Python 程序设计期末大作业

租房数据分析实验报告

吴镇均

班级：2020211306

学号：2020211448

2022 年 12 月 31 日

目录

1 实验目的

可以分为以下几个实验目的：

- 抓取链家官网北上广深 4 个一线城市及常德市的数据。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）。
- 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。
- 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。
- 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。
- 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。
- 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。并分析相对而言在哪个城市租房的性价比最高。
- 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。并分析相对而言在哪个城市租房的负担最重。
- 爬取在网站中标有“业主推荐”标签的租房信息，比较这部分的租房价格特征是否会与总体租房价格特征不同。**(自主设计题目)**

2 实验过程

2.1 信息爬取

2.1.1 网页分析

通过对网页的分析，可以分析得到爬虫时的主要困难，如下：

- **网页中对于同一搜索条件，最多展示一百页数据**。而每页最多 30 条租房信息，因此每次搜索最多只能得到 3000 条租房信息。而对于超出范围的租房信息则无法得到。(通过实践，调整 page 数并不能得到更之后的信息，而是随机重复已有信息。)
- **许多租房信息不以正常形式展示，而是以广告形式展示**。广告显示的是非链家的租房信息，理论上也属于当前城市包含的租房信息，按照实验目的也应当进行获取。同时广告形式和正常形式略有区别，会导致爬虫时不能按照对待正常形式的方式来获取相应数据。
- **由于需要爬取的信息过多，在爬取过程中很容易出现 request timeout 的情况，导致爬虫中途停止**。

经过实践和研究，**得到如下解决方案：**

- **通过选择互斥的搜索条件，来尽可能多地获取同一城市的全部租房信息**。在本实验中，采取了选择租房方式、户型、面积区间和租金区间这几个筛选条件来互斥筛选租房信息。由于这些筛选方式组合起来就是全集，所以只要保证每个筛选组合中的租房信息不超过 3000 条即可。

- **通过分析广告出现的形式，来分类处理对应信息**。经过分析，广告根据在 des 栏的信息数不同，可以分为以下几种：

  – 如果信息数为 8，那么索引为 4、5、6 的信息分别为面积信息、朝向信息、房型信息。

  – 如果信息数为 5，那么索引为 2、3、4 的信息分别为面积信息、朝向信息、房型信息。

  – 如果信息数为 3，那么索引为 0、1、2 的信息分别为面积信息、朝向信息、房型信息。

  – 如果信息数为 9，那么索引为 5、6、7 的信息分别为面积信息、朝向信息、房型信息。

  – 同时，如果在 des 栏的 a 信息栏的信息数少于等于 2，说明该广告中不包含地址信息，则不记录对应信息。

  同时，经过分析，在租房信息数不足 30 的页中，会出现推荐租房信息，规律如下：

  – 若本页没有租房信息，则全部为推荐租房信息，固定为 8 个。

  – 若本页有租房信息，但不足 30 个，则最后 8 个固定为推荐租房信息。

- **通过增加 timeout 时间与使用 ua 池来降低 request timeout 的概率**。

2.1.2 爬虫核心代码

为了节省篇幅，这里省略了 import 等非赘余代码部分。

```
# au池子 降低 request timeout 的概率
def get_ua():
    import random
    user_agents = [
        'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
            Chrome/39.0.2171.95 Safari/537.36 OPR/26.0.1656.60',
        #  ......后面还有很多，为了节省篇幅这里将其省略
    ]
    user_agent = random.choice(user_agents) # 随机抽取对象
    return user_agent

#  文件创建
f1 = open('北京租房数据.csv', mode='a+', encoding='utf-8', newline='')
f2 = open('上海租房数据.csv', mode='a+', encoding='utf-8', newline='')
f3 = open('广州租房数据.csv', mode='a+', encoding='utf-8', newline='')
f4 = open('深圳租房数据.csv', mode='a+', encoding='utf-8', newline='')
f5 = open('常德租房数据.csv', mode='a+', encoding='utf-8', newline='')

csv_writer1 = csv.DictWriter(
    f1, fieldnames=['名称', '区', '板块', '具体地址', '面积（㎡）', '朝向', '房型',
        '租价（元/月）'])
```

```
csv_writer2 = csv.DictWriter(
    f2, fieldnames=['名称', '区', '板块', '具体地址', '面积（㎡）', '朝向', '房型',
        '租价（元/月）'])
csv_writer3 = csv.DictWriter(
    f3, fieldnames=['名称', '区', '板块', '具体地址', '面积（㎡）', '朝向', '房型',
        '租价（元/月）'])
csv_writer4 = csv.DictWriter(
    f4, fieldnames=['名称', '区', '板块', '具体地址', '面积（㎡）', '朝向', '房型',
        '租价（元/月）'])
csv_writer5 = csv.DictWriter(
    f5, fieldnames=['名称', '区', '板块', '具体地址', '面积（㎡）', '朝向', '房型',
        '租价（元/月）'])

csv_writer = [csv_writer1, csv_writer2, csv_writer3, csv_writer4, csv_writer5]
for i in csv_writer:
    i.writeheader()

# 设置筛选信息
city = ['bj', 'sh', 'gz', 'sz', 'changde']
# 整租 合租
manners = ['rt200600000001', 'rt200600000002']
# 租金 由于网页默认提供的租金区间为左闭右闭，因此手动设置
rentPrices = [
    'brp0erp1250', 'brp1251erp1500', 'brp1501erp1750', 'brp1751erp2000',
    'brp2001erp2500', 'brp2501erp3000', 'brp3001erp4000', 'brp4001erp5000',
    'brp5001erp6500', 'brp6501erp8000', 'brp8001erp20000', 'brp20001'
]
# 户型
roomTypes = ['l0', 'l1', 'l2', 'l3']
# 面积
areaTypes = ['ra0', 'ra1', 'ra2', 'ra3', 'ra4', 'ra5']

# 拼接顺序： 页数 方式 朝向 户型 面积 租金
# 设置重复请求次数
s = requests.session()
s.mount('http://', HTTPAdapter(max_retries=3))
s.mount('https://', HTTPAdapter(max_retries=3))
#  组合筛选条件进行互斥筛选
for i in range(0, 5): # 城市遍历
    for manner in range(0, 2): # 方式遍历
        for roomType in range(0, 4): # 户型遍历
            for areaType in range(0, 6): # 面积区间遍历
                for rentPrice in range(0, 12): # 租金区间遍历
                    for page in range(1, 101):
                        if page % 10 == 0: # 每 10 页休眠 1 秒
                            time.sleep(1)
                        url = 'https://' + city[i] + '.lianjia.com/zufang/' + 'pg' +
                            str(page) + manners[manner] + roomTypes[roomType] +
                            areaTypes[areaType] + rentPrices[rentPrice] + '/'
                        headers = {'User-Agent': get_ua()}
```

```
                response = s.get(url=url, headers=headers, timeout=10)
                selector = parsel.Selector(response.text)
                lis = selector.css('.content__list .content__list--item')
                print(url)
                if len(lis) == 8: # 如果只有八套则一定均为推荐房源
                    break
                cnt = 0
                for li in lis:
                    cnt = cnt + 1
                    dit = {}
                    title = li.css('.twoline::text').get().strip('"').strip()
                    if title == '':
                        title = li.css('.content__list--item--title
                            a::text').get().strip('"').strip()
                    dit['名称'] = title
                    if len(
                           li.css('.content__list--item--des a::text').getall())
                               > 2:
                        location1 = li.css('.content__list--item--des
                            a::text').get()
                        dit['区'] = location1
                        location2 = li.css('.content__list--item--des
                            a::text').getall()[1]
                        dit['板块'] = location2
                        location3 = li.css('.content__list--item--des
                            a::text').getall()[2]
                        dit['具体地址'] = location3
                    if len(
                           li.css('.content__list--item--des::text').
                           getall()) == 8:
                        area = li.css('.content__list--item--des::text').getall()
                        [4].strip('"').strip()
                        to = li.css('.content__list--item--des::text').getall()
                        [5].strip('"').strip()
                        room = li.css('.content__list--item--des::text').getall()
                        [6].strip('"').strip()
                    elif len(li.css('.content__list--item--des::text').getall())
                         == 5:
                        area = li.css('.content__list--item--des::text').getall()
                        [2].strip('"').strip()
                        to = li.css('.content__list--item--des::text').getall()
                        [3].strip('"').strip()
                        room = li.css('.content__list--item--des::text').getall()
                        [4].strip('"').strip()
                    elif len(li.css('.content__list--item--des::text').getall())
                         == 3:
                        area = li.css('.content__list--item--des::text').getall()
                        [0].strip('"').strip()
                        to = li.css('.content__list--item--des::text').getall()
                        [1].strip('"').strip()
```

```
                    room = li.css('.content__list--item--des::text').getall()
                    [2].strip('"').strip()
                elif len(li.css('.content__list--item--des::text').getall())
                     == 9:
                    area = li.css('.content__list--item--des::text').getall()
                    [5].strip('"').strip()
                    to = li.css('.content__list--item--des::text').getall()
                    [6].strip('"').strip()
                    room = li.css('.content__list--item--des::text').getall()
                    [7].strip('"').strip()
                else:
                    print(len(li.css('.content__list--item--des::text')
                    .getall()))
                    print(title)
                if '㎡' in area:
                    dit['面积（㎡）'] = area.rstrip('㎡')
                if '租' not in to:
                    dit['朝向'] = to
                dit['房型'] = room
                price = li.css(
                    '.content__list--item-price em::text').get()
                dit['租价（元/月）'] = price
                csv_writer[i].writerow(dit)
                if (len(lis) < 30 and len(lis) - cnt <= 8): #
                     如果本页总租房数小于 30
                     且当前只剩八套，则该八套一定是推荐房源 ，进行break
                    break
            if len(lis) < 30: # 如果已经没有信息了就跳过本次筛选
                break
```

