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.

Prediction Functions

Import parameters from task 1 & 2.

```
# load parameters
beta_list = read.csv("./data/B_list_lastmean.csv")
gamma_list = read.csv("./data/gamma_list.csv")
```

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

```
Speed_Prediction = function(beta, gamma, burn_bindex, burn_times){
  # final parameters to be used
  # the rows of beta_sample means the last 5000, 4000, 3000, 2000 and 1000
  # burn in the MC chains. change this based on the resulting plots
  # if burn-in is set to 8000, then we set burn_bindex as 4 to pick the 4th row of beta_sample
  # index is the useful samples (used for estimates & CIs)
  para_beta = beta[burn_bindex,]
  para_beta = as.matrix(para_beta)
  index = (burn_times + 1):10000
  gamma_sample = gamma[index, ]
  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)
  }

  # 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)
    }
  }
  return(Y_table)
}
```

```
Y_table_set = Speed_Prediction(beta = beta_list, gamma = gamma_list, burn_bindex = 4, burn_times = 8000)

Visual_Table = function(Y_table_input){
  Y_table = Y_table_input
  hurri_res = data.frame(ID = "example",
                         RMSE = 0,
                         R_squared = 0)
  for (i in 1:length(names(Z))) {
    # calculate
    RMSE = sqrt(mean((Y_table[[i]][,1] - Y_table[[i]][,2])^2))
    # calculate R^2
    y = Y_table[[i]][,1]
    y_hat = Y_table[[i]][,2]
    mean_y = mean(y)
    SSR = sum((y_hat - y)^2)
    SST = sum((y - mean_y)^2)
    R_squared = 1 - SSR/SST
    new_row = c(names(Z)[i], RMSE, R_squared)
    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")

  hurri_res$R_squared = as.numeric(hurri_res$R_squared)

  return(hurri_res)
```

```
}
```

Visualization Analysis

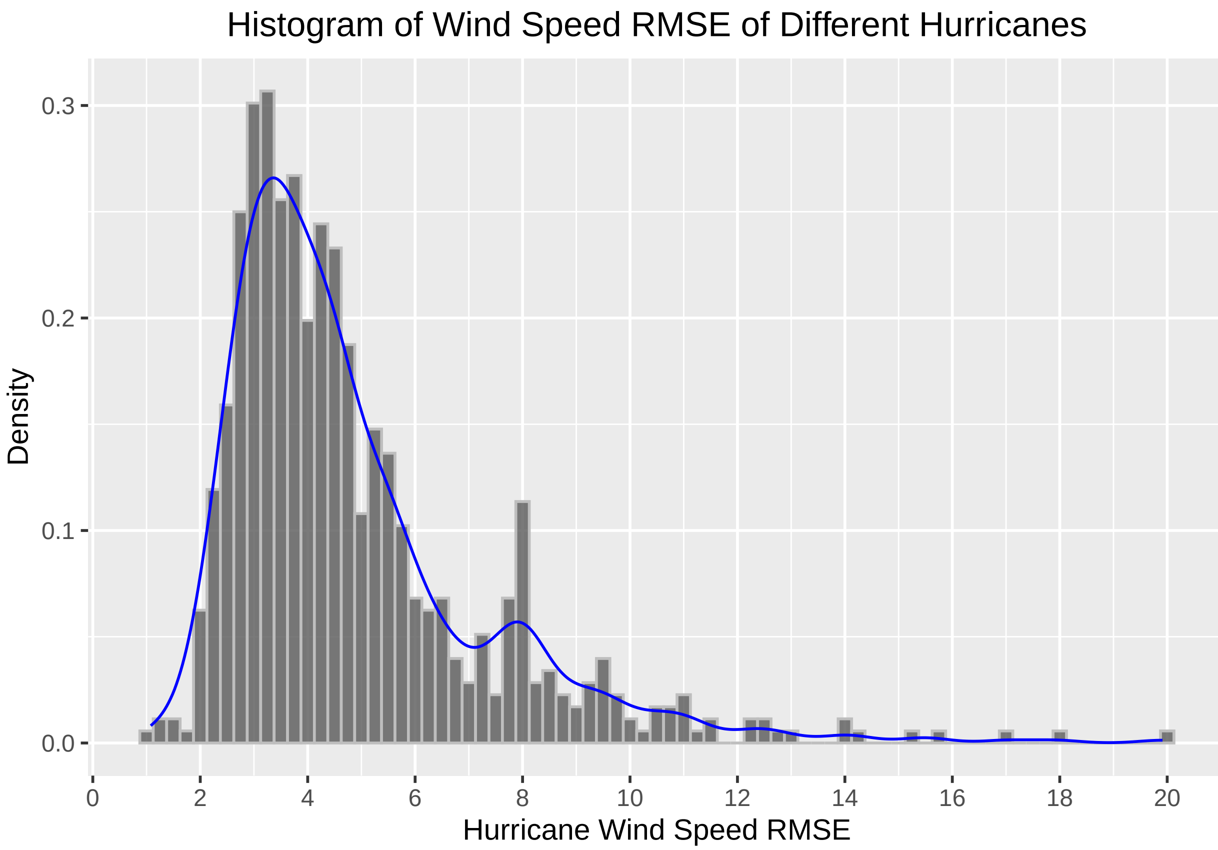
Summary Table

```
hurri_res = Visual_Table(Y_table_set)

hurri_res_brief = hurri_res %>%
  select(c(1:3)) %>%
  head(15)

write.csv(hurri_res_brief, "data/prediction_brief.csv", row.names = FALSE)
write.csv(hurri_res, "data/prediction_all.csv", row.names = FALSE)

hurri_res_brief %>%
  knitr::kable(digits = 4)
```

ID	RMSE	R_squared
ABBY.1960	8.8804	0.7700
ABBY.1964	9.6430	0.3033
ABBY.1968	3.5043	0.9360
ABLE.1950	3.6755	0.9813
ABLE.1951	3.4802	0.9767
ABLE.1952	4.5183	0.9583
AGNES.1972	5.2483	0.8881
ALBERTO.1982	8.0473	0.7499
ALBERTO.1988	2.6121	0.7420
ALBERTO.1994	4.3941	0.8807
ALBERTO.2000	3.7896	0.9625
ALBERTO.2006	4.3591	0.7882
ALBERTO.2012	3.2193	0.8036
ALEX.1998	2.9351	0.7289
ALEX.2004	5.4552	0.9539

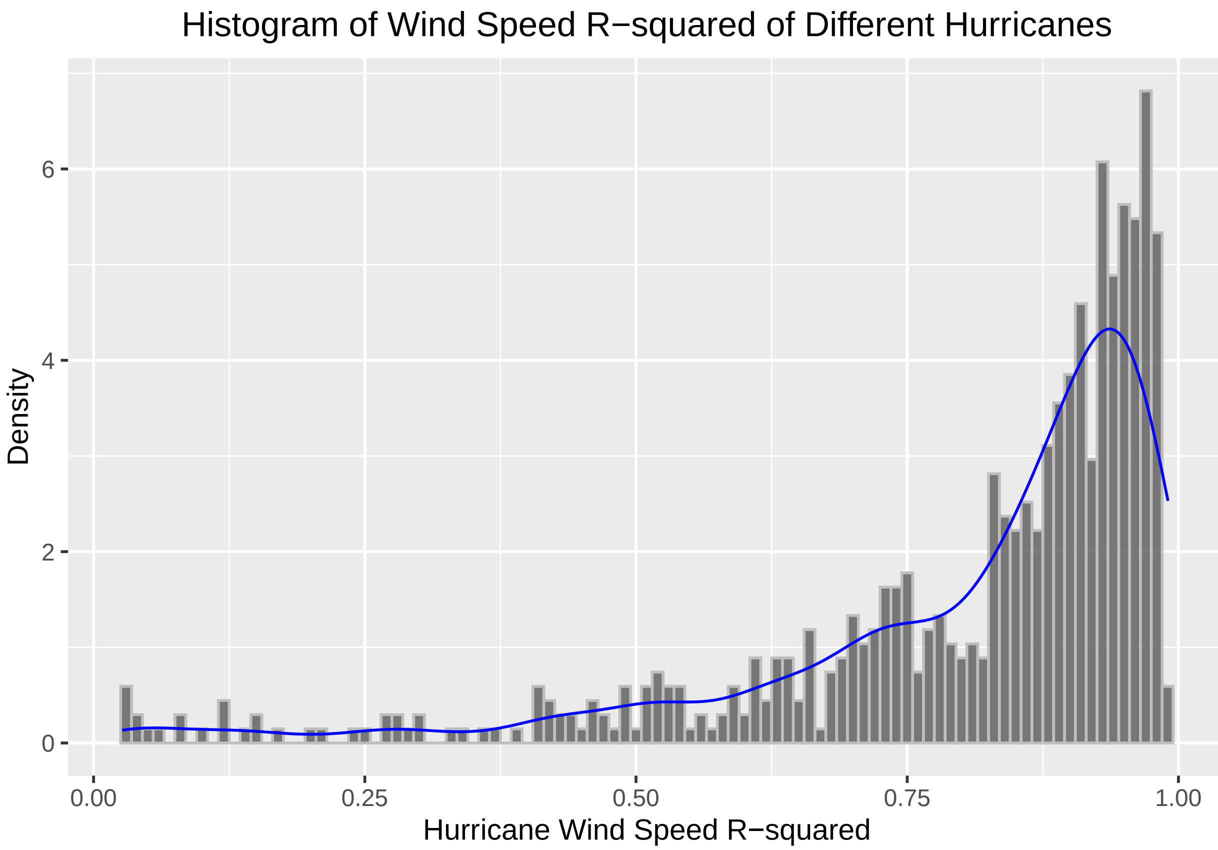
Overall RMSE prediction performance.

```
# overall density of RMSE
ggplot(hurri_res, aes(x = RMSE)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 0.25, alpha = 0.8, color = "grey") +
  geom_density(size = 0.5, color = "blue", lty = 1) +
  scale_x_continuous(breaks = seq(0, 20, by = 2)) +
  labs(title = "Histogram of Wind Speed RMSE of Different Hurricanes", x = "Hurricane Wind Speed RMSE",
  theme(plot.title = element_text(hjust = 0.5))
```



