New York City Subway Ridership

Given a dataset containing NYC Subway turnstile data from multiple stations as well as weather data (located here (https://www.dropbox.com/s/1lpoeh2w6px4diu/improved-dataset.zip?dl=0)), we would like to know if ridership changes with the weather, or what the busiest days/times are. We would then like to come up with a prediction model that, given a time of day and weather data, it can output the number of expected subway riders.

The data set is small (under 10MB) and can be fully loaded into a Pandas dataframe in memory:

In [1]:
```
from pandas import read_csv

df = read_csv("./turnstile_weather_v2.csv")
```

Let's find out how many stations and turnstile units are part of this dataset, as well as which dates are included.

In [2]:
```
station_names = df['station'].value_counts()
print "Station name and number of times it occurs in the dataset for a fe
w examples: "
print station_names[:5]
print "..."

print "Number of stations: " + str(len(station_names))
```

```
Station name and number of times it occurs in the dataset for a few examp
les:
34 ST-PENN STA    558
86 ST             551
LEXINGTON AVE     549
50 ST             539
145 ST            518
dtype: int64
...
Number of stations: 207
```

In [3]:
```
units_total = df['UNIT'].value_counts().size
units_total
```

Out[3]:
```
240
```

In [4]:
```
min_date = df['datetime'].min()
min_date
```

Out[4]:
```
'2011-05-01 00:00:00'
```

In [5]:
```
max_date = df['datetime'].max()
max_date
```

Out[5]:
```
'2011-05-31 20:00:00'
```

This dataset includes 207 stations (almost half of the total across all of New York City's boroughs) and 240 turnstile units and was collected during the month of May, 2011. There are more units than stations since some stations are larger (like connection hubs) and contain more units which have different IDs. That's why in the listing above some stations appear many times.

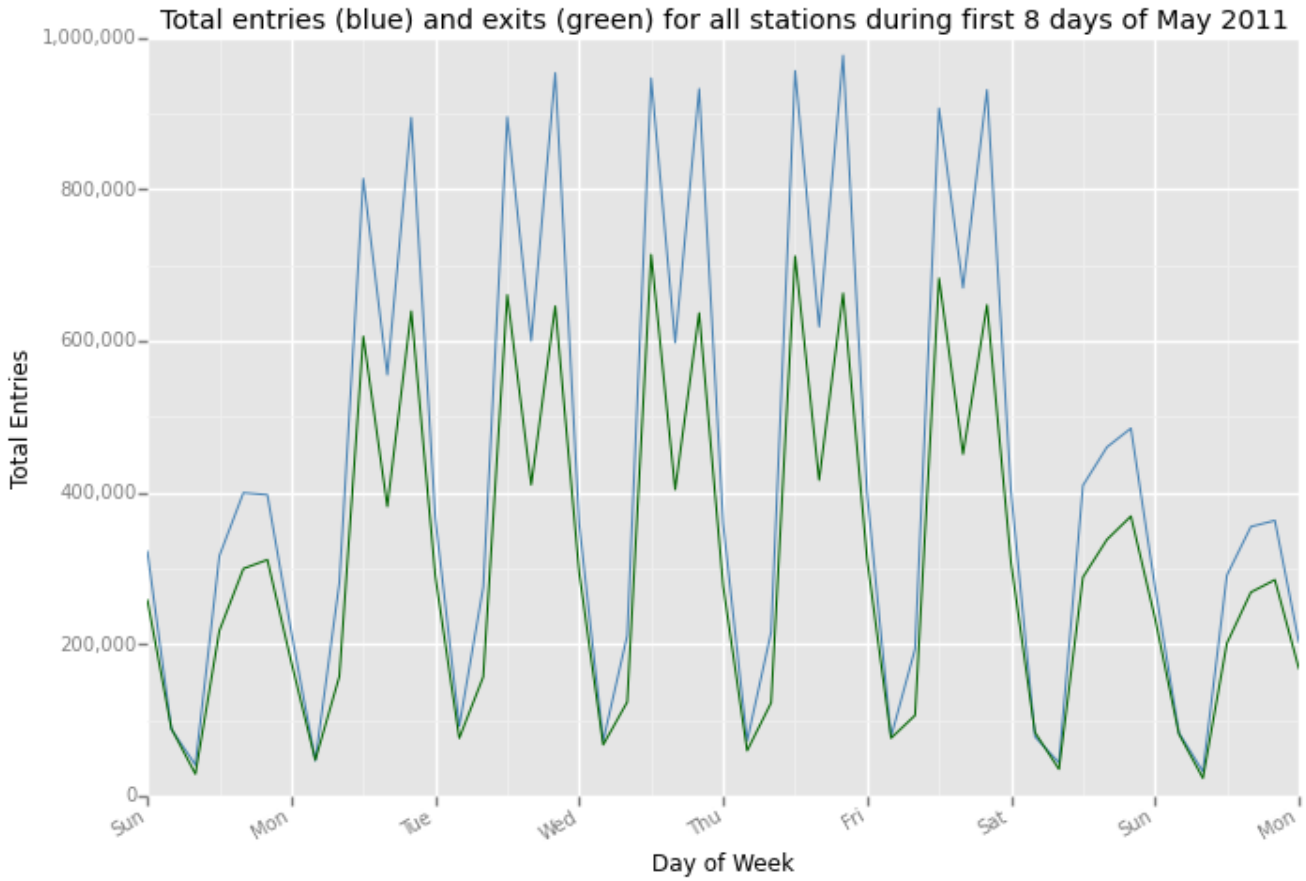
Now let's move on to the interesting part: subway ridership! Let's take a look at the total number of entries, across all stations/units, aggregated at each unique timestamp in our datasets. The improved dataset used in this analysis already has accummulated rows with different timestamps, such that most days will have 6 readings, every 4 hours starting with midnight.

