hw4_pdf

Ruihang Han

2024-09-27

```
library(data.table)
library(lubridate)
```

```
##
##     'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
file_root <- "https://www.ndbc.noaa.gov/view_text_file.php?filename=44013h"
years <- seq(1985, 2023)
tail <- ".txt.gz&dir=data/historical/stdmet/"

all_data <- list()

for (year in years) {
  path <- paste0(file_root, year, tail)

  # Read the first and second lines to determine how many lines to skip
  header <- scan(path, what = 'character', nlines = 1, quiet = TRUE)
  second_line <- scan(path, what = 'character', skip = 1, nlines = 1, quiet = TRUE)
  skip_lines <- ifelse(any(grepl("[a-zA-Z]", second_line)), 2, 1)

  # Read the data
  buoy <- fread(path, header = FALSE, skip = skip_lines, fill = TRUE)

  # Handle column names and ensure column numbers match
  if (length(header) == ncol(buoy)) {
    colnames(buoy) <- header
  } else if (length(header) < ncol(buoy)) {
    extra_cols <- paste0("V", (length(header) + 1):ncol(buoy))
    colnames(buoy) <- c(header, extra_cols)
  } else {
    colnames(buoy) <- header[1:ncol(buoy)]
```

```
  }

  year_col <- grep("^YY$|^YYYY$|^#YY$|^#YYYY$", colnames(buoy), ignore.case = TRUE)
  if (length(year_col) > 0) {
    setnames(buoy, year_col, "YYYY")
    # If the original column is a two-digit year, convert it to four digits
    if (max(buoy$YYYY, na.rm = TRUE) < 100) {
      buoy[, YYYY := ifelse(YYYY < 50, YYYY + 2000, YYYY + 1900)]
    }
  } else {
    # If no year column is found, add one
    buoy[, YYYY := year]
  }

  # Ensure month and day columns exist
  if (!"MM" %in% colnames(buoy)) buoy[, MM := NA]
  if (!"DD" %in% colnames(buoy)) buoy[, DD := NA]

  # Handle wind direction column (WD or WDIR)
  if ("WD" %in% colnames(buoy) && !"WDIR" %in% colnames(buoy)) {
    setnames(buoy, "WD", "WDIR")
  } else if ("WDIR" %in% colnames(buoy) && "WD" %in% colnames(buoy)) {
    buoy[, WDIR := ifelse(is.na(WDIR), WD, WDIR)]
    buoy[, WD := NULL]
  }

  # Handle pressure column (BAR or PRES)
  if ("BAR" %in% colnames(buoy) && !"PRES" %in% colnames(buoy)) {
    setnames(buoy, "BAR", "PRES")
  } else if ("PRES" %in% colnames(buoy) && "BAR" %in% colnames(buoy)) {
    buoy[, PRES := ifelse(is.na(PRES), BAR, PRES)]
    buoy[, BAR := NULL]
  } else if ("PRES" %in% colnames(buoy)) {
    # PRES column already exists, no need to modify
  } else if ("BAR" %in% colnames(buoy)) {
    setnames(buoy, "BAR", "PRES")
  }

  # Store data for each year
  all_data[[as.character(year)]] <- buoy
}
```

```
