Project for P-spline and Multilevel

Choi TaeYoung

2020-07-23

Contents

필요한 패키지

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 구성된 국가 중 GDP TOP100 국가를 추출했다. 그리고 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 2, 14, 48행 이외에는 0을 포함하지 않아서 허들모델이나 zero inflated 방법을 이용하여 모델을 적합할 수 없었다.
- 그래서 우리는 Goodness of Fit(GoF)를 이용하여 일반화 선형모형의 적합도를 검정해보았다.

```
x_list <- x_gdp[,-1] %>% as.data.frame() %>% unlist() %>% as.list()

#How many zero in y_list?

y_list <- obs_y[,-1] %>% t() %>% as.data.frame()
y_zero <- NULL

for(m in 1:120){
  zero <- NULL
  zero <- length(which(y_list[m] == 0))/100
  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] 3
```

GoF 결과

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

```
x_list <- x_gdp[,-1] %>% as.data.frame() %>% unlist() %>% as.list()
y_list <- obs_y[,-1] %>% t() %>% as.data.frame()

#GOF Calculate

goodness <- NULL

  for(m in 1:120){
    result1_out <- NULL
    result2_out <- NULL
    results1 <- glm(unlist(y_list[m]) ~ unlist(x_list), family = poisson, maxit=500)
    results2 <- glm.nb(unlist(y_list[m]) ~ unlist(x_list), maxit=500)

    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)
  }

tail(goodness)
```

```
##        poi_GOF      nb_GOF
## [115,]       0 0.009603768
## [116,]       0 0.010558781
## [117,]       0 0.010008418
## [118,]       0 0.008943346
## [119,]       0 0.007016419
## [120,]       0 0.005388898
```

Multilevel 모델에 적용

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

```
x_list <- x_gdp[,-1] %>% as.data.frame() %>% unlist() %>% as.list()
y_list <- obs_y[,-1] %>% t() %>% as.data.frame()
#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), maxit=500)
      kth_beta_hat <- coef(results2)[2]
```

```
      kth_var <- diag(vcov(results2))[2]
      grain_out <- list(kth_beta_hat, kth_var)
      grain_out
    beta_hat <- rbind(beta_hat,grain_out)
  }
```