To perform the aggregation, we can use *pandasql* and write a SQL query:

In [6]:
```
query = """select
             datetime as 'datetime',
             sum(ENTRIESn_hourly) as 'entries'
           from df
           where datetime <= "2011-05-09 00:00:00"
           group by datetime"""

from pandasql import *
import datetime
from ggplot import *
%matplotlib inline

# fp: make sure that when converting string dates to date objects
# we don't offset them to a local timezone and display them as they are
os.environ['TZ'] = "UTC"

# fp: convert the datetime column to a date object, since it will be used
 in ggplot
# on the X-axis
total_entries = sqldf(query, locals())
total_entries['datetime'] = total_entries['datetime'].astype('datetime64[
s]')
```

What does this dataframe look like? Let's use *ggplot* for that and display ridership by week, including hourly entries and exits:

In [7]:
```
query = """select
             datetime as 'datetime',
             sum(EXITSn_hourly) as 'exits'
           from df
           where datetime <= "2011-05-09 00:00:00"
           group by datetime"""
# exits in first week of May only
total_exits = sqldf(query, locals())

total_exits['datetime'] = total_exits['datetime'].astype('datetime64[s]')

ggplot(total_entries, aes('datetime', 'entries')) + geom_line(colour="ste
elblue") \
        + scale_x_date(breaks=date_breaks('1 day'), labels='%a') \
        + scale_y_continuous(labels='comma') \
        + ggtitle("Total entries (blue) and exits (green) for all station
s during first 8 days of May 2011") \
        + xlab("Day of Week") + ylab("Total Entries") \
        + geom_line(aes('datetime', 'exits'), data=total_exits, colour="d
arkgreen")
```

