Analyzing the NYC Subway Dataset

Intro to Data Science: Final Project 1, Part 2

(Short Questions)

Austin J. Alexander

Section 1. Statistical Test (IntroDS-ProjectOne-Section1.ipynb)

1.1

1.1.a Which statistical test did you use to analyse the NYC subway data? (IntroDS-ProjectOne-Section1.ipynb#1_1_a)

The Mann-Whitney U test was used to determine if there was a statistically significant difference between the number of reported entries on rainy and non-rainy occasions. This nonparametric test of the equality of two population medians from independent samples was used since the distribution of entries is non-normal (right-skewed) and their shape is the same, as seen visually via histograms, probability plots, and box plots, and as the result of the Shapiro-Wilk normality test (see Preparation for Statistical Tests (IntroDS-ProjectOne-DataExploration-Supplement.ipynb#prep-for-stats)). However, since the sample sizes are so large, the parametric Welch's t-test likely could have been used (and, it was implemented for confirmation purposes, along with the nonparametric Wilcoxon signed-rank test; both agreed with the Mann-Whitney U test results).

As witnessed later in the project, when rainy and non-rainy days from the data set are considered populations (as opposed to samples themselves), it takes significantly large sample sizes from each population (e.g., $n = 3000$, which is more than 30% of the total number of rainy days in the data set) to attain low p-values[1] frequently enough to reject the null hypothesis of the Mann-Whitney U test[2] with the critical values proposed below.

Moreover, using Wendt's rank-biserial correlation r and Cohen's d to measure effect size, the relatively low average value of r[3] and the low average value of d[4] both suggest that the difference between the two samples (and, thus, the two populations) is trivial, even though, according to the Mann-Whitney U test, the difference appears to be statistically signficant (and only then with extremely large samples)[5]. In other words, statistical significance $\neq$ practical significance.

Notes

[1] Identical samples would produce a large p (e.g., $p \approx 0.49$); extremely different samples would produce a very small number (e.g., $p \approx 0$).

[2] Identical samples would produce $U = \frac{n^2}{2}$ (e.g., when $n = 450$, $U = 101250$); extremely different samples can produce a U that is orders of magnitude smaller (e.g., when $n = 450$, possibly $U = 1293$).

[3] For very different samples, $r \to 1$; in the above tests, as n increases, $r \to 0$.

[4] For very different samples, $d \to 1$; in the above tests, as n increases, d tends to remain constant, $d \approx 0.06$, even when the sample size is extremely large. d is interpreted as the difference in the number of standard deviations.

[5] On the issue of p-values and large data sets, see Lin, M., Lucas, H.C., and Shmueli, G. Research Commentary—Too big to fail: Large samples and the P-value problem. *Inf. Syst. Res.* 2013; 24: 906–917. PDF here (http://www.galitshmueli.com/system/files/Print%20Version.pdf).

1.1.b Did you use a one-tail or a two-tail P value? (IntroDS-ProjectOne-Section1.ipynb#1_1_b)

A two-tail p-value was selected since an appropriate initial question, given the results of the Weather-Related Data (IntroDS-ProjectOne-DataExploration-Supplement.ipynb#weather-related) section of the *DataExploration* supplement, is simply whether or not there is a statistically significant difference between the populations (i.e., not whether one population is statistically-significantly greater than another).

1.1.c What is the null hypothesis? (IntroDS-ProjectOne-Section1.ipynb#1_1_c)

The Mann-Whitney U test is a nonparametric test of the null hypothesis that the distributions of two populations are the same.

To verify the assumption that the simple, randomly sampled values are independent, the sample sizes should be less than 5% of the population sizes ($n < 0.05N$). Since the maximum number of rainy days is 9585 ($N = 9585$), a reasonable sample size for each group would be 450 ($n = 450$).

Null Hyptohesis
$H_0 : M_1 = M_2$ or $H_0 : M_1 - M_2 = 0$

Alternate Hypothesis
$H_1 : M_1 \neq M_2$ or $H_1 : M_1 - M_2 \neq 0$

A 95% level of confidence would suggest that 95% of samples would produce similar statistical results.

For a 95% level of confidence, the level of significance (i.e., the probability of making a Type I error) $\alpha = (1 - 0.05) \cdot 100\% = 0.05$.

1.1.d What is your p-critical value? (IntroDS-ProjectOne-Section1.ipynb#1_1_d)

$p \leq 0.05$

1.2

1.2 Why is this statistical test applicable to the dataset? (see 1.1.a (IntroDS-ProjectOne-Section1.ipynb#1_1_a))

1.3

1.3 What results did you get from this statistical test? (IntroDS-ProjectOne-Section1.ipynb#1_3)

[N.B. The following values will change each time this notebook is run; however, the final results should not differ.]

$p = 0.34$

Sample 1 (rainy days)
$\bar{x}_1 = 1805.17, M_1 = 922.5$

Sample 2 (non-rainy days)
$\bar{x}_2 = 1733.25, M_2 = 921.0$

1.4

1.4 What is the significance and interpretation of these results? (IntroDS-ProjectOne-Section1.ipynb#1_4)

Statistical-Test Summary and Conclusion

The difference between the number of entries on rainy and non-rainy days ($n_1 = n_2 = 450$) from the data set is not statistically significant based on a two-independent-sample Mann-Whitney U test using scipy.stats.mannwhitneyu (U is extremely high [closer to its max value, $U = 101250$, than its min value] and p is signicantly greater than the proposed critical value 0.05). In addition, Wendt's rank-biserial correlation r and Cohen's d both indicate an essentially non-existent effect size.

Thus, there does not appear to be either a statistical or practical difference between rainy days and non-rainy days.

Section 2. Linear Regression (IntroDS-ProjectOne-Section2.ipynb)

2.1

2.1.a What approach did you use to compute the coefficients theta and produce prediction for ENTRIESn_hourly in your regression model? (IntroDS-ProjectOne-Section2.ipynb#2_1)

After comparing a few different methods (Ordinary Least Squares [OLS] from StatsModels, two different regression techniques from scikit-learn, the Broyden–Fletcher–Goldfarb–Shanno [BFGS] optimization algorithm from Scipy.optimize, and a Normal Equations algebraic attempt), OLS from StatsModels was chosen due to its consistently higher r and R2 values (see notes 1 and 2 below).

Notes

[1] The linear correlation coefficient (r) can take on the following values: $-1 \leq r \leq 1$. If $r = +1$, then a perfect positive linear relation exists between the explanatory and response variables. If $r = -1$, then a perfect negative linear relation exists between the explanatory and response variables.

[2] The coefficient of determination (R^2) can take on the following values: $0 \leq R^2 \leq 1$. If $R^2 = 0$, the least-squares regression line has no explanatory value; if $R^2 = 1$, the least-squares regression line explains 100% of the variation in the response variable.

2.2

2.2 What features (input variables) did you use in your model? Did you use any dummy variables as part of your features? (IntroDS-ProjectOne-Section2.ipynb#2_2)

Quantitative features used: 'hour','day_week','rain','tempi'.

Categorical features used: 'UNIT'. As a categorical feature, this variable required the use of so-called dummy variables.

2.3

2.3 Why did you select these features in your model? (IntroDS-ProjectOne-Section2.ipynb#2_3)

Due to the findings presented in the *DataExploration* supplement (IntroDS-ProjectOne-DataExploration-Supplement.ipynb), it seemed clear that location significantly impacted the number of entries. In addition, the hour and day of the week showed importance. Temperature appeared to have some relationship with entries as well, and so it was included. Based on that exploration and on the statistical and practical evidence offered in Section 1. Statistical Test (IntroDS-ProjectOne-Section1.ipynb), rain was not included as a feature (and, as evidenced by a number of test runs, had marginal if any importance).

As far as the selection of location and day/time variables were concerned, **station** can be captured quantitatively by **latitude** and **longitude**, both of which, as numeric values, should offer a better sense of trend toward something. However, as witnessed by numerous test runs, **latitude** and **longitude** in fact appear to be redundant when using **UNIT** as a feature, which is in fact more signficant (as test runs indicated and, as one might assume, due to, for example, station layouts, where some UNITs would be used more than others) than **latitude** and **longitude**.

Each **DATEn** is a 'one-off', so it's unclear how any could be helpful for modeling/predicting (as those dates literally never occur again). **day_week** seemed to be a better selection in this case.

