In [1]:
```
import re
from pandas import Series, DataFrame
import pandas as pd
from numpy.random import randn
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
# import seaborn as sns
```

First Step: Select Airbnb Homes for Research

The logic is to find the majority of all homes (80%) using normal distribution.

In [2]:
```
fields = ['id', 'accommodates', 'price', 'number_of_reviews', 'review_sc
ores_rating',
          'review_scores_accuracy', 'review_scores_cleanliness', 'review
_scores_checkin',
          'review_scores_communication', 'review_scores_location', 'revi
ew_scores_value',
          'reviews_per_month']
homes_df = pd.read_csv("./Data/listings.csv", encoding = "utf-8", usecol
s = fields)

homes_df.columns = ['home_id', 'accommodates', 'price', 'number_of_revie
ws',
                    'scores_overall', 'scores_accuracy', 'scores_cleanli
ness',
                    'scores_checkin', 'scores_communication', 'scores_lo
cation',
                    'scores_value', 'reviews_per_month']
homes_df = homes_df.dropna() # removed 660 homes which are lack of some
 score information
homes_df.head(3)
```

Out[2]:

	home_id	accommodates	price	number_of_reviews	scores_overall	scores_accuracy	scores
0	241032	4	$85.00	207	95.0	10.0	
1	953595	4	$150.00	43	96.0	10.0	
2	3308979	11	$975.00	20	97.0	10.0	

In [3]:
```
# change price from string to integer such as "$85.00" to "85"
def extract_price(s):
    return int(''.join(re.findall(r"\d+", s.split('.')[0])))

homes_df.price = [extract_price(p) for p in homes_df.price]
homes_df.head(3)
```

Out[3]:

	home_id	accommodates	price	number_of_reviews	scores_overall	scores_accuracy	scores_c
0	241032	4	85	207	95.0	10.0	
1	953595	4	150	43	96.0	10.0	
2	3308979	11	975	20	97.0	10.0	

In [4]:
```
print("There are " + str(len(homes_df)) + " airbnb homes.")
print("The price ranging from $" + str(min(homes_df.price)) + " to $"
      + str(max(homes_df.price)) + ".")
```

```
There are 3158 airbnb homes.
The price ranging from $25 to $1000.
```

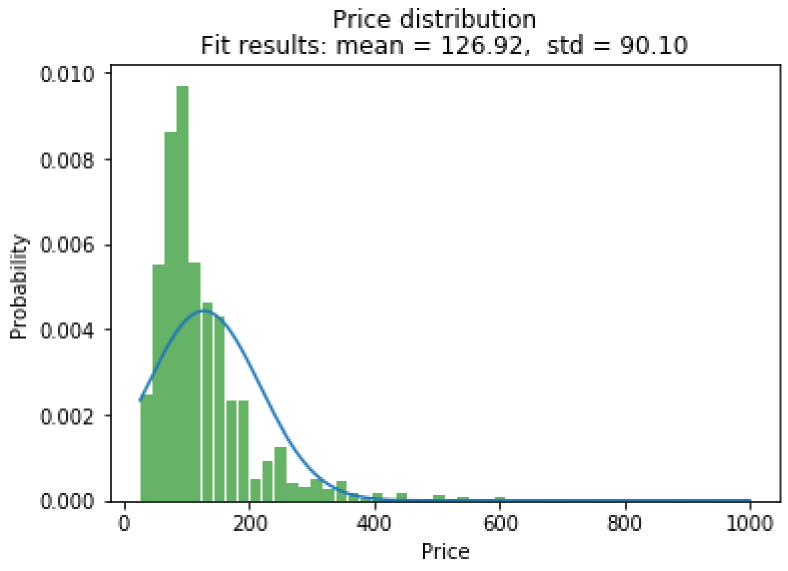
In [5]:
```
import numpy as np
from scipy import stats
from scipy.stats import norm

#正态分布的概率密度函数。可以理解成 x 是 mu（均值）和 sigma（标准差）的函数
def normfun(x,mu,sigma):
    pdf = np.exp(-((x - mu)**2)/(2*sigma**2)) / (sigma * np.sqrt(2*np.pi
))
    return pdf

def cdf(x, mean, std):
    return stats.norm.cdf(x, loc = mean, scale = std)
```

概率密度函数(P.D.F)

