MCMC

Data preprocessing (same as EDA)

```r
origin_df <- read.csv("hurrican703.csv")

hurricane_df <- origin_df %>%
  mutate(
    Active = ifelse(Month %in% month.name[8:10], "Active", "Inactive"),
    Active = factor(Active, levels = c("Inactive", "Active")),
    Month = factor(Month, levels = month.name[-c(2:3)]), # April-January (January ref, may choose anoth
    Nature = as.factor(Nature), # TS,ET,DS,SS,NR (DS ref, may choose another)
    # note: one hurricane can have multiple natures throughout its life
    time = gsub("[()]", "", time),
    time = paste0(ifelse(substr(time, 1, 2) > 23, "19", "20"), time),
    time = as.POSIXct(time, format = "%Y-%m-%d %H:%M:%S"),
    hour = substr(time, 12, 19)
  ) %>%
  # remove data not at six-hour time intervals. (613 observations)
  filter(hour %in% c("00:00:00", "06:00:00", "12:00:00", "18:00:00")) %>%
  dplyr::select(-hour)

# remove hurricanes that has only 2 (<3) observations (change the threshold if you wish)
few_id <- hurricane_df %>%
  group_by(ID) %>%
  summarize(obs = n()) %>%
  filter(obs < 3) %>%
  .$ID
hurricane_df <- hurricane_df %>% filter(!(ID %in% few_id)) # remove 3 hurricanes

# manually correct hurricanes that have same names but are actually different
hurricane_df <-
  hurricane_df %>%
  mutate(
    # 2 hurricanes with the name ALICE.1954
    ID = ifelse(ID == "ALICE.1954" & Month == "June", "ALICE.1954(1)", ID),
    ID = ifelse(ID == "ALICE.1954", "ALICE.1954(2)", ID),
    # 4 hurricanes with the name SUBTROP:UNNAMED.1974
    ID = ifelse(ID == "SUBTROP:UNNAMED.1974" & Month == "June", "SUBTROP:UNNAMED.1974(1)", ID),
    ID = ifelse(ID == "SUBTROP:UNNAMED.1974" & Month == "July", "SUBTROP:UNNAMED.1974(2)", ID),
    ID = ifelse(ID == "SUBTROP:UNNAMED.1974" & Month == "August", "SUBTROP:UNNAMED.1974(3)", ID),
    ID = ifelse(ID == "SUBTROP:UNNAMED.1974", "SUBTROP:UNNAMED.1974(4)", ID),
    # 2 hurricanes with the name SUBTROP:UNNAMED.1976
    ID = ifelse(ID == "SUBTROP:UNNAMED.1976" & Month == "May", "SUBTROP:UNNAMED.1976(1)", ID),
    ID = ifelse(ID == "SUBTROP:UNNAMED.1976", "SUBTROP:UNNAMED.1976(2)", ID)
  ) %>%
  mutate(Season = Season - 1950) # scale the year
```

Data partition (may be useless.)

```
# data partition, no need to partition data
Training <- hurricane_df
```

Create X, Y, Z and $\boldsymbol{m} = (m_i)$ in R

```
n <- length(unique(Training$ID))

Y <- split(Training$Wind.kt, Training$ID) %>%
  lapply(function(x) x[-c(1:2)])

X <- Training %>%
  group_by(ID) %>%
  slice(1) %>%
  dplyr::select(ID, Season, Active, Nature) %>%
  ungroup(ID)
X <- model.matrix(~., X[-1])[,-1]

Z <- Training %>%
  group_by(ID) %>%
  mutate(
    intercept = 1,
    wind_pre = lag(Wind.kt),
    lat_diff = lag(Latitude) - lag(Latitude, 2),
    long_diff = lag(Longitude) - lag(Longitude, 2),
    wind_diff = lag(Wind.kt) - lag(Wind.kt, 2),
  ) %>%
  drop_na %>%
  dplyr::select(ID, intercept, wind_pre, lat_diff, long_diff, wind_diff)
Z <- split(Z[, names(Z)[-1]], Z$ID) %>%
  lapply(as.matrix)

m <- Training %>%
  group_by(ID) %>%
  summarize(obs = n() - 2) %>%
  .$obs
```

