**Analyzing the NYC Subway Dataset - Short Questions**

**Overview**

This project consists of two parts. In Part 1 of the project, I completed the questions in Problem Sets 2, 3, 4, and 5 in the Introduction to Data Science course. Copies of the code and results are included in a separate file,

*ThurstonKarenHammer\_Project1\_OptionalCodeLessons2\_5\_Plus\_ProjectCode.docx*

This current document addresses part 2 of the project. I used the improved data set, turnstile\_weather\_v2.csv as my input file.

**Section 0. Investigation**

Prior to performing the analysis on the NYC Subway Dataset, I queried the data in a variety of ways to better understand the contents. I performed the following queries using Python scripts against the enhanced data set provided for the project: turnstile\_weather\_v2.csv

Number of Rainy Days   
This query shows there were 10 total days with rain. May 17, a Tuesday, reported the most number of unit rows with rain. This information did not provide much insight into the data set. I would have needed to consult outside data sources to know if this number of rainy days in May is typical. I also do not know what other factors other than rain during May, 2011 might have contributed to rider volumes.

**OUTPUT:**number of rainy days = DATEn count(\*)

0 05-04-11 1367

1 05-14-11 332

2 05-15-11 1353

3 05-16-11 1237

4 05-17-11 1417

5 05-18-11 1373

6 05-19-11 897

7 05-20-11 1358

8 05-23-11 143

9 05-30-11 108

>>>

Number of Rainy Days grouped by days of the week  
This query shows the breakdown by days of the week (from 0-6 where Monday is 0), with the count indicating how many rows reported rain (how many rows in the file). The count is 9,585, which is the same count as reported for the ten rainy days. In this case, Wednesday had the most rows reporting rain during May, 2011. Again, no conclusions were drawn from this investigation.

**OUTPUT:**>>> runfile('C:/Users/Karen/Documents/Udacity/Data Analyst Nanodegree/Problem 1/NumRainyDaysByDaysOfWeek\_Turnstile\_Weather\_V2.py', wdir=r'C:/Users/Karen/Documents/Udacity/Data Analyst Nanodegree/Problem 1')

Rows with Rainy Days by Day of Week = day\_week count(\*)

0 0 1488

1 1 1417

2 2 2740

3 3 897

4 4 1358

5 5 332

6 6 1353

>>>

**Weather Conditions for Days When Rain is Reported**I was curious how the ‘conds’ field related to the ‘rain’ field. This query shows the breakdown of weather conditions on days when rain was reported for that calendar day. Many of the rows report ‘clear’ or other conditions that are not indicative of rain. Consulting the file turnstile-weather-variables, I found that the ‘rain’ field is set to ‘1’ if any precipitation is recorded at any time during the day. It is possible that an individual row could indicate ‘Clear’ conditions while the ‘rain’ field is set to ‘1’. **INTERIM CONCLUSION:** At this point in my investigation, I concluded that I should use the ‘Conds’ field instead of the ‘rain’ field.

**OUTPUT:**>>> runfile('C:/Users/Karen/Documents/Udacity/Data Analyst Nanodegree/Problem 1/QueryWeatherConditionsForDaysWhenRainReported\_Turnstile\_Weather\_V2.py', wdir=r'C:/Users/Karen/Documents/Udacity/Data Analyst Nanodegree/Problem 1')

Weather Conditions for Rainy Days = conds count(\*)

0 Clear 459

1 Haze 48

2 Heavy Rain 288

3 Light Drizzle 144

4 Light Rain 1880

5 Mostly Cloudy 1326

6 Overcast 4382

7 Partly Cloudy 36

8 Rain 961

9 Scattered Clouds 61

>>>

**Weather Conditions for Days When No Rain is Reported**This query shows the breakdown of weather conditions on days when no rain was reported for that calendar day. Some of the rows returned were for ‘Light Drizzle’ and ‘Mist’ which indicates that the ‘rain’ field is not changed to a value of ‘1’ for these two conditions. **INTERIM CONCLUSION:** At this point in my investigation, I had additional evidence to indicate that I should use the ‘conds’ field to assess ridership rather than the ‘rain’ field for more accurate results.

**Weather Conditions for Specific Time When Precipitation is Reported to be greater than zero**I would have expected that weather conditions column, ‘conds’ would only include rainy conditions when the precipitation column ‘precipi’ is greater than zero, but 249 rows report ‘Overcast’ when precipitation was reported to be greater than 0. **CONCLUSION:** The ‘Precipi’ field will be used as the rain indicator and not the ‘rain’ or ‘conds’ fields.

**Section 1. Statistical Test**

* 1. ***Which statistical test did you use to analyze the NYC subway data?***I used the Mann-Whitney U-Test, a non-parametric test, to analyze the NYC subway data because I could not assume the data was normally distributed. The frequency histogram shown in Figure 3.1a shows the data appears to have a non-normal distribution.

***Did you use a one-tail or a two-tail P value?***I used a two-tailed P value because I did not predict which group would have the higher median prior to running the test. See <http://www.graphpad.com/guides/prism/6/statistics/index.htm?stat_checklist_mannwhitney.htm> for an explanation of when to use a one-tailed versus two-tailed P value.

***What is the null hypothesis?*Null hypothesis**: Since I am using the Mann-Whitney U-Test, this test requires the null hypothesis be that there is no difference in the two sample populations, or in this case, that the number of people riding the NYC subway is the same when it is not raining versus when it is raining.

