# Analyzing the NYC Subway Dataset

Questions

Overview

This project consists of two parts. In Part 1 of the project, you should have completed the questions in Problem Sets 2, 3, 4, and 5 in the Introduction to Data Science course.
This document addresses part 2 of the project. Please use this document as a template and answer the following questions to explain your reasoning and conclusion behind your work in the problem sets. You will attach a document with your answers to these questions as part of your final project submission.

## Section 0. References

Please include a list of references you have used for this project. Please be specific - for example, instead of including a general website such as stackoverflow.com, try to include a specific topic from Stackoverflow that you have found useful.
Answer:

Histograms with matplotlib
http://pandas.pydata.org/pandas-docs/stable/visualization.html#histograms

Working with text in matplotlib
http://matplotlib.org/users/pyplot_tutorial.html

Remove non-letters from a string
http://www.codecodex.com/wiki/Remove_non-letters_from_a_string#Python

SQL AVG() Function
http://www.w3schools.com/sql/sql_func_avg.asp

Python Dictionaries
http://learnpythonthehardway.org/book/ex39.html

Arithmetics with dates
https://docs.python.org/2/library/datetime.html

Python ggplot examples
https://pypi.python.org/pypi/ggplot/0.4.7

Reading from stdin and files
http://www.tutorialspoint.com/python/file_next.htm

Seaborn plotting library
http://stanford.edu/~mwaskom/software/seaborn/tutorial/quantitative_linear_models.html#plotting-simple-regression-with-regplot

Mann-Whitney U Test :
http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.mannwhitneyu.html
http://wikiofscience.wikidot.com/technology1:mann-whitney-u-test

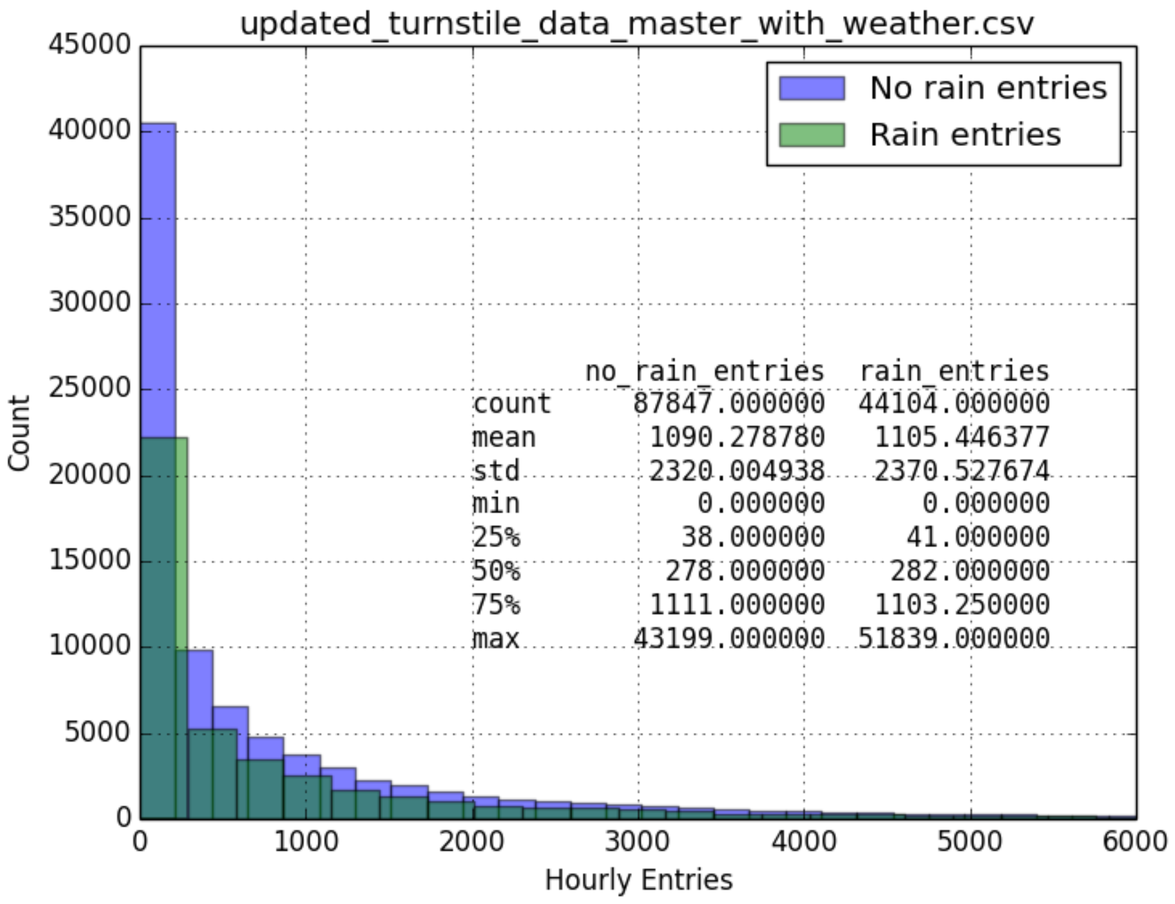
Critical values and p values
http://www.itl.nist.gov/div898/handbook/prc/section1/prc131.htm

