Hurricane Predictions Using a MCMC Algorithm

Margaret Gacheru, Melanie Mayer, Kee-Young Shin, Adina Zhang

May 10, 2020

Introduction

Hurricanes are type of storms with high speed winds that form over tropical or subtropical waters. Typically, hurricanes bring about strong winds, storm surges, and heavy rainfall that can lead to flooding, tornadoes, etc. The Saffir-Simpson Hurricane scale is used to rate hurricanes by their wind speed – the higher the category, the greater the possibility of landfall damage. It is worthwhile to understand aspects of hurricanes and make accurate predictions in order to provide appropriate information and resources to the public, make informed decisions, and potentially save lives.

For this project, we have information about 356 hurricanes in the North Atlantic area since 1989, including the storm's location and maximum windspeed for every 6 hours. Additionally, the dataset includes information about season and month which hurricane occured, nature of the hurricane, and timing of the storm. We are interested in building a bayesian model in order to predict hurricane trajectories, particularly the wind speed at a specific time point. We will utilize Markov Chain Monte-Carlo to estimate the model parameters and their 95% credible intervals through the posterior distributions. Furthermore, we are interested in evaluating our model's ability to accurately predict windspeed and track the hurricanes.

Methods

Bayesian and MCMC Basics

Unlike frequentist approaches, Bayesian analysis treats parameters as random variables and utilizes prior beliefs about the parameters. We start out with a prior distribution of the parameter of interest and then update it with observed data to obtain the posterior distribution. With the posterior distribution, we can obtain parameters estimates such as posterior mean. Bayes Theorem allows us to observe the relationship between prior distribution ($\pi(\theta)$), likelihood ($f(x|\theta)$), and posterior distribution ($\pi(\theta|x)$)

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{m(x)} = \frac{f(x|\theta)\pi(\theta)}{\int f(x|\theta)\pi(\theta)d\theta}$$

Obtaining the posterior distribution is often a difficult task – either there is no closed form or it is computationally intensive to directly sample from $\pi(\theta|x)$. Therefore, MCMC is used to create a Markov chain of the parameter of interest such that their distribution converge to the posterior. A markov chain is a sequence of vectors $X_1, X_2, ..., X_n$ such that the distribution of X_{i+1} depends on only the recent past X_i. In the long run, the sequence generated will be drawn from a stationary distribution, which is the target distribution we want to sample from. The general steps for MCMC include

1. Generate the next sequence for time t_i from the proposal distribution: $\theta \sim q(\theta_i|\theta_{i-1})$
2. Calculate the ratio: $r(\theta_{new}, \theta_{i-1}) = \frac{\pi(\theta_{i-1})q(\theta_i|\theta_{i-1})}{\pi(\theta_i)q(\theta_{i-1}|\theta_i)}$

3. Find the acceptance probability:

$$\alpha(\theta_{new}, \theta_{i-1}) = min[1, r(\theta_{new}, \theta_{i-1})]$$

4. Draw $u \sim uniform(0,1)$. If $u < \alpha(\theta_{new}, \theta_{i-1})$, then $\theta_i = \theta_{new}$. Otherwise $\theta_i = \theta_{i-1}$

There are two approaches to the MCMC algorithm: Gibbs sampling and Metropolis-Hastings. Gibbs sampling is a special case of Metropolis-Hastings when the probability of acceptance is 1. Gibbs sampling involves estimating the stationary distribution by samplling from conditional distributions of the parameters. In our case, since we do not have the conditional distributions of each parameter, we chose to use a Metropolis-Hastings algorithm.

Hurricane Bayesian Model

The Bayesian model used to predict hurricane wind speeds can be expressed as follows: $Y_i(t+6) = \mu_i(t) + \rho_i Y_i(t) + \epsilon_i(t)$ where ρ_j is the autoregressive correlation and the error ϵ_i follows the normal distribution with mean 0 and variance σ^2 independent across t. $\mu_i(t)$ represents the function mean that can be written as follows:

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

where $x_{i,1}(t)$, $x_{i,2}(t)$, and $x_{i,3}(t)$ are the day of the year, calendar year, and type of hurricane at time t, respectively. and

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

are the change of latitude, longitude, and wind speed from time $t-6$ to t. The following prior distributions were assumed: $\pi(\boldsymbol{\beta})$ is jointly normal with mean 0 and variance $diag(1,p)$, $\pi(\rho_j)$ follows a truncated normal $N_{[0,1]}(0.5, 1/5)$, and $\pi(\sigma^{-2})$ follows an inverse-gamma $(0.001, 0.001)$.

