

# 租房数据分析

---

2020211376 马天成

## 0. 实验概述

---

### 0.1 题目要求

---

1. 抓取链家官网北上广深 4 个一线城市，再加上一个离你家乡最近的一个非一线城市/或者你最感兴趣的一个城市的数据。应获取每个城市的全部租房数据（一线城市的数据量应该在万的数量级）。
2. 比较 5 个城市的总体房租情况，包含租金的均价、最高价、最低价、中位数等信息，单位面积租金（元/平米）的均价、最高价、最低价、中位数等信息。采用合适的图或表形式进行展示。
3. 比较 5 个城市一居、二居、三居的情况，包含均价、最高价、最低价、中位数等信息。
4. 计算和分析每个城市不同板块的均价情况，并采用合适的图或表形式进行展示。例如上图中的“海淀-四季青-五福玲珑居北区”，“四季青”即为板块名称。
5. 比较各个城市不同朝向的单位面积租金分布情况，采用合适的图或表形式进行展示。哪个方向最高，哪个方向最低？各个城市是否一致？如果不一致，你认为原因是什么？
6. 查询各个城市的人均 GDP，分析并展示其和单位面积租金分布的关系。相对而言，在哪个城市租房的性价比最高？
7. 查询各个城市的平均工资，分析并展示其和单位面积租金分布的关系。相对而言，在哪个城市租房的负担最重？
8. 围绕各城市租房问题，结合上述数据及分析，设计自己感兴趣的数据分析主题，补充查找或爬取相关数据，完成题目设计、数据获取、数据分析及数据展示过程。

### 0.2 报告要求

---

1. 实验报告中应包含爬虫核心代码、核心数据文件的基本结构、数据处理及数据展示的核心代码。代码及文件中应包含足够的注释。
2. 报告应按照一般实验报告要求，至少包含明确的实验目的、过程、结论。
3. 报告以 pdf 格式提交，总页数不超过 30 页。
4. 报告具体提交方式由助教提供。
5. 报告提交截止时间：2022 年 12 月 31 日 24 点。

### 0.3 说明

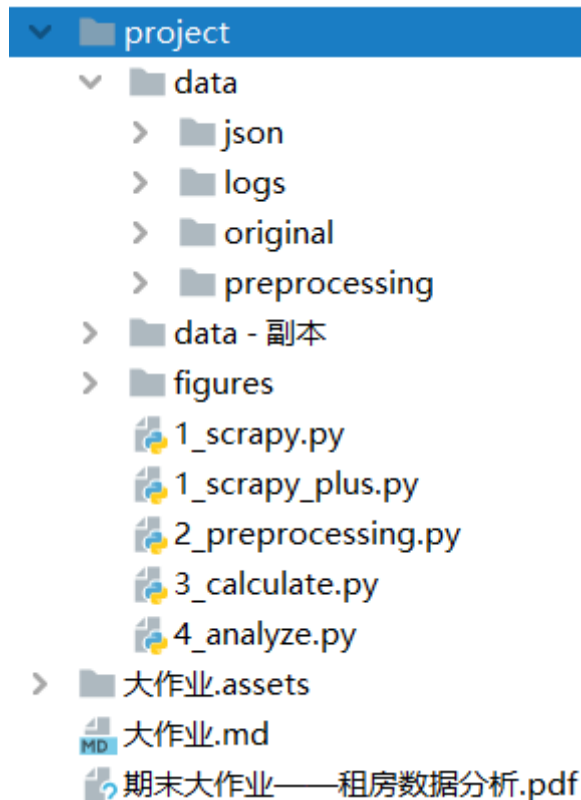
---

1. 我选择 **合肥** 作为我的第五个城市。
2. 我收集了 **人均可支配收入**、**城市租房人数比率**、**城市非户籍常住人口比率** 来进行拓展分析
3. 由于报告30页限制，我将项目说明录制视频的连接放在这里，希望有所帮助：[My Video](#)
4. 我的 Github 仓库[My Repositories \(github.com\)](#)

# 1. 项目概述

---

下面是项目结构：



下面我将简单介绍一下各个文件：

## 1.1 Python代码部分（4个步骤）

---

这里有共有四个Python文件；这些文件的大致功能如下：

1. 数据爬取模块：输入爬取的 html，输出 csv 文件；plus 版是在分析完之后发现 **根据板块获取数据更好** 而做的改进版。
2. 数据预处理模块：输入 csv，输出 csv；将原先的 csv 文件进行 **去重** 和 **计算拓展成列**，生成新的有效 csv 文件，可供分析。
3. 数据计算模块：输入有效的 csv，输出 json；将 csv 的数据进行分析统计，算出**分析阶段需要并且可计算的所有数据**存入 json。
4. 分析模块：输入 json 文件 和 查询数据；进行分析展示，存储图像并展示。

**ATTENTION：**所有的 Python 代码我都是经过：

1. 精心设计，反复修改；
2. 结构工整，思路清晰，无冗余代码；
3. 变量规范，常量定义完整；
4. 绝对不是 **shit mountain!**

## 1.2 data 文件夹（存储中间数据文件）

---

### 1.2.1 original 文件夹

存储了 `1_scrapy_plus.py` 中从网站爬下来的有效数据，根据城市区分，存储为 csv 文件。

### 1.2.2 preprocessing & logs 文件夹

1. 在 `2_preprocessing.py` 进行预处理，中将上一个 original 文件夹中的数据进行 **去重、数据计算拓展**，生成可以直接进行第三部统计的 csv 文件。这个 csv 文件就存储在 **preprocessing 文件夹** 中。
2. 此外，在生成预处理数据数据时，我们会进行 **去重** 和 **删除非法记录**，在这个过程中，我们将：
  1. **记录及其重复次数 & 处理结果** 录入 **城市.log** 中
  2. **记录重复的详细信息** 录入 **城市\_repeat.txt** 中
  3. **非法而删除的记录** 录入 **城市\_error.txt** 中
3. 在介绍预处理代码 - `2_preprocessing.py` 时我们会详细介绍这三个文件的格式和内容。

### 1.2.3 json 文件夹

1. 在 `3_calculate.py` 进行计算，中将上一个 preprocessing 文件夹中的数据进行 **计算**，生成城市的 json 文件，存储在 **json 文件夹**。
2. 这些 json 文件计算了所有在分析阶段需要的数据。该计算已经将所有需要并可以计算的数据都计算出（一些在分析阶段引入的新数据需要重新计算），以字典形式存放为 json；此后就 **不再需要 csv 文件**，只需要用 json 文件进行分析，大大减少数据量。

## 1.3 figure 文件夹

---

可算作分析结果。

存储了 `4_analyze.py` 的输出图像。

## 1.4 data - 副本 文件夹

---

存储了老版数据爬取方式（在城市主页用综合排序页面爬取）的数据。这个数据是比用城区爬取的数据差很多的。

## 2. 实验内容

---

### 2.1 数据爬取模块

---

`1_scrapy_plus.py`

```
from urllib.request import urlopen
```

```

from bs4 import BeautifulSoup
import csv

# target cities
cities = {'北京': 'bj', '上海': 'sh', '广州': 'gz', '深圳': 'sz', '合肥': 'hf'}

# csv outputFile attributes
csv_attributes = ['title', 'name', 'area', 'rooms', 'price_lower',
'price_upper']

# ***
records_in_a_page = 30

# get cities rent data
for name, info in cities.items():
    # root url path
    url_root = 'https://' + info + '.lianjia.com'

    # generate name.csv -----
    with open('data/original/'+name+".csv", 'w', newline='', encoding='utf-8-
sig') as outputFile:
        writer = csv.writer(outputFile)
        writer.writerow(csv_attributes)

    # 1, get districts path in this city -----
    html = BeautifulSoup(urlopen(url_root + '/zufang'), features="lxml")
    # find all district blocks
    blocks = html.findAll(name='li', attrs={'class': 'filter__item--level2',
'data-type': 'district'})
    # get all district path (ignore the first district-'不限')
    district_path_list = list()
    for i in range(1, len(blocks)):
        district_path_list.append(blocks[i].find('a')['href'])

    # 2. get data each from each district pages -----
    for path in district_path_list:
        # 2.1 get page size for this city from city's index page
        html = BeautifulSoup(urlopen(url_root + path), features="lxml")
        size_record = int(html.findAll(name="span", attrs={"class":
"content__title--h1"})[0].text)
        size_page = int(size_record / records_in_a_page)
        if( size_record % records_in_a_page > 0 ):
            size_page = size_page+1
        print(path, size_record, size_page)

        # 2.2 scrapy data in pages & copy data into file
        for i in range(1, size_page+1):
            print(i)
            html = BeautifulSoup(urlopen(url_root + path + 'pg' + str(i)),
features="lxml")

            # get data into list
            title_block = html.findAll("p", {"class": "content__list--item--
title"})

