Overview

Background

With increasing urban population, traffic congestion and saturation and/or lack of public transportation bike sharing proved to be an ingenious environment friendly solution for daily commuters. There has been steady increase in the number of bike share programs worldwide reaching 1608 bike share programs with a fleet of 18.2 million bikes in 2018.

Despite the steady growth in bike sharing programs one of the key challenges faced by aggregators is to estimate the demand for bikes and allocate resources accordingly as the usage rates vary from around three to eight trips per bicycle per day globally2. The variation in usage could be due to multitude of factors one of which we believe are the prevalent weather conditions. We can expect that passengers are more likely to choose bike rides on days when the weather is pleasant without snowfall and/or heavy winds. Another important factor is time during the day. The demand is more during morning and evening peak traffic hours, and lesser during other times of the day.

Further, a study carried out by Bowman Cutter and Matthew Neidell's on the effect of voluntary information disclosure of information on air quality urging people to reduce ozone emissions found that there is an increase in people choosing alternate methods of transportation on days such warnings are issued, supporting the idea that weather parameters have an effect on individual's behavior and choices.

Data Description

The response variable is:
Y (Cnt): Total bikes rented by both casual & registered users together

The predicting variables are:
X_1 (Instant): Record index
X_2 (Dteday): Day on which the observation is made
X_3 (Season): Season which the observation is made (1 = Winter, 2 = Spring, 3 = Summer, 4 = Fall)
X_4 (Yr): Year on which the observation is made
X_5 (Mnth): Month on which the observation is made
X_6 (Hr): Day on which the observation is made (0 through 23)
X_7 (Holiday): Indictor of a public holiday or not (1 = public holiday, 0 = not a public holiday)
X_8 (Weekday): Day of week (0 through 6)
X_9 (Working day): Indicator of a working day (1 = working day, 0 = not a working day)
X_{10} (Weathersit): Weather condition (1 = Clear, Few clouds, Partly cloudy, Partly cloudy, 2 = Mist & Cloudy, Mist & Broken clouds, Mist & Few clouds, Mist, 3 = Light Snow, Light Rain, Thunderstorm & Scattered clouds, Light Rain & Scattered clouds, 4 = Heavy Rain, Ice Pallets, Thunderstorm & Mist, Snow & Fog)
X_{11} (Temp): Normalized temperature in Celsius
X_{12} (Atemp): Normalized feeling temperature in Celsius
X_{13} (Hum): Normalized humidity
X_{14} (Windspeed): Normalized wind speed
X_{15} (Casual): Bikes rented by casual users in that hour
X_{16} (Registered): Bikes rented by registered users in that hour

---

Exploratory Data Analysis

Reading data

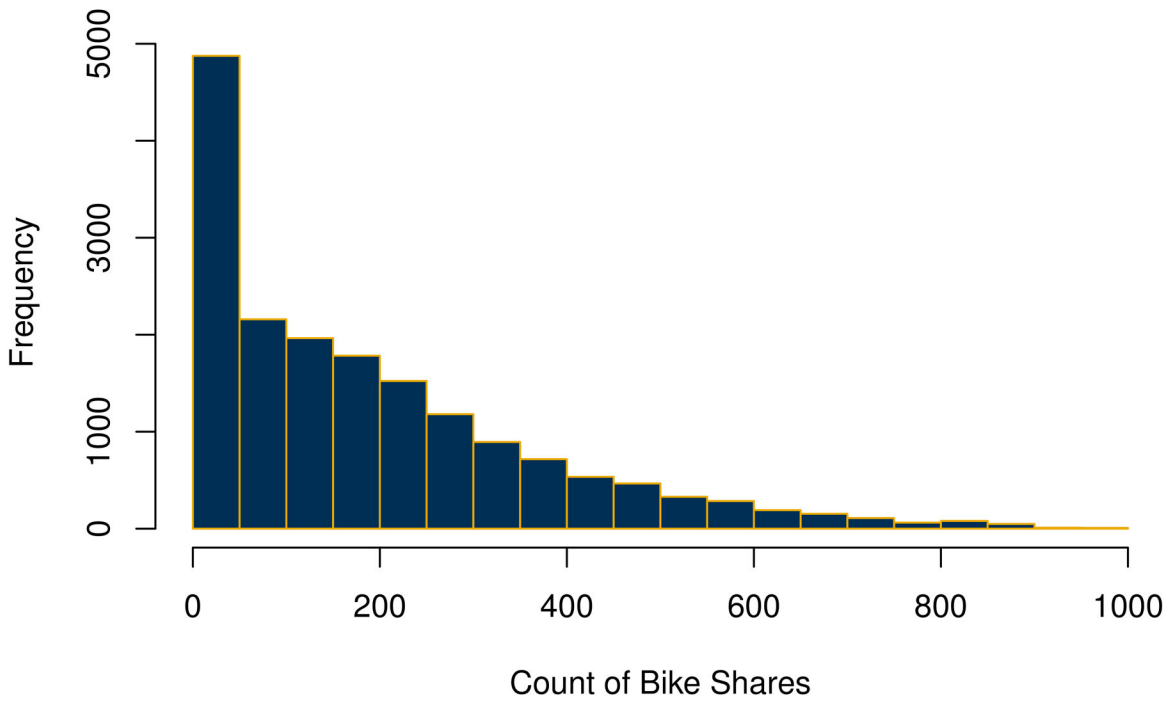
```
# Set colors
gtblue = rgb(0, 48, 87, maxColorValue = 255)
techgold = rgb(179, 163, 105, maxColorValue = 255)
```

```
buzzgold = rgb(234, 170, 0, maxColorValue = 255)
bobbyjones = rgb(55, 113, 23, maxColorValue = 255)
# Read the data using read.csv
data = read.csv("Bikes.csv")
# Show the number of observations
obs = nrow(data)
cat("There are", obs, "observations in the data")
```

```
## There are 17379 observations in the data
```

Response Data Distribution

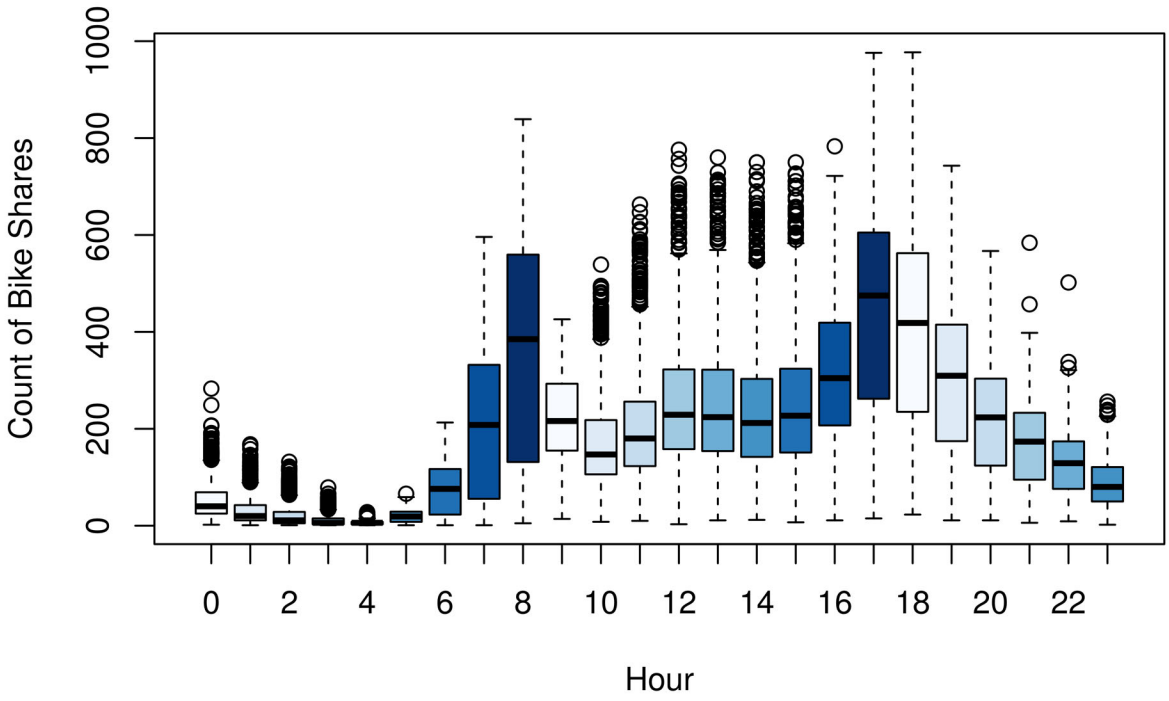
```
# Check the distribution of the response, cnt
hist(data$cnt,
     main="",
     xlab="Count of Bike Shares",
     border=buzzgold,
     col=gtblue)
```



- The frequency of zero bike shares is high, which skews the demand data.