2.4

2.4 What are the coefficients (or weights) of the features in your linear regression model? (IntroDS-ProjectOne-Section2.ipynb#2_4)

[-2.98186953e+14 -1.00885474e+01 -1.23230075e+02 -3.62768713e+14 3.47461736e+13
8.55077133e+13 3.95457990e+13 4.93215540e+13 3.27919219e+12 1.32501142e+15 3.63158848e+14
3.90595392e+13 3.90595392e+13 5.25195291e+14 5.25195291e+14 -3.33646892e+13
4.18593651e+13 -6.43005338e+13 3.48553789e+13 4.71909885e+13 4.71909885e+13
8.90991538e+13 7.67721779e+13 2.98751459e+14 2.67953865e+14 1.43331265e+15 3.90595392e+13
7.10254335e+13 7.10254335e+13 -3.41911030e+13 1.64259366e+13 -2.13851245e+11
2.16027007e+13 9.53346080e+13 -1.60881548e+12 5.49131071e+13 -1.55489589e+15
-1.55489589e+15 2.98751459e+14 1.43331265e+15 2.47998964e+13 2.89338185e+14
3.22923077e+13 3.22923077e+13 4.84345209e+14 3.42500167e+13 6.72581945e+13
-9.93822030e+12 5.03782077e+13 7.76563527e+13 -7.80684225e+12 3.07801248e+13
2.81807568e+13 4.65753724e+13 2.74201203e+13 2.98820259e+13 4.35639251e+13 2.45891578e+15
3.35174852e+13 1.59174038e+14 5.99679225e+13 1.52604489e+13 1.91940058e+14 5.19163570e+13
5.14500887e+13 4.85578758e+13 2.00218798e+13 4.66204123e+13 5.61717207e+13 5.51981189e+13
4.30537071e+13 3.00841483e+13 3.72836651e+13 4.74659693e+13 5.99136725e+13 4.99736315e+13
1.09303050e+13 5.44967598e+13 2.18176947e+13 8.90969031e+12 4.59065947e+13 4.52442216e+13
5.24200053e+13 5.65429622e+13 -3.41911030e+13 1.42460159e+14 7.28922708e+13
3.83122773e+13 1.97757837e+12 1.98524271e+13 2.81523274e+13 6.62408879e+13 3.21655122e+13
3.97254196e+13 2.77119543e+13 8.03721437e+12 3.12730768e+13 3.21655122e+13 5.36516191e+13
2.09635095e+13 2.39586283e+13 5.08100848e+13 3.83511683e+13 3.34582891e+13 3.00841483e+13
1.23500393e+13 2.34435550e+13 3.22561174e+13 3.31012083e+13 3.62244478e+13 4.40242155e+13
4.45055489e+13 2.41979202e+13 3.62244478e+13 8.03721437e+12 4.82501474e+13 3.76235285e+13
7.66106928e+12 4.65753724e+13 4.28865143e+13 1.93628802e+13 2.43644973e+13 4.63626940e+13
4.59065947e+13 3.87850621e+13 3.57217336e+13 4.25004225e+13 6.11943119e+13
-6.94038898e+12 5.71840894e+13 4.78872781e+13 4.40387102e+13 4.08021150e+13
3.62244478e+13 2.41123573e+13 2.73676108e+14 1.19602426e+13 4.14213856e+13
-1.12989949e+13 7.28922708e+13 4.00816485e+13 2.63018286e+13 7.13348492e+13
5.16031160e+13 6.37941980e+13 3.87104156e+13 2.84917040e+13 3.82289919e+13 2.31588376e+13
5.18451435e+13 4.37360490e+13 8.03721436e+12 3.88662660e+13 4.52442216e+13 1.67510482e+13
6.43235553e+12 5.81147619e+13 3.71224800e+13 4.22952594e+13 4.61959442e+13 5.38187476e+13
5.97888006e+13 3.29972520e+13 5.08230117e+13 2.22196950e+13 3.71224800e+13 4.28642697e+13
4.43250535e+13 4.67239258e+13 3.70795448e+13 3.87072959e+13 4.91165088e+12 2.96441906e+13
6.86378380e+13 2.67851651e+12 5.26414878e+13 2.13264925e+13 4.20047702e+13 1.93753589e+13
5.40247861e+13 4.53699055e+13 -1.50947870e+13 3.71421059e+13 4.73192497e+13
4.94518575e+13 5.65429622e+13 2.15405550e+13 4.16380420e+13 4.51578803e+13 1.23967503e+13
5.58618922e+13 3.01110409e+13 2.06820387e+13 4.13469209e+13 3.65737165e+13 1.74086992e+13
2.32845532e+13 1.77776841e+13 3.26637334e+13 4.43523883e+13 2.57833231e+13 3.50506876e+13
4.25765737e+13 6.37941980e+13 2.73704745e+13 5.77956984e+13 3.65343986e+13 3.27919219e+12
3.87934636e+13 3.00549795e+13 4.17272926e+13 2.90234750e+13 3.22923077e+13 2.99505548e+13

3.65737165e+13 9.15144491e+12 4.53217822e+13 4.21900318e+13 2.56098375e+13 7.54183968e+13
5.01742552e+13 4.89785158e+13 2.86985182e+13 5.65429622e+13 5.67400981e+13 3.41216277e+13
7.76563527e+13 6.04684510e+13 6.80068990e+13 3.42698546e+13 2.81398035e+13 3.27799782e+13
4.22952594e+13 2.02399690e+13 4.29506064e+13 4.48932891e+13 1.51172395e+13 4.63626940e+13
5.97888006e+13 5.58618922e+13 7.13348492e+13 -8.31513914e+12 4.18147672e+13
1.45131124e+14 1.45131124e+14 1.45131124e+14 1.45131124e+14 1.45131124e+14 1.45131124e+14
7.99101817e+13 7.99101817e+13 7.99101817e+13 7.99101817e+13 7.99101817e+13 7.99101817e+13
7.99101817e+13 2.23226354e+13 3.06452246e+13 2.38240931e+13 2.91214316e+13 6.39942022e+13
6.98664549e+13 -1.25186577e+15 1.07336750e+14 3.74239135e+13 4.57751726e+13
3.48333698e+13 1.66026849e+13 1.64055490e+13 3.88956304e+13 3.43605850e+13 2.53376327e+11
5.67197105e+13 4.54336928e+13 2.69497029e+13 2.64217213e+13 3.96873581e+13 3.66112485e+13
3.08503877e+13 4.31950923e+13 2.05041593e+13 5.63945989e+13 3.15076051e+13 5.24636084e+13
5.16050921e+13 3.80949596e+13 5.29056781e+13 5.15429464e+13 3.05690734e+13
-2.21889609e+13 6.90475919e+12 5.31237673e+13 2.67829531e+13 1.82325400e+13
4.18725703e+13 1.72837549e+13 2.45877713e+13 3.09556153e+13 4.04819137e+13 2.59546586e+13
3.40861079e+13 5.18191546e+13 3.82902682e+13 4.83457507e+13 -2.90013201e+14
2.12021364e+12 2.13005036e+13 3.30639986e+13 1.37446181e+14 2.16955584e+13 5.88745266e+12
6.51084327e+13 -2.16192537e+14 2.94095982e+13 4.01483951e+13 2.12292901e+13
2.52583690e+13 2.65252349e+13 4.08895297e+13 4.87811498e+13 3.17242615e+13 4.50058437e+13
3.17987262e+13 4.44471289e+13 3.23435321e+13 4.03656689e+13 1.69739264e+13 5.13279524e+13
3.69211994e+13 4.50780911e+12 4.91870188e+13 3.02813774e+13 3.13308799e+13
-4.51070555e+12 5.37702882e+13 3.44352315e+13 4.46539431e+13 2.82523580e+13
2.36937896e+13 2.91069369e+13 4.90332898e+13 3.88757925e+13 2.41671313e+13 2.29713920e+13
-2.27274966e+12 4.75358096e+13 3.42793811e+13 2.86400982e+13 3.60035413e+13
3.58619820e+13 8.09524894e+13 3.11408769e+13 8.00860361e+13 3.55221186e+13 4.49933198e+13
3.43521836e+13 1.62804154e+15 4.00444388e+13 1.32319746e+13 -6.93145123e+13
1.93268995e+13 3.14183545e+13 4.97020921e+13 8.44446420e+13 7.11680687e+13
-1.36016700e+15 2.48954997e+13 2.72390524e+13 4.68438185e+13 5.53679630e+13
5.57369479e+13 4.89477269e+13 3.90240195e+13 4.57255268e+13 4.35914360e+14 2.79014255e+13
1.86488873e+13 7.33594984e+13 4.35014565e+13 4.49648903e+13 3.83994736e+13 6.11854045e+13
5.37827669e+13 4.98610939e+13 4.09801349e+13 -1.23620662e+13 2.23355623e+13
4.41221721e+13 5.09259521e+13 4.23655223e+13 3.60661023e+13 3.60231671e+13 2.31720156e+13
3.01950407e+13 1.50308852e+13 2.15425312e+13 8.82404341e+13 2.88205936e+13 1.81079792e+12
6.22153421e+13 -2.00530461e+14 -3.62653075e+12 3.12862820e+13 4.30614988e+13
4.30346062e+13 6.42359568e+13 3.96281619e+13 1.59615577e+13 3.47944788e+13 3.02591328e+13
5.13874815e+12 4.08533394e+13 -4.52049644e+14 8.30838674e+13 7.47544626e+13
2.07256418e+13 2.77757416e+13 2.87932588e+13 1.94940280e+13 1.53499487e+13 3.00919401e+13
6.07956078e+13 6.82339962e+13 3.49166552e+13 2.27674394e+13 5.78851982e+13 2.65702747e+13
1.91208610e+13 8.14607863e+13 1.26771961e+13 3.34202275e+13 5.80284076e+13
-8.60283904e+13 6.07488968e+13 1.79475282e+13 1.33568465e+13 6.54845778e+13
5.21821377e+13 9.35144915e+12 -2.38577014e+15 1.06510336e+14 7.04671306e+13
2.95817220e+13 2.58263974e+13 3.65719306e+13 -1.18794411e+14 6.67132916e+13
1.19513352e+13 -1.59535067e+13 4.32636212e+13 4.99868095e+13 1.31777246e+13

3.44383512e+13 2.79877668e+13 4.30906676e+13 -2.25605812e+14 2.56796778e+13
5.32932200e+13 2.78238649e+13 4.73623240e+13 3.35998481e+13 -4.11199562e+14

2.5

2.5 What is your model's R^2 (coefficients of determination) value? (IntroDS-ProjectOne-Section2.ipynb#2_5)

The best R^2 value witnessed was 0.54 (with the best r value seen at 0.74).

2.6

2.6.a What does this R^2 value mean for the goodness of fit for your regression model? (IntroDS-ProjectOne-Section2.ipynb#2_6_a)

This R^2 value means that 54% of the proportion of total variation in the response variable is explained by the least-squares regression line (i.e., model) that was created above.

