Moon VS. Weather Patterns

Ibrahim Noman

2024-04-29

Libraries

Data Cleaning

Loading in data

```
moon_data <- read.csv('moon_data.csv')
weather_data <- read.csv('Wildfire_Weather_Merged_new.csv')
```

Since we only want the records for 2022, we will need to manipulate our weather_data. Our moon data starts for January 2nd, 2022, so we will filter it to that.

```
# Filtering for 2022
weather_22 <- weather_data %>%
  filter(Date >= '2022-01-02') %>%
  dplyr::select('Date', 'County', 'tmax', 'tmin', 'tavg', 'prcp')

# Removing unnecessary column in moon data
moon_data <- moon_data[ ,-1]
```

Joining data

Joining the data on the date column

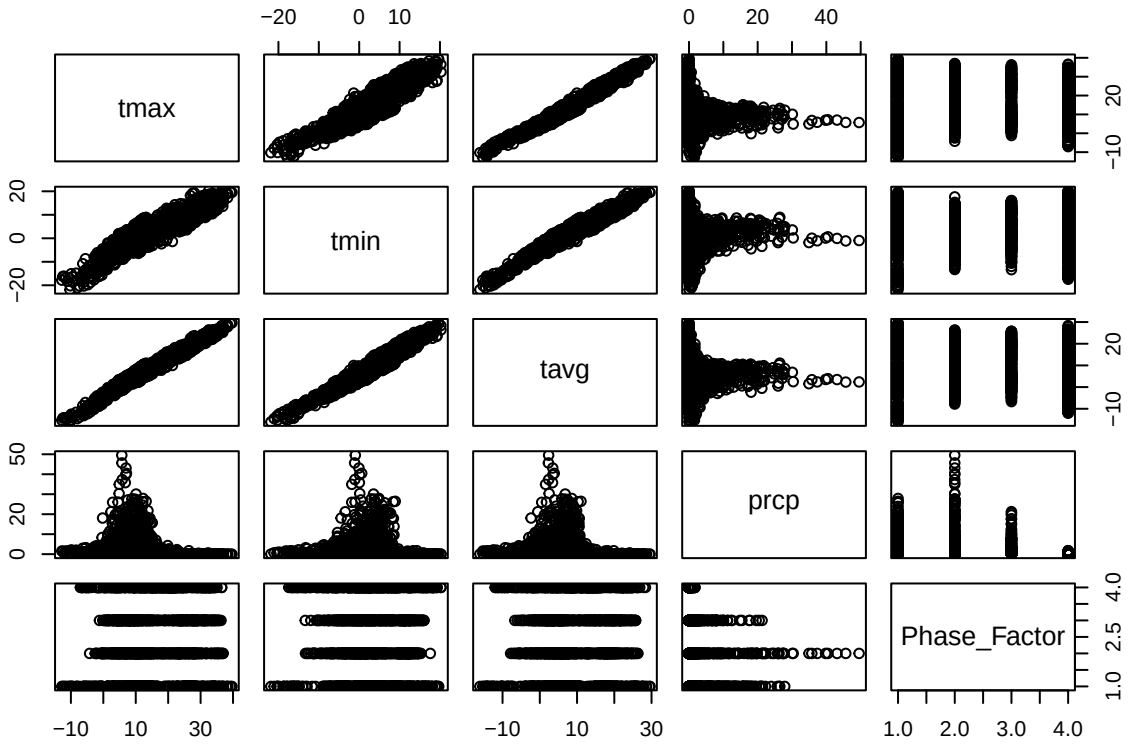
```
# Fixing Date
moon_data$Date <- ymd(moon_data$Date)
weather_22$Date <- ymd(weather_22$Date)

# Join
data <- full_join(moon_data, weather_22, by = 'Date')

# Quick Cleaning
data <- data %>%
  dplyr::select(-c('Date', 'Time', 'County')) %>%
  mutate(Phase_Factor = case_when(Moon_Phase == 'New Moon' ~ 1,
                                  Moon_Phase == 'First Quarter' ~ 2,
                                  Moon_Phase == 'Full Moon' ~ 3,
                                  Moon_Phase == 'Last Quarter' ~ 4)) %>%
  na.omit()
```

Exploratory Data Analysis

Now that our data has been cleaned, we can start analyzing the different predictors.