MCMC

```
B_sample <- function(mu, Sigma, gamma, sigma) {
  Sigma.inv <- solve(Sigma)
  B_mean_cov <- function(i) {
    cov <- solve(Sigma.inv + 1/sigma^2 * t(Z[[i]]) %*% Z[[i]])
    mean <- cov %*% (Sigma.inv %*% mu + 1/sigma^2 * colSums((Y[[i]] - (X[i,] %*% gamma)[,]) * Z[[i]]))
    list(mean = mean, cov = cov)
  }
  mean_cov_list <- lapply(1:n, B_mean_cov)
  B <- sapply(mean_cov_list, function(x) {mvrnorm(mu = x$mean, Sigma = x$cov)})
```

```r
  return(B)
}

mu_sample <- function(B, Sigma) {
  cov <- V %*% solve(n*V + Sigma)
  mean <- cov %*% rowSums(B)
  mu <- mvrnorm(mu = mean, Sigma = cov)
  return(mu)
}

Sigma_sample <- function(B, mu) {
  Sigma.inv <- rWishart(n = 1, Sigma = solve(S + (B - mu) %*% t(B - mu)), df = n + nu)[,,]
  Sigma <- solve(Sigma.inv)
  return(Sigma)
}

gamma_sample <- function(B, sigma) {
  X_trans <- sqrt(m) * X
  cov <- solve(400*diag(6) + 1/sigma^2 * t(X_trans) %*% X_trans)
  total <- rowSums(sapply(1:n, function(i) sum(Z[[i]] %*% B[,i] - Y[[i]]) * X[i,]))
  mean <- cov %*% (-1/sigma^2 * total)
  gamma <- mvrnorm(mu = mean, Sigma = cov)
  return(gamma)
}

# MH algorithm (random walk)
sigma_sample <- function(sigma, B, gamma, a) {
  sigma_new <- sigma + (runif(1) - 0.5) * 2 * a # candidate sigma
  if (sigma_new <= 0) {
    return(sigma)
  }
  RSS <- sum(sapply(1:n, function(i) sum((Y[[i]] - Z[[i]] %*% B[,i] - (X[i,] %*% gamma)[,])^2)))
  log_kernal_ratio <- -sum(m) * log(sigma_new/sigma) +
    log(1 + (sigma/10)^2) - log(1 + (sigma_new/10)^2) -
    0.5 * (1/sigma_new^2 - 1/sigma^2) * RSS
  log_prob <- min(0, log_kernal_ratio)
  sigma <- ifelse(log_prob > log(runif(1)), sigma_new, sigma)
  return(sigma)
}

MCMC <- function(B0, mu0, Sigma0, gamma0, sigma0, a, iter) {
  B <- B0
  mu <- mu0
  Sigma <- Sigma0
  gamma <- gamma0
  sigma <- sigma0
  res <- vector("list", iter)
  for (i in 1:iter) {
    B <- B_sample(mu, Sigma, gamma, sigma)
    mu <- mu_sample(B, Sigma)
    Sigma <- Sigma_sample(B, mu)
    gamma <- gamma_sample(B, sigma)
    sigma <- sigma_sample(sigma, B, gamma, a)
    res[[i]] <- list(B = B, mu = mu, Sigma = Sigma, gamma = gamma, sigma = sigma)
  }
  return(res)
}
```

