Task 4

Task 2: With the estimated model parameters and covariate values, you can calculate the predicted wind speed for each time point using the model equation. This way, you can track the hurricane and compare the predicted wind speeds with the actual wind speeds recorded during the hurricane. Please evaluate how well the estimated Bayesian model can track individual hurricanes.

Solution

Import parameters from task 1 & 2.

```
# load parameters
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

# burn in the MC chains. change this based on the resulting plots
burn = 500
# index of useful samples (used for estimates & CIs)
index = (burn + 1):10000
beta_sample = beta_list
mu_sample = mu_list[index,]
gamma_sample = gamma_list[index,]
```

Implement the prediction process of wind speed for each hurricane in R.

```
# final parameters to be used
para_beta = beta_sample[1,]
para_beta = as.matrix(para_beta)
para_mu = rbind(colMeans(mu_sample))
para_gamma = rbind(colMeans(gamma_sample))

# prediction function
Windspeed_Predict = function(index_hurricane, index_time){
  predict_speed =
    Z[[index_hurricane]][index_time,] %*% para_beta[((index_hurricane - 1) * 5 + 1):((index_hurricane -
    (X %*% t(para_gamma))[index_hurricane, ]
  return(predict_speed)
}

# functional test
Windspeed_Predict(1, 26)
```

```
##          [,1]
## [1,] 23.11151
```

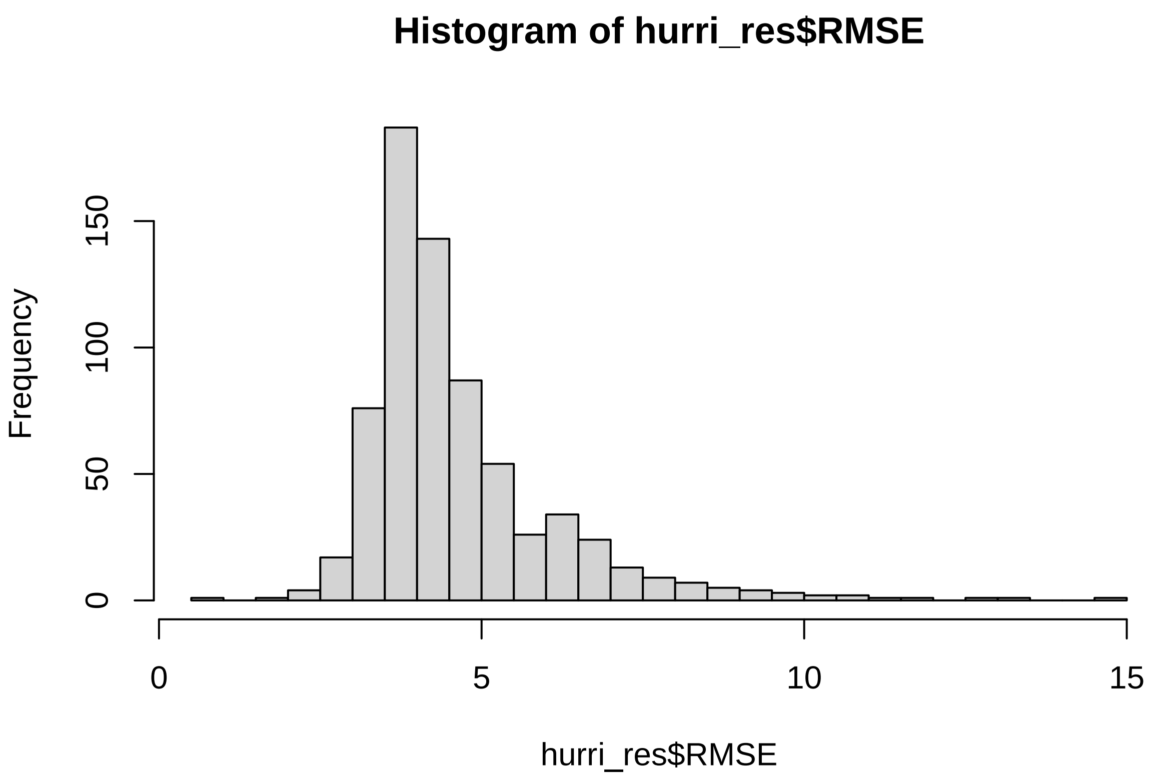
```
# initialize prediction table
Y_table = split(Training$Wind.kt, Training$ID) %>%
  lapply(function(x) x[-c(1:2)]) %>%
  lapply(as.data.frame) %>%
  lapply(function(df) {
    df$wind_obs = df$`X[[i]]`
    df$wind_predict = df$wind_obs
    df = as.matrix(df)
    subset(df, select = c("wind_obs", "wind_predict"))
  })

# updating prediction table
for (i in 1:length(names(Z))) {
 for (j in 1:nrow(Z[[i]])) {
 Y_table[[i]][, 2][j] = Windspeed_Predict(i, j)
  }
}
```

Data visualization for prediction results.

```
hurri_res = data.frame(ID = "example",
                       RMSE = 0)
for (i in 1:length(names(Z))) {
  RMSE = sqrt(mean((Y_table[[i]][,1] - Y_table[[i]])^2))
  new_row = c(names(Z)[i], RMSE)
  hurri_res = rbind(hurri_res, new_row)
}

hurri_res = hurri_res[-1, ]
hurri_res$RMSE = as.numeric(hurri_res$RMSE)
hurricane_info = Training %>%
  group_by(ID) %>%
  slice(1) %>%
  dplyr::select(ID, Season, Month, Nature) %>%
  ungroup(ID) %>%
  mutate(
    Active = ifelse(Month %in% month.name[8:10], "Active", "Inactive"),
    Active = factor(Active, levels = c("Inactive", "Active")))

hurricane_loc = Training %>%
  distinct(ID, .keep_all = TRUE) %>%
  select(ID, Latitude, Longitude, Wind.kt) %>%
  mutate(Start_Lat = Latitude,
         Start_Lon = Longitude,
         Start_Speed = Wind.kt) %>%
  select(ID, Start_Lat, Start_Lon, Start_Speed)

hurricane_info = left_join(hurricane_info, hurricane_loc, by = "ID")

hurri_res = left_join(hurri_res, hurricane_info, by = "ID")
```

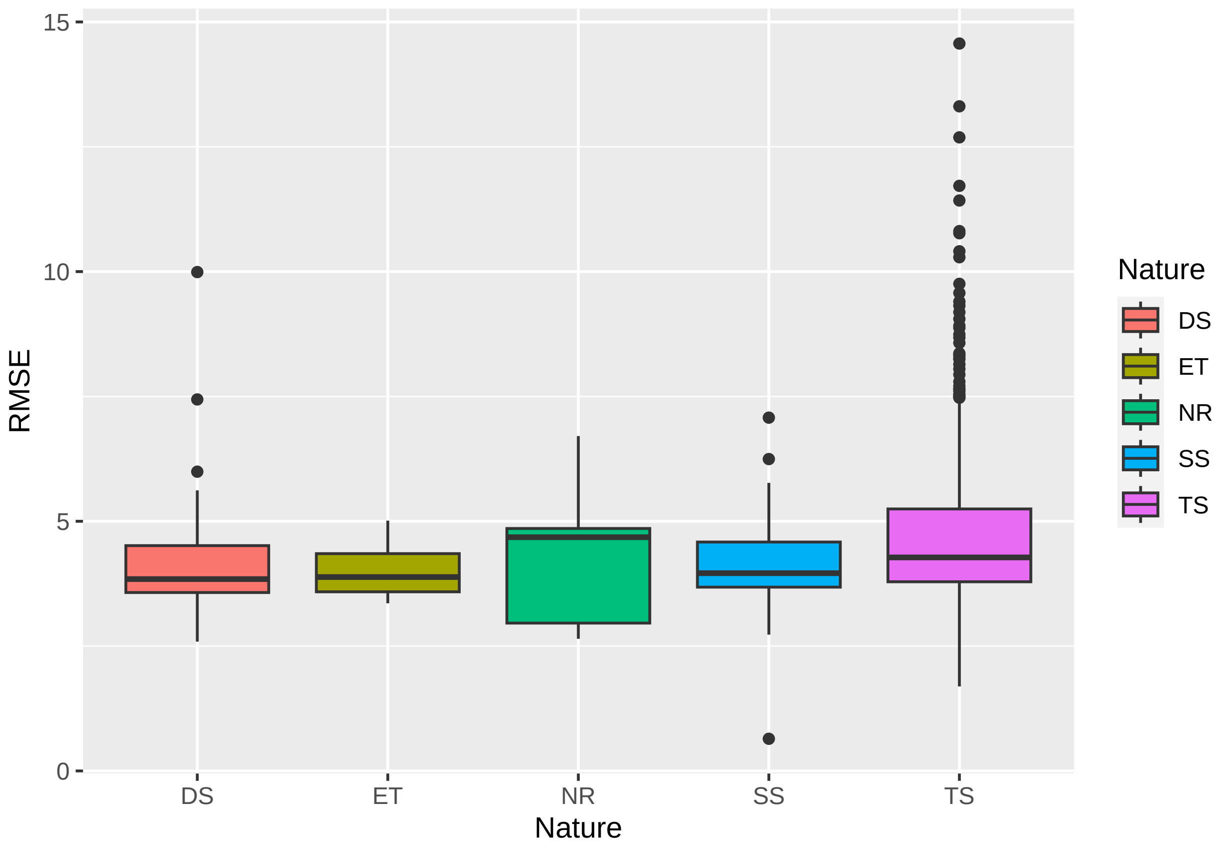
The distribution of overall RMSE in prediction.

```
# density of RMSE
hist(hurri_res$RMSE, breaks = 20)
```