Plots

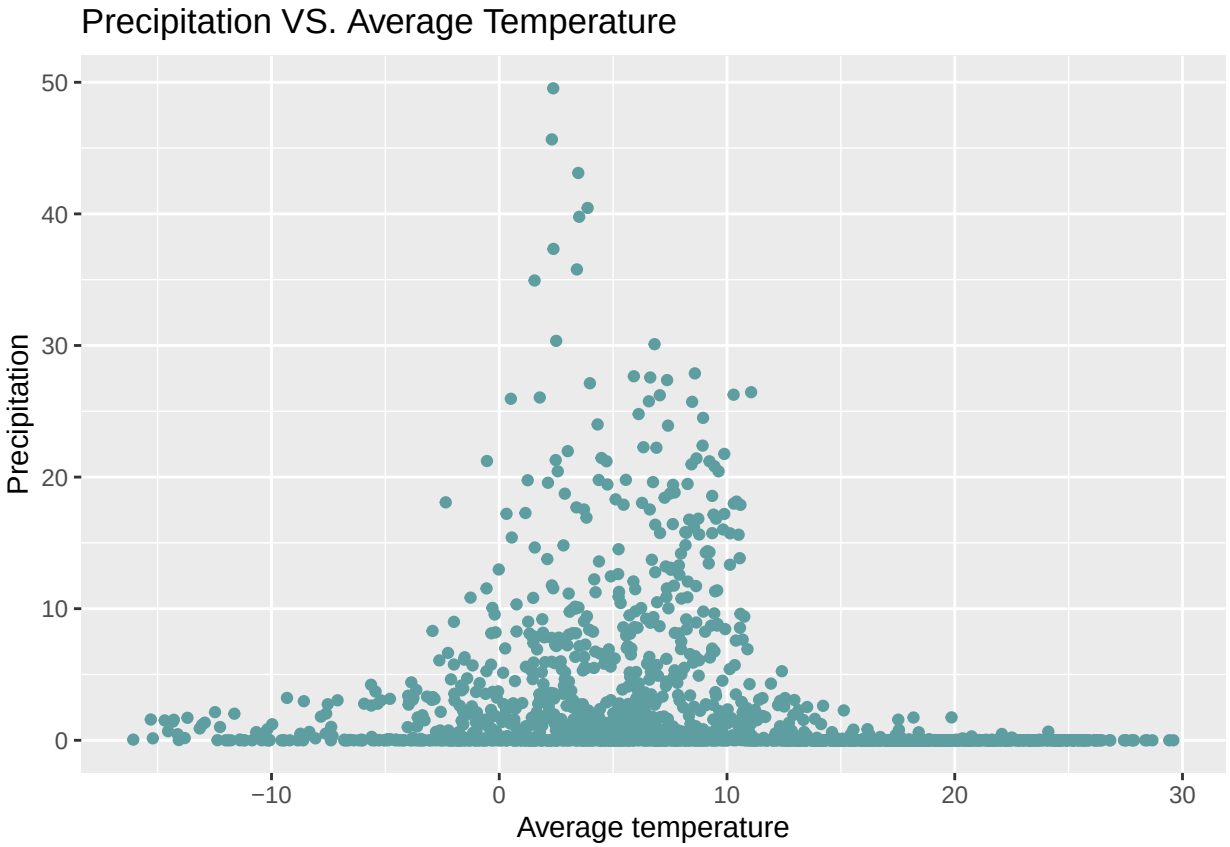
```
# Scatter plot Matrix
plot_data <- data %>%
  dplyr::select(-c('Moon_Phase'))

pairs(plot_data)
```



Temperature and Precipitation

From the plot below, we can see that there isn't a clear linear relationship between Rain and Temperature.

```
ggplot(plot_data) + aes(x = tavg, y = prcp) + geom_point(color = 'cadetblue') + theme_gray() + labs(x =
```



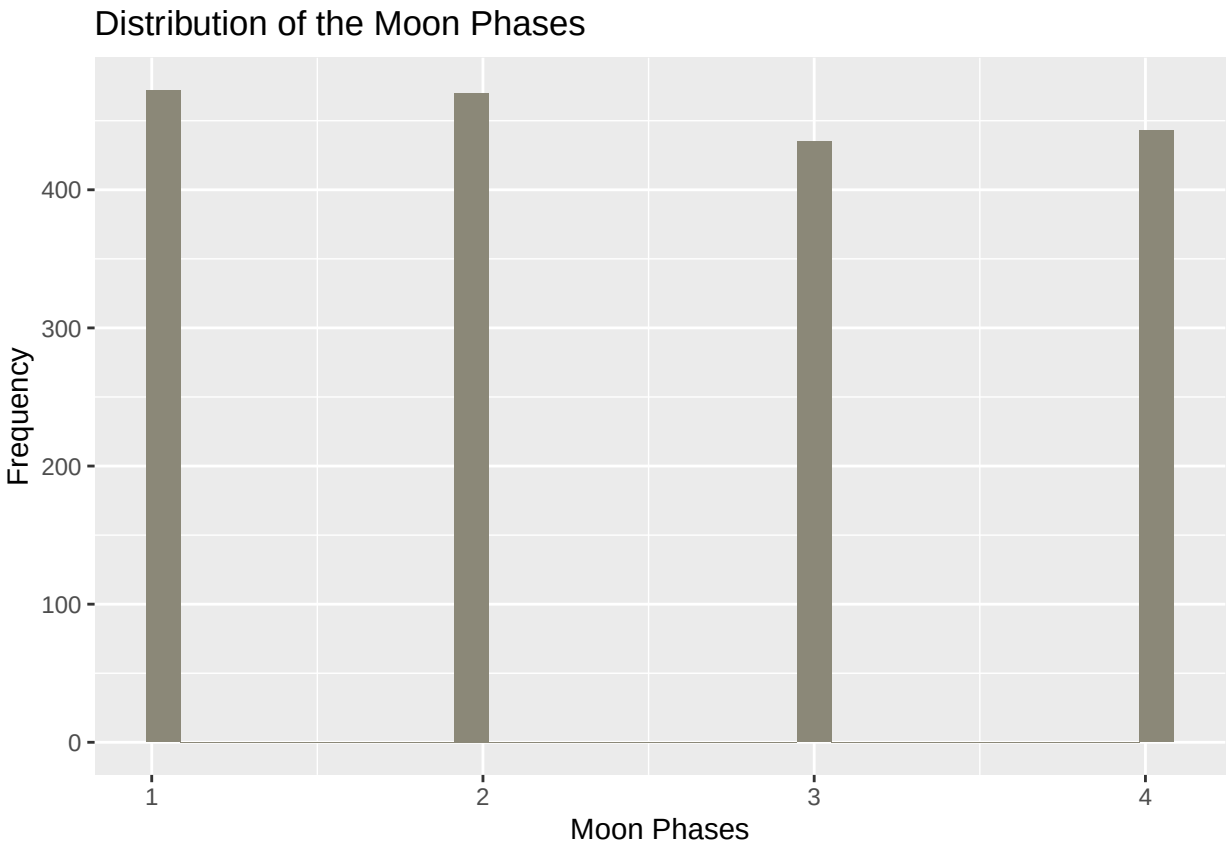
Moon Phases

First we need to see what the distributions of the different phases looks like. If you remember earlier, we converted the phases into the factor type. This will help us later in our linear regression.