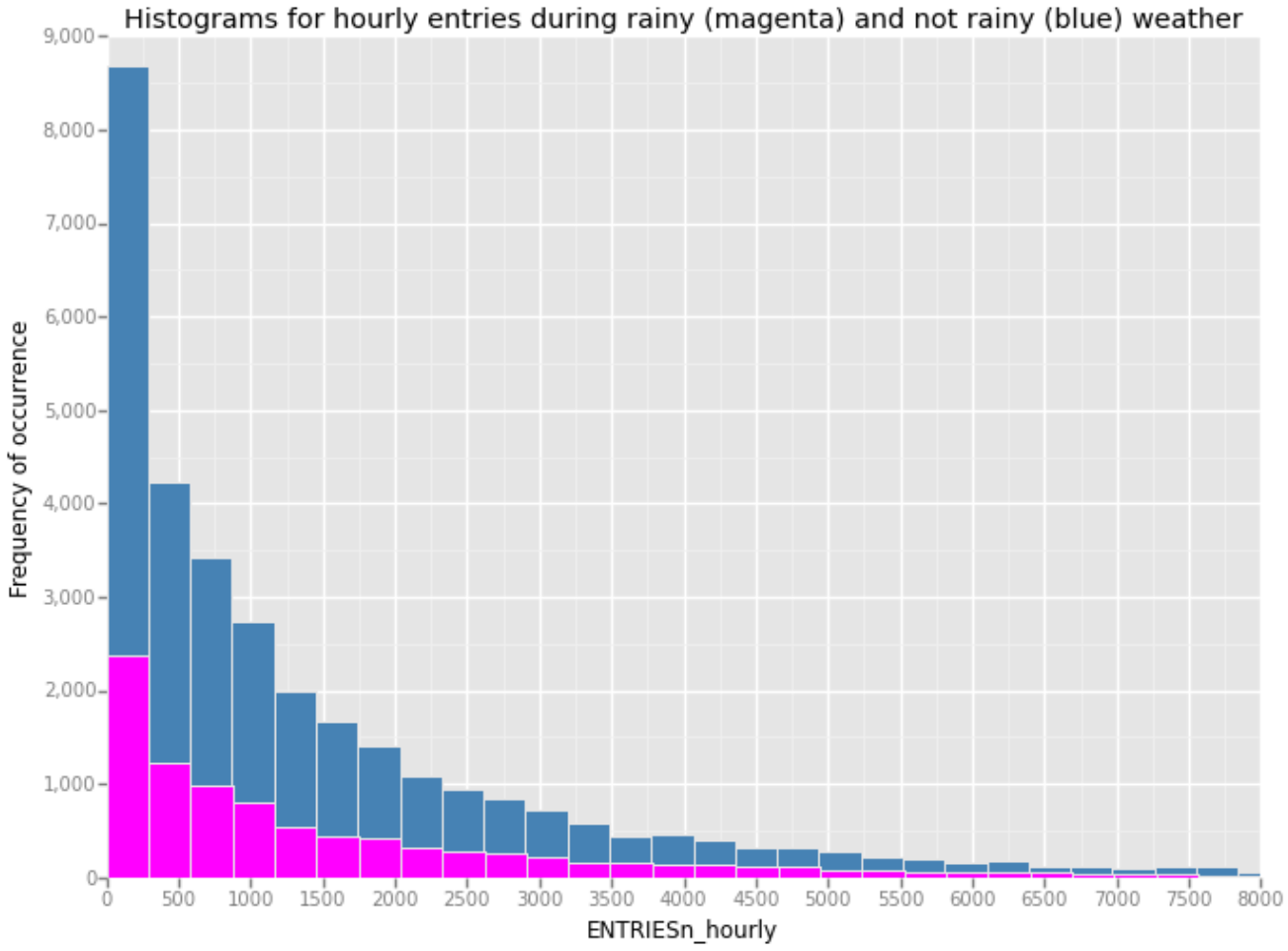


Out[7]: <ggplot: (282135933)>

Weekdays are definitely busier than weekends. Furthermore, weekdays have 2 visible peak times, roughly corresponding to the morning and evening commute, though harder to see because of the 4-hour bucketing of entries.

