Project for P-spline and Multilevel

Choi TaeYoung

2020-04-08

Contents

필요한 패키지 1

데이터 2

데이터 정리 및 Goodness of fit test를 통한 적절한 모델 찾기 2
GoF 결과 . . . 3

Multilevel 모델에 적용 5
x_list 최소값, 최대값 추출 . . . 6
최적의 GCV_vec 찾기 . . . 6

Naive Method(from "ydhwang/mlsplines" in Github) 12
Naive's GCV vector 찾기 . . . 12

그래프 15

필요한 패키지

```
#for data
library(tidyverse)
library(tidyr)
library(devtools)
library(MASS)
library(lubridate)
library(quantmod)
library(PerformanceAnalytics)
library(magrittr)
library(dplyr)
library(data.table)

# for graph
library(ggplot2)
library(dygraphs)
library(highcharter)

# for model
library(pscl)
```

데이터

- Y data : Y데이터의 경우 120달(2009년 1월 ~ 2018년 12월)동안의 한국에 입국한 국적별 외국인 수를 나타냈다.
- X data : X데이터의 경우 120달 동안의 각 나라의 한국 원화 기준 환율을 나타낸 것이다. 나라는 한글 순으로 [가나, 가봉, … , 헝가리, 호주, 홍콩] 174개국으로 구성했으며, 시간의 흐름에 따라 데이터를 나열했다.

데이터 정리 및 Goodness of fit test를 통한 적절한 모델 찾기

- X, Y 데이터 모두 리스트화를 거쳤다. Y데이터가 허들모델이라는 가정으로 각 행마다 0이 얼마나 포함되어 있는지 알아보았다.
- 그 결과 32, 71, 94, 100, 104, 112, 113행에는 0을 포함하지 않아서 허들모델이나 zero inflated 방법을 이용하여 모델을 적합할 수 없었다.
- 그래서 우리는 Goodness of Fit(GoF)를 이용하여 일반화 선형모형의 적합도를 검정해보았다.

```
y_list <- alldata[,-1] %>% t() %>% as.data.frame() %>% as.list()
x_list <- exr[,-1] %>%  t() %>% as.data.frame() %>% as.list()

#How many zero in y_list?
y_list <- alldata[,-1] %>% t() %>% as.data.frame()
y_zero <- NULL
for(m in 1:120){
  zero <- NULL
  zero <- length(which(y_list[m] == 0))/174
  y_zero <- rbind(y_zero,zero)

  zero_count <- length(which(y_zero == 0))
  zero_where <- which(y_zero == 0)
  zero_count
  zero_where
}
zero_count
```

```
## [1] 7
```