We can see that there is no bias in our phase distributions. we have an equal amount of frequency for all 4 phases.

```
ggplot(plot_data) + aes(x = Phase_Factor) + geom_histogram(fill = 'cornsilk4') + theme_gray() + labs(x =
```

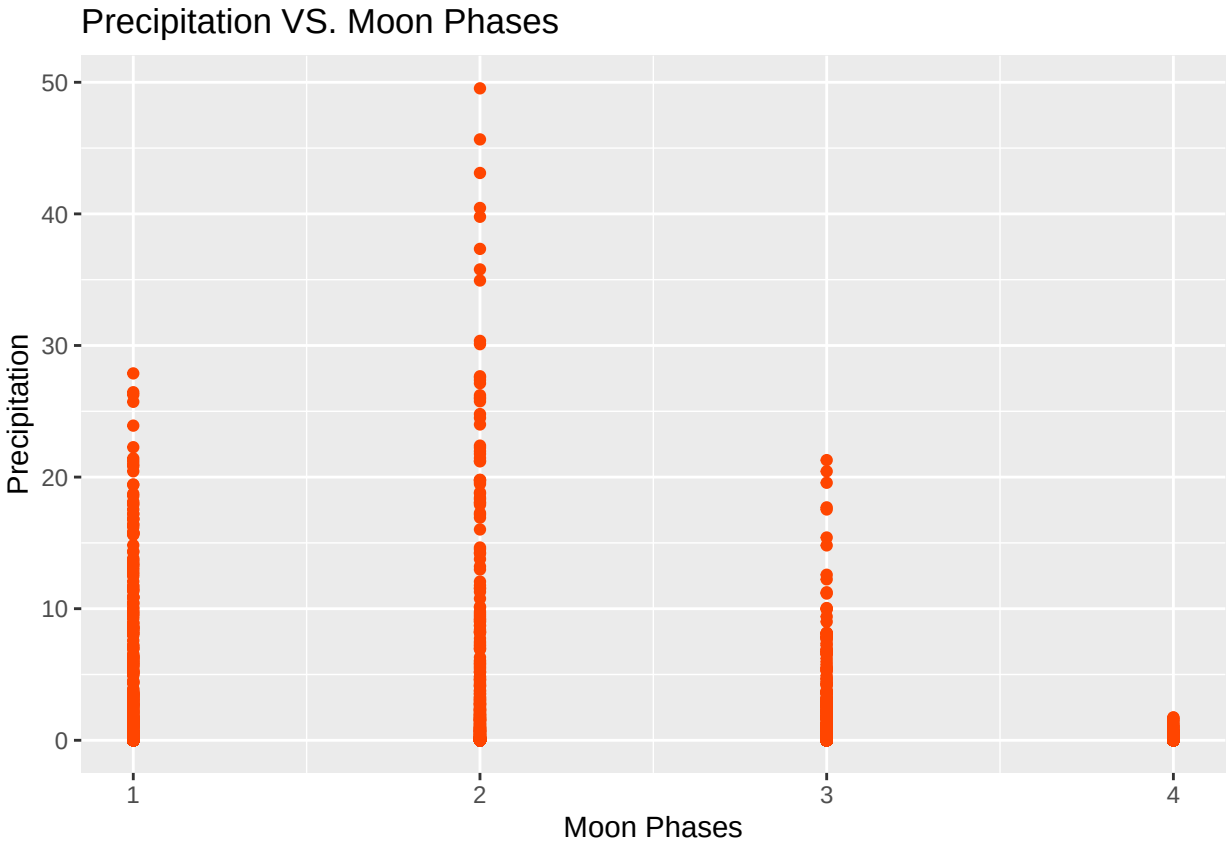
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Precipitation Now let's try to explore the relationships with temperature and precipitation. From the plot below we can see that:

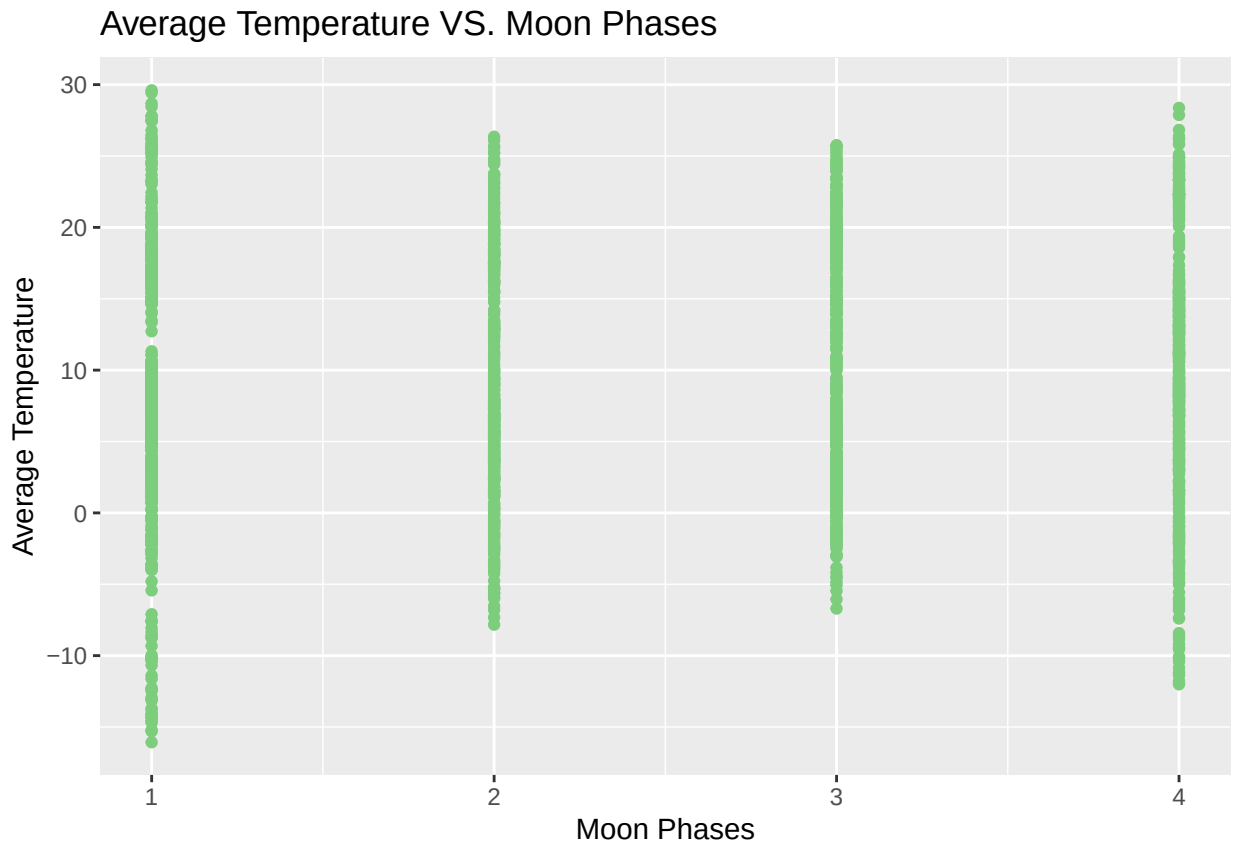
- Majority of rain happens during the first quarter (2)
- The least amount of rain happens during the last quarter (4)

```
ggplot(plot_data) + aes(x = Phase_Factor, y = prcp) + geom_point(color = 'orangered') + theme_gray() + l
```



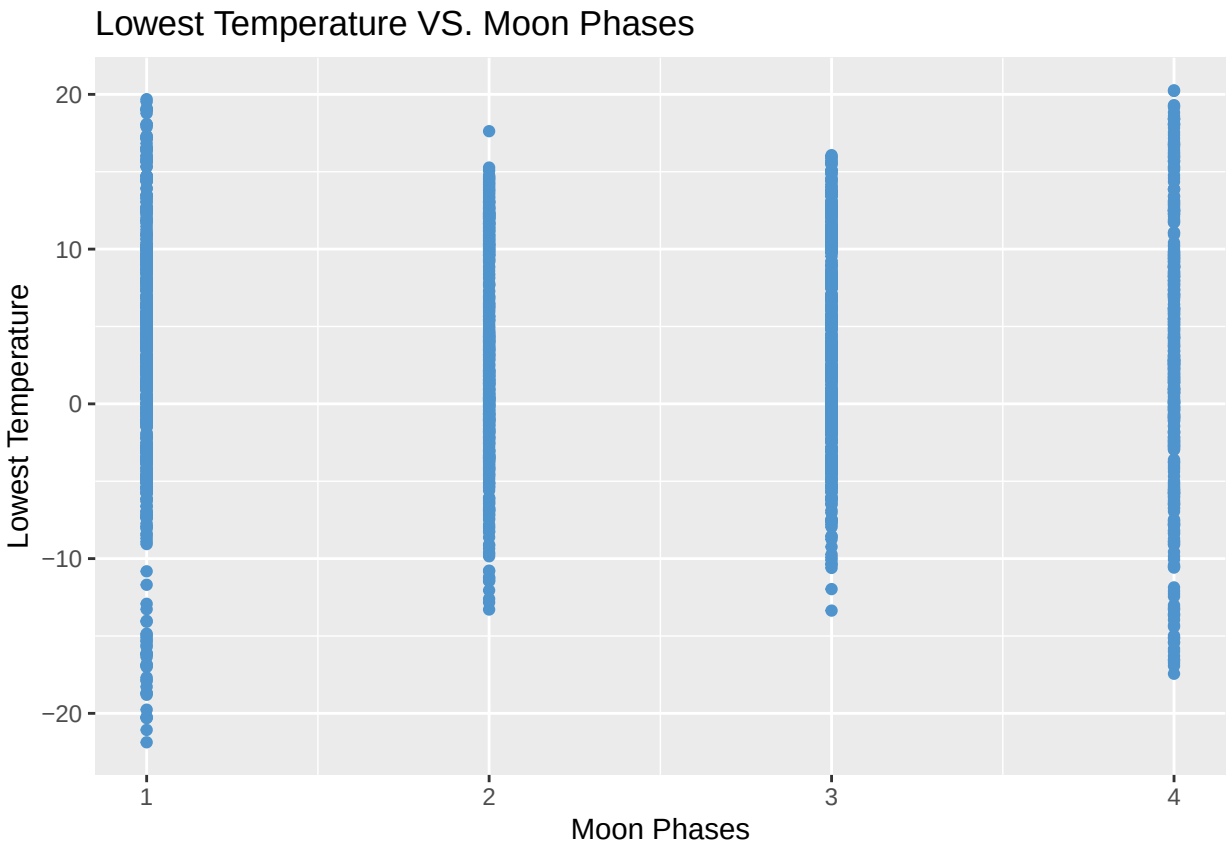
Temperature The plot below shows that the relationship between average temperature and and Moon phases is roughly constant. There aren't any strong differences.

```
ggplot(plot_data) + aes(x = Phase_Factor, y = tavg) + geom_point(color = 'palegreen3') + theme_gray() +
```