Fit lmm model to select starting values

```
df <- hurricane_df %>%
  group_by(ID) %>%
  mutate(
    Season = first(Season),
    Active = first(Active),
    Nature = first(Nature),
    wind_pre = lag(Wind.kt),
    lat_diff = lag(Latitude) - lag(Latitude, 2),
    long_diff = lag(Longitude) - lag(Longitude, 2),
    wind_diff = lag(Wind.kt) - lag(Wind.kt, 2),
  ) %>%
  drop_na %>%
  dplyr::select(ID, Wind.kt, Season, Active, Nature, wind_pre, lat_diff, long_diff, wind_diff)

lmm <- lmer(Wind.kt ~ Season + Active + Nature + wind_pre + lat_diff + long_diff + wind_diff +
            (1 + wind_pre + lat_diff + long_diff + wind_diff | ID),
            data = df)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
summary(lmm)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## Wind.kt ~ Season + Active + Nature + wind_pre + lat_diff + long_diff +
##     wind_diff + (1 + wind_pre + lat_diff + long_diff + wind_diff |      ID)
##    Data: df
##
## REML criterion at convergence: 126056.7
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -11.6203  -0.3792  -0.0443   0.4735  10.1111
##
## Random effects:
##  Groups   Name        Variance  Std.Dev. Corr
##  ID       (Intercept) 2.728e-01 0.52234
##           wind_pre    3.682e-04 0.01919  -0.87
##           lat_diff    3.238e-02 0.17995   0.30 -0.74
##           long_diff   6.932e-02 0.26329   0.88 -0.97  0.66
##           wind_diff   1.694e-02 0.13016   0.39  0.08 -0.68  0.08
##  Residual             2.838e+01 5.32707
## Number of obs: 20283, groups:  ID, 704
```

```
##
## Fixed effects:
##               Estimate Std. Error t value
## (Intercept)   2.899681   0.240848  12.039
## Season       -0.010951   0.002481  -4.415
## ActiveActive  0.346239   0.108215   3.200
## NatureET      0.275297   0.312523   0.881
## NatureNR      0.371966   0.564070   0.659
## NatureSS      0.120112   0.228404   0.526
## NatureTS      0.081464   0.170351   0.478
## wind_pre      0.941229   0.002016 466.818
## lat_diff     -0.023260   0.063885  -0.364
## long_diff    -0.241753   0.033754  -7.162
## wind_diff     0.466916   0.008778  53.191
##
## Correlation of Fixed Effects:
##             (Intr) Season ActvAc NatrET NatrNR NatrSS NatrTS wnd_pr lt_dff
## Season      -0.634
## ActiveActiv -0.299  0.021
## NatureET    -0.362  0.040  0.097
## NatureNR    -0.206  0.045  0.039  0.134
## NatureSS    -0.569  0.234  0.001  0.339  0.187
## NatureTS    -0.742  0.346 -0.050  0.442  0.253  0.674
## wind_pre    -0.346  0.093 -0.096 -0.015 -0.008  0.014 -0.018
## lat_diff    -0.064 -0.010 -0.018  0.044 -0.005  0.045 -0.026 -0.171
## long_diff    0.088  0.018  0.031 -0.064  0.030 -0.090  0.004 -0.201 -0.321
## wind_diff    0.027 -0.004  0.020 -0.005  0.009 -0.023  0.009 -0.043 -0.027
##             lng_df
## Season
## ActiveActiv
## NatureET
## NatureNR
## NatureSS
## NatureTS
## wind_pre
## lat_diff
## long_diff
## wind_diff    0.108
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
mu0 <- as.vector(fixed.effects(lmm)[-c(2:7)])
gamma0 <- as.vector(fixed.effects(lmm)[2:7])
n <- length(unique(df$ID))
B0 <- matrix(rep(mu0, n), nrow = 5)
B0 <- B0 + t(as.matrix(random.effects(lmm)$ID))
sigma0 <- sd(residuals(lmm))
Sigma0 <- as.matrix(Matrix::bdiag(VarCorr(lmm))) + diag(5)*0.001 # computationally singular
```

Apply MCMC on training data

1. Please select appropriate $\boldsymbol{V}$ and $\boldsymbol{S}$ values.
2. Please select appropriate starting values for parameters, especially $\mathbf{B}$.

3. You can save the results `xxx_list` as csv files or otherwise. Note that we need to find the answers to the above 2 problems and also the following questions to make sure that we won't make any updates for these saved data:

- Should we apply MCMC on training hurricanes or all hurricanes? (in other words, should we partition the data into training and test data?) This relies on whether we hope to make predictions only on test hurricanes or on hurricanes used for MCMC.
- Given that samples of **B** is a 10000*2815 matrix, and we only use **B** for predictions on hurricanes used for MCMC, we can decide which hurricanes we are gonna predict (e.g., hurricanes having appropriate number of observations, famous hurricanes, hurricanes of different natures, years, months), and only save the samples of corresponding $\boldsymbol{\beta}_i$'s.
- We need to run the MCMC function and save the results (`xxx_list`) after we solve the above 3 problems. Then we need to check whether the `a` in the random walk for sampling σ is appropriate. (acceptance rate 30% - 60%, see far below) If not, we need to change `a` and regenerate the results.