Interpretation of r^2 and assessment of goodness of fit
http://blog.minitab.com/blog/adventures-in-statistics/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit

## Section 1. Statistical Test

1.1 Which statistical test did you use to analyze the NYC subway data? Did you use a one-tail or a two-tail P value? What is the null hypothesis? What is your p-critical value?
Answer:
I used a Mann-Whitney U test, since the distributions of entries are not normal from visual check. See Fig. 1.

The null and alternative hypothesis are considered as follows:
H0 : samples of entries of rain and no_rain scenarios come from the same distribution
HA : samples of entries of rain and no_rain scenarios come from different distributions.



```
       no_rain_entries   rain_entries
count     87847.000000   44104.000000
mean       1090.278780    1105.446377
std        2320.004938    2370.527674
min           0.000000       0.000000
25%          38.000000      41.000000
50%         278.000000     282.000000
75%        1111.000000    1103.250000
max       43199.000000   51839.000000
```

Fig.1: Histogram of turnstile entries - separated by Rain and No rain

A two-tailed test for α = 0.05 was considered, so p-critical is 0.025.
The scipy function scipy.stats.mannwhitneyu() gives p-values for one-sided, it is required to multiply the obtained p-value by 2.

1.2 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.
Answer:
The  Mann Whitney u test does not assume our data is drawn from any particular underlying probability distribution, and should work for this case where we cannot assume dataset follow normal distribution.

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.
Answer:
The script below shows the calculation of U and p for Mann Whitney u test.

```
w_rain_series  = t_w[t_w.rain == 1]['ENTRIESn_hourly']
no_rain_series = t_w[t_w.rain == 0][‘ENTRIESn_hourly']
rain_mean    = np.mean(w_rain_series)
no_rain_mean = np.mean(no_rain_series)

U, p = scipy.stats.mannwhitneyu(w_rain_series, no_rain_series)
```

and gives the following results:
(rain_mean, no_rain_mean, U, p) =
(1105.4463767458733, 1090.278780151855, 1924409167.0, 0.024999912793489721)

1.4 What is the significance and interpretation of these results?
Answer:
Since the p-value is lower than 0.025 I reject the null hypothesis, that is, we conclude that entries when rain are higher than entries without rain, with a 0.05 chance that I am making a mistake. Note this a result for a generic day, regardless if it is a working day or weekend.

Interestingly, if I restrict analysis only to working days by the following script:

```
w_rain_series_working_day  = t_w[(t_w.rain == 1) & (t_w.weekday == 1)]['ENTRIESn_hourly']
no_rain_series_working_day = t_w[(t_w.rain == 0) & (t_w.weekday == 1)]['ENTRIESn_hourly']
rain_mean_working_day    = np.mean(w_rain_series_working_day)
no_rain_mean_working_day = np.mean(no_rain_series_working_day)

U_working_day, p_working_day = scipy.stats.mannwhitneyu(w_rain_series_working_day,
no_rain_series_working_day)
```

(weekday dummy column was created and indicates whether a row has a date which is a weekend or not. See answer 2.2). In this case, the results are:
(rain_mean, no_rain_mean, U, p) =
(1198.1502978290941, 1304.536235167018, 985531035.5, 3.1874387801245787e-15)
So, for a working day, rainy days show higher average entries. And with a very small p-value indicating that H0 can be rejected with a very low chance of making a Type I error.

On the other hand, the results for weekends, that is for `(weekday == 0)` are
(rain_mean, no_rain_mean, U, p) =
(727.16610989517335, 686.57462735570289, 129024195.5, 0.00030968316696356743)
and the conclusions here are similar to the generic day, which is that on weekends people ride more often when it is raining.

The Mann-Whitney results for these three scenarios are summarized in Table 1.

Also, assuming normality of the distributions, the corresponding Welch two sample t-test results were calculated and are presented also in Table 1. In this case, the only null hypothesis rejection would be for working days.

|  | Mean Entries |  |  |  |
|---|---|---|---|---|
|  | ☔ | ☀️ | **p - value (Mann Whitney u test)** | **p-value (Welch’s two sample t-test)** |
| **generic day** | 1105.44 | 1090.27 | 0.0249999 | 0.269506 |
| **working day** | 1198.15 | 1304.53 | 3.187438E-15 | 9.036605E-10 |
| **weekend** | 727.16 | 686.57 | 0.00030968 | 0.02866 |

Table 1: mean entries in rainy and non-rainy days

## 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:
Gradient descent (as implemented in exercise 3.5)
OLS using Statsmodels
Or something different?
Answer:
I used Gradient descent as implemented in exercise 3.5.