Overall R-squared performance.

```
# overall density of R-squared
ggplot(hurri_res %>% filter(0 <= R_squared & R_squared <= 1), aes(x = R_squared)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 0.01, alpha = 0.8, color = "grey") +
  geom_density(size = 0.5, color = "blue", lty = 1) +
  labs(title = "Histogram of Wind Speed R-squared of Different Hurricanes", x = "Hurricane Wind Speed R
  theme(plot.title = element_text(hjust = 0.5))
```

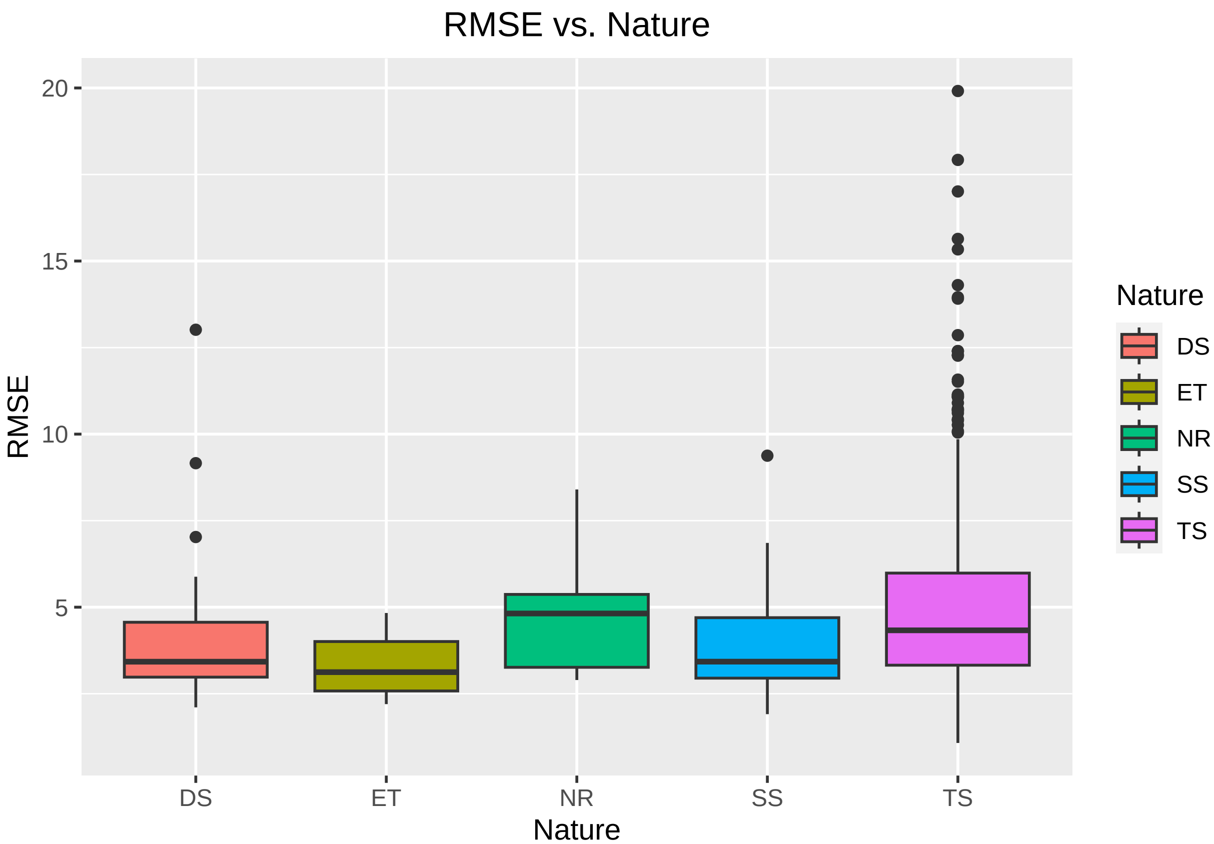


Difference of overall RMSE using different numbers of burn-in.

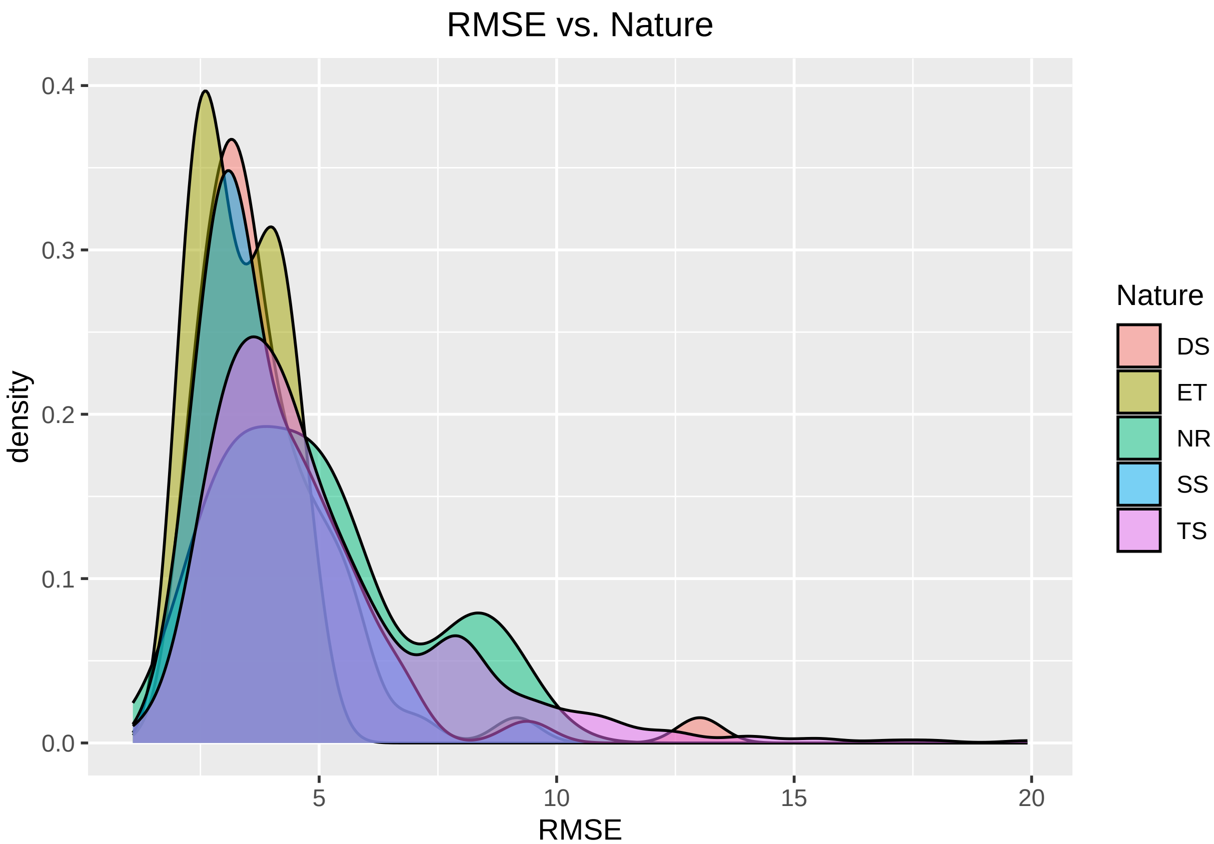
Result shows that there is not obvious difference here.

The difference of RMSE distribution of difference properties.

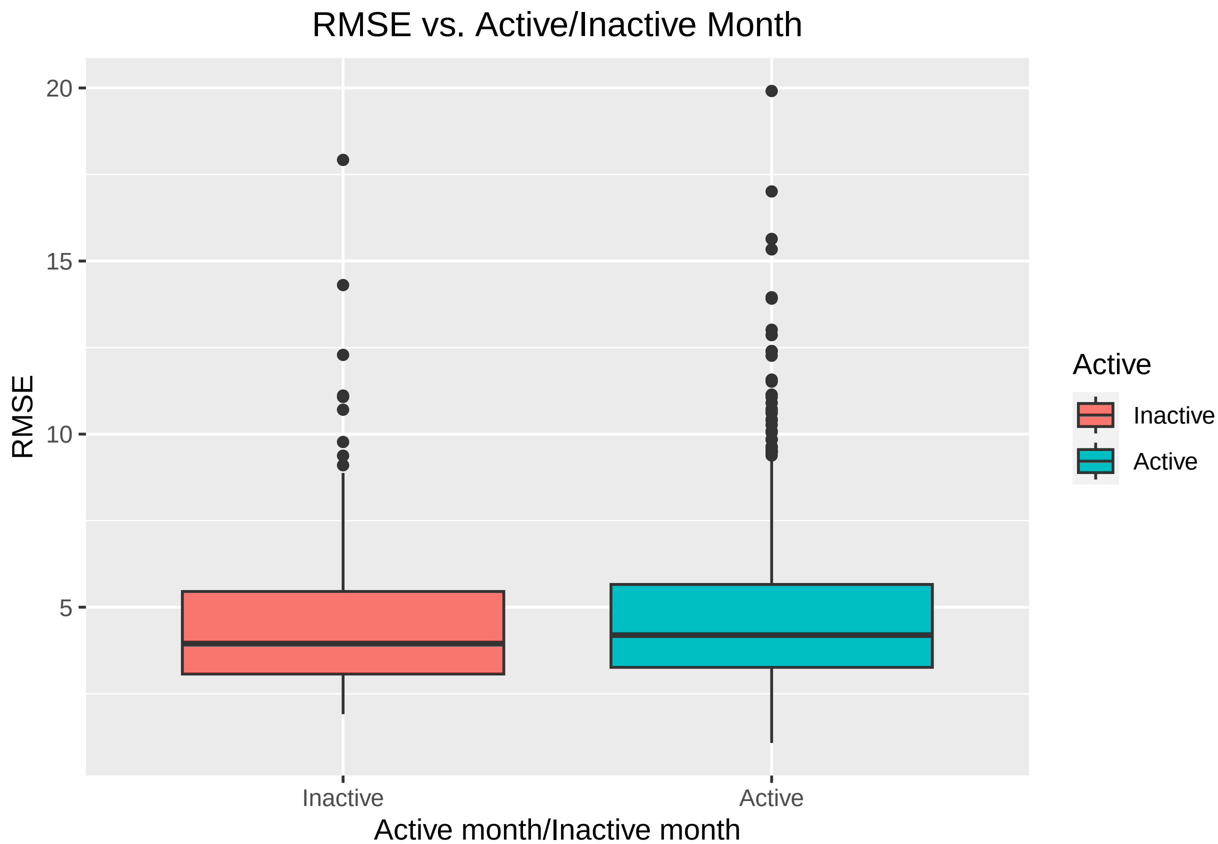
```
# distribution of RMSE on hurricane nature
ggplot(hurri_res, aes(x = Nature, y = RMSE, fill = Nature)) +
  geom_boxplot() +
  scale_color_manual(values = c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2")) +
  labs(title = "RMSE vs. Nature", x = "Nature", y = "RMSE") +
  theme(plot.title = element_text(hjust = 0.5))
```