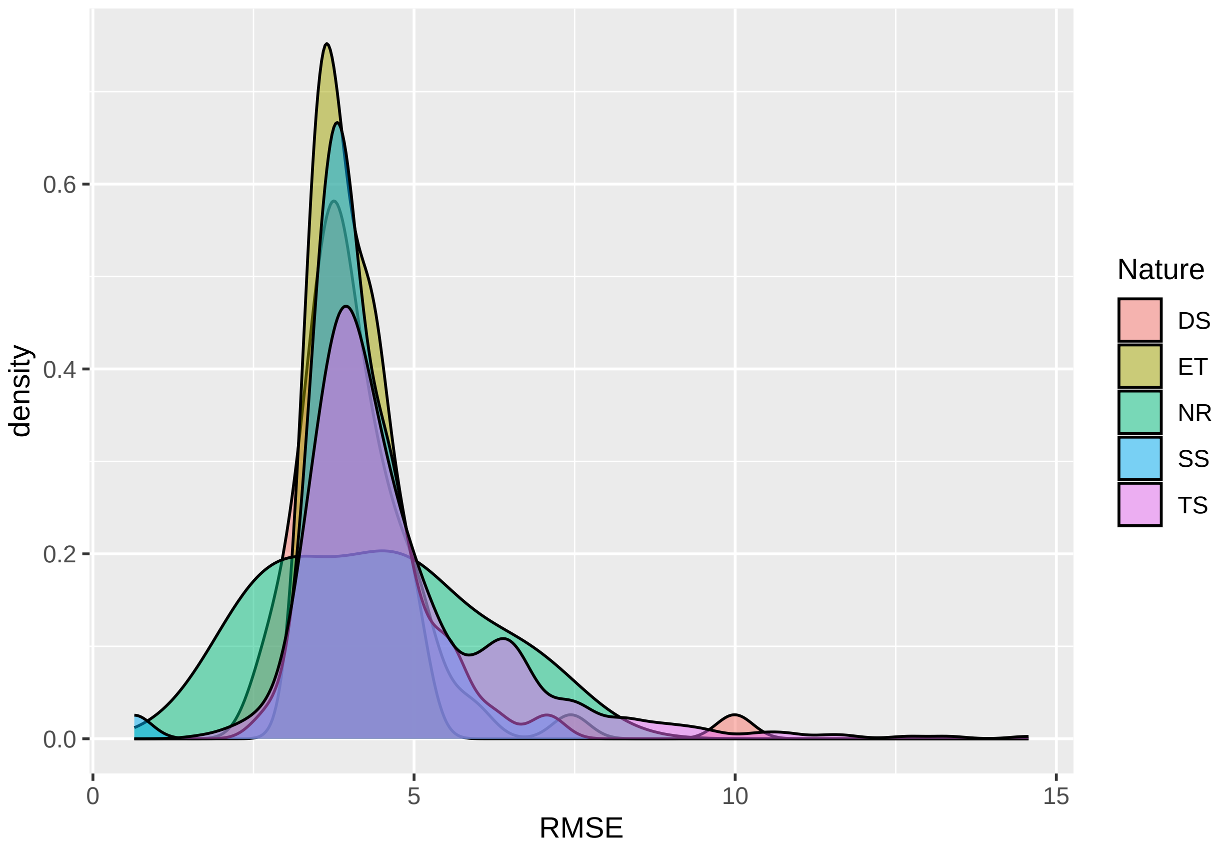


The difference of RMSE distribution of difference properties.

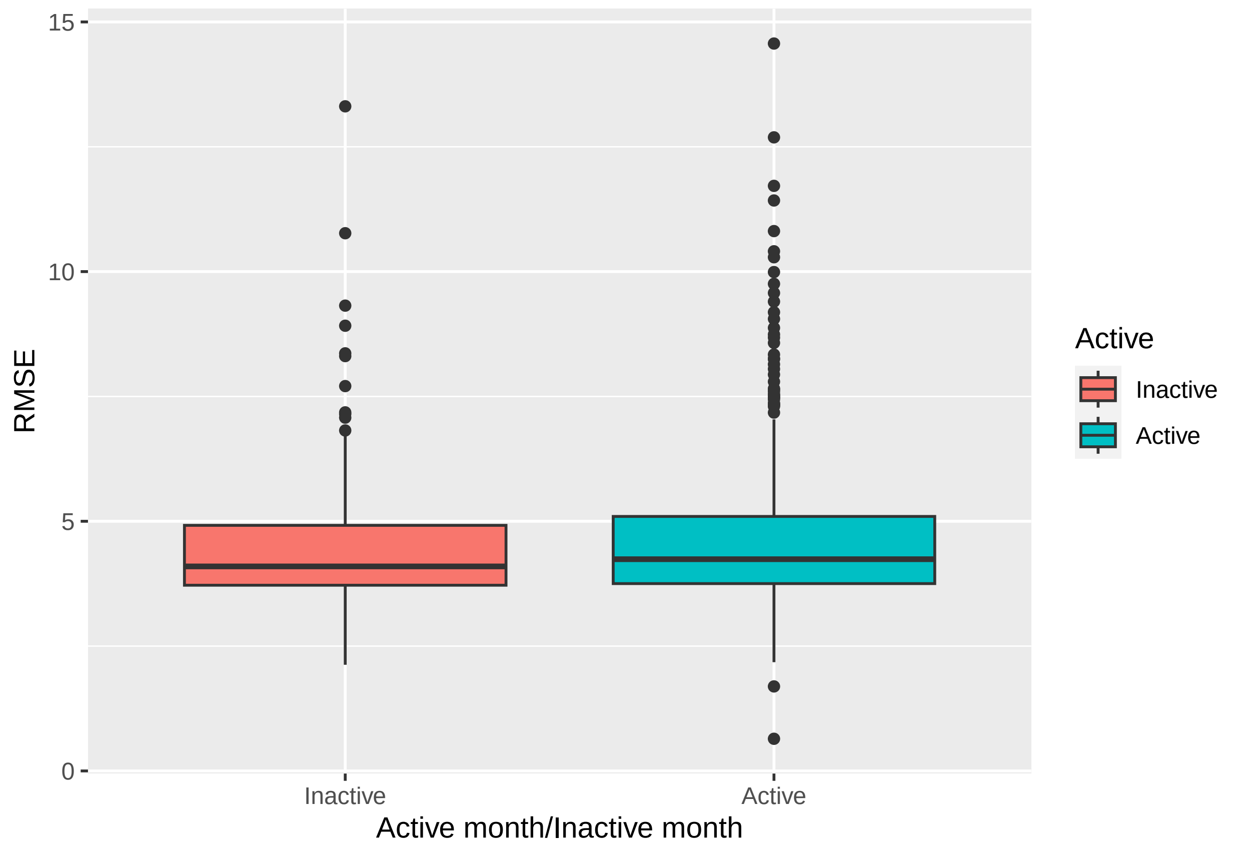
```
# distribution of RMSE on hurricane nature
ggplot(hurri_res, aes(x = Nature, y = RMSE, fill = Nature)) +
  geom_boxplot() +
  scale_color_manual(values = c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2")) +
  labs(x = "Nature", y = "RMSE")
```