## Warning in fread(path, header = FALSE, skip = skip_lines, fill = TRUE): Stopped
## early on line 5114. Expected 16 fields but found 17. Consider fill=17 or even
## more based on your knowledge of the input file. Use fill=Inf for reading the
## whole file for detecting the number of fields. First discarded non-empty line:
## <<2000 08 01 00 78 4.3 5.1 0.58 8.33 5.36 999 1022.9 17.3 17.5 15.0 99.0
## 99.00>>
```

```
test_data <- rbindlist(all_data, use.names = TRUE, fill = TRUE)

# Ensure YYYY, MM, DD columns are at the front
setcolorder(test_data, c("YYYY", "MM", "DD"))
```

```
test_data[, DATE := ymd(paste(YYYY, MM, DD, sep = "-"))]

setcolorder(test_data, c("DATE", "YYYY", "MM", "DD"))

head(test_data)
```

```
##          DATE  YYYY    MM    DD    hh  WDIR  WSPD   GST  WVHT   DPD   APD   MWD
##        <Date> <num> <int> <int> <int> <int> <num> <num> <num> <num> <num> <int>
## 1: 1985-01-01  1985     1     1     0    60     4     5    99    99    99   999
## 2: 1985-01-01  1985     1     1     1    80     4     5    99    99    99   999
## 3: 1985-01-01  1985     1     1     2   100     4     5    99    99    99   999
## 4: 1985-01-01  1985     1     1     3   100     4     5    99    99    99   999
## 5: 1985-01-01  1985     1     1     4   110     4     5    99    99    99   999
## 6: 1985-01-01  1985     1     1     5    90     4     5    99    99    99   999
##      PRES  ATMP  WTMP  DEWP   VIS  TIDE    mm
##     <num> <num> <num> <num> <num> <num> <int>
## 1: 1030.3   4.7   6.7   999    99    NA    NA
## 2: 1030.0   5.1   6.7   999    99    NA    NA
## 3: 1030.1   5.6   6.6   999    99    NA    NA
## 4: 1029.4   5.8   6.7   999    99    NA    NA
## 5: 1028.6   5.8   6.7   999    99    NA    NA
## 6: 1027.8   5.3   6.7   999    99    NA    NA
```

```
# List the columns that need to be checked for 999 values
na_columns <- c("WDIR", "WSPD", "GST", "WVHT", "DPD", "APD", "MWD",
                "PRES", "ATMP", "WTMP", "DEWP", "VIS", "TIDE")

# Handle missing values by replacing 999 with NA
test_data[, (na_columns) := lapply(.SD, function(x) ifelse(x %in% c(99, 999), NA, x)), .SDcols = na_colu

# Check the distribution of NA values in each column
na_summary <- test_data[, lapply(.SD, function(x) sum(is.na(x))), .SDcols = na_columns]
print(na_summary)
```

```
##     WDIR  WSPD   GST   WVHT    DPD    APD    MWD  PRES   ATMP  WTMP   DEWP
##    <int> <int> <int>  <int>  <int>  <int>  <int> <int>  <int> <int>  <int>
## 1: 44175 33183 33485 144269 147961 144269 327167   261 102761 13186 253613
##       VIS   TIDE
##     <int>  <int>
## 1: 443062 462301
```

```
# Visualize the distribution of missing values
library(ggplot2)
library(reshape2)
```

