case_4.3

April 22, 2020

1 What features determine the price of an Airbnb rental?

[1]:
```
import numpy                 as np
import pandas                as pd
import matplotlib.pyplot     as plt
import seaborn               as sns
import sklearn.metrics       as Metrics
import pandas                as pd
import matplotlib.pyplot     as plt


import folium  #needed for interactive map
from folium.plugins import HeatMap

from   collections           import Counter
from   sklearn               import preprocessing
from   datetime              import datetime
from   collections           import Counter
from   math                  import exp
from   sklearn.linear_model  import LinearRegression as LinReg
from   sklearn.metrics       import mean_absolute_error
from   sklearn.metrics       import median_absolute_error
from   sklearn.metrics       import r2_score

%matplotlib inline
sns.set()
```

1.1 Introduction (10 mts)

Business Context. Airbnb is an enormous online marketplace for everyday people to rent places to stay. It is a large and lucrative market, but many vendors are simply individuals who are renting their own primary residence for short visits. Even larger vendors are typically small businesses with only a small number of places to rent. As a result, they have limited ability to assess large-scale trends and set optimal prices.

Airbnb has rolled out a new service to help listers set prices. Airbnb makes a percentage commission off of the listings, so they are incentivized to help listers price optimally; that is, at the maximum

possible point where they will still close a deal. You are an Airbnb consultant helping with this new pricing service.

Business Problem. Your initial task is to explore the data with the goal of answering the question: "What features are most relevant to the price of an Airbnb listing?"

Analytical Context. We will use the publicly available and well-maintained dataset created by the Inside Airbnb advocacy group. We will focus on listings in New York City within the last year, taking advantage of larger datasets when there are important details to explore.

The case is structured as follows: we will (1) do basic data exploration by plotting distributions of key quantities; (2) introduce the concept of correlation to find the key features; (3) introduce the idea of interaction effects to correct for the effects of key features; (4) discuss how to iteratively generate hypotheses and choose data visualizations to support your conclusions; (5) look at one very specific type of interaction effect, the temporal effect, and how to correct for it; and finally (6) pull everything together to identify the key factors that affect the price.

1.2 Some basic data exploration (20 mts)

We begin by loading the data and looking at its basic shape:

[2]:
```
listings = pd.read_csv('airbnb_NYC.csv', delimiter=',')
listings.shape
```

[2]:
```
(30179, 81)
```

Let's also look at the columns of the dataset:

[3]:
```
listings.columns
```

[3]:
```
Index(['id', 'name', 'summary', 'description', 'experiences_offered',
       'neighborhood_overview', 'transit', 'house_rules', 'host_id',
       'host_since', 'host_response_time', 'host_response_rate',
       'host_is_superhost', 'host_listings_count', 'host_identity_verified',
       'street', 'neighbourhood', 'latitude', 'longitude', 'property_type',
       'room_type', 'accommodates', 'bathrooms', 'bedrooms', 'beds',
       'bed_type', 'amenities', 'price', 'guests_included', 'extra_people',
       'minimum_nights', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'number_of_reviews', 'number_of_reviews_ltm',
       'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'instant_bookable', 'cancellation_policy',
       'calculated_host_listings_count',
       'calculated_host_listings_count_entire_homes',
       'calculated_host_listings_count_private_rooms',
       'calculated_host_listings_count_shared_rooms', 'reviews_per_month',
       'check_in_24h', 'air_conditioning', 'high_end_electronics', 'bbq',
```

```
       'balcony', 'nature_and_views', 'bed_linen', 'breakfast', 'tv',
       'coffee_machine', 'cooking_basics', 'white_goods', 'elevator', 'gym',
       'child_friendly', 'parking', 'outdoor_space', 'host_greeting',
       'hot_tub_sauna_or_pool', 'internet', 'long_term_stays', 'pets_allowed',
       'private_entrance', 'secure', 'self_check_in', 'smoking_allowed',
       'accessible', 'event_suitable'],
      dtype='object')
```

[4]:
```
# We display the basic listings data.
pd.options.display.max_columns = 100
listings.head(3)
```

[4]:

	id	name	summary	description	experiences_offered	neighborhood_overview	transit	house_rules	host_id	host_since	host_response_time	host_response_rate	host_is_superhost
0	2539	Clean & quiet apt home by the park	Renovated apt home in elevator building.	Renovated apt home in elevator building. Spaci…	none	Close to Prospect Park and Historic Ditmas Park	Very close to F and G trains and Express bus i…	-The security and comfort of all our guests is…	2787	39698.0	within an hour	1.0	0.0
1	3647	THE VILLAGE OF HARLEM…NEW YORK !	NaN	WELCOME TO OUR INTERNATIONAL URBAN COMMUNITY T…	none	NaN	NaN	Upon arrival please have a legibile copy of yo…	4632	39777.0	within a day	1.0	0.0
2	7750	Huge 2 BR Upper East Cental Park	NaN	Large Furnished 2BR one block to Central Park…	none	NaN	NaN	NaN	17985	39953.0	within a day	1.0	0.0

```
   host_listings_count  host_identity_verified                        street  \
0                  6.0                     1.0  Brooklyn , NY, United States
1                  1.0                     1.0   New York, NY, United States
2                  2.0                     1.0   New York, NY, United States

  neighbourhood  latitude  longitude property_type        room_type  \
0      Brooklyn  40.64749  -73.97237     Apartment     Private room
1        Harlem  40.80902  -73.94190     Apartment     Private room
2        Harlem  40.79685  -73.94872     Apartment  Entire home/apt

   accommodates  bathrooms  bedrooms  beds       bed_type  \
0             2        1.0         1     1       Real Bed
1             2        1.0         1     1  Pull-out Sofa
2             4        1.0         2     2       Real Bed

                                        amenities  price  guests_included  \
0  {TV,"Cable TV",Internet,Wifi,"Wheelchair acces…    149                1
1  {"Cable TV",Internet,Wifi,"Air conditioning",K…    150                2
2  {TV,"Cable TV",Internet,Wifi,"Air conditioning…    190                1

   extra_people  minimum_nights calendar_updated  has_availability  \
0            35               1      3 weeks ago                 1
1            20               3    34 months ago                 1
2             0               7      7 weeks ago                 1

   availability_30  availability_60  availability_90  availability_365  \
0               30               60               90               365
1               30               60               90               365
2                4               14               14               249

   number_of_reviews  number_of_reviews_ltm  review_scores_rating  \
0                  9                      2                  98.0
1                  0                      0                   NaN
2                  0                      0                   NaN

   review_scores_accuracy  review_scores_cleanliness  review_scores_checkin  \
0                    10.0                       10.0                   10.0
1                     NaN                        NaN                    NaN
2                     NaN                        NaN                    NaN

   review_scores_communication  review_scores_location  review_scores_value  \
0                         10.0                    10.0                 10.0
1                          NaN                     NaN                  NaN
2                          NaN                     NaN                  NaN

   instant_bookable          cancellation_policy  \
0                 0                     moderate
```

```
1                0  strict_14_with_grace_period
2                0                     flexible

   calculated_host_listings_count  \
0                               6
1                               1
2                               2

   calculated_host_listings_count_entire_homes  \
0                                            0
1                                            0
2                                            1

   calculated_host_listings_count_private_rooms  \
0                                             5
1                                             1
2                                             1

   calculated_host_listings_count_shared_rooms  reviews_per_month  \
0                                            1               0.21
1                                            0                NaN
2                                            0                NaN

   check_in_24h  air_conditioning  high_end_electronics  bbq  balcony  \
0             1                -1                    -1   -1       -1
1            -1                 1                    -1   -1       -1
2            -1                 1                    -1   -1       -1

   nature_and_views  bed_linen  breakfast  tv  coffee_machine  cooking_basics  \
0                -1          1         -1   1               1               1
1                -1         -1         -1   1              -1              -1
2                -1         -1         -1   1              -1              -1

   white_goods  elevator  gym  child_friendly  parking  outdoor_space  \
0            1         1   -1              -1        1             -1
1           -1        -1   -1              -1       -1             -1
2           -1         1   -1              -1       -1             -1

   host_greeting  hot_tub_sauna_or_pool  internet  long_term_stays  \
0             -1                     -1         1                1
1             -1                     -1         1               -1
2             -1                     -1         1               -1

   pets_allowed  private_entrance  secure  self_check_in  smoking_allowed  \
0            -1                -1       1              1               -1
1            -1                -1      -1             -1               -1
2             1                -1      -1             -1               -1
```

```
   accessible  event_suitable
0           1               1
1          -1              -1
2          -1              -1
```

The following are details about some of the important columns here:

1. `neighbourhood`: which neighborhood the property is in
2. `longitude`, `latitude`: longitude and latitude
3. `property_type`: type of property, such as apartment, condo etc.
4. `bathrooms`: number of bathrooms
5. `bedrooms`: number of bathrooms
6. `price`: price of the listing
7. `number_of_reviews`: number of reviews given by customers who stayed there
8. `parking`: 1 means there is parking available, -1 means there is not

For other categorical variables, such as `outdoor_friendly`, `gym`, etc., the 1,-1 should be interpreted similarly to `parking` as explained above.

1.2.1 Plotting the marginal distributions of key quantities of interest

As you have seen in the Python cases, it is good to first develop an idea of how the values of a few key quantities of interest are distributed. Let's start by doing so for some numeric variables, such as `price`, `bedrooms`, `bathrooms`, `number_of_reviews`:

1.2.2 Exercise 1:

1.1 Use the `describe()` and `quantile()` commands to compute some important summary statistics for the above variables.

Answer. One possible solution is given below:

[5]:
```
listings[['price','bedrooms','bathrooms','number_of_reviews']].describe()
```

[5]:

	price	bedrooms	bathrooms	number_of_reviews
count	30179.000000	30179.000000	30179.000000	30179.000000
mean	132.949965	1.265516	1.151595	4.459889
std	93.151824	0.614659	0.422225	5.265633
min	0.000000	1.000000	0.500000	0.000000
25%	65.000000	1.000000	1.000000	0.000000
50%	100.000000	1.000000	1.000000	2.000000
75%	175.000000	1.000000	1.000000	7.000000
max	500.000000	11.000000	7.500000	20.000000

[6]:
```
listings['price'].quantile([0.9,0.95,0.99])
```

[6]:
```
0.90    250.0
0.95    325.0
0.99    450.0
Name: price, dtype: float64
```