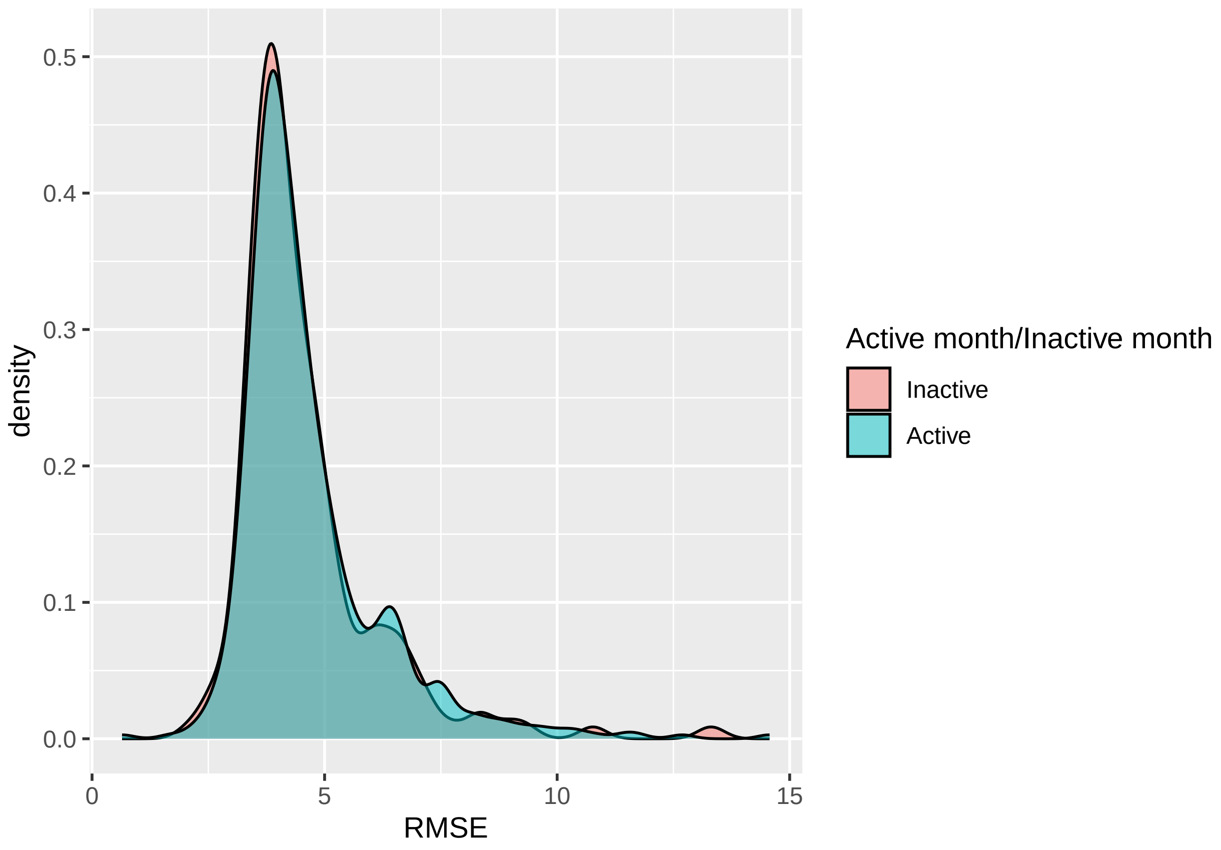
```
ggplot(hurri_res, aes(x = RMSE, fill = Nature)) +
  geom_density(alpha = 0.5) +
  labs(x = "RMSE", fill = "Nature")
```



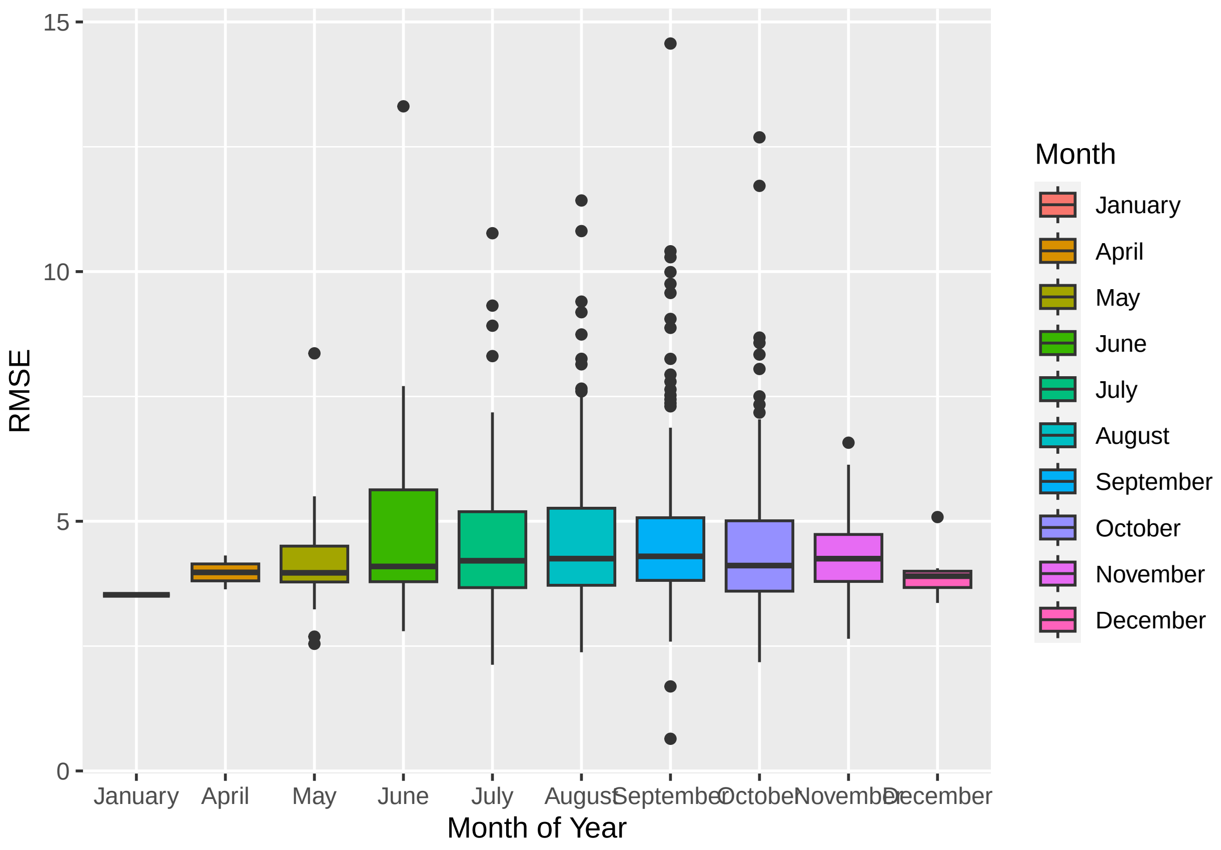
```
# distribution of RMSE on hurricane active months
ggplot(hurri_res, aes(x = Active, y = RMSE, fill = Active)) +
  geom_boxplot() +
  labs(x = "Active month/Inactive month", y = "RMSE")
```



```
ggplot(hurri_res, aes(x = RMSE, fill = Active)) +
  geom_density(alpha = 0.5) +
  labs(x = "RMSE", fill = "Active month/Inactive month")
```

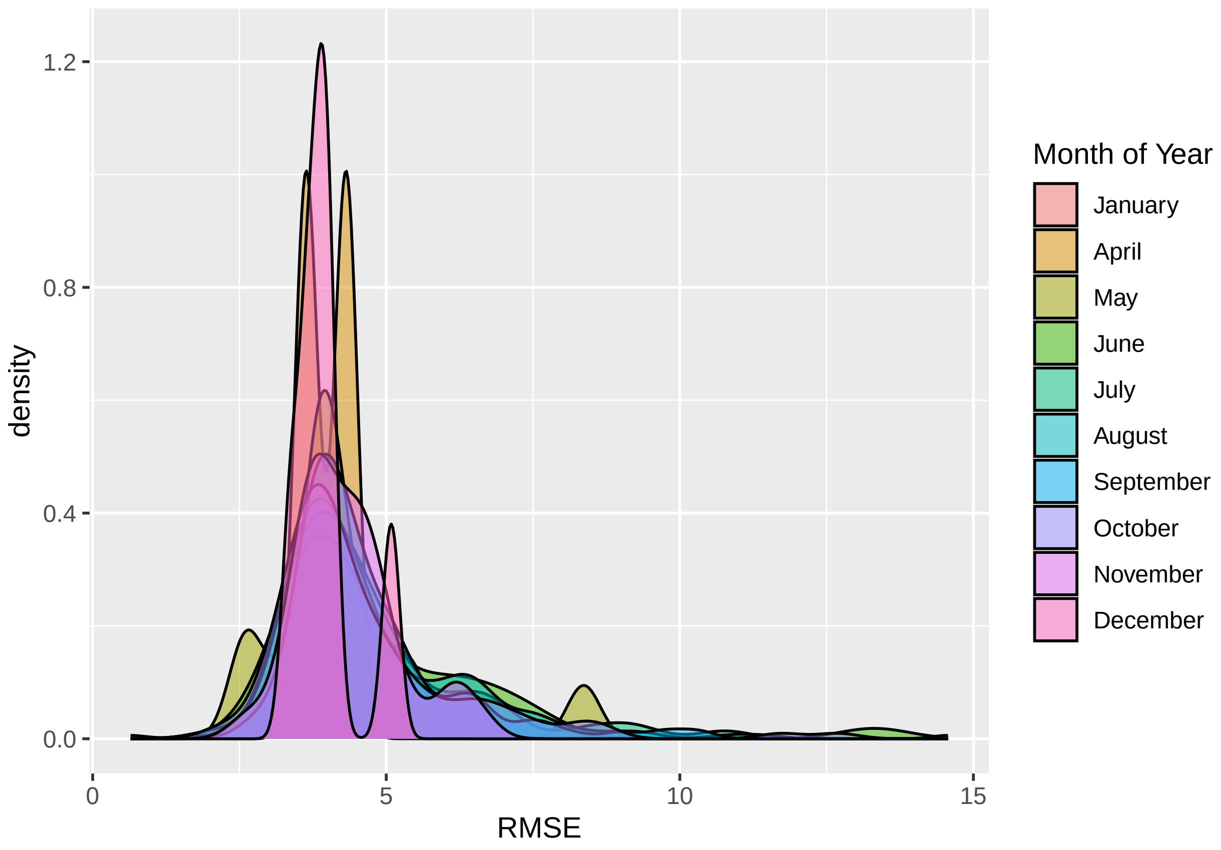


```
ggplot(hurri_res, aes(x = Month, y = RMSE, fill = Month)) +
  geom_boxplot() +
  labs(x = "Month of Year", y = "RMSE")
```



