Project 4: Bayesian modeling of hurrican trajectories

Sitong Cui, Zongchao Liu, Yuanzhi Yu, Mengyu Zhang

Introduction

The 1989 Atlantic hurricane season officially began on June 1, 1989, and lasted until November 30, 1989. However storms can form outside these dates. For example, Tropical Storm Karen lasted until December 4. This season had average activity with 15 depressions, 3 became a tropical storm, 5 became a hurricane, and 2 became a major hurricane. This was a damaging season because this season featured the powerful Hurricane Hugo. Overall, the storms of the season collectively caused 136 fatalities and at least $10.2 billion in damage. Therefore, efforts to obtain deeper understanding of their physical mechanisms such as Wind.kt (Maximum wind speed in Knot at each check point) for the development and tracking is very important for us.

Objective

Randomly select 80% hurricanes and design a MCMC algorithm to estimate the posterior distributions of the model parameters. Estimate the model parameters using posteri means and construct their 95% credibility intervals. Apply the developed model to track the remaining 20% hurricanes, and evaluate how well your model could predict and track these hurricanes.

Data

hurrican356.csv collected the track data of 356 hurricanes in the North Atlantic area since 1989. For all the storms, their location (longitude & latitude) and maximum wind speed were recorded every 6 hours. The data includes the following variables

1. **ID**: ID of the hurricans
2. **Season**: In which the hurricane occurred
3. **Month**: In which the hurricane occurred
4. **Nature**: Nature of the hurricane

- ET: Extra Tropical
- DS: Disturbance
- NR: Not Rated
- SS: Sub Tropical
- TS: Tropical Storm

5. **time**: dates and time of the record
6. **Latitude** and **Longitude**: The location of a hurricane check point
7. **Wind.kt** Maximum wind speed (in Knot) at each check point

To make the model more interpretable, we delete the storms with the “Nature” being “Not Rated”, and then randomly split the data into training data and test data.

Method

Let t be time (in hours) since a hurricane began, and For each hurrican i, we denote $\{Y_{i,1}(t), Y_{i,2}(t), Y_{i,3}(t)\}, j = 1, 2, 3$ be the latitude, longitude, and wind speed at time t. The following Bayesian model was suggested:

$$Y_i(t+6) = \mu_i(t) + \rho Y_i(t) + \epsilon_i(t)$$

And then we have:

$$\mu_i(t) = \beta_0 + x_{i,1}(t)\beta_1 + x_{i,2}\beta_2 +]x_{i,3}\beta_3 + \sum_{k=1}^{3} \beta_{3+k}\Delta_{i,k}(t-6)$$

$$\Delta_{i,k}(t-6) = Y_{i,k}(t) - Y_{i,k}(t-6), k = 1, 2, 3$$

For $\pi(\boldsymbol{\beta})$ ~ $MVN(\mathbf{0},\ diag(1, p))$, $\pi(\rho)$ follows a truncated normal $N_{[0,1]}(0.5, 1/5)$ $\pi((\sigma^2)^{-1})$ follows a inverse-gamma $(0.001, 0.001)$

What we can conclude from these information:

Yi is a linear combination of μ_i and $\rho Y_i(t)$ and $\epsilon_i(t)$. While μ_i and $\rho Y_i(t)$ is a constant. So $Y_i(t+6)$ follow normal distribution with $\mu_{new}(t)$ and variance σ^2:

$$Y_i(t+6) \sim Normal(\mu_{new}(t), \sigma^2)$$

$$\mu_{new}(t) = \mu_i(t) + \rho Y_i(t)$$

To exclude the time series influence on Y_i we choose to use

$$\epsilon_i = Y_i(t+6) - \mu_{new} \sim Normal(0, \sigma^2)$$

We set i is the ith hurricane, n is the total number of hurricane. The k denote the kth speed among this hurricane, m is the total measured speeds among this hurricane.

The posteria distribution

$$f(\epsilon_i|\boldsymbol{\beta}, \rho, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(Y_i(t+6) - \mu_{new} - 0)^2}{2\sigma^2})$$

Poesterian distribution:

$$\pi(\boldsymbol{\beta}, \rho, \sigma^2|\epsilon_i) \propto \prod_{i=1}^{n}\prod_{k=1}^{m} f(\epsilon_i|\boldsymbol{\beta}, \rho, \sigma^2) * \pi(\boldsymbol{\beta}) * \pi(\rho) * \pi((\sigma^2)^{-1})$$

Take the log form:

$$\pi'(\boldsymbol{\beta}, \rho, \sigma^2|\epsilon_i) \propto \sum_{i=1}^{n}\sum_{k=1}^{m} log(f(\epsilon_i|\boldsymbol{\beta}, \rho, \sigma^2)) + log(\pi(\boldsymbol{\beta})) + log(\pi(\rho)) + log(\pi((\sigma^2)^{-1}))$$

