국회 L1 regulization

낭니

8/18/2021

import module

```
library(readxl)
library(glmnet)

## 필요한 패키지를 로딩중입니다: Matrix

## Loaded glmnet 4.1-2

library(ggplot2)
library(graphics)
library(foreign)
library(dplyr)

##
## 다음의 패키지를 부착합니다: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

setwd('/Users/nang/Desktop/ㄴ ㅏㅇ/ㄱㅛㄹㅠㅎ ㅏㄱㅅㅜㄹㅈ ㅔ/regression')
```

import data

```
assembly <- read_excel('국회_final2.xlsx')
class(assembly)

## [1] "tbl_df"     "tbl"        "data.frame"

assembly <- as.data.frame(assembly)
assembly <- assembly[,-c(3,4)] # category, start 삭제
head(assembly,5) ; c('차원: ',dim(assembly))

##                                                                           tit
le
```

```
## 1                                                   국가보안법 개정에 관한 청원
## 2                             인공지능 윤리 및 고인의 AI 구현 법안 제정에 관한 청원
## 3 포괄적 차별금지법 법안 제정에 관한 동의 및 일부 내용 수정 요청에 관한 청원
## 4                            국민연금 수급 개시 연령과 연계한 정년연장에 관한 청원
## 5                                             하수도법 일부개정법률안에 관한 청원
##   count     sadness       anger topic__0 topic__1 topic__3 topic__4 topic_
_5
## 1  1761  0.09668989  0.14101084        0        0        0        1
 0
## 2   223  0.02967435  0.01117003        0        0        0        0
 0
## 3 14025  0.01208298  0.06047738        0        0        0        0
 0
## 4 19807 -0.02827379 -0.05525199        0        0        0        0
 0
## 5   736 -0.05677945 -0.04983796        0        0        0        0
 0
##   topic__6 topic__7 topic__9 topic__10 topic__11 topic__12 topic__13 topic
__14
## 1        0        0        0         0         0         0         0
   0
## 2        0        0        0         0         0         0         0
   0
## 3        0        0        0         0         0         0         0
   0
## 4        1        0        0         0         0         0         0
   0
## 5        0        0        0         0         0         0         0
   0
##   topic__15 topic__16 topic__17 topic__18 topic__19 topic__20 topic__21
## 1         0         0         0         0         0         0         0
## 2         0         0         0         0         0         0         0
## 3         0         0         0         0         0         0         0
## 4         0         0         0         0         0         0         0
## 5         0         0         0         1         0         0         0
##   topic__22 topic__23 topic__25 topic__26 topic__28 topic__29
## 1         0         0         0         0         0         0
## 2         0         0         0         0         0         1
## 3         0         0         0         0         0         1
## 4         0         0         0         0         0         0
## 5         0         0         0         0         0         0

## [1] "차원: " "220"    "30"
```

set x, y

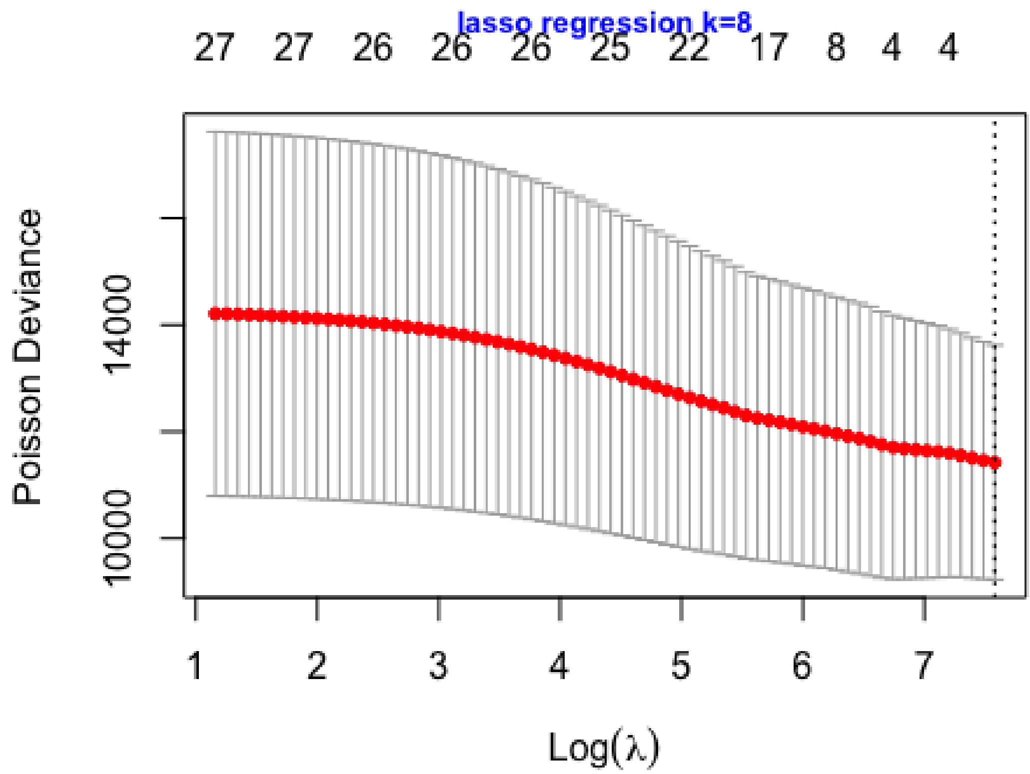
```
y <- assembly$count
names(assembly) <- c('title','count','sad','anger','t0','t1','t3','t4','t5','
t6','t7',
                     't9','t10','t11','t12','t13','t14','t15','t16','t17','t1
8',
                     't19','t20','t21','t22','t23','t25','t26','t28','t29')
X <- as.matrix(assembly[,-c(1,2)])

set.seed(sample(1:1000,1))
train <- sample(1:nrow(X), nrow(X)*0.7)
X_test <- (-train)
y_test <- y[X_test]
```

lasso regression

```
lasso <- cv.glmnet(X[train,], y[train], alpha = 1,
                   nfolds = 8,
                   family = 'poisson')
lasso$glmnet.fit

##
## Call:  glmnet(x = X[train, ], y = y[train], alpha = 1, family = "poisson")

##
##    Df  %Dev  Lambda
## 1   0  0.00 1958.00
## 2   2  1.81 1784.00
## 3   2  3.27 1625.00
## 4   3  4.49 1481.00
## 5   4  6.15 1349.00
## 6   4  8.13 1229.00
## 7   4  9.73 1120.00
## 8   4 11.04 1021.00
## 9   4 12.11  930.00
## 10  4 12.98  847.40
## 11  4 13.70  772.10
## 12  4 14.29  703.50
## 13  5 14.87  641.00
## 14  5 15.47  584.10
## 15  8 16.34  532.20
## 16 10 17.27  484.90
## 17 12 18.20  441.80
## 18 13 19.04  402.60
## 19 14 19.80  366.80
## 20 17 20.54  334.20
## 21 17 21.36  304.50
## 22 17 22.06  277.50
```

```
## 23 17 22.66  252.80
## 24 19 23.22  230.40
## 25 19 23.72  209.90
## 26 19 24.14  191.30
## 27 21 24.54  174.30
## 28 22 24.89  158.80
## 29 22 25.20  144.70
## 30 22 25.46  131.80
## 31 22 25.69  120.10
## 32 22 25.89  109.40
## 33 23 26.08   99.72
## 34 24 26.23   90.86
## 35 25 26.37   82.79
## 36 25 26.49   75.44
## 37 25 26.60   68.74
## 38 26 26.69   62.63
## 39 26 26.77   57.07
## 40 26 26.84   52.00
## 41 26 26.90   47.38
## 42 26 26.95   43.17
## 43 26 26.99   39.33
## 44 26 27.03   35.84
## 45 26 27.06   32.66
## 46 26 27.09   29.75
## 47 26 27.11   27.11
## 48 26 27.13   24.70
## 49 26 27.15   22.51
## 50 26 27.17   20.51
## 51 26 27.18   18.69
## 52 27 27.19   17.03
## 53 27 27.20   15.51
## 54 27 27.21   14.14
## 55 26 27.21   12.88
## 56 26 27.22   11.74
## 57 26 27.22   10.69
## 58 26 27.23    9.74
## 59 26 27.23    8.88
## 60 26 27.23    8.09
## 61 27 27.24    7.37
## 62 27 27.24    6.72
## 63 27 27.24    6.12
## 64 27 27.24    5.58
## 65 27 27.24    5.08
## 66 27 27.24    4.63
## 67 27 27.24    4.22
## 68 27 27.24    3.84
## 69 27 27.24    3.50
## 70 27 27.25    3.19
```

```
par(mfrow=c(1,1))
plot(lasso, main = '')
title(main = list('lasso regression k=8', cex = 0.8, col = 'blue'))
```




```
best.lambda <- lasso$lambda.min

best_lasso <- glmnet(X[train,], y[train], alpha = 1,
                     lambda = best.lambda,
                     family = 'poisson')

pred <- predict(best_lasso, s = best.lambda,
                newx = X[X_test,])
cbind(y_test, exp(pred))

##       y_test       s1
##  [1,]    223 5657.481
##  [2,]  14025 5657.481
##  [3,]  12867 5657.481
##  [4,]   3772 5657.481
##  [5,]   2587 5657.481
##  [6,]    581 5657.481
##  [7,]   2214 5657.481
```

```
##  [8,]    629 5657.481
##  [9,]   1530 5657.481
## [10,]   7867 5657.481
## [11,]  33875 5657.481
## [12,]   3667 5657.481
## [13,]  63240 5657.481
## [14,]    220 5657.481
## [15,]   6577 5657.481
## [16,]  20165 5657.481
## [17,]  24676 5657.481
## [18,]  36756 5657.481
## [19,]   3863 5657.481
## [20,]  45543 5657.481
## [21,]   1710 5657.481
## [22,]   3261 5657.481
## [23,]   2989 5657.481
## [24,]   1153 5657.481
## [25,]   2657 5657.481
## [26,]    581 5657.481
## [27,]   9287 5657.481
## [28,]   1099 5657.481
## [29,]  27321 5657.481
## [30,]  17500 5657.481
## [31,]  10176 5657.481
## [32,]    221 5657.481
## [33,]   1575 5657.481
## [34,]   1087 5657.481
## [35,]    267 5657.481
## [36,]    577 5657.481
## [37,]   7290 5657.481
## [38,]    587 5657.481
## [39,]    370 5657.481
## [40,]    233 5657.481
## [41,]   1911 5657.481
## [42,]   7839 5657.481
## [43,]   2537 5657.481
## [44,]   1367 5657.481
## [45,]    745 5657.481
## [46,]   7447 5657.481
## [47,]   1300 5657.481
## [48,]   1020 5657.481
## [49,]    428 5657.481
## [50,]   2466 5657.481
## [51,]    551 5657.481
## [52,]    147 5657.481
## [53,]    299 5657.481
## [54,]    350 5657.481
## [55,]   2884 5657.481
## [56,]    195 5657.481
```

```
## [57,]  50352 5657.481
## [58,]    400 5657.481
## [59,]    138 5657.481
## [60,]    221 5657.481
## [61,]  68319 5657.481
## [62,]  22692 5657.481
## [63,]  13026 5657.481
## [64,]    899 5657.481
## [65,]    226 5657.481
## [66,]    169 5657.481
```

```
coef(best_lasso, s = lasso$lambda.min)
```