estimated running time: 10 mins for 10,000 iterations

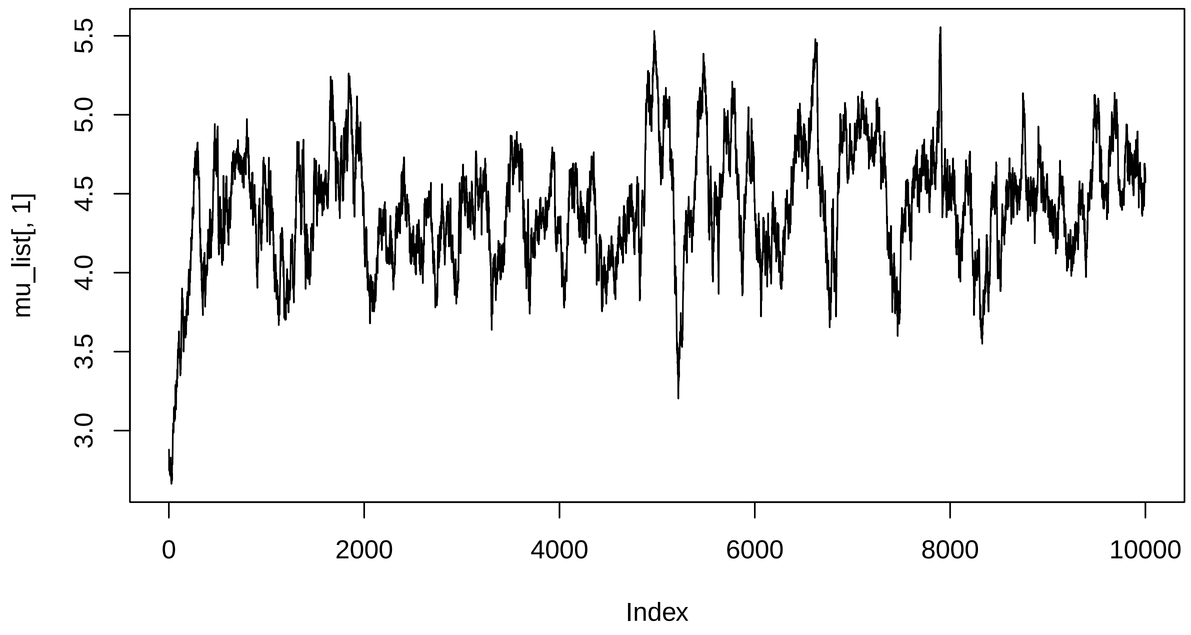
```r
# constants
nu <- 5 # suggested
V <- diag(5) # can try other. see requirement
S <- diag(5) # can try other. see requirement

set.seed(1)
res <- MCMC(B0, mu0, Sigma0, gamma0, sigma0, a = 0.1, iter = 10000) # try larger values

# B. beta_i_j: i: ith hurricane, j: 1-5
B_list <- t(mapply(function(x) x$B, res))
B_names <- apply(expand.grid(0:4, 1:n), 1,
                 function(x) paste("beta", x[2], x[1], sep = "_"))
colnames(B_list) <- B_names
# mu
mu_list <- t(mapply(function(x) x$mu, res))
colnames(mu_list) <- colnames(Z[[1]])
# Sigma (symmetric)
Sigma_list <- t(mapply(function(x) x$Sigma, res))
Sigma_names <- apply(expand.grid(1:5, 1:5), 1,
                     function(x) paste("Sigma_", x[2], x[1], sep = ""))
colnames(Sigma_list) <- Sigma_names
# gamma (month ref: January, nature ref: DS)
gamma_list <- t(mapply(function(x) x$gamma, res))
# sigma
sigma_list <- mapply(function(x) x$sigma, res)


B_mean = bind_rows(colSums(B_list[5001:10000,])/5000, colSums(B_list[6001:10000,])/4000,
                   colSums(B_list[7001:10000,])/3000, colSums(B_list[8001:10000,])/2000,
                   colSums(B_list[9001:10000,])/1000)
write.csv(B_mean, file = "./data/B_list_lastmean.csv", row.names = FALSE) # last 5000 obs
write.csv(mu_list, file = "./data/mu_list.csv", row.names = FALSE)
write.csv(Sigma_list, file = "./data/Sigma_matrix_list.csv", row.names = FALSE)
write.csv(gamma_list, file = "./data/gamma_list.csv", row.names = FALSE)
write.csv(sigma_list, file = "./data/sigma_list.csv", row.names = FALSE)
```

Resulting plots (not complete)