2.2 What features (input variables) did you use in your model? Did you use any dummy variables as part of your features?

In addition to the dummy variables related to the turnstile UNIT, I wrote a program to include day_week & weekday to “turnstile_data_master_with_weather.csv” and saved this new file as “updated_turnstile_data_master_with_weather.csv”

```
import datetime

def add_weekday_day_week(filenames):

    for name in filenames:
        df = pandas.read_csv(name)
        df['date'] = df['DATEn'].apply(lambda x: datetime.datetime.strptime(x , ‘%Y-%m-%d'))
        df['day_week'] = df['date'].apply(lambda x: int(x.weekday()))
        df['weekday'] = df['day_week'].apply(lambda x: 0 if x == 5 or x == 6 else 1)
        df.to_csv("updated_"+name)
```

Adding weekday and day_week to features list:

```
features = dataframe[['rain', 'precipi', 'Hour', 'mintempi', 'maxtempi', 'day_week', 'weekday']]
```

increased r^2 from 0.458554208399 to 0.468759744238.

From Fig. 2 that was created for Problem Set 4, Isee that 30.may.2011 presented low entries. In fact, it was memorial day in the US, a national holiday. So, I added an additional dummy variable with the script below:

```
import holidays
df['holiday'] = df['date'].apply(lambda x: int(x in holidays.US()) )
```

This adds an additional column for holiday (0 = no, 1 = yes).
r^2 is improves to 0.470541486962.

The number of iterations was kept at 75.

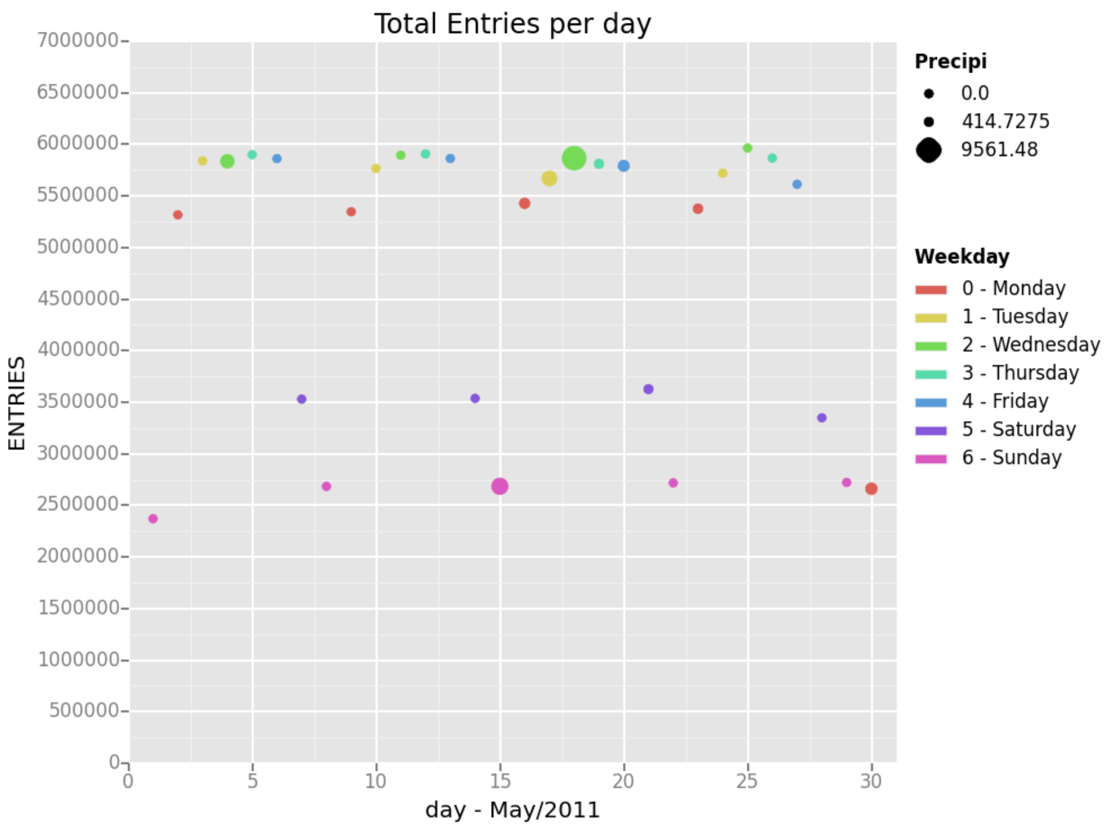


Fig.2: Entries and precipitation per day in May/2011

Finally, per the Fig.3 I see the number of entries do not show a linear relationship with the hour of day. It increases and decreases showing peaks at rush hours. So, forcing a relationship such as $Y = \ldots + \theta_H . X_H + \ldots$, can lead to high errors and not help in predicting the entries.
So, I decided to create additional dummy variables for the hours that were added to the feature set, by the following code:

```
dummy_units = pandas.get_dummies(dataframe['Hour'], prefix='hour')
features = features.join(dummy_units)
```

Removing 'Hour' from the features and with these hour dummy variables, r^2 increased to 0.514958381474.