- p-spline 기법을 활용하여 새롭게 짠 코드로 리얼데이터에 적용

```
lambda <- c(0.0001, 0.001, 0.01, 0.1 ,1, 10, 100, 10^3, 10^4, 10^5)
GCV_vec <- NULL

n <- length(unlist(beta_hat[,1]))
IK <- unlist(beta_hat[,1])[-c(1, n)]
EK <- unlist(beta_hat[,1])[c(1, n)]
B = GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK)

for(i in 1:length(lambda)){
EM_out <- main_EM_p(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])),
                    B=GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK), D=GetDiffMatrix(dim(B)[1], 2),
GCV_vec <- rbind(GCV_vec,EM_out$GCV)
}

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

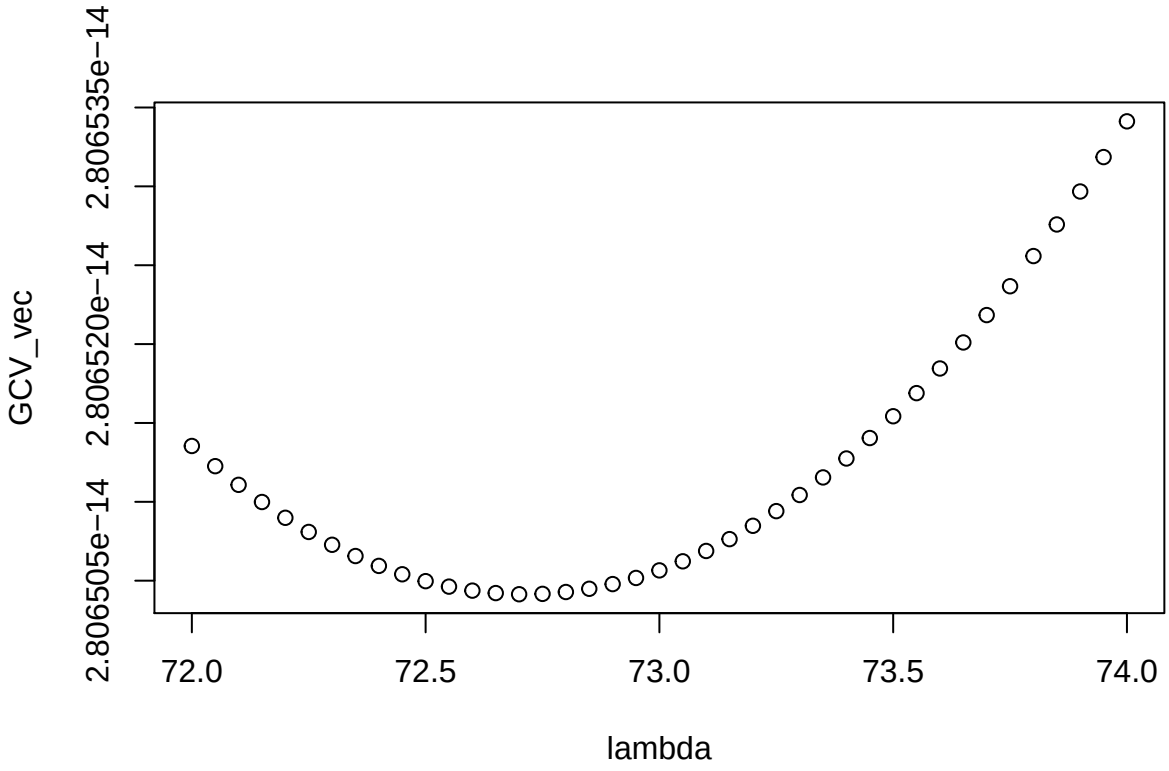
```
## [1] 100
```

- lambda[which.min(GCV_vec)]을 실행할 때, 100이 나온다.
- 그래서 100근처에서 GCV벡터를 더 찾아보기로 한다.

```
lambda <- seq(72, 74, by=.05)
  GCV_vec <- NULL

for(i in 1:length(lambda)){
EM_out <- main_EM_p(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])),
                    B=GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK), D=GetDiffMatrix(dim(B)[1], 2),
GCV_vec <- rbind(GCV_vec,EM_out$GCV)
}


plot(lambda, GCV_vec)
```



- 최적의 GCV_vec로 EM_out구하기 <- mu_hat 구함

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

```
## [1] 72.7
```

```
EM_out <- main_EM_p(beta_hat_vec = unlist(beta_hat[,1]), V = diag(unlist(beta_hat[,2])),
                    B=GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK), D=GetDiffMatrix(dim(B)[1], 2),
tail(EM_out$mu)
```

```
##                 [,1]
## [115,] 5.225773e-07
## [116,] 5.335386e-07
## [117,] 5.234308e-07
## [118,] 5.220587e-07
## [119,] 5.106113e-07
## [120,] 5.526022e-07
```

- Multilevel과 성능을 비교하기위해서 Naive한 방법으로 구해보자.
- Naive기법 역시 P-spline으로 코드를 짠 후 실행했다.

```
#naive
GCV_vec <- NULL
lambda <- c(0.0001, 0.001, 0.01, 0.1 ,1, 10, 100, 10^3, 10^4, 10^5)
for(i in 1:length(lambda)){
  naive_out <- naive_ss_p(beta_hat_vec = unlist(beta_hat[,1]), B=GetBSpline(unlist(beta_hat[,1]), deg =
  GCV_vec <- rbind(GCV_vec,naive_out$GCV)
}