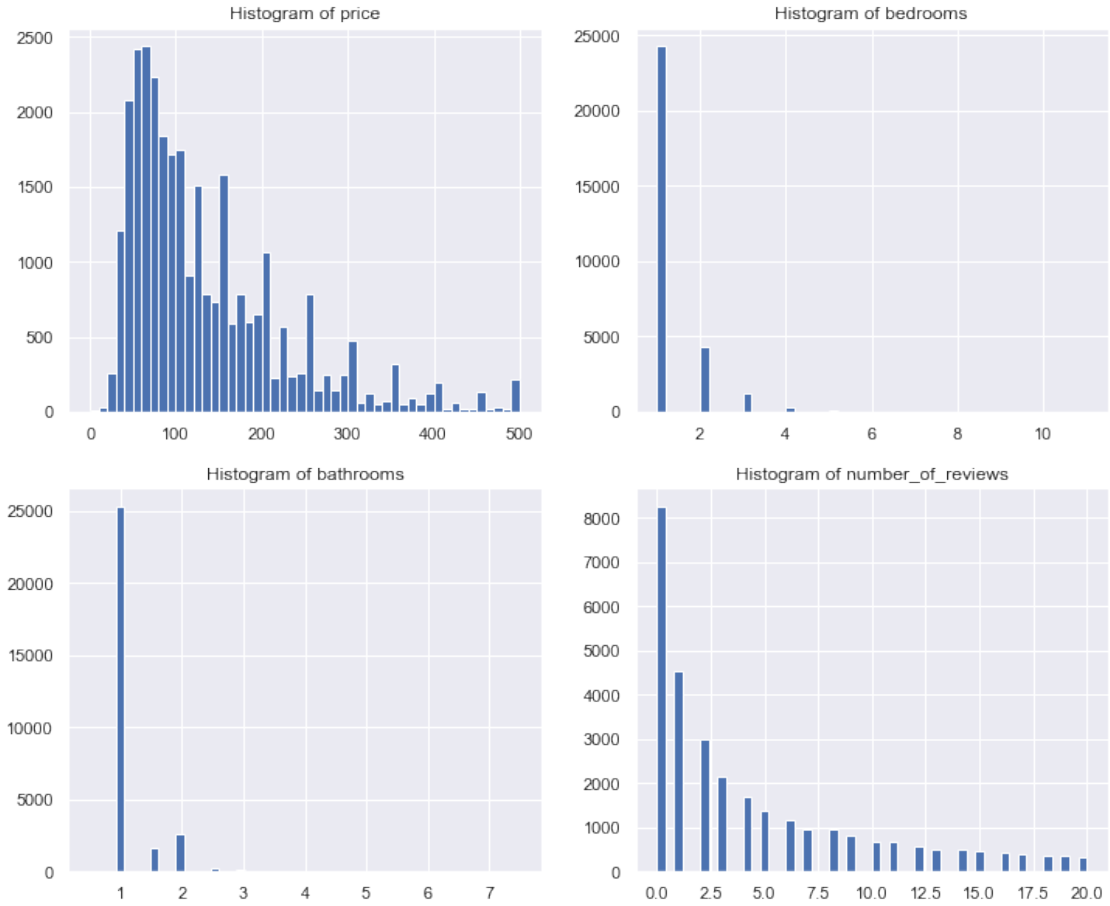
[7]:
```
listings['number_of_reviews'].describe()
```

[7]:
```
count    30179.000000
mean         4.459889
std          5.265633
min          0.000000
25%          0.000000
50%          2.000000
75%          7.000000
max         20.000000
Name: number_of_reviews, dtype: float64
```

1.2 Use the `plt.hist()` function to plot the histogram of the above variables. What are their basic shapes (e.g. normal, skewed, multi-modal, etc.)?

Answer. All look somewhat skewed to the right, though the `bathroom` variable is so concentrated at a single entry that it is hard to tell.

[8]:
```
plt.figure(figsize=(12,10))
vars_to_plot = ['price', 'bedrooms','bathrooms','number_of_reviews']
for i, var in enumerate(vars_to_plot):
    plt.subplot(2,2,i+1)
    plt.hist(listings[var],50)
    title_string = "Histogram of " + var
    plt.title(title_string)
```



1.3 Are the distributions fairly smooth, or do they exhibit "spiky" or "discontinuous" behavior? If the latter, can you explain where it might come from?

Answer. The `price` variable is noticeably spiky. There is a nice bulk of prices between about 25 and 300 dollars, with very obvious spikes at nice, round numbers such as 50, 100, 150, 200, 250, and 300. This probably reflects the fact that people enter in the prices that they wish to list at, and so tend to choose round numbers (or numbers just below round numbers).

1.4 Can you detect any outliers from these histograms? If so, do they suggest (i) data error; or (ii) data that should be omitted from our future analysis?

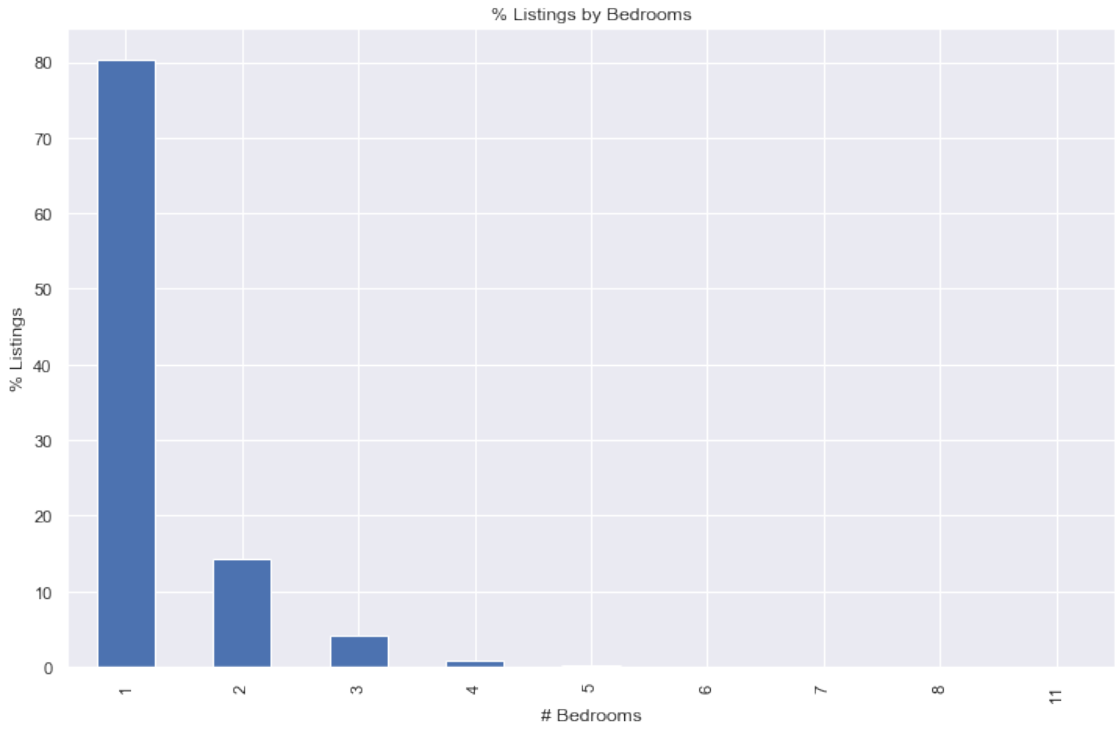
Answer. Very few places had prices of more than $320, and so we might think of these as "outliers". Some of these may represent error, but we guess that most of them are correct – hotels in NYC certainly often go for over 400 dollars per night, and so it is not unreasonable to expect some Airbnb listings of this price. The question as to whether we should omit these outliers is a little more difficult, but we lean towards omitting them for most clients. Even if these prices are correct, we suspect that they are governed by idiosyncratic factors that are not as relevant to the listings that most of our clients are interested in analyzing. Thus, they will tend to give us misleading (or

"biased") results.

1.2.3 Another way to look at the histogram of number of bedrooms

Sometimes, it is better to look at a histogram which plots the relative percentages of values across categories:

[9]:
```
# How many bedrooms
bedrooms_counts = Counter(listings.bedrooms)
tdf = pd.DataFrame.from_dict(bedrooms_counts, orient = 'index').sort_values(by␣
 ↪= 0)
tdf = (tdf.iloc[-10:, :] / len(listings)) * 100

# Sort bedroom dataframe by number
tdf.sort_index(axis = 0, ascending = True, inplace = True)

# Plot percent of listings by bedroom number
ax = tdf.plot(kind = 'bar', figsize = (12, 7.5))
ax.set_xlabel("# Bedrooms")
ax.set_ylabel("% Listings")
ax.set_title('% Listings by Bedrooms')
ax.legend_.remove()

plt.show()

print("Percent of 1 Bedroom Listings: %{0:.2f}".format(tdf[0][1]))
#The syntax 0:.2f denotes that we will print upto to decimal places
#Change it to %{0:.3f to see what happens}
```




```
Percent of 1 Bedroom Listings: %80.37
```

1.3 Inspecting price against variables of interest

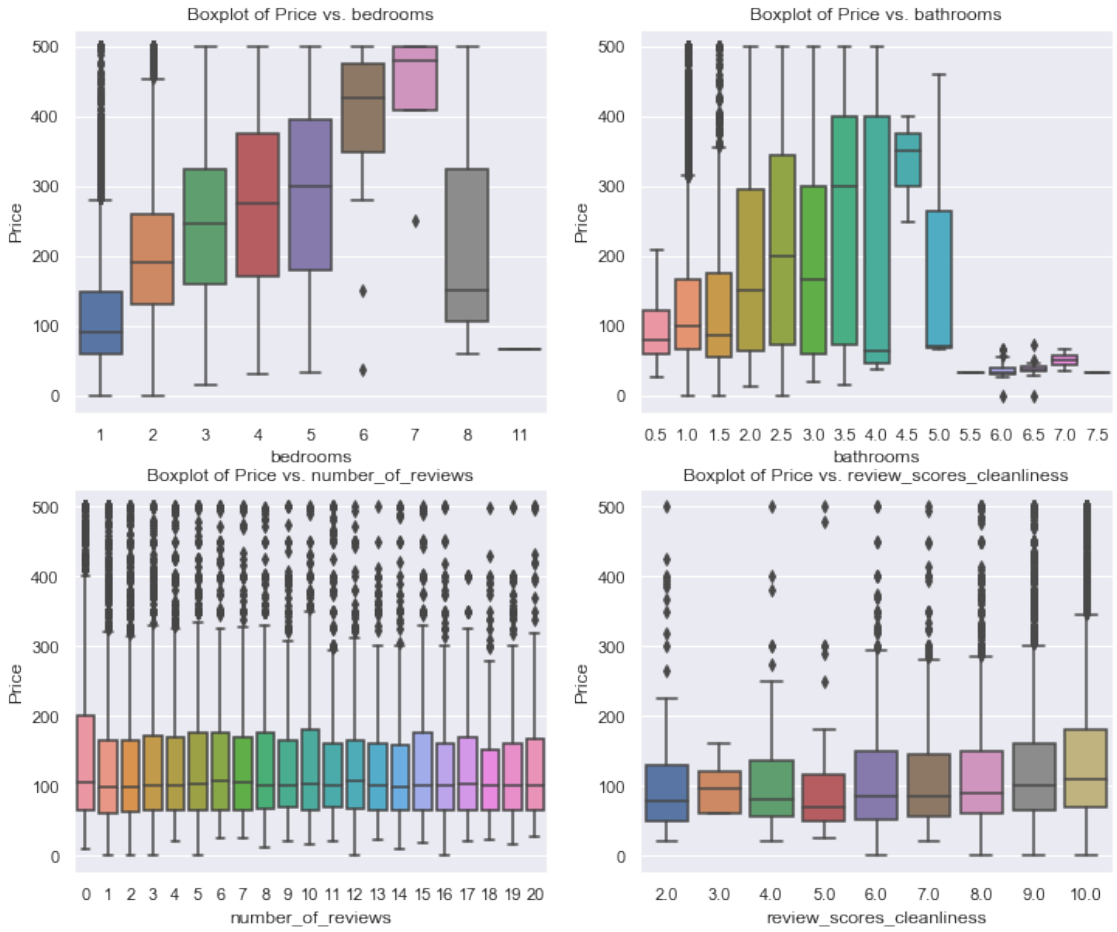
Now that we have looked at the variables of interest in isolation, it makes sense to look at them in relation to price.