2.2 数据文件结构

这里展示北京市的数据文件，作为数据文件结构参考：

	A	B	C	D	E	F	G	H
1	名称	区	板块	具体地址	面积（㎡）	朝向	房型	租价（元/月）
2	整租·京南嘉园 1室1厅 北	房山	窦店	京南嘉园	34.9	北	1室1厅1卫	1100
3	整租·南厂北区 1室1厅 南/北	昌平	南口	南厂北区	38.7	南 北	1室1厅1卫	1450
4	整租·高教大楼 1房间 南	昌平	沙河	高教大楼	18.06	南	1房间1卫	1050
5	整租·高教大楼 1房间 南	昌平	沙河	高教大楼	18.78	南	1房间1卫	1050
6	整租·高教大楼 1房间 西	昌平	沙河	高教大楼	18	西	1房间1卫	1200
7	整租·伟业嘉园北里 1室1厅 南	房山	良乡	伟业嘉园北里	30	南	1室1厅1卫	1100
8	整租·龙湖长城源著2号院 1室0厅 西	密云	古北口镇	龙湖长城源著2号院	36.71	西	1室0厅1卫	1200
9	整租·新景家园西区 1室1厅 南	东城	崇文门	新景家园西区	22.06	南	1室1厅0卫	2000
10	整租·南环里 1室0厅 东	昌平	鼓楼大街	南环里	35.2	东	1室0厅1卫	3000
11	整租·蓝星花园 1室0厅 西	顺义	后沙峪	蓝星花园	30	西	1室0厅1卫	3000
12	整租·北街家园六区 1室0厅 西	昌平	沙河	北街家园六区	26	西	1室0厅1卫	2900
13	整租·西潞园一里 1室1厅 南	房山	良乡	西潞园一里	39.9	南	1室1厅1卫	2300
14	整租·新城东里 1室0厅 南	通州	万达	新城东里	40	南	1室0厅1卫	2800
15	整租·慧华苑 1室0厅 北	昌平	回龙观	慧华苑	17.59	北	1室0厅1卫	2500
16	整租·北街家园八区 1室0厅 东	昌平	沙河	北街家园八区	32	东	1室0厅1卫	2450
17	整租·北街家园八区 1室0厅 东	昌平	沙河	北街家园八区	35	东	1室0厅1卫	2700
18	整租·北街家园六区 1室0厅 西	昌平	沙河	北街家园六区	26	西	1室0厅1卫	2500
19	整租·高井 1室1厅 南	石景山	石景山其它	高井	35.3	南	1室1厅1卫	2500
20	整租·北街家园八区 1室0厅 东	昌平	沙河	北街家园八区	33	东	1室0厅1卫	2700
21	整租·金顶街四区 1室1厅 南	石景山	苹果园	金顶街四区	33.93	南	1室1厅0卫	2800
22	整租·清秀园北区 1室0厅 东	昌平	东关	清秀园北区	35.9	东	1室0厅1卫	2700
23	整租·安福苑 1室1厅 南	昌平	鼓楼大街	安福苑	37.58	南	1室1厅1卫	2700
24	整租·车站北里 1室1厅 南	大兴	黄村火车站	车站北里	31	南	1室1厅1卫	3000
25	整租·八角南路 1室0厅 南	石景山	八角	八角南路	25.71	南	1室0厅1卫	2200
26	整租·北街家园六区 1室0厅 西	昌平	沙河	北街家园六区	29.43	西	1室0厅1卫	2700
27	整租·北下洼子胡同 1室0厅 北	东城	地安门	北下洼子胡同	15	北	1室0厅1卫	2600
28	整租·红松园1号院 1室1厅 南	朝阳	东坝	红松园1号院	33	南	1室1厅1卫	3000
29	整租·西环里 1室1厅 南	昌平	西关环岛	西环里	33.33	南	1室1厅1卫	2700
30	整租·西潞苑小区 1室0厅 西	通州	北关	西潞苑小区	33	西	1室0厅1卫	2700

图 1: 核心文件基本结构示例

在本次爬取中，共得到如下数据量的信息：

- 北京：33960
- 上海：30073
- 广州：98395
- 深圳：56591
- 常德：3570

考虑到网站信息实时变化，与网站给出的城市租房信息总量相差均在可接受范围内，可以认为爬取到的租房信息已是对应城市的全部租房信息。

2.3 比较总体房租情况

2.3.1 核心代码

首先需要对文件进行读取 代码如下：

```
beijing = pd.read_csv('beijing.csv', sep=',', encoding="utf-8")
shanghai = pd.read_csv('shanghai.csv', sep=',', encoding="utf-8")
guangzhou = pd.read_csv('guangzhou.csv', sep=',', encoding="utf-8")
shenzhen = pd.read_csv('shenzhen.csv', sep=',', encoding="utf-8")
changde = pd.read_csv('changde.csv', sep=',', encoding="utf-8")

zufang = [beijing, shanghai, guangzhou, shenzhen, changde]
```

处理表示为区间方式的面积和租价 代码如下：

```
# 处理价格区间
for city in zufang:
    pricecopy = city['租价（元/月）']
    for i in range(0, len(city['租价（元/月）'])):
        if '-' in str(city['租价（元/月）'][i]):
            zone = str(city['租价（元/月）'][i]).split('-')
            pricecopy[i] = (eval(zone[0]) + eval(zone[1])) / 2
    city['租价（元/月）'] = pricecopy
    city['租价（元/月）'] = city['租价（元/月）'].astype(float)

# 处理面积区间
for city in zufang:
    areacopy = city['面积（㎡）'].copy()
    for i in range(0, len(city['面积（㎡）'])):
        if '-' in str(city['面积（㎡）'][i]):
            zone = str(city['面积（㎡）'][i]).split('-')
            areacopy[i] = (eval(zone[0]) + eval(zone[1])) / 2
    city['面积（㎡）'] = areacopy
    city['面积（㎡）'] = city['面积（㎡）'].astype(float)
```

然后需要获取总价的均价、最高价、最低价，中位数，单位面积租金的均价、最高价、最低价、中位数 代码如下：

```
average = []
for city in zufang:
    average.append(city['租价（元/月）'].mean())

highest = []
for city in zufang:
    highest.append(city['租价（元/月）'].max())

lowest = []
for city in zufang:
    lowest.append(city['租价（元/月）'].min())

medium = []
for city in zufang:
    medium.append(city['租价（元/月）'].median())

# 平均
unitData = []
for city in zufang:
    pricecopy = city['租价（元/月）'].copy()
    for i in range(0, len(pricecopy)):
        pricecopy[i] = pricecopy[i] / city['面积（㎡）'][i]
    unitData.append(pricecopy)

unitAverage = []
for city in unitData:
    unitAverage.append(city.mean())

unitHigh = []
for city in unitData:
    unitHigh.append(city.max())

unitLow = []
for city in unitData:
    unitLow.append(city.min())

unitmeidum = []
for city in unitData:
    unitmeidum.append(city.median())
```

