DataFrame.plot.XXX() 默认都是按列画

|  |  |  |  |
| --- | --- | --- | --- |
| bar() | 条形图 | barh | 横向条形图 |
| hist() | 直方图 | hist(orieatation=’horiaontal’) | 横向直方图 |
| box() | 箱型图 | box(vert=False) | 横向箱型图 |
| area() | 区域图 | scatter(X,Y,c,s) | 散点图 |
| factorplot | 柱状图 | kdeplot() | 核估计图 |
| distplot | 单变量分布图 |  |  |
| jointplot | 双变量分布图 |  |  |
| pairplot() | 多对变量之间的关系分布图 |  |  |

matplotlib

import matplotlib.pyplot as plt

matplotlib.style.use(‘ggplot’) #使用ggplot样式

matplotlib输入为一个DataFrame（n\*m），默认按列绘图(将画出m条线)。Series也行

df = pd.DataFrame(np.random.randn(100,4),   
index=pd.date\_range(‘1/1/2000’,periods=1000),  
columns=list(‘ABCD’)) # 也可以写成columns=[‘A’,’B’,’C’,’D’]

df = df.cumsum()

plt.figure()

df.plot() # 默认画折线图

plt.show()

条形图

df.plot(kind=’bar’) # bar条形图

或df.plot.bar()

df.plot.barh() #横向条形图

plt.axhline(0,color=’k’) # 在y=0的位置画一条黑色的线

df.plot.bar(stacked=True) # 叠加条形图

df.plot.barh(stacked=True) # 横向叠加条形图

直方图

plt.figure()

df.plot.hist() # 直方图，默认有不堆叠，但会重叠

df.plot.hist(orieatation=’horiaontal’,cumulative=True) # 横向累积直方图

df.plot.hist(alpha=0.2,color=’g’) # alpha为透明度20%

df.plot.hist(stacked=Ture，bins=20) # 使用堆叠，就不会重叠。bins=20指分为20个区段

箱型图

df.plot.box()

Color=dict(boxes=’DarkGreen’,whiskers=’DrakOrange’,medians=’DarkBlue’,caps=’Gray’) # 设置箱型图颜色

df.plot.box(color=Color)

df.plot.box(vert=False,positions=[1,4,8]) # vert=False水平箱型图，positions指定绘制的地方

区域图

df.plot.area()

df.plot.area(stacked=0.2)

散点图

df.plot.scatter(x=Series1,y=Series2) # 画散点图需要xy

同一画布画多个散点图：

ax = df.plot.scatter(x= Series1,y=Series2,color=’DrakBlue’,label=’Group1’)

df.plot.scatter(x=Series3,y=Series4,color=’RED’,label=’Group2’,ax=ax)

df.plot.scatter(x=Series1,y=Series2,c=Series3,s=Series4) # 参数c为每个点提供颜色，参数s为每个点指定气泡大小

柱状图

years = np.arange(2000,2015)

g = sns.factorplot(data=DataFrame)

g.set\_xticklabels(step=2)

# 分布数据集可视化

## 单变量分布

x=np.random.normal(size=100)

sns.distplot(x) 默认情况下，这将绘制一个直方图，并拟合出核密度估计(KDE)。

sns.distplot(x, bins=20, kde=True,hist=True, rug=True);

x: 数据

bins :柱子个数

kde ：是否画曲线（该曲线名为核估计，英文名为kde）

hist:是否画柱子

rug :是否画地毯图

地毯图：直方图通过在数据的范围内切成数据片段，然后绘制每个数据片段中的观察次数，来表示整体数据的分布。为了说明这一点，添加地毯图，每个观察点绘制一个小的垂直刻度。

