Gibbs Sampling Algorithm

5/4/2022

```r
hurr_df <- read_csv("./data/hurricane703.csv") %>%
  drop_na() %>%
  group_by(ID) %>%
  filter(n() > 1) %>%
  # create delta variables and wind lag - wind speed at time t, Y(t)
  mutate(lat_change = c(0, diff(Latitude, lag = 1)), # I put zero for first entry, could be NA
         lng_change = c(0, diff(Longitude, lag = 1)),
         wind_change = c(0, diff(Wind.kt, lag = 1)),
         wind_lag = lag(Wind.kt, n = 1)) %>%
  dplyr::select(ID, lat_change, lng_change, wind_change, wind_lag, Wind.kt) %>%
  nest(y = Wind.kt, x = lat_change:wind_lag) %>%
  mutate(x = map(.x = x, .f = ~model.matrix(~., data = .x)),
         y = map(.x = y, .f = pull))
```

```
## Rows: 22038 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): ID, Month, Nature, time
## dbl (4): Season, Latitude, Longitude, Wind.kt
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```r
hurr_list <- list(
  y = hurr_df$y,
  x = hurr_df$x
)


# you can extract data like this
hurr_list$x[[1]]
```

```
##    (Intercept) lat_change lng_change wind_change wind_lag
## 2            1        0.6       -0.8           5       35
## 3            1        0.5       -1.1           5       40
## 4            1        0.8       -1.2           5       45
## 5            1        1.0       -1.4           0       50
## 6            1        0.7       -1.1           0       50
## 7            1        0.6       -1.1           5       50
## 8            1        0.7       -1.0           0       55
## 9            1        0.7       -0.6           5       55
## 10           1        0.4       -0.8           0       60
## 11           1        0.3       -0.8           0       60
## 12           1        0.5       -0.6           5       60
## 13           1        0.5       -0.2           0       65
## 14           1        0.4       -0.3           0       65
## 15           1        0.4       -0.3           5       65
## 16           1        0.3       -0.7           5       70
## 17           1        0.2       -0.6           5       75
## 18           1        0.0       -0.6           0       80
## 19           1       -0.2       -0.6           0       80
## 20           1       -0.1       -0.5           5       80
## 21           1        0.0       -0.8           5       85
## 22           1        0.0       -0.9           5       90
## 23           1        0.1       -1.1           5       95
## 24           1        0.4       -0.7           0      100
## 25           1        0.8       -0.6           0      100
## 26           1        0.6       -0.5           0      100
## 27           1        0.6       -0.5           0      100
## 28           1        0.5       -0.4           5      100
## 29           1        0.7       -0.2           0      105
## 30           1        0.8        0.0           0      105
## 31           1        0.8        0.0           0      105
## 32           1        1.0        0.0           5      105
## 33           1        1.1        0.3           0      110
## 34           1        1.6        0.9          -5      110
## 35           1        1.6        1.6          -5      105
## 36           1        1.6        1.7         -10      100
## 37           1        1.7        1.6         -10       90
## 38           1        1.9        2.1         -10       