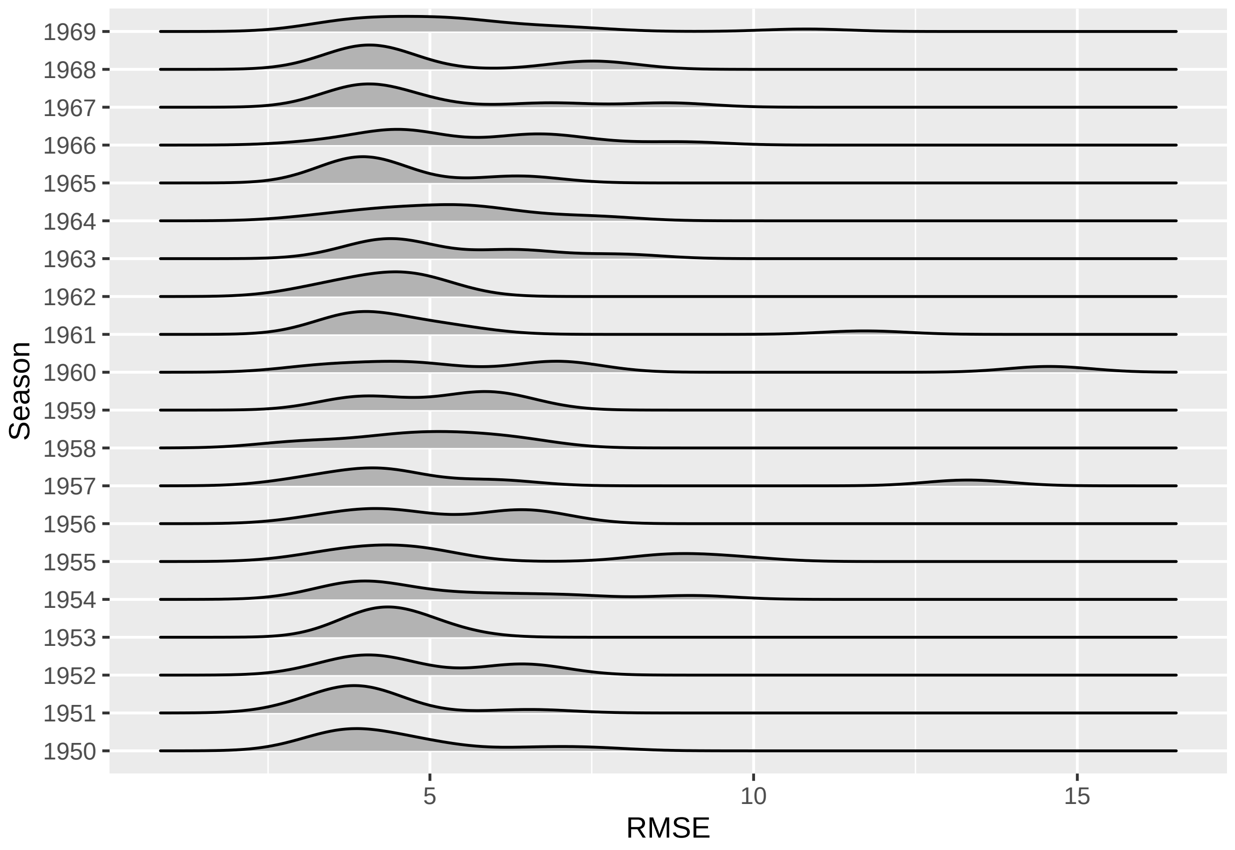
```r
ggplot(hurri_res, aes(x = RMSE, fill = Month)) +
  geom_density(alpha = 0.5) +
  labs(x = "RMSE", fill = "Month of Year")
```

```
## Warning: Groups with fewer than two data points have been dropped.

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



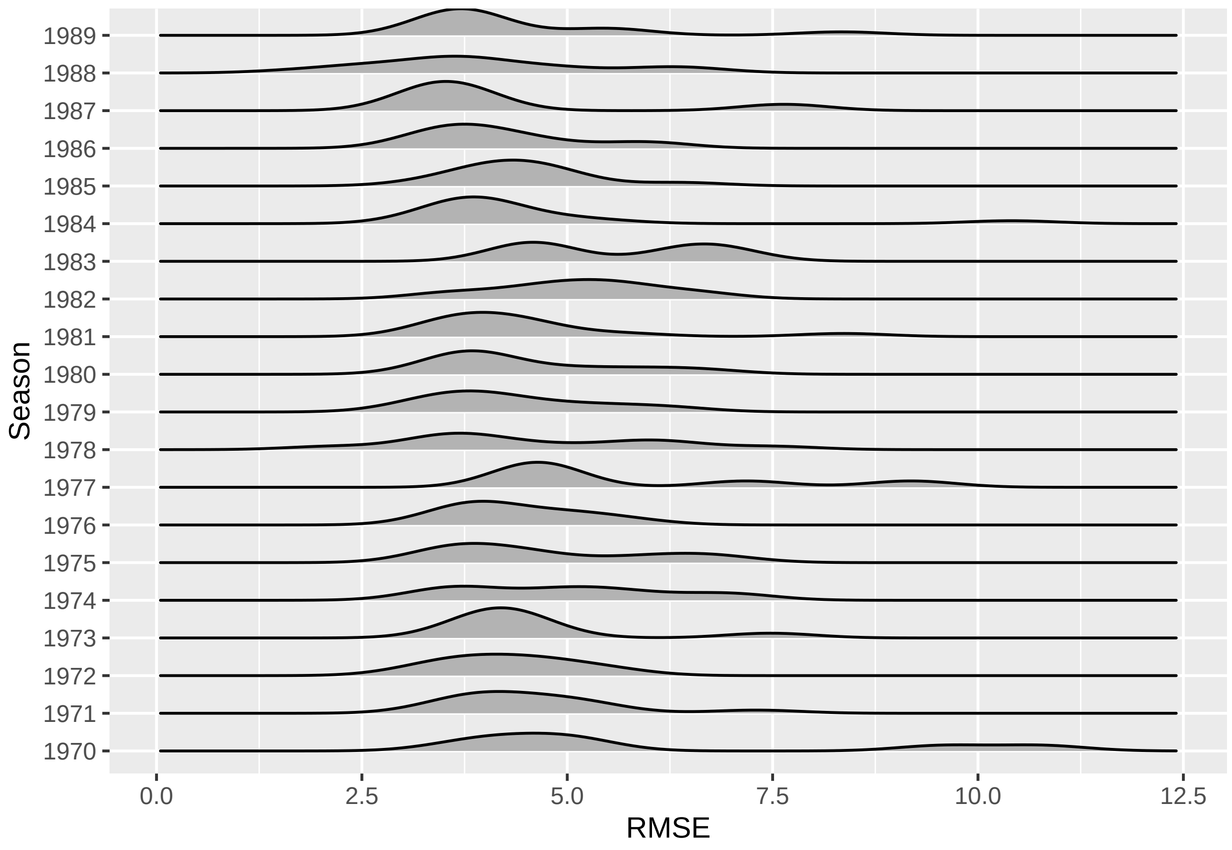
```
# distribution of RMSE on hurricane season
ggplot(hurri_res %>% filter(Season < 1970) %>% mutate(Season = factor(Season)), aes(x = RMSE, y = Seaso
  geom_density_ridges(scale = 0.8) +
  labs(x = "RMSE", y = "Season")
```

