Assigment 2

Henning Anevik, Joar Forsberg

September 19 2024

1 Problem 1: Scraping house prices

[ ]:
```
# just a helper method to clean outputs

def clean_output(matches):
  return [line.replace('\xa0', '').strip() for line in matches]
```

Here's what happens:

We go through each and every html file. We extract some useful html with BeautifulSoup because it was easier to work with smaller snippets (for pattern matching with regexps).

From the html the actual information is extracted using regexps and are cleaned up a bit.

The regexp for addresses is not entirely correct. If an address is made up of two or more names only on of these names will be matched. For example Skårby Station 520 will become Station 520.

[ ]:
```
#################################################TASK␣
↪1################################################
debug = True
from bs4 import BeautifulSoup
import re
import pandas
import os

sales = []
addresses = []
plots = []
prices = []
rooms = []
areas = []
locations = []


for file in os.listdir("/content/"):
  if file.endswith(".html"):
    with open(file, 'r') as f:
      html = f.read()
      soup = BeautifulSoup(html, 'html.parser')

      dos = soup.find_all(class_="hcl-label hcl-label--state hcl-label--sold-at") #␣
↪date of sale
      address = soup.find_all(class_="sold-property-listing__heading␣
↪qa-selling-price-title hcl-card__title")
```

```
    tot_area = soup.find_all(class_="sold-property-listing__subheading␣
↪sold-property-listing__area") # includes no. rooms, living area + area
    location = soup.find_all('div', class_='sold-property-listing__location')
    plot_area = soup.find_all(class_="sold-property-listing__land-area")
    price = soup.find_all(class_="hcl-text hcl-text--medium")
    string_soup = str(soup)


    dos_str = str(dos)
    dos_pattern = r"Såld \d{1,2} \w+ \d{4}"
    #dos_matches = list(soup.find_all(string=dos_re))
    dos_matches = re.findall(dos_pattern, dos_str, re.MULTILINE)
    dos_matches.append(file) # for debugging purposes
    sales.append(dos_matches)

    address_str = str(address)
    address_pattern = r"[A-ZÅÄÖ][a-zåäöA-ZÅÄÖ]+\s+\d+$"
    address_matches = re.findall(address_pattern, address_str, re.MULTILINE)
    address_matches.append(file)
    addresses.append(address_matches)

    plot_area_pattern = r'\d+(?:\s\d+)*\s*m² tomt'
    plot_area_matches = re.findall(plot_area_pattern, str(plot_area), re.MULTILINE)
    cleaned_plot_area_matches = clean_output(plot_area_matches)
    cleaned_plot_area_matches.append(file)
    plots.append(cleaned_plot_area_matches)

    price_str = str(price)
    price_pattern = r"Slutpris\s+[\d\s]+kr"
    price_matches = re.findall(price_pattern, price_str, re.MULTILINE)
    cleaned_price_matches = clean_output(price_matches)
    cleaned_price_matches.append(file)
    prices.append(cleaned_price_matches)

    location_pattern = r"([^\n]+?,\s*Kungälvs kommun)"
    location_matches = re.findall(location_pattern, str(location), re.MULTILINE)
    locations.append([line.strip().replace("\n", "") for line in location_matches])

    room_pattern = r"\b\d+\s*rum\b"
    room_matches = re.findall(room_pattern, str(tot_area), re.MULTILINE)

    cleaned_room_matches = clean_output(room_matches)
    cleaned_room_matches.append(file)

    rooms.append(cleaned_room_matches)
    living_area_pattern = r"(\d+)\s*(?:\+\s*(\d+)\s*)?m²"
    living_area_matches = []

    for element in tot_area:
        text = element.get_text(separator=' ', strip=True)
        match = re.search(living_area_pattern, text)
        if match:
            num1 = match.group(1)
            num2 = match.group(2) if match.group(2) else ''