1.3.1 Exercise 2:

2.1 Write code for making a boxplot of `price` vs. `bedrooms`, `bathrooms`, `number_of_reviews`, `review_scores_cleanliness`.

Answer. One possible solution is given below:

[10]:
```
plt.figure(figsize=(12,10))
vars_to_plot =␣
 ↪['bedrooms','bathrooms','number_of_reviews','review_scores_cleanliness']
for i, var in enumerate(vars_to_plot):
    plt.subplot(2,2,i+1)
    sns.boxplot(x = var, y='price', data = listings)
    title_string = "Boxplot of Price vs. " + var
    plt.ylabel("Price")
    plt.title(title_string)
```



2.2 Comment on the relationship between price and the respective variable in each of the above plots.

Answer.

1. As expected, the median price increases with the number of bedrooms. This relationship also seems linear.
2. Again as expected, the price on average seems to increase with the number of bathrooms. There seems to be some outliers which defy this trend.
3. The number of reviews do not seem to affect the median price.
4. There seems to be a slight increase in median price with increase in cleanliness review scores.

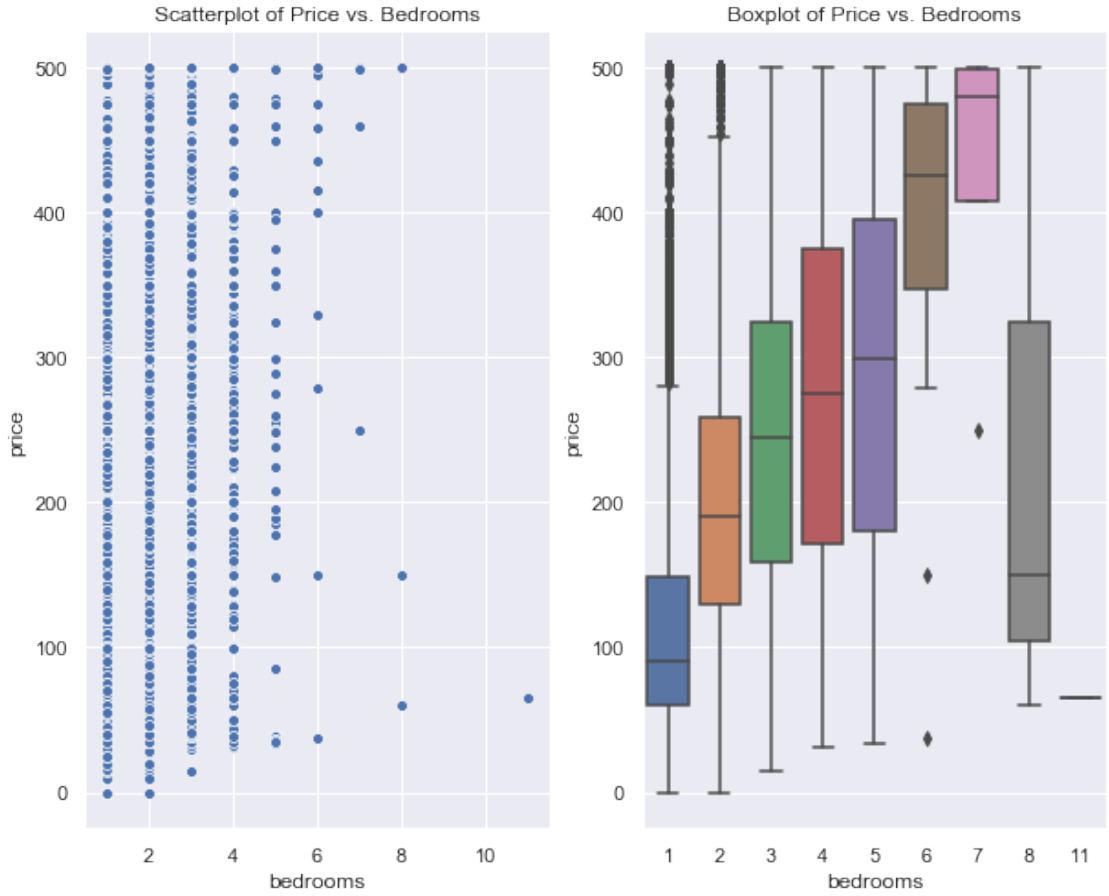
1.3.2 Investigating correlations (15 mts)

Although plotting the relationship between price and a few other variables is a good first step, overall there are too many variables to individually plot and manually inspect. We need a more systematic method. How do we proceed? An easy way to get a quick overview of the key variables that affect the price is via correlation.

Let's look at the `price` vs. `bedrooms` plot again:

[11]:
```
plt.figure(figsize=(10,8))
plt.subplot(121)
sns.scatterplot(x='bedrooms',y = 'price', data = listings)
plt.ylabel("price")
plt.title("Scatterplot of Price vs. Bedrooms")
plt.subplot(122)
sns.boxplot(x='bedrooms',y= 'price', data = listings)
plt.ylabel("price")
plt.title("Boxplot of Price vs. Bedrooms")
```

[11]:
```
Text(0.5, 1.0, 'Boxplot of Price vs. Bedrooms')
```



We see that as the number of bedrooms increases, the price on average increases. The quantity correlation is one way to capture this relationship. The correlation of two quantities is a measurement of how much they tend to increase together, measured on a scale going from -1 to 1. A positive correlation between price and number of bedrooms would indicate that higher-priced listings tend to have more bedrooms. Similarly, a negative correlation between price and number of bedrooms would indicate that higher-priced listings tend to have fewer bedrooms. In our case, we can easily

see that price is positively correlated with bedrooms.

Since correlation is just a single number summarizing an entire joint distribution, it can be misleading and does not eliminate the need to plot and visually inspect the key variables that it suggests are important. Nonetheless, it is quite helpful when quickly scanning for very strong relationships in the data and whittling down a much larger list of potential factors.

[12]:
```
np.corrcoef(listings['price'],listings['bedrooms'])[0,1]
```

[12]:
```
0.4545392952627167
```

The correlation matrix then gives all of the pairwise correlations between all of the variables. We can get a quick overview of the key variables that affect the price by looking at its row in the correlation matrix.

1.3.3 Exercise 3:

3.1 Write code to compute the correlation matrix between the price and other quantities. (use .corr() function).

3.2 Print the columns which are positively correlated, in increasing order of the correlation.

3.3 Print the columns which are negatively correlated, in increasing order of the magnitude of the correlation.

[13]:
```
# Create a correlation matrix
corr    = listings.corr()
pos_cor = corr['price'] >0
neg_cor = corr['price']  <0
corr['price'][pos_cor].sort_values(ascending = False)
#This prints out the coefficients that are positively correlated with price.
```

[13]:
```
price                                          1.000000
accommodates                                   0.571541
bedrooms                                       0.454539
beds                                           0.421355
guests_included                                0.321970
tv                                             0.271563
elevator                                       0.229610
calculated_host_listings_count_entire_homes    0.218890
white_goods                                    0.214283
gym                                            0.209892
child_friendly                                 0.206189
air_conditioning                               0.196582
host_listings_count                            0.194891
calculated_host_listings_count                 0.185098
```

```
bathrooms                                       0.163276
private_entrance                                0.146453
review_scores_location                          0.140308
coffee_machine                                  0.133457
availability_365                                0.118864
bbq                                             0.110258
self_check_in                                   0.108914
bed_linen                                       0.105295
availability_60                                 0.099858
long_term_stays                                 0.096381
review_scores_cleanliness                       0.095329
availability_90                                 0.093483
cooking_basics                                  0.087954
latitude                                        0.079542
balcony                                         0.079413
extra_people                                    0.075439
availability_30                                 0.075208
pets_allowed                                    0.070507
hot_tub_sauna_or_pool                           0.064996
review_scores_rating                            0.064228
reviews_per_month                               0.063172
check_in_24h                                    0.053180
minimum_nights                                  0.048627
outdoor_space                                   0.046216
accessible                                      0.038301
host_response_rate                              0.034280
review_scores_accuracy                          0.033125
host_id                                         0.028931
high_end_electronics                            0.027578
id                                              0.027533
instant_bookable                                0.024415
internet                                        0.024220
secure                                          0.019678
review_scores_communication                     0.017169
review_scores_checkin                           0.014043
event_suitable                                  0.012606
breakfast                                       0.007514
nature_and_views                                0.000173
Name: price, dtype: float64
```

[14]:
```
corr['price'][neg_cor].sort_values()
```

[14]:
```
longitude                                       -0.294196
calculated_host_listings_count_private_rooms    -0.127504
calculated_host_listings_count_shared_rooms     -0.101389
smoking_allowed                                 -0.054131
host_greeting                                   -0.051356
```

```
number_of_reviews                -0.029229
host_identity_verified           -0.022861
parking                          -0.019383
host_is_superhost                -0.016325
number_of_reviews_ltm            -0.011481
host_since                       -0.009671
review_scores_value              -0.005942
Name: price, dtype: float64
```

3.4: From the table above, what factors are most correlated with price? Which correlations are surprising?

Answer. Many of these are unsurprising – for example, the largest correlations are with measures of size (`accommodates`, `bedrooms`, `beds`, etc.). Review scores are only slightly related to price. Looking at the location-related scores, we find that longitude is negatively related to price while latitude is not. This motivates us to plot them on a map (we will do this next).