```

```

des_block = html.findAll("p", {"class": "content_list--item--des"})

price_block = html.findAll("span", {"class": "content_list--item-price"})

for title, des, price in zip(title_block, des_block, price_block):
    # solve title
    title = title.get_text().strip()
    # solve des
    des_list = des.get_text().split('/')
    if len(des_list) == 5:
        name = des_list[0].replace("\n", "").strip()
        area = des_list[1].replace("\n", "").strip()[:-1]
        rooms = des_list[3].replace("\n", "").strip()[0]
    elif len(des_list) == 3:
        name = ""
        area = des_list[0].replace("\n", "").strip()[:-1]
        rooms = des_list[2].replace("\n", "").strip()[0]
    # solve price
    text = price.get_text().split(' ')[0]
    if '-' in text:
        text = text.split('-')
        price_lower = text[0]
        price_upper = text[1]
    else:
        price_lower = text
        price_upper = text
    writer.writerow([title, name, area, rooms, price_lower, price_upper])

```

这里不提供原版的代码了，在附录中会提供。

## 2.1.1 代码过程分析

1. 在开头给出五个城市的 **命名和简写对照 dict**，方便自动遍历和爬取，用循环减少无谓的相似代码。
2. 再给出 csv 文件需要爬取的各个属性（属性行），以及一个页面的租房记录数：30。
3. 打开需要生成的文件：**data/original/城市名.csv**，并写入 csv 的属性行 csv\_attributes
4. **循环**每个城市：
  1. 获取当前页面的 **地区标签相对 path**（藏于 href），存入 **district\_list**。
  2. 根据 **district\_list** 的元素到达每个子页面。获得当前的页面的租房数据数量，计算出该 district 遍历完所需的页面数：**记录数 / 单个页面记录数**（开头已给出定义）。然后**循环**进入每个该地区的顺序页面：
    1. 获取当前城市的html文件；
    2. 根据当前 html体，获取30个记录需要录入的数据，存为 list。循环该 list：
      1. 分析数据，将记录数据拆开，化成 attribute 的形式，一个个**写入 csv** 中。

## 2.1.2 数据转化细节分析

只需要在处理时取出相应数据，然后取出即可。下面有几个数据爬取策略

1. 几室：取出字段出去前导换行空格后的第一个字符。
2. 面积：删除前导后导空格换行，并删除最后一个字符。

## 2.2 数据预处理模块

### 2\_preprocessing.py

```
import pandas as pd
import csv

# target cities
cities = {'北京': 'bj', '上海': 'sh', '广州': 'gz', '深圳': 'sz', '合肥': 'hf'}

# csv outputFile attributes
csv_attributes = ['title', 'name', 'area', 'rooms',
                  'price_lower', 'price_upper', 'price_average',
                  'price_square_lower', 'price_square_upper',
                  'price_square_average']

# preprocessing each city.csv with "deduplicate" and "append"
for name, info in cities.items():

    # read data & convert into list -----
    data = pd.read_csv('data/original/'+name+'.csv', encoding='utf-8-sig')
    data = data.values.tolist()
    original_size = len(data)
    print('preprocessing', name, original_size)

    # open log and error file
    logFile = open('data/logs/' + name + ".log", 'w', encoding='utf-8')
    repeatFile = open('data/logs/' + name + "_repeat.txt", 'w', encoding='utf-8')
    errorFile = open('data/logs/' + name + "_error.txt", 'w', encoding='utf-8')

    # generate deduplicated data & write into file -----
    with open('data/preprocessing/' + name + ".csv", 'w', newline='',
              encoding='utf-8-sig') as outputFile:
        writer = csv.writer(outputFile)
        writer.writerow(csv_attributes)

    # generate price_square_lower, price_square_upper & deduplicate
    i = 0
    cnt_delete = 0
    while i < len(data):
        # solve illegal records depend on rooms
        if str(data[i][3]).isdigit()==False or int(data[i][3])<=0:
            # write into error.log
            errorFile.write(str(data[i])+'\n')
```

```

        del data[i]
        cnt_delete += 1
        continue

    # deduplicate
    #print('deduplicate', i)
    cnt_repeat = 0
    j = i+1
    while j < len(data):
        # find repeat
        if data[i] == data[j]:
            # write into repeat.log
            repeatFile.write('['+str(i)+' , '+str(j)+' ]'+str(data[j])+'\n')
            del data[j]
            j -= 1
            cnt_repeat += 1
            cnt_delete += 1
        j += 1
    # for data[i], repeat cnt times
    if cnt_repeat > 0:
        logFile.write('repeat '+str(cnt_repeat)+' : '+str(data[i])+'\n')

    # append price_average, price_square_lower, price_square_upper,
    price_square_average
    if '-' in str(data[i][2]):
        area = str(data[i][2]).split('-')
        area = (float(area[0]) + float(area[1])) / 2
    else:
        area = float(data[i][2])
    data[i].append((float(data[i][4]) + float(data[i][5])) / 2)
    data[i].append(float(data[i][4]) / area)
    data[i].append(float(data[i][5]) / area)
    data[i].append((float(data[i][7]) + float(data[i][8])) / 2)
    # has deduplicated this record with writing into csv
    writer.writerow(data[i])

    i += 1

    # write delete counts
    summary = 'delete(repeat&illegal) : '+str(cnt_delete)+' | original'+str(original_size)+' remain '+str(len(data))
    logFile.write(summary)
    print(name, summary)

    # close log and error
    logFile.close()
    errorFile.close()

```

## 2.2.1 代码过程分析

1. 在开头给出五个城市的 **命名和简写对照 dict**，方便自动遍历和爬取，用循环减少无谓的相似代码。
2. 再给出扩展后的 csv 文件需要爬取的各个属性（属性行）
3. 打开需要生成的文件：**data/preprocessing/城市名.csv**，并写入 csv 的属性行 csv\_attributes
4. **循环**每个城市：
  1. 读取这个城市的 original csv 文件并读出。
  2. 循环每一条记录：
    1. 当前记录不合法：**几室 - 录入了“未”字 || 数据为0**。删除该数据，放入 **error.txt** 中，continue。
    2. **计算该条数据需要拓展的列数据（属性）**
    3. 从下一条记录开始找到最后，**循环**到最后一条：
      1. 假如记录和循环外的重复，则记录为重复，放入 **repeat.txt** 中。
    4. 假如当前的记录找到了重复，则把该条记录和重复次数放入 **城市名.log** 中。
  3. 将原记录数，删除记录数，剩余记录数最后再放入 **城市名.log** 中。
  4. 将删除后的数据写入新的 csv 文件：**data/preprocessing/城市名.csv**

## 2.2.2 输出文件格式样式

城市.log

```
广州.log [D:\Course\Term5 Python程序设计实践\Homework\8_大作业\project\data\logs] - Notepad3
文件(F) 编辑(E) 查看(V) 外观(P) 设置(S) 帮助(H)
6887 repeat 20 : ['整租·碧桂园天玺湾 3室2厅 东南', '南沙-进港大道-碧桂园天玺湾', '101.00', '3', 2600, 2600]
6888 repeat 20 : ['整租·南沙建滔广场 3室2厅 西', '南沙-南沙区政府-南沙建滔广场', '138.00', '3', 8970, 8970]
6889 repeat 20 : ['整租·时代维港 1室1厅 复式 东南', '南沙-进港大道-时代维港', '59.00', '1', 1800, 1800]
6890 repeat 20 : ['整租·南沙保利南怡湾 3室1厅 西北', '南沙-黄阁-南沙保利南怡湾', '84.00', '3', 1800, 1800]
6891 repeat 20 : ['整租·越秀滨海新城 3室2厅 北', '南沙-进港大道-越秀滨海新城', '97.00', '3', 2200, 2200]
6892 repeat 20 : ['整租·大岗翡翠蓝湾北区 3室2厅 东北', '南沙-大岗镇-大岗翡翠蓝湾北区', '102.00', '3', 2150, 2150]
6893 repeat 20 : ['整租·南沙湾御苑 3室2厅 东南', '南沙-南沙港-南沙湾御苑', '95.00', '3', 2200, 2200]
6894 repeat 20 : ['整租·阳光城丽景湾领悦 2室2厅 东南', '南沙-进港大道-阳光城丽景湾领悦', '89.00', '2', 2000, 2000]
6895 repeat 16 : ['整租·越秀国际总部广场 1室1厅 东/东南', '南沙-金洲-越秀国际总部广场', '50.00', '1', 1800, 1800]
6896 repeat 16 : ['整租·叠翠峰 3室2厅 东南', '南沙-南沙区政府-叠翠峰', '90.00', '3', 2900, 2900]
6897 repeat 16 : ['整租·万科南方公元 3室2厅 南', '南沙-黄阁-万科南方公元', '96.00', '3', 2500, 2500]
6898 repeat 16 : ['整租·越秀滨海悦城 3室2厅 南', '南沙-金洲-越秀滨海悦城', '117.00', '3', 3200, 3200]
6899 delete(repeat&illegal) : 72756 | original 97492 remain 24736
行 1 / 6,899 列 1 / 76 字符 1 / 76 求值 -- 选定 -- 选行 -- 匹配 -- 796 KB Unicode (UTF-8) CR+LF INS STD 文本文件
```