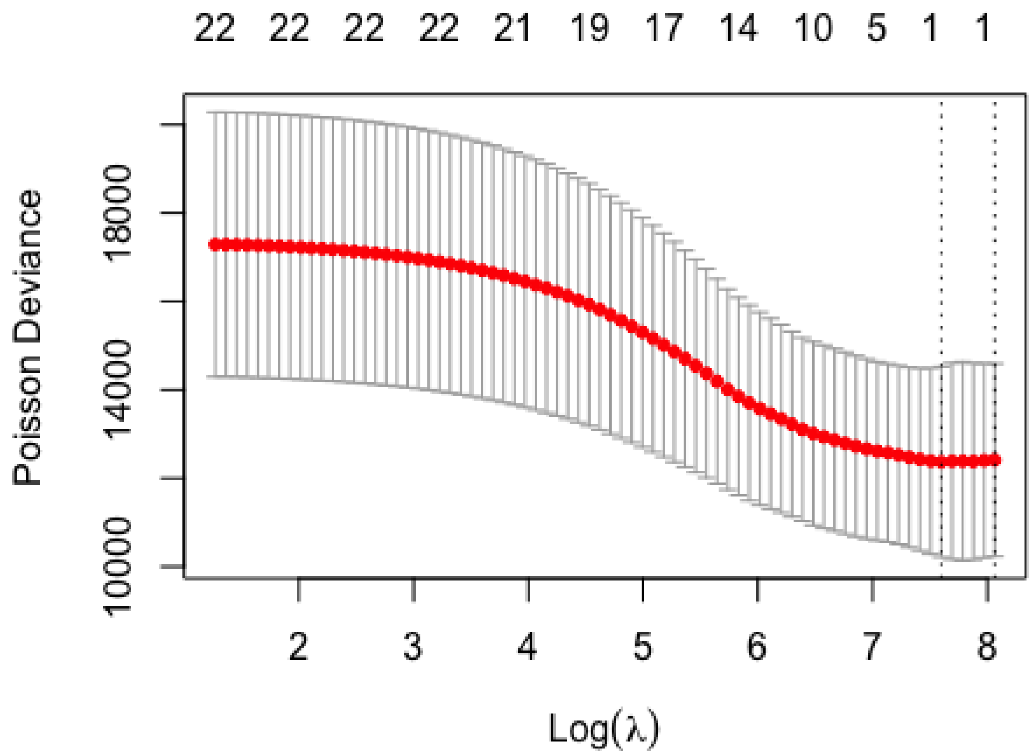
```
## 29 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 8.640734e+00
## sad         .
## anger       .
## t0          .
## t1          .
## t3          .
## t4          .
## t5          .
## t6          .
## t7          .
## t9          .
## t10         .
## t11         .
## t12         .
## t13         1.163185e-15
## t14         .
## t15         .
## t16         .
## t17         .
## t18         .
## t19         .
## t20         .
## t21         .
## t22         .
## t23         .
## t25         .
## t26         .
## t28         .
## t29         .
```

```
summary(best_lasso$beta)
```

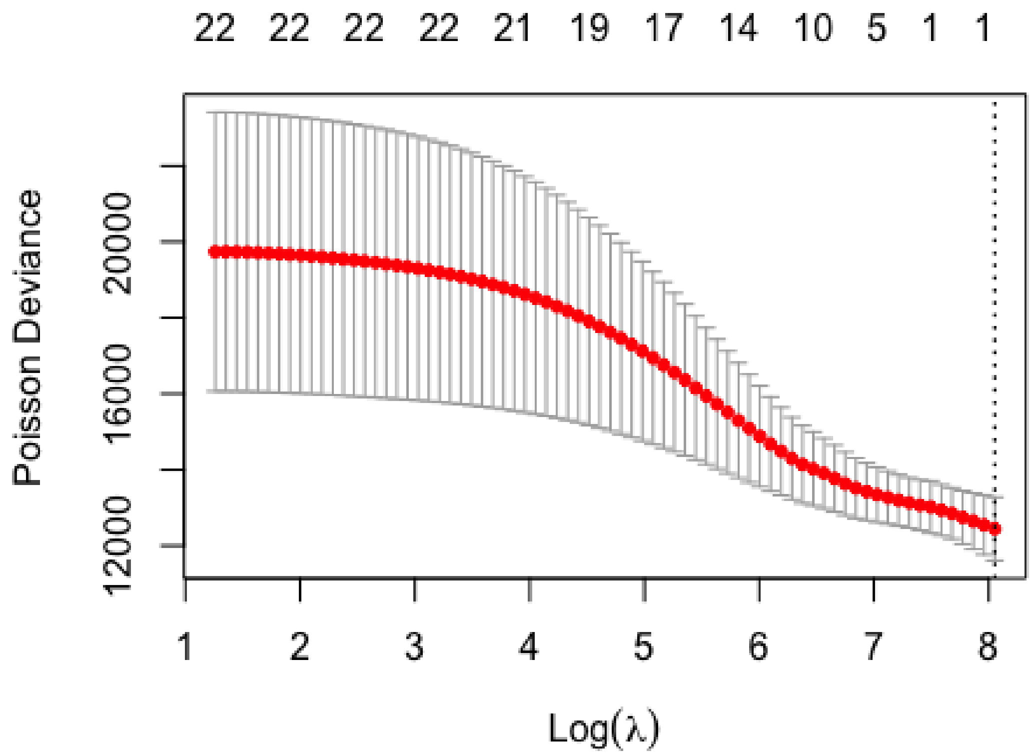
```
## 28 x 1 sparse Matrix of class "dgCMatrix", with 1 entries
##    i j            x
## 1 14 1 1.163185e-15
```