2.6.b Do you think this linear model to predict ridership is appropriate for this dataset, given this R^2 value? (IntroDS-ProjectOne-Section2.ipynb#2_6_b)

It's barely better than guessing in the dark; thus, too much shouldn't be staked on its predictions.

Since the predictions show a discernible, linear, and increasing pattern (and, thus, are not stochastic), it seems apparent that there is in fact not a linear relationship between the explanatory and response variables. Thus, a linear model is not appropriate for the current data set.

Section 3. Visualization

3.1

3.1 One visualization should contain two histograms: one of ENTRIESn_hourly for rainy days and one of ENTRIESn_hourly for non-rainy days (see the ENTRIESn_hourly HISTOGRAM (by RAIN) (IntroDS-ProjectOne-DataExploration-Supplement.ipynb#entries-hist) in the *DataExploration* supplement)

3.2

3.2 One visualization can be more freeform (e.g., see the Combining Station, Latitude, Longitude Data on a Map Layer (IntroDS-ProjectOne-DataExploration-Supplement.ipynb#station-lat-long-map) in the *DataExploration* supplement)

Section 4. Conclusion

4.1

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?

From the current data set and the analyses performed, it remains inconclusive whether rain has any impact on the number of NYC subway entries. However, based on this data set alone, rain seemed to be an insignificant factor as it related to subway ridership. Thus, further analysis is necessary.

On the other hand, based on the data exploration, it seems quite clear that the number of entries is highly dependent on physical location, particularly station position, with specific units having the most importance.

4.2

4.2 What analyses lead you to this conclusion?

Based on exploration, statistical significance tests, and attempts at modeling the data, physical location dominated as an explanatory variable with respect to the number of entries. The Mann-Whitney U test indicated that rainy and non-rainy days were essentially identical. Thus, there did not appear to be either a statistical or, based on the effect size values, practical difference between rainy days and non-rainy days in terms of their respective number of entries.

Section 5. Reflection

5.1

5.1 Please discuss potential shortcomings of the methods of your analysis

The data set under consideration was limited to a single month in the late spring / early summer of a particular year. As a result, among countless other possible factors for which the available data did not account, precipitation may in fact have an increased impact on the number of entries at other times of the year (e.g., during the winter months). Thus, the data set was limited by its temporal locale.

The linear regression model that was created, while having very high r and R^2 values did not, based on residual analysis, adequately model the data. There is in fact *not* a linear relationship between the explanatory and response variables under consideration; thus, a non-linear model would likely be more appropriate for the current data set.

The statistical tests that were employed seemed effective (as long as sample sizes were kept small enough). However, it's unclear how traditional statistical tests relate to massive data sets (esp. since many statitical tests need to be used with relatively small sample sizes; on this point, see 5.2 below).

5.2

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

Assuming all statistical tests and learning models were implemented and interpreted correctly, it became clear that computational power was very important in data science, not due to the ability merely to apply methods to data, but in the ability to repeat numerous tests on random samples of data, which, at least in the case of this analysis, encouraged more confidence in test/model results.

DataExploration Supplement (IntroDS-ProjectOne-DataExploration-Supplement.ipynb)

Rain Supplement (IntroDS-ProjectOne-Rain-Supplement.ipynb)

Unit_Entries Supplement (IntroDS-ProjectOne-Unit_Entries-Supplement.ipynb)

While an inumerable number of online resources were used to attain a better understanding of the statistical matters in this analysis, the primary source for statistical definitions and methods was Michael Sullvan's *Statistics: Informed decisions using data* (4th ed.) (http://www.amazon.com/Statistics-Informed-Decisions-Using-Edition/dp/0321757270).

Analyzing the NYC Subway Dataset

Intro to Data Science: Final Project 1, Part 2

(Short Questions)

Data Exploration Supplement

Austin J. Alexander

Import Directives and Initial DataFrame Creation

In [40]:
```
import inflect # for string manipulation
import numpy as np
import pandas as pd
import scipy as sp
import scipy.stats as st
import matplotlib.pyplot as plt
%matplotlib inline

filename = '/Users/excalibur/py/nanodegree/intro_ds/final_project/i
mproved-dataset/turnstile_weather_v2.csv'

# import data
data = pd.read_csv(filename)
```

Initial Data Exploration

Data Shape

In [41]:
```
print "SHAPE: " + str(data.shape)
data.head(1)
```

```
SHAPE: (42649, 27)
```

Out[41]:

	UNIT	DATEn	TIMEn	ENTRIESn	EXITSn	ENTRIESn_hourly	EXITSn_hourly
0	R003	05-01-11	00:00:00	4388333	2911002	0	0

1 rows × 27 columns

Data Types

In [42]:
```
data.dtypes
```

Out[42]:
```
UNIT                object
DATEn               object
TIMEn               object
ENTRIESn             int64
EXITSn               int64
ENTRIESn_hourly    float64
EXITSn_hourly      float64
datetime            object
hour                 int64
day_week             int64
weekday              int64
station             object
latitude           float64
longitude          float64
conds               object
fog                  int64
precipi            float64
pressurei          float64
rain                 int64
tempi              float64
wspdi              float64
meanprecipi        float64
meanpressurei      float64
meantempi          float64
meanwspdi          float64
weather_lat        float64
weather_lon        float64
dtype: object
```

ENTRIESn_hourly Statistics

In [43]:
```
data['ENTRIESn_hourly'].describe()
```

Out[43]:
```
count    42649.000000
mean      1886.589955
std       2952.385585
min          0.000000
25%        274.000000
50%        905.000000
75%       2255.000000
max      32814.000000
Name: ENTRIESn_hourly, dtype: float64
```

Functions for Getting, Mapping, and Plotting Data

[N.B. Due to decisions described in the *Unit_Entries* supplement (IntroDS-ProjectOne-Unit_Entries-Supplement.ipynb), in the current analysis, unless otherwise noted, *entries* will refer to a summation of **ENTRIESn_hourly** per **UNIT** (i.e., not, as might be expected, simply alues in the **ENTRIESn** column).]

In [44]:
```
entries_hourly_by_row = data['ENTRIESn_hourly'].values
```

In [45]:
```
def map_column_to_entries_hourly(column):
    instances = column.values # e.g., longitude_instances = data['l
ongitude'].values

    # reduce
    entries_hourly = {} # e.g., longitude_entries_hourly = {}
    for i in np.arange(len(instances)):
        if instances[i] in entries_hourly:
            entries_hourly[instances[i]] += float(entries_hourly_b
y_row[i])
        else:
            entries_hourly[instances[i]] = float(entries_hourly_b
y_row[i])

    return entries_hourly # e.g., longitudes, entries
```

In [46]:
```
def display_basic_stats(entries_hourly_dict, column1name):
    # e.g, longitude_df = pd.DataFrame(data=longitude_entries_hourl
y.items(), columns=['longitude','entries'])
    df = pd.DataFrame(data=entries_hourly_dict.items(), columns=[co
lumn1name,'entries'])

    p = inflect.engine()
    print "{0} AND THEIR ENTRIES".format(p.plural(column1name.uppe
r()))
    print df.head(3)

    print
    print pd.DataFrame(df['entries']).describe()
    print "{:<7}".format('range') + "{:0<14}".format(str(np.ptp(ent
ries_hourly_dict.values())))

    return df # e.g, longitude_df
```

In [47]:
```
def plot_data(df, column1name, plot_kind, xaxis_labeled):

    p = inflect.engine()
    if xaxis_labeled == True:
        df.plot(x=column1name, y='entries', title="{0} AND THEIR EN
TRIES".format(p.plural(column1name.upper())), kind=plot_kind, alph
a=0.5, color='green')
        plt.xlabel(column1name)
    else:
        df.plot(title="{0} AND THEIR ENTRIES".format(p.plural(colum
n1name.upper())), kind=plot_kind, alpha=0.5, color='green')
        plt.xlabel("{0} row index".format(column1name))

    plt.ylabel('{0} entries'.format(column1name))
    plt.legend(['entries'])
    plt.show()
```

In [48]:
```
def plot_histogram(df, column_name, num_of_bins):
    df[column_name].plot(kind='hist', bins=num_of_bins, alpha=0.5,
color='green')
    plt.ylabel('frequency')
    plt.show()
```

Function for Basic Statistics

In [49]:

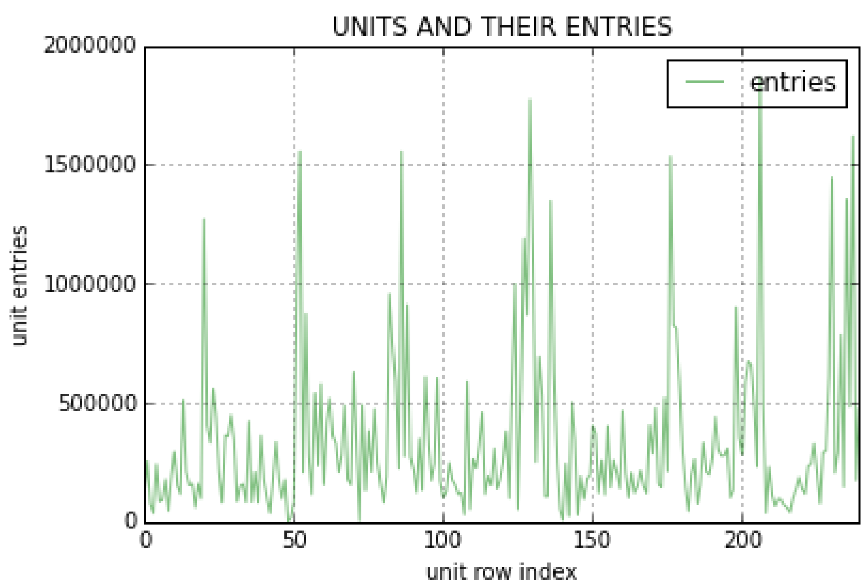
```
def describe_samples(sample1, sample2):
    size1, min_max1, mean1, var1, skew1, kurt1 = st.describe(sample
1)
    size2, min_max2, mean2, var2, skew2, kurt2 = st.describe(sample
2)

    med1 = np.median(sample1)
    med2 = np.median(sample2)

    std1 = np.std(sample1)
    std2 = np.std(sample2)

    print "Sample 1 (rainy days):\n  min = {0}, max = {1},\n  mean
= {2:.2f}, median = {3}, var = {4:.2f}, std = {5:.2f}".format(min_m
ax1[0], min_max1[1], mean1, med1, var1, std1)
    print "Sample 2 (non-rainy days):\n  min = {0}, max = {1},\n  m
ean = {2:.2f}, median = {3}, var = {4:.2f}, std = {5:.2f}".format(m
in_max2[0], min_max2[1], mean2, med2, var2, std2)
```

