# flats

November 12, 2024

### 0.0.1 : .

**1.** 1- 54 9
. 57 . , , “ ”
. , .

: * ( ) *
?

[ ]:
```
import time
import datetime

def timer(f):
    def wrap_timer(*args, **kwargs):
        start = time.time()
        result = f(*args, **kwargs)
        delta = time.time() - start
        print (f'              {f.__name__}       {delta}     ')
        return result
    return wrap_timer
```

[ ]:
```
import logging

def log(f):

    def wrap_log(*args, **kwargs):

        logging.info(f"          {f.__doc__}")
        result = f(*args, **kwargs)
        logging.info(f"     : {result}")
        return result

    return wrap_log

logging.basicConfig(level=logging.INFO)
```

P.S. “ ” ( , tqdm).
, ?

[ ]:
```
import requests
import json
import time
import tqdm
import numpy as np


@timer
@log
def requests_site(N):
    """                    """
    headers = ({'User-Agent':'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)␣
 ↪AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.5 Safari/605.1.15'})
    pages = [1 + i for i in range(N)]
    n = 0

    for i in pages:
        s = f"https://www.cian.ru/cat.php?
 ↪deal_type=rent&engine_version=2&page={i}&offer_type=flat&region=1&room1=1&type=-2"

        response = requests.get(s, headers = headers)
        if response.status_code == 200:
            name = f'sheets/sheet_{i}.txt'
            with open(name, 'w') as f:
                f.write(response.text)
            n += 1
            logging.info(f"            {i}")
        else:
            print(f"        {i}          response.status_code = {response.
 ↪status_code}")
        time.sleep(np.random.randint(7,13))
    return f"           {n}     "

requests_site(300)
```

**2.** BeautifulSoup lxml, , “ ” , , , lxml . , ,

**2.1** . , os, . , .

[ ]:
```
def read_file(filename):
    with open(filename) as input_file:
        text = input_file.read()
    return text
```

[ ]:
```
import tqdm

site_texts = []
pages = [1 + i for i in range(309)]

for i in tqdm.tqdm(pages):
    name = f'sheets/sheet_{i}.txt'
    site_texts.append(read_file(name))

print(f"      {len(site_texts)}     .")
```

**2.2**

[ ]:
```
from bs4 import BeautifulSoup
import re
import pandas as pd
from dateutil.parser import parse
from datetime import datetime, date, time

def parse_tag(tag, tag_value, item):
    key = tag
    value = "None"
    if item.find('div', {'class': tag_value}):
        if key == 'link':
            value = item.find('div', {'class': tag_value}).find('a').get('href')
        elif (key == 'price' or key == 'price_meter'):
            value = parse_digits(item.find('div', {'class': tag_value}).text,␣
 ↪key)
        elif key == 'pub_datetime':
            value = parse_date(item.find('div', {'class': tag_value}).text)
        else:
            value = item.find('div', {'class': tag_value}).text
    return key, value


def parse_digits(string, type_digit):
    digit = 0
    try:
        if type_digit == 'flats_counts':
            digit = int(re.sub(r" ", "", string[:string.find(" ")]))
        elif type_digit == 'price':
            digit = re.sub(r" ", "", re.sub(r" ", "", string))
        elif type_digit == 'price_meter':
            digit = re.sub(r" ", "", re.sub(r" / ²", "", string))
    except:
        return -1
    return digit
```

```
def parse_date(string):
    now = datetime.strptime("15.03.20 00:00", "%d.%m.%y %H:%M")
    s = string
    if string.find('    ') >= 0:
        s = "{} {}".format(now.day, now.strftime("%b"))
        s = string.replace('    ', s)
    elif string.find('   ') >= 0:
        s = "{} {}".format(now.day - 1, now.strftime("%b"))
        s = string.replace('   ',s)
    if (s.find('  ') > 0):
        s = s.replace('  ','mar')
    if (s.find('  ') > 0):
        s = s.replace('  ','feb')
    if (s.find('  ') > 0):
        s = s.replace('  ','jan')
    return parse(s).strftime('%Y-%m-%d %H:%M:%S')


def parse_text(text, index):

    tag_table = '_93444fe79c--wrapper--E9jWb'
    tag_items = ['_93444fe79c--card--_yguQ', '_93444fe79c--card--_yguQ']
    tag_flats_counts = '_93444fe79c--totalOffers--22-FL'
    tags = {
        'link':('c6e8ba5398--info-section--Sfnx-␣
 ↪c6e8ba5398--main-info--oWcMk','undefined c6e8ba5398--main-info--oWcMk'),
        'desc': ('c6e8ba5398--title--2CW78','c6e8ba5398--single_title--22TGT',␣
 ↪'c6e8ba5398--subtitle--UTwbQ'),
        'price': ('c6e8ba5398--header--1df-X', 'c6e8ba5398--header--1dF9r'),
        'price_meter': 'c6e8ba5398--term--3kvtJ',
        'metro': 'c6e8ba5398--underground-name--1efZ3',
        'pub_datetime': 'c6e8ba5398--absolute--9uFLj',
        'address': 'c6e8ba5398--address-links--1tfGW',
        'square': ''
    }

    res = []
    flats_counts = 0
    soup = BeautifulSoup(text)
    if soup.find('div', {'class': tag_flats_counts}):
        flats_counts = parse_digits(soup.find('div', {'class':␣
 ↪tag_flats_counts}).text, 'flats_counts')

    flats_list = soup.find('div', {'class': tag_table})
    if flats_list:
        items = flats_list.find_all('div', {'class': tag_items})
```

```
        for i, item in enumerate(items):

            d = {'index': index}
            index += 1
            for tag in tags.keys():
                tag_value = tags[tag]
                key, value = parse_tag(tag, tag_value, item)
                d[key] = value
            results[index] = d

    return flats_counts, index
```

