Section 0 – References

https://www.dropbox.com/s/1lpoeh2w6px4diu/improved-dataset.zip?dl=0

https://bespokeblog.wordpress.com/2011/07/11/basic-data-plotting-with-matplotlib-part-3-histograms/

http://www.datarobot.com/blog/ordinary-least-squares-in-python/

http://stackoverflow.com/questions/30244251/plt-hist-errors-on-subsetted-data

http://stats.stackexchange.com/questions/116315/problem-with-mann-whitney-u-test-in-scipy

http://www.weather-and-climate.com/average-monthly-precipitation-Rainfall-inches,New-York,United-States-of-America

Section 1 – Statistical Test

We want to investigate the relationship between NYC subway ridership and time and weather. Prior to further analysis, descriptive statistics of the dataset was examined to ensure no missing data, obvious data errors, and to get a general sense of the data structure. The data used for the below analyses is the improved dataset. The time span of our data is the month of May, 2011, recorded at 4-hour time intervals. Sample size is 42649. The mean number of entries hourly is 1886.59, with standard deviation of 2952.39, and max of 32814 (descriptive statistic output at end of report). We'll slice the data further in various ways to examine how, if at all, the weather affects riders' behaviors.

To answer the question whether rain affects ridership in a statistically significant way, Mann-Whitney-U test and Welsh's t-Test were performed (c1) on our two samples, ridership during rainy time versus no rain. We are not assuming equal variance. At critical value of 5%, we reject both null hypotheses that the two have the same mean (two-tail t-test: p=4.63e-7) and the two samples come from same distribution (two-tailed U-test: p=5.48e-6). Now we know when it rains, more people ride the subway. Let's consider the amount of rain, defined by precipitation. Light rain is less than or equal to 0.03, and heavy rain is more than 0.03. U-test again, and two-tail p-value is 1.32e-39. Reject null hypothesis again. So we can see that when it rains lightly (<=0.03), the subway stations are busiest.

Sample	Mean of Hourly Entries
No Rain	1845.54
Yes Rain	2028.20
Light Rain	2151.04
Heavy Rain	1246.01

Section 2 -- Linear Regression

Given what we learned in part 1, we want to further investigate the relationship between ridership and various variables in the dataset, and choose predictor(s) to forecast mean hourly entries. I first examined potential variables individually, with gradient decent, I gathered their R^2 values, summarized below. Using combination of the four variables with highest R^2 ('rain', 'tempi', 'weekday', 'hour'), our prediction model has R^2 of 0.1045. These four variables make sense that they have higher R^2 than others. Weather condition as well as rush hours or not, intuitively help explain the mean number of hourly entries. (C2_GradientDescent)

Variable	R^2
'rain'	0.000667
'precipi'	0.000766
'meanprecipi'	0.001271
'tempi'	0.00803
'weekday'	0.02115
'hour'	0.08225

ENTRIESn_hourly= 50.294*rain + 539.590*tempi +1518.18*weekday + 2007.48*hour

I also tried using OLS Regression to fit data. Only using weather-related variables yield R^2 as low as Gradient Descent. Using dummy might introduce multicollinearity. After examining various combinations, these variables are used to predict ENTRIESn_hourly: intercept, rain, tempi, weekday and hour. All variables are statistically significant, and R^2 is 0.104. (C2_OLS)

ENTRIESn_hourly=-391.42 + 80.62*rain + 5.87*tempi + 951.90*weekday + 120.40*hour

After trying both regression models, I do not believe linear models are appropriate to model our data. Neither methods have an attractive R^2, although all variables are statistically significant.

Section 3 – Visualization

Now, let's examine the data graphically. Histograms and scatter plots are useful tools. Histogram helps investigate frequency and skew of data. We know our ENTRIESn_hourly is highly skewed right, histogram can show graphically how the skew is distributed. Scatter plot can help examine relationship between variables. The variable of interest here are ENTRIESn_hourly and hour, only looking at the the station where max ENTRIESn_hourly in the dataset occurs. How does ENTRIESn_houly distribute given hour? Do they peak at rush hour? Do we have an outlier just one day that's skewing the data? A scatter plot can help answer these questions.