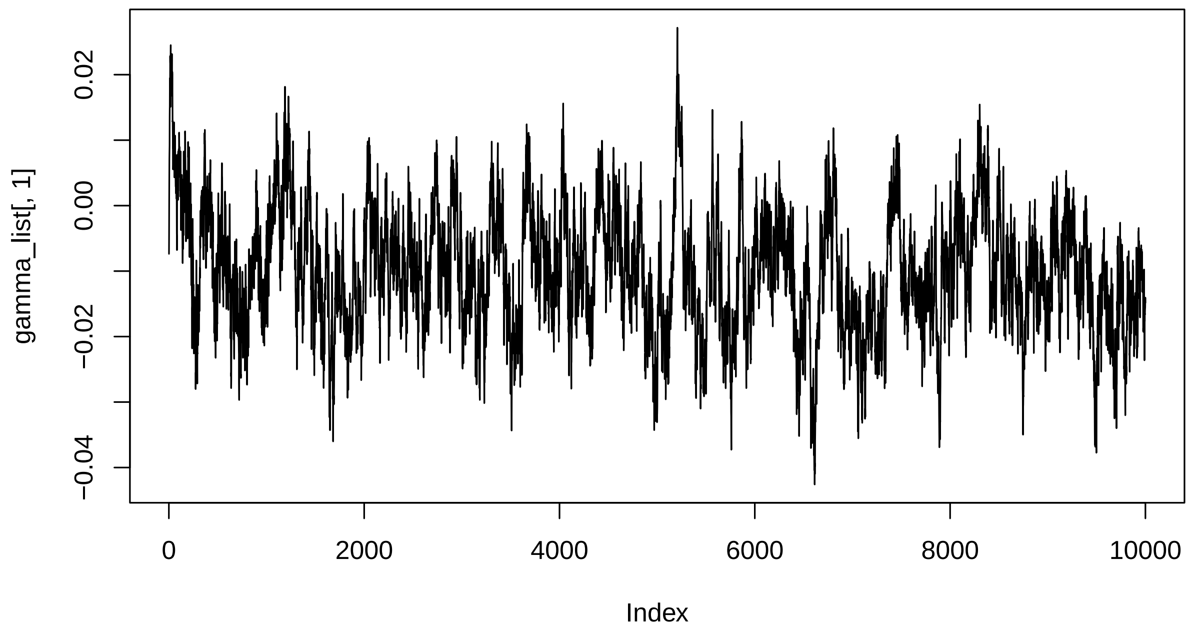
Please make time series plots for all parameters to see if and from which index their samples converge. (may not need for **B**, since it has $5n$ parameters (5 for each hurricane))

Please also make autocorrelation plots.

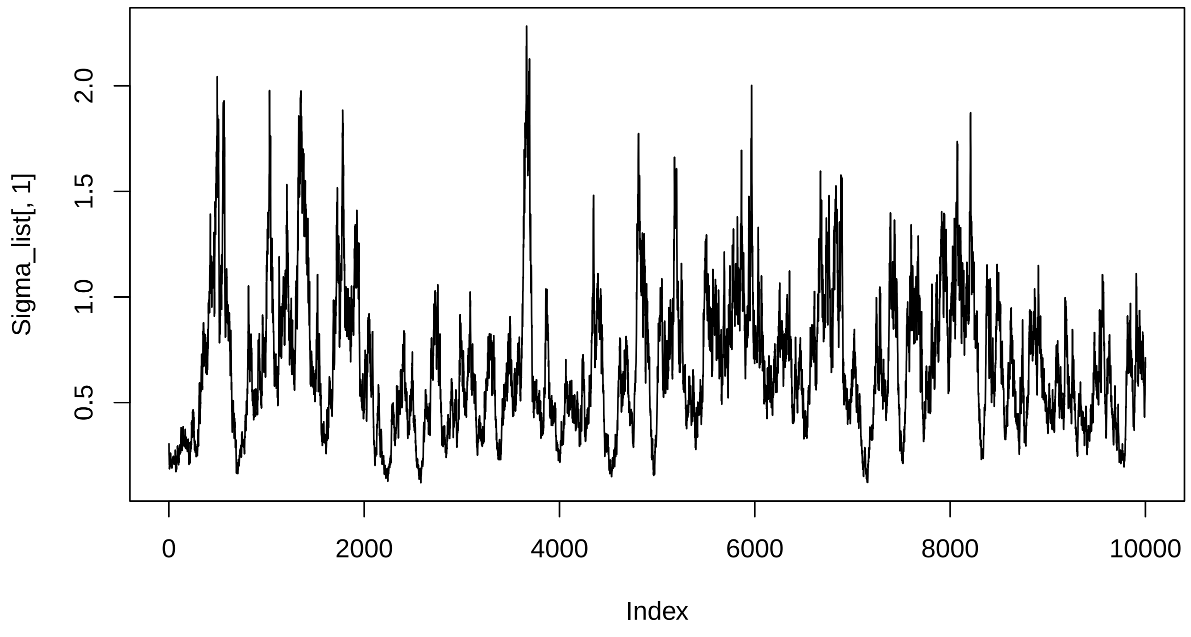
```
beta_list <- read.csv("./data/B_list_lastmean.csv")
mu_list <- read.csv("./data/mu_list.csv")
Sigma_list <- read.csv("./data/Sigma_matrix_list.csv")
gamma_list <- read.csv("./data/gamma_list.csv")
sigma_list <- read.csv("./data/sigma_list.csv")$x

plot(mu_list[,1], type = "l") # plot mu1 (intercept)
```