To find the poeteria distribution, we use the random walk in Metropolis-Hasting algorithm: The probability of accepting :

$$\alpha(\lambda|\boldsymbol{\theta}) = min(1, \frac{\pi(\boldsymbol{\lambda})q(\boldsymbol{\theta}|\boldsymbol{\lambda})}{\pi(\boldsymbol{\theta})q(\boldsymbol{\lambda}\boldsymbol{\theta})})$$

In this situation, we have:

$$q(\theta|\lambda) = q(\lambda|\theta)$$

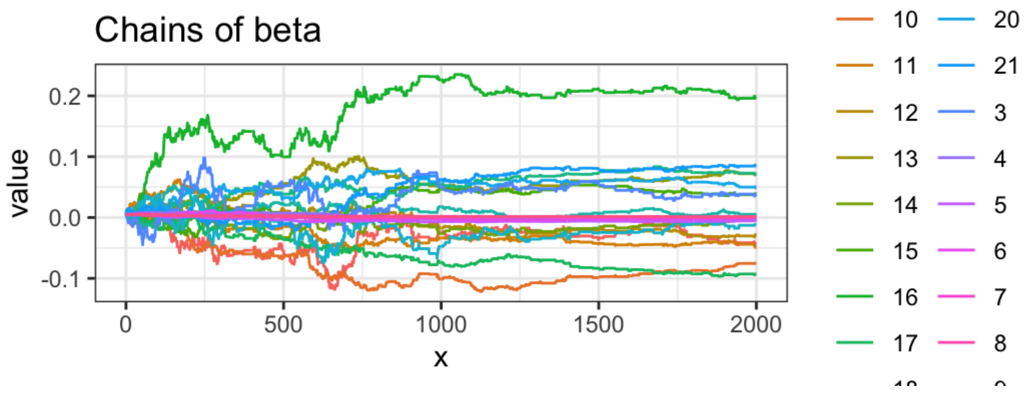
So,

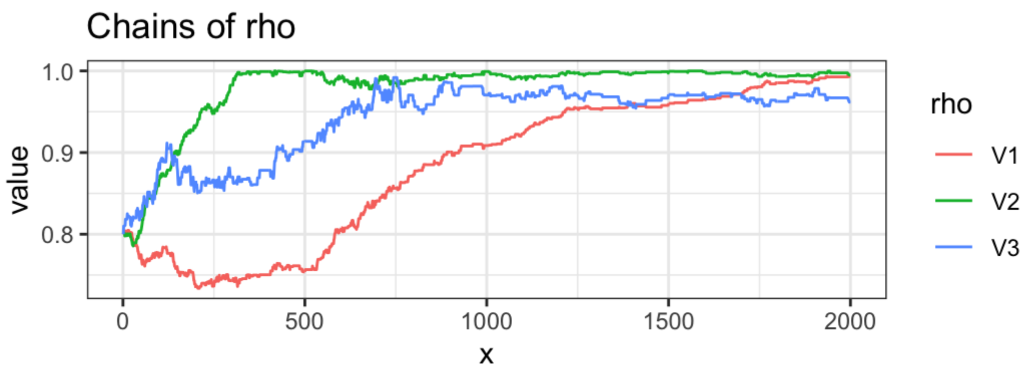
$$\alpha(\lambda|\boldsymbol{\theta}) = min(1, \frac{\pi(\boldsymbol{\lambda})}{\pi(\boldsymbol{\theta})}) = min(1, \pi'(\boldsymbol{\lambda}) - \pi'(\boldsymbol{\theta}))$$

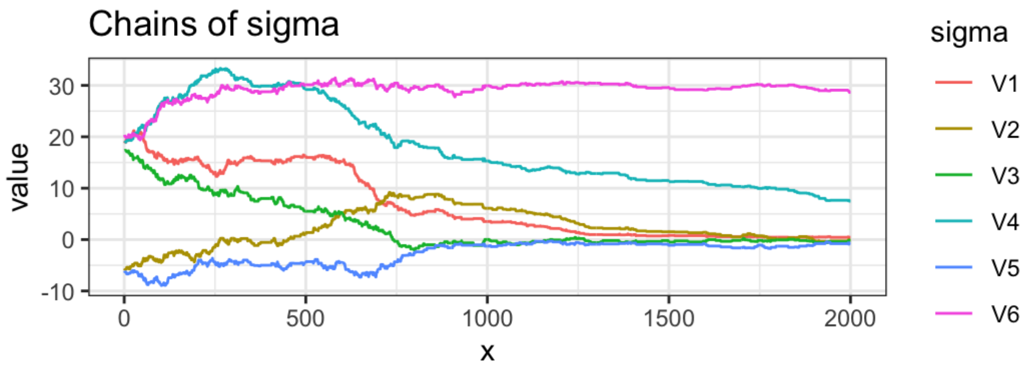
Finally ,we can compare the $\alpha(\lambda|\boldsymbol{\theta})$ with the random drawed uniform(0,1) number,if $\alpha(\lambda|\boldsymbol{\theta})$ is larger, accept the λ, otherwise still accept $ $.

Result

The chain plots of parameters are shown below. Since the accept rate decrease after 1000 iterations, to make transition more efficient, the step size for random walk during first 1000 iterations is set to be different from the second half iterations. Accept rate is $433/2000 = 0.2165$.