, , , ,
. , *5 402* *5343* *5402*,
( ).
, . 54 ,
309 , , 8640 .
.

[ ]:
```
from IPython.display import clear_output

sum_flats = 0
index = 0
results = {}
for i, text in enumerate(site_texts):
    flats_counts, index = parse_text(text, index)
    sum_flats = len(results)
    clear_output(wait=True)
    print(f"    {i + 1} flats = {flats_counts},          {sum_flats}    ")
print(f"   sum_flats ({sum_flats}) = flats_counts({flats_counts})")
```

### 2.3 C .

[ ]:
```
col = {'index','link','desc','price','price_meter', 'address',␣
 ↪'metro','pub_datetime'}
df = pd.DataFrame(results, col).T.sort_values(by="pub_datetime", ascending =␣
 ↪True)
df.to_csv('flats.csv')
```

**3** , 8640 . “ ” .

,

“ ”.

:

1. : - .
, - .
2. - . - “ ”

? ( ) ,

, , , . , , ,

. “ ” :

. “ ” (

) , .

: price_per_month - square - okrug - ,
price_meter - 1

[1]:
```
import pandas as pd

df = pd.read_csv('flats.csv')
df.shape
```

[1]:
```
(8640, 9)
```

[2]:
```
df['price_per_month'] = df['price'].str.strip('/  .').astype(int) #price_int
new_desc = df["desc"].str.split(",", n = 3, expand = True)
df["square"]= new_desc[1].str.strip('  ²').astype(int)
df["floor"]= new_desc[2]
new_address = df['address'].str.split(',', n = 3, expand = True)
df['okrug'] = new_address[1].str.strip(" ")
df['price_per_meter'] = (df['price_per_month'] / df['square']).round(2)␣
↪#price_std

df = df.drop(['index','metro', 'price_meter','link',␣
↪'price','desc','address','pub_datetime','floor'], axis='columns')
```