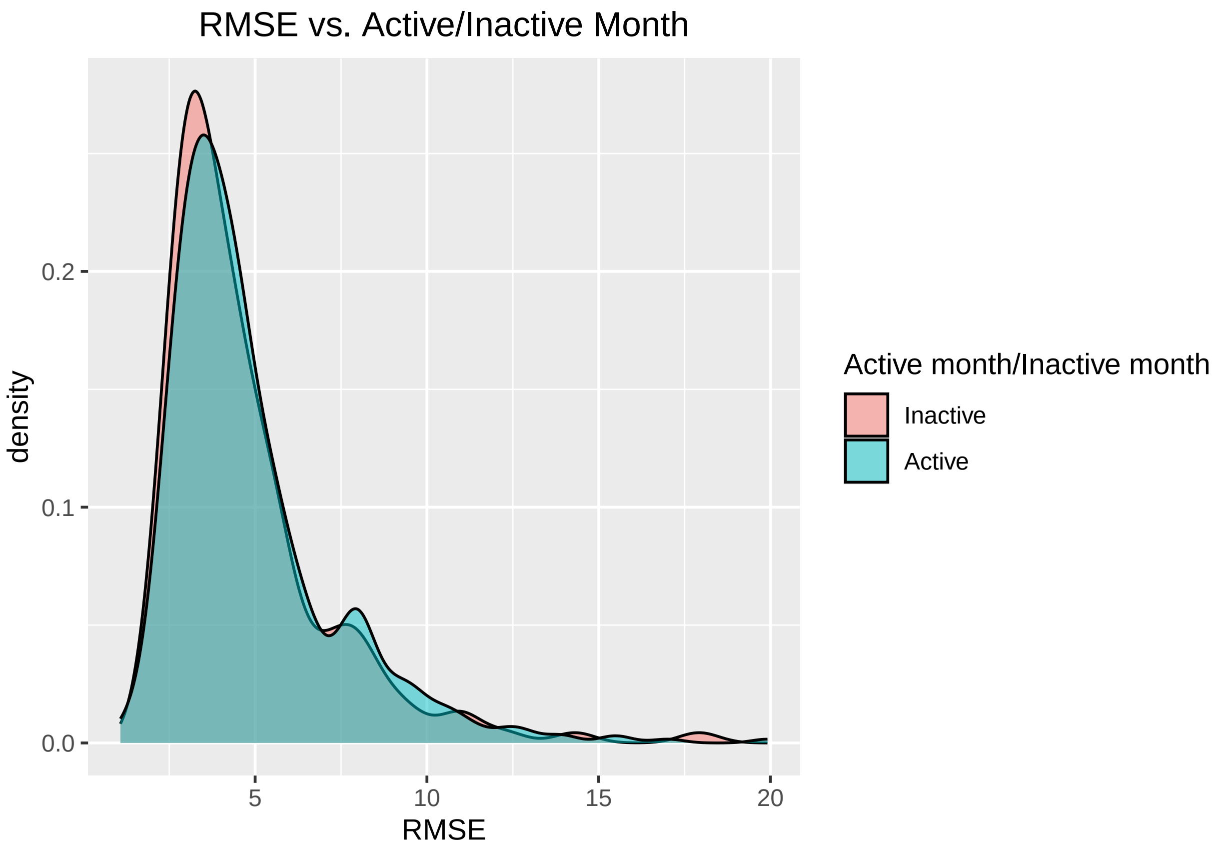

```
ggplot(hurri_res, aes(x = RMSE, fill = Nature)) +
  geom_density(alpha = 0.5) +
  labs(title = "RMSE vs. Nature", x = "RMSE", fill = "Nature") +
  theme(plot.title = element_text(hjust = 0.5))
```



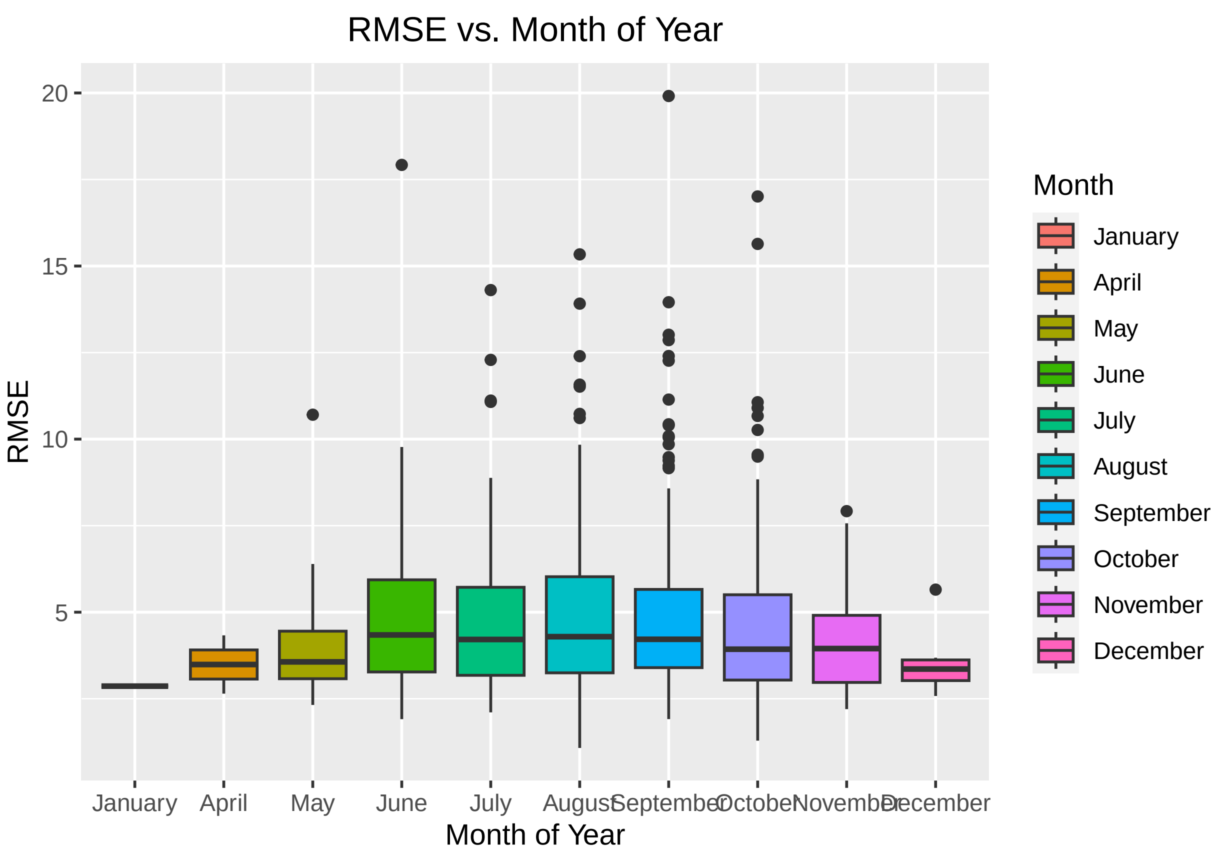

```r
# distribution of RMSE on hurricane active months
ggplot(hurri_res, aes(x = Active, y = RMSE, fill = Active)) +
  geom_boxplot() +
  labs(title = "RMSE vs. Active/Inactive Month", x = "Active month/Inactive month", y = "RMSE") +
  theme(plot.title = element_text(hjust = 0.5))
```