elasticnet

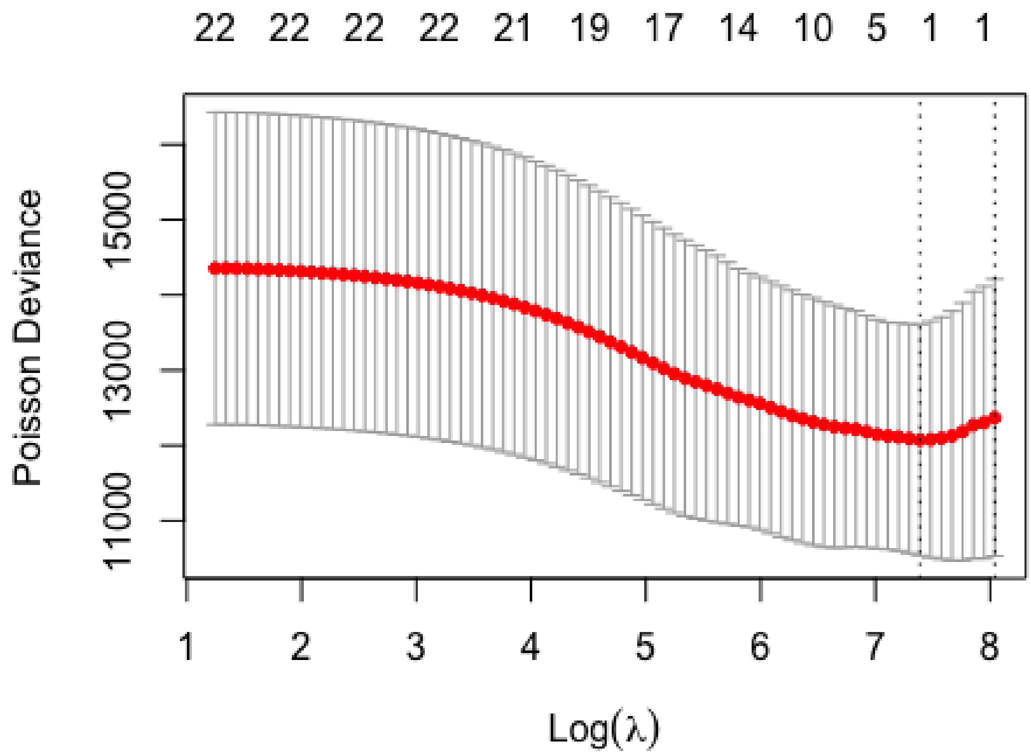
```
y <- assembly$count
names(assembly) <- c('title','count','sad','anger','t0','t1','t3','t4','t5','
t6','t7',
                     't9','t10','t11','t12','t13','t14','t15','t16','t17','t1
8',
                     't19','t20','t21','t22','t23','t25','t26','t28','t29')
X <- as.matrix(assembly[,-c(1,2)])
set.seed(set.seed(sample(1:1000,1)))
train <- sample(1:nrow(X), nrow(X)*0.4)
X_test <- (-train)
y_test <- y[X_test]

alpha <- seq(0.8,0.99,0.01)
best.lambda_min <- rep(0,length(alpha))
best.lambda_1se <- rep(0,length(alpha))
for (i in 1:length(alpha)){
  k = cv.glmnet(X[train,], y[train], alpha = alpha[i],
            nfolds = 5,
            family = 'poisson')

  plot(k)
  best.lambda_min[i] <- k$lambda.min
  best.lambda_1se[i] <- k$lambda.1se
  print(c('alpha :', alpha[i],'best lambda - min: ', best.lambda_min[i],
          'best lambda - 1se: ', best.lambda_1se[i]))
}
```



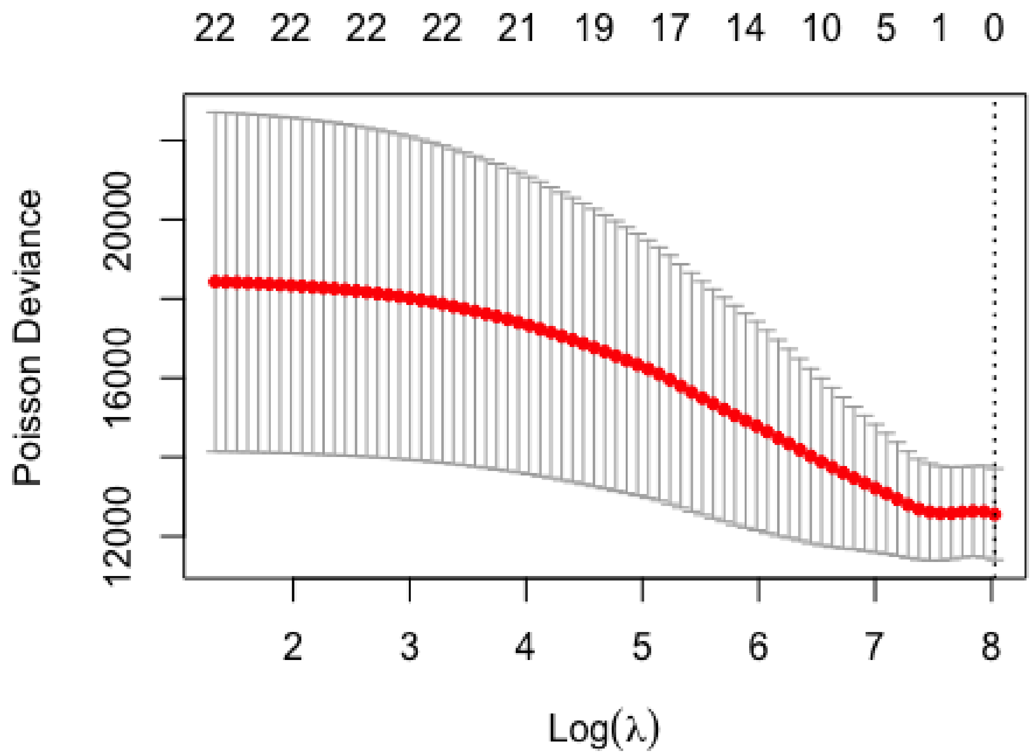

```
## [1] "alpha :"              "0.8"                  "best lambda - min: "
## [4] "1996.7795076409"      "best lambda - 1se: " "3179.43765073531"
```



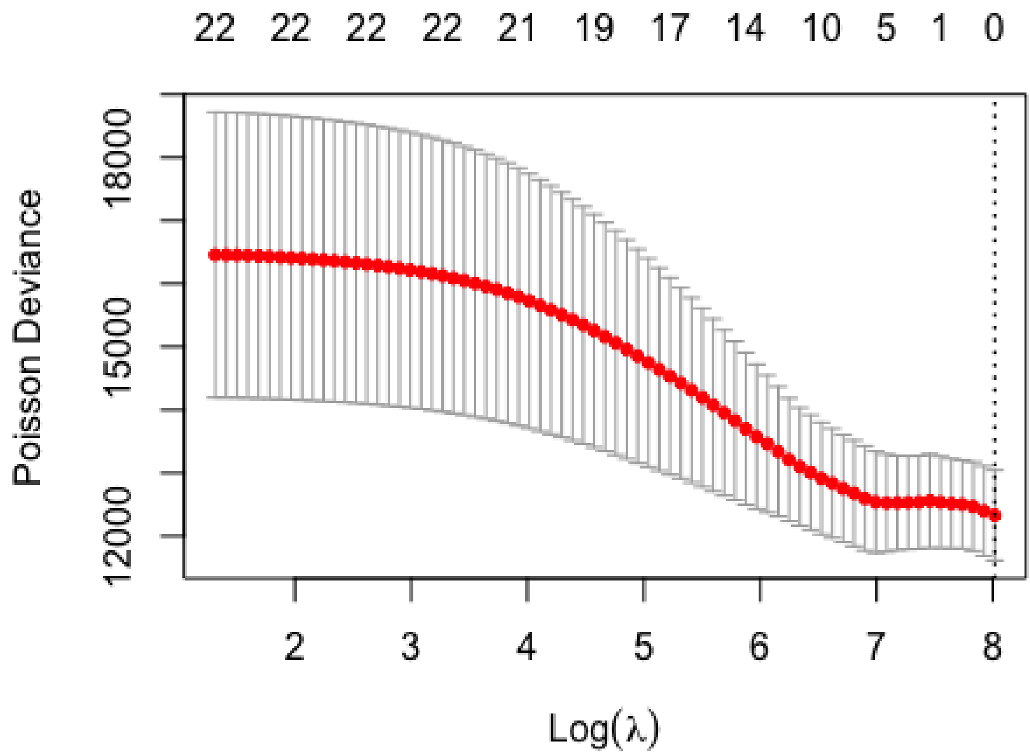

```
## [1] "alpha :"             "0.81"                "best lambda - min: "
## [4] "3140.18533405956"    "best lambda - 1se: " "3140.18533405956"
```