repeat.txt

```
广州_repeat.txt [D:\Course\Term5 Python程序设计实践\Homework\8_大作业\project\data\logs] - Notepad3
文件(F) 编辑(E) 查看(V) 外观(P) 设置(S) 帮助(H)
1 [34, 654] ['整租·尚东君御雅苑 1室0厅 北', '天河-珠江新城-尚东君御雅苑', '39.00', '1', 6200, 6200]
2 [34, 1075] ['整租·尚东君御雅苑 1室0厅 北', '天河-珠江新城-尚东君御雅苑', '39.00', '1', 6200, 6200]
3 [48, 510] ['整租·南国花园 3室2厅 东南/西', '天河-珠江新城-南国花园', '108.00', '3', 10000, 10000]
4 [54, 68] ['整租·双城国际 1室0厅 北', '天河-珠江新城-双城国际', '34.00', '1', 4000, 4000]
5 [56, 62] ['整租·富力盈丰大厦 1室0厅 北', '天河-珠江新城-富力盈丰大厦', '51.00', '1', 5500, 5500]
6 [58, 60] ['整租·汇金国际金融中心 1室0厅 西', '天河-金融城-汇金国际金融中心', '42.00', '1', 5800, 5800]
7 [62, 1705] ['整租·御华苑 3室2厅 南/北', '天河-龙口东-御华苑', '108.59', '3', 5700, 5700]
8 [81, 87] ['整租·南国嘉园 3室2厅 西南', '天河-东圃-南国嘉园', '89.00', '3', 4420, 4420]
9 [92, 2731] ['整租·东圃广场 3室1厅 东/西', '天河-东圃-东圃广场', '77.00', '3', 3900, 3900]
10 [98, 700] ['整租·天雅阁 2室2厅 南', '天河-天润路-天雅阁', '83.29', '2', 5000, 5000]
11 [102, 2538] ['整租·翠屏富通雅苑 2室2厅 北', '天河-东圃-翠屏富通雅苑', '70.77', '2', 4200, 4200]
12 [105, 1951] ['整租·利雅湾 3室2厅 北', '天河-珠江新城-利雅湾', '125.00', '3', 13500, 13500]
13 [106, 2355] ['整租·华翰大厦 2室1厅 东南', '天河-体育中心-华翰大厦', '80.00', '2', 7800, 7800]
14 [108, 381] ['整租·名门大厦 2室1厅 南', '天河-珠江新城-名门大厦', '117.00', '2', 8000, 8000]
行 1 / 72,612 列 1 / 75 字符 1 / 75 求值 -- 字节 -- 选行 -- 匹配 -- 8.38 MB Unicode (UTF-8) CR+LF INS STD 文本文件
```

error.txt





```

        price_square_total = price_square_total + float(record[9])
        price_square_lower = min(float(record[7]), price_square_lower)
        price_square_upper = max(float(record[8]), price_square_upper)

# calculate average and middle
price_average = price_total / len(data)
sorted(data, key=(lambda x: x[6])) # sort by price_average
index = int(len(data)/2)
if len(data) & 1 == 1:
    price_middle = float(data[index][6])
else:
    price_middle = (float(data[index][6]) + float(data[index+1][6])) / 2

# calculate price_square average and middle for room_i
price_square_average = price_square_total / len(data)
sorted(data, key=(lambda x: x[9])) # sort by price_square_average
index = int(len(data)/2)
if len(data) & 1 == 1:
    price_square_middle = float(data[index][9])
else:
    price_square_middle = (float(data[index][9]) + float(data[index+1][9]))
/ 2

# store calculate data into json
city_dict['size'] = len(data)
city_dict['price'] = {'average': price_average,
                     'lower': price_lower,
                     'upper': price_upper,
                     'middle': price_middle}
city_dict['price_square'] = {'average': price_square_average,
                             'lower': price_square_lower,
                             'upper': price_square_upper,
                             'middle': price_square_middle}

# generate city_json[3:5] (room1:3 calculate) -----
# variables (list for room_1, room_2, room_3)
rooms = [list(), list(), list()]
price_total = [0, 0, 0]
price_lower = [0x3F3F3F, 0x3F3F3F, 0x3F3F3F]
price_upper = [0, 0, 0]
# judge for each record
for record in data:
    if record[3] <= 3:
        index = record[3]-1
        # store this record
        rooms[index].append(record)
        # update price-total, lower, upper
        price_total[index] = price_total[index] + float(record[6])
        price_lower[index] = min(float(record[4]), price_lower[index])
        price_upper[index] = max(float(record[5]), price_upper[index])

# variables
price_average = [0, 0, 0]
price_middle = [0, 0, 0]
for i in range(3):

```

```

# calculate average and middle
price_average[i] = price_total[i]/len(rooms[i])
sorted(rooms[i], key=(lambda x: x[6])) # sort by price_average
index = int(len(rooms[i])/2)
if len(rooms[i])&1 == 1:
    price_middle[i] = float(rooms[i][index][6])
else:
    price_middle[i] = float(float(rooms[i][index][6])+float(rooms[i]
[index+1][6]))/2

# store calculate data into city.json
key = 'room_'+str(i+1)
city_dict[key] = {'size': len(rooms[i]),
                  'average': price_average[i],
                  'lower': price_lower[i],
                  'upper': price_upper[i],
                  'middle': price_middle[i]}

# generate city_json[6:9] (north & south & west & east) -----
size = [0, 0, 0, 0]
price_square_total = [0, 0, 0, 0]
for record in data:
    if '北' in record[0]:
        size[0] += 1
        price_square_total[0] += float(record[9])
    if '南' in record[0]:
        size[1] += 1
        price_square_total[1] += float(record[9])
    if '西' in record[0]:
        size[2] += 1
        price_square_total[2] += float(record[9])
    if '东' in record[0]:
        size[3] += 1
        price_square_total[3] += float(record[9])

# store calculate data into city.json
city_dict['North'] = {'size': size[0], 'price_square_average':
price_square_total[0] / size[0]}
city_dict['South'] = {'size': size[1], 'price_square_average':
price_square_total[1] / size[1]}
city_dict['West'] = {'size': size[2], 'price_square_average':
price_square_total[2] / size[2]}
city_dict['East'] = {'size': size[3], 'price_square_average':
price_square_total[3] / size[3]}

# generate city_json[10] (plate) -----
# store plates' name
plate_set = set()
for record in data:
    # cause some records has no plate!!!
    if type(record[1]) == str and record[1] != 'nan':
        plate_set.add(record[1].split('-')[1])

# generate each plates' price_average
plate_list = list(plate_set)

```

```

plate_size = [0] * len(plate_list)
plate_total = [0] * len(plate_list)
for record in data:
    if type(record[1]) == str and record[1] != 'nan':
        index = plate_list.index(record[1].split('-')[1])
        plate_size[index] += 1
        plate_total[index] += record[6]
plate_dict = dict()
for i in range(len(plate_list)):
    plate_dict[plate_list[i]] = {'size': plate_size[i], 'average':
plate_total[i] / plate_size[i]}

# store calculate data into city.json
city_dict['plates'] = {'size': len(plate_list), 'data': plate_dict}

# write city.json -----
with open('data/json/' + name + ".json", 'w') as jsonFile:
    jsonFile.write(json.dumps(city_dict, indent=1, ensure_ascii=False))

```

### 2.3.1 代码过程分析

1. 在开头给出五个城市的 **命名和简写对照 dict**，方便自动遍历和爬取，用循环减少无谓的相似代码。
2. 循环每个城市：
  1. 读入文件：**data/preprocessing/城市名.csv**
  2. 计算 **租房价格** 的：平均值，最小值，最大值，中位数。
  3. 计算 **单位面积租房价格** 的：平均值，最小值，最大值，中位数。
  4. 计算一室二室三室的 **租房价格** 的：平均值，最小值，最大值，中位数。
  5. 计算东南西北 **单位面积租房价格** 的：平均值，最小值，最大值，中位数。
  6. 计算出板块集合，循环每一个板块：
    1. 找到所有的板块，记录相应数据。
  7. 算出各个板块的 **平均单位面积租房价格**。
  8. 将上述数据存入 **data/json/城市.json** 中。

