The Relationship Between the Price of California House, its Location and Structure.

Author: Aichen Liu

[118]:
```python
import pandas as pd
import qeds
%matplotlib inline
# activate plot theme
import seaborn as sns
from IPython.display import display
import matplotlib.pyplot as plt
from scipy.stats import norm
import matplotlib.mlab as mlab
import geopandas as gpd
import numpy as np
from shapely.geometry import Point
import qeds
qeds.themes.mpl_style();
import requests
from bs4 import BeautifulSoup
from urllib.request import urlopen
import re
```

3.1 Introduction

California has been experiencing an extended and increasing housing shortage. Many realtors and potential buyers are interested in how and why the housing price has fluctuated in the past. There are lots of factors that affect housing prices. This research explores how housing prices can be affected by the structure of the house, the size of the house, the age of the house, the location of the house, population density, and the potential buyer's income.

The data downloaded from Kaggle is originally from Aurélien Géron's recent book 'Hands-On Machine learning with Scikit-Learn and TensorFlow'. The data contains 20640 observations and the information on the following variables in 1990 California: longitude latitude housing_median_age total_rooms total_bedrooms population of people residing within a block households median_income median_house_value ocean_proximity. The sampling unit is block.

To further analyze the research question, set median_house_value to be the dependent variable

Y, and house_median_age, total_rooms, population, ocean_proximity, longitude, latitude and median_income to be the independent variables Xi.

The below codes give a glimpse of the data and the type of object in each data cell.

[119]:
```
data = pd.read_csv("housing.csv")
data.head()
```

[119]:
```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
```

[120]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

3.2 Data Cleaning

The data used is pretty tidy. As indicated below, total_bedrooms is the only variable with missing values. Since it is reasonable that there may exist houses without bedrooms, such as commercial real estate. I decided to replace missing values with 0. The detailed code is shown below.

[121]:
```
data.isnull().sum()
```

[121]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

[122]:
```
data.fillna(value=0, axis=1, inplace=True)
```

[123]:
```
data.isnull().sum()
```

[123]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

On the other hand, Ocean_proximity is the only categorical variable in the data set. Having a closer look into this variable, we can see from the following table, ocean_proximity has five levels: <1H OCEAN, INLAND, NEAR OCEAN, NEAR BAY, ISLAND. To make further analysis easier each level is turned into an individual binary variable.

[124]:
```
data.ocean_proximity.value_counts()
```

[124]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

[125]:
```
data['<1H Ocean'] = np.where(data['ocean_proximity']== '<1H OCEAN', 1, 0)
data['Inland'] = np.where(data['ocean_proximity']== 'INLAND', 1, 0)
data['NEAR OCEAN'] = np.where(data['ocean_proximity']== 'NEAR OCEAN', 1, 0)
data['NEAR BAY '] = np.where(data['ocean_proximity']== 'NEAR BAY ', 1, 0)
data['Island '] = np.where(data['ocean_proximity']== 'Island ', 1, 0)
```

```
data.info()
```

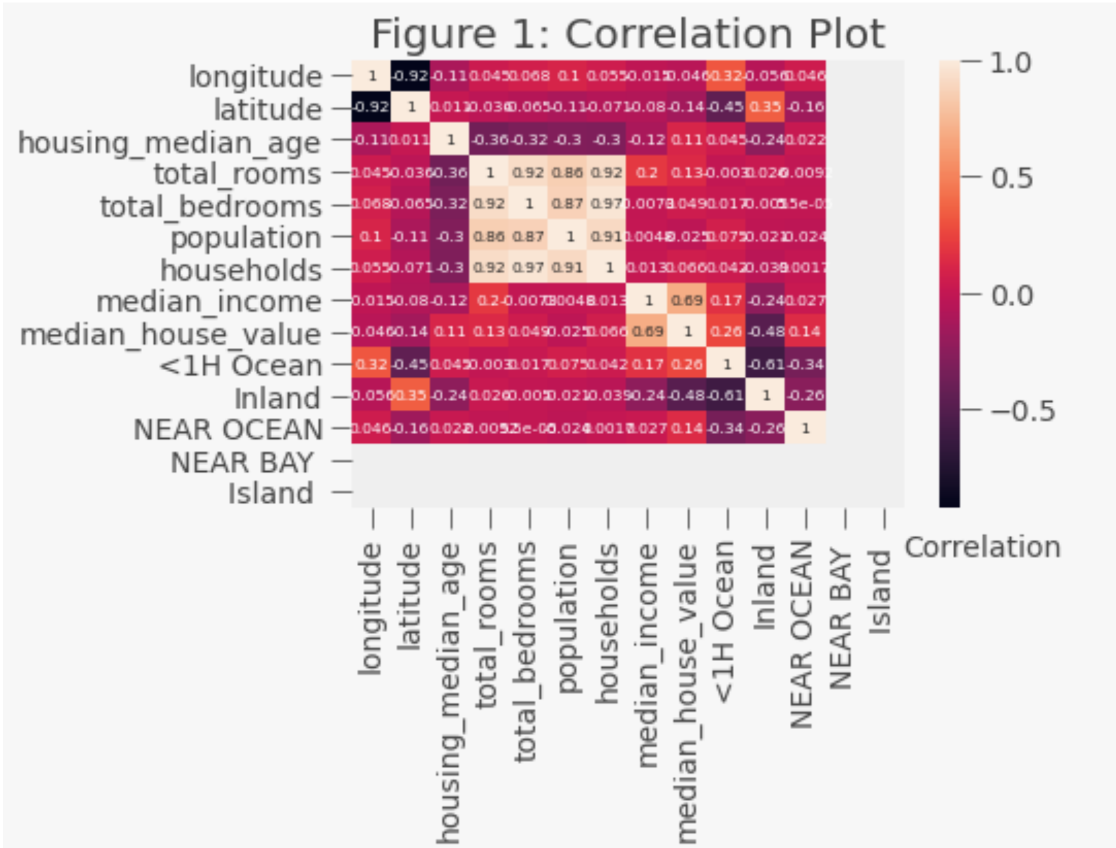
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20640 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
 10  <1H Ocean           20640 non-null  int64
 11  Inland              20640 non-null  int64
 12  NEAR OCEAN          20640 non-null  int64
 13  NEAR BAY            20640 non-null  int64
 14  Island              20640 non-null  int64
dtypes: float64(9), int64(5), object(1)
memory usage: 2.4+ MB
```

After creating the new variables, a heatmap of the correlation between each variable is shown below in figure 1. If the colour is close to black, it means that the correlation is close to negative one; If the colour is close to coral, it means that the correlation is close to one.

The table below summarizes the linear correlation between each variable and median_house_value. As indicated, there is a strong positive correlation between median_house_price and median_income, which is 0.68. On the other hand, median_house_price has a weakly negative correlation between latitude, longitude and population.

[126]:
```
sns.heatmap(data.corr(),annot=True,annot_kws={"size":7.5})
plt.annotate('Correlation',xy=(0.8, 0.35),  xycoords='figure fraction')
plt.title('Figure 1: Correlation Plot')
```

[126]:
```
Text(0.5, 1.0, 'Figure 1: Correlation Plot')
```



[127]:
```python
corr_matrix=data.corr()
corr_matrix.median_house_value.sort_values(ascending=False)
```

[127]:
```
median_house_value    1.000000
median_income         0.688075
<1H Ocean             0.256617
NEAR OCEAN            0.141862
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049148
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Inland               -0.484859
NEAR BAY                   NaN
Island                     NaN
Name: median_house_value, dtype: float64
```

3.3 Summary Statistics and Plots

To further explore the structure of data, A summary table of statistics for Y and Xi is shown below. It contains standard deviations, min and max values, and percentiles, which give us a better understanding of the range and the distribution of each variable.

[128]:
```
data[["housing_median_age", "total_rooms",
      "population","median_income","median_house_value", "ocean_proximity"]].
 ↪describe()
```

[128]:
```
       housing_median_age   total_rooms    population  median_income  \
count        20640.000000  20640.000000  20640.000000   20640.000000
mean            28.639486   2635.763081   1425.476744       3.870671
std             12.585558   2181.615252   1132.462122       1.899822
min              1.000000      2.000000      3.000000       0.499900
25%             18.000000   1447.750000    787.000000       2.563400
50%             29.000000   2127.000000   1166.000000       3.534800
75%             37.000000   3148.000000   1725.000000       4.743250
max             52.000000  39320.000000  35682.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

