Regression Analysis of Bike Sharing Demand

Chance Robinson, Jayson Barker and Neha Dixit

Master of Science in Data Science, Southern Methodist University, USA

1 Introduction

A bike sharing system is a means of renting a bicycle via a network of kiosk locations throughout a city and returning it to a different place as needed.

For this analysis, we are looking at two years' worth of bikeshare data from Capital Bikeshare in Washington D.C. Our data was retrieved from the *Bike Sharing Demand*[1] Kaggle competition.

The objective is to predict the total count of bikes rented during each hour covered by the test set.

2 Data Description

The dataset we chose for this project was a publicly shared, hourly bike sharing dataset made available through Kaggle in csv format.

This data set is divided into two distinct sets – a train and test set. The train set consists of 10,886 rows (titled "train.csv") and the test set consists of 6,493 rows (titled "test.csv"). Within the training set, the first 19 days of each month are captured whereas in the test data set, the 20th day to the end of each month is present. The entirety of the data spans from 1/1/2011 through 12/31/2012 - encompassing two full years of bike sharing data. Interestingly, the time component of this analysis is captured in hours of each day meaning we have a calendar date represented 24 times (for each hour of that day) in the data, along with it's associated attribute values.

In train data set, there are a total of 12 attributes which capture multiple variables related to bike rentals. Some of these attributes are categorical, and others are continuous. All attributes are summarized in the table below:

Column Name	Type	Description
1. datetime	Date	YYYY-MM-DD HH24 (example: 2011-01-01 04:00:00)
2. season	Integer	(1-4)
3. holiday	Integer	(0 or 1)
4. workingday	Integer	(0 or 1)
5. weather	Integer	(1-4)
6. temp	Float	temparture in Celcius
7. atemp	Float	"feels like" temperature in Celsius
8. humidity	Integer	relative humidity
9. windspeed	Float	wind speed
10. casual	Integer	count of casual users
11. registered	Integer	count of registered users
12. count	Integer	count of total users (*response variable*)

Note that the test data set lacks the casual, registered and count variables.

3 Exploratory Data Analysis

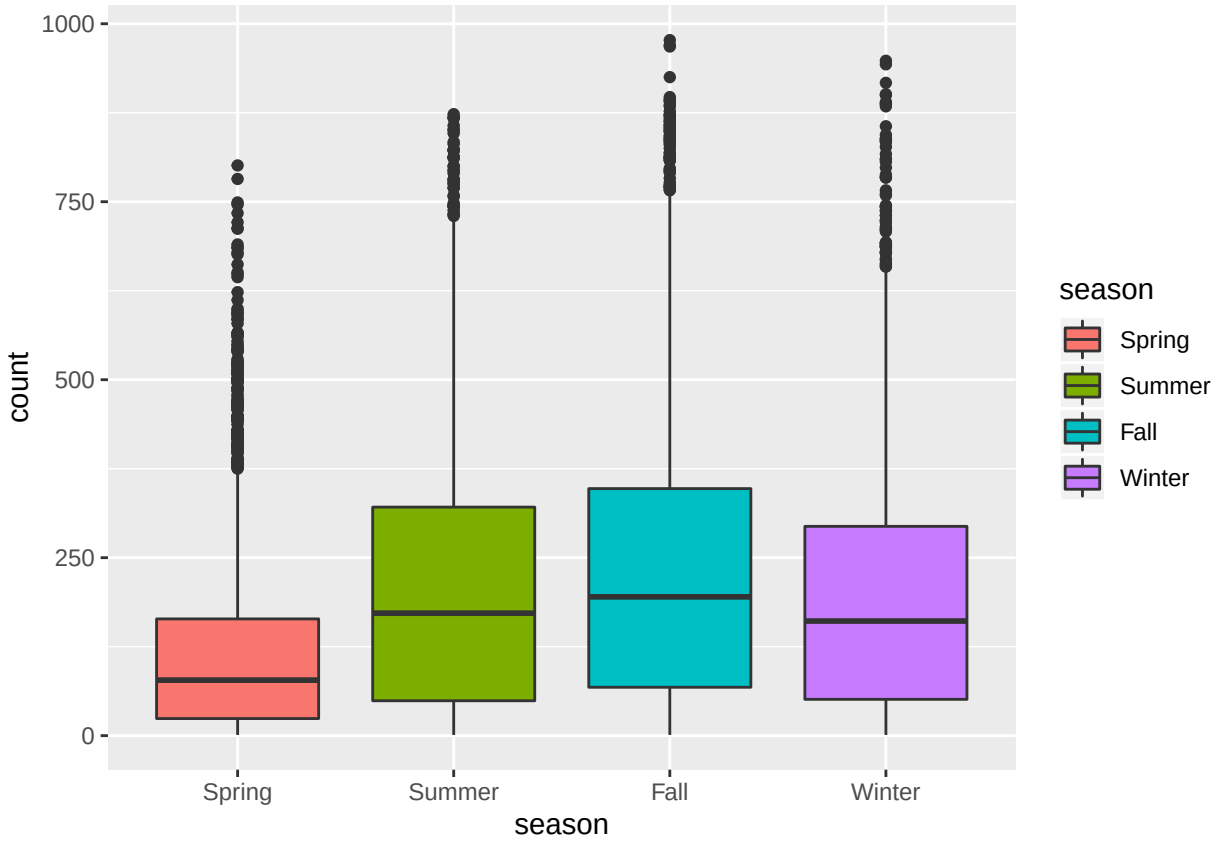
3.1 Categorical Variables Plots

Several numeric variables were found to be better suited to categorization and were converted to factors.

3.1.1 Season

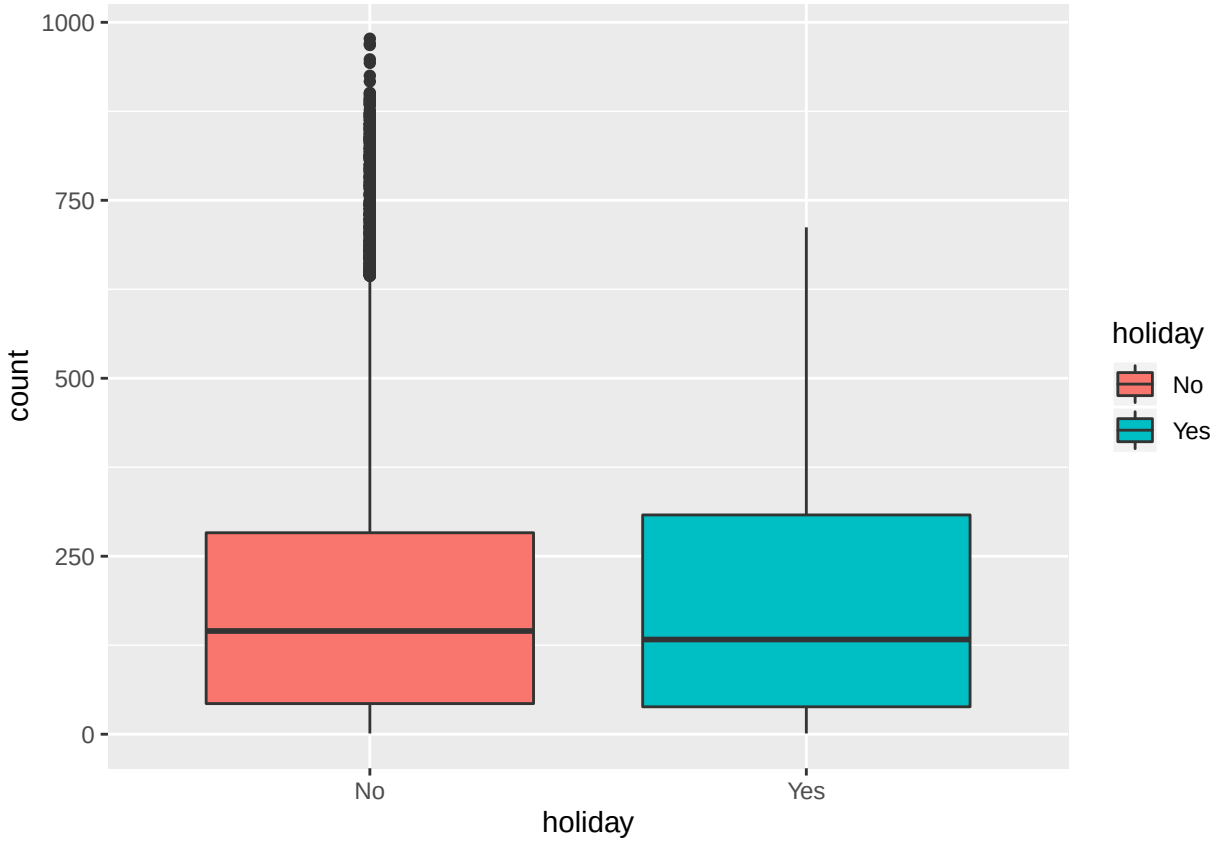
The Summer and Fall months show the highest seasonal averages.

