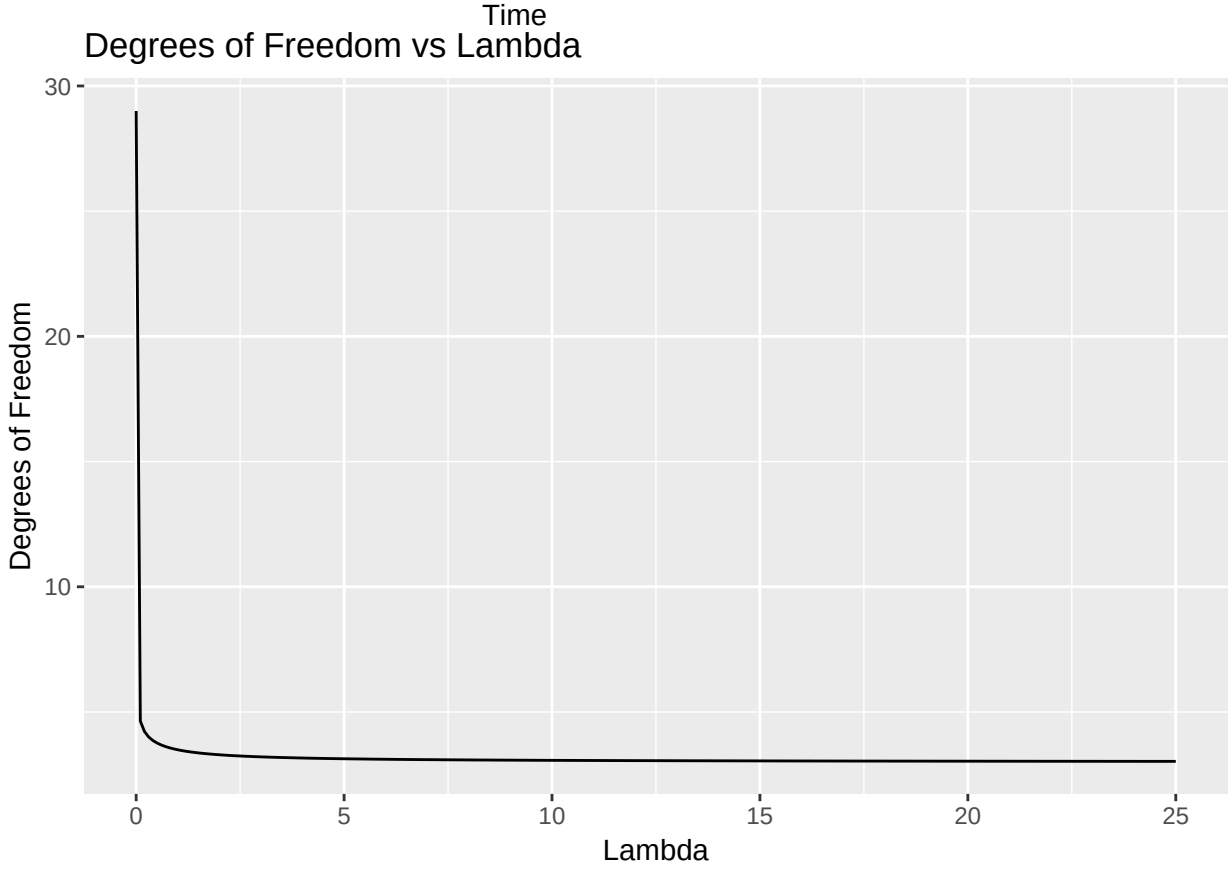
Finally the relationship of the degrees of freedom with respect to λ is given by the following formula.

$$df_\lambda = \sum_{k=1}^{N} \frac{1}{1 + \lambda d_k}$$

It's visible from the formula that the greater the λ the lower the degrees of freedom. This relationship can be seen on the last plot, where the effective degrees of freedom are shown against the penalty factor. We see a rapid decrease of the degrees which confirms this relationship.



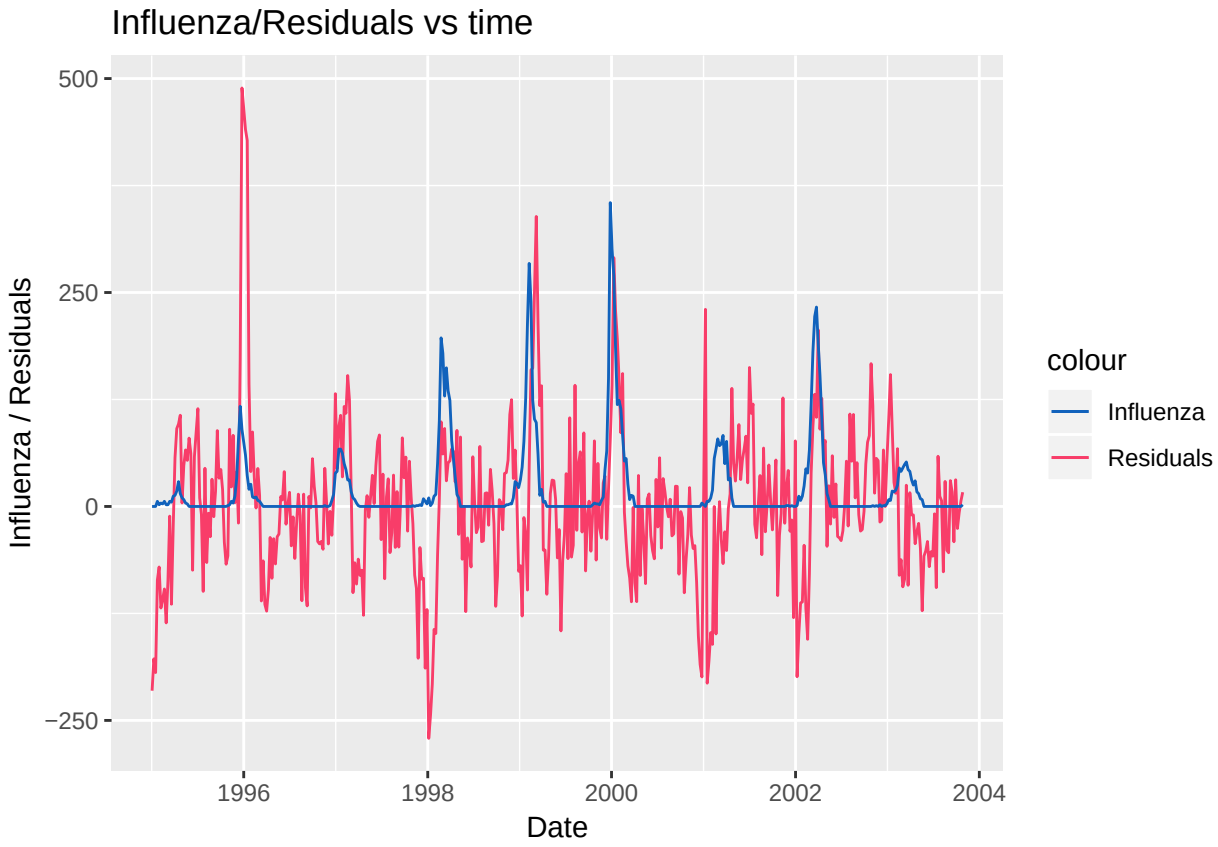

Mortality vs Time
Mortality
2400
2100
1800
1500
1996
1998
2000
2002
2004
Time
Type of prediction
True values
low (0.01)
high (10.0)


Degrees of Freedom vs Lambda
Degrees of Freedom
30
20
10
0
5
10
15
20
25
Lambda

Task 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?