```
## Picking joint bandwidth of 0.654
```



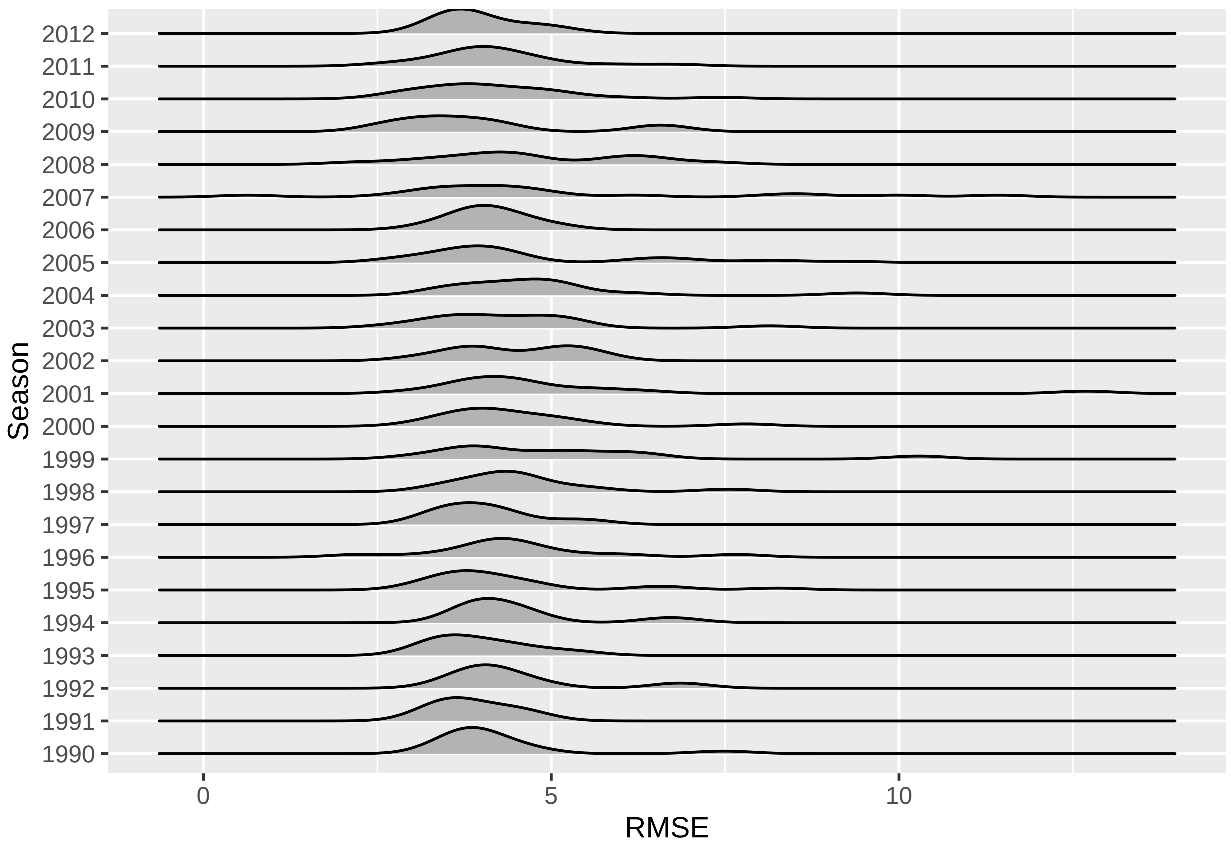
```
ggplot(hurri_res %>% filter(Season >= 1970 & Season < 1990) %>% mutate(Season = factor(Season)), aes(x =
  geom_density_ridges(scale = 0.8) +
  labs(x = "RMSE", y = "Season")
```

```
## Picking joint bandwidth of 0.548
```



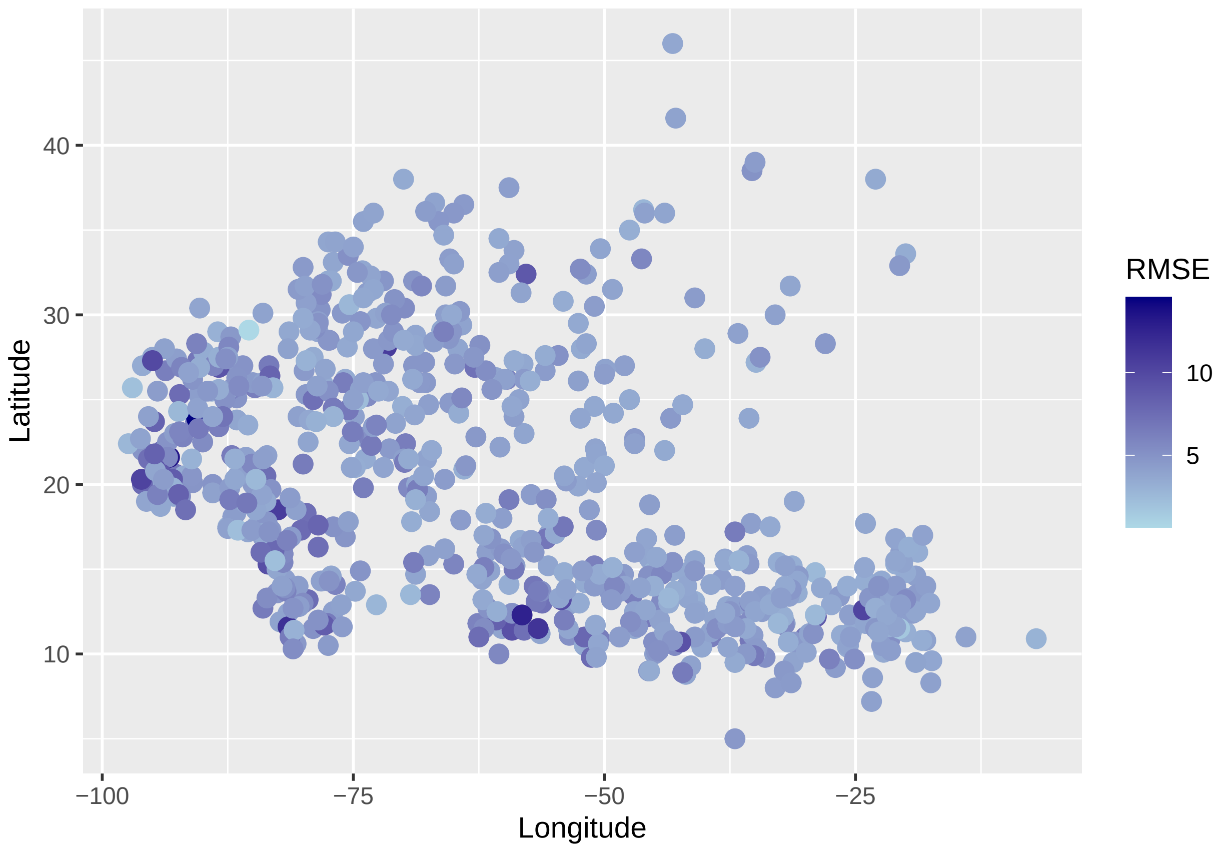
```r
ggplot(hurri_res %>% filter(Season >= 1990 & Season < 2013) %>% mutate(Season = factor(Season)), aes(x =
  geom_density_ridges(scale = 0.8) +
  labs(x = "RMSE", y = "Season")
```

