Data Science Final Project: Milestone 3

November 5, 2016

1 Airbnb Pricing Prediction: Milestone 3

James Gearheart Danny Zhuang Bob Saludo Ryan Wallace
Harvard University Fall 2016
TF: Christine Hwang Due Date: Saturday, November 5th, 2016

1.1 Summary of Work and Insights

1.1.1 Data Overview

Formatting The data was retrieved from the data.beta.nyc website by zip file download. The only complication in storing the data is that since the file sizes are over 100MB, the files can not be stored on GitHub. If it proves necessary, GitHub Large File Storage may provide a solution for this issue.

In each of the three datasets, (listings, calendar, and reviews), the data are well formatted overall in CSV format with one record (listing, listing-date, or review, respectively) per row. Given the varied nature of the data, Pandas is a good choice for initial processing, and each of the three datasets is read into its own Pandas data frame. Manual examination of the data shows that the 'id' field in the listings data set corresponds to the 'listing_id' fields in the reviews and calendar data, so joining on these fields is feasible, as demonstrated in the proof of concept above.

There are occasional formatting irregularities throughout the data that have been fixed. For example, some fields contained excess parentheses or string data types where floats or booleans are more appropriate. The price fields were orignally encoded as strings with dollar sign and comma values. These were converted to floats for use in learning models.

Missing Values While the datasets are very complete on average, there are a significant number of missing values. In the listings dataset, roughly 30% of all review scores, the vast majority of the square footage values, and low percentages of other listing characteristics are missing. In the calendar dataset, 28% of the listing-dates do not have price information. In the reviews dataset, an insignificant number of comments are missing.

The missing values were handled in various ways to suit the nature of the data. Missing price data is a roadblock because it is inadvisable to attempt to impute missing response variables. Thus, records with missing price data are dropped. For categorical variables, it is imaginable that knowing which listings have descriptors absent is associated with the price (likely in a negative fashion). Therefore, missing categorical values were replaced with a new category 'unspecified'. Missing review values were imputed with the global mean, as the correlation between price and the majority of review metrics is so low that attempting to use regression for imputation would be useless. Likewise, the data are too noisy for KNN to be of use. For other continuous feature variables such as bedrooms, bathrooms, and beds with relatively high correlation with price, missing values were imputed using linear regression.

After all imputation was finished, the only missing values were those of weekly and monthly prices. These missing values are potential response variables, and will be dealt with on an individual model basis.

Size of Data There are a total of about 27,000 listings in the listings data set, each with 52 features. The calendar dataset contains one entry for each day of the year for each of the approximately 27,000 listings. The reviews dataset contains about 28,000 reviews for about 19,000 distinct listings.

After cleaning, the majority of the data remains. All of the orignal listings remain in the data; over 70% of the calendar records are useful; and the vast majority of the reviews have complete comments.

1.1.2 Features and their Distributions

There are three potential response variables in the data, nighly price, weekly price, and mothly price. Since over half of the weekly and monthly price data are not present, it seems reasonable to focus on predicting nightly price. Nightly prices are right-skewed, with a few very high outliers. The majority of the prices are below 300 dollars per night. The price distribution looks approximately lognormal, so a log transformation of the price variable resulting in a normal distribution of prices may play better with regression techniques.

There are over fifty predictors in the listings dataset, both continuous and categorical. Histograms of the continuous variables of interest and lists of the most frequent categorical variables of interest appear above. Interesting factors to note are that the majority of listings have one bedroom and one bathroom, and that reviews are extremely left-skewed.

1.1.3 Visualize the supply/price of Airbnb homes by location

We notice that in general, the overwhelming majority of rental properties are concentrated in the center of the grid (between 40.6 - 40.9 Latitude, and -73.8 - -74.1 Longitude. Outside of this dense range, we have sparse units in the outskirts of NYC which are predominantly listed under the 'unspecified' category for neighborhood and are own the lower end of the pricing spectrum.

On the whole, the most expensive rentals are in the area around 40.7 Latitude and -74.0 Longitude. As we move outwards in all directions, prices generally become less expensive. One possible way that we could use this observation is to determine the epicenter in terms of high-priced housing, and then calculate the euclidean distance column for all points in the dataset and adding this distance into our model as a predictor. Additionally, we could use the neighborhood predictor variable in our model because the top 10 neighborhoods are each separate and extremely concentrated. We think it is logical to suspect that there will be clear differences between the neighborhoods of NYC, with clustering within the neighborhoods and at least some pricing separation between the neighborhoods. We see this in the average rental prices for each of the 191 neighborhoods in our dataset that we have printed above.

For our final presentation, we would likely want to re-produce these scatterplots except using the map of NYC as the background instead of a whitespace background.

1.1.4 Correlations between Price and Features

The Pearson correlations between price and each feature variable of interest is plotted above in decreasing order of correlation. From the chart, we see that the basic physical features of the listings are most strongly linearly related with price: number of individuals accomodated, number of beds and bedrooms, and number of bathrooms have the highest correlation values. The strongest negative linear relationships with price are for the number of listings by the given host, number of reviews, and the longitude of the listing. These results all make intuitive sense. Hosts with many listings may be less attentive to their tenants, tenants may be more inclined to leave a negative review, and knowledge of the demographics of New York supports the observed negative relationship between price and longitude.

While studying these correlation values are useful, they only describe the magnitude of the linear relationship between price and the predictors, so additional visual anlaysis is necessary.

1.1.5 Relationship between Price and Time

There are strong and interestings relationships between the prices of listings and both the time of year and the day of the week. Two different sets of statistics were computed to investigate these relationships. First, the simple average price over all the listings was calculated for each day of the year, and for each day of the week. Since this may overlook certain patterns, the differences between the price of a property on each day of the year and its average price throughout the year was also calculated, and averaged accross all the properties for each day of the year.

Both sets of metrics show very similar trends. Late in the month of January is very cheap: all of the fifteen dates with the largest negative average difference from mean listing price occurred in the month of January. Most dates are up to 8 dollars cheaper on average. In striking contrast, early January (presumably because of New Years celebrations) are the most expensive, with the first few days averaging over 10 dollars more expensive than usual. The spring and early summer months are also more expensive by about 5 dollars on average for peak times.

The relationship between the day of the week and the average difference in listing price from its mean is also very strong. Weekdays and Sunday are all similarly priced. However, Fridays and Saturdays average 3.50 dollars more expensive than weeknights.

In the modelling stage, it is clear that time (in terms of both date and day of the week) will be an important feature. The interaction between the date and day of the week may also prove helpful.

1.1.6 Analyze Target Variable and need for Transformation

The target variable of "price" was analyzed to determine the variable's skewness. Histograms and boxplots were created to visualize the distribution of the target variable. These visualizations confirmed that price is highly skewed to the right. This indicates that the price variable contains several extreme outliers and the need to transform this variable to ensure that it is normally distributed and prepared for predictive modeling.

The "price" variable was log transformed and new visualizations were created to demonstrate the normal distribution of the log transformed target variable. Boxplots were also created to demonstrate that the transformation contained the outliers. Several additional visualizations were created that plotted the log transformed target variable against several of the possible preditors.

In [2]:
```
# import necessary libraries
import csv
import datetime
import operator
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression as LinReg
%matplotlib inline
```

1.2 Read and Format Data

In [3]:
```
# remove commas from prices in calendar CSV
with open('datasets/calendar.csv', 'rb') as f, open('datasets/calendar_clea
    writer = csv.writer(g, delimiter=',')
    for line in f:
        # split into four columns
        row = line.split(',', 3)
        writer.writerow(row)

# read the three datasets
listings_df = pd.read_csv('datasets/listings.csv', delimiter=',')
reviews_df = pd.read_csv('datasets/reviews.csv', delimiter=',')
calendar_df = pd.read_csv('datasets/calendar_cleaned.csv')
```

In [4]:
```
print 'listings dataframe: \n'
listings_df.head()
```

```
listings dataframe:
```

Out[4]:
```
        id     scrape_id last_scraped                                   name
0  1069266  2.015010e+13       1/2/15           Stay like a real New Yorker!
1  1846722  2.015010e+13       1/2/15       Apartment 20 Minutes Times Square
2  2061725  2.015010e+13       1/2/15     Option of 2 Beds w Private Bathroom
3    44974  2.015010e+13       1/3/15     Charming Bright West Village Studio
4  4701675  2.015010e+13       1/2/15           Charming Apartment in Chelsea

                                         picture_url   host_id host_name  \
0  https://a0.muscache.com/pictures/50276484/larg...   5867023   Michael
1  https://a1.muscache.com/pictures/35865039/larg...   2631556    Denise
2  https://a2.muscache.com/pictures/50650147/larg...   4601412      Miao
3  https://a1.muscache.com/pictures/20489905/larg...    198425      Sara
4  https://a2.muscache.com/pictures/60588955/larg...  22590025   Charles
```

```
  host_since                                   host_picture_url  \
0    4/10/13  https://a2.muscache.com/ic/users/5867023/profi...
1    6/13/12  https://a2.muscache.com/ic/users/2631556/profi...
2     1/5/13  https://a0.muscache.com/ic/users/4601412/profi...
3    8/11/10  https://a0.muscache.com/ic/users/198425/profil...
4   10/15/14  https://a2.muscache.com/ic/users/22590025/prof...

                                              street        ...          \
0  East 53rd Street, New York, NY 10022, United S...        ...
1     West 155th Street, New York, NY, United States        ...
2  Van Buren Street, Brooklyn, NY 11221, United S...        ...
3   Greenwich Ave, New York, NY 10011, United States        ...
4  West 22nd Street, New York, NY 10011, United S...        ...

  first_review last_review review_scores_rating review_scores_accuracy  \
0      4/28/13    12/17/14                 86.0                    9.0
1       1/5/14    12/29/14                 85.0                    8.0
2       2/4/14    12/29/14                 98.0                   10.0
3      10/8/10    10/30/14                 96.0                   10.0
4      12/8/14     12/8/14                100.0                   10.0

  review_scores_cleanliness review_scores_checkin review_scores_communicati
0                       7.0                   9.0                         9
1                       8.0                   9.0                         8
2                      10.0                  10.0                        10
3                       9.0                  10.0                        10
4                      10.0                  10.0                        10

   review_scores_location  review_scores_value host_listing_count
0                    10.0                  9.0                  1
1                     7.0                  8.0                  2
2                     9.0                 10.0                  4
3                    10.0                  9.0                  1
4                    10.0                 10.0                  1

[5 rows x 52 columns]
```

```
In [5]: print 'reviews dataframe: \n'
        reviews_df.head()
```

```
reviews dataframe:
```

```
Out[5]:    listing_id        id        date  reviewer_id reviewer_name  \
        0     1180670  14705995  2014-06-24     10875598       Gregory
        1     4457617  24432844  2014-12-28     24502047         Amber
```

```
2      722394   9248781  2013-12-16      6821360         Giri
3     4074444  23983183  2014-12-15      8822691        Wendy
4       68046  11797670  2014-04-15     12231047     Virginie

                                               comments
0  Ok, if you like the location and don't mind an...
1  Kleine süße WG, super gelegen, sehr freundlich...
2                          Extremely disappointed.
3                            Exactly as described.
4  Appartement très sympa, accueillant. A quelque...
```

```
In [6]: print 'calendar dataframe: \n'
        calendar_df.head()
```

```
calendar dataframe:
```

```
Out[6]:    "listing_id"        "date" "available"  "price"\n
        0       3604481  "2015-01-01"           t  $600.00\n
        1       3604481  "2015-01-02"           t  $600.00\n
        2       3604481  "2015-01-03"           t  $600.00\n
        3       3604481  "2015-01-04"           t  $600.00\n
        4       3604481  "2015-01-05"           t  $600.00\n
```

```
In [7]: # method to smartly convert price strings to floats
        def convert_float(val):
            try:
                if type(val) == float:
                    return val
                else:
                    return float(val.strip('[\$\n]').replace(',',''))
            except ValueError:
                return np.nan
```

```
In [8]: # fix wonky formatting of data and field names
        # remove surronding parens and newlines from column names
        old_columns = listings_df.columns.values
        new_columns = [col.replace('"', '').replace('\n','') for col in old_columns
        listings_df.columns = new_columns

        old_columns = reviews_df.columns.values
        new_columns = [col.replace('"', '').replace('\n','') for col in old_columns
        reviews_df.columns = new_columns

        old_columns = calendar_df.columns.values
        new_columns = [col.replace('"', '').replace('\n','') for col in old_columns
        calendar_df.columns = new_columns
```

```
# remove parens around date in calendar_df
calendar_df['date'] = calendar_df['date'].apply(lambda x: x.replace('"', ''

# change t and f to 1 and 0 in calendar_df
calendar_df['available'] = calendar_df['available'].apply(lambda x: 1 if x

# change string types to floats when possible
listings_df['price'] = listings_df['price'].apply(lambda x: convert_float(x
listings_df['weekly_price'] = listings_df['weekly_price'].apply(lambda x: c
listings_df['monthly_price'] = listings_df['monthly_price'].apply(lambda x:
calendar_df['price'] = calendar_df['price'].apply(lambda x: convert_float(x
```

```
In [9]: print 'listings dataframe: \n'
        listings_df.head()
```

```
listings dataframe:
```

```
Out[9]:         id     scrape_id last_scraped                               name
        0  1069266  2.015010e+13       1/2/15        Stay like a real New Yorker!
        1  1846722  2.015010e+13       1/2/15    Apartment 20 Minutes Times Square
        2  2061725  2.015010e+13       1/2/15  Option of 2 Beds w Private Bathroom
        3    44974  2.015010e+13       1/3/15  Charming Bright West Village Studio
        4  4701675  2.015010e+13       1/2/15        Charming Apartment in Chelsea

                                                 picture_url   host_id host_name  \
        0  https://a0.muscache.com/pictures/50276484/larg...   5867023   Michael
        1  https://a1.muscache.com/pictures/35865039/larg...   2631556    Denise
        2  https://a2.muscache.com/pictures/50650147/larg...   4601412      Miao
        3  https://a1.muscache.com/pictures/20489905/larg...    198425      Sara
        4  https://a2.muscache.com/pictures/60588955/larg...  22590025   Charles

          host_since                                  host_picture_url  \
        0    4/10/13  https://a2.muscache.com/ic/users/5867023/profi...
        1    6/13/12  https://a2.muscache.com/ic/users/2631556/profi...
        2     1/5/13  https://a0.muscache.com/ic/users/4601412/profi...
        3    8/11/10  https://a0.muscache.com/ic/users/198425/profil...
        4   10/15/14  https://a2.muscache.com/ic/users/22590025/prof...

                                                      street        ...         \
        0  East 53rd Street, New York, NY 10022, United S...        ...
        1     West 155th Street, New York, NY, United States        ...
        2  Van Buren Street, Brooklyn, NY 11221, United S...        ...
        3   Greenwich Ave, New York, NY 10011, United States        ...
        4  West 22nd Street, New York, NY 10011, United S...        ...

          first_review last_review review_scores_rating review_scores_accuracy  \
```

```
0      4/28/13    12/17/14                  86.0                    9.0
1       1/5/14    12/29/14                  85.0                    8.0
2       2/4/14    12/29/14                  98.0                   10.0
3      10/8/10    10/30/14                  96.0                   10.0
4      12/8/14     12/8/14                 100.0                   10.0

  review_scores_cleanliness review_scores_checkin review_scores_communicati
0                       7.0                   9.0                         9
1                       8.0                   9.0                         8
2                      10.0                  10.0                        10
3                       9.0                  10.0                        10
4                      10.0                  10.0                        10

   review_scores_location  review_scores_value host_listing_count
0                    10.0                  9.0                  1
1                     7.0                  8.0                  2
2                     9.0                 10.0                  4
3                    10.0                  9.0                  1
4                    10.0                 10.0                  1

[5 rows x 52 columns]
```

```
In [10]: print 'reviews dataframe: \n'
         reviews_df.head()
```

```
reviews dataframe:
```

```
Out[10]:    listing_id        id        date  reviewer_id reviewer_name  \
         0     1180670  14705995  2014-06-24     10875598       Gregory
         1     4457617  24432844  2014-12-28     24502047         Amber
         2      722394   9248781  2013-12-16      6821360          Giri
         3     4074444  23983183  2014-12-15      8822691         Wendy
         4       68046  11797670  2014-04-15     12231047     Virginie

                                                    comments
         0  Ok, if you like the location and don't mind an...
         1  Kleine süße WG, super gelegen, sehr freundlich...
         2                          Extremely disappointed.
         3                            Exactly as described.
         4  Appartement très sympa, accueillant. A quelque...
```

```
In [11]: print 'calendar dataframe: \n'
         calendar_df.head()
```

```
calendar dataframe:
```