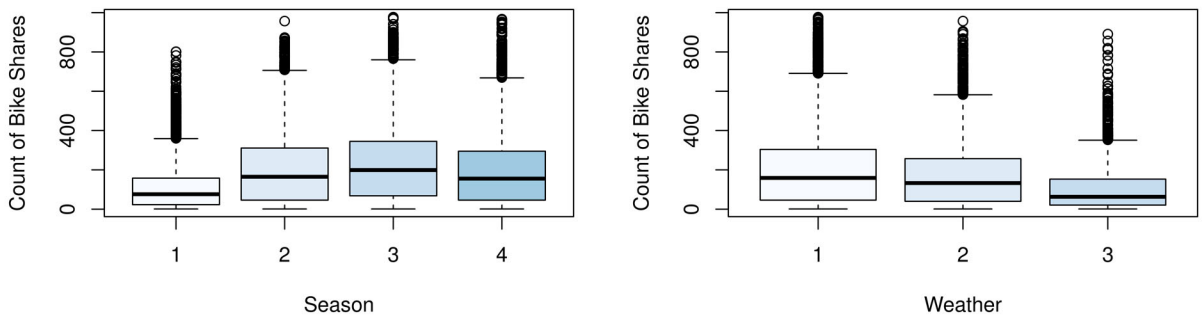
```
# Check the response, cnt, against time of day
boxplot(cnt~hr,
        main="",
        xlab="Hour",
        ylab="Count of Bike Shares",
        col=blues9,
```

```
        data=data)
```



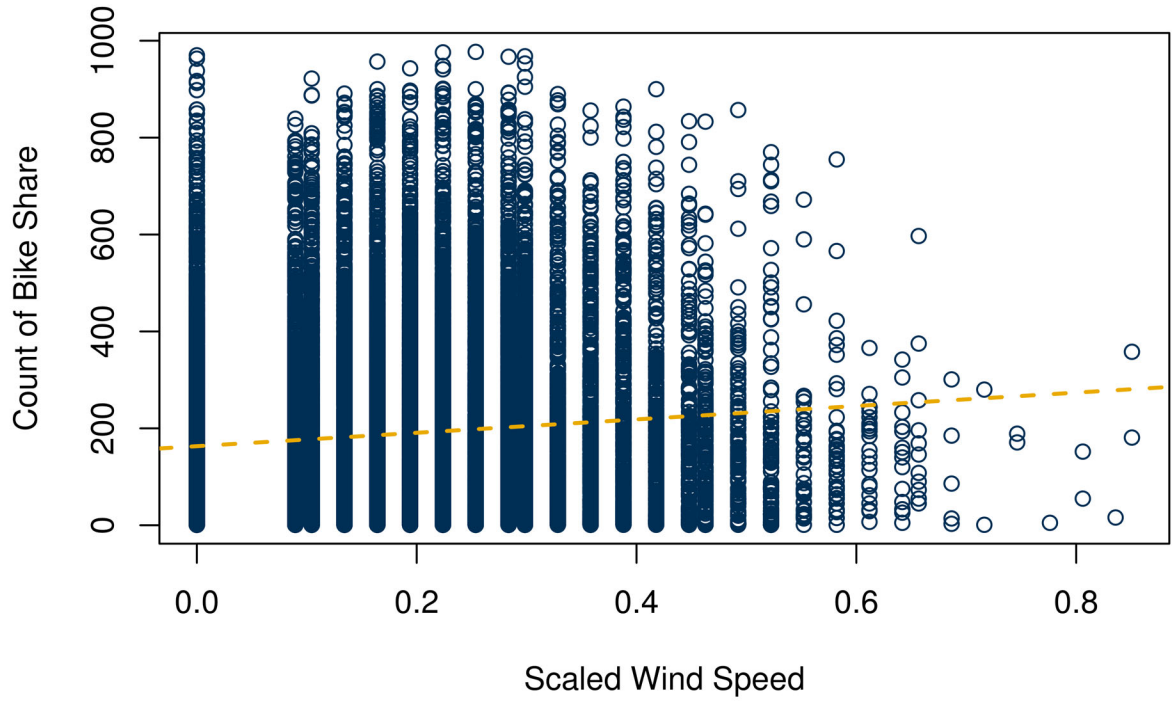
The number of bike shares between hour 0 and hour 6 is low. The majority activity as expected is focused between hour 7 and hour 23, peaking at hour 8 and hour 17.

```
par(mfrow=c(1, 2))

# Plot cnt against season
boxplot(cnt~season,
        main="",
        xlab="Season",
        ylab="Count of Bike Shares",
        col=blues9,
        data=data)

# Plot cnt against weather
boxplot(cnt~weathersit,
        main="",
        xlab="Weather",
        ylab="Count of Bike Shares",
        col=blues9,
        data=data)
```



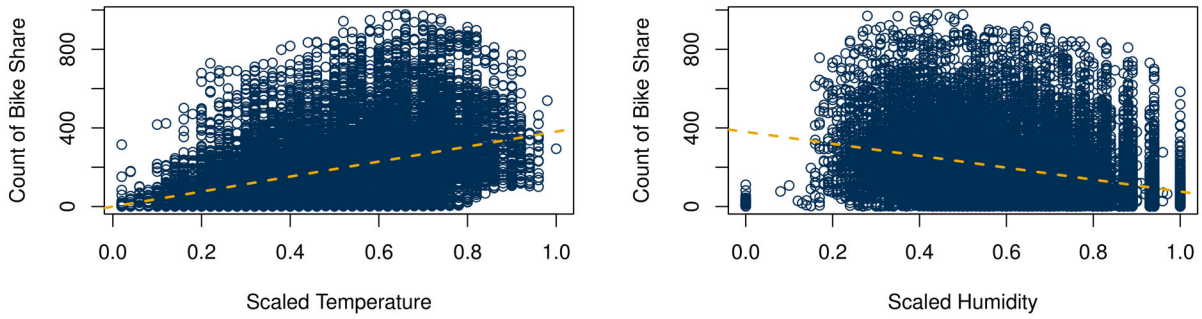
The number of bikes rented during winter are the lowest. The number of bikes decreases as the weather becomes unfavorable.

```
plot(data$windspeed,
     data$cnt,
     xlab="Scaled Wind Speed",
     ylab="Count of Bike Share",
     main="",
     col=gtblue)
abline(lm(cnt~windspeed, data=data),
       col=buzzgold,
       lty=2, lwd=2)
```



The count of rental bikes seems to decrease as windspeed increases. <- ***Need to discuss this as the OLS line contradicts this statement.***

```
par(mfrow=c(1, 2))

plot(data$temp,
     data$cnt,
     xlab="Scaled Temperature",
     ylab="Count of Bike Share",
     main="",
     col=gtblue)
abline(lm(cnt~temp, data=data), col=buzzgold, lty=2, lwd=2)

plot(data$hum,
     data$cnt,
     xlab="Scaled Humidity",
     ylab="Count of Bike Share",
     main="",
     col=gtblue)
abline(lm(cnt~hum, data=data), col=buzzgold, lty=2, lwd=2)
```



The count of rental bikes seems to decrease as humidity increases although the demand varies within similar ranges at varying humidity levels. The count of rental bikes seems to increase as temperature increases however with much wider variability at larger temperature levels.