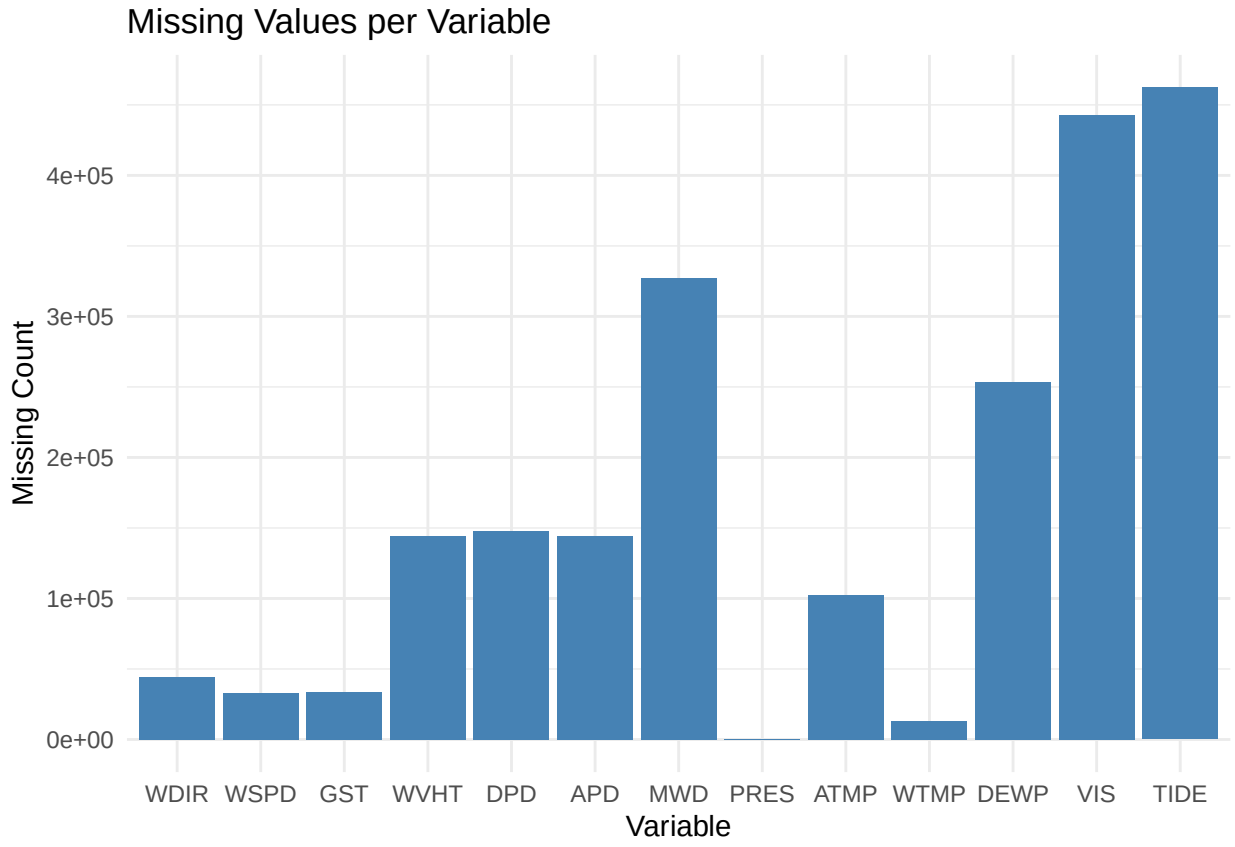
```
##
##    'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```
# Convert missing value summary into a format suitable for visualization
na_melted <- melt(na_summary, variable.name = "Variable", value.name = "Missing_Count")
```

```
## No id variables; using all as measure variables
```

```
ggplot(na_melted, aes(x = Variable, y = Missing_Count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  theme_minimal() +
  labs(title = "Missing Values per Variable", x = "Variable", y = "Missing Count")
```




```
# Load necessary libraries
library(ggplot2)
library(dplyr)
```

```
##
##     'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
library(forecast)
```