Season	Label	Description
1	Spring	Dec 21 ~ Mar 20
2	Summer	Mar 21 ~ Jun 20
3	Fall	Jun 21 ~ Sep 20
4	Winter	Sep 21 ~ Dec 20



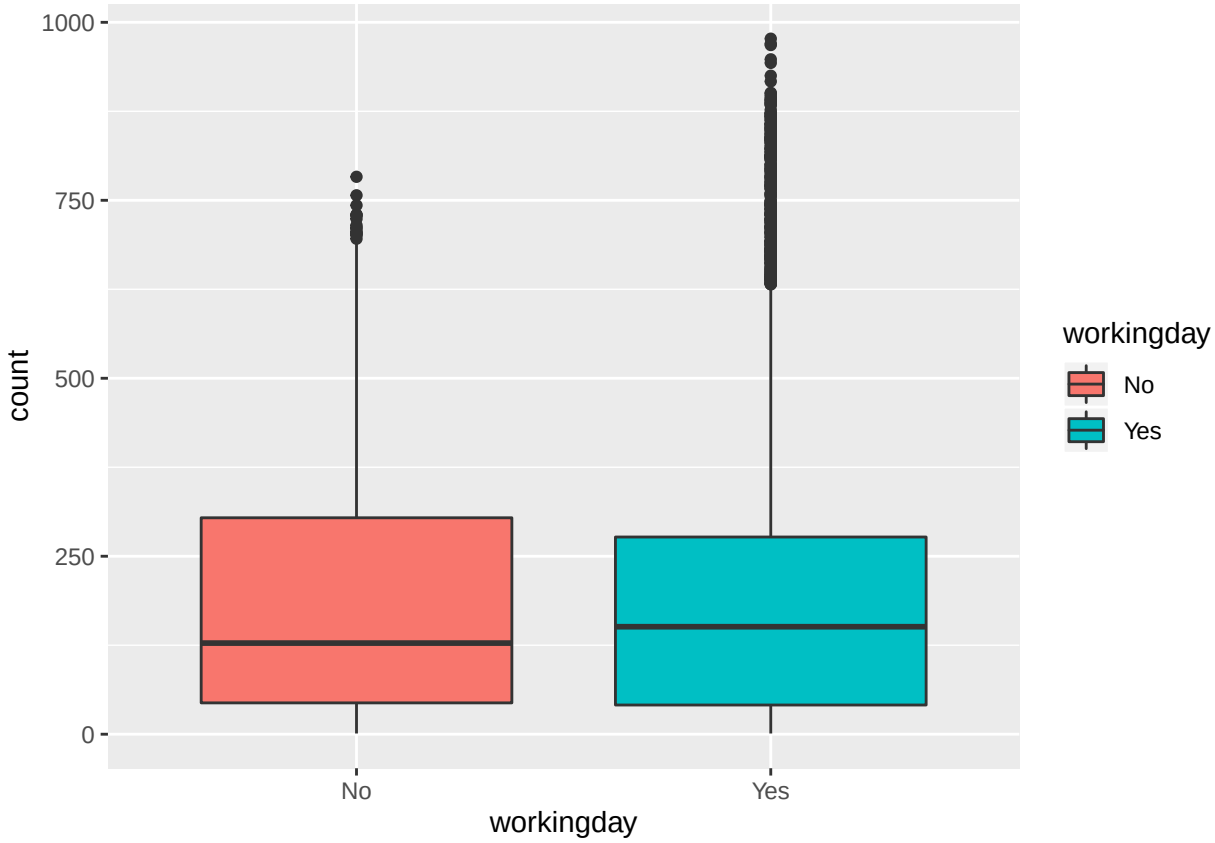
3.1.2 Holiday

Whether or not the day is a holiday, oddly enough, has no noticeable impact on the average counts.



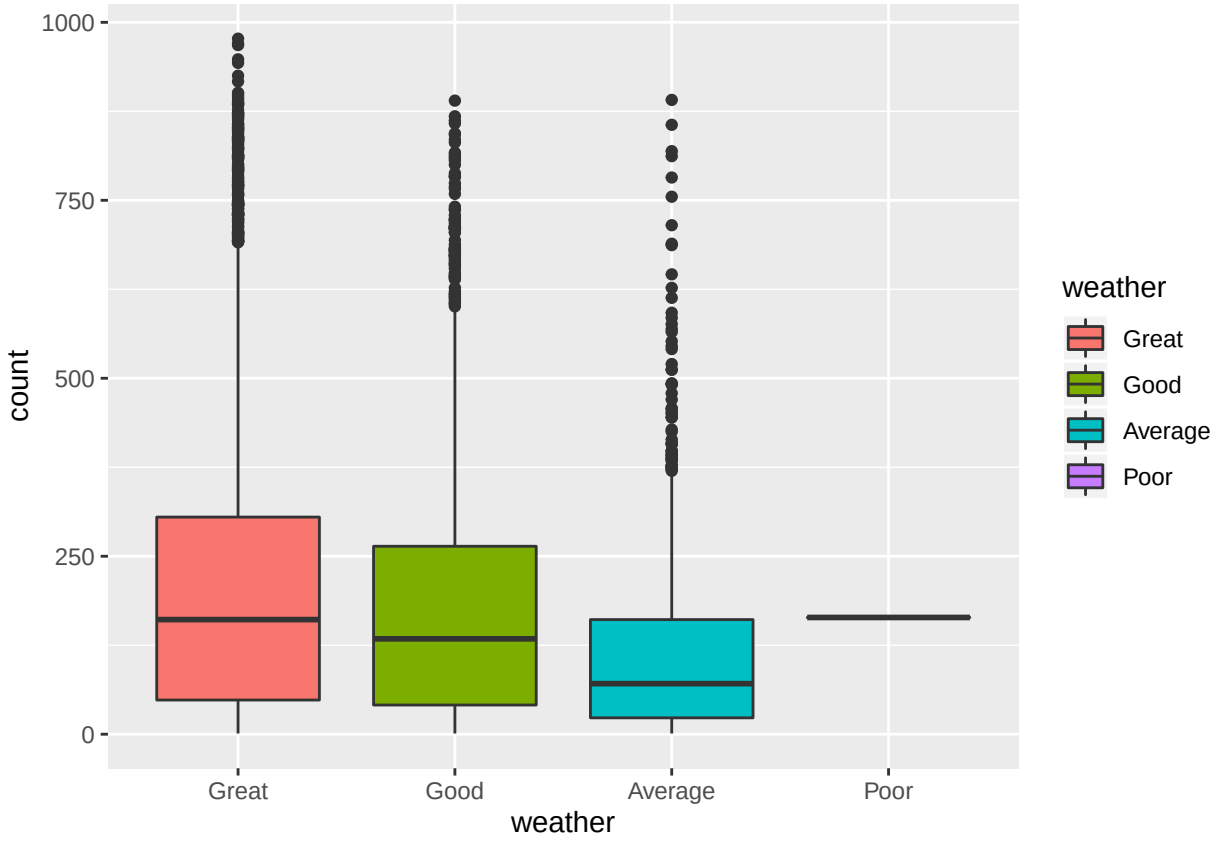
3.1.3 Working Day

The same could be said of whether or not the recording was on a working day or weekend.