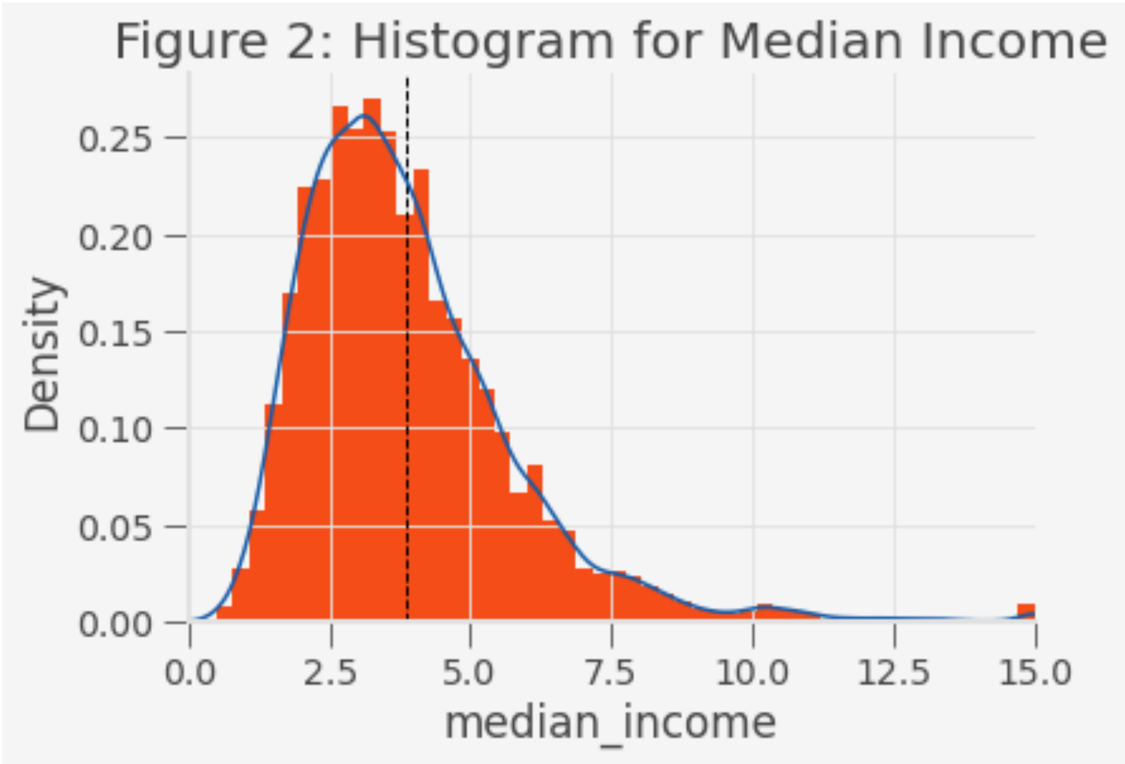
Histograms provide better visualization for the distribution of variables. For each histogram below, the vertical dashed line indicates the mean and the solid curve illustrates the density line.

Figures 2, 3 and 4 each represent the density plot for Median_income, total_rooms and housing_median_age. Figure 5 represents the frequency plot of the median_house_value. Median_income(Figure 2), total_rooms(Figure 3) are right-skewed, whereas housing_median_age(Figure 4) is multimodal with some extreme values at the right tail. median_house_value(Figure 5) is also right-skewed with some extreme values at the right tail.

[129]:
```
fig, ax = plt.subplots()
data.plot(
    kind="hist", y="median_income", color=(244/255, 77/255, 24/255),
    bins=50, legend=False, density=True, ax=ax
)
plt.axvline(data["median_income"].mean(), color='k', linestyle='dashed',␣
 ↪linewidth=1)
ax.set_facecolor((0.96, 0.96, 0.96))
fig.set_facecolor((0.96, 0.96, 0.96))
plt.xlabel('median_income')
```

```
data["median_income"].plot.density(xlim=(0, 15))

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.set_title("Figure 2: Histogram for Median Income")
```

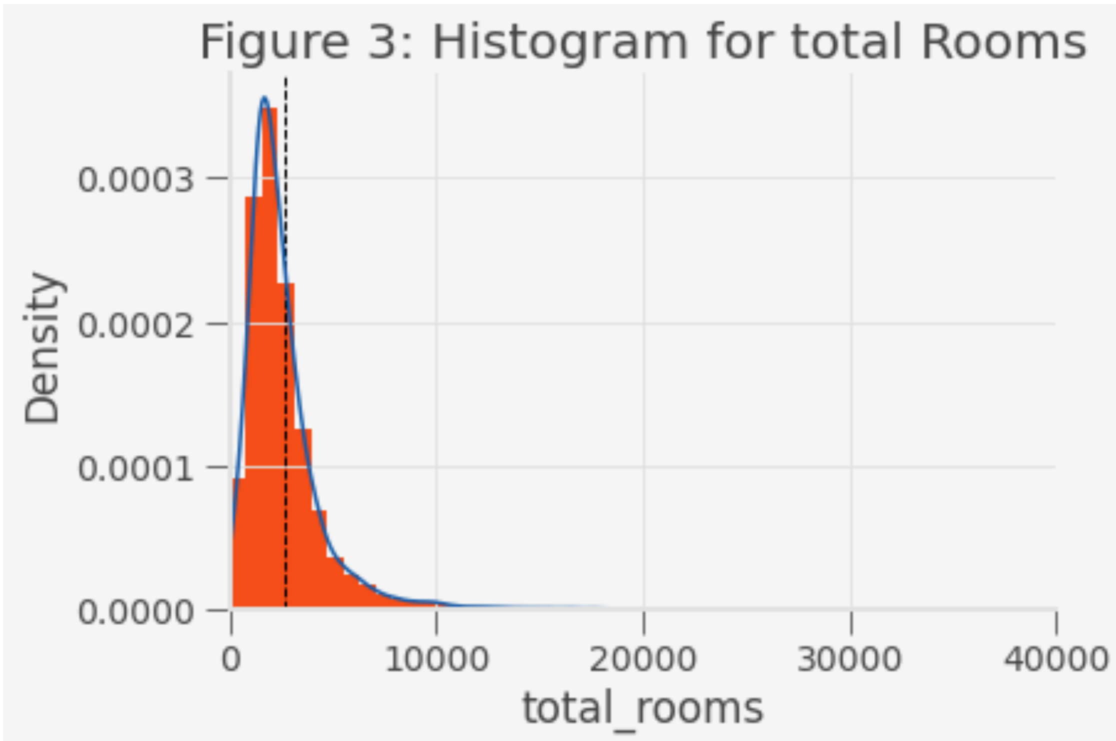
[129]: Text(0.5, 1.0, 'Figure 2: Histogram for Median Income')