GoF 결과

- 그 결과 포아송 GoF는 모두 0으로 나왔으며, 음이항분포 GoF는 낮은 값을 보였다. 즉, 포아송분포를 사용하였을 때 과대산포가 발생하므로, 음이항분포를 이용하여 모형적합을 시도했다.

```r
#GOF Calculate
goodness <- NULL
  for(m in 1:120){
    result1_out <- NULL
    result2_out <- NULL
    results1 <- glm(unlist(y_list[m]) ~ unlist(x_list[m]), family = poisson)
    results2 <- glm.nb(unlist(y_list[m]) ~ unlist(x_list[m]))

    poi_GOF <- 1 - pchisq(summary(results1)$deviance,
           summary(results1)$df.residual
           )
    nb_GOF <- 1 - pchisq(summary(results2)$deviance,
           summary(results2)$df.residual
           )
    out <- cbind(poi_GOF,nb_GOF)
    goodness <- rbind(goodness, out)
  }

goodness
```

```
##        poi_GOF       nb_GOF
##   [1,]       0 4.216903e-04
##   [2,]       0 2.336926e-04
##   [3,]       0 1.315852e-04
##   [4,]       0 9.886678e-05
##   [5,]       0 1.266799e-04
##   [6,]       0 1.770270e-04
##   [7,]       0 1.265186e-04
##   [8,]       0 1.198591e-04
##   [9,]       0 1.189589e-04
##  [10,]       0 7.047179e-05
##  [11,]       0 1.729070e-04
##  [12,]       0 2.746954e-04
##  [13,]       0 2.718776e-04
##  [14,]       0 4.169687e-04
##  [15,]       0 2.464476e-04
##  [16,]       0 1.910588e-04
##  [17,]       0 1.463103e-04
##  [18,]       0 6.808432e-05
##  [19,]       0 1.111294e-04
##  [20,]       0 7.162165e-05
##  [21,]       0 8.499500e-05
##  [22,]       0 6.994961e-05
##  [23,]       0 1.256059e-04
##  [24,]       0 2.103422e-04
##  [25,]       0 4.892183e-04
##  [26,]       0 1.771561e-04
##  [27,]       0 1.972107e-04
##  [28,]       0 1.196807e-04
##  [29,]       0 1.081669e-04
```

```
## [30,]        0 7.931204e-05
## [31,]        0 1.681955e-04
## [32,]        0 3.461796e-05
## [33,]        0 9.215112e-05
## [34,]        0 5.097686e-05
## [35,]        0 5.128410e-05
## [36,]        0 1.359643e-04
## [37,]        0 8.970739e-05
## [38,]        0 8.980080e-05
## [39,]        0 1.160048e-04
## [40,]        0 1.064577e-04
## [41,]        0 5.722055e-05
## [42,]        0 7.164615e-05
## [43,]        0 5.300821e-05
## [44,]        0 4.891635e-05
## [45,]        0 4.451827e-05
## [46,]        0 5.241495e-05
## [47,]        0 7.143896e-05
## [48,]        0 1.645429e-04
## [49,]        0 9.390959e-05
## [50,]        0 4.021199e-05
## [51,]        0 7.856806e-05
## [52,]        0 1.038062e-04
## [53,]        0 9.369332e-05
## [54,]        0 9.959533e-05
## [55,]        0 5.618779e-05
## [56,]        0 4.224859e-05
## [57,]        0 5.497540e-05
## [58,]        0 7.046496e-05
## [59,]        0 7.993177e-05
## [60,]        0 1.247997e-04
## [61,]        0 8.578268e-05
## [62,]        0 1.881258e-04
## [63,]        0 6.718039e-05
## [64,]        0 4.767934e-05
## [65,]        0 6.892362e-05
## [66,]        0 4.620945e-05
## [67,]        0 5.040982e-05
## [68,]        0 5.502358e-05
## [69,]        0 6.497065e-05
## [70,]        0 4.817688e-05
## [71,]        0 4.883417e-05
## [72,]        0 5.103125e-05
## [73,]        0 1.142503e-04
## [74,]        0 3.483053e-05
## [75,]        0 4.538004e-05
## [76,]        0 4.358062e-05
## [77,]        0 3.772975e-05
## [78,]        0 9.036233e-05
## [79,]        0 1.095758e-04
## [80,]        0 5.842371e-05
## [81,]        0 4.230658e-05
## [82,]        0 5.560191e-05
## [83,]        0 5.523466e-05
```

```
##  [84,]       0 8.876616e-05
##  [85,]       0 1.188986e-04
##  [86,]       0 7.465211e-05
##  [87,]       0 3.314827e-05
##  [88,]       0 4.577717e-05
##  [89,]       0 5.355614e-05
##  [90,]       0 5.010204e-05
##  [91,]       0 4.287172e-05
##  [92,]       0 4.080181e-05
##  [93,]       0 4.634219e-05
##  [94,]       0 3.973385e-05
##  [95,]       0 8.560771e-05
##  [96,]       0 7.777267e-05
##  [97,]       0 6.010377e-05
##  [98,]       0 4.120811e-05
##  [99,]       0 8.310839e-05
## [100,]       0 5.238254e-05
## [101,]       0 9.241253e-05
## [102,]       0 5.573194e-05
## [103,]       0 7.851748e-05
## [104,]       0 5.419524e-05
## [105,]       0 9.261278e-05
## [106,]       0 6.262879e-05
## [107,]       0 5.987134e-05
## [108,]       0 6.512857e-05
## [109,]       0 7.555126e-05
## [110,]       0 5.772463e-05
## [111,]       0 6.153935e-05
## [112,]       0 4.684681e-05
## [113,]       0 5.459322e-05
## [114,]       0 5.385093e-05
## [115,]       0 8.349387e-05
## [116,]       0 5.697890e-05
## [117,]       0 5.314116e-05
## [118,]       0 5.207198e-05
## [119,]       0 5.298167e-05
## [120,]       0 6.381532e-05
```

Multilevel 모델에 적용

- 논문의 방법인 EM알고리즘을 통해 multilevel spline 방법으로 최적의 μ 벡터를 찾았다.

```
#multilevel

  #beta_hat_vector 구하기
  grain_out <- NULL
  J=120
  beta_hat <- NULL
  for(m in 1:120){
    result2_out <- NULL
    results2 <- glm.nb(unlist(y_list[m]) ~ unlist(x_list[m]))
      kth_beta_hat <- coef(results2)[2]
      kth_var <- diag(vcov(results2))[2]
      grain_out <- list(kth_beta_hat, kth_var)
```

```r
      grain_out
    beta_hat <- rbind(beta_hat,grain_out)
  }

# x, y를 다시 리스트화
x_list <- exr[,-1]
x_list <- x_list[colSums(is.na(x_list))<nrow(x_list)] %>% t() %>% as.data.frame() %>% as.list()

y_list <- alldata[,-1]
y_list <- y_list[colSums(is.na(y_list))<nrow(y_list)] %>% t() %>% as.data.frame() %>% as.list()
```