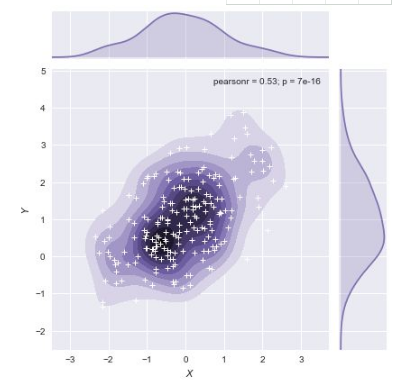
jointplot()函数使用JointGrid来管理。为了获得更多的灵活性，您可能需要直接使用JointGrid绘制图形。jointplot()在绘制后返回JointGrid对象，您可以使用它来添加更多图层或调整可视化的其他方面：

g = sns.jointplot(x="x", y="y", data=df, kind="kde", color="m")

g.plot\_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")

g.ax\_joint.collections[0].set\_alpha(0)

g.set\_axis\_labels("$X$", "$Y$");



## 双变量分布

jointplot() 函数中创建一个多面板数字，显示两个变量之间的双变量（或联合）关系以及每个变量的单变量（或边际）分布和轴。

默认样式为散点图 即kind=’scatter’

可修改为 ‘hex’ (hexbin图) ‘kde’(核估计) ‘reg’(线性回归拟合)

## 呈现数据集中成对的关系

pairplot()函数。这将创建一个轴的矩阵，并显示DataFrame中每对列的关系。默认情况下，它也绘制每个变量在对角轴上的单变量：

iris = sns.load\_dataset("iris")

sns.pairplot(iris)

# 线性关系的可视化

函数regplot()和lmplot()是密切相关的，并且共享了大部分的核心功能。

在最简单的调用中，两个函数绘制了两个变量x和y的散点图，然后拟合回归模型y〜x并绘制了该回归线的结果回归线和95％置信区间：

sns.regplot(x="total\_bill", y="tip", data=DATA);

等价于

sns.lmplot(x="total\_bill", y="tip", data= DATA);

为离散值添加一些随机噪声的“抖动”(jitter)，使得这些值的分布更加明晰。

值得注意的是，抖动仅适用于散点图数据，且不会影响拟合的回归线本身

sns.lmplot(x="size", y="tip", data= DATA, x\_jitter=.05)

在每个独立的数据分组中对观察结果进行折叠，以绘制中心趋势的估计以及置信区间

sns.lmplot(x="size", y="tip", data= DATA, x\_estimator=np.mean)

其他参数：

点的形状：markers=["o", "x"]

使用颜色区分（自动的）hue="smoker"

# 分类数据的绘制

eaborn.factorplot(x=None, y=None, hue=None, data=None, row=None, col=None, col\_wrap=None, estimator=<function mean>, ci=95, n\_boot=1000, units=None, order=None, hue\_order=None, row\_order=None, col\_order=None, kind='point', size=4, aspect=1, orient=None, color=None, palette=None, legend=True, legend\_out=True, sharex=True, sharey=True, margin\_titles=False, facet\_kws=None, \*\*kwargs)

Parameters：

* x,y,hue 数据集变量 变量名
* date 数据集 数据集名
* row,col 更多分类变量进行平铺显示 变量名
* col\_wrap 每行的最高平铺数 整数
* estimator 在每个分类中进行矢量到标量的映射 矢量
* ci 置信区间 浮点数或None
* n\_boot 计算置信区间时使用的引导迭代次数 整数
* units 采样单元的标识符，用于执行多级引导和重复测量设计 数据变量或向量数据
* order, hue\_order 对应排序列表 字符串列表
* row\_order, col\_order 对应排序列表 字符串列表
* kind : 可选：point 默认, bar 柱形图, count 频次, box 箱体, violin 提琴, strip 散点，swarm 分散点（具体图形参考文章前部的分类介绍）
* size 每个面的高度（英寸） 标量
* aspect 纵横比 标量
* orient 方向 "v"/"h"
* color 颜色 matplotlib颜色
* palette 调色板 seaborn颜色色板或字典
* legend hue的信息面板 True/False
* legend\_out 是否扩展图形，并将信息框绘制在中心右边 True/False
* share{x,y} 共享轴线 True/False
* facet\_kws [FacetGrid](https://link.zhihu.com/?target=http%3A//seaborn.pydata.org/generated/seaborn.FacetGrid.html%23seaborn.FacetGrid)的其他参数 字典

seabron基于matplotlib，在其基础上渲染

导入模块：

import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt

sns.set\_style('darkgrid') # 设置背景色，使用这个就行了。浅灰色背景

plt.show() #显示图像

Seaborn将matplotlib参数分成两个独立的组。第一组设定了**美学风格**，第二组则是不同的**度量元素**，这样就可以很容易地添加到代码当中了操作这些参数的接口是两对函数。为了**控制样式**，使用axes\_style()和set\_style()函数。为了**扩展绘图**，请使用plotting\_context()和set\_context()函数。在这两种情况下，第一个函数返回一个参数字典，第二个函数则设置matplotlib默认属性。