3.1.4 Weather

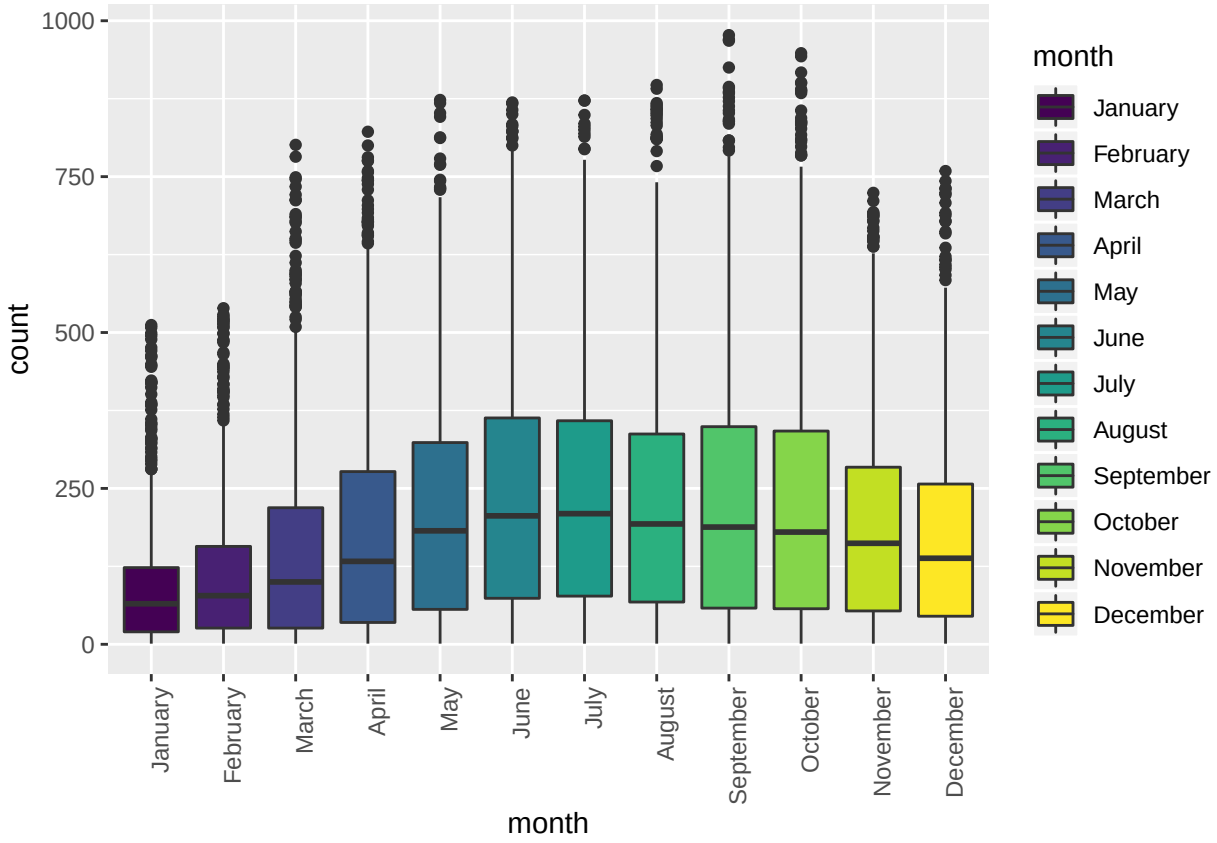
Better weather shows improved rental rates as expected.



3.1.5 Count by Month

The datetime column was broken out into multiple factors as well so that we could visualize the components of each date and aggregate by different dimensions of the timestamp. We felt this was also necessary due to the nature of how the train/ test data sets had been pre-split. (i.e. . . with the first 19 days of the month holding the only true counts to validate our models against.)

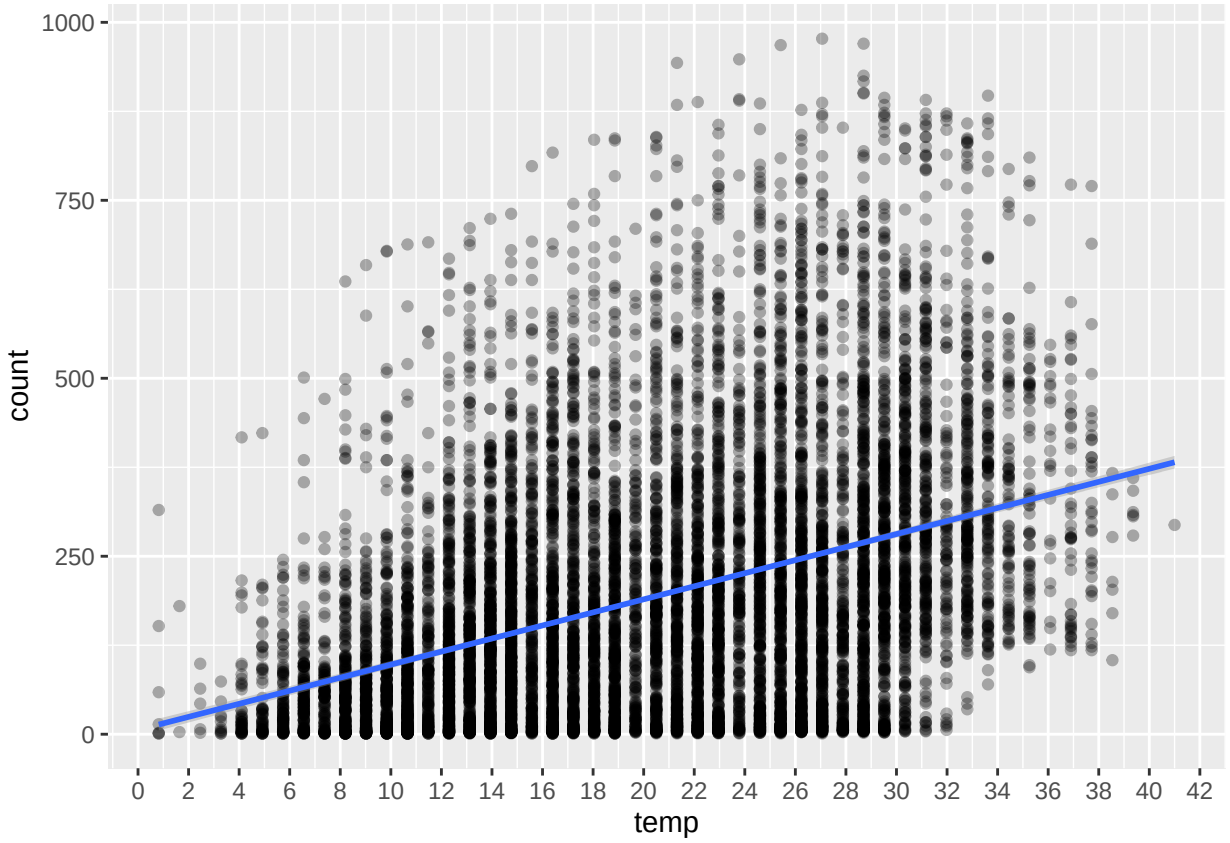
- Year
- Month
- Day
- Hour



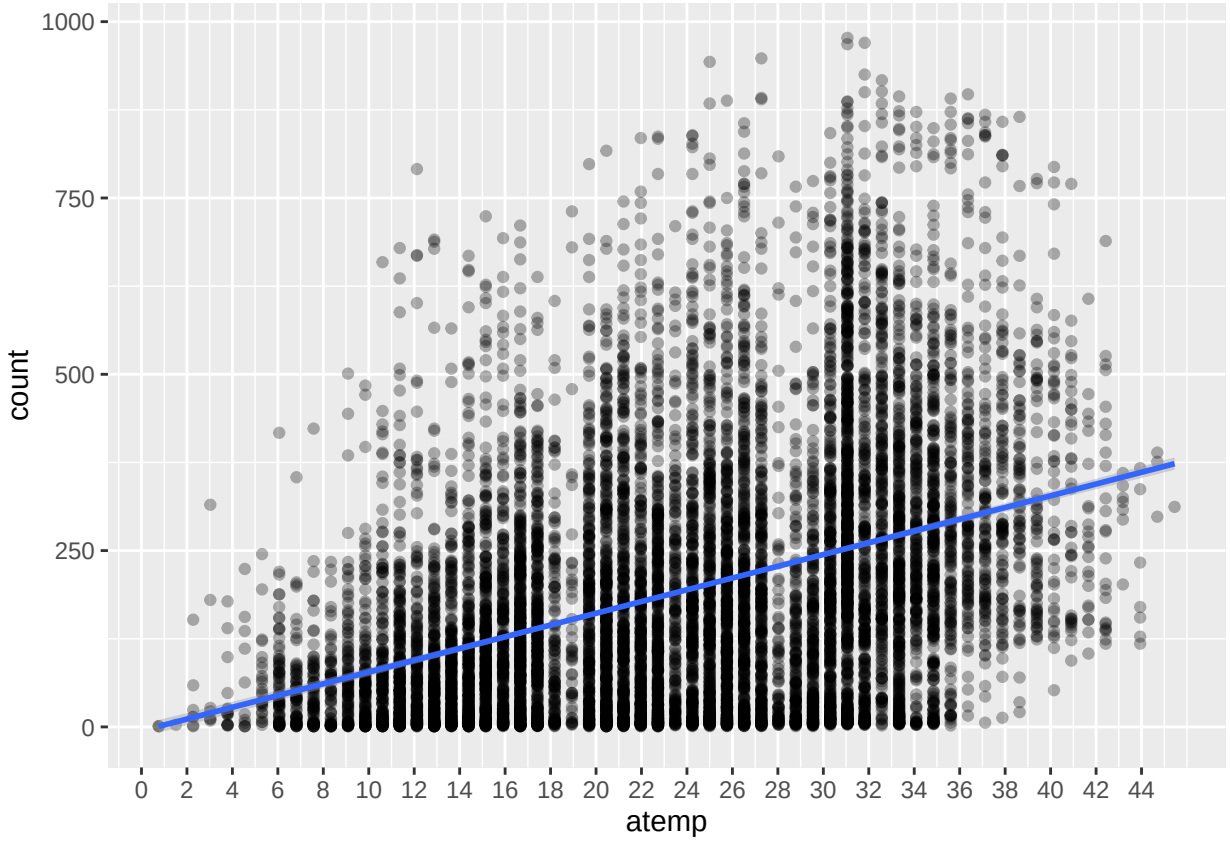
3.2 Continuous Variable Plots

3.2.1 Count by Temperature

The upper and lower extremes of temperature readings appear to coincide with fewer rentals.