x_list 최소값, 최대값 추출

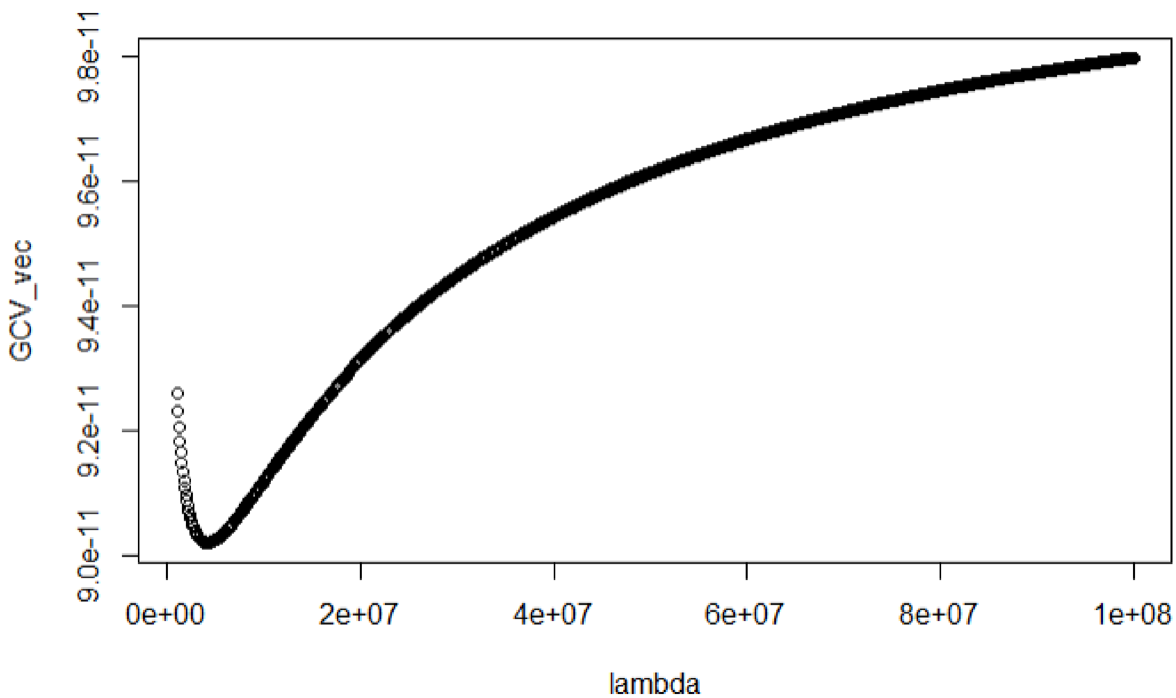
```r
# x의 범위로 make_K만들기
aaa <- unlist(x_list)
aaa <-  ifelse(is.na(aaa), mean(aaa, na.rm=TRUE), aaa)
min_x <- min(aaa)
max_x <- max(aaa)

      for(u in 1:120){
      z <- seq(min_x, max_x, length.out = 120)
      K <- mlsplines::make_K(z)
      }
```

