目录

**[1、 seaborn 库 2](#_Toc23145)**

**[2、 Pandas库 12](#_Toc14565)**

[创建 12](#_Toc14436)

[排序 12](#_Toc8573)

[索引 13](#_Toc17954)

[插入与删除 14](#_Toc13399)

**[3、 se 库 18](#_Toc8568)**

# seaborn 库

seaborn.relplot(x=None, y=None, hue=None, size=None,

style=None, data=None, row=None, col=None, col\_wrap=None,

row\_order=None, col\_order=None, palette=None, hue\_order=None,

hue\_norm=None, sizes=None, size\_order=None, size\_norm=None,

markers=None, dashes=None, style\_order=None, legend='brief',

kind='scatter', height=5, aspect=1, facet\_kws=None, \*\*kwargs)

参数：

x和y这个就是我们通常意义上的x和y即自变量和因变量。

hue表示的是色彩，可以理解为不同的值最终展现出来的颜色是不一样的

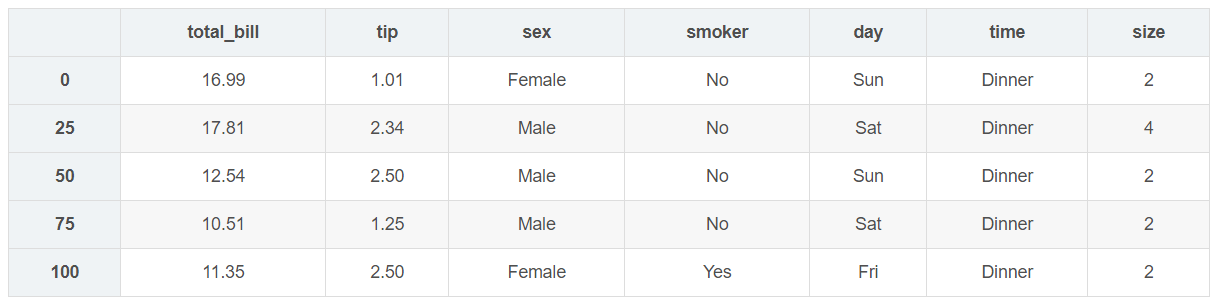
size 表示大小，把元素按照不同的大小来显示

style表示按照不同的风格来显示，比如加号和×等

data这个是要加载对应的数据，这个是必要

**Python Seaborn Tutorial For Beginners**

<https://www.datacamp.com/community/tutorials/seaborn-python-tutorial#load>



#首先先导入几个可能用到的库

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

下一步看一下接下来能够用到的几个函数的基本语法

seaborn.relplot(x=None, y=None, hue=None, size=None,

style=None, data=None, row=None, col=None, col\_wrap=None,

row\_order=None, col\_order=None, palette=None, hue\_order=None,

hue\_norm=None, sizes=None, size\_order=None, size\_norm=None,

markers=None, dashes=None, style\_order=None, legend='brief',

kind='scatter', height=5, aspect=1, facet\_kws=None, \*\*kwargs)¶

就以relplot这个函数来说，可以看到有很多参数，所以理论上来讲可以通过加入不同的参数然后改变图形的样子，得到不同的图形，这里我们以比较常见的几个来举几个例子，

x和y这个就是我们通常意义上的x和y即自变量和因变量。

hue表示的是色彩，可以理解为不同的值最终展现出来的颜色是不一样的

size 表示大小，把元素按照不同的大小来显示

style表示按照不同的风格来显示，比如加号和×等

data这个是要加载对应的数据，这个是必要的

其他的还有很多参数，就不一以介绍了，现在就按照刚刚提到的几个参数来看一下具体的图形是什么样的

这里我们就以seaborn自带的tips这个数据集为例

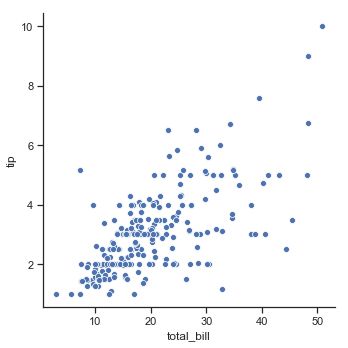
import seaborn as sns

sns.set(style='ticks')

tips = sns.load\_dataset('tips')

g = sns.relplot(x='total\_bill',y='tip',data=tips)