In summary, the cumulative improvements in r^2 are as follows:

| | r^2 |
|---|---|
| original feature set | 0.4585 |
| add weekday and day_week | 0.4687 |
| add holiday | 0.4705 |
| add hour dummy variables | 0.5150 |

Table 2: r^2 improvements for different feature sets

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.”
Answer:
The choice of weekday, day_week and holiday and are based on observation of the graph in Fig.2 explained in 2.2 and justified by improvement of r^2.
The dummy variables for UNITS came already chosen with the code already set by default. Removing them decreases the r^2 a lot, to around 0.0458122001164, so entries are highly dependent on the unit.

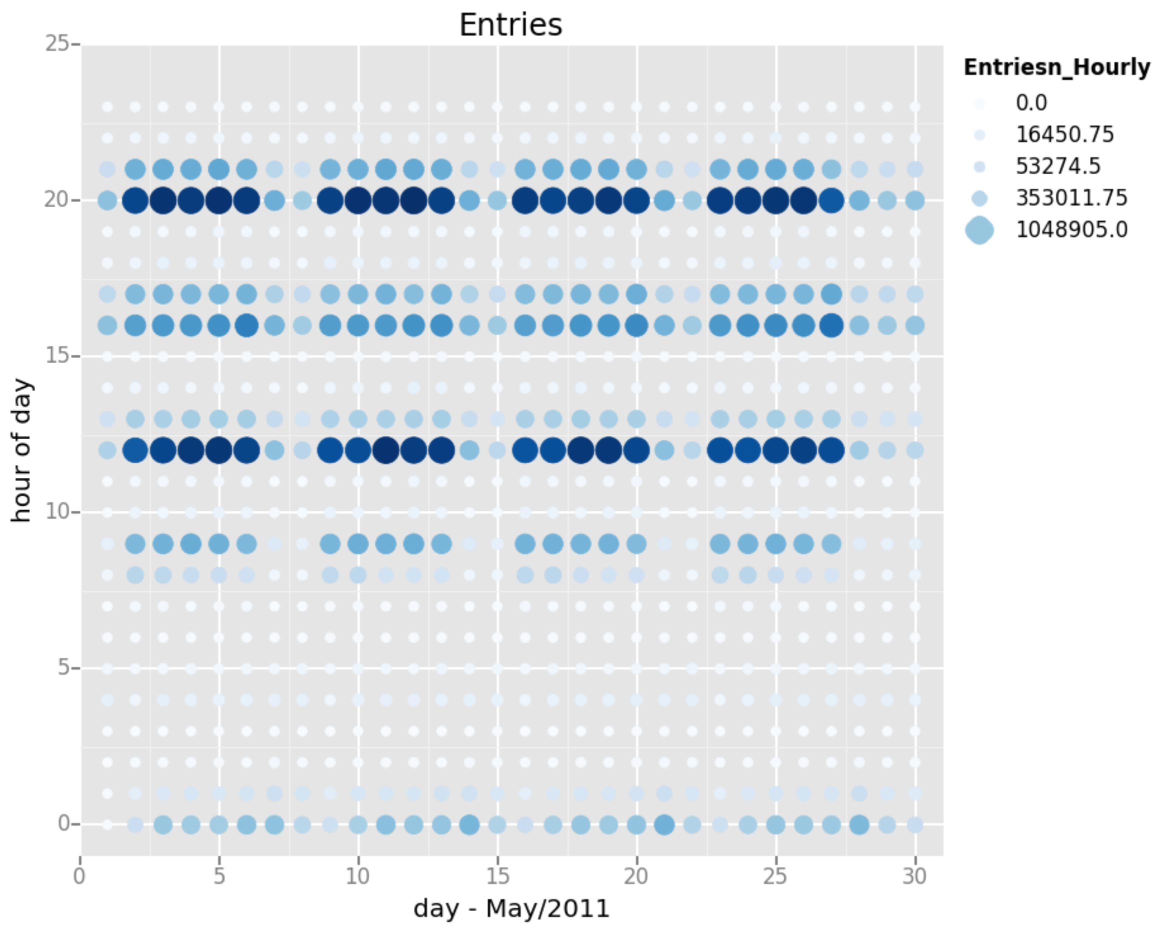


Fig.3: Entries per hour and day in May/2011

The dummy variables for hour of day were chosen, with reasons already explained in 2.2.