80
## 39           1        2.1        2.3          -5       70
## 40           1        1.3        1.3           0       65
## 41           1        0.9        1.1         -20       65
## 42           1        2.4        2.8         -10       45
## 43           1        2.1        3.0          -5       35
## 44           1        2.0        3.0           0       30
## 45           1        1.6        3.1           0       30
## 46           1        1.1        3.0           5       30
## 47           1        0.6        2.9           0       35
## 48           1        0.0        3.0           0       35
## 49           1       -0.8        4.1          -5       35
## 50           1       -1.0        4.0           0       30
## 51           1       -1.0        3.4           0       30
## attr(,"assign")
## [1] 0 1 2 3 4
```

```
hurr_list$y[[1]]
```

```
##  [1]  35  40  45  50  50  50  55  55  60  60  60  65  65  65  70  75  80  80  80
## [20]  85  90  95 100 100 100 100 100 105 105 105 105 110 110 105 100  90  80  70
## [39]  65  65  45  35  30  30  30  35  35  35  30  30  30
```

```
# list of theta parameters
theta <- list(
  "beta" = list("beta" = matrix(data = 1, nrow = 1, ncol = 5),
                "mu" = 0,
                "sigma" = 0),
  "B" = list(matrix(data = 1, nrow = nrow(hurr_df), ncol = 5),
             "mu" = 0,
             "sigma" = 0),
  "sigma2" = c(3),
  "sigma_m" = matrix(data = 1, nrow = 5, ncol = 5) + diag(1, 5, 5)
)
```

```
# log posterior of sigma^2
log_sigma2 <- function(data, i){
  # alpha parameter of distribution
  alpha = 1 + sum(dim(data[[i]]$y)) / 2
