5 散点图

5.1 普通散点图

用matplotlib实现

df.plot.scatter() : 对matplotlib.pyplot.scatter的打包调用。

In [ ]:

```
ccss.plot.scatter('s3', 'index1')
```

In [ ]:

```
plt.scatter(ccss.s3, ccss.index1, s = ccss.index1, c = ccss.index1)
```

用seaborn实现

scatterplot函数的绝大部分参数含义和lineplot非常类似，因此不再详细解释。

seaborn.scatterplot(

```
    x, y, hue : names of variables in data
    data : 用于绘图的数据框
    hue_order : 分类变量/hue变量各类别取值的绘图顺序

    estimator = None : 对y变量的汇总方式，为None时绘制所有原始值
    legend = 'brief' : 图例的显示方式
        “brief”, “full”, or False, optional

    格式设定:
        size : 线段宽度所对应的变量/数值
        sizes : list, dict, or tuple, 用于进一步设置线宽如何确定
        size_norm : 进一步指定数值的标准化方法用于线段宽度
        size_order : list, 线宽在各线段中的使用顺序

        style : 线段形状所对应的变量/数值
        style_order : list, 线形的使用顺序

        markers : 数据点的显示方式
            boolean, list, or dictionary, optional

        palette : hue变量所对应的调色盘设定，dict/seaborn调色盘格式。
        hue_norm : 当hue变量为数值时，可进一步指定数值的标准化方法用于颜色映射

        alpha : float, 散点的不透明度比例
```

)

In [ ]:

```
sns.scatterplot(ccss.s3, ccss.index1)
```

In [ ]:

```
# 分组散点图
sns.scatterplot(ccss.s3, ccss.index1, ccss.s2)
```

In [ ]:

```
sns.scatterplot(ccss.s3, ccss.index1, ccss.s2, size = ccss.Qs9)
```

5.2 变量间的回归趋势考察

matplotlib/seaborn中的相应功能只是采用图形方式对回归趋势进行观察，并不是要替代statsmodels中相应的建模分析功能。

lmplot()：功能比较完备，可考察各种常见的线性/曲线回归趋势，可按照分类变量分组/分行列面板，并同时计算可信区间。
regplot()：可看作lmplot()的一个简单子集，绘制时使用一个Axes对象，无分行列面板考察的功能。
jointplot()：调用regplot()同时呈现回归趋势和单变量分布特征。
pairplot()：将regplot()和PairGrid相结合，实现在矩阵中两两考察回归趋势的功能。

5.2.1 两变量基本回归关系的考察

seaborn.lmplot(

x, y, data :

格式设定：
palette : palette name, list, or dict, optional
height : 高度(in inches)
aspect : 宽高比
markers : matplotlib marker，散点标记，有分组变量时提供list
{x,y}_jitter : floats，绘图时对散点加入随机噪声
当数值为离散值时可以改善散点图效果
truncate = False : bool，是否只在x取值范围内绘制回归线

基本绘图设定：
x_estimator : 对x运用该函数并显示结果
x_bins : int/vector，对变量x进行分箱，然后计算其集中趋势和ci
该操作只影响散点图，不影响回归趋势线的估计
x_ci : "ci", "sd", int in [0, 100] or None，计算的可信区间大小
为ci时，另外使用ci参数给出百分比
scatter : bool，是否绘制散点图
fit_reg : bool，是否拟合回归模型
n_boot : int，Bootstrap抽样次数
units : variable name in data，抽样基本单位对应的变量

{x,y}_partial : strings in data/matrices，绘图/拟合时希望控制的协变量
注意此处x/y的设定方向和一般使用习惯相反！

{scatter,line}_kws : dict，plt.scatter和plt.plot中的其余参数

)

In [ ]:

```
sns.lmplot(x = "s3", y="index1", data=ccss)
```

In [ ]:

```
# 对年龄进行分段散点图呈现,分段后呈现因变量CI而不是原始数值
sns.lmplot(x = "s3", y="index1", data=ccss, x_bins = 15)
```

In [ ]:

```
# 不分段呈现因变量CI而不是原始数值
sns.lmplot(x = "s3", y="index1", data=ccss, x_estimator=np.mean)
```

In [ ]:

```
# 分段进行特殊指标的CI估计
sns.lmplot(x = "s3", y="index1", data=ccss, x_bins = 15,
           x_estimator=np.median)
```

In [ ]:

```
# 在模型中控制更多自变量，0.9版本会报错，老版本可以运行
sns.lmplot(x = "s3", y="index1", data=ccss, y_partial = "Qs9")
```

5.2.2 特殊回归趋势的拟合

seaborn.lmplot(

复杂曲线拟合：

order : int，所拟合曲线的阶数，大于1时使用numpy.polyfit进行曲线拟合

logistic : bool，是否拟合logistic回归曲线

lowess : bool，是否拟合lowess曲线

robust : bool，是否拟合稳健回归

logx : bool，是否拟合y ~ log(x)的对数曲线，但仍按照原始的x/y数值输出

)

In [ ]:

```
anscombe = sns.load_dataset("anscombe")
anscombe.head()
```

In [ ]:

```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'I'"),
           ci=None)
```

In [ ]:

```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'II'"),
           ci=None)
```

In [ ]:

```
# 直接指定拟合高次项
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'II'"),
           order=2, ci=None)
```

In [ ]:

```
# 生成所需的高次项
data2 = anscombe.query("dataset == 'II'")
data2['x2'] = (data2.x - 9)**2
data2
```