Non-Weather-Related Data

Unit Statistics

In [50]:
```
unit_entries_hourly = map_column_to_entries_hourly(data['UNIT'])
unit_df = display_basic_stats(unit_entries_hourly, 'unit')
plot_data(unit_df, 'unit', 'line', False)
```

```
UNITS AND THEIR ENTRIES
   unit  entries
0  R318   112098
1  R319   254531
2  R312    73913

              entries
count      240.000000
mean    335254.895833
std     334849.388932
min          0.000000
25%     131148.000000
50%     221479.500000
75%     409285.750000
max    1868674.000000
range  1868674.000000
```



Top-5 Units

In [51]:
```
unit_df.sort(columns='entries', ascending=False).head(5)
```

Out[51]:

	unit	entries
206	R084	1868674
129	R022	1773372
237	R012	1618262
52	R046	1555117
86	R055	1554806

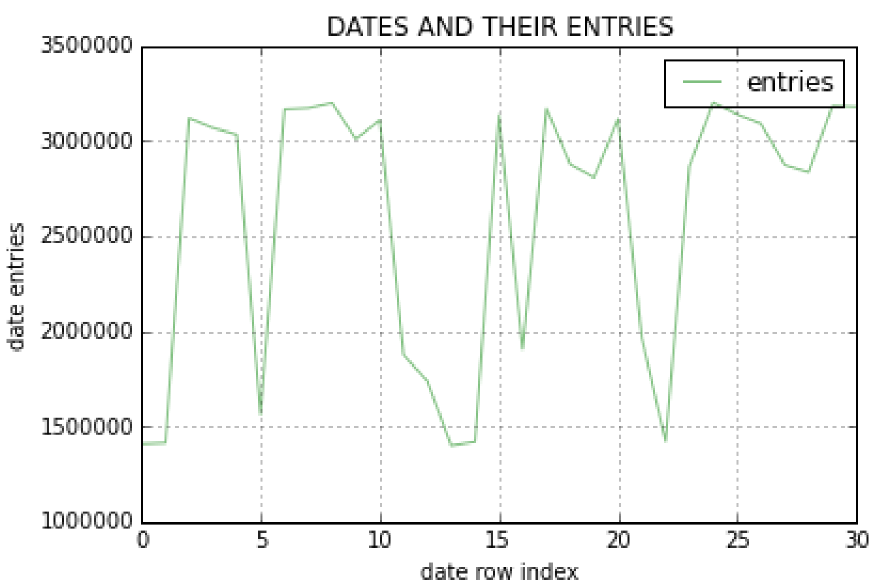
Unit Summary

Clearly, certain units received more entries than other units.

Date Statistics

In [52]:
```
date_entries_hourly = map_column_to_entries_hourly(data['DATEn'])
date_df = display_basic_stats(date_entries_hourly, 'date')
plot_data(date_df, 'date', 'line', False)
```

```
DATES AND THEIR ENTRIES
       date  entries
0  05-30-11  1409572
1  05-15-11  1413929
2  05-04-11  3118915

              entries
count       31.000000
mean   2595521.774194
std     710440.834289
min    1400098.000000
25%    1891834.000000
50%    3009536.000000
75%    3137683.000000
max    3201840.000000
range  1801742.000000
```



May 2011 Calendar

Image from: calendarbar.org

Top-5 Dates

In [53]:
```
date_df.sort(columns='entries', ascending=False).head(5)
```

Out[53]:

	date	entries
24	05-12-11	3201840
8	05-05-11	3199002
29	05-03-11	3183128
30	05-06-11	3179032
7	05-26-11	3172004

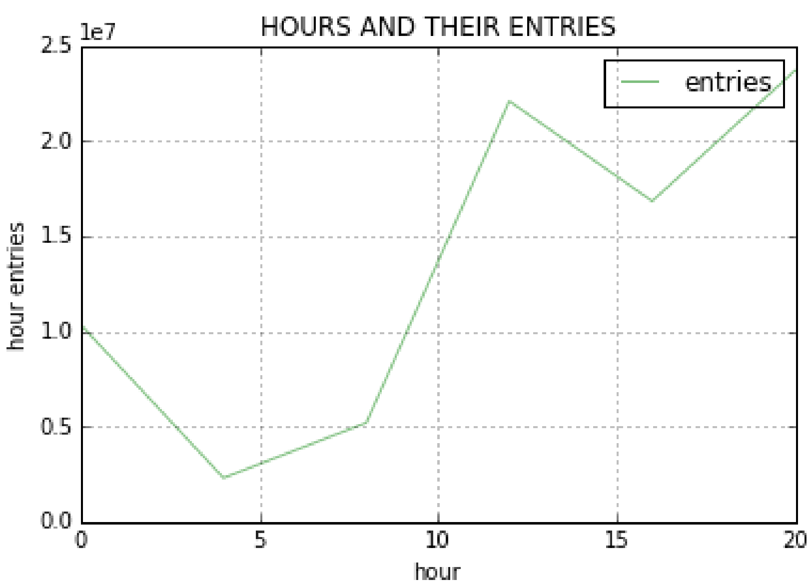
Date Summary

Clearly, certain dates received more entries than other dates (the top 5 included 1 Tuesday, 3 Thursdays , and 1 Friday).

Hour Statistics

In [54]:
```
hour_entries_hourly = map_column_to_entries_hourly(data['hour'])
hour_df = display_basic_stats(hour_entries_hourly, 'hour')
plot_data(hour_df, 'hour', 'line', True)
```

```
HOURS AND THEIR ENTRIES
   hour   entries
0     0  10353167
1     4   2300788
2     8   5198583

               entries
count         6.000000
mean   13410195.833333
std     8863957.086415
min     2300788.000000
25%     6487229.000000
50%    13593103.500000
75%    20772247.000000
max    23690281.000000
range  21389493.00000
```



Top-5 Hours

In [55]:
```
hour_df.sort(columns='entries', ascending=False).head(5)
```

Out[55]:

	hour	entries
5	20	23690281
3	12	22085316
4	16	16833040
0	0	10353167
2	8	5198583

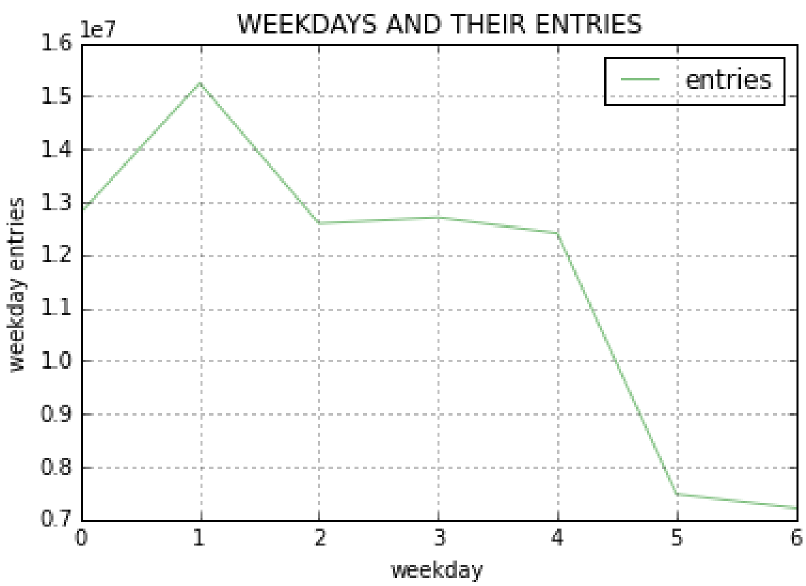
Hour Summary

Clearly, certain hours recorded more entries than other hours (the top 2 were, in order, 10:00pm and 12:00pm).

Weekday Statistics

In [56]:
```
weekday_entries_hourly = map_column_to_entries_hourly(data['day_wee
k'])
weekday_df = display_basic_stats(weekday_entries_hourly, 'weekday')
plot_data(weekday_df, 'weekday', 'line', True)
```

```
WEEKDAYS AND THEIR ENTRIES
   weekday   entries
0        0  12795107
1        1  15246943
2        2  12592691

                entries
count          7.000000
mean    11494453.571429
std      2989933.638739
min      7218706.000000
25%      9949293.000000
50%     12592691.000000
75%     12752124.500000
max     15246943.000000
range   8028237.000000
```



Top-5 Weekdays

(0: Mon, 1: Tue, 2: Wed, 3: Thu, 4: Fri, 5: Sat, 6: Sun)

In [57]:
```
weekday_df.sort(columns='entries', ascending=False).head(5)
```

Out[57]:

	weekday	entries
1	1	15246943
0	0	12795107
3	3	12709142
2	2	12592691
4	4	12411679

Weekday Summary

Clearly, certain weekdays received more entries than other weekdays (the top 2 were, in order, Tuesday and Monday -- somewhat strange given the Date results above).