***What is your p-critical value?***I used the standard p-critical value of .05. The Udacity Lesson 3 Notes describing the MannWhitney U Test had a warning about p values calculated by scipy since it does a one-tailed test, so I will need to double the value from scipy:

|  |
| --- |
| *Normally, the formulation of the hypothesis when using the MannWhitney U test is twotailed,so be certain to double the p generated by the scipy function in order to report the proper pvalue.* |

* 1. ***Why is this statistical test applicable to the dataset? In particular, consider the assumptions that the test is making about the distribution of ridership in the two samples.***The handout from the Udacity Data Analysis course which provides additional detail for this test describes the MannWhitney U test as

|  |
| --- |
| *“a nonparametric test than can be used to test, for two populations with unknown distributions, if we draw randomly from each distributions, whether one distribution is more likely to generate a higher value than the other.”* |

This description matches our situation. We have two populations with unknown distributions, and only know the distribution over one month, drawn at random.

***1.3 What results did you get from this statistical test? These should include the following numerical values: p-values, as well as the means for each of the two samples under test.***

***NOTE****: I had to use the original input data file used for the class, not the enhanced turnstile\_weather\_v2.csv, because the p value was returned as NAN, see Piazza @339. This code and resulting output illustrate the test just as well, but with slightly different input data.*

See Appendix A for the code used to run this test. The output was:

Precipitation > Zero Mean =1105.45, No Precipitation Mean=1090.28, U=1924409167, p=0.019

***1.4 What is the significance and interpretation of these results?***  
My resulting p value from scipy is .019, and after doubling the value to get a two-tailed result, it is .038, less than the p-critical value, indicating that if the null hypothesis were true, we would see the sample values only 3.8% of the time. This p-value indicates that we should reject the null hypothesis, and conclude that there IS a difference in the distribution of the two samples. The mean for days with precipitation is higher (1105.45) than the mean on dry days (1090.28), so the difference seems to be slightly in favor of higher numbers of riders on rainy days.

(The p value of .019 is reported to two significant digits, citing recommendations of statistical authorities Bailar and Mosteller, 1988.)

The U value is very large. According to the Udacity Lesson 3 notes, smaller values of *U* “are suggestive of more extreme deviations from the null hypothesis.” This very large U value seems to suggest that there were not extreme deviations from the null hypothesis, even though the p value was very low.

My U value of 1,924,409,167 seems too large for the sample size (9,585 \* 33,064 = 316,918,440 is the max U value possible), but since I changed to a different data set, I did a quick check of the same size and found it to be larger than the original set:

**Number of rows with precip=44,104 Number of rows without precip = 87,847**

The maximum U value in this case would be 44,104 \* 87,847 which is 3,874,404,088. According to an explanation I found at <http://www.graphpad.com/guides/prism/6/statistics/index.htm?how_the_mann-whitney_test_works.htm>,

***the value of U should be about one half the maximum if the null hypothesis is true.***

**My U value is 1,924,409,167 which is just under one half of the maximum:**

**1,924,409,167 / 3,874,404,088 = .497**

This explanation from graphpad seems to contradict the Udacity Lesson 3 notes, unless the small amount below one half is significant, in which case U can be considered small.

I calculated the following values in addition to U and p:

**Precip Mean=1105.4464**

**Precip Var=5619274.0418**

**Precip Median=282.0000**

**No Precip Mean=1090.2788**

**No Precip Var=6656676.1959**

**No Precip Median=278.0000**

The mean and median values of the two samples are close, but not the same:

**Precip Mean/No Precip Mean = 1105.45 / 1090.28 = 1.01**

**Precip Median / No Precip Median = 282/278 = 1.01**

**This information indicates that the two samples are not very different, but different enough to generate a low p value.**  If the U value as a percentage of U(max) is significant, then my conclusion is to reject the null hypothesis and instead say that there is a difference between the numbers of people who ride the subway when there is precipitation versus when the weather is dry.

**Section 2. Linear Regression**

***2.1 What approach did you use to compute the coefficients theta and produce prediction for ENTRIESn\_hourly in your regression model:***

1. ***Gradient descent (as implemented in exercise 3.5)***
2. ***OLS using Statsmodels***
3. ***Or something different?***

I used Gradient Descent as implemented in exercise 3.5 to compute the coefficients theta and produce prediction for ENTRIESn\_hourly in my regression model. See Figure 3.3.

***2.2 What features (input variables) did you use in your model? Did you use any dummy variables as part of your features?***I used ‘hour’, ‘precipi’, ‘fog’ and ‘weekday’ as the input variables in my model. I also added ‘UNITS’ with dummy variables as part of my features.

***2.3 Why did you select these features in your model? We are looking for specific reasons that lead you to believe that the selected features will contribute to the predictive power of your model.***

* ***Your reasons might be based on intuition. For example, response for fog might be: “I decided to use fog because I thought that when it is very foggy outside people might decide to use the subway more often.”***
* ***Your reasons might also be based on data exploration and experimentation, for example: “I used feature X because as soon as I included it in my model, it drastically improved my R2 value.”***

I added ‘precipi’ because I found in my exploration of the data prior to analysis that it was a more reliable indicator than ‘rain’.  I also used the ‘hour’ and ‘weekday’ features because from personal experience riding public transportation, I know that the time of day is a predictor of rider volume. I also used ‘fog’ because foggy conditions can mean wet conditions. The ‘UNITS’ input variable was the dummy variable, and it generated about 200 columns in the ‘features’ array, one for each unique column value of ‘UNITS’ in the input file. I experimented with other combinations of features such as location (‘latitude’ and ‘longitude’), but these did not increase the R\*\*2 value.