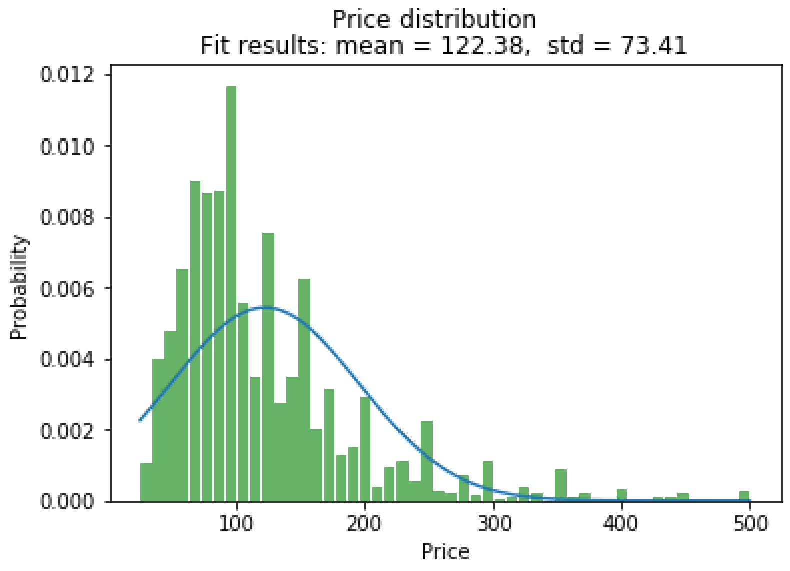
$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

In [6]:
```
def price_dtrb_hist_plot(homes_df, price_range):

    # Select data from dataframe
    data = homes_df[homes_df.price <= price_range[1]].price
    mean, std = norm.fit(data)
    # 设定 x 轴前两个数字是 x 轴的开始和结束，第三个数字表示步长，或者区间的间隔长度
    x = np.arange(price_range[0], price_range[1], 0.1)
    #设定 y 轴，载入刚才的正态分布函数
    y = normfun(x, mean, std)
    plt.plot(x,y)
    #画出直方图，最后的“normed”参数，是赋范的意思，数学概念
    plt.hist(data, bins=50, rwidth=0.9, density=True, alpha=0.6, color=
'g')
    subtitle = "Fit results: mean = %.2f,  std = %.2f" % (mean, std)
    plt.suptitle('Price distribution')
    plt.title(subtitle)
    plt.xlabel('Price')
    plt.ylabel('Probability')
    #输出
    plt.show()

    # 计算置信区间
    # 这里的0.8是置信水平
    conf_interval = stats.norm.interval(0.8, loc=mean, scale=std)
    print(conf_interval)
    prob = cdf(conf_interval[1], 126, 90) - cdf(conf_interval[0], 126, 9
0)
    print(prob)
```

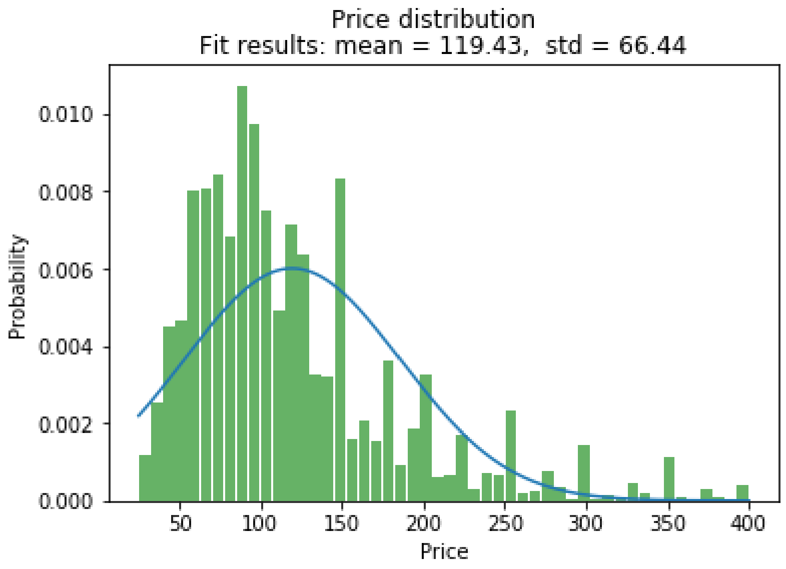
In [7]:
```
price_dtrb_hist_plot(homes_df, [25, 1000])
price_dtrb_hist_plot(homes_df, [25, 500])
price_dtrb_hist_plot(homes_df, [25, 400])
price_dtrb_hist_plot(homes_df, [25, 300])
price_dtrb_hist_plot(homes_df, [25, 200])

print()
print("Using conf_intveral and  cumulative density function, we can find
that the original distribution is a normal distrobution because the prob
is equal to the confidence_interval we set as 0.8.")
print("In this research, I want to deal with the majority of airbnb home
s. So, I will use the price range of [25, 243] to select data. The selec
ted homes will be the 80% in the price normal distribution")
```