首先是不考虑hue这个参数的



是不是感觉杂乱无章，因为小数点全都堆叠在了一起

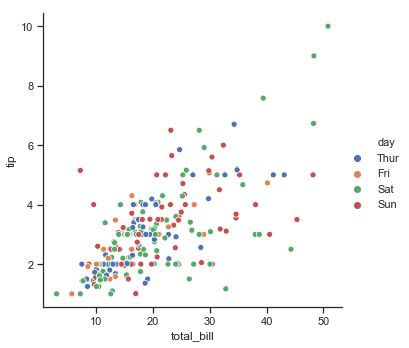
那么现在考虑加入hue这个参数

sns.set(style='ticks')

tips = sns.load\_dataset('tips')

g = sns.relplot(x='total\_bill',y='tip',hue='day',data=tips)

图形如下



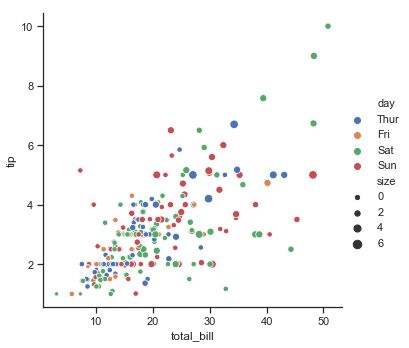
这样是不是就比第一个图形好多了？hue在这里就是起到了按照day的不同，显示不同颜色的作用

那么我们在在此基础上加上一个size这个参数，看看有什么效果

sns.set(style='ticks')

tips = sns.load\_dataset('tips')

g = sns.relplot(x='total\_bill',y='tip',hue='day',size='size',data=tips)



这个图相比于上面的图，就更加好读一点了，当然，size这里也可以是自定义的大小

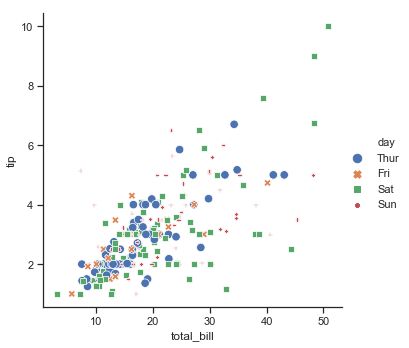
下面我们看一下style参数的使用

sns.set(style='ticks')

tips = sns.load\_dataset('tips')

g = sns.relplot(x='total\_bill',y='tip',hue='day',size='day',style='day',data=tips)

图形如下



能够看到圆点变成了多个风格，当然这个图形已经很难读出比较有效的东西了，所以对于可视化来说，不建议使用太多参数，一个好的图形应该是一目了然的，而不是花里胡哨

我们在来看一看matplotlib的基本操作

同样找到官方文档[https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py](https://link.zhihu.com/?target=https://matplotlib.org/tutorials/introductory/pyplot.html%23sphx-glr-tutorials-introductory-pyplot-py" \t "https://zhuanlan.zhihu.com/p/_blank)

由于matplotlib这个库包含大量的参数，我就不展开了

我们需要知道有以下常用的功能

plt.figure()这个是当创建的时候图形是在图中的时候使用的

颜色，标记和线的类型

使用例如

a.plot(x,y,linestyle='--',color='g')的方式来定义

而对于绘图范围，刻度位置以及刻度标签，则可以使用xlim,ylim,xticks,yticks,xticktable,yticktable等

学习了上面的操作以后，我们开始绘制常见的可视化图形

散点图

散点图在英文中被称为scatter plot，讲两个变量值放在二维坐标中，能够很好的展现出两个变量之间的关系

#准备数据

N = 100

x = np.random.randn(N)

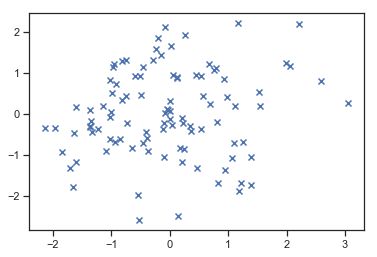
y = np.random.randn(N)

#用Matplotlib绘制散点图

plt.scatter(x,y,marker='x')

plt.show()