2.4 What are the coefficients (or weights) of the non-dummy features in your linear regression model?
Answer:
The coefficients for the non-dummy variables are:

rain:       6.0959
precipi:  1.6881
mintempi: -6.9026
maxtemp:  23.0811
day_week:   19.4411
weekday:  281.2628
holiday: -119.0045
ones: 1095.34848

The positive value for rain agrees with the first line 'generic day' of Table 1: More rain, more riders.

Now, to check the negative effect of rain on ridership for working days, I ran regression only with samples coming from those days by restricting `dataframe = dataframe[dataframe.weekday == 1]` and removing weekday from the features list. As expected, the coefficient for rain became -1.30550547, a value consistent with line 'working day' of Table 1: More rain, less riders.

2.5 What is your model's R2 (coefficients of determination) value?
Answer:
The value of r^2 is 0.514958381474.

2.6 What does this R2 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 R2  value?
Answer:
This shows that about 51% of the variation in entries can be explained by the features chosen.
I think the model crudely predicts ridership. Experimenting other columns as features did not bring significant results. I think experimenting with fields only available in the improved dataset, such as geographical location or other weather conditions could be a valid attempt.

**Section 3. Visualization**

Please include two visualizations that show the relationships between two or more variables in the NYC subway data.
Remember to add appropriate titles and axes labels to your plots. Also, please add a short description below each figure commenting on the key insights depicted in the figure.
3.1 One visualization should contain two histograms: one of  ENTRIESn_hourly for rainy days and one of ENTRIESn_hourly for non-rainy days.
You can combine the two histograms in a single plot or you can use two separate plots.
If you decide to use to two separate plots for the two histograms, please ensure that the x-axis limits for both of the plots are identical. It is much easier to compare the two in that case.
For the histograms, you should have intervals representing the volume of ridership (value of ENTRIESn_hourly) on the x-axis and the frequency of occurrence on the y-axis. For example, each interval (along the x-axis), the height of the bar for this interval will represent the number of records (rows in our data) that have ENTRIESn_hourly that falls in this interval.
Remember to increase the number of bins in the histogram (by having larger number of bars). The default bin width is not sufficient to capture the variability in the two samples.
Answer:

The histogram in Fig.1 provides the visualization requested.

This visualization shows the counts of entries corresponding to rain and no-rain in defined ranges (bin sizes). Isee that it is not shaped as a normal distribution, and that no-rain entries dominates at lower values. The code to generate this histogram is as follows:

```
plot_df = turnstile_weather[['DATEn','ENTRIESn_hourly','rain','Hour', 'precipi', 'mintempi',
'meantempi', 'maxtempi', 'weekday']]
```

```
plot_df['no_rain_entries'] = turnstile_weather[(turnstile_weather.rain == 0)]['ENTRIESn_hourly']
plot_df['rain_entries']    = turnstile_weather[(turnstile_weather.rain == 1)]['ENTRIESn_hourly']
plot_df['no_rain_entries'].plot(kind = 'hist', alpha = 0.5, bins = 200, label='No rain entries')
plot_df['rain_entries'].plot(kind = 'hist', alpha = 0.5, bins = 180, label='Rain entries')
plt.xlim([0,6000])
plt.xlabel('Hourly Entries')
plt.ylabel('Count')
plt.legend()
plt.title(filename)
plt.text(2000, 10000, plot_df[['no_rain_entries', 'rain_entries']].describe(), fontdict={'family' :
'monospace'})
```

3.2 One visualization can be more freeform. You should feel free to implement something that we discussed in class (e.g., scatter plots, line plots) or attempt to implement something more advanced if you'd like. Some suggestions are:
Ridership by time-of-day
Ridership by day-of-week
Answer:
Examples of plots with conclusions were presented in Fig. 2 and Fig. 3.
In addition, I tried the check whether the temperatures and barometric pressure had correlation with the entries and the kind of relationship.