***2.4 What are the coefficients (or weights) of the non-dummy features in your linear regression model?***

The final coefficients on the last of the 25 iterations was:

Hour: 849.11917577, Precipi: -60.12599773 Fog: -36.12648452, Weekday: 445.24372557

To see how well these coefficients can predict ridership (ENTRIESn\_hourly), I used row 453 (chosen randomly) in the file:

Where Hour=12, Precipi=0, Fog=0 and Weekday=1:

12\*(837.92908065)+0\*(-60.12599773)+0\*(-36.12648452)+1\*(446.05590831)=3352+1176=10,055 predicted riders.

The actual number of entries (ENTRIESn) for this record is 2,486.

These coefficients are interesting because the two that are negative (reduce the ridership), are the weather-related coefficients of ‘Precipi’ and ‘Fog’. The ‘Weekday’ coefficient is very strong, and the ‘Hour’ coefficient is nearly twice as large as the ‘Weekday’ coefficient. Based on the negative influence of the weather coefficients on ridership, the linear regression model will tend to predict that fewer people will ride the subway when there is precipitation or fog.

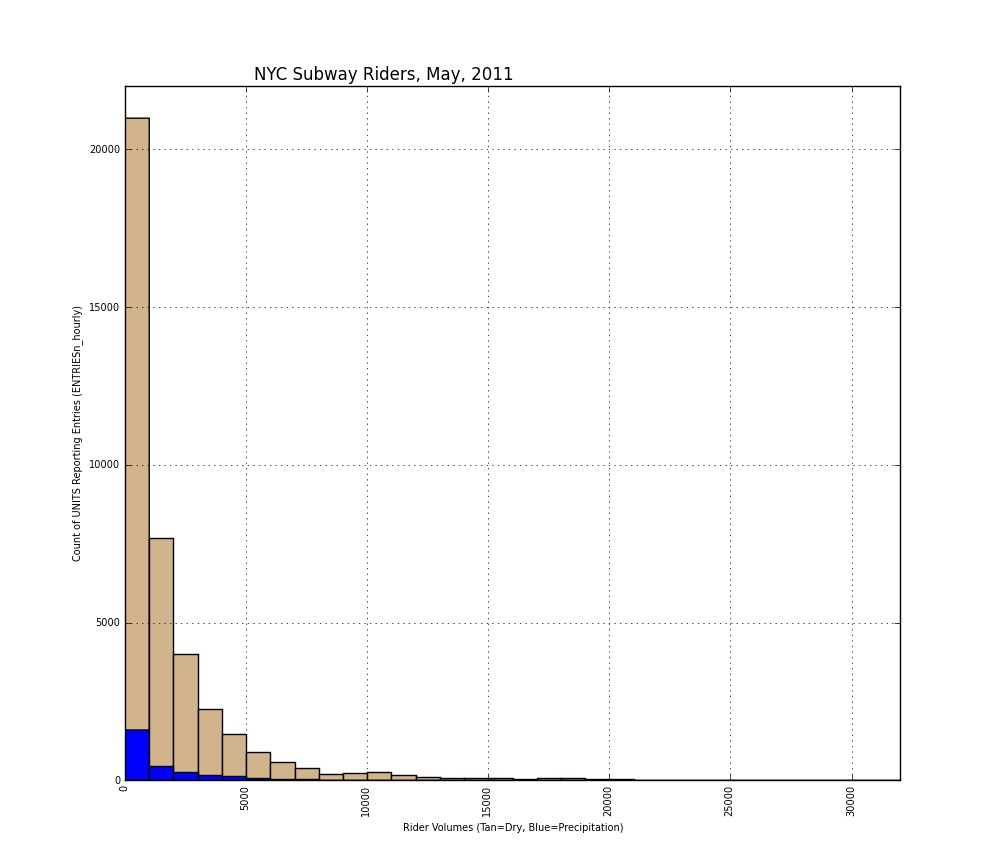
***2.5 What is your model’s R\*\*2 (coefficients of determination) value?***My model’s R\*\*2 value is 0.482. The ideal value would be closer to 1.

***2.6 What does this R\*\*2 value mean for the goodness of fit for your regression model? Do you think this linear model to predict ridership is appropriate for this dataset, given this R\*\*2  value?***The R\*\*2 value of .482 is not as close to 1 as I would like to have achieved, but I was not successful in finding features that would increase the value. This seems to indicate that this linear model was not very valuable or appropriate for this dataset or that the data was too limited. I tried running the model with only one tenth the data, and the model performed even more poorly, giving an R\*\*2 value much lower.

Additionally, when I used the coefficients generated by the model (see question 2.4 above), my predicted values were not very close to the actual values of ENTRIESn\_hourly, additional indication that the model did not yield very accurate coefficients.

