# BikeShare Analysis

Casey Rodriguez

2022-09-26

## Introduction

For the purpose of this presentation we are interested in the factors that contribute to the frequency in which bicycles are rented from a bike share program. The data we will be using is sourced from Kaggle, though few notes are given on the origin of said data. The full source is available at https://www.kaggle.com/datasets/shrutipandit707/bikesharing.

We are interested in determining factors that can predict the total daily ridership of a specific day, given the weather conditions and what day we are preparing for.

The information provided in the dataset is difficult to parse at first glance and we will need to work on cleaning the data before we can begin analysis. A quick preview also reveals that many of the variables are unclear, often leaving discrete entries as numerical when explanatory text would be helpful. One seemingly incorrect detail is the `workingday` variable, later renamed to `is_workday`, which tracks whether or not a given day is a working day. Our assumption was that this indicates weekend days, but that is not the case, given the inconsistency the variable has when compared to the day of the week a specific date would be.

```
##   instant   dteday season yr mnth holiday weekday workingday weathersit
## 1       1 1/1/2018      1  0    1       0       6          0          2
## 2       2 2/1/2018      1  0    1       0       0          0          2
## 3       3 3/1/2018      1  0    1       0       1          1          1
## 4       4 4/1/2018      1  0    1       0       2          1          1
## 5       5 5/1/2018      1  0    1       0       3          1          1
## 6       6 6/1/2018      1  0    1       0       4          1          1
##        temp    atemp     hum windspeed casual registered  cnt
## 1 14.110847 18.18125 80.5833 10.749882    331        654  985
## 2 14.902598 17.68695 69.6087 16.652113    131        670  801
## 3  8.050924  9.47025 43.7273 16.636703    120       1229 1349
## 4       8.2  10.6061 59.0435 10.739832    108       1454 1562
## 5  9.305237  11.4635 43.6957   12.5223     82       1518 1600
## 6  8.378268 11.66045 51.8261 6.0008684     88       1518 1606
```

## Data Cleaning

Several modifications to the data frame would benefit readability and improve foundation work later. Important changes include renaming the columns, unifying the date formatting, replacing the numbers in discrete variables, and truncating the digits for temperature, humidity, and wind speed.

```
# Rename each column
df <- df %>%
    rename(id           = instant,
```

```r
         date         = dteday,
         season       = season,
         year         = yr,
         month        = mnth,
         is_holiday   = holiday,
         day_of_week  = weekday,
         is_workday   = workingday,
         weather      = weathersit,
         temperature  = temp,
         ambient_temp = atemp,
         humidity     = hum,
         wind_speed   = windspeed,
         casual_rides = casual,
         return_rides = registered,
         total_rides  = cnt) %>%
  # Modify column data
  mutate(# Unify date format
         date = lubridate::parse_date_time(date, orders = c('dmy')),
         # Convert discrete numbers to strings
         month        = month.name[month],
         year         = case_when(year == 0 ~ "2018",
                                  year == 1 ~ "2019"),
         season       = case_when(season == 1 ~ "Spring",
                                  season == 2 ~ "Summer",
                                  season == 3 ~ "Autumn",
                                  season == 4 ~ "Winter"),
         weather      = case_when(weather == 1 ~ "Clear",
                                  weather == 2 ~ "Overcast",
                                  weather == 3 ~ "Storm"),
         day_of_week  = wday(date, week_start=2,label=TRUE),
         is_holiday   = as.logical(is_holiday),
         is_workday   = as.logical(is_workday),
         # Vars temperature, ambient_temp, humidity, and wind_speed are characters
         # Convert them to doubles for use as continuous variables
         temperature  = as.double(temperature),
         ambient_temp = as.double(ambient_temp),
         humidity     = as.double(humidity),
         wind_speed   = as.double(wind_speed),
         # Convert relevant columns to factors
         year         = fct_relevel(year, "2018", "2019"),
         month        = fct_relevel(month, "January", "February", "March",
                                    "April", "May", "June", "July",
                                    "August", "September", "October",
                                    "November", "December"),
         season       = fct_relevel(season, "Winter", "Spring", "Summer", "Autumn"),
         weather      = fct_relevel(weather, "Clear", "Overcast", "Storm"),
         day_of_week  = fct_relevel(day_of_week, "Mon", "Tue", "Wed", "Thu",
                                          "Fri", "Sat", "Sun"))

# Finally, preview the new data
head(df)
```

```
##   id       date season year   month is_holiday day_of_week is_workday  weather
```

```
## 1  1 2018-01-01 Spring 2018 January      FALSE         Mon      FALSE Overcast
## 2  2 2018-01-02 Spring 2018 January      FALSE         Tue      FALSE Overcast
## 3  3 2018-01-03 Spring 2018 January      FALSE         Wed       TRUE    Clear
## 4  4 2018-01-04 Spring 2018 January      FALSE         Thu       TRUE    Clear
## 5  5 2018-01-05 Spring 2018 January      FALSE         Fri       TRUE    Clear
## 6  6 2018-01-06 Spring 2018 January      FALSE         Sat       TRUE    Clear
##   temperature ambient_temp humidity wind_speed casual_rides return_rides
## 1   14.110847     18.18125  80.5833  10.749882          331          654
## 2   14.902598     17.68695  69.6087  16.652113          131          670
## 3    8.050924      9.47025  43.7273  16.636703          120         1229
## 4    8.200000     10.60610  59.0435  10.739832          108         1454
## 5    9.305237     11.46350  43.6957  12.522300           82         1518
## 6    8.378268     11.66045  51.8261   6.000868           88         1518
##   total_rides
## 1         985
## 2         801
## 3        1349
## 4        1562
## 5        1600
## 6        1606
```

### Column Description

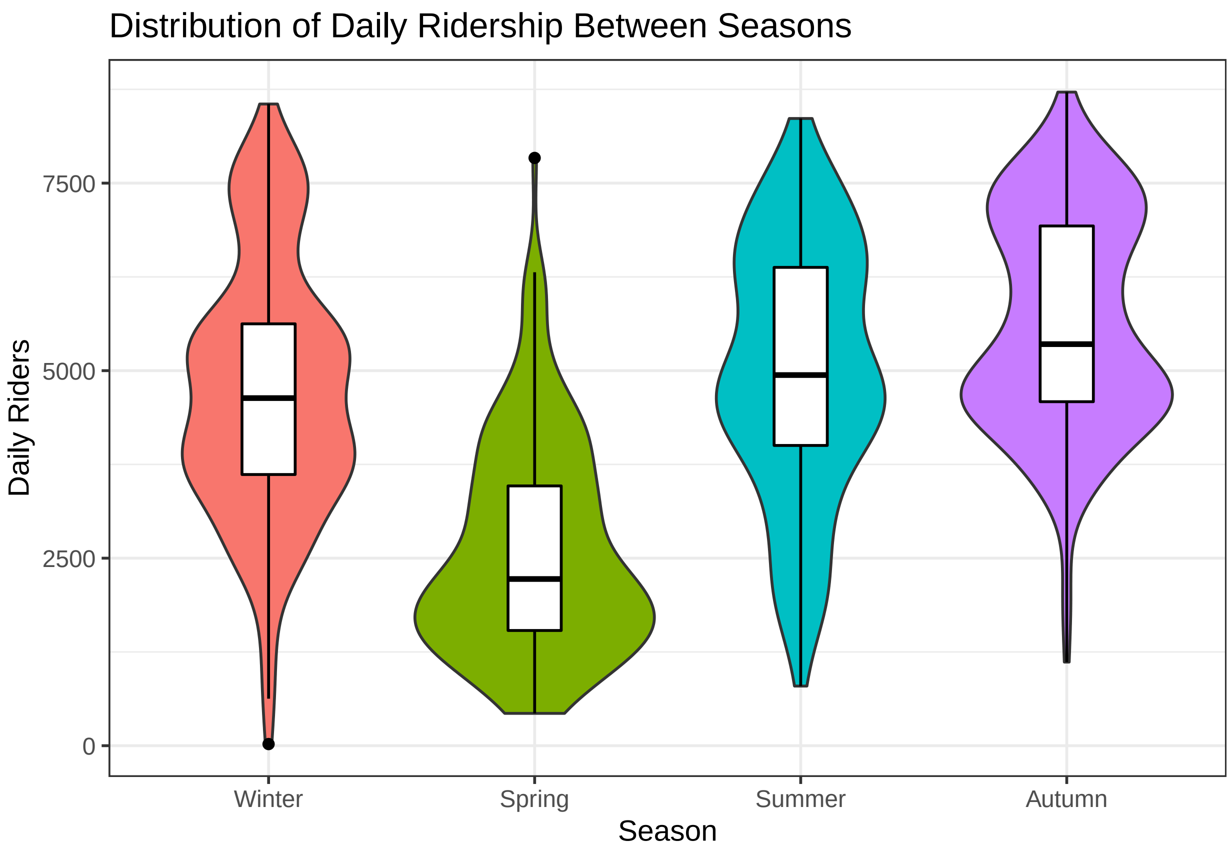
| Variables | Description |
|---|---|
| id | A unique daily identifier |
| date | The date data was collected |
| season | The current season |
| year | The current year |
| month | The current month |
| is_holiday | If the day was a holiday |
| day_of_week | The day of the week |
| is_workday | If the day was a workday |
| weather | The weather conditions of a given day |
| temperature | Reported daily temperature throughout the day |
| ambient_temp | The ambient temperature throughout the day |
| humidity | The humidity throughout the day |
| wind_speed | The wind speed throughout the day |
| casual_rides | How many non-registered users rented a bike |
| return_rides | How many registered users rented a bike |
| total_rides | Count of times a bike was rented on a given day |

