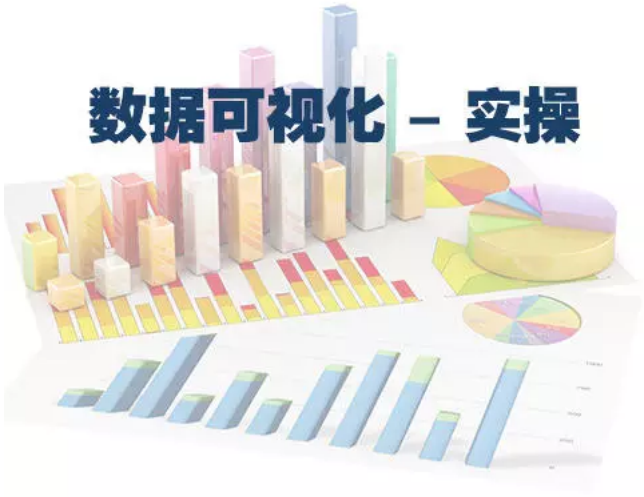
Python数据可视化-练习

陈为林_Joe (/u/d7b32cb5c108) + 关注
0.1 2017.07.21 14:53* 字数 408 阅读 2311 评论 0 喜欢 8
(/u/d7b32cb5c108)



Dr.fish 发来一份 Worldindex.csv

让我们一起来探索一下吧.

打开Jupyter notebook 新建一个python 3文档

导入模块

```
import pandas as pd              # 导入pandas模块 并简写成 pd
import matplotlib.pyplot as plt  # 导入绘图包(今天的主角)
import numpy as np               # 导入numpy模块 并简写成 np

%config InlineBackend.figure_format = 'retina' # 设置图像清晰度
```

导入文件

```
data = pd.read_csv('WorldIndex.csv')
```

观察数据

```
data.head()
```

	Country	Continent	Life_expectancy	GDP_per_capita	Population
0	Algeria	Africa	75.042537	4132.760292	39871528.0
1	Angola	Africa	52.666098	3695.793748	27859305.0
2	Benin	Africa	59.720707	783.947091	10575952.0
3	Botswana	Africa	64.487415	6532.060501	2209197.0
4	Burundi	Africa	57.107049	303.681022	10199270.0

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 177 entries, 0 to 176
Data columns (total 5 columns):
Country            177 non-null object
Continent          177 non-null object
Life_expectancy    169 non-null float64
GDP_per_capita     169 non-null float64
Population         176 non-null float64
dtypes: float64(3), object(2)
memory usage: 7.0+ KB
```

观察后,大概可知晓这是全球各国关于人口年龄 GDP 的一份数据表,
数据总体完整,但也有一小部分缺失值.为了方便练习,这里缺失值做丢弃处理.

数据清洗