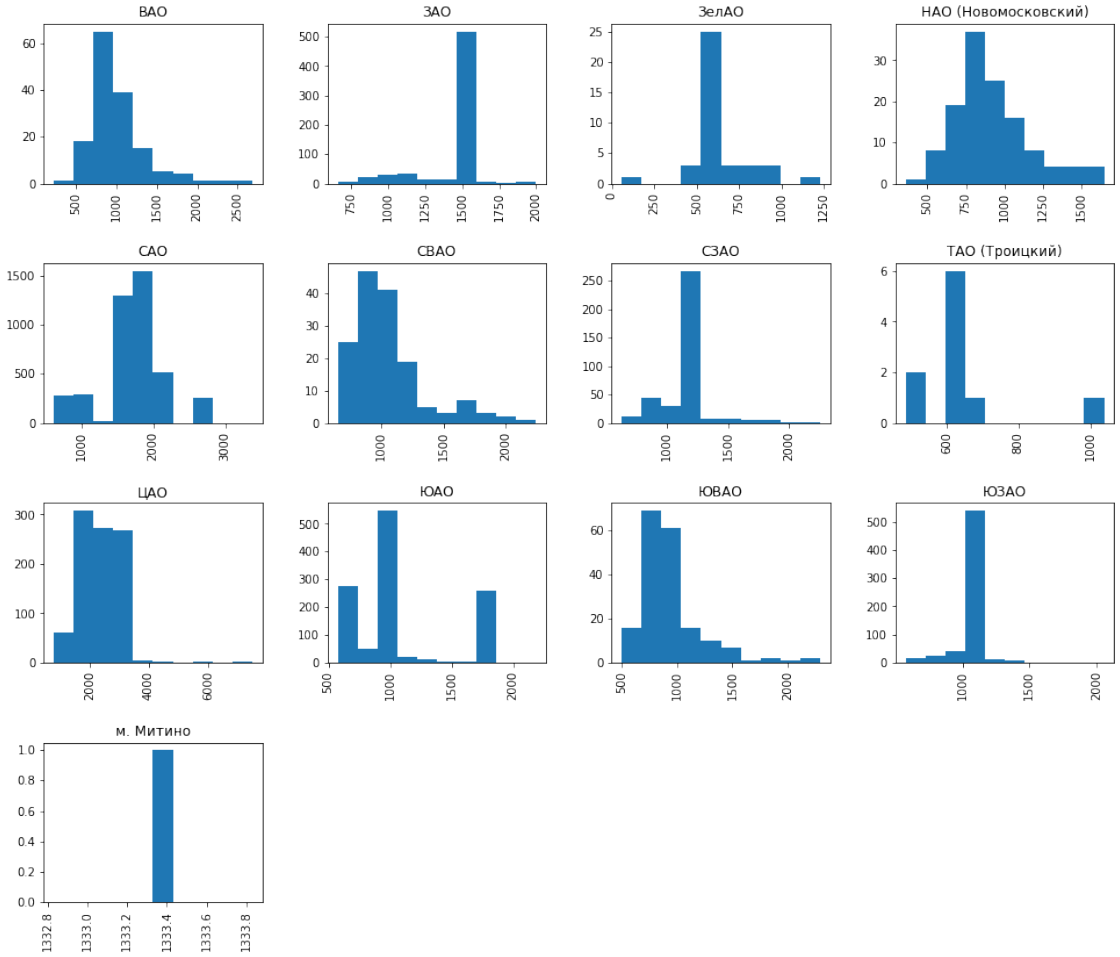
“ ” . : matplotlib, seaborn plotly.

1. . Matplotlib ,
. , 1 , .
: ( 500 ) 1000 ., ( 300 )
1700 . - .

[7]:
```
import matplotlib
hists = df['price_per_meter'].hist(by=df['okrug'], figsize=(16, 14), color =␣
↪"tab:blue", grid = True)
hists
```