최적의 GCV_vec 찾기

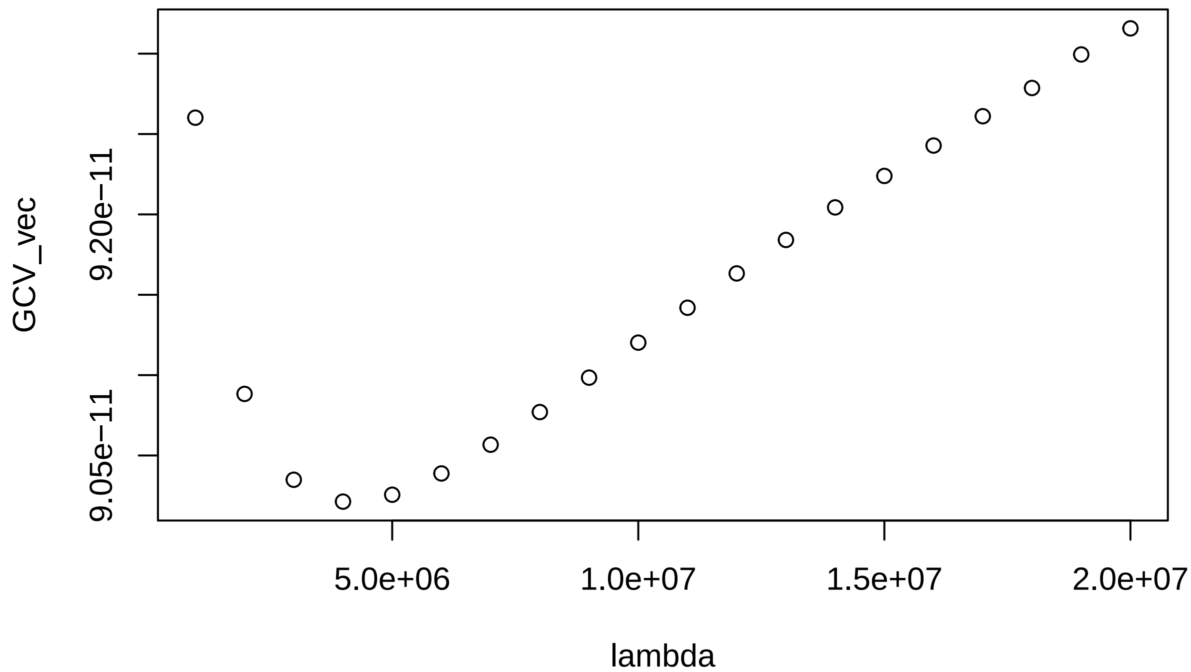
```r
lambda <- c(10^5, 10^6, 10^7, 10^8, 10^9)
  GCV_vec <- NULL

  for(i in 1:length(lambda)){
  EM_out <- mlsplines::main_EM(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])), K =
  GCV_vec <- rbind(GCV_vec,EM_out$GCV)
  }
```

- lambda[which.min(GCV_vec)]을 실행할 때, 10^7이 나온다.
- 그래서 10^7근처에서 GCV벡터를 더 찾아보기로 한다.

```r
knitr::include_graphics("11.PNG")
```

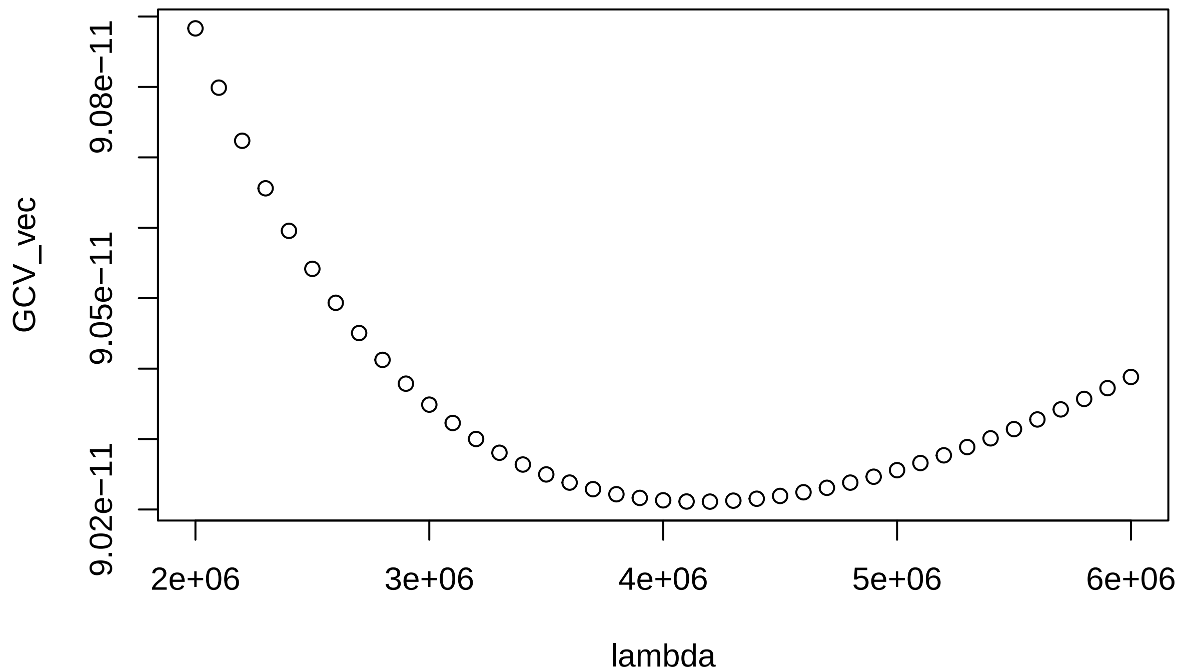



```
lambda <- seq(10^6, 2e+07, by=10^6)
  GCV_vec <- NULL

  for(i in 1:length(lambda)){
  EM_out <- mlsplines::main_EM(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])), K =
  GCV_vec <- rbind(GCV_vec,EM_out$GCV)
  }

plot(lambda, GCV_vec)
```