```
# 删除包含缺失值的行
df = data.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 164 entries, 0 to 175
Data columns (total 5 columns):
Country            164 non-null object
Continent          164 non-null object
Life_expectancy    164 non-null float64
GDP_per_capita     164 non-null float64
Population         164 non-null float64
dtypes: float64(3), object(2)
memory usage: 7.7+ KB
```

现在数据剩有164条完整数据.
看了一下,字段名有点长.我们重新定义一下吧

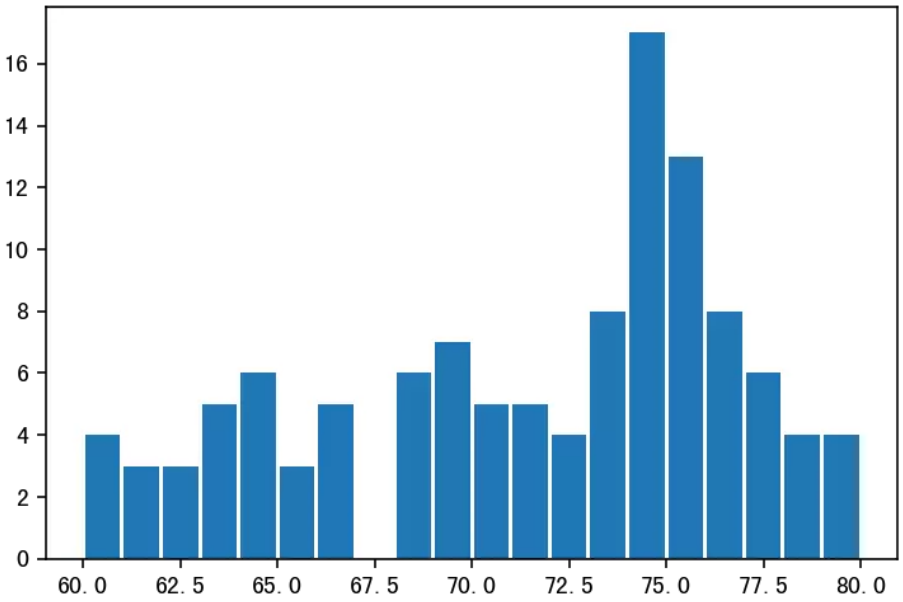
```
# 重新定义列名
df.columns = ['country', 'continent', 'life', 'gdp', 'popu']
```

数据展现

- 直方图

  绘制关于人均寿命数据的直方图

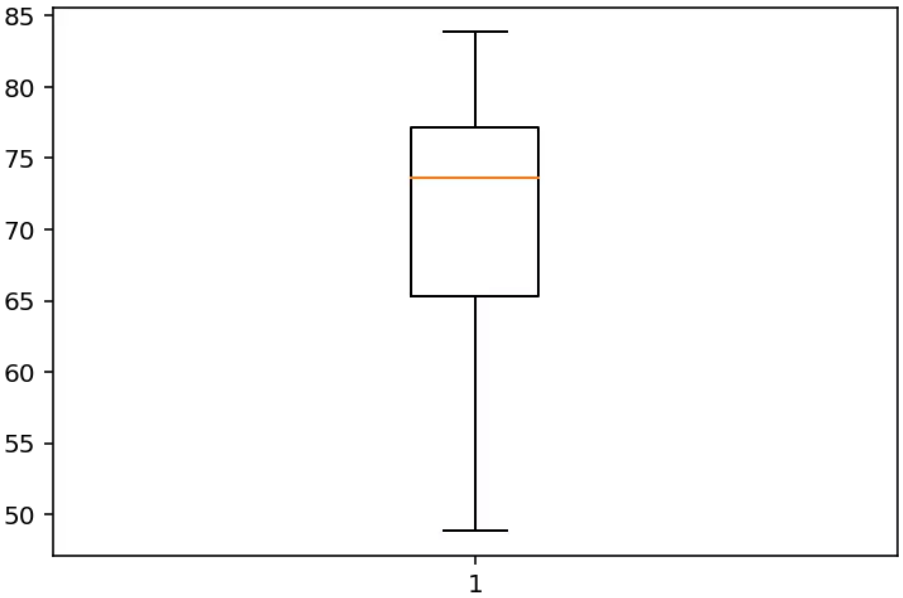
```
plt.hist(df.life, range=(60,80), bins=20, rwidth=0.9)
# 参数range 表示想要展示数据的区间.原来数据表信息不会修改到.在区间范围外的数据只是隐藏起来了
# 参数bins 表示将数据平均分成多少份来展示. 如20,则表示将X轴上数据平均分成20份.即显示20条柱状
# 参数rwidth (取值范围1~0). '0.9'表示设置单条柱状的宽度显示占比90%.那么就有10%是空白的,呈现
plt.show()
# 显示图表出来的命令, (上节课:我们是用%matplotlib inline 直接展现).这节课:是逐一显示.画完一
```



人均寿命数据的直方图

- 箱图
- 绘制人均寿命的箱图

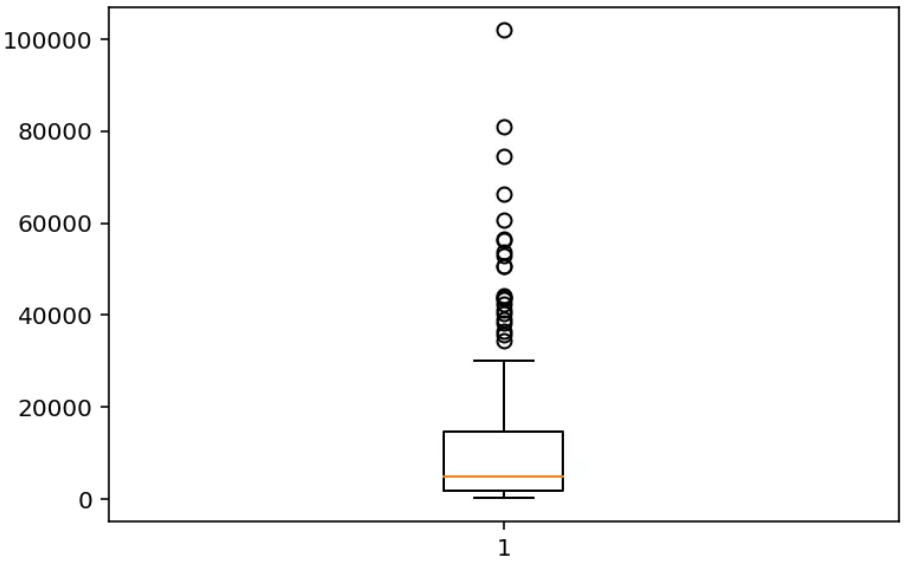
```
plt.boxplot(df.life)
plt.show()
```



人均寿命的箱图

- 绘制人均GDP数据的箱图