Metropolis-Hastings Algorithm

To estimate the parameters, a random walk Metropolis-Hastings (MH) algorithm with component wise updates was utiliized, with a uniform proposal for β and ρ, and a inverse-gamma proposal for σ^2. The MH algorithm constructs a Markov chain by accepting candidate points with probability

$$\alpha(y|x^{(t)}) = min(1, \frac{\pi(y)q(x^{(t)}|y)}{\pi(x^{(t)})q(y|x^{(t)})})$$

We used the uniform distribution as random-walk for the proposal. The random-walk Metropolis is a special class where the proposed transition q is symmetric. As a result, the acceptance probability is only related to the prior distributions and the likelihood. The response wind speed was assumed to follow a normal distribution and the log-likelihood could be expressed as

$$l(\beta, \rho) = -\frac{n}{2} log(2\pi) - nlog(\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y - X^T\beta - \rho Y)^2$$

The partial posterior distribution can be formed by the product of the log-likelihood and the priors. For the MH algorithm, a burn in was incorporated to ensure analysis of a stationary distribution. Additionally, we utlized component-wise MH in which the parameters are updated one by one rather than simultaneously.

The window defines the space with which we will do the random-walk. Tuning these window parameters can help with efficiency of the algorithm. The windows were chosen in a way that the probability of acceptance for each of the parameters were within the range of 30%-70%. The windows for the betas from β_0 to β_6 were 0.13, 0.001, 0.001, 0.08, 0.1, 0.2, and 0.05, respectively. The start values are not as crucial since the MH algorithm will lead to convergence after certain number of steps. For our project, the start values for the betas, rho, and sigma were 2, 0.6, and 10, respectively.

Results

In order to test our predictions, we ran the MH algorithm on 80% of the hurricanes in our data, using 20,000 steps. We then explored the convergence plots in order to assure we achieved convergence and to choose a proper cut-off to consider as the burn-in steps. An example of such plots can be seen in Figure 1, where we observe the random walk for the distribution of beta 5. Here we see a large decrease and increase from 0 to around 9000 until it begins to fluctuate around -0.2. We therefore choose to use step 10000 as our burn-in and used steps 10,001-20,000 to estimate our parameter distribution. The same was done for all parameters.

Our final posterior means and their respective credible intervals (CI) can be found in Table 1. The CI are found by taking the 2.5th and 97.5th quantiles of the posterior distribution of each parameter. We see most of the intervals are narrow, an indication that the spread is small and only marginally fluctuates around the mean. The means of the parameters for the changes from time t to $t+6$ (i.e. latitude, longitude, windspeed) vary. Interestingly, difference in latitudinal mapping has a null effect on the predictions (estimate: -0.05, CI: -0.36, -0.12) while difference in longitudinal mapping has a larger effect (estimate: -0.24, CI: 4.66, 5.82). It appears hurricanes' longitudinal shift has more predictive ability on windspeeds, but this may be due to multicollinearity. It could be helpful to account for their close relationship in our model.

To check how accurate our predictions were, we ran our model on the testing dataset (20% of the hurricanes). We found a root mean squared error (RMSE) of 5.1. To put this on the scale of wind speed, the mean wind speed is 48.06 (25th quantile = 30, 75th quantile = 60). Therefore this is a small yet non-negligible deviation from the true value. However, in Figure 2 we can see the distribution of these residuals. We find they are centered around zero, as we would like. The distribution is left skewed with an outlier around -60, causing the non-zero RMSE. There is a hurricane in our testing data set who's wind speed change from time t to $t+6$ does not follow the same trend as the other hurricanes in our model.

We further explored our predictions by looking at the models ability to predict each hurricanes' wind speed longitudinally. Figure 3 shows how our model performed on four different hurricanes in our testing data set. For hurricanes Ivan and Hortense our model appears to perform well. The predicted wind speed is very close to the observed wind speed. There is no systematic under/over prediction occuring such that the residuals would not be centered around zero, as we see for hurricane Edouard and Arlene. We are systematically overpredicting the windspeed and the residuals are centered around 2 for hurricane Arlene, although the trajectory of the windspeed is roughly correctly modeled.

Discussion

Modeling hurricane wind speeds is an important yet difficult task. Our model aims to get close predictions using an MCMC algorithm . We find that for many hurricanes we are able to get close predictions using this model, although it varies greatly across hurricanes. It would be of interest to further explore the differences in the features of the correctly predicted hurricanes versus the incorrectly predicted hurricanes in order to have more certainty in the accuracy of our model.