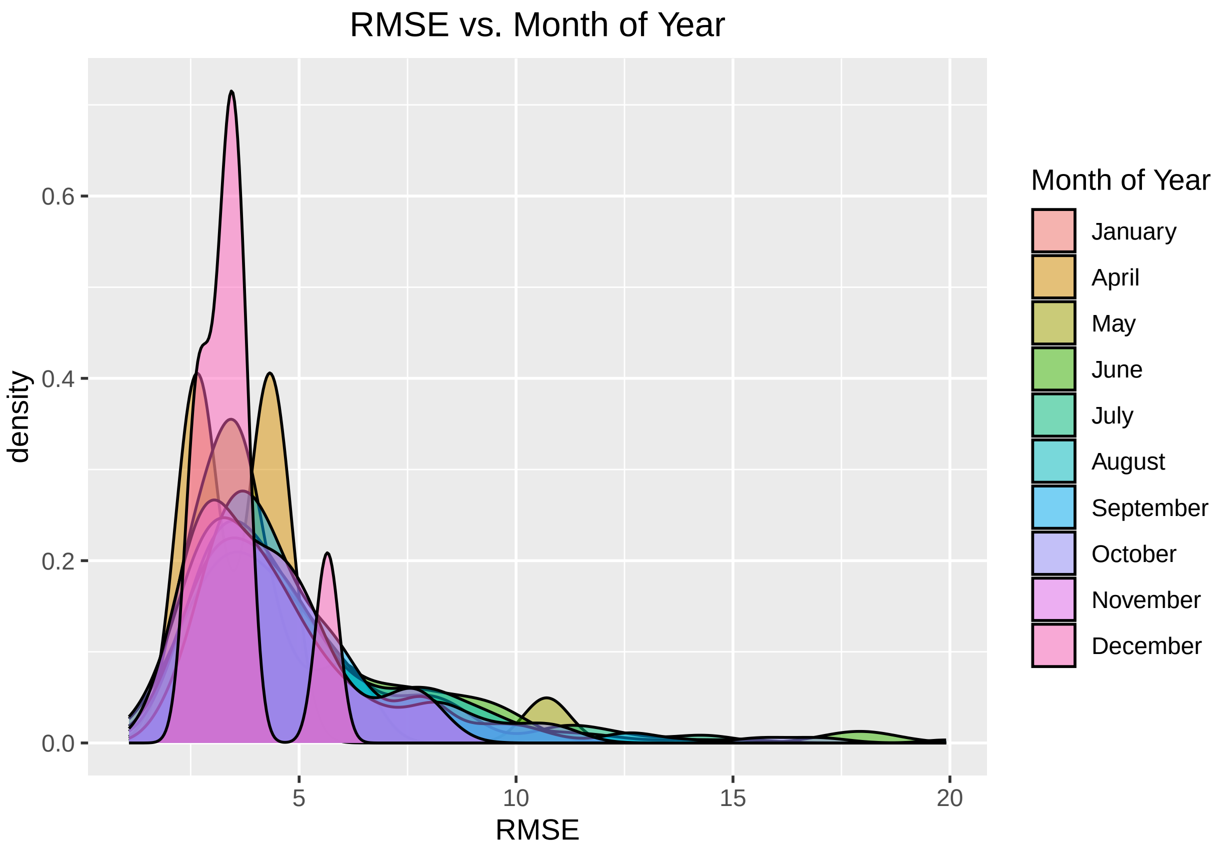

```
ggplot(hurri_res, aes(x = RMSE, fill = Active)) +
  geom_density(alpha = 0.5) +
  labs(title = "RMSE vs. Active/Inactive Month", x = "RMSE", fill = "Active month/Inactive month") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
ggplot(hurri_res, aes(x = Month, y = RMSE, fill = Month)) +
  geom_boxplot() +
  labs(title = "RMSE vs. Month of Year", x = "Month of Year", y = "RMSE") +
  theme(plot.title = element_text(hjust = 0.5))
```

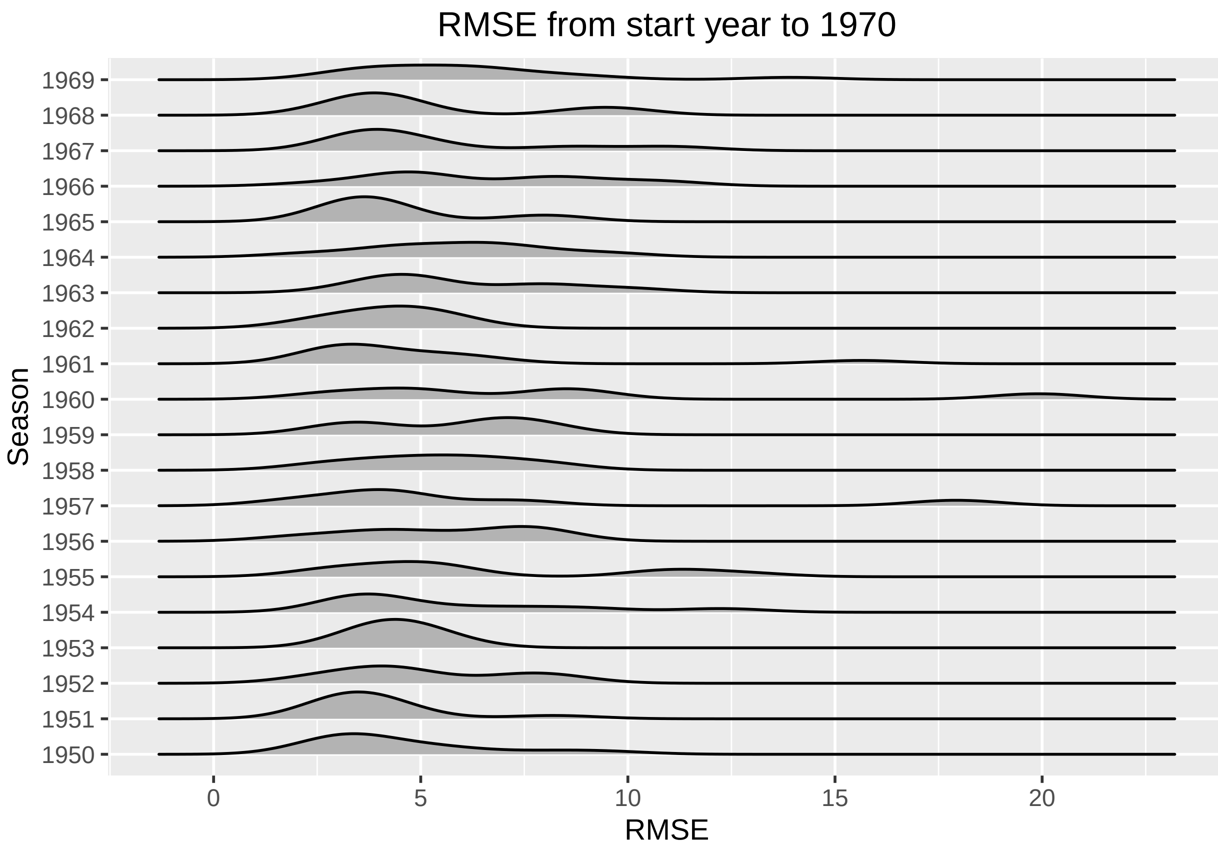



```
ggplot(hurri_res, aes(x = RMSE, fill = Month)) +
  geom_density(alpha = 0.5) +
  labs(title = "RMSE vs. Month of Year",x = "RMSE", fill = "Month of Year") +
  theme(plot.title = element_text(hjust = 0.5))
```



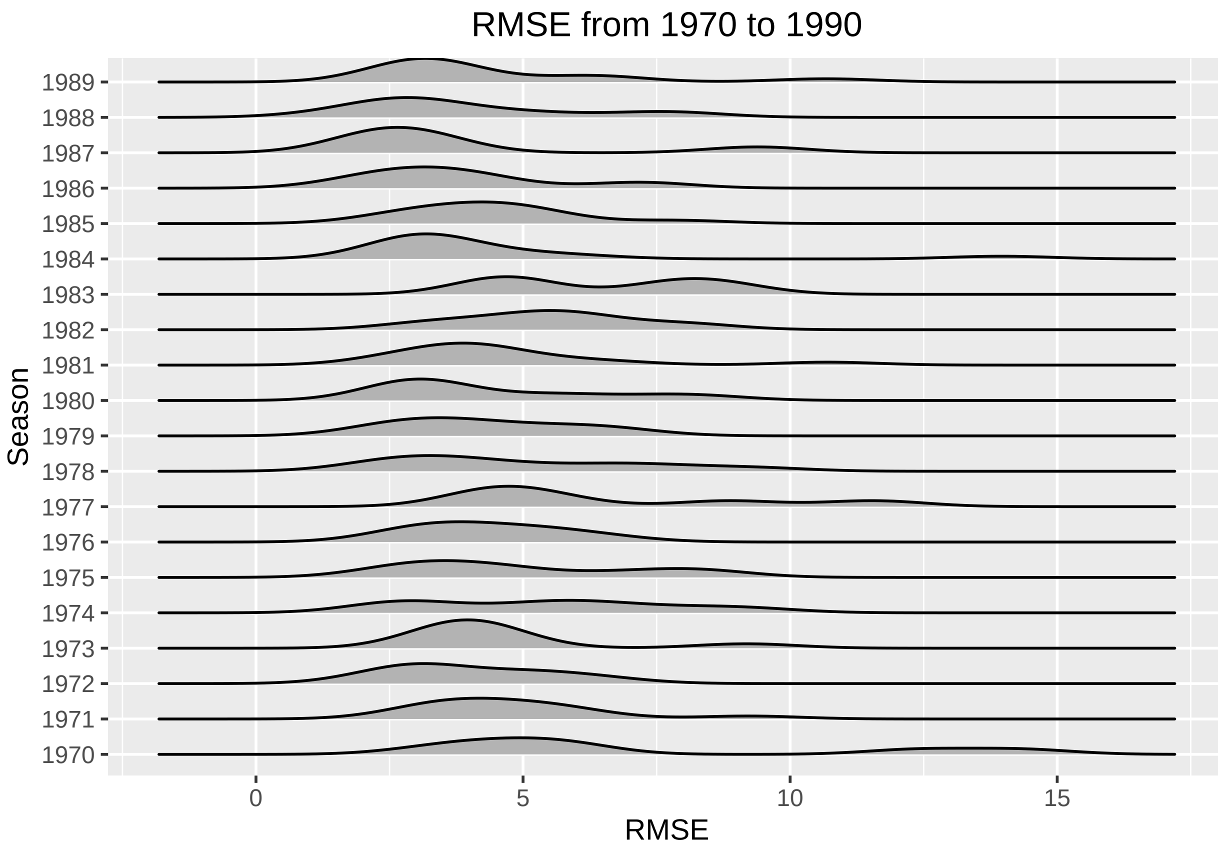

```
# distribution of RMSE on hurricane season
ggplot(hurri_res %>% filter(Season < 1970) %>% mutate(Season = factor(Season)), aes(x = RMSE, y = Seaso
  geom_density_ridges(scale = 0.8) +
  labs(title = "RMSE from start year to 1970",x = "RMSE", y = "Season") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Picking joint bandwidth of 1.1
```