Section 3. Visualization

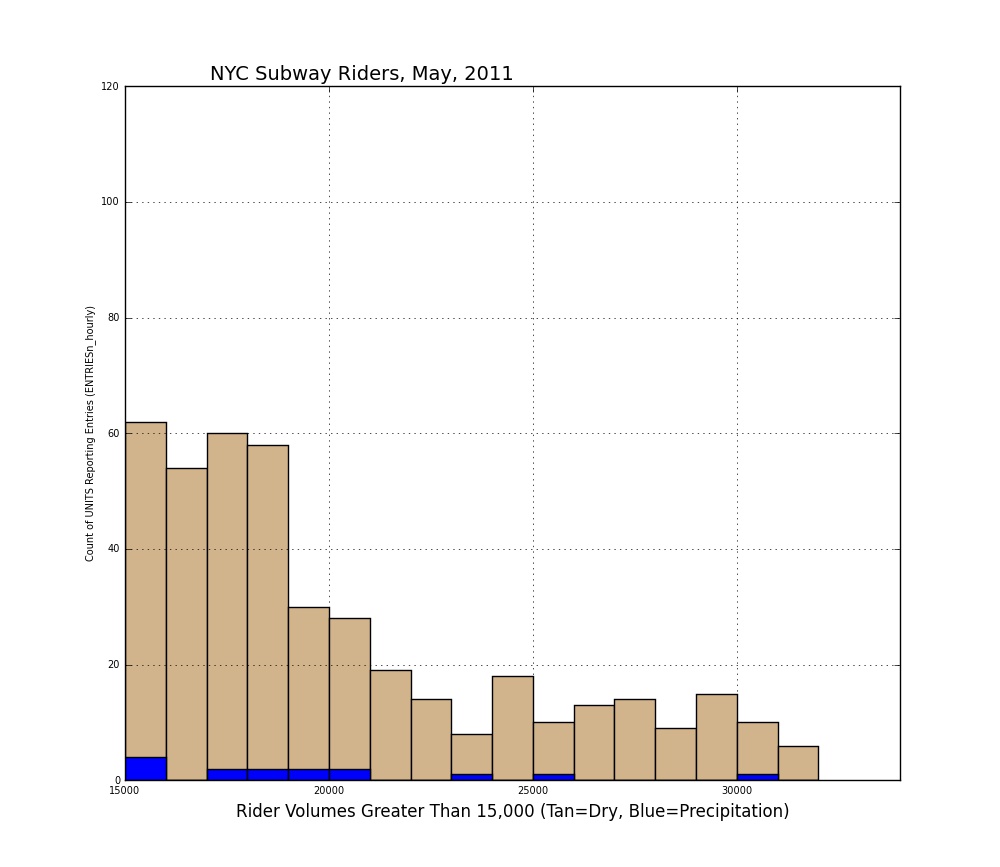
* 1. ***The histograms in Figure 3.1.a, 3.1.b, and 3.1c show rider volumes for days with and without precipitation.***The volume of ridership (value of ENTRIESn\_hourly) is on the x-axis and the number of units reporting is on the y-axis.

******

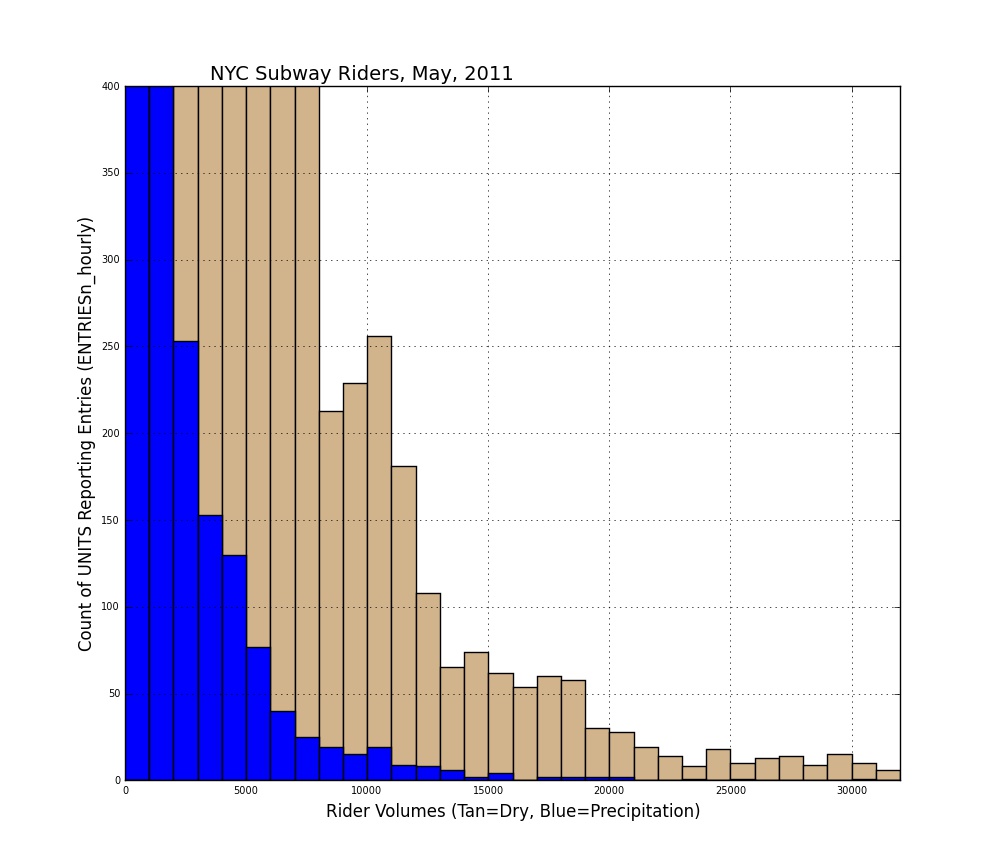
**Figure 3.1a: # of Units Reporting Rider Volumes on Rainy and Dry Days in May, 2011**

This frequency histogram shows the number of units reporting rider volume data for each range of rider volumes.