We also notice a few correlations that seem a bit surprising. For example:

1. Parking is negatively correlated to price. This correlation with parking is very suspicious – why would parking be bad? I suspect that it is "spurious", caused by the fact that parking is more common in less expensive neighborhoods. Let's investigate this by looking at parking in a region-by-region manner.
2. Being a superhost is negatively correlated; we don't follow up on it here.
3. The total number of listings is positively correlated. This seems counterintuitive, as one would want large-scale listers to be able to rent more cheaply due to economies of scale.

1.4 Location, location, location! (20 mts)

In Exercise 3, we found quite a few variables that are reasonably correlated with price. We could continue our exploration by looking at each of these variables in turn, but we know that in real estate, location data is quite special, and so we will first explore how location affects the price in greater detail.

We will use the `folium` package. Make sure that you have installed the package (if not, do it now!). The following gives an interactive map for plotting the listings on a map of New York City:

[15]:
```
folium_map = folium.Map(location=[40.738, -73.98],
                        zoom_start=13,
                        tiles="OpenStreetMap")
folium_map
# This sets up a basic map of NYC. You can try to change the "tiles" option␣
 ↪above.
# The options you have are: "OpenStreetMap", Mapbox Bright","Stamen␣
 ↪Toner","Mapbox Control Room","Stamen Terrain"
```

[15]:
```
<folium.folium.Map at 0x1a1b71f9b0>
```

[33]:
```
folium_map = folium.Map(location=[40.738, -73.98],
                        zoom_start=13,
                        tiles="OpenStreetMap")
#Now we can have a scatter plot of the first 1000 data points on the above map
for i in range(0,1000):
    marker = folium.
 ↪CircleMarker(location=[listings["latitude"][i],listings["longitude"][i]],radius=5,color="r"
    marker.add_to(folium_map)

folium_map
```

[33]:
```
<folium.folium.Map at 0x1a1f3fecf8>
```

1.4.1 Using heatmaps to understand the price distribution with location

Next, we create a heatmap of the price of apartments in NYC. This will give us a sense of where the important locations are:

[17]:
```
max_amount = float(listings['price'].max())

folium_hmap = folium.Map(location=[40.738, -73.98],
                        zoom_start=13,
                        tiles="OpenStreetMap")

hm_wide = HeatMap( list(zip(listings['latitude'], listings['longitude'],␣
 ↪listings['price'])),
                   min_opacity=0.2,
                   max_val=max_amount,
                   radius=8, blur=6,
                   max_zoom=15,
                  )

folium_hmap.add_child(hm_wide)
```

[17]:
```
<folium.folium.Map at 0x1a1d5e5978>
```

1.4.2 Exercise 4:

4.1 What areas in NYC have expensive rentals?

Answer. Manhattan and Brooklyn are very pricey; Queens is relatively less so. In general, things farther away from Manhattan and Brooklyn are cheaper.

4.2 Looking at this map, you can (roughly) see the correlation between price and longitude/latitude. Does location appear to be strongly related to price? Does it seem likely that

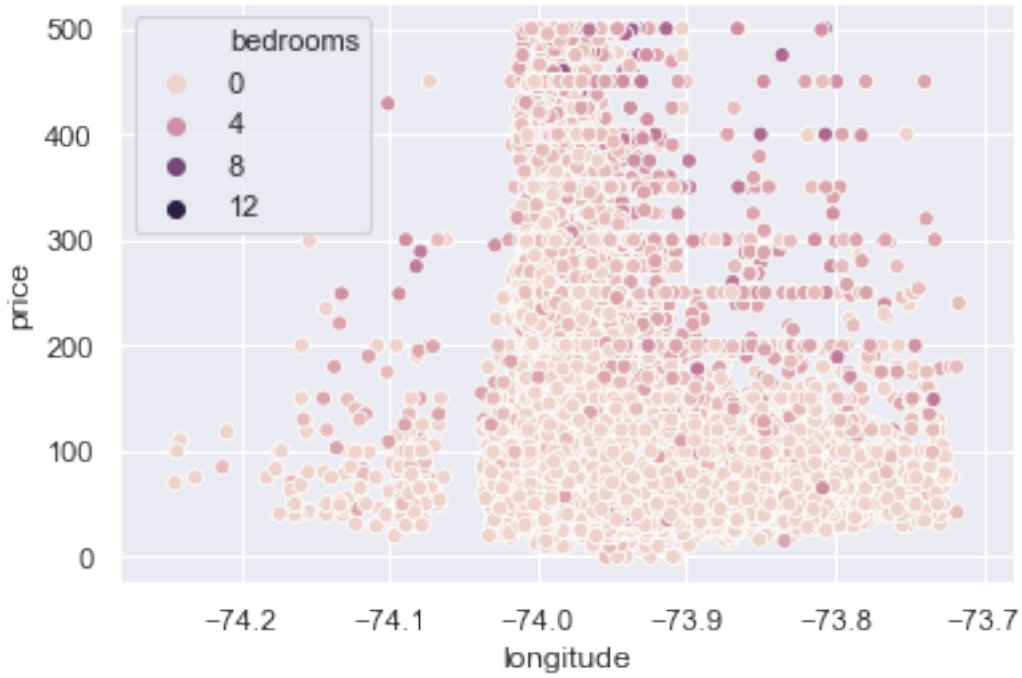
this relationship could be captured well by a linear model?

Answer. The location does seem to be strongly related to price; however, the relationship doesn't seem to be close to linear. This suggests that, when we model price, we will need to incorporate location data and our method cannot be via a linear mapping.

4.3 Write code to make a scatterplot between price and longitude, with number of bedrooms categorized by color. Answer: We can also plot `longitude` vs. `price`:

[18]:
```
sns.scatterplot(x= listings['longitude'], y = listings['price'], hue =␣
↪listings['bedrooms'])
```

[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1be3d5c0>
```



When looking at the list of correlations, `parking` stood out as having a surprisingly negative correlation with price. We've seen that location has a strong influence on price; let's see if it can help explain the negative correlation exhibited by `parking`.

1.4.3 Exercise 5:

Write code here to plot the first 1000 locations on the map where parking is available by blue color, and the first 1000 locations where parking is not available by red color. (Hint: You can use the command: color = "blue" and "red" respectively.)

Answer. One possible solution is given below:

[19]:
```
lat_log_parking_yes = listings.loc[ listings['parking']==1.0,␣
↪["latitude","longitude" ] ]
lat_log_parking_no = listings.loc[ listings['parking']==-1.0,␣
↪["latitude","longitude" ] ]
folium_map = folium.Map(location=[40.738, -73.98],
                        zoom_start=13,
                        tiles="OpenStreetMap")
for i in range(1000):
    marker = folium.CircleMarker(location=[lat_log_parking_yes["latitude"].
↪iloc[i],lat_log_parking_yes["longitude"].
↪iloc[i]],radius=5,color="blue",fill=True)
    marker.add_to(folium_map)

for i in range(1000):
    marker = folium.CircleMarker(location=[lat_log_parking_no["latitude"].
↪iloc[i],lat_log_parking_no["longitude"].
↪iloc[i]],radius=5,color="red",fill=True)
    marker.add_to(folium_map)

folium_map
```

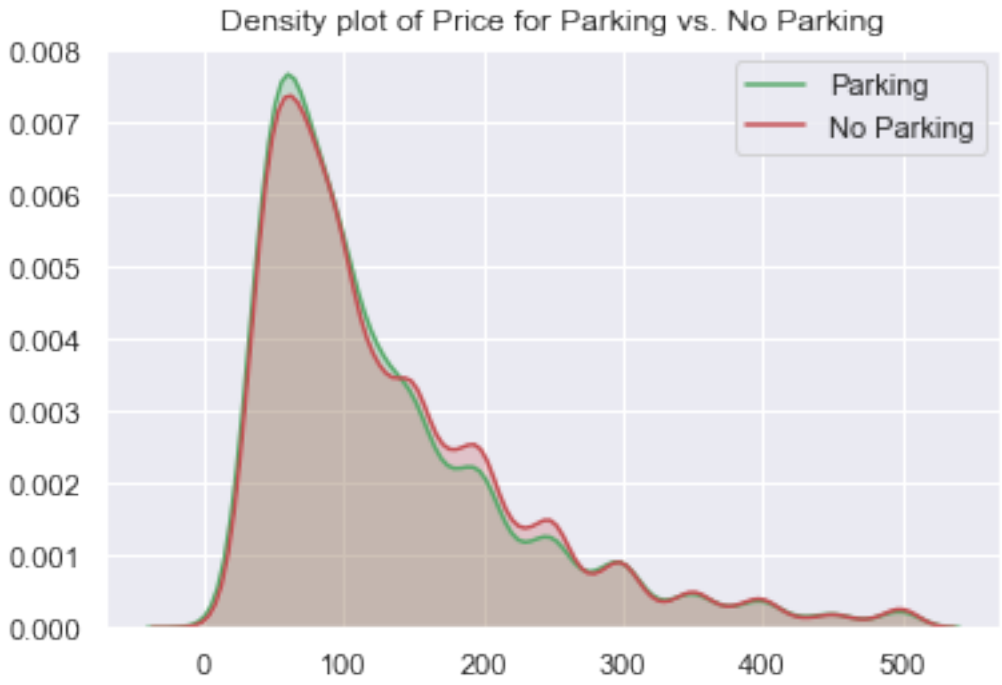
[19]:
```
<folium.folium.Map at 0x1a1c16cda0>
```

1.5 Interaction effects and iterative hypotheses (15 mts)

Now that we have explored some of the factors that are expected to affect price, let's focus on understanding the unexpected correlations, such as the negative correlation with parking. We start with the latter:

[34]:
```
# First, plot parking vs. non-parking prices.
sns.kdeplot(listings.loc[listings['parking'] == 1,'price'],shade = True,␣
↪label="Parking",color="g")
sns.kdeplot(listings.loc[listings['parking'] == -1,'price'],shade = True,␣
↪label="No Parking",color="r")
plt.title("Density plot of Price for Parking vs. No Parking");
```

Kernel Density Estimate



We saw before that the correlation between price and parking is -0.019383. Since parking is desirable, we expect the price to increase with parking. When we see a pattern like this, we should suspect the existence of **interaction effects** that are complicating the parking vs. price relationship. Interaction effects are when the relationship between two variables is **conditional**, or depends on the value of a third, hidden variable.

What could this third variable potentially be? Well, we have seen that location has a huge impact on prices. Perhaps high-price areas don't have many parking spots, whereas low-price areas do? We don't know this for sure, but it's a worthwhile guess.

More formally, we hypothesize that this observed negative correlation is the result of interaction effects arising from location. In order to investigate this hypothesis, we ought to break down the locations by neighborhood and see if this negative correlation between price and parking still holds within neighborhoods. The neighborhoods are discrete and there are many listings per neighborhood, so we can simply compute the correlation for every neighborhood individually. Mathematically, this is exactly the same thing as conditioning on the neighborhood and computing the conditional correlation.

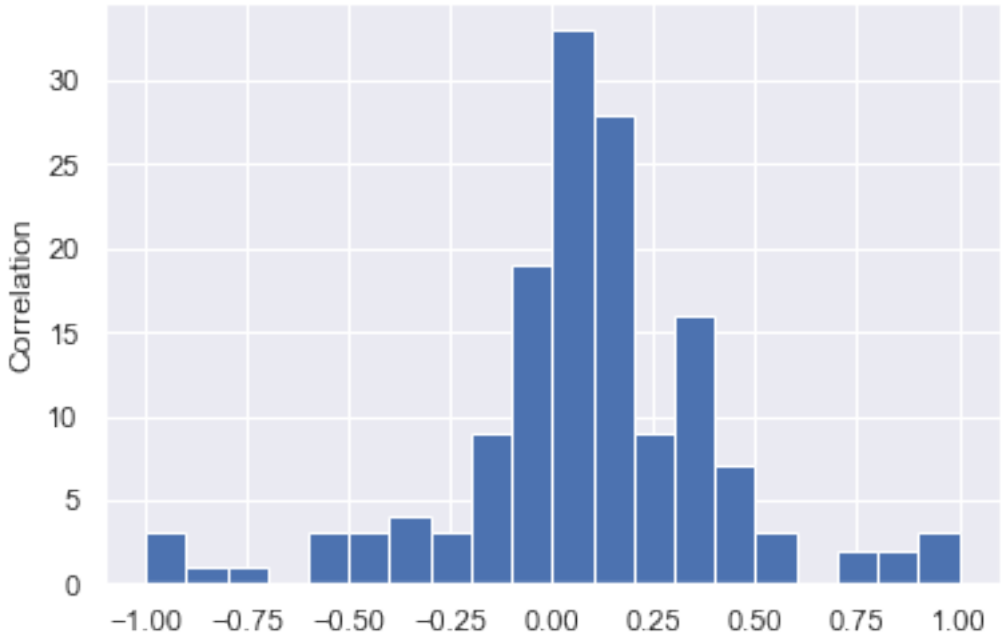
1.5.1 Exercise 6:

6.1 Write code to make a dictionary in which the keys are the `neighbourhoods` in the dataset and the values are the correlation between price and parking for that neighborhood.

6.2 Next plot a histogram of these correlations.

[21]:
```
neighbourhoods = listings.neighbourhood.unique()
cvec = list()
cvec = dict()