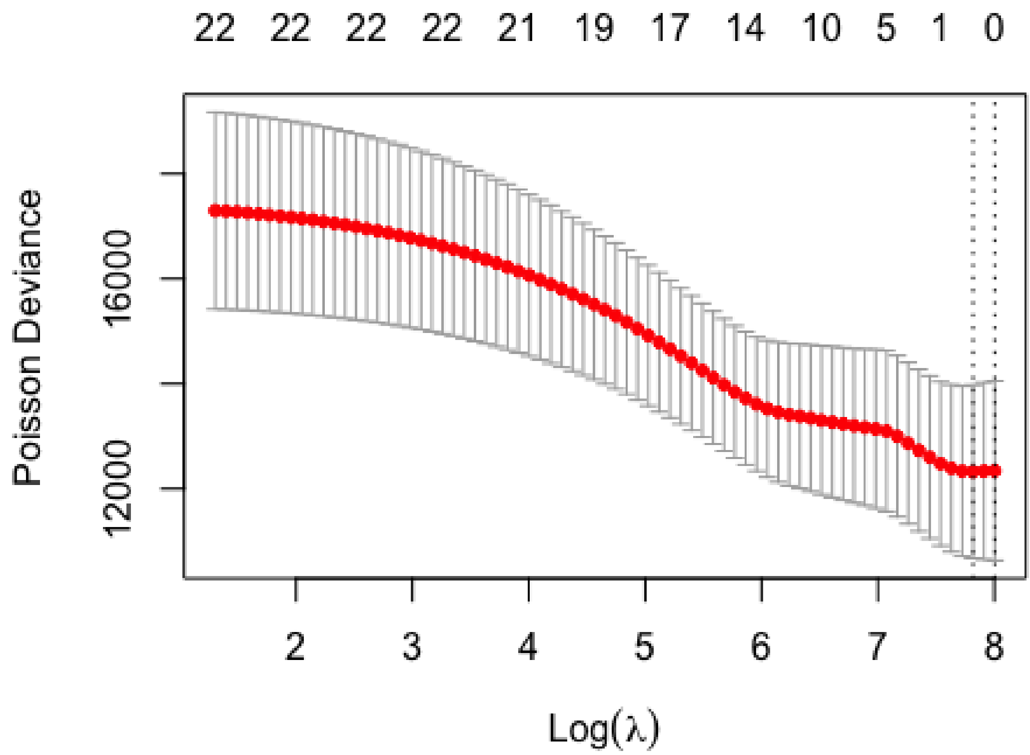

```
## [1] "alpha :"              "0.82"                 "best lambda - min: "
## [4] "1617.32822167479"     "best lambda - 1se: " "3101.89039096127"
```



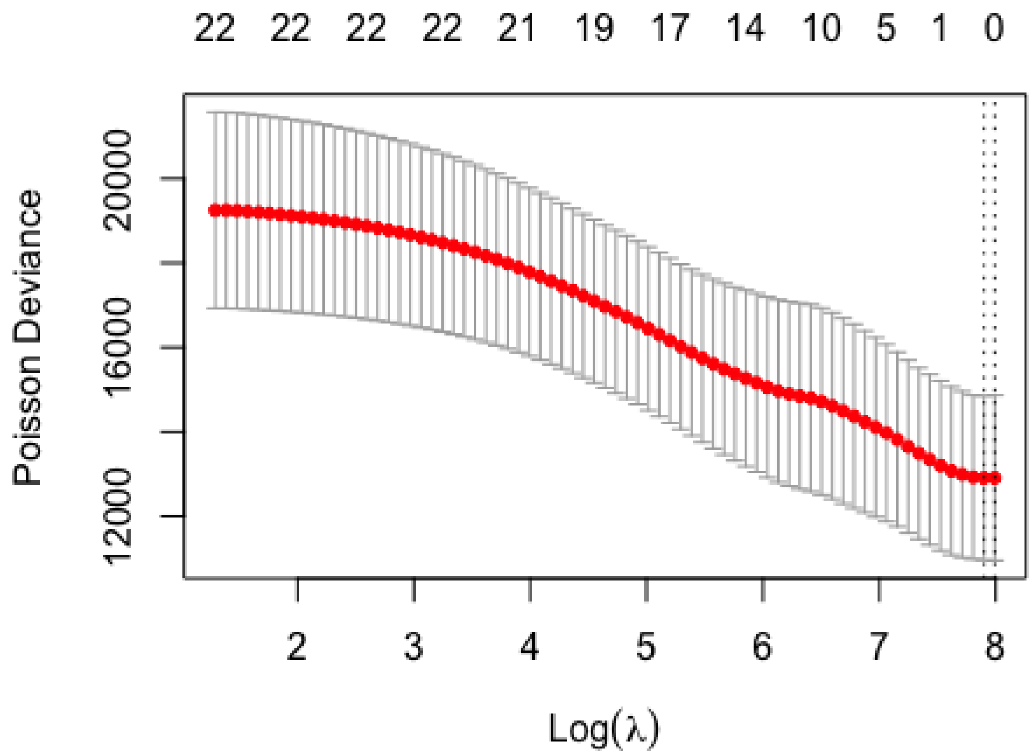

```
## [1] "alpha :"              "0.83"                 "best lambda - min: "
## [4] "3064.51821757619"     "best lambda - 1se: " "3064.51821757619"
```



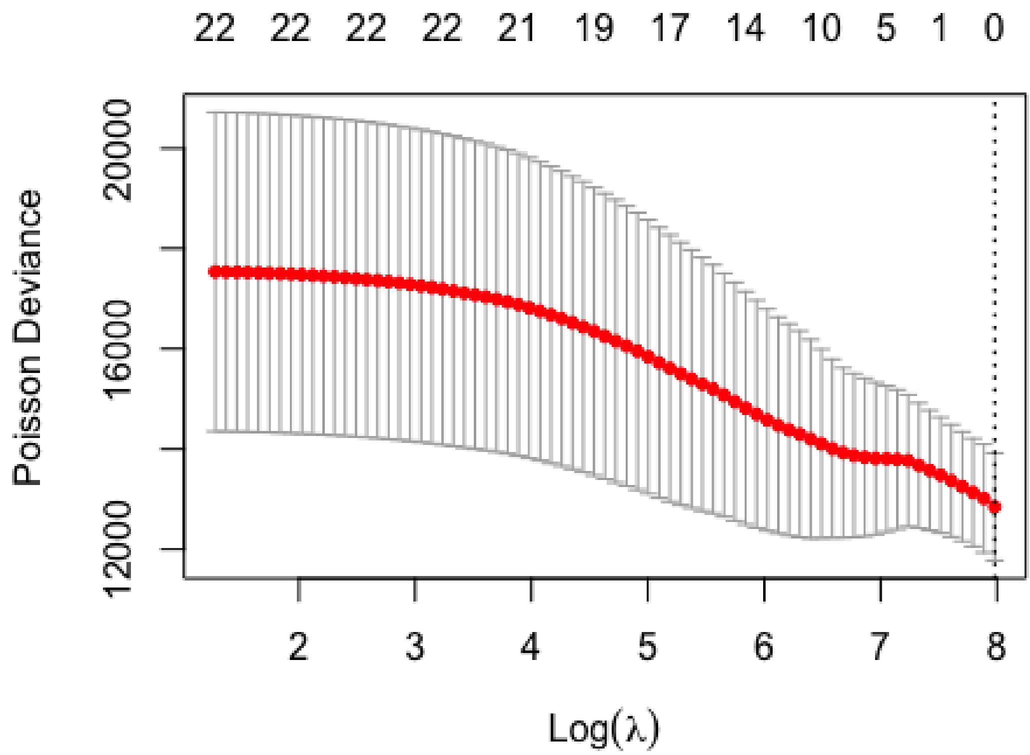

```
## [1] "alpha :"               "0.84"                  "best lambda - min: "
## [4] "3028.03585784315"      "best lambda - 1se: " "3028.03585784315"
```



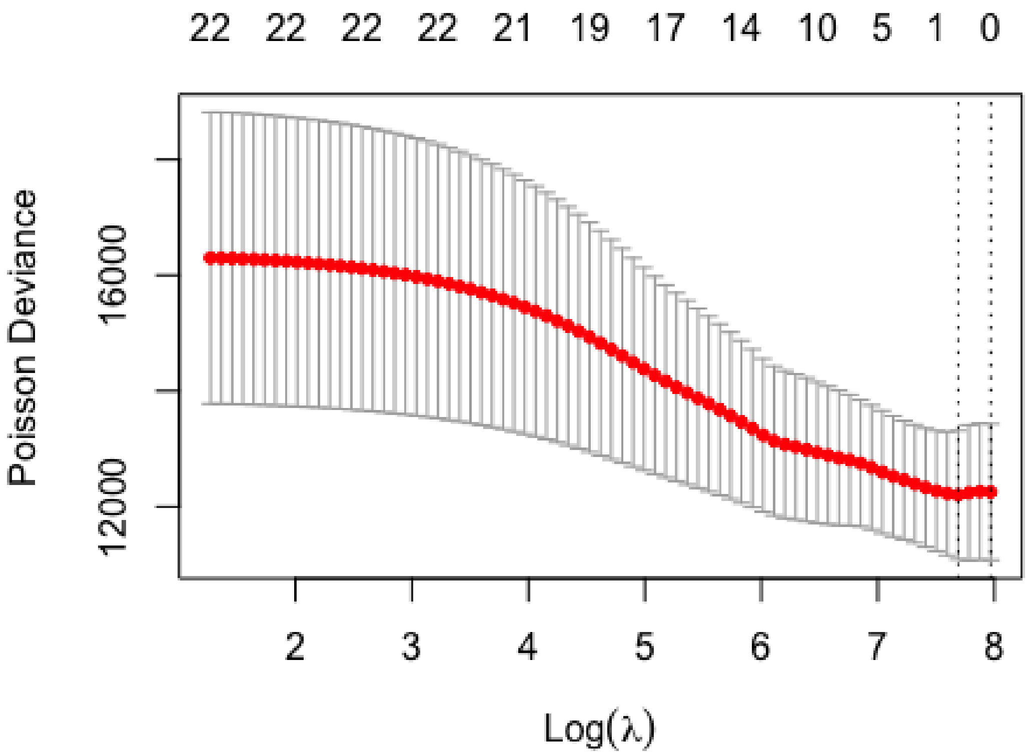

```
## [1] "alpha :"              "0.85"                 "best lambda - min: "
## [4] "2484.35293635588"     "best lambda - 1se: " "2992.4119065744"
```