- 좀더 상세한 값을 위해 계속해서 찾는다.

```
lambda <- seq(2e+6, 6e+6, by=10^5)
  GCV_vec <- NULL

  for(i in 1:length(lambda)){
  EM_out <- mlsplines::main_EM(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])), K =
  GCV_vec <- rbind(GCV_vec,EM_out$GCV)
  }

plot(lambda, GCV_vec)
```



- 최적의 GCV_vec로 EM_out구하기 <- mu_hat 구함

```
EM_out <- mlsplines::main_EM(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])), K = K
EM_out$mu
```

```
##                 [,1]
##    [1,] -7.427522e-05
##    [2,] -7.009564e-05
##    [3,] -6.622116e-05
##    [4,] -6.247196e-05
##    [5,] -5.909422e-05
##    [6,] -5.684149e-05
##    [7,] -5.592085e-05
##    [8,] -5.571018e-05
##    [9,] -5.566451e-05
##   [10,] -5.561054e-05
##   [11,] -5.533990e-05
##   [12,] -5.468899e-05
##   [13,] -5.350190e-05
##   [14,] -5.156563e-05
##   [15,] -4.944430e-05
##   [16,] -4.769435e-05
##   [17,] -4.646352e-05
##   [18,] -4.559899e-05
##   [19,] -4.482009e-05
##   [20,] -4.362121e-05
##   [21,] -4.190682e-05
```

```
## [22,] -4.005951e-05
## [23,] -3.821602e-05
## [24,] -3.620174e-05
## [25,] -3.352750e-05
## [26,] -3.001180e-05
## [27,] -2.720859e-05
## [28,] -2.627585e-05
## [29,] -2.733484e-05
## [30,] -2.973345e-05
## [31,] -3.306552e-05
## [32,] -3.669581e-05
## [33,] -3.991425e-05
## [34,] -4.198098e-05
## [35,] -4.247702e-05
## [36,] -4.149453e-05
## [37,] -3.946914e-05
## [38,] -3.708545e-05
## [39,] -3.571693e-05
## [40,] -3.590254e-05
## [41,] -3.759787e-05
## [42,] -4.039134e-05
## [43,] -4.348298e-05
## [44,] -4.576265e-05
## [45,] -4.662139e-05
## [46,] -4.592819e-05
## [47,] -4.413192e-05
## [48,] -4.176075e-05
## [49,] -3.937223e-05
## [50,] -3.710838e-05
## [51,] -3.603731e-05
## [52,] -3.680348e-05
## [53,] -3.911145e-05
## [54,] -4.207383e-05
## [55,] -4.462506e-05
## [56,] -4.565200e-05
## [57,] -4.459129e-05
## [58,] -4.199536e-05
## [59,] -3.874990e-05
## [60,] -3.527457e-05
## [61,] -3.195251e-05
## [62,] -2.886834e-05
## [63,] -2.598172e-05
## [64,] -2.462604e-05
## [65,] -2.509749e-05
## [66,] -2.732105e-05
## [67,] -3.092124e-05
## [68,] -3.478100e-05
## [69,] -3.751796e-05
## [70,] -3.831573e-05
## [71,] -3.770525e-05
## [72,] -3.647505e-05
## [73,] -3.444480e-05
## [74,] -3.116925e-05
## [75,] -2.720298e-05
```

```
##  [76,] -2.417198e-05
##  [77,] -2.230152e-05
##  [78,] -2.146481e-05
##  [79,] -2.233224e-05
##  [80,] -2.448340e-05
##  [81,] -2.632830e-05
##  [82,] -2.709118e-05
##  [83,] -2.700555e-05
##  [84,] -2.615364e-05
##  [85,] -2.415241e-05
##  [86,] -2.110753e-05
##  [87,] -1.867063e-05
##  [88,] -1.775017e-05
##  [89,] -1.835634e-05
##  [90,] -1.960751e-05
##  [91,] -2.075561e-05
##  [92,] -2.102902e-05
##  [93,] -2.015201e-05
##  [94,] -1.902165e-05
##  [95,] -1.829992e-05
##  [96,] -1.735064e-05
##  [97,] -1.451899e-05
##  [98,] -8.644870e-06
##  [99,] -1.418345e-06
## [100,]  4.813357e-06
## [101,]  8.652594e-06
## [102,]  1.040972e-05
## [103,]  1.051627e-05
## [104,]  9.883227e-06
## [105,]  9.355451e-06
## [106,]  8.268125e-06
## [107,]  5.981831e-06
## [108,]  3.731549e-06
## [109,]  3.343161e-06
## [110,]  5.569690e-06
## [111,]  8.978010e-06
## [112,]  1.133909e-05
## [113,]  1.165345e-05
## [114,]  1.105671e-05
## [115,]  9.517038e-06
## [116,]  7.733556e-06
## [117,]  6.995289e-06
## [118,]  6.306414e-06
## [119,]  4.180327e-06
## [120,]  8.947412e-07
```