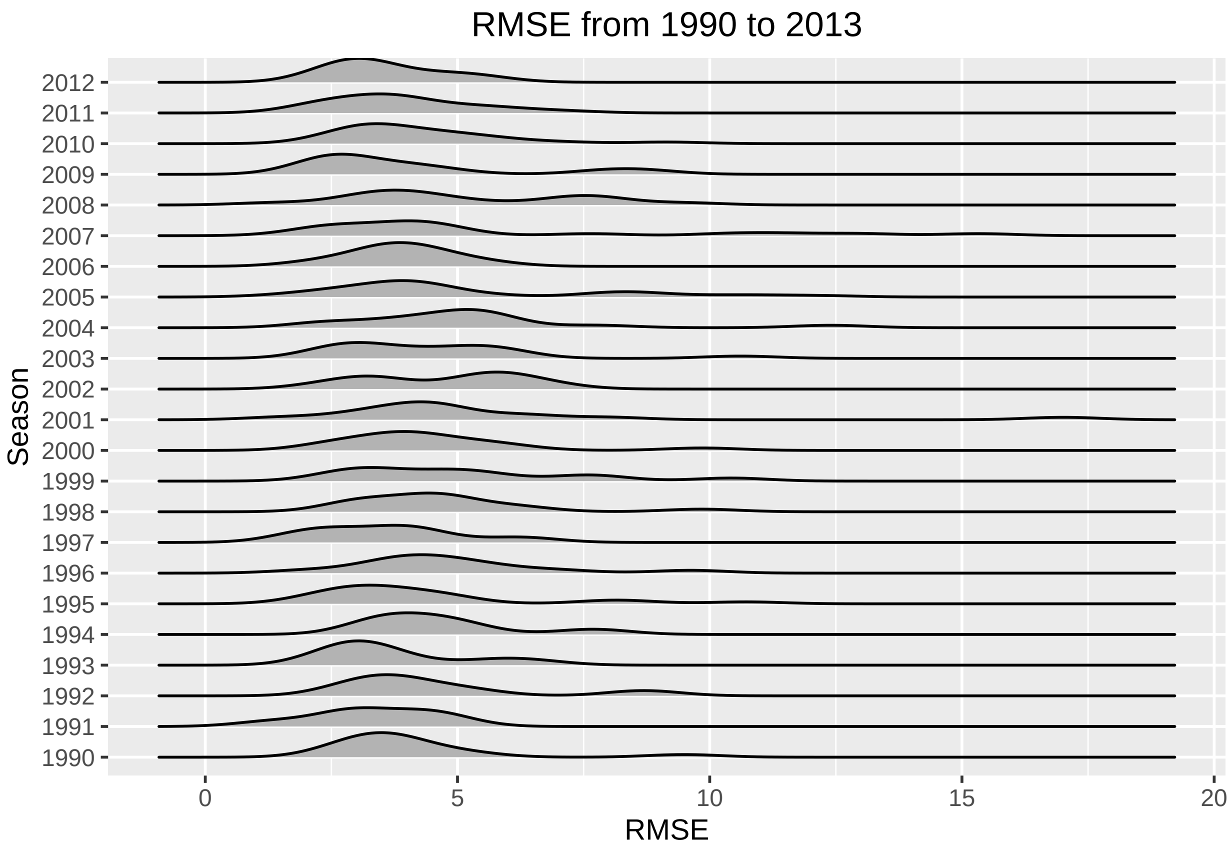

```
ggplot(hurri_res %>% filter(Season >= 1970 & Season < 1990) %>% mutate(Season = factor(Season)), aes(x =
  geom_density_ridges(scale = 0.8) +
  labs(title = "RMSE from 1970 to 1990", x = "RMSE", y = "Season") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Picking joint bandwidth of 0.965
```




```
ggplot(hurri_res %>% filter(Season >= 1990 & Season < 2013) %>% mutate(Season = factor(Season)), aes(x =
  geom_density_ridges(scale = 0.8) +
  labs(title = "RMSE from 1990 to 2013", x = "RMSE", y = "Season") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Picking joint bandwidth of 0.736
```



Relation between RMSE and the start location information.

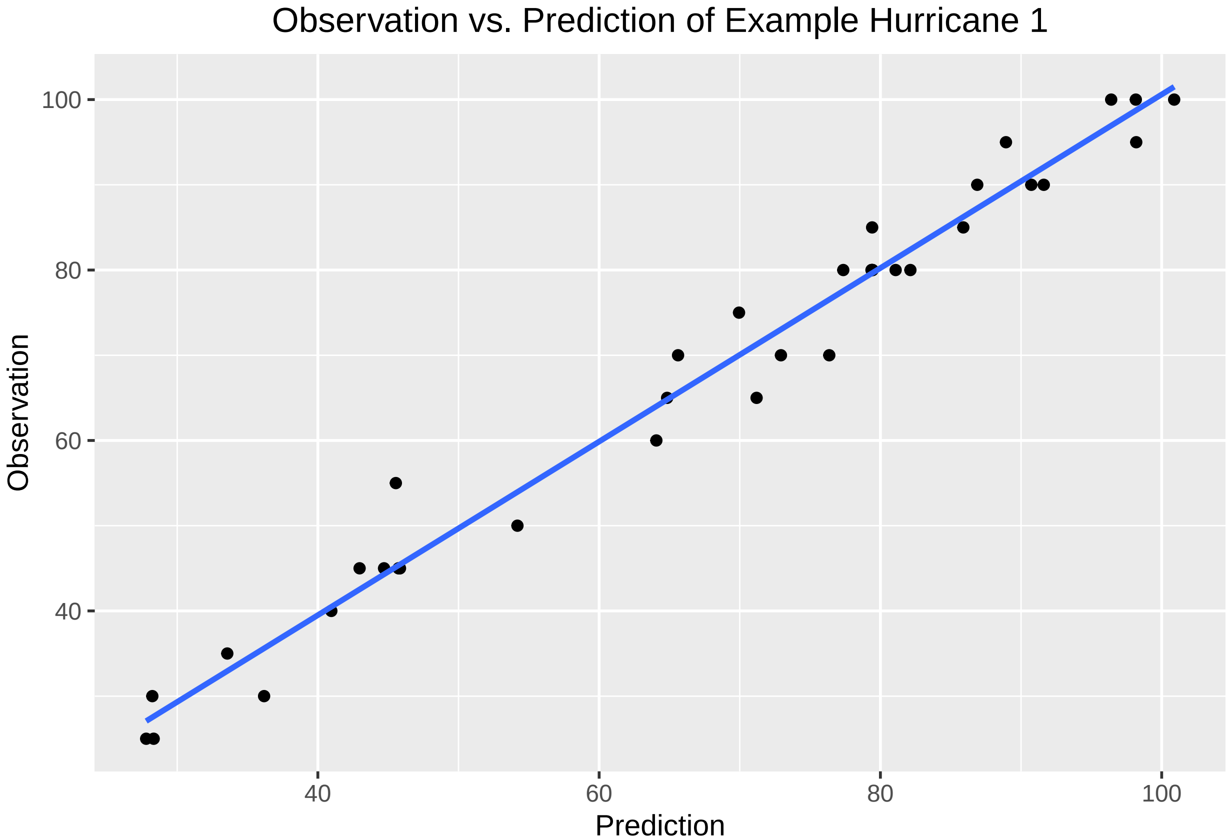
(not finally included since there is not much useful information in there)

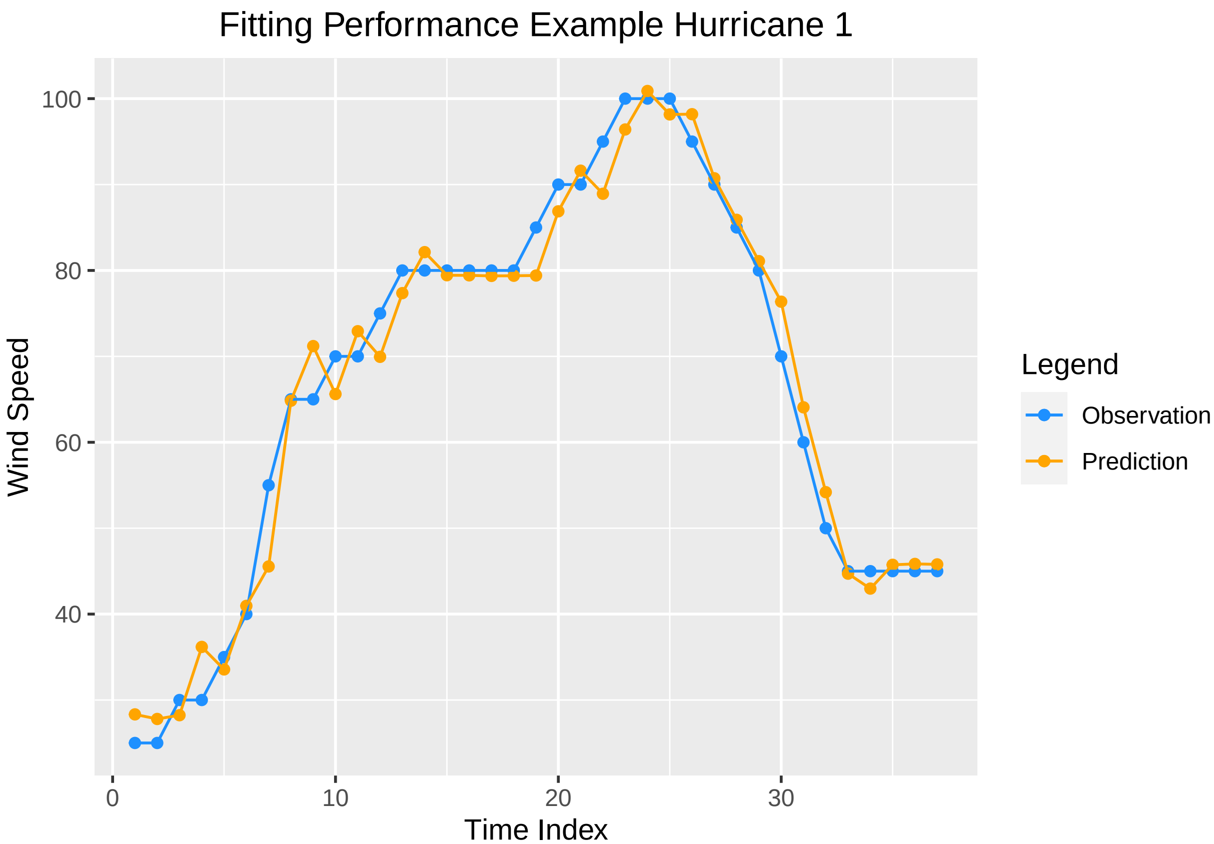
Prediction performance on some specific example hurricanes.

```
for (i in 1:5) {
  example_hurri = as.data.frame(Y_table_set[[i*5]])
  example_hurri$index = 1:nrow(example_hurri)

  # example visualization of example hurricanes
  graph_a = ggplot(example_hurri, aes(x = wind_predict, y = wind_obs)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    labs(title = paste("Observation vs. Prediction of", "Example Hurricane", i), x = "Prediction", y =
    theme(plot.title = element_text(hjust = 0.5))

  graph_b = 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 = paste("Fitting Performance", "Example Hurricane", i), x = "Time Index", y = "Wind Spee
    scale_color_manual(name = "Legend",
                       values = c("Observed" = "#1E90FF", "Predicted" = "orange"),
                       labels = c("Observation", "Prediction")) +
    theme(plot.title = element_text(hjust = 0.5))
```