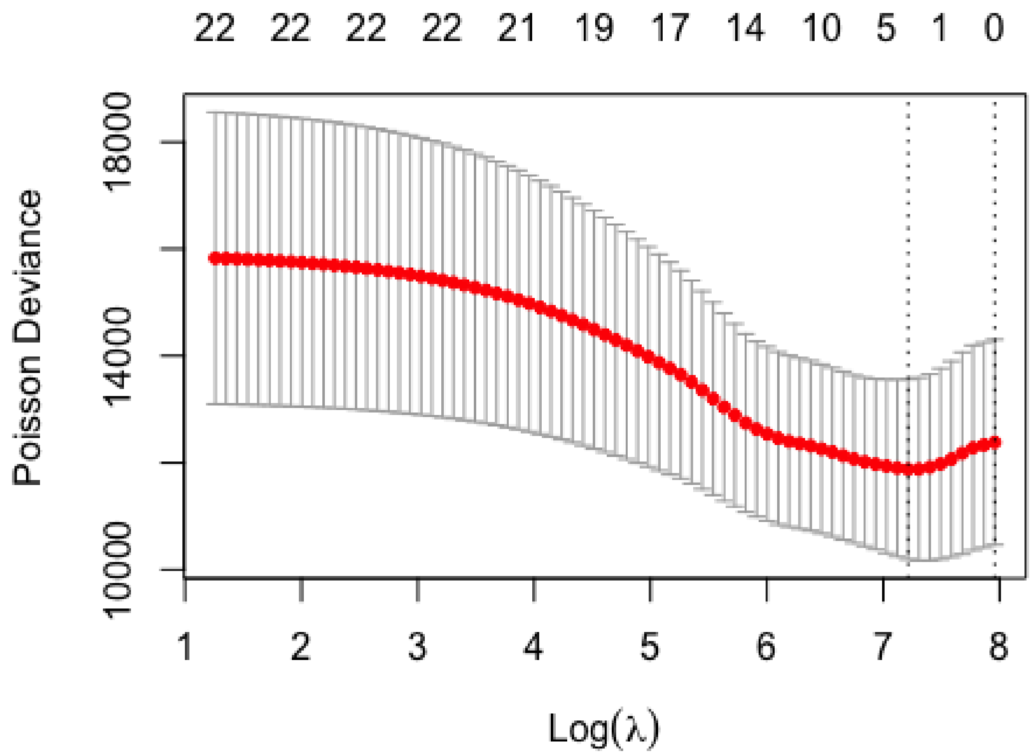

```
## [1] "alpha :"              "0.86"                 "best lambda - min: "
## [4] "2694.86992836497"     "best lambda - 1se: " "2957.61641928865"
```



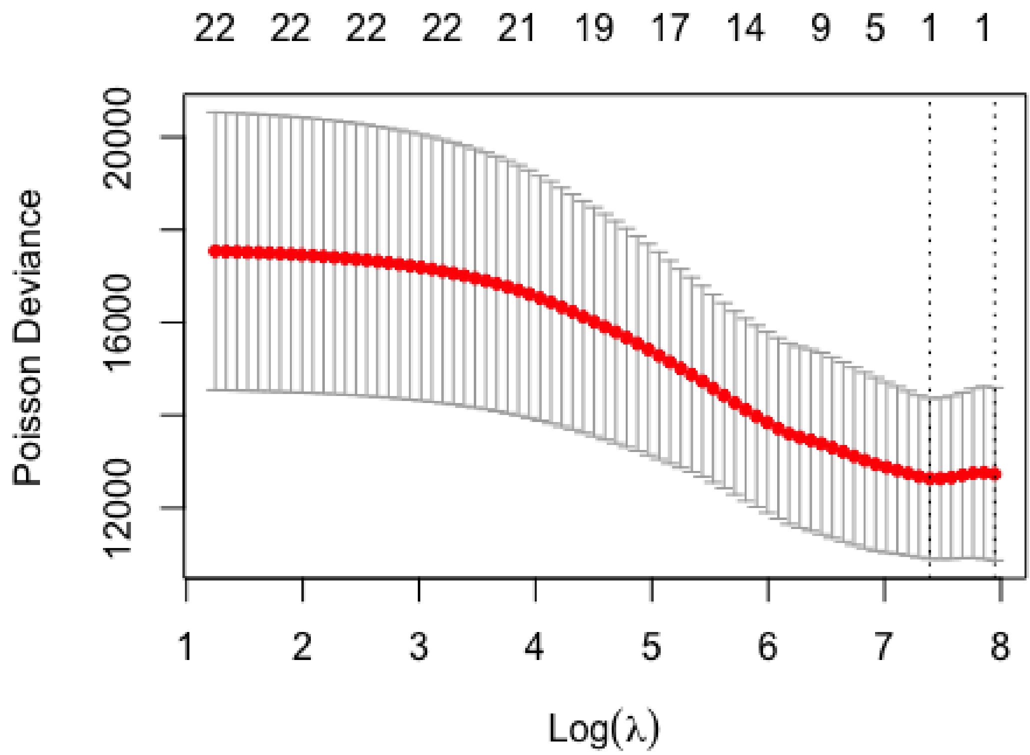

```
## [1] "alpha :"              "0.87"                 "best lambda - min: "
## [4] "2923.62082826235"     "best lambda - 1se: " "2923.62082826235"
```



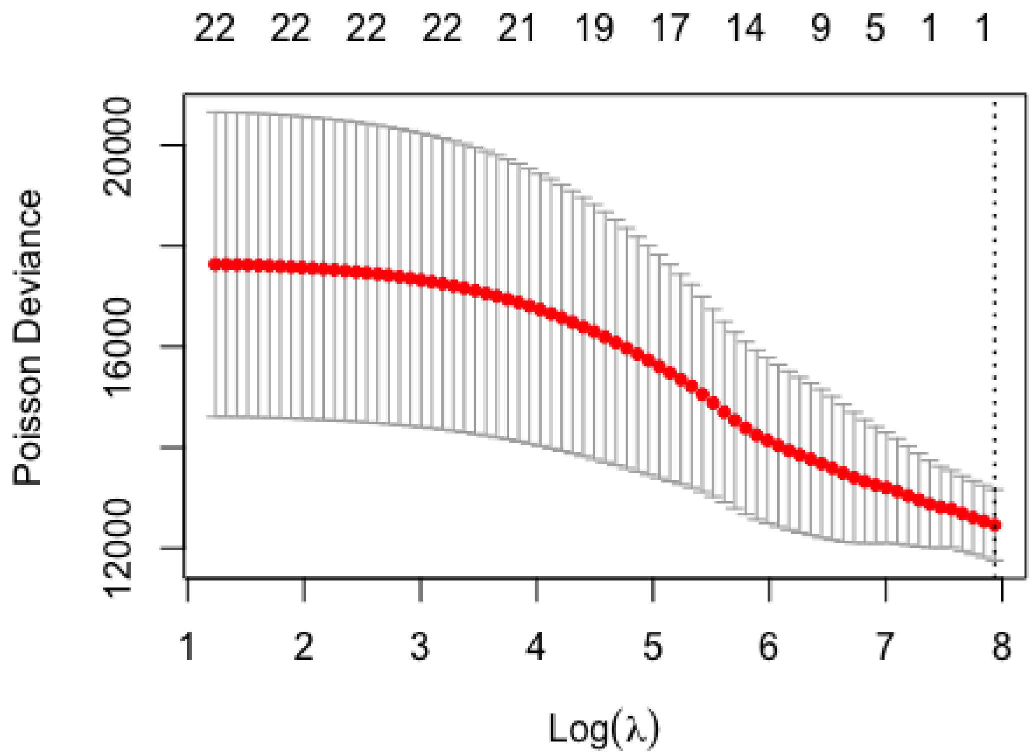

```
## [1] "alpha :"               "0.88"                  "best lambda - min: "
## [4] "2186.47998695716"      "best lambda - 1se: " "2890.39786430482"
```



