02 数据可视化-matplotlib

如果你想要用Python进行数据分析，就需要在项目初期开始进行探索性的数据分析，这样方便你对数据有一定的了解。其中最直观的就是采用数据可视化技术，这样，数据不仅一目了然，而且更容易被解读。同样在数据分析得到结果之后，我们还需要用到可视化技术，把最终的结果呈现出来。

可视化视图都有哪些?

按照数据之间的关系，我们可以把可视化视图划分为4类，它们分别是比较、联系、构成和分布。我来简单介绍下这四种关系的特点：

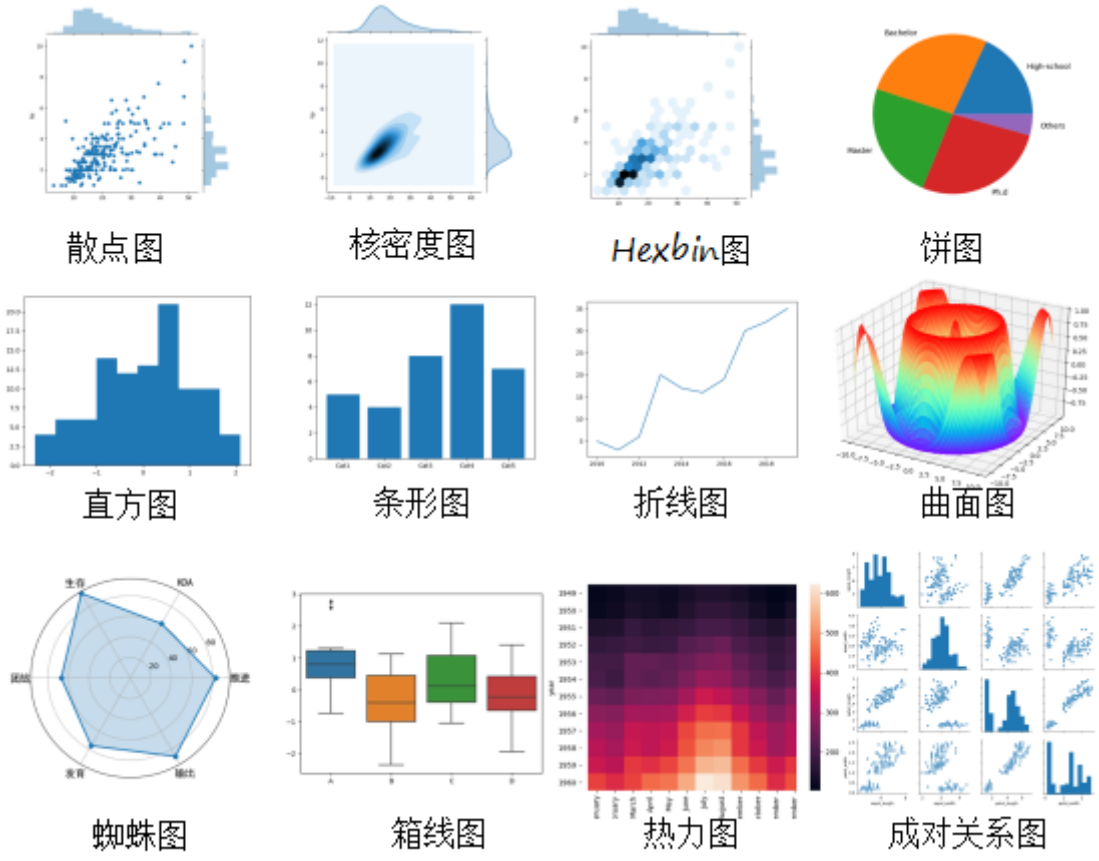
1. 比较：比较数据间各类别的关系，或者是它们随着时间的变化趋势，比如折线图；
2. 联系：查看两个或两个以上变量之间的关系，比如散点图；
3. 构成：每个部分占整体的百分比，或者是随着时间的百分比变化，比如饼图；
4. 分布：关注单个变量，或者多个变量的分布情况，比如直方图。

同样，按照变量的个数，我们可以把可视化视图划分为单变量分析和多变量分析。

单变量分析指的是一次只关注一个变量。比如我们只关注“身高”这个变量，来看身高的取值分布，而暂时忽略其他变量。

多变量分析可以让你在一张图上可以查看两个以上变量的关系。比如“身高”和“年龄”，你可以理解是同一个人的两个参数，这样在同一张图中可以看到每个人的“身高”和“年龄”的取值，从而分析出来这两个变量之间是否存在某种联系。

可视化的视图可以说是分门别类，多种多样，今天我主要介绍常用的10种视图，这些视图包括了散点图、折线图、直方图、条形图、箱线图、饼图、热力图、蜘蛛图、二元变量分布和成对关系。



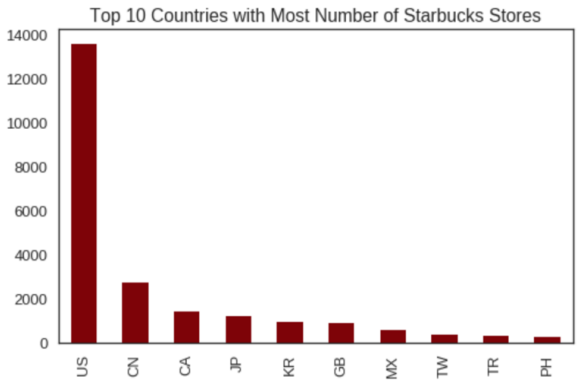
一、数据分析中常用图

可视化是在整个数据挖掘的关键辅助工具，可以清晰的理解数据，从而调整我们的分析方法。

- 能将数据进行可视化,更直观的呈现
- 使数据更加客观、更具说服力

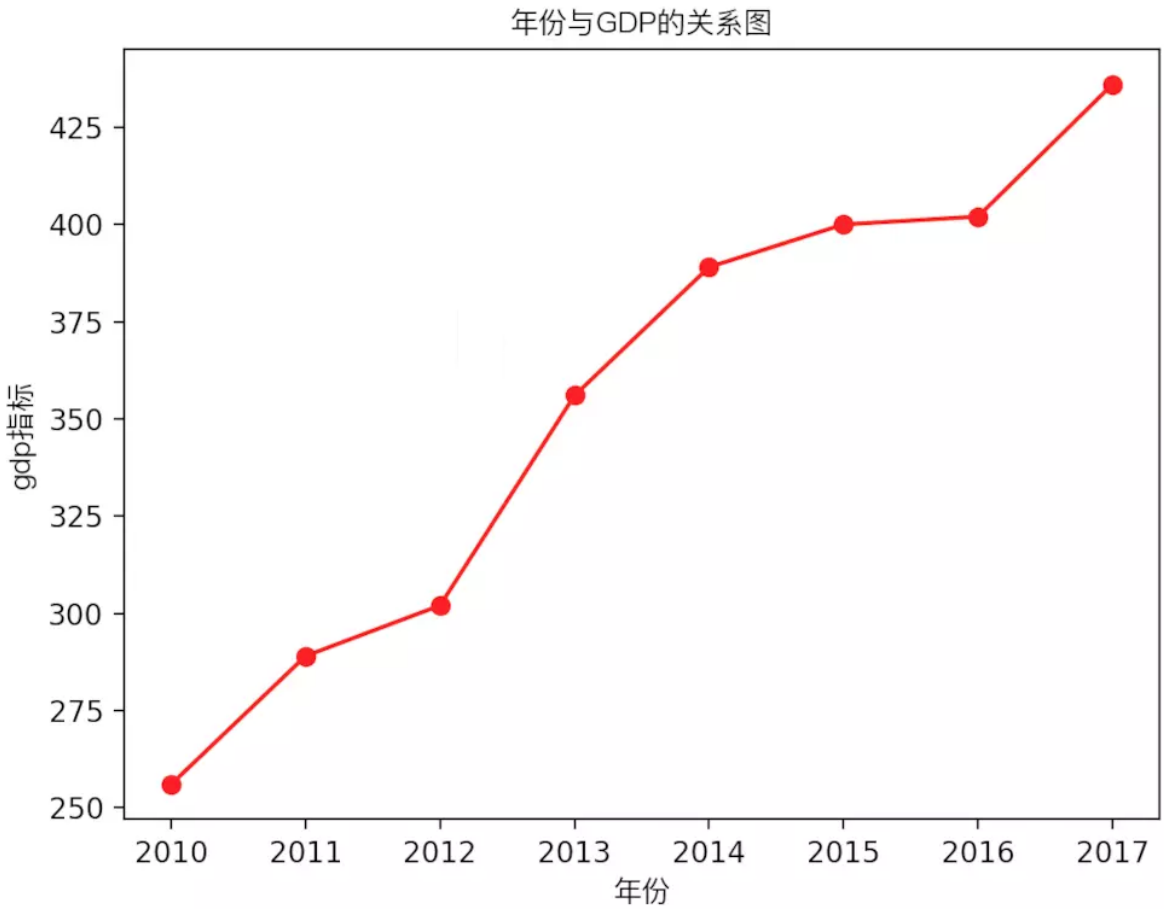
例如下面两个图为数字展示和图形展示：

```
US    13608
CN     2734
CA     1468
JP     1237
KR      993
GB      901
MX      579
TW      394
TR      326
PH      298
Name: Country, dtype: int64
```



1、1 折线图

折线图用于显示数据在一个连续的时间间隔或者时间跨度上的变化，它的特点是反映事物随时间或有序类别而变化的趋势。示例图如下：



折线图应用场景：

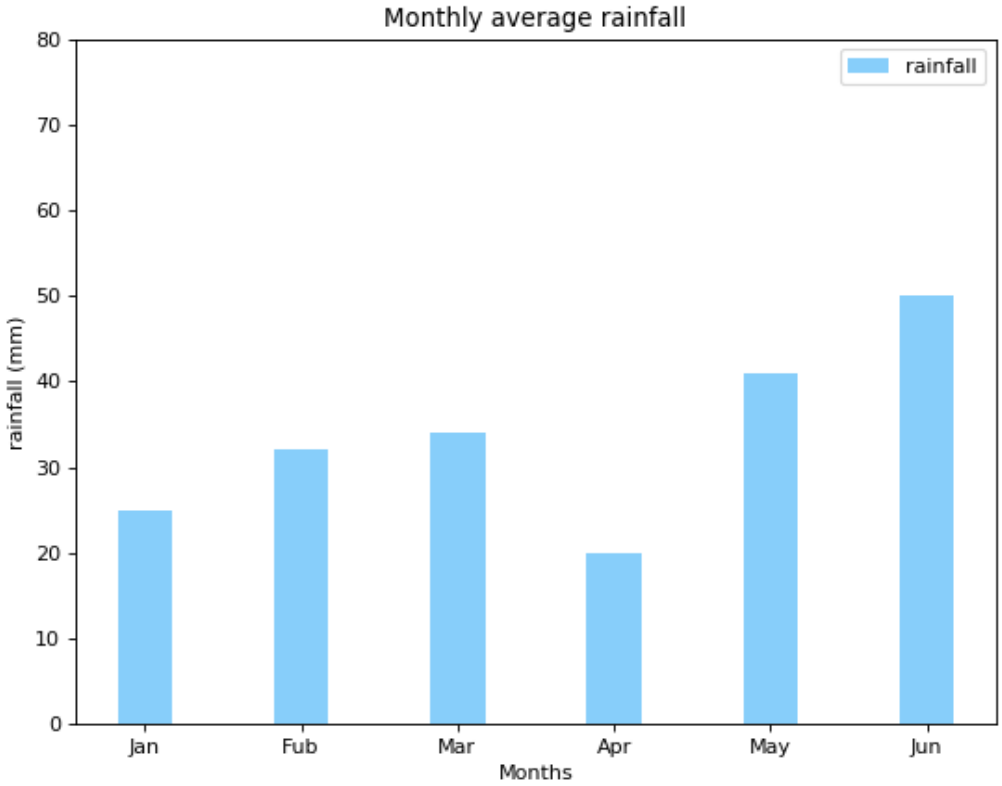
1. 折线图适合`x`轴是一个连续递增或递减的，对于没有规律的，则不适合使用折线图，建议使用柱状图。