Preparing the Data

```
# Set a seed for reproducibility
set.seed(9)

# Remove the irrelevant columns
clean_data = data[-c(1,2,9,15,16)]

# Convert the numerical categorical variables to predictors
clean_data$season = as.factor(clean_data$season)
clean_data$yr = as.factor(clean_data$yr)
clean_data$mnth = as.factor(clean_data$mnth)
clean_data$hr = as.factor(clean_data$hr)
clean_data$holiday = as.factor(clean_data$holiday)
```

```r
clean_data$weekday = as.factor(clean_data$weekday)
clean_data$weathersit = as.factor(clean_data$weathersit)

# 80% Train 20% Test split
sample_size = floor(0.8*nrow(clean_data))
picked = sample(seq_len(nrow(clean_data)), size=sample_size)
train = clean_data[picked,]
test = clean_data[-picked,]
```

Creating the Model

```r
# Create a multiple linear regression model
model1 = lm(cnt ~ ., data=train)
summary(model1)
```

```
##
## Call:
## lm(formula = cnt ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -388.07  -61.20   -7.46   51.52  432.04
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -79.4356     7.4390 -10.678  < 2e-16 ***
## season2      34.9268     5.4110   6.455 1.12e-10 ***
## season3      27.0055     6.4438   4.191 2.80e-05 ***
## season4      65.3435     5.4690  11.948  < 2e-16 ***
## yr1          85.3415     1.7487  48.804  < 2e-16 ***
## mnth2         4.1666     4.3853   0.950 0.342060
## mnth3        16.4733     4.9267   3.344 0.000829 ***
## mnth4        12.5834     7.3038   1.723 0.084936 .
## mnth5        26.4616     7.8357   3.377 0.000735 ***
## mnth6        11.5056     8.0535   1.429 0.153131
## mnth7        -7.8872     9.0547  -0.871 0.383736
## mnth8        13.1813     8.8362   1.492 0.135789
## mnth9        39.4297     7.8463   5.025 5.09e-07 ***
## mnth10       20.8547     7.2558   2.874 0.004056 **
## mnth11       -6.9377     6.9849  -0.993 0.320614
## mnth12       -0.3485     5.5593  -0.063 0.950014
## hr1         -20.0931     5.9691  -3.366 0.000764 ***
## hr2         -26.8251     6.0232  -4.454 8.51e-06 ***
## hr3         -39.2407     6.0946  -6.439 1.25e-10 ***
## hr4         -40.0625     6.0560  -6.615 3.84e-11 ***
## hr5         -24.7054     6.0829  -4.061 4.90e-05 ***
## hr6          32.5635     6.0460   5.386 7.32e-08 ***
## hr7         170.0286     5.9759  28.452  < 2e-16 ***
## hr8         302.5048     6.0425  50.063  < 2e-16 ***
## hr9         162.6816     6.0247  27.003  < 2e-16 ***
## hr10        109.1344     6.0158  18.141  < 2e-16 ***
## hr11        132.2292     6.1238  21.593  < 2e-16 ***
```

```
## hr12        169.3083     6.1091  27.714  < 2e-16 ***
## hr13        165.0598     6.1608  26.792  < 2e-16 ***
## hr14        149.5507     6.2557  23.906  < 2e-16 ***
## hr15        158.7465     6.2540  25.383  < 2e-16 ***
## hr16        221.3557     6.2020  35.691  < 2e-16 ***
## hr17        373.5963     6.1865  60.389  < 2e-16 ***
## hr18        346.0519     6.1500  56.268  < 2e-16 ***
## hr19        236.7916     6.1312  38.621  < 2e-16 ***
## hr20        154.5982     6.0567  25.525  < 2e-16 ***
## hr21        105.4905     6.0769  17.359  < 2e-16 ***
## hr22         68.0205     6.0584  11.227  < 2e-16 ***
## hr23         29.2676     5.9982   4.879 1.08e-06 ***
## holiday1    -29.5963     5.3678  -5.514 3.58e-08 ***
## weekday1     10.8324     3.3062   3.276 0.001054 **
## weekday2     10.4986     3.2435   3.237 0.001211 **
## weekday3     13.2372     3.2285   4.100 4.15e-05 ***
## weekday4     12.8713     3.2424   3.970 7.23e-05 ***
## weekday5     18.8042     3.2251   5.831 5.64e-09 ***
## weekday6     15.7637     3.2243   4.889 1.02e-06 ***
## weathersit2 -12.3071     2.1390  -5.754 8.92e-09 ***
## weathersit3 -63.7651     3.6679 -17.385  < 2e-16 ***
## temp        125.6739    32.1120   3.914 9.14e-05 ***
## atemp       115.4957    33.2228   3.476 0.000510 ***
## hum         -86.1635     6.2104 -13.874  < 2e-16 ***
## windspeed   -31.2492     7.8785  -3.966 7.33e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 101.8 on 13851 degrees of freedom
## Multiple R-squared:  0.6852, Adjusted R-squared:  0.684
## F-statistic:   591 on 51 and 13851 DF,  p-value: < 2.2e-16
```

Finding Insignificant Variables

```
which(summary(model1)$coeff[,4]>0.05)
```

```
##  mnth2  mnth4  mnth6  mnth7  mnth8 mnth11 mnth12
##      6      8     10     11     12     15     16
```

```
summary(data$weathersit)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.425   2.000   3.000
```

Coding Dummy Variables

```
# Create dummy variables
weathersit.1 = ifelse(data$weathersit == 1, 1, 0)
weathersit.2 = ifelse(data$weathersit == 2, 1, 0)
weathersit.3 = ifelse(data$weathersit == 3, 1, 0)

# Include all dummy variables without intercept
fit.1 = lm(cnt ~ weathersit.1 + weathersit.2
           + weathersit.3 - 1, data=data)
# Include 2 dummy variables with intercept
```

```r
fit.2 = lm(cnt ~ weathersit.2 + weathersit.3,
           data=data)
# Use categorical variable
fit.3 = lm(cnt ~ weathersit, data=clean_data)

summary(fit.1)
```

```
##
## Call:
## lm(formula = cnt ~ weathersit.1 + weathersit.2 + weathersit.3 -
##     1, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -203.87 -141.87  -45.17   88.98  781.83
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## weathersit.1  204.869      1.680  121.97   <2e-16 ***
## weathersit.2  175.165      2.662   65.80   <2e-16 ***
## weathersit.3  111.501      4.758   23.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 179.4 on 17376 degrees of freedom
## Multiple R-squared:  0.5321, Adjusted R-squared:  0.532
## F-statistic:  6585 on 3 and 17376 DF,  p-value: < 2.2e-16
```

```r
summary(fit.2)
```

```
##
## Call:
## lm(formula = cnt ~ weathersit.2 + weathersit.3, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -203.87 -141.87  -45.17   88.98  781.83
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   204.869      1.680 121.972   <2e-16 ***
## weathersit.2  -29.704      3.148  -9.437   <2e-16 ***
## weathersit.3  -93.369      5.046 -18.503   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 179.4 on 17376 degrees of freedom
## Multiple R-squared:  0.02148,    Adjusted R-squared:  0.02137
## F-statistic: 190.7 on 2 and 17376 DF,  p-value: < 2.2e-16
```

```r
summary(fit.3)
```