  # beta parameter of distribution
  b = (1/2) * sum(t(data[[i]]$y - data[[i]]$x %*% t(theta$beta)) %*% diag(length(data[[i]]$y)) %*% (data
  # pulling sigma^2 from theta list
  tau = theta$sigma2

  return(alpha * log(b) - log(gamma(alpha)) + (alpha - 1) * log(tau) - tau * b)
}

# log posterior of beta
log_beta <- function(){
  v = sum(theta$sigma_m)
  u = sum(theta$sigma_m %*% t(theta$beta))

  return(-(1/2) * log(det(2 * pi * solve(v))) - (1/2) * t((theta$beta - solve(v) %*% u)) %*% solve(v) %
}

# log posterior of B
log_B <- function(data, i){
  v = t(data$x[[i]]) %*% (1 / diag(theta$sigma2)) %*% data[[i]]$x + theta$sigma_m
  u = data[[i]]$x %*% (1 / diag(theta$sigma2)) %*% data[[i]]$y + theta$sigma_m %*% t(theta$beta)

  return(-(1/2) * log(det(2 * pi * solve(v))) - (1/2) * t((theta$B - solve(v) %*% u)) %*% solve(v) %*%
}
```

```
beta_dist <- function(sigma, B){
  # sigma stored as inverse
  v = nrow(hurr_df) * sigma

  u_matrix <- matrix(0, nrow = 5, ncol = nrow(hurr_df))
  for(i in 1:nrow(hurr_df)){

    u_matrix[,i] = sigma %*% t(matrix(B[i,], ncol = 5))

  }

  u = rowSums(u_matrix)

  mu = (solve(sigma, tol = 1e-95) / nrow(hurr_df)) %*% matrix(u, nrow = 5)
  sigma = solve(sigma, tol = 1e-95) / nrow(hurr_df)

  sample = mvrnorm(n = 1, mu, sigma)
  return(list("beta" = sample,
              "mu" = mu,
              "sigma" = sigma))
}

B_dist <- function(x, y, sigma2, sigma_m, beta){
  B_list <- list()

  v_list <- list()
  for(i in 1:nrow(hurr_df)){
    if(length(y[[i]]) < 2){
      next
    }
    attr(x[[i]], which = "assign") <- NULL
    attr(x[[i]], which = "dimnames") <- NULL
    y_adj <- y[[i]][-1]

    mat <- diag(sigma2, nrow = length(y_adj), ncol = length(y_adj))