2. 如果折线图条数过多，则不应该都绘制在一个图上。

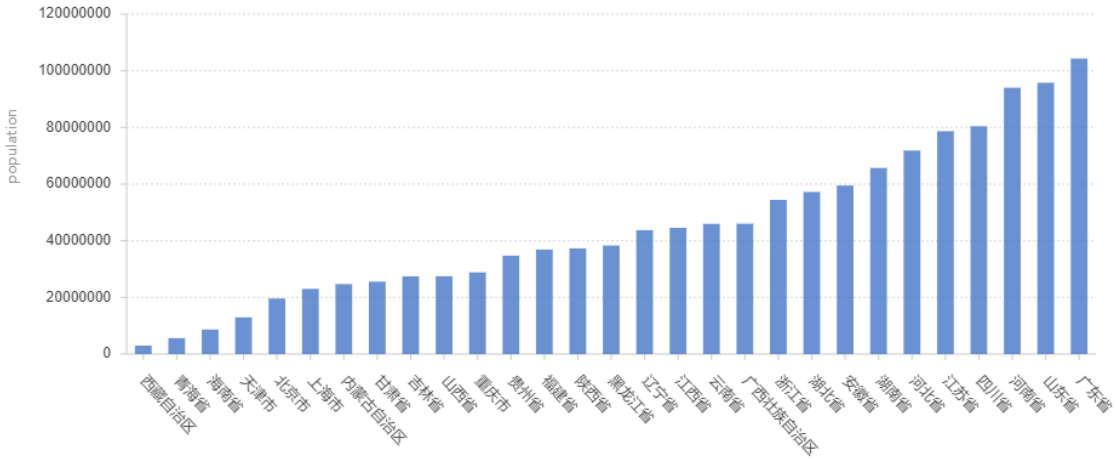
1、2 柱状图

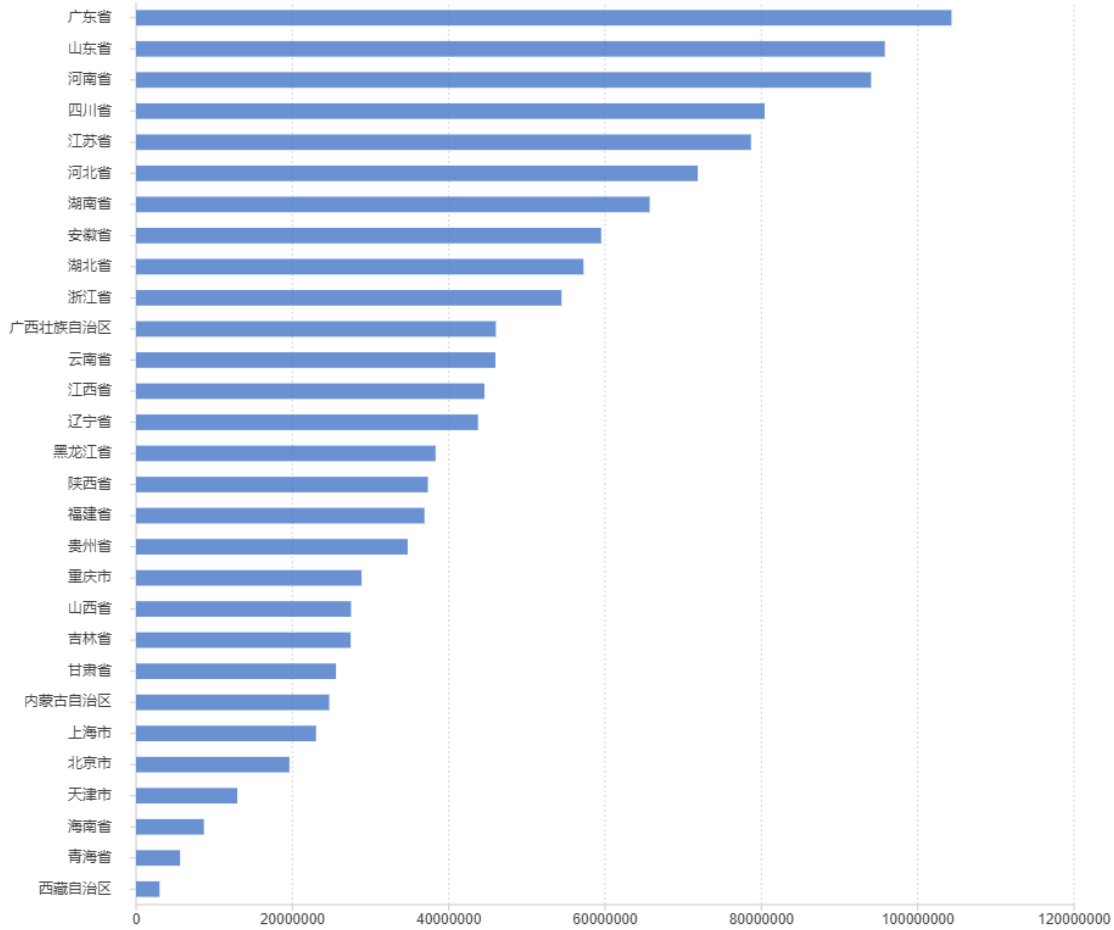
典型的柱状图（又名条形图），使用垂直或水平的柱子显示类别之间的数值比较。其中一个轴表示需要对比的分类，另一个轴代表相应的数值。常于数据的对比，通常用于A和B在想同时间的数据对比。

柱状图有别于直方图，柱状图无法显示数据在一个区间内的连续变化趋势。柱状图描述的是分类数据，回答的是每一个分类中"有多少？"这个问题。 示例图如下：



柱状图应用场景：

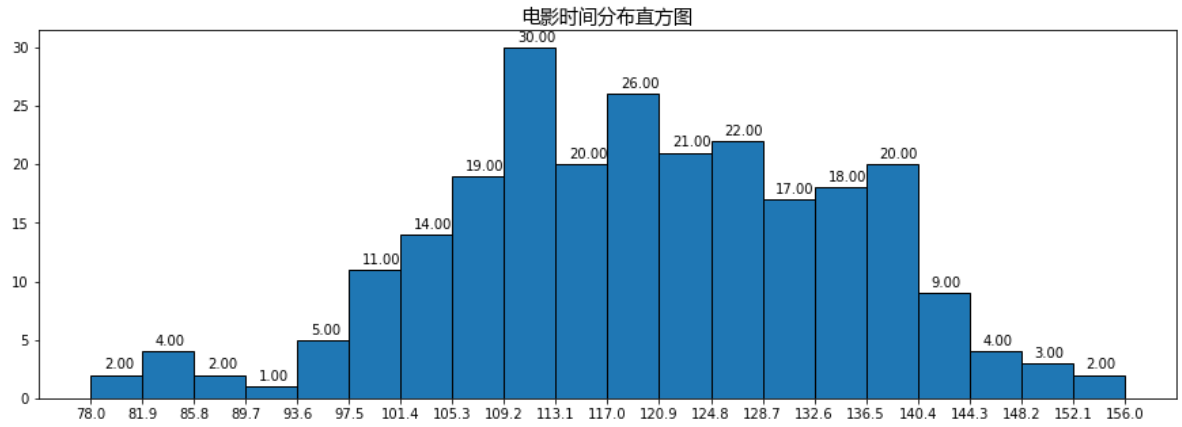
1. 适用于分类数据对比。
2. 垂直条形图最多不超过12个分类（也就是12个柱形），横向条形图最多不超过30个分类。如果垂直条形图的分类名太长，那么建议换成横向条形图。
3. 

4. 

5. 柱状图不适合表示趋势，如果想要表示趋势，应该使用折线图。

1、3 直方图

直方图(Histogram)，又称质量分布图，是一种统计报告图，由一系列高度不等的条纹表示数据分布的情况。一般用横轴表示数据类型，纵轴表示分布情况。 直方图是数值数据分布的精确图形表示。为了构建直方图，第一步是将值的范围分段，即将整个值的范围分成一系列间隔，然后计算每个间隔中有多少值。这些值通常被指定为连续的，不重叠的变量间隔。间隔必须相邻，并且通常是（但不是必须的）相等的大小。



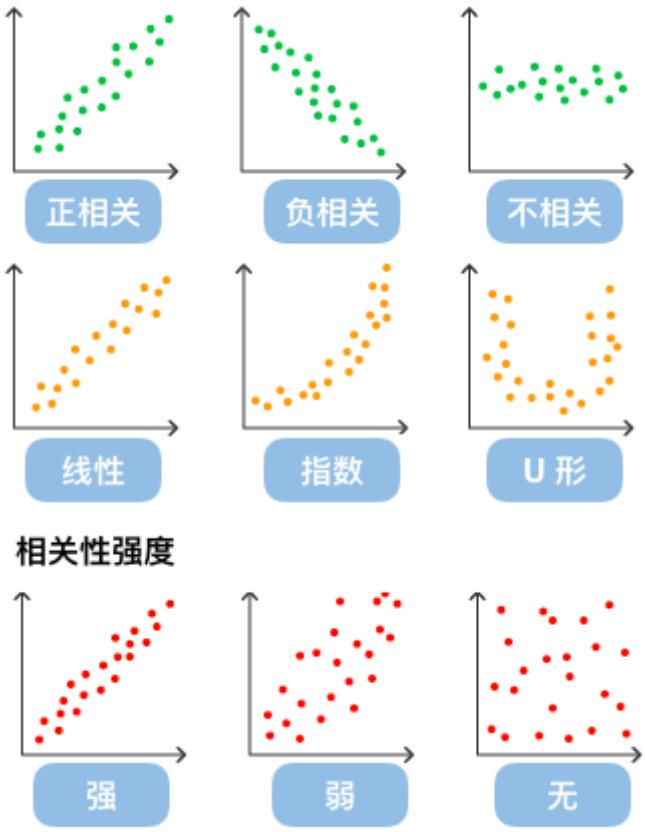
直方图的应用场景：

1. 显示各组数据数量分布的情况。
2. 用于观察异常或孤立数据。
3. 抽取的样本数量过小，将会产生较大误差，可信度低，也就失去了统计的意义。因此，样本数不应少于50个。

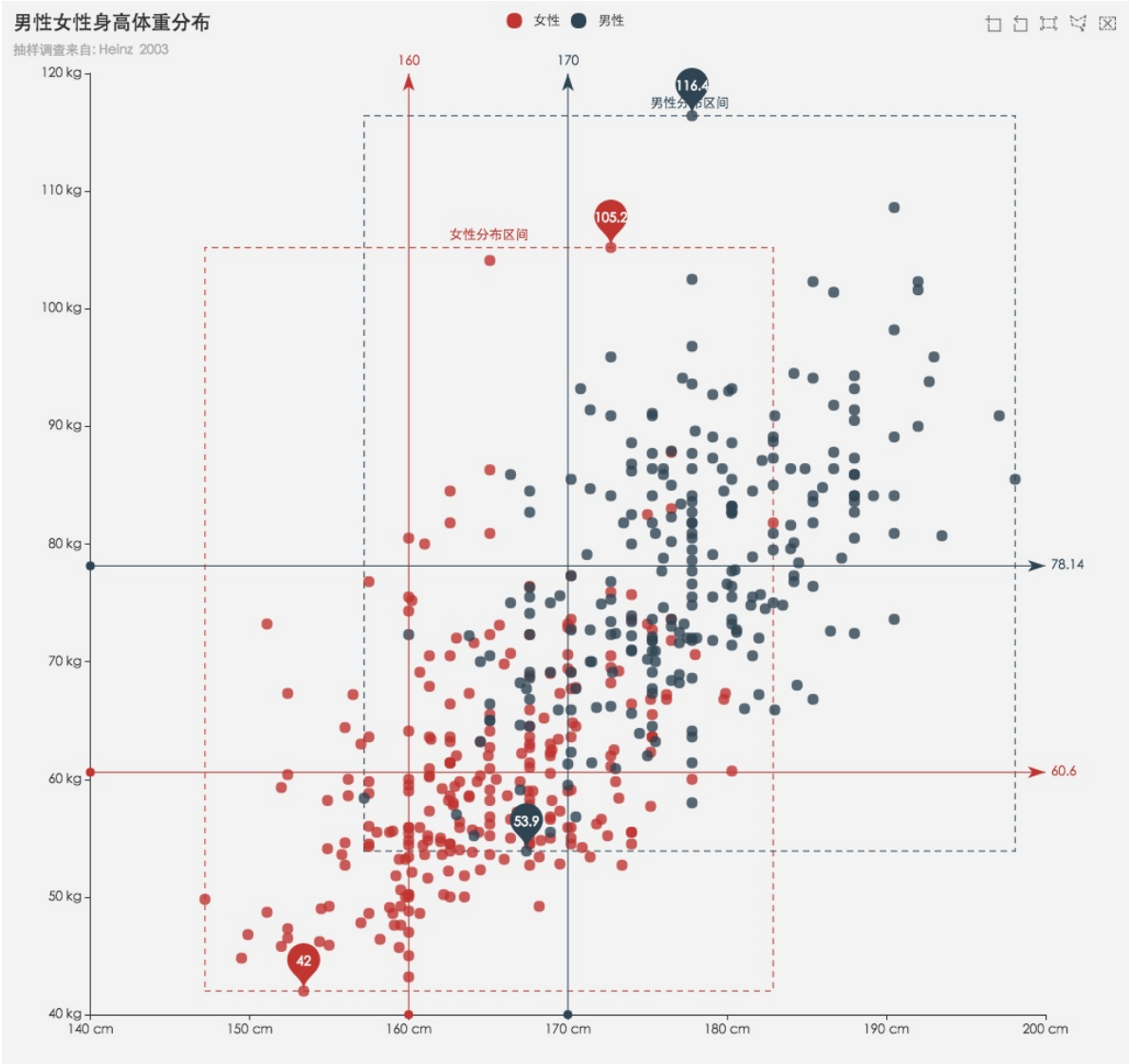
1、4 散点图

散点图也叫 X-Y 图，它将所有的数据以点的形式展现在直角坐标系上，以显示变量之间的相互影响程度，点的位置由变量的数值决定。

通过观察散点图上数据点的分布情况，我们可以推断出变量间的相关性。如果变量之间不存在相互关系，那么在散点图上就会表现为随机分布的离散的点，如果存在某种相关性，那么大部分的数据点就会相对密集并以某种趋势呈现。数据的相关关系主要分为：正相关（两个变量值同时增长）、负相关（一个变量值增加另一个变量值下降）、不相关、线性相关、指数相关等，表现在散点图上的大致分布如下图所示。那些离点集群较远的点我们称为离群点或者异常点。