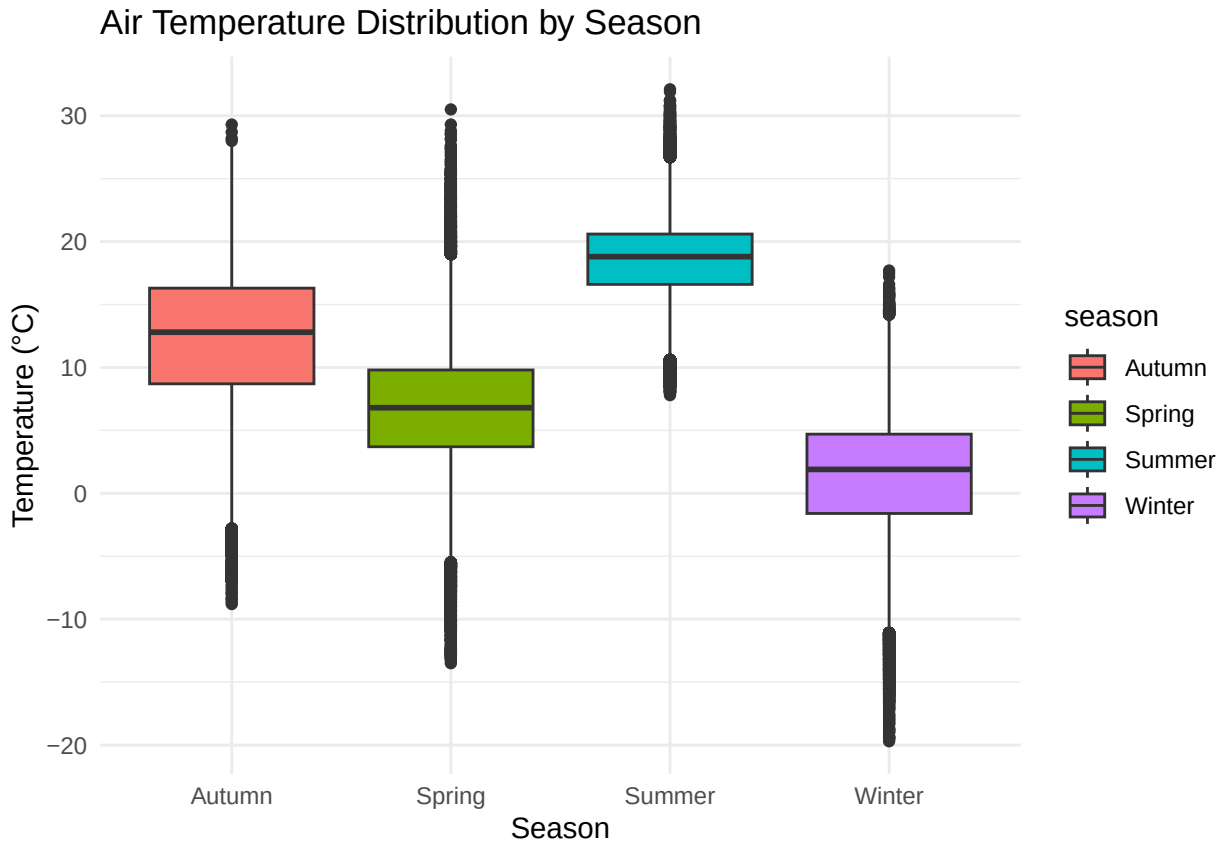
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
# Load data
buoy_data <- test_data
buoy_data$DATE <- as.Date(buoy_data$DATE, format = "%Y-%m-%d")