### Data Exploration

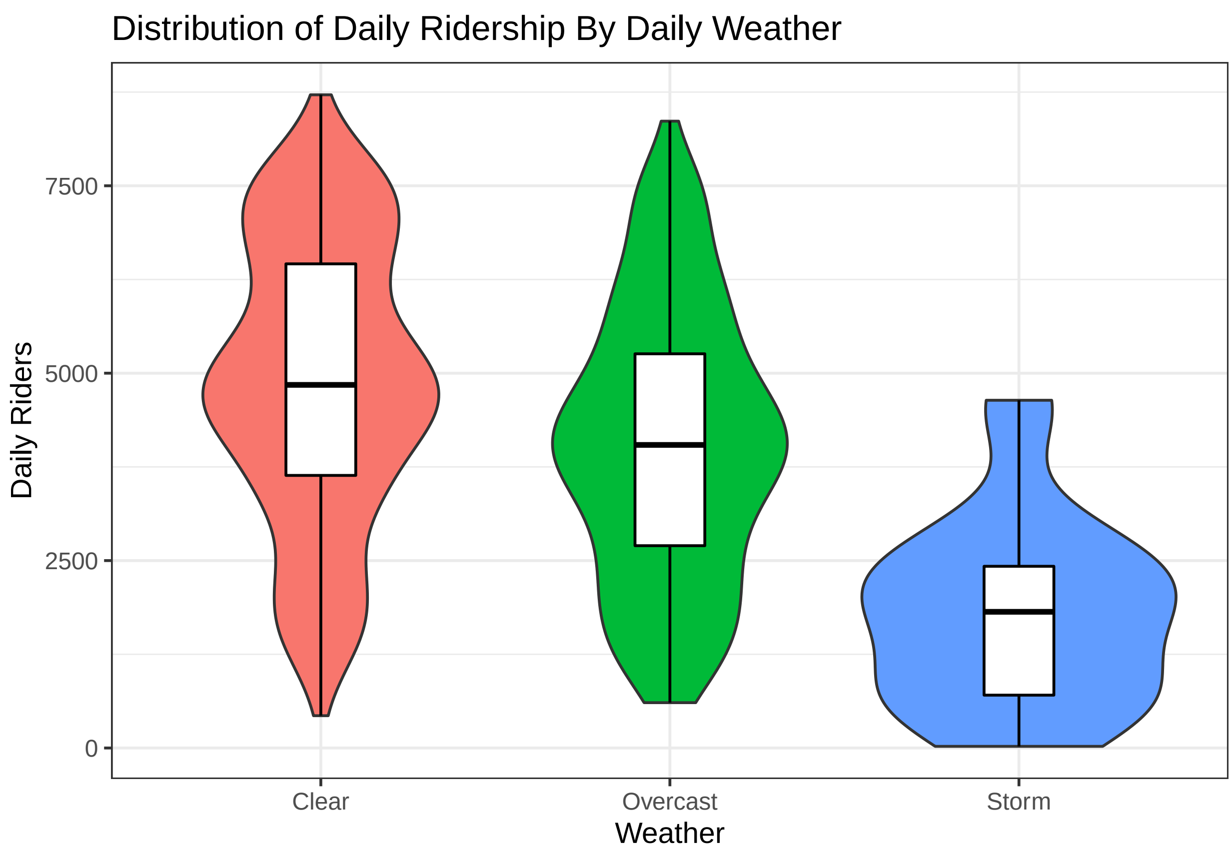
Some quick plots of ride counts across several variables can give us a quick glance at which variables are worth investigating as good predictors of `total_rides`.

#### Categorical Data

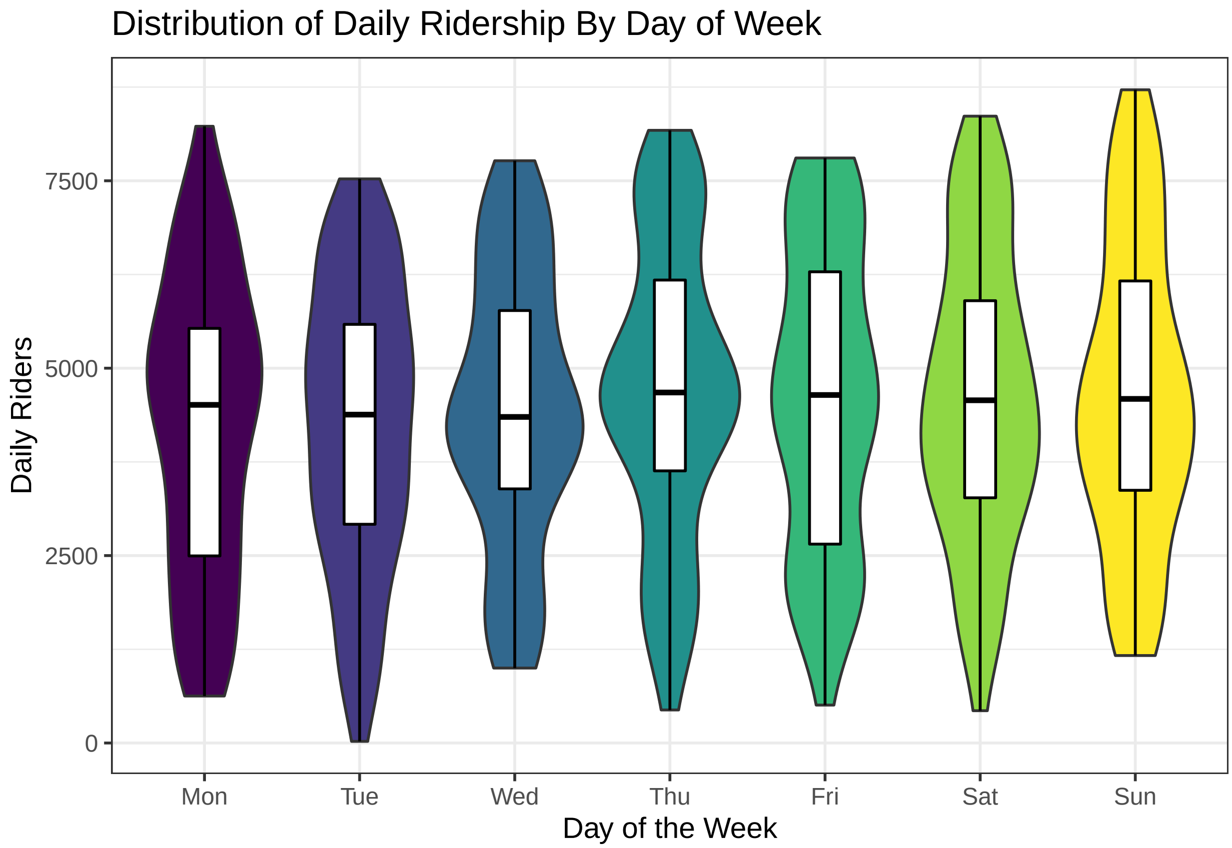
```r
df %>%
    ggplot(aes(x = season, y = total_rides, fill = season)) +
    geom_violin() +
    geom_boxplot(width=0.2, color="black", fill="white") +
    labs(x = "Season",
         y = "Daily Riders",
         title = "Distribution of Daily Ridership Between Seasons") +
    theme_bw() +
    theme(legend.position = "none")
```