```
plt.boxplot(df.gdp)
plt.show()
```



人均GDP数据的箱图

- 条形图
- 绘制各大洲国家数量分布的条形图

```
# 统计每个州的国家数
conti_count = df.continent.value_counts()
conti_count
```

```
Africa           48
Europe           41
Asia             36
North America    19
South America    11
Oceania           9
Name: continent, dtype: int64
```

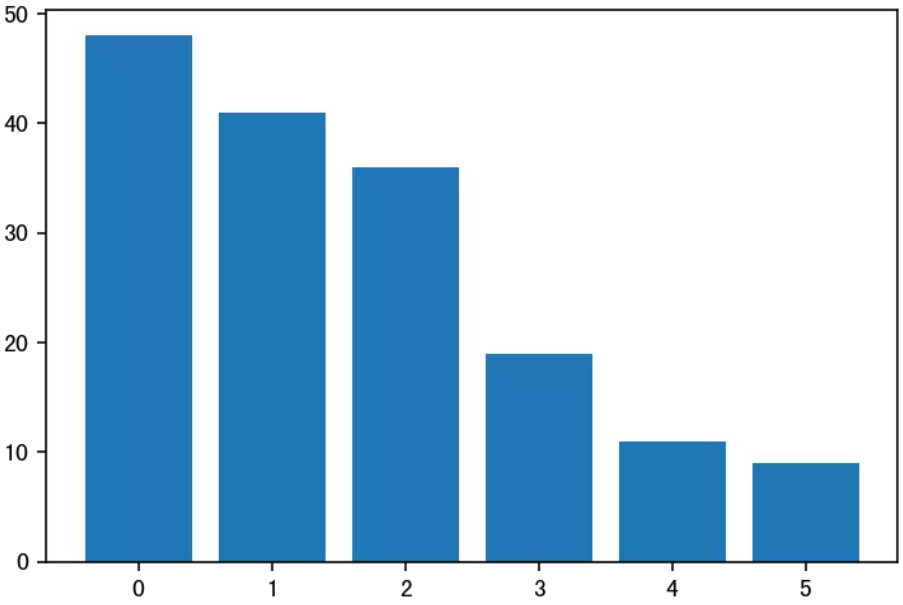
```
# 获取各大州名称
conti = list(conti_count.index)
conti
```

```
['Africa', 'Europe', 'Asia', 'North America', 'South America', 'Oceania']
```

```
有6个洲(七大洲的南极洲无人烟),我们来生成一个数字数组 6个(0~5)
# 画图时.输入的数据必须是数值形式.不能是字符串.
x = np.arange(len(conti))
x
```