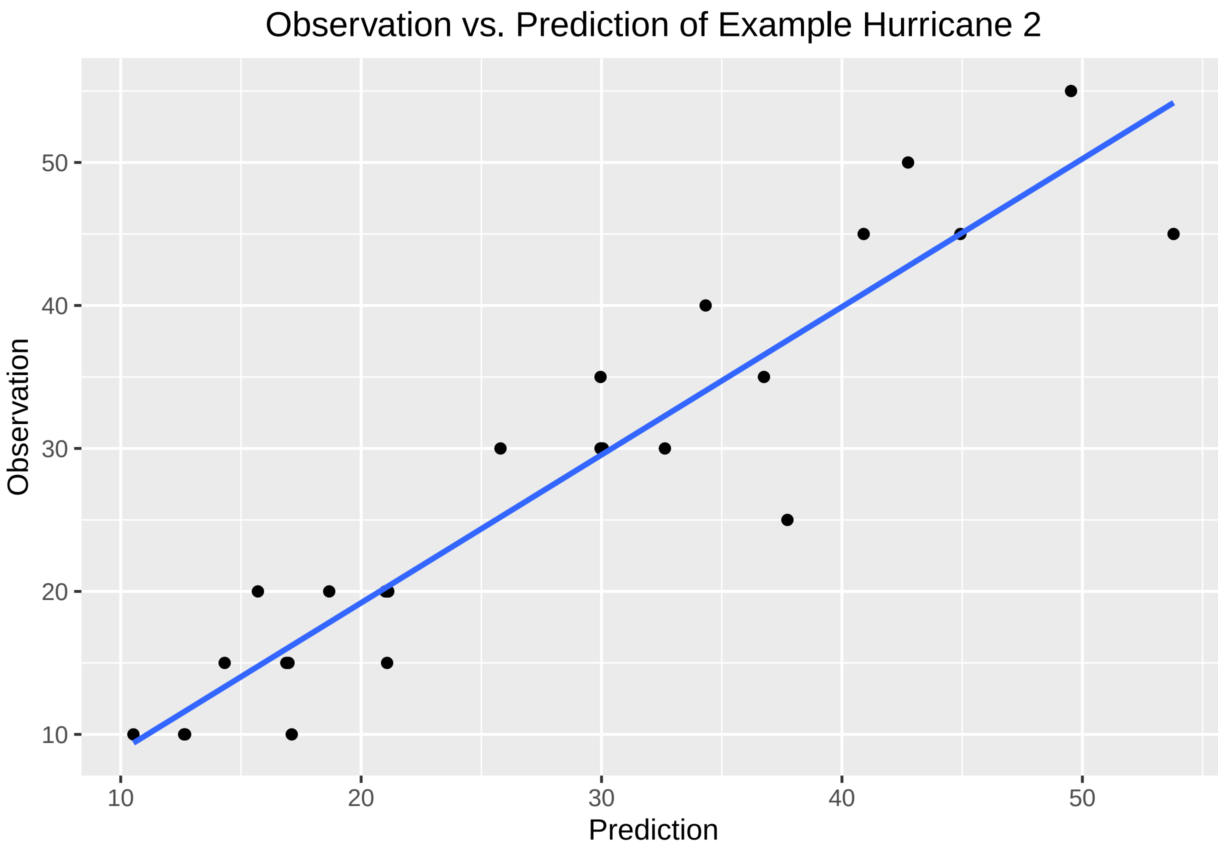
```
  print(graph_a)
  print(graph_b)
}
```

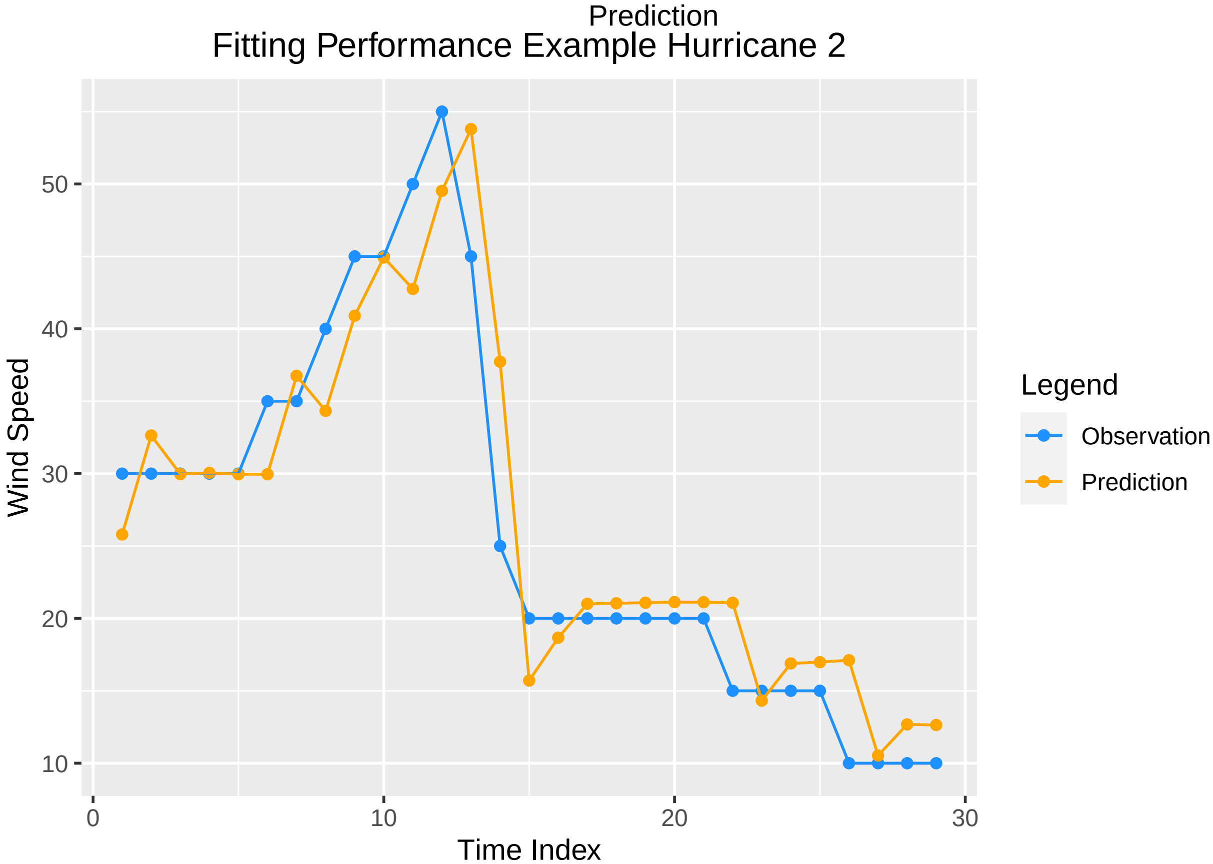
```
## `geom_smooth()` using formula = 'y ~ x'
```