[130]:
```
fig, ax = plt.subplots()
data.plot(
    kind="hist", y="total_rooms", bins=50,color=(244/255, 77/255, 24/255),
    legend=False, density=True, ax=ax
)
plt.axvline(data["total_rooms"].mean(), color='k', linestyle='dashed',␣
 ↪linewidth=1)
ax.set_facecolor((0.96, 0.96, 0.96))
fig.set_facecolor((0.96, 0.96, 0.96))

plt.xlabel('total_rooms')
data["total_rooms"].plot.density(xlim=(0, 40000))

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
```

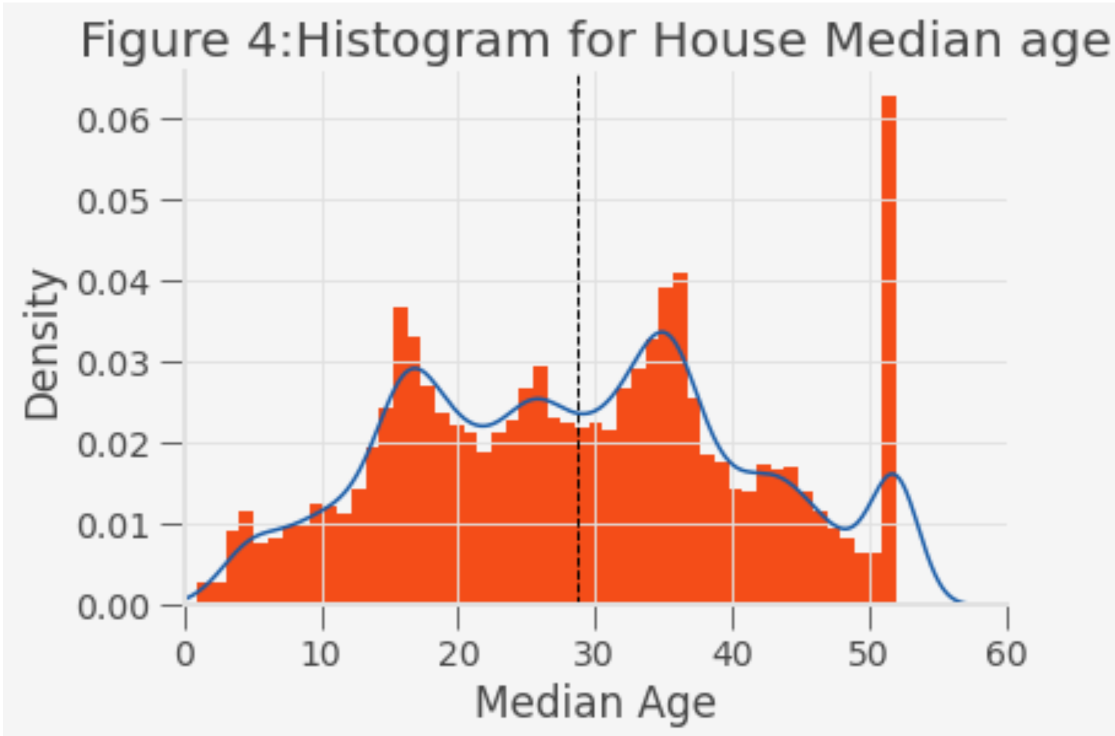
```python
ax.set_title("Figure 3: Histogram for total Rooms")
```

[130]: Text(0.5, 1.0, 'Figure 3: Histogram for total Rooms')



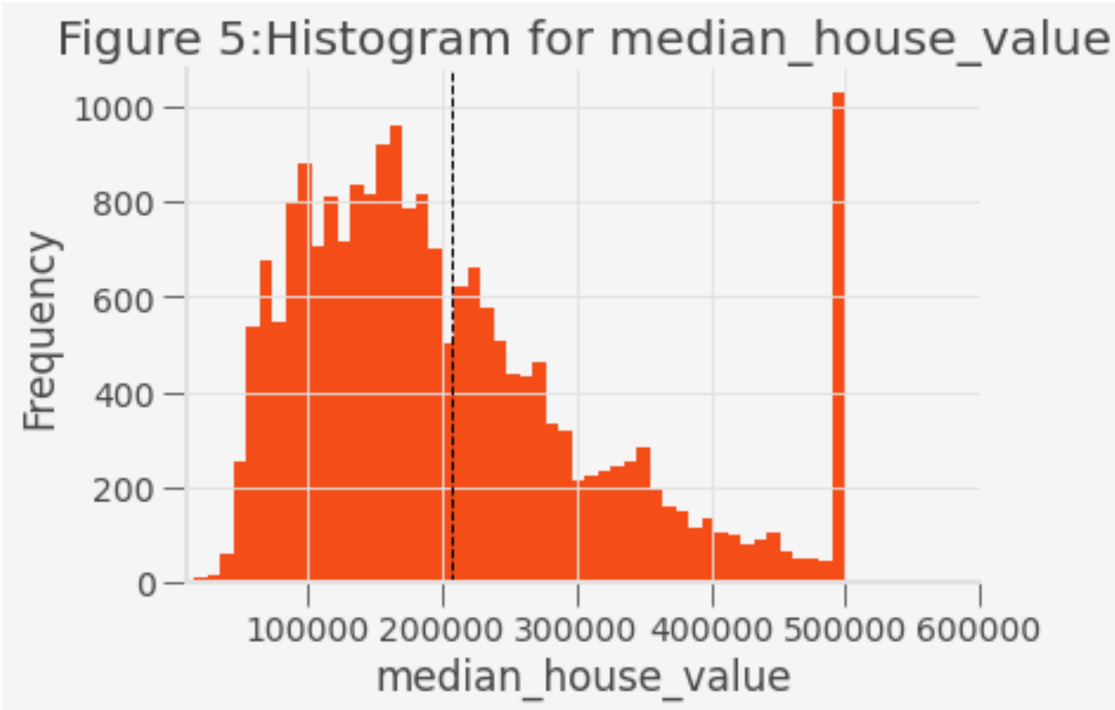
[131]:
```python
fig, ax = plt.subplots()
data.plot(
    kind="hist", y="housing_median_age", bins=50,color=(244/255, 77/255, 24/
 ↪255),
    legend=False, density=True, ax=ax
)
plt.axvline(data["housing_median_age"].mean(), color='k', linestyle='dashed',␣
 ↪linewidth=1)
ax.set_facecolor((0.96, 0.96, 0.96))
fig.set_facecolor((0.96, 0.96, 0.96))

plt.xlabel('Median Age')
data["housing_median_age"].plot.density(xlim=(0, 60))

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.set_title("Figure 4:Histogram for House Median age")
```

[131]: Text(0.5, 1.0, 'Figure 4:Histogram for House Median age')



[132]:
```
fig, ax = plt.subplots()
data.plot(
    kind="hist", y="median_house_value", bins=50,color=(244/255, 77/255, 24/
 ↪255),xlim=(10000, 600000),
    legend=False, density=False, ax=ax
)
plt.axvline(data["median_house_value"].mean(), color='k', linestyle='dashed',␣
 ↪linewidth=1)
ax.set_facecolor((0.96, 0.96, 0.96))
fig.set_facecolor((0.96, 0.96, 0.96))

plt.xlabel('median_house_value')

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.set_title("Figure 5:Histogram for median_house_value")
```