for x in neighbourhoods:
    temp = listings[listings['neighbourhood'] == x]
    cvec[x] = temp.corr()['price']['parking']


res = list(cvec.values())
res = [x for x in res if str(x) != 'nan']
res.sort()


plt.hist(res, bins=20)
plt.ylabel('Correlation')
plt.show()


print('Average correlation: ', sum(res)/len(res))
# print(cvec)

#print(list(cvec).sort())
```

Histograma de Correlaciones de Barrios en NYC




```
Average correlation:  0.08178058684472027
```

6.3 Explain the relationship between the histogram and our finding that parking is negatively correlated with price. **Answer.** Our original correlation of about -0.02 was the correlation between price and parking for all listings in NYC – that is, the conditional correlation between price and parking given that you are in NYC. The number `res['Brooklyn']` is the correlation between price and parking for all listings in Brooklyn – that is, the conditional correlation between price and parking given that you are in Brooklyn.

The histogram shows us that most of the conditional correlations within neighborhoods are positive, even though the correlation across all of NYC is negative. Roughly speaking, this means that the following are all occurring:

1. Within neighborhoods, parking is positively associated with price.
2. Different neighborhoods have very different typical prices (as we saw last section).
3. Parking tends to be concentrated in cheaper neighborhoods.

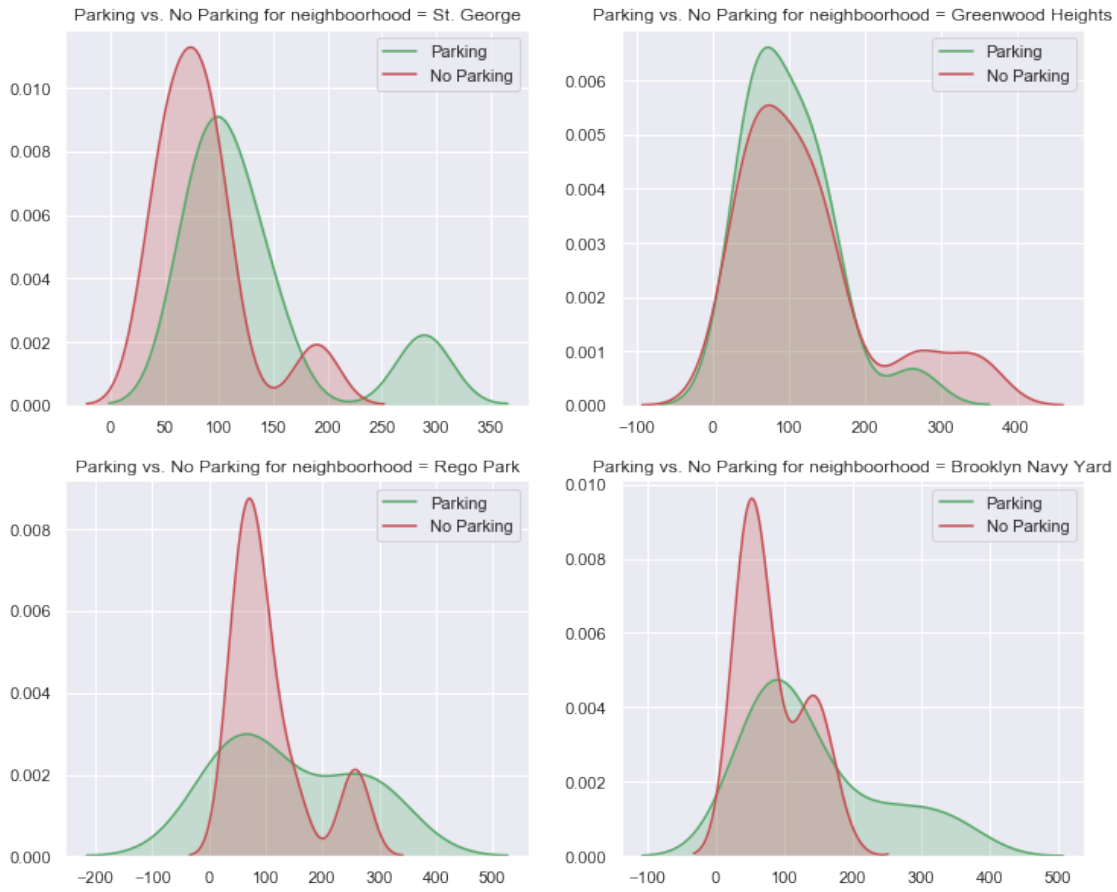
The correlation values of 1 and -1 are presumably due largely to neighborhoods with very few listings, and should essentially be ignored. Viewing the histogram, however, we can see that a clear majority of correlations are at least slightly positive, for an average correlation of 0.08.

6.4 Plot the histogram that overlays the distribution of price for parking and non-parking (use sns.kdeplot) for the neighborhoods: `St. George`,`Greenwood Heights`,`Rego Park`,`Brooklyn Navy Yard`.

If we plot this by neighborhood for a few neighborhoods, we can see this somewhat positive correlation of parking vs. no parking visually:

[23]:
```
plt.figure(figsize=(12,10))
neigh_to_look = ['St. George','Greenwood Heights','Rego Park','Brooklyn Navy␣
 ↪Yard']
for i, neigh in enumerate(neigh_to_look):
 plt.subplot(2,2,i+1)
 sns.kdeplot(listings.loc[(listings['parking'] == 1) &␣
 ↪(listings['neighbourhood'] == neigh),'price'],shade = True,␣
 ↪label="Parking",color="g")
 sns.kdeplot(listings.loc[(listings['parking'] == -1) &␣
 ↪(listings['neighbourhood'] == neigh),'price'],shade = True, label="No␣
 ↪Parking",color="r")
 plt.title("Parking vs. No Parking for neighboorhood = " + str(neigh));
```

the histogram that overlays the distribution of price for parking and non-parking



As we have seen, the existence of unexpected correlations should spur investigation into potential interaction effects, which lead to potentially interesting hypotheses. Thus, one good way of generating iterative hypotheses is to find and think about potential interaction effects.

1.5.2 Finding more interactions: how does price vary by property type?

We saw that finding conditional correlations or interactions is a good way to generate further hypotheses, as many interesting lines of investigation arise from investigating these **confounding variables**. Here is another example: let's now look at how price varies with property type. The following code plots the price of a one bedroom listing broken down by the property type:

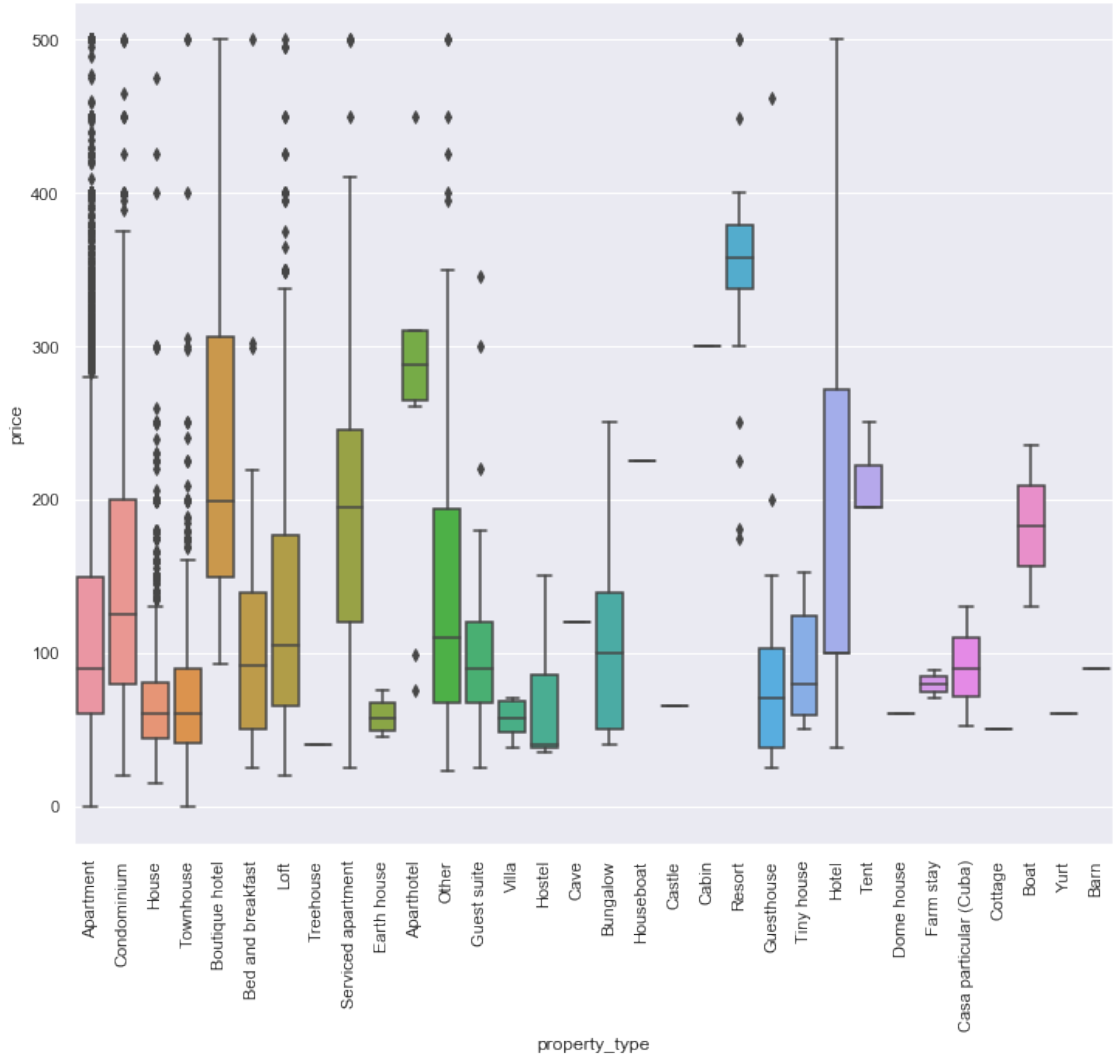
1.6 Exercise 7

7.1 Write code to make a boxplot of price of one bedroom property across all property types.

[24]:
```
plt.figure(figsize=(12,10))
sns.boxplot(y=listings.loc[listings['bedrooms']==1,'price'], x= listings.
 ↪loc[listings['bedrooms']==1,'property_type'])
```