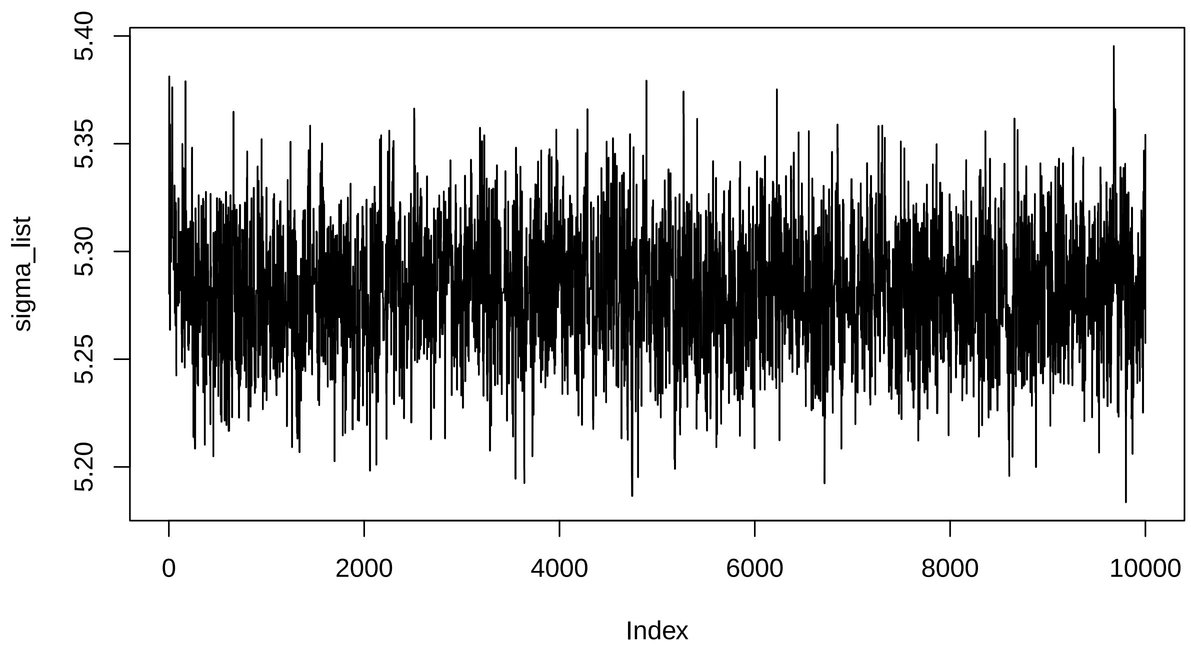
```
plot(gamma_list[,1], type = "l") # e.g. plot gamma1 (year)
```



```r
plot(Sigma_list[,1], type = "l") # plot Sigma11
```



```
plot(sigma_list, type = "l") # plot sigma
```



Burn in (based on resulting plots)

`xxx_sample` will be used for doing inference (estimates, CI, hypothesis test) and making predictions.

Please select “burn in” numbers based on all time series plots and autocorrelation plots.

```
burn <- 8000 # burn in the MC chains. change this based on the resulting plots
index <- (burn + 1):length(res) # index of useful samples (used for estimates & CIs)
B_sample <- B_list[index - 5000,]
mu_sample <- mu_list[index,]
Sigma_sample <- Sigma_list[index,] # may be useless
gamma_sample <- gamma_list[index,]
sigma_sample <- sigma_list[index] # may be useless
```

Acceptance rate for σ samples

30% - 60% is good. if not, please change `a` in random walk. If really need to change `a`, you need to rerun the MCMC function and update the results saved in the files.

```
length(unique(sigma_list[index]))/length(sigma_list[index])
```

```
## [1] 0.4005
```