[7]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11686d550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x116d04790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x116d38f90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x116d7a7d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x116dabfd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x116dee810>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x116e22f90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x116e61850>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x116e6d3d0>,
```

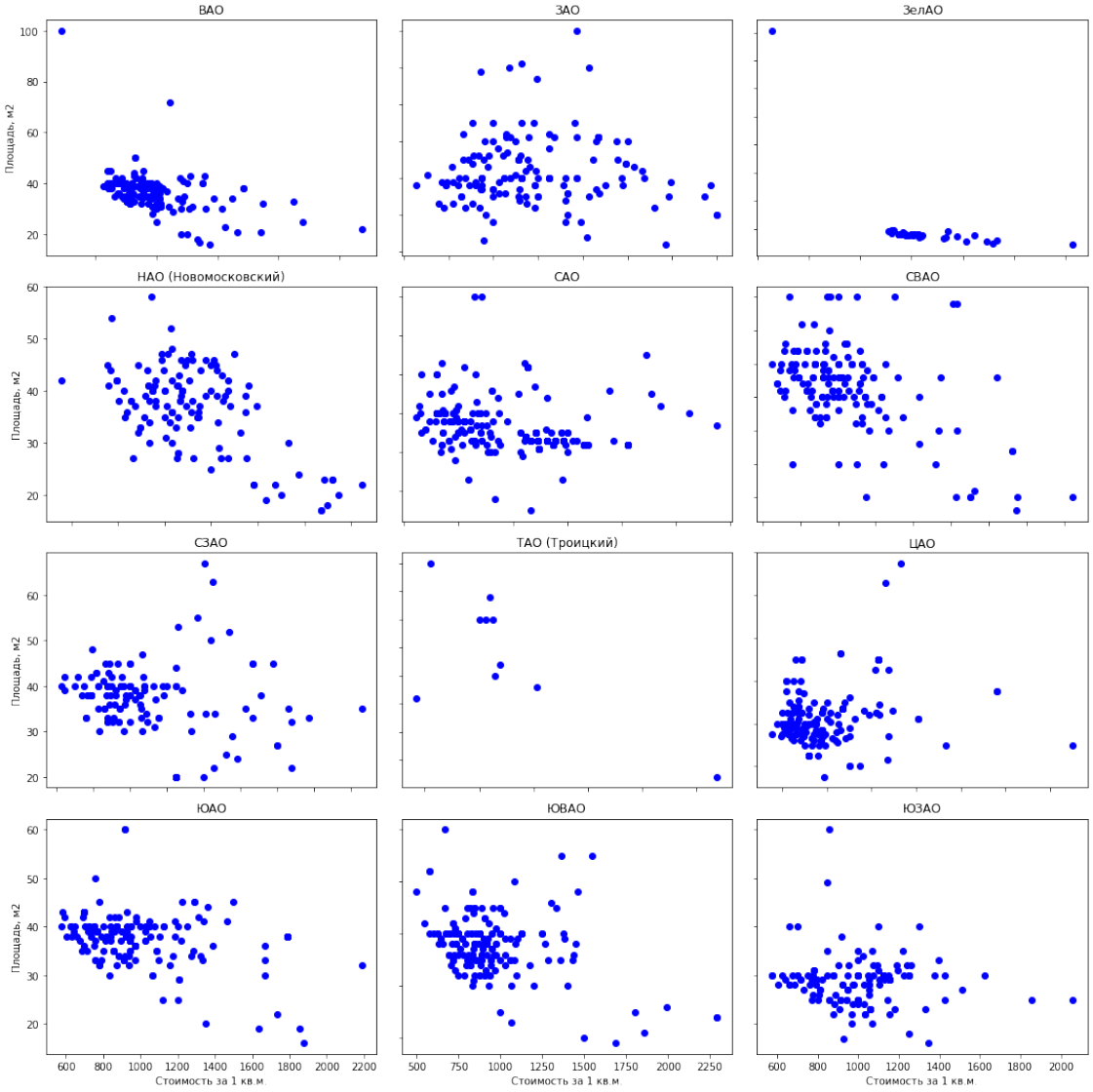
```
  <matplotlib.axes._subplots.AxesSubplot object at 0x116ea1d50>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x116f18c10>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x116f4d8d0>],
 [<matplotlib.axes._subplots.AxesSubplot object at 0x116f8bc50>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x116fc3910>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x117000c90>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x117038950>]],
dtype=object)
```




[8]:
```
df_copy = df.copy(deep = True)
#1          ,
df_copy.drop(df_copy[df_copy['okrug'] == ' .     '].index, inplace=True)
```

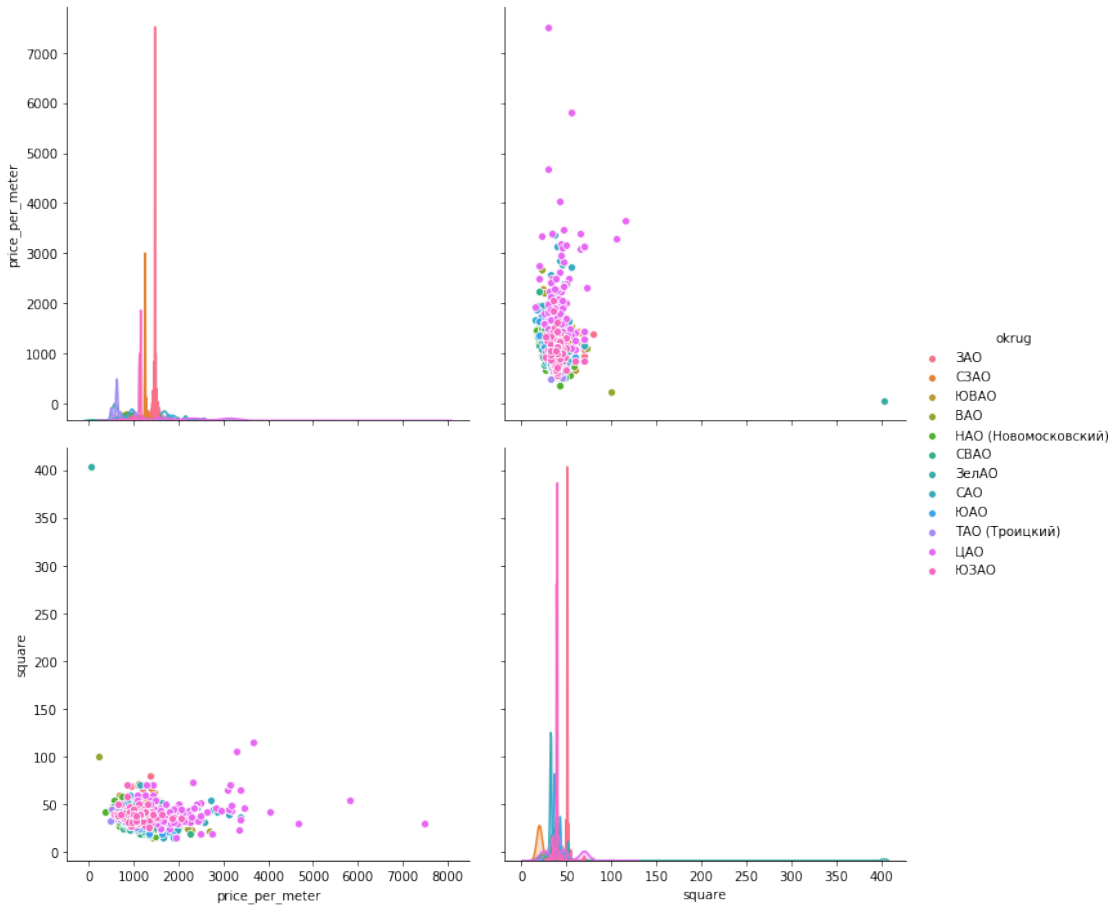
2.  .  . seaborn  -  ,  plotly
,  “  ”,  .

[9]:
```
import matplotlib
import matplotlib.pyplot as plt