Out[11]:
```
   listing_id        date  available  price
0     3604481  2015-01-01          1  600.0
1     3604481  2015-01-02          1  600.0
2     3604481  2015-01-03          1  600.0
3     3604481  2015-01-04          1  600.0
4     3604481  2015-01-05          1  600.0
```

In [12]:
```
# examine size of data
print 'Listings Data: \nnumber of listings: ', listings_df.shape[0]

print '\nCalendar Data: \nnumber of listing-dates: ', calendar_df.shape[0]
print 'number of distinct listings on calendar: ', len(calendar_df['listin
print 'number of dates on calendar per listing: ', calendar_df.shape[0] /

print '\nReviews Data: \nnumber of reviews: ', reviews_df.shape[0]
print 'number of distinct listings reviewed: ', len(reviews_df['listing_ic
```

```
Listings Data:
number of listings:  27392

Calendar Data:
number of listing-dates:  9998080
number of distinct listings on calendar:  27392
number of dates on calendar per listing:  365

Reviews Data:
number of reviews:  277659
number of distinct listings reviewed:  19028
```

In [13]:
```
# determine which values features have missing data
print 'Listings Missing Data:'
for feature in listings_df.columns.values:
    num_nulls = sum(listings_df[feature].isnull())
    if num_nulls != 0:
        print '\t' + feature + ': ' + str(num_nulls) + ', ' + str(round(1(

print '\nCalendar Missing Data:'
for feature in calendar_df.columns.values:
    num_nulls = sum(calendar_df[feature].isnull())
    if num_nulls != 0:
        print '\t' + feature + ': ' + str(num_nulls) + ', ' + str(round(1(

print '\nReviews Missing Data:'
for feature in reviews_df.columns.values:
    num_nulls = sum(reviews_df[feature].isnull())
    if num_nulls != 0:
        print '\t' + feature + ': ' + str(num_nulls) + ', ' + str(round(1(
```

```
Listings Missing Data:
        neighbourhood: 2027, 7.0%
        state: 2, 0.0%
        zipcode: 162, 1.0%
        country: 1, 0.0%
        property_type: 6, 0.0%
        bathrooms: 463, 2.0%
        bedrooms: 140, 1.0%
        beds: 98, 0.0%
        square_feet: 26386, 96.0%
        weekly_price: 15374, 56.0%
        monthly_price: 17558, 64.0%
        first_review: 8364, 31.0%
        last_review: 8364, 31.0%
        review_scores_rating: 8657, 32.0%
        review_scores_accuracy: 8727, 32.0%
        review_scores_cleanliness: 8731, 32.0%
        review_scores_checkin: 8729, 32.0%
        review_scores_communication: 8731, 32.0%
        review_scores_location: 8732, 32.0%
        review_scores_value: 8734, 32.0%

Calendar Missing Data:
        price: 2796197, 28.0%

Reviews Missing Data:
        comments: 164, 0.0%
```

```
In [14]: # impute missing data
         # some features can be dropped because they are all the same
         listings_df = listings_df.drop(['country', 'state'], 1, errors='ignore')

         # there is so little of some data it is useless
         listings_df = listings_df.drop(['square_feet'], 1, errors='ignore')

         # some data is necessary so observations must be dropped
         calendar_df = calendar_df[pd.notnull(calendar_df['price'])]
         reviews_df = reviews_df[pd.notnull(reviews_df['comments'])]

         # review dates are mostly irrelevant
         listings_df = listings_df.drop(['first_review', 'last_review'], 1, errors=

         # neighbourhood, is categorical, so create new 'unspecified' category to a
         listings_df['neighbourhood'] = listings_df['neighbourhood'].apply(lambda x
             'unspecified' if pd.isnull(x) else x)

         # impute zipcode and property_type with modes
```

```
listings_df['zipcode'] = listings_df['zipcode'].apply(lambda x:
    11211 if pd.isnull(x) else x)
listings_df['property_type'] = listings_df['property_type'].apply(lambda x
    'Apartment' if pd.isnull(x) else x)

# weekly and montly price are response variables so missing values can be
# specifically predicting these values

# reviews have low correlation with price; no KNN too costly, use means
review_scores_rating_mean = np.mean(listings_df['review_scores_rating'])
review_scores_accuracy_mean = np.mean(listings_df['review_scores_accuracy'
review_scores_cleanliness_mean = np.mean(listings_df['review_scores_cleanl
review_scores_checkin_mean = np.mean(listings_df['review_scores_checkin'])
review_scores_communication_mean = np.mean(listings_df['review_scores_comm
review_scores_location_mean = np.mean(listings_df['review_scores_location'
review_scores_value_mean = np.mean(listings_df['review_scores_value'])

listings_df['review_scores_rating'] = listings_df['review_scores_rating'].
    review_scores_rating_mean if np.isnan(x) else x)
listings_df['review_scores_accuracy'] = listings_df['review_scores_accurac
    review_scores_accuracy_mean if np.isnan(x) else x)
listings_df['review_scores_cleanliness'] = listings_df['review_scores_clea
    review_scores_cleanliness_mean if np.isnan(x) else x)
listings_df['review_scores_checkin'] = listings_df['review_scores_checkin'
    review_scores_checkin_mean if np.isnan(x) else x)
listings_df['review_scores_communication'] = listings_df['review_scores_co
    review_scores_communication_mean if np.isnan(x) else x)
listings_df['review_scores_location'] = listings_df['review_scores_locatio
    review_scores_location_mean if np.isnan(x) else x)
listings_df['review_scores_value'] = listings_df['review_scores_value'].ap
    review_scores_value_mean if np.isnan(x) else x)

# bathrooms, bedrooms, beds have relatively high correlation with price so
for feature in ['bathrooms', 'bedrooms', 'beds']:
    model = LinReg()
    Xy = listings_df[[feature,'price']]
    Xy = Xy[pd.notnull(Xy[feature])]
    model.fit(np.array(Xy['price']).reshape(-1, 1), np.array(Xy[feature]))
    for index, row in listings_df.iterrows():
        if pd.isnull(listings_df.iloc[index][feature]):
            listings_df.set_value(index, feature, model.predict(listings_d
```

```
In [15]: # see what missing values remain
         print 'Listings Missing Data (after cleaning):'
         for feature in listings_df.columns.values:
             num_nulls = sum(listings_df[feature].isnull())
             if num_nulls != 0:
                 print '\t' + feature + ': ' + str(num_nulls) + ', ' + str(round(10
```

```
print '\nCalendar Missing Data (after cleaning):'
for feature in calendar_df.columns.values:
    num_nulls = sum(calendar_df[feature].isnull())
    if num_nulls != 0:
        print '\t' + feature + ': ' + str(num_nulls) + ', ' + str(round(10

print '\nReviews Missing Data (after cleaning):'
for feature in reviews_df.columns.values:
    num_nulls = sum(reviews_df[feature].isnull())
    if num_nulls != 0:
        print '\t' + feature + ': ' + str(num_nulls) + ', ' + str(round(10
```

```
Listings Missing Data (after cleaning):
        weekly_price: 15374, 56.0%
        monthly_price: 17558, 64.0%

Calendar Missing Data (after cleaning):

Reviews Missing Data (after cleaning):
```

In [16]:
```
# examine size of data after cleaning
print 'Listings Data: \nnumber of listings (cleaned): ', listings_df.shape

print '\nCalendar Data: \nnumber of listing-dates (cleaned): ', calendar_d
print 'number of distinct listings on calendar (cleaned): ', len(calendar_
print 'number of dates on calendar per listing (cleaned): ', calendar_df.s

print '\nReviews Data: \nnumber of reviews (cleaned): ', reviews_df.shape[
print 'number of distinct listings reviewed (cleaned): ', len(reviews_df['
```

```
Listings Data:
number of listings (cleaned):  27392

Calendar Data:
number of listing-dates (cleaned):  7201883
number of distinct listings on calendar (cleaned):  26803
number of dates on calendar per listing (cleaned):  268

Reviews Data:
number of reviews (cleaned):  277495
number of distinct listings reviewed (cleaned):  19025
```

In [17]:
```
# merge listings and reviews by listing_id - proof of concept
# first standardize column name
new_columns = listings_df.columns.values
```

```
new_columns[0] = 'listing_id'
listings_df.columns = new_columns

# merge listings and reviews (only one per listing) by listing_id
# left merge because only reviews with listing info are valuable
listings_reviews_df = listings_df.join(reviews_df, on='listing_id', how='l

print 'combined listing and reviews dataframe: \n'
listings_reviews_df.head()
```

```
combined listing and reviews dataframe:
```

```
Out[17]:    listing_id_listing     scrape_id last_scraped  \
         0             1069266  2.015010e+13       1/2/15
         1             1846722  2.015010e+13       1/2/15
         2             2061725  2.015010e+13       1/2/15
         3               44974  2.015010e+13       1/3/15
         4             4701675  2.015010e+13       1/2/15

                                         name  \
         0        Stay like a real New Yorker!
         1    Apartment 20 Minutes Times Square
         2  Option of 2 Beds w Private Bathroom
         3  Charming Bright West Village Studio
         4        Charming Apartment in Chelsea

                                                 picture_url   host_id host_name
         0  https://a0.muscache.com/pictures/50276484/larg...   5867023   Michael
         1  https://a1.muscache.com/pictures/35865039/larg...   2631556    Denise
         2  https://a2.muscache.com/pictures/50650147/larg...   4601412      Miao
         3  https://a1.muscache.com/pictures/20489905/larg...    198425      Sara
         4  https://a2.muscache.com/pictures/60588955/larg...  22590025   Charles

           host_since                                   host_picture_url  \
         0    4/10/13  https://a2.muscache.com/ic/users/5867023/profi...
         1    6/13/12  https://a2.muscache.com/ic/users/2631556/profi...
         2     1/5/13  https://a0.muscache.com/ic/users/4601412/profi...
         3    8/11/10  https://a0.muscache.com/ic/users/198425/profil...
         4   10/15/14  https://a2.muscache.com/ic/users/22590025/prof...

                                                 street  \
         0  East 53rd Street, New York, NY 10022, United S...
         1     West 155th Street, New York, NY, United States
         2  Van Buren Street, Brooklyn, NY 11221, United S...
         3   Greenwich Ave, New York, NY 10011, United States
         4  West 22nd Street, New York, NY 10011, United S...
```

```
                         ...                             \
0                        ...
1                        ...
2                        ...
3                        ...
4                        ...

  review_scores_communication review_scores_location review_scores_value
0                         9.0                   10.0                 9.0
1                         8.0                    7.0                 8.0
2                        10.0                    9.0                10.0
3                        10.0                   10.0                 9.0
4                        10.0                   10.0                10.0

  host_listing_count listing_id          id        date reviewer_id  \
0                  1        NaN         NaN         NaN         NaN
1                  2        NaN         NaN         NaN         NaN
2                  4        NaN         NaN         NaN         NaN
3                  1   871973.0  15182818.0  2014-07-04  15393693.0
4                  1        NaN         NaN         NaN         NaN

  reviewer_name                                           comments
0           NaN                                                NaN
1           NaN                                                NaN
2           NaN                                                NaN
3         Maeva  Very comfortable bed, plenty of room in the dr...
4           NaN                                                NaN

[5 rows x 53 columns]
```