### 2.3.1 json 文件样例



```

for name in city_name:
    with open(path_json + name + '.json', 'r') as f:
        data_dict[name] = json.load(f)

# analyze preparations -----

# set each data size
city_size = 5
price_size = 4
room_size = 3
direction_size = 4

# set keys to find in data_dict
price_keys = ['average', 'lower', 'upper', 'middle']
room_keys = ['room_1', 'room_2', 'room_3']
direction_keys = ['North', 'South', 'West', 'East']

# set limits (has manually selected from json)
price_limit = 400000
price_square_limit = 3000

# set color list
color_city = ['orange', 'deepskyblue', 'orchid', 'g', 'y']
color_room = ['orange', 'deepskyblue', 'orchid', 'g']
color_direction = ['orange', 'deepskyblue', 'orchid', 'g']

# set pie label
pie_room = ['room_1', 'room_2', 'room_3', 'room_>=4'] # add last two show

# set font
plt.rcParams['font.sans-serif'] = ['SimHei']

# Task 1: compare 5 cities -----

# 1.1: compare 5 cities with price

plt.figure(figsize=(40, 20))
# find each price category with i (generate 4 sub figures)
for i in range(price_size):
    # basic settings
    plt.subplot(1, price_size, i + 1)
    plt.title('Price-' + price_keys[i], fontsize=30)
    plt.xlabel('cities', fontsize=15)
    plt.ylabel('price(¥/m^2)', fontsize=15)

    # generate x & y data
    x = [i for i in range(1, city_size+1)]
    y = list()
    for name in city_name:
        y.append(data_dict[name]['price'][price_keys[i]])

```

```

# generate subplot
plt.ylim((0, price_limit))
plt.bar(x, y, color=color_city)
# add text
plt.xticks(x, city_name, size=30)
for x_value, y_value in zip(x, y):
    plt.text(x_value, y_value, '%.4f' % y_value, ha='center', fontsize=15)

# print & save plot
plt.suptitle('Price Compare', fontsize=30)
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()

```

# 1.2: compare 5 cities with price\_average

```

plt.figure(figsize=(40, 20))
# find each price category with i (generate 4 sub figures)
for i in range(price_size):
    # basic settings
    plt.subplot(1, price_size, i + 1)
    plt.title('Price_square-' + price_keys[i], fontsize=30)
    plt.xlabel('cities', fontsize=15)
    plt.ylabel('price(¥/m^2)', fontsize=15)

    # generate x & y data
    x = [i for i in range(1, city_size+1)]
    y = list()
    for name in city_name:
        y.append(data_dict[name]['price_square'][price_keys[i]])

    # generate subplot
    plt.ylim((0, price_square_limit))
    plt.bar(x, y, color=color_city)
    # add text
    plt.xticks(x, city_name, size=30)
    for x_value, y_value in zip(x, y):
        plt.text(x_value, y_value, '%.4f' % y_value, ha='center', fontsize=15)

# print & save plot
plt.suptitle('Price Square Compare', fontsize=30)
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()

```

# Task 2: compare each cities' room1,2,3 -----

# 2.1: self compare rate

```

plt.figure(figsize=(40, 20))
# find each city with i (generate 5 sub figures)
for i in range(city_size):

```

```

# basic settings
plt.subplot(1, city_size, i + 1)
plt.title(city_name[i], fontsize=100)

# generate room1, 2, 3, >=4 sizes
sizes = list()
for j in range(room_size):
    sizes.append(100 * data_dict[city_name[i]][room_keys[j]]['size'] /
data_dict[city_name[i]]['size'])
# generate the rest size for >=4
total_123 = 0
for item in sizes:
    total_123 += item
sizes.append(100-total_123)

# generate subplot with text
patches, l_text, p_text = \
    plt.pie(sizes, labels=pie_room, colors=color_room, autopct='%1.1f%%',
startangle=90)
plt.axis('equal')
# adjust words size
for item in p_text:
    item.set_size(20)

# print & save plot
plt.suptitle('Rooms Components', fontsize=30)
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()

# 2.2: compare room1, 2, 3 between cities depend on price

# find room_1, 2, 3 with i (generate 3 figures)
for i in range(room_size):
    plt.figure(figsize=(40, 20))

# find price category with i (generate 4 sub figures)
for j in range(price_size):
    # basic settings
    plt.subplot(1, price_size, j + 1)
    plt.title('Price-' + price_keys[j], fontsize=30)
    plt.xlabel('cities', fontsize=15)
    plt.ylabel('price(¥/m^2)', fontsize=15)

    # generate x & y data
    x = [i for i in range(1, city_size+1)]
    y = list()
    # find each city - room_i+1 - price_key - price number
    for name in city_name:
        y.append(data_dict[name][room_keys[i]][price_keys[j]])

    # generate subplot
    plt.bar(x, y, color=color_city)
    # add text

```



```

plt.xticks(x, city_name, size=15)
for x_value, y_value in zip(x, y):
    plt.text(x_value, y_value, '%.4f' % y_value, ha='center',
fontsize=15)

# print & save plot
plt.suptitle(room_keys[i]+' Compare (no limit to flush y)', fontsize=30)
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()

# Task 3: compare each cities' plates -----

# compare each city's plates (generate 5 figures)
for name in city_name:
    plt.figure(figsize=(40, 20))
    # basic settings
    plt.title(name+' plates average price', fontsize=30)
    plt.xlabel('plates', fontsize=15)
    plt.ylabel('price(¥/m^2)', fontsize=15)

    # select plates witch has big size(top 10 percent)
    temp = data_dict[name]['plates']['data']    # get all plates' information
    # store each plates tuple(name, size)
    temp_list = list()
    for info_name, info_data in temp.items():
        temp_list.append([info_name, info_data['size']])
    sorted(temp_list, key=(lambda x: x[1]), reverse=True)

    # generate x data
    x_size = int(data_dict[name]['plates']['size'] / 10)
    x = [i for i in range(1, x_size+1)]

    # generate x label and y data
    x_name = list()
    y = list()
    for i in range(x_size):
        info_name = temp_list[i][0]
        x_name.append(info_name)
        y.append(data_dict[name]['plates']['data'][info_name]['average'])

    # generate subplot
    plt.bar(x, y, color=color_city)
    # add text
    plt.xticks(x, x_name, size=30)
    for x_value, y_value in zip(x, y):
        plt.text(x_value, y_value, '%.4f' % y_value, ha='center', fontsize=15)

    # print & save plot
    plt.savefig(path_figure + str(cnt_figure))
    cnt_figure += 1
    plt.show()

```

# Task 4: compare each cities' directions -----

# compare each city's plates (generate 5 figures)

```
plt.figure(figsize=(40, 20))
for i in range(city_size):
    # basic settings
    plt.subplot(1, city_size, i+1)
    plt.title(city_name[i]+' directions\naverage square price', fontsize=30)
    plt.xlabel('direction', fontsize=15)
    plt.ylabel('price(¥/m^2)', fontsize=15)

    # generate x data
    x = [i for i in range(1, direction_size+1)]

    # generate y data
    y = list()
    for info in direction_keys:
        y.append(data_dict[city_name[i]][info]['price_square_average'])

    # generate subplot
    plt.bar(x, y, color=color_direction)
    # add text
    plt.xticks(x, direction_keys, size=30)
    for x_value, y_value in zip(x, y):
        plt.text(x_value, y_value, '%.4f' % y_value, ha='center', fontsize=15)

# print & save plot
plt.suptitle('Direction Compare (no limit to flush y)', fontsize=30)
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()
```

# Task 5: compare each cities' GDP & price\_square -----

```
GDP = {'北京': 183980, '上海': 173630, '广州': 151200, '深圳': 174600, '合肥': 121800}
```

'''ATTENTION:

we suppose that each person need rent a room with 20m^2  
such we induct a rate to represent:  
20m^2 room price / GDP

'''

# generate this rate

```
square = 20
months = 12
rates = list()
for name, number in GDP.items():
    rates.append(months * square * data_dict[name]['price_square']['average'] /
number)
```

# basic settings

```

plt.figure(figsize=(40, 20))
plt.title('Compare GDP & price_square\n with rate: 20m^2 room price / GDP',
fontsize=30)
plt.xlabel('cities', fontsize=30)
plt.ylabel('rate(1)', fontsize=30)

# generate x & y data
x = [i for i in range(1, city_size+1)]
y = rates

# generate subplot
plt.bar(x, y, color=color_city)
# add text
plt.xticks(x, city_name, size=30)
for x_value, y_value in zip(x, y):
    plt.text(x_value, y_value, '%.4f' % y_value, ha='center', fontsize=30)

# print & save plot
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()

# Task 6: compare each cities' Income & price_square -----

Income = {'北京': 127535, '上海': 136752, '广州': 118133, '深圳': 153471, '合肥':
104729}

'''ATTENTION:
    we suppose that each person need rent a room with 20m^2
    such we induct a rate to represent:
        20m^2 room price / Income
...

# generate this rate
square = 20
months = 12
rates = list()
for name, number in Income.items():
    rates.append(months * square * data_dict[name]['price_square']['average'] /
number)

# basic settings
plt.figure(figsize=(40, 20))
plt.title('Compare Income & price_square\n with rate: 20m^2 room price /
Income', fontsize=30)
plt.xlabel('cities', fontsize=30)
plt.ylabel('rate(1)', fontsize=30)

# generate x & y data
x = [i for i in range(1, city_size+1)]
y = rates

# generate subplot