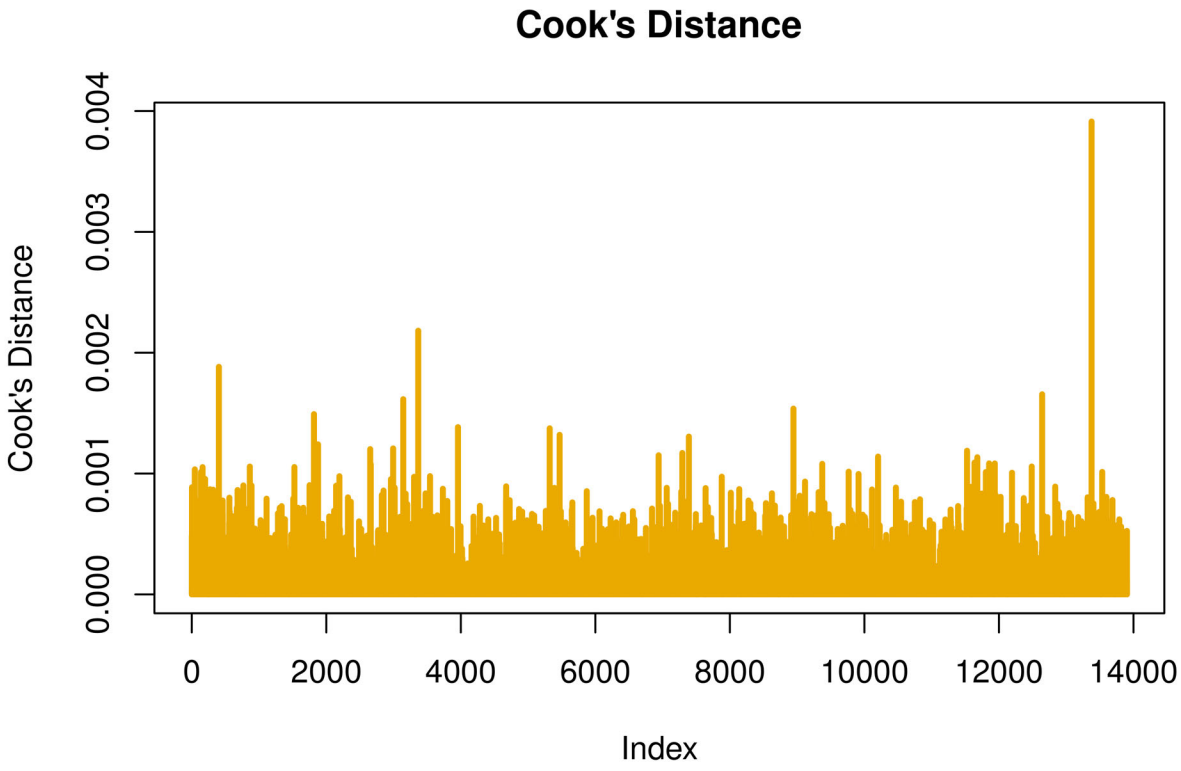
```
##
## Call:
## lm(formula = cnt ~ weathersit, data = clean_data)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -203.87 -141.87  -45.17   88.98  781.83
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  204.869      1.680 121.972   <2e-16 ***
## weathersit2  -29.704      3.148  -9.437   <2e-16 ***
## weathersit3  -93.369      5.046 -18.503   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 179.4 on 17376 degrees of freedom
## Multiple R-squared:  0.02148,    Adjusted R-squared:  0.02137
## F-statistic: 190.7 on 2 and 17376 DF,  p-value: < 2.2e-16
```

Model Assessment

Outliers

```
# Cook's Distance
cook = cooks.distance(model1)
plot(cook,
     type="h",
     lwd=3,
     col=buzzgold,
     ylab = "Cook's Distance",
     main="Cook's Distance")
```



There is one observation with a Cook's Distance noticeably higher than the other observations. However, its Cook's distance is close to 0.004, suggesting that there are likely no outliers.

Multicollinearity

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.6.2
```

```
## Loading required package: carData
```

```
round(vif(model1),3)
```

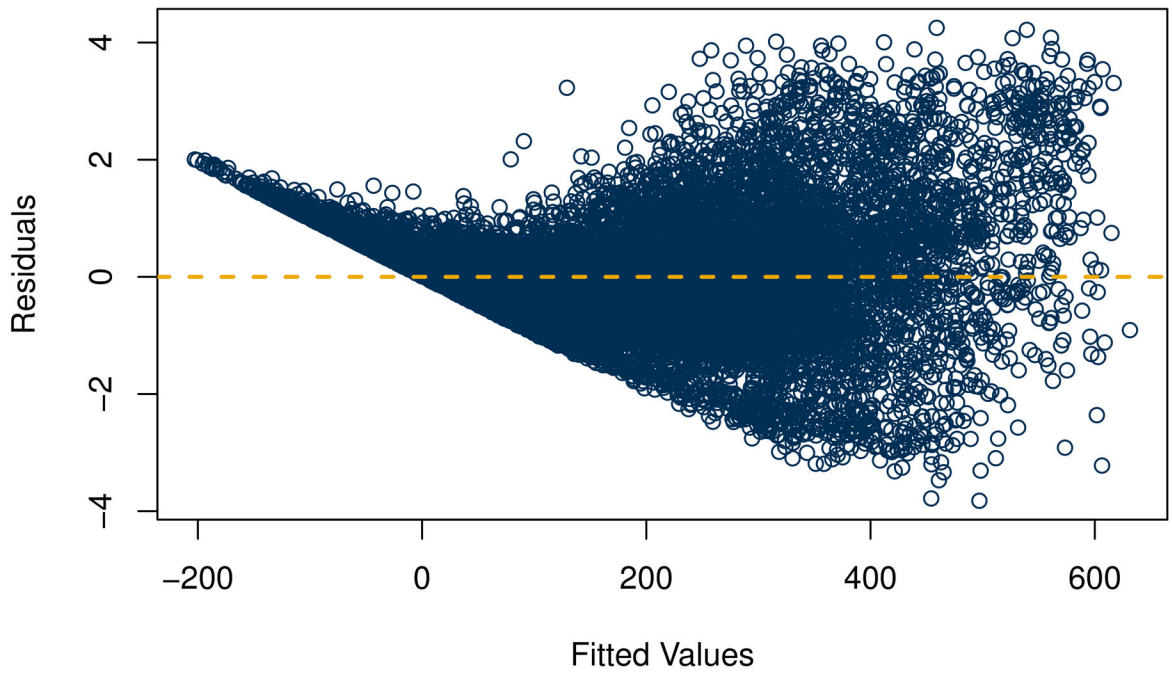
```
##                GVIF Df GVIF^(1/(2*Df))
## season      165.308  3           2.343
## yr            1.025  1           1.012
## mnth        323.778 11           1.300
## hr            1.771 23           1.012
## holiday       1.121  1           1.059
## weekday       1.137  6           1.011
## weathersit    1.386  2           1.085
## temp         51.283  1           7.161
## atemp        43.748  1           6.614
## hum           1.921  1           1.386
## windspeed     1.251  1           1.118
```

As VIFs of the season, mnth, temp, atemp factors are greater than max(10, 1/(1-R2)), it indicates there is a problem of multicollinearity in the linear model. So, we should not use all the predictors in the model.

Goodness of Fit

Constant Variance Assumption

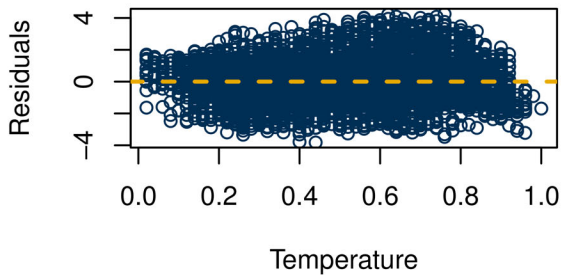
```
# Extract the standardized residuals
resids = rstandard(model1)
fits = model1$fitted