1.3 Examine Fields

```
In [18]: # investigate fields
         print 'listings fields:\n',
         for field in listings_df.columns.values: print '\t' + field
         print '\ncalendar fields:\n',
         for field in calendar_df.columns.values: print '\t' + field
         print '\n reviews fields:\n',
         for field in reviews_df.columns.values: print '\t' + field
```

```
listings fields:
        listing_id
        scrape_id
        last_scraped
        name
```

```
        picture_url
        host_id
        host_name
        host_since
        host_picture_url
        street
        neighbourhood
        neighbourhood_cleansed
        city
        zipcode
        market
        latitude
        longitude
        is_location_exact
        property_type
        room_type
        accommodates
        bathrooms
        bedrooms
        beds
        bed_type
        price
        weekly_price
        monthly_price
        guests_included
        extra_people
        minimum_nights
        maximum_nights
        calendar_updated
        availability_30
        availability_60
        availability_90
        availability_365
        calendar_last_scraped
        number_of_reviews
        review_scores_rating
        review_scores_accuracy
        review_scores_cleanliness
        review_scores_checkin
        review_scores_communication
        review_scores_location
        review_scores_value
        host_listing_count

calendar fields:
        listing_id
        date
        available
```

```
        price

 reviews fields:
        listing_id
        id
        date
        reviewer_id
        reviewer_name
        comments
```

1.4 Examine Values

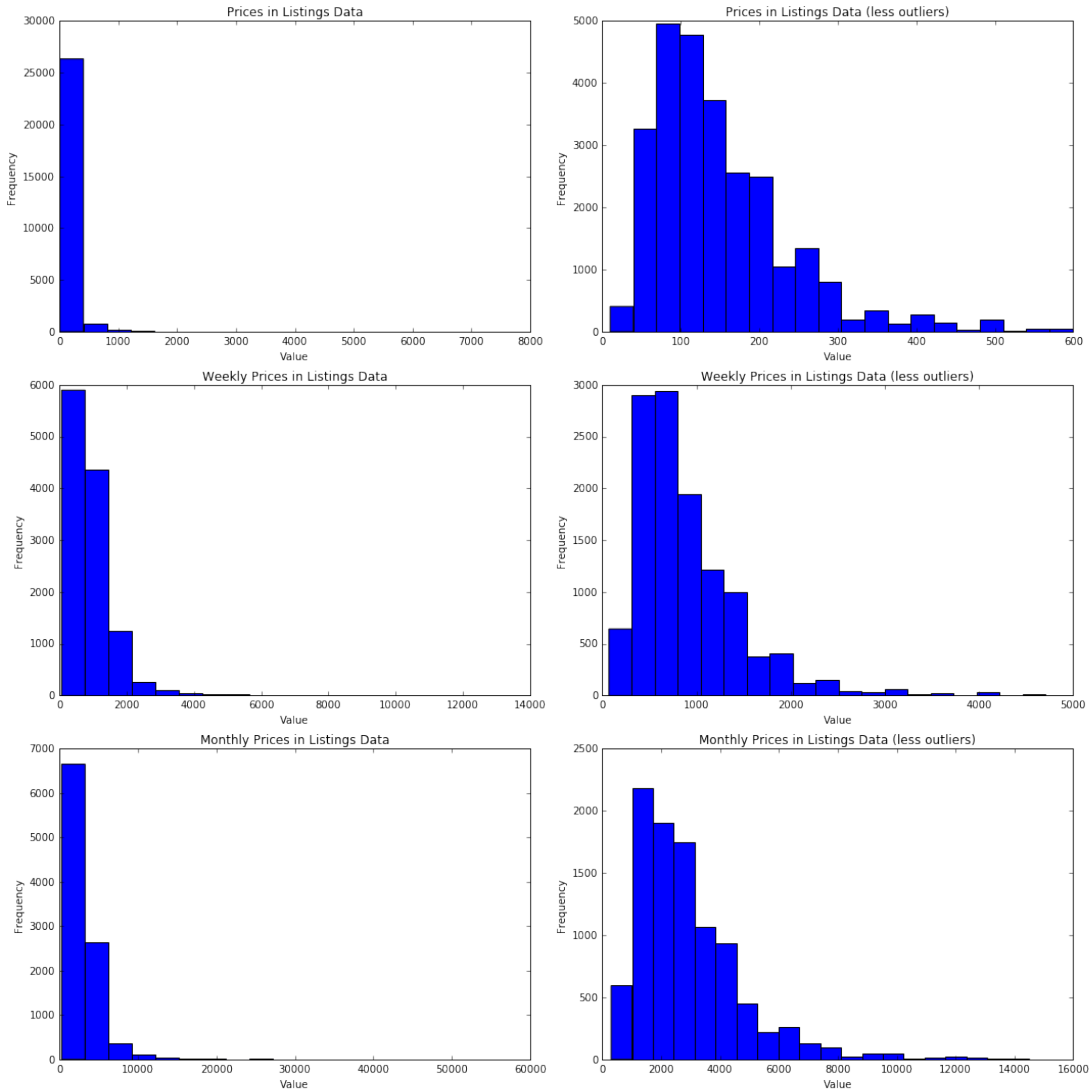
1.4.1 Distribution of Data

Let's first investigate the spread of individual variables of interest.

```
In [19]: # plots a simple histogram of the data, x.
         def plot_histogram(x, title, ax, num_bins):
             # Plot data
             ax.hist(x, bins=num_bins)

             # Label axes, set title
             ax.set_title(title)
             ax.set_xlabel('Value')
             ax.set_ylabel('Frequency')

             return ax
```

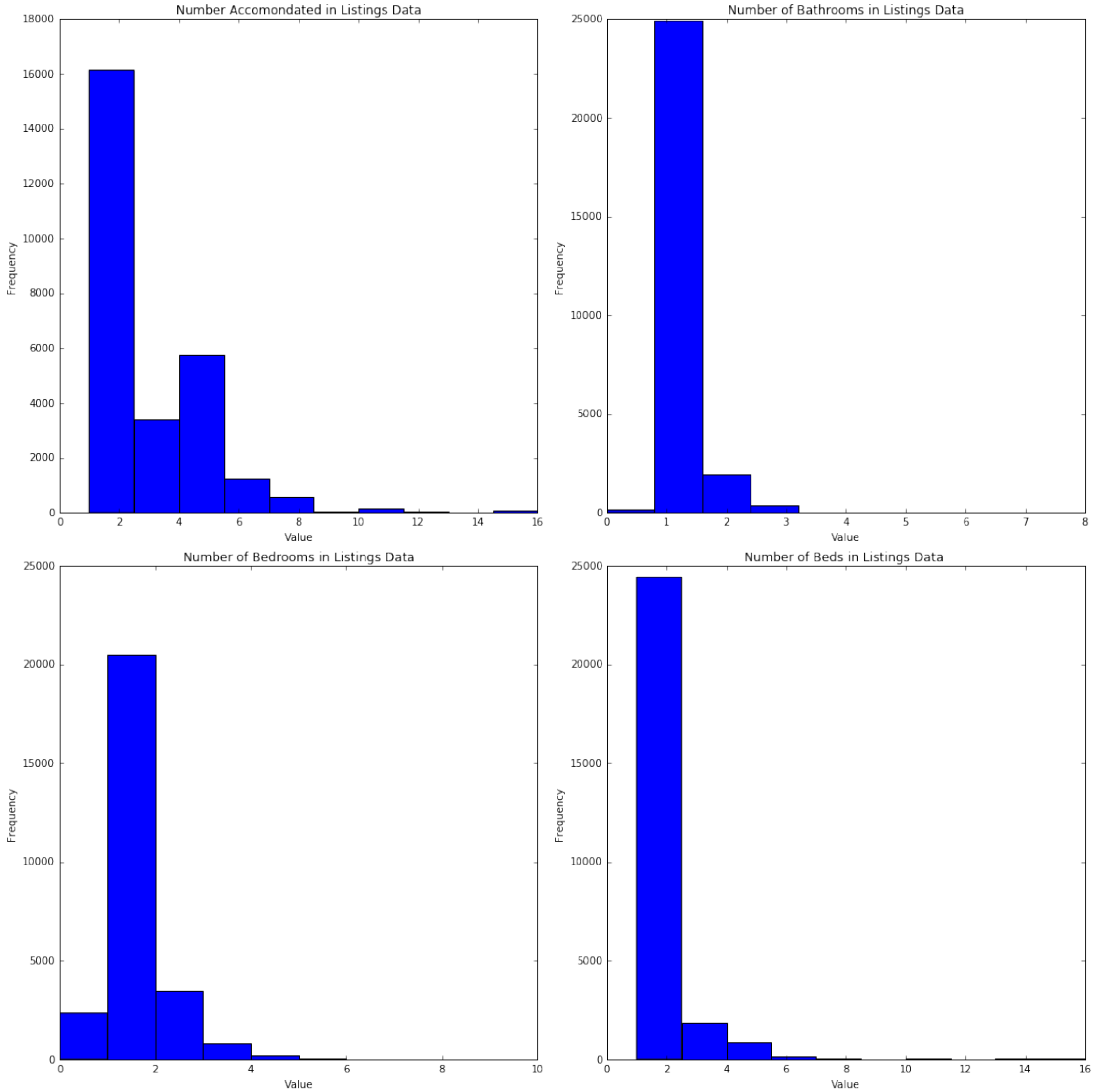
```
In [20]: # np arrays for numerical series
         price = [val for val in np.array(listings_df['price']) if ~np.isnan(val)]
         weekly_price = [val for val in np.array(listings_df['weekly_price']) if ~n
         monthly_price = [val for val in np.array(listings_df['monthly_price']) if

         # remove out outliers
         price_out = [p for p in price if p < 600]
         weekly_price_out = [p for p in weekly_price if p < 5000]
         monthly_price_out = [p for p in monthly_price if p < 15000]

         # price related variables
         fig, ax = plt.subplots(3, 2, figsize=(15, 15))
         ax[0][0] = plot_histogram(price, 'Prices in Listings Data', ax[0][0], 20)
         ax[0][1] = plot_histogram(price_out, 'Prices in Listings Data (less outlie

         ax[1][0] = plot_histogram(weekly_price, 'Weekly Prices in Listings Data',
         ax[1][1] = plot_histogram(weekly_price_out, 'Weekly Prices in Listings Dat
```

```
ax[2][0] = plot_histogram(monthly_price, 'Monthly Prices in Listings Data'
ax[2][1] = plot_histogram(monthly_price_out, 'Monthly Prices in Listings D
plt.tight_layout()
plt.show()
```