示例图如下：

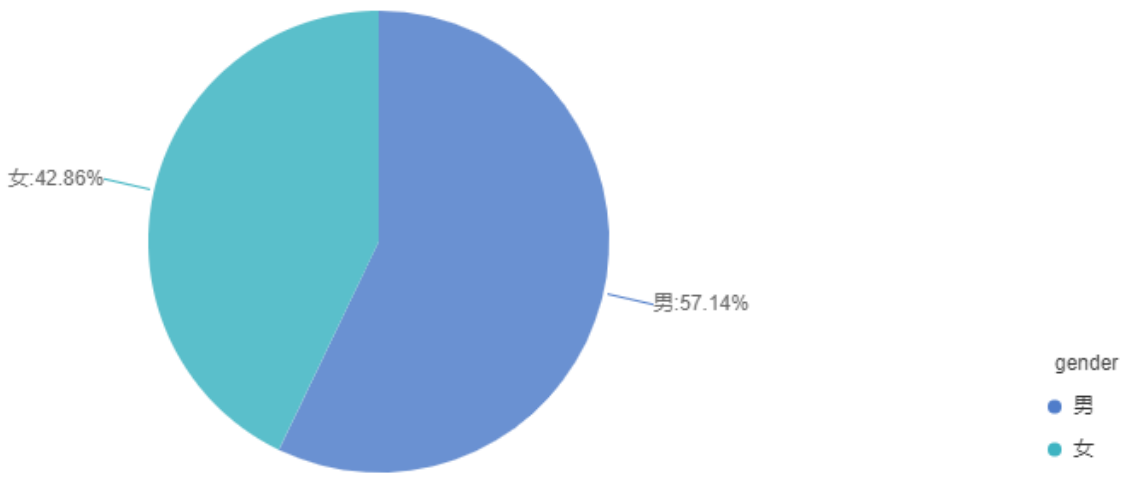


散点图的应用场景：

1. 观察数据集的分布情况。
2. 通过分析规律，根据样本数据特征计算出回归方程。

1、5 饼状图

饼状图通常用来描述量、频率和百分比之间的关系。在饼图中，每个扇区的弧长大小为其所表示的数量的比例。

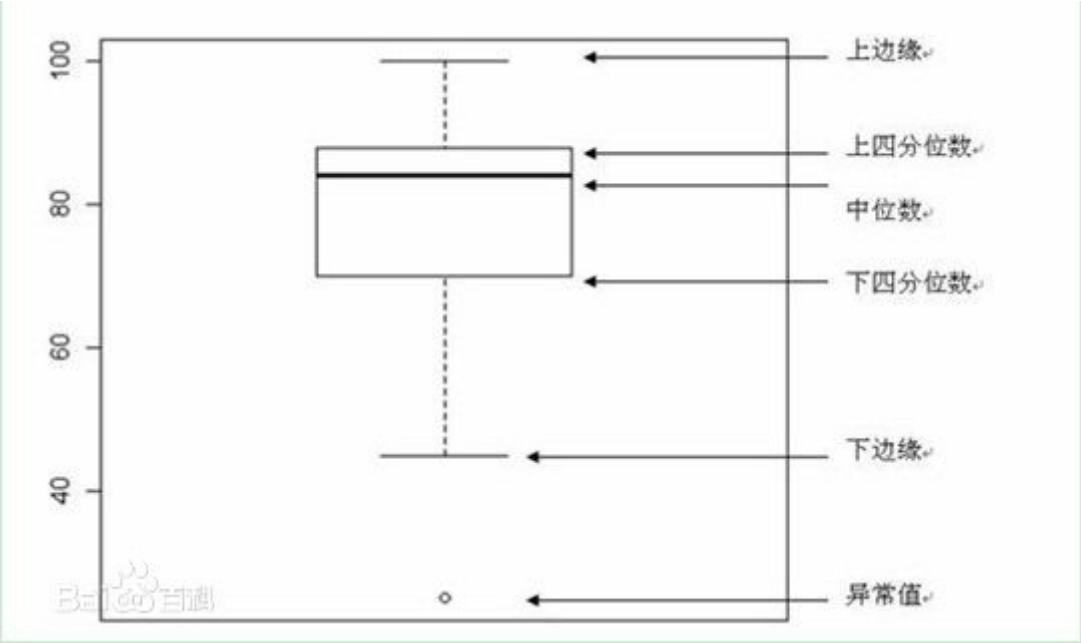


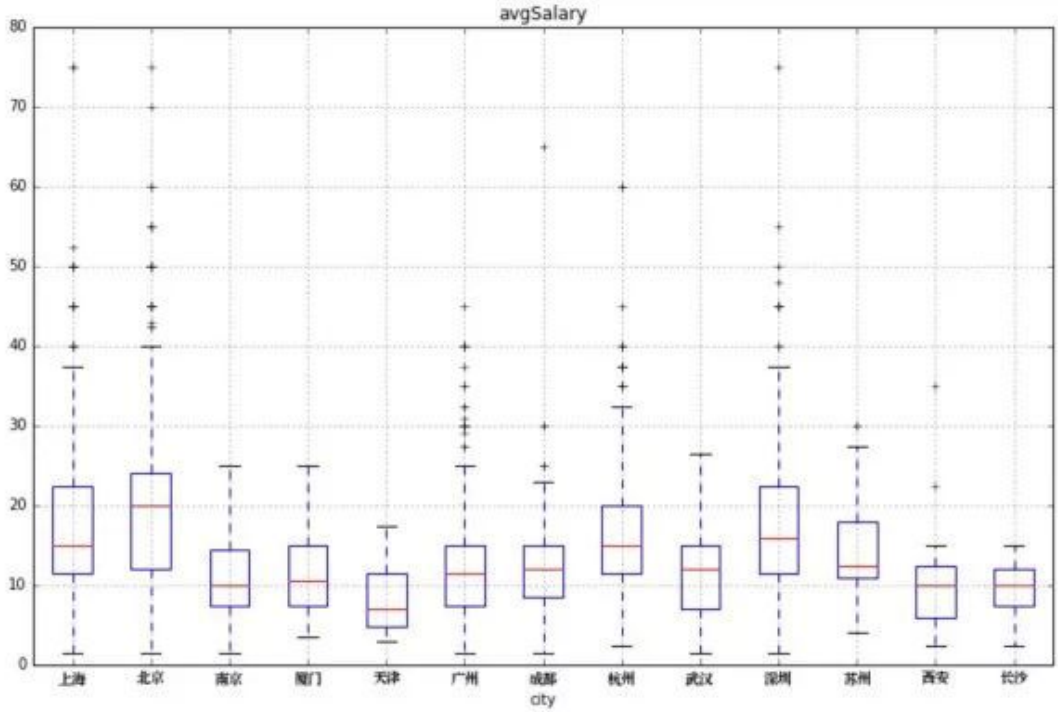
饼状图的应用场景：

1. 展示多个分类的占比情况，分类数量建议不超过9个。
2. 对于一些占比值非常接近的，不建议使用饼状图，可以使用柱状图。

1、6 箱线图

箱线图（Box-plot）又称为盒须图、盒式图或箱型图，是一种用作显示一组数据分散情况资料的统计图。因形状如箱子而得名。在各种领域也经常被使用，它主要用于反映原始数据分布的特征，还可以进行多组数据分布特征的比较。箱线图的绘制方法是：先找出一组数据的**上限值、下限值、中位数（Q2）和下四分位数（Q1）以及上四分位数（Q3）**；然后，连接两个四分位数画出箱子；再将最大值和最小值与箱子相连接，中位数在箱子中间。





> 四分位数（Quartile）也称四分位点，是指在统计学中把所有数值由小到大排列并分成四等份，处于三个分割点位置的数值。多应用于统计学中的箱线图绘制。它是一组数据排序后处于25%和75%位置上的值。四分位数是通过3个点将全部数据等分为4部分，其中每部分包含25%的数据。很显然，中间的四分位数就是中位数，因此通常所说的四分位数是指处在25%位置上的数值（称为下四分位数）和处在75%位置上的数值（称为上四分位数）。与中位数的计算方法类似，根据未分组数据计算四分位数时，首先对数据进行排序，然后确定四分位数所在的位置，该位置上的数值就是四

> 分位数。与中位数不同的是，四分位数位置的确定方法有几种，每种方法得到的结果会有一定差异，但差异不会很大。
>
> 上限的计算规则是： IQR=Q3-Q1 上限=Q3+1.5IQR 下限=Q1-1.5IQR

箱线图的应用场景：

1. 直观明了地识别数据中的异常值。
2. 利用箱线图判断数据的偏态。
3. 利用箱线图比较几批数据的形状。
4. 箱线图适合比较多组数据，如果知识要看一组数据的分布情况，建议使用直方图。

更多图标：https://antv.vision/zh

二、Matplotlib

`Matplotlib` 是一个 `Python` 的 `2D` 绘图库，通过 `Matplotlib` ，开发者可以仅需要几行代码，便可以生成折线图，直方图，条形图，饼状图，散点图等。

- 是专门用于开发2D图表(包括3D图表)
- 以渐进、交互式方式实现数据可视化

```
# 安装模块
pip install matplotlib
```

2、1 基础使用

1. matplotlib.pyplot 模块包含了一系列类似于 matlab 的画图函数。

   ```
   import matplotlib.pyplot as plt
   ```

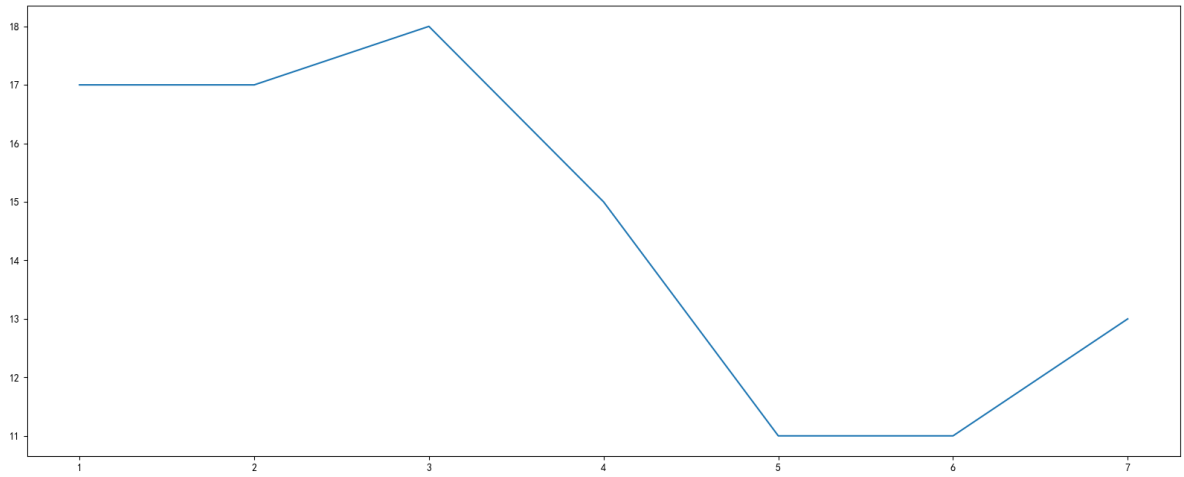
2. 创建画布 -- plt.figure()

   ```
   plt.figure(figsize=(), dpi=)
   # figsize:指定图的长宽
   # dpi:图像的清晰度
   # 返回fig对象
   ```

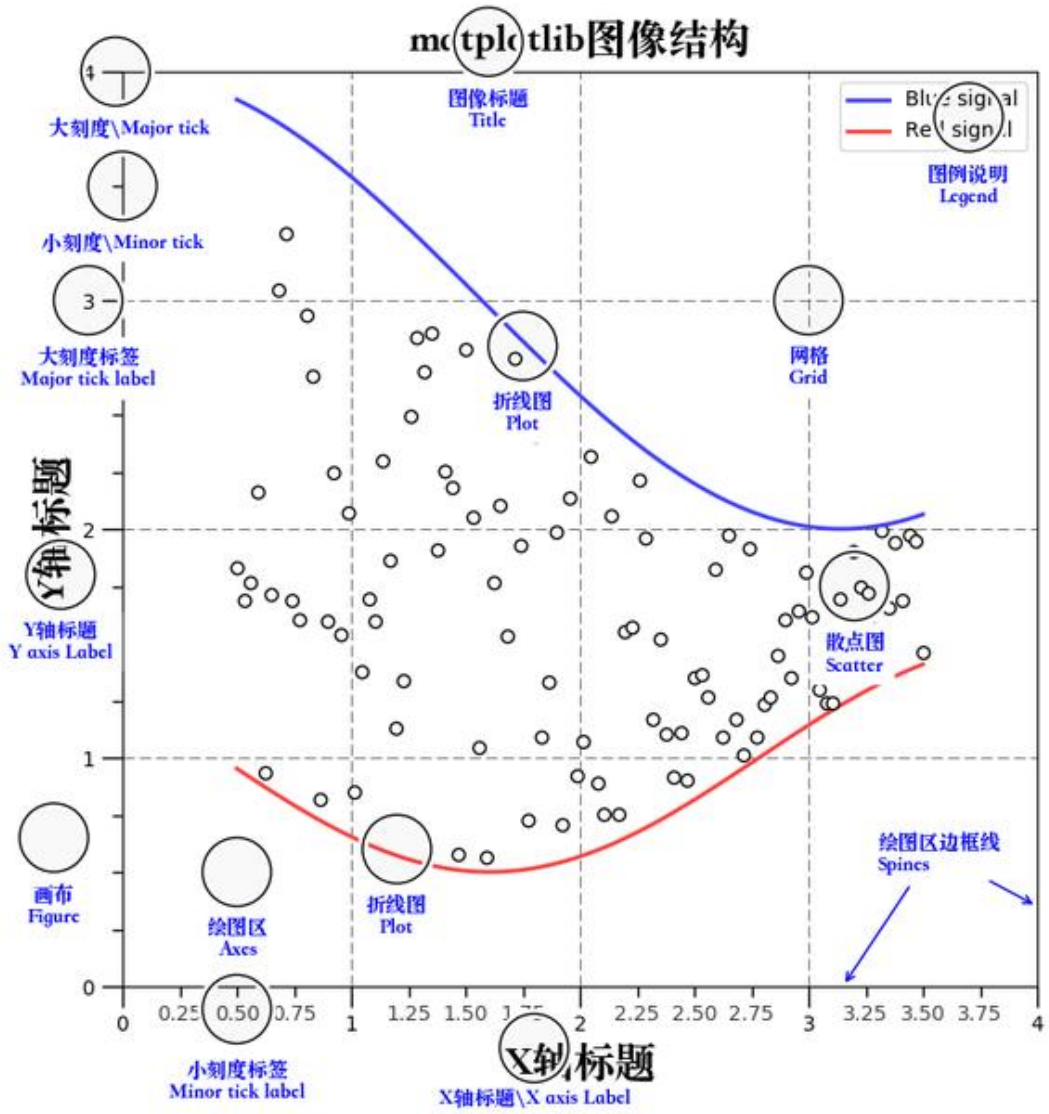
3. 绘制图像 -- plt.plot(x, y)
4. 显示图像 -- plt.show()