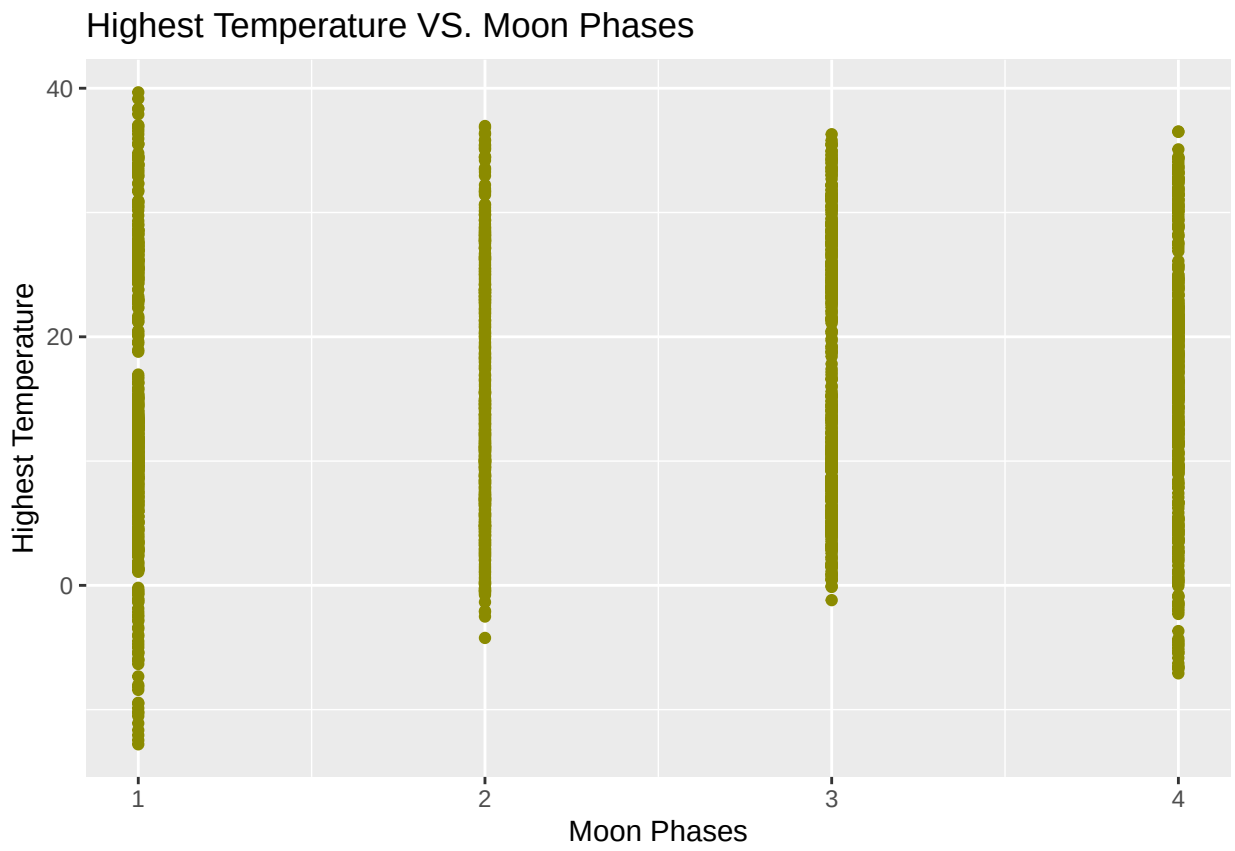


Now, let's explore when the temperature is the lowest/maximum.

```
ggplot(plot_data) + aes(x = Phase_Factor, y = tmin) + geom_point(color = 'steelblue3') + theme_gray() +
```




```
ggplot(plot_data) + aes(x = Phase_Factor, y = tmax) + geom_point(color = 'yellow4') + theme_gray() + lab
```



From the plots above we notice:

- Temperature varies the most during the New Moon (1) and the Last Quarter (4).
- Temperature remains roughly the same for the First Quarter (2) and the Full Moon (3)

Linear Regression

Now let's understand if there is any linear relationship when predicting rainfall.

```
# Taking out Moon Phases since I changed them to categorical
model_data <- data %>%
  dplyr::select(-c('Moon_Phase'))

model_data$Phase_Factor <- as.factor(model_data$Phase_Factor)

# Model
model <- lm(prcp ~., data = model_data)
summary(model)
```

```
##
## Call:
## lm(formula = prcp ~ ., data = model_data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.749 -2.473 -0.677  1.243 43.111
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.3069     0.3507  29.392  < 2e-16 ***
## tmax          -11.6477    15.3408  -0.759    0.448
## tmin          -10.2815    15.3436  -0.670    0.503
## tavg           22.0425    30.6841   0.718    0.473
## Phase_Factor2   0.4037     0.2963   1.362    0.173
## Phase_Factor3  -1.6711     0.3023  -5.527 3.73e-08 ***
## Phase_Factor4  -2.0978     0.3063  -6.849 1.01e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.538 on 1813 degrees of freedom
## Multiple R-squared:  0.3005, Adjusted R-squared:  0.2982
## F-statistic: 129.8 on 6 and 1813 DF,  p-value: < 2.2e-16
```