```
## Picking joint bandwidth of 0.427
```

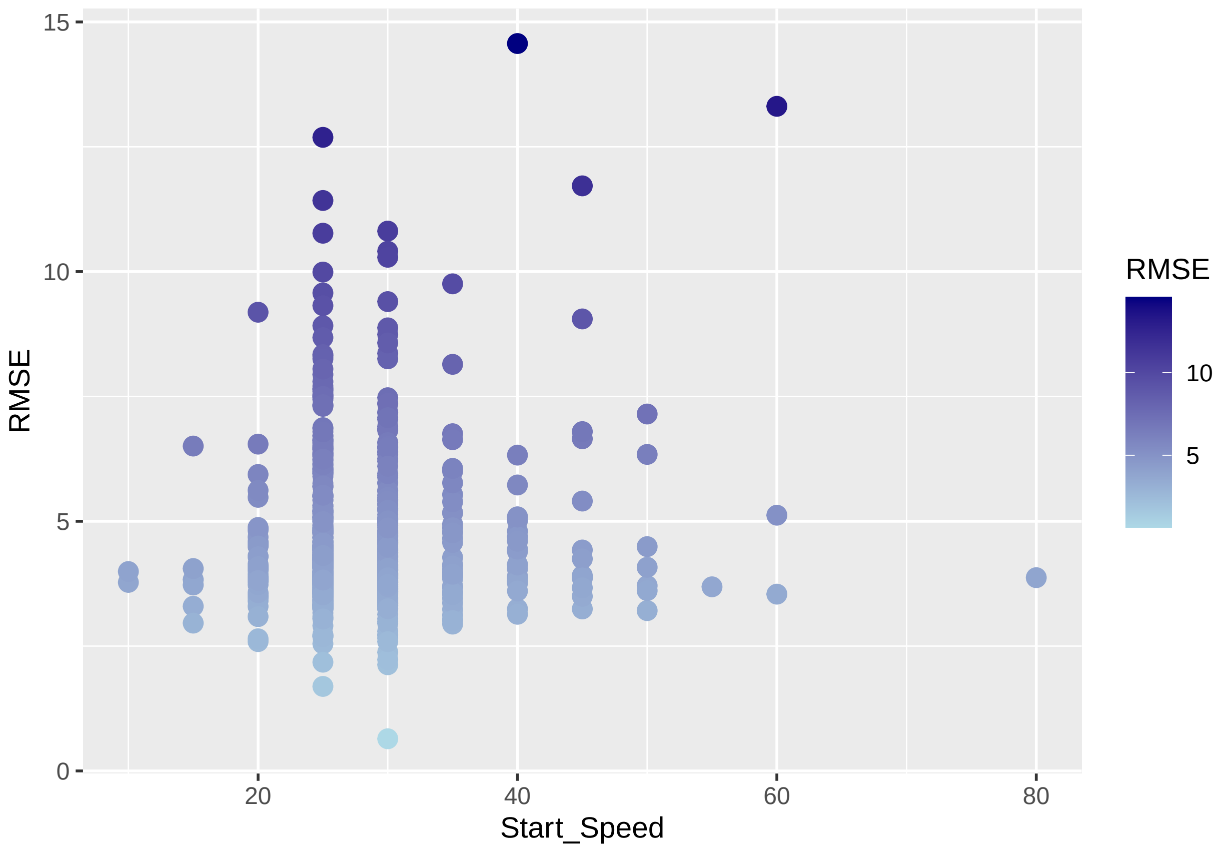


Relation between RMSE and the start location information.

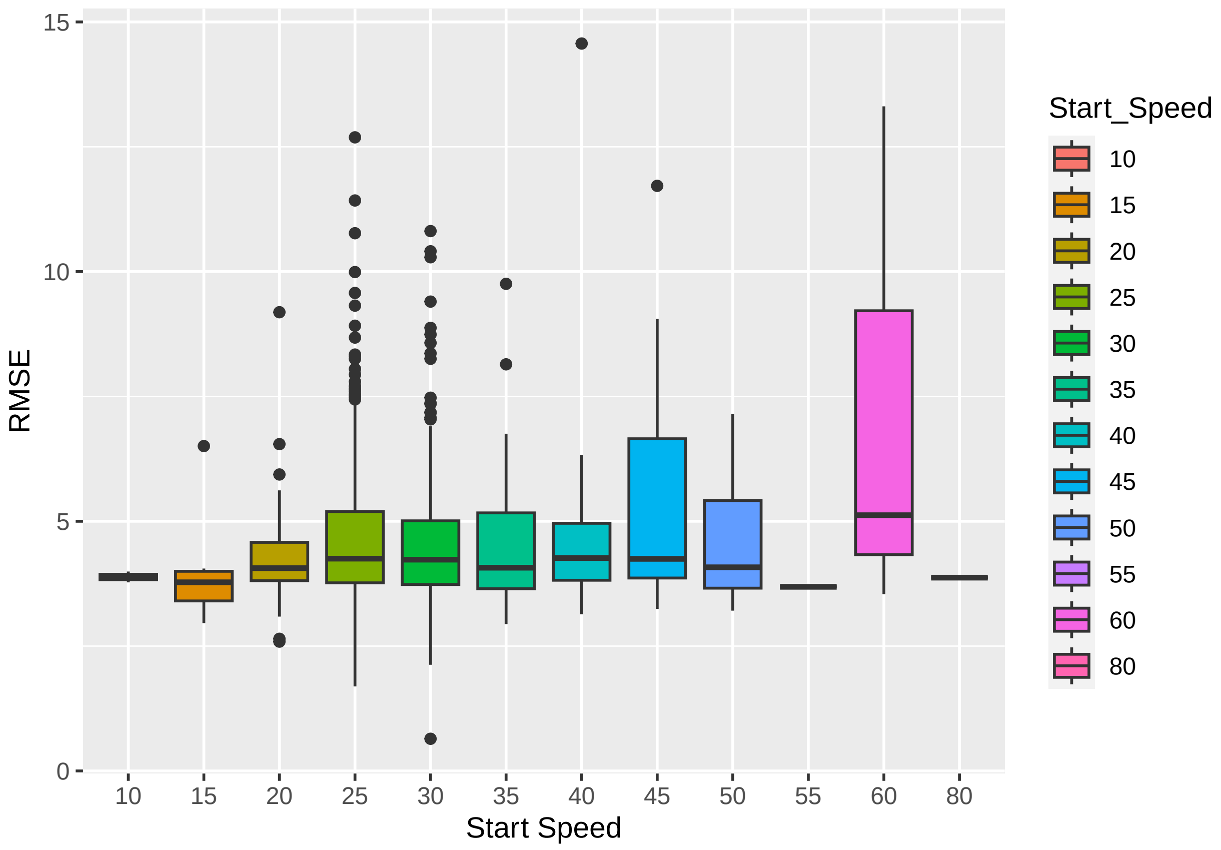
```
# distribution of RMSE between start information
ggplot(hurri_res, aes(x = Start_Lon, y = Start_Lat, color = RMSE)) +
  geom_point(size = 3) +
  scale_color_gradient(low = "#ADD8E6", high = "#000080") +
  labs(x = "Longitude", y = "Latitude", color = "RMSE")
```



```r
ggplot(hurri_res, aes(x = Start_Speed, y = RMSE, color = RMSE)) +
  geom_point(size = 3) +
  scale_color_gradient(low = "#ADD8E6", high = "#000080") +
  labs(x = "Start_Speed", y = "RMSE", color = "RMSE")
```

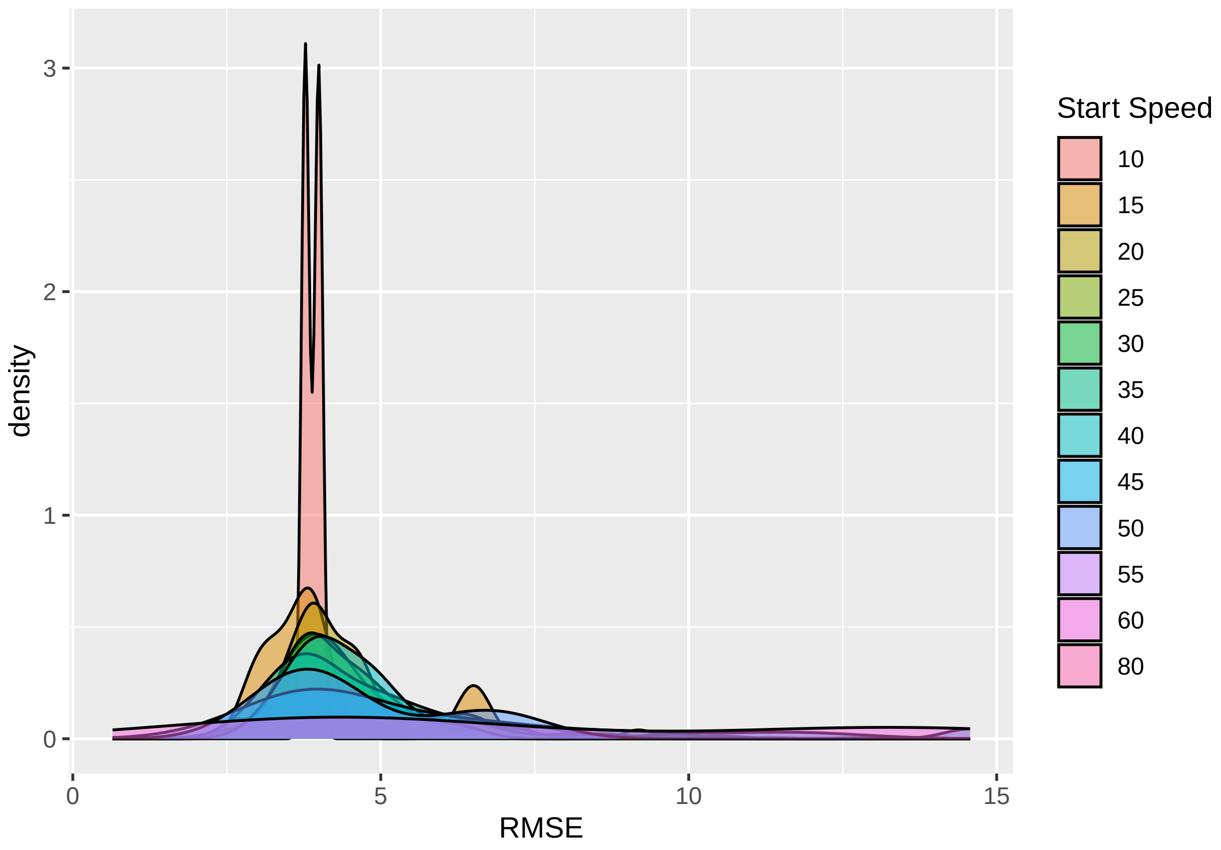