We used data from six hours prior to time t to inform the predictions at time t, creating a model that updates with the progression of the hurricane in real time. However, we found that there was some data recorded at time periods less than 6 hours apart. This made it such that the recordings of some hurricanes were not comparable to the others and we could not use them in training our model.

In order to use a Bayesian approach in modeling we had to impose a prior distribution on our parameters of interest. A test to see how sensitive our model is to our prior distributions may inform on the stability of our model and can add value. A choice of priors proposal distribution was also made, we used a uniform distribution. We could try another distribution with an appropriate support to see if our predictions perform better. We found our model was very sensitive to changes in the windown used. Care must be taken to correctly tun the parameters used in this model. However, we managed to achieve a relatively low RMSE with the properly tuned parameters, an encouraging result.

Appendix

Figure 1. Convergence plot for rho estimate

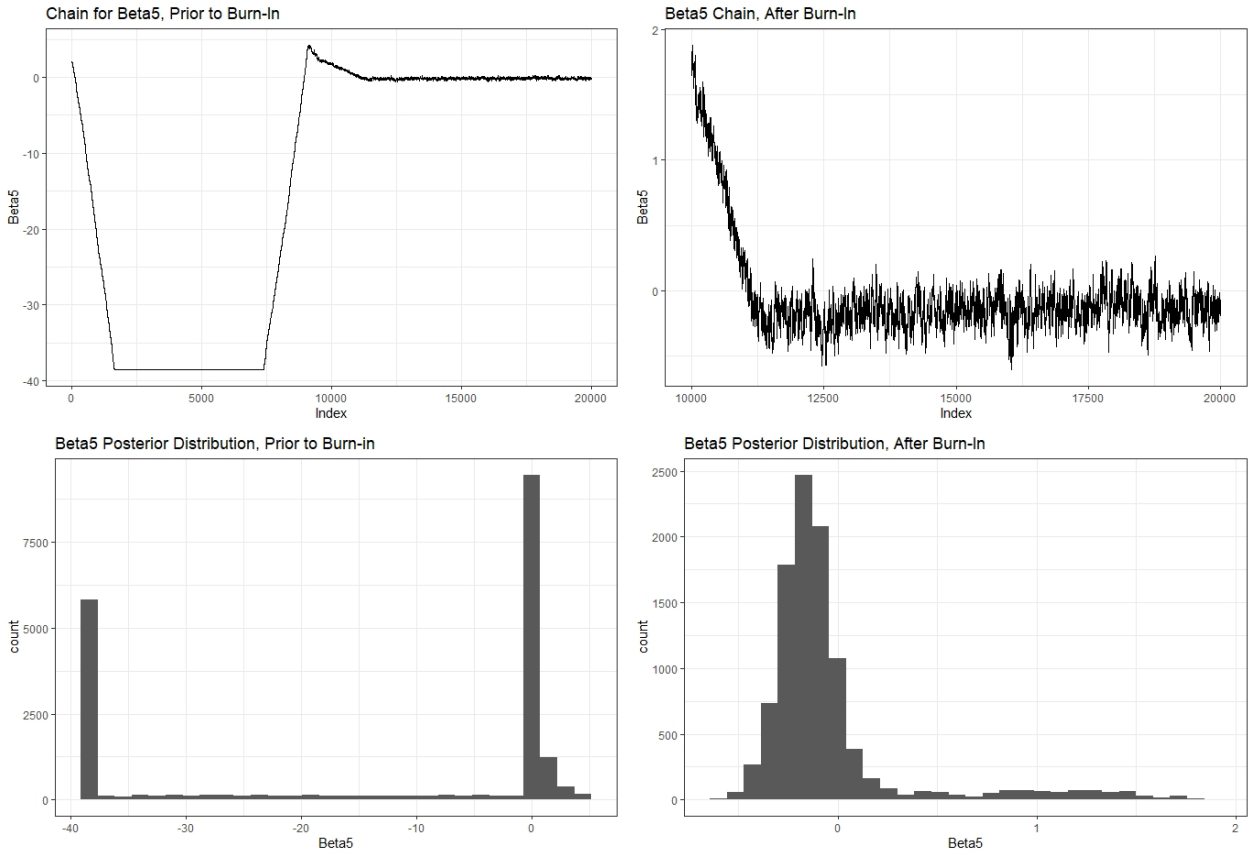


Table 1. Parameter estimates and credible intervals

Parameters	Posterior.Mean	L95.	U95.
Intercept	-11.12	-31.88	-0.98
Day of Year	0.00	-0.01	0.00
Year	0.01	0.00	0.02
Nature/Type	0.34	0.00	2.06
Diff Lat	-0.05	-0.40	1.29
Diff Long	-0.24	-0.36	-0.12
Diff Wind	0.50	0.47	0.59
Rho	0.94	0.71	0.96
Sigma	43.52	39.11	44.34