Figure 1 shows the histogram of ENTRIESn_hourly, when it rains versus when it doesn't rain. We can clearly see that, when it does not rain, frequency of the first bin (hourly entries: 0-500) is much higher than when it does rain. So when it does not rain, the stations are less busy. Figure 2 zooms in on the right tail.

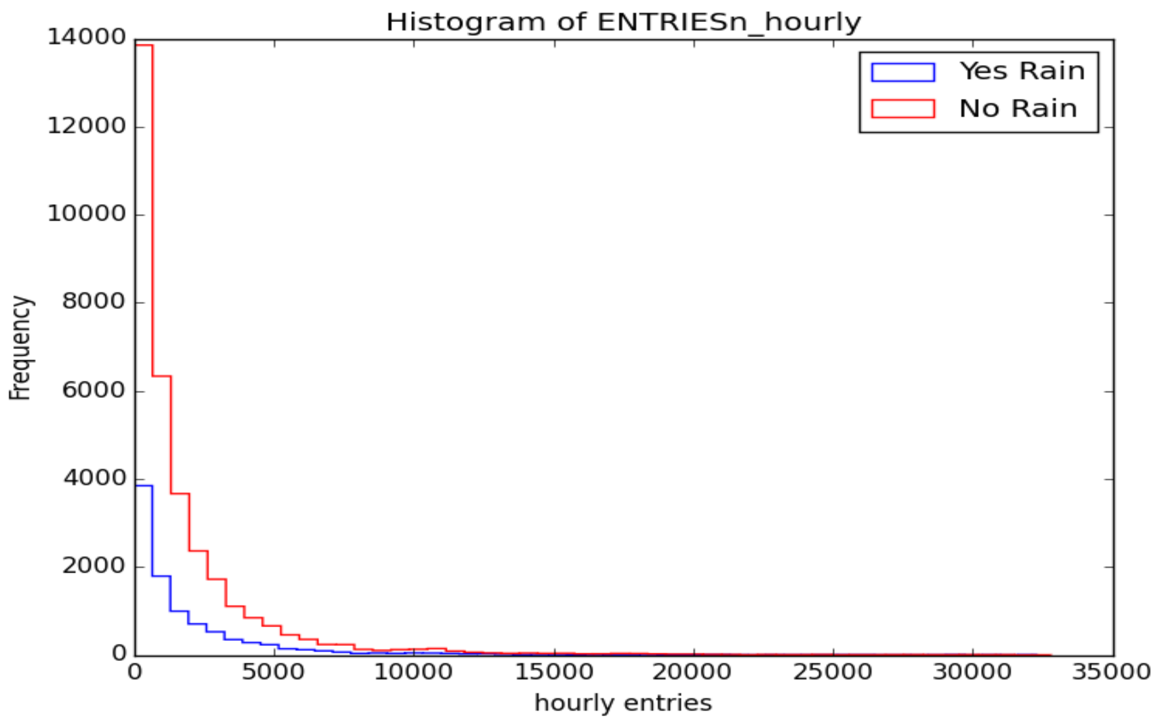


Figure 1

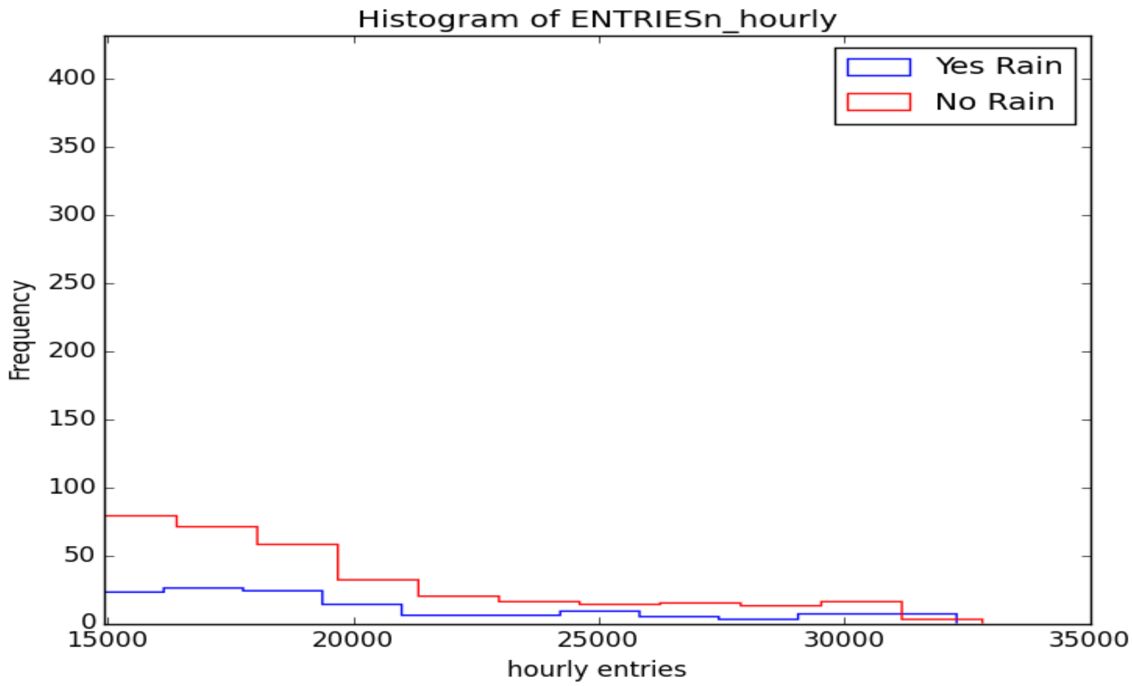


Figure 2

Figure 3 is a scatter plot of ENTRIESn_hourly by hour at the station where the max ENTRIESn_hourly lie, to examine its behavior (R084). The horizontal line indicates the 75th percentile ENTRIESn_hourly of the dataset, and we can see that this station is above the line on almost record. The busiest time at R084 is 8pm, and from noon to midnight, all the records are above the 75th percentile line.