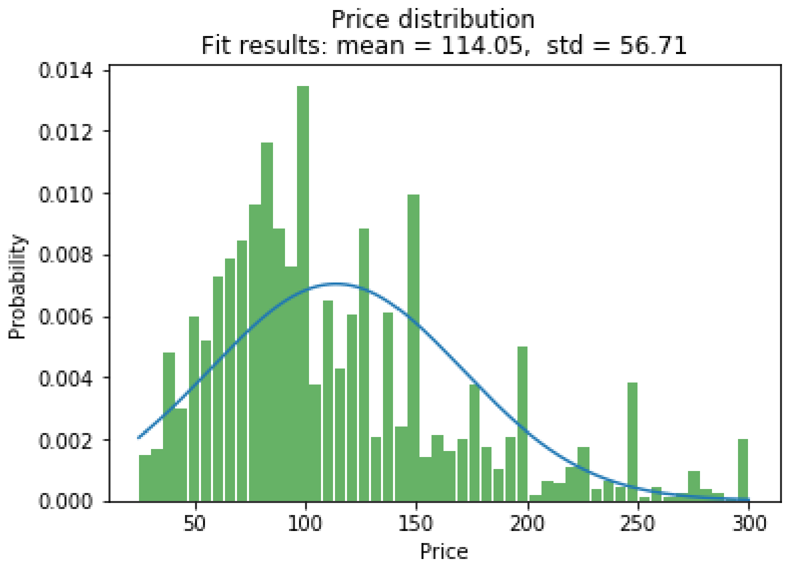
```
(11.454521584190857, 242.39348348230692)
0.8004822141897883
```



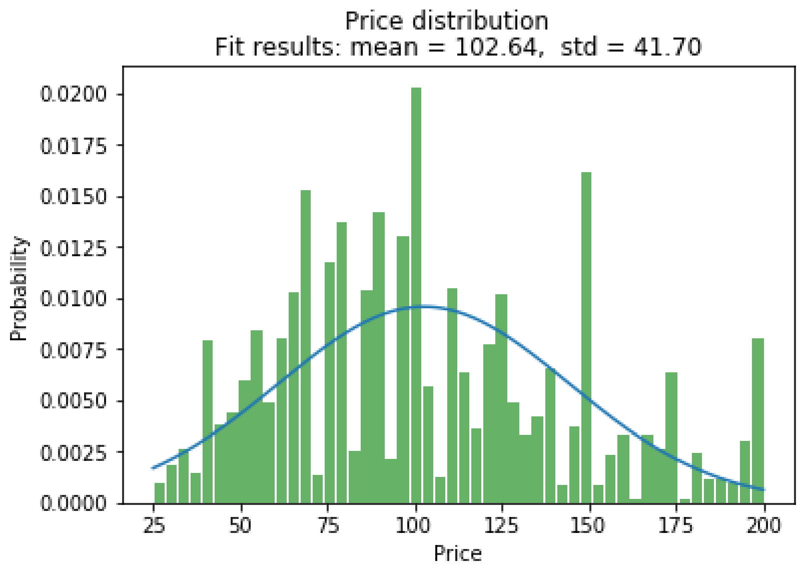
```
(28.306770830881604, 216.457991377864)
0.7037184652285017
```



```
(34.280987575449814, 204.57091197380967)
0.6545935083717964
```



```
(41.36639145927616, 186.72517639316123)
0.57656146756483
```



```
(49.18976983807186, 156.0819188445415)
0.4341963724177405
```