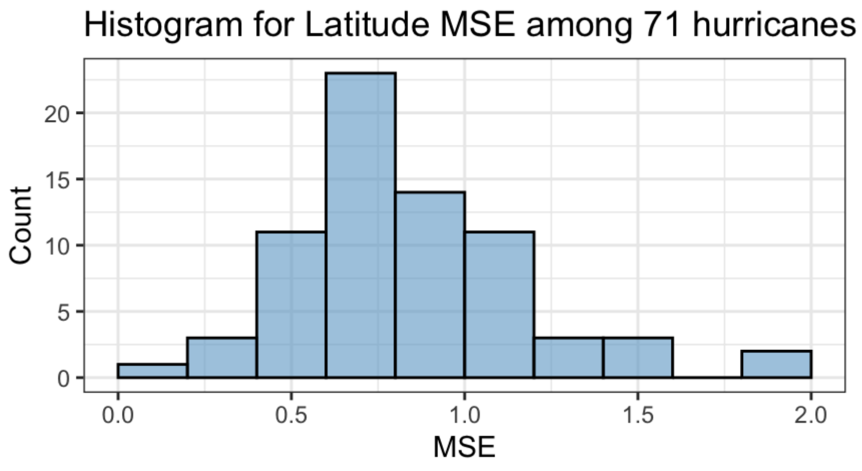


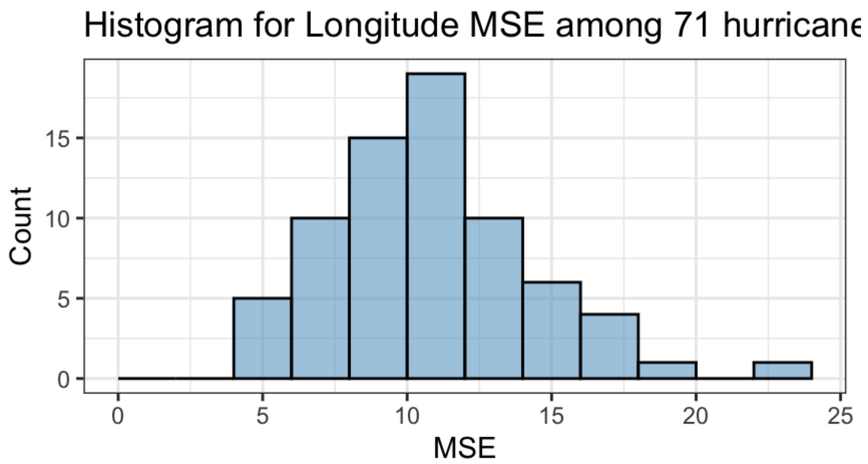


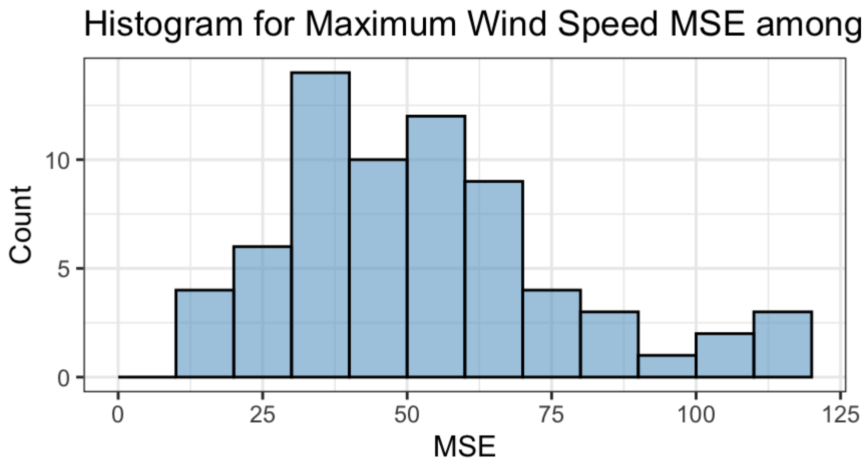
Almost every parameters converge after 1500 iterations, so we average last 500 iterations to get estimates of parameters. The final parameters estimate are shown below.

```
##         Latitude Longitude Speed
## hat_rho    0.978     0.997 0.968
## 2.5%       0.960     0.993 0.958
## 97.5%      0.993     1.000 0.974
##          Latitude_intercpt Longitude_intercpt Speed_intercpt
## hat_beta            -0.027             -0.017          0.039
## 2.5%                -0.042             -0.025          0.030
## 97.5%               -0.009             -0.006          0.053
##          Latitude_beta1 Longitude_beta1 Speed_beta1 Latitude_beta2
## hat_beta         -0.002          -0.006      -0.002          0.001
## 2.5%             -0.002          -0.007      -0.003          0.001
## 97.5%            -0.001          -0.006       0.000          0.001
##          Longitude_beta2 Speed_beta2 Latitude_beta3 Longitude_beta3
## hat_beta           0.001       0.001         -0.088          -0.044
## 2.5%               0.001       0.001         -0.097          -0.047
## 97.5%              0.001       0.001         -0.075          -0.040
##          Speed_beta3 Latitude_beta4 Longitude_beta4 Speed_beta4
## hat_beta      -0.025          0.066          -0.011       0.043
## 2.5%          -0.032          0.061          -0.017       0.036
## 97.5%         -0.018          0.076          -0.001       0.052
##          Latitude_beta5 Longitude_beta5 Speed_beta5 Latitude_beta6
## hat_beta          0.206          -0.090       0.076          0.008
## 2.5%              0.196          -0.096       0.073          0.003
## 97.5%             0.214          -0.083       0.083          0.014
##          Longitude_beta6 Speed_beta6
## hat_beta           0.059       0.081
## 2.5%               0.050       0.076
## 97.5%              0.065       0.086
```