准备绘图，并编写绘图函数 代码如下：

```
def plot(data, labels, visualName):
    plt.rcParams['font.sans-serif'] = ['SimHei'] # 显示中文
    width = 0.15
    plt.figure(figsize=(10, 10))
```

```
    plt.ylabel('租金', fontsize=12)
    plt.title(visualName)
    plt.ticklabel_format(style='plain')
    x = np.arange(len(labels))
    plt.xticks(x, labels=labels)
    plt.bar(x - 2*width, data[0], width=width, color='darkorange')
    plt.bar(x - width, data[1], width=width, color='deepskyblue')
    plt.bar(x, data[2], width=width, color='g')
    plt.bar(x + width, data[3], width=width, color='y')
    plt.bar(x + 2*width, data[4], width=width, color='cyan')

    for cdata in range(0, 5):
        for a, b in zip(x, data[cdata]):
            plt.text(a + (cdata - 2) * width, b, "{:.2f}".format(b), ha='center',
                va='bottom', fontsize=8)

    plt.legend(['北京', '上海', '广州', '深圳', '常德'])
    plt.savefig(visualName + '.png')
    plt.show()
```

依次进行绘图 代码如下：

```
# 清空bjdata、shdata、gzdata、szdata、cddata，初始化data
bjdata = []
shdata = []
gzdata = []
szdata = []
cddata = []
data = [bjdata, shdata, gzdata, szdata, cddata]

for i in range(0, 5):
    data[i].append(average[i])
    # data[i].append(highest[i])
    data[i].append(lowest[i])
    data[i].append(medium[i])

plot(data, ['均价', '最低价', '中位数'], '总体房租情况（总价）')

# 清空bjdata、shdata、gzdata、szdata、cddata，初始化data，操作与上面代码一致，这里省略

for i in range(0, 5):
    data[i].append(highest[i])

plot(data, ['最高价'], '总体房租情况（总价最高价）')

# 清空bjdata、shdata、gzdata、szdata、cddata，初始化data，操作与上面代码一致，这里省略

for i in range(0, 5):
    data[i].append(unitAverage[i])
```

```
    # data[i].append(unitHigh[i])
    data[i].append(unitLow[i])
    data[i].append(unitmeidum[i])

plot(data, ['均价', '最低价', '中位数'], '总体房租情况（均价）')

# 清空bjdata、shdata、gzdata、szdata、cddata，初始化data，操作与上面代码一致，这里省略

for i in range(0, 5):
    data[i].append(unitHigh[i])

plot(data, ['最高价'], '总体房租情况（均价最高价）')
```