Let's also look at the hourly entry histograms for rainy and not rainy weather:

In [15]:
```
df_rain = df[df.rain == 1]
df_rain = df_rain.reset_index(drop=True)

df_no_rain = df[df.rain == 0]
df_no_rain = df_no_rain.reset_index(drop=True)

width = 290

ggplot(df_no_rain, aes(x="ENTRIESn_hourly")) + geom_histogram(fill="steel
blue", binwidth=width) \
    + geom_histogram(df_rain, aes(x="ENTRIESn_hourly"), fill="magenta", b
inwidth=width) \
    + scale_x_continuous(limits=(0,8000)) + scale_y_continuous(labels="co
mma") \
    + ggtitle("Histograms for hourly entries during rainy (magenta) and n
ot rainy (blue) weather") \
    + ylab("Frequency of occurrence")
```



Out[15]: <ggplot: (284790225)>

Statistical Analysis

To analyze the subway hourly entries data, during rainy and not rainy weather, we can use the Mann-Whitney U-test since it is a non-parametric test, which does not make any assumptions on the population distributions. In our case, the distribution of ridership is not normal so we cannot use a t-test.

Given 2 sample sets that come from unknown distributions, the U-test will help determine whether one of the sets is more likely to generate higher values or not. We can apply it in our case to two sets, one containing total subway entries when it is raining and the other when it is not raining.

The null hypothesis states that, by randomly drawing an element x from the first set and an element y from the second set, the outcomes $x < y$ and $y > x$ are equally likely. The two sets need not have the same distribution for this to hold so we will need to report additional descriptive statistics for our two sets.

Let's create the two sets, for rainy and not rainy weather, and select the subway entries column:

In [9]:
```
import numpy as np
import scipy

rainy = df[df.rain == 1]['ENTRIESn_hourly']
not_rainy = df[df.rain == 0]['ENTRIESn_hourly']

mean_not_rainy = np.mean(not_rainy)
mean_rainy = np.mean(rainy)

median_not_rainy = np.median(not_rainy)
median_rainy = np.median(rainy)
```

Now we can apply the Mann-Whitney rank test using the scipy implementation:

In [10]:
```
U, p = scipy.stats.mannwhitneyu(rainy, not_rainy)

print "Mann-Whitney test:\t", "U =", U, "\tp =", format(p * 2,"f"), "(two
-sided)"
print "U-statistic distribution mean:\t", (rainy.size * not_rainy.size) /
 2.
print "Mean entries during rainy weahter: ", mean_rainy,"\nMean entries d
uring not rainy weather: ", mean_not_rainy
print "Median for rainy weahter: ", median_rainy,"\nMedian for not rainy
weather: ", median_not_rainy
```

```
Mann-Whitney test:      U = 153635120.5         p = 0.000005 (two-sided)
U-statistic distribution mean:  158459220.0
Mean entries during rainy weahter:  2028.19603547
Mean entries during not rainy weather:  1845.53943866
Median for rainy weahter:  939.0
Median for not rainy weather:  893.0
```