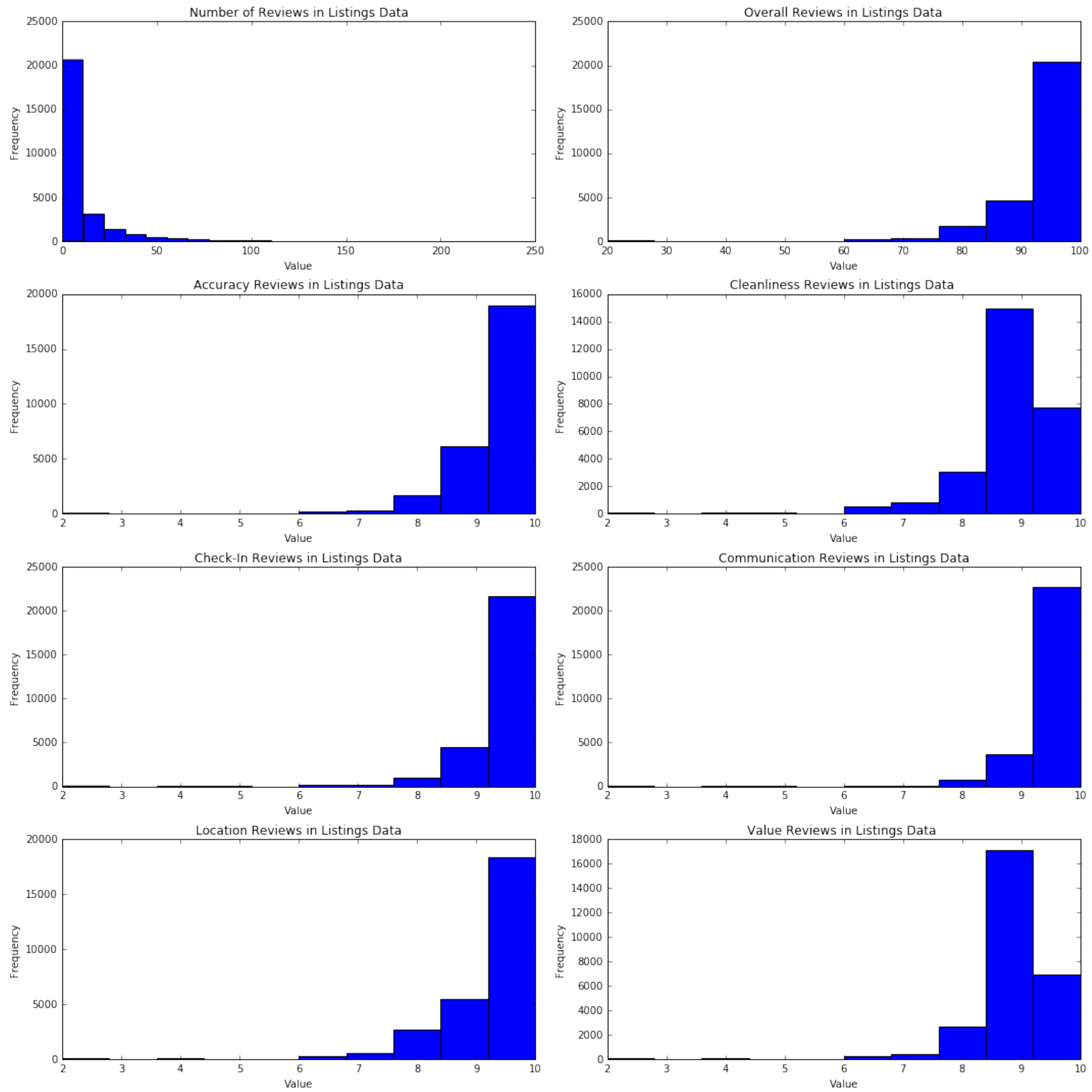

```
In [21]: # accomodations related variables
         accommodates = np.array(listings_df['accommodates'])
         bathrooms = [val for val in np.array(listings_df['bathrooms']) if ~np.isna
         bedrooms = [val for val in np.array(listings_df['bedrooms']) if ~np.isnan(
         beds = [val for val in np.array(listings_df['beds']) if ~np.isnan(val)]

         fig, ax = plt.subplots(2, 2, figsize=(15, 15))
         ax[0][0] = plot_histogram(accommodates, 'Number Accomondated in Listings D
```

```
ax[0][1] = plot_histogram(bathrooms, 'Number of Bathrooms in Listings Data

ax[1][0] = plot_histogram(bedrooms, 'Number of Bedrooms in Listings Data',
ax[1][1] = plot_histogram(beds, 'Number of Beds in Listings Data ', ax[1][

plt.tight_layout()
plt.show()
```




```
In [22]: # reviews related variables
         number_of_reviews = np.array(listings_df['number_of_reviews'])
         review_scores_rating = [val for val in np.array(listings_df['review_scores
         review_scores_accuracy = [val for val in np.array(listings_df['review_scor
         review_scores_cleanliness = [val for val in np.array(listings_df['review_s
```

```
review_scores_checkin = [val for val in np.array(listings_df['review_score
review_scores_communication = [val for val in np.array(listings_df['review
review_scores_location = [val for val in np.array(listings_df['review_scor
review_scores_value = [val for val in np.array(listings_df['review_scores_

fig, ax = plt.subplots(4, 2, figsize=(15, 15))
ax[0][0] = plot_histogram(number_of_reviews, 'Number of Reviews in Listing
ax[0][1] = plot_histogram(review_scores_rating, 'Overall Reviews in Listin

ax[1][0] = plot_histogram(review_scores_accuracy, 'Accuracy Reviews in Lis
ax[1][1] = plot_histogram(review_scores_cleanliness, 'Cleanliness Reviews 

ax[2][0] = plot_histogram(review_scores_checkin, 'Check-In Reviews in List
ax[2][1] = plot_histogram(review_scores_communication, 'Communication Revi

ax[3][0] = plot_histogram(review_scores_location, 'Location Reviews in Lis
ax[3][1] = plot_histogram(review_scores_value, 'Value Reviews in Listings 

plt.tight_layout()
plt.show()
```




```
In [23]: # categorical geographical variables by examining most frequent entries
         top_neighborhood = sorted(listings_df.groupby('neighbourhood').size().to_d
         top_zipcode = sorted(listings_df.groupby('zipcode').size().to_dict().items
         top_property_type = sorted(listings_df.groupby('property_type').size().to_

         print 'Top Neighborhoods'
         for k, v in top_neighborhood[:10]:
             print str(k) + ': ' + str(v)

         print '\nTop Zipcodes'
         for k, v in top_zipcode[:10]:
             print str(k) + ': ' + str(v)
```

```
        print '\nTop Property Types'
        for k, v in top_property_type[:10]:
            print str(k) + ': ' + str(v)
```

```
Top Neighborhoods
unspecified: 2027
Williamsburg: 1878
Upper West Side: 1307
Upper East Side: 1183
Hell's Kitchen: 1177
Bedford-Stuyvesant: 1112
Bushwick: 1023
Lower East Side: 942
Harlem: 861
East Village: 842

Top Zipcodes
11211: 1200
10002: 1130
10009: 1062
10003: 948
10011: 855
11238: 814
10014: 797
10019: 760
11216: 697
10012: 685

Top Property Types
Apartment: 24915
House: 1575
Loft: 601
Bed & Breakfast: 170
Dorm: 49
Other: 48
Boat: 11
Treehouse: 6
Villa: 4
Cabin: 3
```

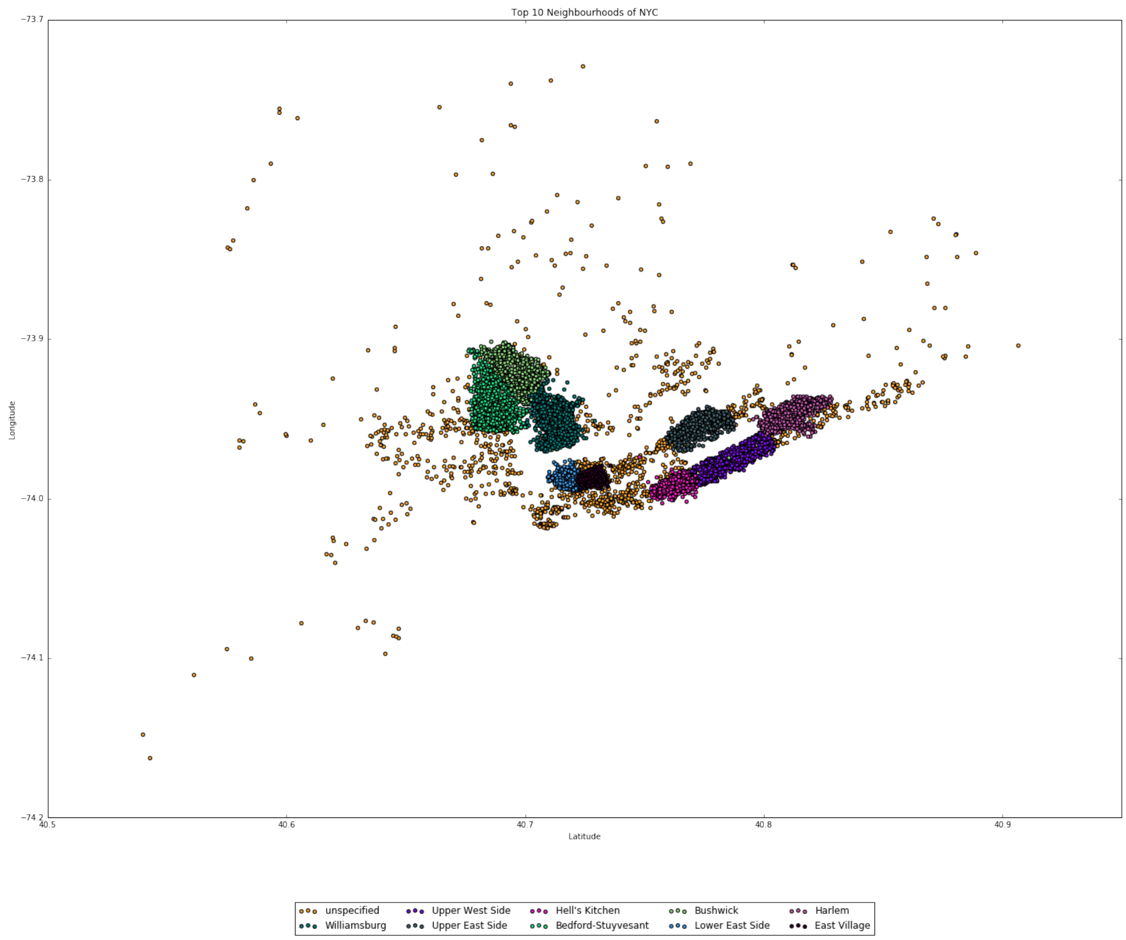
1.4.2 Visualize the supply of Airbnb homes by location and be sure to reflect the price of the average home in each region

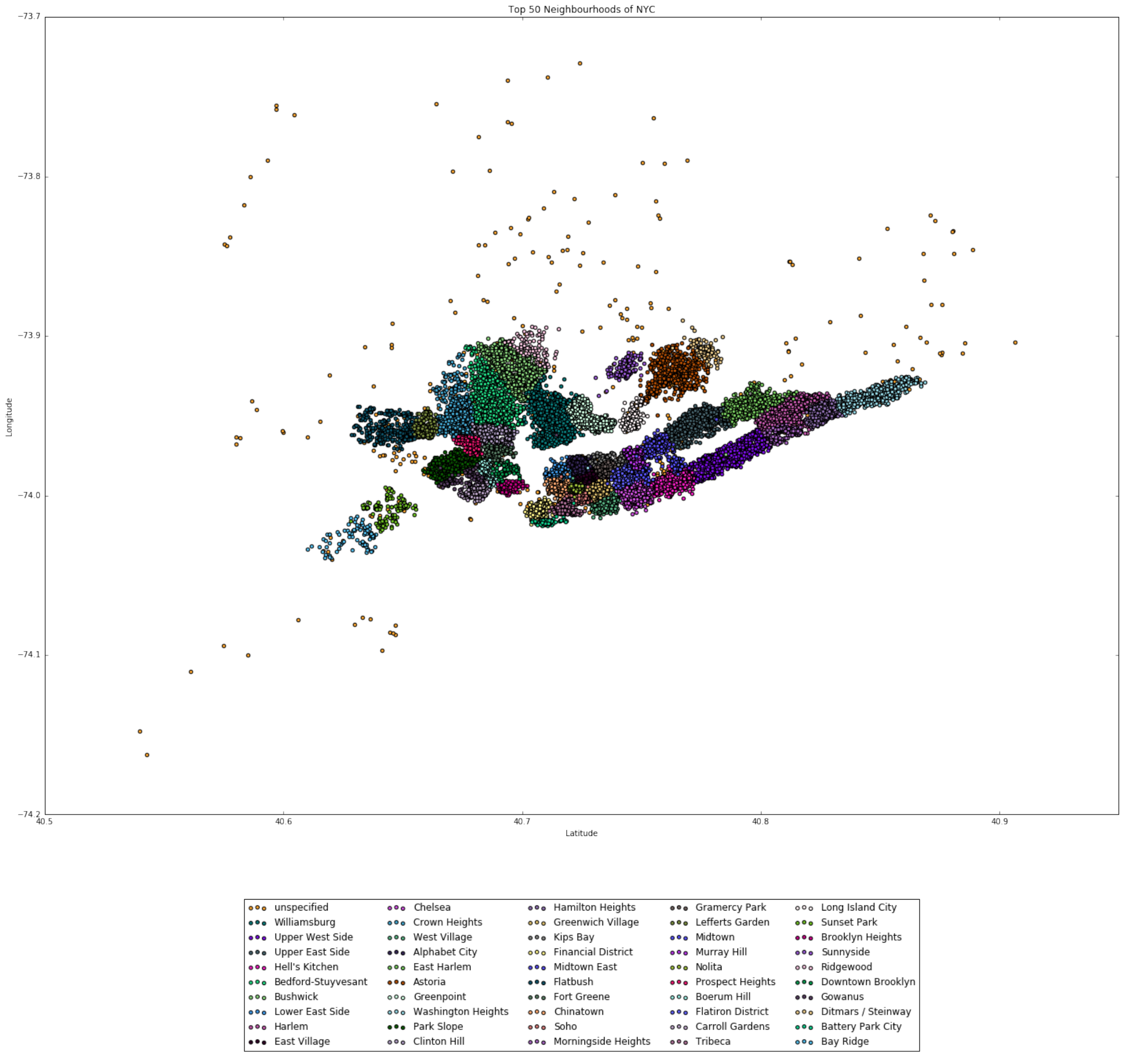
```
In [24]: # plot 2D neighborhood to visualize supply of rental homes in each area wi
         def plot_neighborhoods_by_price_2D():
             fig = plt.figure(figsize=(20, 15))
```