# Clean data: Remove rows with missing air temperature (ATMP)
buoy_data_clean <- buoy_data %>%
  filter(!is.na(ATMP))

# Create a season column
buoy_data_clean <- buoy_data_clean %>%
  mutate(season = case_when(
    month(DATE) %in% c(3, 4, 5)  ~ "Spring",
    month(DATE) %in% c(6, 7, 8)  ~ "Summer",
    month(DATE) %in% c(9, 10, 11) ~ "Autumn",
    month(DATE) %in% c(12, 1, 2) ~ "Winter"
  ))

# Calculate the average air temperature for each season
seasonal_avg <- buoy_data_clean %>%
  group_by(season) %>%
  summarise(mean_temp = mean(ATMP, na.rm = TRUE),
            sd_temp = sd(ATMP, na.rm = TRUE))

# Print the average air temperature for each season
print(seasonal_avg)
```

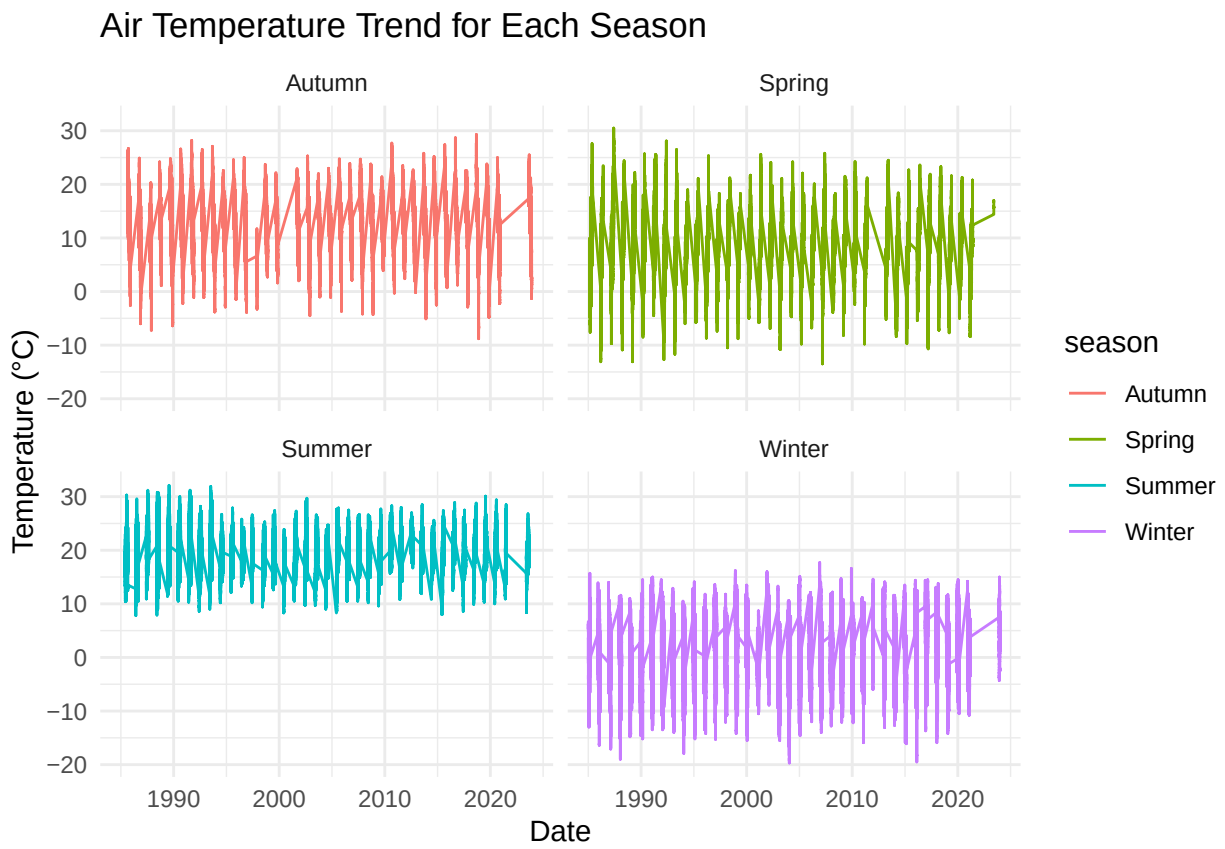
```
## # A tibble: 4 x 3
##   season mean_temp sd_temp
##   <chr>      <dbl>   <dbl>
## 1 Autumn     12.4     5.18
## 2 Spring      6.79    4.86
## 3 Summer     18.5     3.12
## 4 Winter      1.37    4.98
```

```
# Plot the distribution of air temperature for each season (boxplot)
ggplot(buoy_data_clean, aes(x = season, y = ATMP, fill = season)) +
  geom_boxplot() +
```

```
  labs(title = "Air Temperature Distribution by Season", x = "Season", y = "Temperature (°C)") +
  theme_minimal()
```