Answer: From the plot shown below it seems that there is some temporal correlation between the residuals and the outbreaks of influenza. This is expected since it was seen on task 1.1 that the mortality and outbreaks of influenza are closely related and that the variable `Influeza` was not added to our linear model. So, the residuals must show this pattern that is not explain solely by the years and week number.

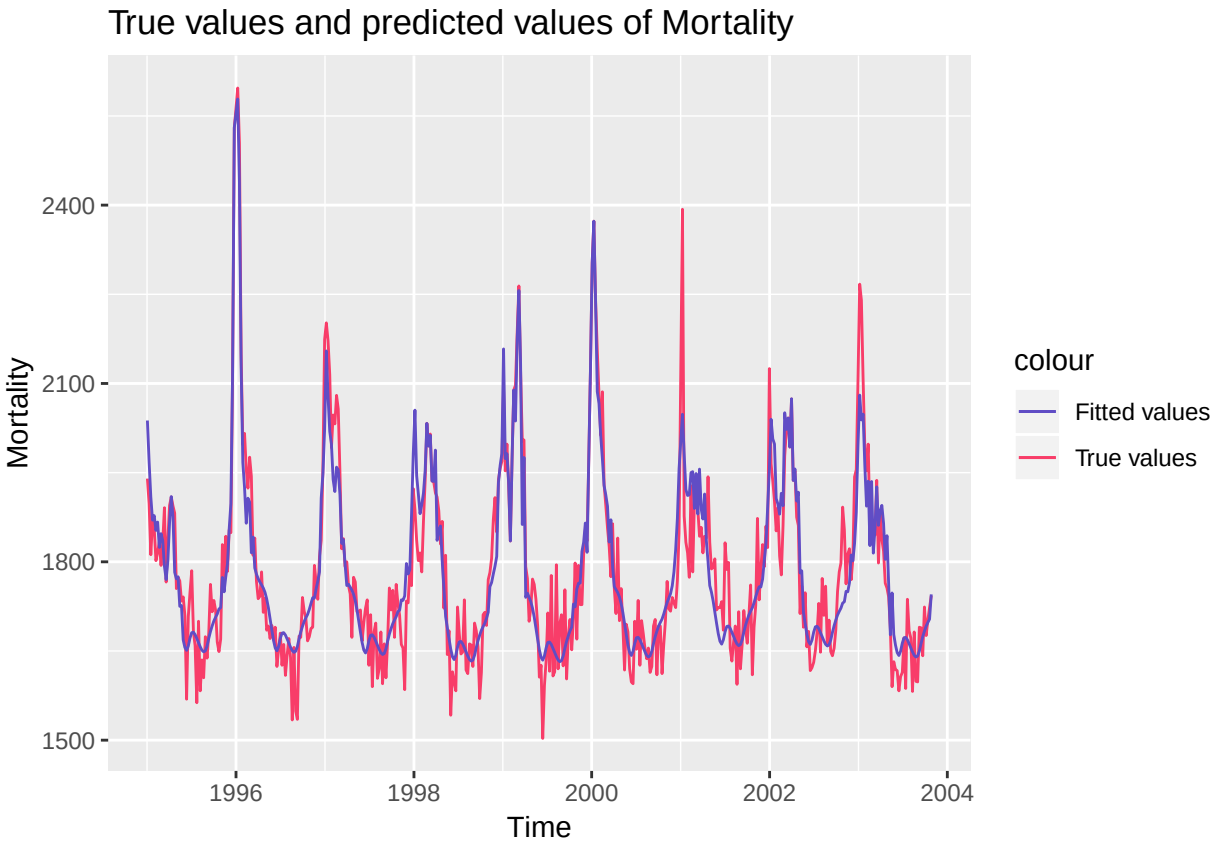


Task 1.6

Task: Fit a `GAM` model in R in which moratlity 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.

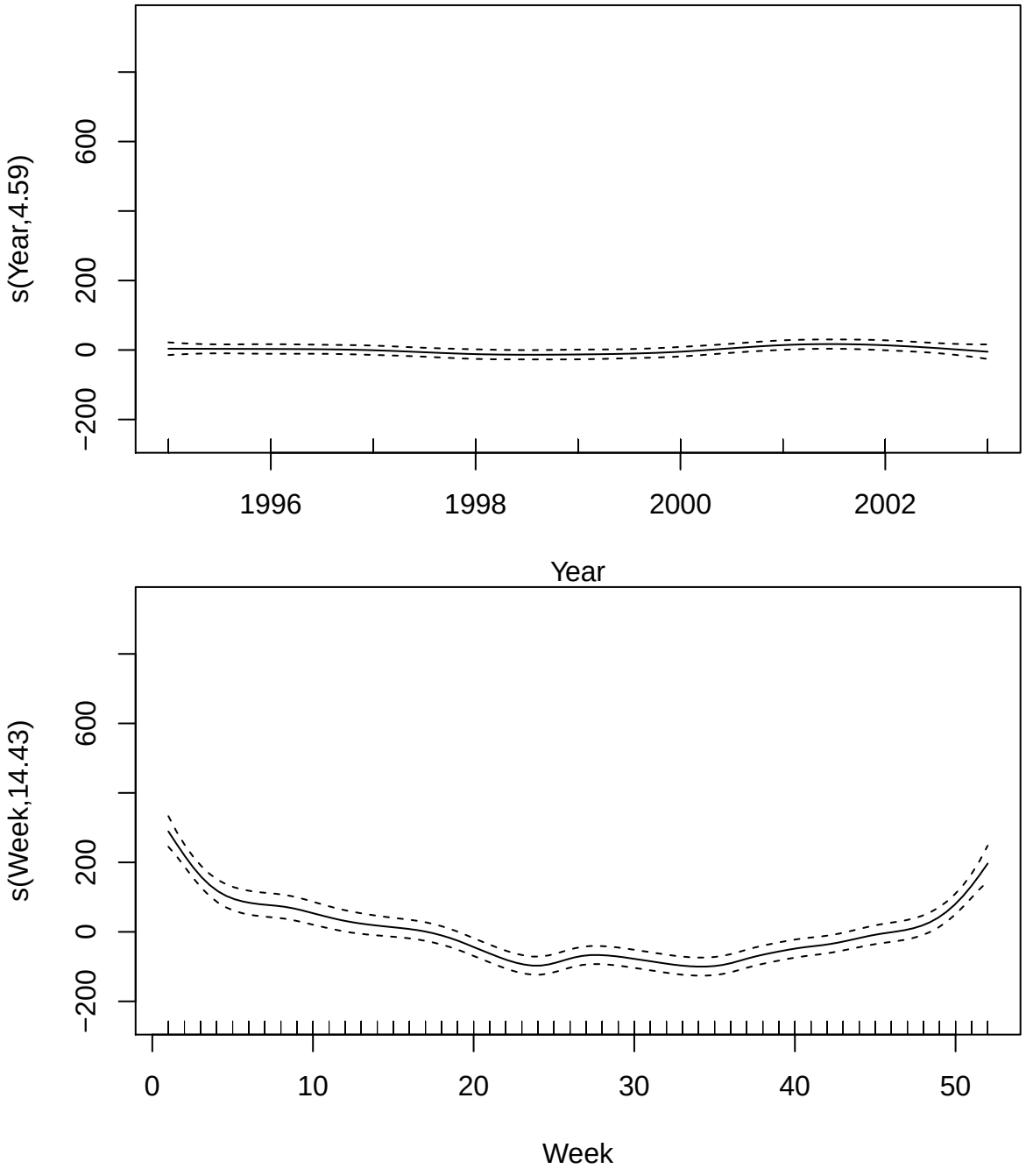
Answer: From the first plot we see that the model now present a better fit than the last model by incorporating the years and the influenza as a spline. From the summary we can also see that the year is not statistically significant which is congruent with the result shown previously. While the variables `Week` and `Influenza` are statistically significant. These two facts can be seen from the spline components. The components from `Year` are almost zero, while for `Week` and for `Influenza` are different and follows a pattern which is descriptive to the model. From the last two plots we can see that the residuals no longer follows the pattern of the influenza since this pattern is being captured by the new spline and that the residuals seems to behave as a normal distribution, which indicates that the model is a good fit. With all this information taken