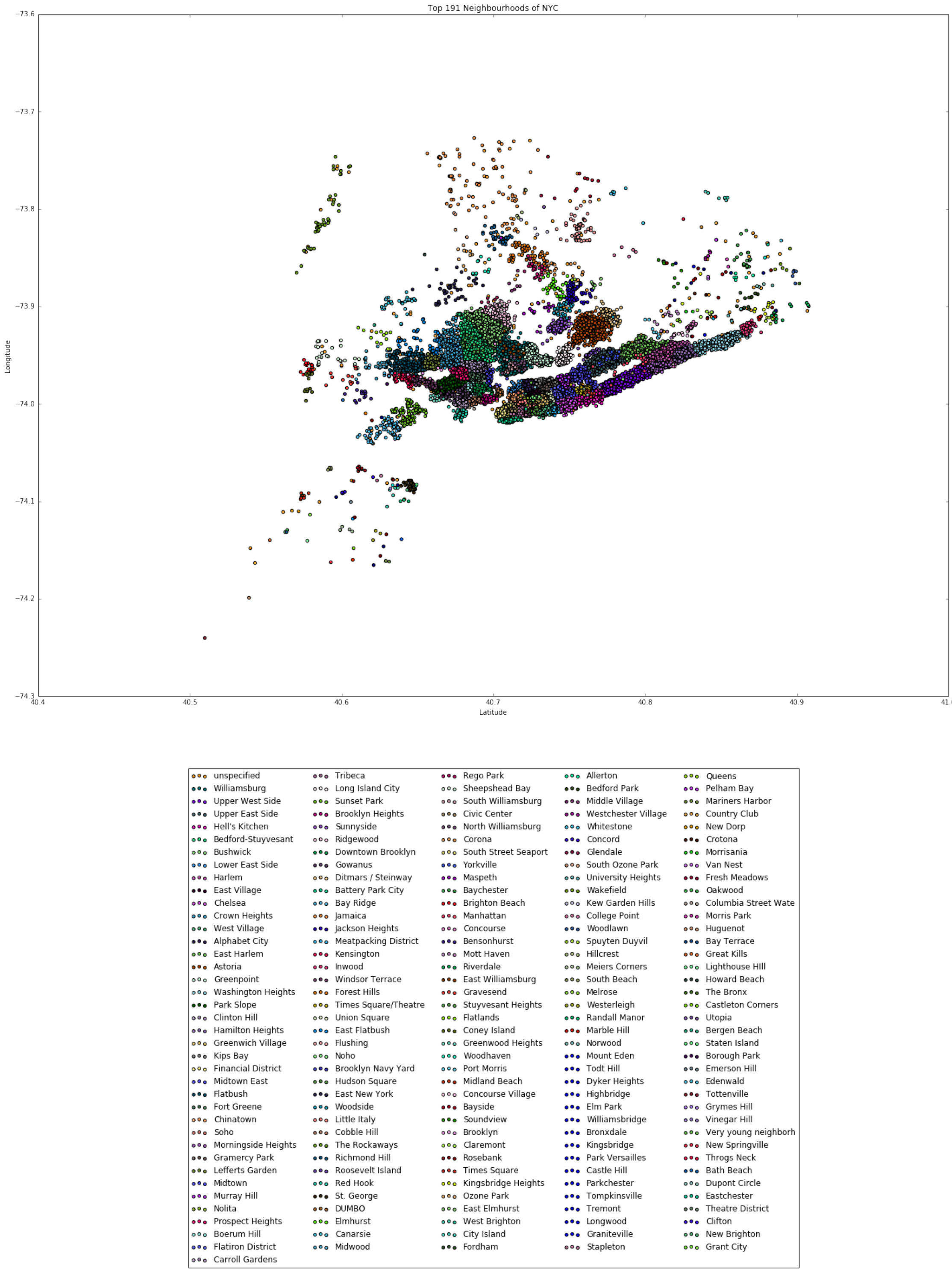
```
        num_class = len(top_neighborhood)

        ax = fig.add_subplot(1, 1, 1)
        ax.scatter(listings_df['latitude'], listings_df['longitude'], c=rgb_gr

        ax.set_xlabel('Latitude')
        ax.set_ylabel('Longitude')
        ax.set_title('Price gradient for NYC')

        import matplotlib.patches as mpatches
        classes = ['Less expensive', 'More expensive']
        class_colours = ['g', 'r']
        recs = []
        for i in range(0,len(class_colours)):
            recs.append(mpatches.Rectangle((0,0),1,1,fc=class_colours[i]))
        plt.legend(recs,classes,loc=4)

        plt.tight_layout()
        plt.show()

    # plot 3D neighborhood to visualize supply of rental homes in each area wi
    def plot_neighborhoods_by_price_3D():
        fig = plt.figure(figsize=(20, 15))

        num_class = len(top_neighborhood)

        ax = fig.add_subplot(1, 1, 1, projection='3d')
        ax.scatter(listings_df['latitude'], listings_df['longitude'], listings

        ax.set_xlabel('Latitude')
        ax.set_ylabel('Longitude')
        ax.set_zlabel('Nightly Rental Price')
        ax.set_title('Price gradient for NYC')

        import matplotlib.patches as mpatches
        classes = ['Less expensive', 'More expensive']
        class_colours = ['g', 'r']
        recs = []
        for i in range(0,len(class_colours)):
            recs.append(mpatches.Rectangle((0,0),1,1,fc=class_colours[i]))
        plt.legend(recs,classes,loc=4)

        plt.tight_layout()
        plt.show()

    # plot neighborhood to visualize supply of rental homes in each area with
    def plot_neighborhood_locations(num_neighborhoods, colors):
        fig = plt.figure(figsize=(20, 15))
```

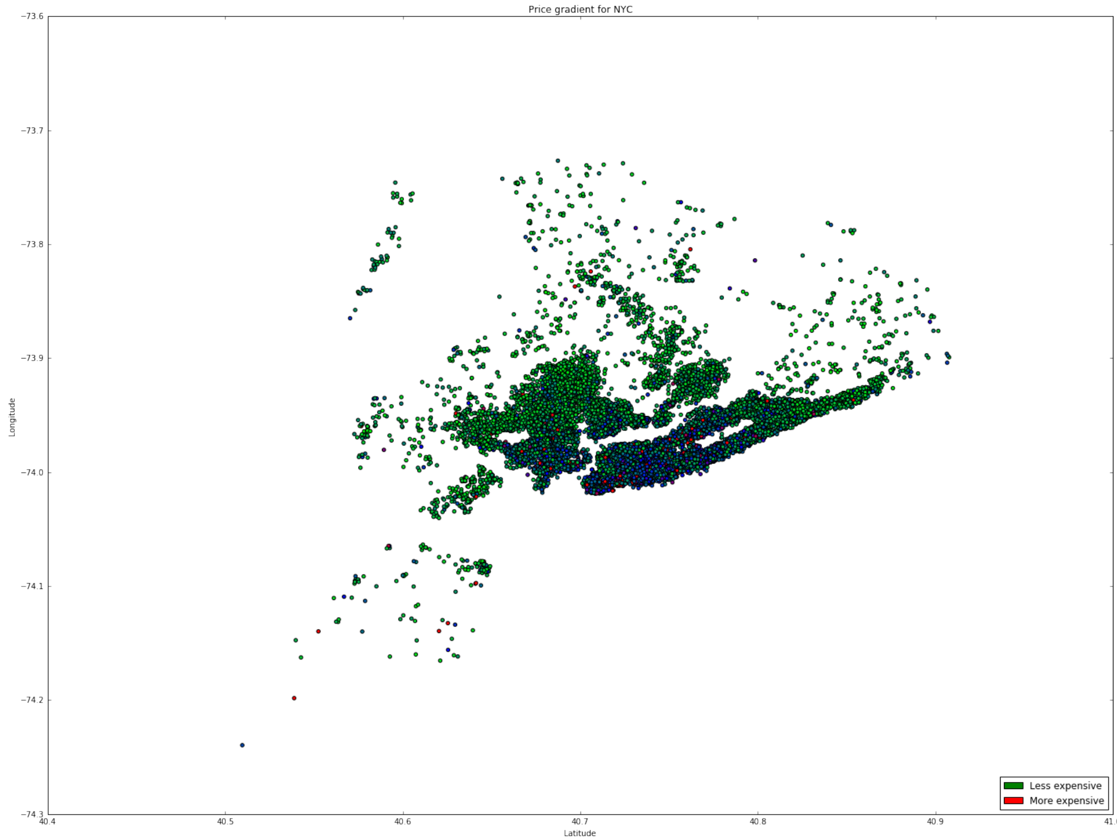
```
    i = 0
    num_class = len(top_neighborhood)
    for (neighborhood, num) in top_neighborhood[0:num_neighborhoods]:
        # adjust length of the printed neighborhood
        if len(neighborhood)>20:
            legend_label = neighborhood[0:20]
        else:
            legend_label = neighborhood

        ax = fig.add_subplot(1, 1, 1)
        ax.scatter(listings_df[listings_df['neighbourhood'] == str(neighbo
                   listings_df[listings_df['neighbourhood'] == str(neighbo
                   c=colors[i], label=legend_label)
        i = i + 1

    ax.set_xlabel('Latitude')
    ax.set_ylabel('Longitude')
    ax.set_title('Top ' + str(num_neighborhoods) + ' Neighbourhoods of NYC
    ax.legend(loc=9, bbox_to_anchor=(0.5, -0.1), ncol=5)
    plt.tight_layout()
    plt.show()

# for calculating the rgb tuble for the given gradient; red = expensive, g
def rgb(minimum, maximum, value):
    minimum, maximum = float(minimum), float(maximum)
    ratio = 2 * (value-minimum) / (maximum - minimum)
    r = int(max(0, 255*(ratio - 1)))
    g = int(max(0, 255*(1 - ratio)))
    b = 255 - g - r
    return max(min(r/255., 1.), 0), max(min(g/255., 1.), 0), max(min(b/255

# build color matrix for coloring-by-neighborhood
import random
random.seed(1)
colors=np.random.random((191, 3))

# build color gradient by price (green -> red)
price = listings_df['price']
price_min = np.min(price_out)
price_max = np.max(price_out)
rgb_grad = np.array([rgb(price_min, price_max, x) for x in price])

# plot top 10 most populous neighborhoods in NYC, coloring by neighborhoo
plot_neighborhood_locations(10, colors)
# plot top 50 most populous neighborhoods in NYC, coloring by neighborhoo
plot_neighborhood_locations(50, colors)
# plot all neighborhoods in NYC, coloring by neighborhood
```

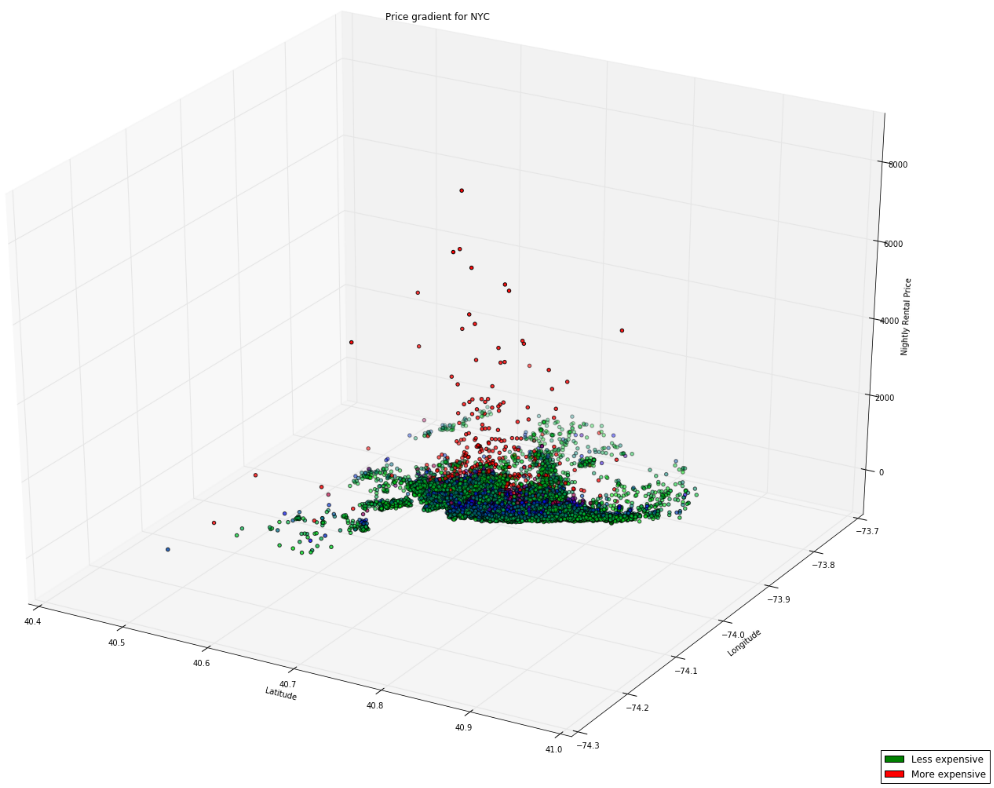
```
plot_neighborhood_locations(191, colors)

# plot all homes with color gradient (green -> red) denoting price
plot_neighborhoods_by_price_2D()
# plot all homes with color gradient (green -> red denoting price) with z-
plot_neighborhoods_by_price_3D()
```