            living_area_matches.append((num1, num2))
    living_area_matches.append(file)
    areas.append(living_area_matches)

if debug:
  print(plots)
  print("")
  print(rooms)
  print("")
  print(areas)
  print("")
  print(sales)
  print("")
  print(addresses)
  print("")
  print(prices)
  print("")
  print(locations)
```

```
[['735m² tomt', '1339m² tomt', '1860m² tomt', '1815m² tomt', '2081m² tomt',
'928m² tomt', '1411m² tomt', '612m² tomt', '2230m² tomt', '780m² tomt', '4215m²
tomt', '459m² tomt', '688m² tomt', '11627m² tomt', '1455m² tomt', '1263m² tomt',
'263m² tomt', '1300m² tomt', '860m² tomt', '446m² tomt', '1696m² tomt', '1661m²
tomt', '262m² tomt', 'kungalv_slutpris_page_40.html']]

[['6rum', '6rum', '4rum', '2rum', '4rum', '4rum', '8rum', '4rum', '4rum',
'4rum', '6rum', '5rum', '3rum', '7rum', '2rum', '3rum', '5rum', '4rum', '6rum',
'3rum', '2rum', '5rum', '4rum', 'kungalv_slutpris_page_40.html']]

[[('174', '14'), ('120', '45'), ('62', ''), ('37', ''), ('123', '45'), ('118',
''), ('150', ''), ('107', ''), ('92', '60'), ('95', '3'), ('110', '68'), ('107',
'5'), ('65', '15'), ('160', '80'), ('45', ''), ('70', ''), ('122', ''), ('128',
'32'), ('185', '36'), ('65', '65'), ('61', ''), ('140', '50'), ('103', ''),
'kungalv_slutpris_page_40.html']]

[['Såld 24 maj 2013', 'Såld 24 maj 2013', 'Såld 23 maj 2013', 'Såld 23 maj
2013', 'Såld 16 maj 2013', 'Såld 14 maj 2013', 'Såld 9 maj 2013', 'Såld 8 maj
2013', 'Såld 30 april 2013', 'Såld 30 april 2013', 'Såld 30 april 2013', 'Såld
21 april 2013', 'Såld 18 april 2013', 'Såld 17 april 2013', 'Såld 17 april
2013', 'Såld 13 april 2013', 'Såld 11 april 2013', 'Såld 10 april 2013', 'Såld 8
april 2013', 'Såld 21 mars 2013', 'Såld 20 mars 2013', 'Såld 8 mars 2013', 'Såld
26 februari 2013', 'kungalv_slutpris_page_40.html']]

[['Gårdskiftegatan 41', 'Kuskalundsvägen 36', 'Vävra 170', 'Stinneröd 160',
'Grokareby 150', 'Duvesjön 370', 'Alekärr 153', 'Munkegärdegatan 353', 'Staby
490', 'Skeppsholmsgatan 3', 'Flateby 500', 'Stenåldersgatan 4', 'Ulvesund 419',
'Trädal 360', 'Dösebacka 640', 'Österväg 28', 'Plogvägen 188', 'Ringby 164',
'Jorsalagatan 5', 'Egnahemsgatan 10', 'Källeröd 220', 'Bokstigen 4',
'Hällristningsgatan 8', 'kungalv_slutpris_page_40.html']]

[['Slutpris 3075000kr', 'Slutpris 1850000kr', 'Slutpris 1385000kr', 'Slutpris
595000kr', 'Slutpris 2620000kr', 'Slutpris 2770000kr', 'Slutpris 2800000kr',
'Slutpris 2630000kr', 'Slutpris 2500000kr', 'Slutpris 3200000kr', 'Slutpris
2610000kr', 'Slutpris 2775000kr', 'Slutpris 2150000kr', 'Slutpris 3400000kr',
'Slutpris 1600000kr', 'Slutpris 1550000kr', 'Slutpris 3300000kr', 'Slutpris
3250000kr', 'Slutpris 4100000kr', 'Slutpris 2825000kr', 'Slutpris 1480000kr',
'Slutpris 3330000kr', 'Slutpris 2725000kr', 'kungalv_slutpris_page_40.html']]

[['Ytterby-Tega,          Kungälvs kommun', 'Vävra,          Kungälvs kommun',
'Hålta,          Kungälvs kommun', 'Romelanda,          Kungälvs kommun',
'Kareby,          Kungälvs kommun', 'Duvesjön,          Kungälvs kommun',
'Romelanda,          Kungälvs kommun', 'Munkegärde,          Kungälvs kommun',
'Kärna,          Kungälvs kommun', 'Centralt,          Kungälvs kommun', 'Kärna
- Harestad,          Kungälvs kommun', 'Ytterby - Stället,          Kungälvs
kommun', 'Ytterby,          Kungälvs kommun', 'Ytterby - Östra Trädal,
Kungälvs kommun', 'Dösebacka,          Kungälvs kommun', 'Bohuslän,
Kungälvs kommun', 'Ullstorp,          Kungälvs kommun', 'Kareby,
Kungälvs kommun', 'Iskällan,          Kungälvs kommun', 'Centralt,
Kungälvs kommun', 'Kareby,          Kungälvs kommun', 'Centralt,
Kungälvs kommun', 'Ytterby - Stället,          Kungälvs kommun']]
```

