Formal Analysis
Correlation and Regression

1. Introduction

The big question that drove my interest is: Can we predict a house price based on its area? In other words, I am interested in knowing how helpful is the area of the house in predicting the house price? Hence, I tried to tackle the question by exploring whether there is a correlation between the houses prices and areas. I also created a linear regression model to examine how much variation of houses prices can be predicted based on variation in houses area.

2. Dataset

I obtained this dataset from kaggle datasets. In the original dataset, there are 19 house features along with the price and the id columns, and 21613 entries . Originally, there is information about houses built from 1900 till 2015, but this report analyzes only house prices in 2014 and 2015.The variables of interest are price and area of the house. The area of the house is the predictor (independent) and quantitative variable measured in square footage unit. The house price is the response(dependent), quantitative variable measured in USD.[1]

3. Methods

I used pandas package and matplotlib to read and analyze the dataset in python. First, python was used to examine the descriptive statistics for each variable of interest. Table 1. Provides the summary statistics for area and price of houses (they are computed in Appendix A).

[1] #variables: identify variables included in the report with an explanation of the relation between dependent and independent variables.

The sample distributions for the two variables are displayed in Figures 1 and 2 (these are created in Appendix B).[2]

Table 1: Summary statistics for houses prices and areas		
	House Price (USD)	House Area (square footage)
Count	597	597
Mean	688641	2614.17
Median	599950	2640.0
Mode	1st 550000.0 2nd 635000.0 3rd 1050000.0	1690
Standard Deviation	377699	911.5
Range	3244997	5080

[2] Descriptivestats: provide a table that has the summary of the descriptive statistics including mean, median, mode, standard deviation and range. Then made conclusion based on this information.

Predicting the price of the house based on its area

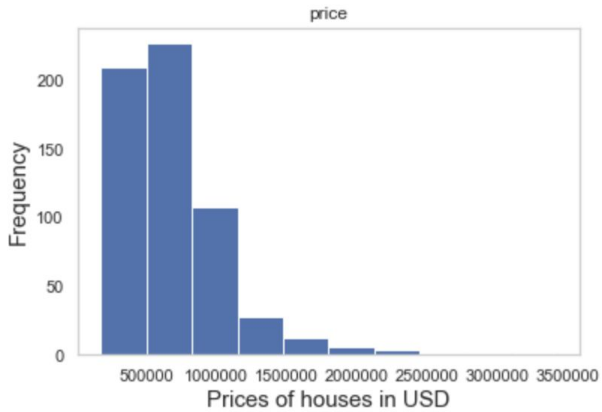


Figure 1: Histogram for Prices of Houses

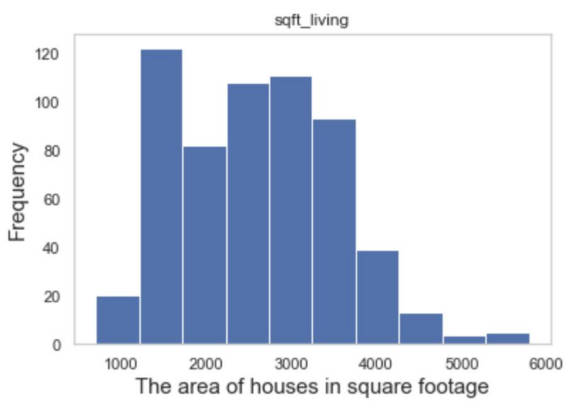


Figure 2: Histogram for Area of Houses

By looking at the mean and median values in Table 1, we find that for the house price the mean is greater than the median, and the data is skewed to the right ,which agrees with what is seen in figure 1 as the data is concentrated to the left. On the other hand, the mean and the median for the houses area are so close to each other, which means the distribution is almost symmetric with slight skewness. That skewness showed in the data might influence the effectiveness of the regression model. The standard deviation of the house prices is larger than that houses area which corresponds to more spread of data.[3]

To assess the association between the house area and its price, we will calculate pearson's correlation coefficient. Furthermore, to assess how well we can predict the house price based on its area, we will create linear regression model. But first From OpenIntro 7.2.2, we should check that the following conditions are satisfied before proceeding:

1. Linearity of data/constant variation: based on the first graph, we confirm that the data shows roughly a linear trend with almost constant variability.
2. Nearly normal residuals: Based on the histogram, the residuals almost follow a nearly normal distribution with a little skewness to the right. Also, as shown in the second graph, the residuals are almost symmetric about zero with similar degree of spread throughout the whole range. The variability is higher for the bigger house areas, but apart from that, it is approximately constant with small diversion from homoscedasticity.
3. We assume that the observations are based on independent sampling.