Figure 2. Histogram of Residuals on Testing Dataset

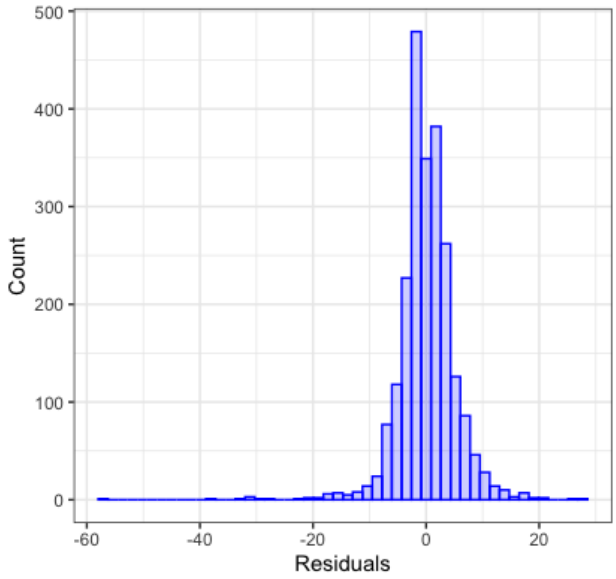
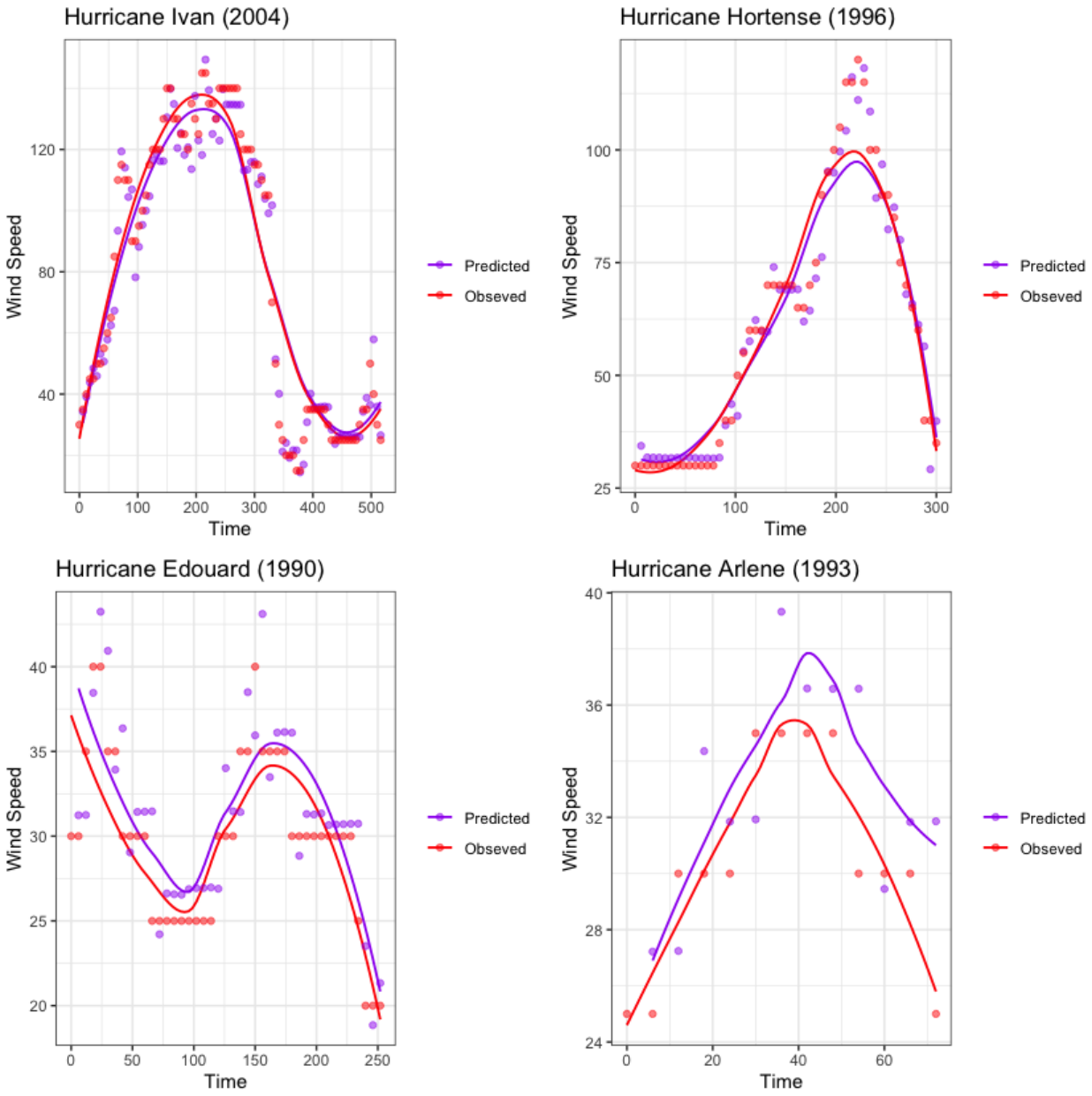