Naive Method(from “ydhwang/mlsplines” in Github)

- Multilevel과 성능을 비교하기위해서 Naive한 방법으로 구해보자.

```
#naive
GCV_vec <- NULL
lambda <- c(10^7,10^8,10^9,10^10,10^11)
for(i in 1:length(lambda)){
  naive_out <- mlsplines::naive_ss(beta_hat_vec = unlist(beta_hat[,1]), lambda = lambda[i], K = K)
  GCV_vec <- rbind(GCV_vec,naive_out$GCV)
}

lambda[which.min(GCV_vec)]
```

```
## [1] 1e+08
```

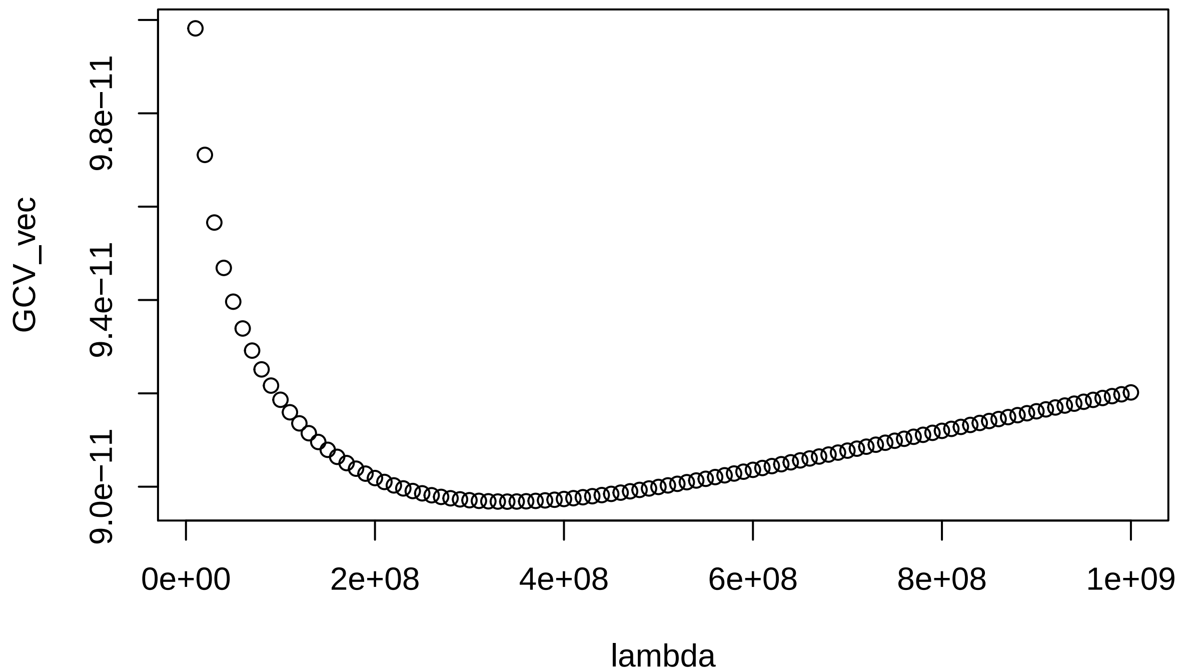
Naive’s GCV vector 찾기

- Naive 역시 비슷한 방법으로 풀어나간다.

```
GCV_vec <- NULL
lambda <- seq(10^7,10^9,by=10^7)
for(i in 1:length(lambda)){
  naive_out <- mlsplines::naive_ss(beta_hat_vec = unlist(beta_hat[,1]), lambda = lambda[i], K = K)
  GCV_vec <- rbind(GCV_vec,naive_out$GCV)
}
lambda[which.min(GCV_vec)]
```

```
## [1] 3.4e+08
```

```
plot(lambda, GCV_vec)
```




```r
naive_out <- mlsplines::naive_ss(beta_hat_vec = unlist(beta_hat[,1]), lambda = lambda[which.min(GCV_vec)

naive_out$mu
```

```
##                  [,1]
##   [1,] -7.494229e-05
##   [2,] -7.058187e-05
##   [3,] -6.678170e-05
##   [4,] -6.284697e-05
##   [5,] -5.897712e-05
##   [6,] -5.644235e-05
##   [7,] -5.565384e-05
##   [8,] -5.568000e-05
##   [9,] -5.576931e-05
##  [10,] -5.580761e-05
##  [11,] -5.560472e-05
##  [12,] -5.499757e-05
##  [13,] -5.378180e-05
##  [14,] -5.158627e-05
##  [15,] -4.918632e-05
##  [16,] -4.732987e-05
##  [17,] -4.618478e-05
##  [18,] -4.553886e-05
##  [19,] -4.503021e-05
##  [20,] -4.396843e-05
##  [21,] -4.225989e-05
```