```

```

plt.bar(x, y, color=color_city)
# add text
plt.xticks(x, city_name, size=30)
for x_value, y_value in zip(x, y):
    plt.text(x_value, y_value, '%.4f' % y_value, ha='center', fontsize=30)

# print & save plot
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()

# Task 7: further analyze -----

# 人均可支配收入（税后）
Income_Discretionary = {'北京': 75002, '上海': 78027, '广州': 74416, '深圳': 70847,
                        '合肥': 46009}

big_city_name = ['北京', '上海', '广州', '深圳']
# 租房人数比例（未查询到合肥）
Rate_rent = {'北京': 0.333, '上海': 0.384, '广州': 0.508, '深圳': 0.768}
# 非本地户籍人口比例
Rate_inward = {'北京': 0.385, '上海': 0.421, '广州': 0.489, '深圳': 0.665}

# Q1: Continue GDP and Income, how about Income_Discretionary? -----

'''ATTENTION:
    we suppose that each person need rent a room with 20m^2
    such we induct a rate to represent:
        20m^2 room price / Income_Discretionary
...

# generate this rate
square = 20
months = 12
rates = list()
for name, number in Income_Discretionary.items():
    rates.append(months * square * data_dict[name]['price_square']['average'] /
                 number)

# basic settings
plt.figure(figsize=(40, 20))
plt.title('Compare Income & price_square\n with rate: 20m^2 room price /
Income_Discretionary', fontsize=30)
plt.xlabel('cities', fontsize=30)
plt.ylabel('rate(1)', fontsize=30)

# generate x & y data
x = [i for i in range(1, city_size+1)]
y = rates

# generate subplot
plt.bar(x, y, color=color_city)

```

```

# add text
plt.xticks(x, city_name, size=30)
for x_value, y_value in zip(x, y):
    plt.text(x_value, y_value, '%.4f' % y_value, ha='center', fontsize=30)

# print & save plot
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()

# Q2: How about Rate_rent and Rate_inward? -----

'''ATTENTION:
    we suppose that each person need rent a room with 20m^2
    such we induct a rate to represent:
        20m^2 room price / Income_Discretionary
...'''

# generate this rate: rent in inward
rent_in_inward = list()
for s, t in zip(Rate_rent.values(), Rate_inward.values()):
    rent_in_inward.append(s/t)

# basic settings
plt.figure(figsize=(40, 20))
plt.title('Rate_rent / Rate_inward', fontsize=30)
plt.xlabel('cities', fontsize=30)
plt.ylabel('rate(1)', fontsize=30)

# generate x & y data
x = [i for i in range(1, len(rent_in_inward)+1)]
y = rent_in_inward

# generate subplot
plt.bar(x, y, color=color_city)

# add text
plt.xticks(x, big_city_name, size=30)
for x_value, y_value in zip(x, y):
    plt.text(x_value, y_value, '%.4f' % y_value, ha='center', fontsize=30)

# print & save plot
plt.savefig(path_figure + str(cnt_figure))
cnt_figure += 1
plt.show()

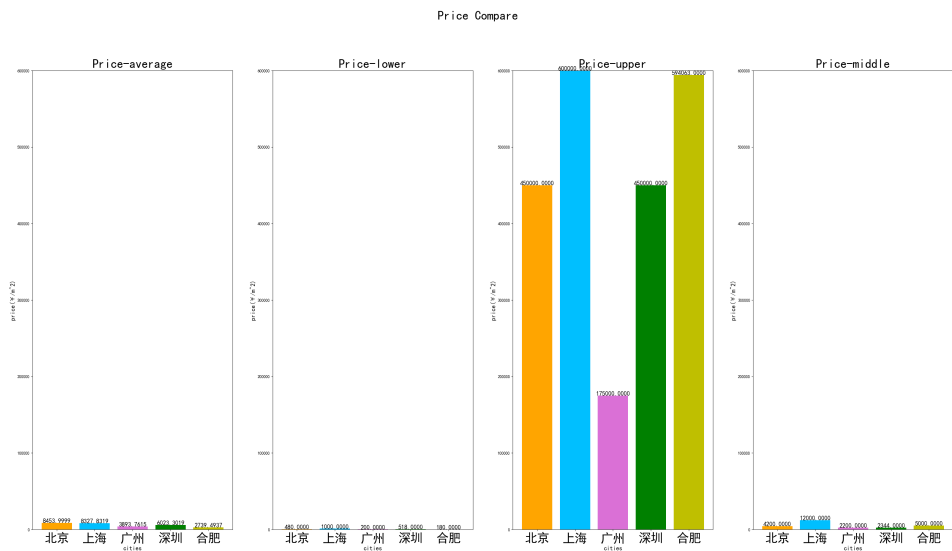
```