折线图绘制与显示

```
# 0.准备数据
x = [1, 2, 3, 4, 5, 6, 7]
y_shanghai = [17, 17, 18, 15, 11, 11, 13]

# 1.创建画布
plt.figure(figsize=(20, 8), dpi=100)

# 2.绘制图像
plt.plot(x, y_shanghai)

# 3.图像显示
plt.show()
```



2、1、1 Matplotlib图像结构(了解)

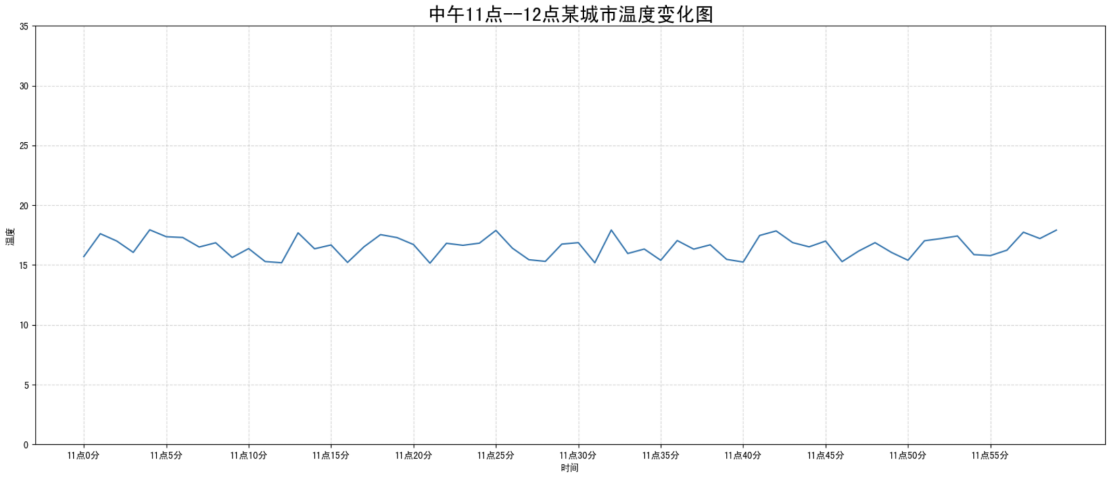


2、2 添加辅助功能

为了更好地理解所有基础绘图功能，我们通过天气温度变化的绘图来融合所有的基础API使用

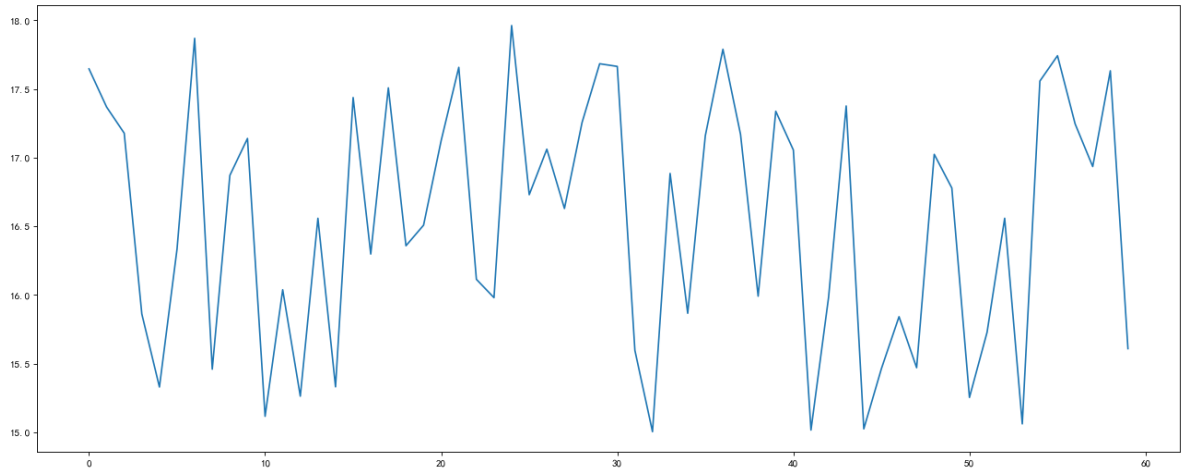
需求：画出某城市11点到12点1小时内每分钟的温度变化折线图，温度范围在15度~18度

效果：



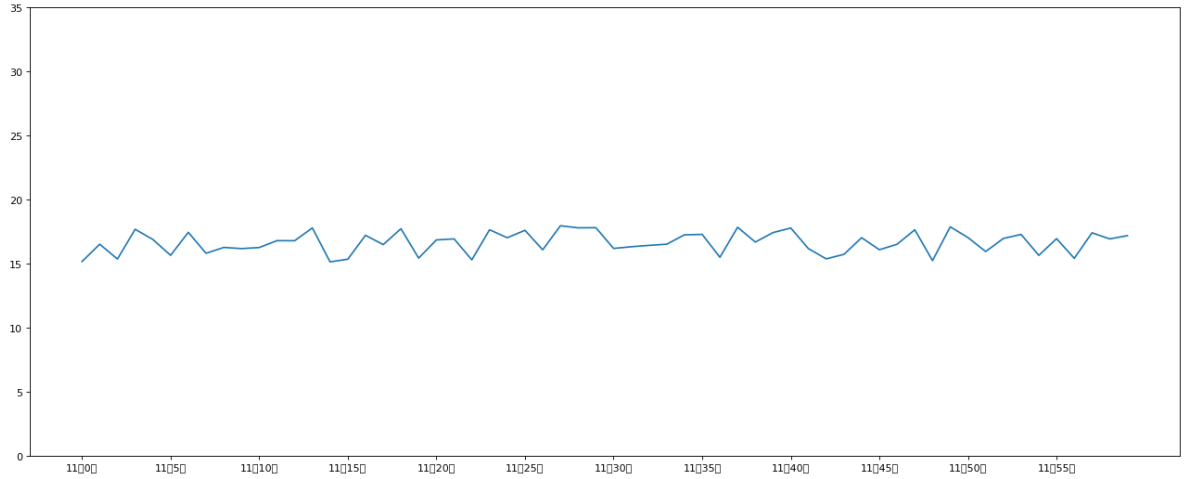
2、2、1 准备数据并画出初始折线图

```
import matplotlib.pyplot as plt
import random

# 画出温度变化图

# 0.准备x, y坐标的数据
x = range(60)
y_shanghai = [random.uniform(15, 18) for i in x]

# 1.创建画布
plt.figure(figsize=(20, 8), dpi=80)

# 2.绘制折线图
plt.plot(x, y_shanghai)

# 3.显示图像
plt.show()
```



2、2、2 添加自定义x,y刻度

```
# 增加以下两行代码

# 2.1 添加x,y轴刻度
# 设置x,y轴刻度
x_ticks_label = ["11点{}分".format(i) for i in x]
y_ticks = range(40)

# 修改x,y轴坐标刻度显示
# plt.xticks(x_ticks_label[::5]) # 坐标刻度不可以直接通过字符串进行修改
plt.xticks(x[::5], x_ticks_label[::5])
plt.yticks(y_ticks[::5])
```

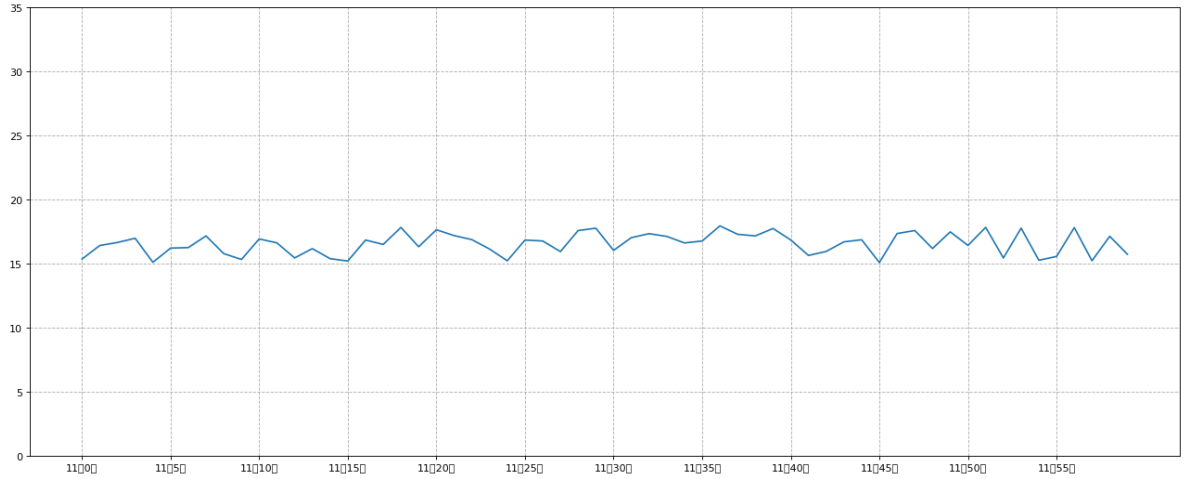


这一步可能会出现中文乱码问题，请查看附录乱码解决方案

2、2、3 添加网格显示

为了更加清楚地观察图形对应的值

```
# 2.2 添加网格显示
plt.grid(True, linestyle="--", alpha=1)
```



2、2、4 添加描述信息

添加x轴、y轴描述信息及标题

> 通过fontsize参数可以修改图像中字体的大小

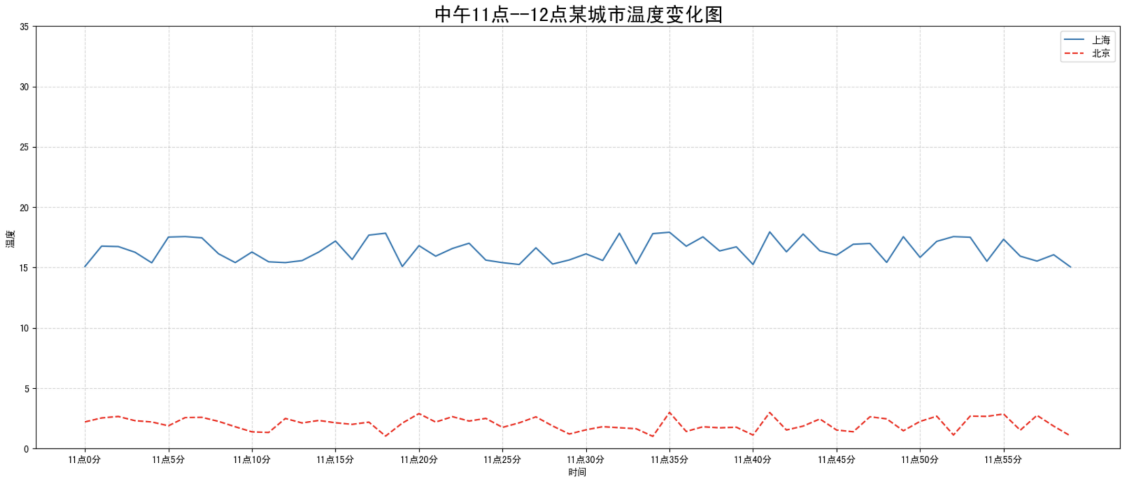
```
# 2.3 添加描述信息
plt.xlabel("时间")
plt.ylabel("温度")
plt.title("中午11点-12点某城市温度变化图", fontsize=20)
```

2、3 绘制多个图像

2、3、1 多次plot

需求：再添加一个城市的温度变化

收集到北京当天温度变化情况，温度在1度到3度。怎么去添加另一个在同一坐标系当中的不同图形，**其实很简单只需要再次plot即可**，但是需要区分线条，如下显示




```
# 增加北京的温度数据
y_beijing = [random.uniform(1, 3) for i in x]

# 2.绘制图像
plt.plot(x, y_shanghai, label="上海")
plt.plot(x, y_beijing, color="r", linestyle="--", label="北京")  # 新增绘制北京的数据
```

我们仔细观察，用到了两个新的地方，一个是对于不同的折线展示效果，一个是添加图例。

2、3、2 设置图形风格

颜色字符	风格字符
r 红色	- 实线
g 绿色	- - 虚线
b 蓝色	-. 点划线
w 白色	: 点虚线
c 青色	' ' 留空、空格
m 洋红	
y 黄色	
k 黑色	

2、3、3 显示图例

注意：如果只在plt.plot()中设置label还不能最终显示出图例，还需要通过plt.legend()将图例显示出来。

```
# 绘制折线图
plt.plot(x, y_shanghai, label="上海")
# 使用多次plot可以画多个折线
plt.plot(x, y_beijing, color='r', linestyle='--', label="北京")

# 显示图例
plt.legend(loc="best")
```

Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

完整代码：

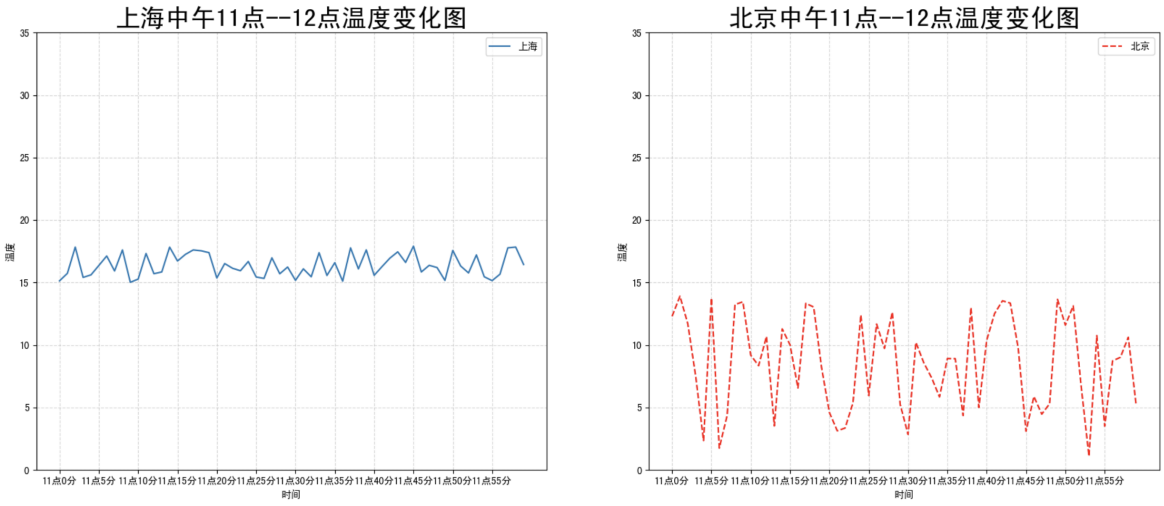
```
# 0.准备数据
x = range(60)
y_shanghai = [random.uniform(15, 18) for i in x]
y_beijing = [random.uniform(1,3) for i in x]

# 1.创建画布
plt.figure(figsize=(20, 8), dpi=100)

# 2.绘制图像
plt.plot(x, y_shanghai, label="上海")
plt.plot(x, y_beijing, color="r", linestyle="--", label="北京")

# 2.1 添加x,y轴刻度
# 构造x,y轴刻度标签
x_ticks_label = ["11点{}分".format(i) for i in x]
y_ticks = range(40)

# 刻度显示
```

```
plt.xticks(x[::5], x_ticks_label[::5])
plt.yticks(y_ticks[::5])

# 2.2 添加网格显示
plt.grid(True, linestyle="--", alpha=0.5)

# 2.3 添加描述信息
plt.xlabel("时间")
plt.ylabel("温度")
plt.title("中午11点--12点某城市温度变化图", fontsize=20)

# 2.4 图像保存
plt.savefig("./test.png")

# 2.5 添加图例
plt.legend(loc=0)


# 3.图像显示
plt.show()
```

2、3、4 多个坐标系显示

plt.subplots(面向对象的画图方法)

如果我们想要将上海和北京的天气图显示在同一个图的不同坐标系当中，效果如下：



可以通过subplots函数实现(旧的版本中有subplot，使用起来不方便)，推荐subplots函数

- matplotlib.pyplot.subplots(nrows=1, ncols=1, **fig_kw) 创建一个带有多个axes(坐标系/绘图区)的图

```
Parameters:

nrows, ncols : 设置有几行几列坐标系
    int, optional, default: 1, Number of rows/columns of the subplot grid.

Returns:
fig : 图对象
axes : 返回相应数量的坐标系

设置标题等方法不同:
```

```
    set_xticks
    set_yticks
    set_xlabel
    set_ylabel
```

关于axes子坐标系的更多方法：参考 https://matplotlib.org/stable/api/axes_api.html

- 注意：**plt.函数名()**相当于面向过程的画图方法，**axes.set_方法名()**相当于面向对象的画图方法。

```
# 0.准备数据
x = range(60)
y_shanghai = [random.uniform(15, 18) for i in x]
y_beijing = [random.uniform(1, 5) for i in x]

# 1.创建画布
# plt.figure(figsize=(20, 8), dpi=100)
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20, 8), dpi=100)

# 2.绘制图像
# plt.plot(x, y_shanghai, label="上海")
# plt.plot(x, y_beijing, color="r", linestyle="--", label="北京")
axes[0].plot(x, y_shanghai, label="上海")
axes[1].plot(x, y_beijing, color="r", linestyle="--", label="北京")

# 2.1 添加x,y轴刻度
# 构造x,y轴刻度标签
x_ticks_label = ["11点{}分".format(i) for i in x]
y_ticks = range(40)

# 刻度显示
# plt.xticks(x[::5], x_ticks_label[::5])
# plt.yticks(y_ticks[::5])
axes[0].set_xticks(x[::5])
axes[0].set_yticks(y_ticks[::5])
axes[0].set_xticklabels(x_ticks_label[::5])
axes[1].set_xticks(x[::5])
axes[1].set_yticks(y_ticks[::5])
axes[1].set_xticklabels(x_ticks_label[::5])

# 2.2 添加网格显示
# plt.grid(True, linestyle="--", alpha=0.5)
axes[0].grid(True, linestyle="--", alpha=0.5)
axes[1].grid(True, linestyle="--", alpha=0.5)

# 2.3 添加描述信息
# plt.xlabel("时间")
# plt.ylabel("温度")
# plt.title("中午11点--12点某城市温度变化图", fontsize=20)
axes[0].set_xlabel("时间")
axes[0].set_ylabel("温度")
axes[0].set_title("中午11点--12点某城市温度变化图", fontsize=20)
axes[1].set_xlabel("时间")
axes[1].set_ylabel("温度")
axes[1].set_title("中午11点--12点某城市温度变化图", fontsize=20)
```

```
# # 2.4 图像保存
plt.savefig("./test.png")

# # 2.5 添加图例
# plt.legend(loc=0)
axes[0].legend(loc=0)
axes[1].legend(loc=0)

# 3.图像显示
plt.show()
```

小结

- 添加x,y轴刻度【知道】
  - plt.xticks()
  - plt.yticks()
  - **注意:在传递进去的第一个参数必须是数字,不能是字符串,如果是字符串吗,需要进行替换操作**
- 添加网格显示【知道】
  - plt.grid(linestyle="--", alpha=0.5)
- 添加描述信息【知道】
  - plt.xlabel()
  - plt.ylabel()
  - plt.title()
- 图像保存【知道】
  - plt.savefig("路径")
- 多次plot【了解】
  - 直接进行添加就OK
- 显示图例【知道】
  - plt.legend(loc="best")
  - **注意:一定要在plt.plot()里面设置一个label,如果不设置,没法显示**
- 多个坐标系显示【了解】
  - plt.subplots(nrows=, ncols=)
- 折线图的应用【知道】
  - 1.应用于观察数据的变化
  - 2.可是画出一些数学函数图像

三、常见图形绘制

Matplotlib能够绘制**折线图、散点图、柱状图、直方图、饼图。**

我们需要知道不同的统计图的意义，以此来决定选择哪种统计图来呈现我们的数据。

3、1 折线图

以折线的上升或下降来表示统计数量的增减变化的统计图

特点：能够显示数据的变化趋势，反映事物的变化情况。(变化)

api：plt.plot(x, y)

3、2 柱状图（条形图）

排列在工作表的列或行中的数据可以绘制到柱状图中。

特点：绘制连离散的数据,能够一眼看出各个数据的大小,比较数据之间的差别。(统计/对比)

api：plt.bar(x, width, align='center', **kwargs)

应用场景：数量统计，频率统计。

条形图的绘制方式跟折线图非常的类似，只不过是换成了`plt.bar`方法。`plt.bar`方法有以下常用参数：

1. `x`：一个数组或者列表，代表需要绘制的条形图的x轴的坐标点。
2. `height`：一个数组或者列表，代表需要绘制的条形图y轴的坐标点。
3. `width`：每一个条形图的宽度，默认是0.8的宽度。
4. `bottom`：`y`轴的基线，默认是0，也就是距离底部为0.
5. `align`：对齐方式，默认是`center`，也就是跟指定的`x`坐标居中对齐，还有为`edge`，靠边对齐，具体靠右边还是靠左边，看`width`的正负。
6. `color`：条形图的颜色。

返回值为`BarContainer`，是一个存储了条形图的容器，而条形图实际上的类型是`matplotlib.patches.Rectangle`对象。

更多参考：https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html#matplotlib.pyplot.bar

3、2、1 条形图的绘制

比如现在有`2019`年贺岁片票房的数据（数据来源：https://piaofang.maoyan.com/dashboard）

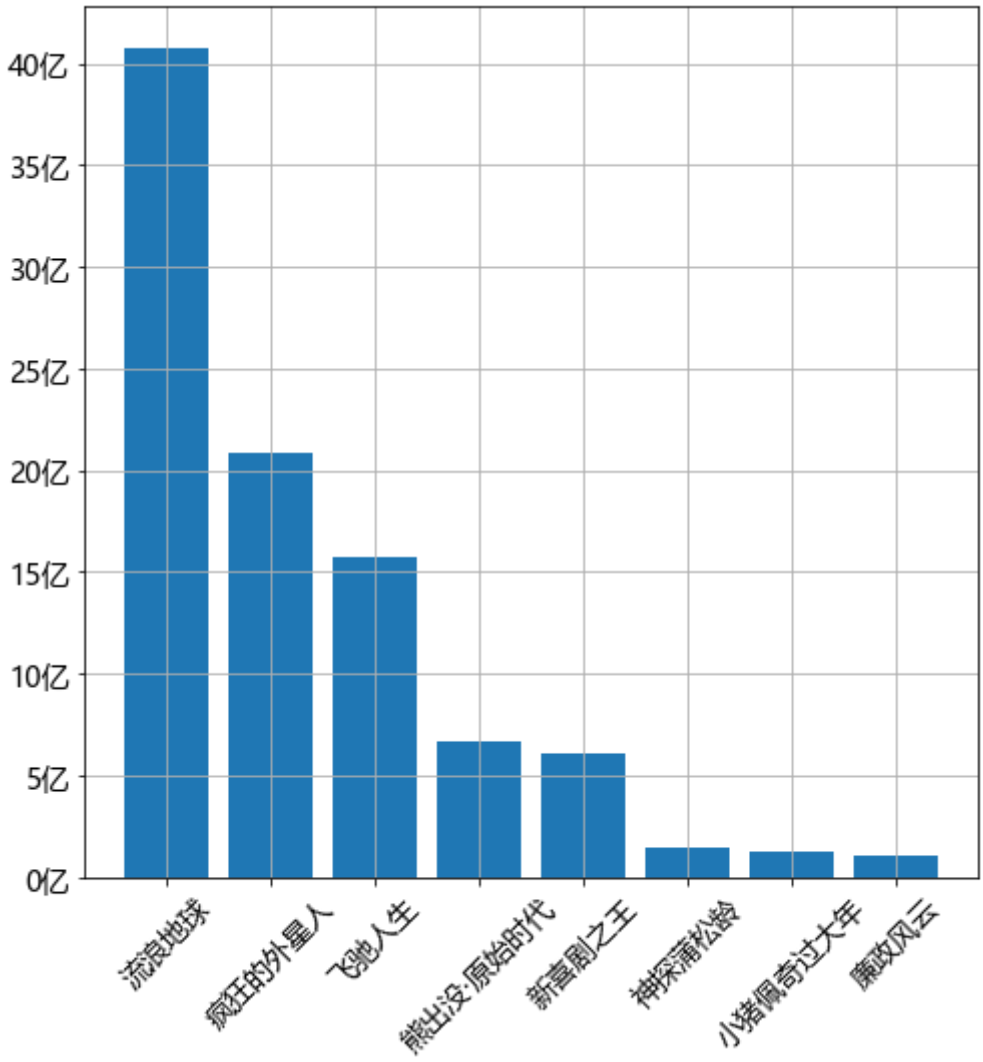
```
#票房单位亿元
movies = {
    "流浪地球":40.78,
    "飞驰人生":15.77,
    "疯狂的外星人":20.83,
    "新喜剧之王":6.10,
    "廉政风云":1.10,
    "神探蒲松龄":1.49,
    "小猪佩奇过大年":1.22,
    "熊出没·原始时代":6.71
}
```

用条形图绘制每部电影及其票房的代码如下：

```
import matplotlib.pyplot as plt
import random
# 设置显示中文字体
plt.rcParams['font.sans-serif'] = ['SimHei'] # 指定默认字体
# 设置正常显示符号
plt.rcParams['axes.unicode_minus'] = False
```

```
plt.figure(figsize=(20, 8), dpi=100)

movies = {
    "流浪地球": 40.78,
    "飞驰人生": 15.77,
    "疯狂的外星人": 20.83,
    "新喜剧之王": 6.10,
    "廉政风云": 1.10,
    "神探蒲松龄": 1.49,
    "小猪佩奇过大年": 1.22,
    "熊出没·原始时代": 6.71
}

plt.bar(range(len(movies)), list(movies.values()))
plt.xticks(range(len(movies)), list(movies.keys()))
plt.grid()
```

效果图如下：



其中`xticks`和`yticks`的用法跟之前的折线图一样。这里新出现的方法是`bar`，`bar`常用的有3个参数，分别是`x`（x轴的坐标点），`y`（y轴的坐标点）以及`width`（条形的宽度）。