Top 50 Neighbourhoods of NYC
Longitude
Latitude
−73.7
−73.8
−73.9
−74.0
−74.1
−74.2
40.5
40.6
40.7
40.8
40.9
unspecified
Williamsburg
Upper West Side
Upper East Side
Hell's Kitchen
Bedford-Stuyvesant
Bushwick
Lower East Side
Harlem
East Village
Chelsea
Crown Heights
West Village
Alphabet City
East Harlem
Astoria
Greenpoint
Washington Heights
Park Slope
Clinton Hill
Hamilton Heights
Greenwich Village
Kips Bay
Financial District
Midtown East
Flatbush
Fort Greene
Chinatown
Soho
Morningside Heights
Gramercy Park
Lefferts Garden
Midtown
Murray Hill
Nolita
Prospect Heights
Boerum Hill
Flatiron District
Carroll Gardens
Tribeca
Long Island City
Sunset Park
Brooklyn Heights
Sunnyside
Ridgewood
Downtown Brooklyn
Gowanus
Ditmars / Steinway
Battery Park City
Bay Ridge


Top 191 Neighbourhoods of NYC
Longitude
Latitude
−73.6
−73.7
−73.8
−73.9
−74.0
−74.1
−74.2
−74.3
40.4
40.5
40.6
40.7
40.8
40.9
41.0
unspecified
Williamsburg
Upper West Side
Upper East Side
Hell's Kitchen
Bedford-Stuyvesant
Bushwick
Lower East Side
Harlem
East Village
Chelsea
Crown Heights
West Village
Alphabet City
East Harlem
Astoria
Greenpoint
Washington Heights
Park Slope
Clinton Hill
Hamilton Heights
Greenwich Village
Kips Bay
Financial District
Midtown East
Flatbush
Fort Greene
Chinatown
Soho
Morningside Heights
Gramercy Park
Lefferts Garden
Midtown
Murray Hill
Nolita
Prospect Heights
Boerum Hill
Flatiron District
Carroll Gardens
Tribeca
Long Island City
Sunset Park
Brooklyn Heights
Sunnyside
Ridgewood
Downtown Brooklyn
Gowanus
Ditmars / Steinway
Battery Park City
Bay Ridge
Jamaica
Jackson Heights
Meatpacking District
Kensington
Inwood
Windsor Terrace
Forest Hills
Times Square/Theatre
Union Square
East Flatbush
Flushing
Noho
Brooklyn Navy Yard
Hudson Square
East New York
Woodside
Little Italy
Cobble Hill
The Rockaways
Richmond Hill
Roosevelt Island
Red Hook
St. George
DUMBO
Elmhurst
Canarsie
Midwood
Rego Park
Sheepshead Bay
South Williamsburg
Civic Center
North Williamsburg
Corona
South Street Seaport
Yorkville
Maspeth
Baychester
Brighton Beach
Manhattan
Concourse
Bensonhurst
Mott Haven
Riverdale
East Williamsburg
Gravesend
Stuyvesant Heights
Flatlands
Coney Island
Greenwood Heights
Woodhaven
Port Morris
Midland Beach
Concourse Village
Bayside
Soundview
Brooklyn
Claremont
Rosebank
Times Square
Kingsbridge Heights
Ozone Park
East Elmhurst
West Brighton
City Island
Fordham
Allerton
Bedford Park
Middle Village
Westchester Village
Whitestone
Concord
Glendale
South Ozone Park
University Heights
Wakefield
Kew Garden Hills
College Point
Woodlawn
Spuyten Duyvil
Hillcrest
Meiers Corners
South Beach
Melrose
Westerleigh
Randall Manor
Marble Hill
Norwood
Mount Eden
Todt Hill
Dyker Heights
Highbridge
Elm Park
Williamsbridge
Bronxdale
Kingsbridge
Park Versailles
Castle Hill
Parkchester
Tompkinsville
Tremont
Longwood
Graniteville
Stapleton
Queens
Pelham Bay
Mariners Harbor
Country Club
New Dorp
Crotona
Morrisania
Van Nest
Fresh Meadows
Oakwood
Columbia Street Wate
Morris Park
Huguenot
Bay Terrace
Great Kills
Lighthouse HIll
Howard Beach
The Bronx
Castleton Corners
Utopia
Bergen Beach
Staten Island
Borough Park
Emerson Hill
Edenwald
Tottenville
Grymes Hill
Vinegar Hill
Very young neighborh
New Springville
Throgs Neck
Bath Beach
Dupont Circle
Eastchester
Theatre District
Clifton
New Brighton
Grant City


Price gradient for NYC
Longitude
Latitude
−73.6
−73.7
−73.8
−73.9
−74.0
−74.1
−74.2
−74.3
40.4
40.5
40.6
40.7
40.8
40.9
41.0
Less expensive
More expensive



In [25]:
```
# build numpy array of average home prices in each region
neighborhood_prices = top_neighborhood
i = 0
for (neighborhood, num) in top_neighborhood:
    neighborhood_prices[i] = [neighborhood, np.mean(np.array(listings_df[l
    i += 1

neighborhood_prices
```

Out[25]:
```
[['unspecified', 160.10557474099656],
 ['Williamsburg', 141.77103301384452],
 ['Upper West Side', 219.63045141545524],
 ['Upper East Side', 202.34319526627218],
 ["Hell's Kitchen", 221.71282922684793],
 ['Bedford-Stuyvesant', 108.0341726618705],
 ['Bushwick', 85.854349951124149],
 ['Lower East Side', 185.75902335456476],
 ['Harlem', 124.31126596980255],
 ['East Village', 197.63895486935866],
 ['Chelsea', 253.42839951865221],
 ['Crown Heights', 107.24738219895288],
```

```
['West Village', 265.93038821954485],
['Alphabet City', 172.76512968299713],
['East Harlem', 131.85128205128206],
['Astoria', 107.58752166377816],
['Greenpoint', 141.02631578947367],
['Washington Heights', 110.08128544423441],
['Park Slope', 153.54684512428298],
['Clinton Hill', 185.64932562620425],
['Hamilton Heights', 105.3507972665148],
['Greenwich Village', 240.1846153846154],
['Kips Bay', 249.80462724935734],
['Financial District', 236.36559139784947],
['Midtown East', 248.81440443213296],
['Flatbush', 106.60778443113773],
['Fort Greene', 153.97719869706842],
['Chinatown', 168.2051282051282],
['Soho', 310.12867647058823],
['Morningside Heights', 133.45275590551182],
['Gramercy Park', 217.1844262295082],
['Lefferts Garden', 99.607594936708864],
['Midtown', 219.84848484848484],
['Murray Hill', 295.7837837837838],
['Nolita', 240.69585253456222],
['Prospect Heights', 129.48356807511738],
['Boerum Hill', 187.18652849740931],
['Flatiron District', 309.30337078651684],
['Carroll Gardens', 181.46428571428572],
['Tribeca', 397.63636363636363],
['Long Island City', 159.57664233576642],
['Sunset Park', 77.624060150375939],
['Brooklyn Heights', 205.5078125],
['Sunnyside', 89.960629921259837],
['Ridgewood', 82.13274336283186],
['Downtown Brooklyn', 190.3362831858407],
['Gowanus', 177.42574257425741],
['Ditmars / Steinway', 93.291666666666671],
['Battery Park City', 348.58241758241758],
['Bay Ridge', 83.670588235294119],
['Jamaica', 75.776470588235298],
['Jackson Heights', 92.444444444444443],
['Meatpacking District', 288.64383561643837],
['Kensington', 102.38888888888889],
['Inwood', 86.588235294117652],
['Windsor Terrace', 139.93548387096774],
['Forest Hills', 100.18032786885246],
['Times Square/Theatre District', 232.49180327868854],
['Union Square', 378.06666666666666],
['East Flatbush', 97.775862068965523],
```

```
['Flushing', 98.98181818181818184],
['Noho', 295.33962264150944],
['Brooklyn Navy Yard', 124.72340425531915],
['Hudson Square', 248.06382978723406],
['East New York', 72.765957446808514],
['Woodside', 75.276595744680847],
['Little Italy', 409.52173913043481],
['Cobble Hill', 164.48837209302326],
['The Rockaways', 114.74418604651163],
['Richmond Hill', 170.65853658536585],
['Roosevelt Island', 124.33333333333333],
['Red Hook', 150.84210526315789],
['St. George', 141.27027027027026],
['DUMBO', 231.20588235294119],
['Elmhurst', 69.090909090909093],
['Canarsie', 82.454545454545453],
['Midwood', 232.875],
['Rego Park', 89.400000000000006],
['Sheepshead Bay', 103.85714285714286],
['South Williamsburg', 147.96428571428572],
['Civic Center', 198.03999999999999],
['North Williamsburg', 152.13636363636363],
['Corona', 76.666666666666671],
['South Street Seaport', 217.42857142857142],
['Yorkville', 152.36842105263159],
['Maspeth', 72.89473684210526],
['Baychester', 62.944444444444443],
['Brighton Beach', 118.70588235294117],
['Manhattan', 127.94117647058823],
['Concourse', 90.86666666666666],
['Bensonhurst', 102.07142857142857],
['Mott Haven', 66.642857142857139],
['Riverdale', 115.07142857142857],
['East Williamsburg', 101.0],
['Gravesend', 104.53846153846153],
['Stuyvesant Heights', 162.46153846153845],
['Flatlands', 84.07692307692308],
['Coney Island', 123.61538461538461],
['Greenwood Heights', 123.30769230769231],
['Woodhaven', 92.0],
['Port Morris', 106.3],
['Midland Beach', 85.400000000000006],
['Concourse Village', 78.799999999999997],
['Bayside', 121.0],
['Soundview', 51.222222222222221],
['Brooklyn', 123.11111111111111],
['Claremont', 52.625],
['Rosebank', 68.875],
```

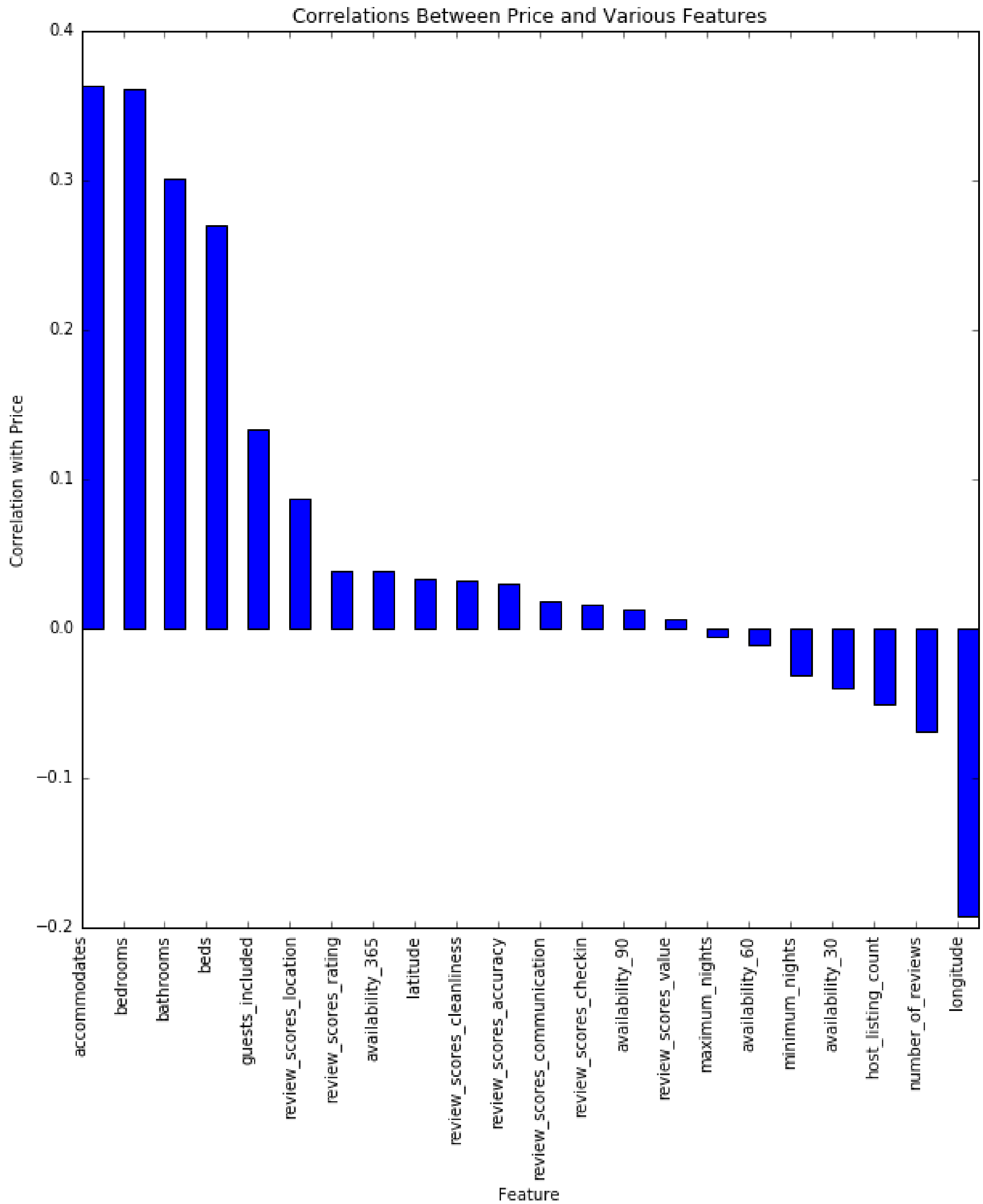
```
['Times Square', 186.875],
['Kingsbridge Heights', 89.375],
['Ozone Park', 53.571428571428569],
['East Elmhurst', 72.0],
['West Brighton', 54.571428571428569],
['City Island', 96.714285714285708],
['Fordham', 79.428571428571431],
['Allerton', 54.5],
['Bedford Park', 75.333333333333329],
['Middle Village', 122.33333333333333],
['Westchester Village', 74.666666666666671],
['Whitestone', 140.5],
['Concord', 96.333333333333329],
['Glendale', 92.166666666666671],
['South Ozone Park', 124.8],
['University Heights', 72.0],
['Wakefield', 99.0],
['Kew Garden Hills', 131.59999999999999],
['College Point', 127.2],
['Woodlawn', 140.59999999999999],
['Spuyten Duyvil', 106.8],
['Hillcrest', 111.8],
['Meiers Corners', 89.0],
['South Beach', 243.75],
['Melrose', 60.75],
['Westerleigh', 608.75],
['Randall Manor', 254.25],
['Marble Hill', 97.5],
['Norwood', 86.0],
['Mount Eden', 46.666666666666664],
['Todt Hill', 58.333333333333336],
['Dyker Heights', 67.333333333333329],
['Highbridge', 65.0],
['Elm Park', 126.66666666666667],
['Williamsbridge', 81.333333333333329],
['Bronxdale', 78.333333333333329],
['Kingsbridge', 75.0],
['Park Versailles', 56.666666666666664],
['Castle Hill ', 65.0],
['Parkchester', 110.0],
['Tompkinsville', 88.0],
['Tremont', 66.333333333333329],
['Longwood', 123.33333333333333],
['Graniteville', 126.66666666666667],
['Stapleton', 65.5],
['Queens', 86.5],
['Pelham Bay', 64.5],
['Mariners Harbor', 90.0],
```

```
 ['Country Club', 119.5],
 ['New Dorp', 189.5],
 ['Crotona', 65.5],
 ['Morrisania', 116.0],
 ['Van Nest', 50.0],
 ['Fresh Meadows', 250.0],
 ['Oakwood', 79.0],
 ['Columbia Street Waterfront', 155.0],
 ['Morris Park', 69.0],
 ['Huguenot', 700.0],
 ['Bay Terrace', 59.0],
 ['Great Kills', 1500.0],
 ['Lighthouse HIll', 175.0],
 ['Howard Beach', 119.0],
 ['The Bronx', 65.0],
 ['Castleton Corners', 100.0],
 ['Utopia', 125.0],
 ['Bergen Beach', 100.0],
 ['Staten Island', 59.0],
 ['Borough Park', 60.0],
 ['Emerson Hill', 100.0],
 ['Edenwald', 135.0],
 ['Tottenville', 219.0],
 ['Grymes Hill', 86.0],
 ['Vinegar Hill', 120.0],
 ['Very young neighborhood yet a bit removed from the drunken craziness sc
  115.0],
 ['New Springville', 58.0],
 ['Throgs Neck', 120.0],
 ['Bath Beach', 49.0],
 ['Dupont Circle', 250.0],
 ['Eastchester', 80.0],
 ['Theatre District', 115.0],
 ['Clifton', 65.0],
 ['New Brighton', 95.0],
 ['Grant City', 200.0]]
```