Here lists are flattenend and padded so that they're all the same size so we can construct a DataFrame

```
[ ]: import numpy as np

# here we pop the last element of each sublist because that element will be the␣
↪filename
# which I added for beugging purposes
for sublist in sales:
  sublist.pop()

for sublist in addresses:
  sublist.pop()

for sublist in plots:
  sublist.pop()

for sublist in prices:
  sublist.pop()

for sublist in rooms:
  sublist.pop()

for sublist in areas:
  sublist.pop()

sales_flat = [item for sublist in sales for item in sublist]
addresses_flat = [item for sublist in addresses for item in sublist]
plots_flat = [item for sublist in plots for item in sublist]
prices_flat = [item for sublist in prices for item in sublist]
rooms_flat = [item for sublist in rooms for item in sublist]
areas_flat = [item for sublist in areas for item in sublist]
locations_flat = [item for sublist in locations for item in sublist]

if debug:
  print(sales_flat)
  print(" ")
  print(addresses_flat)
  print(" ")
  print(plots_flat)
  print(" ")
  print(prices_flat)
  print(" ")
  print(rooms_flat)
  print(" ")
  print(areas_flat)

# pad lists to make sure everything is the same length
max_length = max(len(sales_flat), len(addresses_flat), len(plots_flat),␣
 ↪len(prices_flat), len(rooms_flat), len(areas_flat))

sales_flat = sales_flat + [np.nan] * (max_length - len(sales_flat))
addresses_flat = addresses_flat + [np.nan] * (max_length - len(addresses_flat))
plots_flat = plots_flat + [np.nan] * (max_length -  len(plots_flat))
prices_flat = prices_flat + [np.nan] * (max_length -  len(prices_flat))
rooms_flat = rooms_flat + [np.nan] * (max_length -  len(rooms_flat))
areas_flat = areas_flat + [np.nan] * (max_length -  len(areas_flat))
locations_flat = areas_flat + [np.nan] * (max_length -  len(areas_flat))


df = pandas.DataFrame({
    "Date of sale": [line.replace("Såld", "") for line in sales_flat],
    "Address": addresses_flat,
    "Price": [line.replace("Slutpris", "") if isinstance(line, str) else line for line␣
 ↪in prices_flat],
    "No. rooms": rooms_flat,
    "Living area": areas_flat,
    "Plot size": [line.replace("tomt", "") if isinstance(line, str) else line for line␣
 ↪in plots_flat]
})
df.to_csv("out.csv")

if debug:
  print(df)
```

```
[ ]: #Debugging
with open("/content/kungalv_slutpris_page_34.html", 'r') as f:
  html = f.read()
  soup = BeautifulSoup(html, 'html.parser')
  no_rooms = soup.find_all('div', class_="sold-property-listing__subheading␣
 ↪sold-property-listing__area") # includes no. rooms, living area + area
  matches = re.findall(r'\b\d+\s*rum\b', str(no_rooms))
print(no_rooms)
print(matches)
```

2 Problem 2: Analyzing 2022 house sales

Starting with importing the necesary libaries and cleaning the data to the right format needed for calculations and future ploting. The function “FiveNumSun” used to calculate the five-number summary comes from [1]. Some imprefections was removed like dupliction of the index as well as blankspace in the column “Date of sale".

```
[ ]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     df = pd.read_csv("out.csv")
     df = df.drop("Unnamed: 0", axis=1)  #Remove dublicate id
     df["Date of sale"] = df["Date of sale"].str.slice(start=1)  #Remove added blank space␣
      ↪from the data
     df = df[df["Date of sale"].str.contains("2022")]
     df["Price"] = pd.to_numeric(df["Price"].str.replace("kr", "")) #Remove kr and change␣
      ↪the data to numeric

     def FiveNumSum(data):
         return np.percentile(data, [0, 25, 50, 75, 100], method="midpoint")

     print("Here is the five-number summary ", FiveNumSum(df["Price"]))
```

```
Here is the five-number summary [ 1650000. 4025000. 5000000. 5790000. 10500000.]
```