图形如下



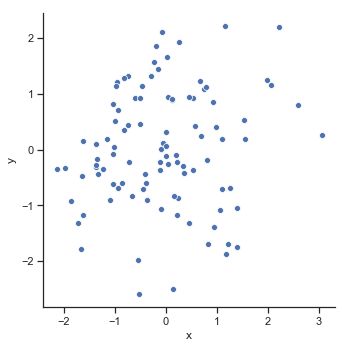
使用seaborn绘图

df = pd.DataFrame({'x':x,'y':y})

sns.relplot(x='x',y='y',data=df,kind='scatter')

plt.show()

图形如下



折线图

折线图可以用来表示数据随着时间变化的趋势

在Matplotlib中，我们可以使用plt.plot()函数，当然在绘图之前，我们要先把数据按照x轴的大小进行排序，要不画出来的折线图无法按照x轴递增的顺序显示

而在seaborn中，我们可以直接使用sns.lineplot(x,y,data=None)函数

x = [2001,2002,2003,2004,2005,2006,2007,2008,2009,2010]

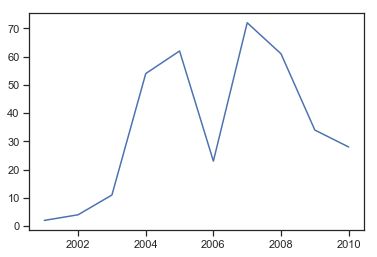
y = [2,4,11,54,62,23,72,61,34,28]

#使用Matplotlib绘制折线图

plt.plot(x,y)

plt.show()

图形如下

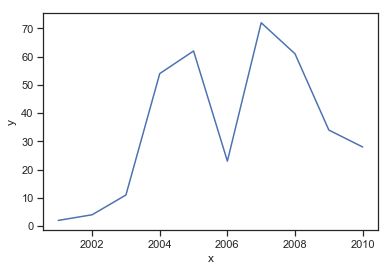


df = pd.DataFrame({'x':x,'y':y})

sns.lineplot(x='x',y='y',data=df)

plt.show()

使用seaborn绘制，图形如下



直方图

直方图是我们比较常见的图，把坐标轴为一定数量的小区间，这些小区间内用矩形条展示该区间的值

在Matplotlib中，我们使用plt.hist(x,bins=10)函数，其中x代表的是一维数组

bins是直方图中区间数量，默认是10

在seaborn中，我们使用sns.distplot(x,bins=10,kde=True)函数，x和bins的含义Matplotlib的含义基本一致，而kde代表的是核密度估计，默认为True,我们也可以通过设置为False的方法，核密度估计是通过核函数来帮我们估计概率密度的方法

#准备数据

a = np.random.randn(100)

s = pd.Series(a)

#用Matplotlib绘制直方图

plt.hist(s)

plt.show()

#用Seaborn绘制直方图

#kde设置为False

sns.distplot(s,kde=False)

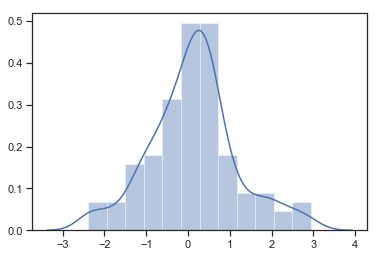
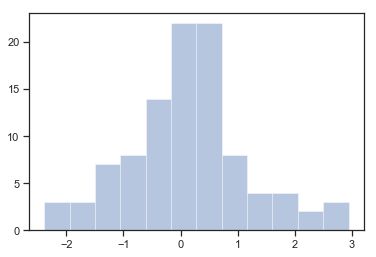
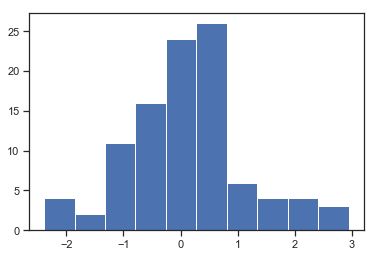
plt.show()

#kde设置为True

sns.distplot(s,kde=True)

plt.show()