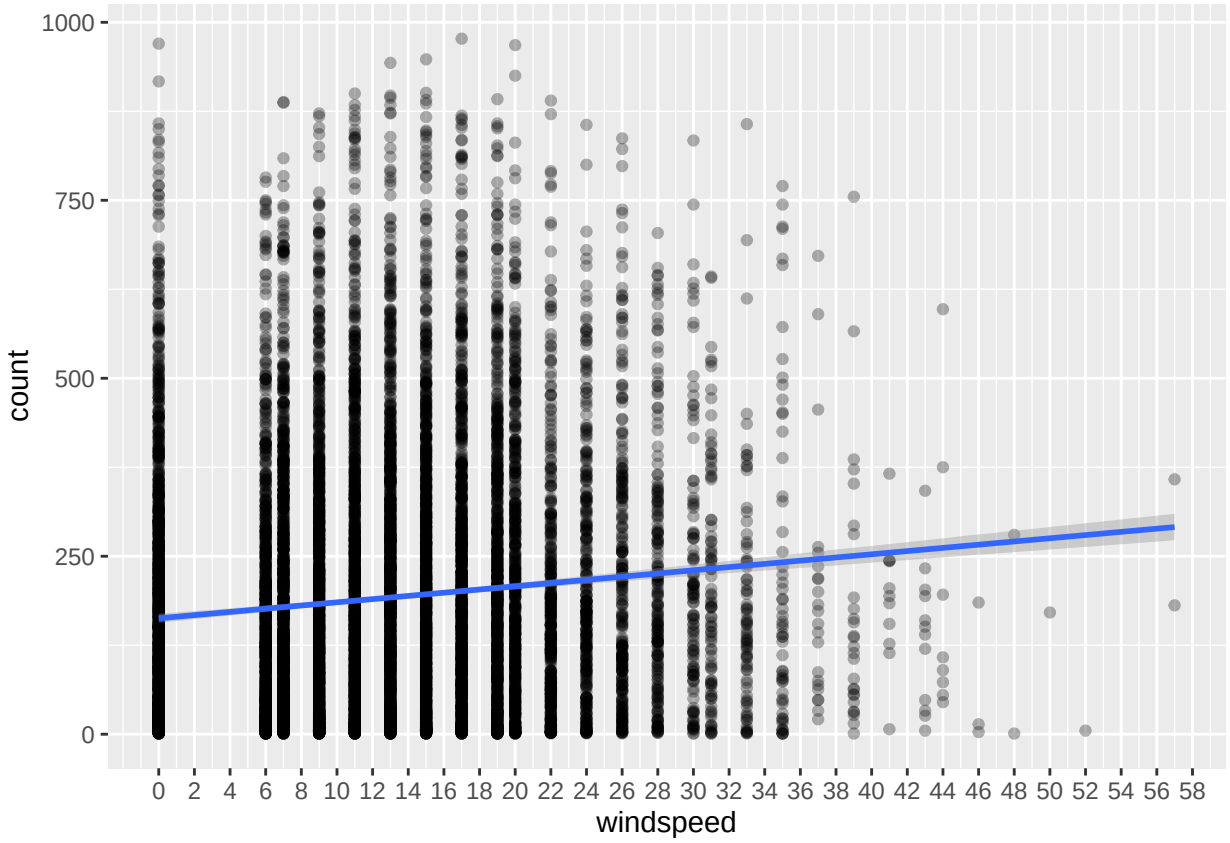


3.2.2 Count by "Feels like" Temperature



3.2.3 Count by Wind Speed

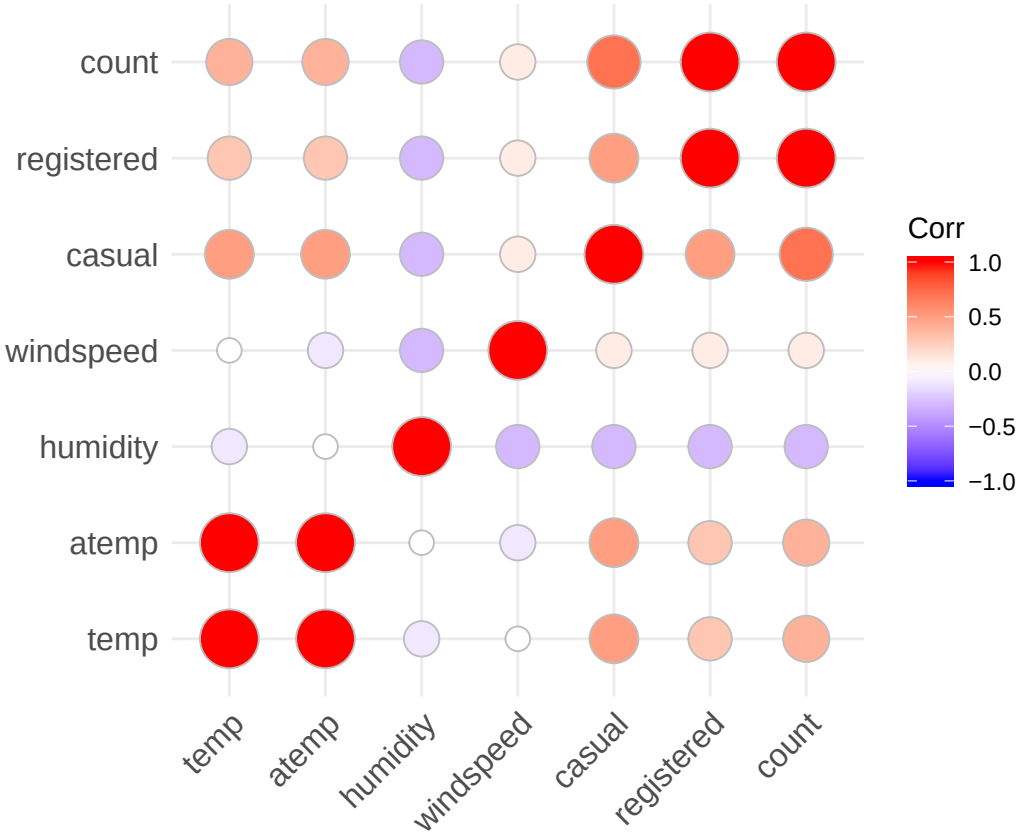
There are fewer rentals recorded as the windspeed increases beyond a certain amount.



3.3 Correlation Matrix

There are several variables with a relatively high level of covariance from the training set. The following columns should therefore be removed so as not to be picked up by any automated modelling techniques.

