DAT565/DIT407 Assignment 2

Ola Bratt
ola.bratt@gmail.com

Patrick Attimont
patrickattimont@gmail.com

2024-01-xx

This paper is addressing the assignment 2 study queries within the *Introduction to Data Science & AI* course, DIT407 at the University of Gothenburg and DAT565 at Chalmers. The main source of information for this project is derived from the lectures and Skiena [1].

Problem 1: Scrapping house prices

Problem 1 have been solved using BeautifulSoup together with simple string operations such as

```
split, replace and strip,
```

also regaular expressions have been used to idefity certain information. The code can be found in the appendix.

Problem 2: Analyzing 2022 house sales

To caluculate the five-number summary of the closing prices of the houses prices we simply used

```
describe()
```

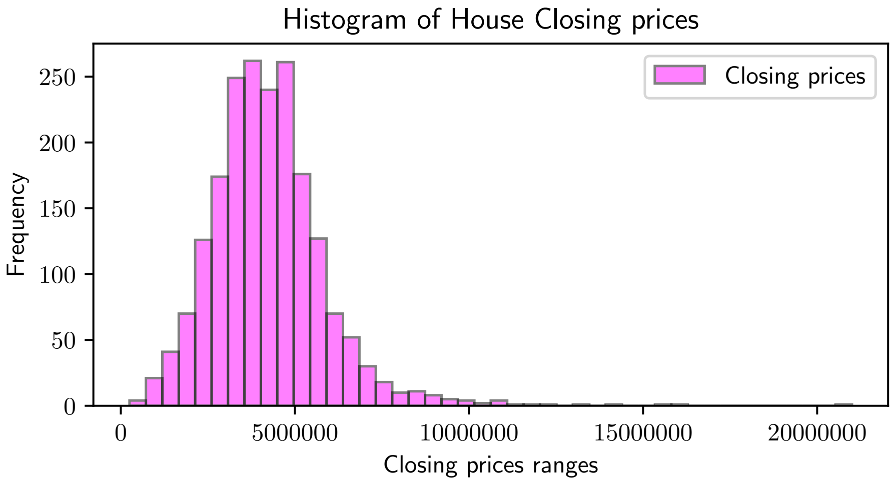
on the dataframe containing the closing prices. The result can be seen in table 1.

When plotting the histogram of the closing prices, see figure 1a we used *square root method* to decide bin size. It seems appropriate since it revelas trends without hiding the details. The plot is skewed to the right, which is expected since there are few houses with high prices. The plot of closing price vs house area is shown in figure 1b. The plot of closing price vs house area with color is shown in figure 1c.

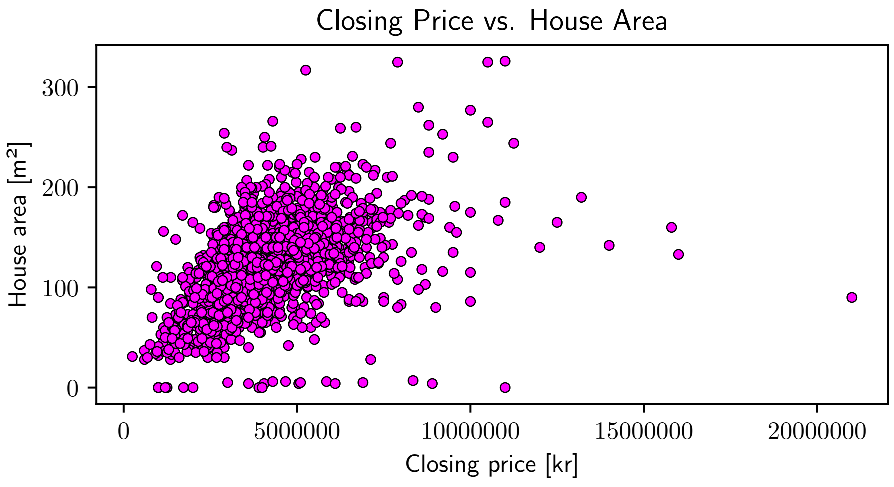
min	250000
25%	3200000
50%	4100000
75%	5035000
max	21000000