三幅图分别如下



可以看到前面两幅图没有区别，而最后一幅图是kde默认为True的情形，这个图加了一条集中趋势线

条形图

直方图表示变量数值分布，而条形图能够帮助我们查看类别的特征，在条形图中，长条形的长度表示的是频数，宽度表示类别

在Matplotlib中，使用plt.bar(x,height)函数，其中x代表x轴的位置序列，heighti表示的是y轴的数值，也即柱子的高度

在seaborn中，我们可以使用sns.barplot(x=None,y=None,data=None)函数

#准备数据

x = ['col1','col2','col3','col4','col5']

y = [6,4,7,11,9]

#用Matplotlib绘制条形图

plt.bar(x,y)

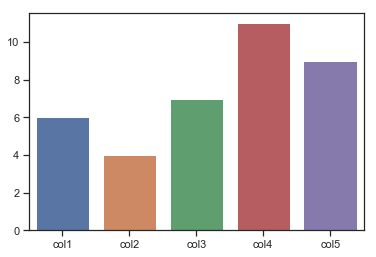
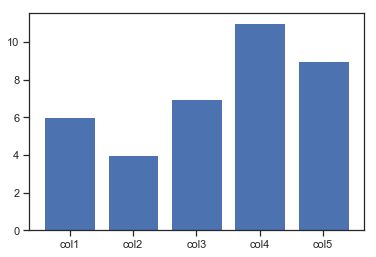
plt.show()

#用seaborn绘制条形图

sns.barplot(x,y)

plt.show()

图形结果如下



可以看到seaborn自动的给不同类别的条形图加了颜色，看起来就比较方便了

箱线图

箱线图常常和分位数结合在一起，可以根据箱线图的图形判断数据的差异性，离散程度，以及异常值

在Matplotlib中，使用plt.boxplot(x.labels=None)函数，x代表要绘制的箱线图数据，labels代表缺省值，可以为箱线图加上标签

在seaborn'中，使用sns.boxplot(x=None,y=None,data=None)函数

# 准备数据

# 0-1 之间的 生成10\*4 维度数据

data=np.random.normal(size=(10,4))

lables = ['A','B','C','D']

# 用 Matplotlib 绘制箱线图

plt.boxplot(data,labels=lables)

plt.show()

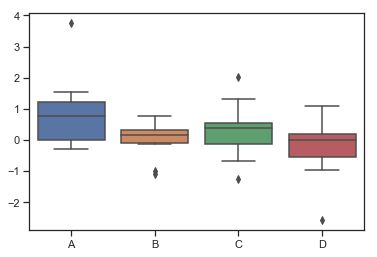
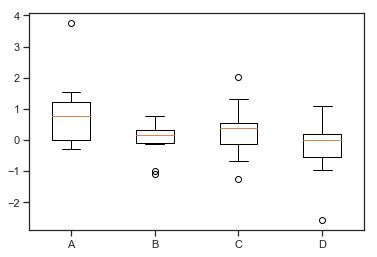
# 用 Seaborn 绘制箱线图

df = pd.DataFrame(data, columns=lables)

sns.boxplot(data=df)

plt.show()

图形如下



饼图

饼图是我们常见的一种图，能够显示每个模块的占比，但是如果占比相差不大时可能会不明显

这里就直接使用Matplotlib，plt.pie(x,labels=None)

#准备数据

nums = [20, 36, 40, 37, 6]

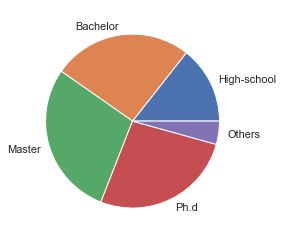
labels = ['High-school','Bachelor','Master','Ph.d', 'Others']

# 用 Matplotlib 绘制饼图

plt.pie(x = nums, labels=labels)

plt.show()

图形如下



热力图，

热力图英文名叫heat map，是矩阵的一种表示方法，矩阵的元素值用不同的颜色来带代表，不同颜色代表大小不同的值，通过颜色的深浅可以直观了解每个位置数值的大小，这里推荐seaborn官方文档[https://seaborn.pydata.org/generated/seaborn.heatmap.html#seaborn.heatmap](https://link.zhihu.com/?target=https://seaborn.pydata.org/generated/seaborn.heatmap.html%23seaborn.heatmap" \t "https://zhuanlan.zhihu.com/p/_blank),这里面有详细的不同参数的操作，很有意思，感兴趣的可以看一下