2.3.2 绘图展示及分析

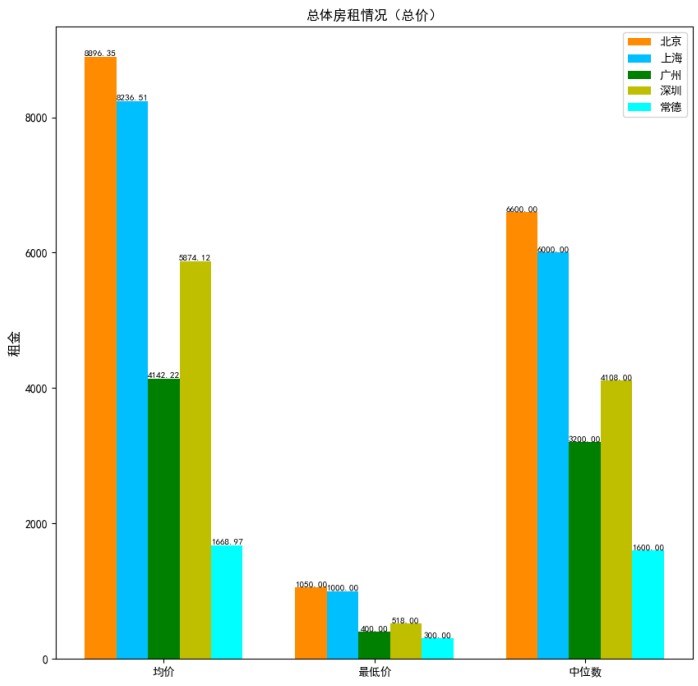

图 2: 总价的均价、最低价、中位数

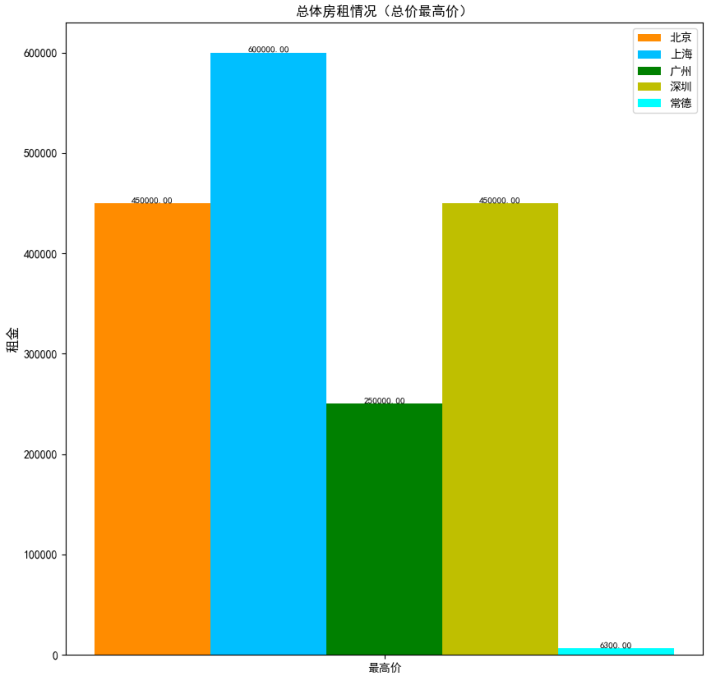

图 3: 总价的最高价

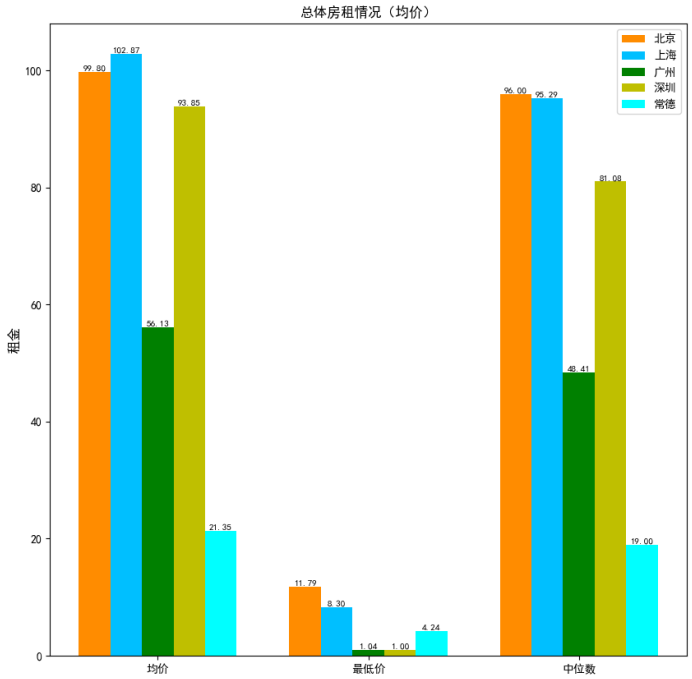

图 4: 单位面积租金的均价、最低价、中位数

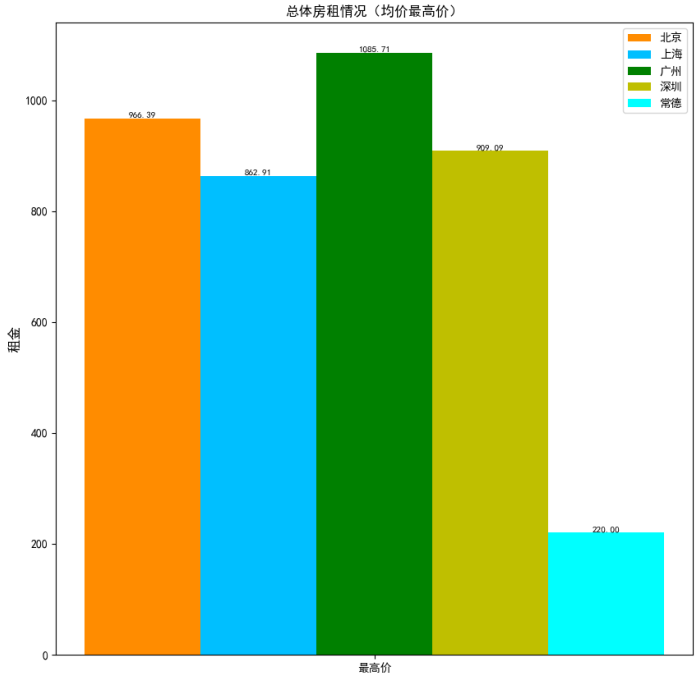

图 5: 单位面积租金的最高价

注：由于最高价和其余三种信息一起绘出会导致其余几种信息贴近 x 轴，显示效果差，因此这里将最高价单独绘出。

分析结论 首先可以看出，非一线城市常德市在大部分数据上，都和其余四个城市有较大差距。只有在最低价上会相差不多。

而在四个一线城市中，广州的租金在均价、最低价和中位数上都会明显更低一些，只有单位面积租金最高价处于第一位。深圳则是普遍高于广州，但是低于北京和上海。

北京和上海在均价、最低价和中位数上相差不大。相对而言，北京在总价的均价、最低价和中位数上更高一些，但上海的总价最高价远远领先。而在单位面积租金上，二者则相差更小，均价上上海相对更高，而在最低价和中位数上低于北京。

总的来说，非一线城市常德市的租金最低，而一线城市中，北京略高于上海，上海高于深圳，深圳高于广州。

2.4 比较一居、二居、三居情况

2.4.1 核心代码

首先需要对文件进行读取 这里代码与前面一节中的代码一致，可见代码2.3.1（点击可跳转，后续亦然）

接着是处理并分类一居、二居、三居信息。 代码如下：

```
bjData = []
shData = []
gzData = []
szData = []
cdData = []
Data = [bjData, shData, gzData, szData, cdData]
for index in range(0, 5):
    city = zufang[index]
    room = city['房型'].copy()
    price = city['租价（元/月）'].copy()
    #  1-3分别表示一居、二居、三居的价格数组
    price1 = []
    price2 = []
    price3 = []
    for i in range(0, len(city)):
        if '1室' in str(city['房型'][i]):
            if '-' in str(city['租价（元/月）'][i]):
                zone = str(city['租价（元/月）'][i]).split('-')
                price1.append((eval(zone[0]) + eval(zone[1])) / 2)
            else:
                tmp = str(city['租价（元/月）'][i])
                price1.append(eval(tmp))
        elif '2室' in str(city['房型'][i]):
            if '-' in str(city['租价（元/月）'][i]):
                zone = str(city['租价（元/月）'][i]).split('-')
                price2.append((eval(zone[0]) + eval(zone[1])) / 2)
            else:
                tmp = str(city['租价（元/月）'][i])
                price2.append(eval(tmp))
        elif '3室' in str(city['房型'][i]):
            if '-' in str(city['租价（元/月）'][i]):
                zone = str(city['租价（元/月）'][i]).split('-')
                price3.append((eval(zone[0]) + eval(zone[1])) / 2)
            else:
                tmp = str(city['租价（元/月）'][i])
                price3.append(eval(tmp))
    #  将对应信息加入城市居室信息中
    Data[index].append(price1)
    Data[index].append(price2)
    Data[index].append(price3)
```

依次绘图　代码如下，对应绘图函数参照2.3.1：

```
roomLabels = ['一居', '二居', '三居']
for jushi in range(0, 3):
    bjdata = []
    shdata = []
    gzdata = []
```

```
    szdata = []
    cddata = []
    data = [bjdata, shdata, gzdata, szdata, cddata]

    # 均价 最低价 中位数
    for i in range(0, 5):
        data[i].append(np.mean(Data[i][jushi]))
        data[i].append(np.max(Data[i][jushi]))
        data[i].append(np.min(Data[i][jushi]))
        data[i].append(np.median(Data[i][jushi]))
    plot(data, ['均价', '最高价', '最低价', '中位数'], roomLabels[jushi])
```