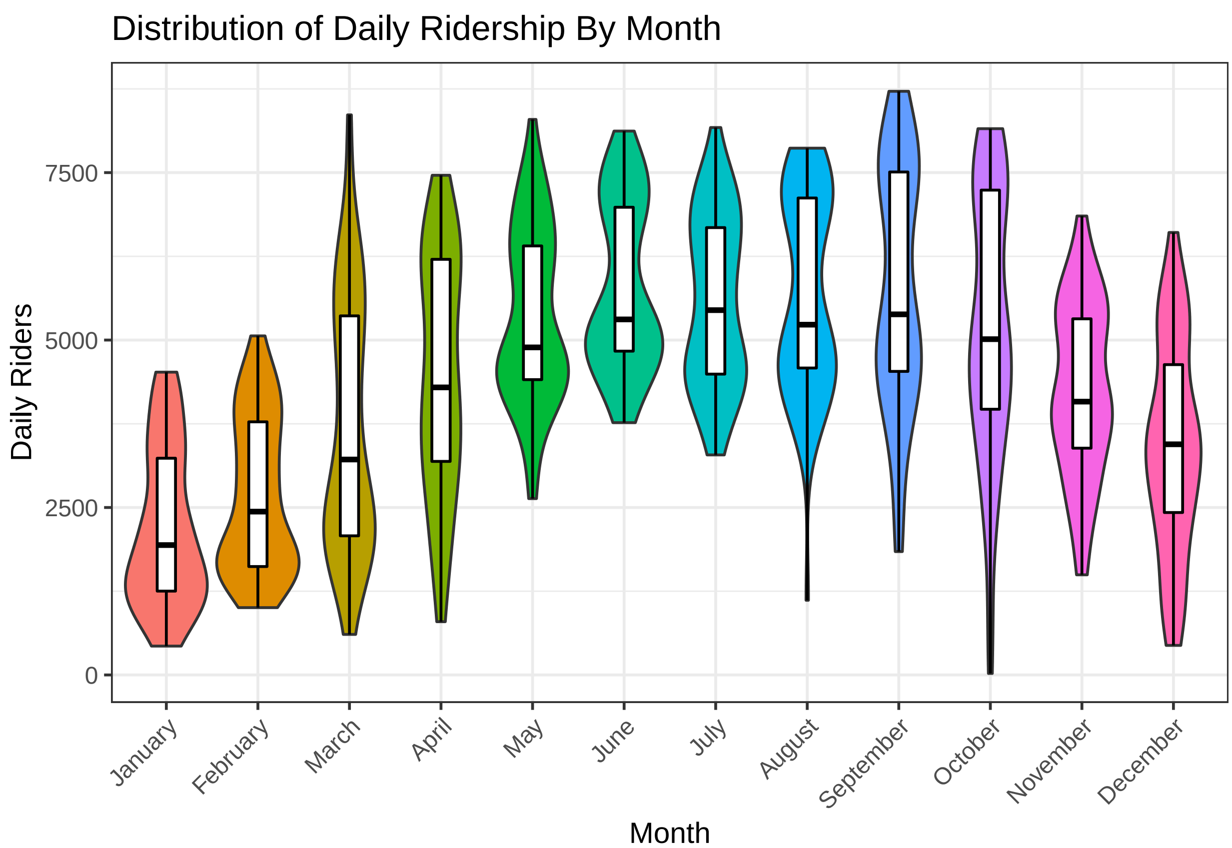

```r
df %>%
    ggplot(aes(x = weather, y = total_rides, fill = weather)) +
    geom_violin() +
    geom_boxplot(width=0.2, color="black", fill="white") +
    labs(x = "Weather",
         y = "Daily Riders",
         title = "Distribution of Daily Ridership By Daily Weather") +
    theme_bw() +
    theme(legend.position = "none")
```




```
df %>%
    ggplot(aes(x = day_of_week, y = total_rides, fill = day_of_week)) +
    geom_violin() +
    geom_boxplot(width=0.2, color="black", fill="white") +
    labs(x = "Day of the Week",
         y = "Daily Riders",
         title = "Distribution of Daily Ridership By Day of Week") +
    theme_bw() +
    theme(legend.position = "none")
```




```
df %>%
    ggplot(aes(x = month, y = total_rides, fill = month)) +
    geom_violin() +
    geom_boxplot(width=0.2, color="black", fill="white") +
    labs(x = "Month",
         y = "Daily Riders",
         title = "Distribution of Daily Ridership By Month") +
    theme_bw() +
    theme(legend.position = "none", axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```




```
ds <- df %>%
    select(season, month, is_holiday, day_of_week, is_workday, weather)
sapply(ds, function(x) table(x)/nrow(ds)*100)
```