我们此处就直接使用seaborn中的sns.heatmap(data)函数，采用seaborn中自带的数据集flights，该数据集记录了1949-1960年间每个航班乘客的数量

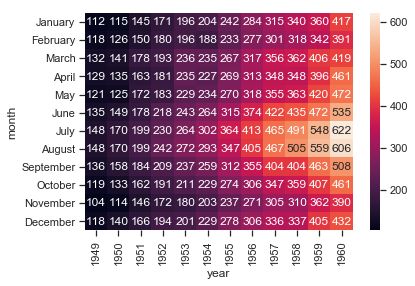
flights = sns.load\_dataset('flights')

flights = flights.pivot('month','year','passengers')

ax = sns.heatmap(flights,annot=True,fmt='d')

这里我使用直接显示数值的方式

看图



# Pandas库

https://www.pypandas.cn/docs/user\_guide/

这是一个非常好用的库，总结一下 Pandas DataFrame 常见用法，用来复制粘贴。

import numpy as np

import pandas as pd

## 创建

data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

c = ['a', 'b', 'c']

r = ['A', 'B', 'C']

df = pd.DataFrame(data=data, columns=c, index=r)

## 排序

### 按列、行名排序

# 行名排序 降序

df.sort\_index(axis=0, ascending=False)# 列名排序 降序

df.sort\_index(axis=0, ascending=False)

### 按值排序

****拿出来排序****

df["a"].sort\_values(ascending = False)

****df 内排序****

df.sort\_values( ['a', 'b', 'c'])

## 索引

虽然标准的 Python/Numpy 的选择和设置表达式都能够直接派上用场，但是作为工 程使用的代码，我们推荐使用经过优化的 pandas 数据访问方式： .at , .iat , .loc , .iloc 和 .ix 。

### 位置索引

df.iloc[2] # 选择第二行所有数据, 是Series类型

df.iloc[[2]] # 选择第二行所有数据, 是DataFrame类型

df.iloc[:, 2] # 选择第二列所有数据, 是Series类型

df.iloc[:, [2]] # 选择第二列所有数据, 是DataFrame类型

df.iloc[:, 0:2] # 选择0到2列所有数据

df.iloc[[2,3], 0:2] # 选择2和3行, 0到2列所有数据

df.iat[1, 1] # 根据位置快速取出数据, 获取单个数据推荐这种方法

### 自定义索引

df.loc['top'] # 选择指定行数据, 是Series类型

df.loc[['top']] # 选择指定行数据, 是DataFrame类型

df.loc[:, 'xm'] # 选择指定列数据, 是Series类型(不推荐)

df.loc[:, ['xm']] # 选择指定列数据, 是DataFrame类型(不推荐)

df.loc[:, ['bj','xm']] # 选择多列数据(不推荐)

df.loc[:, 'bj':'xb'] # 选择多列之间所有数据, 列切片只能用这种方法

df.loc[['top','count'], 'bj':'xb'] # 选择指定行, 指定列数据

df.at['top', 'xm'] # 根据自定义索引快速取出数据, 获取单个数据推荐这种方法

### 布尔索引

# 选取所有出生日期大于等于1998年的数据, 这里是字符串比较

df[df['csrq']>='1998'] # 选取所有出生日期大于等于1997年小于1999年的数据

df[(df['csrq']>='1997')&(data['csrq']<'1999')]# 选取所有出生日期大于等于1997年小于1999年的数据

df[df['csrq'].between('1997', '1999')]# 选取所有出生日期大于等于1997年或者姓名为张三的数据

df[(df['csrq']>='1997')|(data['xm']=='张三')]# 另一种选取方式(不推荐, 实测效率比上面低)

df[df.csrq>='1998'] # 选择字段值为指定内容的数据

df[df['xm'].isin(['张三','李四'])]

## 插入与删除

# 假设cj列本来不存在, 这样会在列尾添加新的一列cj, 值为s(Series对象), 原地

df['cj'] = s # 在第1列位置插入一列dz(地址), 值为s, 原地

df.insert(0, 'dz', s)# 在df中添加内容为df2(必须是DataFrame对象)的新列(添加列), 非原地

df.join(df2)