```
# Plot the trend of air temperature changes for each season
ggplot(buoy_data_clean, aes(x = DATE, y = ATMP, color = season)) +
  geom_line() +
  facet_wrap(~ season) + # Plot by season
  labs(title = "Air Temperature Trend for Each Season", x = "Date", y = "Temperature (°C)") +
  theme_minimal()
```



This analysis reveals significant differences in temperature distribution across seasons. Summer exhibits the highest temperatures, ranging from 15°C to 30°C, while winter shows the lowest, between -10°C and 10°C. Spring and autumn serve as transitional seasons with broader temperature variations, with spring generally being cooler than autumn. The long-term trend analysis indicates clear seasonal cyclic patterns: summer temperatures are relatively stable, while winter shows more significant fluctuations, with some years displaying extreme low temperatures. Although there is no strong evidence of a clear long-term warming trend, certain anomalous fluctuations, particularly in spring and winter, could be indicative of climate change effects, such as more frequent extreme weather events.

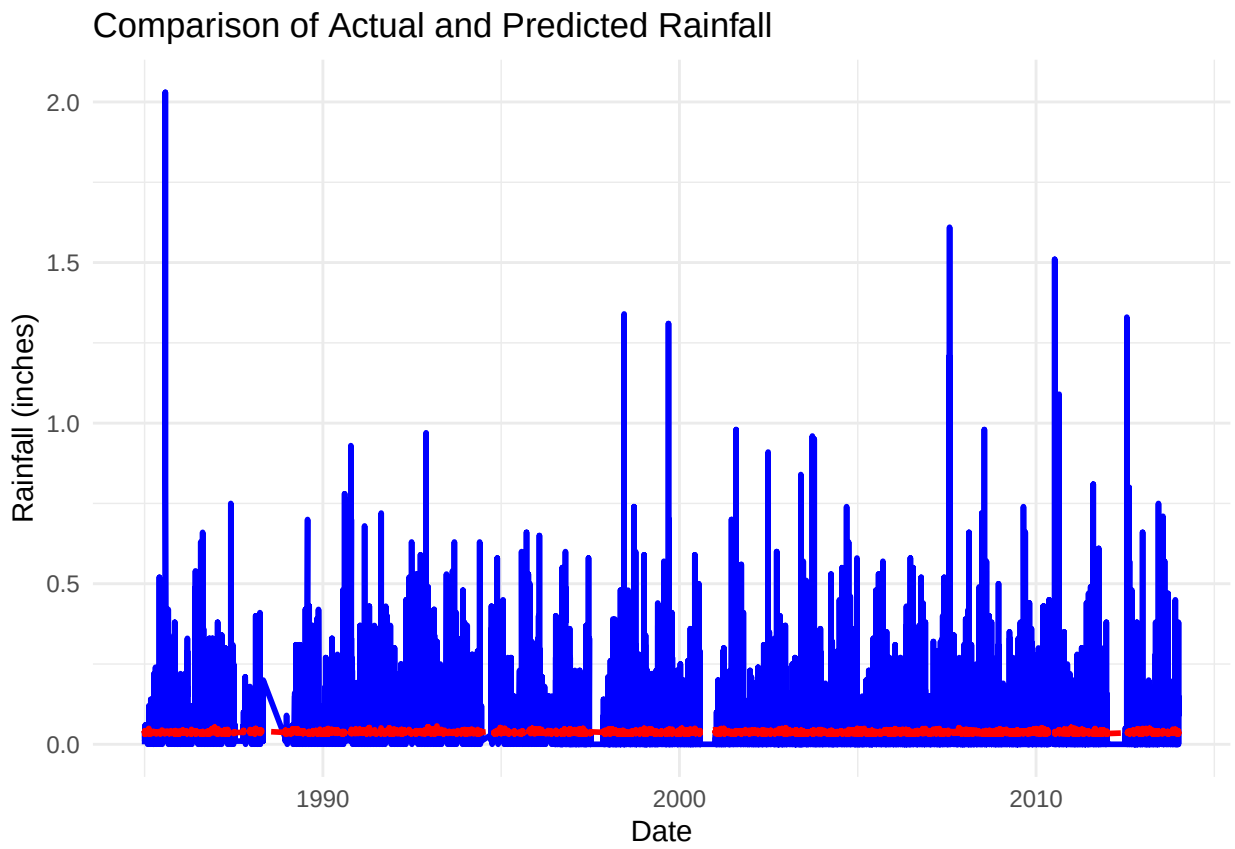
This analysis suggests that further investigation is needed, especially through fitting seasonal time-series models like ARIMA, to explore any long-term temperature trends. Additionally, a comparative study of average temperatures across different periods could help determine whether climate change has led to shifts in seasonal temperature distributions, such as more frequent extreme heat in summer or increased extreme cold events in winter.