```
array([0, 1, 2, 3, 4, 5])
```

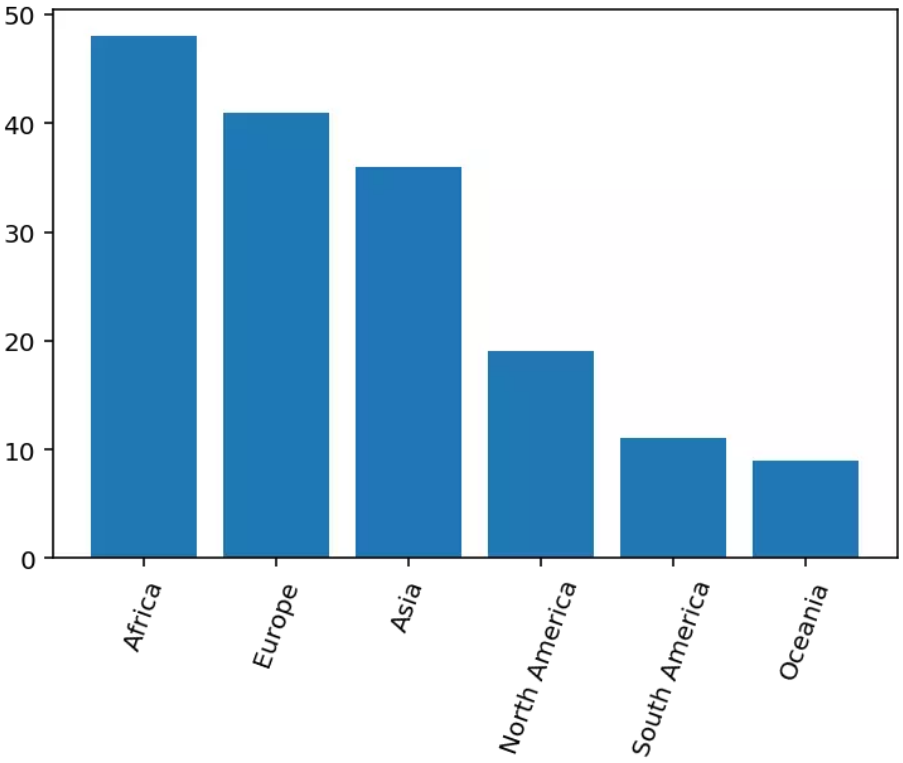
```
plt.bar(x,conti_count)
```



横坐标是数字.看不出哪个数字对应哪个洲

我们来设置调整一下横坐标

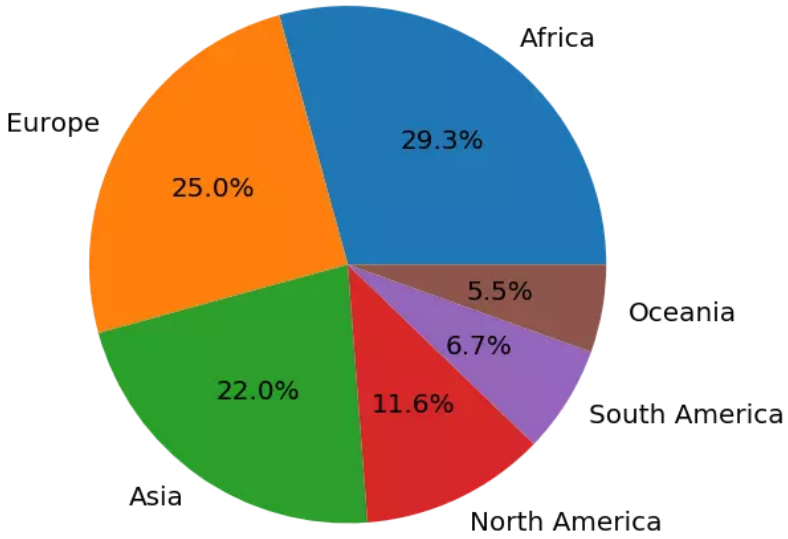
```
# 条形图
plt.bar(x, conti_count)
# 设置横坐标
plt.xticks(x, conti, rotation=70)  # rotation 旋转横坐标标签
```



这下一目了然了

- 饼图
- 绘制各大洲国家数量占比的饼图

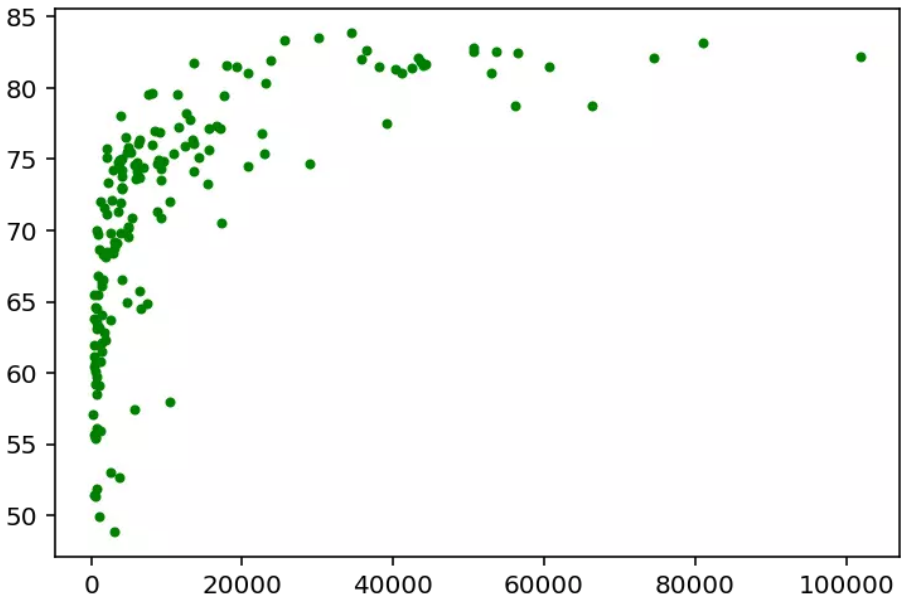
```
plt.pie(conti_count, labels=conti, autopct='%1.1f%%')  # autopct 显示占比
plt.axis('equal')  # 调整坐标轴的比例
plt.show()
```



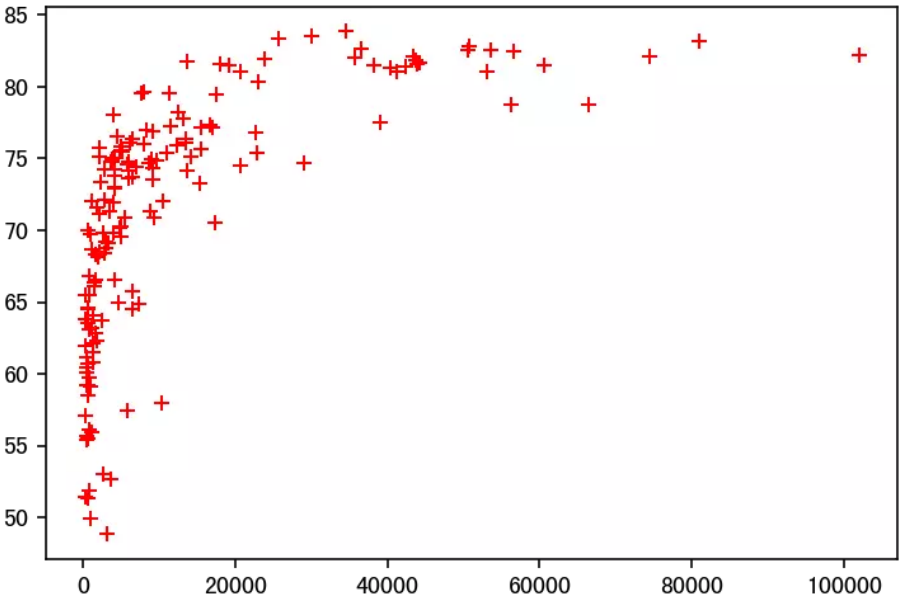
各大洲国家数量占比的饼图

- 散点图
- 绘制人均寿命和人均GDP的关系
  - 方法一