fig, axes = plt.subplots(nrows=4,ncols=3,figsize=(15,15))

for i, (name, group) in enumerate(df_copy.groupby('okrug')):
    axes = axes.flatten()
    axes[i].scatter(group['price_per_meter'],group['square'], color ='blue')
    axes[i].set_title(name)
    axes[i].set(xlabel='        1   . .', ylabel='    ,  2')
    axes[i].label_outer()

fig.tight_layout()
```



[10]:
```
import seaborn as sns
sns.pairplot(vars=["price_per_meter","square"], data=df_copy, hue="okrug",␣
↪height=5)
```

[10]: <seaborn.axisgrid.PairGrid at 0x11801cb10>




[11]:
```
import plotly.express as px

for i, (name, group) in enumerate(df_copy.groupby('okrug')):
    fig = px.scatter(group, x="price_per_meter", y="square", facet_col="okrug",
                     width=400, height=400)
    fig.update_layout(
        margin=dict(l=20, r=20, t=20, b=20),
        paper_bgcolor="LightSteelBlue",
    )
    fig.show()
```

[12]:
```
indexes = []

indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 1142.86)].index)
indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 1214.29)].index)

indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']␣
 ↪== 57.07)].index)
indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']␣
 ↪== 1227.27)].index)

indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 1375)].index)

indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 220)].index)
indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 1111.11)].index)
indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 2681.82)].index)

indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 3301.89)].index)
indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 3684.21)].index)
indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 7500)].index)

indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']␣
 ↪== 1405.61)].index)
indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']␣
 ↪== 1449.21)].index)
indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']␣
 ↪== 2257.14)].index)

indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']␣
 ↪== 2291.67)].index)
indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']␣
 ↪== 666.67)].index)

indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 916.67)].index)
indexes.append(df_copy[(df_copy['okrug'] == '  ')&(df_copy['price_per_meter']␣
 ↪== 2666.67)].index)
```

```
indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']
 ↪== 857.14)].index)
indexes.append(df_copy[(df_copy['okrug'] == '   ')&(df_copy['price_per_meter']
 ↪== 2057.14)].index)

print(len(indexes))
print(indexes)
```

```
20
[Int64Index([5208], dtype='int64'), Int64Index([3955], dtype='int64'),
Int64Index([148], dtype='int64'), Int64Index([243], dtype='int64'),
Int64Index([54, 244, 5813], dtype='int64'), Int64Index([1244], dtype='int64'),
Int64Index([3657], dtype='int64'), Int64Index([2834], dtype='int64'),
Int64Index([1249], dtype='int64'), Int64Index([], dtype='int64'),
Int64Index([1242], dtype='int64'), Int64Index([7685], dtype='int64'),
Int64Index([7428], dtype='int64'), Int64Index([5515], dtype='int64'),
Int64Index([1695, 2526], dtype='int64'), Int64Index([1586, 6099],
dtype='int64'), Int64Index([17, 1649], dtype='int64'), Int64Index([],
dtype='int64'), Int64Index([430, 3981, 4524], dtype='int64'), Int64Index([5193],
dtype='int64')]
```

[13]:
```
for index in indexes:
    df_copy.drop(index, inplace=True)
```

[14]:
```
df_copy.to_csv('flats_clear_data.csv')
df_copy.head(5)
```

[14]:

| | Unnamed: 0 | price_per_month | square | okrug | price_per_meter |
|---|---|---|---|---|---|
| 0 | 1381 | 60000 | 30 | | 2000.00 |
| 1 | 1174 | 65000 | 35 | | 1857.14 |
| 2 | 1142 | 32000 | 33 | | 969.70 |
| 3 | 1390 | 80000 | 71 | | 1126.76 |
| 4 | 1350 | 48000 | 50 | | 960.00 |

4.

, . 8602 . .

[15]:
```
import pandas as pd