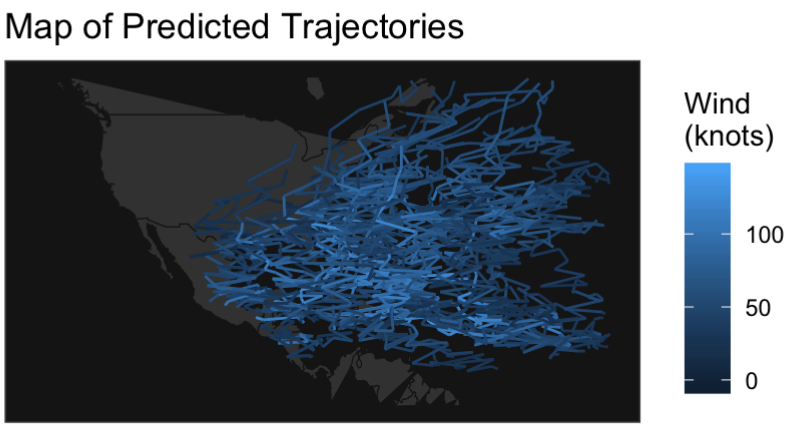
For prediction, we first paste the first two observations from real data and predict the following time points by using the model with estimated parameters. To evaluate the model's performence of prediction, for each storm in test dataset, we calculate the mse for latitudes, longitude and maximum wind speed respectively. The distribution of mse among 71 storms is shown as the following plots. Compared with all the pre-set parameters we have tried according to the distribution of MSE, such as starting values and step size in MH algorithm, this model have the best prediciton performance. However, the prediction is not very well especially for speed prediction.

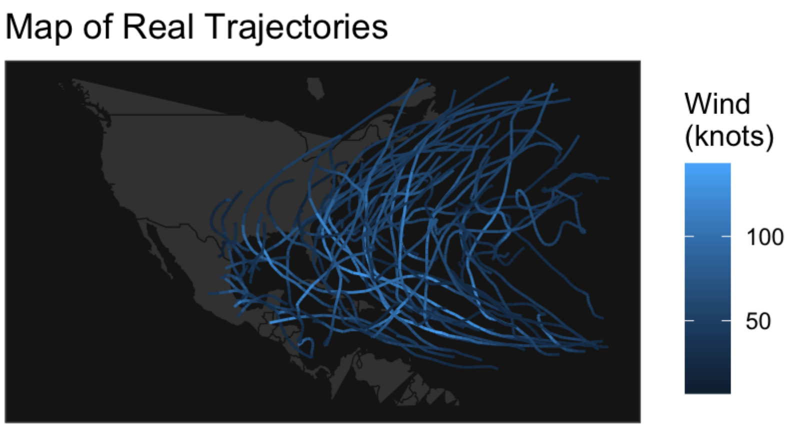




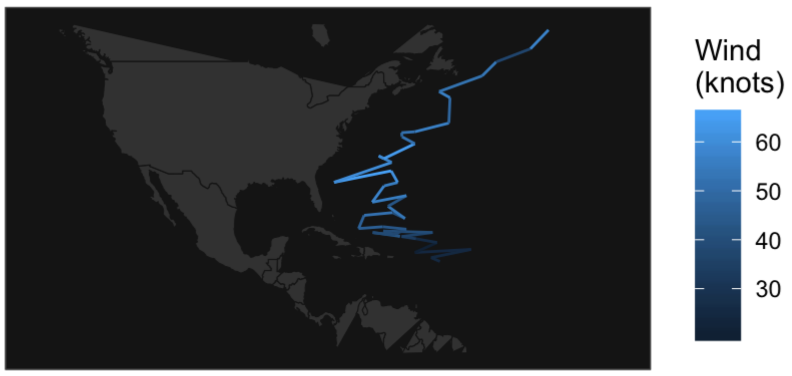


Therefore, for all storms in test dataset, we can draw map to show how well the prediction is. For storm CHANTAL.1995, the trajectory is not as smooth as the real trajectory, since the prediction errors are large.