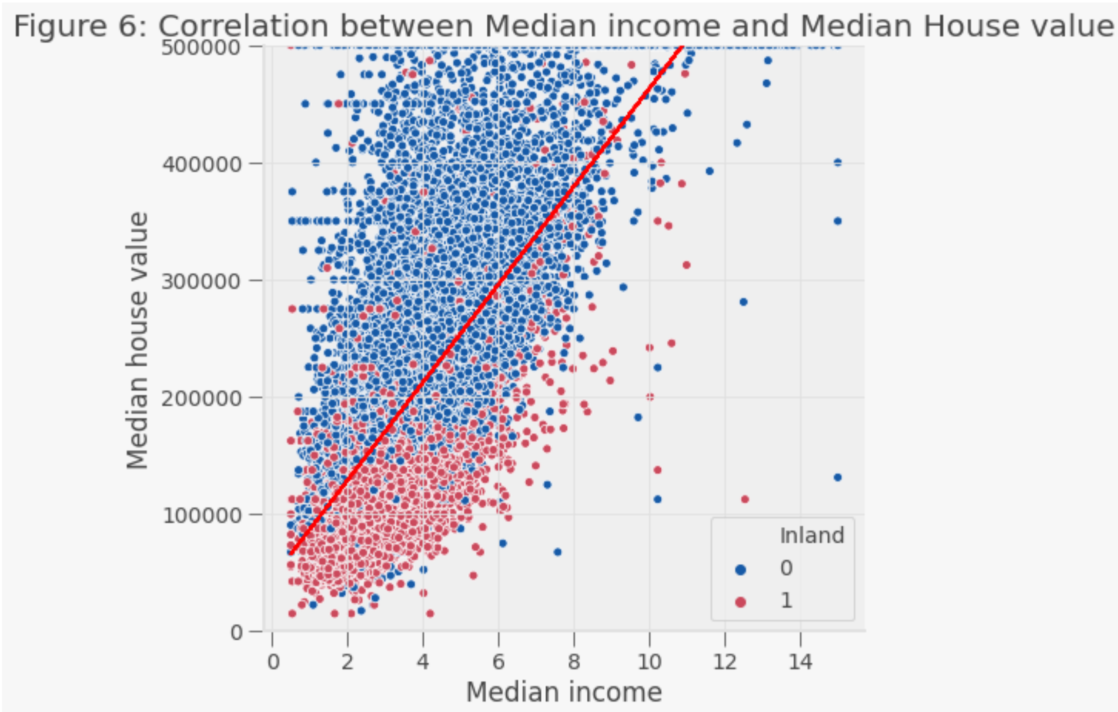
[132]:
```
Text(0.5, 1.0, 'Figure 5:Histogram for median_house_value')
```



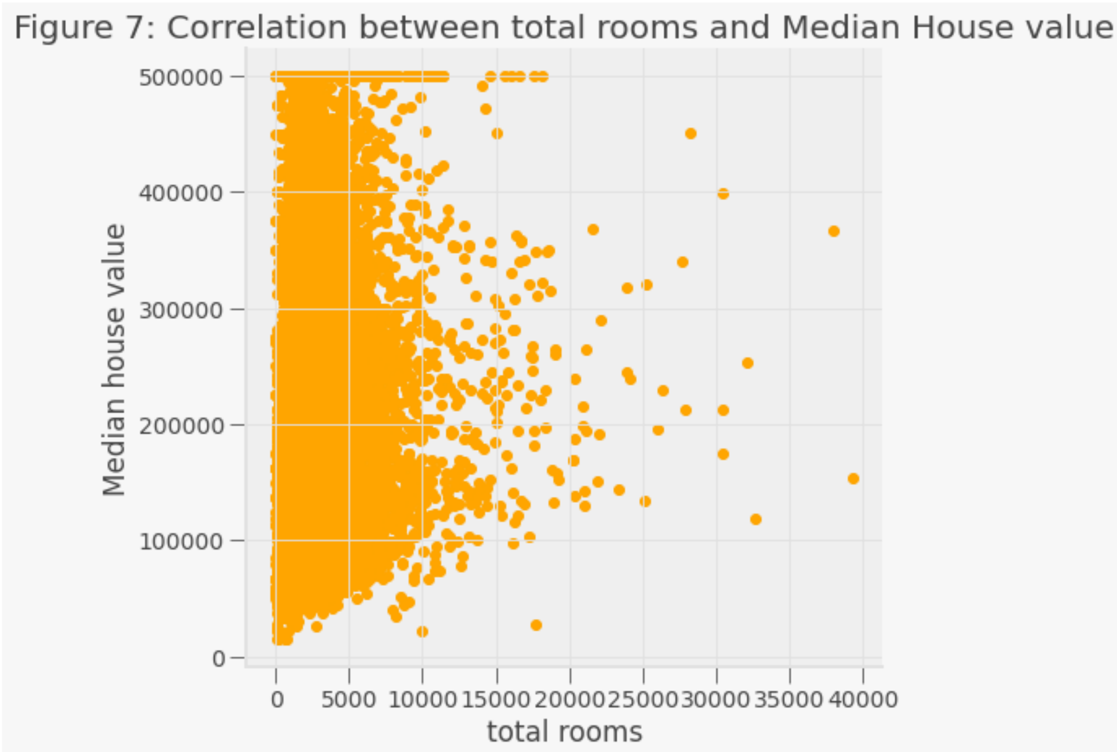
The following scatterplots are to indicate the correlation between variables. Figure 6 shows the correlation between median house value and median income, and the red line is the line of best fit. As indicated, these two variables are positively correlated, which means that people with higher incomes tend to buy a more expensive house. On the other hand, red dots indicate inland houses and blue dots represent other houses. Red dots are clutter below the line of best fit, which shows that inland houses are not as expensive as other types of houses.

Figure 7 indicates the correlation between median house value and total rooms. Most blocks have total rooms within the range of 0-10000, but the house price scatters haphazardly. Thus, there is no significant relationship between housing prices and the number of rooms.

[133]:
```
plt.figure(figsize=(7,7))
sns.scatterplot(x="median_income", y="median_house_value",hue="Inland",
              data=data);

m, b = np.polyfit(data["median_income"], data["median_house_value"], 1)
plt.ylim(0, 500000)
plt.plot(data["median_income"], m*data["median_income"] + b,color = "r")
plt.xlabel('Median income')
plt.ylabel('Median house value')
plt.title('Figure 6: Correlation between Median income and Median House value')
plt.show()
```



[134]:
```
plt.figure(figsize=(7,7))
plt.scatter(data["total_rooms"], data["median_house_value"],c='orange')
plt.xlabel('total rooms')
plt.ylabel('Median house value')
plt.title('Figure 7: Correlation between total rooms and Median House value')
plt.show()
```