Table 1: Five-number summary of closing prices

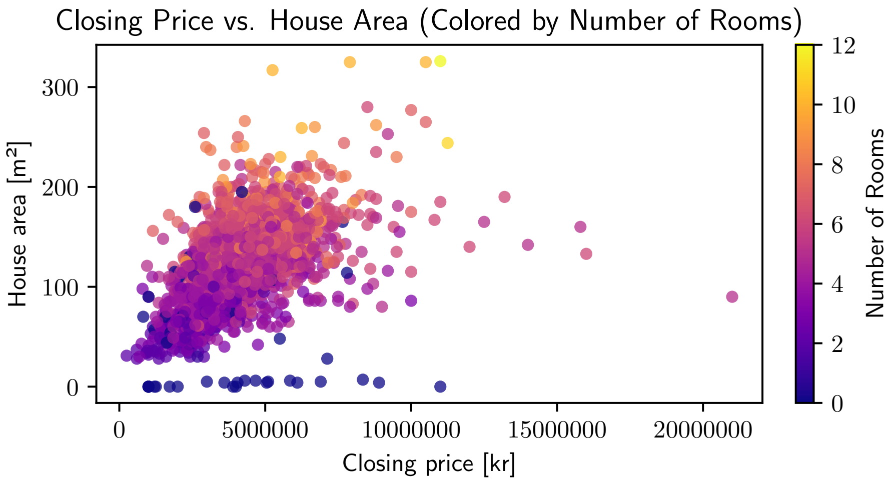
Discussion



(a) Closing prices of houses



(b) Closing price vs house area



(c) Closing price vs house area with color

Figure 1: Plots of house prices

References

[1] Steven S Skiena. *The Data Science Design Manual*. Retrieved 2024-01-20. 2024. URL: `https://ebookcentral.proquest.com/lib/gu/detail.action?docID=6312797`.