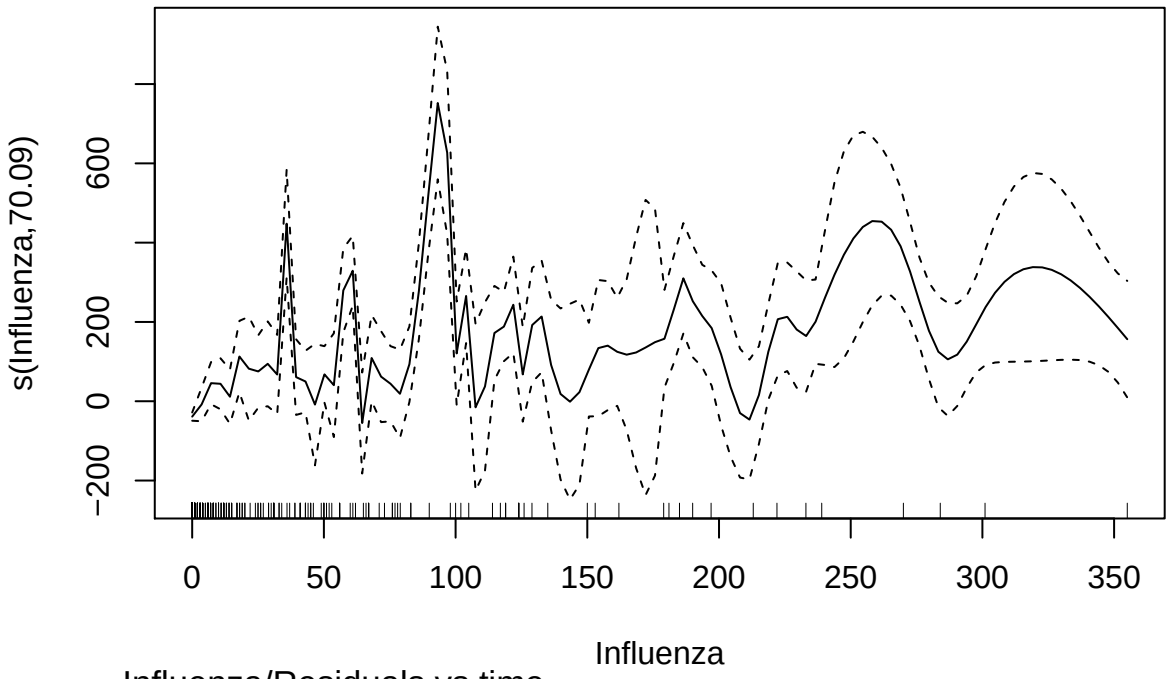
into account, is possible to conclude that `Influenza` seems to influence the `Mortality` in Sweden and that this model is a better fit compared to the one found in task 1.3 since it has a better R^2 (0.819 vs 0.677).