1.4.3 Correlations between price and features of home

In [26]:
```
# get correlations on price
corrs_dict = (listings_df.corr())['price'].to_dict()

# delete irrelevant features
for var in ['price', 'weekly_price', 'monthly_price', 'scrape_id', 'host_i
    del corrs_dict[var]

# sort by correlation value
price_corrs = sorted(corrs_dict.items(), key=operator.itemgetter(1), rever
```

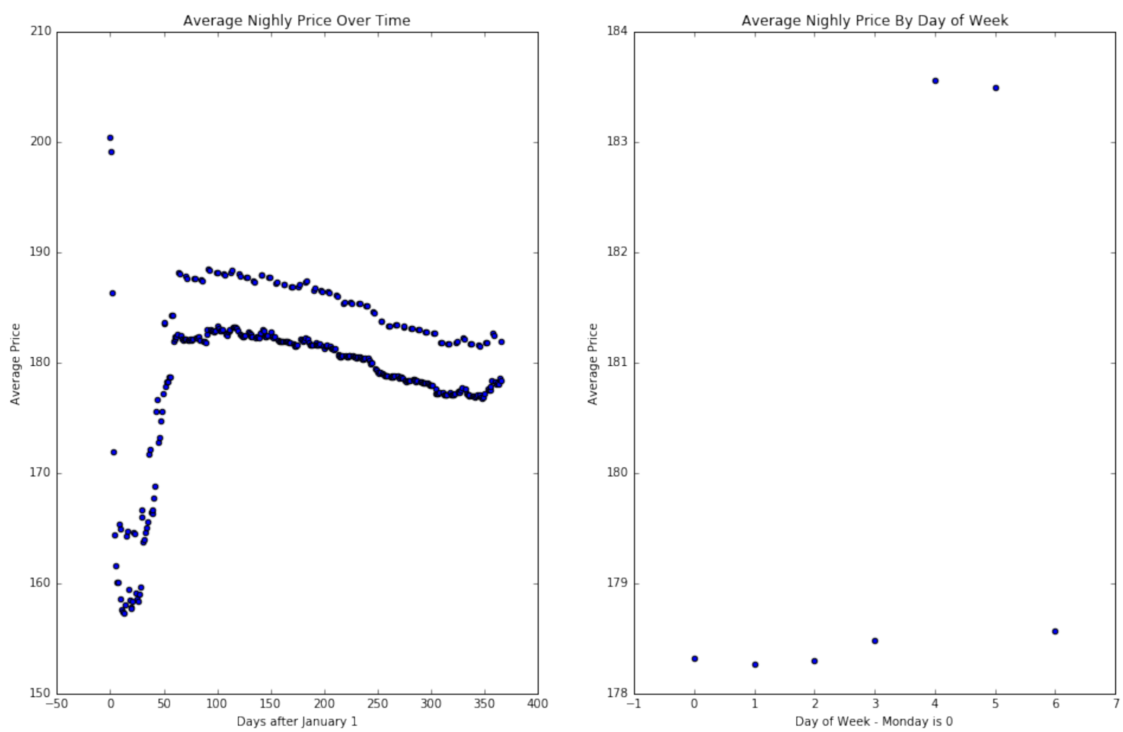
```
# break into two lists
features, corrs = zip(*price_corrs)

# plot correlations
fig, ax = plt.subplots(1, figsize=(10, 10))
ax.set_ylabel('Correlation with Price')
ax.set_xlabel('Feature')
ax.set_title('Correlations Between Price and Various Features')
plt.xticks(range(len(corrs)), features, rotation='vertical')
ax.bar(range(len(corrs)), corrs, .5, color="blue")
plt.show()
```



1.4.4 Relationship between price and time

```
In [27]: # method to convert date to day of week
         def get_day(date):
             return datetime.datetime.strptime(date, '%Y-%m-%d').strftime('%A')
```

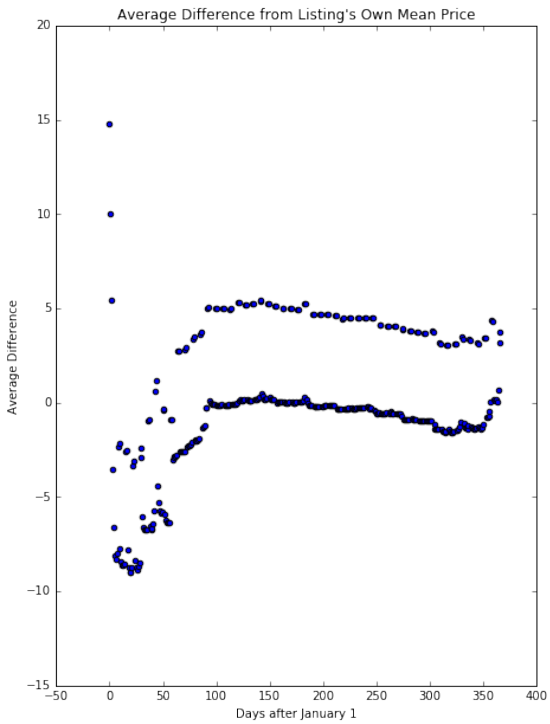
```
In [28]: date_prices = {date: 0 for date in calendar_df['date'].unique()}
         day_prices = {get_day(date): 0 for date in calendar_df['date'].unique()}
         day_counts = {get_day(date): 0 for date in calendar_df['date'].unique()}

         for date in calendar_df['date'].unique():
             prices = [p for p in np.array(calendar_df[calendar_df['date'] == date]
             date_prices[date] = np.mean(prices)
             day_prices[get_day(date)] += np.sum(prices)
             day_counts[get_day(date)] += len(prices)

In [29]: for day, price_sum in day_prices.iteritems():
             day_prices[day] = float(day_prices[day]) / float(day_counts[day])

In [30]: fig, ax = plt.subplots(1, 2, figsize=(16, 10))
         ordered_date_prices = [value for key, value in sorted(date_prices.items())
         ax[0].scatter(range(367), ordered_date_prices)
         ax[1].scatter(range(7), [day_prices['Monday'], day_prices['Tuesday'], day_

         # Label axes, set title
         ax[0].set_title('Average Nighly Price Over Time')
         ax[0].set_xlabel('Days after January 1')
         ax[0].set_ylabel('Average Price')

         ax[1].set_title('Average Nighly Price By Day of Week')
         ax[1].set_xlabel('Day of Week - Monday is 0')
         ax[1].set_ylabel('Average Price')

         plt.show()
```

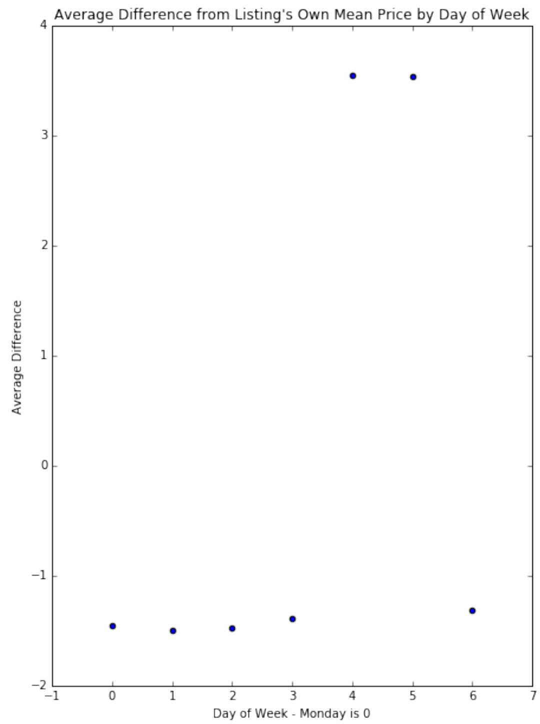


In [31]:
```
# dictionary to contain means
listing_means = {listing: 0 for listing in calendar_df['listing_id'].uniqu

# calculate mean prices for each listing
for listing in calendar_df['listing_id'].unique():
    listing_means[listing] = np.mean(calendar_df[calendar_df['listing_id']

# convert to numpy for efficiency
prices = np.array(calendar_df['price'])
ids = np.array(calendar_df['listing_id'])
diff_means = []
for i in range(len(prices)):
    diff_means.append(prices[i] - listing_means[ids[i]])

# add to calendar df
calendar_df['diff_mean'] = pd.Series(np.array(diff_means), index=calendar_
```

In [32]:
```
# calculate average difference from means over all listings
avg_diff_means = {date: 0 for date in calendar_df['date'].unique()}
for date, mean in avg_diff_means.iteritems():
    avg_diff_means[date] = np.mean(calendar_df[calendar_df['date'] == date

# calculate average difference from means by day of week
# convert to numpy for speed
dates = np.array(calendar_df['date'])
day_avg_diff_means = {get_day(date): 0 for date in calendar_df['date'].uni
for i in range(len(prices)):
    day_avg_diff_means[get_day(dates[i])] += diff_means[i]
for day, avg_diff in day_avg_diff_means.iteritems():
    day_avg_diff_means[day] = float(day_avg_diff_means[day]) / float(day_a
```

In [33]:
```
fig, ax = plt.subplots(1, 2, figsize=(16, 10))
ordered_avg_diff_means = [value for key, value in sorted(avg_diff_means.it
ax[0].scatter(range(367), ordered_avg_diff_means)
ax[1].scatter(range(7), [day_avg_diff_means['Monday'], day_avg_diff_means[

# Label axes, set title
ax[0].set_title('Average Difference from Listing\'s Own Mean Price')
ax[0].set_xlabel('Days after January 1')
ax[0].set_ylabel('Average Difference')

ax[1].set_title('Average Difference from Listing\'s Own Mean Price by Day
ax[1].set_xlabel('Day of Week - Monday is 0')
ax[1].set_ylabel('Average Difference')

plt.show()
```






```
In [34]: # examine priciest and cheapest dates
         sorted_avg_diffs = sorted(avg_diff_means.items(), key=lambda x: x[1])
         print 'Dates with lowest average differences from mean:'
         for date, diff in sorted_avg_diffs[:15]:
             print date + ', ' + get_day(date) + ': $' + str(round(diff, 2))

         print '\nDates with highest average differences from mean:'
         for date, diff in sorted_avg_diffs[-15:]:
             print date + ', ' + get_day(date) + ': $' + str(round(diff, 2))
```

```
Dates with lowest average differences from mean:
2015-01-21, Wednesday: $-9.01
2015-01-20, Tuesday: $-8.99
2015-01-27, Tuesday: $-8.85
2015-01-22, Thursday: $-8.76
2015-01-26, Monday: $-8.76
2015-01-19, Monday: $-8.71
2015-01-28, Wednesday: $-8.68
2015-01-14, Wednesday: $-8.6
2015-01-13, Tuesday: $-8.59
2015-01-15, Thursday: $-8.55
2015-01-29, Thursday: $-8.48
2015-01-12, Monday: $-8.44
2015-01-25, Sunday: $-8.35
```