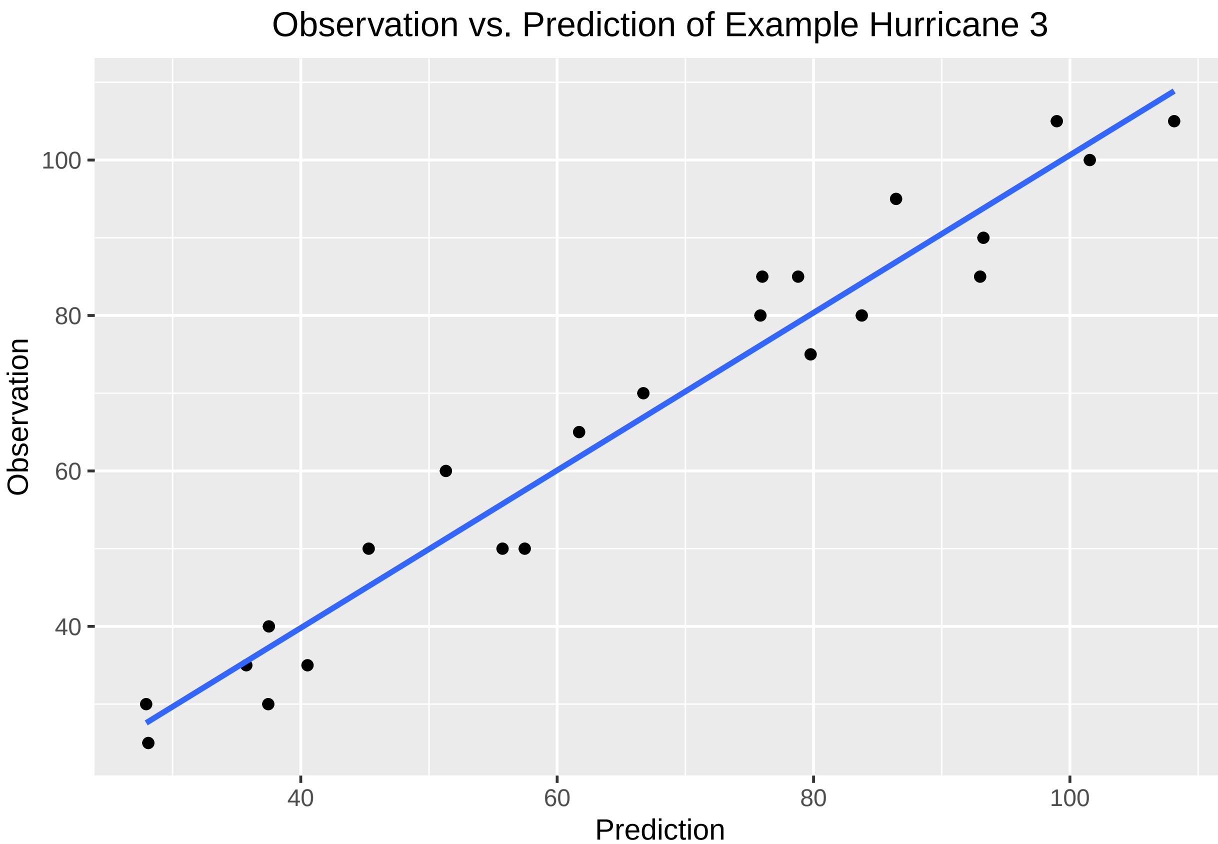


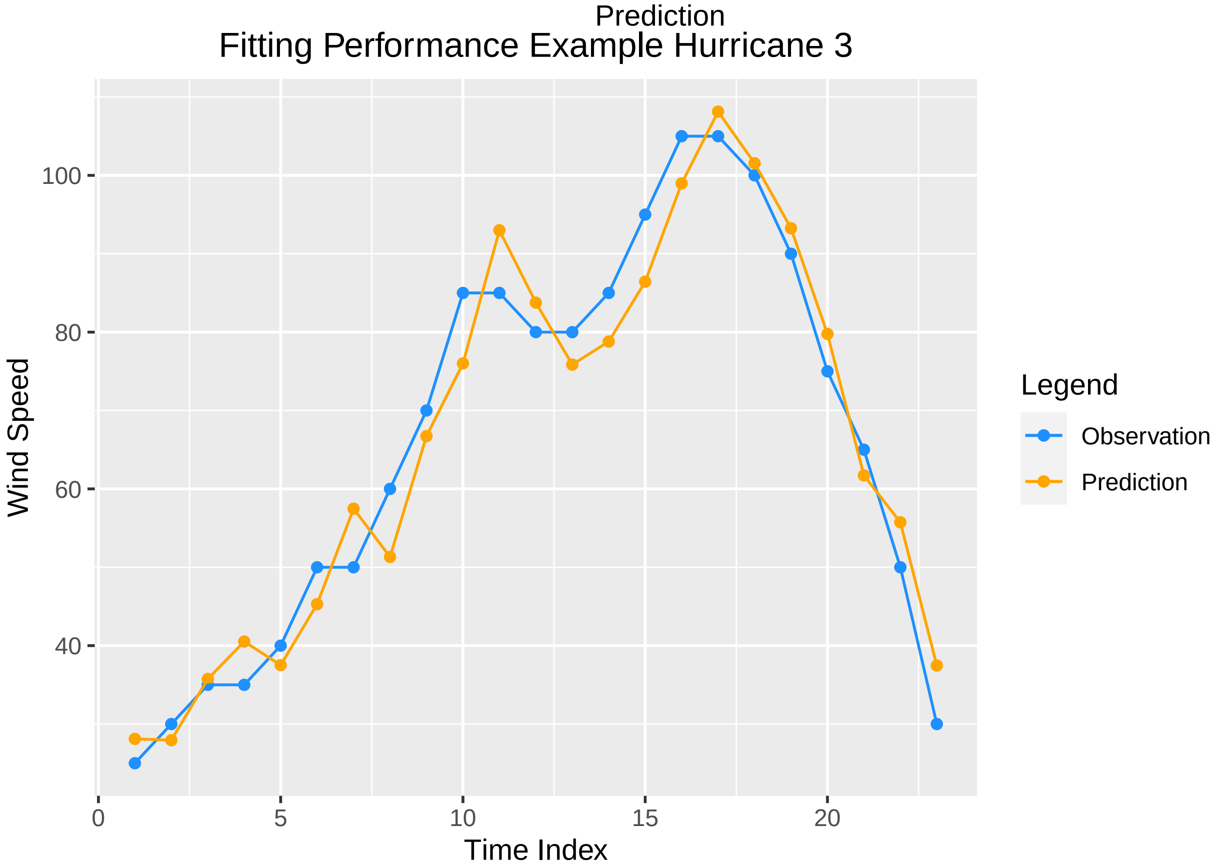

```
## `geom_smooth()` using formula = 'y ~ x'
```