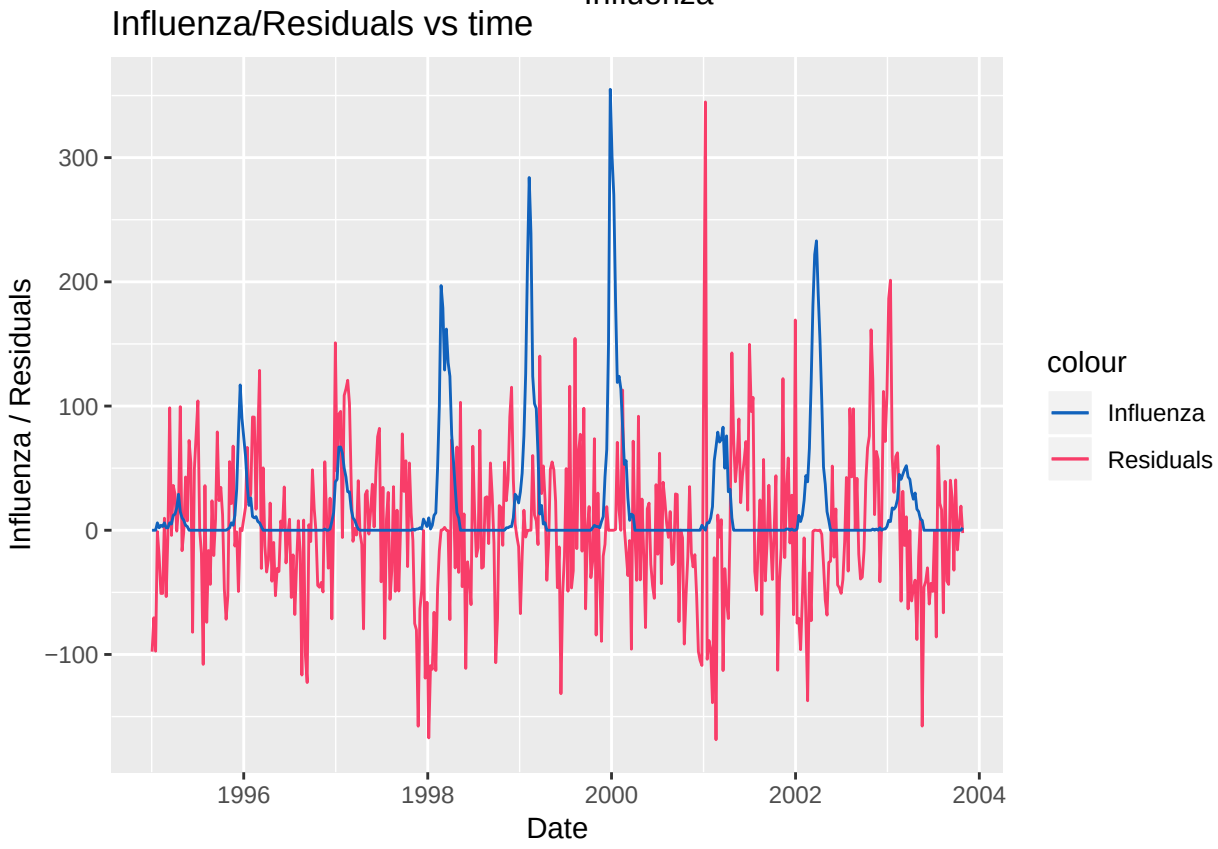


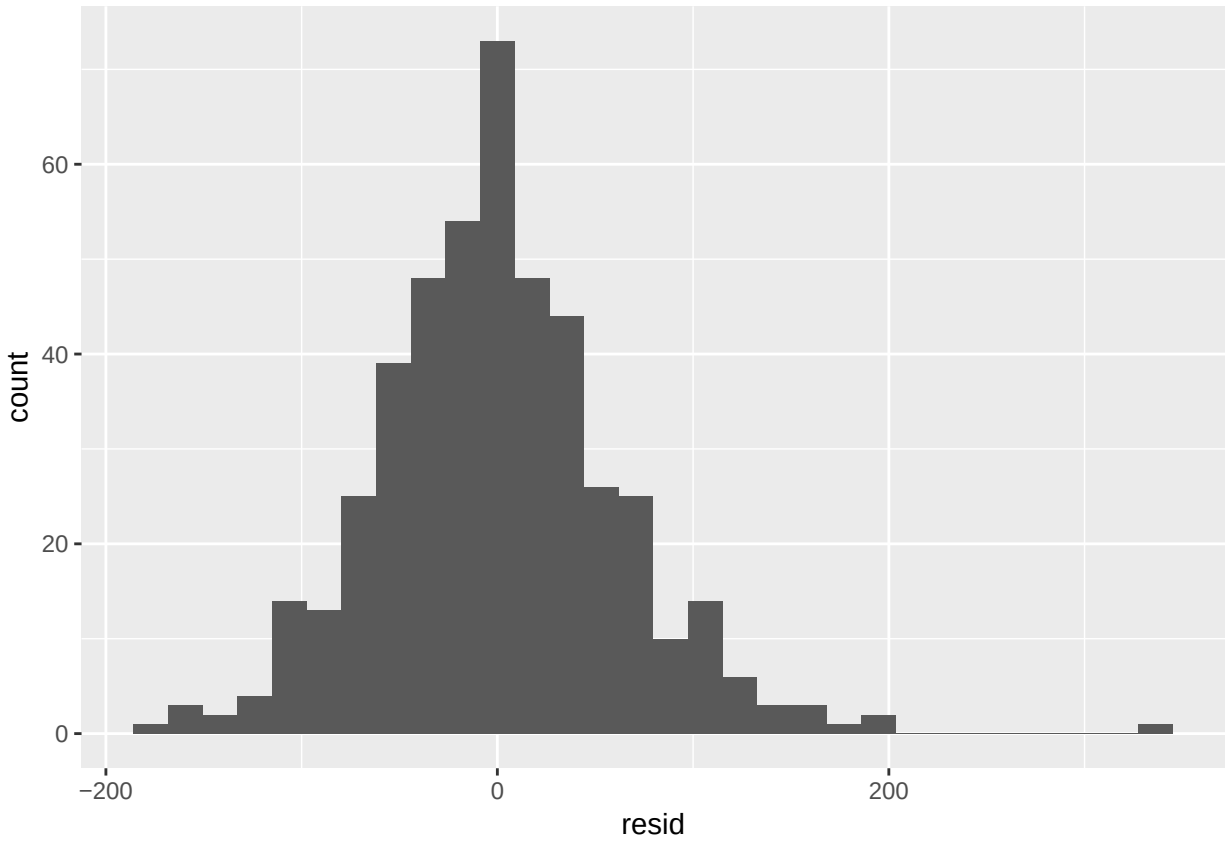

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ s(Year, k = length(unique(data$Year))) + s(Week,
##     k = length(unique(data$Week))) + s(Influenza, k = length(unique(data$Influenza)))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1783.765      3.198   557.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df      F p-value
## s(Year)      4.587  5.592  1.500   0.178
## s(Week)     14.431 17.990 18.763  <2e-16 ***
## s(Influenza) 70.094 72.998  5.622  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 134/144
## R-sq.(adj) =  0.819   Deviance explained = 85.4%
```