- atemp
- causual
- registered



4 Objective I Analysis

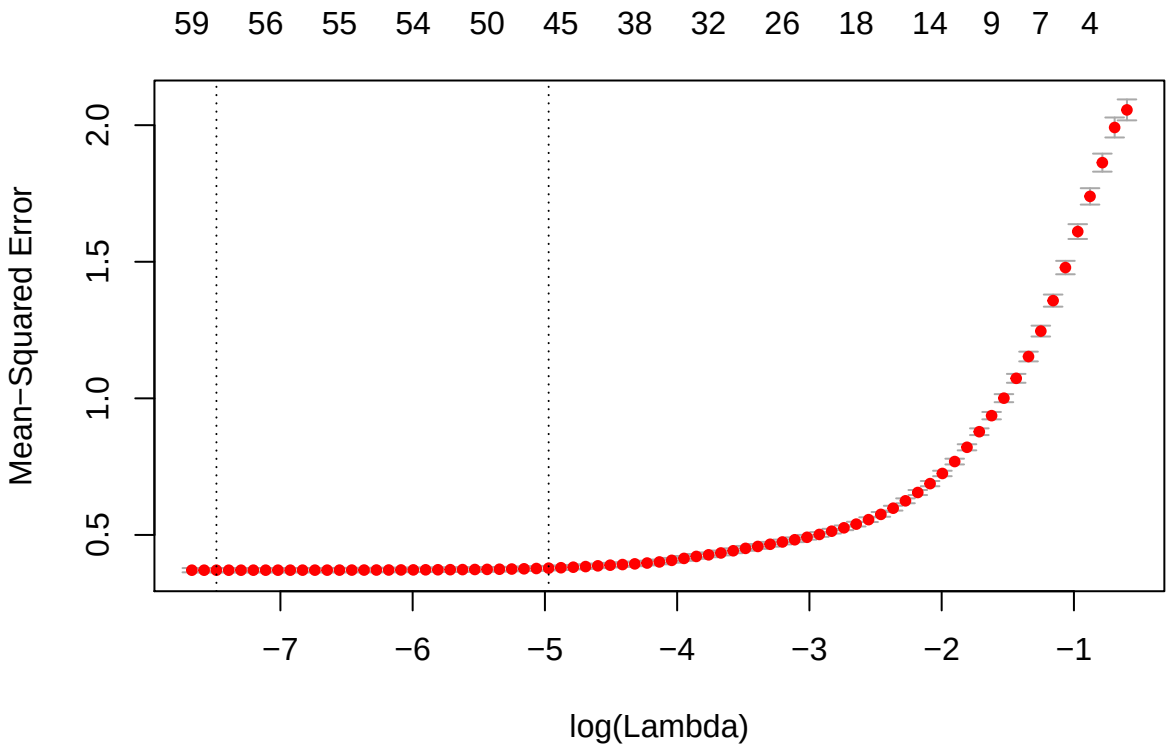
4.1 Question of Interest

The team be utilizing the multiple linear regression techniques we've learned up to this point in the program to predict the bike rental demand on a given date and time. The model will be evaluated on the Root Mean Squared Logarithmic Error, or RMSLE. As this data represents hourly data collected, there is an obvious time component associated with this competition. We wanted to gauge how effective multiple linear regression would be when the assumption of independence is clearly violated.

4.2 Model Selection

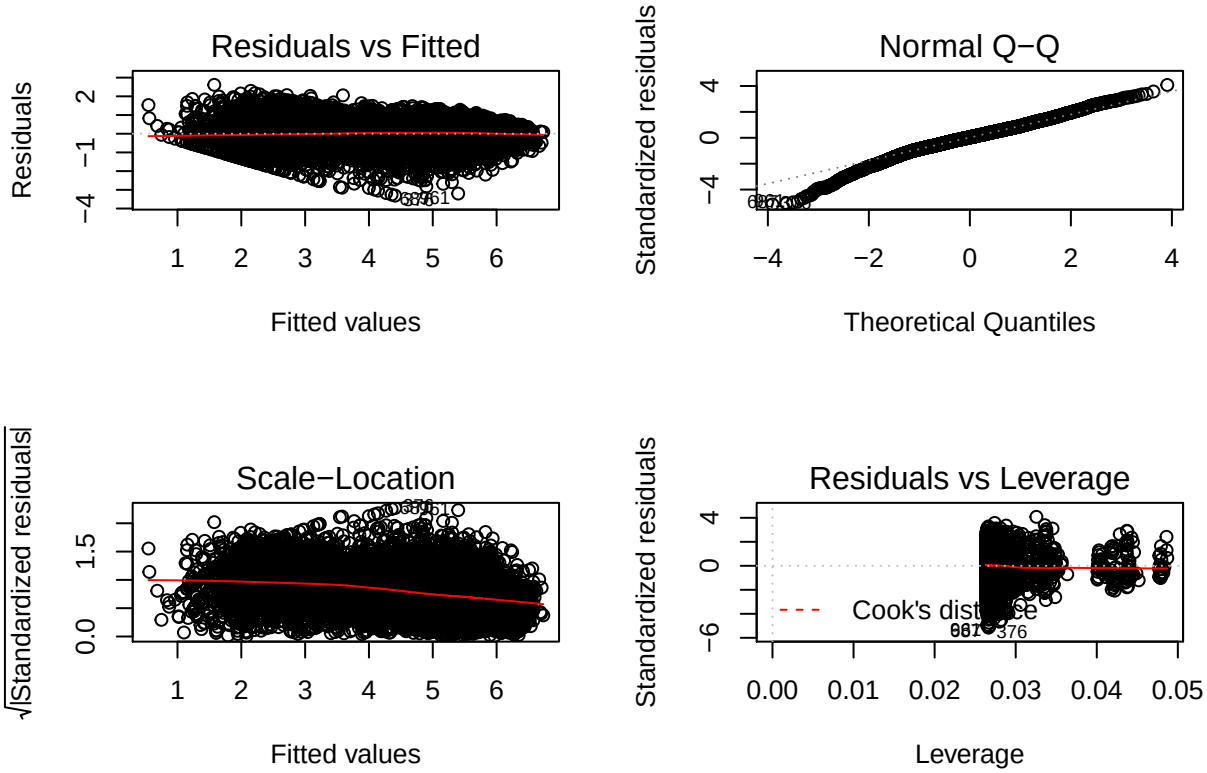
4.2.1 Lasso

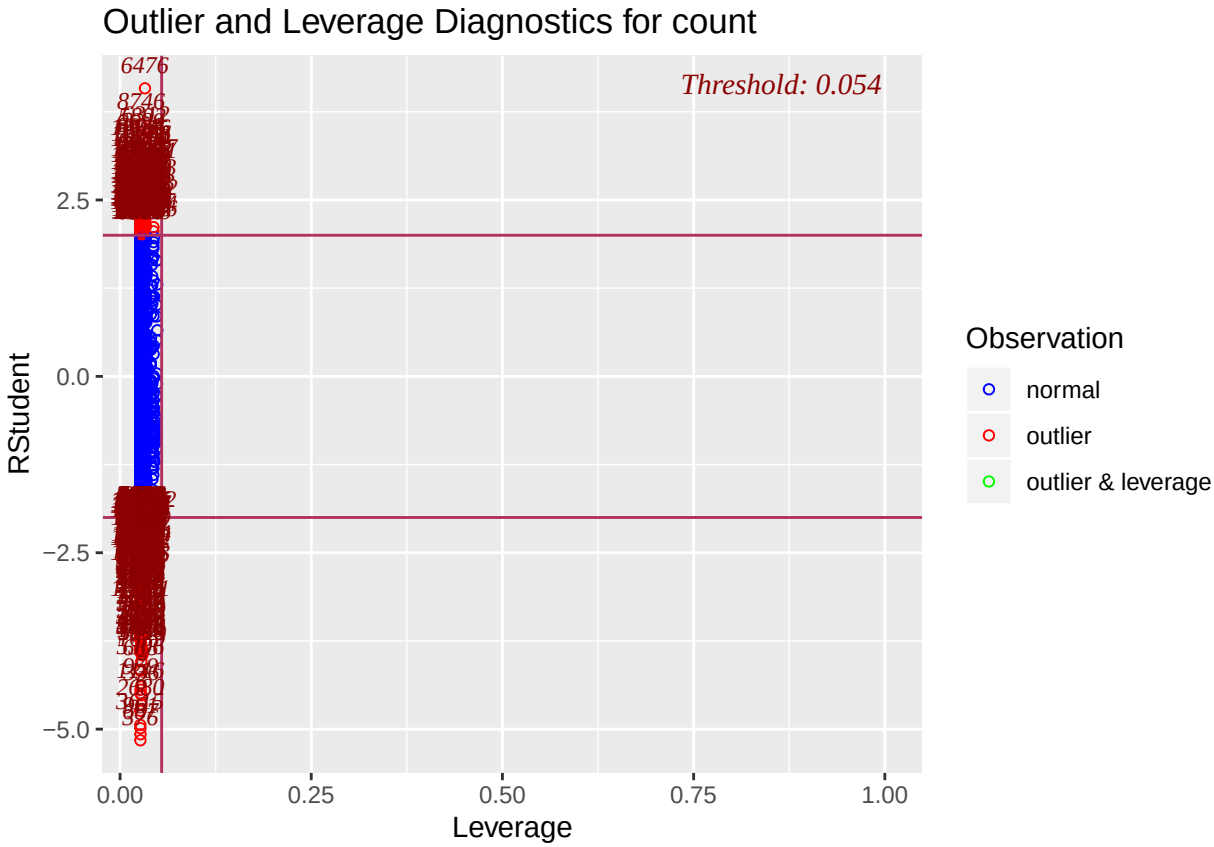
We leveraged the LASSO approach to assist with variable selection for this model. Interestingly, the number of variables identified by our LASSO approach is very similar to the Stepwise approach.