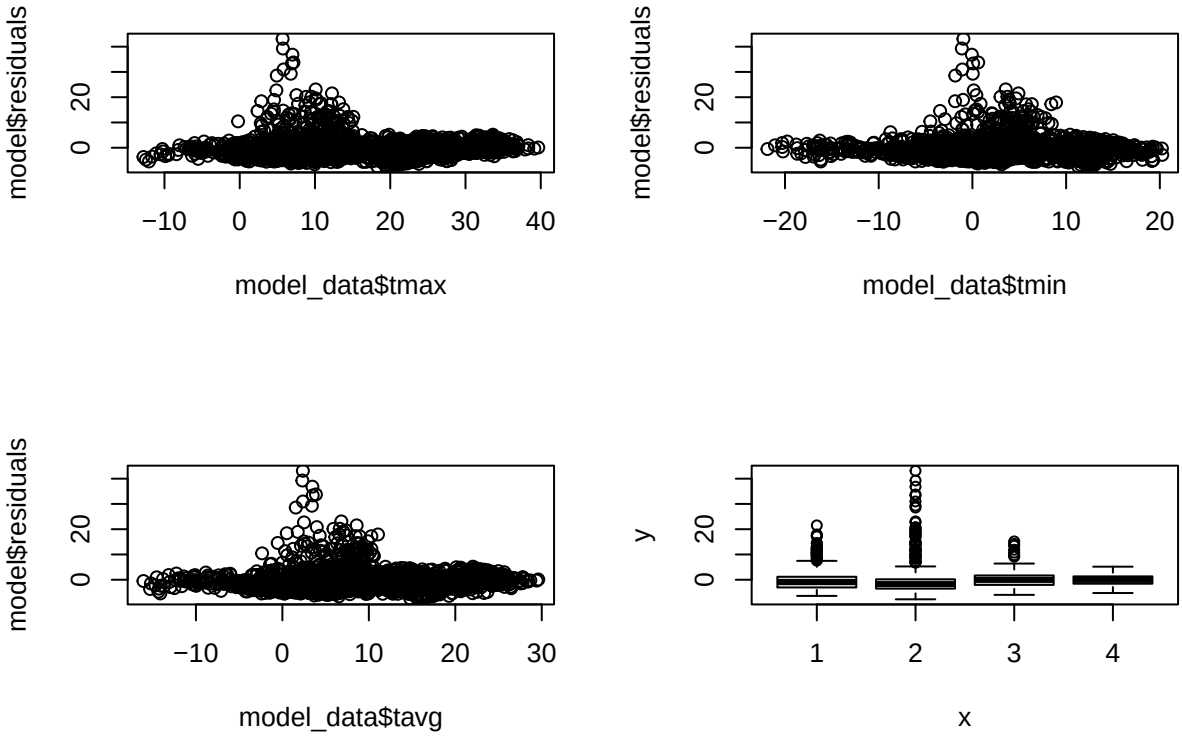
After constructing the first model, we can see that only the Moon phases turn out to statistically significant. Let's see if the Regression Assumptions hold true.

Linear Assumptions

Linearity + Constant Variance

From the residual plots below we can see that the variance isn't constant. There are some patterns that need to be fixed.

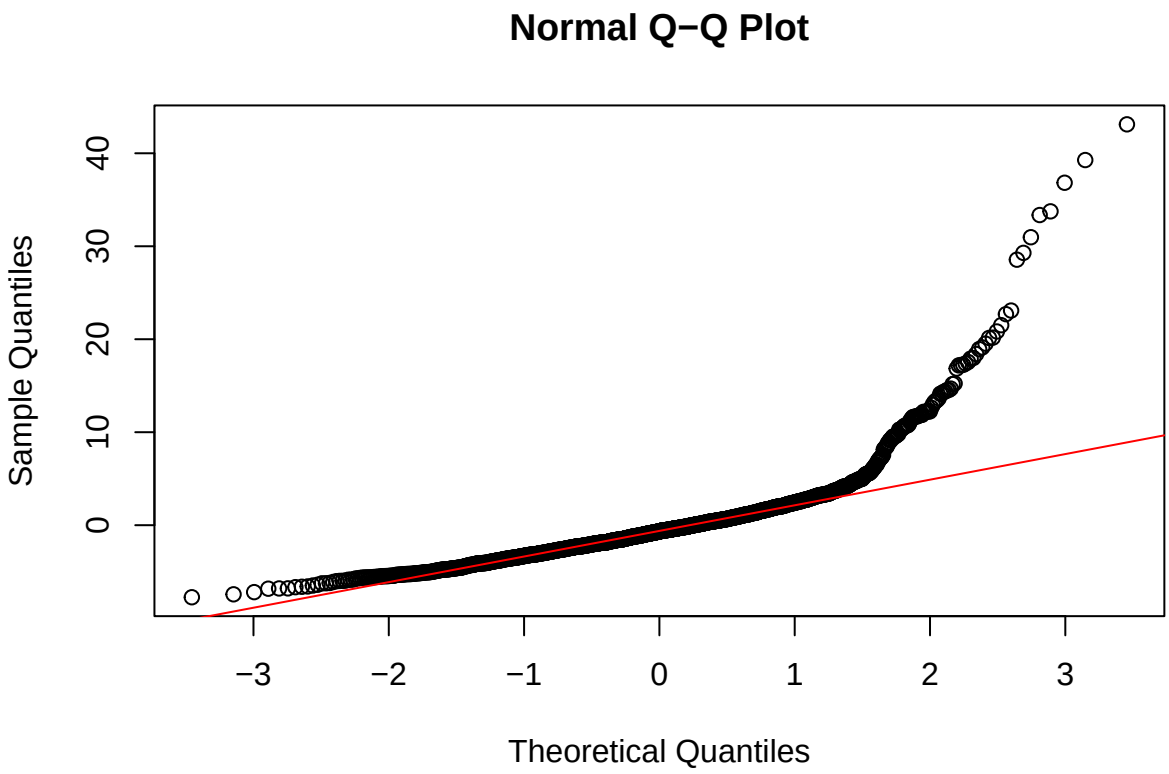
```
# Residuals vs. Predictor Variables
par(mfrow = c(2,2))
plot(x = model_data$tmax, y = model$residuals)
plot(x = model_data$tmin, y = model$residuals)
plot(x = model_data$tavg, y = model$residuals)
plot(x = model_data$Phase_Factor, y = model$residuals)
```