# Plot the standardized residuals against
# fitted values
plot(fits, resids,
     xlab="Fitted Values",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
```

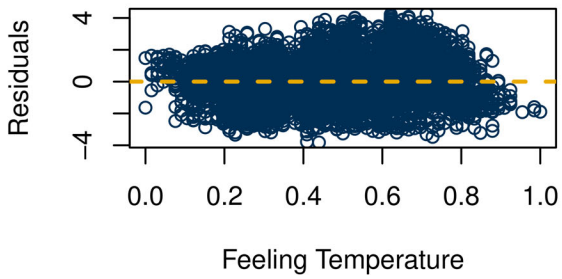


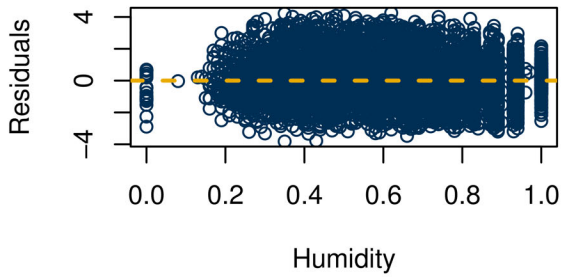
- The constant variance assumption does not hold – the variance increases when moving from lower to higher fitted values.
- The residuals, at low y values, seem to follow a straight-line pattern. The linear pattern in the beginning suggests that the response variable stays constant for a range of predictor values.

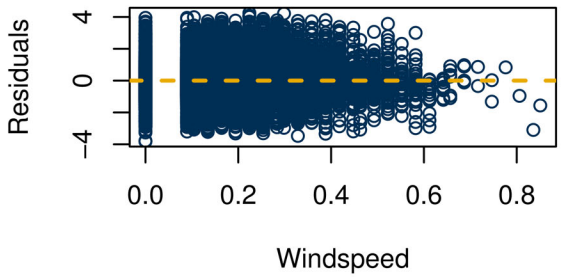
Linearity Assumption

```r
par(mfrow=c(2,2))
# Plot the standardized residuals against
# temperature
plot(train$temp, resids,
     xlab="Temperature",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train$atemp, resids,
     xlab="Feeling Temperature",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train$hum, resids,
     xlab="Humidity",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train$windspeed, resids,
     xlab="Windspeed",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
```





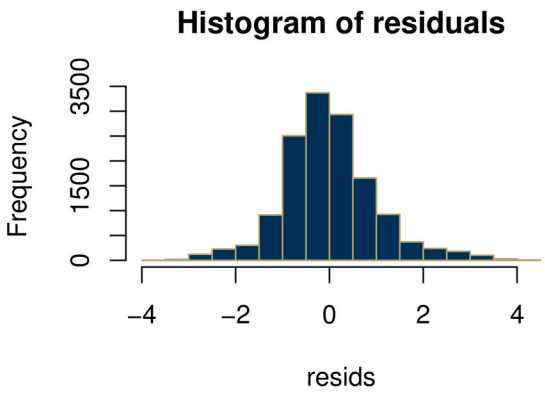


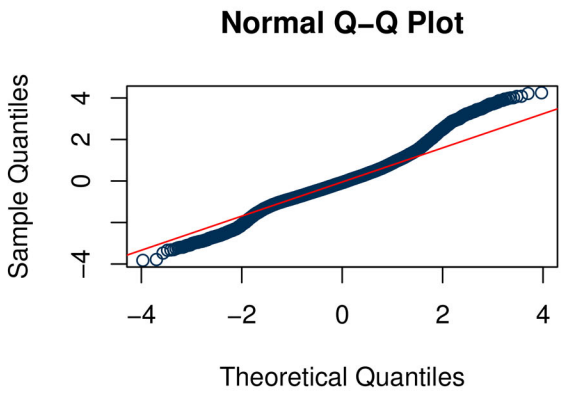


- Analysis here

Normality Assumption

```
par(mfrow=c(1,2))
# Plot histogram of std residuals
hist(resids,
     nclass=20,
     col=gtblue,
     border=techgold,
     main="Histogram of residuals")

# qq plot of std residuals
qqnorm(resids,
       col=gtblue)
qqline(resids,
       col="red")
```



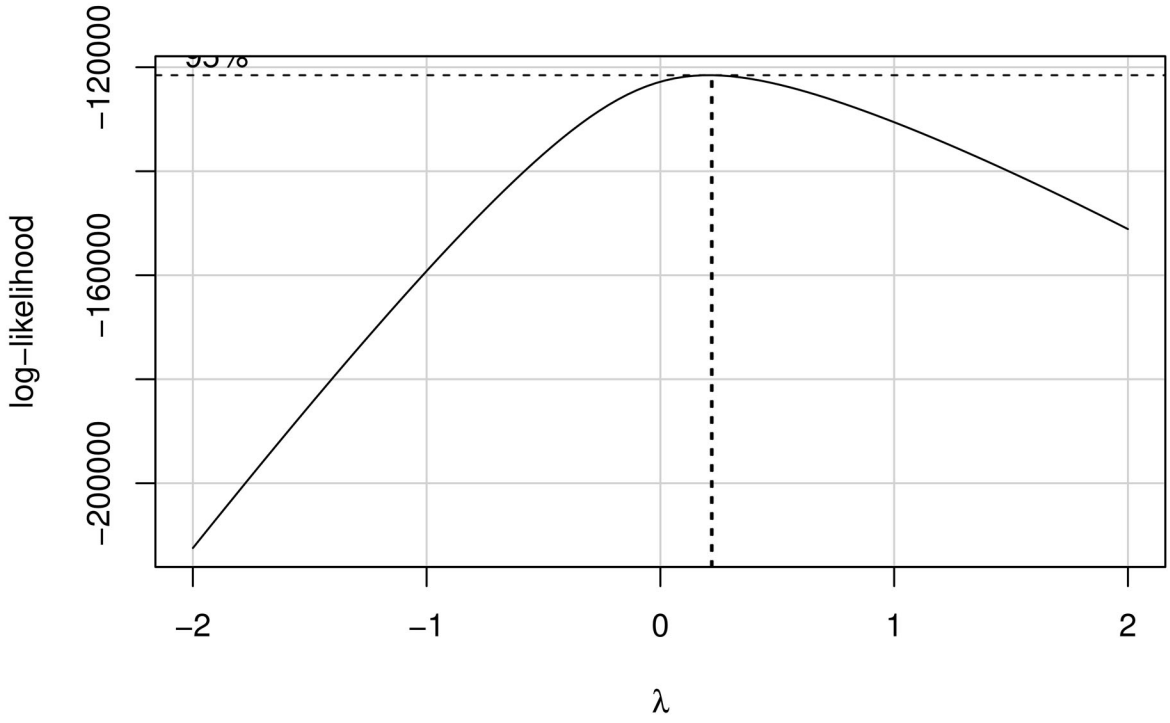


- Analysis here

Transforming to improve Goodness of Fit

Transforming the Response Variable

```
# Box-Cox Transformation
bc = boxCox(model1)
```



```r
lambda = bc$x[which(bc$y==max(bc$y))]
cat("Optimal lambda:", lambda)
```

```
## Optimal lambda: 0.2222222
```

```r
# Fitting the model with square root
# transformation
model2 = lm(sqrt(cnt)~., data=train)
summary(model2)
```

```
##
## Call:
## lm(formula = sqrt(cnt) ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.5004  -1.9154  -0.1174   2.0551  10.2010
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.67339    0.22788   7.343 2.20e-13 ***
## season2      1.36725    0.16576   8.248  < 2e-16 ***
## season3      1.38112    0.19740   6.997 2.74e-12 ***
## season4      2.72181    0.16753  16.246  < 2e-16 ***
## yr1          2.80465    0.05357  52.357  < 2e-16 ***
## mnth2        0.37929    0.13434   2.823 0.004759 **
## mnth3        0.67649    0.15092   4.482 7.44e-06 ***
## mnth4        0.51555    0.22374   2.304 0.021224 *
## mnth5        1.10776    0.24004   4.615 3.97e-06 ***
## mnth6        0.49865    0.24671   2.021 0.043279 *
## mnth7       -0.32604    0.27738  -1.175 0.239848
## mnth8        0.30032    0.27068   1.109 0.267239
## mnth9        1.05184    0.24036   4.376 1.22e-05 ***
## mnth10       0.51581    0.22227   2.321 0.020322 *
## mnth11      -0.24118    0.21397  -1.127 0.259693
## mnth12      -0.03754    0.17030  -0.220 0.825541
## hr1         -1.63371    0.18285  -8.934  < 2e-16 ***
## hr2         -2.57126    0.18451 -13.935  < 2e-16 ***
## hr3         -3.76624    0.18670 -20.173  < 2e-16 ***
## hr4         -4.18981    0.18552 -22.585  < 2e-16 ***
## hr5         -2.35923    0.18634 -12.661  < 2e-16 ***
## hr6          1.48525    0.18521   8.019 1.15e-15 ***
## hr7          6.82450    0.18306  37.279  < 2e-16 ***
## hr8         10.73916    0.18510  58.017  < 2e-16 ***
## hr9          7.49801    0.18456  40.627  < 2e-16 ***
## hr10         5.44242    0.18429  29.532  < 2e-16 ***
## hr11         6.20970    0.18759  33.102  < 2e-16 ***
## hr12         7.44803    0.18714  39.799  < 2e-16 ***
## hr13         7.30991    0.18873  38.733  < 2e-16 ***
## hr14         6.76786    0.19164  35.316  < 2e-16 ***
## hr15         7.08816    0.19158  36.998  < 2e-16 ***
## hr16         9.01922    0.18999  47.472  < 2e-16 ***
## hr17        12.73239    0.18952  67.184  < 2e-16 ***
## hr18        12.12571    0.18840  64.362  < 2e-16 ***
```