4.2.2 Custom Variable Selection

We developed our custom model by adding an interactive term for month and hour based on the seasonal nature that the box plots exhibited.




Outlier and Leverage Diagnostics for count
Threshold: 0.054
RStudent
2.5
0.0
−2.5
−5.0
0.00
0.25
0.50
0.75
1.00
Leverage
Observation
normal
outlier
outlier & leverage

4.2.2.1 Kaggle Score

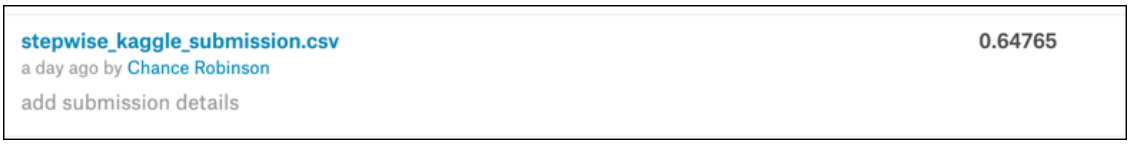
The Root Mean Squared Logarithmic Error Loss (RMSLE) for the Kaggle submission was 0.67517 for our custom model. We scored better than around 24% of all public submission for the competition with this technique.

4.2.3 Stepwise

Next, we leveraged the Stepwise automatic variable selection methodology. As mentioned previously, the variables selected by the Stepwise methodology are very similar to the LASSO recommendation.

4.2.3.1 Kaggle Score

The Root Mean Squared Logarithmic Error Loss (RMSLE) for the Kaggle submission was 0.64765 for our stepwise model. Again, we scored better than around 24% of all public submission for the competition with this technique, which was almost identical to that of the custom model with an interaction.



4.3 Model Assumptions Assessment

- The residual plots are approximately normal based on the QQ plot and Histogram
- The log transformation of the response, in addition to extreme observation removal, has removed both outliers and observations with high leverage
- The independence assumption is clearly broken due to the nature of the hourly collections. We will proceed knowing that MLR may not be the best suited model for this particular data set.

4.4 Comparing Competing Models

We created three models using the training bike sharing data set. All three models performed very similar across our scoring metrics and no single model stood out as the front runner. The table below summarizes the performance of the models using the adjusted R2, AIC, BIC, and RMSE:

Model	Adjusted R2	AIC
1. Custom	0.7915	21642.12
2. Lasso	0.7946	21250.6
3. Stepwise	0.8224	19668.86

4.5 Parameters

The following summarizes the parameters for estimates for the interaction model(custom) and the stepwise model.

4.5.1 Stepwise Model Summary

```
stw.sm
```

```
##
## Call:
## lm(formula = stw.model.formula, data = train.mod.1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.15355 -0.29296  0.03003  0.36343  2.28175
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.3174555  0.0542954  61.100  < 2e-16 ***
## hour01         -0.6274075  0.0399811 -15.693  < 2e-16 ***
## hour02         -1.1359994  0.0404183 -28.106  < 2e-16 ***
## hour03         -1.6615266  0.0411443 -40.383  < 2e-16 ***
## hour04         -1.9322047  0.0414142 -46.656  < 2e-16 ***
## hour05         -0.9274850  0.0402628 -23.036  < 2e-16 ***
## hour06          0.2884532  0.0400794   7.197 6.56e-13 ***
## hour07          1.2738710  0.0399734  31.868  < 2e-16 ***
## hour08          1.9021997  0.0398896  47.687  < 2e-16 ***
## hour09          1.5748200  0.0399255  39.444  < 2e-16 ***
## hour10          1.2474375  0.0400854  31.119  < 2e-16 ***
## hour11          1.3669137  0.0403745  33.856  < 2e-16 ***
## hour12          1.5552795  0.0407026  38.211  < 2e-16 ***
## hour13          1.5299500  0.0410548  37.266  < 2e-16 ***
## hour14          1.4475331  0.0413260  35.027  < 2e-16 ***
## hour15          1.5061812  0.0414038  36.378  < 2e-16 ***
## hour16          1.7681031  0.0413060  42.805  < 2e-16 ***
## hour17          2.1811506  0.0410659  53.113  < 2e-16 ***
## hour18          2.0968370  0.0407964  51.398  < 2e-16 ***
## hour19          1.8037735  0.0403534  44.699  < 2e-16 ***
## hour20          1.5036316  0.0401264  37.472  < 2e-16 ***
## hour21          1.2442347  0.0399632  31.135  < 2e-16 ***
## hour22          1.0003851  0.0398915  25.078  < 2e-16 ***
## hour23          0.6000841  0.0398591  15.055  < 2e-16 ***
## month.L         0.6999912  0.0246473  28.400  < 2e-16 ***
## month.Q        -0.3389488  0.0425804  -7.960 1.89e-15 ***
## month.C         0.1248240  0.0217868   5.729 1.04e-08 ***
## month^4        -0.0322233  0.0226041  -1.426 0.154028
## month^5        -0.1711829  0.0210762  -8.122 5.08e-16 ***
## month^6         0.0058324  0.0202813   0.288 0.773679
```

```
## month^7         0.1613935  0.0204054   7.909 2.84e-15 ***
## month^8        -0.0330624  0.0201660  -1.640 0.101136
## month^9        -0.0205400  0.0201300  -1.020 0.307577
## month^10        0.0446766  0.0201412   2.218 0.026564 *
## month^11       -0.0053160  0.0200316  -0.265 0.790720
## year2012        0.4839078  0.0117932  41.033  < 2e-16 ***
## weatherGood    -0.0552311  0.0143444  -3.850 0.000119 ***
## weatherAverage -0.5515117  0.0242992 -22.697  < 2e-16 ***
## weatherPoor    -0.1034753  0.6025957  -0.172 0.863664
## temp            0.0267535  0.0017862  14.978  < 2e-16 ***
## humidity       -0.0033135  0.0004201  -7.888 3.36e-15 ***
## workingdayYes  -0.0762112  0.0128696  -5.922 3.28e-09 ***
## windspeed      -0.0036725  0.0007753  -4.737 2.20e-06 ***
## holidayYes     -0.0632055  0.0363666  -1.738 0.082238 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6012 on 10736 degrees of freedom
## Multiple R-squared:  0.823,  Adjusted R-squared:  0.8223
## F-statistic:  1161 on 43 and 10736 DF,  p-value: < 2.2e-16
```

4.5.2 Stepwise Model Confidence Intervals

```
stw.model.conf
```

```
##                       2.5 %       97.5 %
## (Intercept)     3.211026394  3.423884533
## hour01         -0.705777933 -0.549037038
## hour02         -1.215226626 -1.056772119
## hour03         -1.742176911 -1.580876202
## hour04         -2.013384298 -1.851025149
## hour05         -1.006407588 -0.848562369
## hour06          0.209890114  0.367016273
## hour07          1.195515864  1.352226204
## hour08          1.824008596  1.980390741
## hour09          1.496558675  1.653081418
## hour10          1.168862680  1.326012291
## hour11          1.287772330  1.446055168
## hour12          1.475494788  1.635064118
## hour13          1.449474949  1.610425000
## hour14          1.366526484  1.528539618
## hour15          1.425022148  1.587340244
## hour16          1.687135770  1.849070459
## hour17          2.100653854  2.261647266
## hour18          2.016868484  2.176805446
## hour19          1.724673301  1.882873603
## hour20          1.424976440  1.582286771
## hour21          1.165899502  1.322569880
## hour22          0.922190315  1.078579819
## hour23          0.521952963  0.678215327
## month.L         0.651677911  0.748304457
## month.Q        -0.422414228 -0.255483406
## month.C         0.082117898  0.167530202
## month^4        -0.076531522  0.012085007
## month^5        -0.212496092 -0.129869713
## month^6        -0.033922711  0.045587412
## month^7         0.121395096  0.201391945
## month^8        -0.072591494  0.006466684
## month^9        -0.059998566  0.018918535
## month^10        0.005196073  0.084157154
## month^11       -0.044581694  0.033949624
## year2012        0.460791009  0.507024554
## weatherGood    -0.083348733 -0.027113558
## weatherAverage -0.599142667 -0.503880774
## weatherPoor    -1.284674358  1.077723669
```

```
## temp            0.023252232  0.030254717
## humidity       -0.004136865 -0.002490104
## workingdayYes  -0.101438121 -0.050984345
## windspeed      -0.005192272 -0.002152635
## holidayYes     -0.134490754  0.008079730
```