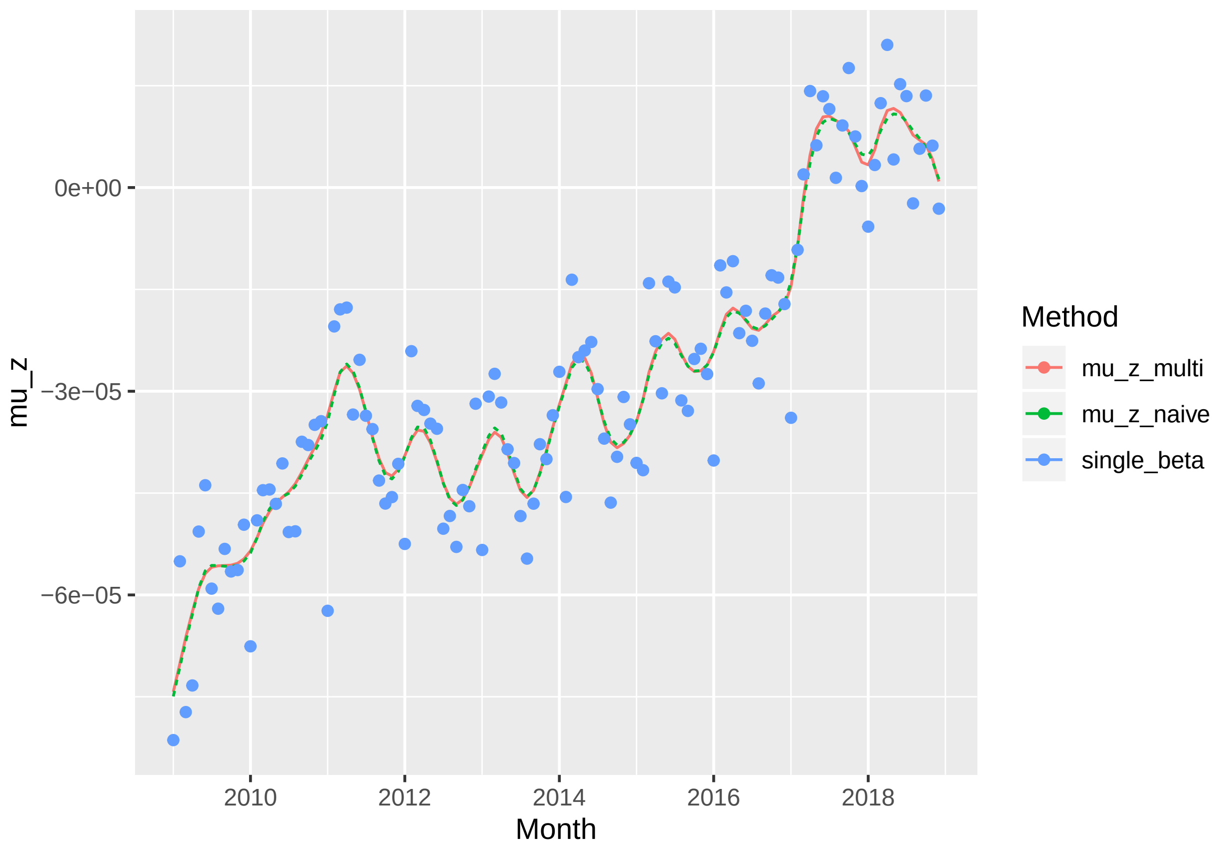
```
## [22,] -4.047928e-05
## [23,] -3.883279e-05
## [24,] -3.706669e-05
## [25,] -3.442286e-05
## [26,] -3.046859e-05
## [27,] -2.717473e-05
## [28,] -2.599269e-05
## [29,] -2.703199e-05
## [30,] -2.952609e-05
## [31,] -3.302754e-05
## [32,] -3.685502e-05
## [33,] -4.025177e-05
## [34,] -4.242110e-05
## [35,] -4.291713e-05
## [36,] -4.182947e-05
## [37,] -3.957939e-05
## [38,] -3.685621e-05
## [39,] -3.526569e-05
## [40,] -3.541242e-05
## [41,] -3.722459e-05
## [42,] -4.023731e-05
## [43,] -4.356807e-05
## [44,] -4.597902e-05
## [45,] -4.682442e-05
## [46,] -4.600104e-05
## [47,] -4.402061e-05
## [48,] -4.147194e-05
## [49,] -3.895769e-05
## [50,] -3.656545e-05
## [51,] -3.540873e-05
## [52,] -3.619594e-05
## [53,] -3.861028e-05
## [54,] -4.169920e-05
## [55,] -4.436698e-05
## [56,] -4.550477e-05
## [57,] -4.458820e-05
## [58,] -4.212619e-05
## [59,] -3.893609e-05
## [60,] -3.548876e-05
## [61,] -3.220548e-05
## [62,] -2.922056e-05
## [63,] -2.652566e-05
## [64,] -2.526300e-05
## [65,] -2.570674e-05
## [66,] -2.774745e-05
## [67,] -3.098167e-05
## [68,] -3.444728e-05
## [69,] -3.699599e-05
## [70,] -3.792212e-05
## [71,] -3.753742e-05
## [72,] -3.638809e-05
## [73,] -3.437525e-05
## [74,] -3.124022e-05
## [75,] -2.753985e-05
```

```
##  [76,] -2.466360e-05
##  [77,] -2.289759e-05
##  [78,] -2.218430e-05
##  [79,] -2.293416e-05
##  [80,] -2.472857e-05
##  [81,] -2.630830e-05
##  [82,] -2.701733e-05
##  [83,] -2.695711e-05
##  [84,] -2.613909e-05
##  [85,] -2.425222e-05
##  [86,] -2.142398e-05
##  [87,] -1.912117e-05
##  [88,] -1.814085e-05
##  [89,] -1.850407e-05
##  [90,] -1.950518e-05
##  [91,] -2.051427e-05
##  [92,] -2.089153e-05
##  [93,] -2.035729e-05
##  [94,] -1.941068e-05
##  [95,] -1.841167e-05
##  [96,] -1.692095e-05
##  [97,] -1.384703e-05
##  [98,] -8.483762e-06
##  [99,] -2.096491e-06
## [100,]  3.592695e-06
## [101,]  7.494546e-06
## [102,]  9.618145e-06
## [103,]  1.019947e-05
## [104,]  9.848113e-06
## [105,]  9.194238e-06
## [106,]  8.082532e-06
## [107,]  6.379993e-06
## [108,]  4.905172e-06
## [109,]  4.705326e-06
## [110,]  6.146456e-06
## [111,]  8.395550e-06
## [112,]  1.018889e-05
## [113,]  1.084857e-05
## [114,]  1.068295e-05
## [115,]  9.721268e-06
## [116,]  8.361802e-06
## [117,]  7.217337e-06
## [118,]  5.916351e-06
## [119,]  3.867192e-06
## [120,]  1.239381e-06
```