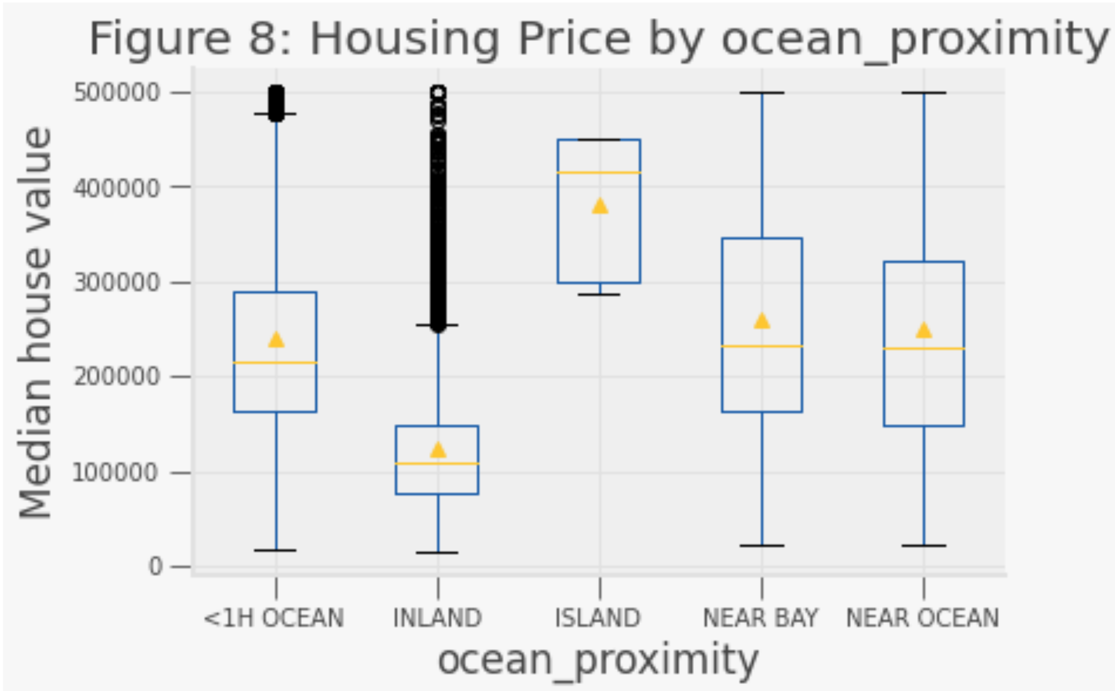


The following boxplot explains the correlation between house price and ocean_proximity. The range of IQR for inland houses is the lowest with no overlap with others, but with some outliers at the top. Therefore we can conclude that the majority of inland houses have a low price. The range of IQR for houses on the island is the highest with no overlap with others. Thus, in general, the housing price on the island is more expensive. For <1H ocean, by bay and by ocean, the distribution and range of the house price are similar.

[135]:

```
plt.figure(figsize=(8,8))
data.boxplot(column="median_house_value",␣
↪by="ocean_proximity",fontsize=10,showfliers=True, showmeans=True)

plt.ylabel('Median house value')
plt.title("Figure 8: Housing Price by ocean_proximity")

plt.suptitle("")
plt.show()
```

```
<Figure size 576x576 with 0 Axes>
```



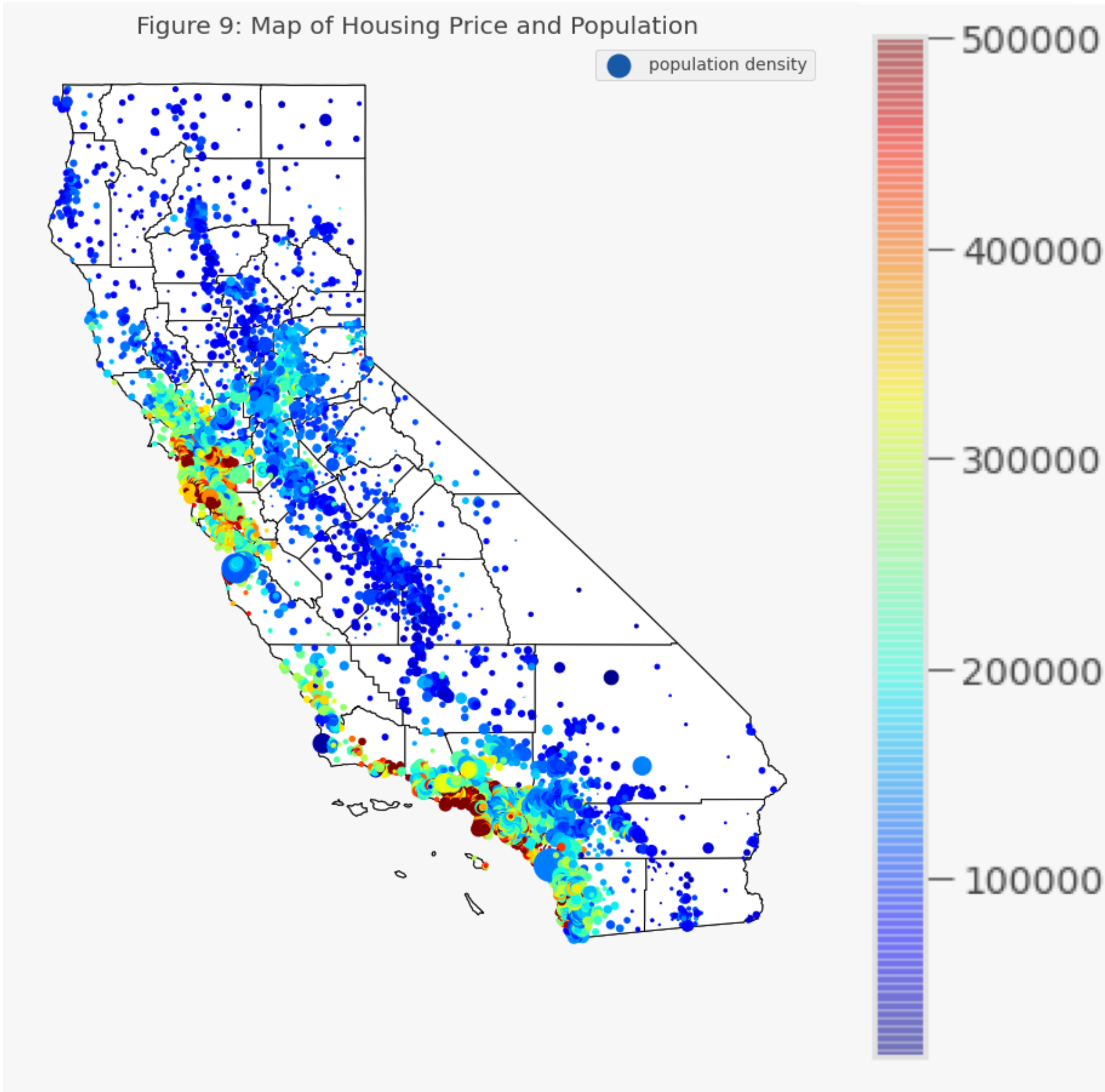
3.4 Map

Figure 9 is a map of California where each dot represents one observation in the dataset. The lower border is the shore, the upper border is the inland area. As indicated from the scale on the right, the colour of dots is corresponding to the price of houses. The more the colour is close to red and yellow, the more expensive the house is; The more the colour is close to blue and purple, the cheaper the house is. Besides, the size of the dots represents the population density of the corresponding neighbourhood. The larger the circle, the higher the population density around that area. We observed that the red dots and yellow dots gather around the shore, which means that houses by the ocean or by the bay are more expensive than houses inland. Also, the size of the circle is larger for those closer to the shore, hence we know that the population density is greater in areas close to the shore.

The majority of the large circles are in light blue, whereas the relatively small circles are in either dark blue or orange. This observation indicates that inland areas with low population density have low housing prices, whereas the near-shore areas with low population density tend to have higher housing prices. High population density doesn't imply high housing prices or low housing prices.