I ran the script for this histogram three times, the first time selecting rows for all volumes of riders. However, there were very few units reporting large volumes of riders for both dry and rainy days (‘precipi’ = 0 and ‘precipi’ > 0). It was difficult to tell from this visualization whether the dry or rainy days reported more than 15,000 riders, due to the very small number of units reporting these volumes, so I then ran the script only selecting units with more than 15,000 riders. See Figure 3.1b below for the result.

****

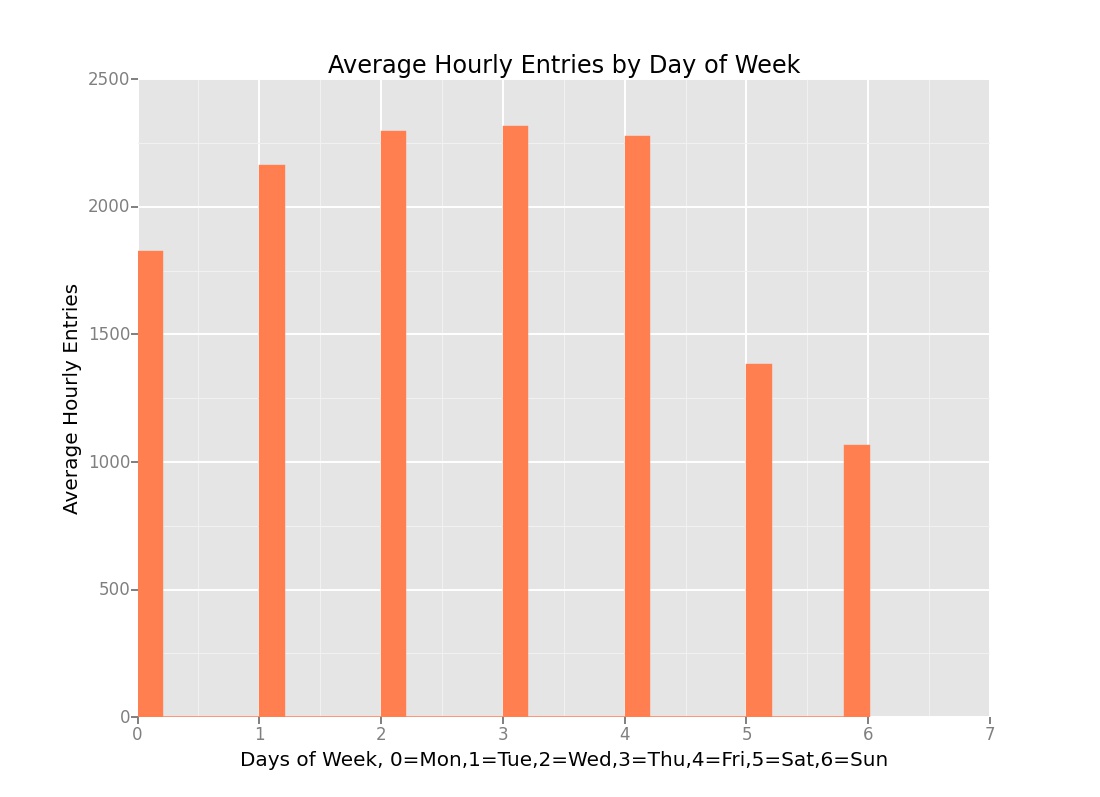
**Figure 3.1b # of Units Reporting Rider Volumes greater than 15,000 on Dry and Rainy Days in May, 2011**This frequency histogram shows the number of units reporting rider volumes greater than 15,000. Although more units reported high volumes on dry days, both dry and rainy days had at least some units reporting very large rider volumes. The highest for dry days was over 32,000 and for rainy days was about 30,000.

**

**Figure 3.1c: # of Units Reporting Rider Volumes on Days with/without Precipitation in May, 2011 (detail view < 400 units)**

The last view of rider volumes in the histogram below shows similar detail, but was the result of running the first script for the histogram but specifying a different range on the y-axis which showed greater detail for between 0 and 400 units reporting. The number of units reporting < 2,000 volume is not captured on this view because of the zoomed in scale.

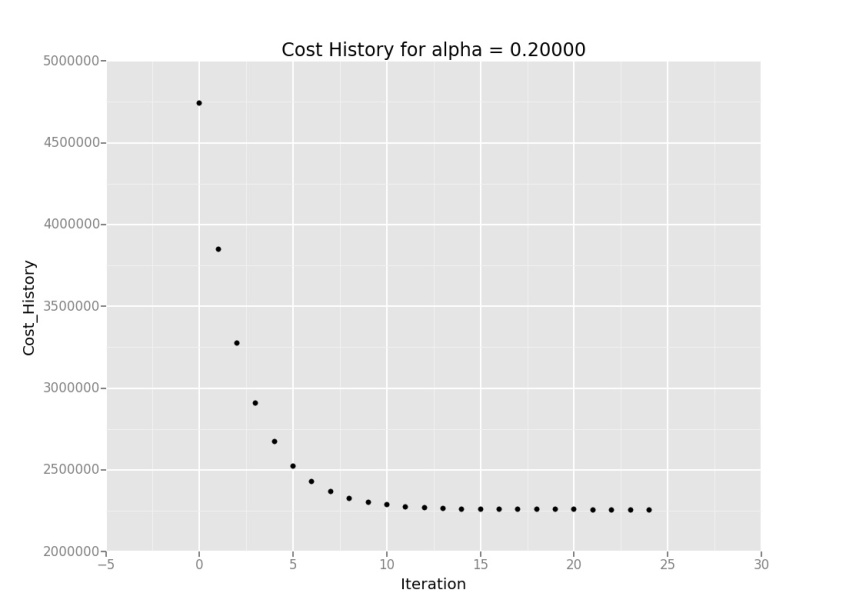
* 1. ***The bar chart in Figure 3.2 shows ridership by day-of-week***The volume of riders (average of ENTRIESn\_hourly values) is on the y-axis and each day is on the x-axis.