```
## GCV = 5840.5  Scale est. = 4693.7    n = 459
```




s(Influenza,70.09)
−200
0
200
600
0
50
100
150
200
250
300
350
Influenza


Influenza/Residuals vs time
Influenza / Residuals
300
200
100
0
−100
1996
1998
2000
2002
2004
Date
colour
Influenza
Residuals



Assignment 2

The data file **data.csv** contains information about 64 em-mails which were manually collected from DBWorld mailing list. They were classified as: 'announces of conferences' (1) and 'everything else' (0) (variable Conference)

Task 2.1

Task: 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.

Answer: We see that the optimal value for the threshold is around 1 and 2. In this case, the value 1.25 was selected. The number of features selected given this threshold is 231. These features can be seen below on the centroid plot. This plot shows the shrunken differences d'_{ik} for feature i and class k. This difference can be thought of as a distance measure of the intra-feature centroid per class $\overline{x}_{ik}$ versus the class centroid $\overline{x}_i$. So, this plot shows the features which have the biggest distance between the overall mean of the feature versus the intra-class mean of the feature. So, for example, we can see that `papers` is the variable that has the biggest difference between observations that came from a `Conference` email and observations who doesn't, since it has the largest shrunken difference per class.

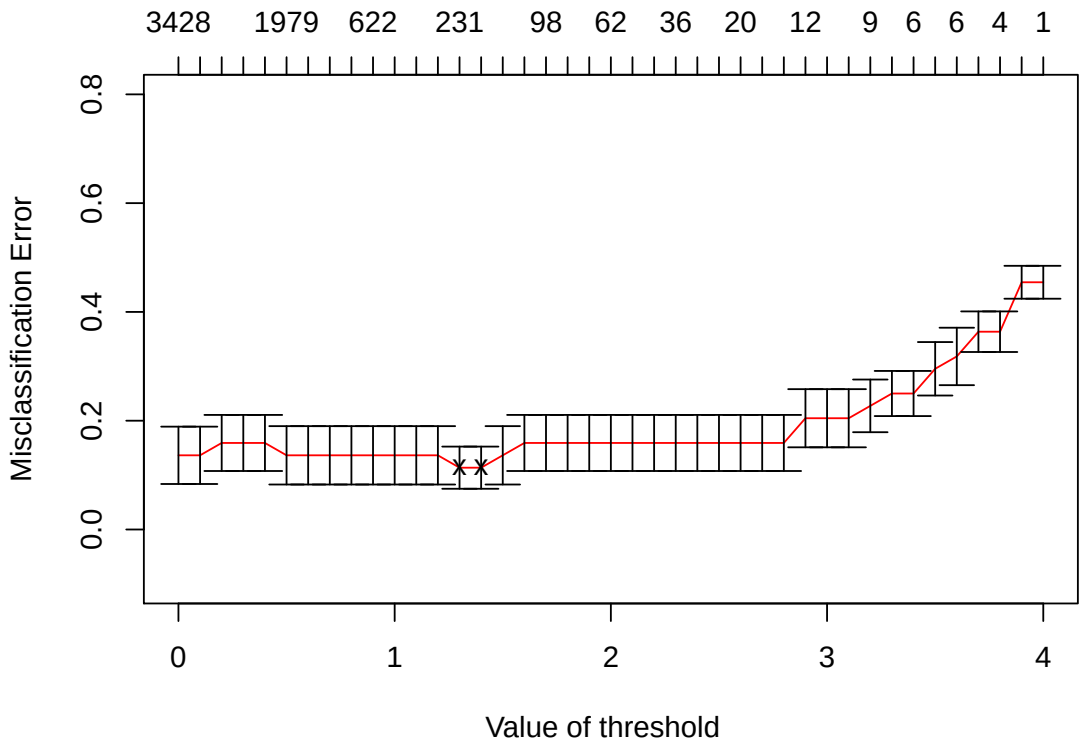
The top contributing features are presented on the table 1. It seems reasonable that these features will have a strong effect on the discrimination between the conference mails and other mails since it present terms related

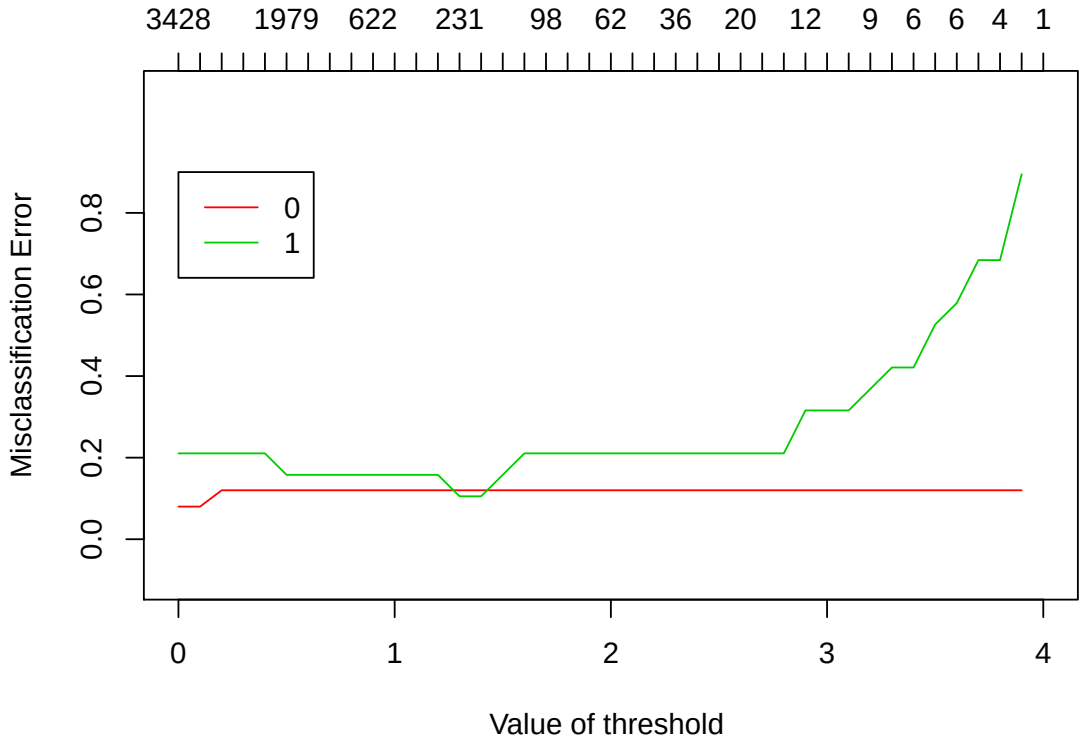
with `Conferences`, like `papers`, `published` and even `conference` itself. The classifier NSC have a really good performance in the sence that it has an FPR of 0, a missclasification of 10% and a precision of 100%.

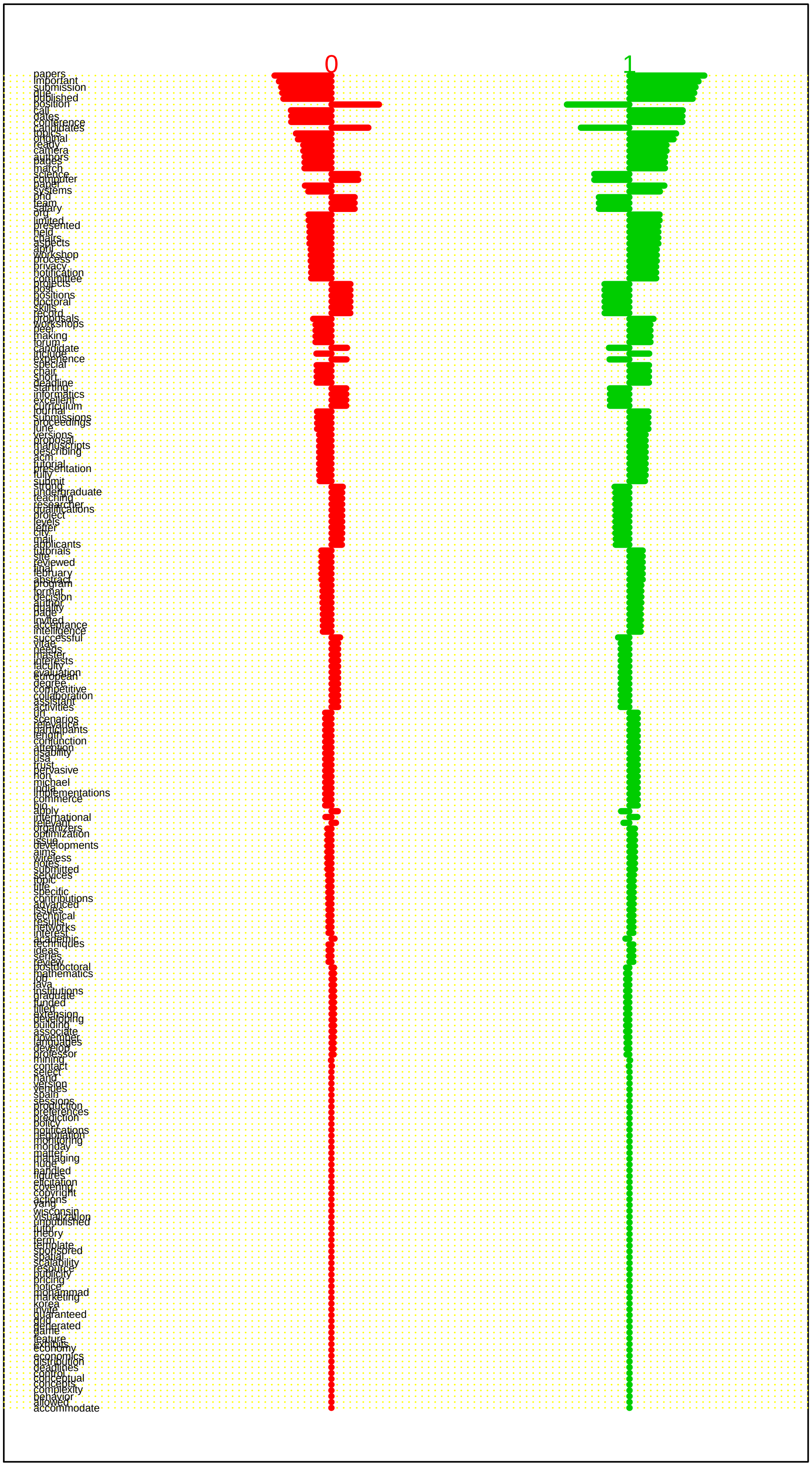
Selected the threshold by:

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

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


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


0
1