2.4.2 绘图展示及分析

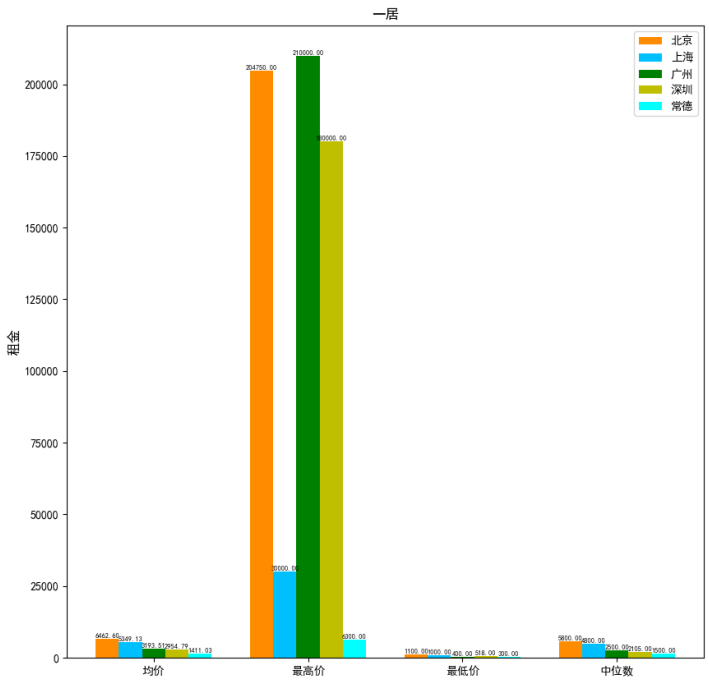

图 6: 一居的均价、最高价、最低价、中位数

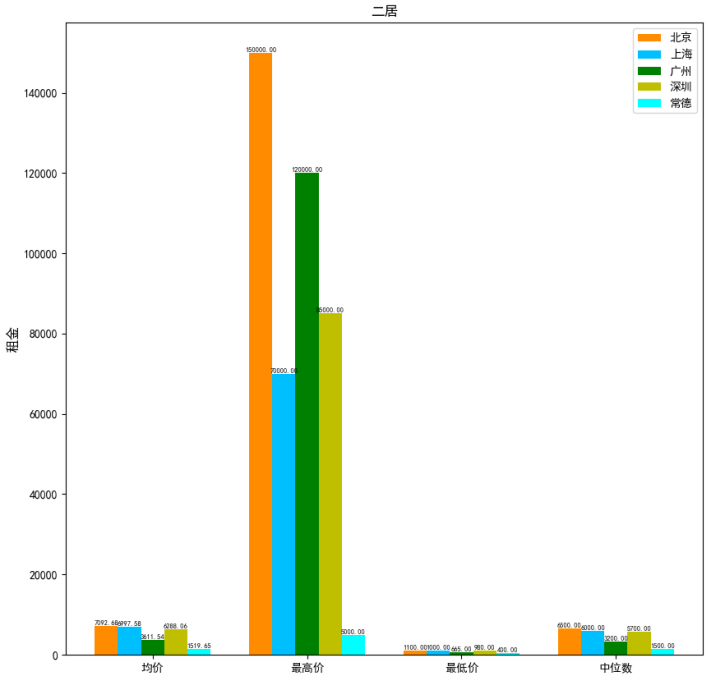

图 7: 二居的均价、最高价、最低价、中位数

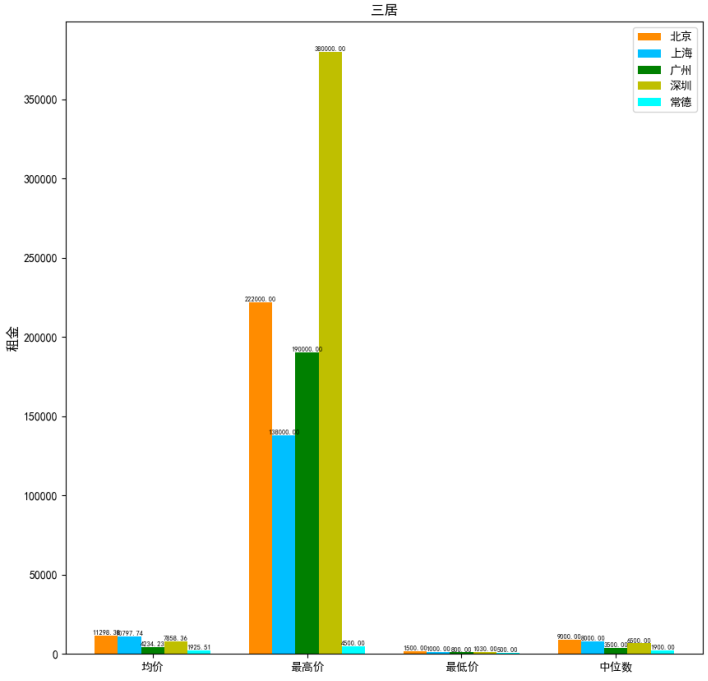

图 8: 三居的均价、最高价、最低价、中位数

分析结论 首先可以看出，非一线城市常德市的数据普遍最低。

同时在均价、最低价和中位数上，四个一线城市相差不大，普遍符合北京高于上海，上海高于深圳，深圳高于广州的规律。

而在最高价上，上海市的一居、二居、三居的最高价明显低于其余几个一线城市。其中一居广州最高价最高，二居北京最高价最高，三居深圳最高价最高。

2.5 比较板块租金均价情况

2.5.1 选用图表分析

由于每个城市中的板块较多，因此如果采用柱状图、直方图或者折线图表示，都会出现 x 轴无法写下所有板块名的情况。饼图则会出现分块过多的情况。

而如果采用分多图展示，则会超出报告应有篇幅。

因此这里考量到板块本身与地理位置联系紧密，为了更好地分析不同板块租价不同的原因，并合理展示所有板块均价，这里采用地图的方式表示。

2.5.2 核心代码

首先需要对文件进行读取 这里代码与前面一节中的代码一致，可见代码2.3.1

然后获取板块均价信息 代码如下：

```
# 删除 板块 为空的行
for i in range(0, 5):
    zufang[i] = zufang[i].dropna(axis=0, subset='板块')

# 获取每座城市的板块均价
for index in range(0, 5):
    city = zufang[index]
    curPlateName = city['板块'].unique().tolist() # 获取每座城市的板块名
    plateName.append(curPlateName)
    plateList = []
    for k in range(0, len(curPlateName)):
        plateList.append([])
    for i in city.index.tolist():
        if '-' in str(city['租价（元/月）'][i]):
            zone = str(city['租价（元/月）'][i]).split('-')
            plateList[curPlateName.index(city['板块'][i])].append((eval(zone[0]) +
                eval(zone[1])) / 2)
        else:
            tmp = str(city['租价（元/月）'][i])
            plateList[curPlateName.index(city['板块'][i])].append(eval(tmp))
    for i in plateList:
        data[index].append(np.mean(i))
```

接着使用百度 Api 获取每个板块的经纬度信息 代码如下：

```
cityName = ['北京市', '上海市', '广州市', '深圳市', '常德市']

locations = [{}, {}, {}, {}, {}]
apiurl = 'http://api.map.baidu.com/geocoding/v3/?'
# 获取板块对应的经纬度信息
for index in range(0, 5):
    for name in plateName[index]:
        params = {
            'address': name,
            'city': cityName[index],
            'output': 'json',
            'ak': 'hbVOogf05TdAXcd67WCnyhpYf0yjVpv0'
        }
        res = requests.get(apiurl, params=params)
        answer = res.json()
        if answer['status'] == 0:
            tmpList = answer['result']
            coordString = tmpList['location']
            coordList = [coordString['lng'], coordString['lat']]
        print(name + ',' + str(float(coordList[0])) + ',' + str(float(coordList[1])))
        locations[index][name] = [float(coordList[0]), float(coordList[1])]
```

注：由于申请的百度 ak 已经使用至接近限额，如需要运行这段代码，请更改百度 ak。

最后使用 pyecharts 进行绘图 代码如下：

```
for index in range(0, 5):
    g = Geo()
    g.add_schema(maptype=cityName[index].strip('市'))
    for key, value in locations[index].items():
        g.add_coordinate(key, value[0], value[1])
    data_pair = [list(z) for z in zip(plateName[index], data[index])]
    g.add('总租价均价', data_pair, symbol_size=8)
    g.set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    g.set_global_opts(visualmap_opts=opts.VisualMapOpts(max_=np.max(data[index])),
        title_opts=opts.TitleOpts(title=cityName[index] + '板块房租分布图'))
    g.render(path=cityName[index] + '板块房租分布图' + ".html")
```