Based on results in Appendix C, we can interpret that:

- The pearson's correlation coefficient is = $\sqrt{0.451} = 0.672$, which shows linear, positive, and roughly from medium to strong relation between the area of the house and its price. In other words, an increase in the area of the house is accompanied by an increase in the prices of houses.[4]
- R-squared value(the coefficient of determination = 0.451), so we can say that about 45.1% of the total variation in the house price can be predicted and explained by the variation the area of the house.

[3] Dataviz: creating a data visualization using python that shows the distribution of prices and area of houses.
[4] Correlation: calculated pearson's correlation coefficient with an explanation of what we can interpret from its value.

Predicting the price of the house based on its area

- The predictive equation is (price = 278.397 * sqft_living + -39137.129). For each extra square foot in the area of the house, the price increases by 278.397 USD. Also if we assume that the area of the house is zero, then the price would be -39137.129 [5]

Hypothesis testing

I conducted hypothesis test to assess whether there is a linear relationship between the area of a house and its price or not.

H0 (null hypothesis) : there is no linear relationship between the area of the house and its price in the population; $\beta = 0$

HA (alternative hypothesis) : there is a positive correlation between the area of the house and its price in the population; $\beta > 0$

To assess statistical significance, we first compute the T-score using the following formula, $T = \frac{slope - 0}{standard\ error\ of\ the\ slope}$. To calculate the standard error of the slope, we use the following formula :SE(b1) = $\sqrt{\frac{(1-R^2)}{n-2}} * \frac{Sy}{Sx}$ .We calculate the degrees of freedom of the t-distribution by $df =$ n-2 , where n is sample size, then the t-score is converted into p-value. Finally, we compare p-value and significance level 0.05.(See Appendix D)[6]. Also, to compute the confidence interval for the slope coefficient , I used the general formula: [slope $\pm\, t_{df=n-2} \times$ SE].

---

[5] Regression: calculated coefficient of determination and explained what it says about the relation between the two variables.

[6] Significance: clearly identify significance level, tails, calculation of t and p values, beside the statistical significance with well-justified interpretation of them.

4. Results and Conclusions

The 95% confidence interval for the slope is [253.67, 303.12] (outputted in Appendix C), which means we are 95% confident that slope coefficient of the true population will be within that interval [253.67, 303.12] , which also means that if we are doing the sampling many times, 95% of the intervals would contain the true slope of the population.[7]

In Appendix D, we see the results for the test of statistical significance. The T-score of 22.11 results in a one-tailed p-value of 7.95609109433037e-80 < 0.05, which means that the result is statistically significant.

Thus we conclude that we reject the null hypothesis and it is valid that there is a positive correlation between the two variables as the slope is larger than zero according to the alternative hypothesis, which also agrees with the confidence interval.

In the conditions we assumed that the observations were sampled independently, and we still see sligh skewness in residuals graph. Thus, there might be some limitations to this model.

5. References

Kaggle. (2016). House Sales in King County, USA. Retrieved from https://www.kaggle.com/harlfoxem/housesalesprediction

[7] Confidenceintervals: applying thorough calculations of the confidence interval, and clearly interpret its meaning.