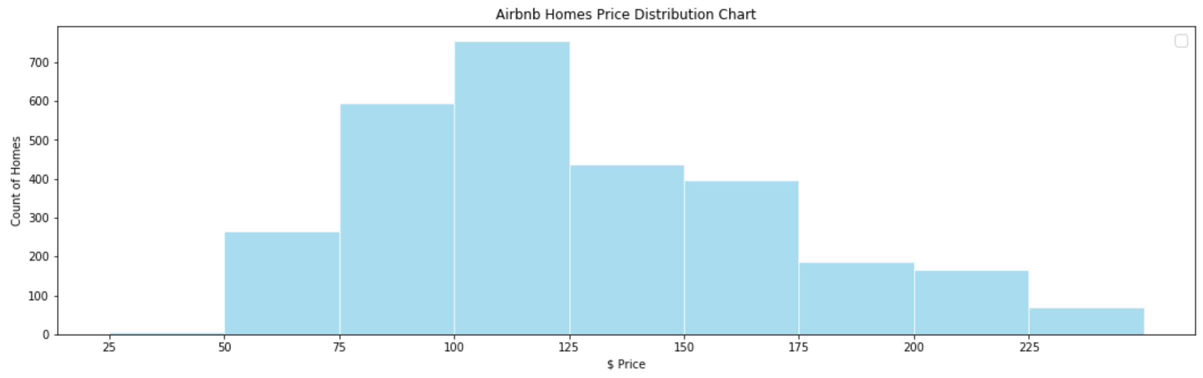
Using conf_intveral and cumulative density function, we can find that the original distribution is a normal distrobution because the prob is equal to the confidence_interval we set as 0.8.
In this research, I want to deal with the majority of airbnb homes. So, I will use the price range of [25, 243] to select data. The selected homes will be the 80% in the price normal distribution

In [8]:
```
# bar chart plot function

def price_dtrb_bar_plot(df, width):
    prices = df.index
    counts = df.num_price

    # Make the same graph
    f, ax = plt.subplots(figsize=(18,5))
    plt.bar(prices, counts, color="skyblue", width=width, linewidth = 1,
            edgecolor='white', align='edge', alpha=0.7)

    # Add titles
    plt.title("Airbnb Homes Price Distribution Chart", loc="center")
    plt.xlabel("$ Price")
    plt.ylabel("Count of Homes")
    plt.xticks(prices)
    ax.legend(fontsize = 14)

    plt.show()
```

In [9]:
```
home_price_count =  homes_df['price'].value_counts()

home_price_count_df = DataFrame(home_price_count.values,
                                index = home_price_count.index,
                                columns = ['num_price'])
home_price_count_df.index.names = ['$price']
```

In [10]:
```
# Segment the prices into several price range to plot.
# First, I tried the `$25` as price range.

def plot_vars(bin_range, max_price):
    binedges = np.arange(0, max_price, bin_range)
    binlabels = np.arange(bin_range, max_price, bin_range)
    tmp = home_price_count_df.groupby(pd.cut(home_price_count_df.index,
                                              bins = binedges,
                                              labels = binlabels)
                                      ).num_price.sum()

    tmp_df = tmp.to_frame()
    tmp_df.index.names = ['price_range']
#     price_dtrb_area_plot(tmp_df)
    price_dtrb_bar_plot(tmp_df, bin_range)





plot_vars(25, 243)
```

```
No handles with labels found to put in legend.
```



In [11]:
```
homes_sample_df = homes_df[homes_df['price'] > 25]
homes_sample_df = homes_sample_df[homes_sample_df['price'] <= 243]
```

In [12]:
```
homes_sample_df.to_csv('./Data/selected_data_for_research.csv', sep='\t'
, encoding='utf-8')
```

Next Step: Combine selected homes with its reviews

In [13]:
```
# Load Necessary Data: reviews
reviews_df = pd.read_csv("./Data/reviews.csv", encoding="utf-8")
reviews_df.columns = ['home_id', 'review_id', 'date', 'reviewer_id', 're
viewer_name', 'comments']
reviews_df.dropna()
reviews_df.head(2)
```

Out[13]:

	home_id	review_id	date	reviewer_id	reviewer_name	comments
0	7202016	38917982	2015-07-19	28943674	Bianca	Cute and cozy place. Perfect location to every...
1	7202016	39087409	2015-07-20	32440555	Frank	Kelly has a great room in a very central locat...