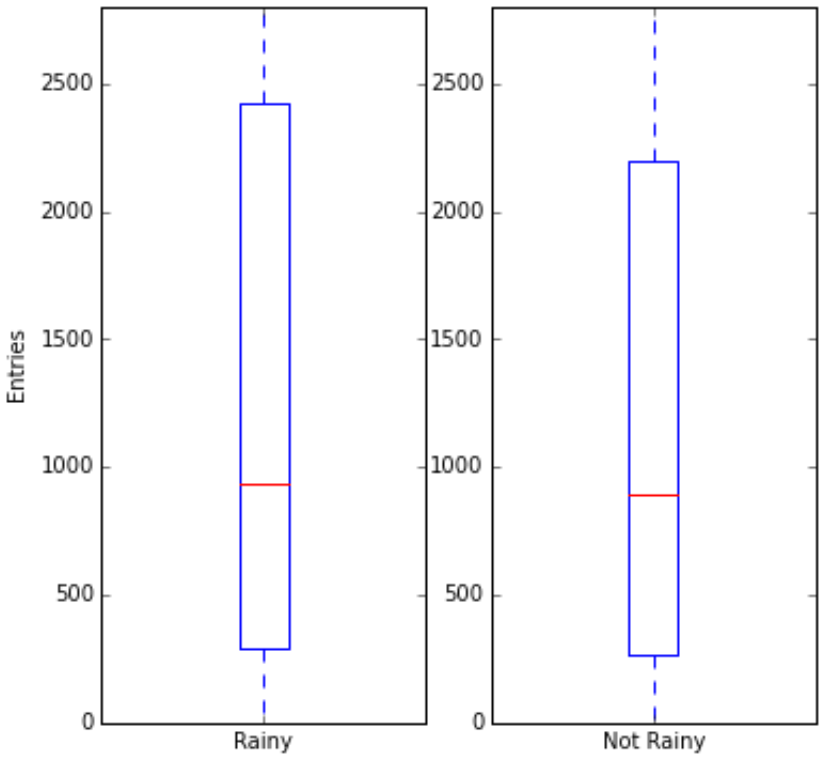
The variable *p* holds the one-sided p-value (as described in the scipy documentation (!http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html)) but we are doubling it since our hypothesis formulation assumes a two-sided test.

The resulting p-value falls below the critical p-value threshold of 0.05, which suggests that the U-statistic is also significantly smaller than the mean of the U-statistic distribution under the null hypothesis. Therefore this would be indicative that the null hypothesis is not the true state of the world.

However, the U-statistic returned by the test may not seem much smaller than the U-statistic distribution mean and we should also look at the descriptive statistics for the two sets. Then we can see that both the mean number of entries and the median for rainy weather are higher than in the case of not rainy weather. Even so, the difference of the medians is quite small.

Let's look at the interquartile range (IQR), which can best be seen on a boxplot (cropped below):

In [11]:
```
import matplotlib.pyplot as plt

rainy = rainy.reset_index(drop=True)
not_rainy = not_rainy.reset_index(drop=True)

fig, axes = plt.subplots(1,2, figsize=(6,6))
axes[0].boxplot(rainy, labels=['Rainy'])
axes[0].set_ylabel("Entries")
axes[1].boxplot(not_rainy, labels=['Not Rainy'])

for ax in axes.flatten():
    ax.set_ylim([0,2800])
```



We notice a slight increase in ridership entries when it is raining, more so at Q3:

In [12]:
```
print "Upper quartile for rainy weather: ", np.percentile(rainy, 75)
print "Upper quartile for not rainy weather: ", np.percentile(not_rainy,
75)
```

```
Upper quartile for rainy weather:  2424.0
Upper quartile for not rainy weather:  2197.0
```