2.5.3 绘图展示及分析

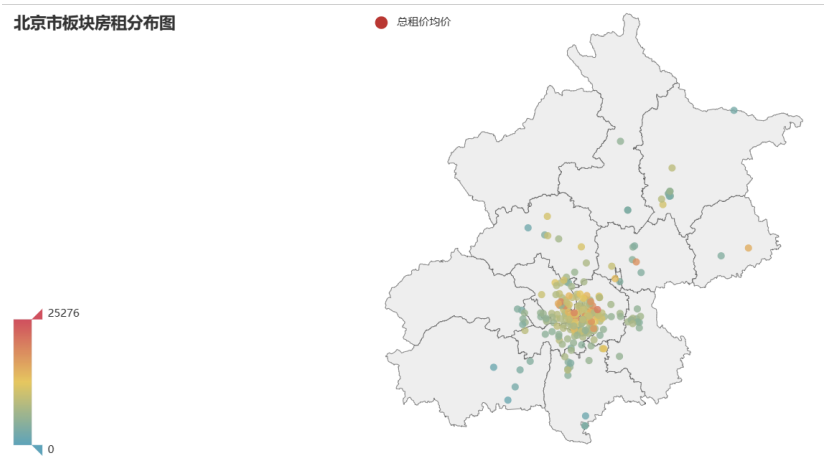

图 9: 北京板块租价均价

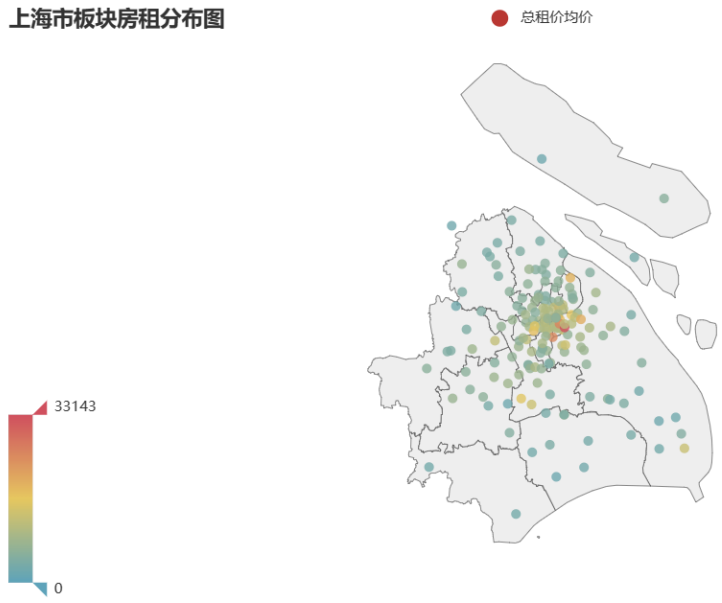

图 10: 上海板块租价均价

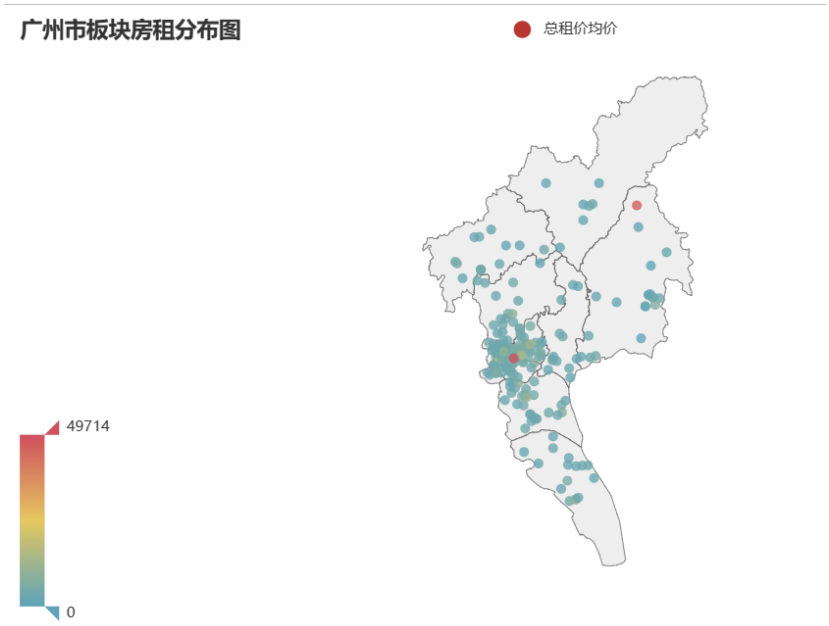

图 11: 广州板块租价均价

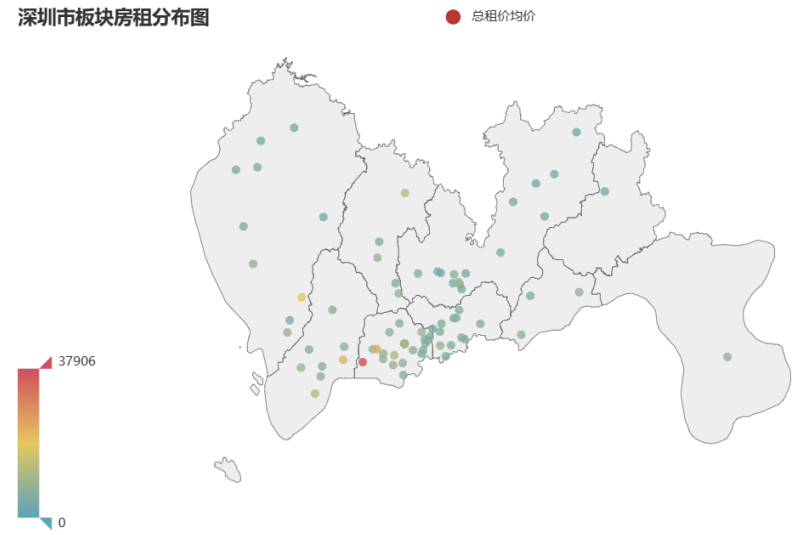

图 12: 深圳板块租价均价

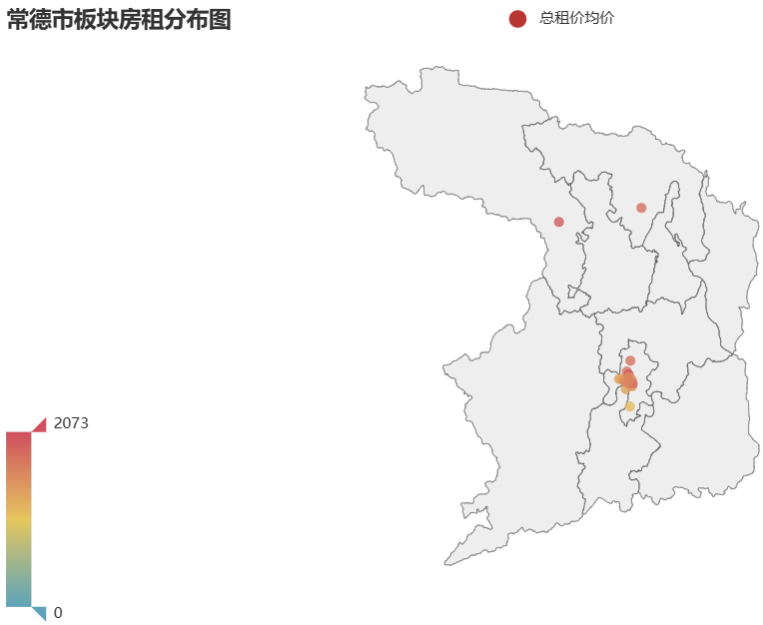

图 13: 常德板块租价均价

注：由于本报告中只能插入静态图片，因此不能完全展示绘图内容。实际代码运行会存储可互动图为 html，展示效果好

分析结论 不同地区分别分析:

- **北京市：**北京市中在东城、西城、海淀、朝阳等区的板块租价均价较高，往外扩散式地降低。
- **上海市：**上海市中黄浦区板块的租价均价相对最高，但静安区和长宁区的板块均价也相对较高。往外扩散时板块密度会降低，但租金均价降低并不明显。
- **广州市：**广州市越秀区的板块的租价均价相对最高，且板块集中在越秀区、天河区和海珠区。往外扩散时版块密度会降低，租金也会下降。
- **深圳市：**深圳市福田区和南山区的的租价均价显然更高，而板块也集中在福田区、南山区和罗湖区。往外扩散时版块密度会降低，租金也会下降。
- **常德市：**常德市的板块主要集中在武陵区，在石门县和澧县也有零散分布。其中租价均价相差不大。

2.6 比较朝向租金情况

2.6.1 核心代码

首先需要对文件进行读取 这里代码与前面一节中的代码一致，可见代码2.3.1

处理表示为区间方式的面积和租价 可见代码2.3.1

获取每个城市的朝向均价 代码如下：

```
# 获取每座城市的朝向均价
toMap = {'东': 0, '东南': 1, '南': 2, '西南': 3, '西': 4, '西北': 5, '北': 6, '东北':
    7}
for index in range(0, 5):
    city = zufang[index]
    towardList = []
    for k in range(0, 8): # 东 东南 南 西南 西 西北 北 东北 东南
        towardList.append([])
    for i in city.index.tolist():
        tos = city['朝向'][i].split() # 每个租房信息可能有多个朝向，计入包含的每个朝向中
        for to in tos:
            towardList[toMap[to]].append(city['租价（元/月）'][i] /
                city['面积（㎡）'][i])
    for i in towardList:
        if len(i) != 0:
            data[index][0].append(np.max(i))
            data[index][1].append(np.min(i))
            data[index][2].append(np.mean(i))
        else:
            data[index][0].append(0)
            data[index][1].append(0)
            data[index][2].append(0)
```

绘图 为了体现租金跟随朝向发生的变化，这里采用折线图，代码如下：

```
plt.rcParams['font.sans-serif'] = ['SimHei'] # 显示中文

cityName = ['北京', '上海', '广州', '深圳', '常德']
toName = ['东', '东南', '南', '西南', '西', '西北', '北', '东北']
information = ['最高价', '最低价', '均价']
for i in range(0, 5):
    plt.xlabel('朝向')
    plt.ylabel('平均单位面积租金（元/平米）')
    colors = ['.r-', '.b-', '.g-']
    lines = []
    for index in range(0, 3):
        p, = plt.plot(toName, data[i][index], colors[index])
        for a, b in zip(toName, data[i][index]):
            plt.text(a, b, "{:.2f}".format(b), ha='center', va='bottom', fontsize=8)
        lines.append(p)
    plt.legend(lines, information, loc='upper right')
    plt.grid(linestyle='--')
    plt.tick_params(axis='y', direction='in', color='r', grid_color='r')
    plt.title(cityName[i] + '朝向分析图')
    plt.savefig('towards_' + str(i) + '.png')
    plt.show()
```