3、2、2 横向条形图

横向条形图需要使用`plt.barh`这个方法跟`bar`非常的类似，只不过把方向进行旋转。参数跟`bar`类似，但也有区别。如下：

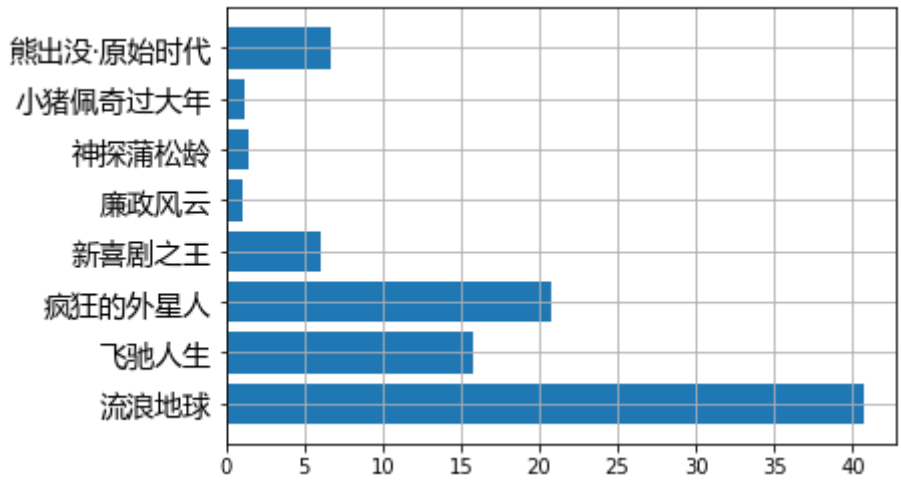
1. `y`：数组或列表，代表需要绘制的条形图在`y`轴上的坐标点。
2. `width`：数组或列表，代表需要绘制的条形图在`x`轴上的值（也就是长度）。
3. `height`：条形图的高度，默认是0.8。
4. `left`：条形图的基线，也就是距离y轴的距离。
5. 其他参数跟`bar`一样。

返回值也是`BarContainer`容器对象。

还是以以上数据为例，将电影名和票房反转一下。示例代码如下：

```
plt.figure(figsize=(20, 8), dpi=100)

movies = {
    "流浪地球": 40.78,
    "飞驰人生": 15.77,
    "疯狂的外星人": 20.83,
    "新喜剧之王": 6.10,
    "廉政风云": 1.10,
    "神探蒲松龄": 1.49,
    "小猪佩奇过大年": 1.22,
    "熊出没·原始时代": 6.71
}

plt.barh(range(len(movies)), list(movies.values()))
plt.yticks(range(len(movies)), list(movies.keys()))  # 修改轴坐标
plt.grid()
```

效果图如下：



3、3、3 分组条形图

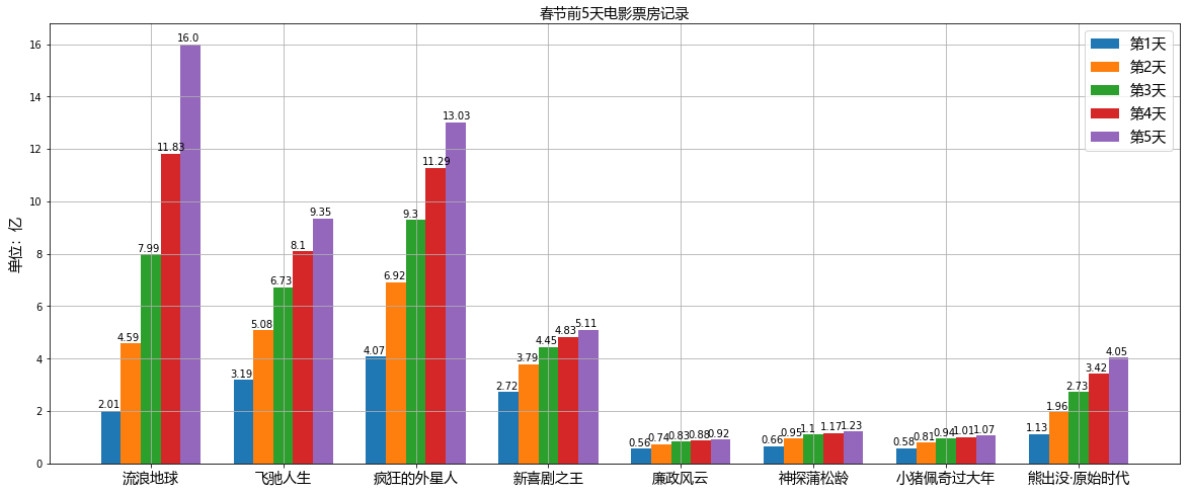
现在有一组数据，是2019年春节贺岁片前五天的电影票房记录。 示例代码如下：

```
import matplotlib.pyplot as plt
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']  # 指定默认字体
plt.rcParams['axes.unicode_minus'] = False

movies = {
    "流浪地球": [2.01, 4.59, 7.99, 11.83, 16],
    "飞驰人生": [3.19, 5.08, 6.73, 8.10, 9.35],
    "疯狂的外星人": [4.07, 6.92, 9.30, 11.29, 13.03],
    "新喜剧之王": [2.72, 3.79, 4.45, 4.83, 5.11],
    "廉政风云": [0.56, 0.74, 0.83, 0.88, 0.92],
    "神探蒲松龄": [0.66, 0.95, 1.10, 1.17, 1.23],
    "小猪佩奇过大年": [0.58, 0.81, 0.94, 1.01, 1.07],
    "熊出没·原始时代": [1.13, 1.96, 2.73, 3.42, 4.05]
}
plt.figure(figsize=(20, 8))
width = 0.75
bin_width = width / 5

ind = range(0, len(movies))

movie_data = list(movies.values())
print(movie_data)

every_day = []
for i in range(len(movie_data[0])):
    every_day.append([
        movie_data[0][i],
        movie_data[1][i],
        movie_data[2][i],
        movie_data[3][i],
        movie_data[4][i],
        movie_data[5][i],
        movie_data[6][i],
        movie_data[7][i],
    ])

print(every_day)
# 第一种方案

# plt.bar([i - bin_width * 2 for i in ind], every_day[0], width=bin_width,
label='第一天')
#
# plt.bar([i - bin_width for i in ind], every_day[1], width=bin_width, label='第
二天')
#
# plt.bar(ind, every_day[2], width=bin_width, label='第三天')
#
# plt.bar([i + bin_width for i in ind], every_day[3], width=bin_width, label='第
四天')
#
# plt.bar([i + bin_width * 2 for i in ind], every_day[4], width=bin_width,
label='第五天')

# 第二种方案
for index in range(len(every_day)):
    day_tickets = every_day[index]
```

```
    xs = [i - (bin_width * (2 - index)) for i in ind]
    plt.bar(xs, day_tickets, width=bin_width, label="第%d天" % (index + 1))
    # 添加坐标上的数字
    for ticket, x in zip(day_tickets, xs):
        plt.annotate(ticket, xy=(x, ticket), xytext=(x - 0.1, ticket + 0.1))

# 设置图例
plt.legend()
plt.ylabel("单位：亿")
plt.title("春节前5天电影票房记录")
# 设置x轴的坐标
plt.xticks(ind, movies.keys())
plt.grid(True)
plt.show()
```

示例图如下：

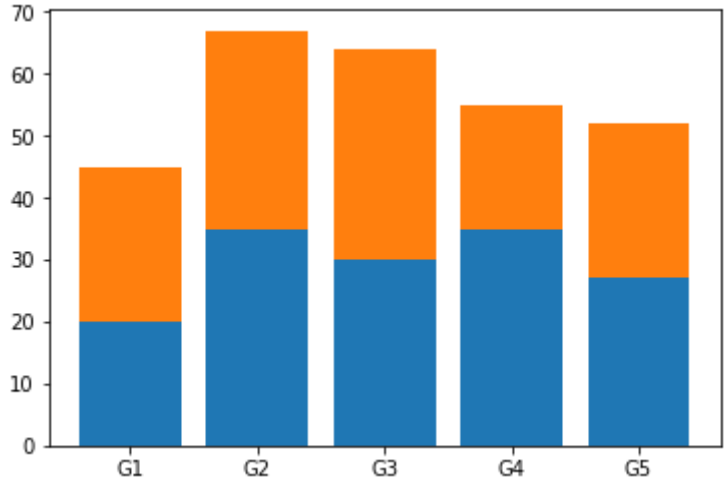


3、3、4 堆叠条形图

堆叠条形图，是将一组相关的条形图堆叠在一起进行比较的条形图。比如以下案例：

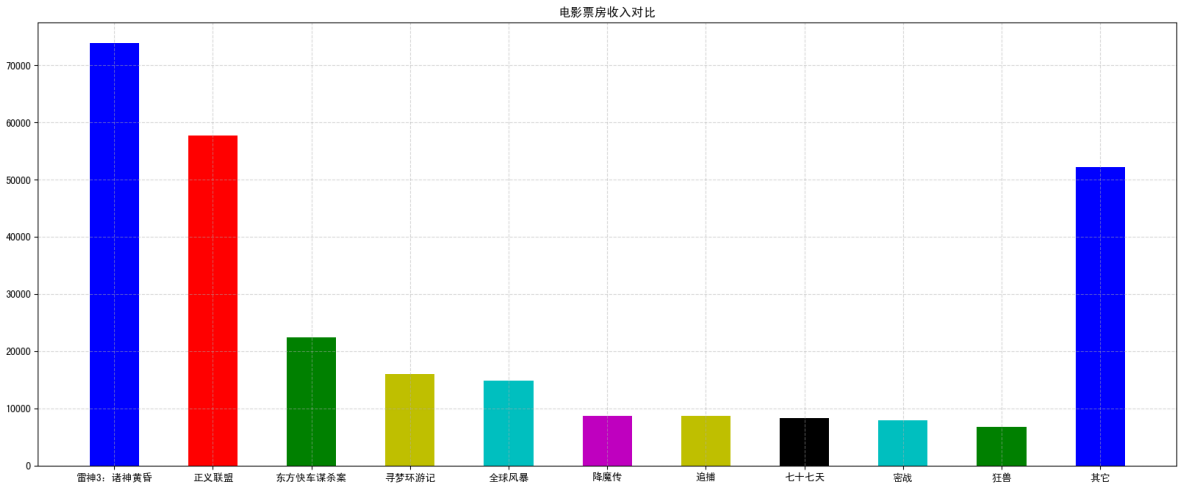
```
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']  # 指定默认字体
plt.rcParams['axes.unicode_minus'] = False

# 准备数据
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
groupNames = ('G1', 'G2', 'G3', 'G4', 'G5')

# 生成序号
xs = range(len(menMeans))
plt.bar(xs, menMeans)
plt.bar(xs, womenMeans, bottom=menMeans)
plt.xticks(xs, groupNames)
plt.show()
```

效果图如下：



在绘制女性得分的条形图的时候，因为要堆叠在男性得分的条形图上，所以使用到了一个 `bottom` 参数，就是距离 `x` 轴的距离。通过对贴条形图，我们就可以清楚的知道，哪一个队伍的综合排名是最高的，并且在每个队伍中男女的得分情况。

作业-需求-对比每部电影的票房收入



电影数据如下图所示：

排名.影片名	单月票房(万)	月度占比	平均票价	场均人次	上映日期	口碑指数	月内天数
1.雷神3：诸神黄昏	73853	26.6%	35	15	2017-11-03	-	28
2.正义联盟	57767	20.8%	35	15	2017-11-17	-	14
3.东方快车谋杀案	22354	8.1%	31	12	2017-11-10	-	21
4.寻梦环游记	15969	5.8%	34	19	2017-11-24	-	7
5.全球风暴	14839	5.3%	34	9	2017-10-27	-	30
6.降魔传	8725	3.1%	34	7	2017-11-17	-	14
7.追捕	8716	3.1%	32	8	2017-11-24	-	7
8.七十七天	8318	3.0%	33	11	2017-11-03	-	28
9.密战	7916	2.9%	31	8	2017-11-03	-	28
10.狂兽	6764	2.4%	35	6	2017-11-10	-	21
其他	52222	18.8%	32	7	-	-	-

准备数据

```
['雷神3：诸神黄昏','正义联盟','东方快车谋杀案','寻梦环游记','全球风暴', '降魔传','追捕','七十七天','密战','狂兽','其它']
[73853,57767,22354,15969,14839,8725,8716,8318,7916,6764,52222]
```

绘制柱状图

```
# 0.准备数据
# 电影名字
movie_name = ['雷神3：诸神黄昏','正义联盟','东方快车谋杀案','寻梦环游记','全球风暴','降魔传','追捕','七十七天','密战','狂兽','其它']
# 横坐标
x = range(len(movie_name))
# 票房数据
y = [73853,57767,22354,15969,14839,8725,8716,8318,7916,6764,52222]

# 1.创建画布
plt.figure(figsize=(20, 8), dpi=100)

# 2.绘制柱状图
plt.bar(x, y, width=0.5, color=['b','r','g','y','c','m','y','k','c','g','b'])

# 2.1b修改x轴的刻度显示
plt.xticks(x, movie_name)

# 2.2 添加网格显示
plt.grid(linestyle="--", alpha=0.5)

# 2.3 添加标题
plt.title("电影票房收入对比")
```

```
# 3.显示图像
plt.show()
```

> 参考链接：https://matplotlib.org/index.html

3、3 散点图

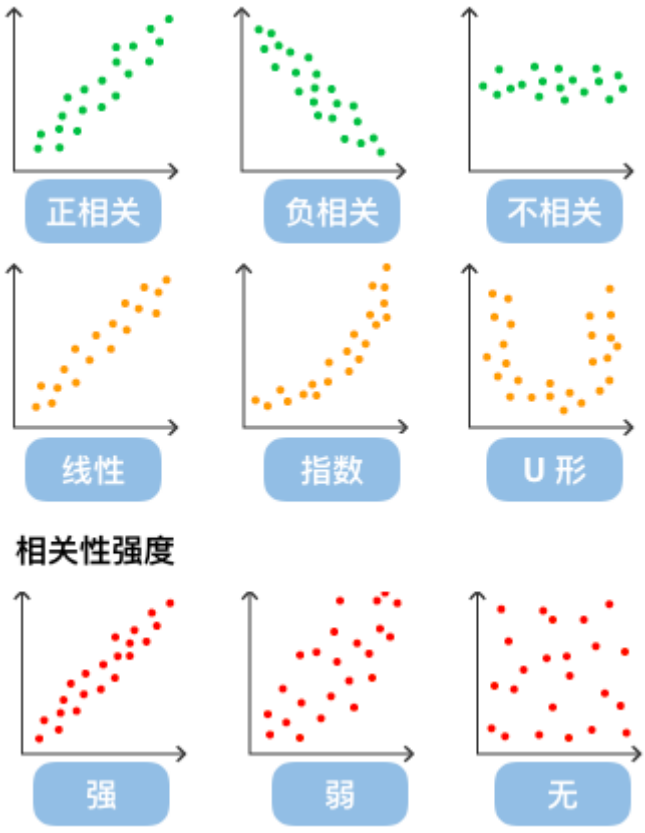
用两组数据构成多个坐标点，考察坐标点的分布,判断两变量之间是否存在某种关联或总结坐标点的分布模式。