Predicting the price of the house based on its area

6. Appendix

Appendix A: Import and Analyze Data

In [3]:

```
#importing packages
import pandas
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import matplotlib
%matplotlib inline
import statsmodels.api as statsmodels #useful stats package with linear regression functions
import seaborn as sns #very nice plotting package
sns.set(color_codes=True)

#import data
filename = 'house_data.csv'
data = pandas.read_csv(filename)
data=data.dropna() # delete empty cells
data
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above
0	9.385200e+09	20150512T000000	729500.0	3.0	2.50	1660.0	1091.0	3.0	0.0	1.0	...	9.0	1530.0
1	1.832100e+09	20140625T000000	597326.0	4.0	4.00	3570.0	8250.0	2.0	0.0	0.0	...	10.0	2860.0
2	3.076501e+09	20141029T000000	385195.0	1.0	1.00	710.0	6000.0	1.5	0.0	0.0	...	6.0	710.0
3	8.077100e+09	20150422T000000	631000.0	3.0	2.25	1670.0	1396.0	2.0	0.0	0.0	...	9.0	1250.0
4	9.520900e+09	20141231T000000	614285.0	5.0	2.75	2730.0	6401.0	2.0	0.0	0.0	...	8.0	2730.0
5	1.250200e+09	20140624T000000	455000.0	2.0	1.50	1200.0	1259.0	2.0	0.0	0.0	...	8.0	1000.0
6	5.581001e+08	20150312T000000	628000.0	5.0	2.75	2600.0	8160.0	2.0	0.0	0.0	...	8.0	2600.0
7	8.156600e+09	20150326T000000	1290000.0	5.0	3.50	2980.0	5100.0	2.0	0.0	0.0	...	10.0	2370.0
8	2.770602e+09	20140826T000000	500000.0	2.0	2.25	1570.0	1269.0	2.0	0.0	0.0	...	9.0	1280.0
9	2.770602e+09	20150421T000000	671000.0	4.0	2.75	1890.0	1475.0	2.0	0.0	0.0	...	9.0	1200.0
10	9.126100e+09	20140617T000000	350000.0	3.0	2.00	1380.0	3600.0	3.0	0.0	0.0	...	8.0	1380.0
11	9.161100e+09	20150318T000000	673000.0	4.0	2.25	2580.0	2875.0	2.0	0.0	0.0	...	9.0	2580.0
12	9.126101e+09	20140801T000000	455000.0	3.0	1.75	1320.0	1014.0	3.0	0.0	0.0	...	9.0	1320.0
13	2.768101e+09	20150402T000000	649000.0	3.0	2.00	1530.0	1442.0	3.0	0.0	0.0	...	9.0	1530.0
14	7.853440e+09	20150409T000000	802945.0	5.0	3.50	4000.0	9234.0	2.0	0.0	0.0	...	9.0	4000.0
15	8.011100e+09	20150306T000000	530000.0	4.0	2.75	2740.0	7872.0	2.0	0.0	0.0	...	10.0	2740.0
16	7.853431e+09	20150127T000000	572800.0	3.0	2.50	3310.0	4682.0	2.0	0.0	0.0	...	9.0	2380.0
17	2.768101e+09	20150422T000000	659000.0	2.0	2.50	1450.0	1213.0	2.0	0.0	0.0	...	9.0	1110.0
18	7.853440e+09	20150505T000000	771005.0	5.0	4.50	4000.0	6713.0	2.0	0.0	0.0	...	9.0	4000.0
19	7.132301e+09	20150411T000000	500000.0	3.0	1.75	1530.0	825.0	3.0	0.0	0.0	...	8.0	1530.0
20	4.385700e+09	20150407T000000	1800000.0	4.0	3.50	3480.0	4000.0	2.0	0.0	0.0	...	9.0	2460.0
21	9.103000e+09	20150424T000000	920000.0	4.0	3.25	2190.0	4265.0	2.0	0.0	0.0	...	9.0	1540.0
22	8.691440e+09	20150202T000000	1290000.0	5.0	4.00	4360.0	8030.0	2.0	0.0	0.0	...	10.0	4360.0

Predicting the price of the house based on its area

click to scroll output; double click to hide