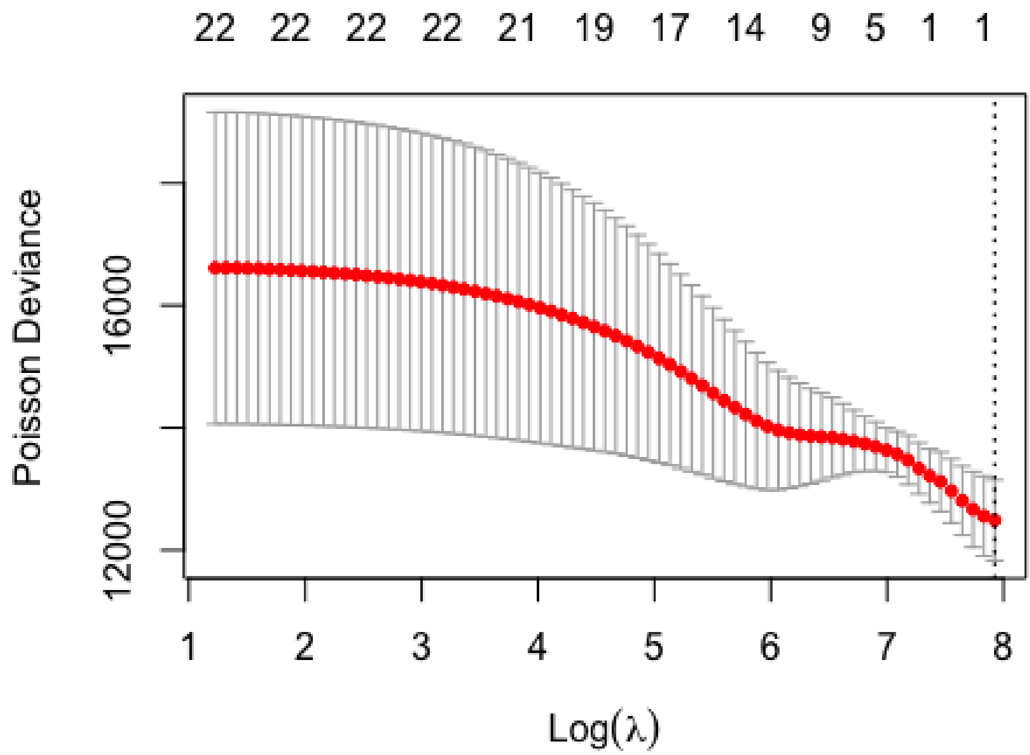

```
## [1] "alpha :"              "0.89"                 "best lambda - min: "
## [4] "1357.74424371735"     "best lambda - 1se: " "2857.92148380702"
```



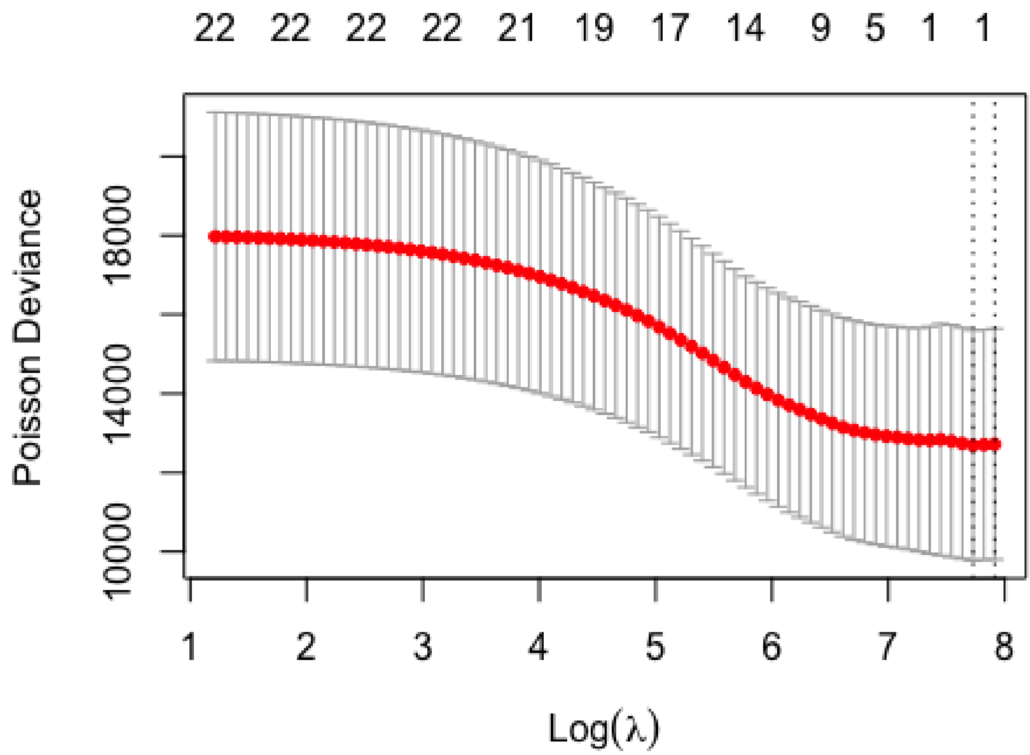

```
## [1] "alpha :"              "0.9"                  "best lambda - min: "
## [4] "1617.23655083969"     "best lambda - 1se: " "2826.1668006536"
```



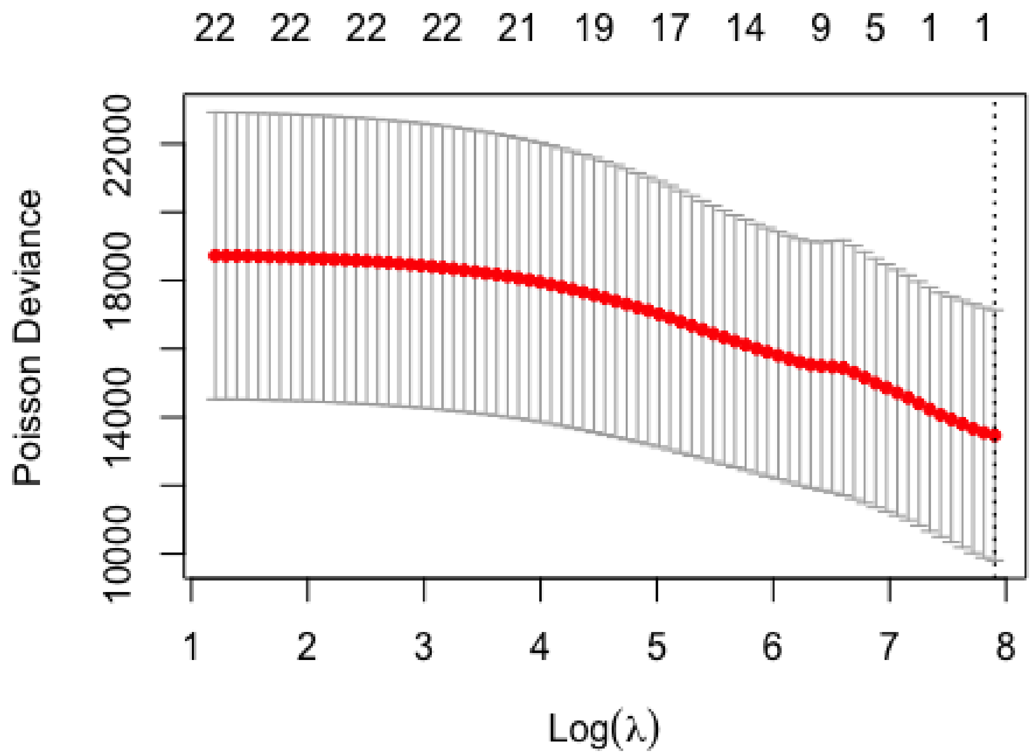

```
## [1] "alpha :"             "0.91"                "best lambda - min: "
## [4] "2795.11002262444"    "best lambda - 1se: " "2795.11002262444"
```



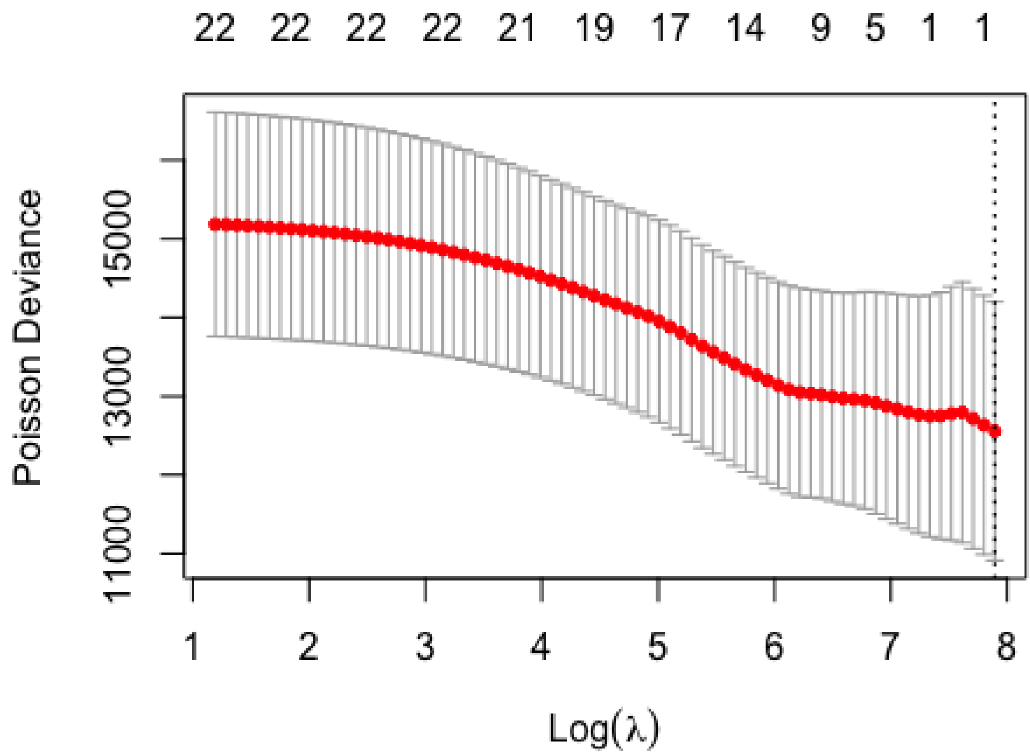

```
## [1] "alpha :"             "0.92"                 "best lambda - min: "
## [4] "2764.72839194374"    "best lambda - 1se: " "2764.72839194374"
```