```
## hr19         9.43773    0.18782  50.249  < 2e-16 ***
## hr20         7.02243    0.18554  37.849  < 2e-16 ***
## hr21         5.37490    0.18616  28.873  < 2e-16 ***
## hr22         3.86367    0.18559  20.818  < 2e-16 ***
## hr23         1.99954    0.18375  10.882  < 2e-16 ***
## holiday1    -0.98570    0.16444  -5.994 2.09e-09 ***
## weekday1     0.22887    0.10128   2.260 0.023849 *
## weekday2     0.17409    0.09936   1.752 0.079781 .
## weekday3     0.28347    0.09890   2.866 0.004161 **
## weekday4     0.34388    0.09933   3.462 0.000538 ***
## weekday5     0.72314    0.09880   7.320 2.62e-13 ***
## weekday6     0.53031    0.09877   5.369 8.04e-08 ***
## weathersit2 -0.34621    0.06553  -5.284 1.29e-07 ***
## weathersit3 -2.65358    0.11236 -23.617  < 2e-16 ***
## temp         3.84655    0.98371   3.910 9.26e-05 ***
## atemp        4.87870    1.01773   4.794 1.65e-06 ***
## hum         -2.58165    0.19025 -13.570  < 2e-16 ***
## windspeed   -1.10133    0.24135  -4.563 5.08e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.12 on 13851 degrees of freedom
## Multiple R-squared:  0.7849, Adjusted R-squared:  0.7841
## F-statistic: 990.8 on 51 and 13851 DF,  p-value: < 2.2e-16
```

Finding Insignificant Variables

```
which(summary(model2)$coeff[,4]>0.05)
```

```
##    mnth7    mnth8   mnth11   mnth12 weekday2
##       11       12       15       16       42
```

Explanatory Power

```
cat('R-squared:', summary(model2)$r.squared)
```

```
## R-squared: 0.7848631
```

Multicollinearity

```
vif(model2)
```

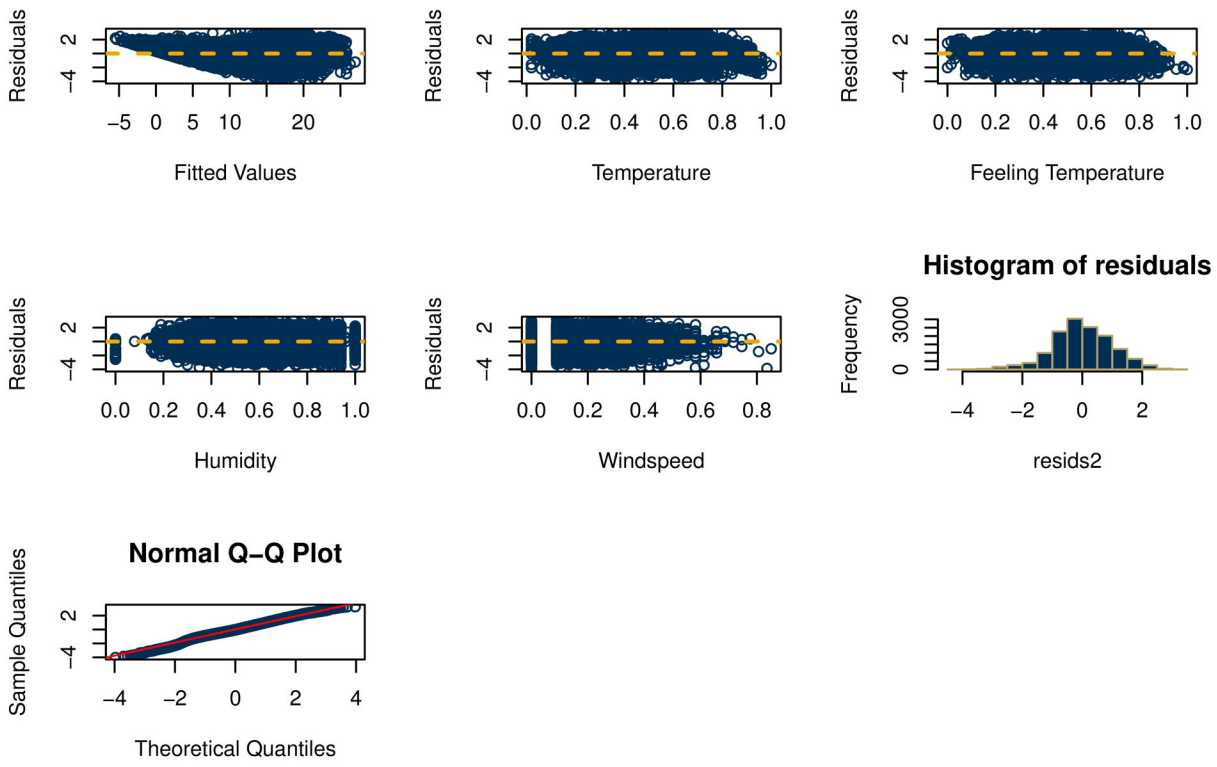
```
##                  GVIF Df GVIF^(1/(2*Df))
## season     165.308269  3        2.342695
## yr           1.024763  1        1.012306
## mnth       323.777775 11        1.300475
## hr           1.770765 23        1.012499
## holiday      1.121207  1        1.058870
## weekday      1.137323  6        1.010781
## weathersit   1.385718  2        1.084973
## temp        51.283316  1        7.161237
## atemp       43.747556  1        6.614193
## hum          1.921250  1        1.386091
## windspeed    1.250741  1        1.118365
```

Explanatory power of the new model is better with respect to the coefficient of determination, but multicollinearity is still a problem.

Goodness of Fit

```
# Extract the standardized residuals
resids2 = rstandard(model2)
fits2 = model2$fitted

par(mfrow=c(3,3))
# Constant Variance Assumption
plot(fits2, resids2,
     xlab="Fitted Values",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
# Plot the standardized residuals against
# temperature
plot(train$temp, resids2,
     xlab="Temperature",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train$atemp, resids2,
     xlab="Feeling Temperature",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train$hum, resids2,
     xlab="Humidity",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train$windspeed, resids2,
     xlab="Windspeed",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
# Plot histogram of std residuals
```

```r
hist(resids2,
     nclass=20,
     col=gtblue,
     border=techgold,
     main="Histogram of residuals")

# qq plot of std residuals
qqnorm(resids2,
       col=gtblue)
qqline(resids2,
       col="red")
```



The constant variance assumption is still violated. The transformation has not improved the goodness of fit even though the model performance is better with respect to the coefficient of determination.

Removing Low Demand Data to improve Goodness of Fit

```r
# Set a seed for reproducibility
set.seed(9)

# Remove data for hours 0-6
hrs = as.numeric(data$hr)
data_red = clean_data[which(hrs>=7),]
```

```r
# 80% Train 20% Test split
sample_size = floor(0.8*nrow(data_red))
picked = sample(seq_len(nrow(data_red)), size=sample_size)
train_red = data_red[picked,]
test_red = data_red[-picked,]

# Fitting the model with square root transformation
model3 = lm(sqrt(cnt)~.,data=train_red)
summary(model3)$r.squared
```