그래프

```
# hat_all
single_beta <- unlist(beta_hat[,1]) %>% as.vector()
mu_z_naive <- naive_out$mu %>% as.vector()
mu_z_multi <- EM_out$mu %>% as.vector()
```

```r
hat_all <- cbind(mu_z_naive,mu_z_multi,single_beta) %>% as.data.frame

test_mon <- fread("~\\Github\\main\\main\\fordata22.csv")
test_mon <- test_mon[,1]
hat_all <- cbind(test_mon,hat_all)

hat_all$Month <- parse_date_time(hat_all$Month, "ym")
hat_all$Month <- as.Date(hat_all$Month, format="%Y-%m-%d")
hat_all <- as.data.frame(hat_all)
hat_all <- hat_all %>% mutate_if(is.character,parse_number)

# gather 사용
df1 <- gather(hat_all[, c("Month", "mu_z_naive", "mu_z_multi")],
             key = "Method", value = "mu_z", -Month)

df2 <- cbind(test_mon,single_beta)
df2$Month <- parse_date_time(df2$Month, "ym")
df2$Month <- as.Date(df2$Month, format="%Y-%m-%d")
df2 <- as.data.frame(df2)
df2 <- df2 %>% mutate_if(is.character,parse_number)


g <- ggplot(df1) +
  geom_line(aes(x = Month, y = mu_z, color = Method, linetype = Method)) +
  geom_point(data=df2, aes(x = Month, y = single_beta, color = "single_beta")) +
  guides(linetype = "none") +
  scale_color_discrete(name = "Method")
g
```


0e+00
−3e−05
−6e−05
mu_z
2010
2012
2014
2016
2018
Month
Method
mu_z_multi
mu_z_naive
single_beta