```
# 用plt.plot函数来写
plt.plot(df.gdp, df.life, 'g.')  # 'g.' 表示用绿色的点绘制
plt.show()
```

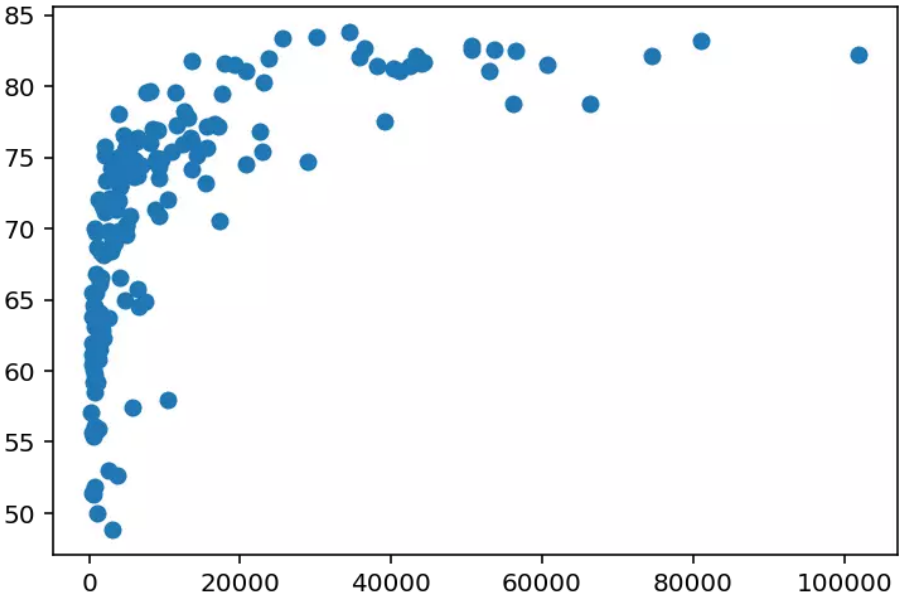