df_clear = pd.read_csv('flats_clear_data.csv')
df_clear.shape
```

[15]: (8614, 6)

[16]:
```
df_counts = df_clear['okrug'].value_counts().to_frame().reset_index()
df_counts.rename(columns={'index': 'okrug', 'okrug': 'counts'}, inplace=True)
```

```
df_means = df_clear.groupby(['okrug']).agg({'price_per_meter': 'mean',␣
 ↪'price_per_month': 'mean', 'square': 'mean'}).round(2)
df_means.rename(columns={'price_per_meter': 'mprice_per_meter',␣
 ↪'price_per_month': 'mprice_per_month', 'square': 'msquare'}, inplace=True)

df_result = pd.merge(df_counts, df_means, on='okrug')
df_result['price_mean'] = df_result['mprice_per_meter'] * df_result['msquare']
df_result['price_mean'] = df_result['price_mean'].round(2)

df_medians = df_clear.groupby(['okrug']).agg({'price_per_meter': 'median'}).
 ↪round(2)
df_medians.rename(columns={'price_per_meter': 'medprice_per_meter'},␣
 ↪inplace=True)
df_result = pd.merge(df_result, df_medians, on='okrug')

df_std = df_clear.groupby(['okrug']).agg({'price_per_meter': 'std'}).round(2)
df_std.rename(columns={'price_per_meter': 'stdprice_per_meter'}, inplace=True)
df_result = pd.merge(df_result, df_std, on='okrug')


df_result = df_result.sort_values(['price_mean'], ascending=False)
df_result.to_csv('flats_results.csv')

df_result_copy = df_result.copy(deep = True)
df_result_copy.rename(columns={'okrug': '   ',
                               'counts': '            ',
                               'mprice_per_meter': '          1  . .',
                               'mprice_per_month': '                ',
                               'msquare': '         ',
                               'price_mean': '          ',
                               'medprice_per_meter': '           1  . .',
                               'stdprice_per_meter': '                 1  . .'
                              }, inplace=True)

df_result_copy
```

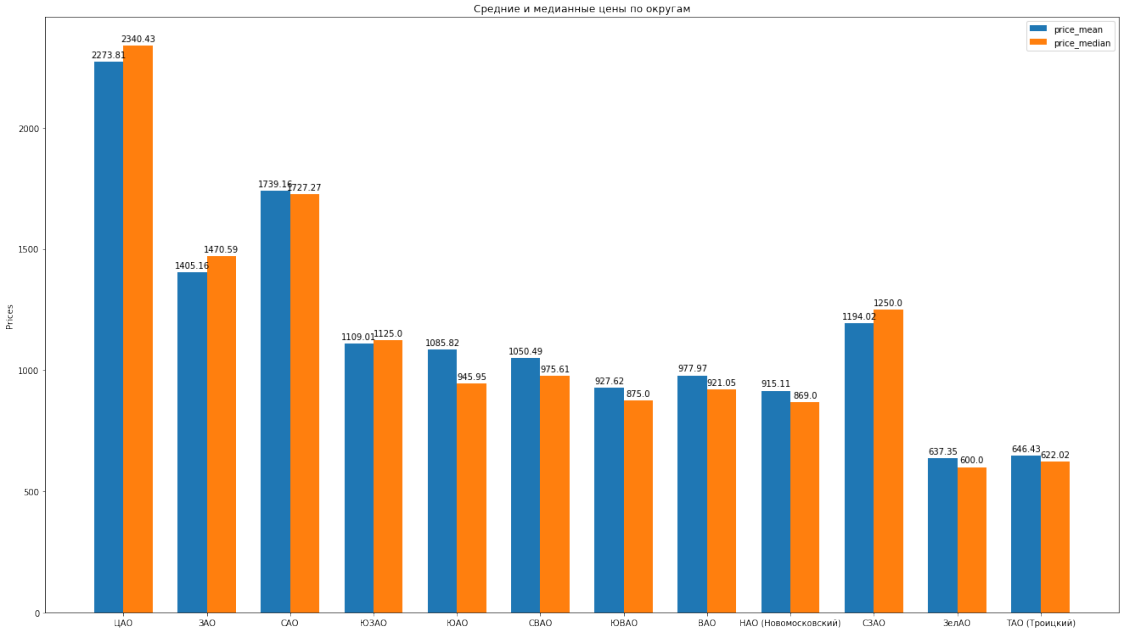
[16]:

| | | 1 . . \ |
|---|---|---|
| 2 | 915 | 2273.81 |
| 3 | 647 | 1405.16 |
| 0 | 4217 | 1739.16 |
| 4 | 637 | 1109.01 |
| 1 | 1167 | 1085.82 |
| 7 | 153 | 1050.49 |
| 6 | 181 | 927.62 |
| 8 | 147 | 977.97 |
| 9 ( ) | 126 | 915.11 |
| 5 | 377 | 1194.02 |