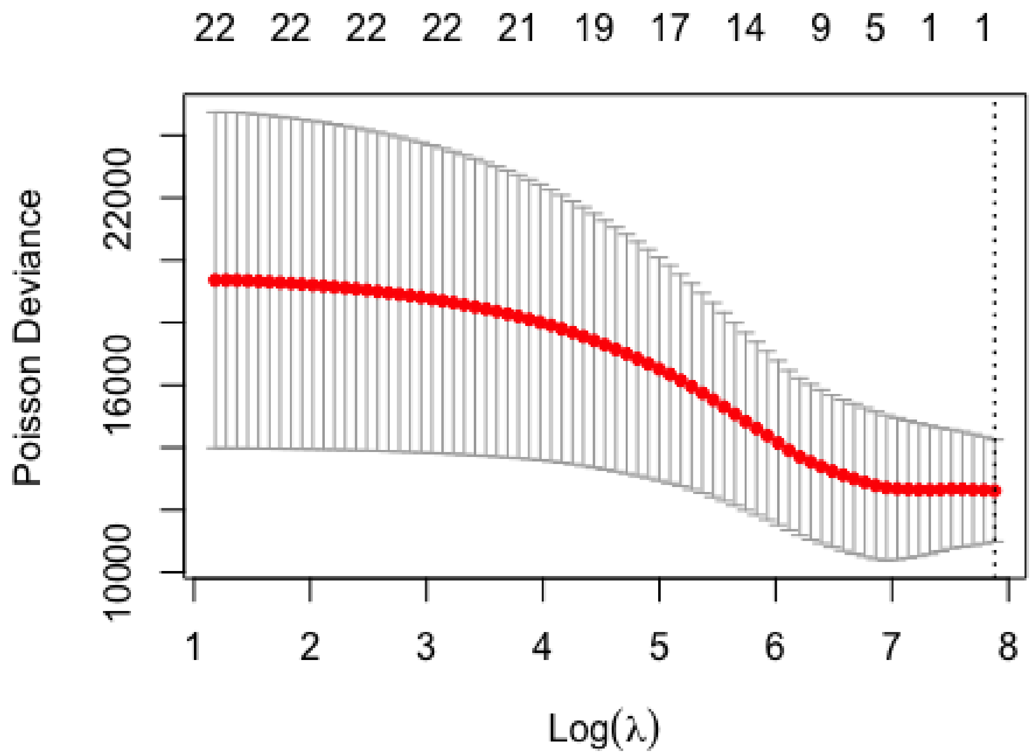

```
## [1] "alpha :"              "0.93"                 "best lambda - min: "
## [4] "2270.6451568844"      "best lambda - 1se: " "2735.00012966478"
```



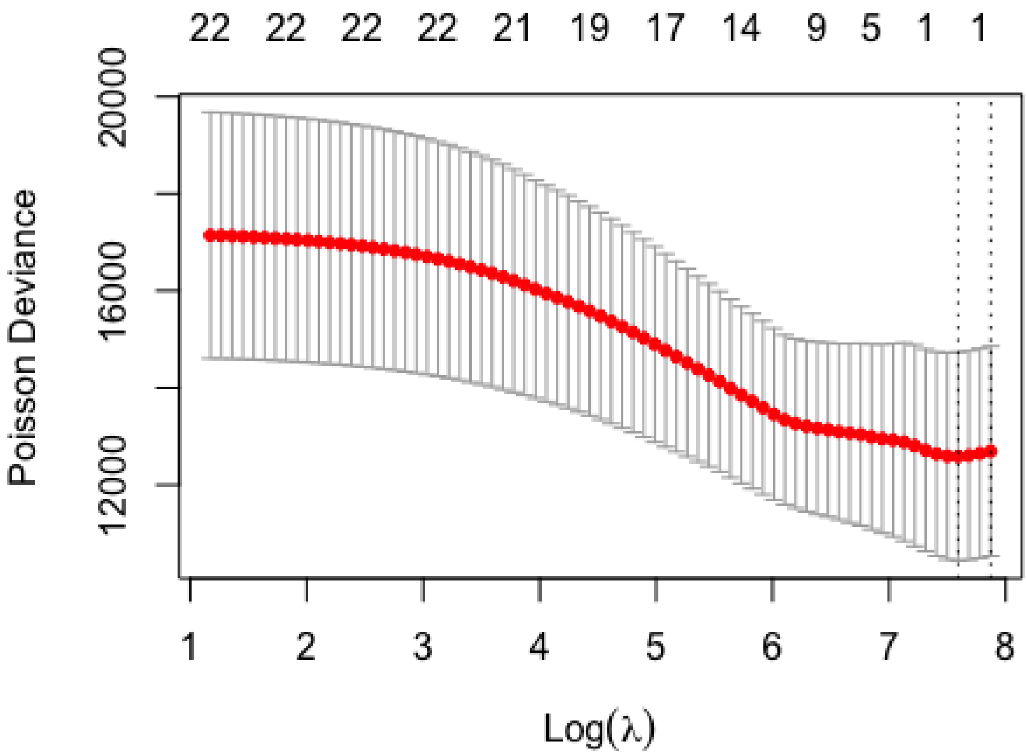

```
## [1] "alpha :"              "0.94"                 "best lambda - min: "
## [4] "2705.90438360452"     "best lambda - 1se: " "2705.90438360452"
```



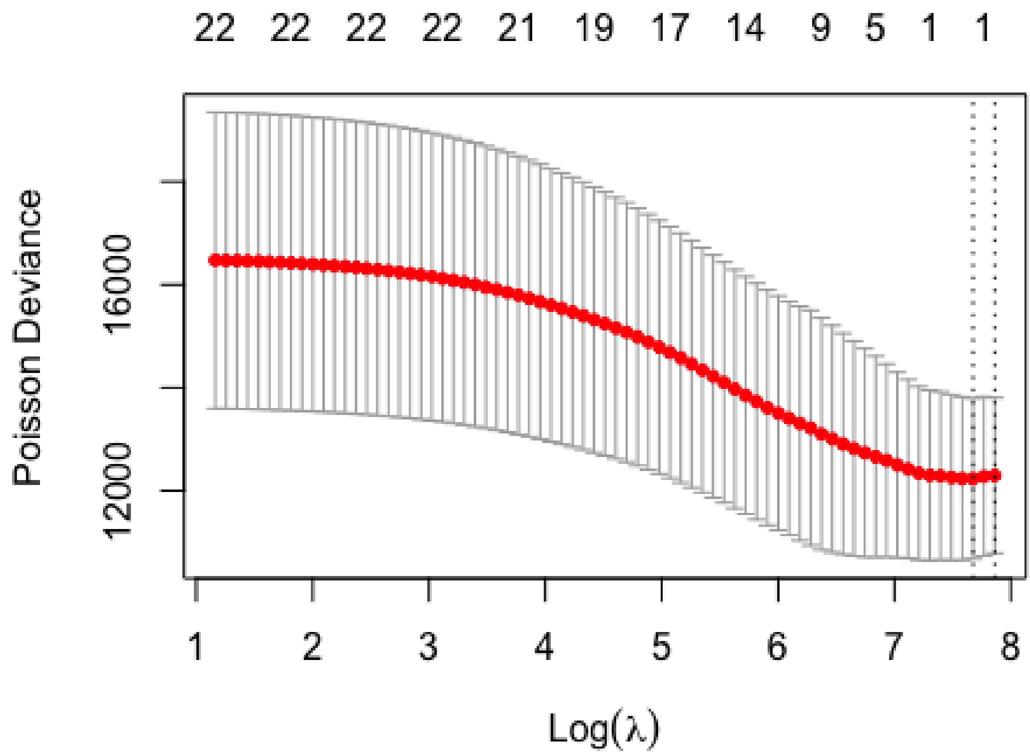

```
## [1] "alpha :"              "0.95"                 "best lambda - min: "
## [4] "2677.42117956657"     "best lambda - 1se: " "2677.42117956657"
```