4.6 Model Interpretation

For the Stepwise model, keeping the temp and humidity constant a unit increase in windspeed is associated with a multiplicative change of 0.997 units in median of counts. The 95% confidence interval for the multiplicative change is = (0.994, .998)

Keeping the temp and windspeed constant a unit increase in humidity is associated with a multiplicative change of 0.997 unit in median of counts. The 95% confidence interval for the multiplicative change is =(0.996, .998)

Keeping humidity and temperature constant a unit change in temperature is associated with a multiplicative change of 1.03 units in median of counts. The 95% confidence interval for the multiplicative change is = (1.02, 1.031)

Stepwise Model

$$\begin{aligned}\mu\{log(count)\} = \hat{\beta}_0 + \hat{\beta}_1(hour) + \hat{\beta}_2(month) + \\ \hat{\beta}_3(year) + \hat{\beta}_4(weather) + \hat{\beta}_5(temp) + \hat{\beta}_6(humidity) + \\ \hat{\beta}_7(workingday) + \hat{\beta}_8(windspeed) + \hat{\beta}_9(holiday)\end{aligned} \tag{1}$$

4.7 Conclusion

To conclude, based on the initial EDA, it was identified that a log-linear model was a be better fit for Multiple Linear Regression. Also, once the outliers were identified and excluded, we ran LASSO, stepwise and custom model selection on the dataset. As we can see from the above sections, the results from the stepwise and custom models were very close. We find that the stepwise performed slightly better than the custom and LASSO models with the lowest standard error measures (AIC = 19668, BIC = 19996 and RMSE = 0.599).

5 Objective II Analysis

5.1 Question of Interest

As the independence assumption from Objective I appears to have been violated, the team wanted to apply the time series analysis techniques we've learned up to this point in the MSDS program in an attempt to address the issue. The primary goal was to compare and contrast the performance of auto arima models versus those that we modeled on our own.

Additionally, we wanted to compare the Kaggle submission to that from Objective I to see if our predictions were better or worse than from the prior objective.

5.2 Data Preperation

As the data had been pre-split monthly, we needed to make our predictions from the 20th of each month and on for each year of recorded bike rentals. (2011 and 2012) The approach has been summarized with the steps below.

1. Log the response variable
2. Loop through years
3. Loop through months
4. Fit AR model
5. Forcast x number of observations based on the number of rows from test data frame and impute the count from the time series forecast

5.3 Comparing Competing Models

5.3.1 Auto Arima

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,4) with non-zero mean
## Q* = 10.049, df = 3, p-value = 0.01816
##
## Model df: 9.   Total lags used: 12
```

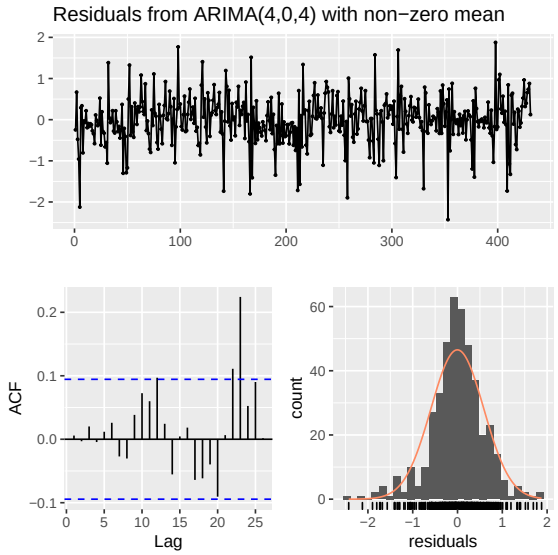


Figure 1: Auto Arima Residual Plots

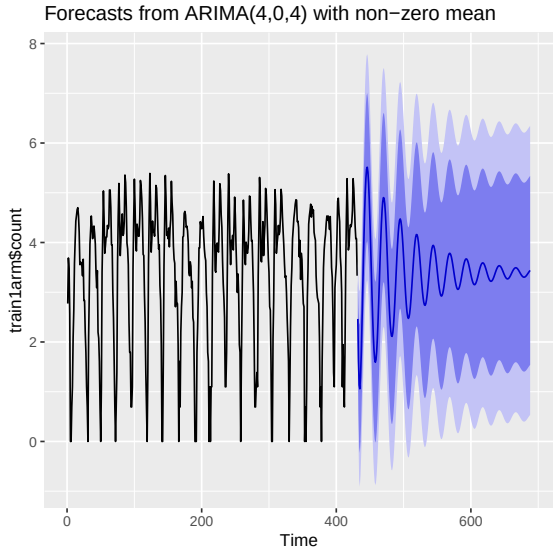


Figure 2: Auto Arima Forecasts

5.3.2 Auto Regression

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(25,0,0) with non-zero mean
## Q* = 11.131, df = 3, p-value = 0.01104
##
## Model df: 26.   Total lags used: 29
```

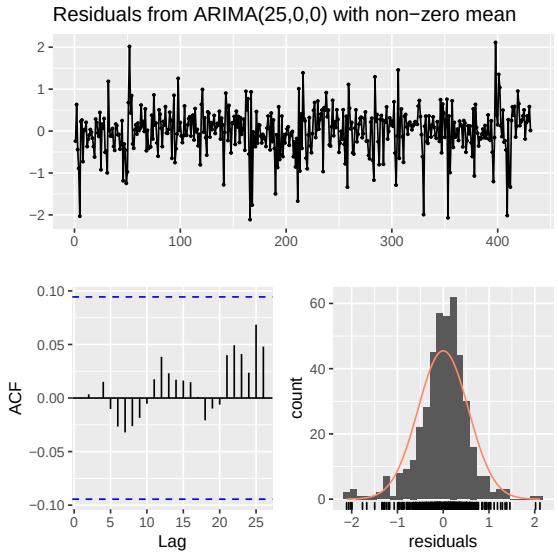


Figure 3: Arima (25) Residual Plots

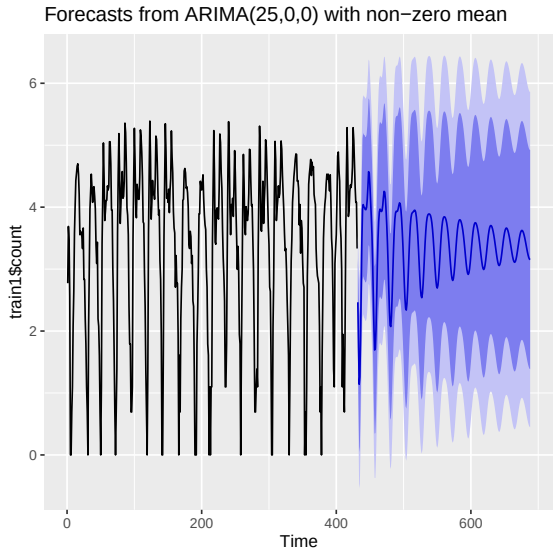


Figure 4: Arima (25) Forecasts

5.3.3 Accuracy Metrics

The accuracy metrics appear to go in favor of the AR25 model. In some year/ month combinations the difference was more obvious and was more reason to utilize the AR25 method.

Model	RMSLE	AIC	Variance
ARIMA	0.2418941	791.5698	0.3541476
AR25	0.2265693	768.3810	0.3025818

5.3.3.1 Kaggle Score

The Root Mean Squared Logarithmic Error Loss (RMSLE) for the Kaggle submission was 1.01847 for our AR25 model. We scored better than only 14.48% of all public submission for the competition which was considerably worse than our score from all of the multiple linear regression models above.

5.4 Model Assumption Assessment

Both the ARIMA and AR 25 models showed a constant mean and variance for the timespan of the observations. The auto-correlations however were noticeably better for the AR 25 graph. The error terms were also slightly improved which is the reason we elected to not go with the automated selection approach.

- Constant Mean
- Constant Variance
- Constant auto-correlations

5.5 Conclusion

Although the data set was found to be serially correlated, ultimately applying time series techniques was not enough to improve our RMSLE above and beyond the multiple linear regression techniques from objective one. Again, this likely had more to do with the fact that we were attempting to predict future observations 10 plus days in advance on a monthly basis. And with the seasonal nature of our hourly observations, the accuracy of the prediction became less accurate over time and showed a regression to the mean after only a few days' worth of predictions. The immediate observations however, within 24 to 48 hours, visually at least showed much better predictive trends.