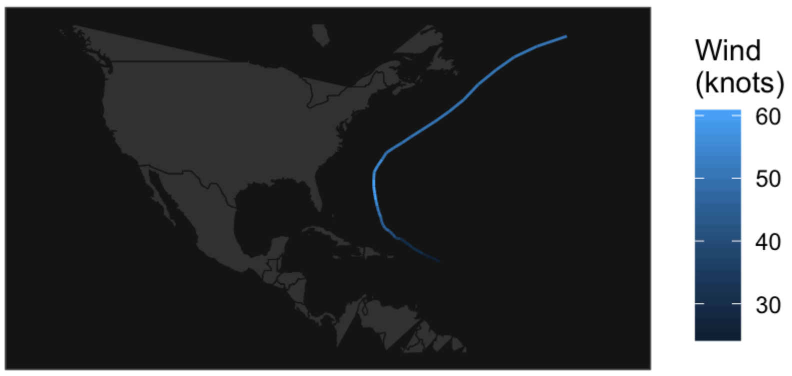




Map of Predicted Trajectory (CHANTAL.1995)



Map of Real Trajectory (CHANTAL.1995)



Discussion and Conclusions

In this project, we constructed a complex Bayesian model to predict the trajectories of the hurricanes. The model predicted each hurricane's location and wind-speed. By implementing the regular Metropolis-Hastings algorithm for 2000 iterations, we finally obtained the converged estimates of the 30 parameters in the model. The result showed that our final model gave us a set of acceptable predicted values.

By plotting the predictive values and comparing them to the true values, we see that the trajectories of the hurricanes that we predict are not that smooth like the true trajectories. This is because of the predictive errors of the latitude and longitude values in the data. If we choose the simplified model, the resulting predicted trajectories would seem to be better because the latitude and longitude values are always correct under this scenario. Hence, The plot would only show the predictive errors of the windspeed and the trajectories would be smooth. It is hard to say the full model performs well if we consider the errors we get. The results are acceptable since the predicted hurricanes are generally in the correct locations and the wind-speeds are similar to the original true values.

For our current model, we run only 2000 iterations to obtain the model because constructing the full model takes a long time. We did not have enough time for the full model since we might need some extra time to adjust the initial values. Therefore, we chose 2000 iterations to check the chain and see if the parameters were converged

To improve the model, it is recommended to try some other methods like component-wise MH algorithm, Gibb sampling to update the parameters. It is important that in the process of updating parameters, we should periodically check the values of the parameters to make sure everything is on the right pathway. It is highly recommended to calculate the accept rate every 100 iterations as a way to efficiently adjust the modeling process. Specifically, we found that in the original dataset, there were 85 winds that were not rated. These data might not have much influence on the model performance due to the large sample size we had, but might bring some problems to model interpretation. Thus, we finally deleted these 85 observations to ensure a better interpretation of the results.