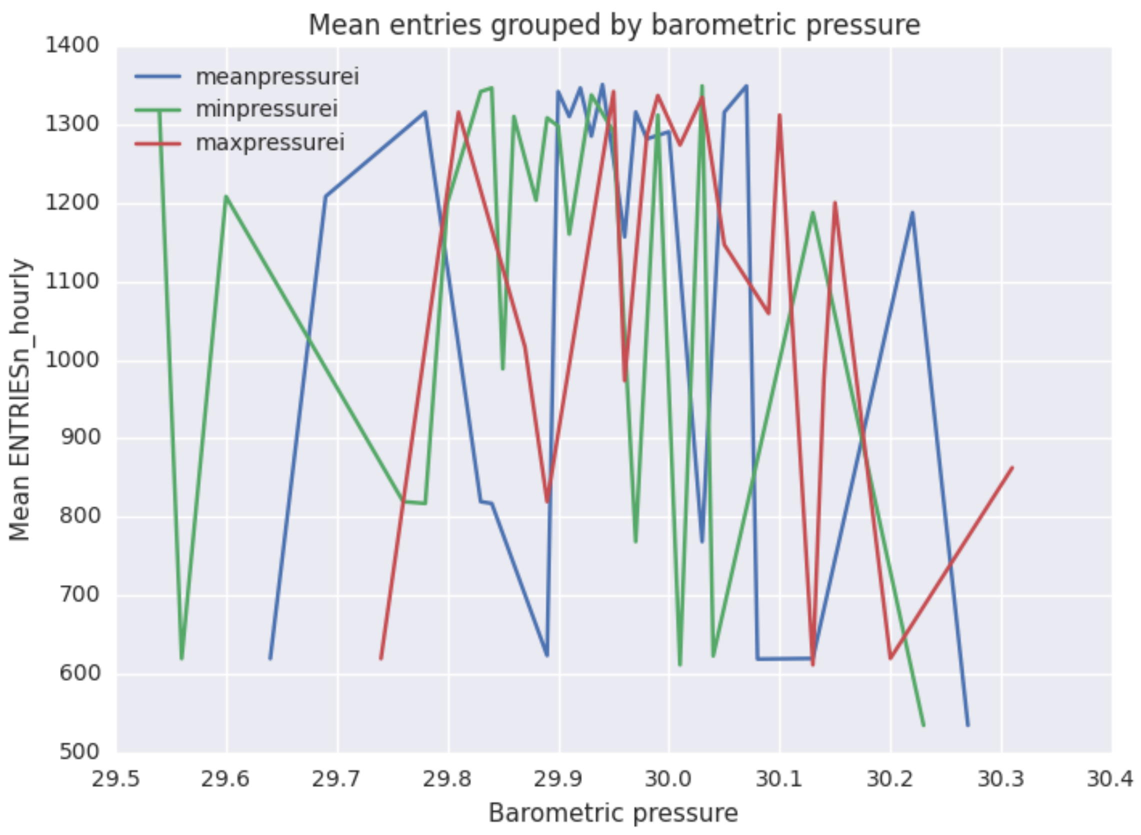


Fig.4: Mean entries grouped by temperatures

From Fig. 4 I can see the mean entries grouped by three temperature columns: meantempi, mintempi, maxtempi. By chosing only meantempi and mintempi, only a very small improvement is observed: r^2 = 0.514964513027, when compared to the last r^2 value obtained in the last features choice of answer to 2.2.

Similarly, no linearity observation can be obtained by Fig. 5, indicating that barometric pressure would not bring a significant improvement to r^2, found out to be r^2 = 0.515053815292.