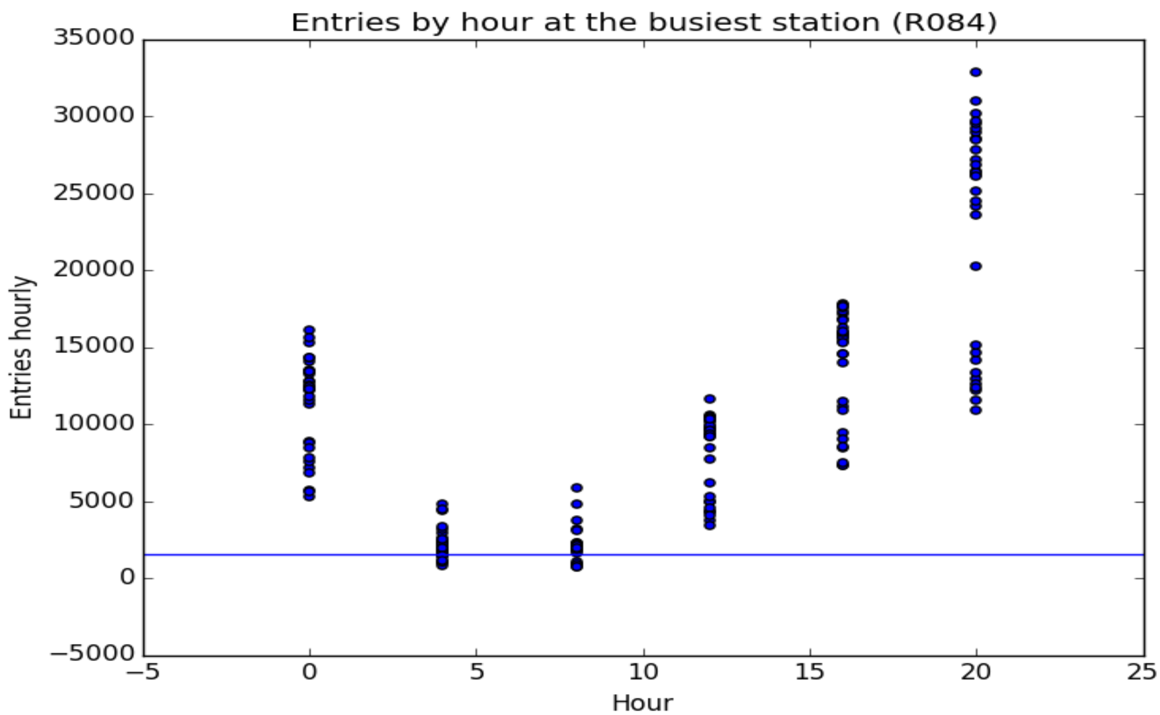


Figure 3

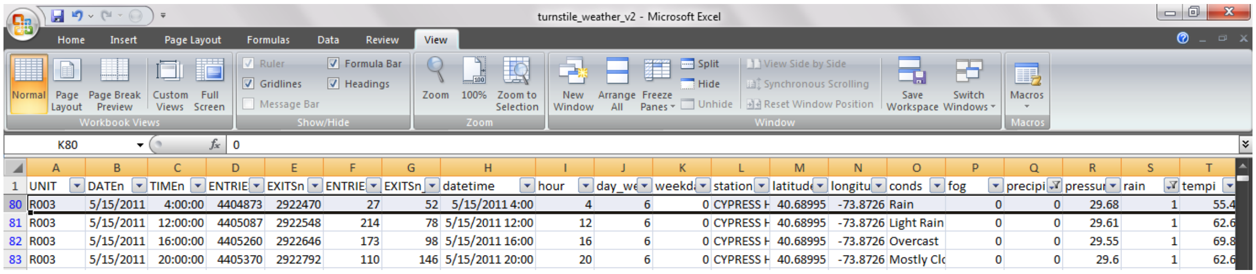
Section 4 – Conclusion

The question we are trying to answer is, do more people ride the NYC subway when it rains. Both our statistical tests confirm that the answer is yes. We showed that the mean of ridership is statistically different whether it rains, namely higher when it rains. Counter-intuitively, with light rain NYC subway is statistically significantly busier than heavy rain. I believe this is partially due to a much smaller sample size (8 heavy rain observations).

I am hesitant to make conclusion based on our linear regression model, due to multi-colinearity issues and extremely low coefficient of determination. I tried gradient descent as well as OLS, both yield the same results. Our data is likely to be better modeled by nonlinear regression.

Section 5 – Reflection

Our observations might be unusually low in precipitation. The average May precipitation is approximately 4 inches, and the average of our dataset is 0.005 inches. Maybe rain would be a more significant predictor if we had more precipitation in our observed data. Also, the rain column and precipitation seem contradicting. The rain column would indicate 1 (yes rain), and precipitation column would indicate 0 (0 inches of rain observed).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	UNIT	DATEn	TIMEn	ENTRIE	EXITSn	ENTRIE	EXITSn_	datetime	hour	day_we	weekda	station	latitude	longitu	conds	fog	precipi	pressur	rain	tempi
80	R003	5/15/2011	4:00:00	4404873	2922470	27	52	5/15/2011 4:00	4	6	0	CYPRESS H	40.68995	-73.8726	Rain	0	0	29.68	1	55.4
81	R003	5/15/2011	12:00:00	4405087	2922548	214	78	5/15/2011 12:00	12	6	0	CYPRESS H	40.68995	-73.8726	Light Rain	0	0	29.61	1	62.6
82	R003	5/15/2011	16:00:00	4405260	2922646	173	98	5/15/2011 16:00	16	6	0	CYPRESS H	40.68995	-73.8726	Overcast	0	0	29.55	1	69.8
83	R003	5/15/2011	20:00:00	4405370	2922792	110	146	5/15/2011 20:00	20	6	0	CYPRESS H	40.68995	-73.8726	Mostly Clc	0	0	29.6	1	62.6