# 将df2中的行添加到df的尾部(添加行), 非原地

df.append(df2)# 删除单列, 并返回删除的列, 原地

df.pop('xm')# 删除指定行, 非原地

df.drop(1) # 删除指定列, axis=1指第2维, axis默认0, 非原地

df.drop(['xm', 'xh'], axis=1)

### DataFrame 重要方法与属性

'''重要属性'''

df.values # 查看所有元素的value

df.dtypes # 查看所有元素的类型

df.index # 查看所有行名

df.index = ['总数', '不同', '最多', '频率'] # 重命名行名

df.columns # 查看所有列名

df.columns = ['班级', '姓名', '性别', '出生日期'] # 重命名列名

df.T # 转置后的df, 非原地

'''查看数据'''

df.head(n) # 查看df前n条数据, 默认5条

df.tail(n) # 查看df后n条数据, 默认5条

df.shape() # 查看行数和列数

df.info() # 查看索引, 数据类型和内存信息

'''数据统计'''

df.describe() # 查看数据值列的汇总统计, 是DataFrame类型

df.count() # 返回每一列中的非空值的个数

df.sum() # 返回每一列的和, 无法计算返回空, 下同

df.sum(numeric\_only=True) # numeric\_only=True代表只计算数字型元素, 下同

df.max() # 返回每一列的最大值

df.min() # 返回每一列的最小值

df.argmax() # 返回最大值所在的自动索引位置

df.argmin() # 返回最小值所在的自动索引位置

df.idxmax() # 返回最大值所在的自定义索引位置

df.idxmin() # 返回最小值所在的自定义索引位置

df.mean() # 返回每一列的均值

df.median() # 返回每一列的中位数

df.var() # 返回每一列的方差

df.std() # 返回每一列的标准差

df.isnull() # 检查df中空值, NaN为True, 否则False, 返回一个布尔数组

df.notnull() # 检查df中空值, 非NaN为True, 否则False, 返回一个布尔数组

### 转换成 Numpy

df.values

np.array(df)

作者：madao756  
链接：https://www.jianshu.com/p/6e35d37e7709  
来源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

Series 一维标记数组

##pd.series(data, index, dtype, copy)

##data: 数据； index: 索引；

data = np.array(['a','b','c','d'])

s = pd.Series(data, index=[1,2,3,4])

print(s)

----->

当前目录：D:\Codes\数据分析

1 a

2 b

3 c

4 d

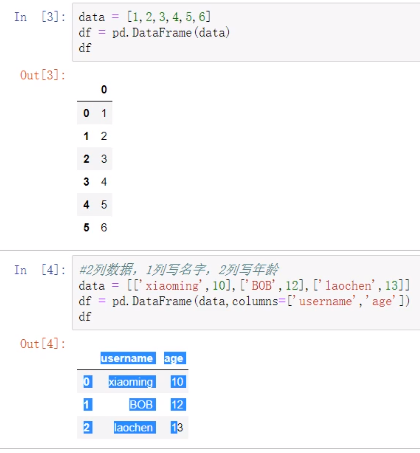
dtype: object

>>>

使用字典及标量来创建



数据帧：



##数据帧 pd.DataFrame(Data,index,columns,dtype)

df = pd.DataFrame(data)

print(df)

## 字典创建帧

dict1 = {

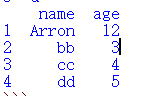
'name':["Arron","bb","cc","dd"],

"age":[12,3,4,5]

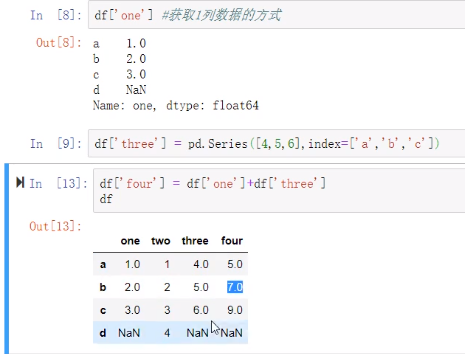
}

df1 = pd.DataFrame(dict1, index=[1,2,3,4])

print(df1)



https://blog.csdn.net/ahmcwt/article/details/104646246/?utm\_medium=distribute.pc\_relevant.none-task-blog-title-6&spm=1001.2101.3001.4242