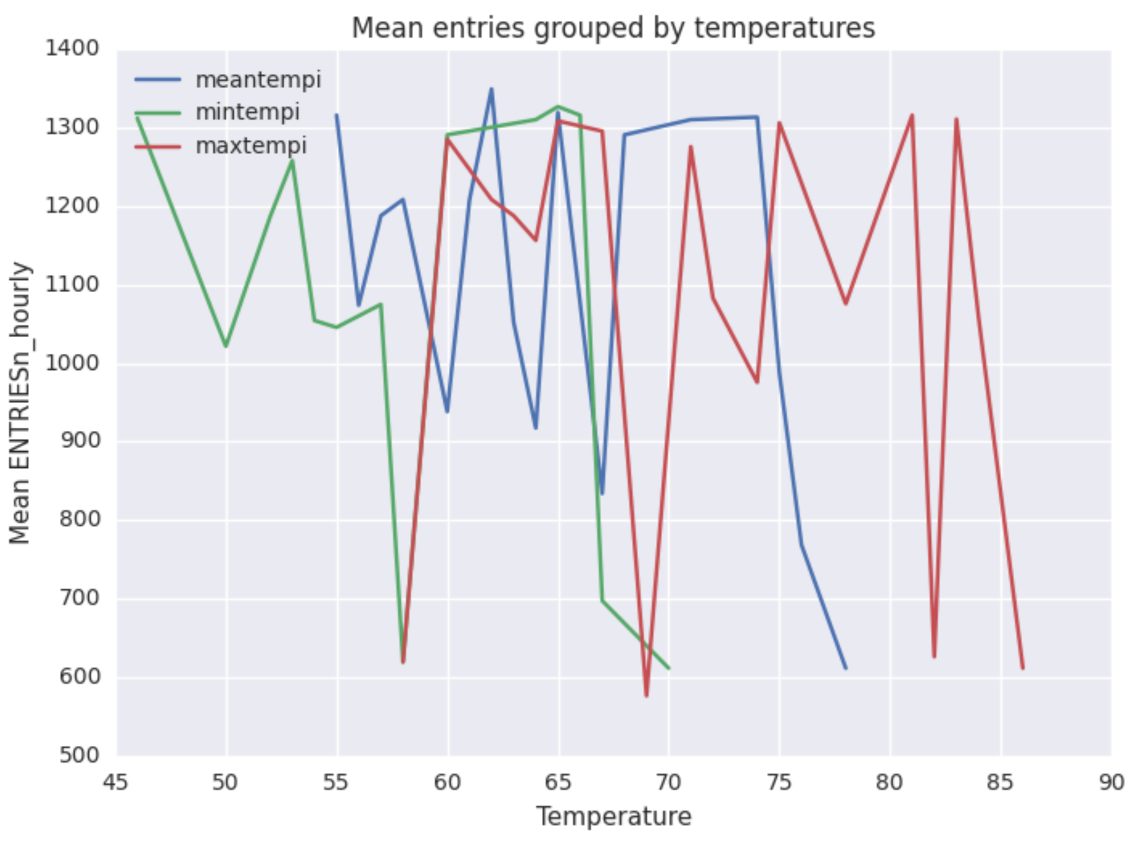


Fig.5: Mean entries grouped by barometric pressure

The code to generate Fig. 4 and Fig. 5 is as follows:

```
if False:
    plot_df_mean = plot_df.groupby(plot_df.meantempi).mean()
    plot_df_mean.reset_index(inplace=True)
    plt.plot(plot_df_mean.meantempi, plot_df_mean.ENTRIESn_hourly)

    plot_df_min = plot_df.groupby(plot_df.mintempi).mean()
    plot_df_min.reset_index(inplace=True)
    plt.plot(plot_df_min.mintempi, plot_df_min.ENTRIESn_hourly)

    plot_df_max = plot_df.groupby(plot_df.maxtempi).mean()
    plot_df_max.reset_index(inplace=True)
    plt.plot(plot_df_max.maxtempi, plot_df_max.ENTRIESn_hourly)

    plt.ylabel('Mean ENTRIESn_hourly')
    plt.xlabel('Temperature')
    plt.legend(['meantempi', 'mintempi', 'maxtempi'], loc='upper left')
    plt.title('Mean entries grouped by temperatures')

if True: # minpressurei, meanpressurei, maxpressurei
    plot_df_mean = plot_df.groupby(plot_df.meanpressurei).mean()
    plot_df_mean.reset_index(inplace=True)
    plt.plot(plot_df_mean.meanpressurei, plot_df_mean.ENTRIESn_hourly)

    plot_df_min = plot_df.groupby(plot_df.minpressurei).mean()
    plot_df_min.reset_index(inplace=True)
    plt.plot(plot_df_min.minpressurei, plot_df_min.ENTRIESn_hourly)

    plot_df_max = plot_df.groupby(plot_df.maxpressurei).mean()
    plot_df_max.reset_index(inplace=True)
    plt.plot(plot_df_max.maxpressurei, plot_df_max.ENTRIESn_hourly)
```

```
        plt.ylabel('Mean ENTRIESn_hourly')
        plt.xlabel('Barometric pressure')
        plt.legend(['meanpressurei', 'minpressurei', 'maxpressurei'], loc='upper left')
        plt.title('Mean entries grouped by barometric pressure')
```

## Section 4. Conclusion

Please address the following questions in detail. Your answers should be 1-2 paragraphs long.
4.1 From your analysis and interpretation of the data, do more people ride
the NYC subway when it is raining or when it is not raining?
Answer:
The results of Table 1 of Question 1.4 along with the analysis of coefficients of Question 2.4, support this answer.

If a generic day is randomly chosen, we have more riders when it is raining. This conclusion considers α = 0.05. If I had chosen α = 0.01 this would not be valid and the I would conclude that there is no difference between ridership volume between rainy and non-rainy days.

However, when I analyze working days and weekends separately, the conclusion is:
a) for a working day, non-rainy days have more riders than rainy days
b) for weekends, rainy days have more riders than non-rainy days

4.2 What analyses lead you to this conclusion? You should use results from both your statistical tests and your linear regression to support your analysis.
Answer:
The Mann-Whitney u tests confirm that the conclusions in 4.1 are valid since p-values are very low. In fact, for both weekends and working days, we can safely reject the null hypothesis that both come from same distribution, since $p < 0.1\%$.

Also, from answer to question 2.4, the coefficient of rain feature is positive, so for a generic day the more rain, the more riders.

It can also be noted that coefficient for weekday (281.2628) is positive and its magnitude higher than other features. It is an indication that among the chosen features, this is the most important non-dummy factor on the prediction of the number of riders, more than other features such as rain or temperature. The values of different coefficients θ can be compared because of the normalization.

Dummy variables show high coefficients, but since these variables are of 'yes'/'no' type, with most of the time being a 'no' (which means close to zero after normalization), they only affect ridership when the focus is in one particular turnstile unit. It almost seems like I am 'cheating', trying to conform a prediction formula to a bunch of discrete 0/1 variables, increasing the computational time because of quadratic nature of matrices product, instead of trying to achieve a more elegant and compact polynomial.