```r
# Load necessary R packages
library(readxl)
library(ggplot2)
library(dplyr)
library(lubridate)

# 1. Load data
buoy_data <- test_data  # Ensure test_data is properly loaded
rainfall_data <- read.csv("Rainfall.csv")

# 2. Use lubridate to process dates
# Convert the DATE column in the rainfall data to a date format
```

```r
rainfall_data <- rainfall_data %>%
  mutate(date_only = ymd_hm(DATE))  # Ensure the DATE column format is compatible with ymd_hm

# Convert year, month, day, and hour columns in buoy data to a datetime format
buoy_data$date_only <- ymd_h(paste(buoy_data$YYYY, buoy_data$MM, buoy_data$DD, buoy_data$hh, sep='-'))

# 3. Merge data and remove unnecessary columns
rain_subset <- rainfall_data[,c("date_only", "HPCP")]
rain_subset <- rain_subset %>%
  left_join(select(buoy_data, date_only, WSPD), by = "date_only")

# Remove rows with missing values
rain_subset <- na.omit(rain_subset)

# 4. Build a linear regression model to analyze the relationship between rainfall (HPCP) and wind speed
model <- lm(HPCP ~ WSPD, data = rain_subset)

# 5. View the model summary
summary(model)
```

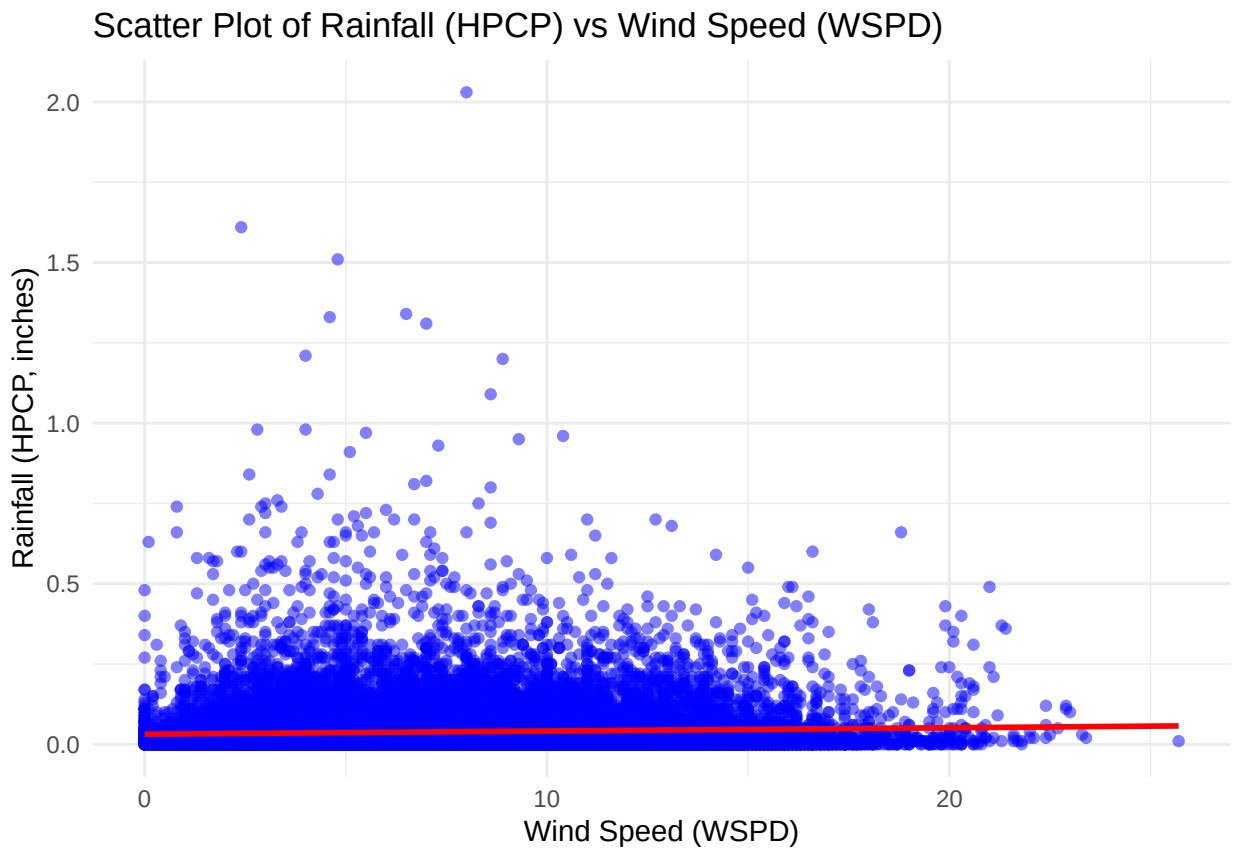
```
##
## Call:
## lm(formula = HPCP ~ WSPD, data = rain_subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.05329 -0.03630 -0.02620  0.00420  1.99058
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0313792  0.0009445   33.22   <2e-16 ***
## WSPD        0.0010049  0.0001174    8.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07598 on 28869 degrees of freedom
## Multiple R-squared:  0.002531,   Adjusted R-squared:  0.002497
## F-statistic: 73.27 on 1 and 28869 DF,  p-value: < 2.2e-16
```