过程这里就不详细分析了，其实就是根据 json 画图。在下一个实验结果分析中详细阐述。

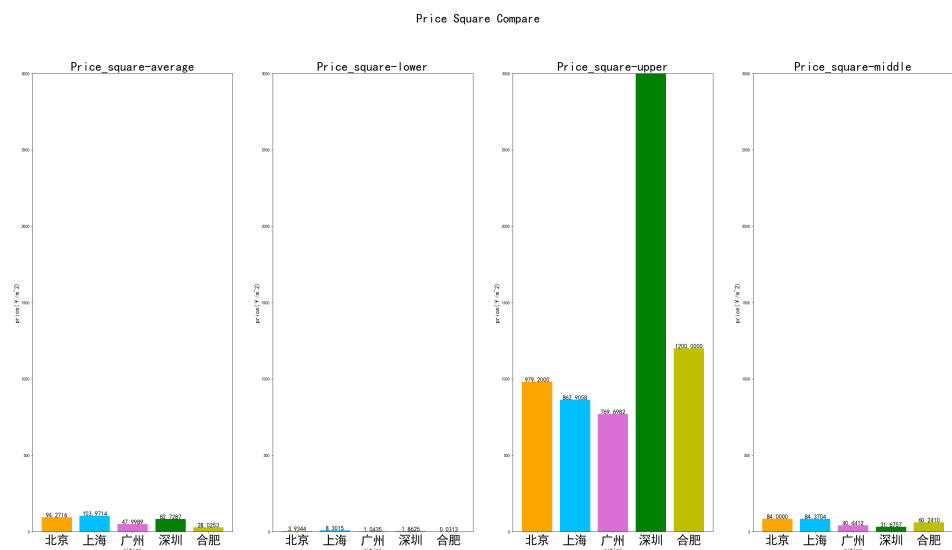
### 3. 实验结果分析

# Q1：比较 5 个城市的总体房租情况

价格：



单位面积价格：



主要比较北京上海：

1. 北京平均价格偏高但中位数价格低很多：上海住房价格较为集中，北京有更多高房价租房。
2. 单位面积上海比北京高，上海住房面积小。

# Q2：比较 5 个城市一居、二居、三居的情况

各个占比：

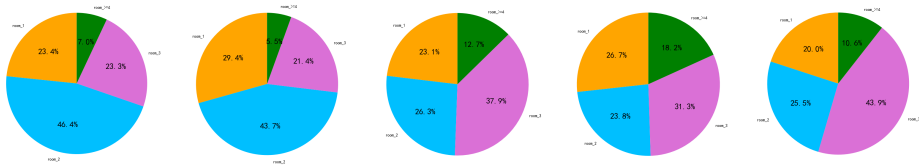
北京

上海

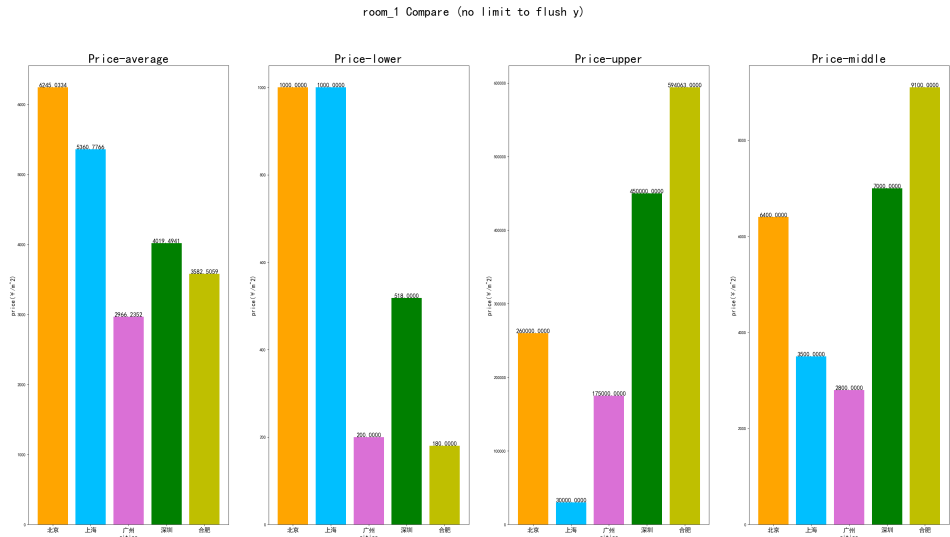
广州

深圳

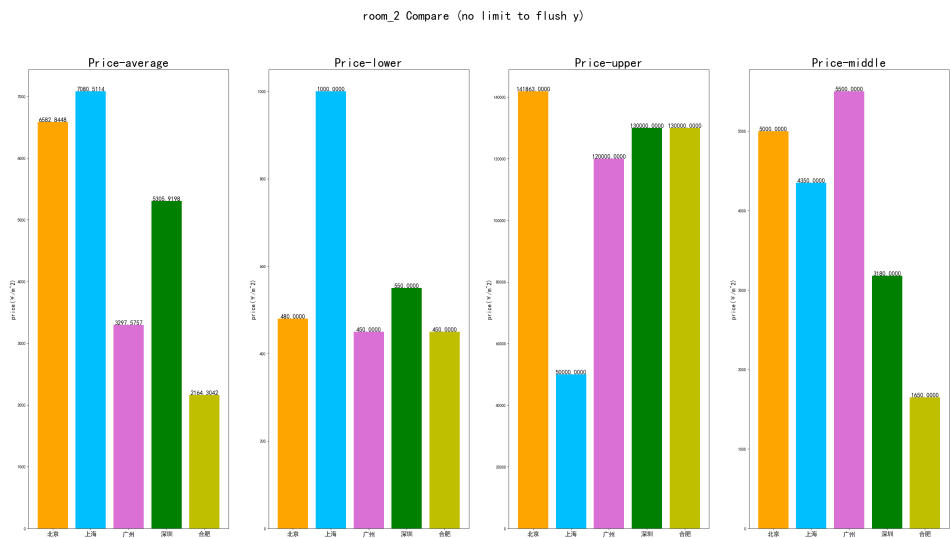
合肥



一室价格:



二室价格:



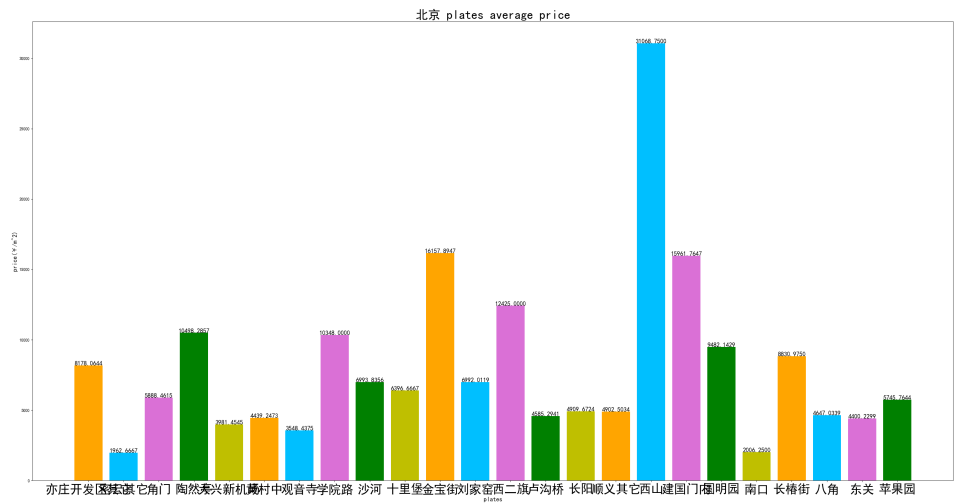
三室价格:

- 1. 北京一室很贵，上海二室三室很贵
- 2. 合肥一室很贵，广州二室三室很贵，深圳比他俩都高但比上海北京低。

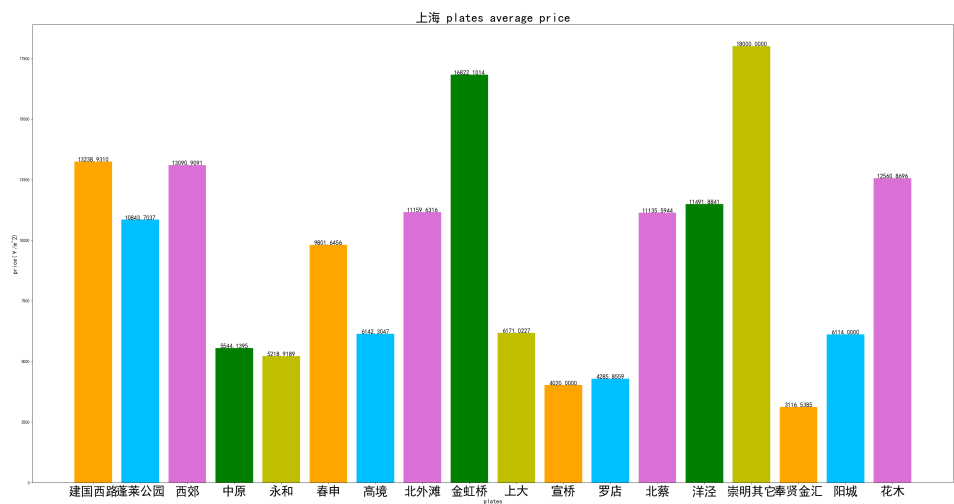
### Q3：计算和分析每个城市不同板块的均价情况

ATTENTION：由于板块数量巨多，我们只选取前 10% 展示

北京：

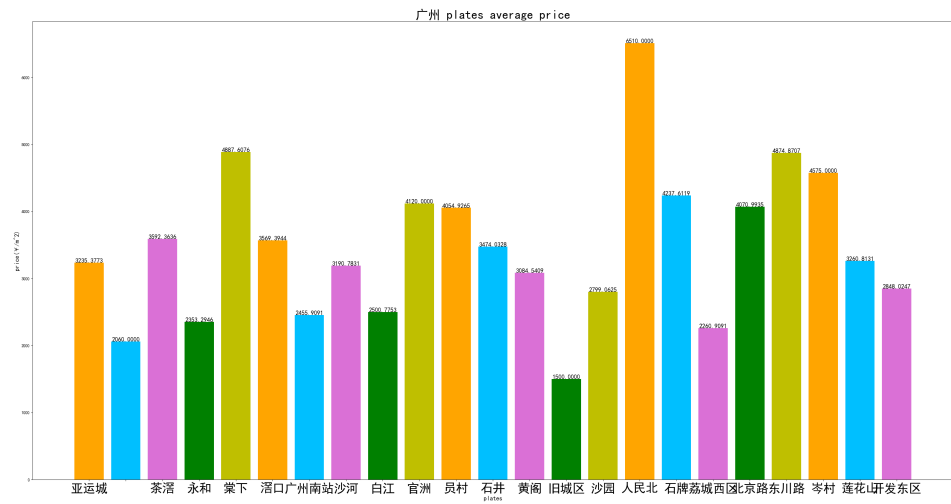


上海：

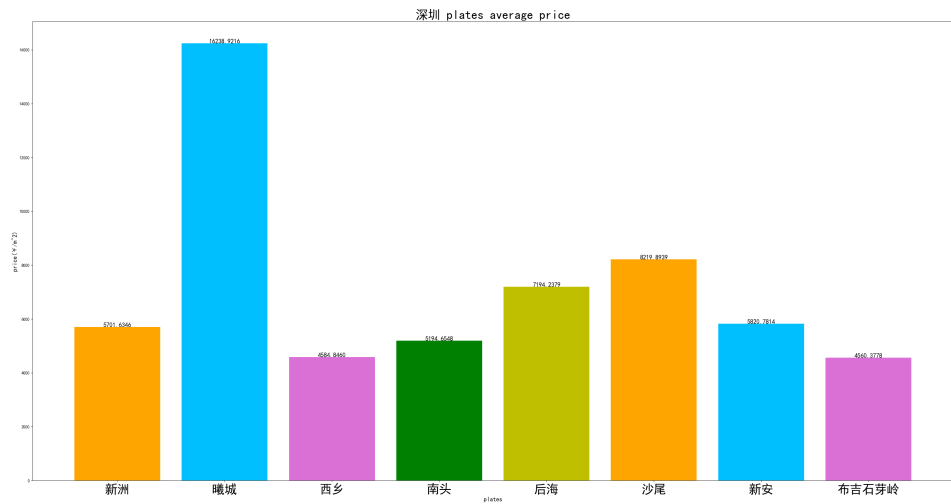


广州：

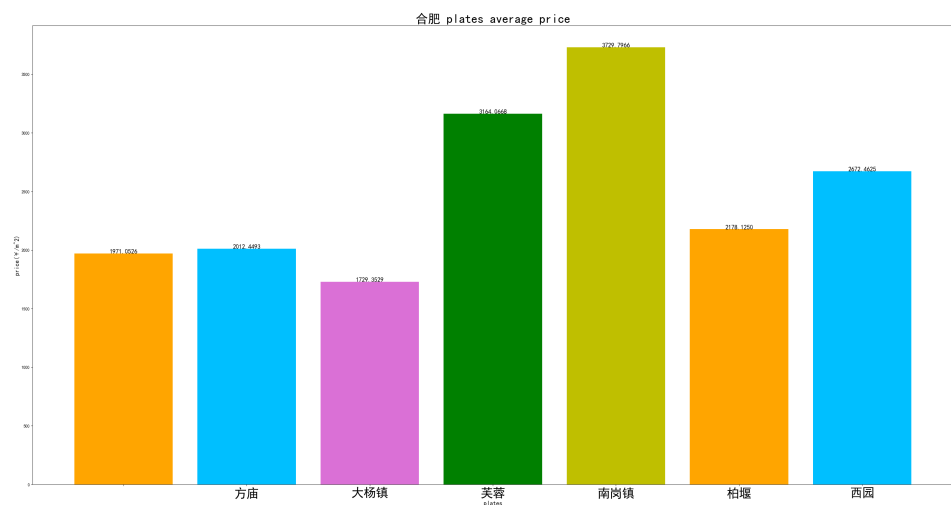




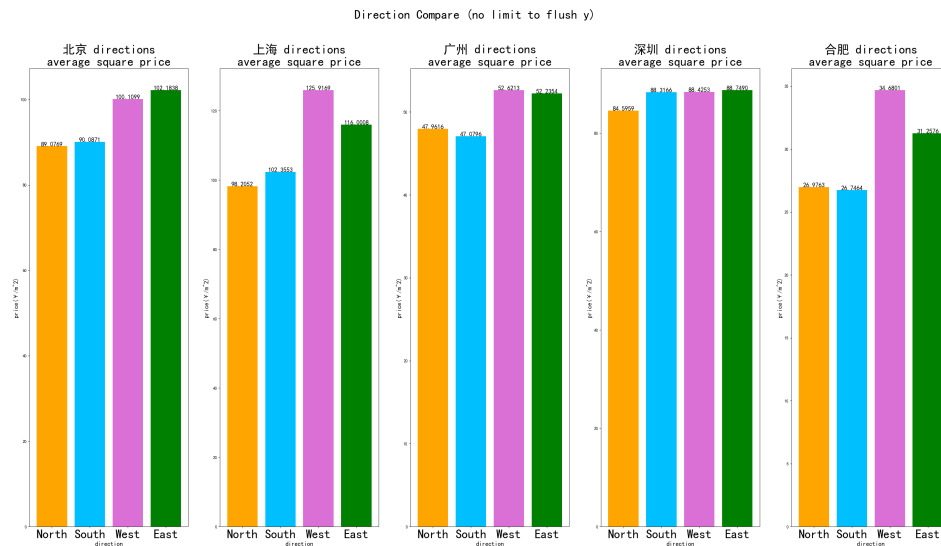
深圳：



合肥：



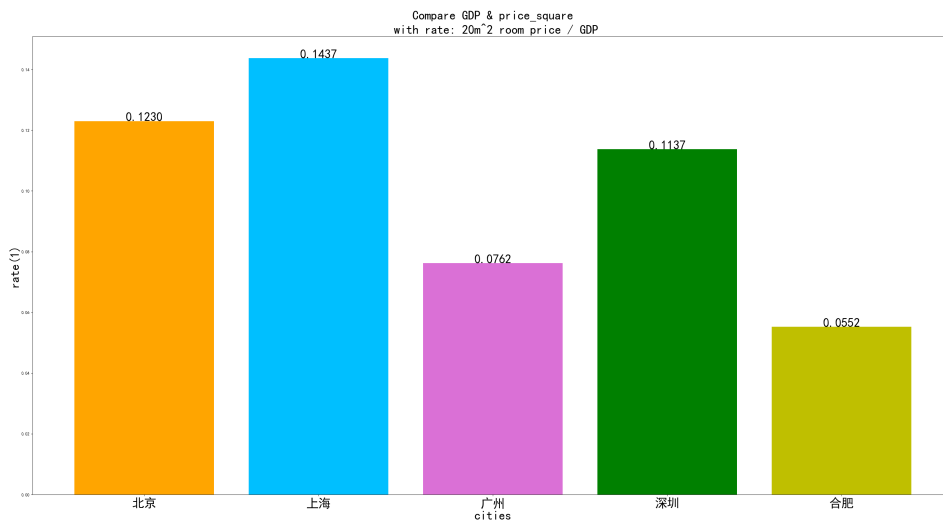
Q4：比较各个城市不同朝向的单位面积租金分布情况



1. 各个城市不一致，比如深圳居然西边价格明显贵一点。
2. 东西朝向的房屋价格普遍比南北朝向贵（一个小前提）。
3. 沿海城市更看重南北朝向。
4. 大体上还是东南贵一点。