Other weather factors, such as barometric pressure, temperature show only marginal contributions to the prediction accuracy of ridership.

**Section 5. Reflection**

Please address the following questions in detail. Your answers should be 1-2 paragraphs long.
5.1 Please discuss potential shortcomings of the methods of your analysis, including:
Dataset,
Analysis, such as the linear regression model or statistical test.

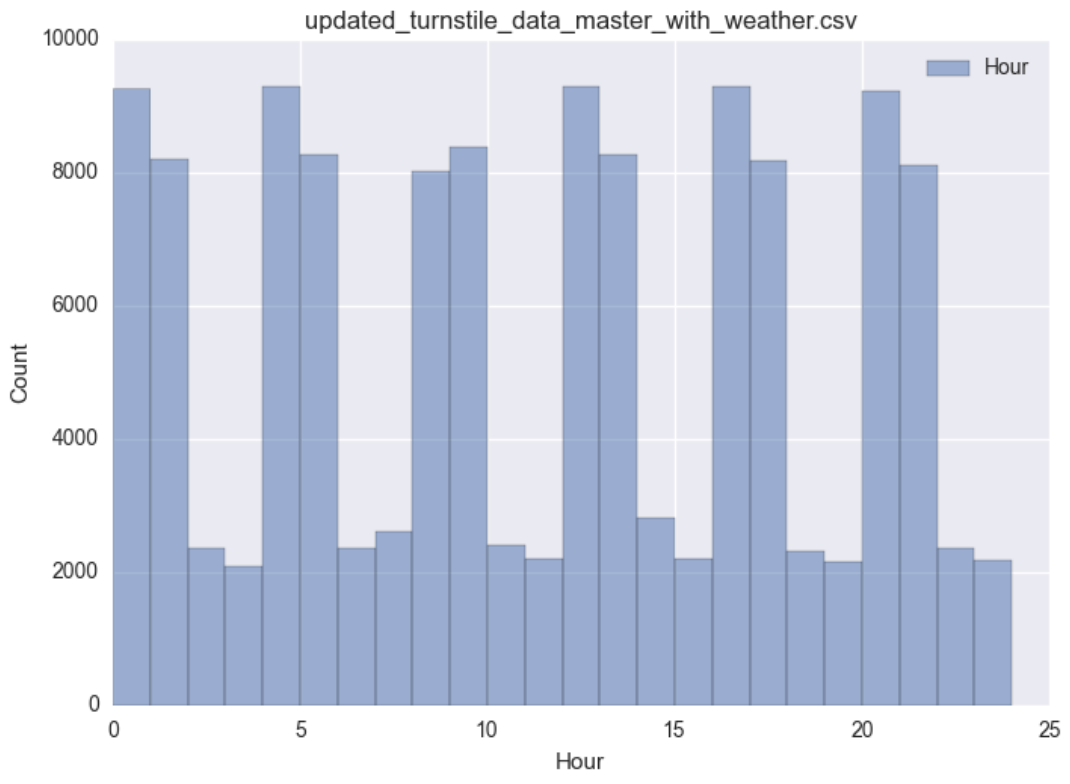


Fig.6: Hour value histogram

Answer:
the original data set showed inconsistent entries for hour. While most of the rows fit into the (0, 4, 8, 12, 16, 20) window, with a comparable number in the (1, 5, 9, 13, 17, 21) window, and the rest distributed among the remaining time slots. Initially, I was thinking this inconsistency would privilege low Hour values, but it was not the case. In any case, more checks are needed to see if this affects unevenly rain vs no-rain entries or working day versus weekend entries. In quick glance on the data I suspect these are more related to UNIT than to other factors.

Three shortcomings, that is lack of day_week, weekday and holiday, were corrected by a script that added those columns to the file. Using these columns as features of the regression model proved to be effecting in increasing the r^2.

Finally, I think that one month of data tells us only what could happen in that month in different years, that is, from data of May/11, we could predict ridership of May/12, May/13,…. .
Seasonal increases or decreases on ridership would be captured by this model.

5.2 (Optional) Do you have any other insight about the dataset that you would like to share with us?

From the improved dataset, I see that there are more columns available for analysis.
For example, ‘conds’ column (expanded into dummy columns) could be an interesting feature to check - to see if improves the accuracy of prediction.

Also, in the improved dataset, a test I would try is to check if stations in downtown have the same rain/no-rain ridership as stations away from downtown. My intuition is that in downtown, riders would not have choice of not taking the subway, while in less dense areas, people may use more cars when it is raining - but I may be wrong. To check if a station is in downtown or not, I would use latitude and longitude along with some criteria to define which are downtown stations.