2.6.2 绘图展示及分析

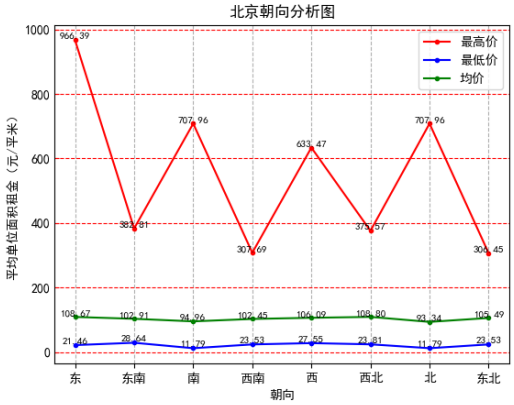

图 14: 北京朝向租价情况

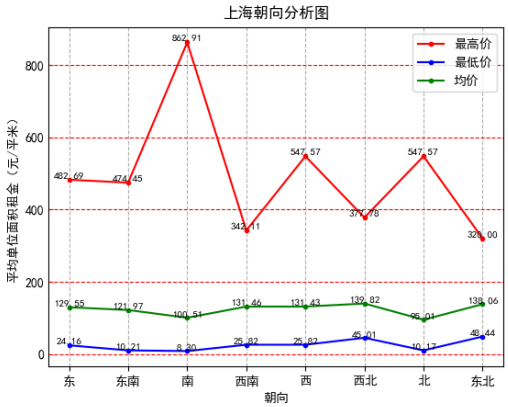

图 15: 上海朝向单位面积租金情况

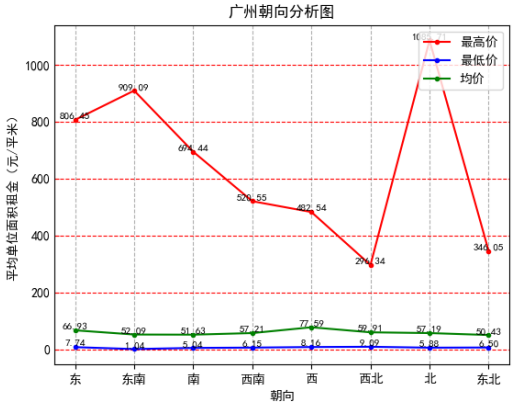

图 16: 广州朝向单位面积租金情况

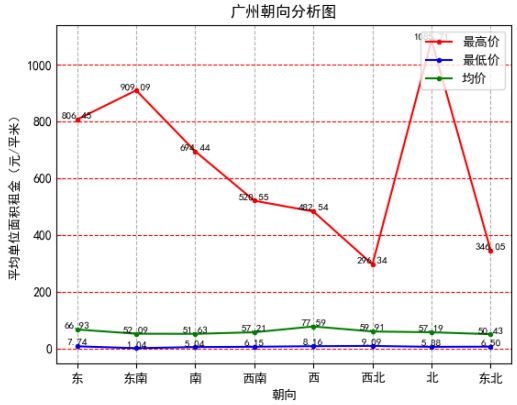

图 17: 深圳朝向单位面积租金情况

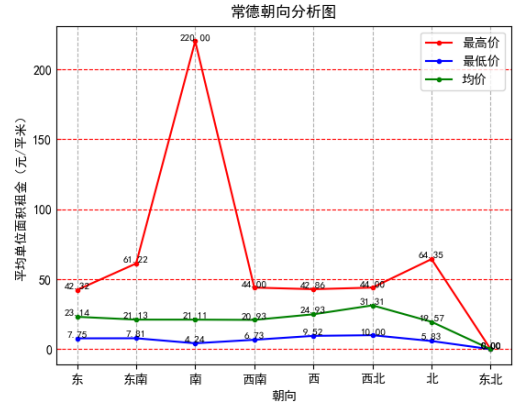

图 18: 常德朝向单位面积租金情况

分析结论　注：这里主要考虑均价（篇幅所限图片设置较小，可以放大查看。）

不同地区分别分析：

- **北京市：**西北最高，北最低。
- **上海市：**西北最高，北最低。
- **广州市：**西最高，南最低。
- **深圳市：**西最高，东南最低。
- **常德市：**西北最高，北最低。**注：常德市爬取的数据中无东北朝向的数据，因此对应数据为0。**

总体而言，各个城市的最高和最低的朝向并不完全一致，但基本都呈现出偏向东和西的方向单位面积租金均价会更高一些，而偏向南和北的单位面积租金均价会更低一些。

至于各个城市情况不一致的原因，我个人认为是各个城市的地理情况和租房宣传策略所影响的。

2.7 人均 GDP 和平均工资与单位面积租金分布的关系

由于这两个问题采用的是同一套核心代码，因此放在一起描述。

2.7.1 核心代码

首先需要对文件进行读取 这里代码与前面一节中的代码一致，可见代码2.3.1

处理表示为区间方式的面积和租价 可见代码2.3.1

获取单位面积租金均价 代码如下：

```
# 平均
unitData = []
for city in zufang:
    pricecopy = city['租价（元/月）'].copy()
    for i in range(0, len(pricecopy)):
        pricecopy[i] = pricecopy[i] / city['面积（㎡）'][i]
    unitData.append(pricecopy)

data = []
for index in unitData:
    data.append(np.mean(index))
```

绘制双 Y 轴柱状图 代码如下：

```
#  查询得到的平均GDP
aveGdp = [183937.45, 173756.71, 151162.22, 174628.38, 76796.23]
#  查询得到的平均工资
aveWage = [35549, 35487, 31421, 31889, 7270]

que = [aveGdp, aveWage]
plt.rcParams['font.sans-serif'] = ['SimHei'] # 显示中文
queName = ['人均Gdp', '人均工资']
for ques in range(0, 2):
    width = 0.3
    labels = ['北京', '上海', '广州', '深圳', '常德']
    x = np.arange(len(labels))
    # 创建图层
    fig, ax1 = plt.subplots(figsize=(16, 16))
    # 绘制柱形图1
    b1 = ax1.bar(x, data, width=width, label='平均单位面积租金（元/平米）', color='g',
        tick_label=labels)
    # 绘制柱形图2---双Y轴
    ax2 = ax1.twinx()
    b2 = ax2.bar(x + width, que[ques], width=width, label=queName[ques], color='y')
    # 坐标轴标签设置
    ax1.set_title('总体房租情况（均价）与' + queName[ques] + '关系展示', fontsize=14)
```

```
    ax1.set_xlabel('城市', fontsize=12)
    ax1.set_ylabel('平均单位面积租金（元/平米）', fontsize=12)
    ax2.set_ylabel(queName[ques], fontsize=12)
    # x轴标签旋转
    ax1.set_xticklabels(ax1.get_xticklabels(), rotation=25)

    for a, b in zip(x, data):
        ax1.text(a, b, "{:.2f}".format(b), ha='center', va='bottom', fontsize=8)
    for a, b in zip(x, que[ques]):
        ax2.text(a + width, b, "{:.2f}".format(b), ha='center', va='bottom',
            fontsize=8)

    plt.legend(handles=[b1, b2])
    plt.savefig('ques_' + str(ques) + '.png')
    plt.show()
```