```
# 自己尝试 改变一下
plt.plot(df.gdp, df.life, 'r+')   # "r" 表示红色 "+"表示用+号来绘图
plt.show()
```



- 方法二(推荐)

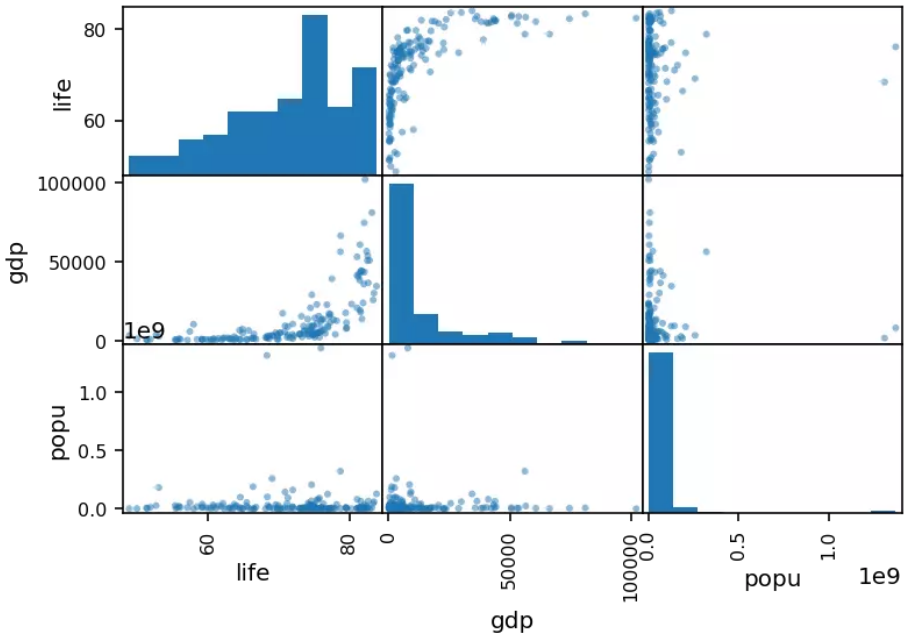
```
# 用plt.scatter函数来写 (推荐)
plt.scatter(df.gdp, df.life)
plt.show()
```



人均寿命和人均GDP的关系 散点图

- 矩阵图
- 多个变量时，为了快速观察，可绘制矩阵图

```
pd.scatter_matrix(df)
plt.show()
```

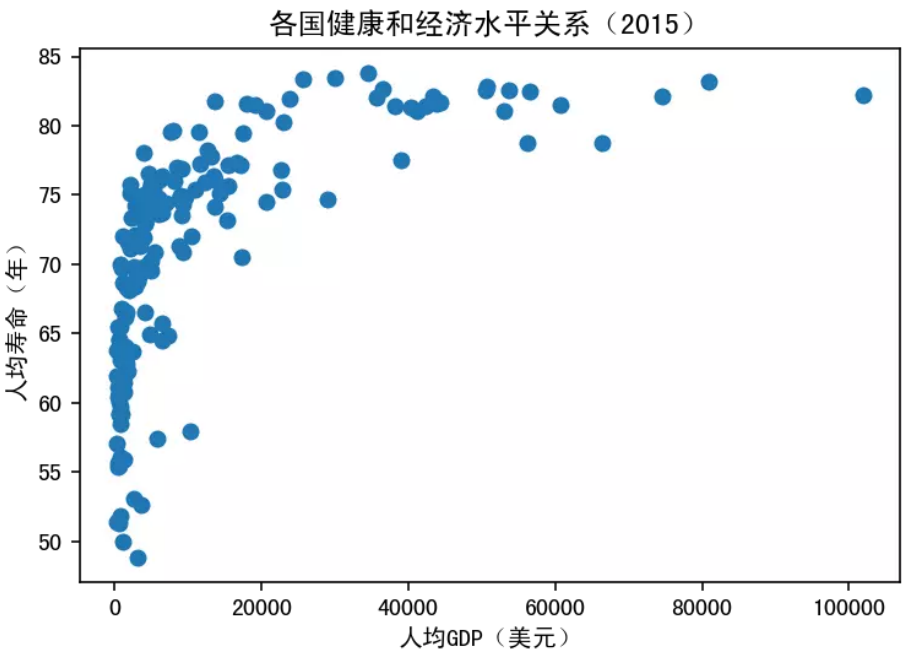


矩阵图

什么?千篇一律的图表看厌了.能不能来点新意.那么.下面的私人高端定制适合你.且看:

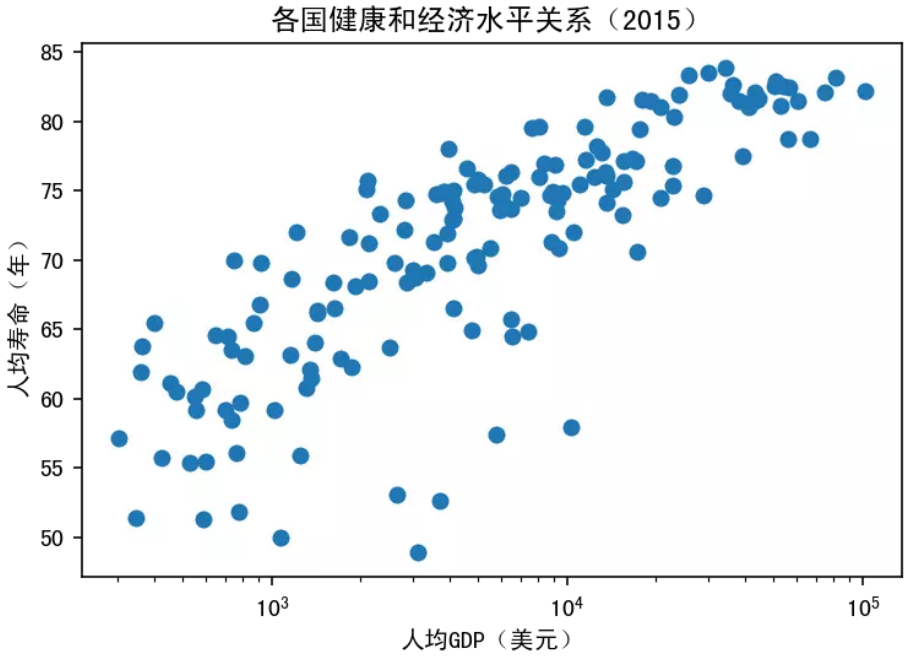
- 数据图表的定制
- 设置坐标轴名称和图像名称