[136]:
```
county_df = gpd.read_file("http://www2.census.gov/geo/tiger/GENZ2016/shp/
↪cb_2016_us_county_5m.zip")
state_df = gpd.read_file("http://www2.census.gov/geo/tiger/GENZ2016/shp/
↪cb_2016_us_state_5m.zip")
data["Coordinates"] = list(zip(data.longitude, data.latitude))
data["Coordinates"] = data["Coordinates"].apply(Point)
g_data = gpd.GeoDataFrame(data, geometry="Coordinates")
```

```
county_df_C = county_df.query("STATEFP == '06'")

fig, gax = plt.subplots(figsize=(15, 15))
state_df.query("NAME == 'California'").plot(ax=gax, edgecolor="black",␣
↪color="white")
county_df_C.plot(ax=gax, edgecolor="black", color="white")

data.plot(ax=gax,kind='scatter',x='longitude', y='latitude',
         s=data['population']/50, label='population density',
         c=data['median_house_value'],
         cmap=plt.get_cmap('jet'),
         colorbar=True,
         figsize=(15,15))

gax.annotate('Housing Price',xy=(0.84, 0.06),  xycoords='figure fraction')
plt.axis('off')
plt.title("Figure 9: Map of Housing Price and Population")
plt.show()
```



3.5 Web-Scraping

Besides the structure of the houses, other factors are affecting the price of the house, such as the surrounding school districts, surrounding secondary schools rating, and the frequency of natural disasters. Since we don't have available data, we can conduct web-scarping for the two websites I found and collect the information from the scratch. Here are the addresses of the two websites. https://school-ratings.com/svg/index.html and http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_021708_Earthquakes

The first website includes an interactive map about the school-ratings of California for each of nearly 9000 public schools. The rating is determined by a school's API Score in comparison to all other schools in California where 1 is the worst and 10 is the best. I chose an image over a dataset because the datasets from the California Department of Education are not public. I think this map is very useful because I could scrape this image and compare it with the map I generated above to visualize the pattern and the correlation between the housing price and the goodness of the surrounding schools. Since the website I chose is an image, we don't need to merge or run it over time.

The scarped image is in the URL below. I have attached the image in the appendix. In this image, each dot represents one school, the greener the dots, the better the school. There are three clusters of green dots; One is on the lower shore, one is around the bay, and the other one is in the middle inland area. Compare these clusters of green dots with the housing price in figure 9, we can observe the pattern of the schools, the more expensive the housing in that neighbourhood.

[137]:
```
html = urlopen('https://school-ratings.com/svg/index.html')
imge="https://www.school-ratings.com/svg/caState2.svgz"
bs = BeautifulSoup(html, 'html.parser')
images = bs.find_all('img')
for image in images:
    print("The images are:")
    print(image['src']+'\n')
    print(imge)
```

```
The images are:
../schoolRatingsSmall.gif

https://www.school-ratings.com/svg/caState2.svgz
The images are:
http://creativecommons.org/images/public/somerights20.gif

https://www.school-ratings.com/svg/caState2.svgz
```

The second website is the SOCR Data - California Earthquake Data from the Northern California Earthquake Data Center (NCEDC) provided by the professor. This dataset contains a table of the information of the earthquakes from 1969 October to 2007 November; It contains the longitude, latitude, magnitude, depth, and the date of the earthquakes. Since both my original data and the earthquake data have longitude and latitude, I plan to merge the new earthquake data onto my original data by both longitude and latitude. After merging the data, I will generate a scatterplot of the relationship between median_house_value versus the magnitude of the earthquakes. If I were a buyer, I would take into account the environmental factors. So I hypothesized that there might be a relationship between the magnitude of the earthquakes and the housing price. The following codes show the scraping process of the earthquake data.

[138]:
```
web_url = 'http://wiki.stat.ucla.edu/socr/index.php/
 ↪SOCR_Data_021708_Earthquakes'
response = requests.get(web_url)
print('Status code\n', response.status_code)
soup_object = BeautifulSoup(response.content)
data_table = soup_object.find_all('table', 'wikitable')[1]
all_values = data_table.find_all('tr')
```

```
Status code
```

200

[139]:
```
Earthquake = pd.DataFrame(columns = ['Date', 'Time',␣
↪'latitude','longitude','Depth','Mag','Magt','Nst','Gap','Clo','RMS','SRC','EventID'])
ix = 0

for row in all_values[1:]:
    values = row.find_all('td') # Extract all elements with tag <td>
    # Pick only the text part from the <td> tag
    date = values[0].text
    time = values[1].text
    lat = float(values[2].text)
    long= float(values[3].text)
    depth = values[4].text
    mg  = float(values[5].text)
    mgt= values[6].text
    nst= values[7].text
    gap= values[8].text
    clo= values[9].text
    rms= values[10].text
    src= values[11].text
    eid= values[12].text.replace('\n','')
    Earthquake.loc[ix] = [date, time,␣
↪lat,long,depth,mg,mgt,nst,gap,clo,rms,src,eid] # Store it in the dataframe␣
↪as a row
    ix += 1

Earthquake.head()
```

[139]:

	Date	Time	latitude	longitude	Depth	Mag	Magt	Nst	\
0	1969/10/02	04:56:45.30	38.4978	-122.6640	0.22	5.6	ML	38	
1	1969/10/02	06:19:56.39	38.4500	-122.7535	5.14	5.7	ML	53	
2	1972/02/24	15:56:50.99	36.5903	-121.1905	4.18	5.1	ML	10	
3	1974/11/28	23:01:24.59	36.9202	-121.4673	5.48	5.2	ML	51	
4	1975/06/07	08:46:23.51	40.5415	-124.2763	23.48	5.3	ML	15	

	Gap	Clo	RMS	SRC	EventID
0	104	52	0.22	NCSN	-1003132
1	139	58	0.22	NCSN	-1003135
2	128	6	0.06	NCSN	-1009260
3	61	4	0.13	NCSN	-1021953
4	176	5	0.04	NCSN	-1024134

[140]:
```
Earthquake.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 168 entries, 0 to 167
Data columns (total 13 columns):
```

```
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       168 non-null    object
 1   Time       168 non-null    object
 2   latitude   168 non-null    float64
 3   longitude  168 non-null    float64
 4   Depth      168 non-null    object
 5   Mag        168 non-null    float64
 6   Magt       168 non-null    object
 7   Nst        168 non-null    object
 8   Gap        168 non-null    object
 9   Clo        168 non-null    object
 10  RMS        168 non-null    object
 11  SRC        168 non-null    object
 12  EventID    168 non-null    object
dtypes: float64(3), object(10)
memory usage: 18.4+ KB
```

The three chunks of code above show the detailed process of web-scraping. Firstly, find the URL of the website and check the status code. If the status code is within the range of 200-299, it means that the request was successfully completed and it is OK the continue scraping. Then use the function BeautifulSoup() to extract the content of the website into the object called soup_object. By reading through the content of the website, we can locate the data table that we want to scrap is the second one on the website. The data table has a HTML tag called

and with class . Then we can further extract the data table information from the website using the code <soup_object.find_all('table', 'wikitable')[1]> and store it in the variable data_table. Lastly, create an empty data frame called Earthquake. Then simply iterate over the rows of the dataset and insert the value from each data-contained cell to the empty data frame. The values in column longitude, latitude and magnitude are converted from string type to float type, in order to merge with the original data.

The scraped data has 168 observations. It is stored in variable Earthquake. The column information and the first six rows of the data are shown above.