6 Appendix

6.1 Code

6.1.1 R Code For Objective I

```r
### Library Imports
library(tidyverse)
# Date manipulation
library(lubridate)
# Plotting
library(olsrr)
# RMLSE
library(MLmetrics)


### Load the csv data
train <- read_csv('../../../data/train.csv')
test <- read_csv('../../../data/test.csv')

### Identify Dimensions

no_of_rows_trn <- dim(train)[1]
no_of_rows_tst <- dim(test)[1]

no_of_cols_trn <- dim(train)[2]
no_of_cols_tst <- dim(test)[2]

# no_of_rows_trn
# no_of_rows_tst
#
# no_of_cols_trn
# no_of_cols_tst

### Missing Data (Both)

# train
any(is.na(train))

# test
any(is.na(test))


### Log the response variable
```

```r
train$count <- log(train$count)


### Factors
train$season <- factor(train$season, labels = c("Spring", "Summer", "Fall", "Winter"))
test$season <- factor(test$season, labels = c("Spring", "Summer", "Fall", "Winter"))

table(train$season)

train$holiday <- factor(train$holiday, labels = c("No", "Yes"))
test$holiday <- factor(test$holiday, labels = c("No", "Yes"))

table(train$holiday)

train$workingday <- factor(train$workingday, labels = c("No", "Yes"))
test$workingday <- factor(test$workingday, labels = c("No", "Yes"))

table(train$workingday)

train$weather <- factor(train$weather, labels = c("Great", "Good", "Average", "Poor"))
test$weather <- factor(test$weather, labels = c("Great", "Good", "Average", "Poor"))

table(train$weather)


### Split Date-Time (Both)
train <- train %>%
  mutate(year = as.factor(format(datetime, format = "%Y")),
         month = as.numeric(format(datetime, format = "%m")),
         day = as.factor(format(datetime, format = "%d")),
         hour = as.factor(format(datetime, format = "%H")))

test <- test %>%
  mutate(year = as.factor(format(datetime, format = "%Y")),
         month = as.numeric(format(datetime, format = "%m")),
         day = as.factor(format(datetime, format = "%d")),
         hour = as.factor(format(datetime, format = "%H")))



### Convert Months to Ordered Factor (Both)
train$month <-month(train$datetime, label = TRUE, abbr = FALSE)
test$month <-month(test$datetime, label = TRUE, abbr = FALSE)
```

```r
### Outlier
outliers <- train[   train$count < median(train$count) - (sd(train$count) * 3), ]
# outliers

train %>%
  group_by(month) %>%
  summarize(mean = mean(count), sd = sd(count), median = median(count), observations = n())


# remove outliers
train <- train %>%
  filter(!datetime %in% outliers$datetime)

# Extreme Observation
# train[5555, ] # 2012-01-09 18:00:00
train <- train %>%
  filter(datetime != '2012-01-09 18:00:00')



### Model Fitting

### Custom Model
model.base.formula = count ~ weather +
                             windspeed +
                             temp +
                             month +
                             hour +
                             month:hour

  # datetime


model  <- lm(formula = model.base.formula, data = train)

plot(model)

summary(model)

ols_plot_resid_lev(model)
```

```r
### Stepwise Model
train.mod.1$datetime <- NULL
train.mod.1$datetime <- NULL

# Fit the model with all parameters
fit1 <- lm(count ~ ., data=train.mod.1)
# Fit the model with only 1 parameter
fit2 <- lm(count ~ 1, data=train.mod.1)


# stw.model <- stepAIC(fit2,direction="both",scope=list(upper=fit1,lower=fit2))
# summary(stw.model)

stw.model.formula <- count ~ hour + month + year + weather + temp + humidity +
  workingday + windspeed + holiday

stw.model <- lm(stw.model.formula,
                data = train.mod.1)

stw.sm <- summary(stw.model)
stw.sm.coe <- stw.sm$coefficients

# get the CIs for the coefficients
stw.model.conf <- confint(stw.model)
```

6.1.2 R Code For Objective II

```
### Library Imports

library(tidyverse)
# Date manipulation
library(lubridate)
# RMLSE
library(MLmetrics)
# Time Series Analysis
library(tseries)
library(forecast)


### Load the csv data
train <- read_csv('../../../data/train.csv')
test <- read_csv('../../../data/test.csv')


### Categorical Factors
train$season <- factor(train$season, labels = c("Spring", "Summer", "Fall", "Winter"))
test$season <- factor(test$season, labels = c("Spring", "Summer", "Fall", "Winter"))

table(train$season)

train$holiday <- factor(train$holiday, labels = c("No", "Yes"))
test$holiday <- factor(test$holiday, labels = c("No", "Yes"))

table(train$holiday)

train$workingday <- factor(train$workingday, labels = c("No", "Yes"))
test$workingday <- factor(test$workingday, labels = c("No", "Yes"))

table(train$workingday)

train$weather <- factor(train$weather, labels = c("Great", "Good", "Average", "Poor"))
test$weather <- factor(test$weather, labels = c("Great", "Good", "Average", "Poor"))

table(train$weather)


### Split Date-Time
train <- train %>%
```

```
  mutate(year = as.factor(format(datetime, format = "%Y")),
         month = as.numeric(format(datetime, format = "%m")),
         day = as.factor(format(datetime, format = "%d")),
         hour = as.factor(format(datetime, format = "%H")))

test <- test %>%
  mutate(year = as.factor(format(datetime, format = "%Y")),
         month = as.numeric(format(datetime, format = "%m")),
         day = as.factor(format(datetime, format = "%d")),
         hour = as.factor(format(datetime, format = "%H")))


### Convert months to ordinal factor
train$month <-month(train$datetime, label = TRUE, abbr = FALSE)
test$month <-month(test$datetime, label = TRUE, abbr = FALSE)


## 2011

### January

#### Auto Arima

train1arm <- train %>%
  filter(year == '2011' & month == 'January') %>%
  select(datetime, count)

test1arm <- test %>%
  filter(year == '2011' & month == 'January') %>%
  mutate(count = NA) %>%
  select(datetime, count)


### Log the response variable
train1arm$count = log(train1arm$count)


autoarm <- auto.arima(train1arm$count, D=1)


number = nrow(test1arm)

acf(autoarm$residuals)
```

```r
pacf(autoarm$residuals)

checkresiduals(autoarm)

fcst <- forecast(autoarm, h=number)

autoplot(fcst)

# point estimate (mean)
test1arm$count <- fcst$mean

RMSLE(y_pred = fcst$fitted, y_true = train1arm$count)

summary(autoarm)


#### AR 25

train1 <- train %>%
  filter(year == '2011' & month == 'January') %>%
  select(datetime, count)

test1 <- test %>%
  filter(year == '2011' & month == 'January') %>%
  mutate(count = NA) %>%
  select(datetime, count)


### Log the response variable
train1$count = log(train1$count)


AR25 <- arima(train1$count,order=c(25,0,0))

number = nrow(test1)

acf(AR25$residuals)
pacf(AR25$residuals)

checkresiduals(AR25)

fcst <- forecast(AR25, h=number)
```

```r
autoplot(fcst)

# point estimate (mean)
test1$count <- fcst$mean


RMSLE(y_pred = fcst$fitted, y_true = train1$count)
summary(AR25)


#
#
#

## 2012

### December

train24 <- train %>%
  filter(year == '2012' & month == 'December') %>%
  select(datetime, count)

test24 <- test %>%
  filter(year == '2012' & month == 'December') %>%
  mutate(count = NA) %>%
  select(datetime, count)


### Log the response variable
train24$count = log(train24$count)


AR25 <- arima(train24$count,order=c(25,0,0))

number = nrow(test24)

acf(AR25$residuals)
pacf(AR25$residuals)

checkresiduals(AR25)

fcst <- forecast(AR25, h=number)
```

```r
autoplot(fcst)

# point estimate (mean)
test24$count <- fcst$mean


RMSLE(y_pred = fcst$fitted, y_true = train24$count)

summary(AR25)


### Combine all of the individual data frames

combined <- data.frame(datetime=character(),
                 count=double(),
                 stringsAsFactors=FALSE)



combined <- bind_rows(test1, test2, test3, test4, test5,
                      test6, test7, test8, test9, test10,
                      test11, test12,test13, test14, test15,
                      test16, test17, test18, test19, test20,
                      test21, test22, test23, test24)


combined <- combined %>%
  mutate(count = round(exp(count)))


# combined
# write.csv(combined, file = "./ts_kaggle_submission.csv", row.names = F)
```

References

[1] Kaggle (2014). Bike sharing demand dataset. Data retrieved from the Kaggle website, https://www.kaggle.com/c/bike-sharing-demand/data.