Gaussianity

Same can be said for the Gaussian assumption, since the top tail deviates quite significantly.

```
qqnorm(model$residuals)
qqline(model$residuals, col = 'red')
```



Fixing

To fix the issues mentioned above we will try the following things:

- Apply transformations
- Split the data into training and test sets

```
# Accounting for 0 prcp values
model_data_2 <- model_data
model_data_2$prcp <- log(model_data$prcp + 0.01)

# Number of observations
n = nrow(model_data_2)

# 70% to train
n_train = n * 0.7

# Creating Indices
train <- sample(x = 1:n, size = n_train)

train_set <- model_data_2[train, ]
test_set <- model_data_2[-train, ]
```

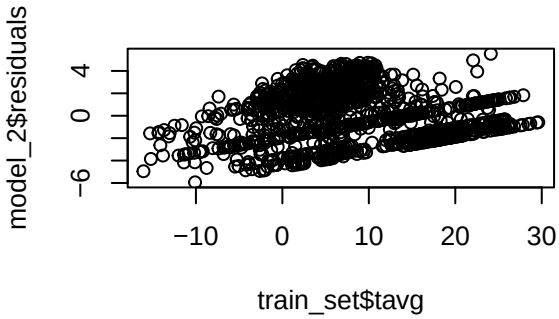
```r
# Model
model_2 = lm(prcp ~ tavg + Phase_Factor, data = train_set)
summary(model_2)
```

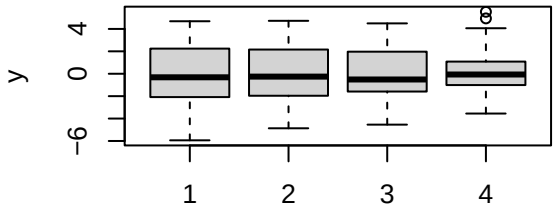
```
##
## Call:
## lm(formula = prcp ~ tavg + Phase_Factor, data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9402 -1.7559 -0.2204  1.7543  5.5273
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.027840   0.138862  -0.200    0.841
## tavg          -0.134935   0.007386 -18.269  < 2e-16 ***
## Phase_Factor2 -0.282536   0.178087  -1.587    0.113
## Phase_Factor3 -0.773193   0.182842  -4.229 2.52e-05 ***
## Phase_Factor4 -2.646659   0.182870 -14.473  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.3 on 1269 degrees of freedom
## Multiple R-squared:  0.3283, Adjusted R-squared:  0.3262
## F-statistic:   155 on 4 and 1269 DF,  p-value: < 2.2e-16
```

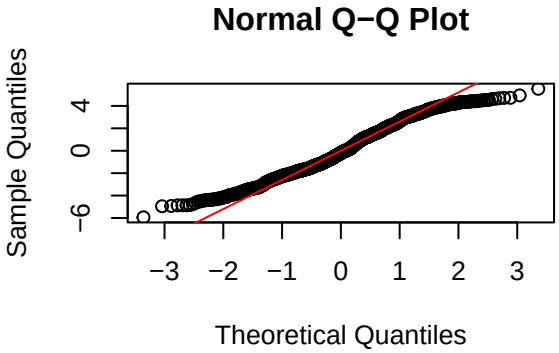
After applying the log transformation we can see that the average temperature is now significant in the model.

Even after transforming the data, we can see that the constant variance assumption does not hold, however the Gaussian Assumption looks a lot better.

```r
par(mfrow = c(2,2))
plot(x = train_set$tavg, y = model_2$residuals)
plot(x = train_set$Phase_Factor, y = model_2$residuals)

qqnorm(model_2$residuals)
qqline(model_2$residuals, col = 'red')
```