    u <- t(x[[i]]) %*% solve(mat, tol = 1e-95) %*% y_adj + sigma_m %*% t(beta)
    v <- t(x[[i]]) %*% solve(mat, tol = 1e-95) %*% x[[i]] + sigma_m

    mu <- solve(v, tol = 1e-95) %*% u
    sigma <- solve(v, tol = 1e-95)
    v_list[[i]] <- v

    B_list[[i]] <- mvrnorm(n = 1, mu, sigma)

  }

  B_matrix <- do.call(rbind, B_list)

  return(list("B" = B_matrix,
              "mu" = mu,
              "v" = v_list))

}

sigma2_dist <- function(B, x, y){
  length_vec = sapply(y, length)

  beta_vec <- c()
  for(i in 1:nrow(hurr_df)){
    attr(x[[i]], which = "assign") <- NULL
    attr(x[[i]], which = "dimnames") <- NULL
    y_adj <- y[[i]][-1]

    res <- t(y_adj - x[[i]] %*% B[i,]) %*% diag(length(y_adj)) %*% (y_adj - x[[i]] %*% (B[i,]))

    beta_vec[i] <- res
  }

  # alpha parameter of distribution
  alpha = sum(length_vec) / 2
  # beta parameter of distribution
  b = (1/2) * sum(beta_vec)

  sample = rinvgamma(n = 1, shape = alpha, scale = b)
  return(sample)
}

sigma_m_dist <- function(beta, B){
 prev_matrix <- matrix(0, nrow = 5, ncol = 5)

 for(i in 1:nrow(hurr_df)){

   matrix <- t(B[i,] - beta) %*% (B[i,] - beta) + prev_matrix

   prev_matrix <- matrix
 }
 matrix <- matrix + diag(1, 5, 5)

 #mat_param <- solve(matrix)

 n = nrow(hurr_df)

 sample = matrix(riwish(v = n + 6, S = matrix),
                 ncol = 5)

 #sample = matrix(rWishart(n = 1, df = n + 6, Sigma = mat_param),
 #                 ncol = 5)

  return(sample)
}
```

```
beta_dist
```

```