The construction of the histogram the bin estimation method used was Freedman-Diaconis Rule. There exist load of more method with their respective pros and cons. The reasoning for using Freedman was the data was not normal distrubted and it had some outliers. The method for calculating the size comes from [2].

```
[ ]: iqr = np.subtract(*np.percentile(df["Price"], [75, 25]))
     bin_width = 2 * iqr * len(df["Price"])**(-1/3)
     num_bins = int((max(df["Price"]) - min(df["Price"])) / bin_width)

     plt.hist(df["Price"], bins=num_bins, color="yellow", edgecolor = "black")
     plt.title("Closing Price Histogram")
     plt.ylabel("Frequency")
     plt.xlabel("Price")
     plt.show()
```

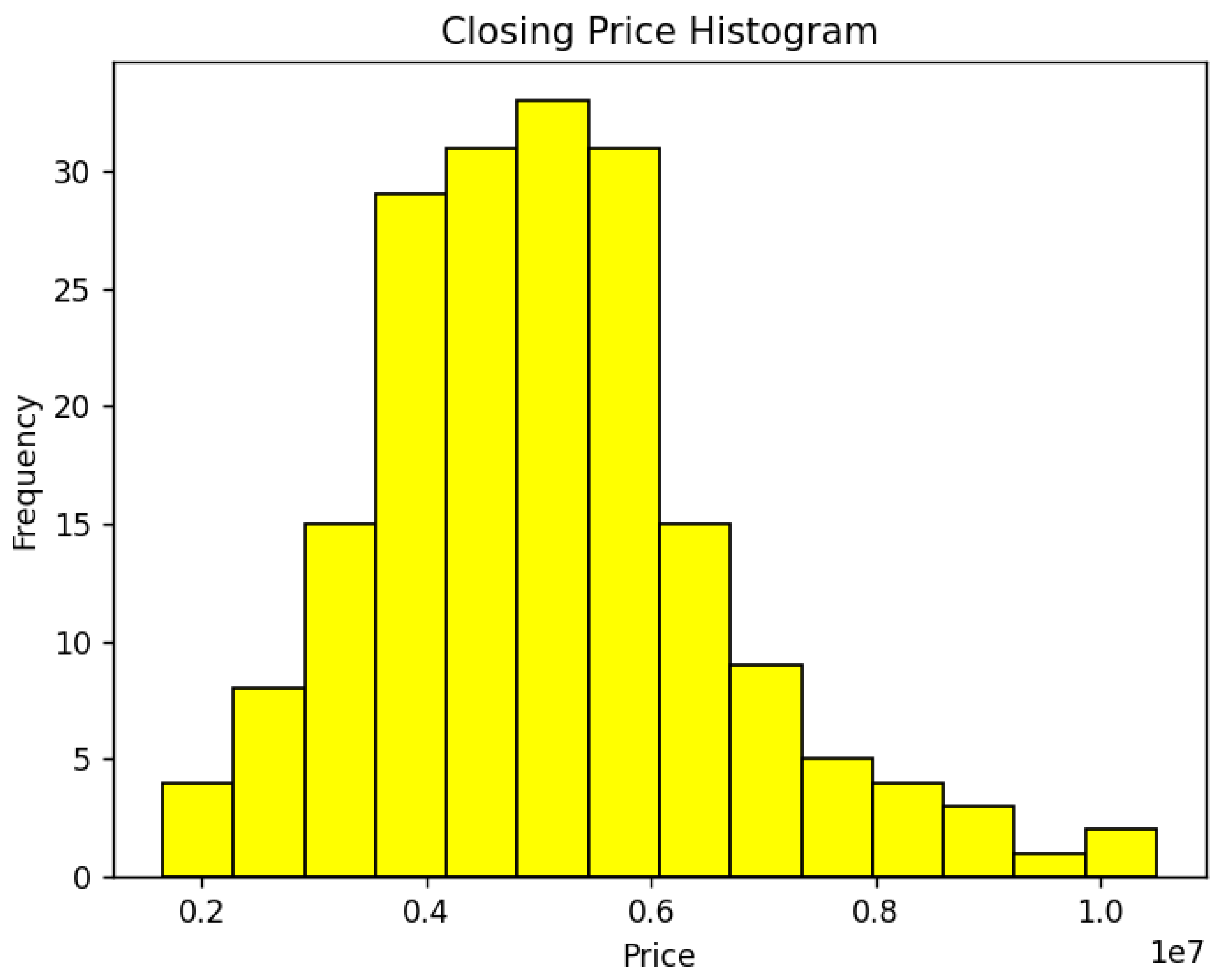

Fig. 1 Histogram price frequency

For the histogram exatraction of the boarea was needed.

```
[ ]: df['Living area'] = pd.to_numeric(df['Living area'].str.extract(r"\('(\d+)'")[0])␣
      ↪#Filter so only boarea is included
     plt.scatter(df["Price"], df["Living area"])
     plt.title("Scatter plot relation price, boarea")
     plt.xlabel("Price")
     plt.ylabel("Boarea")
     plt.show()
```