```
##        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
```

Table 1: Top contributing features

Features
papers
important
submission
due
published
position
call
conference
dates
candidates

Table 2: Confusion matrix of the traint set

True / Predicted	Not Conference	Conference
Not Conference	22	3
Conference	1	18

Table 3: Confusion matrix of the test set

True / Predicted	Not Conference	Conference
Not Conference	10	0
Conference	2	8

Table 4: Metrics of the train / test set

	Train	Test
Recall	0.9473684	0.8
Precision	0.8571429	1.0
Missclasification	0.0909091	0.1

	Train	Test
Accuracy	0.9090909	0.9
FPR	0.1200000	0.0

Task 2.2

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

a. Elastic net with the binomial response and $\alpha = 0.5$ in which penalty is selected by the cross-validation

b. Support vector machine with "vanilladot" kernel.

Compare the results of these models with the results of the nearest shrunken centroids (make a comparative table). Which model would you prefer and why?

Answer: In this case the best model given the test set is SVM because it outperforms every other model. However it uses all of the features while NSCM have a similar performance but only uses 231 features out of 4703. In this case, the more favorable model is SVM because so far the calculations are not so computational heavy, however, if the data set where to increase and it needs to be deployed on a computer with little resources, NSCM would be the best model overall.

Table 5: Comparison between all models

	NSCM Train	NSCM Test	Elastic Net Train	Elastic Net Test	SVM Train	SVM Test
Recall	0.9473684	0.8	0.6315789	0.6	1.0000000	0.90
Precision	0.8571429	1.0	0.8000000	1.0	0.9500000	1.00
Missclasification	0.0909091	0.1	0.2272727	0.2	0.0227273	0.05
Accuracy	0.9090909	0.9	0.7727273	0.8	0.9772727	0.95
FPR	0.1200000	0.0	0.1200000	0.0	0.0400000	0.00
N_Features	231.0000000	231.0	32.0000000	32.0	4703.0000000	4703.00

Task 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: With a predefined false discovery rate (FDR) of 5% we end up selecting 39 features which are shown below with the respective $p - value$ plots and the BH rejection threshold. We basically are testing multiple hypothesis at the same time and we know the following bound exist over the real false discovery rate.

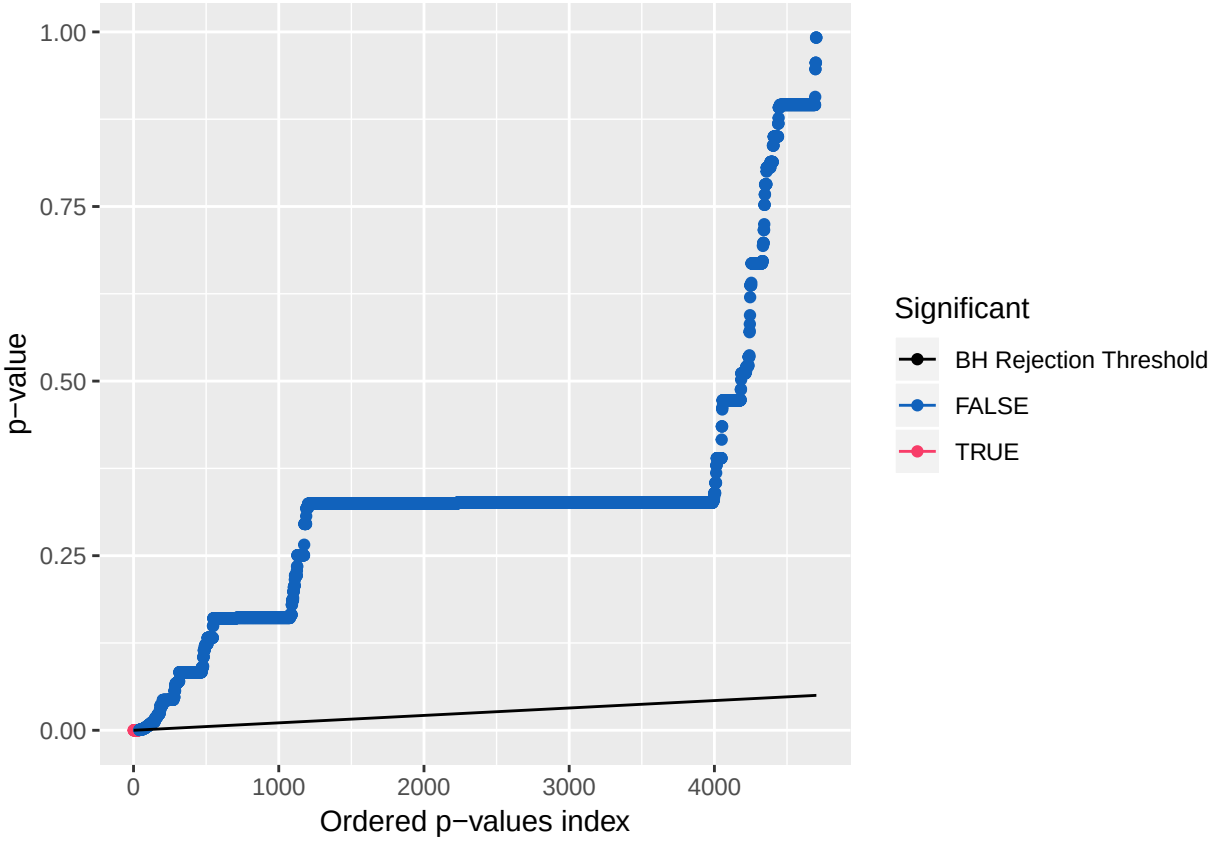
$$FDR \leq \alpha \frac{M_0}{M} \leq \alpha$$

With this bound in mind we create a set of significant features which is defined as follows:

$$S = \left\{ j : p_{(j)} < \alpha \frac{j}{M} \right\}$$

Where $p_{(j)}$ is the p-value of the ordered feature j. So this set of features are those who, under the assumptions of the t-test, have a statistically different mean for the observations that belong to a `Conference` and those who doesn't. We see that the selected features are coherent with was told above. For example, the word `phd` or `workshop` are strongly related with a `Conference` and not so much with other type of emails, which means

that one would expect a difference in mean under the two populations (one that came from conferences and the other who doesn't).



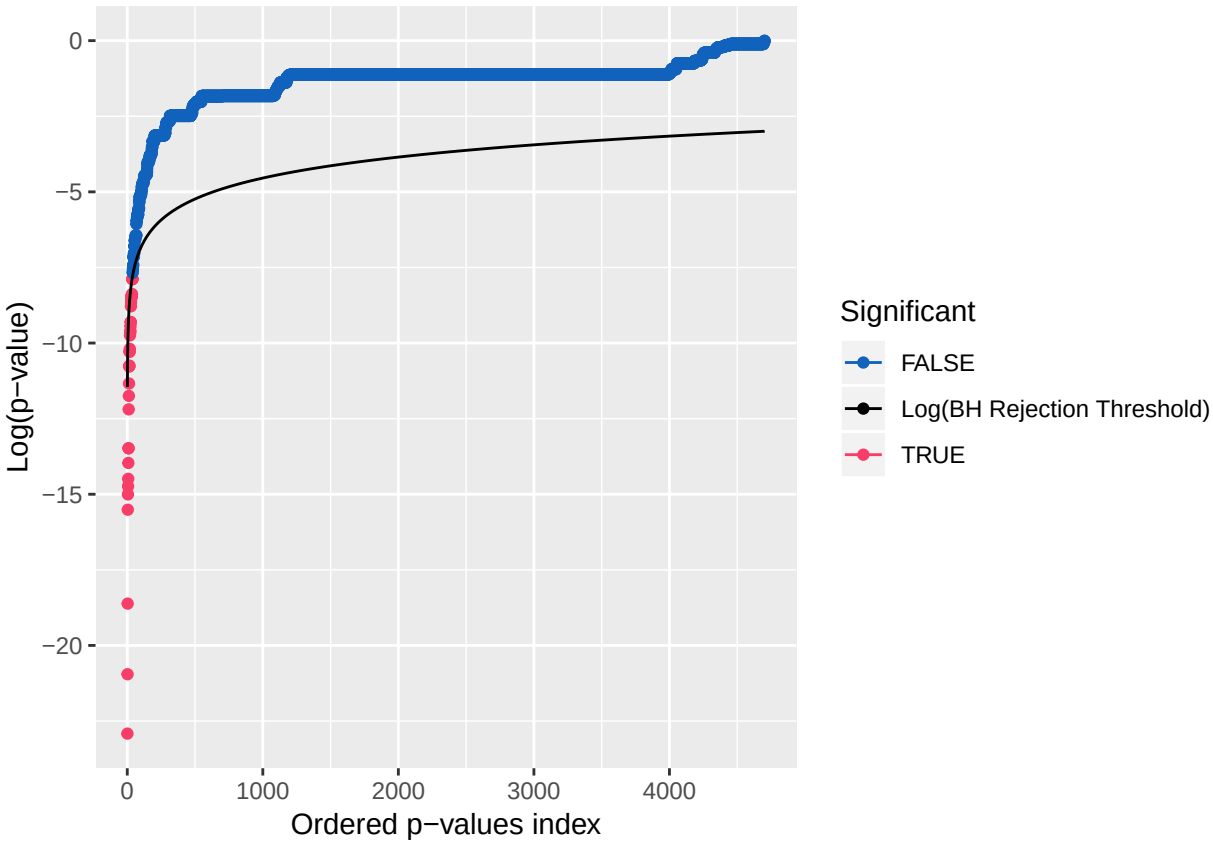


Table 6: Features selected by the Benjamini-Hochberg method

Features
papers
submission
position
published
important
call
conference
candidates
dates
paper
topics
limited
candidate
camera
ready
authors
phd
projects
org
chairs
due
original
notification

Features
salary
record
skills
held
team
pages
workshop
committee
proceedings
apply
strong
international
degree
excellent
post
presented

Appendix

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

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

# Loading the data and preprocessing.
data = read_xlsx("Influenza.xlsx")
data$Date = as.Date(paste(data$Year, data$Week, 1, sep="-"), "%Y-%U-%u")


# Plotting the time dependancy of
# influenza and moratlity rate.
p = ggplot() +
    geom_line(aes(x=data$Date, y=data$Mortality)) +
    labs(x='Date',
         y='Mortality',
         title='Mortality over the years')

print(p)

p = ggplot() +
    geom_line(aes(x=data$Date, y=data$Influenza)) +
    labs(x='Date',
```

```
         y='Influenza',
         title='Influenza over the years')

print(p)

ggplot(data) +
  geom_line(aes(x = data$Time, y = data$Mortality,
                colour = "Mortality Time Series")) +
  geom_line(aes(x = data$Time, y = data$Influenza,
                colour = "Influenza Time Series")) +
  labs(title = "Superimposed Time Series", y = "Values",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("#604dc5","#f83d69"))

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

# Fitting the model.
reg = gam(Mortality ~ Year + s(Week, k=length(unique(data$Week))),
          data=data)


summary(reg)
plot(reg)

#############
# TASK 1.3 #
#############

# Getting the fitted values and
# creating a plot of
# y vs yhat.
y_hat = fitted(reg)

# Plotting the real values and the fitted ones.
p = ggplot() +
    geom_line(aes(x=data$Date, y=data$Mortality, color='True values')) +
    geom_line(aes(x=data$Date, y=y_hat, color='Fitted values')) +
    scale_colour_manual(values=c("#604dc5","#f83d69")) +
    labs(x='Time',
         y='Mortality',
         title='True values and predicted values of Mortality')

print(p)

# Output of the regression object.
summary(reg)

# Plotting the spline component.
plot(reg)

# Plotting the mortality by week and by year.
```