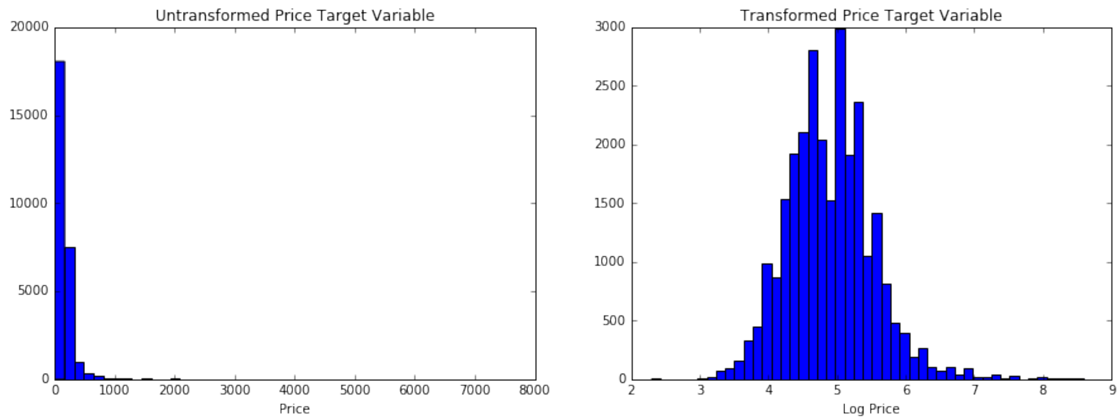
```
2015-01-07, Wednesday: $-8.3
2015-01-06, Tuesday: $-8.13

Dates with highest average differences from mean:
2015-05-08, Friday: $5.18
2015-05-09, Saturday: $5.19
2015-05-15, Friday: $5.22
2015-07-03, Friday: $5.23
2015-05-16, Saturday: $5.24
2015-05-29, Friday: $5.24
2015-05-30, Saturday: $5.24
2015-07-04, Saturday: $5.25
2015-05-02, Saturday: $5.29
2015-05-01, Friday: $5.29
2015-05-22, Friday: $5.39
2015-05-23, Saturday: $5.42
2015-01-03, Saturday: $5.44
2015-01-02, Friday: $10.0
2015-01-01, Thursday: $14.76
```

1.4.5 Visualize the target variable to understand skewness and identify transformation that might be necessary

The target variable “price” was analyzed to investigate the the distribution of the data. The histogram below shows that the variable is skewed right. This indicates that there are large outliers in price and that the target variable would need to be transformed to create a normal distribution that would be beneficial for predictive modeling.

A log transformation was conducted on the target variable “price”. This transformation resulted in a normal distribution that would be ideal for predictive modeling.

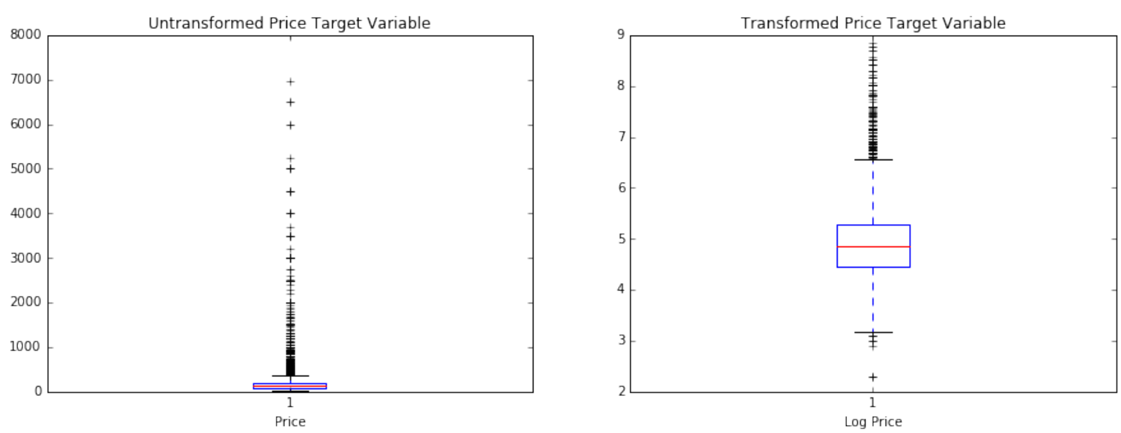
```
In [35]: listings_df['price_log'] = np.log(listings_df['price'])

         fig, ax = plt.subplots(1, 2, figsize=(15, 5))
         ax[0].hist(listings_df['price'],bins=50)
         ax[0].set_title('Untransformed Price Target Variable')
         ax[0].set_xlabel('Price')

         ax[1].hist(listings_df['price_log'],bins=50)
         ax[1].set_title('Transformed Price Target Variable')
         ax[1].set_xlabel('Log Price')

Out[35]: <matplotlib.text.Text at 0x13c1defd0>
```



As a second level of analysis, the price variable was visulaized by a boxplot. This visualization confirms the extreme outliers for price and the need to transform the target variable.

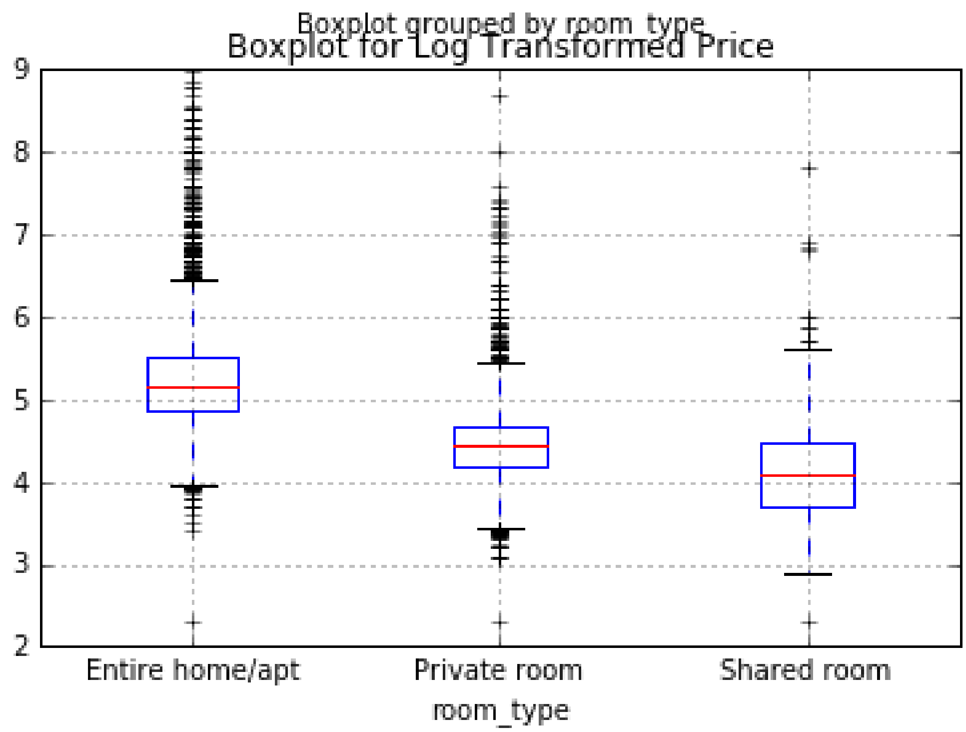
The log transformed "price" variable was analyzed further by a boxplot to confirm that the outlier prices were now within a reasonable range.

```
In [36]: fig, ax = plt.subplots(1, 2, figsize=(15, 5))
         ax[0].boxplot(listings_df['price'])
         ax[0].set_title('Untransformed Price Target Variable')
         ax[0].set_xlabel('Price')

         ax[1].boxplot(listings_df['price_log'])
         ax[1].set_title('Transformed Price Target Variable')
         ax[1].set_xlabel('Log Price')

Out[36]: <matplotlib.text.Text at 0x14c51ed10>
```

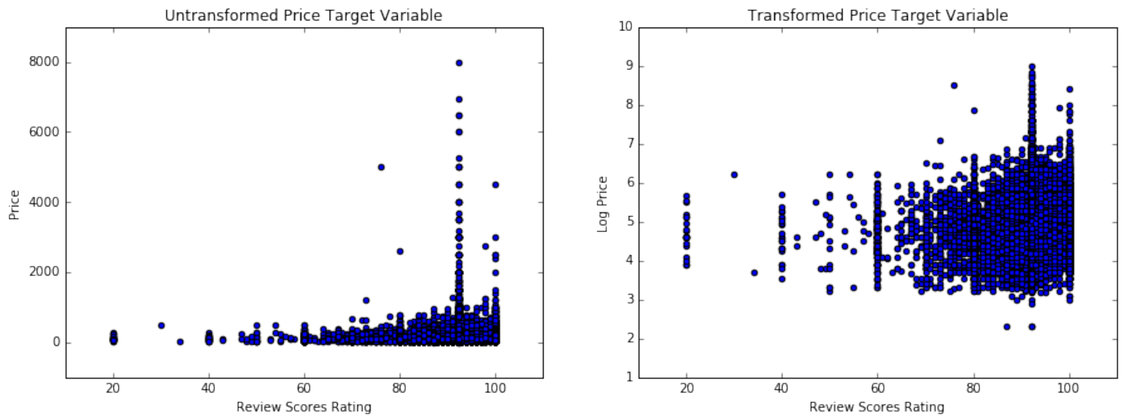



```
In [37]: listings_df.boxplot(column='price_log', by ='room_type')
         plt.title('Boxplot for Log Transformed Price')
```

```
Out[37]: <matplotlib.text.Text at 0x1692532d0>
```

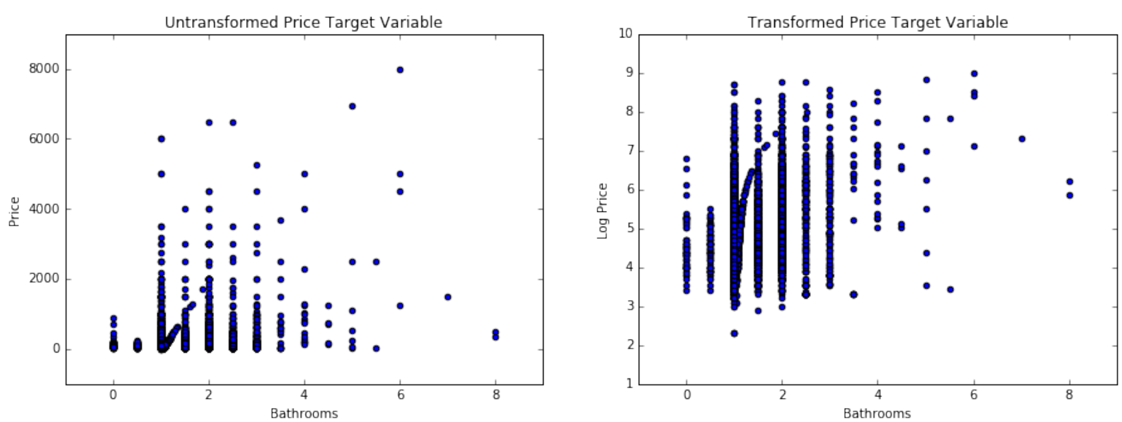



```
In [38]: fig, ax = plt.subplots(1, 2, figsize=(15, 5))
         ax[0].scatter(listings_df['review_scores_rating'], listings_df['price'])
         ax[0].set_title('Untransformed Price Target Variable')
         ax[0].set_xlabel('Review Scores Rating')
         ax[0].set_ylabel('Price')
         ax[1].scatter(listings_df['review_scores_rating'], listings_df['price_log'
         ax[1].set_title('Transformed Price Target Variable')
         ax[1].set_ylabel('Log Price')
         ax[1].set_xlabel('Review Scores Rating')

Out[38]: <matplotlib.text.Text at 0x169c7ad90>
```




```
In [39]: fig, ax = plt.subplots(1, 2, figsize=(15, 5))
         ax[0].scatter(listings_df['bathrooms'], listings_df['price'])
         ax[0].set_title('Untransformed Price Target Variable')
         ax[0].set_xlabel('Bathrooms')
         ax[0].set_ylabel('Price')
         ax[1].scatter(listings_df['bathrooms'], listings_df['price_log'])
         ax[1].set_title('Transformed Price Target Variable')
         ax[1].set_ylabel('Log Price')
         ax[1].set_xlabel('Bathrooms')
```

```
Out[39]: <matplotlib.text.Text at 0x13c81efd0>
```




```
In [40]: fig, ax = plt.subplots(1, 2, figsize=(15, 5))
         ax[0].scatter(listings_df['bedrooms'], listings_df['price'])
         ax[0].set_title('Untransformed Price Target Variable')
         ax[0].set_xlabel('Bedrooms')
         ax[0].set_ylabel('Price')
```

```
ax[1].scatter(listings_df['bedrooms'], listings_df['price_log'])
ax[1].set_title('Transformed Price Target Variable')
ax[1].set_ylabel('Log Price')
ax[1].set_xlabel('Bedrooms')
```

```
Out[40]: <matplotlib.text.Text at 0x13cf62650>
```