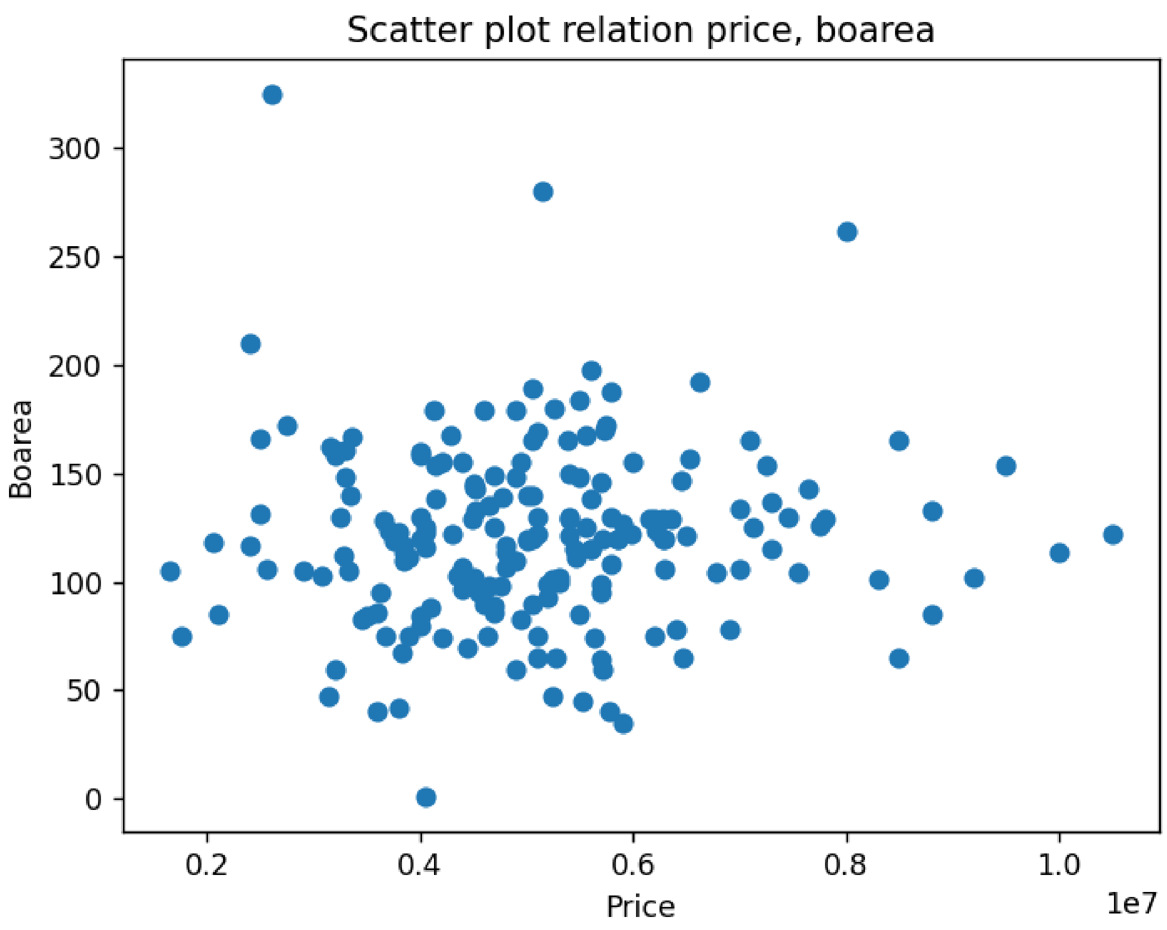

Fig. 2 Histogram showing price and boarea relationship

Lastly a histogram where the number of rooms changes the color of the dots which can be seen in the following fig.

```
[ ]: df["No. rooms"] = pd.to_numeric(df["No. rooms"].str.replace("rum", "")) #Remove kr and␣
      ↪change the data to numeric
     plt.scatter(df["Price"], df["Living area"], c=df["No. rooms"], cmap="cividis")
     plt.colorbar().set_label("No. rooms")
     plt.title("Scatter plot relation price, boarea")
     plt.xlabel("Price")
     plt.ylabel("Boarea")
     plt.show()
```