******

**Figure 3.2 Average hourly entries per unit (station) by day of week**

Sunday (day 6) is the day with the lowest total number of riders reported, with Saturday having the next fewest. Tuesday through Friday were each at least double the Sunday hourly averages, with Wednesday and Thursday (days 2 and 3) having the highest average hourly entries of riders during May, 2011. Averages were used since there were not an equal number of days in the file for each day of the week, and additionally, May has a Monday holiday (Memorial Day) which may have also influenced the data. This figure does not provide any insight into ridership on rainy versus dry days.

***3.3 The cost history, shown on the y-axis in Figure 3.3 is output from the gradient descent function***

******

***Figure 3.3 Cost history for gradient descent linear regression, alpha=0.2 for 25 iterations, R\*\*2 = .482***The graph above illustrates the gradient descent output using the features 'hour', 'precipi', 'fog', and ‘weekday' to predict the number of riders. From experience, I know that the time of day is a significant factor in determining the number of riders on public transportation. Figure 3.2 revealed that the day of the week influenced volumes, but the ‘weekday’ feature yielded a higher R\*\*2 value than the ‘day\_week’ feature. I experimented with different features, but these four features, used together, gave the best result. I used ‘UNIT’ because I suspect the location of the station might in itself predict ridership, especially if the station is located where working people need the subway to get to work. I tried ‘fog’ because I wondered if reduced visibility might have an effect on ridership. The four factors together don’t predict ridership very well, but other combinations were even less effective, including fewer factors or more than these four. An R\*\*2 value close to 1.0 is the desired result.

**Section 4. Conclusion**

***4.1 From my analysis and interpretation of the data, I cannot conclude that there is no difference in ridership on days with precipitation versus days without***

A quick look at the data prior to analysis suggested that possibly more people ride the NYC subway when it is dry. After a thorough analysis, this initial hunch was shown to be incorrect. Although I rejected the null hypothesis for the Mann-Whitney U Test which told me there was a difference in the two sample populations, the mean for riders on days with precipitation was actually higher than the mean value of riders on dry days.

***4.2 What analyses lead you to this conclusion?***I used the Mann-Whitney U-Test, which assumes as the null hypothesis that two populations are the same. I rejected the null hypothesis due to the low p value of .038. I also performed a linear regression which showed that possibly fewer people ride the subway when it is raining since the parameter coefficients for weather-related variable were negative. However, the R\*\*2 value was only .482, so it was not a very strong indicator. The means of the two populations were very close, with the mean on days with precipitation being slightly higher, which contradicts the results of the Gradient Descent test.

**Section 5. Reflection**

***5.1 Potential shortcomings of the data set and the methods of analysis***During my pre-analysis of the data I noticed discrepancies between the fields ‘rain’, ‘conds’ and ‘precipi’. For example, there were records with ‘rain’ = ‘0’, but with ‘precip’ greater than ‘0’. The total number of riders for this situation was 3,351,053 spread over multiple UNIT rows. This number of riders would not have changed the results of my analysis, but for a given UNIT may have been statistically significant although I did not test this hypothesis.

The methods of analysis that I used were limited by the limited amount of data and by the source of the data. To make more confident predictions, more than one month of data could be used, and a thorough analysis would include trend analysis to understand if ridership is changing (increasing or decreasing in volume) or staying the same over many months or years. In addition, a more thorough analysis would incorporate data from other sources detailing local special events or criminal activity on the subway that could affect ridership. Another question to ask on future analysis would be if the same units with high rider volumes on rainy days also had high volumes on dry days. It could be possible that the unit reporting the data had unique rider behavior different from other rider behavior for other units.

***5.2 (Optional) Do you have any other insight about the dataset that you would like to share with us?***The ENTRIESn\_hourly value was somewhat uncertain in its meaning. UNIT rows were sampled every four hours, not every hour, so it was unclear if the actual number of riders should be multiplied by four to get the actual total. For purposes of comparing ‘rain’ or ‘precipi’ fields, this uncertainty had no bearing on the analysis.

NOTE: I had trouble getting ggplot to plot values instead of counts on the y axis in plot 3.2. I found this post which helped: <http://forums.udacity.com/questions/100155698/anyone-having-any-real-luck-with-geom_bar-in-ggplot>.