Figure 3. Wind Speed Predictions over Time



Code

```
library(tidyverse)
library(caret)
library(lubridate)
library(msm)
library(invgamma)


#Data Cleaning:

dt = read.csv("./hurrican356.csv") %>%
  arrange(-desc(time)) %>%
  separate(time, into = c("date", "time"), sep = " ") %>%
  mutate(time = str_replace(time, "\\)", ""),
         date = str_replace(date, "\\(", ""),
         day_of_yr = ymd(date) - ymd(paste0(year(date), "-01-01")),
         day_of_yr = as.numeric(day_of_yr, units = "days")) %>%
  separate(time, into = c("time", "temp", "temp2"), sep = ":") %>%
  mutate(time = as.numeric(time)) %>%
  select(-c(X, date, temp, temp2)) %>%
  filter(time %in% c(0, 6, 12, 18)) %>%
  group_by(ID) %>%
  mutate(t = 1,
         t = (cumsum(t) - 1) * 6,
         diff_lat = Latitude - lag(Latitude, default = NA),
         diff_long = Longitude - lag(Longitude, default = NA),
         diff_spd = Wind.kt - lag(Wind.kt, default = NA),
         Wind.t6 = lead(Wind.kt),
         Nature = factor(Nature,
                         levels = c("NR", "DS", "ET", "SS", "TS")),
         Nature = as.numeric(Nature)) %>%
  rename(year = Season) %>%
  ungroup()

# functions for posteriors
# use log to make it additive

rho_prior = function(rho){
  prob = dtnorm(rho, mean = 0.5, sd = sqrt(1/5), lower = 0, upper = 1)
  return(log(prob))
}

sigma_prior = function(sigma){
  prob = dinvgamma(1/sigma, 0.001, 0.001)
  return(log(prob))
}

beta_prior = function(beta){
  prob = dnorm(beta, 0, 1)
  return(sum(log(prob)))
}
```

```
loglike = function(Y, X, betas, rhos, sigma){
  resp = as.matrix(Y)
  X = data.matrix(X)
  coef = c(betas, rhos)
  mean = X %*% coef
  # constants removed since they will be canceled out anyway
  loglik = (-0.5/sigma)*(resp - mean)^2
  return(sum(loglik))
}

posteriors = function(Y, X, betas, rhos, sigma) {
  loglik = loglike(Y, X, betas, rhos, sigma)
  beta_p = beta_prior(betas)
  rho_p = rho_prior(rhos)
  sigma_p = sigma_prior(sigma)
  return(rho_p + sigma_p + beta_p + loglik)
}

# create MH algorithm
mh = function(Y, X, start_b, start_r, start_s,
              window_b, window_r, window_s, n){
  p = length(start_b)
  r = length(start_r)
  res_b = start_b
  res_r = start_r
  res_s = start_s
  chain = c(1, start_b, start_r, start_s)
  for (i in 2:n){
    for (j in 1:p) {
      proposed_b = res_b
      proposed_b[j] = res_b[j] + (runif(1) - 0.5)*2*window_b[j]
      numer = posteriors(Y, X, proposed_b, res_r, res_s)
      denom = posteriors(Y, X, res_b, res_r, res_s)
      if (log(runif(1)) < (numer - denom)) {
        res_b = proposed_b
      }
    }
    for (m in 1:r) {
      proposed_r = res_r
      proposed_r[m] = res_r[m] + (runif(1) - 0.5)*2*window_r
      numer = posteriors(Y, X, res_b, proposed_r, res_s)
      denom = posteriors(Y, X, res_b, res_r, res_s)
      if (log(runif(1)) < (numer - denom)) {
        res_r = proposed_r
      }
    }
    proposed_s = 1/rinvgamma(1, .001*window_s, .001*window_s)
    numer = posteriors(Y, X, res_b, res_r, proposed_s)
    denom = posteriors(Y, X, res_b, res_r, res_s)
    if (log(runif(1)) < (numer - denom)) {
      res_s = proposed_s
    }
    chain = rbind(chain, c(i, res_b, res_r, res_s))
```

```
  }
  return(chain)
}

#Filter missing data
dt = dt %>% filter(!is.na(Wind.t6))

# select 80% of hurricanes as training set
hurricanes_id = as.character(unique(dt$ID))
train_ind = sample(1:length(hurricanes_id),
                   size = floor(.8*length(hurricanes_id)),
                   replace = FALSE)
train_hurr = hurricanes_id[train_ind]

# create datasets for response and predictors for
# training and test sets
train_X = dt %>%
  filter(ID %in% train_hurr) %>%
  mutate(intercept = 1,
         diff_lat = ifelse(is.na(diff_lat), 0, diff_lat),
         diff_long = ifelse(is.na(diff_long), 0, diff_long),
         diff_spd = ifelse(is.na(diff_spd), 0, diff_spd)) %>%