```
plt.xticks(rotation = 90)
```

[24]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]),
 <a list of 32 Text xticklabel objects>)
```



7.2 What can you conclude about the variation in price of a one bedroom by the property type?

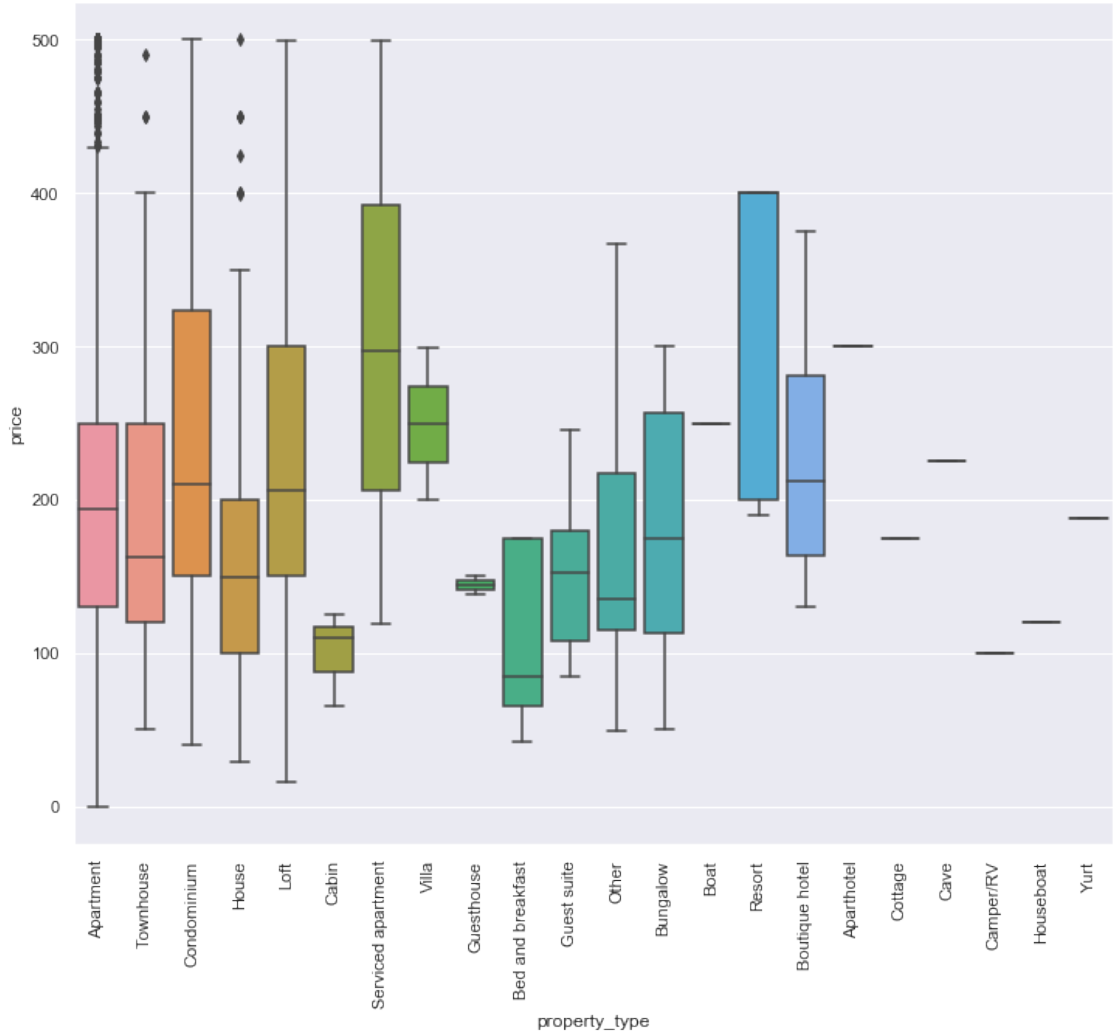
Answer. There is significant variation in price according to the property type; a room in a house or a loft is the cheapest, while cabins, boutique hotels, and boats are very expensive! It is also interesting to see huge variations in hotel prices.

7.3 Do the same price vs. property type plot for two bedroom listings.

Answer. One possible solution is given below:

[25]:
```
plt.figure(figsize=(12,10))
sns.boxplot(y=listings.loc[listings['bedrooms']==2,'price'], x= listings.
 ↪loc[listings['bedrooms']==2,'property_type'])
plt.xticks(rotation = 90)
```

[25]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21]), <a list of 22 Text xticklabel objects>)
```



7.4 (20 mts) Pick any other variable of your choice and make another interactive plot, showing the variation of price broken down by sub-categories of that variable.

1.6.1 More hypothesis generation (10 mts)

So far, we have primarily been interested in understanding what factors influence the price of an existing listing. However, a natural question to ask is what you can do to increase the price of your own listing without spending too much money?

Looking through the list of correlations, two immediately stand out:

Object	Correlation
AC	0.18
TV	0.26

Some air conditioning is relatively cheap, and has a large correlation of 0.189. Televisions are very cheap, and have an enormous correlation of 0.26. This suggests that one of the best things I can do for a listing is run out and buy a TV if I don't have one – a 300 dollar TV might increase the value of a listing by 30 dollars per night!

1.6.2 Question:

Do you really believe this could work? Can you come up with any simple plot or explanation that would argue one way or the other?

Apartments with TVs are systematically better than those without – indeed, presence of a television is positively correlated with many other positive price signals, from the size of the listing (e.g. number of people it accommodates) to essentially all other amenities. Nevertheless, it is important to not quickly conclude everything simply based on correlations. We will discuss this much more carefully in a later class on causal inference.

1.7 Exploring temporal effects: summer in Rio and winter in Moscow (10 mts)

We have seen that conditional plots can be a useful way to "correct" comparisons by taking into account interaction effects.

Time is a very common interaction effect that appears across lots of datasets. For Airbnb data, this is especially important, as Airbnb is often more expensive near holidays, and so reasonable price estimates must take this into account. In practice this is one of the most important corrections offered by Airbnb pricing consultancy firms, and corrections usually take advantage of data pooled from many somewhat similar cities. This is vital to achieving good corrections, but it is easy to make mistakes by failing to account for important city-to-city differences.

We begin by opening up the calendar data and counting (i) the number of rentals per day; and (ii) their total prices:

[26]:
```
cal = pd.read_csv('scal.csv', delimiter=',')
cal.head()
# Count rentals and total price on each date.
```

[26]:

	date	price
0	20190709	149.0
1	20190709	76.0
2	20190710	76.0
3	20190711	76.0
4	20190712	76.0

[27]:
```
rcount = dict()
rprice = dict()

for row in cal.itertuples(index=True, name='Pandas'):
    rcount[str(row[1])] = rcount.get(str(row[1]), 0) + 1
    rprice[str(row[1])] = rprice.get(str(row[1]), 0) + row[2]
```

[28]:
```
rcount
rprice
```

[28]:
```
{'20190709': 7311122.0,
 '20190710': 7336477.0,
 '20190711': 7364554.0,
 '20190712': 7574463.0,
 '20190713': 7600433.0,
 '20190714': 7374435.0,
 '20190715': 7402541.0,
 '20190716': 7416168.0,
 '20190717': 7420430.0,
 '20190718': 7434145.0,
 '20190719': 7608526.0,
 '20190720': 7633376.0,
 '20190721': 7404288.0,
 '20190722': 7397609.0,
 '20190723': 7430512.0,
 '20190724': 7444154.0,
 '20190725': 7448882.0,
 '20190726': 7639731.0,
 '20190727': 7672200.0,
 '20190728': 7431108.0,
 '20190729': 7428820.0,
 '20190730': 7433812.0,
 '20190731': 7428124.0,
 '20190801': 7464241.0,
 '20190802': 7653010.0,
 '20190803': 7677334.0,
 '20190804': 7469903.0,
 '20190805': 7463023.0,
 '20190806': 7483988.0,
 '20190807': 7488712.0,
```

```
'20190808': 7491771.0,
'20190809': 7666070.0,
'20190810': 7685216.0,
'20190811': 7473277.0,
'20190812': 7471003.0,
'20190813': 7478440.0,
'20190814': 7484016.0,
'20190815': 7483085.0,
'20190816': 7660942.0,
'20190817': 7673251.0,
'20190818': 7463552.0,
'20190819': 7449766.0,
'20190820': 7468115.0,
'20190821': 7475174.0,
'20190822': 7487863.0,
'20190823': 7674081.0,
'20190824': 7702397.0,
'20190825': 7487806.0,
'20190826': 7469476.0,
'20190827': 7475308.0,
'20190708': 5649873.0,
'20190828': 7486646.0,
'20190829': 7523269.0,
'20190830': 7728517.0,
'20190831': 7775389.0,
'20190901': 7605130.0,
'20190902': 7532915.0,
'20190903': 7544932.0,
'20190904': 7569062.0,
'20190905': 7599174.0,
'20190906': 7776643.0,
'20190907': 7797763.0,
'20190908': 7573393.0,
'20190909': 7612472.0,
'20190910': 7640578.0,
'20190911': 7625951.0,
'20190912': 7632536.0,
'20190913': 7789154.0,
'20190914': 7802255.0,
'20190915': 7581844.0,
'20190916': 7600393.0,
'20190917': 7626463.0,
'20190918': 7632884.0,
'20190919': 7637768.0,
'20190920': 7820873.0,
'20190921': 7837326.0,
'20190922': 7610490.0,
```