Appendix

```r
# data cleaning
data = read.csv("./hurrican356.csv")
shift <- function(x, n=1){
  c(x[-(seq(n))], rep(NA, n))
}
data1 = read.csv("hurrican356.csv") %>%
  janitor::clean_names() %>%
  filter(nature != "NR") %>%
  mutate(year = season,
         date_hour = time) %>%
  separate(date_hour, into = c("date", "hour"), sep = " ") %>%
  filter(hour == "00:00:00)" | hour == "06:00:00)" | hour == "12:00:00)" | hour ==
"18:00:00)") %>%
  mutate(hour = str_replace(hour, ":00:00\\)", ""),
         hour = as.numeric(hour),
         date = str_replace(date, "\\(", ""),
         date = yday(date),
         nature = as.numeric(as.factor(nature))) %>%
  group_by(id) %>%
  mutate(delta1 = c(NA, diff(latitude)),
         delta2 = c(NA, diff(longitude)),
         delta3 = c(NA, diff(wind_kt)),
         latitude_d = shift(latitude),
         longitude_d = shift(longitude),
         windkt_d = shift(wind_kt)) %>%
  ungroup() %>%
  na.omit() %>%
  select(id, latitude, longitude, wind_kt, latitude_d, longitude_d, windkt_d, date,
 year, nature, delta1, delta2, delta3)

#head(data1)
#summary(data1)
```

```r
#split the data into train and test
set.seed(123)
id = unique(data1$id)
num_id = length(id)
train_id = sample(id, 0.8*num_id)

train_data = data1[which(data1$id %in% train_id),] %>%
  select(-id)
```

```
# Starting points
set.seed(111)
rho = rep(0.8, 3)
sigma = bayesm::rwishart(3,diag(0.1,3))$IW
sigma = c(sigma[1,], sigma[2,c(2,3)], sigma[3,3])
beta = rep(0.005,21)

# Density function.
# for each yi
logdy = function(obs, beta, rho, sigma){
  x = c(1,obs[7:12])
  y = obs[1:3]
  mu = beta %*% x + rho*obs[1:3]
  dy = dmvnorm(obs[4:6], mean = mu, sigma = sigma)
  return(log(dy))
}

#traintest = train_data[c(1:100),]
#betatest = rep(0.008,21)
#apply(traintest, 1, logdy, beta.=betatest)

logdensity = function(data=train_data, beta.=beta, rho.=rho, sigma.=sigma){
  beta_m = matrix(beta.,3)
  sigma_m = matrix(c(sigma.[c(1:3)], sigma.[2], sigma.[c(4,5)], sigma.[c(3,5)], sigm
a.[6]), 3)
  logdy = apply(data, 1, logdy, beta=beta_m, rho=rho., sigma=sigma_m)
  logdens = sum(logdy) + log(dmvnorm(beta., rep(0,21), diag(1,21))) + log(dtruncnorm
(rho.[1], a=0, b=1, mean = 0.5, sd = 0.2)) + log(dtruncnorm(rho.[2], a=0, b=1, mean
 = 0.5, sd = 0.2)) + log(dtruncnorm(rho.[3], a=0, b=1, mean = 0.5, sd = 0.2)) + log
(MCMCpack::diwish(sigma_m, 3, diag(0.1,3)))
  return(logdens)
}
```

```
# MH
regularMHstep = function(startpars, niter = 1000, rhoa, betaa, sigmaa){
  beta_m = matrix(NA, niter, 21)
  rho_m = matrix(NA, niter, 3)
  sigma_m = matrix(NA, niter, 6)
  beta_m[1,] = startpars$beta
  rho_m[1,] = startpars$rho
  sigma_m[1,] = startpars$sigma
  for (i in 2:1000) {   # correlated issue
    print(str_c("####################  ", i, "/",niter, "  ####################"))
    temp_beta = beta_m[i-1,] + runif(21,-1,1)*beta_a
    temp_rho = rho_m[i-1,] + runif(3,-1,1)*rho_a
    temp_sigma = sigma_m[i-1,] + runif(6,-1,1)*sigma_a
    temp_sigma_m = matrix(c(temp_sigma[c(1:3)], temp_sigma[2], temp_sigma[c(4,5)], t
emp_sigma[c(3,5)], temp_sigma[6]), 3)
    if (sum(ifelse(temp_rho<1, 0, 1))==0 & is.positive.definite(temp_sigma_m)) {
      if (log(runif(1)) < logdensity(beta.=temp_beta, rho.=temp_rho, sigma.=temp_sig
ma) - logdensity(beta.=beta_m[i-1,], rho.=rho_m[i-1,], sigma.=sigma_m[i-1,])){
        beta_m[i,] = temp_beta
        rho_m[i,] = temp_rho
        sigma_m[i,] = temp_sigma
        }
      else{
        beta_m[i,] = beta_m[i-1,]
        rho_m[i,] = rho_m[i-1,]
        sigma_m[i,] = sigma_m[i-1,]
      }}
    else{
      beta_m[i,] = beta_m[i-1,]
      rho_m[i,] = rho_m[i-1,]
      sigma_m[i,] = sigma_m[i-1,]
      }
  }
  for (i in 1001:niter) {   # correlated issue
    print(str_c("####################  ", i, "/",niter, "  ####################"))
    temp_beta = beta_m[i-1,] + runif(21,-1,1)*beta_a/2
    temp_rho = rho_m[i-1,] + runif(3,-1,1)*rho_a/2
    temp_sigma = sigma_m[i-1,] + runif(6,-1,1)*sigma_a/2
    temp_sigma_m = matrix(c(temp_sigma[c(1:3)], temp_sigma[2], temp_sigma[c(4,5)], t
emp_sigma[c(3,5)], temp_sigma[6]), 3)
    if (sum(ifelse(temp_rho<1, 0, 1))==0 & is.positive.definite(temp_sigma_m)) {
      if (log(runif(1)) < logdensity(beta.=temp_beta, rho.=temp_rho, sigma.=temp_sig
ma) - logdensity(beta.=beta_m[i-1,], rho.=rho_m[i-1,], sigma.=sigma_m[i-1,])){
        beta_m[i,] = temp_beta
        rho_m[i,] = temp_rho