```
## $season
## x
##   Winter   Spring   Summer   Autumn
## 24.38356 24.65753 25.20548 25.75342
##
## $month
## x
##   January February     March     April       May      June      July    August
##  8.493151  7.671233  8.493151  8.219178  8.493151  8.219178  8.493151  8.493151
## September   October  November  December
##  8.219178  8.493151  8.219178  8.493151
##
## $is_holiday
## x
##     FALSE      TRUE
## 97.123288  2.876712
##
## $day_of_week
## x
##      Mon      Tue      Wed      Thu      Fri      Sat      Sun
## 14.38356 14.38356 14.24658 14.24658 14.24658 14.24658 14.24658
```

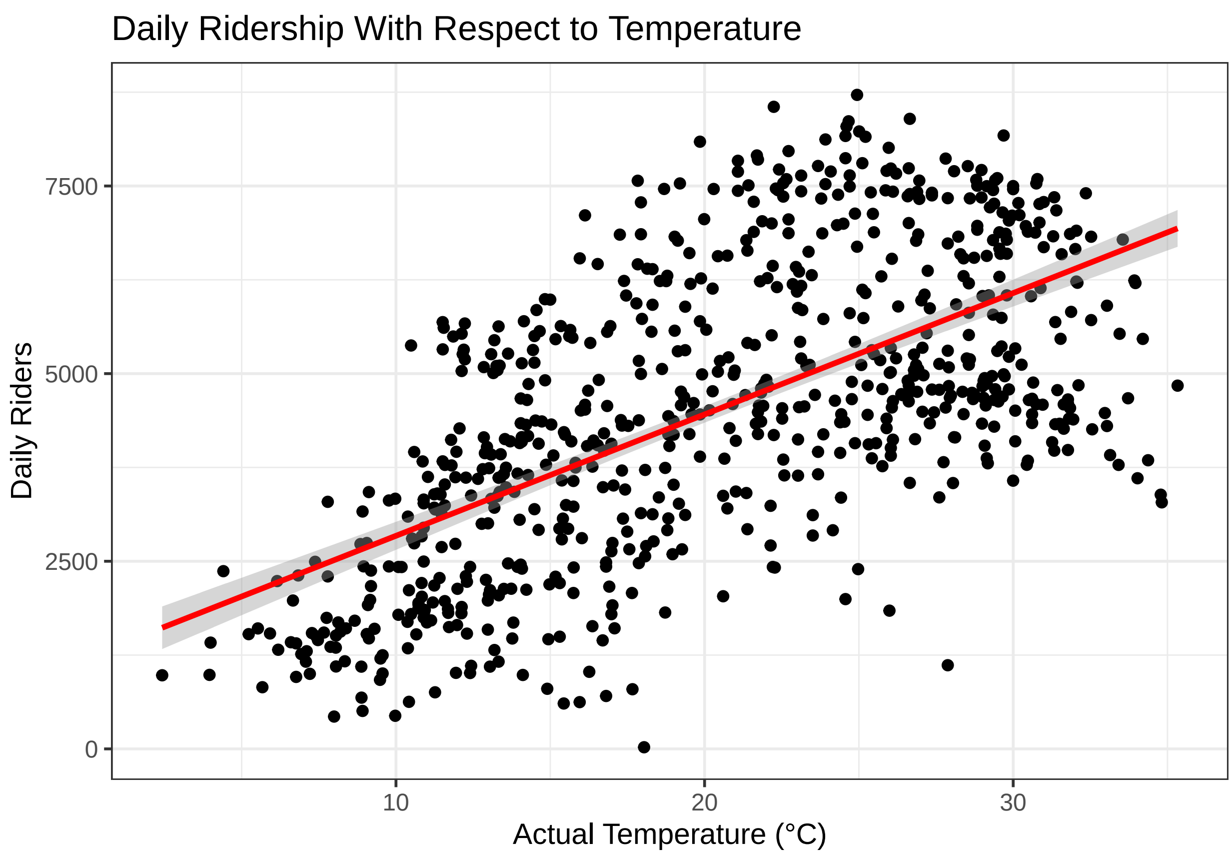
```
##
## $is_workday
## x
##    FALSE     TRUE
## 31.64384 68.35616
##
## $weather
## x
##     Clear  Overcast     Storm
## 63.424658 33.698630  2.876712
```

Categorical variables out of the way, the vast majority (97%) of our data points take place on non-holiday days, making it unlikely to be a good predictor for future total ride counts. However, we still need to explore our numerical variables.

### Continuous Data

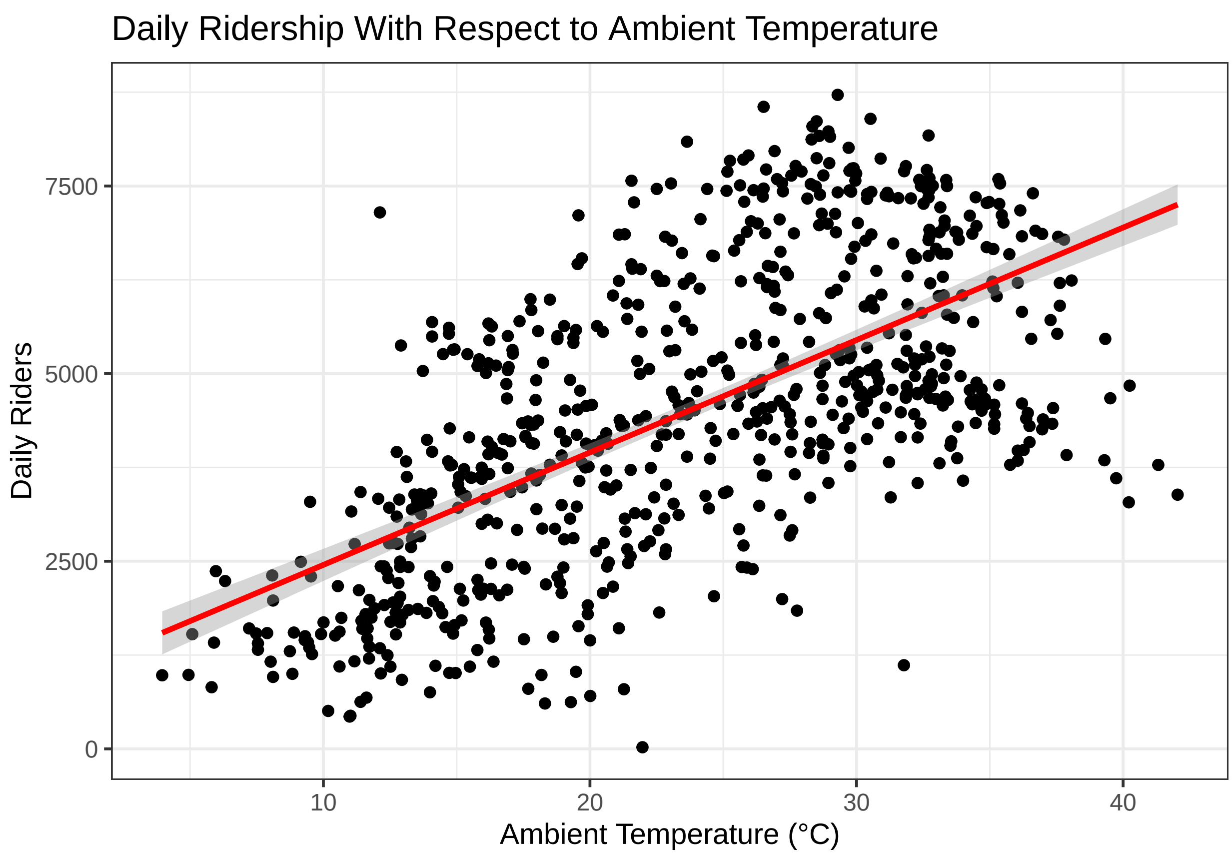
```
df %>%
    ggplot(aes(x = temperature, y = total_rides)) +
    geom_point() +
    geom_smooth(method = "lm", se = TRUE, color = "red") +
    labs(x = "Actual Temperature (°C)",
         y = "Daily Riders",
         title = "Daily Ridership With Respect to Temperature") +
    theme_bw()
```

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



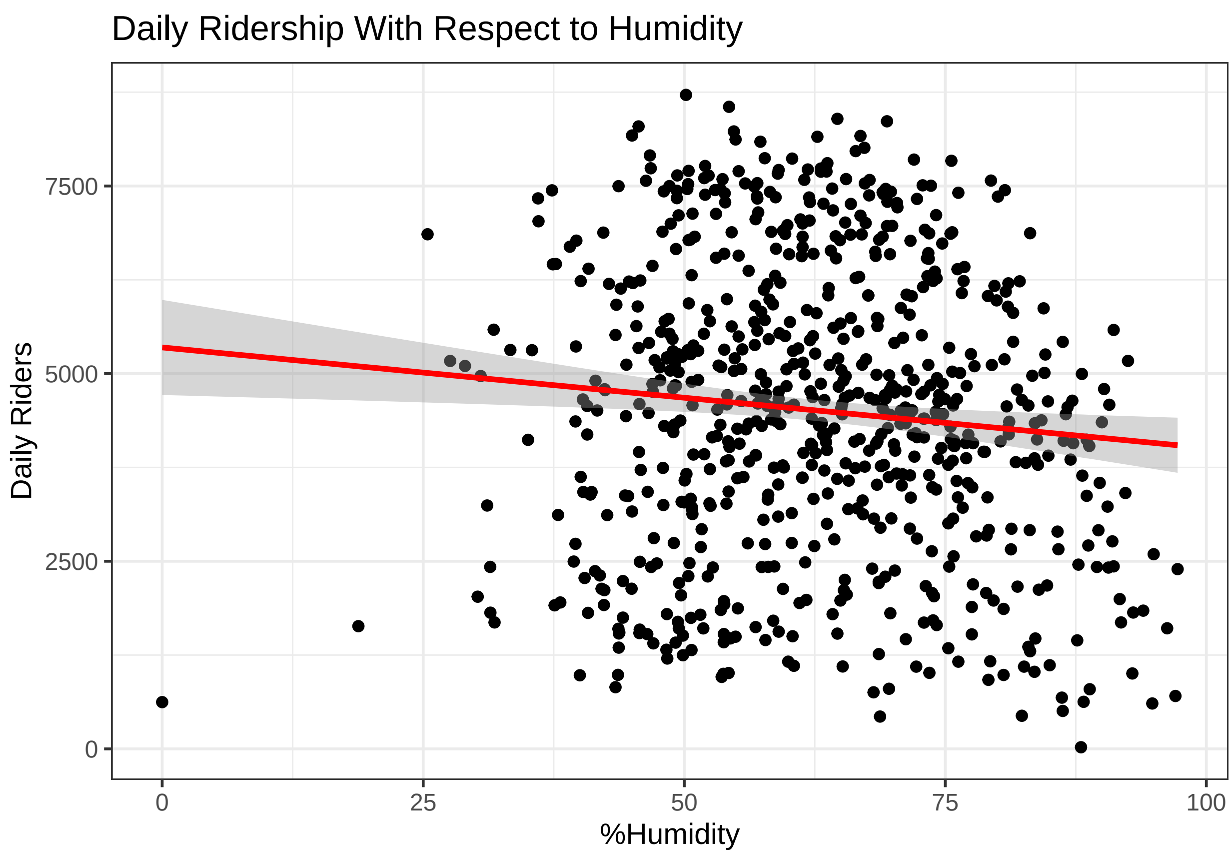