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

```
## [1] 100
```

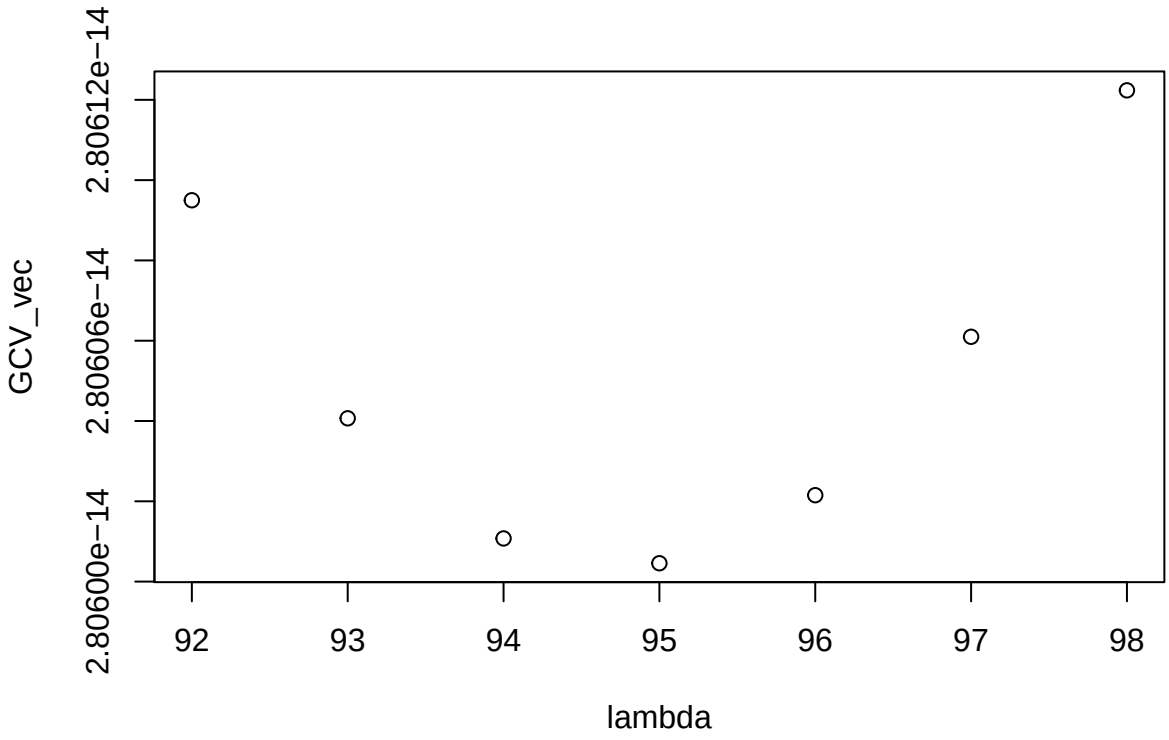
Naive's GCV vector 찾기

- Naive 역시 비슷한 방법으로 풀어나간다.

```
GCV_vec <- NULL
lambda <- seq(92, 98, by=1)
for(i in 1:length(lambda)){
  naive_out <- naive_ss_p(beta_hat_vec = unlist(beta_hat[,1]), lambda = lambda[i],
                          B=GetBSpline(unlist(beta_hat[,1]), deg = 3, IK, EK), D=GetDiffMatrix(dim(B)[1]
  GCV_vec <- rbind(GCV_vec,naive_out$GCV)
}
lambda[which.min(GCV_vec)]
```

```
## [1] 95
```

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




```
naive_out <- naive_ss_p(beta_hat_vec = unlist(beta_hat[,1]), B=GetBSpline(unlist(beta_hat[,1]), deg = 3


tail(naive_out$mu)
```

```
##                [,1]
## [115,] 5.223756e-07
## [116,] 5.336052e-07
## [117,] 5.235909e-07
## [118,] 5.218619e-07
## [119,] 5.109880e-07
## [120,] 5.515082e-07
```

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()

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

test_mon <- fread("S:\\Teo\\OneDrive - 성균관대학교\\1석사\\연구\\내 연구\\PSpline\\Computing\\obs_y.csv")
test_mon <- test_mon[,1]
hat_all <- cbind(test_mon,hat_all)
```

```r
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
```