**Appendix. Python Code Documentation**

**Code and results for Section 1.3 Mann\_Whitney U-Test:**

import numpy as np``

import scipy

import scipy.stats

import pandas

def mann\_whitney\_test(filename):

turnstile\_weather = pandas.read\_csv(filename)

x = (turnstile\_weather['ENTRIESn\_hourly'][turnstile\_weather['precipi']>0])

y = (turnstile\_weather['ENTRIESn\_hourly'][turnstile\_weather['precipi']==0])

rain\_mean = np.mean(x)

norain\_mean = np.mean(y)

rain\_var = np.var(x)

norain\_var - np.var(y)

rain\_median = np.median(x)

norain\_median = np.median(y)

print("Precip Mean=%7.4f, Precip Var=%7.4f, Precip Median=%7.4f, No Precip Mean=%7.4f, No Precip Var=%7.4f, No Precip Median=%7.4f" % (rain\_mean,rain\_var,rain\_median,norain\_mean,norain\_var,norain\_median))

U, p = scipy.stats.mannwhitneyu(x,y)

return U, p

rain\_mean, norain\_mean, U, p = mann\_whitney\_test('C:\Users\Karen\\turnstile\_data\_master\_with\_weather.csv')

#rain\_mean, norain\_mean, U, p = mann\_whitney\_test('C:turnstile\_weather\_v2.csv') **THIS FILE RETURNED p=nan, see @339 in Piazza**

print("U=%d, p=%.3f" % (U, p))

**OUTPUT:**

>>> runfile('C:/Users/Karen/Documents/Udacity/Data Analyst Nanodegree/Project 1/PlotMannWhitneyUTest\_Turnstile\_Weather\_v2.py', wdir=r'C:/Users/Karen/Documents/Udacity/Data Analyst Nanodegree/Project 1')

Precipitation > Zero Mean =1105.45, No Precipitation Mean=1090.28, U=1924409167, p=0.019

>>>

**Code for Figure 3.1a**

import matplotlib.pyplot as plt

import pandas

def entries\_histogram(NYCMTA\_weather):

# plt.figure(

plt.figure(figsize=(10,12))

plt.axis([0,32000,0,22000])

plt.title('NYC Subway Riders, May, 2011',fontsize=12,horizontalalignment='right')

plt.ylabel('Count of UNITS Reporting Entries (ENTRIESn\_hourly)',fontsize=7)

plt.yticks(fontsize=7)

plt.xlabel('Rider Volumes (Tan=Dry, Blue=Precipitation)',fontsize=7)

plt.xticks(fontsize=7, rotation='vertical')

# your code here to plot a histogram for hourly entries when it is not raining

NYCMTA\_weather['ENTRIESn\_hourly'][NYCMTA\_weather['precipi']==0].hist(histtype='bar', bins=[0,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,14000,15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,26000,27000,28000,29000,30000,31000,32000], color=['tan'])

# your code here to plot a histogram for hourly entries when it is raining'

NYCMTA\_weather['ENTRIESn\_hourly'][NYCMTA\_weather['precipi']>0].hist(histtype='bar', bins=[0,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,14000,15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,26000,27000,28000,29000,30000,31000,32000], color=['blue'])

return plt

NYCMTA\_weather = pandas.read\_csv('C:turnstile\_weather\_v2.csv')

plt = entries\_histogram(NYCMTA\_weather)

plt.legend(loc='upper right')

plt.show()plt.show()

**Code for Figure 3.1b**

import matplotlib.pyplot as plt

import pandas

def entries\_histogram(NYCMTA\_weather):

plt.figure(figsize=(10,12))

plt.axis([15000,34000,0,120])

plt.title('NYC Subway Riders, May, 2011',fontsize=14,horizontalalignment='right')

plt.ylabel('Count of UNITS Reporting Entries (ENTRIESn\_hourly)',fontsize=7)

plt.yticks(fontsize=7)

plt.xlabel('Rider Volumes Greater Than 15,000 (Tan=Dry, Blue=Precipitation)',fontsize=12)

plt.xticks(fontsize=7)

# your code here to plot a histogram for hourly entries when it is not raining

NYCMTA\_weather['ENTRIESn\_hourly'][NYCMTA\_weather['ENTRIESn\_hourly'] > 15000][NYCMTA\_weather['precipi']==0].hist(histtype='bar',bins=[0,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,14000,15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,26000,27000,28000,29000,30000,31000,32000],color=['tan'])

# your code here to plot a histogram for hourly entries when it is raining'

NYCMTA\_weather['ENTRIESn\_hourly'][NYCMTA\_weather['ENTRIESn\_hourly'] > 15000][NYCMTA\_weather['precipi']>0].hist(histtype='bar',bins=[0,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,14000,15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,26000,27000,28000,29000,30000,31000,32000],color=['blue'])

return plt

NYCMTA\_weather = pandas.read\_csv('C:turnstile\_weather\_v2.csv')

plt = entries\_histogram(NYCMTA\_weather)

plt.legend(loc='upper right')

plt.show()

**Code for Figure 3.1c**

import matplotlib.pyplot as plt

import pandas

def entries\_histogram(NYCMTA\_weather):

plt.figure(figsize=(10,12))

plt.axis([0,32000,0,400])

plt.title('NYC Subway Riders, May, 2011',fontsize=14,horizontalalignment='right')

plt.ylabel('Count of UNITS Reporting Entries (ENTRIESn\_hourly)',fontsize=12)

plt.yticks(fontsize=7)

plt.xlabel('Rider Volumes (Tan=Dry, Blue=Precipitation)',fontsize=12)

plt.xticks(fontsize=7)

# your code here to plot a histogram for hourly entries when it is not raining

NYCMTA\_weather['ENTRIESn\_hourly'][NYCMTA\_weather['precipi']==0].hist(histtype='bar', bins=[0,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,14000,15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,26000,27000,28000,29000,30000,31000,32000],color=['tan'])