Descriptive Statistic:

	ENTRIESn	EXITSn	ENTRIESn_hourly	EXITSn_hourly	\
count	4.264900e+04	4.264900e+04	42649.000000	42649.000000	
mean	2.812486e+07	1.986993e+07	1886.589955	1361.487866	
std	3.043607e+07	2.028986e+07	2952.385585	2183.845409	
min	0.000000e+00	0.000000e+00	0.000000	0.000000	
25%	1.039762e+07	7.613712e+06	274.000000	237.000000	
50%	1.818389e+07	1.331609e+07	905.000000	664.000000	
75%	3.263049e+07	2.393771e+07	2255.000000	1537.000000	
max	2.357746e+08	1.493782e+08	32814.000000	34828.000000	

	hour	day_week	weekday	latitude	longitude	\
count	42649.000000	42649.000000	42649.000000	42649.000000	42649.000000	
mean	10.046754	2.905719	0.714436	40.724647	-73.940364	
std	6.938928	2.079231	0.451688	0.071650	0.059713	
min	0.000000	0.000000	0.000000	40.576152	-74.073622	
25%	4.000000	1.000000	0.000000	40.677107	-73.987342	
50%	12.000000	3.000000	1.000000	40.717241	-73.953459	
75%	16.000000	5.000000	1.000000	40.759123	-73.907733	
max	20.000000	6.000000	1.000000	40.889185	-73.755383	

	fog	...	pressurei	rain	tempi	\
count	42649.000000	...	42649.000000	42649.000000	42649.000000	
mean	0.009824	...	29.971096	0.224741	63.103780	
std	0.098631	...	0.137942	0.417417	8.455597	
min	0.000000	...	29.550000	0.000000	46.900000	
25%	0.000000	...	29.890000	0.000000	57.000000	
50%	0.000000	...	29.960000	0.000000	61.000000	
75%	0.000000	...	30.060000	0.000000	69.100000	
max	1.000000	...	30.320000	1.000000	86.000000	

	wspdi	meanprecipi	meanpressurei	meantempi	meanwspdi	\
count	42649.000000	42649.000000	42649.000000	42649.000000	42649.000000	
mean	6.927872	0.004618	29.971096	63.103780	6.927872	
std	4.510178	0.016344	0.131158	6.939011	3.179832	
min	0.000000	0.000000	29.590000	49.400000	0.000000	
25%	4.600000	0.000000	29.913333	58.283333	4.816667	
50%	6.900000	0.000000	29.958000	60.950000	6.166667	
75%	9.200000	0.000000	30.060000	67.466667	8.850000	
max	23.000000	0.157500	30.293333	79.800000	17.083333	

	weather_lat	weather_lon
count	42649.000000	42649.000000
mean	40.728555	-73.938693
std	0.065420	0.059582
min	40.600204	-74.014870
25%	40.688591	-73.985130
50%	40.720570	-73.949150
75%	40.755226	-73.912033
max	40.862064	-73.694176

Code:

C1_ Mann-Whiteney-U Test

```
 def mann_whitney_plus_means(t):
    u=scipy.stats.mannwhitneyu(YesRain['ENTRIESn_hourly'], NoRain['ENTRIESn_hourly'],
   use_continuity=True)[0]
    m_u = len(YesRain['ENTRIESn_hourly'])*len(NoRain['ENTRIESn_hourly'])/2
      sigma_u = np.sqrt ( len(YesRain['ENTRIESn_hourly']) *len(NoRain['ENTRIESn_hourly'])
      *(len(YesRain['ENTRIESn_hourly'])+len(NoRain['ENTRIESn_hourly'])+1)/12)
    z = (u - m_u)/sigma_u
      p=pval = 2*scipy.stats.norm.cdf(z)
    print p
```

C1_Welsh’s t-Test

```
def ttest(t):
    ttest_1=scipy.stats.ttest_ind(YesRain['ENTRIESn_hourly'], NoRain['ENTRIESn_hourly'],
equal_var=False)
    if (ttest_1[1]<0.05):
    print ttest_1
      return (False, ttest_1)
    else:
      return (True, ttest_1)

ttest(t)
```

C2_OLS

```
import numpy as np
import pandas
import scipy
import scipy.stats
import time
import matplotlib.pyplot as plt
import statsmodels as sm
from ggplot import *

print time.ctime()

#read csv file
t = pandas.read_csv('turnstile_weather_v2.csv')
features = t[['pressurei', 'precipi', 'tempi', 'weekday', 'hour']]
#dummy_units = pandas.get_dummies(t['UNIT'], prefix='unit')
#features = features.join(dummy_units)
features = sm.tools.tools.add_constant(features)
values = t['ENTRIESn_hourly']
```

```
mod = sm.regression.linear_model.OLS(values,features)
res = mod.fit()
print res.summary()
```

C2_Gradient Decent

```
import numpy as np
import pandas
import scipy
import scipy.stats
import time
import matplotlib.pyplot as plt
import sys
import csv
from ggplot import *

print time.ctime()

#read csv file
t = pandas.read_csv('turnstile_weather_v2.csv')


#gradient decent
def normalize_features(t):
    #Normalize the features in the data set.
    mu = t.mean()
    sigma = t.std()

    if (sigma == 0).any():
        raise Exception("One or more features had the same value for all samples, and thus could " + \
                        "not be normalized. Please do not include features with only a single value " + \
                        "in your model.")
    t_normalized = (t - t.mean()) / t.std()

    return t_normalized, mu, sigma

def compute_cost(features, values, theta):
    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):

    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)
    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(t):

    # Select Features (try different features!)
    features = t[['rain', 'meantempi', 'weekday', 'hour']]

    # Add UNIT to features using dummy variables
    #dummy_units = pandas.get_dummies(t['UNIT'], prefix='unit')
    #features = features.join(dummy_units)

    # Values
    values = t['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 = 0.1 # please feel free to change this value
    num_iterations = 75 # 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)
```

```
    plot = plot_cost_history(alpha, cost_history)


    predictions = np.dot(features_array, theta_gradient_descent)
    print gradient_descent(features_array, values_array, theta_gradient_descent, alpha, num_iterations)

    #print features_array


predictions(t)
```

C3_ Histogram:

```
YesRain =  t[t['rain']==1]
NoRain =  t[t['rain']==0]
with_rain = YesRain['ENTRIESn_hourly']
without_rain = NoRain['ENTRIESn_hourly']
plt.hist(with_rain.values, bins=20, histtype='step', color='b', label='Yes Rain')
plt.hist(without_rain.values, bins=20, histtype='step', color='r',label='No Rain')
plt.title("Histogram of ENTRIESn_hourly")
plt.xlabel("hourly entries")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

C3_Scatter Plot:

```
busy=t[t['UNIT']=='R084']
plt.scatter(busy['hour'], busy['ENTRIESn_hourly'])
plt.axhline(1537)
plt.title("Entries by hour at the busiest station (R084)")
plt.xlabel("Hour")
plt.ylabel("Entries hourly")
plt.legend()
plt.show()
```