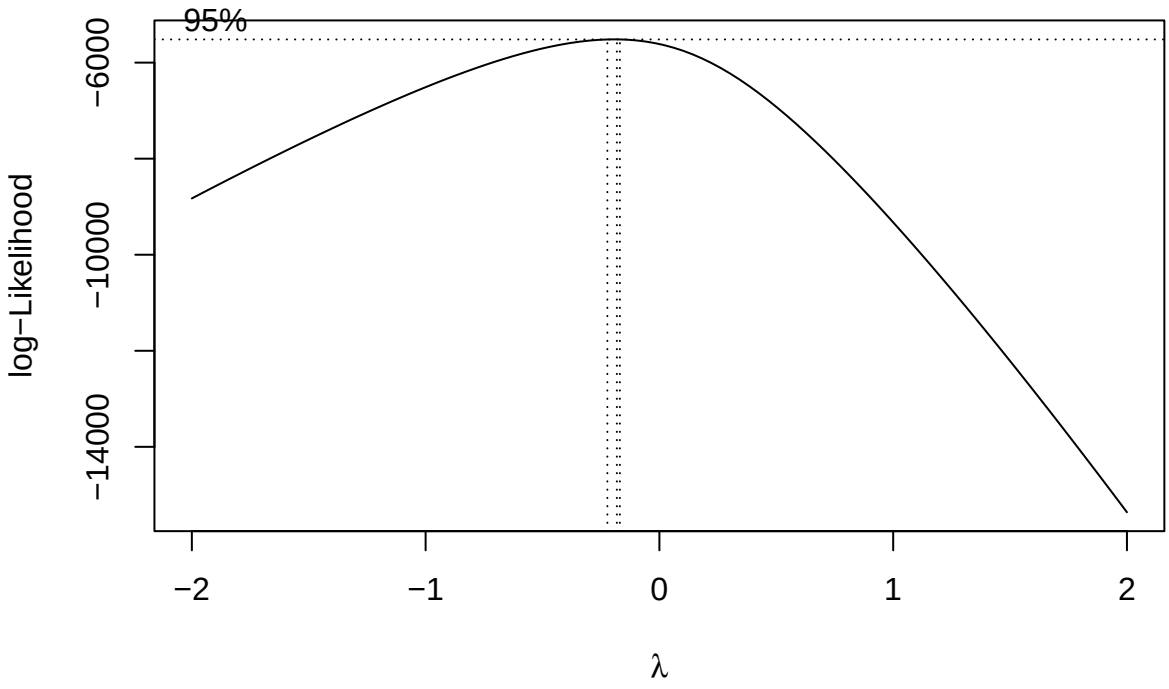




Boxcox

Let's now try the Box Cox transformation which might help us get the job done.

```
# Creating new df
box_data <- model_data %>%
  mutate(prcp = prcp + 0.01)

# Splitting data
train_box <- box_data[train, ]

# Boxcox
b <- boxcox(lm(prcp ~ tavg + Phase_Factor, data = train_box))
```




```
lambda <- b$x[which.max(b$y)]

# Transforming prcp
train_box$prcp_transformed <- (train_box$prcp^lambda - 1) / lambda
```

From the summary table below we can see that not much changed.

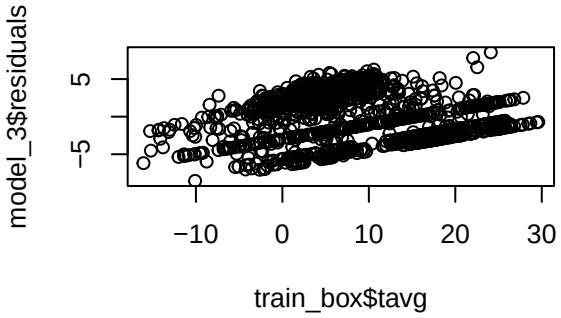
```
# Model
model_3 <- lm(prcp_transformed ~ tavg + Phase_Factor, data = train_box)
summary(model_3)
```

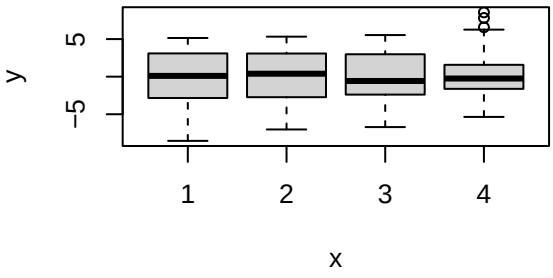
```
##
## Call:
## lm(formula = prcp_transformed ~ tavg + Phase_Factor, data = train_box)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.5441 -2.4801 -0.1243  2.7350  8.5575
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.65402    0.18949  -3.452 0.000576 ***
## tavg          -0.19726    0.01008 -19.573  < 2e-16 ***
## Phase_Factor2 -0.36420    0.24301  -1.499 0.134200
## Phase_Factor3 -0.91952    0.24950  -3.685 0.000238 ***
```

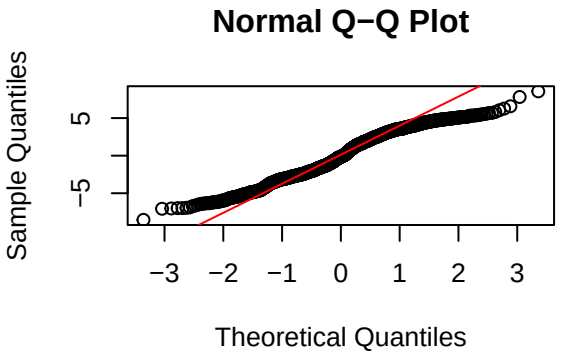
```
## Phase_Factor4 -3.56281    0.24954 -14.278  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.138 on 1269 degrees of freedom
## Multiple R-squared:  0.3444, Adjusted R-squared:  0.3423
## F-statistic: 166.6 on 4 and 1269 DF,  p-value: < 2.2e-16
```

Same can be said for the assumptions.

```
# Residul Plots
par(mfrow = c(2,2))
plot(x = train_box$tavg, y = model_3$residuals)
plot(x = train_box$Phase_Factor, y = model_3$residuals)

# Gaussian
qqnorm(model_3$residuals)
qqline(model_3$residuals, col = 'red')
```







Conclusion

I don't really see this project as a failure. I tried exploring if there is a linear relationship that could help predict rain. As our plots showed early on, there was no linear relationship. Even after applying transformations and splitting data the assumptions still didn't uphold.

Future

There is definitely a relationship, and in the future it will be important to eexplore non-linear relationships.