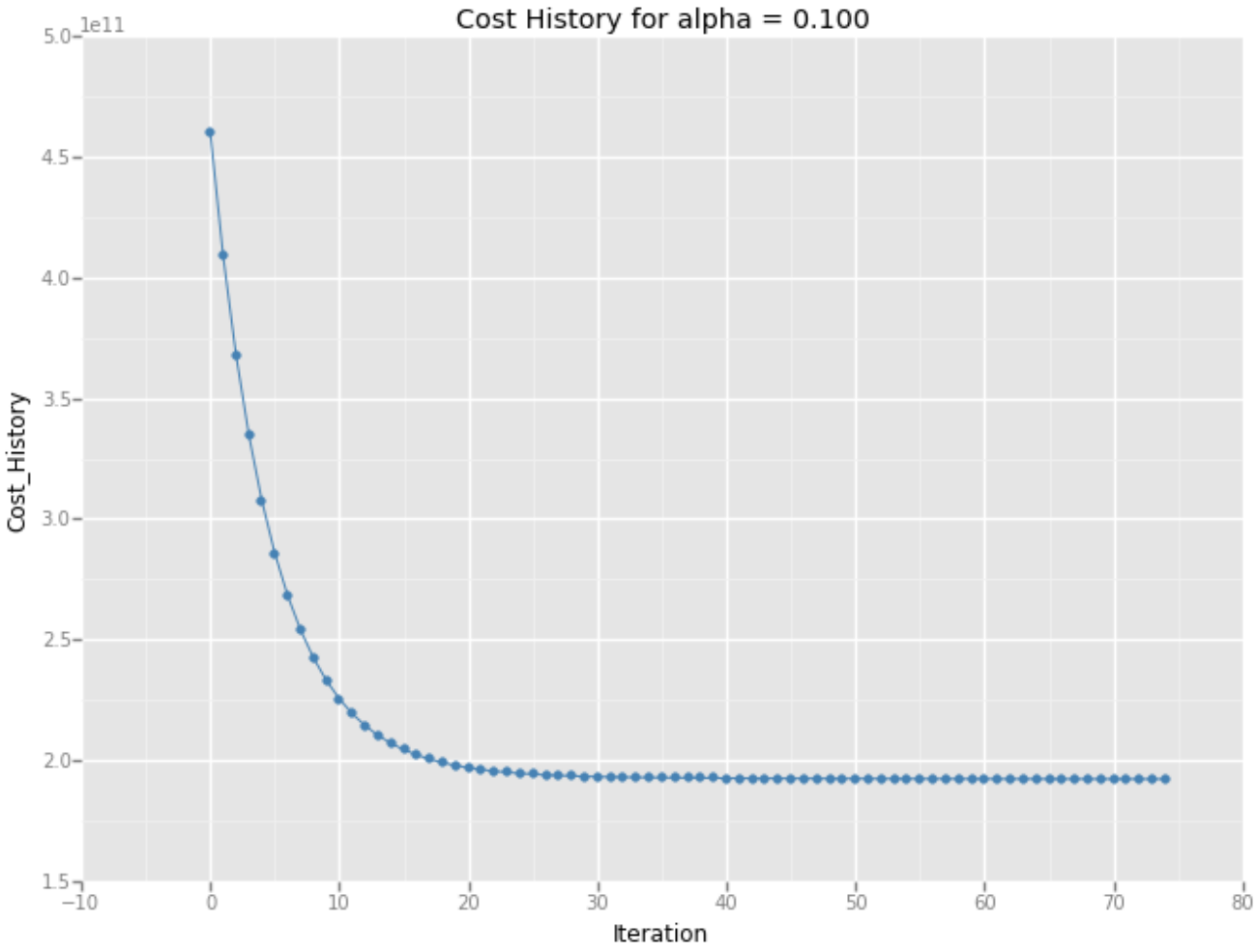
Regression Analysis

We can use linear regression with gradient descent to calculate the theta coefficients and produce predictions for ENTRIESn_hourly.

Intuitively, the features that are likely to influence the ridership are related to the time of day, day of week, whether it's the weekend and the weather. As can be seen in the statistical analysis above, rain did not influence the hourly entry counts by a large amount. Indeed, by including it in the feature list, the change in the coefficient of determination (R squared) was of the order of thousandths. Instead, let's use mean temperature.

In [13]:
```
execfile("./regression.py")

# specify non-dummy features here
non_dummy_features = ['hour', 'weekday', 'day_week', 'meantempi']
total_non_dummy_features = len(non_dummy_features)

# predictions(df, features) returns:
# => predicted values for the training examples
# => the observed values
# => a list of all the features, including dummy variables for categorica
l data (by default, UNIT is included)
# => the coefficients
# => a plot of the cost history over the iterations
predicted, values, features, theta, plot = predictions(df, non_dummy_feat
ures)
print "R squared: ", compute_r_squared(values, predicted)

print "Coefficients for non-dummy variables: "
weights = zip(features.columns, theta)
for pair in weights[:(total_non_dummy_features + 4)]:
    print pair
print "..."
print weights[-1]
print plot
```

```
R squared:  0.482987444101
Coefficients for non-dummy variables:
('hour', 855.73872506180828)
('weekday', 485.30153456533111)
('day_week', 73.540240331780751)
('meantempi', -88.687334520869925)
('unit_R003', -105.64002951400263)
('unit_R004', -86.784416590231118)
('unit_R005', -86.306913319172068)
('unit_R006', -80.565569301134261)
...
('ones', 1885.8919386563641)
```