```
df %>%
    ggplot(aes(x = ambient_temp, y = total_rides)) +
    geom_point() +
    geom_smooth(method = "lm", se = TRUE, color = "red") +
    labs(x = "Ambient Temperature (°C)",
         y = "Daily Riders",
         title = "Daily Ridership With Respect to Ambient Temperature") +
    theme_bw()
```

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



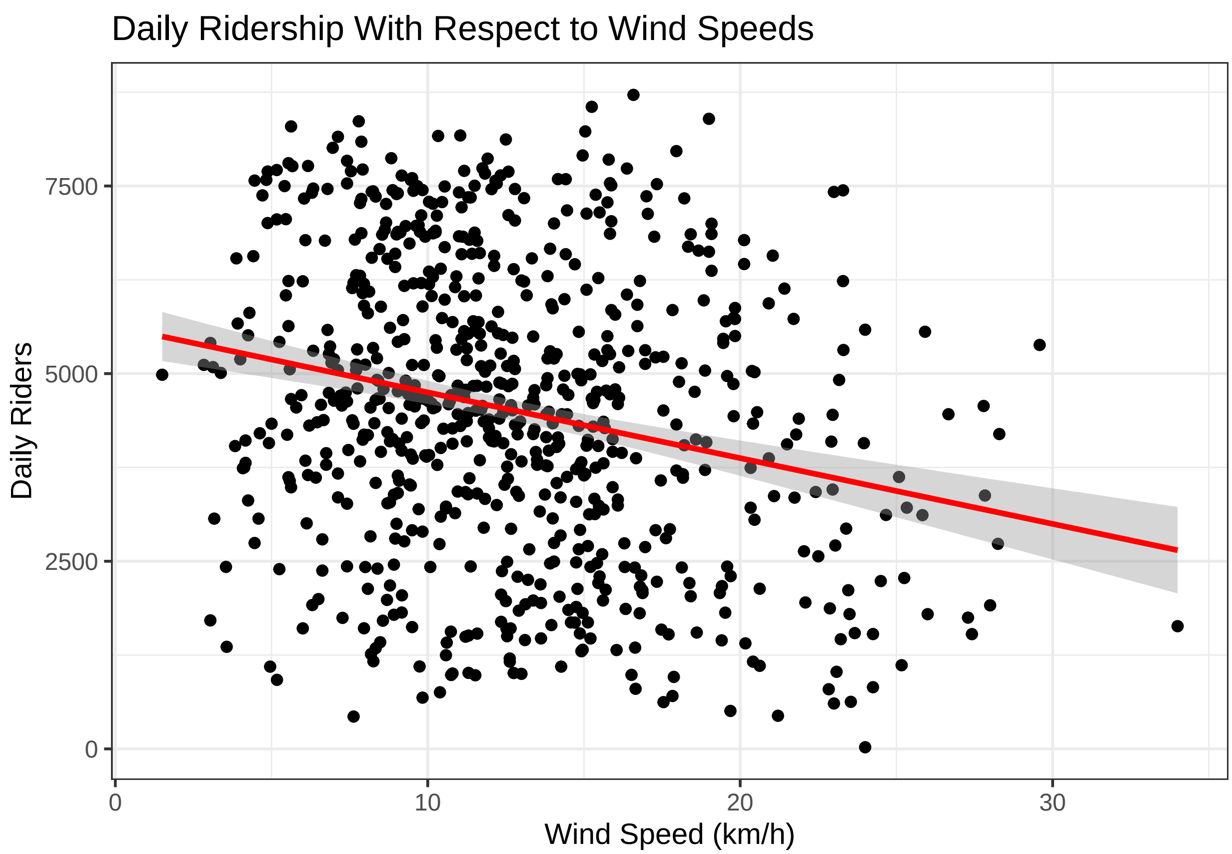

```
df %>%
    ggplot(aes(x = humidity, y = total_rides)) +
    geom_point() +
    geom_smooth(method = "lm", se = TRUE, color = "red") +
    labs(x = "%Humidity",
         y = "Daily Riders",
         title = "Daily Ridership With Respect to Humidity") +
    theme_bw()
```

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




```
df %>%
    ggplot(aes(x = wind_speed, y = total_rides)) +
    geom_point() +
    geom_smooth(method = "lm", se = TRUE, color = "red") +
    labs(x = "Wind Speed (km/h)",
         y = "Daily Riders",
         title = "Daily Ridership With Respect to Wind Speeds") +
    theme_bw()
```

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




From our plotting of the daily rider total against our numerical data, we see correlation between both actual and ambient temperatures and total daily rides. By combining our findings from our categorical and continuous exploration, we can start to look at building our first model.

## Making a Model

Making a model is easy. Making a good model takes time. Sometimes the simplest of models is enough to make a sufficient prediction, taking a single variable and gauging how well it can predict a second variable. Let's see how simple models fare at calculating daily ride totals.

```
season_rides = lm(total_rides ~ season, data=df)
summary(season_rides)
```

```
##
## Call:
## lm(formula = total_rides ~ season, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4706.2 -1067.6  -183.4  1219.4  5227.6
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4728.2      117.6  40.193  < 2e-16 ***
## seasonSpring   -2119.8      165.9 -12.777  < 2e-16 ***
```

```
## seasonSummer    264.2      165.0   1.601     0.11
## seasonAutumn    916.1      164.1   5.582 3.37e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1569 on 726 degrees of freedom
## Multiple R-squared:  0.3455, Adjusted R-squared:  0.3428
## F-statistic: 127.7 on 3 and 726 DF,  p-value: < 2.2e-16
```

```r
rain_rides = lm(total_rides ~ weather, data=df)
summary(rain_rides)
```

```
##
## Call:
## lm(formula = total_rides ~ weather, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4445.8 -1250.3   -13.8  1398.2  4317.2
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        4876.79      85.54  57.011  < 2e-16 ***
## weatherOvercast    -831.97     145.22  -5.729 1.48e-08 ***
## weatherStorm      -3073.50     410.67  -7.484 2.09e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1841 on 727 degrees of freedom
## Multiple R-squared:  0.09858,    Adjusted R-squared:  0.0961
## F-statistic: 39.75 on 2 and 727 DF,  p-value: < 2.2e-16
```

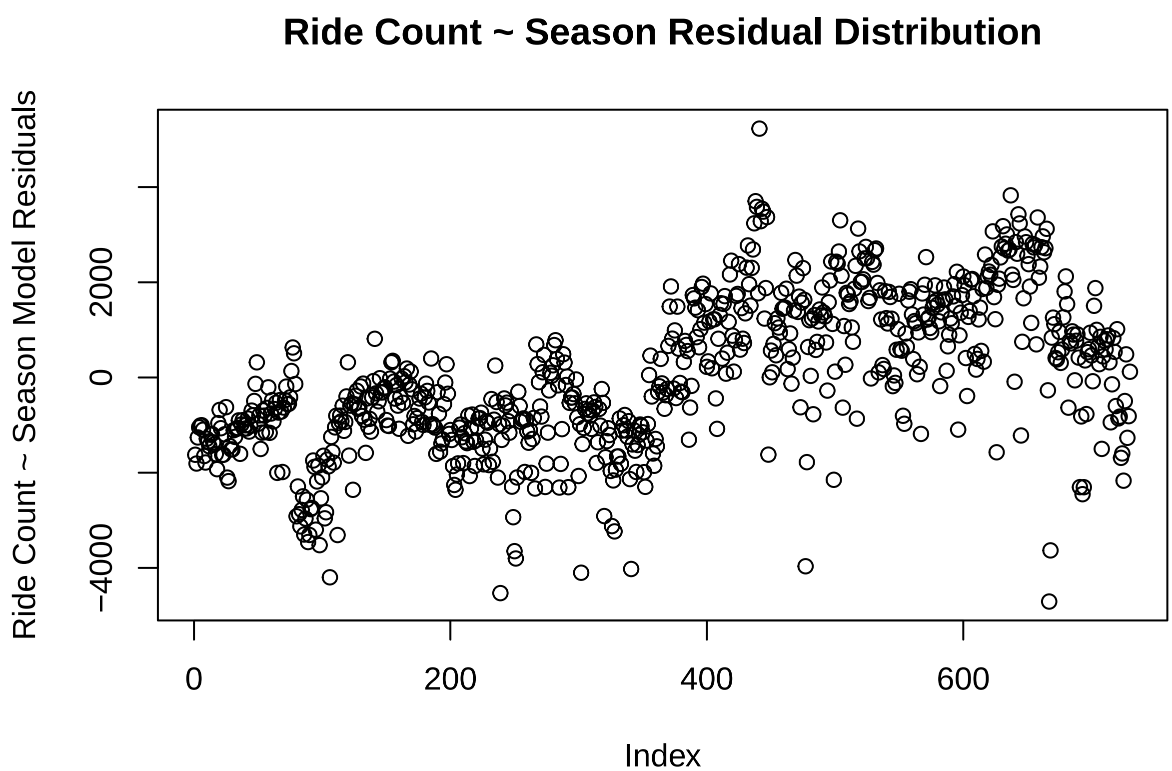
```r
month_rides = lm(total_rides ~ month, data = df)
summary(month_rides)
```