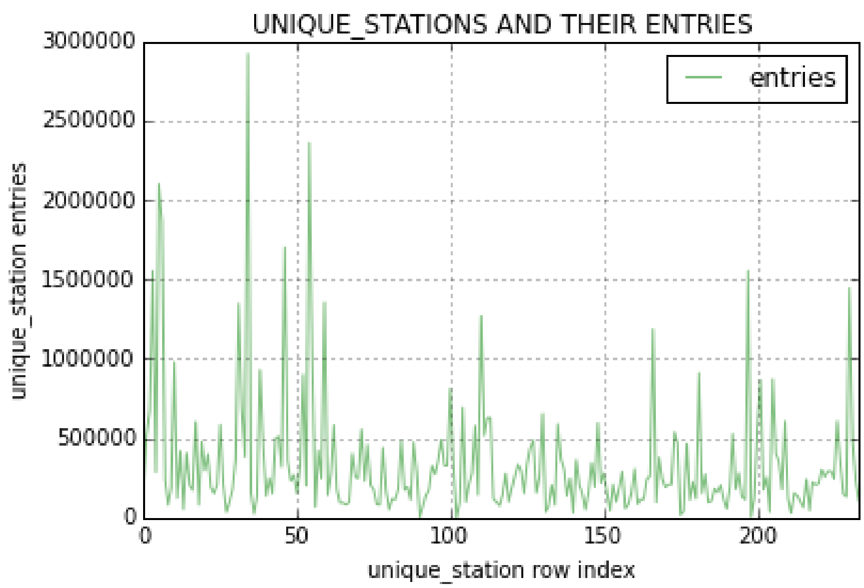
Station Statistics

[N.B. Some stations have the same name but different locations, so unique identifiers needed to be created.]

In [58]:
```
data['unique_station'] = data['station'] + " (" + data['latitud
e'].map(str) + ", " + data['longitude'].map(str) + ")"
```

In [59]:
```
station_entries_hourly = map_column_to_entries_hourly(data['unique_station'])
station_df = display_basic_stats(station_entries_hourly, 'unique_station')
plot_data(station_df, 'unique_station', 'line', False)
```

```
UNIQUE_STATIONS AND THEIR ENTRIES
                             unique_station  entries
0            176 ST (40.848635, -73.912497)   151399
1  168 ST-BROADWAY (40.840778, -73.940091)   521054
2      57 ST-7 AVE (40.764755, -73.980646)   674799

              entries
count      234.000000
mean    343851.175214
std     393424.158576
min          0.000000
25%     130422.000000
50%     217648.000000
75%     402551.250000
max    2920887.000000
range  2920887.000000
```



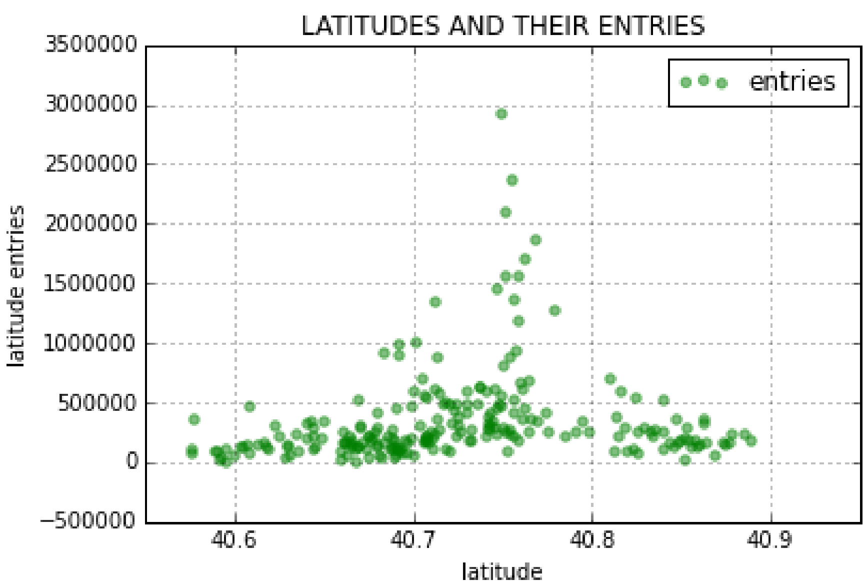
Station Summary

Clearly, certain stations received more entries than other stations.

Latitude Statistics

In [60]:
```
latitude_entries_hourly = map_column_to_entries_hourly(data['latitu
de'])
latitude_df = display_basic_stats(latitude_entries_hourly, 'latitud
e')
plot_data(latitude_df, 'latitude', 'scatter', True)
```

```
LATITUDES AND THEIR ENTRIES
    latitude  entries
0  40.852417     7559
1  40.707840   209745
2  40.643982   102508

                entries
count        233.000000
mean      345326.931330
std       393653.267874
min            0.000000
25%       131511.000000
50%       218938.000000
75%       402883.000000
max      2920887.000000
range    2920887.000000
```



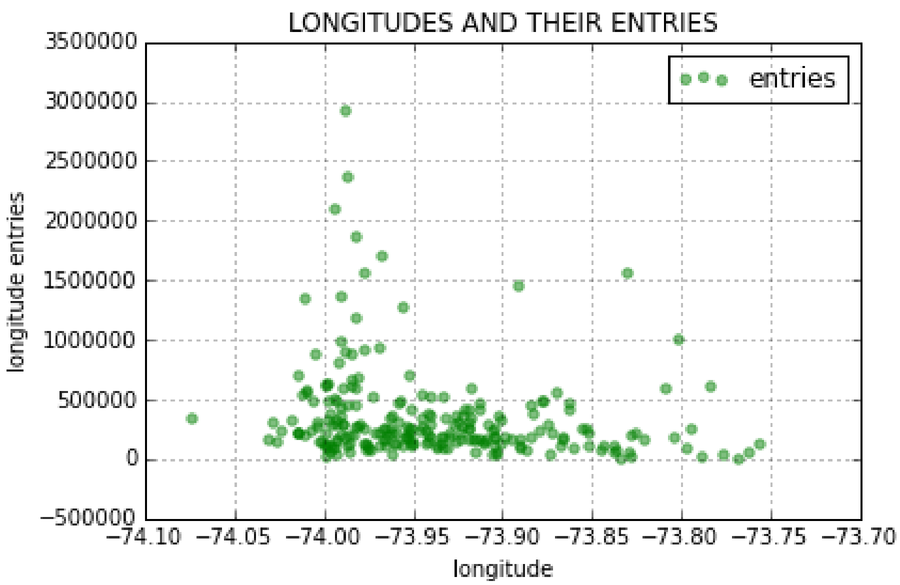
Latitude Summary

Clearly, certain latitudes received more entries than other latitudes (in particular, those between latitudes 40.7 and 40.8).

Longitude Statistics

In [61]:
```
longitude_entries_hourly = map_column_to_entries_hourly(data['longi
tude'])
longitude_df = display_basic_stats(longitude_entries_hourly, 'longi
tude')
plot_data(longitude_df, 'longitude', 'scatter', True)
```

```
LONGITUDES AND THEIR ENTRIES
   longitude  entries
0 -73.977417   911174
1 -73.828125   193792
2 -74.014099   694605

                entries
count        234.000000
mean      343851.175214
std       393424.158576
min            0.000000
25%       130422.000000
50%       217648.000000
75%       402551.250000
max      2920887.000000
range    2920887.000000
```



Longitude Summary

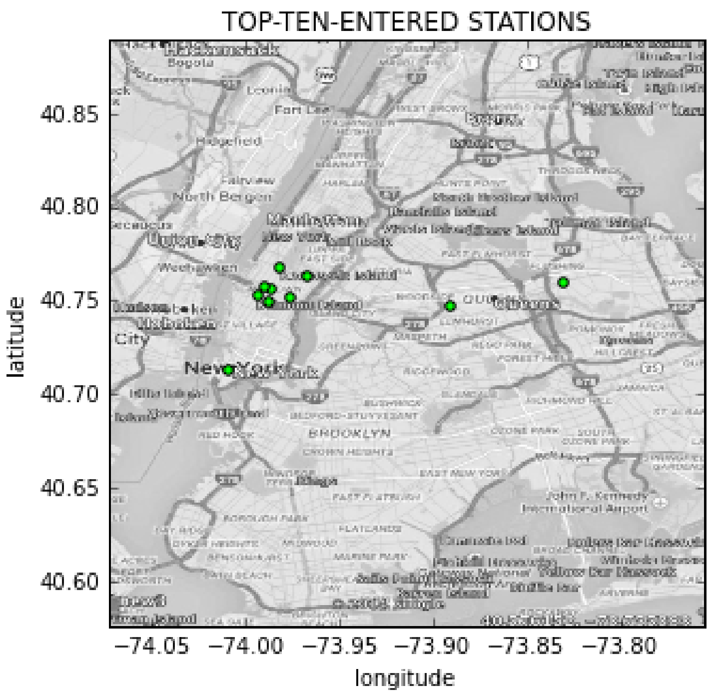
Clearly, certain longitudes received more entries than other longitudes (in particular, those between longitudes -74.00 and -73.95).