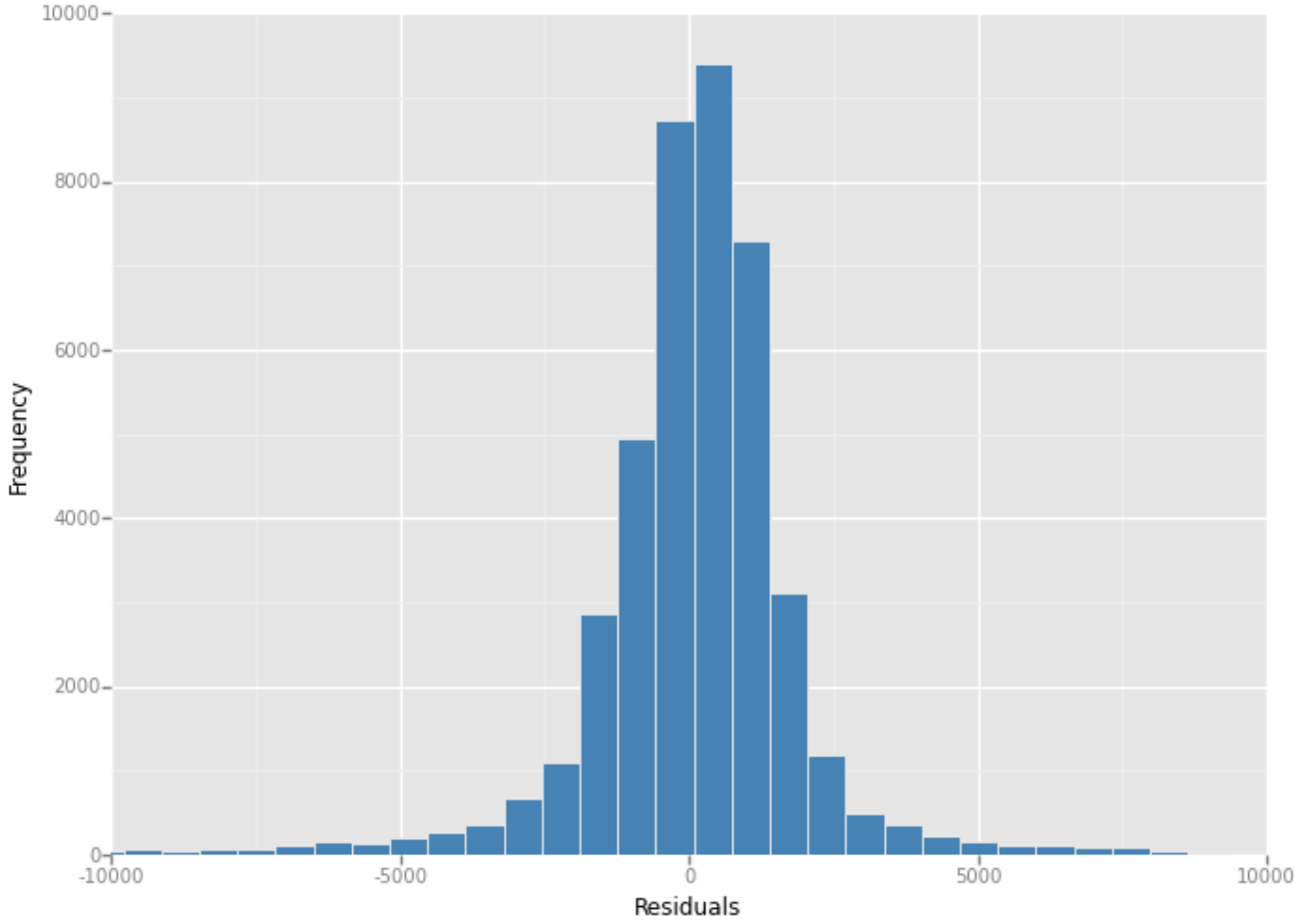

```
<ggplot: (286697753)>
```

Notice that dummy features for each possible value of the UNIT column were added to the model. They are flags which mark whether a training example belongs to a certain unit or not. Only one of them will be 1 for each training example and the rest will be 0.

As it turns out, for features: 'hour', 'weekday', 'day_week' and 'meantempi', the coefficient of determination is about 0.48. The ratio tells us how well the model fits our trainig data, or that 48% of the total variation in the examples is explained by the model. Obtaining a higher value than that is not necessarily the objective here (it may even be alarming if R squared would be 100%, as it may indicate a model that closely fits our training data,

but which performs badly on new examples). To get a better idea we should look at a histogram of residuals, which are differences between the observed examples and predicted values for our dependent variable (the number of entries):

In [14]:
```
residuals = predicted - values
residuals_df = pd.DataFrame({"x": residuals})

ggplot(residuals_df, aes("x")) + geom_bar(fill="steelblue", binwidth=650)
 + \
    scale_x_continuous(limits=(-10000, 10000)) + xlab("Residuals") + ylab
("Frequency")
```



Out[14]: <ggplot: (278059473)>

The histogram looks like a bell-curve approximately centerred on 0. This means that, in probability terms, the model outputs a difference of +x or -x equally often, especially as frequency decreases. However, this is not as accurate for frequently occuring residuals. The histogram suggests that the variation unexplained by this simple model, in the form of random error, follows a Gaussian distribution with mean 0 and some variance. There is no other immediately noticeable source of error that the model does not already take into account.

However, the bell shape of the histogram does not confirm this simple linear model captures the entire story either. For this particular dataset, the model may be sufficient to predict ridership at a given station and date/time to some modest degree of accuracy. As was discovered above, including the features for time of day and day of week had a significant impact on R^2. A more complex model that combines features like station neighborhood, time of day and temperature/precipitation may perform better.