```
plc.rcParams['font.sans-serif']=['SimHei']  #用来正常显示中文标签
plt.scatter(df.gdp, df.life)
plt.xlabel('人均GDP（美元）')  # x轴名称
plt.ylabel('人均寿命（年）')   # y轴名称
plt.title('各国健康和经济水平关系（2015）')  # 图标题
```



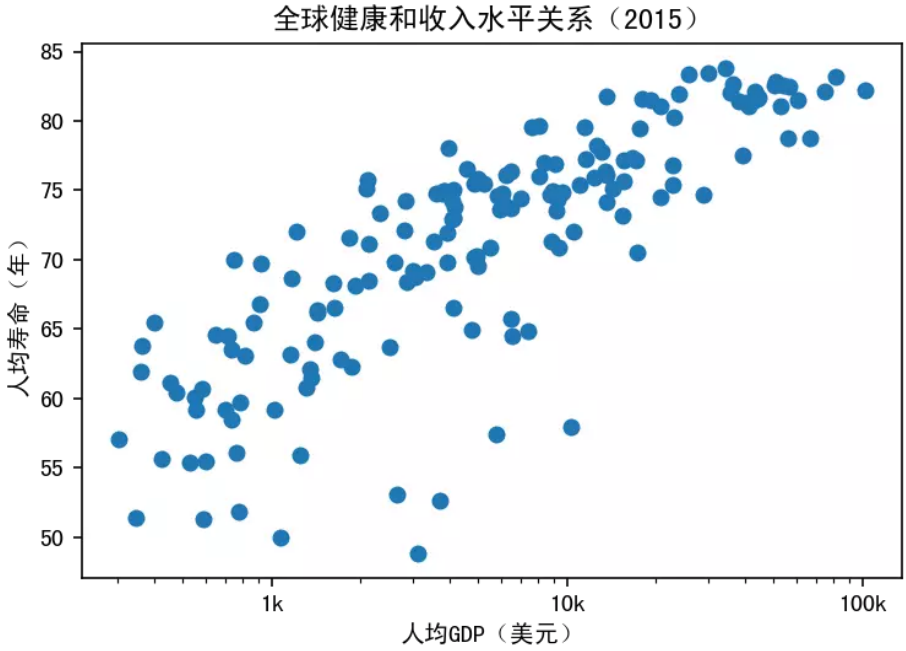
- 将横轴进行对数变换

```
plt.scatter(df.gdp, df.life)
plt.xscale('log')  # 对x轴采用对数刻度
plt.xlabel('人均GDP（美元）')
plt.ylabel('人均寿命（年）')
plt.title('各国健康和经济水平关系（2015）')
plt.show()
```

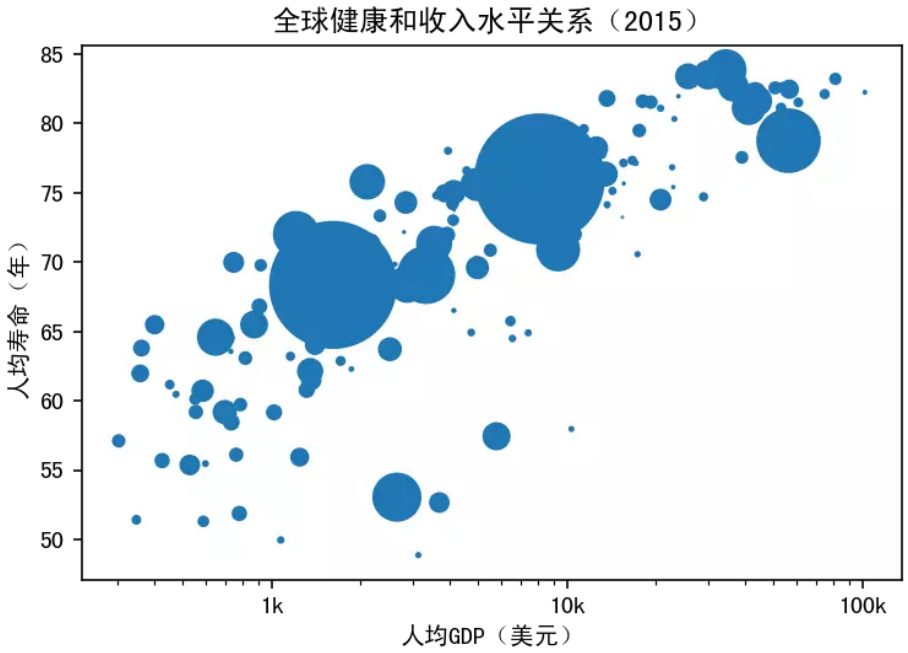
- 设置轴刻度的显示形式