```
'20190923': 7662028.0,
'20190924': 7678796.0,
'20190925': 7686533.0,
'20190926': 7664102.0,
'20190927': 7801627.0,
'20190928': 7809135.0,
'20190929': 7569281.0,
'20190930': 7547162.0,
'20191001': 7565986.0,
'20191002': 7600706.0,
'20191003': 7624577.0,
'20191004': 7821212.0,
'20191005': 7839690.0,
'20191006': 7570563.0,
'20191007': 7560211.0,
'20191008': 7580872.0,
'20191009': 7582916.0,
'20191010': 7596753.0,
'20191011': 7784847.0,
'20191012': 7809348.0,
'20191013': 7577700.0,
'20191014': 7558855.0,
'20191015': 7603868.0,
'20191016': 7610924.0,
'20191017': 7619754.0,
'20191018': 7781042.0,
'20191019': 7799124.0,
'20191020': 7565998.0,
'20191021': 7585231.0,
'20191022': 7617754.0,
'20191023': 7625660.0,
'20191024': 7619348.0,
'20191025': 7773520.0,
'20191026': 7781576.0,
'20191027': 7553366.0,
'20191028': 7547367.0,
'20191029': 7557903.0,
'20191030': 7562987.0,
'20191031': 7592592.0,
'20191101': 7741503.0,
'20191102': 7812999.0,
'20191103': 7578017.0,
'20191104': 7551582.0,
'20191105': 7552777.0,
'20191106': 7558275.0,
'20191107': 7572006.0,
'20191108': 7740407.0,
```

```
'20191109': 7769233.0,
'20191110': 7569229.0,
'20191111': 7565817.0,
'20191112': 7587439.0,
'20191113': 7596191.0,
'20191114': 7609888.0,
'20191115': 7773662.0,
'20191116': 7783499.0,
'20191117': 7574316.0,
'20191118': 7578081.0,
'20191119': 7607580.0,
'20191120': 7621928.0,
'20191121': 7640623.0,
'20191122': 7803628.0,
'20191123': 7827522.0,
'20191124': 7630929.0,
'20191125': 7619805.0,
'20191126': 7651433.0,
'20191127': 7719548.0,
'20191128': 7777991.0,
'20191129': 7981717.0,
'20191130': 7968948.0,
'20191201': 7736005.0,
'20191202': 7773335.0,
'20191203': 7802570.0,
'20191204': 7821771.0,
'20191205': 7838132.0,
'20191206': 8029319.0,
'20191207': 8042367.0,
'20191208': 7767553.0,
'20191209': 7768780.0,
'20191210': 7782659.0,
'20191211': 7780699.0,
'20191212': 7780008.0,
'20191213': 7961551.0,
'20191214': 7974846.0,
'20191215': 7717094.0,
'20191216': 7702629.0,
'20191217': 7706277.0,
'20191218': 7714856.0,
'20191219': 7738801.0,
'20191220': 7999159.0,
'20191221': 8070385.0,
'20191222': 7909280.0,
'20191223': 7952998.0,
'20191224': 8027895.0,
'20191225': 8061024.0,
```

```
'20191226': 8097005.0,
'20191227': 8318620.0,
'20191228': 8379392.0,
'20191229': 8240935.0,
'20191230': 8300043.0,
'20191231': 8565703.0,
'20200101': 8092336.0,
'20200102': 7890223.0,
'20200103': 8006255.0,
'20200104': 7981667.0,
'20200105': 7770598.0,
'20200106': 7745646.0,
'20200107': 7736960.0,
'20200108': 7736878.0,
'20200109': 7740304.0,
'20200110': 7954006.0,
'20200111': 7962447.0,
'20200112': 7737314.0,
'20200113': 7732832.0,
'20200114': 7733122.0,
'20200115': 7730242.0,
'20200116': 7734227.0,
'20200117': 7954599.0,
'20200118': 7960156.0,
'20200119': 7740960.0,
'20200120': 7726873.0,
'20200121': 7730391.0,
'20200122': 7733269.0,
'20200123': 7736238.0,
'20200124': 7953032.0,
'20200125': 7956735.0,
'20200126': 7735359.0,
'20200127': 7734405.0,
'20200128': 7738230.0,
'20200129': 7743188.0,
'20200130': 7752560.0,
'20200131': 7966937.0,
'20200201': 7964173.0,
'20200202': 7742444.0,
'20200203': 7734559.0,
'20200204': 7739044.0,
'20200205': 7742796.0,
'20200206': 7749857.0,
'20200207': 7972098.0,
'20200208': 7979165.0,
'20200209': 7744420.0,
'20200210': 7742095.0,
```

```
'20200211': 7746663.0,
'20200212': 7747607.0,
'20200213': 7757161.0,
'20200214': 7994919.0,
'20200215': 8010988.0,
'20200216': 7769952.0,
'20200217': 7754889.0,
'20200218': 7754747.0,
'20200219': 7754648.0,
'20200220': 7759557.0,
'20200221': 7981052.0,
'20200222': 7985023.0,
'20200223': 7751915.0,
'20200224': 7750522.0,
'20200225': 7752532.0,
'20200226': 7752252.0,
'20200227': 7762965.0,
'20200228': 7989602.0,
'20200229': 8006905.0,
'20200301': 7830523.0,
'20200302': 7832558.0,
'20200303': 7838129.0,
'20200304': 7837950.0,
'20200305': 7843559.0,
'20200306': 8088142.0,
'20200307': 8095565.0,
'20200308': 7860078.0,
'20200309': 7858204.0,
'20200310': 7868716.0,
'20200311': 7870895.0,
'20200312': 7873576.0,
'20200313': 8101061.0,
'20200314': 8111919.0,
'20200315': 7869675.0,
'20200316': 7870305.0,
'20200317': 7875420.0,
'20200318': 7876176.0,
'20200319': 7874681.0,
'20200320': 8101356.0,
'20200321': 8120174.0,
'20200322': 7892259.0,
'20200323': 7895453.0,
'20200324': 7905288.0,
'20200325': 7911619.0,
'20200326': 7915798.0,
'20200327': 8146021.0,
'20200328': 8153304.0,
```

```
'20200329': 7904671.0,
'20200330': 7909096.0,
'20200331': 7919027.0,
'20200401': 7947523.0,
'20200402': 7960211.0,
'20200403': 8192954.0,
'20200404': 8209870.0,
'20200405': 7959800.0,
'20200406': 7960613.0,
'20200407': 7979974.0,
'20200408': 7984081.0,
'20200409': 7986842.0,
'20200410': 8223138.0,
'20200411': 8236026.0,
'20200412': 7978375.0,
'20200413': 7985428.0,
'20200414': 8004233.0,
'20200415': 8057387.0,
'20200416': 8064438.0,
'20200417': 8293517.0,
'20200418': 8290325.0,
'20200419': 8027685.0,
'20200420': 8026092.0,
'20200421': 8041386.0,
'20200422': 8046026.0,
'20200423': 8052096.0,
'20200424': 8279807.0,
'20200425': 8283560.0,
'20200426': 8024278.0,
'20200427': 8026436.0,
'20200428': 8040613.0,
'20200429': 8053315.0,
'20200430': 8059123.0,
'20200501': 8339386.0,
'20200502': 8423898.0,
'20200503': 8164915.0,
'20200504': 8173374.0,
'20200505': 8185135.0,
'20200506': 8181478.0,
'20200507': 8182786.0,
'20200508': 8406319.0,
'20200509': 8410226.0,
'20200510': 8157904.0,
'20200511': 8168145.0,
'20200512': 8188612.0,
'20200513': 8197356.0,
'20200514': 8195462.0,
```

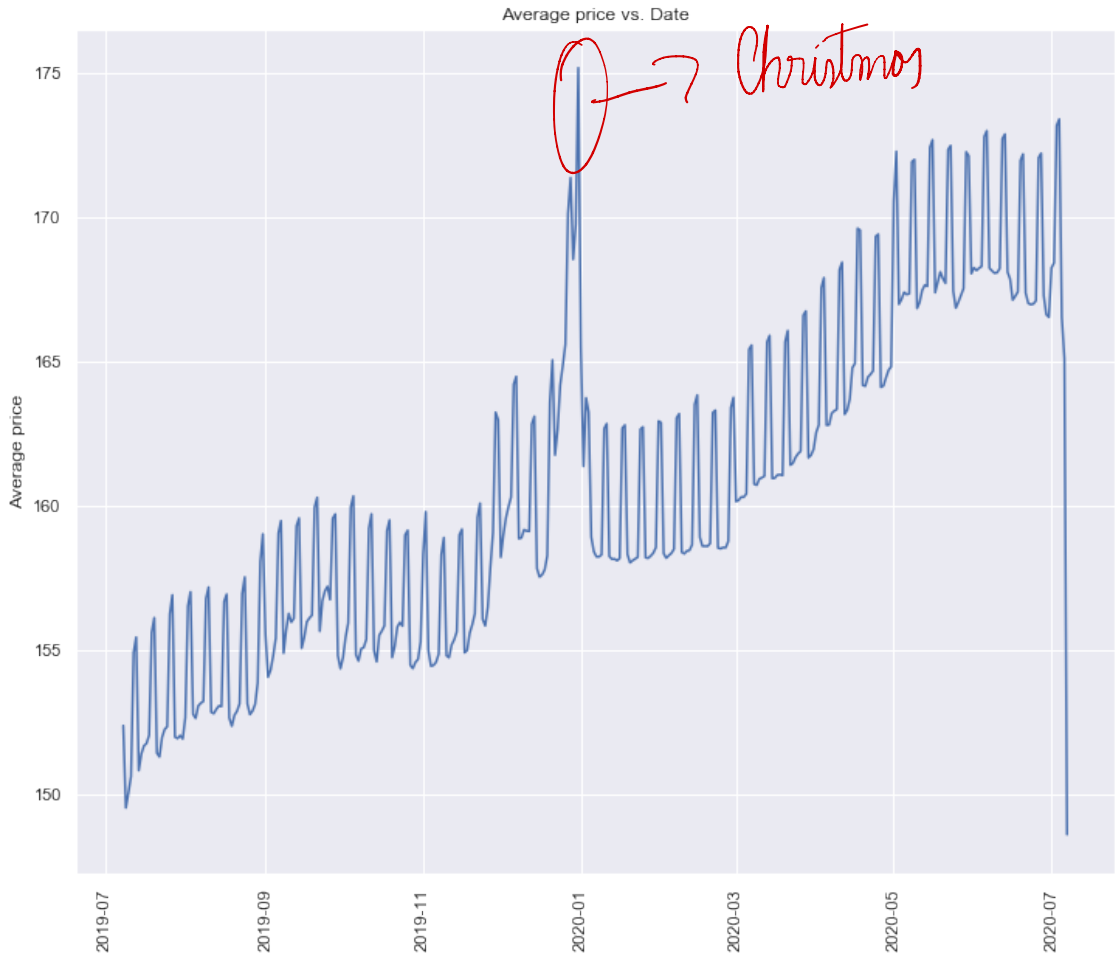
```
'20200515': 8430919.0,
'20200516': 8443555.0,
'20200517': 8184723.0,
'20200518': 8202092.0,
'20200519': 8219659.0,
'20200520': 8208609.0,
'20200521': 8200535.0,
'20200522': 8427099.0,
'20200523': 8433840.0,
'20200524': 8187212.0,
'20200525': 8158563.0,
'20200526': 8168635.0,
'20200527': 8180523.0,
'20200528': 8191667.0,
'20200529': 8423076.0,
'20200530': 8416337.0,
'20200531': 8216848.0,
'20200601': 8226700.0,
'20200602': 8222319.0,
'20200603': 8226169.0,
'20200604': 8229121.0,
'20200605': 8449018.0,
'20200606': 8459009.0,
'20200607': 8226173.0,
'20200608': 8221541.0,
'20200609': 8217699.0,
'20200610': 8218600.0,
'20200611': 8225627.0,
'20200612': 8445414.0,
'20200613': 8453293.0,
'20200614': 8219563.0,
'20200615': 8207525.0,
'20200616': 8172199.0,
'20200617': 8178614.0,
'20200618': 8186103.0,
'20200619': 8408628.0,
'20200620': 8419472.0,
'20200621': 8184346.0,
'20200622': 8166680.0,
'20200623': 8164307.0,
'20200624': 8165509.0,
'20200625': 8170921.0,
'20200626': 8413140.0,
'20200627': 8420824.0,
'20200628': 8179649.0,
'20200629': 8147380.0,
'20200630': 8143038.0,
```