In [14]:
```
sample1_df = pd.read_csv("./Data/selected_data_for_research.csv", sep='
\t', encoding="utf-8")
sample1_df = sample1_df.drop("Unnamed: 0", axis=1)
sample1_df.head(2)
```

Out[14]:

	home_id	accommodates	price	number_of_reviews	scores_overall	scores_accuracy	scores_c
0	241032	4	85	207	95.0	10.0	
1	953595	4	150	43	96.0	10.0	

In [15]:
```
# Merge the reviews and homes in the sample data.
df1 = sample1_df[['home_id', 'scores_cleanliness', 'scores_location']]
df2 = reviews_df[['home_id', 'review_id', 'comments']]
sample1_rh_df = pd.merge(df1, df2, on="home_id")
sample1_rh_df.head(3)
# sample1_rh_df.stack()[0].comments
```

Out[15]:

	home_id	scores_cleanliness	scores_location	review_id	comments
0	241032	10.0	9.0	682061	Excellent all the way around. \r\n\r\nMaija wa...
1	241032	10.0	9.0	691712	Maija's apartment was a wonderful place to sta...
2	241032	10.0	9.0	702999	one of the most pleasant stays i've had in my ...

Last Step: Get homes and reviews data for sentence analysis

Find data for `cleanliness` and `location` aspects

Brief Summary & Necessary Functions

In [16]:
```
# sample2_rh_df is a copy of sample1_rh_df to
# in case unexpected modification for original data.
sample2_rh_df = sample1_rh_df
print('*' * 40 + '\nThere are:\n' + '-' * 40)
print(str(len(sample2_rh_df.groupby('home_id'))) + " Airbnb homes in tot
al.\n" + '-' * 40)
print(str(len(sample2_rh_df)) + " reviews in total.\n" + '-' * 40)
```

```
****************************************
There are:
----------------------------------------
2890 Airbnb homes in total.
----------------------------------------
81275 reviews in total.
----------------------------------------
```

In [17]:
```
# group by aspect scores
def overview_df(aspect):
    df1 = pd.DataFrame(sample2_rh_df.groupby([aspect])['home_id'].nuniqu
e())
    df2 = pd.DataFrame(sample2_rh_df.groupby([aspect])['review_id'].nuni
que())
    overview_df = pd.merge(df1, df2, on = aspect)
    overview_df.columns = ['number of homes', 'number of reveiws']
    return overview_df
# what makes people comments

# Input: the expecting aspect and score
# Output: the DataFrame with selected homes and their reviews informatio
n
def home_sample_review(score, aspect, overview_df):
    num_home = overview_df.loc[score].values[0]
    num_review = overview_df.loc[score].values[1]
    tmp = num_review/1500
    size = num_home/tmp
    # Generate a uniform random sample using random
    data = list(set(sample2_rh_df[sample2_rh_df[aspect] == score].home_i
d))
    sample_home = np.random.choice(data, int(size), replace=False)

    while not sample_validation(sample_home):
        sample_home = np.random.choice(data, int(size), replace=False)
        if sample_validation(sample_home):
            break
        else:
            continue

    sample_home_review = sample2_rh_df[sample2_rh_df['home_id'].isin(sam
ple_home)]
    print("Selected " + str(int(size)) + " homes among " + str(num_home)
+ " homes with score " + str(score) + ".\nSample home_ids:")
    print(sample_home)
    print("With in total: " + str(len(sample_home_review)) + ' reviews.'
)

    return sample_home_review

# This is a function is to guarantee:
# the number of reviews of the selected homes keeps around 1500.
def sample_validation(sample_home):
    sample_home_review = sample2_rh_df[sample2_rh_df['home_id'].isin(sam
ple_home)]
    return True if np.abs(len(sample_home_review) - 1500) <= 100 else Fa
lse
```

Get data for `cleanliness` aspect

Group homes with <= 7.0 cleanliness score together. Based on literature review, the airbnb review score has positive bias. So, for score equal to or less than 7, we can assume there is a negetive implication of the homes. As for homes with >= 9.0 score, randomly select sample as long as gurrantee 1500 number of reviews.