特点：判断变量之间是否存在数量关联趋势,展示离群点(分布规律)

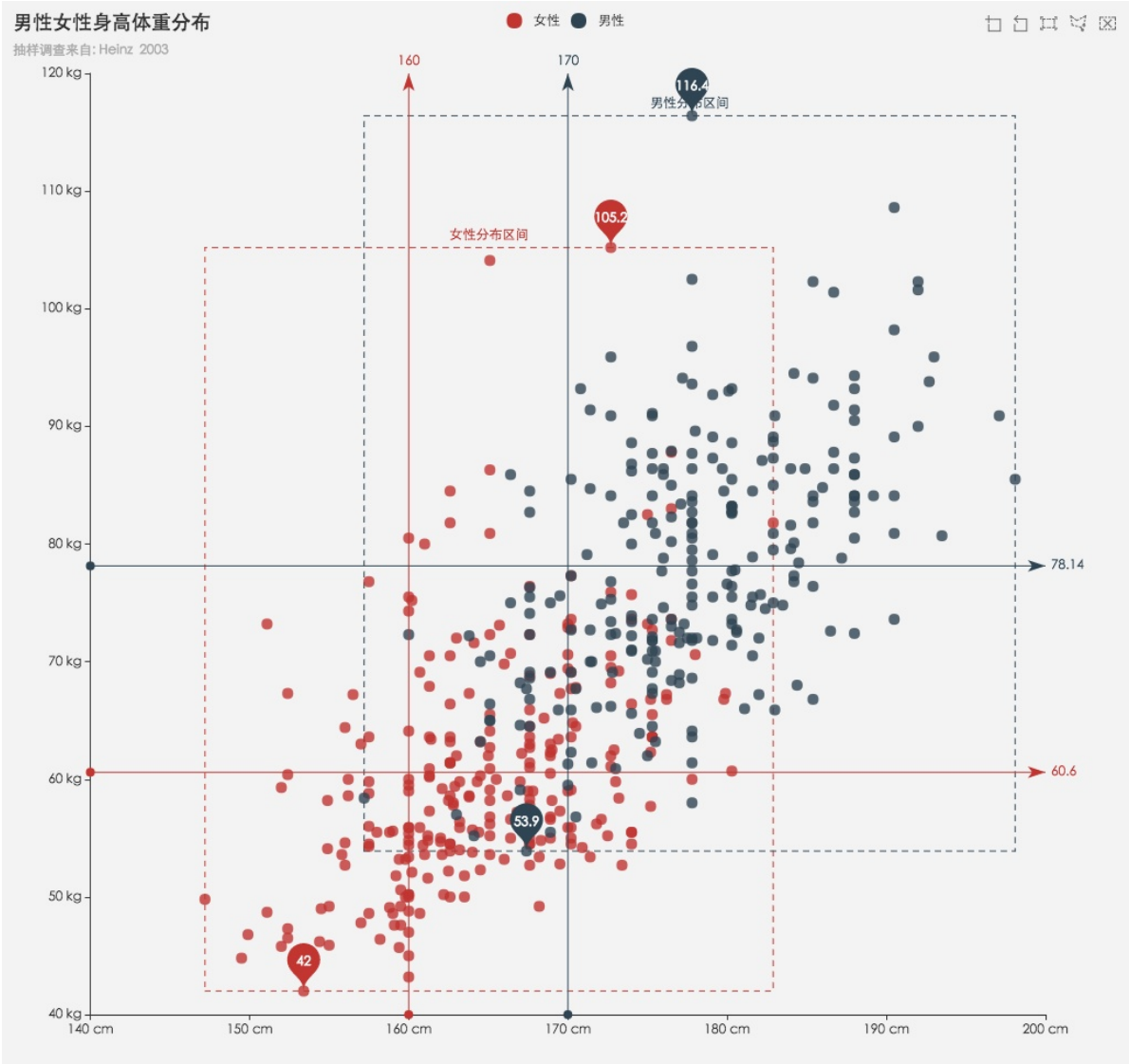
api：plt.scatter(x, y)

散点图也叫 X-Y 图，它将所有的数据以点的形式展现在直角坐标系上，以显示变量之间的相互影响程度，点的位置由变量的数值决定。

通过观察散点图上数据点的分布情况，我们可以推断出变量间的相关性。如果变量之间不存在相互关系，那么在散点图上就会表现为随机分布的离散的点，如果存在某种相关性，那么大部分的数据点就会相对密集并以某种趋势呈现。数据的相关关系主要分为：正相关（两个变量值同时增长）、负相关（一个变量值增加另一个变量值下降）、不相关、线性相关、指数相关等，表现在散点图上的大致分布如下图所示。那些离点集群较远的点我们称为离群点或者异常点。



示例图如下：



绘制散点图

散点图的绘制，使用的是`plt.scatter`方法，这个方法有以下参数：

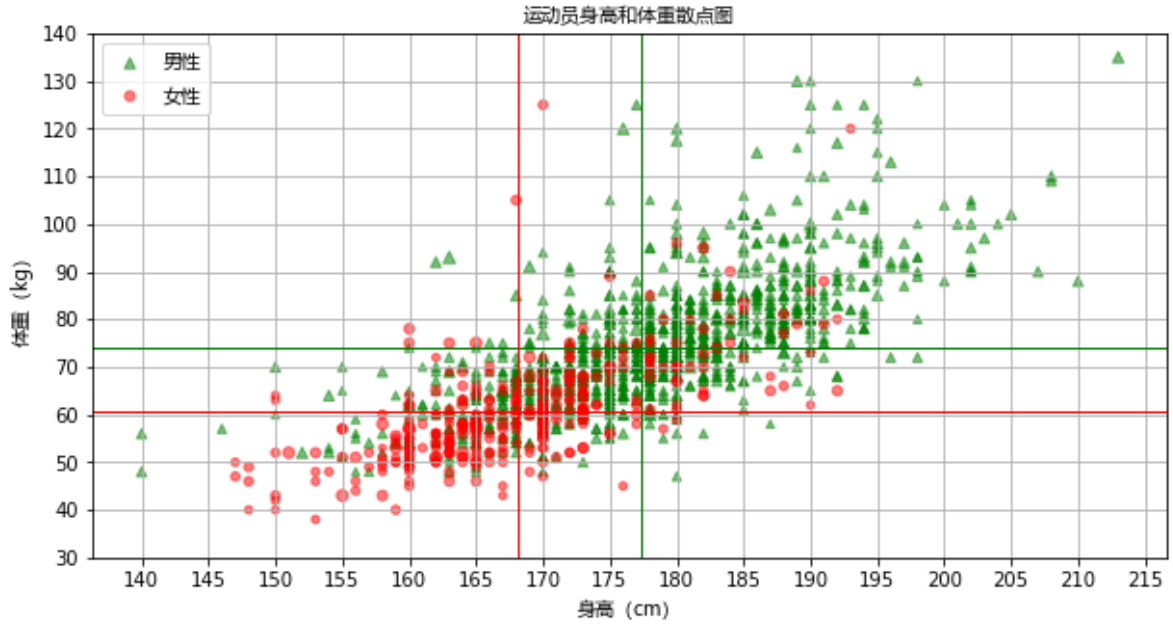
1. `x,y`：分别是x轴和y轴的数据集。两者的数据长度必须一致。
2. `s`：点的尺寸。如果是一个具体的数字，那么散点图的所有点都是一样大小，如果是一个序列，那么这个序列的长度应该和x轴数据量一致，序列中的每个元素代表每个点的尺寸。
3. `c`：点的颜色。可以为具体的颜色，也可以为一个序列或者是一个`cmap`对象。
4. `marker`：标记点，默认是圆点，也可以换成其他的。
5. 其他参数：`https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html#matplotlib.pyplot.scatter`。

比如有一组运动员身高和体重以及年龄的数据，那么可以通过以下代码来绘制散点图：

```
male_athletes = athletes[athletes['Sex'] == 'M']
female_athletes = athletes[athletes['Sex'] == 'F']
male_mean_height = male_athletes['Height'].mean()
female_mean_height = female_athletes['Height'].mean()
male_mean_weight = male_athletes['Weight'].mean()
female_mean_weight = female_athletes['Weight'].mean()

plt.figure(figsize=(10,5))
```

```
plt.scatter(male_athletes['Height'],male_athletes['Weight'],s=male_athletes['Age
'],marker='^',color='g',label='男性',alpha=0.5)
plt.scatter(female_athletes['Height'],female_athletes['Weight'],color='r',alpha=
0.5,s=female_athletes['Age'],label='女性')
plt.axvline(male_mean_height,color="g",linewidth=1)
plt.axhline(male_mean_weight,color="g",linewidth=1)
plt.axvline(female_mean_height,color="r",linewidth=1)
plt.axhline(female_mean_weight,color="r",linewidth=1)
plt.xticks(np.arange(140,220,5))
plt.yticks(np.arange(30,150,10))
plt.legend(prop=font)
plt.xlabel("身高（cm）",fontproperties=font)
plt.ylabel("体重（kg）",fontproperties=font)
plt.title("运动员身高和体重散点图",fontproperties=font)
plt.grid()
plt.show()
```

效果图如下：



绘制回归曲线

有一组数据后，我们可以对这组数据进行回归分析，回归分析可以帮助我们了解这组数据的大体走向。回归分析按照涉及的变量的多少，分为一元回归和多元回归分析；按照自变量的多少，可分为简单回归分析和多重回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。如果在回归分析中，只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量，且自变量之间存在线性相关，则称为多重线性回归分析。

自变量数量	是否线性	回归类型
1个	是	一元线性回归
多个	是	多元线性回归
1个	否	一元非线性回归
多个	否	多元非线性回归

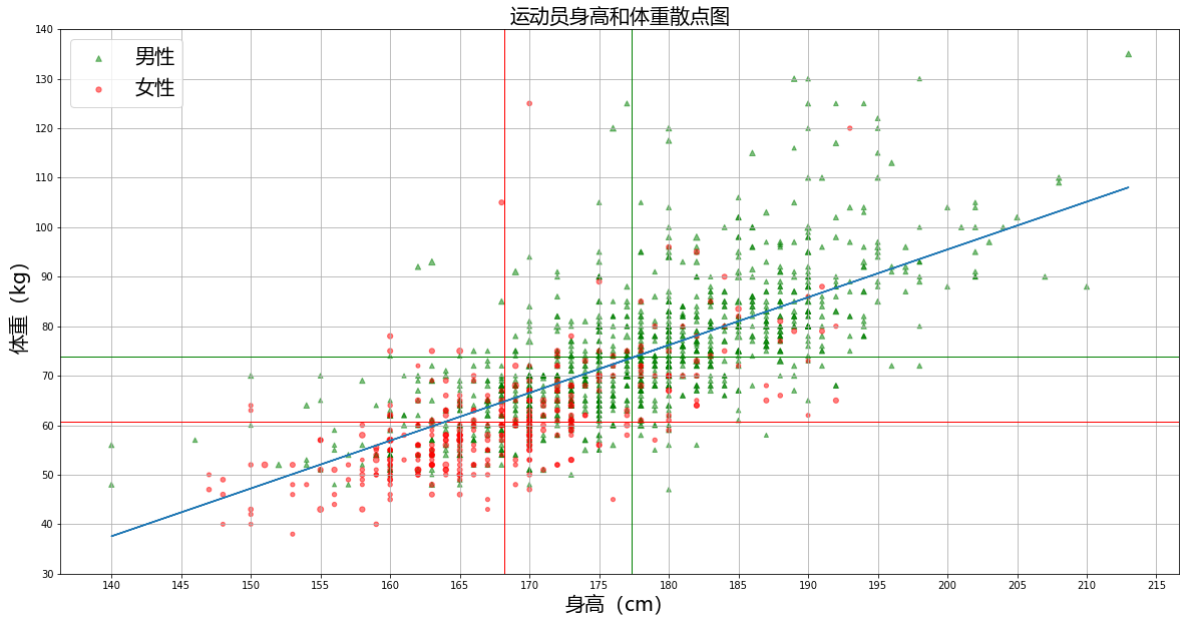
通过以上运动员散点图的分析，我们总体上可以看出来是满足线性回归的，因此可以在图上绘制一个线性回归的线条。想要绘制线性回归的线条，需要先按照之前的数据计算出线性方程，假如`x`是自变量，`y`是因变量，那么线性回归的方程可以用以下几个来表示：

```
y = 截距+斜率*x+误差
```

只要把这个方程计算出来了，那么后续我们就可以根据`x`的值，大概的估计出`y`的取值范围，也就是预测。如果我们针对以上运动员的身高和体重的关系，只要有身高，那么就可以大概的估计出体重的值。回归方程的绘制我们需要借助`scikit-learn`库，这个库是专门做机器学习用的，我们需要使用里面的线性回归类`sklearn.liear_regression.LinearRegression`。示例代码如下：

```
from sklearn.linear_model import LinearRegression
male_athletes = athletes[athletes['Sex'] == 'M'].dropna()
female_athletes = athletes[athletes['Sex'] == 'F'].dropna()
xtrain = male_athletes['Height']
ytrain = male_athletes['Weight']
# 生成线性回归对象
model = LinearRegression()
# 喂训练数据进去，但是需要把因变量转换成1列多行的数据
model.fit(xtrain[:,np.newaxis],ytrain)
# 打印斜率
print(model.coef_)
# 打印截距
print(model.intercept_)
line_xticks = xtrain
# 根据回归方程计算出的y轴坐标
line_yticks = model.predict(xtrain[:,np.newaxis])
```