Conclusion

From the analysis conducted above, it seems that there is an increase in the number of riders on the NYC Subway when it is raining, albeit a small one.

The statistical analysis using a Mann-Whitney U test resulted in a p-value falling under the critical threshold. This suggests that, given two sets of hourly entries for rainy and not rainy data, by randomly selecting entries from each set and comparing them, one of the sets is likely to generate higher values. However, the U test is a rank test and does not provide an accurate picture on its own, regarding the similarity of the two distributions. We should also look at descriptive statistics like the median or IQR, which favoured the set containing entries for rainy weather as having higher values. Furthermore, in our regression analysis, adding the rain variable as a feature in the model did not significantly improve the value of R^2 and its theta (the weight) was much lower than those of features like hour or time of day, suggesting that rainy weather does not strongly affect the hourly entries.

However, there are also potential shortcomings of the dataset that need to be considered. In its current form, the dataset may not capture the entire story regarding the number of entries to the subway system. Originally, turnstile data is captured at different times in different stations but for consistency it is placed into 4-hour buckets. These time slots, starting at midnight, may not exactly capture the variation corresponding to NYC riders' commuting times and may also not associate as accurately with the weather data. Furthermore, this analysis used the weather data without taking into account the distance between the subway station and its associated weather station which reported the data. There is a possibility that including the distance will result in a stronger correlation between the number of entries and rainy weather.

Finally, the model used in the regression analysis is very simple and suffers from high bias. Providing additional training data will not help either since, in its current form, the model is not explaining a large portion of the total variation. More complex features of higher degrees may improve the model and the predicted results.