```
plt.scatter(df.gdp, df.life)
plt.xscale('log')
plt.xlabel('人均GDP（美元）')
plt.ylabel('人均寿命（年）')
plt.title('全球健康和收入水平关系（2015）')
tick_val = [1000,10000,100000]
tick_lab = ['1k','10k','100k']
plt.xticks(tick_val, tick_lab)  # 重置x坐标刻度
plt.show()
```



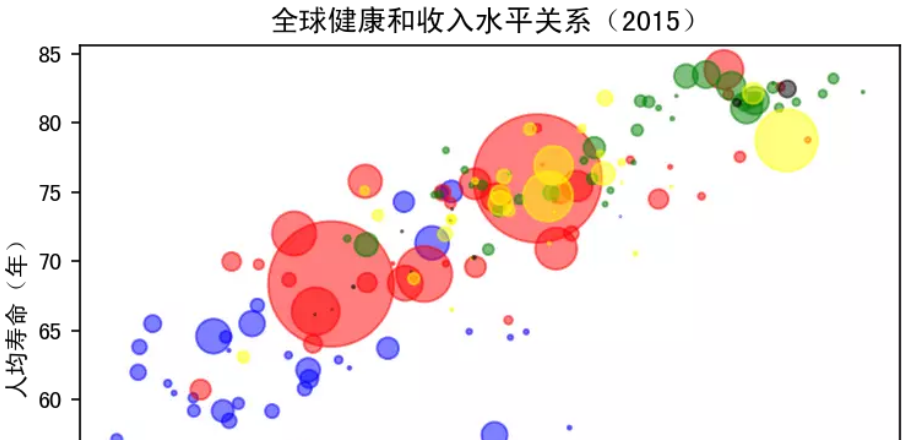
- 设置各个数据点的大小，与人口数成正比

```
size = df.popu / 1e6 * 2  # 数据点大小，正比于人口数
plt.scatter(x=df.gdp, y=df.life, s=size)  # 参数s设置点的大小
plt.xscale('log')
plt.xlabel('人均GDP（美元）')
plt.ylabel('人均寿命（年）')
plt.title('全球健康和收入水平关系（2015）')
tick_val = [1000,10000,100000]
tick_lab = ['1k','10k','100k']
plt.xticks(tick_val, tick_lab)
plt.show()
```



- 赋予不同州的国家不同的颜色

```
# 新建一个字典来存取各洲对应的颜色.
map_dict = {
    'Asia':'red',
    'Europe':'green',
    'Africa':'blue',
    'North America':'yellow',
    'South America':'yellow',
    'Oceania':'black'
}
colors = df.continent.map(map_dict)   # 将国家按所在州对于不同的颜色
size = df.popu / 1e6 * 2
plt.scatter(x=df.gdp, y=df.life, s=size, c=colors, alpha=0.5)  # 参数c设置颜色，alphai
plt.xscale('log')
plt.xlabel('人均GDP（美元）')
plt.ylabel('人均寿命（年）')
plt.title('全球健康和收入水平关系（2015）')
tick_val = [1000,10000,100000]
tick_lab = ['1k','10k','100k']
plt.xticks(tick_val, tick_lab)
plt.show()
```



- 添加文本和网格

```
map = {
    'Asia':'red',
    'Europe':'green',
    'Africa':'blue',
    'North America':'yellow',
    'South America':'yellow',
    'Oceania':'black'
}
colors = df.continent.map(map_dict)
size = df.popu / 1e6 * 2
plt.scatter(x=df.gdp, y=df.life, s=size, c=colors, alpha=0.5)
plt.xscale('log')
plt.xlabel('人均GDP（美元）')
plt.ylabel('人均寿命（年）')
plt.title('全球健康和收入水平关系（2015）')
tick_val = [1000,10000,100000]
tick_lab = ['1k','10k','100k']
plt.xticks(tick_val, tick_lab)
plt.text(1550, 73, 'India')   # 在图中添加文本
plt.text(5700, 81, 'China')
plt.grid(True)  # 添加网格
plt.show()
```