In [18]:
```
clean_overview_df = overview_df('scores_cleanliness')
clean_overview_df
```

Out[18]:

	number of homes	number of reveiws
scores_cleanliness		
3.0	1	2
4.0	4	9
5.0	5	25
6.0	29	110
7.0	34	465
8.0	169	3511
9.0	682	21338
10.0	1966	55815

In [19]:
```
# number of review selection around 1500.
clean_score7_df = sample2_rh_df[sample2_rh_df['scores_cleanliness'] <=
7.0]
print('Selected all homes with score no more than 7.0. \nSample home_id
s:')
print(list(set(clean_score7_df.home_id)))
print("With in total: " + str(len(clean_score7_df)) + ' reviews.')
print('*' * 90)
clean_score8_df = home_sample_review(8.0, 'scores_cleanliness', clean_ov
erview_df)
print('*' * 90)
clean_score9_df = home_sample_review(9.0, 'scores_cleanliness', clean_ov
erview_df)
print('*' * 90)
clean_score10_df = home_sample_review(10.0, 'scores_cleanliness', clean_
overview_df)
print('*' * 90)

clean_score7_df.to_csv('./Data/clean_score7_df.csv', sep='\t', encoding=
'utf-8')
clean_score8_df.to_csv('./Data/clean_score8_df.csv', sep='\t', encoding=
'utf-8')
clean_score9_df.to_csv('./Data/clean_score9_df.csv', sep='\t', encoding=
'utf-8')
clean_score10_df.to_csv('./Data/clean_score10_df.csv', sep='\t', encodin
g='utf-8')
```

```
Selected all homes with score no more than 7.0.
Sample home_ids:
[7965184, 3291777, 6363779, 8754180, 3770248, 1764233, 8594059, 404186
8, 3766285, 9151374, 5479566, 666897, 7245586, 7934356, 1520533, 709391
0, 639130, 3889050, 3888924, 3534364, 716829, 6250399, 613151, 9509279,
5252515, 8934054, 4951079, 5353512, 3226793, 8083242, 8555304, 299817,
6865200, 7985714, 7203765, 5261239, 6992696, 670009, 8922554, 4340410,
8050232, 2769088, 1520581, 670021, 3697351, 5639238, 4082250, 1520593,
7844444, 5126365, 9183838, 9519968, 8863714, 5078244, 6958436, 9075558,
6623079, 6959336, 7732071, 2856806, 5637990, 8067053, 4773614, 3593582,
6120046, 3424242, 6717555, 7975026, 2357110, 3052151, 30712, 3732094, 3
096191]
With in total: 611 reviews.
***********************************************************************
*******************
Selected 72 homes among 169 homes with score 8.0.
Sample home_ids:
[1549973 9532861 8391954 9331449 5062445 3916050 8483477 6705584 392557
2
 1039766 9448215 9300972 5487653 1773803 7872980 8212190 1520549 534024
2
 3594885 3732076 9186256 5931372 5473498 3263722 1815304 7828509 426404
3
 4566609 3939683 9134196  319768 7219541 8463726 4264056 6216116 480817
3
 1263470 1840671 7999692 9866461 8355276 8737284 3155785 6278181 641198
6
 6796066 4863533 3424991 4664312 1950446 7291403 4038347 9411935 814721
5
 1831338 8755762 5020861 1107845 6575407 3888986 6629657 1499596  38543
8
 4258515 7649837 7900056 9316399 8255196 1002835 4735761   15108  88227
4]
With in total: 1470 reviews.
***********************************************************************
*******************
Selected 47 homes among 682 homes with score 9.0.
Sample home_ids:
[5559643 1633025 4589654  555858 2776890 7455706 7067650  935671 498867
9
 1163345  366301 2693137 5671843 3040278  877203 5330475 3416217 516458
1
 8340819   59827  442487  685600 5415077  443736 5719631 2263643 796784
4
 4418480 6716380  226495  722537 6646894 7601333 7936712 1148517 697126
0
 7353834 5580773 6466129 6759038 7363462 3242605 7435912 3592838 915627
3
   19611  486344]
With in total: 1569 reviews.
***********************************************************************
*******************
Selected 52 homes among 1966 homes with score 10.0.
Sample home_ids:
[7922063 5744931 6750264 7402190 8472954  708774 1472532 7763613 763868
9
 4708075 3352685 1796302 6387576 7093738 5241773 7035498 5310503 836932
1
 4577542 1090307 7809455 6209191 8566242 3986793 7037291 1252147 473772
7
 4760181 8645226 1251707 6748502 6747473 8743202 5834820 2761092  95847
5
 1961671   41401 3208330 8474584 7763298 3024336 6569950 7517684 608745
1
 8926060 5900224 7418814  261912 7116795 7429207 7324993]
With in total: 1445 reviews.
**********************************************************************
******************
```

Get data for `location` aspect

Group homes with <= 7.0 cleanliness score together.