Appendix: Source Code

```python
import numpy as np
import pandas as pd
import glob
import errno
import re
import locale
import datetime
import matplotlib as mpl
from matplotlib import pyplot
from bs4 import BeautifulSoup
locale.setlocale(locale.LC_TIME, "sv_SE") # For Swedish dates

date_obj = lambda dateText: datetime.datetime.strptime(dateText.
    ↪ replace('S ld ',''').strip(), '%d %B %Y')

def cleanLocation(locationText):
    locationText.span.decompose()
    stripped = locationText.text.strip().replace("\n", "")
    splitted = stripped.split(',')
    locationList = list(map(lambda x: x.strip(), splitted))
    return ", ".join(locationList)

def areaAndRoom(areaText):
    areaText.span.decompose() if areaText.span else areaText
    areaAndRoom = re.findall(r'\d+', areaText.text.strip())
    areaAndRoomList = list(map(lambda x: x.strip(), areaAndRoom))
    intList = [eval(i) for i in areaAndRoomList]
    area = 0
    room = 0
    errors = 0
    try:
        area = intList[0]
        room = intList[1]
    except IndexError:
        errors += 1
    #print('Errors ' + errors.__str__())
    return area, room

def cleanLandArea(landAreaText):
    landAreaText = landAreaText.replace('\u00a0','')
    return zeroIfNoNumber(landAreaText)

def cleanPrice(priceText):
    priceText = priceText.replace('Slutpris','')
    priceText = priceText.replace('kr','')
    priceText = priceText.replace('\u00a0','')
    return zeroIfNoNumber(priceText)

def zeroIfNoNumber(valueText):
    value = re.findall(r'\d+', valueText)
    if value.__len__() > 0:
        value = int(value[0])
    else:
        value = 0
    return value

def parseObject(obj):
        dateText = obj.find('span',attrs={'class':'hcl-label hcl-label--state hcl-label--sold-at'}).text
        addressText = obj.find('h2',attrs={'class':'sold-property-listing__heading qa-selling-price-title hcl-card__title'}).text
        locationText = obj.find('span',attrs={'class':'property-icon property-icon--result'}).parent
        areaText = obj.find('div',attrs={'class':'sold-property-listing__subheading sold-property-listing__area'})
        extraAreaText = obj.find('span',attrs={'class':'listing-card__attribute--normal-weight'}).text if obj.find('span',attrs={'class':'listing-card__attribute--normal-weight'}) else ''
        landAreaText = obj.find('div',attrs={'class':'sold-property-listing__land-area'}).text if obj.find('div',attrs={'class':'sold-property-listing__land-area'}) else ''
        priceText = obj.find('span',attrs={'class':'hcl-text hcl-text--medium'}).text
        area, room = areaAndRoom(areaText)
        extraArea = zeroIfNoNumber(extraAreaText)
        return [date_obj(dateText), addressText.strip(), cleanLocation(locationText), area, extraArea, area + extraArea , room, cleanLandArea(landAreaText), cleanPrice(priceText)]


dir_path = '../kungalv_slutpriser/*.html'
files = glob.glob(dir_path)
entities = pd.DataFrame(columns=['Date', 'Address', 'Location', 'Area', 'ExtraArea', 'TotalArea', 'Rooms', 'LandArea', 'Price'])
for name in files:
    try:
        with open(name) as f:
            soup = BeautifulSoup(f, "html.parser")
            objects = soup.findAll('li',attrs={'class':'sold-results__normal-hit'})
            for obj in objects:
                entity = parseObject(obj)
                entities.loc[len(entities.index)] = entity
    except IOError as exc:
        if exc.errno != errno.EISDIR:
            raise


entities.to_csv('entities.csv', index=False, encoding='utf-8')


pyplot.rcParams['text.usetex'] = True
entities = pd.read_csv('entities.csv')
#print(entities.head())
print(entities['Price'].describe())

# Plot histogram of closing prices
num_bins = int(len(entities['Price']) ** 0.5) # Determine the number of bins using the square root choice method
```

```python
fig1, ax1 = pyplot.subplots(figsize=(5, 2.7), layout='constrained')
ax1.hist(entities['Price'], bins=num_bins, color='magenta', edgecolor='black', linewidth=1, alpha=0.5, label='Closing prices')
ax1.set_xlabel('Closing prices ranges')  # Add an x-label to the axes.
ax1.set_ylabel('Frequency')  # Add a y-label to the axes.
ax1.set_title("Histogram of House Closing prices")  # Add a title to the axes.
ax1.legend(loc='upper right')
ax1.ticklabel_format(useOffset=1, style='plain', axis='x')
fig1.savefig('histogram_closing_price.pdf', bbox_inches='tight')


# Plot Closing Price vs. House Area
fig2, ax2 = pyplot.subplots(figsize=(5, 2.7), layout='constrained')
ax2.scatter(entities['Price'], entities['Area'], s=15, color='magenta', edgecolor='black', linewidth=0.5)
ax2.set_xlabel('Closing price [kr]')  # Add an x-label to the axes.
ax2.set_ylabel('House area [m ]')  # Add a y-label to the axes.
ax2.set_title("Closing Price vs. House Area")  # Add a title to the axes.
ax2.ticklabel_format(useOffset=1, style='plain', axis='x')
fig2.savefig('closing_price_house_ares.pdf', bbox_inches='tight')


# Plot Closing Price vs. House Area (Colored by Number of Rooms)
fig3, ax3 = pyplot.subplots(figsize=(5, 2.7), layout='constrained')
ax3.scatter(entities['Price'], entities['Area'],  c=entities['Rooms'], cmap='plasma', s=15, alpha=0.75)
ax3.set_xlabel('Closing price [kr]')  # Add an x-label to the axes.
ax3.set_ylabel('House area [m ]')  # Add a y-label to the axes.
ax3.set_title("Closing Price vs. House Area (Colored by Number of Rooms)")  # Add a title to the axes.
sm = pyplot.cm.ScalarMappable(cmap='plasma')
sm.set_array(entities['Rooms'])
fig3.colorbar(sm,  label='Number of Rooms', ax=pyplot.gca())
ax3.ticklabel_format(useOffset=1, style='plain', axis='x')
fig3.savefig('closing_price_house_ares_color.pdf', bbox_inches='tight')
```