```
##
## Call:
## lm(formula = total_rides ~ month, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5177.2 -1098.5  -242.1  1293.7  4669.7
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        2176.3      193.6  11.241  < 2e-16 ***
## monthFebruary       493.6      281.0   1.756   0.0794 .
## monthMarch         1515.9      273.8   5.537 4.33e-08 ***
## monthApril         2308.6      276.1   8.362 3.20e-16 ***
## monthMay           3173.4      273.8  11.590  < 2e-16 ***
## monthJune          3596.0      276.1  13.025  < 2e-16 ***
## monthJuly          3387.3      273.8  12.371  < 2e-16 ***
## monthAugust        3488.1      273.8  12.739  < 2e-16 ***
```

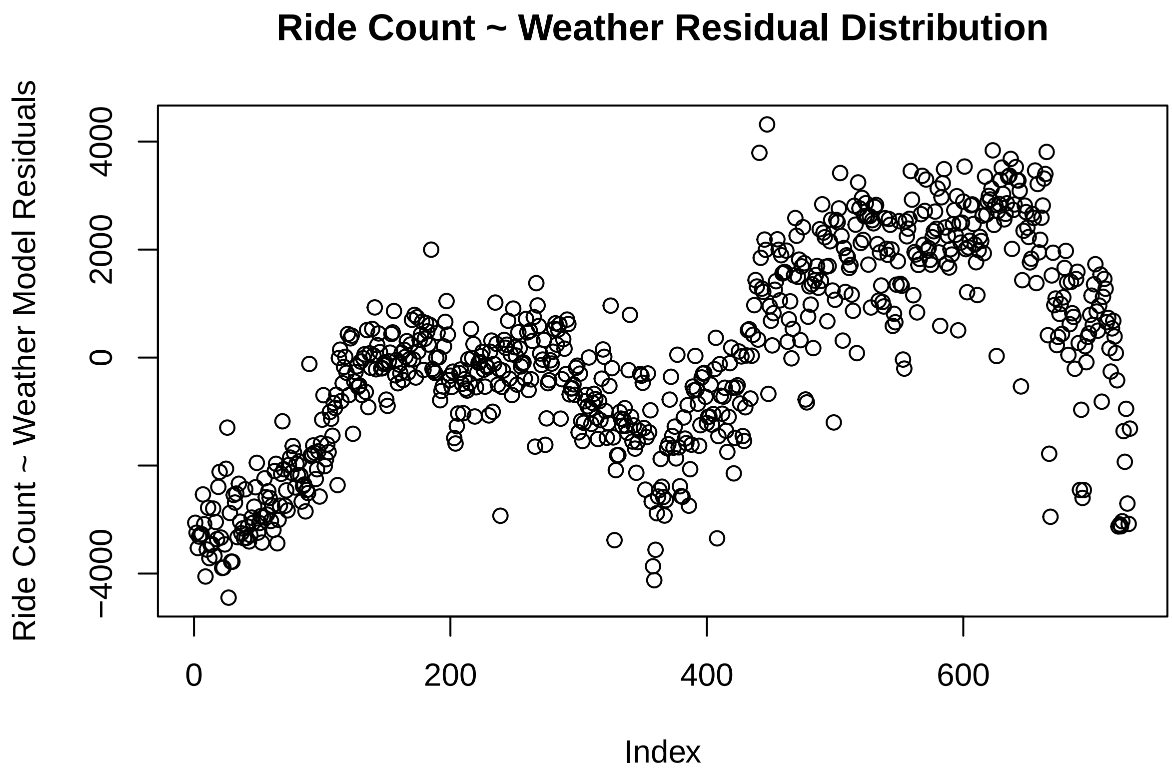
```
## monthSeptember   3590.2       276.1  13.004  < 2e-16 ***
## monthOctober     3022.9       273.8  11.040  < 2e-16 ***
## monthNovember    2070.8       276.1   7.501 1.88e-13 ***
## monthDecember    1227.5       273.8   4.483 8.56e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1524 on 718 degrees of freedom
## Multiple R-squared:  0.3893, Adjusted R-squared:   0.38
## F-statistic: 41.61 on 11 and 718 DF,  p-value: < 2.2e-16
```

While the single, categorical models look promising, running a sanity check by looking for a random distribution of residuals tells a different story.

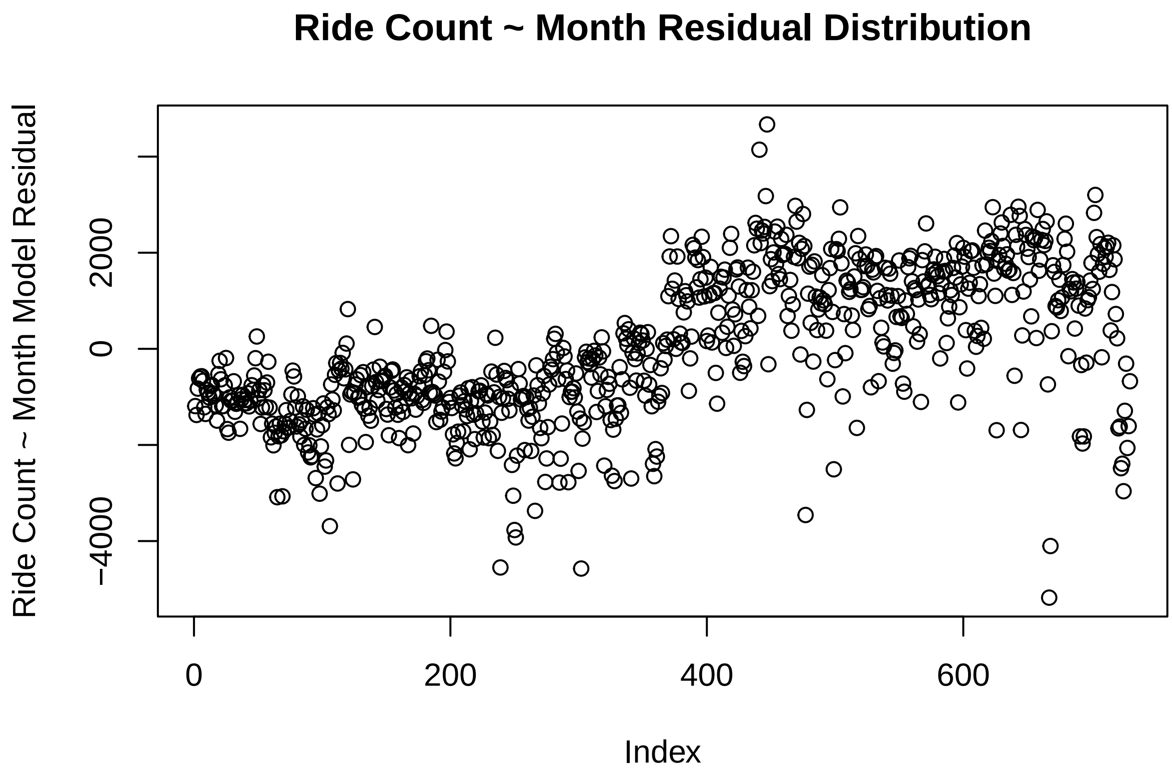
```
plot(season_rides$residuals, ylab = "Ride Count ~ Season Model Residuals",
     main="Ride Count ~ Season Residual Distribution")
```




```
plot(rain_rides$residuals, ylab = "Ride Count ~ Weather Model Residuals",
     main="Ride Count ~ Weather Residual Distribution")
```




```
plot(month_rides$residuals, ylab = "Ride Count ~ Month Model Residual",
     main="Ride Count ~ Month Residual Distribution")
```




The plots of each model's residuals have clear patterns, indicating that a more complex model is warranted. Given the jump in the middle of the chart, `year` might be more relevant than initially thought. Additionally, each variable tested above might have unexplored correlation.

Before moving on to multivariate linear regression, modeling how our continuous variables match to daily rides can give more indication on what factors correlate.