  select(intercept, day_of_yr, year, Nature,
         diff_lat, diff_long, diff_spd, Wind.kt)

train_Y = dt %>%
  filter(ID %in% train_hurr) %>%
  select(Wind.t6)

test_X = dt %>%
  filter(!(ID %in% train_hurr)) %>%
  mutate(intercept = 1) %>%
  select(intercept, day_of_yr, year, Nature,
         diff_lat, diff_long, diff_spd, Wind.kt)

test_Y = dt %>%
  filter(!(ID %in% train_hurr)) %>%
  select(Wind.t6)

#Run algorithm
set.seed(24)
chains = mh(train_Y, train_X, start_b = rep(2,7),
            start_r = 0.6, start_s = 10, window_b = rep(3,7),
            window_r = 0.02, window_s = 150, n = 10000)

## Unique values
length(unique(chains[,2]))
length(unique(chains[,3]))
length(unique(chains[,4]))
length(unique(chains[,5]))
length(unique(chains[,6]))
length(unique(chains[,7]))
length(unique(chains[,8]))
```

```r
length(unique(chains[,9]))
length(unique(chains[,10]))

## Visualization of posterior distributions and chains to find proper burn-in
# Visualize chains
chains_dat = chains %>% as.data.frame()

chains_dat %>%
  ggplot(aes(x = V1, y = V10)) +
  geom_line() +
  labs(x = "Index",
       y = "Variance")

chains_dat %>%
  ggplot(aes(x = V1, y = V9)) +
  geom_line() +
  labs(x = "Index", y = "Rho")

chains_dat %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_line() +
  labs(x = "Index", y = "Intercept")

chains_dat %>%
  ggplot(aes(x = V1, y = V4)) +
  geom_line() +
  labs(x = "Index", y = "Beta3")

#Visualize posteriors distributions
chains_dat %>%
  ggplot(aes(x = V2)) +
  geom_histogram() +
  labs(x = "Intercept")

chains_dat %>%
  ggplot(aes(x = V10)) +
  geom_histogram() +
  labs(x = "Variance")

chains_dat %>%
  ggplot(aes(x = V9)) +
  geom_histogram() +
  labs(x = "Rho") +
  theme_bw()

chains_dat %>%
  ggplot(aes(x = V4)) +
  geom_histogram() +
  labs(x = "Beta3")

chains_dat %>%
  ggplot(aes(x = V6)) +
  geom_histogram() +
```

```
  labs(x = "Beta5")

## Apply burn-in
##Remove first 10000 from chain
burn_in_chains = chains_dat[10001:dim(chains_dat)[1],]

# Visualize posteriors and chains - convergence plot
burn_in_chains %>%
  ggplot(aes(x = V1, y = V10)) +
  geom_line() +
  labs(x = "Index",
       y = "Variance")

burn_in_chains %>%
  ggplot(aes(x = V1, y = V9)) +
  geom_line() +
  labs(x = "Index", y = "Rho")

burn_in_chains %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_line() +
  labs(x = "Index", y = "Intercept")

burn_in_chains %>%
  ggplot(aes(x = V1, y = V4)) +
  geom_line() +
  labs(x = "Index", y = "Beta3")

burn_in_chains %>%
  ggplot(aes(x = V1, y = V6)) +
  geom_line() +
  labs(x = "Index", y = "Beta5")


# Visualize posterior distributions
burn_in_chains %>%
  ggplot(aes(x = V2)) +
  geom_histogram() +
  labs(x = "Intercept")

burn_in_chains %>%
  ggplot(aes(x = V10)) +
  geom_histogram() +
  labs(x = "Variance")

burn_in_chains %>%
  ggplot(aes(x = V9)) +
  geom_histogram() +
  labs(x = "Rho")

burn_in_chains %>%
  ggplot(aes(x = V4)) +
  geom_histogram() +
```

```
  labs(x = "Beta3")

burn_in_chains %>%
  ggplot(aes(x = V6)) +
  geom_histogram() +
  labs(x = "Beta5")

## Prediction and Inference
# Take average of coefficients
beta_avg = burn_in_chains %>% summarise_at(c("V2", "V3", "V4", "V5", "V6", "V7", "V8"),
                                           mean, na.rm = T)
rho_avg = burn_in_chains %>% summarise_at("V9", mean, na.rm = T)
sigma_avg = burn_in_chains %>% summarise_at("V10", mean, na.rm = T)

# 95% CI
beta_avg_l95 = burn_in_chains %>% summarise_at(c("V2", "V3", "V4", "V5", "V6", "V7", "V8"),
                                               quantile, 0.025, na.rm = T)
beta_avg_u95 = burn_in_chains %>% summarise_at(c("V2", "V3", "V4", "V5", "V6", "V7", "V8"),
                                               quantile, 0.975, na.rm = T)

rho_avg_l95 = burn_in_chains %>% summarise_at("V9", quantile, 0.025, na.rm = T)