**样式控制：axes\_style() and set\_style()**

有5个seaborn的主题，适用于不同的应用和人群偏好：

sns.set\_style('darkgrid')

darkgrid 黑色网格（默认）

whitegrid 白色网格

dark 黑色背景

white 白色背景

ticks 应该是**四周都有刻度线**的白背景？

## 用despine()进行边框控制

* 可以删除上方和右方坐标轴上不需要的边框

sns.despine() # 默认无参数状态，就是删除上方和右方的边框

* 一些图的边框可以通过数据移位，当然调用despine()也能做同样的事。当边框没有覆盖整个数据轴的范围的时候，trim参数会限制留存的边框范围。

sns.despine(offset=10, trim=True) # offset 两坐标轴离开距离；

* 你也可以通过往despine()中添加参数去控制边框

sns.despine(left=True) # 删除左边边框

* 从plot()函数中移除顶部或右边的边框

despine(fig=None, ax=None, top=True, right=True, left=False, bottom=False, offset=None, trim=False)

## seaborn样式中最重要的元素

如果您想要定制seanborn的样式，可以将参数字典传递给axes\_style()和set\_style()的rc参数。注意，只能通过该方法覆盖样式定义的一部分参数。(然而，更高层次的set()函数接受任何matplotlib参数的字典)。

如果您想要查看包含哪些参数，您可以只调用该函数而不带参数，这将返回当前设置的字典:

sns.axes\_style()

{'axes.axisbelow': True,

'axes.edgecolor': 'white',

'axes.facecolor': '#EAEAF2',

'axes.grid': True,

'axes.labelcolor': '.15',

'axes.linewidth': 0.0,

'figure.facecolor': 'white',

'font.family': ['sans-serif'],

'font.sans-serif': ['Arial',

'Liberation Sans',

'Bitstream Vera Sans',

'sans-serif'],

'grid.color': 'white',

'grid.linestyle': '-',

'image.cmap': 'Greys',

'legend.frameon': False,

'legend.numpoints': 1,

'legend.scatterpoints': 1,

'lines.solid\_capstyle': 'round',

'text.color': '.15',

'xtick.color': '.15',

'xtick.direction': 'out',

'xtick.major.size': 0.0,

'xtick.minor.size': 0.0,

'ytick.color': '.15',

'ytick.direction': 'out',

'ytick.major.size': 0.0,

'ytick.minor.size': 0.0}

或许，你可以试试不同种类的参数效果

sns.set\_style("darkgrid", **{"axes.facecolor": ".9"}**) 传字典

## 通过 plotting\_context() 和 set\_context() 调整绘图元素

首先，可以通过sns.set()重置参数。

sns.set()

四种预设，按相对尺寸的顺序(线条越来越粗)，分别是paper，notebook, talk, and poster。notebook的样式是默认的，上面的绘图都是使用默认的notebook预设。

sns.set\_context("paper")

plt.figure(figsize=(8,6))

通过更改context还可以独立地扩展字体元素的大小。(这个选项也可以通过顶级set()函数获得），通过提供参数值的字典来覆盖参数。

sns.set\_context("notebook", font\_scale=1.5, rc={"lines.linewidth": 2.5})

注释

plt.annotate(text,xy=arrow\_crd,xytext=text\_crd,arrowprops=dict)

text:要注释的字符串

xy：箭头坐标

xytext:字符串坐标

arrowprops:颜色大小等属性。 例子：

plt.annotate(str,xy=(2,1),xytext=(3,15),arrowprops=dict(shrink=1,width=2))