```
temp_rides = lm(total_rides ~ temperature, data = df)
summary(temp_rides)
```

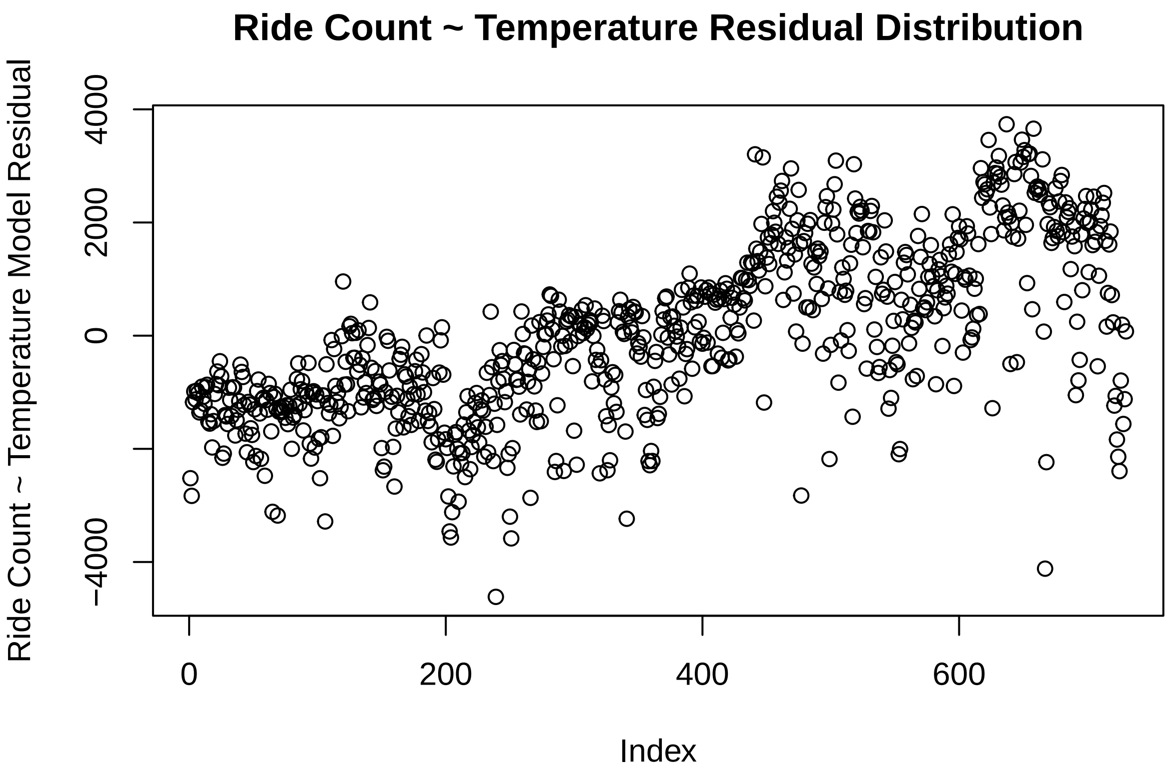
```
##
## Call:
## lm(formula = total_rides ~ temperature, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4615.7 -1136.2   -98.7  1047.0  3736.0
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1222.040    161.278   7.577 1.07e-13 ***
## temperature  161.717      7.446  21.719  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1509 on 728 degrees of freedom
```

```
## Multiple R-squared:  0.3932, Adjusted R-squared:  0.3924
## F-statistic: 471.7 on 1 and 728 DF,  p-value: < 2.2e-16
```

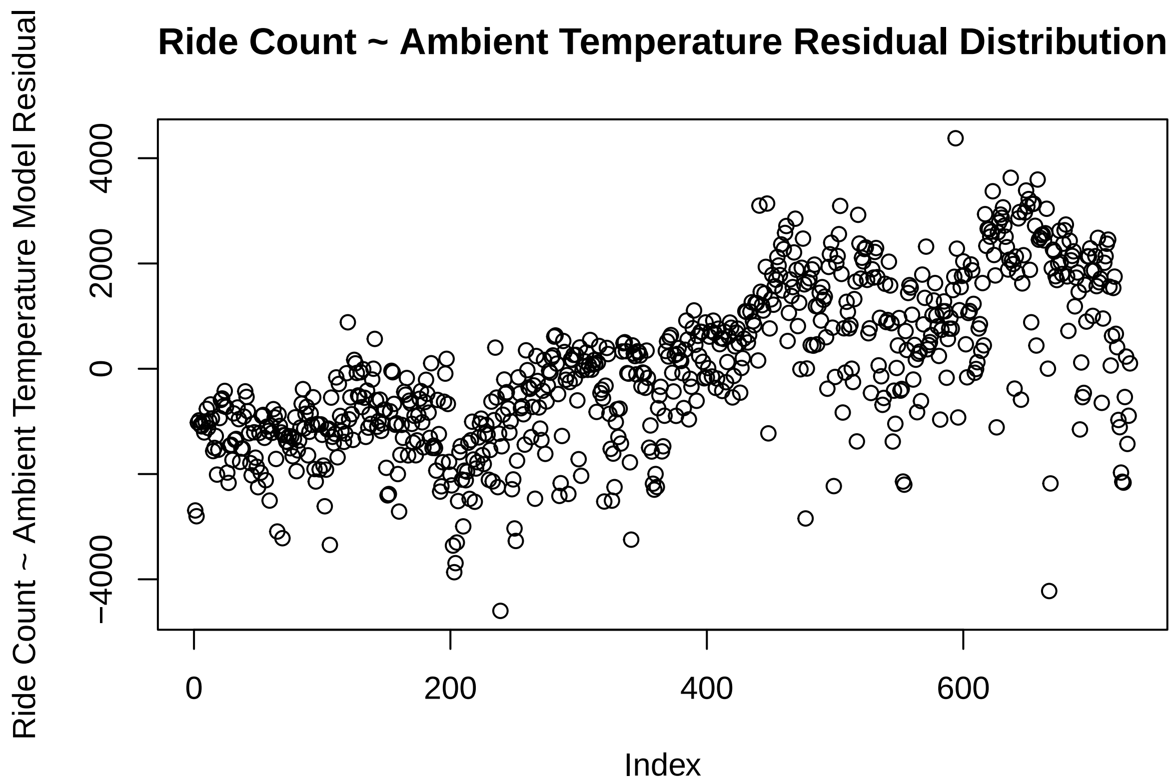
```
ambient_rides = lm(total_rides ~ ambient_temp, data = df)
summary(ambient_rides)
```

```
##
## Call:
## lm(formula = total_rides ~ ambient_temp, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4599.2 -1092.2   -92.8  1073.7  4378.8
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   953.518    171.385   5.564 3.71e-08 ***
## ambient_temp  149.812      6.832  21.928  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1503 on 728 degrees of freedom
## Multiple R-squared:  0.3978, Adjusted R-squared:  0.3969
## F-statistic: 480.8 on 1 and 728 DF,  p-value: < 2.2e-16
```

```
plot(temp_rides$residuals, ylab = "Ride Count ~ Temperature Model Residual",
     main="Ride Count ~ Temperature Residual Distribution")
```




```
plot(ambient_rides$residuals, ylab = "Ride Count ~ Ambient Temperature Model Residual",
     main="Ride Count ~ Ambient Temperature Residual Distribution")
```




### Making Better Models

Where single-variable regression fails, multiple-variable regression saves the day. By introducing a more complex model, we both decrease the intuitiveness and increase the accuracy the model provides.

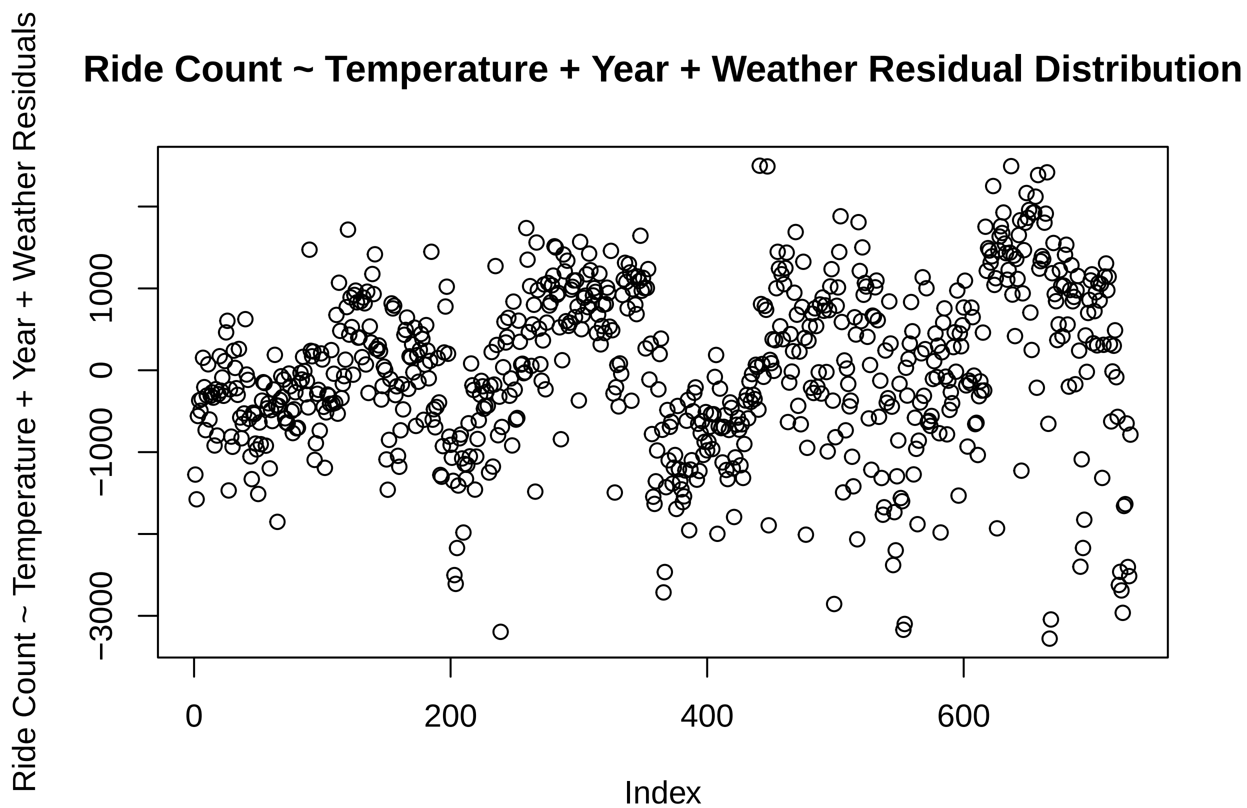
Our first attempt at a multivariate model uses ambient temperatures, the year, and the weather to predict a day's total ride count.