Combining Station, Latitude, Longitude Data on a Map Layer

The top-ten most-entered stations are the ones that someone familiar with New York City might expect (with a focus on Manhattan Island):

In [62]:
```
station_location = data[['unique_station', 'latitude', 'longitud
e']]
station_location.drop_duplicates(inplace=True)
station_location_entries = pd.merge(station_location, station_df, o
n='unique_station')
station_location_entries.sort(columns='entries', ascending=False, i
nplace=True)
top_ten = station_location_entries.head(10)
print top_ten[['unique_station', 'entries']]

plt.figure(figsize = (5,5))
plt.title('TOP-TEN-ENTERED STATIONS')
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.xlim(station_location_entries['longitude'].min(), station_locat
ion_entries['longitude'].max())
plt.ylim(station_location_entries['latitude'].min(), station_locati
on_entries['latitude'].max())
img = plt.imread('NYmap.png')

plt.scatter(top_ten['longitude'], top_ten['latitude'], color='#00FF
00', edgecolors='black', zorder=1)

plt.imshow(img, zorder=0, extent=[station_location_entries['longitu
de'].min(), station_location_entries['longitude'].max(), station_lo
cation_entries['latitude'].min(), station_location_entries['latitud
e'].max()])

plt.show()
```

	unique_station	entries
14	34 ST-HERALD SQ (40.749533, -73.987899)	2920887
21	42 ST-TIMES SQ (40.755905, -73.986504)	2360981
8	34 ST-PENN STA (40.752247, -73.993456)	2101634
58	59 ST-COLUMBUS (40.76811, -73.981891)	1868674
34	LEXINGTON AVE (40.762796, -73.967686)	1701440
32	42 ST-GRD CNTRL (40.751849, -73.976945)	1555117
38	MAIN ST (40.759578, -73.830056)	1554806
10	ROOSEVELT AVE (40.746655, -73.891361)	1444569
7	42 ST-PA BUS TE (40.757303, -73.989787)	1355492
18	WORLD TRADE CTR (40.712557, -74.009807)	1347727



Station, Latitude, and Longitude Summary

Again, it seems quite clear that number of entries is dependent on *location, location, location*!

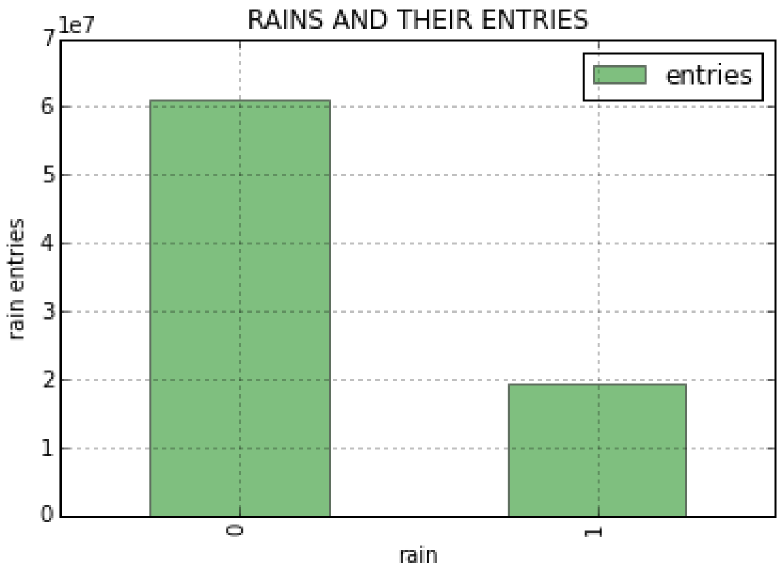
Weather-Related Data

While the above simple observations seem to indicate that non-weather-related variables have a tremendous impact on transit usage, it would only be fair to explore possible weather-related influences as well (esp. since the project guidelines expect it). Out of the weather-related data that has been made available, the two most-sensible categories to check are the (binary) rain and temperature (in Fahrenheit) variables.

Rain Statistics

In [63]:
```
rain_entries_hourly = map_column_to_entries_hourly(data['rain'])
rain_df = display_basic_stats(rain_entries_hourly, 'rain')
plot_data(rain_df, 'rain', 'bar', True)
```

```
RAINS AND THEIR ENTRIES
   rain   entries
0     0  61020916
1     1  19440259

                entries
count          2.000000
mean    40230587.500000
std     29401964.530892
min     19440259.000000
25%     29835423.250000
50%     40230587.500000
75%     50625751.750000
max     61020916.000000
range   41580657.00000
```



In [64]:
```
rain_days = data[data['rain'] == 1]
no_rain_days = data[data['rain'] == 0]

print "RAIN DAYS"
print rain_days['ENTRIESn_hourly'].describe()
print
print "NO-RAIN DAYS"
print no_rain_days['ENTRIESn_hourly'].describe()
```

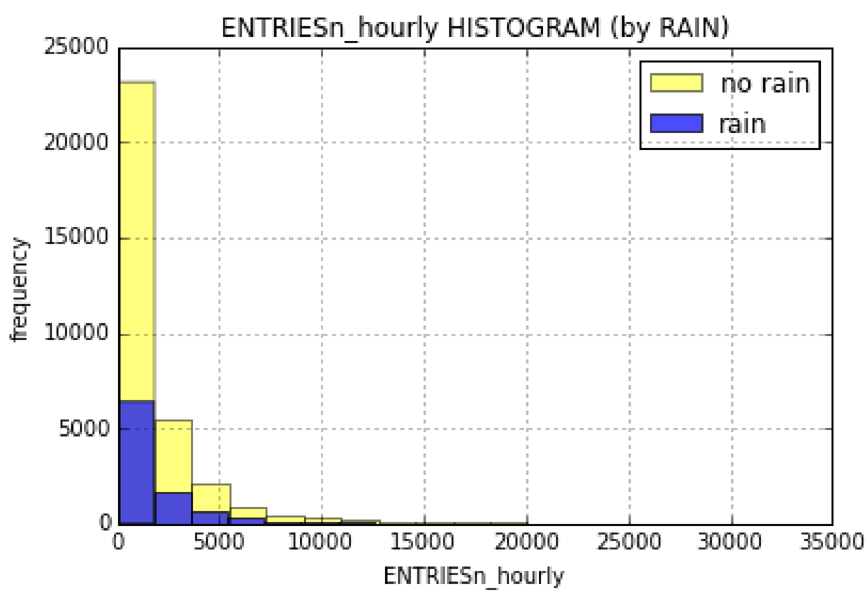
```
RAIN DAYS
count     9585.000000
mean      2028.196035
std       3189.433373
min          0.000000
25%        295.000000
50%        939.000000
75%       2424.000000
max      32289.000000
Name: ENTRIESn_hourly, dtype: float64

NO-RAIN DAYS
count    33064.000000
mean      1845.539439
std       2878.770848
min          0.000000
25%        269.000000
50%        893.000000
75%       2197.000000
max      32814.000000
Name: ENTRIESn_hourly, dtype: float64
```

ENTRIESn_hourly HISTOGRAM (by RAIN)

In [76]:

```
print "Number of non-rainy days:", no_rain_days['ENTRIESn_hourl
y'].count()
print "Number of rainy days:",  rain_days['ENTRIESn_hourly'].coun
t()

no_rain_days['ENTRIESn_hourly'].plot(kind='hist', bins=18, alph
a=0.5, color='yellow')
rain_days['ENTRIESn_hourly'].plot(kind='hist', bins=18, alpha=0.7,
color='blue')
plt.title('ENTRIESn_hourly HISTOGRAM (by RAIN)')
plt.xlabel('ENTRIESn_hourly')
plt.ylabel('frequency')
plt.legend(['no rain', 'rain'])
plt.show()
```

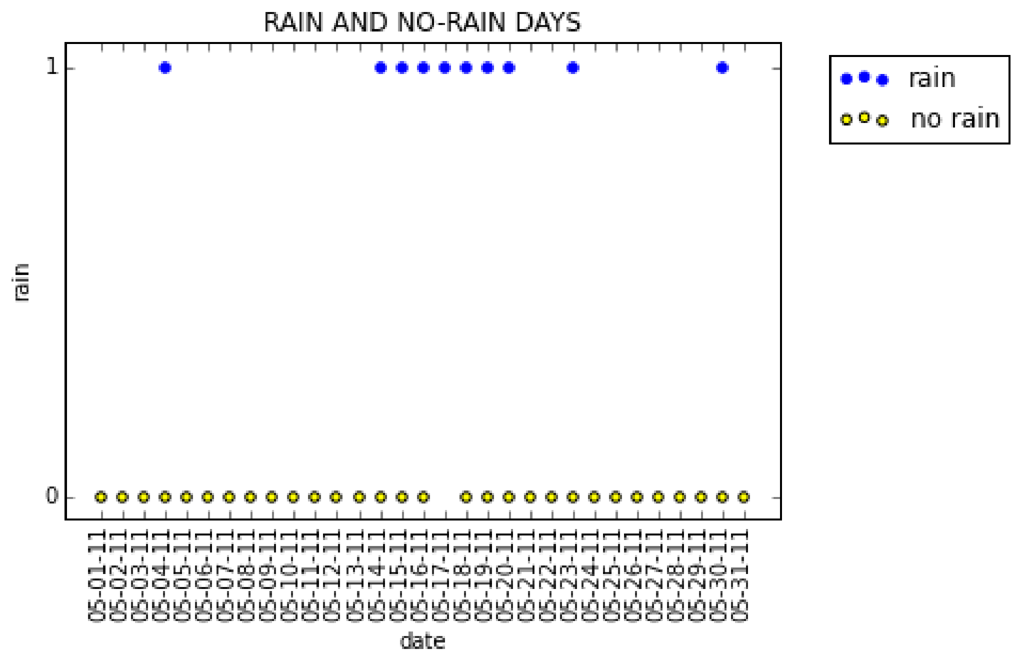
```
Number of non-rainy days: 33064
Number of rainy days: 9585
```