```
ggplot(hurri_res %>% mutate(Start_Speed = factor(Start_Speed)), aes(x = Start_Speed, y = RMSE, fill = St
  geom_boxplot() +
  labs(x = "Start Speed", y = "RMSE")
```



```
ggplot(hurri_res %>% mutate(Start_Speed = factor(Start_Speed)), aes(x = RMSE, fill = Start_Speed)) +
  geom_density(alpha = 0.5) +
  labs(x = "RMSE", fill = "Start Speed")
```

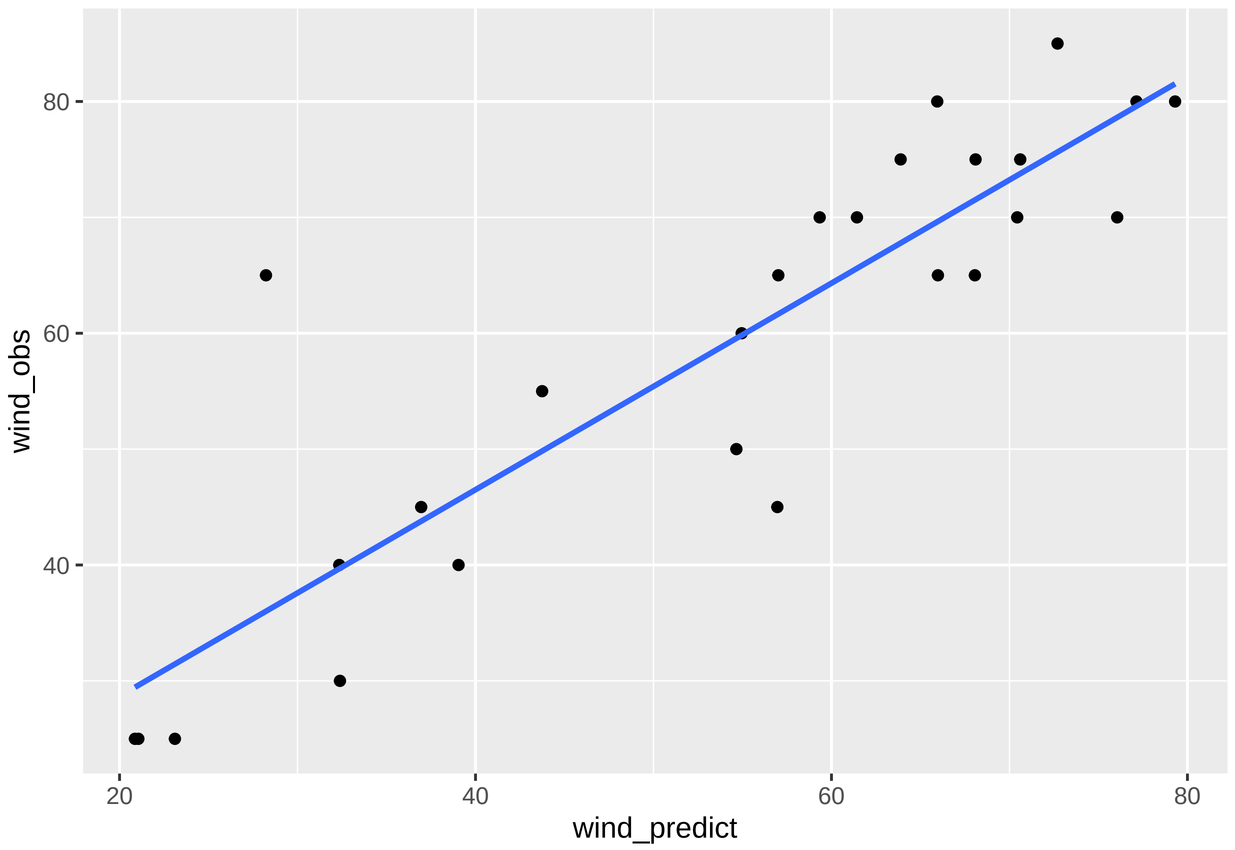
```
## Warning: Groups with fewer than two data points have been dropped.
## Groups with fewer than two data points have been dropped.