In [20]:
```
loc_overview_df = overview_df('scores_location')
loc_overview_df
```

Out[20]:

	number of homes	number of reveiws
scores_location		
4.0	1	1
6.0	7	13
7.0	19	155
8.0	121	1762
9.0	823	28732
10.0	1919	50612

In [21]:
```
# number of review selection around 1500.
loc_score7_df = sample2_rh_df[sample2_rh_df['scores_location'] <= 7.0]
print('Selected all homes with score no more than 7.0. \nSample home_id
s:')
print(list(set(loc_score7_df.home_id)))
print("With in total: " + str(len(loc_score7_df)) + ' reviews.')
print('*' * 90)
loc_score8_df = home_sample_review(8.0, 'scores_location', loc_overview_
df)
print('*' * 90)
loc_score9_df = home_sample_review(9.0, 'scores_location', loc_overview_
df)
print('*' * 90)
loc_score10_df = home_sample_review(10.0, 'scores_location', loc_overvie
w_df)
print('*' * 90)

loc_score7_df.to_csv('./Data/loc_score7_df.csv', sep='\t', encoding='utf
-8')
loc_score8_df.to_csv('./Data/loc_score8_df.csv', sep='\t', encoding='utf
-8')
loc_score9_df.to_csv('./Data/loc_score9_df.csv', sep='\t', encoding='utf
-8')
loc_score10_df.to_csv('./Data/loc_score10_df.csv', sep='\t', encoding='u
tf-8')
```

```
Selected all homes with score no more than 7.0.
Sample home_ids:
[7965184, 8061699, 190984, 4041868, 1549973, 6215199, 613151, 6250399,
4951079, 5353512, 8083242, 5792683, 7071021, 6019762, 1022135, 23356, 4
126284, 5104077, 7415378, 6701018, 9183838, 3706719, 5637990, 6959336,
7839723, 1263470, 9157232]
With in total: 169 reviews.
***********************************************************************
******************
Selected 103 homes among 121 homes with score 8.0.
Sample home_ids:
[1815472 9473312 8342968 7922663 5407311 7934356 2056276 4340410 951996
8
 1179538 6907671 4589654  385438 8988178 5126077 8799588 7902068 632528
3
 7095802  286712 1340668 1571230 9199982 1484651 8921924 3630581  61150
0
 7013085 5340242 3316219 2134911  693956 6249164 4708075 7500000 913419
6
 9532861 1566487 9300972 8168876 8952253 3449059 2586350 9507115 372639
1
 9238818  264829 1499596 2610187  609610  571640 3544964  877203 721954
1
 4130112  716829 6714817 4395654 3849918 9151374 6425652 3303857 720544
3
 7388899 4105081 7048843 6545246 5376433 5744931 1672979 4163851 385988
2
 3329962 6226666  666897 6416765 7420488 6133684 9186256 6411986 973694
0
 2800448 1905473  571651 7429207 8525825 5219336 3533112 7431247 298076
2
 4144767 7027507 4395578 5931372 3904056 1652107 2986056 8848854  69600
4
 9564093 3951768  278192 5471427]
With in total: 1488 reviews.
***********************************************************************
******************
Selected 42 homes among 823 homes with score 9.0.
Sample home_ids:
[5164581  153967 6002165 1063855 3206305 4583161 1407502 8034871  25714
0
 4082250 7561272 5793477 4258515 4061207 7664565 3732103 3889050 758568
8
 6782993 3115801 4106041 1264287  158953 1599856 4518037 8409962 342289
4
 7483750 1856970  492287 6482409 5620928  491958 6421243 5593399 771458
5
 4374326   15108 9091301 4681687  826436  443580]
With in total: 1470 reviews.
***********************************************************************
******************
Selected 56 homes among 1919 homes with score 10.0.
Sample home_ids:
[7077910 6823781 5249067 7564942 3394936 7243974 3888986 1534622 390607
6
 7455068 9615941 9151865 1790020  261912 8009814 8065531 8418650 627622
4
```

```
 3505739 4410544 4933447 6344566 7828222 3479851 7745196 6992387 612797
4
 4869458 3489083 7870538 4993710  565703 4053644 4660460 9236961 108408
4
 6835466 7561333 7953211 5067177 8473056  170273 1039766 7900497 859768
7
 7803176 9727246 6316917 8102349 4532538 8150395 2386589 9075656 801244
3
 3971137   58503]
With in total: 1466 reviews.
***********************************************************************
*******************
```

```
In [ ]:
```