[148]:
```
Earthquake=Earthquake.round({'latitude': 2, 'longitude': 2})
merged = pd.merge(data, Earthquake, how='left',on=["latitude", "longitude"])
merged.head()
```

[148]:
```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity  \
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
```

```
2       496.0       177.0        7.2574            352100.0        NEAR BAY
3       558.0       219.0        5.6431            341300.0        NEAR BAY
4       565.0       259.0        3.8462            342200.0        NEAR BAY

   …  Depth  Mag  Magt  Nst  Gap  Clo  RMS  SRC EventID  Coordinates_y
0  …    NaN  NaN   NaN  NaN  NaN  NaN  NaN  NaN     NaN           None
1  …    NaN  NaN   NaN  NaN  NaN  NaN  NaN  NaN     NaN           None
2  …    NaN  NaN   NaN  NaN  NaN  NaN  NaN  NaN     NaN           None
3  …    NaN  NaN   NaN  NaN  NaN  NaN  NaN  NaN     NaN           None
4  …    NaN  NaN   NaN  NaN  NaN  NaN  NaN  NaN     NaN           None

[5 rows x 28 columns]
```

[143]:

```
merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20640 entries, 0 to 20639
Data columns (total 27 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20640 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
 10  <1H Ocean           20640 non-null  int64
 11  Inland              20640 non-null  int64
 12  NEAR OCEAN          20640 non-null  int64
 13  NEAR BAY            20640 non-null  int64
 14  Island              20640 non-null  int64
 15  Coordinates         20640 non-null  geometry
 16  Date                11 non-null     object
 17  Time                11 non-null     object
 18  Depth               11 non-null     object
 19  Mag                 11 non-null     float64
 20  Magt                11 non-null     object
 21  Nst                 11 non-null     object
 22  Gap                 11 non-null     object
 23  Clo                 11 non-null     object
 24  RMS                 11 non-null     object
 25  SRC                 11 non-null     object
 26  EventID             11 non-null     object
```

```
dtypes: float64(10), geometry(1), int64(5), object(11)
memory usage: 4.4+ MB
```

[144]:
```
merged.isnull().sum()
```

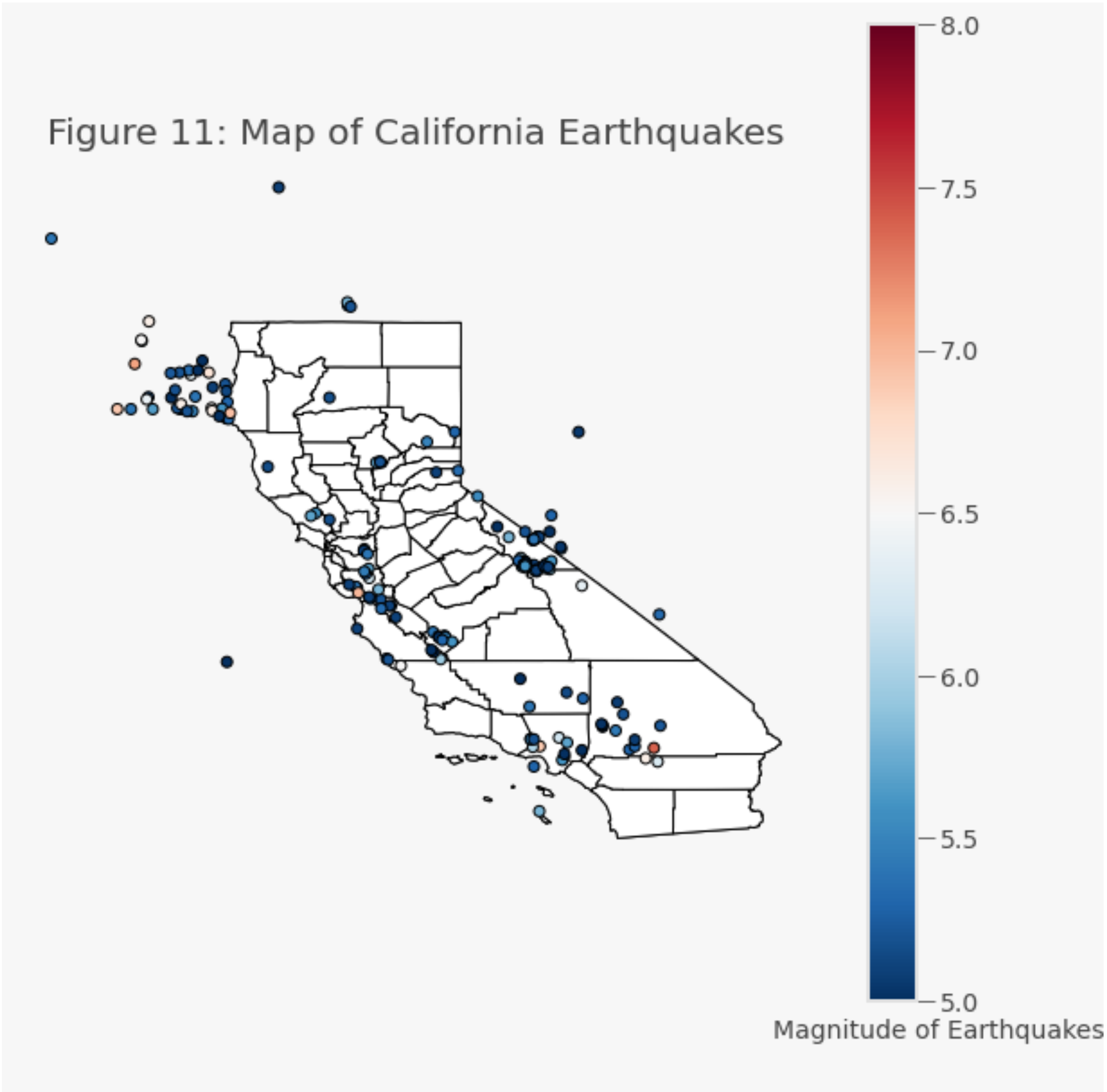
[144]:
```
longitude                0
latitude                 0
housing_median_age       0
total_rooms              0
total_bedrooms           0
population               0
households               0
median_income            0
median_house_value       0
ocean_proximity          0
<1H Ocean                0
Inland                   0
NEAR OCEAN               0
NEAR BAY                 0
Island                   0
Coordinates              0
Date                 20629
Time                 20629
Depth                20629
Mag                  20629
Magt                 20629
Nst                  20629
Gap                  20629
Clo                  20629
RMS                  20629
SRC                  20629
EventID              20629
dtype: int64
```

The longitude and latitude in the new earthquake data are rounded to two decimal places, to merge it with the original data. Since not all the locations had earthquakes before, left join is used in this case to keep all the observations in the original data. The merged dataset is stored in the variable merged. The column names and the first six rows of the data are shown above.

The newly merged dataset contains a lot of missing values because we used left merge. Among the 20640 observations, only 11 observations have the earthquake values.