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



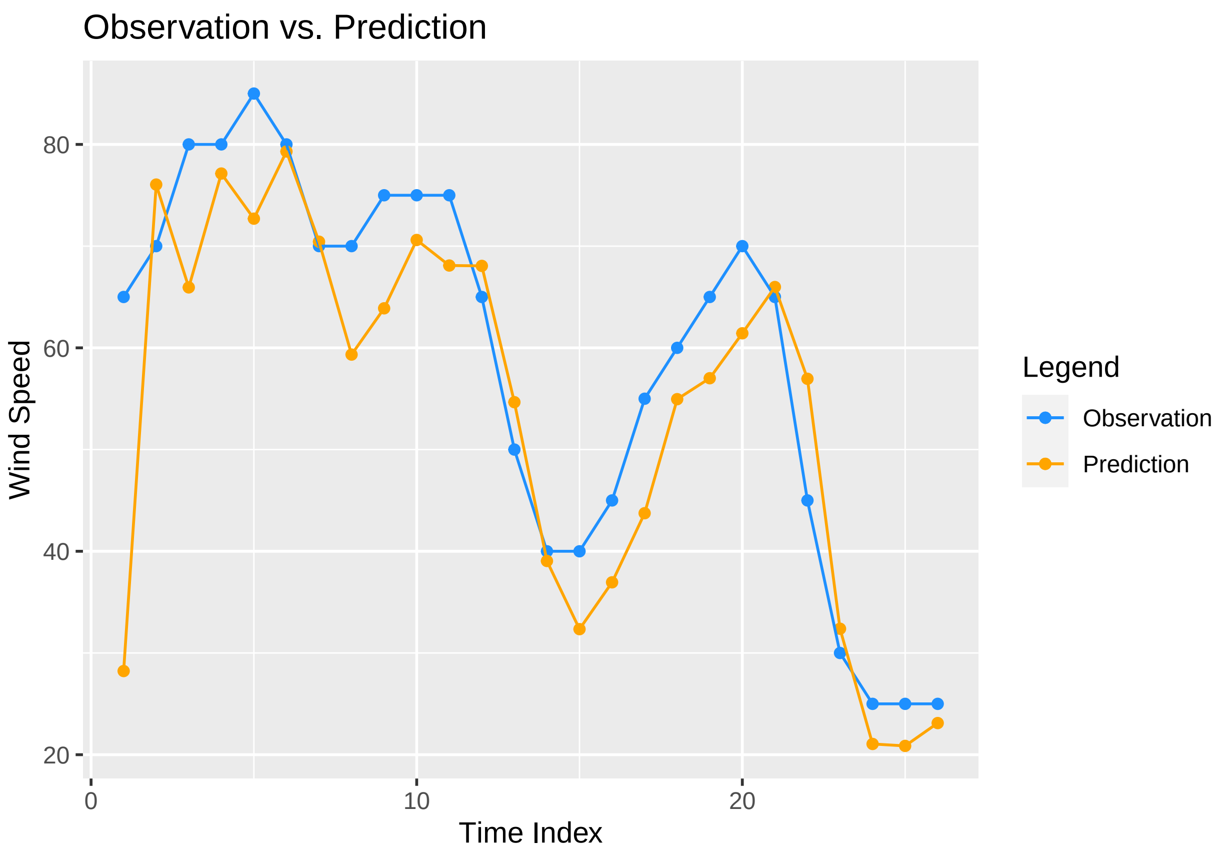
The prediction performance on some specific hurricanes.

```
example_hurri = as.data.frame(Y_table[[1]])
example_hurri$index = 1:nrow(example_hurri)

# example visualization of index 1 hurricane
ggplot(example_hurri, aes(x = wind_predict, y = wind_obs)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

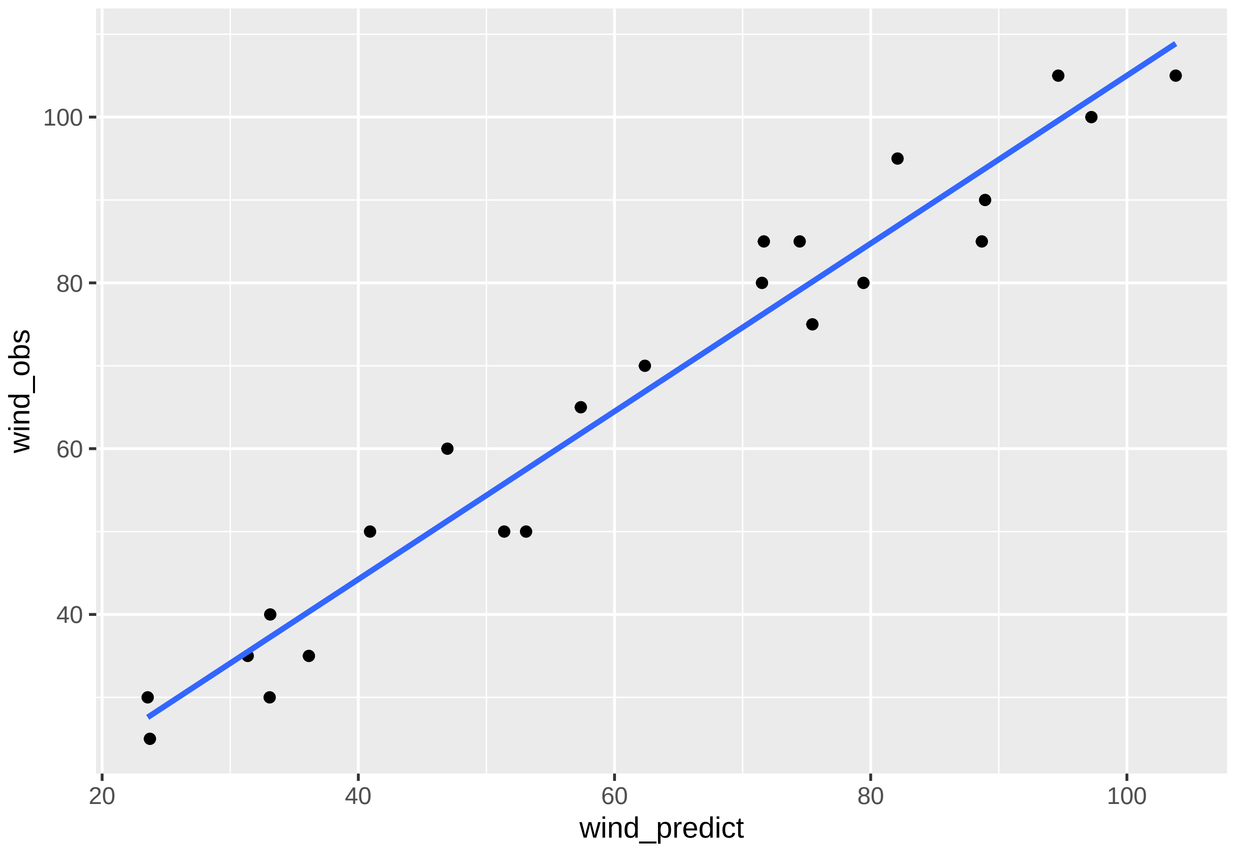


```r
ggplot(example_hurri, aes(x = index)) +
  geom_point(aes(y = wind_obs, color = "Observed")) +
  geom_point(aes(y = wind_predict, color = "Predicted")) +
  geom_line(aes(y = wind_obs, color = "Observed")) +
  geom_line(aes(y = wind_predict, color = "Predicted")) +
  labs(title = "Observation vs. Prediction", x = "Time Index", y = "Wind Speed") +
  scale_color_manual(name = "Legend",
                     values = c("Observed" = "#1E90FF", "Predicted" = "orange"),
                     labels = c("Observation", "Prediction"))
```

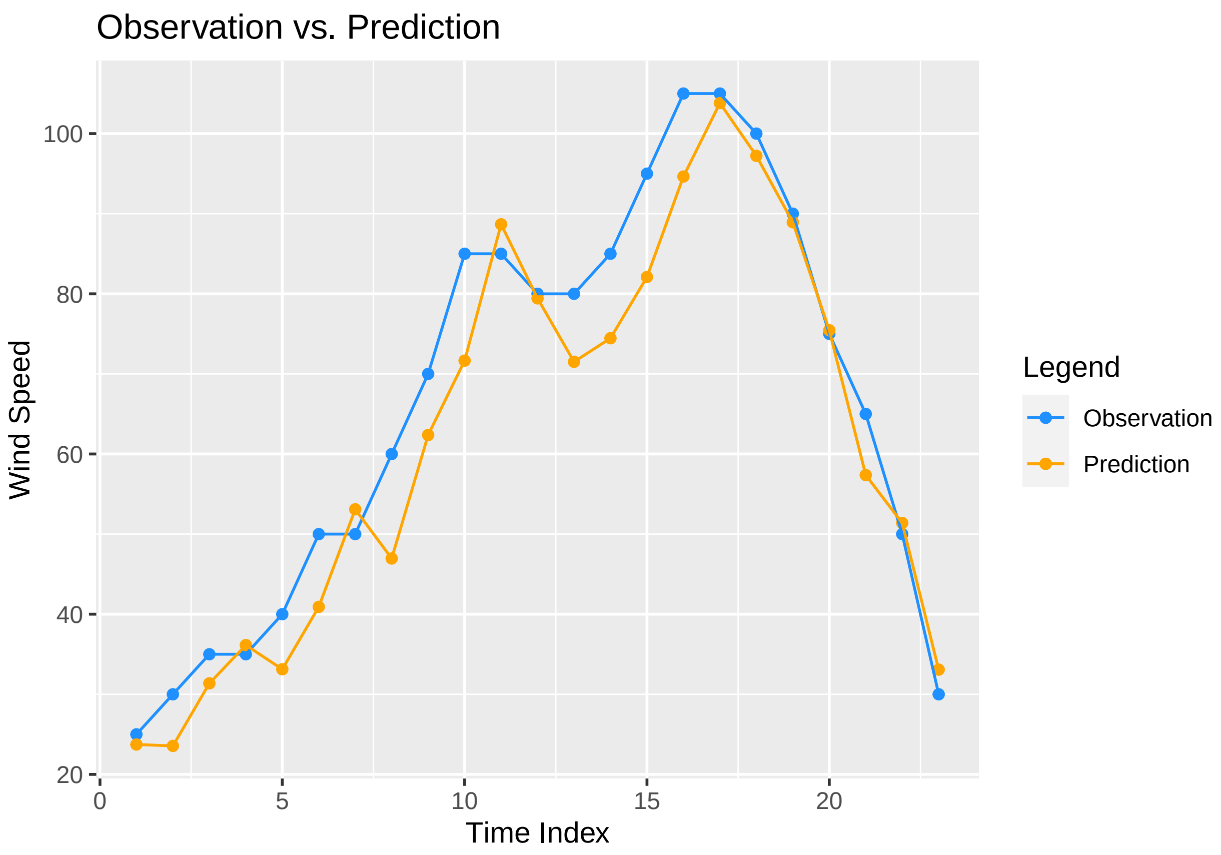


```
example_hurri = as.data.frame(Y_table[[15]])
example_hurri$index = 1:nrow(example_hurri)
# example visualization of index 2 hurricane
ggplot(example_hurri, aes(x = wind_predict, y = wind_obs)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
ggplot(example_hurri, aes(x = index)) +
  geom_point(aes(y = wind_obs, color = "Observed")) +
  geom_point(aes(y = wind_predict, color = "Predicted")) +
  geom_line(aes(y = wind_obs, color = "Observed")) +
  geom_line(aes(y = wind_predict, color = "Predicted")) +
  labs(title = "Observation vs. Prediction", x = "Time Index", y = "Wind Speed") +
  scale_color_manual(name = "Legend",
                     values = c("Observed" = "#1E90FF", "Predicted" = "orange"),
                     labels = c("Observation", "Prediction"))
```



Observation vs. Prediction
Wind Speed
Time Index
20
40
60
80
100
0
5
10
15
20
Legend
Observation
Prediction