```
 '20200701': 8226484.0,
 '20200702': 8235179.0,
 '20200703': 8467860.0,
 '20200704': 8478685.0,
 '20200705': 8142857.0,
 '20200706': 8072829.0,
 '20200707': 1831370.0}
```

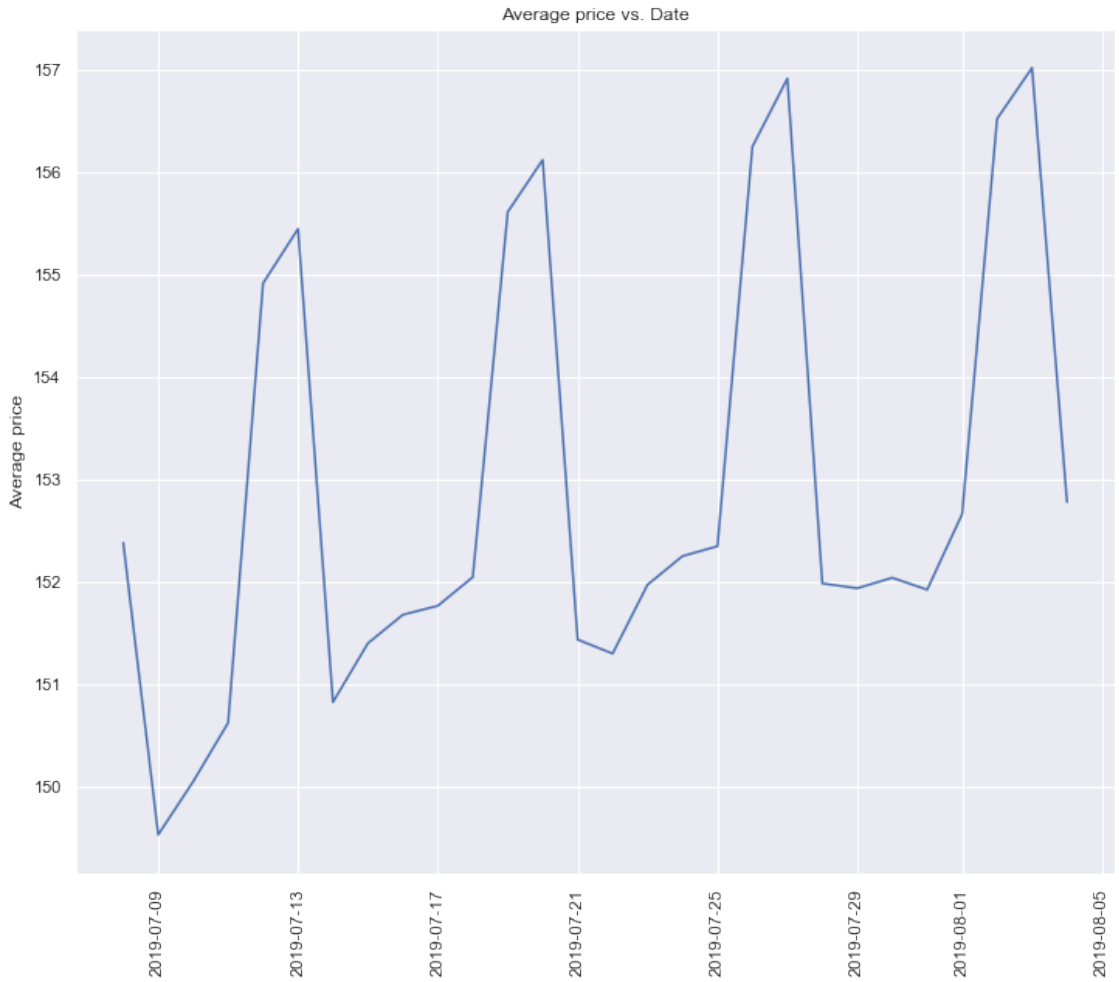
[29]:
```
# Next, plot the results.
tempcount = sorted(rcount.items())
x, y = zip(*tempcount)
tempprice = sorted(rprice.items())
u,v = zip(*tempprice)

# plt.plot(x, y)
# plt.show()
```

[30]:
```
# Next, we look at average price

tempprice = sorted(rprice.items())
u,v = zip(*tempprice)

ratio = lambda a,b: float(a)/float(b)

avgprice = list(map(ratio, v,y))

xd = pd.to_datetime(x)
```

[31]:
```
plt.figure(figsize=(12,10))
plt.plot(xd,avgprice)
plt.xticks(rotation = 'vertical')
plt.ylabel('Average price')
plt.title("Average price vs. Date")
plt.show()
```



[32]:
```
#Let us also plot a smaller time interval
plt.figure(figsize=(12,10))
plt.plot(xd[0:28],avgprice[0:28])
plt.xticks(rotation = 'vertical')
plt.ylabel('Average price')
plt.title("Average price vs. Date")
plt.show()
```



When analyzing time series data like this, it is common to view it as a sum of several contributing effects over time plus noise. The two common types of summands in such a representation are:

1. **Seasonal effects**: this is a summand that is periodic, often with period corresponding to the calendar (week, month or year).
2. **Trend effects**: this is a smooth summand that goes up or down slowly over an entire series, representing long-term trends such as price inflation.

1.7.1 Exercise 8:

8.1 Visually, can you see any strong seasonal or trend components? What do they mean?

Answer.

1. There is an extremely strong cyclical component that repeats every week. This corresponds to the fact that weekend travel to NYC is very different from weekday travel.
2. There is a trend of increasing prices over time.

3. Calendar components: this is a component with sharp "spikes" that is designed to correct for any idiosyncratic elements of our calendar. This might include: (i) a monthly time series with a dip in February (since it is the shortest month); (ii) spikes in months that contain five Saturdays (since there may be more spending on weekends); or (iii) a daily time series with a dip on Labor Day (when stores are closed).

8.2 What is the enormous spike that you see in this chart? Is it real, and how would you describe what is going on in layman's terms?

Answer. This spike occurs at Christmas, the busiest holiday season. We expect it every year and must incorporate it in any reasonable model.

8.3 Can we guess the busiest season (excluding Christmas) from this raw chart?

Answer. This would be difficult. Notice that this chart covers about a year, but there is a clear discontinuity if you try to "wrap" the data (i.e. the difference between the first and last day on this chart is significant). This is caused by an underlying trend of increasing prices every year. To figure out the best season, you would need to extract out this trend, which is difficult to do from a single year's data in a single city.

This brings us to an important topic: bringing in auxiliary datasets! The Inside Airbnb website includes calendar data for many cities, and we can use these to adjust for the trend component. To get some diversity, we should make sure to source some data from: (i) a city close to NYC; (ii) a city in the US with very different weather; and (iii) some cities very far away.

1.8 Conclusions (10 mts)

In this case, we saw that Airbnb prices are influenced by many factors. Some of the main ones include location, date, number of bedrooms, number of guests, and property type.

Any future model we build should feature these factors. Incorporating some of these factors, such as the number of bedrooms, should be straightforward, as this has a large and nearly linear relationship to price. But others, such as location, exhibit very non-linear relationships. We will learn how to deal with these types of complex relationships in future cases.

We also found some surprising correlations, such as the negative correlation between price and parking. However, after breaking the data down by neighborhoods and incorporating the interaction effect of location, this negative correlation went away entirely.

Temporal effects are a very specific type of interaction effect which must be dealt with separately. Our exploration tells us that any model of AirBnB pricing should take into account strong seasonal components as well as strong spikes around major holidays.

1.9 Takeaways (5 mts)

In this case, you learned the following exploration process:

1. Start by looking at marginal distributions of quantities of interest to look for interesting patterns and/or outliers.

2. A correlation matrix can quickly reveal the most promising candidate variables for further investigation.
3. Investigate each of these candidate variables in turn. Note which ones exhibit interesting and unexpected correlations.
4. Explore potential interaction effects for the variables with unexpected correlations. Suspected important interactions should be looked at directly with further plotting.
5. Finally, take some time to carefully plot any interactions that you know to be important from domain knowledge. In our case, we looked at two features that are common to many datasets: location data and temporal data. Both of these contained very important signals that were immediately visually apparent, but which were strongly non-linear and could not easily be reduced to correlations or other simple summaries.

This process can be a bit daunting at first, but it is widely used by veteran data analysts and scientists and is extremely effective in most situations. By iteratively generating hypotheses throughout this process and investigating them, you can uncover great insight about what is going on without building a single formal model. Formal modelling will be discussed in future cases.