In [66]:

```
date_and_rain = data[['DATEn', 'rain']].drop_duplicates()
date_and_rain.sort(columns='DATEn', inplace=True)
print date_and_rain.head()

dates = data['DATEn'].unique()
rain_dates = date_and_rain[date_and_rain['rain'] == 1]['DATEn'].uni
que()
no_rain_dates = date_and_rain[date_and_rain['rain'] == 0]['DATE
n'].unique()

indices_of_rain_dates = []
for rain_date in rain_dates:
    indices_of_rain_dates.append(np.where(dates == rain_date)[0]
[0])

indices_of_no_rain_dates = []
for no_rain_date in no_rain_dates:
    indices_of_no_rain_dates.append(np.where(dates == no_rain_date)
[0][0])

plt.title('RAIN AND NO-RAIN DAYS')
plt.xticks(np.arange(len(dates)), dates, rotation='vertical')
plt.yticks([0,1])
plt.xlabel('date')
plt.ylabel('rain')

plt.scatter(indices_of_rain_dates, np.ones(len(indices_of_rain_date
s)), color='blue')
plt.scatter(indices_of_no_rain_dates, np.zeros(len(indices_of_no_ra
in_dates)), color='yellow', edgecolors='black')

plt.legend(['rain', 'no rain'], bbox_to_anchor=(1.05, 1), loc=2)
plt.show()
```

```
       DATEn  rain
0      05-01-11     0
5      05-02-11     0
11     05-03-11     0
16     05-04-11     1
32542  05-04-11     0
```

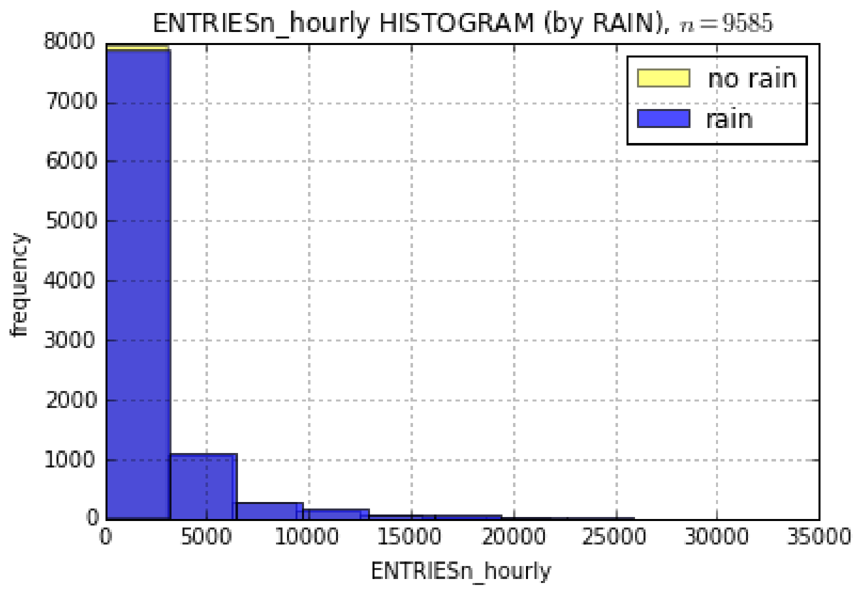


One possible explanation for the difference might be, simply, that there were more non-rainy days in May 2011, as indicated in the graph above. [N.B. Certain days are reported as being rainy and non-rainy days. For a brief exploration of this phenomenon, see the *Rain* supplement (IntroDS-ProjectOne-Rain-Supplement.ipynb).]

What if the number of non-rainy days was limited to the total number of rainy days in the data set?

In [67]:
```
random_row_indices = np.random.choice(no_rain_days['ENTRIESn_hourl
y'].index.values, 9585)
```

In [68]:
```
no_rain_days['ENTRIESn_hourly'].loc[random_row_indices].plot(kin
d='hist', bins=10, alpha=0.5, color='yellow')
rain_days['ENTRIESn_hourly'].plot(kind='hist', bins=10, alpha=0.7,
color='blue')
plt.title(r'ENTRIESn_hourly HISTOGRAM (by RAIN), $n = 9585$')
plt.xlabel('ENTRIESn_hourly')
plt.ylabel('frequency')
plt.legend(['no rain', 'rain'])
plt.show()
```



With the number of samples and bins equal, the similarites between the two groups are obvious.

Rain Summary

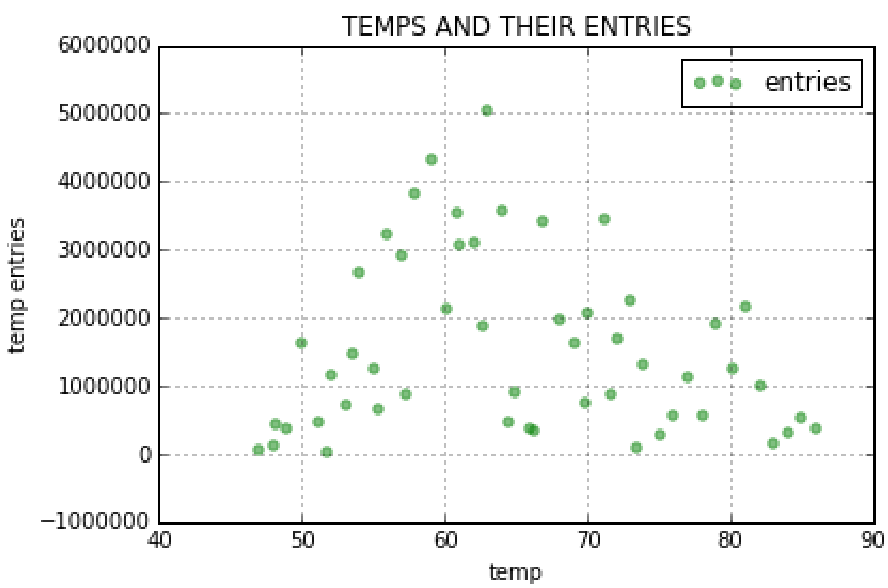
While non-rainy days occur in greater number in this data set (thus, contributing to their higher-frequency counts), the distribution of *ENTRIESn_hourly* for rainy and non-rainy days seems otherwise comparable according to the above histograms.

Temperature Statistics

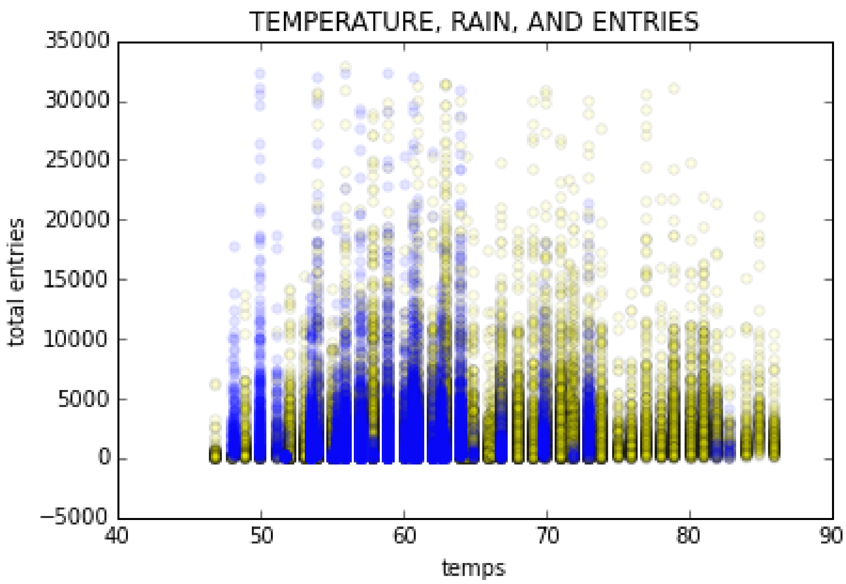
In [69]:

```
temp_entries_hourly = map_column_to_entries_hourly(data['tempi'])
temp_df = display_basic_stats(temp_entries_hourly, 'temp')
plot_data(temp_df, 'temp', 'scatter', True)
```

```
TEMPS AND THEIR ENTRIES
   temp  entries
0  51.1   471715
1  57.9  3831355
2  86.0   362824

              entries
count       52.000000
mean   1547330.288462
std    1276860.663369
min      32822.000000
25%     471789.250000
50%    1203004.000000
75%    2193937.250000
max    5037972.000000
range  5005150.000000
```



In [70]:
```
entries_rain_temp = data[['ENTRIESn_hourly', 'rain', 'tempi']].drop_duplicates()
entries_no_rain_temp = entries_rain_temp[entries_rain_temp['rain'] == 0]
entries_rain_temp = entries_rain_temp[entries_rain_temp['rain'] == 1]

plt.scatter(entries_no_rain_temp['tempi'], entries_no_rain_temp['ENTRIESn_hourly'], color='yellow', edgecolors='black', alpha=0.1)
plt.scatter(entries_rain_temp['tempi'], entries_rain_temp['ENTRIESn_hourly'], color='blue', alpha=0.1)
plt.title('TEMPERATURE, RAIN, AND ENTRIES')
plt.xlabel('temps')
plt.ylabel('total entries')
plt.show()
```



Temperature Summary

Based on the above scatter plots, there does seem to be some type of relationship between temperatures and number of entries. In general, temperatures between 55°F and 66°F received more entries.

Visually-speaking, cold and rainy days seemed to attract approximately the same number of entries as warm and non-rainy (esp. if it's remembered that there were more non-rainy days in the data set).