效果图如下：



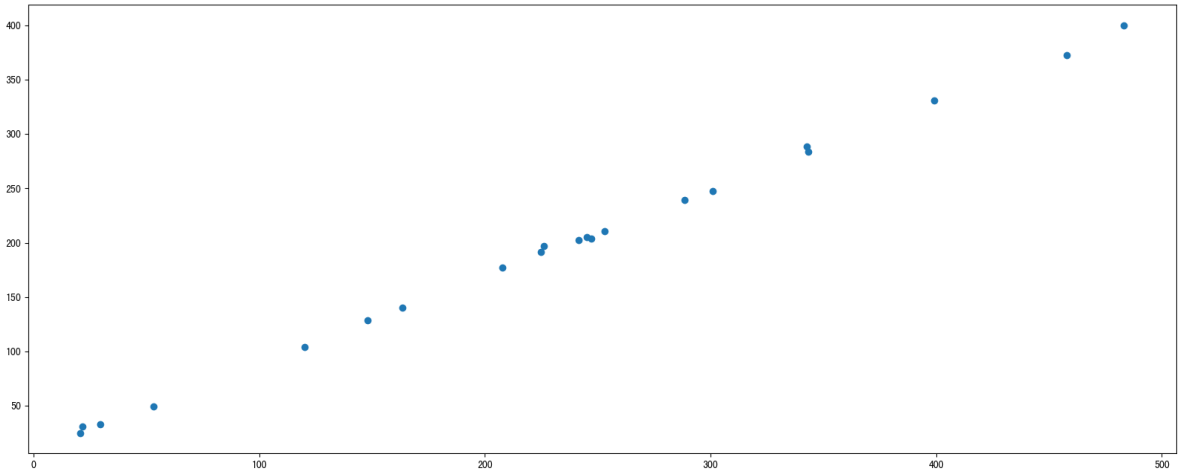
需求：探究房屋面积和房屋价格的关系

房屋面积数据：

```
x = [225.98, 247.07, 253.14, 457.85, 241.58, 301.01,  20.67, 288.64,163.56,
120.06, 207.83, 342.75, 147.9 ,  53.06, 224.72,  29.51,21.61, 483.21, 245.25,
399.25, 343.35]
```

房屋价格数据：

```
y = [196.63, 203.88, 210.75, 372.74, 202.41, 247.61,  24.9 , 239.34,
       140.32, 104.15, 176.84, 288.23, 128.79,  49.64, 191.74,  33.1 ,
        30.74, 400.02, 205.35, 330.64, 283.45]
```




```
# 0.准备数据
x = [225.98, 247.07, 253.14, 457.85, 241.58, 301.01,  20.67, 288.64,163.56,
120.06, 207.83, 342.75, 147.9 ,  53.06, 224.72,  29.51,21.61, 483.21, 245.25,
399.25, 343.35]
y = [196.63, 203.88, 210.75, 372.74, 202.41, 247.61,  24.9 , 239.34,140.32,
104.15, 176.84, 288.23, 128.79,  49.64, 191.74,  33.1 ,30.74, 400.02, 205.35,
330.64, 283.45]

# 1.创建画布
plt.figure(figsize=(20, 8), dpi=100)

# 2.绘制散点图
plt.scatter(x, y)

# 3.显示图像
plt.show()
```

3、4 饼图

用于表示不同分类的占比情况，通过弧度大小来对比各种分类。饼图可以看成数据的合计后的占比，适合突出表现份额。

特点：分类数据的占比情况(占比)

api：plt.pie(x, labels=,autopct=,colors)

饼图是一个划分为几个扇形的圆形统计图表，用于描述量、频率或百分比之间的相对关系的。 在`matplotlib`中，可以通过`plt.pie`来实现，其中的参数如下：

1. `x`：饼图的比例序列。
2. `labels`：饼图上每个分块的名称文字。
3. `explode`：设置某几个分块是否要分离饼图。
4. `autopct`：设置比例文字的展示方式。比如保留几个小数等。
5. `shadow`：是否显示阴影。
6. `textprops`：文本的属性（颜色，大小等）。
7. 其他参数：https://matplotlib.org/api/_as_gen/matplotlib.pyplot.pie.html#matplotlib.pyplot.pie

返回值：

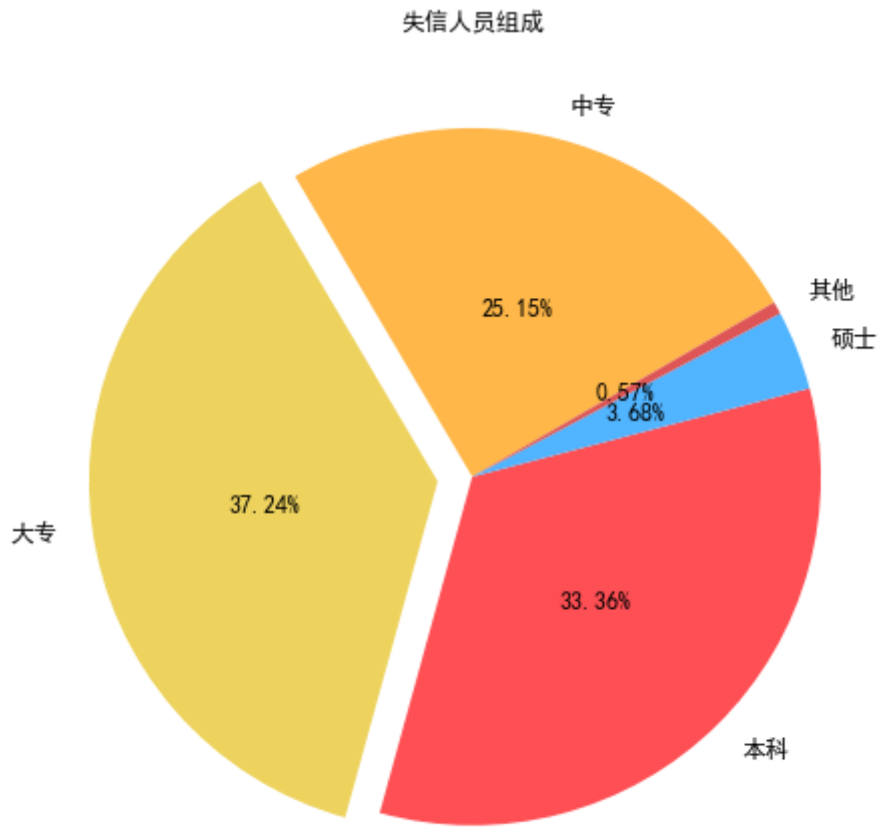
1. `patches`：饼图上每个分块的对象。
2. `texts`：分块的名字文本对象。
3. `autotexts`：分块的比例文字对象。

```
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']  # 指定默认字体
plt.rcParams['axes.unicode_minus'] = False

# 设置绘画的主题风格
plt.figure(figsize=(10, 8))
# 服务行业单位个数
# 数据来源
http://www.stats.gov.cn/tjsj/pcsj/jbdwpc/decjbdwpcsj/201612/t20161229_1447977.ht
ml

edu = [59104, 1467937, 3579974]
labels = ['第一产业', '第二产业', '第三产业']

explode = [0, 0, 0.1, ]  # 用于突出第三产业
colors = ['#FEB748', '#EDD25D', '#FE4F54']  # 自定义颜色

plt.axes(aspect='equal')  # 保证饼图是圆  不是默认的椭圆
plt.pie(x=edu,  # 数据
        labels=labels,  # 标签名称
        autopct='%.2f%%',  # 设置百分比格式  保留几位小数
        colors=colors,  # 使用自定义颜色
        labeldistance=1.1,  # 设置教育标签与圆心的距离
        # startangle=30,  # 设置饼图的初始角度  逆时针
        textprops={'fontsize': 12, 'color': 'k'},  # 设置文本标签的属性值
        explode=explode,  # 突出显示第三产业
        pctdistance=0.5,  # 占比和图距离
        )
plt.title('服务行业单位个数')
plt.show()
```

案例

假如现在我们有一组数据，用来记录各个操作系统的市场份额的。那么用饼状图表示如下：

```
# 设置绘画的主题风格
plt.figure(figsize=(10,8))
# 构造数据
edu = [0.2515,0.3724,0.3336,0.0368,0.0057]
labels = ['中专','大专','本科','硕士','其他']

explode = [0,0.1,0,0,0]  # 用于突出大专
colors=['#FEB748','#EDD25D','#FE4F54','#51B4FF','#dd5555'] # 自定义颜色

plt.axes(aspect='equal')  # 保证饼图是圆  不是默认的椭圆
plt.pie(x=edu,  # 数据
        labels=labels,   # 标签名称
        autopct='%.2f%%',  # 设置百分比格式  保留几位小数
        colors=colors,  # 使用自定义颜色
#         radius = 1,   # 设置饼图半径
#         center = (80,80),  # 设置圆点
        labeldistance = 1.1, # 设置教育水平标签与圆心的距离
        startangle =30, # 设置饼图的初始角度  逆时针
        textprops = {'fontsize':12, 'color':'k'}, # 设置文本标签的属性值
        explode=explode, # 突出显示大专人群
        pctdistance=0.5,   # 占比和图距离
#         shadow=True,    # 阴影
#         frame=True    # frame 显示
       )
plt.title('失信人员组成')
```



3、5 雷达图

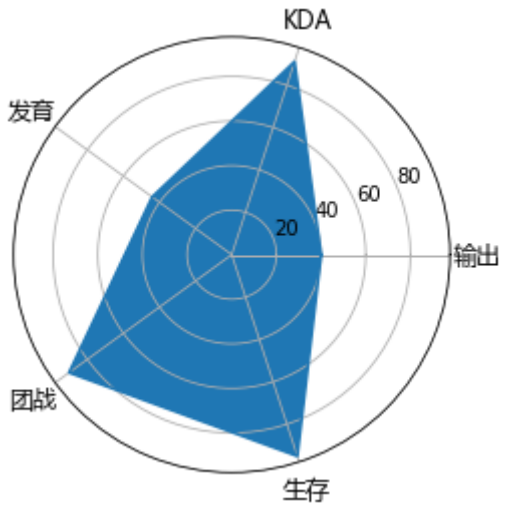
雷达图（Radar Chart）又被叫做蜘蛛网图，适用于显示三个或更多的维度的变量的强弱情况。比如英雄联盟中某个影响的属性（法术伤害，物理防御等），或者是某个企业在哪些业务方面的投入等，都可以用雷达图方便的表示。

3、5、1 plt.polar绘制雷达图

在`matplotlib.pyplot`中，可以通过`plt.polar`来绘制雷达图，这个方法的参数跟`plt.plot`非常的类似，只不过是`x`轴的坐标点应该为弧度（2*PI=360°）。示例代码如下：

```
properties = ['输出', 'KDA', '发育', '团战', '生存']
values = [40, 91, 44, 90, 95, 40]
theta = np.linspace(0, np.pi*2, 6)
plt.polar(theta, values)
plt.xticks(theta, properties, fontproperties=font)
plt.fill(theta, values)
plt.show()
```

效果图如下：



其中有几点需要注意：

1. 因为`polar`并不会完成线条的闭合绘制，所以我们在绘制的时候需要在`theta`中和`values`中在最后多重复添加第0个位置的值，然后在绘制的时候就可以和第1个点进行闭合了。
2. `polar`只是绘制线条，所以如果想要把里面进行颜色填充，那么需要调用`fill`函数来实现。
3. `polar`默认的圆圈的坐标是角度，如果我们想要改成文字显示，那么可以通过`xticks`来设置。

小结

- 折线图【知道】
  - 能够显示数据的变化趋势，反映事物的变化情况。(变化)
  - plt.plot()
- 散点图【知道】
  - 判断变量之间是否存在数量关联趋势,展示离群点(分布规律)

    - plt.scatter()
- 柱状图【知道】
    - 绘制连离散的数据,能够一眼看出各个数据的大小,比较数据之间的差别。(统计/对比)
    - plt.bar(x, width, align="center")
- 直方图【知道】
    - 绘制连续性的数据展示一组或者多组数据的分布状况(统计)
    - plt.hist(x, bins)
- 饼图【知道】
    - 用于表示不同分类的占比情况，通过弧度大小来对比各种分类
    - plt.pie(x, labels, autopct, colors)

附录：中文显示问题解决

解决方案一：

下载中文字体（黑体，看准系统版本）

- 步骤一：下载 SimHei 字体（或者其他的支持中文显示的字体也行）
- 步骤二：安装字体
    - linux下：拷贝字体到 usr/share/fonts 下：

      ```
      sudo cp ~/SimHei.ttf /usr/share/fonts/SimHei.ttf
      ```

    - windows和mac下：双击安装
- 步骤三：删除~/.matplotlib中的缓存文件

  ```
  cd ~/.matplotlib
  rm -r *
  ```

- 步骤四：修改配置文件matplotlibrc

  ```
  vi ~/.matplotlib/matplotlibrc
  ```

  将文件内容修改为：

  ```
  font.family         : sans-serif
  font.sans-serif     : SimHei
  axes.unicode_minus  : False
  ```

解决方案二：

在Python脚本中动态设置matplotlibrc,这样也可以避免由于更改配置文件而造成的麻烦，具体代码如下：

```
import matplotlib.pyplot as plt
# 设置显示中文字体
plt.rcParams['font.sans-serif'] = ['SimHei'] # 指定默认字体
```

有时候，字体更改后，会导致坐标轴中的部分字符无法正常显示，此时需要更改axes.unicode_minus参数：

```
# 设置正常显示符号
plt.rcParams['axes.unicode_minus'] = False
```