```
multi_rides = lm(total_rides ~ temperature + year + weather, data = df)
summary(multi_rides)
```

```
##
## Call:
## lm(formula = total_rides ~ temperature + year + weather, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3278.3  -621.0   -25.1   779.3  2498.6
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       708.531    120.006   5.904 5.45e-09 ***
## temperature       148.958      5.019  29.680  < 2e-16 ***
## year2019         2039.786     74.972  27.207  < 2e-16 ***
## weatherOvercast  -551.107     80.050  -6.885 1.26e-11 ***
```

```
## weatherStorm    -2135.185    226.275  -9.436  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1009 on 725 degrees of freedom
## Multiple R-squared:  0.7299, Adjusted R-squared:  0.7284
## F-statistic: 489.7 on 4 and 725 DF,  p-value: < 2.2e-16
```

```
plot(multi_rides$residuals, ylab = "Ride Count ~ Temperature + Year + Weather Residuals",
     main = "Ride Count ~ Temperature + Year + Weather Residual Distribution")
```




While the first attempt did produce a more accurate chart, nearly doubling the R-squared values of the previous, single-variable models, it still could be better. In particular, the residual chart seems to indicate a regular, repeating pattern. Including seasonality in our model improves the design once more.

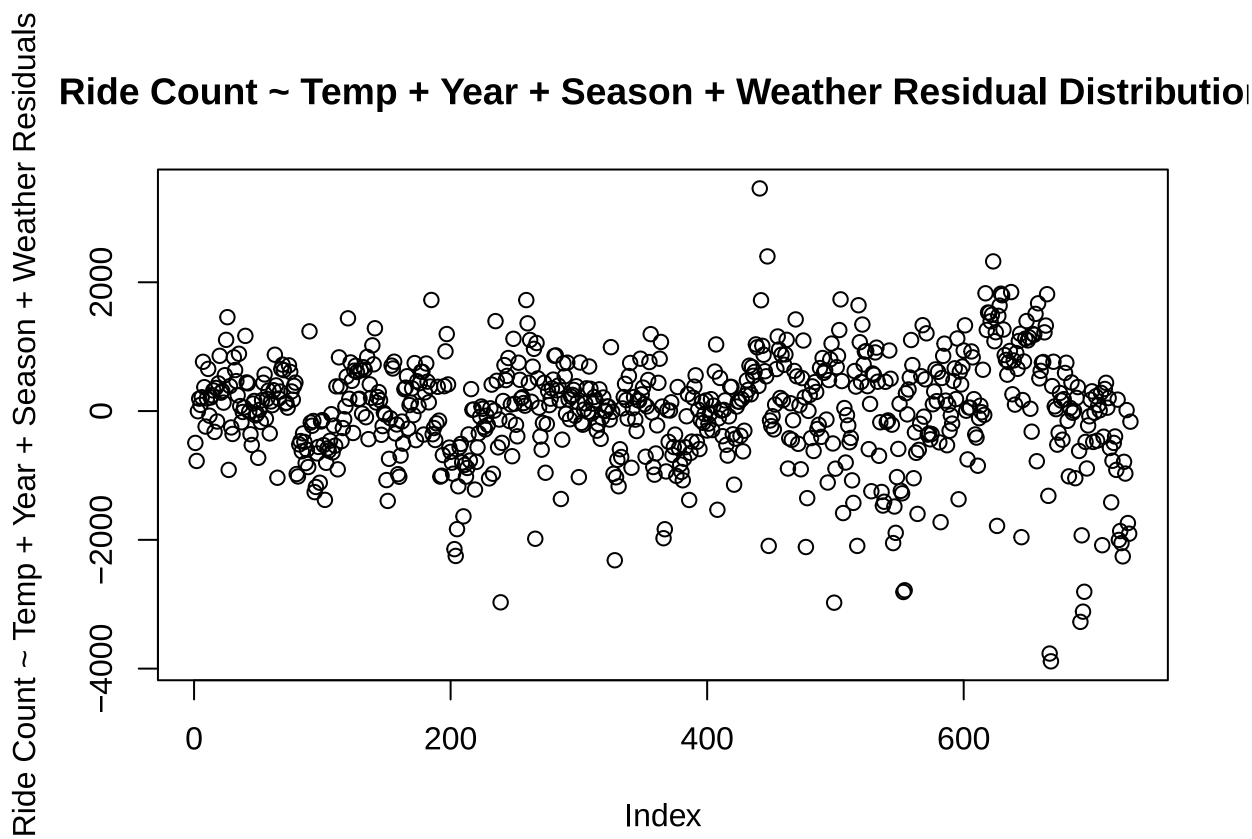
### The Final Model

```
all_rides = lm(total_rides ~ temperature + year + season + weather, data = df)
summary(all_rides)
```

```
##
## Call:
## lm(formula = total_rides ~ temperature + year + season + weather,
```

```
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3886.9  -444.6    87.0   513.2  3457.5
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1954.161    152.787  12.790  < 2e-16 ***
## temperature        121.391      7.614  15.943  < 2e-16 ***
## year2019          2052.907     63.837  32.159  < 2e-16 ***
## seasonSpring     -1579.812     99.081 -15.945  < 2e-16 ***
## seasonSummer      -433.917     97.875  -4.433 1.07e-05 ***
## seasonAutumn      -645.202    125.766  -5.130 3.72e-07 ***
## weatherOvercast   -607.758     68.142  -8.919  < 2e-16 ***
## weatherStorm     -2408.860    193.095 -12.475  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 857 on 722 degrees of freedom
## Multiple R-squared:  0.8059, Adjusted R-squared:  0.804
## F-statistic: 428.3 on 7 and 722 DF,  p-value: < 2.2e-16
```

```r
plot(all_rides$residuals, ylab = "Ride Count ~ Temp + Year + Season + Weather Residuals",
     main = "Ride Count ~ Temp + Year + Season + Weather Residual Distribution")
```



With a residual distribution that lacks a clear pattern, a p-value that is well below the standard 0.05 required for a 95% null-hypothesis rejection certainty, and an R-squared value at approximately 0.8, the new model incorporating the area's temperature and weather alongside the record's year and astronomical season provides a worthwhile prediction for the day's total ride count.

## Final Model Findings

Based on the final model, our top three predictor variables are:

| Variable | Description |
|---|---|
| Temperature | A coefficient value of 121.391 indicates that a unit increase in temperature increases the day's ride count by 121.391 rides. |
| Year | A coefficient value of 2052.907 indicates that, with respect to 2018, the same day in the year 2019 will see an increase in the day's ride count by 2052.907 rides. |
| (Stormy) Weather | A coefficient value of -2408.860 indicates that, with respect to a sunny day, stormy weather will see 2408.860 less daily riders. |

Put in common tongue, a bike rental company can expect more customers the longer they are in business and the warmer the day is. However, they can expect a noticeable drop in customers should the weather forecast predict rain.

Our remaining variables are also worth considering, winter brings in the most members, with respect to spring, summer, and autumn days. Finally, sunny weather sees the most foot traffic, with respect to both overcast and stormy weather days. For planning purposes, the rental company should expect high demands on sunny, warm, winter days, especially if the company has been operating in the area for over a year. On colder, cloudy, spring days newer bike rentals can expect lower ridership numbers.

## Further Analysis

With our model complete, we can predict total ridership for a given date should we know the expected weather conditions in advance. However, there are more details we can explore, leaving questions for further analysis down the line. In future study we would like to investigate how weather conditions affect ridership between members and casual users.