```
10                                          37                  637.35
11          (    )                      10                  646.43

                                            \
2                     115901.14             46.41       105527.52
3                      69236.55             48.95        68782.58
0                      59647.94             34.82        60557.55
4                      43568.27             39.29        43573.00
1                      41171.70             38.29        41576.05
7                      37374.35             36.47        38311.37
6                      34697.08             37.80        35064.04
8                      34135.83             35.71        34923.31
9                      32445.15             36.71        33593.69
5                      29859.87             25.69        30674.37
10                     24162.11             38.62        24614.46
11                     23450.00             37.10        23982.55

               1  . .                      1  . .
2                     2340.43                                699.94
3                     1470.59                                182.11
0                     1727.27                                409.03
4                     1125.00                                115.37
1                      945.95                                399.59
7                      975.61                                294.54
6                      875.00                                242.25
8                      921.05                                297.91
9                      869.00                                238.80
5                     1250.00                                192.21
10                     600.00                                116.24
11                     622.02                                151.49
```

[17]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

labels = df_result['okrug']
price_mean = df_result['mprice_per_meter']
price_median = df_result['medprice_per_meter']

x = np.arange(len(labels))
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, price_mean, width, label='price_mean')
rects2 = ax.bar(x + width/2, price_median, width, label='price_median')

ax.set_ylabel('Prices')
```

```
ax.set_title('                        ')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()


def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')


autolabel(rects1)
autolabel(rects2)

fig.set_size_inches(18.5, 10.5, forward=True)

fig.tight_layout()

plt.show()
```




:

- , , , .
  “ ” , (
  ). , , ,
  , , , “ ” .
  , , .
- . .
  « ».
  , « » — . .
  . , , , , ,
  - , , . . .
- , “ ” ,
  . , , — .

6.

, , .
,

, OpenStreetMap , : geopandas, cartoframes ( , ?) folium,
:) .

[18]:
```
import folium
import geopandas
import pandas as pd
import json

mo_df = geopandas.GeoDataFrame.from_file('atd/mo.shp')
gjson = mo_df.to_json()

df_result.okrug.replace('   (        )','        ', inplace=True)
df_result.okrug.replace('   (    )','    ', inplace=True)


full_df = pd.merge(left=mo_df, right=df_result, left_on='ABBREV_AO',␣
↪right_on='okrug')
full_gf = geopandas.GeoDataFrame(full_df)
df_result.head(15)
```

[18]:

| | okrug | counts | mprice_per_meter | mprice_per_month | msquare | \ |
|---|---|---|---|---|---|---|
| 2 | | 915 | 2273.81 | 115901.14 | 46.41 | |
| 3 | | 647 | 1405.16 | 69236.55 | 48.95 | |
| 0 | | 4217 | 1739.16 | 59647.94 | 34.82 | |
| 4 | | 637 | 1109.01 | 43568.27 | 39.29 | |
| 1 | | 1167 | 1085.82 | 41171.70 | 38.29 | |
| 7 | | 153 | 1050.49 | 37374.35 | 36.47 | |
| 6 | | 181 | 927.62 | 34697.08 | 37.80 | |
| 8 | | 147 | 977.97 | 34135.83 | 35.71 | |
| 9 | 126 | | 915.11 | 32445.15 | 36.71 | |

```
5                    377           1194.02          29859.87    25.69
10                    37            637.35          24162.11    38.62
11                    10            646.43          23450.00    37.10

    price_mean  medprice_per_meter  stdprice_per_meter
2    105527.52             2340.43              699.94
3     68782.58             1470.59              182.11
0     60557.55             1727.27              409.03
4     43573.00             1125.00              115.37
1     41576.05              945.95              399.59
7     38311.37              975.61              294.54
6     35064.04              875.00              242.25
8     34923.31              921.05              297.91
9     33593.69              869.00              238.80
5     30674.37             1250.00              192.21
10    24614.46              600.00              116.24
11    23982.55              622.02              151.49
```