```
ribe() # descriptive stats
```

Out[86]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_abo
count	5.970000e+02	5.970000e+02	597.000000	597.000000	597.000000	597.000000	597.000000	597.000000	597.000000	597.0	597.000000	597.0000
mean	4.531750e+09	6.886416e+05	3.720268	2.747487	2614.175879	5923.112228	2.132328	0.003350	0.110553	3.0	8.582915	2433.9798
std	3.013101e+09	3.776990e+05	0.839747	0.598686	911.529269	9867.967086	0.392988	0.057831	0.541412	0.0	0.838609	885.3634
min	7.600065e+06	1.750030e+05	1.000000	1.000000	710.000000	638.000000	1.000000	0.000000	0.000000	3.0	6.000000	710.0000
25%	1.926059e+09	4.325000e+05	3.000000	2.500000	1800.000000	2250.000000	2.000000	0.000000	0.000000	3.0	8.000000	1658.0000
50%	3.797002e+09	5.999500e+05	4.000000	2.500000	2640.000000	4911.000000	2.000000	0.000000	0.000000	3.0	9.000000	2475.0000
75%	7.299600e+09	8.399900e+05	4.000000	3.250000	3266.000000	6565.000000	2.000000	0.000000	0.000000	3.0	9.000000	3080.0000
max	9.834201e+09	3.420000e+06	6.000000	5.000000	5790.000000	144619.000000	3.000000	1.000000	4.000000	3.0	11.000000	5450.0000

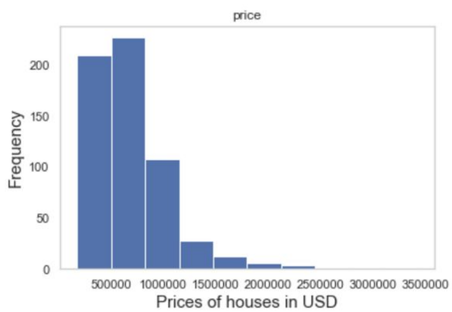
```
In [87]: data['price'].describe() # descriptive stats for price

Out[87]: count    5.970000e+02
         mean     6.886416e+05
         std      3.776990e+05
         min      1.750030e+05
         25%      4.325000e+05
         50%      5.999500e+05
         75%      8.399900e+05
         max      3.420000e+06
         Name: price, dtype: float64

In [103]: print(" the median of the houses prices is ", data.loc[:,"price"].median()) # print median

           the median of the houses prices is  599950.0

In [104]: print(" the median of the houses prices is ", data.loc[:,"price"].mode()) #print mode

           the median of the houses prices is  0     550000.0
          1     635000.0
          2    1050000.0
          dtype: float64
```

```
In [89]: data['sqft_living'].describe() #descriptive stats for house areas

Out[89]: count     597.000000
         mean     2614.175879
         std       911.529269
         min       710.000000
         25%      1800.000000
         50%      2640.000000
         75%      3266.000000
         max      5790.000000
         Name: sqft_living, dtype: float64

In [101]: print(" the median of the houses area is", data.loc[:,"sqft_living"].median()) #print median

           the median of the houses area is 2640.0
           the mode of the houses area is 0    1690.0
          dtype: float64

In [102]: print(" the mode of the houses area is", data.loc[:,"sqft_living"].mode()) #print mode

           the mode of the houses area is 0    1690.0
          dtype: float64
```

Appendix B: Visualize Data

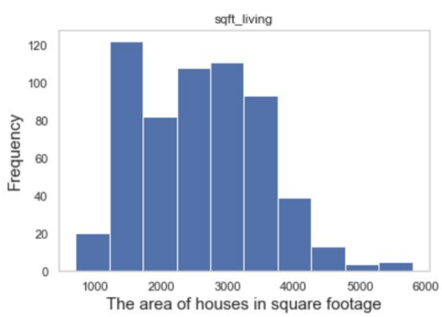
In [106]:
```
import matplotlib.pyplot as plt # importing the library
fig=plt.figure(figsize=(17,10)) #adjusting the size of the histogram
data.hist(column='price') # identifying which column for the data
plt.xlabel('Prices of houses in USD', fontsize=15) #labelling the x-axis
plt.ylabel('Frequency', fontsize=15) #labelling the y-axis
plt.grid(False) #hide gridlines
```

```
<Figure size 1224x720 with 0 Axes>
```



In [105]:
```
import matplotlib.pyplot as plt # importing the library
fig=plt.figure(figsize=(17,10)) #adjusting the size of the histogram
data.hist(column='sqft_living') # identifying which column for the data
plt.xlabel('The area of houses in square footage', fontsize=15) #labelling the x-axis
plt.ylabel('Frequency', fontsize=15) #labelling the y-axis
plt.grid(False) #hide gridlines
```