## function(sigma, B){
##   # sigma stored as inverse
##   v = nrow(hurr_df) * sigma
##
##   u_matrix <- matrix(0, nrow = 5, ncol = nrow(hurr_df))
##   for(i in 1:nrow(hurr_df)){
##
##     u_matrix[,i] = sigma %*% t(matrix(B[i,], ncol = 5))
##
##   }
##
##   u = rowSums(u_matrix)
##
##   mu = (solve(sigma, tol = 1e-95) / nrow(hurr_df)) %*% matrix(u, nrow = 5)
##   sigma = solve(sigma, tol = 1e-95) / nrow(hurr_df)
##
##   sample = mvrnorm(n = 1, mu, sigma)
##
##   return(list("beta" = sample,
##                  "mu" = mu,
##                  "sigma" = sigma))
## }
```

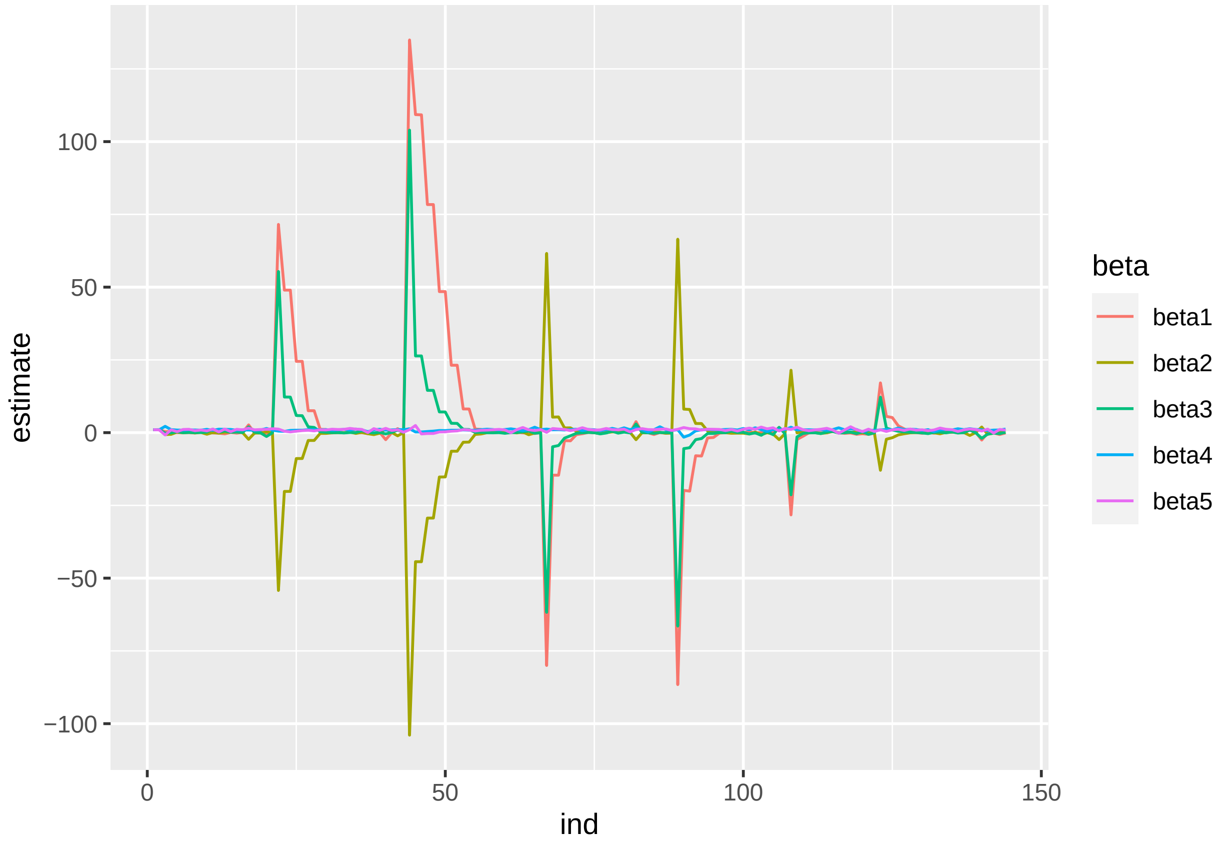
```
gibbs_fun <- function(iter, start, data){
  set.seed(052022)
  beta_list <- list()
  sigma_list <- list()
  sigma2_list <- list()
  B_list <- list()

  beta_list[[1]] <- start$beta
  sigma_list[[1]] <- start$sigma_m
  sigma2_list[[1]] <- start$sigma2
  B_list[[1]] <- theta$B
  # added code to catch data before error
  tryCatch(expr = {
  for (i in 2:iter){
    #browser()
    beta_list[[i]] <- beta_dist(sigma = sigma_list[[i-1]], B = B_list[[i-1]][[1]])
    sigma_list[[i]] <- sigma_m_dist(beta = matrix(beta_list[[i]][[1]], ncol = 5), B = B_list[[i-1]][[1]
    sigma2_list[[i]] <- sigma2_dist(B = B_list[[i-1]][[1]], x = data$x, y = data$y)
    B_list[[i]] <- B_dist(x = data$x, y = data$y, sigma2 = sigma2_list[[i]],
                          sigma_m = sigma_list[[i]], beta = matrix(beta_list[[i]][[1]], ncol = 5))
  }
  },
  error = function(e){print(e)},
  finally = list(beta = beta_list, sigma_m = sigma_list,
              sigma2 = sigma2_list, B = B_list)
  )
  return(list(beta = beta_list, sigma_m = sigma_list,
              sigma2 = sigma2_list, B = B_list))

}
test <- gibbs_fun(iter = 1000, start = theta, data = hurr_list)
```

```
## <simpleError in mvrnorm(n = 1, mu, sigma): 'Sigma' is not positive definite>
```