        sigma_m[i,] = temp_sigma
        }
      else{
        beta_m[i,] = beta_m[i-1,]
        rho_m[i,] = rho_m[i-1,]
        sigma_m[i,] = sigma_m[i-1,]
      }}
    else{
      beta_m[i,] = beta_m[i-1,]
      rho_m[i,] = rho_m[i-1,]
      sigma_m[i,] = sigma_m[i-1,]
      }
```

```r
  }
  return(list(est_beta = beta_m, est_rho = rho_m, est_sigma = sigma_m))
}
```

```r
# starting points
startpars = list(rho = rho, beta = beta, sigma = sigma)
rho_a = c(0.005,0.005,0.01)
sigma_a = rep(0.5, 6)
beta_a = c(rep(0.01, 3), rep(0.0005, 2), 0.001, rep(0.0001, 3), rep(0.005,6), 0.01,
 rep(0.005, 5))

set.seed(123)
MHresults = regularMHstep(startpars, niter = 2000, rhoa = rho_a, betaa = beta_a, sig
maa = sigma_a)

#save(MHresults, file="result_2000_2.RData")
```

```r
# accept rate
uni_beta = rep(NA,21)
for (i in 1:21){
  uni_beta[i] = length(unique(MHresults$est_beta[1:2000,i]))
}
uni_beta

uni_sigma = rep(NA,6)
for (i in 1:6) {
  uni_sigma[i] = length(unique(MHresults$est_sigma[1:2000,i]))
}
uni_sigma

uni_rho = rep(NA,3)
for (i in 1:3) {
  uni_rho[i] = length(unique(MHresults$est_rho[1:2000,i]))
}
uni_rho
```

```r
# print chain plots
niter = 2000
beta_results = as.data.frame(MHresults$est_beta) %>%
  mutate(x = 1:niter) %>%
  gather(key = beta, value = value, V1:V21) %>%
  mutate(beta = str_replace(beta, "V", ""))
beta_plot = ggplot(beta_results, aes(x = x, y = value, color = beta)) +
  geom_line()
beta_plot

rho_results = as.data.frame(MHresults$est_rho) %>%
  mutate(x = 1:niter) %>%
  gather(key = rho, value = value, V1:V3)
rho_plot = ggplot(rho_results, aes(x = x, y = value, color = rho)) +
  geom_line()
rho_plot

sigma_results = as.data.frame(MHresults$est_sigma) %>%
  mutate(x = 1:niter) %>%
  gather(key = sigma, value = value, V1:V6)
sigma_plot = ggplot(sigma_results, aes(x = x, y = value, color = sigma)) +
  geom_line()
sigma_plot
```

```r
# estimates
hat_sigmas = MHresults$est_sigma
hat_rhos = MHresults$est_rho
hat_betas = MHresults$est_beta

hatsigma = as.matrix(hat_sigmas[1500:nrow(hat_sigmas),])
hatrho = as.matrix(hat_rhos[1500:nrow(hat_rhos),])
hatbeta = as.matrix(hat_betas[1500:nrow(hat_betas),])

hat_sigma = apply(hatsigma,2, mean)
hat_rho = apply(hatrho,2,mean)
hat_beta = apply(hatbeta,2,mean)

hat_sigmam = matrix(c(hat_sigma[c(1:3)], hat_sigma[2], hat_sigma[c(4,5)], hat_sigma
[c(3,5)], hat_sigma[6]), 3)
hat_betam = matrix(hat_beta,3)

hat_sigmam
hat_betam
hat_rho

ci_sigma = apply(hatsigma, 2, quantile, probs = c(0.025, 0.975))
ci_rho = apply(hatrho, 2, quantile, probs = c(0.025, 0.975))
ci_beta = apply(hatbeta, 2, quantile, probs = c(0.025, 0.975))

rbind(hat_sigma, ci_sigma)

rbind(hat_rho, ci_rho)

rbind(hat_beta, ci_beta)
```

```r
# prediction
test_id = setdiff(id,train_id)
test1 = data1[which(data1$id %in% test_id),]
test_ls = test1 %>% group_split(id)

# for each yi
predy = function(obs){
  y <- NULL
  for (i in 1:2){
    temp = obs[i,2:4]
    y = rbind(y, temp)
  }

  for (i in 3:nrow(obs)){
    d = y[i-1,]-y[i-2,]
    #x = c(1, as.matrix(obs[i,8:10]),as.matrix(d))
    x = c(1, as.matrix(obs[i,8:13]))
    #mu = t(hat_betam %*% x) + hat_rho * y[i-1,]
    mu = t(hat_betam %*% x) + hat_rho * as.matrix(obs[i-1,2:4])
    epsilon = rmvnorm(1, mean = rep(0,3), sigma = hat_sigmam)
    y[i,] = mu + epsilon
  }
  err_latitude = mean(as.matrix((y[,1] - obs[,2])^2))
  err_longitude = mean(as.matrix((y[,2] - obs[,3])^2))
  err_wk = mean(as.matrix((y[,3] - obs[,4])^2))

  return(list(predy = y, err = cbind(err_latitude, err_longitude, err_wk)))
}

pred_results = map(test_ls, predy)
```

```r
y <- NULL
for (i in 1:length(pred_results)) {
  y = rbind(y, pred_results[[i]]$predy)
}

test_ord = test1 %>% arrange(id)

dt1 <- cbind(test_ord,y) %>%
  janitor::clean_names() %>%
  select(-latitude_d, -longitude_d, -windkt_d, -date,-year,-nature,-delta1,-delta2,-
delta3)



##### mse
mse <- NULL
for (i in 1:length(pred_results)) {
  mse = rbind(mse, pred_results[[i]]$err)
}

test_ord = test1 %>% arrange(id)
dt2 <- data.frame(mse) %>%
  mutate(id = as.character(unique(test_ord$id))) %>%
  select(id, everything())
```

```r
map = ggplot(dt1, aes(x = longitude_2, y = latitude_2, group = id)) +
  geom_polygon(data = map_data("world"),
               aes(x = long, y = lat, group = group),
               fill = "gray25", colour = "gray10", size = 0.2) +
  geom_path(data = dt1, aes(group = id, colour = wind_kt_2), size = 0.5) +