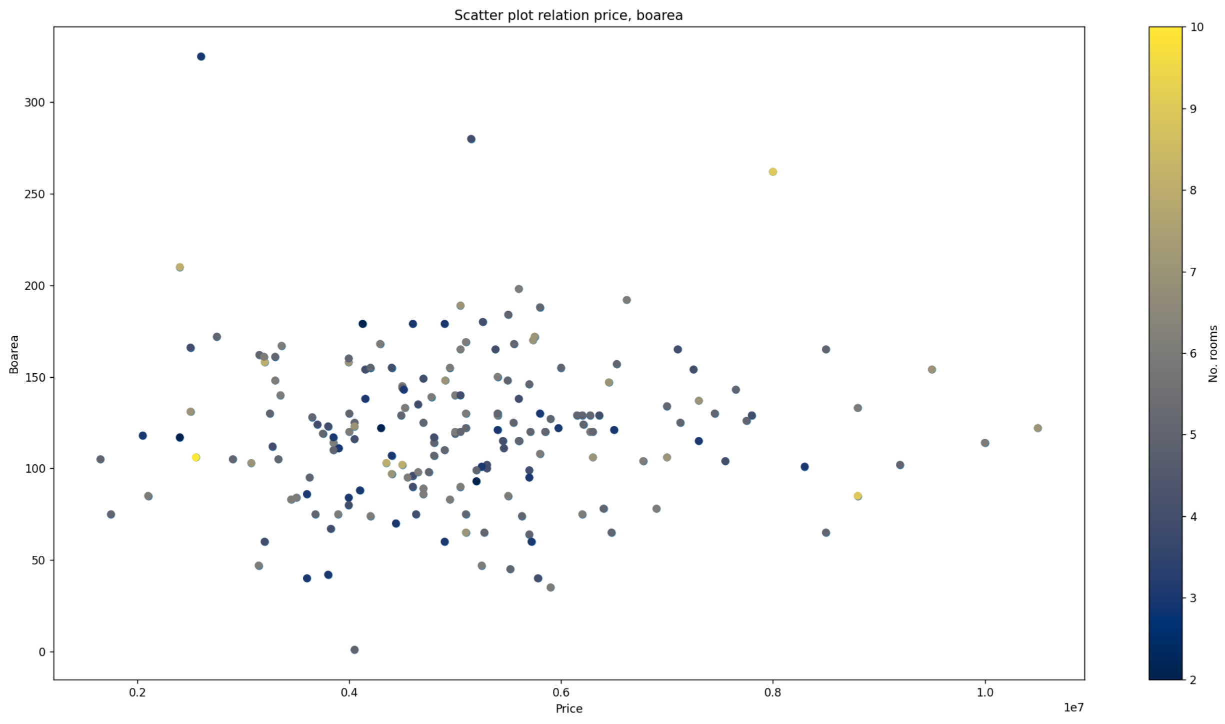

Fig. 3 Histogram showing price and boarea relationship and colors connected to no. rooms

3 Discussion

Start by analyzing each figure individually and then discuss their relations. Fig. 1 is not normally distributed. It's more right-skewed, meaning there are some houses that are more expensive. But the majority of the houses are grouped together in the same price category.

Fig. 2 shows a scatter plot that represents the relationship between the different prices of the houses and their respective "boarea". It's hard to create a clear conclusion due to the wide scatter. We can also see that a bigger boarea doesn't have to mean a higher end price for the house, as there are some outliers.

In Fig. 3, the majority of the houses have 4-6 rooms, and this doesn't necessarily mean higher prices. We can also see that some outliers with 8+ rooms go for both high and low prices. So it's hard to draw a linear relationship between the different aspects.

In conclusion, it's hard to make a solid statement about the different variables and their relationships. While boarea and rooms are important, it doesn't mean there is a direct impact on the end price. Some houses may be in bigger or more popular cities, which influences the price more than boarea and the number of rooms. To be able to draw better and more precise conclusions, more data is needed and more parameters need to be considered. 190 entries are not enough to create or see a clear pattern.

4 Refrences

[1] Wikipedia, "Five-number summary" Wikipedia, [Online] Available: https://en.wikipedia.org/wiki/Five-number_summary [Accessed: Sep. 18 2024].

[2] How matplotlib, "How to Optimize Bin Size in Matplotlib Histogram for Data Visualization" How2matplotlib, [Online] Available: https://how2matplotlib.com/bin-size-in-matplotlib-histogram.html [Accessed: Sep. 18 2024].