```r
# 6. Predict rainfall and add the predicted values to the data frame
rain_subset <- rain_subset %>%
  mutate(predicted_rainfall = predict(model, .))

# 7. Plot actual rainfall and predicted rainfall comparison
ggplot(rain_subset, aes(x = date_only)) +
  geom_line(aes(y = HPCP), color = "blue", linewidth = 1) +  # Actual rainfall
  geom_line(aes(y = predicted_rainfall), color = "red", linetype = "dashed", linewidth = 1) +  # Predic
  labs(title = "Comparison of Actual and Predicted Rainfall", x = "Date", y = "Rainfall (inches)") +
  theme_minimal()
```




```
# Scatter plot of Rainfall (HPCP) vs Wind Speed (WSPD)
ggplot(rain_subset, aes(x = WSPD, y = HPCP)) +
  geom_point(color = "blue", alpha = 0.5) +  # Plot scatter points
  geom_smooth(method = "lm", color = "red", se = FALSE) +  # Add regression line, no confidence interva
  labs(title = "Scatter Plot of Rainfall (HPCP) vs Wind Speed (WSPD)",
       x = "Wind Speed (WSPD)",
       y = "Rainfall (HPCP, inches)") +
  theme_minimal()
```

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



The first image shows a time-series plot comparing actual rainfall (in blue) and predicted rainfall (in red) over time. The predicted rainfall remains quite flat and does not capture the variability or peaks in the actual rainfall. This suggests that the linear regression model, which predicts rainfall based on wind speed (WSPD), does not accurately capture the relationship between the two variables.

The second image shows the output of the linear regression model. The coefficient for wind speed (WSPD) is positive and statistically significant (p-value $<$ 2.2e-16), indicating that there is a relationship between wind speed and rainfall. However, the R-squared value is extremely low (0.0025), suggesting that wind speed explains only a small portion of the variance in rainfall. The model's predictive power is weak, as indicated by the low adjusted R-squared, meaning that wind speed is not a strong predictor of rainfall in this case.

In summary, while wind speed has a statistically significant relationship with rainfall, it explains very little of the variation, and the model does not predict rainfall effectively based on this single factor.