rho_avg_u95 = burn_in_chains %>% summarise_at("V9", quantile, 0.975, na.rm = T)

sig_avg_l95 = burn_in_chains %>% summarise_at("V10", quantile, 0.025, na.rm = T)
sig_avg_u95 = burn_in_chains %>% summarise_at("V10", quantile, 0.975, na.rm = T)

# Table of parameters
par_avg = cbind(beta_avg, rho_avg, sigma_avg) %>% round(digits = 2)
par_l95 = cbind(beta_avg_l95, rho_avg_l95, sig_avg_l95) %>% round(digits = 2)
par_u95 = cbind(beta_avg_u95, rho_avg_u95, sig_avg_u95) %>% round(digits = 2)
parameters = c("Intercept", "Day of Year", "Year", "Nature/Type",
               "Diff Lat", "Diff Long", "Diff Wind", "Rho", "Sigma")

par_table = cbind(parameters, t(par_avg), t(par_l95), t(par_u95)) %>% as.data.frame()
names(par_table) = c("Parameters", "Posterior Mean", "L95%", "U95%")
row.names(par_table) = c()
write_csv(par_table, "table_parameters.csv")

# Predict wind speed at time t
par = unlist(c(beta_avg, rho_avg))

# Test data
test_X = dt %>%
  filter(!(ID %in% train_hurr)) %>%
  mutate(intercept = 1) %>%
  select(intercept, day_of_yr, year, Nature,
         diff_lat, diff_long, diff_spd, Wind.kt)

test_Y = dt %>%
  filter(!(ID %in% train_hurr)) %>%
  select(Wind.t6)

test.fit = data.matrix(test_X) %*% par
```

```
test.res = test_Y - test.fit

# Pull IDs from original data
test.hurricanes = dt %>%
  filter(!(ID %in% train_hurr)) %>%
  select(ID)
# Data set with Predicted and Accurate values
hurricane_predict = tibble(
  id = test.hurricanes,
  test_Y = test_Y,
  pred_Y = test.fit
)

preds <- cbind(id = test.hurricanes, test_X, true = test_Y, preds = test.fit, res = test.res) %>%
  group_by(ID) %>%
  mutate(t = 1,
         t = (cumsum(t) - 1) * 6) %>%
  rename(resids = Wind.t6.1)


###Explore model accuracy
preds %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                 fill="blue",
                 alpha = .2,
                 bins = 50) +
  labs(title="", x="Residuals", y="Count") +
  theme_bw()


resids.no.na <- preds %>% filter(!is.na(preds))
mse(resids.no.na$Wind.t6, resids.no.na$preds)

###Explore model on individual hurricanes
preds %>%
  filter(ID == "IVAN.2004") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.35, se = F, size = 0.65) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Ivan (2004)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
    scale_color_manual(labels = c("Predicted", "Obseved") , values = c("purple", "red")) +
  theme_bw()


preds %>%
  filter(ID == "IVAN.2004") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
```

```
                fill="blue",
                alpha = .2) +
  labs(title="Felix 1995", x="Residuals", y="Count") +
  theme_bw()


preds %>%
  filter(ID == "KYLE.2002") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.5, se = F) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Dennis (1999)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
    scale_color_manual(labels = c("Predicted", "Obseved") , values = c("lightblue", "red")) +
  theme_bw()


preds %>%
  filter(ID == "KYLE.2002") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                fill="blue",
                alpha = .2) +
  labs(title="Felix 1995", x="Residuals", y="Count") +
  theme_bw()

preds %>%
  filter(ID == "ERIKA.1997") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.35, se = F, size = 0.65) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Erika (1997)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
  scale_color_manual(labels = c("Predicted", "Obseved") , values = c("purple", "red")) +
  theme_bw()

preds %>%
  filter(ID == "ERIKA.1997") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                fill="blue",
                alpha = .2) +
  labs(title=" ", x="Residuals", y="Count") +
  theme_bw()

preds %>%
  filter(ID == "LUIS.1995") %>%
```

```
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.35, se = F, size = 0.65) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Luis (1995)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