```
## [1] 0.6579021
```

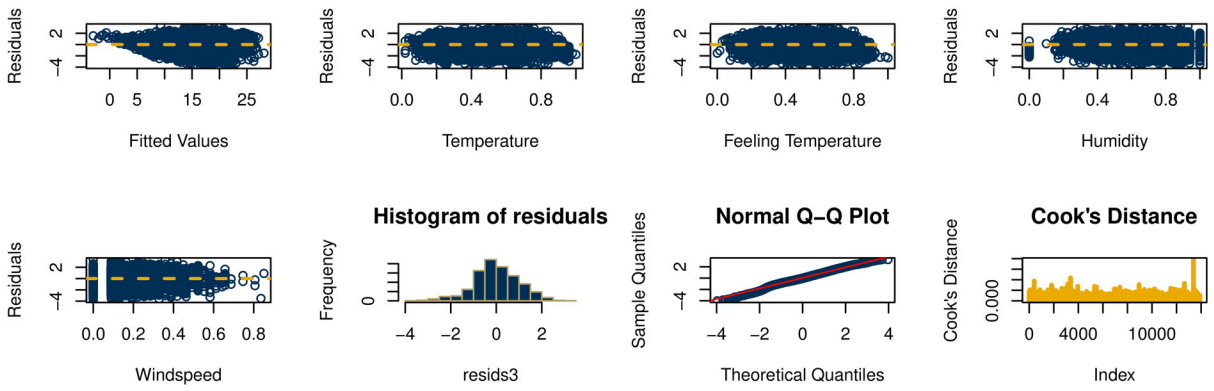
```r
which(summary(model3)$coeff[,4]>0.05)
```

```
##  mnth7 mnth11 mnth12   hr14   hr15   hr20
##     11     15     16     23     24     29
```

Goodness of Fit

```r
# Extract the standardized residuals
resids3 = rstandard(model3)
fits3 = model3$fitted

par(mfrow=c(2,4))
# Constant Variance Assumption
plot(fits3, resids3,
     xlab="Fitted Values",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
# Plot the standardized residuals against
# temperature
plot(train_red$temp, resids3,
     xlab="Temperature",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train_red$atemp, resids3,
     xlab="Feeling Temperature",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train_red$hum, resids3,
     xlab="Humidity",
     ylab="Residuals",
```

```
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
plot(train_red$windspeed, resids3,
     xlab="Windspeed",
     ylab="Residuals",
     main="",
     col=gtblue)
abline(0, 0,
       col=buzzgold,
       lty=2, lwd=2)
# Plot histogram of std residuals
hist(resids3,
     nclass=20,
     col=gtblue,
     border=techgold,
     main="Histogram of residuals")

# qq plot of std residuals
qqnorm(resids2,
       col=gtblue)
qqline(resids3,
       col="red")

# Cook's Distance
cook3 = cooks.distance(model1)
plot(cook3,
     type="h",
     lwd=3,
     col=buzzgold,
     ylab = "Cook's Distance",
     main="Cook's Distance")
```



Prediction

```
# Build a prediction of the models with the new test data
# Specify whether a confidence or prediction interval
pred1 = predict(model1, test, interval = 'prediction', level=0.95)
pred2 = predict(model2, test, interval = 'prediction', level=0.95)
pred3 = predict(model3, test_red, interval = 'prediction', level=0.95)

head(pred1, 12)
```

```
##            fit        lwr       upr
## 2   -93.551819 -293.50680 106.40316
## 10  113.991376  -85.98471 313.96746
## 25  -66.591692 -266.62963 133.44625
## 34   26.095391 -173.86259 226.05337
## 45   36.808102 -163.13933 236.75554
## 46   -3.996574 -203.93777 195.94462
## 54  213.550029   13.59758 413.50248
## 57   65.030257 -134.91746 264.97798
## 60  106.907557  -93.02962 306.84474
## 67   22.919850 -177.01790 222.85760
## 72 -119.149923 -319.08272  80.78288
## 74 -122.920611 -322.86025  77.01903
```

Model1 Accuracy

```
## Save Predictions to compare with observed data
test.pred1 = pred1[,1]
test.lwr1 = pred1[,2]
test.upr1 = pred1[,3]

# Mean Squared Prediction Error (MSPE)
mean((test.pred1-test$cnt)^2)
```

```
## [1] 10304.95
```

```
# Mean Absolute Prediction Error (MAE)
mean(abs(test.pred1-test$cnt))
```

```
## [1] 74.52024
```

```
# Mean Absolute Percentage Error (MAPE)
mean(abs(test.pred1-test$cnt)/test$cnt)
```

```
## [1] 2.724609
```

```
# Precision Measure (PM)
sum((test.pred1-test$cnt)^2)/sum((test$cnt-mean(test$cnt))^2)
```

```
## [1] 0.3101164
```

```
# Calculate number of values outside UCL and LCL
sum(test$cnt<test.lwr1 | test$cnt>test.upr1) / nrow(test)
```

```
## [1] 0.06904488
```

Model2 Accuracy

```
## Save Predictions to compare with observed data
test.pred2 = pred2[,1]^2
test.lwr2 = pred2[,2]
test.upr2 = pred2[,3]

# Mean Squared Prediction Error (MSPE)
mean((test.pred2-test$cnt)^2)
```

```
## [1] 8955.41
```

```
# Mean Absolute Prediction Error (MAE)
mean(abs(test.pred2-test$cnt))
```

```
## [1] 62.48756
```

```
# Mean Absolute Percentage Error (MAPE)
mean(abs(test.pred2-test$cnt)/test$cnt)
```

```
## [1] 0.8051396
```

```
# Precision Measure (PM)
sum((test.pred2-test$cnt)^2)/sum((test$cnt-mean(test$cnt))^2)
```

```
## [1] 0.2695035
```

```
# CI Measure (CIM)
sum(sqrt(test$cnt)<test.lwr2 | sqrt(test$cnt)>test.upr2) / nrow(test)
```

```
## [1] 0.0618527
```

Model3 Accuracy

```
## Save Predictions to compare with observed data
test.pred3 = pred3[,1]^2
test.lwr3 = pred3[,2]
test.upr3 = pred3[,3]

# Mean Squared Prediction Error (MSPE)
mean((test.pred3-test_red$cnt)^2)
```

```
## [1] 11271.78
```

```
# Mean Absolute Prediction Error (MAE)
mean(abs(test.pred3-test_red$cnt))
```

```
## [1] 78.67701
```

```
# Mean Absolute Percentage Error (MAPE)
mean(abs(test.pred3-test_red$cnt)/test_red$cnt)
```

```
## [1] 0.5172032
```

```
# Precision Measure (PM)
sum((test.pred3-test_red$cnt)^2)/sum((test_red$cnt-mean(test_red$cnt))^2)
```

```
## [1] 0.3616828
```

```r
# CI Measure (CIM)
sum(sqrt(test_red$cnt)<test.lwr3 | sqrt(test_red$cnt)>test.upr3) / nrow(test_red)
```