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

print(p)

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

# Creating grid for the penalty factor.
lambdas = seq(0.0 , 25, 0.1)

# Data frame that is going to hold
# the information of all the models.
results = NULL

for (lambda in lambdas)
{
  # Fitting a GAM for each penalty factor lambda.
  g_reg = gam(Mortality ~ Year + s(Week,
                                   k=length(unique(data$Week)),
                                   sp=lambda),
              data=data)

  # Adding the information to the data frame.
  results = rbind(results, data.frame(list(lambda=lambda,
                                           deviance=g_reg$deviance,
                                           df=sum(influence(g_reg)))))
}

# Plotting the deviance vs lambda.
p = ggplot() +
    geom_line(aes(x=results$lambda, y=results$deviance)) +
    labs(x='Lambda',
         y='Deviance',
         title='Deviance vs Lambda')


print(p)

# High vs low penalty factors.
low_reg = gam(Mortality ~ Year + s(Week,
                                   k=length(unique(data$Week)),
                                   sp=0.01),
              data=data)

high_reg = gam(Mortality ~ Year + s(Week,
```

```
                                    k=length(unique(data$Week)),
                                    sp=10),
               data=data)

preds = NULL
preds = rbind(preds, data.frame(list(y=data$Mortality,
                                     x=data$Date,
                                     type='True values')))
preds = rbind(preds, data.frame(list(y=fitted(low_reg),
                                     x=data$Date,
                                     type='low (0.01)')))
preds = rbind(preds, data.frame(list(y=fitted(high_reg),
                                     x=data$Date,
                                     type='high (10.0)')))


# Plotting the predictions with low/high penalty
# and the real values.

p = ggplot() +
    geom_line(aes(x=preds$x, y=preds$y, colour=preds$type)) +
    scale_colour_manual(values=c("#9699bc", "#1162bc", "#f83d69")) +
    labs(x='Time',
         y='Mortality',
         colour='Type of prediction',
         title='Mortality vs Time')

print(p)

# Plotting the deviance vs lambda.
p = ggplot() +
    geom_line(aes(x=results$lambda, y=results$df)) +
    labs(x='Lambda',
         y='Degrees of Freedom',
         title='Degrees of Freedom vs Lambda')

print(p)

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

# Getting the residuals.
resid = residuals(reg)

# Plotting the residuals and influenza vs the time.
p = ggplot() +
    geom_line(aes(x=data$Date,y=resid, colour='Residuals')) +
    geom_line(aes(x=data$Date,y=data$Influenza, colour='Influenza')) +
    labs(x='Date',
         y='Influenza / Residuals',
         title='Influenza/Residuals vs time') +
    scale_colour_manual(values=c("#1162bc", "#f83d69"))
```

```
print(p)

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

# Fitting the new model.
reg = gam(Mortality ~ s(Year, k=length(unique(data$Year))) +
                      s(Week, k=length(unique(data$Week))) +
                      s(Influenza, k=length(unique(data$Influenza))),
                      data=data)

# Getting the fitted values and
# creating a plot of
# y vs yhat.
y_hat = fitted(reg)

# Plotting the real values and the fitted ones.
p = ggplot() +
    geom_line(aes(x=data$Date, y=data$Mortality, color='True values')) +
    geom_line(aes(x=data$Date, y=y_hat, color='Fitted values')) +
    scale_colour_manual(values=c("#604dc5","#f83d69")) +
    labs(x='Time',
         y='Mortality',
         title='True values and predicted values of Mortality')

print(p)

# Output of the regression object.
summary(reg)

# Plotting the spline component.
plot(reg)

# Getting the residuals.
resid = residuals(reg)

# Plotting the residuals and influenza vs the time.
p = ggplot() +
    geom_line(aes(x=data$Date,y=resid, colour='Residuals')) +
    geom_line(aes(x=data$Date,y=data$Influenza, colour='Influenza')) +
    labs(x='Date',
         y='Influenza / Residuals',
         title='Influenza/Residuals vs time') +
    scale_colour_manual(values=c("#1162bc", "#f83d69"))

print(p)

# Plotting the histogram from the residuals.
p = ggplot() +
    geom_histogram(aes(resid))

print(p)
```

```
############
# TASK 2.1 #
############

# Loading the data and doing preprocessing.
data = read.csv('data.csv', sep=';', encoding='latin1')
data$Conference = as.factor(data$Conference)