[19]:
```
y = json.loads(gjson)
list_names = []
dict_coordinates = dict()

for mo in y['features']:
    name = mo['properties']['ABBREV_AO']

    if name not in list_names:
        list_names.append(name)
        if (len(df_result[df_result['okrug'] == name]['price_mean']) != 0):
            vote = df_result[df_result['okrug'] == name]['price_mean'].values[0]
            x = mo['geometry']['coordinates'][0][0][1]
            y = mo['geometry']['coordinates'][0][0][0]
            dict_coordinates[name] = (y,x)
print(dict_coordinates)

#
dict_coordinates2 = {'    ': ('Troitskii',37.1031012, 55.3408329),
 '  ': ('ZAO',37.4276499, 55.7482092),
 '       ': ('NewMoscow',37.4315575, 55.5203129),
 '   ': ('ZelAO',37.1685294, 55.98000),
 '  ': ('SZAO',37.4461022, 55.7944941),
 '  ': ('UAO',37.6534248, 55.65539),
 '  ': ('SAO',37.4962542, 55.8524795),
 '  ': ('UZAO',37.5395575, 55.6273129),
 '  ': ('CAO',37.6139298, 55.7584219),
 '  ': ('SVAO',37.5194795, 55.9417633),
 '  ': ('VAO',37.7103095, 55.8029193),
 '  ': ('UVAO',37.6606903, 55.7307034)}
```

```
{'    ': ([36.8031012, 55.4408329], [36.8031903, 55.4416007]), '  ':
(37.4276499, 55.7482092), '        ': (37.4395575, 55.6273129), '   ':
(37.1785294, 56.0079518), '  ': (37.4461022, 55.7944941), '  ': (37.6534248,
55.65539), '  ': (37.4962542, 55.8924795), '   ': (37.4557187, 55.6370488),
'  ': (37.5139298, 55.7584219), '   ': (37.5194795, 55.9417633), '  ':
(37.6503095, 55.7929193), '   ': (37.6606903, 55.7307034)}
```

[20]:
```
import json
from folium import FeatureGroup, Marker, LayerControl

def popform(name, price_mean, price_per_meter):
    p ='<h5> OKRUG  {} price_mean {} price_per_meter {} </h5>'.format(name,␣
 ↪price_mean, price_per_meter)
    return p

full_gf.crs = ({'init' :'epsg:3857'})
full_gf = full_gf.to_crs({'init' :'epsg:4326'})

m = folium.Map(location=[55.764414, 37.647859], zoom_start=9)

a = folium.Choropleth(
    geo_data=gjson,
    name='choropleth',
    data=full_gf[['okrug', 'price_mean']],
    columns=['okrug', 'price_mean'],
    key_on='feature.properties.ABBREV_AO',
    fill_color='YlGnBu',
    line_weight=1,
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Prices',
    highlight = True
)
m.add_child(a)

feature_group = FeatureGroup(name='Some icons')

y = json.loads(gjson)
list_names = []
for mo in y['features']:
    name = mo['properties']['ABBREV_AO']
    if name not in list_names:
        list_names.append(name)
        if (len(df_result[df_result['okrug'] == name]['price_mean']) != 0):
            price_mean = df_result[df_result['okrug'] == name]['price_mean'].
 ↪values[0].round(2)
```

```
            price_per_meter = df_result[df_result['okrug'] ==␣
 ↪name]['mprice_per_meter'].values[0].round(2)
            lat = dict_coordinates2[name][2]
            lon = dict_coordinates2[name][1]
            name_okrug = dict_coordinates2[name][0]
            Marker(location=[lat,lon], popup=popform(name_okrug, price_mean,␣
 ↪price_per_meter), tooltip=dict_coordinates2[name][0]).add_to(feature_group)

feature_group.add_to(m)
LayerControl().add_to(m)

m
```

```
/Users/mary/opt/anaconda3/lib/python3.7/site-packages/pyproj/crs/crs.py:55:
FutureWarning:

'+init=<authority>:<code>' syntax is deprecated. '<authority>:<code>' is the
preferred initialization method. When making the change, be mindful of axis
order changes: https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-
changes-in-proj-6
```

[20]:
```
<folium.folium.Map at 0x1a1e186290>
```

[ ]:
```
m.save('flat_map.html')
```

[ ]:
```

```