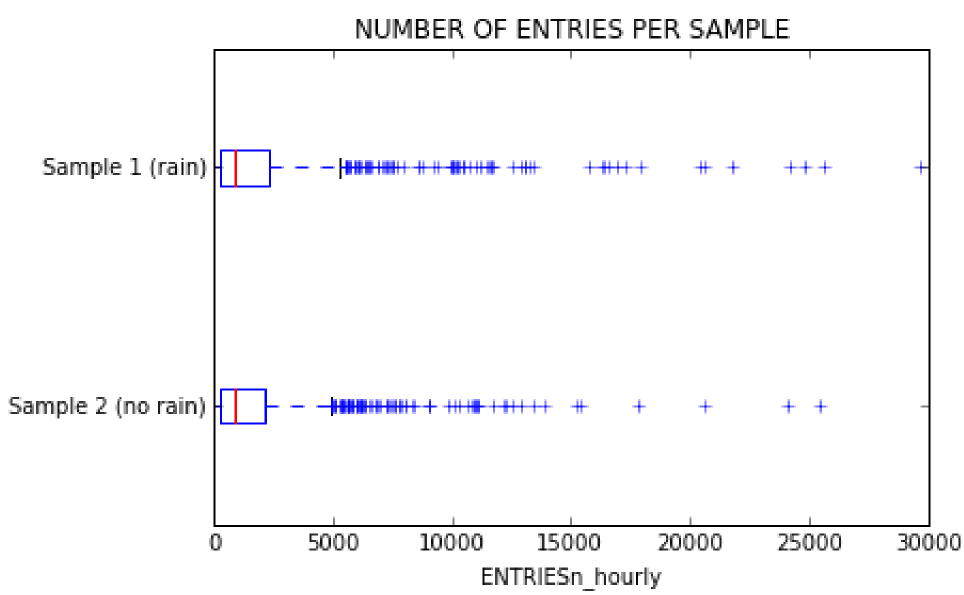
Preparation for Statistical Tests: Looking for Normality

Taking simple random samples, where n: sample size, and $n = 1000$, the following basic statistics reveal themselves.

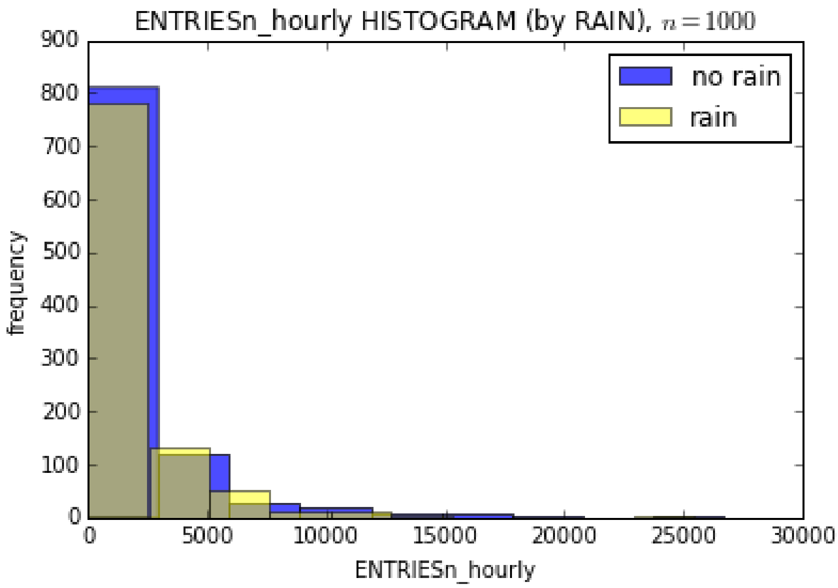
In [71]:
```
n = 1000
sample1 = np.random.choice(rain_days['ENTRIESn_hourly'], size=n, re
place=False)
sample2 = np.random.choice(no_rain_days['ENTRIESn_hourly'], size=n,
replace=False)

describe_samples(sample1,sample2)
```

```
Sample 1 (rainy days):
  min = 0.0, max = 29665.0,
  mean = 1975.33, median = 923.0, var = 10059424.13, std = 3170.07
Sample 2 (non-rainy days):
  min = 0.0, max = 25457.0,
  mean = 1803.53, median = 875.5, var = 6855341.46, std = 2616.96
```

In [72]:
```
plt.boxplot([sample2, sample1], vert=False)
plt.title('NUMBER OF ENTRIES PER SAMPLE')
plt.xlabel('ENTRIESn_hourly')
plt.yticks([1, 2], ['Sample 2 (no rain)', 'Sample 1 (rain)'])

plt.show()
```



In [77]:
```
plt.hist(sample1, color='blue', alpha=0.7, bins=10)
plt.hist(sample2, color='yellow', alpha=0.5, bins=10)
plt.title(r'ENTRIESn_hourly HISTOGRAM (by RAIN), $n = 1000$')
plt.xlabel('ENTRIESn_hourly')
plt.ylabel('frequency')
plt.legend(['no rain', 'rain'])
plt.show()
```



Testing for Normality

Treating the rain and non-rain days as two independent populations, and although visually apparent from the above histogram, the following statistical test seeks to determine whether rainy and non-rainy day distributions are normal.

The Shapiro-Wilk normality test is a test of the null hypothesis that a sample is from a population with a normal distribution.

The test confirms the visually apparent non-normality with a small-enough sample size; so here, $n = 30$.

A 95% level of confidence would suggest that 95% of samples would produce similar statistical results.

For a 95% level of confidence, the level of significance (i.e., the probability of making a Type I error) $\alpha = (1 - 0.05) \cdot 100\% = 0.05$.

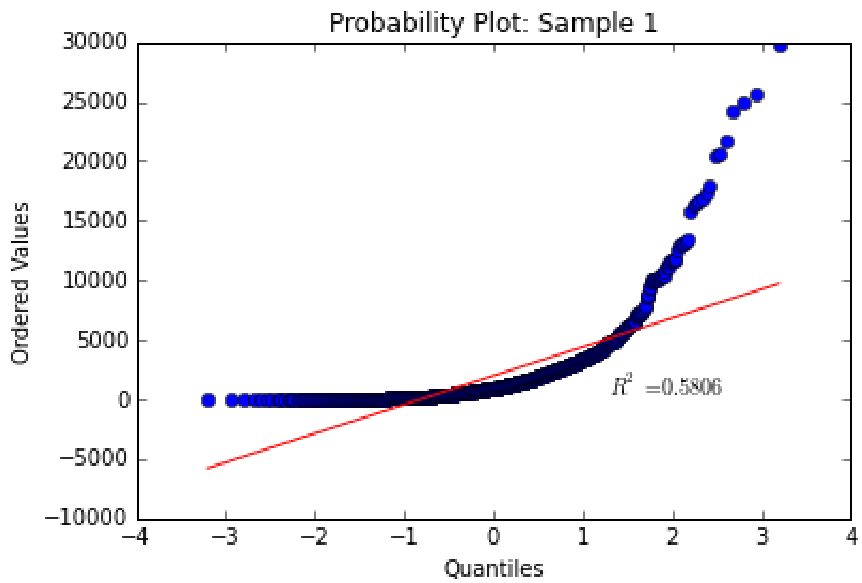
```
In [74]: n = 30
alpha = 0.05
small_sample1 = np.random.choice(sample1, size=n, replace=False)
small_sample2 = np.random.choice(sample1, size=n, replace=False)
W1, p1 = st.shapiro(small_sample1)
W2, p2 = st.shapiro(small_sample2)

print "p1 < {0}: {1}".format(alpha, (p1 < alpha))
print "p2 < {0}: {1}".format(alpha, (p2 < alpha))
```

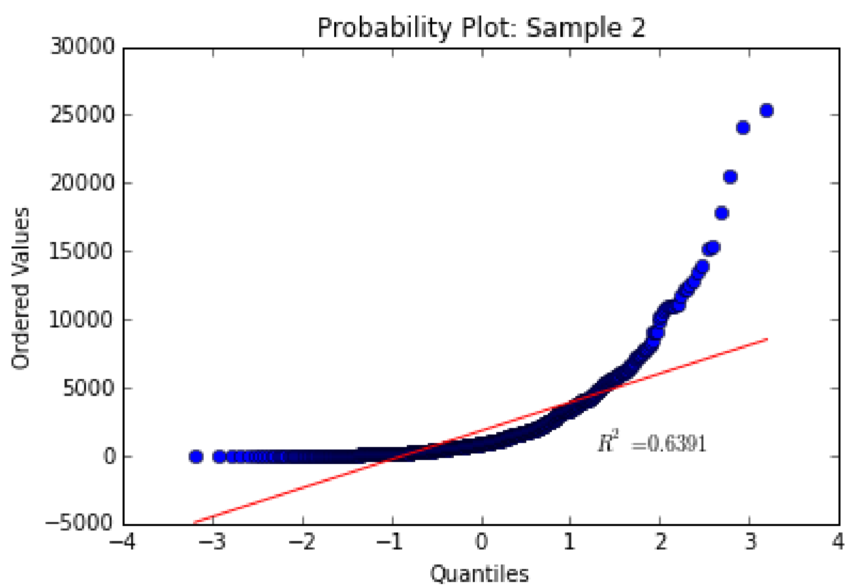
```
p1 < 0.05: True
p2 < 0.05: True
```

In both cases, $p < 0.05$, so the null hypotheses that the samples come from normally distributed populations are rejected.

Moreover, the following probability plots seal the deal (samples from normal distributions would hug the red regression line throughout the plot):

In [75]:
```
st.probplot(sample1, plot=plt)
plt.title('Probability Plot: Sample 1')
plt.show()

st.probplot(sample2, plot=plt)
plt.title('Probability Plot: Sample 2')
plt.show()
```





Apparent Conclusions

Rainy/Non-rainy days and their number of entries are not normally distributed.