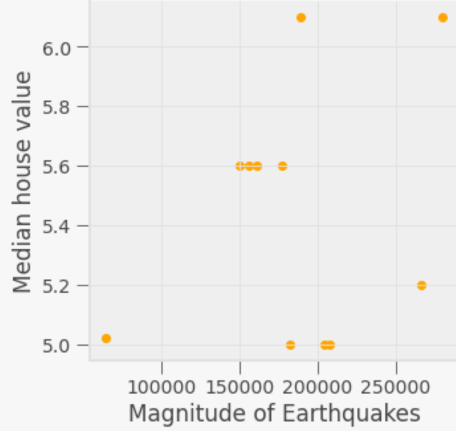
In figure 11, all 168 earthquake data are plotted on the map. The majority of the earthquakes happened at the border of California with a magnitude around 5 to 6. And some of the earthquakes happened outside of California. In figure 12, we used the merged data to plot the correlation between the magnitude of the earthquakes and the housing price. As indicated, the 11 data points randomly scatter in the plot. Therefore there is no significant relationship existed.

[145]:
```python
Earthquake["Coordinates"] = list(zip(Earthquake.longitude,Earthquake.latitude))
Earthquake["Coordinates"] = Earthquake["Coordinates"].apply(Point)
g_Earthquake = gpd.GeoDataFrame(Earthquake, geometry="Coordinates")
#filter geometry information for California
county_df_C = county_df.query("STATEFP == '06'")
#Plot
fig, gax = plt.subplots(figsize=(10, 10))
state_df.query("NAME == 'California'").plot(ax=gax, edgecolor="black",␣
↪color="white")
county_df_C.plot(ax=gax, edgecolor="black", color="white")

g_Earthquake.plot(
    ax=gax, edgecolor='black',column='Mag', legend=True, cmap='RdBu_r',
    vmin=5, vmax=8)

gax.annotate('Magnitude of Earthquakes',xy=(0.7, 0.06),  xycoords='figure␣
↪fraction')
plt.axis('off')
plt.title("Figure 11: Map of California Earthquakes")
plt.show()
```



[146]:
```
plt.figure(figsize=(5,5))
plt.scatter(merged["median_house_value"], merged["Mag"],c='orange')
plt.xlabel('Magnitude of Earthquakes')
plt.ylabel('Median house value')
plt.title('Figure 12:Correlation between Magnitude of Earthquakes and Median␣
↪House value')
plt.show()
```

Figure 12:Correlation between Magnitude of Earthquakes and Median House value



3.6 Summary

After a brief data cleaning and some visualizations of the data, we gained a more comprehensive understanding of the background and the structure of the data. By reviewing the plots, we can make some tentative conclusions about the research question. As indicated in the plots, there exists a positive relationship between income and house prices. People with higher incomes have a greater propensity to spend. Therefore they tend to buy more expensive houses than those who have lower income.

On the other hand, houses on the island are more expensive, whereas the inland houses have a lower price than the others. Besides, we found out that high population density doesn't imply high housing price or low housing price; Inland areas with low population density have low housing price, whereas near-shore areas with low population density tend to have higher housing price. Furthermore, houses in the good-school neighbourhood tend to have a higher price. Since real estate has low liquidity, location is the decisive factor in whether the property appreciates in the future.

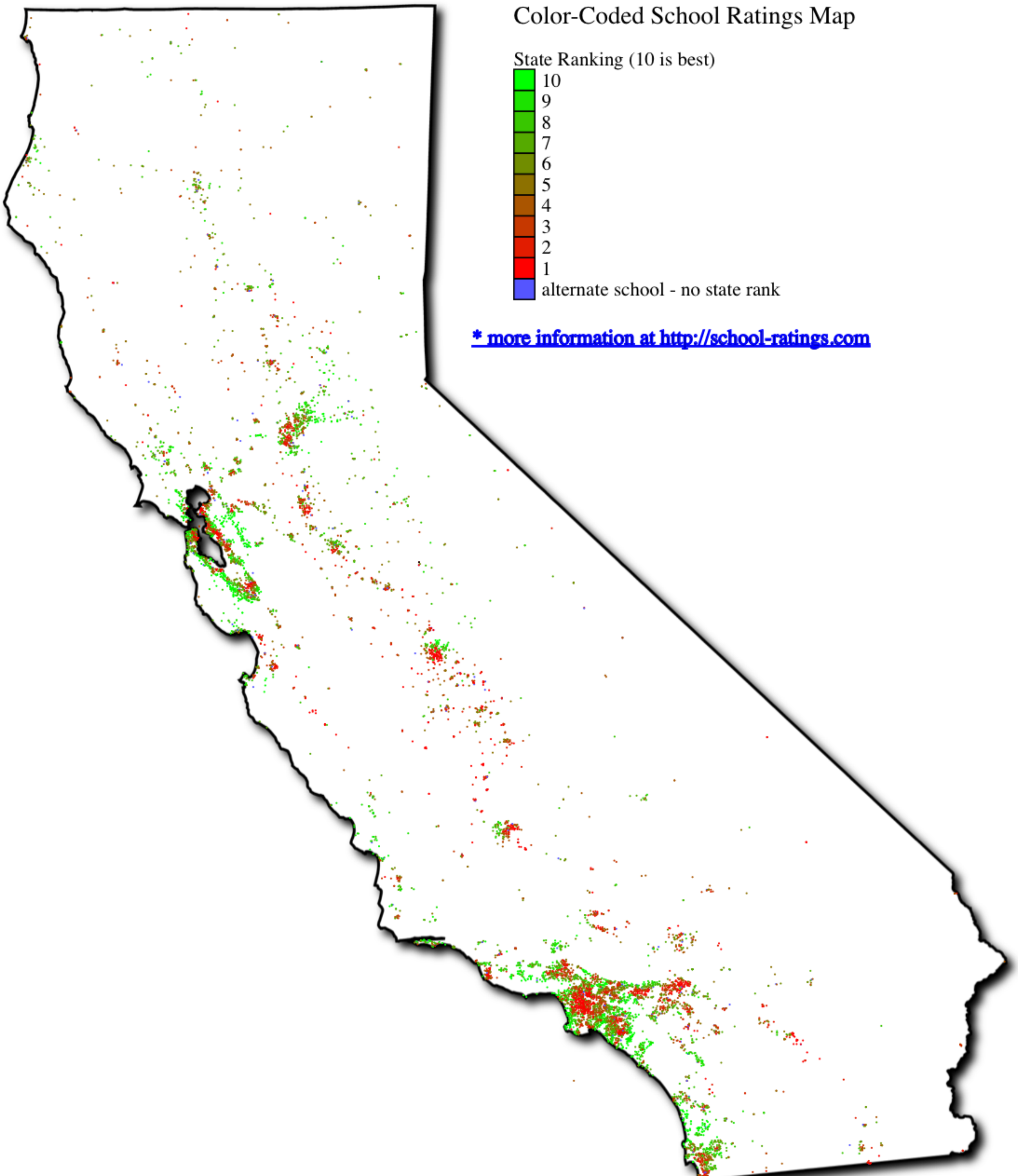
And at last, the magnitude of the earthquakes and housing price may not have a significant relationship.

3.7 Conclusion

To sum up, in project three, I updated the introduction and some wording. I updated the data cleaning part by adding a new correlation plot. I also fine-tuned the map by adding a population density component to it. Some new information such as the goodness of schools in the neighbourhood and earthquakes are taken into account. By conducting exploratory data analysis, and visualization of variables, we now have some intuition about the research question. Some tentative conclusions are drawn in the summary part.

Since some findings are ambiguous by only observing the plots. Therefore in the next step, a model or a machine learning algorithm is required to test the statistical significance of the tentative conclusions and to draw more accurate conclusions.

3.8 Appendix


Color-Coded School Ratings Map
State Ranking (10 is best)
10
9
8
7
6
5
4
3
2
1
alternate school - no state rank
* more information at http://school-ratings.com