```
## [1] 0.06098546
```

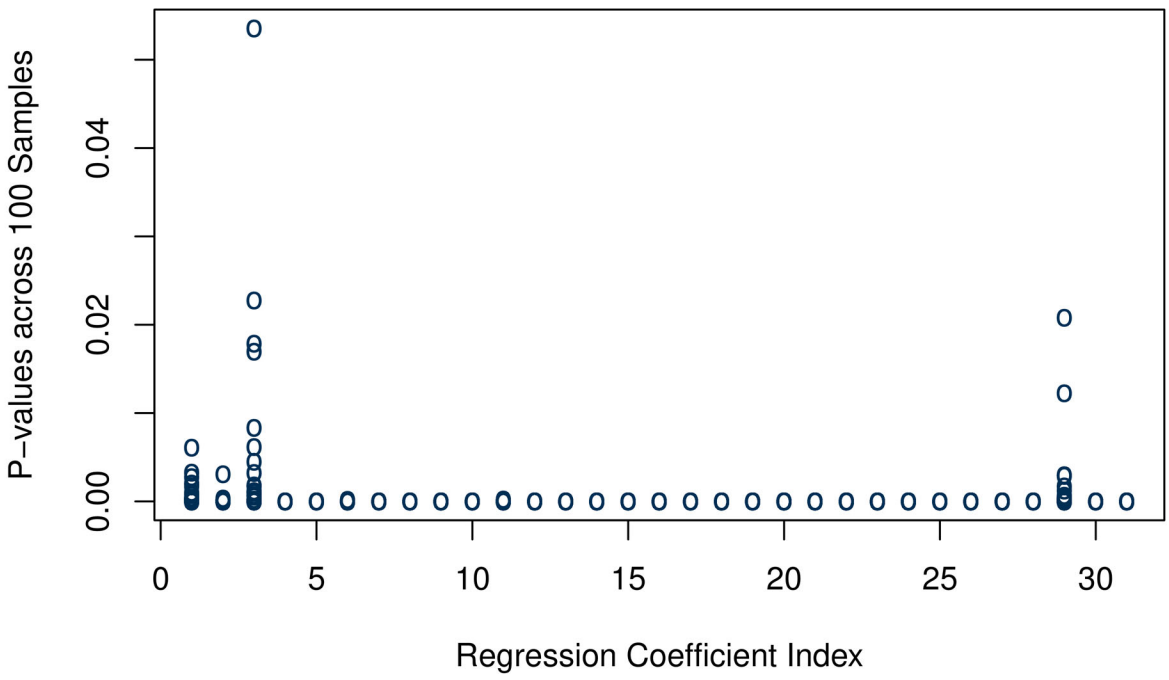
P-values and Large Sample Size

P-value Problem

```r
# Approach: Subsample 40% of the initial data sample & repeat 100 times
count = 1
n = nrow(train)
B = 100
ncoef = dim(summary(model1)$coeff)[1]
pv_matrix = matrix(0,nrow = ncoef,ncol = B)

while (count <= B) {
  # 40% random sample of indices
  subsample = sample(n, floor(n*0.4), replace=FALSE)
  # Extract the random subsample data
  subdata = train[subsample,]
  # Fit the regression for each subsample
  submod = lm(sqrt(cnt)~.,data=subdata)
  # Save the p-values
  pv_matrix[,count] = summary(submod)$coeff[,4]
  # Increment to the next subsample
  count = count + 1
}

# Count pvalues smaller than 0.01 across the 100 (sub)models
alpha = 0.01
pv_significant = rowSums(pv_matrix < alpha)
```

Statistically Significant Coefficients

```r
# Which regression coefficients are statistically significant?
idx_scoef = which(pv_significant>=95)

# Plot the 100 p-values of the significant coefficients
matplot(pv_matrix[idx_scoef,],
     xlab="Regression Coefficient Index",
     ylab="P-values across 100 Samples",
     type="p",
     pch="o",
     col=gtblue)
```



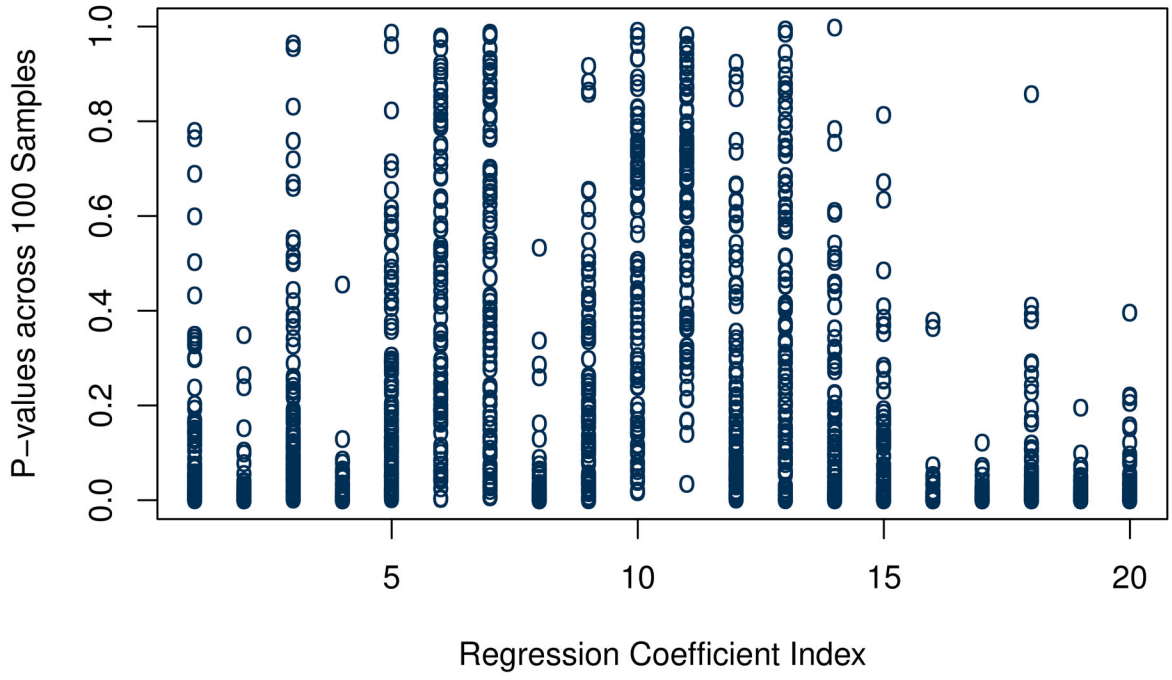

```
# Show the p-values of the significant coefficients in model2
cbind(round(summary(model2)$coeff[idx_scoef,c(1,4)],3),
      Freq=pv_significant[idx_scoef])
```

```
##             Estimate Pr(>|t|) Freq
## (Intercept)    1.673        0  100
## season2        1.367        0  100
## season3        1.381        0   96
## season4        2.722        0  100
## yr1            2.805        0  100
## hr1           -1.634        0  100
## hr2           -2.571        0  100
## hr3           -3.766        0  100
## hr4           -4.190        0  100
## hr5           -2.359        0  100
## hr6            1.485        0  100
## hr7            6.824        0  100
## hr8           10.739        0  100
## hr9            7.498        0  100
## hr10           5.442        0  100
## hr11           6.210        0  100
## hr12           7.448        0  100
## hr13           7.310        0  100
## hr14           6.768        0  100
## hr15           7.088        0  100
## hr16           9.019        0  100
```

```
## hr17          12.732        0  100
## hr18          12.126        0  100
## hr19           9.438        0  100
## hr20           7.022        0  100
## hr21           5.375        0  100
## hr22           3.864        0  100
## hr23           2.000        0  100
## weekday5       0.723        0   98
## weathersit3   -2.654        0  100
## hum           -2.582        0  100
```

Coefficients Not Statsitically Significant

```
## Which regression coefficients are not statistically significant?
idx_icoef = which(pv_significant<85)

# Plot the 100 p-values of the coefficients not statistically
# significant
matplot(pv_matrix[idx_icoef,],
     xlab="Regression Coefficient Index",
     ylab="P-values across 100 Samples",
     type="p",
     pch="o",
     col=gtblue)
```



```r
# Show the p-values of coefficients not stastically significant
# in model2
cbind(round(summary(model2)$coeff[idx_icoef,c(1,4)],3),
      Freq=pv_significant[idx_icoef])
```

```
##             Estimate Pr(>|t|) Freq
## mnth2          0.379    0.005   15
## mnth3          0.676    0.000   70
## mnth4          0.516    0.021    5
## mnth5          1.108    0.000   73
## mnth6          0.499    0.043    4
## mnth7         -0.326    0.240    2
## mnth8          0.300    0.267    3
## mnth9          1.052    0.000   57
## mnth10         0.516    0.020    7
## mnth11        -0.241    0.260    0
## mnth12        -0.038    0.826    0
## weekday1       0.229    0.024    7
## weekday2       0.174    0.080    7
## weekday3       0.283    0.004   18
## weekday4       0.344    0.001   34
## weekday6       0.530    0.000   78
## weathersit2   -0.346    0.000   80
## temp           3.847    0.000   45
## atemp          4.879    0.000   70
## windspeed     -1.101    0.000   53
```