2.7.2 绘图展示及分析

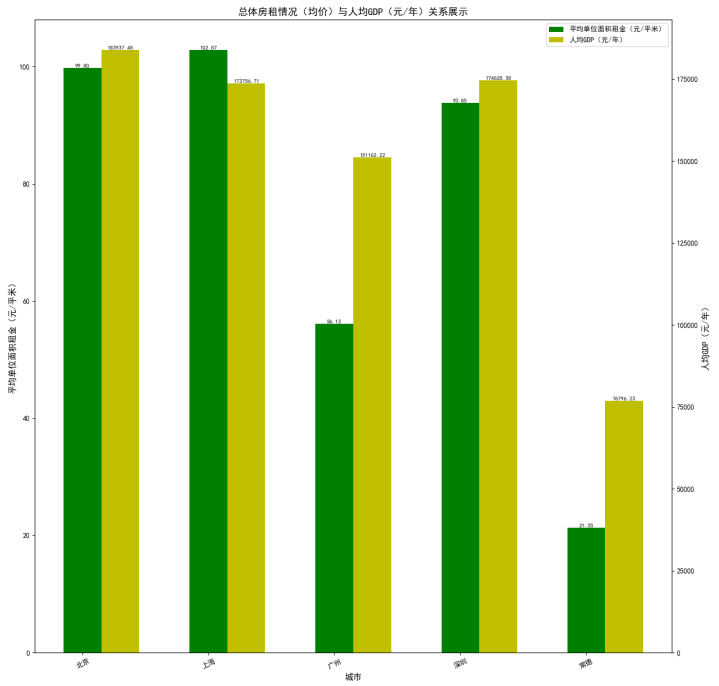

图 19: 人均 GDP 与单位面积租金均价

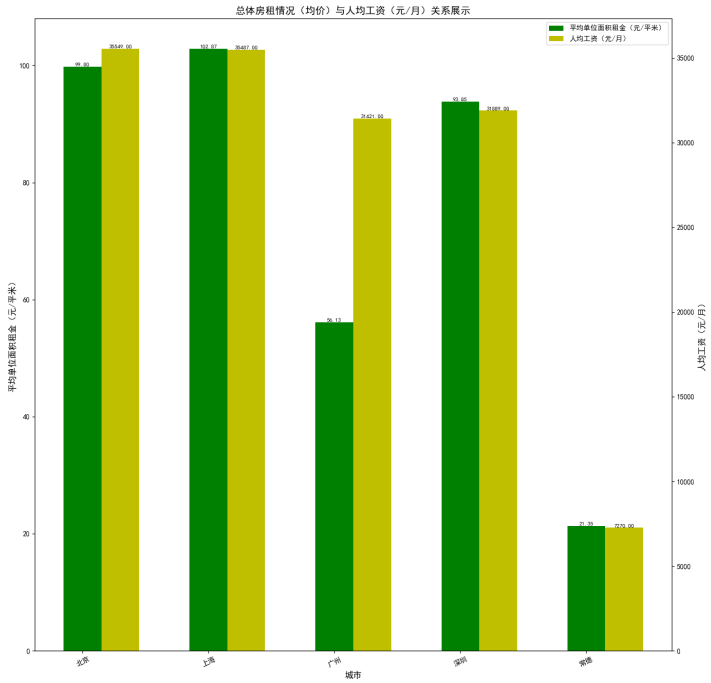

图 20: 平均工资与单位面积租金均价

分析结论

- **性价比：**从图中可以看出，在相对关系中，广州市和常德市的人均 GDP 显然高于单位面积租金均价，其中常德市的相对关系中，人均 GDP 几乎是单位面积租金的两倍（单论柱状长度），因此在常德市租房性价比最高。
- **负担：**从图中可以看出，在相对关系中，只有广州市的平均工资显然高于单位面积租金均价。因此在广州市租房负担最小。

2.8 分析有“业主推荐”标签的租房信息特征与总体的区别（自主设计题目）

2.8.1 信息爬取

爬虫核心代码 代码与2.1.2中的代码几乎一致，只有 url 前缀及遍历条件不同，如下：

```
# ......上面与前面代码一致
for i in range(0, 5): # 城市遍历
    for page in range(1, 101):
        if page % 10 == 0:
            time.sleep(1)
        url = 'https://' + city[i] + '.lianjia.com/zufang/' + 'orec1pg' + str(
            page) + '/'
# ......后面与前面代码一致
```

爬取得到的数据量　注：爬取得到的文件基本结构和之前展示的结构一致，因此这里不重复展示。

各城市包含“业主推荐”的租房信息量如下：

- **北京市：**2493；
- **上海市：**260；
- **广州市：**1146；
- **深圳市：**217；
- **常德市：**4。

2.8.2　数据分析及展示

首先需要对文件进行读取　可见代码2.3.1，区别在于将文件名依据情况进行了一定修改。

处理表示为区间方式的面积和租价　代码可见2.3.1。

然后需要获取总价的均价、最高价、最低价，中位数，单位面积租金的均价、最高价、最低价、中位数　代码可见2.3.1。

准备绘图，并编写绘图函数　绘图函数代码可见2.3.1。

绘图　代码如下：

```
# 清空bjdata、shdata、gzdata、szdata、cddata，初始化data，操作与之前代码一致，这里省略
# 总价：  均价 最低价 中位数
for i in range(0, 5):
    data[i].append(average[i])
    data[i].append(highest[i])
    data[i].append(lowest[i])
    data[i].append(medium[i])

plot(data, ['均价', '最高价', '最低价', '中位数'], '业主推荐房租情况（总价）')

# 平均价 均价 最低价 中位数

# 清空bjdata、shdata、gzdata、szdata、cddata，初始化data，操作与之前代码一致，这里省略
for i in range(0, 5):
    data[i].append(unitAverage[i])
    data[i].append(unitHigh[i])
    data[i].append(unitLow[i])
    data[i].append(unitmeidum[i])

```

```
20 plot(data, ['均价', '最高价', '最低价', '中位数'], '业主推荐房租情况（单位面积租价）')
```

图表展示 绘图得到信息如下：

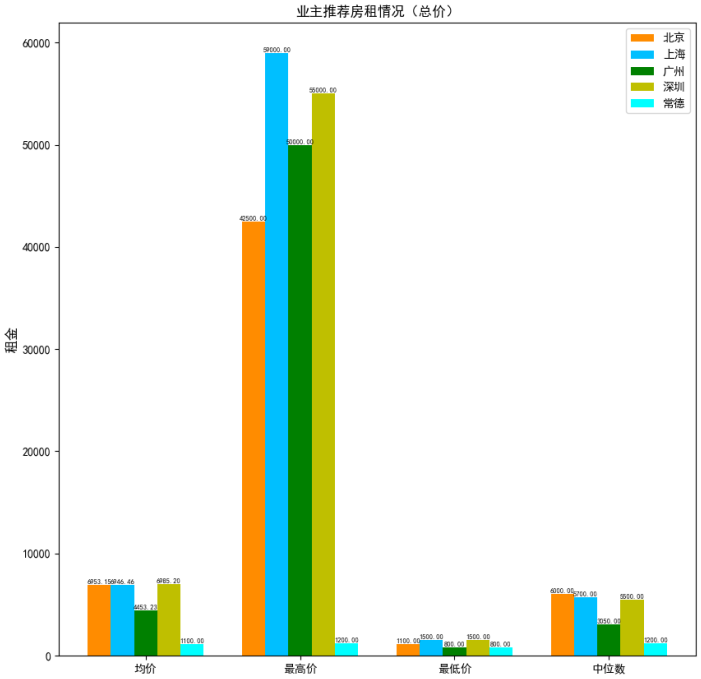

图 21: 业主推荐房租情况（总价）

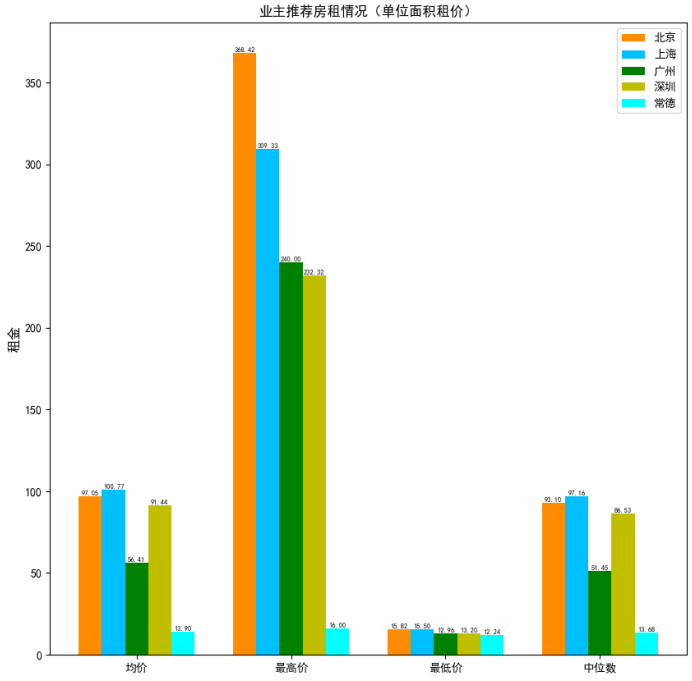

图 22: 业主推荐房租情况（单位面积租价）

分析结论 通过对比业主推荐房租展示和之前的总体展示可以发现，整体而言，包含“业主推荐”标签的房租的均价都会更低一些，但与总体相差不多。因此结论可以说“业主推荐”标签对房租影响并不大。

3 实验结论

每个实验目的的结论在对应的分析过程中已给出。具体可见：2.3.2、2.4.2、2.5.3、2.6.2、2.7.2、2.8.2。（点击即可跳转）

本次大作业实验对我的 Python 运用能力作了较为详尽的考察，通过本次实验我的 Python 代码编写能力、问题分析能力和文档查询能力得到了极大的提高。相信会成我以后学习道路上一笔宝贵的财富。