    scale_color_manual(labels = c("Predicted", "Obseved") , values = c("purple", "red")) +
  theme_bw()

preds %>%
  filter(ID == "LUIS.1995") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                 fill="blue",
                 alpha = .2) +
  labs(title=" ", x="Residuals", y="Count") +
  theme_bw()

preds %>%
  filter(ID == "HUGO.1989") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.5, se = F) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Hugo (1989)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
    scale_color_manual(labels = c("Predicted", "Obseved") , values = c("lightblue", "red"))

preds %>%
  filter(ID == "HUGO.1989") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                 fill="blue",
                 alpha = .2) +
  labs(title="Irene 2005", x="Residuals", y="Count") +
  theme_bw()


preds %>%
  filter(ID == "ARLENE.1993") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.35, se = F, size = 0.65) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Arlene (1993)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
  scale_color_manual(labels = c("Predicted", "Obseved") , values = c("purple", "red")) +
```

```
  theme_bw()

preds %>%
  filter(ID == "ARLENE.1993") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                 fill="blue",
                 alpha = .2) +
  labs(title="", x="Residuals", y="Count") +
  theme_bw()


preds %>%
  filter(ID == "NOEL.1995") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.35, se = F, size = 0.65) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Noel (1995)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
  scale_color_manual(labels = c("Predicted", "Obseved") , values = c("purple", "red")) +
  theme_bw()

preds %>%
  filter(ID == "NOEL.1995") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                 fill="blue",
                 alpha = .2) +
  labs(title="", x="Residuals", y="Count") +
  theme_bw()


preds %>%
  filter(ID == "EMILY.1993") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.35, se = F, size = 0.65) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Emily (1993)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
  scale_color_manual(labels = c("Predicted", "Obseved") , values = c("purple", "red")) +
  theme_bw()

preds %>%
  filter(ID == "EMILY.1993") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                 fill="blue",
```

```
                alpha = .2) +
  labs(title="", x="Residuals", y="Count") +
  theme_bw()

preds %>%
  filter(ID == "EDOUARD.1990") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.35, se = F, size = 0.65) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Edouard (1990)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
  scale_color_manual(labels = c("Predicted", "Obseved") , values = c("purple", "red")) +
  theme_bw()

preds %>%
  filter(ID == "EDOUARD.1990") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                 fill="blue",
                 alpha = .2) +
  labs(title="", x="Residuals", y="Count") +
  theme_bw()

preds %>%
  filter(ID == "HORTENSE.1996") %>%
  gather(key = type, value = wind.speed, c("preds", "Wind.t6")) %>%
  ggplot(aes(x = t, y = wind.speed, group = type, colour = type)) +
  geom_smooth(alpha = 0.35, se = F, size = 0.65) +
  geom_point(alpha = 0.5) +
      labs(title = "Hurricane Hortense (1996)",
           x = "Time",
         y = "Wind Speed",
         color = "") +
  scale_color_manual(labels = c("Predicted", "Obseved") , values = c("purple", "red")) +
  theme_bw()

preds %>%
  filter(ID == "HORTENSE.1996") %>%
  ggplot(aes(x = resids)) +
  geom_histogram(col="blue",
                 fill="blue",
                 alpha = .2) +
  labs(title="", x="Residuals", y="Count") +
  theme_bw()
```