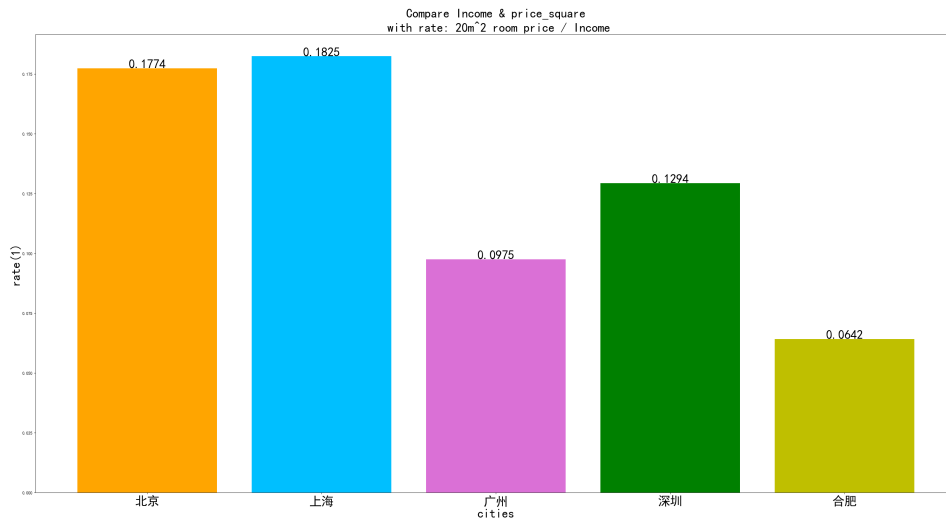
## Q5: GDP & 单位面积租金分布

ATTENTION: 我们引入变量: 20平方米一年所支付的租金占总金额数值的比例



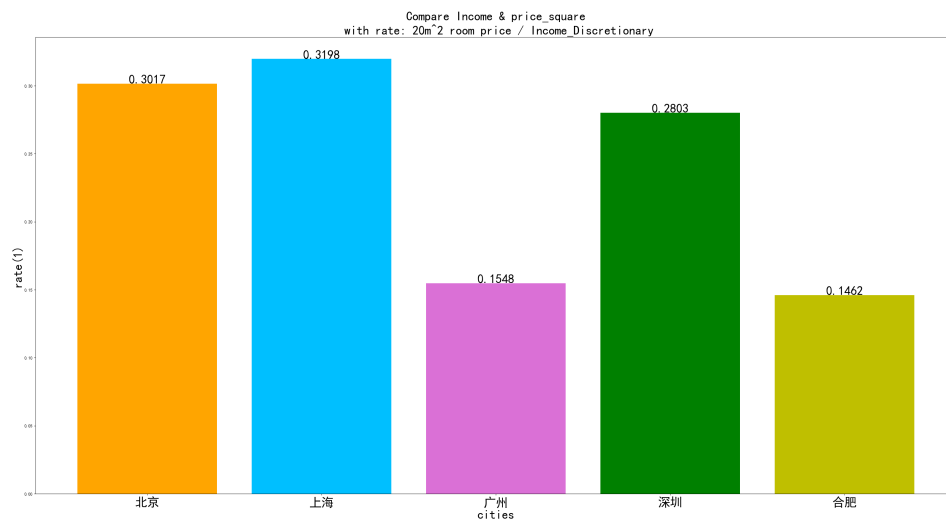
可以看到上海最贵，北京和深圳差不多，合肥最低。

## Q6: 平均工资 & 单位面积租金分布



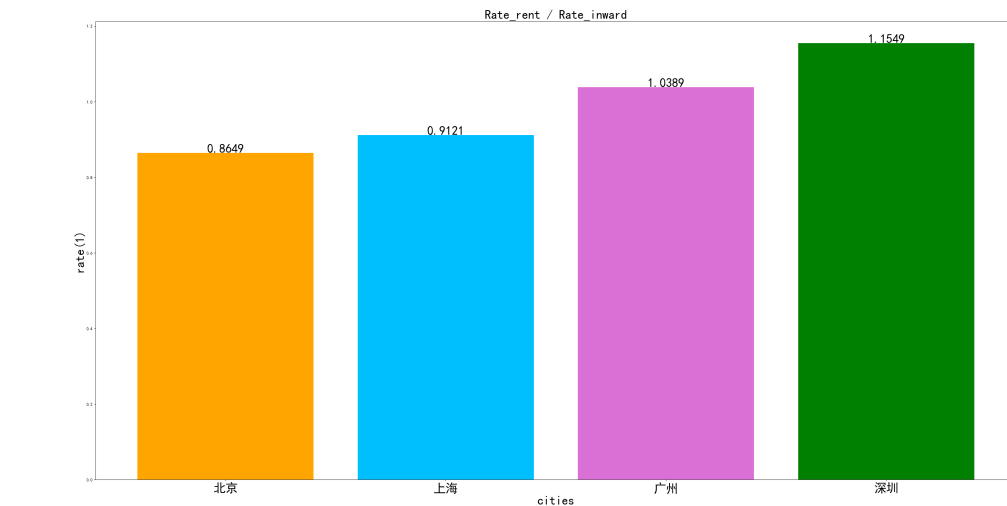
平均工资更能反映：北京上海差不多，其余三个明显低于上海北京，依旧合肥最低。

## Q7：拓展 - 可支配收入 & 单位面积租金分布



这里就看出来问题了：广州降到了合肥的水准！说明在可支配收入中房租占比广州很小，说明房租压力不是很大。

## Q8：拓展 - 租房人口比例 & 常住非户籍人口比例



这是更有意思的一组数据。我们的租房人口占非户籍人口的比重，北京上海广州深圳依次增高，而北京上海小于1。这说明了入籍的难易程度：北京住了也难入籍，深圳你甚至租房也是本地户籍。所以可以看出深圳的招揽人才在户籍方面下了大功夫，而北京上海则对户籍严把关。

## 4. 遇到的问题 & 改进

1. 爬取数据：用板块方式取出的数据更可靠（综合排序页面爬取去重去脏数据后32000剩3000，地区爬取剩16000）
2. 取出换行和空格：数据要注意规整，尤其是像下图出现的数据缺省或非法。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
154	整租 南沙-金洲	47	1	1500	1500																		
155	整租 春庭 白云-元下	62.81	2	2100	2100																		
156	整租 奥园 番禺-汉溪	48	2	3400	3400																		
157	整租 华南 番禺-华南	139	4	7000	7000																		
158	整租 嘉诚 越秀-水荫	53	1	4500	4500																		
159	整租 萝岗 黄埔-科学	96	3	3600	3600																		
160	整租 锦林 黄埔-科学	92	3	5000	5000																		
161	整租 保利 南沙-黄阁	79	3	1800	1800																		
162	整租 奥园 花都-镜湖	124	4	4000	4000																		
163	整租 石化 黄埔-文冲	38	1	2000	2000																		
164	整租 南沙 海珠-同福	55	2	2600	2600																		
165	整租 新鸿 花都-狮岭	135	4	4500	4500																		
166	整租 灵山 南沙-明珠	68	2	2800	2800																		
167	整租 富丽 番禺-大石	115	5	3300	3300																		
168	整租 岗贝 白云-机场	120	3	4500	4500																		
169	整租 凤湖 黄埔-知识	60	2	2000	2000																		
170	整租 佳信 海珠-赤岗	84	2	5500	5500																		
171	整租 湖溪 越秀-东湖	108	3	5600	5600																		
172	整租 越秀 南沙-金洲	43	1	1200	1200																		
173	整租 中海 南海-金沙	107	3	3600	3600																		
174	整租 时代 南海-金沙	77	2	2300	2300																		
175	整租 花都 花都-新区	127	2	1700	1700																		
176	整租 珠光 天河-珠江	25	1	2080	2080																		
177	整租 招商 黄埔-知识	121	4	1500	1500																		
178	整租 华南 番禺-华南	145	3	4500	4500																		
179	整租 保利 黄埔-知识	87	3	2300	2300																		
180	整租 展创 番禺-广州	32	1	1200	1200																		
181	整租 侨德 白云-同德	85	未	2200	2200																		
182	整租 龙湖 黄埔-知识	99	3	3800	3800																		
183	独栋 源公 黄埔-知识	99	3	1350	1350																		
184	独栋 源公 黄埔-知识	99	3	1900	1900																		
185	独栋 源公 黄埔-知识	99	3	2200	2200																		
186	整租 信达 天河-沙太	46	1	5000	5000																		
187	整租 东涌 番禺-市桥	93	3	1800	1800																		

3. price的处理：记录最高和最低。有 - 则最低最高各一个，没有则都赋值该值。
4. 脏数据：

文件 开始 插入 绘图 页面布局 公式 数据 审阅 视图 帮助 操作说明搜索										广州.csv - Excel										天成										
A783										独栋米家美大学城店 21特一楼一房一开间																				
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W							
764	独栋临海	南沙·金洲	83	2	2499	2499	2499	30.10843	30.10843	30.10843																				
765	整租	人民荔湾·龙津	28	2	2680	2680	2680	95.71429	95.71429	95.71429																				
766	整租	朝天越·越秀·西门	49.07	2	3000	3000	3000	61.13715	61.13715	61.13715																				
767	整租	华南碧桂园·华康	317.96	7	30000	30000	30000	94.35149	94.35149	94.35149																				
768	整租	时代南沙·沙湾	180	4	6000	6000	6000	33.33333	33.33333	33.33333																				
769	整租	碧桂园·凤凰	95	3	2500	2500	2500	26.31579	26.31579	26.31579																				
770	独栋	本地·增城·凤凰	95	3	1980	1980	1980	20.84211	20.84211	20.84211																				
771	整租	芳村·荔湾·鹤顶	62	3	2500	2500	2500	40.32258	40.32258	40.32258																				
772	独栋	米家·增城·鹤顶	62	3	1180	1180	1180	19.03226	19.03226	19.03226																				
773	整租	碧桂园·南沙·进港	98	3	2400	2400	2400	24.4898	24.4898	24.4898																				
774	整租	金·增城·凤凰	70	3	3500	3500	3500	50	50	50																				
775	整租	品·增城·新塘	105	3	3000	3000	3000	28.57143	28.57143	28.57143																				
776	整租	广州·增城·沙村	93	3	2500	2500	2500	26.88172	26.88172	26.88172																				
777	整租	米家·增城·沙村	93	3	899	899	899	6.66667	6.66667	6.66667																				
778	整租	汇·增城·白江	96	3	3200	3200	3200	33.33333	33.33333	33.33333																				
779	整租	亚运·番禺·亚运	141	3	4600	4600	4600	32.62411	32.62411	32.62411																				
780	整租	米家·番禺·亚运	141	3	880	1080	980	6.241135	7.659574	6.950355																				
781	独栋	米家·天河·天河	555	1	44000	44000	44000	79.27928	79.27928	79.27928																				
782	独栋	米家·天河·天河	555	1	1580	1580	1580	2.846847	2.846847	2.846847																				
783	独栋	米家·天河·天河	555	1	799	799	799	1.43964	1.43964	1.43964																				
784	整租	新塘·增城·沙村	99.02	3	3000	3000	3000	30.29691	30.29691	30.29691																				
785	独栋	米家·增城·沙村	99.02	3	2280	2280	2280	23.02565	23.02565	23.02565																				
786	合租	吗·增城·沙村	99.02	3	1560	1560	1560	15.75439	15.75439	15.75439																				
787	整租	东风·越秀·东风	58	2	3600	3600	3600	62.06897	62.06897	62.06897																				
788	合租	天河·越秀·天河	58	2	1560	1560	1560	26.89655	26.89655	26.89655																				
789	合租	新·越秀·天河	58	2	1160	1160	1160	20	20	20																				
790	合租	春江·越秀·天河	58	2	1590	1590	1590	27.41379	27.41379	27.41379																				
791	整租	汇·金·天河·金·金	42	1	5300	5300	5300	126.1905	126.1905	126.1905																				
792	整租	越秀·南沙·进港	120	4	3000	3000	3000	25	25	25																				
793	合租	富力·南沙·进港	120	4	1660	1660	1660	33.83333	33.83333	33.83333																				
794	整租	西兴·番禺·西兴	134	5	4500	4500	4500	33.58209	33.58209	33.58209																				
795	整租	科·增城·增城·朱村	118	4	3000	3000	3000	25.42373	25.42373	25.42373																				
796	整租	现代·增城·新塘	125	3	3000	3000	3000	24	24	24																				
797	合租	如意·增城·新塘	125	3	1460	1460	1460	11.68	11.68	11.68																				
798	合租	四季·增城·新塘	125	3	2190	2190	2190	17.52	17.52	17.52																				

广州
 

就绪
 辅助功能: 不可用

平均值: 5877.320721    计数: 30    求和: 141055.6973

100%