通过标签选择数据：

>>> df.sort\_index(axis=0, ascending=True)

A B C D

2020-12-09 1.948187 0.196630 -0.543099 1.201738

2020-12-10 0.263322 -0.653471 0.153266 -0.822625

2020-12-11 0.234130 -0.169499 -0.763881 0.761132

2020-12-12 -0.501687 -0.741316 -0.320843 1.722590

2020-12-13 1.268059 -0.370238 0.223310 1.104557

2020-12-14 0.993159 0.134015 0.355654 0.277222

>>> df.sort\_values("B")

A B C D

2020-12-12 -0.501687 -0.741316 -0.320843 1.722590

2020-12-10 0.263322 -0.653471 0.153266 -0.822625

2020-12-13 1.268059 -0.370238 0.223310 1.104557

2020-12-11 0.234130 -0.169499 -0.763881 0.761132

2020-12-14 0.993159 0.134015 0.355654 0.277222

2020-12-09 1.948187 0.196630 -0.543099 1.201738

>>> df['B'] #通过列名进行选择

2020-12-09 0.196630

2020-12-10 -0.653471

2020-12-11 -0.169499

2020-12-12 -0.741316

2020-12-13 -0.370238

2020-12-14 0.134015

Freq: D, Name: B, dtype: float64

>>> df[1:3] #选择部分，进行切片，选择第1~3行

A B C D

2020-12-10 0.263322 -0.653471 0.153266 -0.822625

2020-12-11 0.234130 -0.169499 -0.763881 0.761132

>>> df["20201210":"20201213"]

A B C D

2020-12-10 0.263322 -0.653471 0.153266 -0.822625

2020-12-11 0.234130 -0.169499 -0.763881 0.761132

2020-12-12 -0.501687 -0.741316 -0.320843 1.722590

2020-12-13 1.268059 -0.370238 0.223310 1.104557

>>> df.loc[dates[1]] #通过标签选择

A 0.263322

B -0.653471

C 0.153266

D -0.822625

Name: 2020-12-10 00:00:00, dtype: float64

>>> df.loc[dates[2],'B'] #获取一个量，第2行，B列

-0.16949935349762021

>>> df.loc['20201209','A']

1.9481868597691128

>>> df.at['20201210','B']

-0.6534714066384203

>>> df.iloc[1:4,0:3] #通过数值进行切片，数值即为第“几”行或列

A B C

2020-12-10 0.263322 -0.653471 0.153266

2020-12-11 0.234130 -0.169499 -0.763881

2020-12-12 -0.501687 -0.741316 -0.320843

>>>

>>> df.iloc[[1,2,5],[0,2]] #通过数值进行切片

A C

2020-12-10 0.263322 0.153266

2020-12-11 0.234130 -0.763881

2020-12-14 0.993159 0.355654

>>> pd.Series(np.random.randint(0,8,size=10)) #准备画图

0 4

1 6

2 6

3 4

4 1

5 7

6 5

7 1

8 3

9 0

dtype: int32

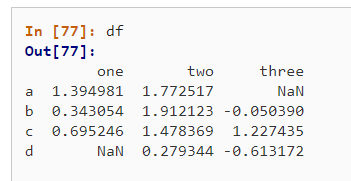
>>>

绘图相关：

https://pandas.pydata.org/pandas-docs/stable/user\_guide/visualization.html

函数库

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.cumsum.html#pandas.DataFrame.cumsum



**https://www.cnblogs.com/daysn/p/10833729.html**

# [Python通过pandas操作excel常用功能](https://www.cnblogs.com/xhuangtao/p/11963279.html)

<https://www.cnblogs.com/xhuangtao/p/11963279.html>

## **在写入表格时候给自己挖了个坑，图省事直接用to\_excel（fileuri，sheetname）写入，怎么写怎么覆盖  。后来隐约感觉到是自动关闭writer的锅，改成writer读写后就好了。**

### **writer = pd.ExcelWriter(filepath) companydf.to\_excel(excel\_writer=writer,sheet\_name='公司维度表') goodsdf.to\_excel(excel\_writer=writer, sheet\_name='货物维度表') writer.save() writer.close()**

# se 库