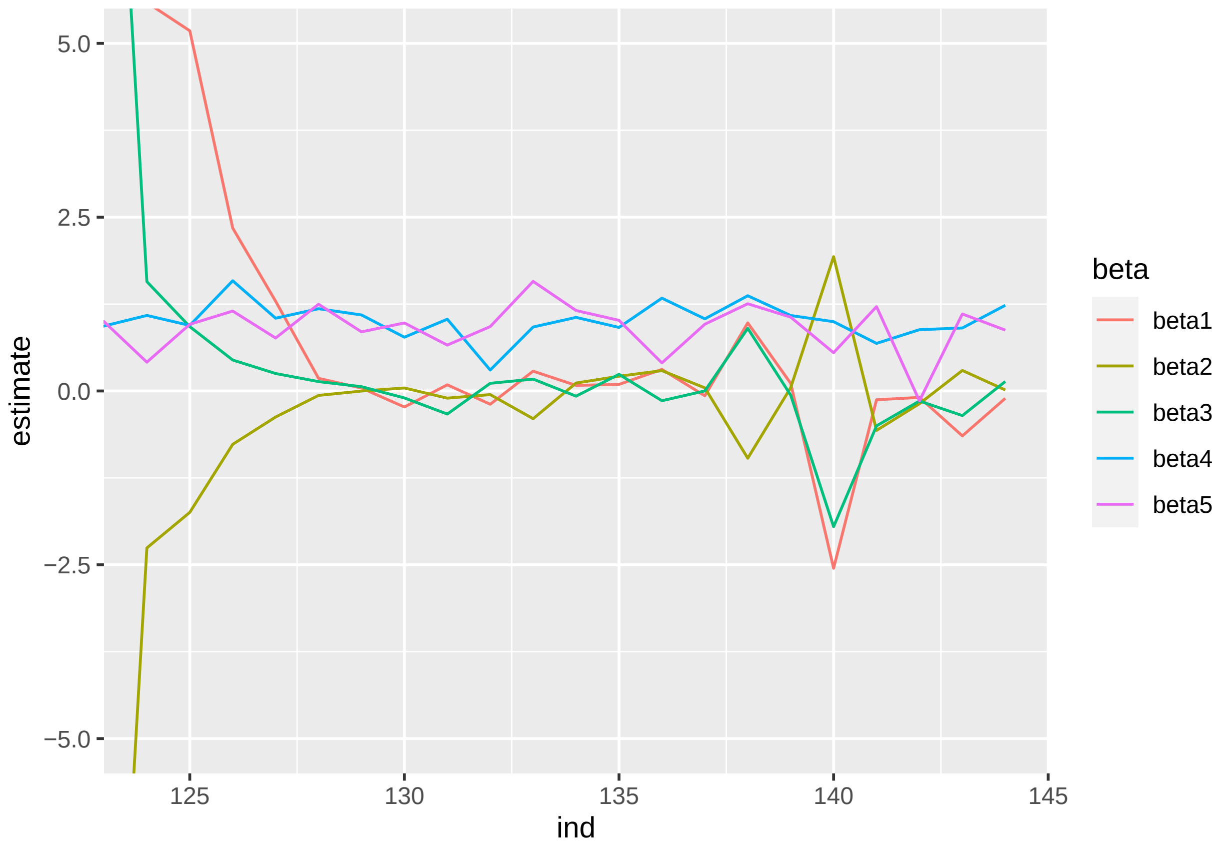
```
# the following code reproduced error for me in B_dist function
B_dist(hurr_list$x, hurr_list$y, test$sigma2[[144]], test$sigma_m[[144]], matrix(test$beta[[144]]$beta,
```

```
## Error in mvrnorm(n = 1, mu, sigma): 'Sigma' is not positive definite
```

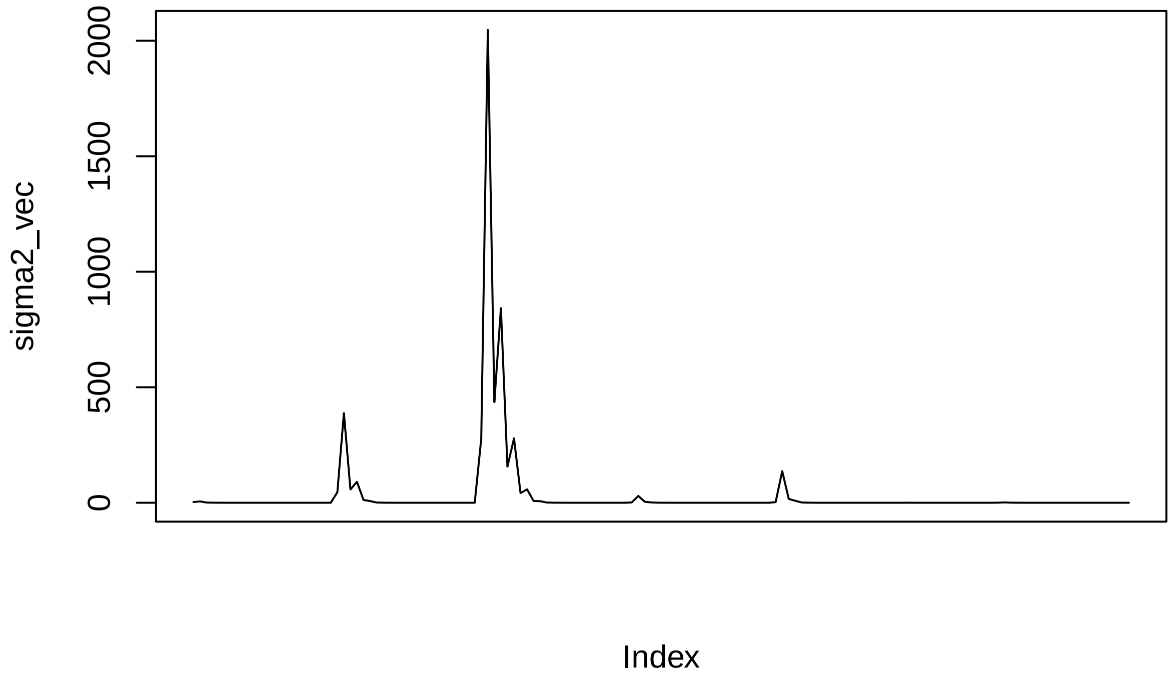
```
# function to extract beta values from gibbs output
beta_compiler <- function(list){
  beta_res_list <- list()

  for (i in 1:length(list)){
    beta_res_list[[i]] <- list[[i]]$beta
  }

  return(beta_res_list)
}
# visualizing beta values
beta_res_list <- beta_compiler(list = test$beta)
beta_vec <- data.frame(do.call(rbind, beta_res_list)) %>%
  rename(beta1 = 1, beta2 = 2, beta3 = 3, beta4 = 4, beta5 = 5) %>%
  pivot_longer(cols = beta1:beta5, names_to = "beta", values_to = "estimate") %>%
  group_by(beta) %>%
  mutate(ind = c(1:length(beta_res_list)))

ggplot(data = beta_vec, aes(y = estimate, col = beta)) +
  geom_line(aes(ind))
```



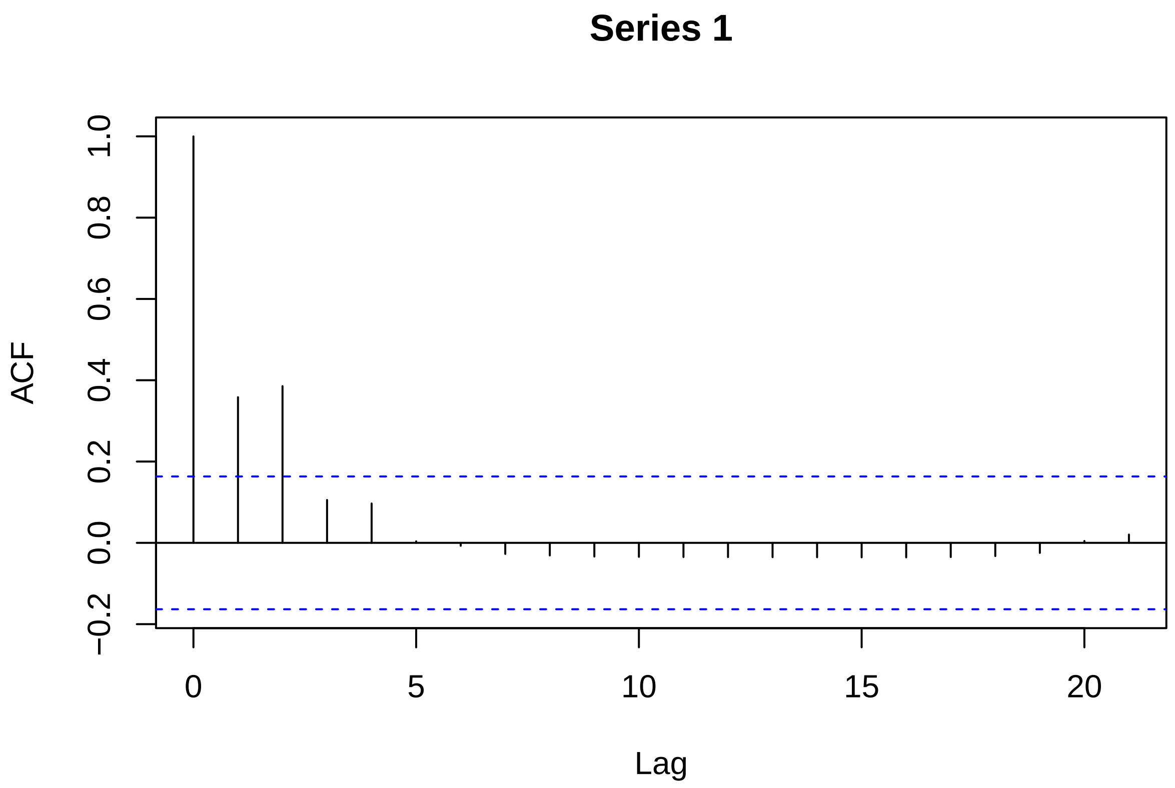
```
# zooming in on end of graph
ggplot(data = beta_vec, aes(y = estimate, col = beta)) +
  geom_line(aes(ind)) +
  coord_cartesian(xlim = c(length(beta_res_list) - 20, length(beta_res_list)),
                  ylim = c(-5, 5))
```



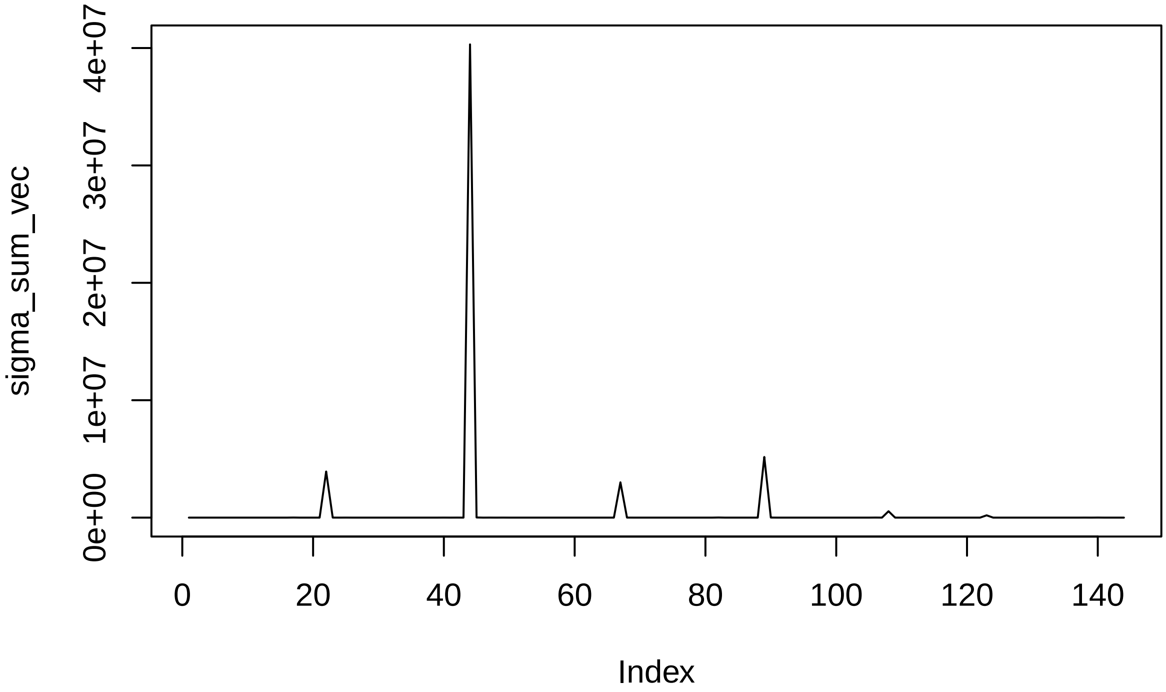
```
# visualizing sigma2 and sigma values
sigma2_vec <- do.call(rbind, test$sigma2)
plot(x = sigma2_vec, xaxt = "n", type = "l")
```



```
acf(sigma2_vec, pl = TRUE)
```



```
sigma_sum_vec <- do.call(rbind, lapply(test$sigma_m, sum))
plot(x = sigma_sum_vec, type = "l")
```


sigma_sum_vec
0e+00
1e+07
2e+07
3e+07
4e+07
0
20
40
60
80
100
120
140
Index