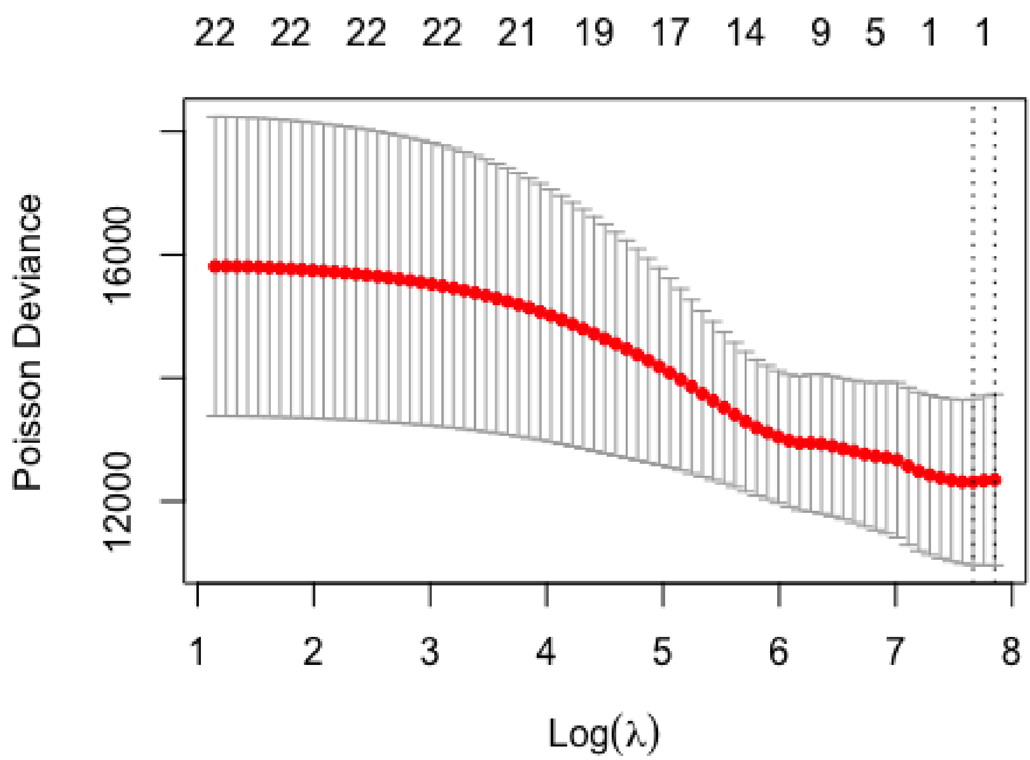

```
## [1] "alpha :"             "0.96"                 "best lambda - min: "
## [4] "2649.53137561275"    "best lambda - 1se: " "2649.53137561275"
```




```
## [1] "alpha :"             "0.97"                "best lambda - min: "
## [4] "1983.6107098168"     "best lambda - 1se: " "2622.21661916314"
```




```
## [1] "alpha :"              "0.98"                 "best lambda - min: "
## [4] "2154.79591418622"     "best lambda - 1se: " "2595.4593067227"
```




```
## [1] "alpha :"              "0.99"                 "best lambda - min: "
## [4] "2133.03029889141"     "best lambda - 1se: " "2569.24254604873"

min(best.lambda_min)

## [1] 1357.744

which.min(best.lambda_min)

## [1] 10

best.elasticnet <- glmnet(X[train,], y[train], alpha = alpha[which.min(best.l
ambda_min)],
                          lambda = best.lambda_min[which.min(best.lambda_min)],
                          family = 'poisson')
pred <- predict(best.elasticnet, s = best.lambda_min[which.min(best.lambda_mi
n)],
                newx = X[X_test,])

cbind(y_test,exp(pred))

##         y_test       s1
##   [1,]  14025 5186.754
```

```
##   [2,]  19807 5186.754
##   [3,]    736 5186.754
##   [4,]   1197 5186.754
##   [5,]    752 5186.754
##   [6,]  12867 5186.754
##   [7,]   3772 8668.689
##   [8,]  24822 5186.754
##   [9,]   2587 5186.754
##  [10,]  12582 5186.754
##  [11,]    581 5186.754
##  [12,]   2214 5186.754
##  [13,]    387 5186.754
##  [14,]   1530 5186.754
##  [15,]  25355 5186.754
##  [16,]   5577 5186.754
##  [17,]   7867 5186.754
##  [18,]   5913 5186.754
##  [19,]   7841 5186.754
##  [20,]   1849 5186.754
##  [21,]   4366 5186.754
##  [22,]   4486 5186.754
##  [23,]   3667 5186.754
##  [24,]   6223 5186.754
##  [25,]   5736 5186.754
##  [26,]   7936 5186.754
##  [27,]  63240 5186.754
##  [28,]   1955 5186.754
##  [29,]   1700 5186.754
##  [30,]    220 5186.754
##  [31,]   9291 5186.754
##  [32,]   6577 5186.754
##  [33,]   5837 5186.754
##  [34,]    429 5186.754
##  [35,]  20165 8668.689
##  [36,]   4194 5186.754
##  [37,]   3863 8668.689
##  [38,]  20626 5186.754
##  [39,]   2989 8668.689
##  [40,]  11057 8668.689
##  [41,]  11032 8668.689
##  [42,]    389 5186.754
##  [43,]   1214 5186.754
##  [44,]   1945 5186.754
##  [45,]   2978 5186.754
##  [46,]   1983 5186.754
##  [47,]   3007 5186.754
##  [48,]    961 8668.689
##  [49,]    581 8668.689
##  [50,]   1377 5186.754
```

```
##  [51,]    353 8668.689
##  [52,]   3291 5186.754
##  [53,]   3412 5186.754
##  [54,]    472 5186.754
##  [55,]    239 8668.689
##  [56,]   9287 5186.754
##  [57,]    739 5186.754
##  [58,]   1179 5186.754
##  [59,]   1099 5186.754
##  [60,]  27321 5378.286
##  [61,]  17500 5186.754
##  [62,]    699 8668.689
##  [63,]    700 5186.754
##  [64,]    125 5186.754
##  [65,]  13347 5186.754
##  [66,]    889 8668.689
##  [67,]    184 5186.754
##  [68,]    206 5186.754
##  [69,]    898 5186.754
##  [70,]    267 5186.754
##  [71,]    344 5186.754
##  [72,]  19695 5186.754
##  [73,]    419 8668.689
##  [74,]   9281 5186.754
##  [75,]    685 5186.754
##  [76,]    587 5186.754
##  [77,]    370 5186.754
##  [78,]    233 5378.286
##  [79,]   1911 5186.754
##  [80,]    273 8668.689
##  [81,]   7839 8668.689
##  [82,]  13631 5186.754
##  [83,]   1986 5186.754
##  [84,]  63489 5186.754
##  [85,]   1723 5186.754
##  [86,]    853 5186.754
##  [87,]   1367 5186.754
##  [88,]    745 5186.754
##  [89,]  56899 5186.754
##  [90,]    236 5186.754
##  [91,]   7447 5186.754
##  [92,]   3693 5186.754
##  [93,]   3761 5186.754
##  [94,]   1300 5186.754
##  [95,]   5097 5186.754
##  [96,]   1020 5186.754
##  [97,]    173 5186.754
##  [98,]    428 5186.754
##  [99,]  17288 8668.689
```

```
## [100,]   2466 5186.754
## [101,]   6247 5186.754
## [102,]    817 5186.754
## [103,]  14002 5186.754
## [104,]    551 5186.754
## [105,]    147 5186.754
## [106,]   4776 5186.754
## [107,]    232 8668.689
## [108,]    299 5186.754
## [109,]    350 5186.754
## [110,]    653 5186.754
## [111,]   2884 5186.754
## [112,]    195 5186.754
## [113,]  50352 5186.754
## [114,]    762 5186.754
## [115,]    133 5186.754
## [116,]   7856 5186.754
## [117,]    400 5186.754
## [118,]    250 5186.754
## [119,]    688 5186.754
## [120,]    221 5186.754
## [121,]  68319 5186.754
## [122,]  50273 5186.754
## [123,]   1653 5186.754
## [124,]  13026 5186.754
## [125,]  14929 5186.754
## [126,]  12051 5186.754
## [127,]   1240 5186.754
## [128,]    899 5186.754
## [129,]    285 5186.754
## [130,]    226 5186.754
## [131,]    161 5186.754
## [132,]    152 5186.754
```

```
coef(best.elasticnet, s = best.lambda_min[which.min(best.lambda_min)])
```

```
## 29 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept) 8.55386343
## sad         .
## anger       .
## t0          .
## t1          0.51360942
## t3          .
## t4          .
## t5          .
## t6          .
## t7          0.03626163
## t9          .
```

```
## t10         .
## t11         .
## t12         .
## t13         .
## t14         .
## t15         .
## t16         .
## t17         .
## t18         .
## t19         .
## t20         .
## t21         .
## t22         .
## t23         .
## t25         .
## t26         .
## t28         .
## t29         .
```

```
summary(best.elasticnet$beta)
```

```
## 28 x 1 sparse Matrix of class "dgCMatrix", with 2 entries
##   i j          x
## 1 4 1 0.51360942
## 2 9 1 0.03626163
```