# Traing / test split.
set.seed(12345)
n = dim(data)[1]
id = sample(1:n, floor(n * 0.7))
train = data[id,]
test = data[-id,]

rownames(train) = 1:nrow(train)
X_train = t(train[, -ncol(train)])
y_train = train$Conference

rownames(test) = 1:nrow(test)
X_test = t(test[, -ncol(test)])
y_test = test$Conference

# Doing weird stuff because that's what
# the library wants.
mydata_train = list(x=X_train,
                    y=y_train,
                    geneid=as.character(1:nrow(X_train)),
                    genenames=rownames(X_train))

mydata_test = list(x=X_test,
                    y=y_test,
                    geneid=as.character(1:nrow(X_test)),
                    genenames=rownames(X_test))

# Training the model and getting the best parameter by cv.
model = pamr.train(mydata_train, threshold=seq(0, 4, 0.1))
cv_model = pamr.cv(model, mydata_train)

th = cv_model$threshold[which.min(cv_model$error)]


cv_model$threshold[which.min(cv_model$error)]

pamr.plotcv(cv_model)

pamr.plotcen(model, mydata_train, threshold=th)

# Helper function to calculate the
# classification metrics given a
# confusion matrix.
# Rows (0, 1) must be from the classifier
# Columns (0, 1) must be true values.
```

```
analyze_cm = function(cm)
    {
        cm_df = as.data.frame(cm)
        recall = cm_df[4, "Freq"] / (cm_df[4, "Freq"] + cm_df[2, "Freq"])
        precision = cm_df[4, "Freq"] / (cm_df[4, "Freq"] + cm_df[3, "Freq"])
        accuracy = (cm_df[1, "Freq"] + cm_df[4, "Freq"]) / sum(cm_df[, "Freq"])
        mcr = 1 - accuracy
        fpr = cm_df[3, "Freq"] / (cm_df[1, "Freq"] + cm_df[3, "Freq"])

        return(list(Recall=recall,
                    Precision=precision,
                    Missclasification=mcr,
                    Accuracy=accuracy,
                    FPR=fpr))
}

# Helper function to get metrics from a confusion matrix.
get_metrics_cm = function(cm_train, cm_test)
{
  metrics = data.frame(analyze_cm(cm_train))
  metrics = rbind(metrics, analyze_cm(cm_test))
  rownames(metrics) = c("Train", "Test")

  return(t(metrics))
}

# Transforms a confusion matrix
# to a data frame. It assumes
# a 2 by 2 binary confusion matrix.
prettyfy_cm = function(cm_table)
{
  pretty_table = data.frame(list(titles=c("Not Conference", "Conference"),
                                 Bad=c(cm_table[1, 1], cm_table[2, 1]),
                                 Good=c(cm_table[1, 2], cm_table[2, 2])))

  colnames(pretty_table) = c("True / Predicted", "Not Conference", "Conference")

  return(pretty_table)
}

genes = pamr.listgenes(model, mydata_train, threshold=th)
top_genes = as.data.frame(colnames(data)[as.numeric(genes[1:10, 1])])
colnames(top_genes) = 'Features'
kable(top_genes, caption='Top contributing features')

# Getting the prediction.
yhat_train = pamr.predict(model, newx=X_train, threshold=th)
yhat_test = pamr.predict(model, newx=X_test, threshold=th)

# Computing the confusion matrixes.
cm_train = table(train$Conference, yhat_train)
cm_test = table(test$Conference, yhat_test)
```

```
# Pretty printing.
kable(prettyfy_cm(cm_train), caption='Confusion matrix of the traint set')
kable(prettyfy_cm(cm_test), caption='Confusion matrix of the test set')

kable(get_metrics_cm(cm_train, cm_test), caption='Metrics of the train / test set')

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

# Creating the train/test data.
X_train = as.matrix(train[, -ncol(train)])
y_train = train$Conference

X_test = as.matrix(test[, -ncol(test)])
y_test = test$Conference

# Creating the elastic net and SVM.
elastic = cv.glmnet(x=X_train, y=y_train, alpha=0.5, family="binomial")
svm = ksvm(X_train, y_train, kernel='vanilladot', scale=FALSE)

# Getting the confusion matrixes.
cm_train_elastic = table(y_train,
                         predict(elastic,
                                 newx=X_train,
                                 type='class'))
cm_test_elastic = table(y_test,
                        predict(elastic,
                                newx=X_test,
                                type='class'))

cm_train_svm = table(y_train,
                     predict(svm,
                             X_train,
                             type='response'))
cm_test_svm = table(y_test,
                    predict(svm,
                            X_test,
                            type='response'))

# Getting the number of features used
# by each algorithm.
elastic_coef = sum(coef(elastic, s='lambda.min') != 0) - 1
svm_coef = dim(data)[2]

# Getting the metrics for each algorithm.
nscm_metrics = get_metrics_cm(cm_train, cm_test)
colnames(nscm_metrics) = c("NSCM Train", "NSCM Test")
nscm_metrics = rbind(nscm_metrics,
                     N_Features=c(dim(genes)[1],
                                  dim(genes)[1]))

elastic_metrics = get_metrics_cm(cm_train_elastic, cm_test_elastic)
```

```
colnames(elastic_metrics) = c("Elastic Net Train", "Elastic Net Test")
elastic_metrics = rbind(elastic_metrics,
                        N_Feature=c(elastic_coef,
                                    elastic_coef))

svm_metrics = get_metrics_cm(cm_train_svm, cm_test_svm)
colnames(svm_metrics) = c("SVM Train", "SVM Test")
svm_metrics = rbind(svm_metrics,
                    N_Feature=c(svm_coef,
                                svm_coef))

all_metrics = cbind(nscm_metrics, elastic_metrics, svm_metrics)



kable(all_metrics, caption="Comparison between all models")

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

# Variable that is going to store
# all of the p-values of each feature.
p_values = c()

# Calculating all of the p-values given a t-test.
for (i in 1:(ncol(data) - 1))
{
  test = t.test(data[, i] ~ data$Conference)
  p_values = c(p_values, test$p.value)
}

# Ordering the p-values.
ordered_idxs = base::order(p_values)
ordered_pvalues = p_values[ordered_idxs]

# Defining the BJ rejection threshold.
bh_rejection = function(j, alpha=0.15, M=4702)
{
  return(alpha * j / M)
}

# Populating the threshold.
fpr = 0.05
threshold = sapply(1:4702, bh_rejection, fpr)

# Merging everything into a dataframe.
results = data.frame(list(index=1:(ncol(data) - 1),
                          ordered_pvalues=ordered_pvalues,
                          threshold=threshold,
                          significant=threshold > ordered_pvalues))

p = ggplot(results) +
```

```
    geom_point(aes(x=index, y=ordered_pvalues, colour=significant)) +
    geom_line(aes(x=index, y=threshold, colour='BH Rejection Threshold')) +
    scale_colour_manual(values=c("#000000", "#1162bc", "#f83d69")) +
    labs(x='Ordered p-values index', y='p-value', colour='Significant')

print(p)

p = ggplot(results) +
    geom_point(aes(x=index, y=log(ordered_pvalues), colour=significant)) +
    geom_line(aes(x=index, y=log(threshold), colour='Log(BH Rejection Threshold)')) +
    scale_colour_manual(values=c("#1162bc", "#000000", "#f83d69")) +
    labs(x='Ordered p-values index', y='Log(p-value)', colour='Significant')

print(p)

# Slicing the selected features.
Features = colnames(data)[ordered_idxs[1:39]]
Features = data.frame(Features)

# Printing the selected features.
kable(Features,
      caption='Features selected by the Benjamini-Hochberg method')
```