# your code here to plot a histogram for hourly entries when it is raining'

NYCMTA\_weather['ENTRIESn\_hourly'][NYCMTA\_weather['precipi']>0].hist(histtype='bar', bins=[0,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,14000,15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,26000,27000,28000,29000,30000,31000,32000],color=['blue'])

return plt

NYCMTA\_weather = pandas.read\_csv('C:turnstile\_weather\_v2.csv')

plt = entries\_histogram(NYCMTA\_weather)

plt.legend(loc='upper right')

plt.show()

**Code for Figure 3.2**

from pandas import \*

from ggplot import \*

import numpy as np

import matplotlib.pyplot as plt

def plot\_weather\_data(turnstile\_weather):

temp = turnstile\_weather.groupby('day\_week').mean()['ENTRIESn\_hourly']

d = {'day\_week': temp.index, 'ENTRIESn\_hourly' : temp.values}

tw\_s = pandas.DataFrame(d)

# scale\_y\_continuous(labels='comma') + \

plot = ggplot(tw\_s, aes(x='day\_week', weight = 'ENTRIESn\_hourly'))

plot = plot + geom\_bar(color = 'coral', fill = 'coral')

plot = plot + xlab('Days of Week, 0=Mon,1=Tue,2=Wed,3=Thu,4=Fri,5=Sat,6=Sun') + ylab('Average Hourly Entries')

plot = plot + ggtitle('Average Hourly Entries by Day of Week')

return plot

weather\_data = pandas.read\_csv('c:turnstile\_weather\_v2.csv')

MyPlot = plot\_weather\_data(weather\_data)

print (MyPlot)

**Code for Figure 3.3: Gradient Descent**

import numpy as np

import pandas

from ggplot import \*

import scipy

import matplotlib.pyplot as plt

import sys

def compute\_r\_squared(data, predictions):

SST = ((data-np.mean(data))\*\*2).sum()

SSReg = ((predictions-data)\*\*2).sum()

r\_squared = 1 - SSReg / SST

return r\_squared

def normalize\_features(array):

"""

Normalize the features in the data set.

"""

array\_normalized = (array-array.mean())/array.std()

mu = array.mean()

sigma = array.std()

return array\_normalized, mu, sigma

def compute\_cost(features, values, theta):

"""

Compute the cost of a list of parameters, theta, given a list of features

(input data points) and values (output data points).

"""

# m is number of data points, aka values

m = len(values)

sum\_of\_square\_errors = np.square(np.dot(features, theta) - values).sum()

cost = sum\_of\_square\_errors / (2\*m)

return cost

def gradient\_descent(features, values, theta, alpha, num\_iterations):

"""

Perform gradient descent given a data set with an arbitrary number of features.

"""

m = len(values)

cost\_history = []

for i in range(num\_iterations):

predicted\_values = np.dot(features,theta)

theta = theta - alpha / m \* np.dot((predicted\_values - values), features)

# cost = compute\_cost(features,values,theta)

# cost\_history.append(cost)

cost\_history.append(compute\_cost(features, values, theta))

return theta, pandas.Series(cost\_history)

def plot\_cost\_history(alpha, cost\_history):

cost\_df = pandas.DataFrame({

'Cost\_History': cost\_history,

'Iteration': range(len(cost\_history))

})

return ggplot(cost\_df, aes('Iteration', 'Cost\_History')) + \

geom\_point() + ggtitle('Cost History for alpha = %.3f' % alpha )

def predictions(dataframe):

# Select Features (try different features!)

features = dataframe[['hour', 'precipi', 'fog', 'weekday']]

# Add UNIT to features using dummy variables

dummy\_units = pandas.get\_dummies(dataframe['UNIT'], prefix='unit')

features = features.join(dummy\_units)

# Values

values = dataframe['ENTRIESn\_hourly']

m = len(values)

features, mu, sigma = normalize\_features(features)

features['ones'] = np.ones(m) # Add a column of 1s (y intercept)

# Convert features and values to numpy arrays

features\_array = np.array(features)

values\_array = np.array(values)

# Set values for alpha, number of iterations.

alpha = 0.2 # please feel free to change this value

num\_iterations = 25 # please feel free to change this value

# Initialize theta, perform gradient descent

theta\_gradient\_descent = np.zeros(len(features.columns))

theta\_gradient\_descent, cost\_history = gradient\_descent(features\_array,

values\_array,

theta\_gradient\_descent,

alpha,

num\_iterations)

# -------------------------------------------------

# Uncomment the next line to see your cost history

# -------------------------------------------------

plot = plot\_cost\_history(alpha, cost\_history)

predictions = np.dot(features\_array, theta\_gradient\_descent)

r\_squared = compute\_r\_squared(values\_array, predictions)

return predictions, plot, r\_squared

weather\_data = pandas.read\_csv('C:\Users\Karen\\turnstile\_data\_master\_with\_weather.csv')

MyPredictions, MyPlot, r\_squared = predictions(weather\_data)

print (MyPlot)

print (MyPredictions)

print ("R\*\*2=%.2f" % r\_squared)

**OUTPUT:**

>>>runfile('C:/Users/Karen/Documents/Udacity/Data Analyst Nanodegree/Project 1/PlotGradientDescentLinearRegression.py', wdir=r'C:/Users/Karen/Documents/Udacity/Data Analyst Nanodegree/Project 1')

<ggplot: (33447081)>

[-1723.53529767 -1234.05383428 -255.0909075 ..., 510.3706259

999.85208929 1489.33355267]

R\*\*2=0.482