```
<Figure size 1224x720 with 0 Axes>
```

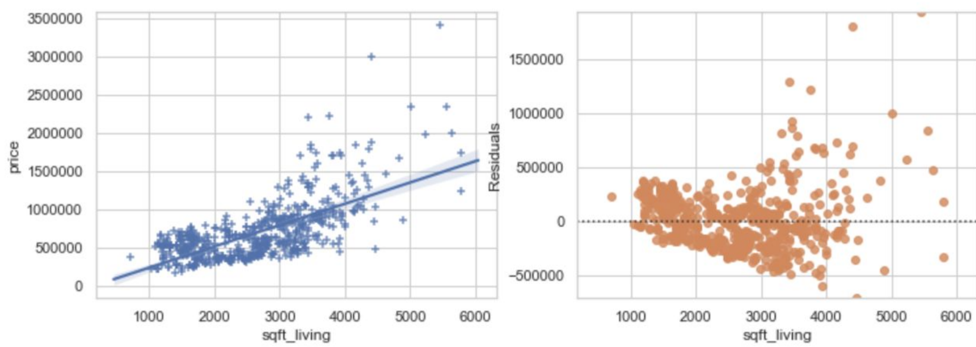


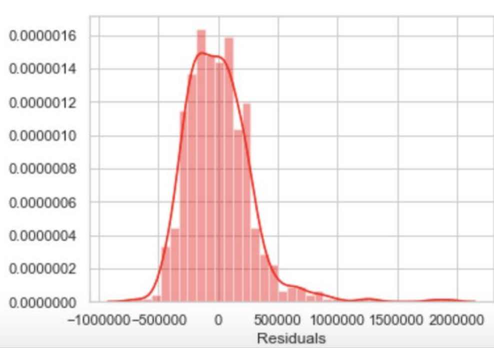
Predicting the price of the house based on its area

Appendix C: Regression model

In [5]:
```
def regression_model(column_x, column_y): # define a function


    #fit the regression line using "statsmodels" library:
    X = statsmodels.add_constant(data[column_x])
    Y = data[column_y]
    regressionmodel = statsmodels.OLS(Y,X).fit() # "ordinary least squares"

    #extract regression parameters from model, rounded to 3 decimal places:
    Rsquared = round(regressionmodel.rsquared,3)
    slope = round(regressionmodel.params[1],3)
    intercept = round(regressionmodel.params[0],3)

    #make plots:
    sns.set_style("whitegrid")
    fig, (ax1, ax2) = plt.subplots(ncols=2, sharex=True, figsize=(12,4))
    sns.regplot(x=column_x, y=column_y, data=data, marker="+", ax=ax1) #scatter plot
    sns.residplot(x=column_x, y=column_y, data=data, ax=ax2) #residual plot
    ax2.set(ylabel='Residuals')
    ax2.set_ylim(min(regressionmodel.resid)-1,max(regressionmodel.resid)+1)
    plt.figure(figsize=(5.5,4)) #histogram
    sns.distplot(regressionmodel.resid, kde=True, axlabel='Residuals', color='red') #histogram

    #print R-squared and the regression equation
    print("R-squared = ",Rsquared)
    print("Regression equation: "+column_y+" = ",slope,"* "+column_x+" + ",intercept)
```

```
R-squared =  0.451
Regression equation: price =  278.397 * sqft_living +  -39137.129
```





Appendix D: Confidence interval

In [79]:
```
R =  0.451 # R-squared value
n= 597 #sample size
Sy = 3.776990e+05 #standard deviation of house price
Sx = 911.529269  #standard deviation of house area without basement
SE = ((1-R)/(n-2))**0.5 *(Sy/Sx) #formula to calculate standard error of slope

print(SE) # print standard error of slope
```

```
12.586439889264593
```

In [80]:
```
slope = 278.397 #slope from the equation
T = (slope - 0)/SE #calculating
print(T)
```

```
22.118804240860385
```

In [83]:
```

pvalue= stats.t.sf(22.11,595) #calculating the p-value using the T-score and degree of fredoom
print(pvalue)
```

```
7.95609109433037e-80
```

In [82]:
```

t = stats.t.ppf (1 - 0.025, df) #calculating t
def confidence_interval(slope, t, SE): #defining a function that calculate confidence interval
    lowbound= slope - t*SE
    highbound= slope + t*SE

    return lowbound, highbound
print(confidence_interval(slope, t, SE))
```

```
(253.67766353548518, 303.1163364645148)
```