  xlim(-138, -20) + ylim(3, 55) +
  labs(x = "", y = "", colour = "Wind \n(knots)", title="Map of Prediction") +
  theme(panel.background = element_rect(fill = "gray10", colour = "gray30"),
        axis.text.x = element_blank(), axis.text.y = element_blank(),
        axis.ticks = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
map

map1 = ggplot(dt1, aes(x = longitude, y = latitude, group = id)) +
  geom_polygon(data = map_data("world"),
               aes(x = long, y = lat, group = group),
               fill = "gray25", colour = "gray10", size = 0.2) +
  geom_path(data = dt1, aes(group = id, colour = wind_kt), size = 0.5) +
  xlim(-138, -20) + ylim(3, 55) +
  labs(x = "", y = "", colour = "Wind \n(knots)",title="Map of Real") +
  theme(panel.background = element_rect(fill = "gray10", colour = "gray30"),
        axis.text.x = element_blank(), axis.text.y = element_blank(),
        axis.ticks = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
map1
```

```
dt11 = dt1 %>%
  filter(id == unique(dt1$id)[10])

# CHANTAL.1995
CHANTAL.1995.pre = dt11 %>%
  ggplot(aes(x = longitude_2, y = latitude_2, group = id)) +
  geom_polygon(data = map_data("world"),
               aes(x = long, y = lat, group = group),
               fill = "gray25", colour = "gray10", size = 0.2) +
  geom_path(aes(group = id, colour = wind_kt_2), size = 0.5) +
  xlim(-138, -20) + ylim(3, 55) +
  labs(x = "", y = "", colour = "Wind \n(knots)", title="Map of Prediction (CHANTAL.
1995)") +
  theme(panel.background = element_rect(fill = "gray10", colour = "gray30"),
        axis.text.x = element_blank(), axis.text.y = element_blank(),
        axis.ticks = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
CHANTAL.1995.pre

CHANTAL.1995.real = dt11 %>%
  ggplot(aes(x = longitude, y = latitude, group = id)) +
  geom_polygon(data = map_data("world"),
               aes(x = long, y = lat, group = group),
               fill = "gray25", colour = "gray10", size = 0.2) +
  geom_path(aes(group = id, colour = wind_kt), size = 0.5) +
  xlim(-138, -20) + ylim(3, 55) +
  labs(x = "", y = "", colour = "Wind \n(knots)", title="Map of Real (CHANTAL.1995)"
) +
  theme(panel.background = element_rect(fill = "gray10", colour = "gray30"),
        axis.text.x = element_blank(), axis.text.y = element_blank(),
        axis.ticks = element_blank(), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())

CHANTAL.1995.real
```

```
ggsave("beta_plot1000.jpeg", beta_plot, width = 8, height = 4)
ggsave("rho_plot1000.jpeg", rho_plot, width = 8, height = 4)
ggsave("sigma_plot1000.jpeg", sigma_plot, width = 8, height = 4)

ggsave("beta_plot2000.jpeg", beta_plot, width = 8, height = 4)
ggsave("rho_plot2000.jpeg", rho_plot, width = 8, height = 4)
ggsave("sigma_plot2000.jpeg", sigma_plot, width = 8, height = 4)


save(dt1, file="pred_res.RData")
save(dt2, file="err.RData")

ggsave("map_pred_2000.jpeg", map, width = 8, height = 4)
ggsave("map_real_2000.jpeg", map1, width = 8, height = 4)
ggsave("CHANTAL_1995_pre_2000.jpeg", CHANTAL.1995.pre, width = 8, height = 4)
ggsave("CHANTAL_1995_real_2000.jpeg", CHANTAL.1995.real, width = 8, height = 4)

#save(MHresults, file="result_2000.RData")
```

```
err_latitude_hist = dt2 %>% ggplot(aes(err_latitude)) +
  geom_histogram(breaks = seq(0,3, by = 0.2),
                 col="black",
                 fill="skyblue3",
                 alpha = .6) +
  labs(title="Histogram for Latitude MSE among 71 hurricanes", x="MSE", y="Count")+
  theme_bw()
err_latitude_hist

err_longitude_hist = dt2 %>% ggplot(aes(err_longitude)) +
  geom_histogram(breaks = seq(0,30, by = 2),
                 col="black",
                 fill="skyblue3",
                 alpha = .6) +
  labs(title="Histogram for Longitude MSE among 71 hurricanes", x="MSE", y="Count")+
  theme_bw()
err_longitude_hist

err_wk_hist = dt2 %>% ggplot(aes(err_wk)) +
  geom_histogram(breaks = seq(0,150, by = 10),
                 col="black",
                 fill="skyblue3",
                 alpha = .6) +
  labs(title="Histogram for Maximum Wind Speed MSE among 71 hurricanes", x="MSE", y=
"Count")+
  theme_bw()
err_wk_hist


#ggsave("err_latitude_hist.jpeg", err_latitude_hist, width = 6, height = 4)

#ggsave("err_longitude_hist.jpeg", err_longitude_hist, width = 6, height = 4)

#ggsave("err_wk_hist.jpeg", err_wk_hist, width = 6, height = 4)
```