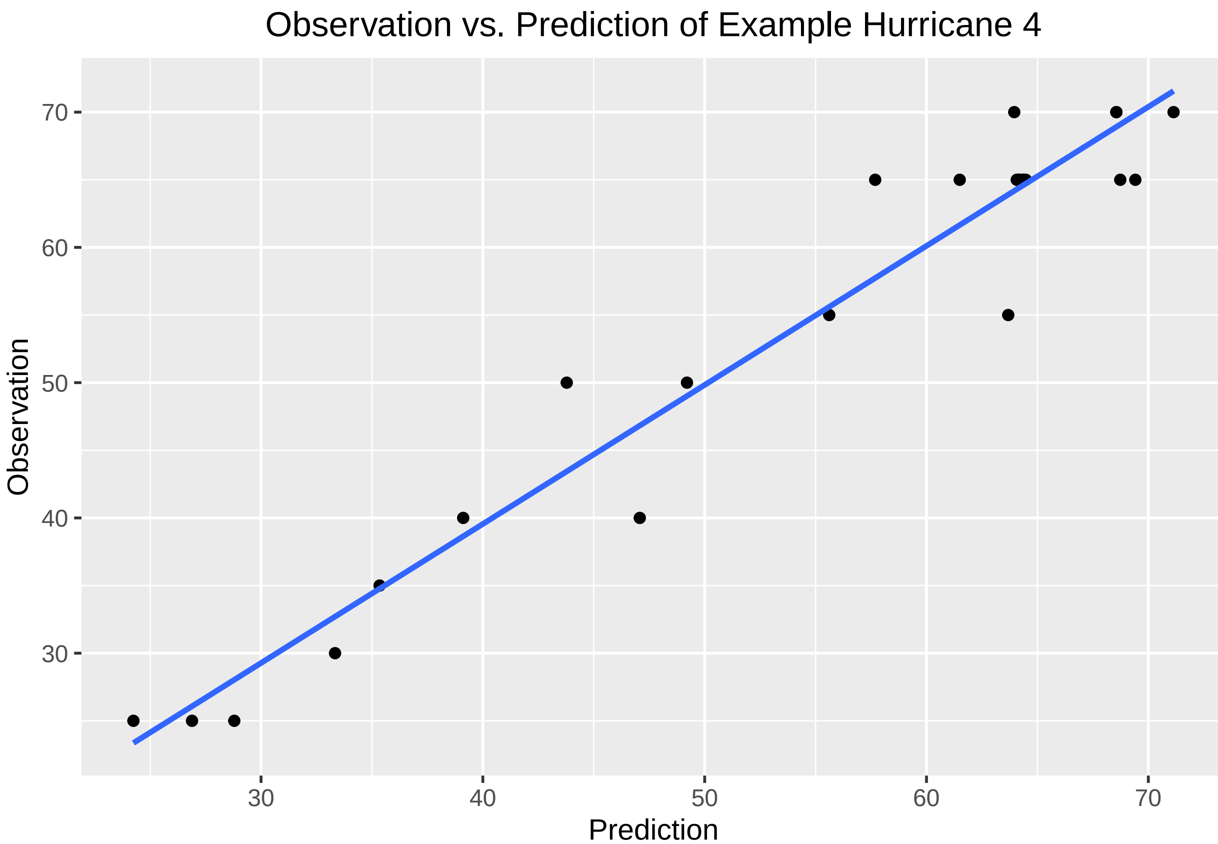


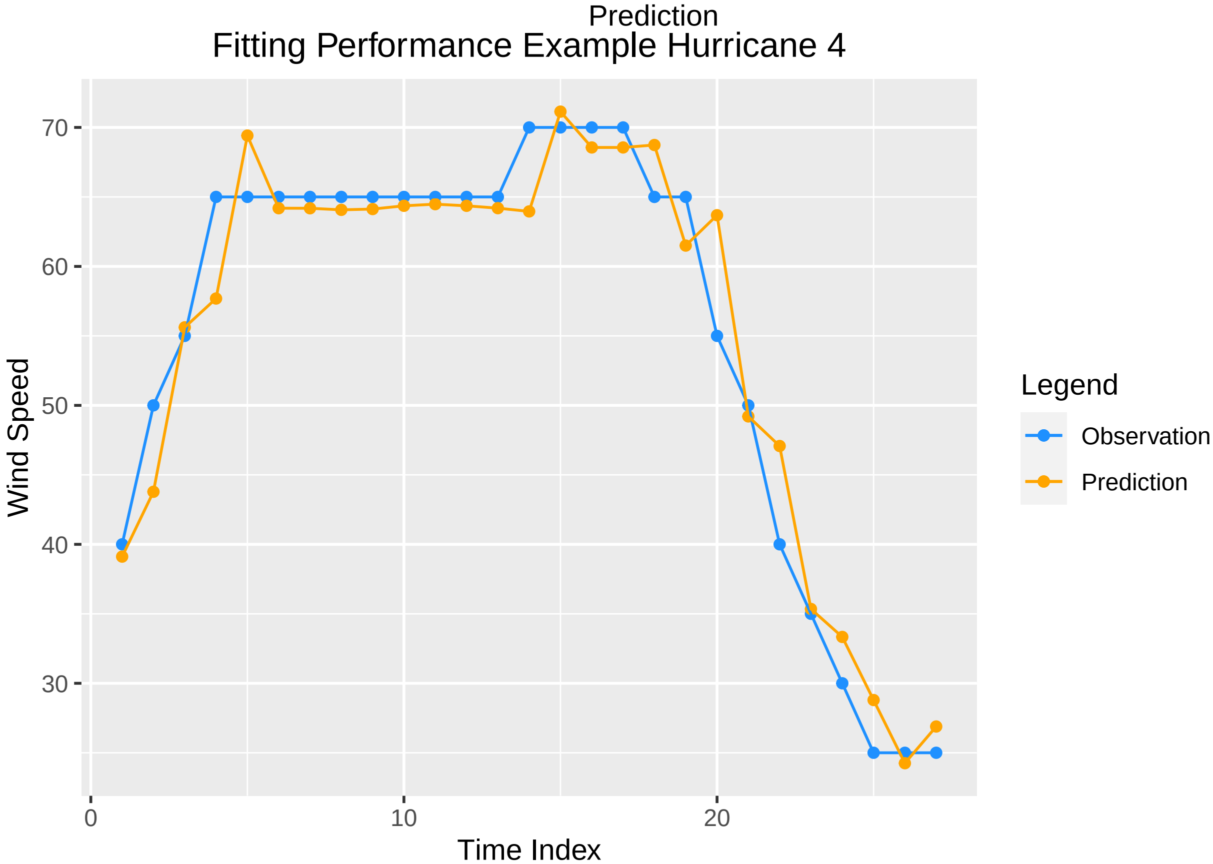

```
## `geom_smooth()` using formula = 'y ~ x'
```



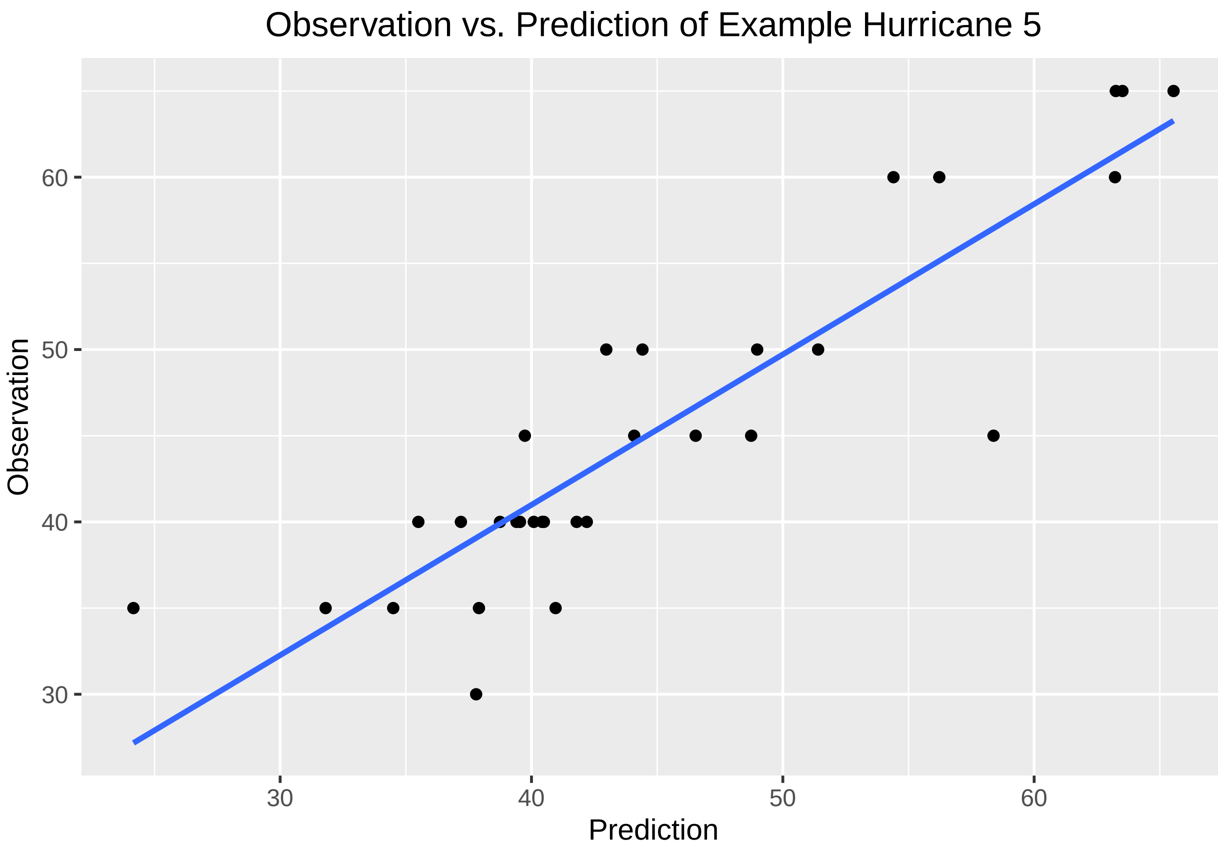


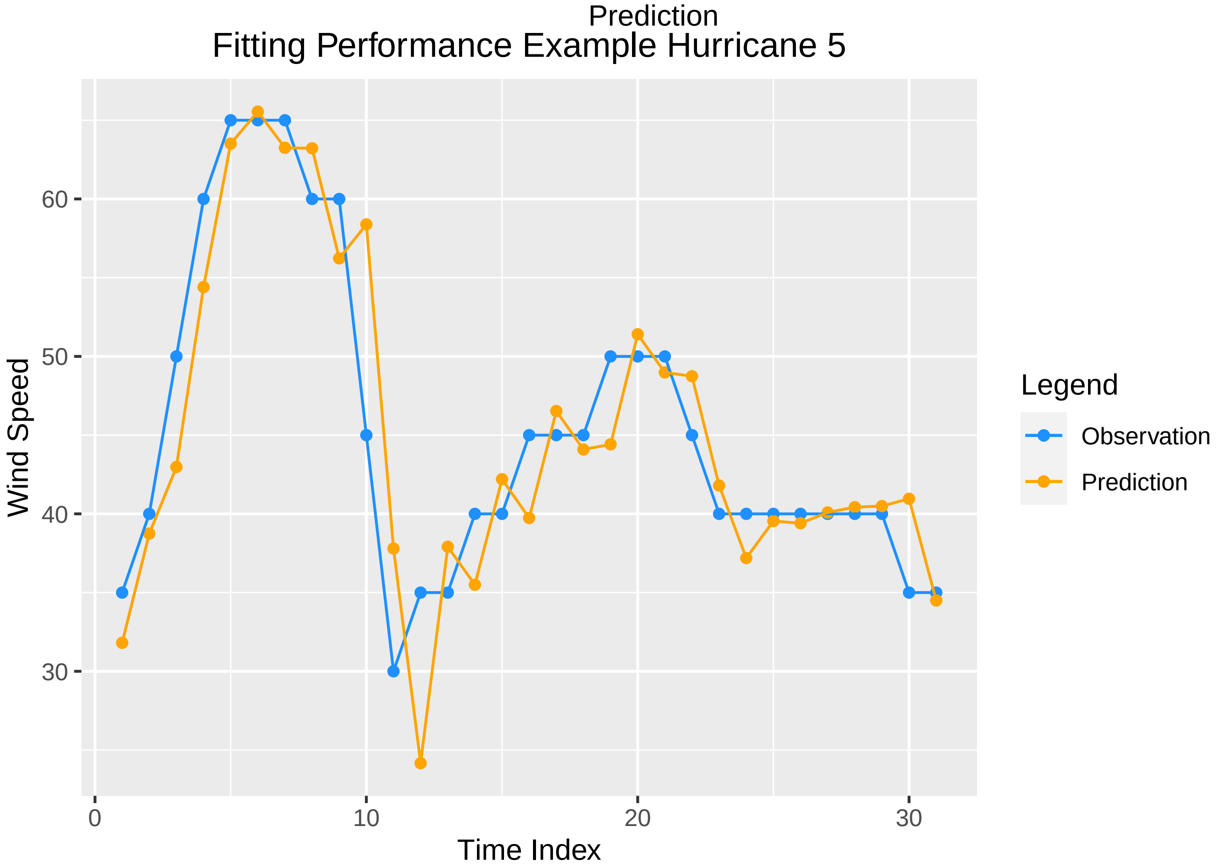

```
## `geom_smooth()` using formula = 'y ~ x'
```






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


Observation vs. Prediction of Example Hurricane 5
Observation
60
50
40
30
30
40
50
60
Prediction


Fitting Performance Example Hurricane 5
Wind Speed
60
50
40
30
0
10
20
30
Time Index
Legend
Observation
Prediction