In [ ]:

```
# 在回归模型中控制高次项，0.9版本会报错，老版本可以运行
sns.lmplot(x = "x", y = "y", data = data2, y_partial = 'x2')
```

In [ ]:

```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'III'"),
           ci=None)
```

In [ ]:

```
# 拟合稳健回归模型
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'III'"),
           robust=True, ci=None)
```

In [ ]:

```
ccss.O1.value_counts()
```

In [ ]:

```
ccss['O1logic'] = ccss.O1 == '有'
ccss.O1logic.value_counts()
```

In [ ]:

```
# 拟合logistic回归曲线
sns.lmplot(x="s3", y="O1logic", data=ccss, logistic=True)
```

In [ ]:

```
# 拟合lowess曲线
sns.lmplot(x = "s3", y="index1", data=ccss, lowess=True)
```

In [ ]:

```
# 放大回归细节以便观察
sns.lmplot(x = "s3", y="index1", data=ccss, lowess=True)
plt.ylim(80,110)
```

5.2.3 残差考察

seaborn.residplot()使用的参数均在seaborn.lmplot()出现，这里不再复述。

In [ ]:

```
sns.residplot(x="x", y="y", data=anscombe.query("dataset == 'I'"))
```

In [ ]:

```
# 处理分布不理想的残差
sns.residplot(x="x", y="y", data=anscombe.query("dataset == 'II'"))
```

In [ ]:

```
sns.residplot(x="x", y="y", data=anscombe.query("dataset == 'II'"),
              order = 2)
```

In [ ]:

```
# 0.9版本会报错，老版本可以运行
sns.residplot(x = "x", y = "y", y_partial = "x2", data = data2)
```

5.2.4 分组考察回归关系

seaborn.lmplot(

数据分组：

hue, col, row : strings，用于定义分组、列面板、行面板的变量名

col_wrap : int，当超过指定的宽度后，列面板折叠至下一行显示

share{x,y} : bool, 'col', or 'row'，有行列面板时是否共用x/y轴

{hue,col,row}_order : lists，相应分类变量的显示顺序

legend = True : bool，是否显示图例

legend_out = True : bool，是否在图形外面显示图例

)

In [ ]:

```
# 图例分组
sns.lmplot(x = "s3", y="index1", hue = "s2", data = ccss)
```

In [ ]:

```
# 图例分组
sns.lmplot(x = "s3", y="index1", hue = "s2", data = ccss, ci = None)
plt.ylim(80, 110)
```

In [ ]:

```
sns.lmplot(x = "s3", y="index1", hue = "s2", data = ccss, lowess=True)
```

In [ ]:

```
# 行列面板
sns.lmplot(x = "s3", y="index1", hue = "s2", col = "time",
           data = ccss, lowess=True)
```

In [ ]:

```
# 行列面板
sns.lmplot(x = "s3", y="index1", hue = "s2", col = "time",
           data = ccss, lowess=True, col_wrap = 2)
```

In [ ]:

```
# 原始的行列面板显示
sns.lmplot(x = "s3", y="index1", hue = "s2", row = "s0",
           col = "time", data = ccss)
```

In [ ]:

```
# 调整回归线显示范围
sns.lmplot(x = "s3", y="index1", hue = "s2", row = "s0",
           col = "time", data = ccss, truncate = True)
```

In [ ]:

```
# 不共用x轴
sns.lmplot(x = "s3", y="index1", hue = "s2", row = "s0",
           col = "time", data = ccss, sharex = False)
```

5.3 散点图的衍生图形

5.3.1 联合变量分布的散点图

jointplot()函数可以创建一个多面板图形来展示两个变量之间的联合关系，并同时展示每个轴上单变量的分布情况。

seaborn.jointplot(

x, y ： 绘图用数据，可以是数据框内的变量名
data = None ： 数据框名称

kind = 'scatter' ： 绘制的图形种类
{ “scatter” | “reg” | “resid” | “kde” | “hex” }
stat_func = <function pearsonr> ： 需要计算的统计量
该参数在新版本中已经取消

图形格式：
color ： matplotlib color
size = 6 ： 图形大小（默认方形）
ratio = 5 ： 联合图和边上的边际分布图宽度比例
space = 0.2 ： 联合图和边际分布图中间的缝隙宽度

dropna = True ： 是否删除缺失数据
{x, y}lim ： two-tuples, x/y轴的刻度范围
{joint, marginal, annot}_kws ： dicts, 各图形元素的属性设定

) 返回：JointGrid对象。

In [ ]:

```
sns.scatterplot(x = "s3", y = "index1", data = ccss)
```

In [ ]:

```
sns.jointplot(x = "s3", y = "index1", data = ccss)
```

In [ ]:

```
sns.jointplot(x = "s3", y = "index1", data = ccss, kind = 'reg')
```

In [ ]:

```
# 错误但不影响使用的调用方法
sns.jointplot(x = ccss.s3, y = ccss.index1, data = ccss)
```

5.3.2 Hexbin plot

对于大数据集，散点图将会过于密集，难以考察关联趋势。

hexbin图（也称向日葵图）可以将x、y变量按照六角形单元格为单位进行汇总，然后用颜色深浅展示单元格内的样本频数，这种图形对于相对大的数据集效果最好。

该图形可以通过matplotlib的plt.hexbin函数使用，也可以作为jointplot的一种类型参数使用

In [ ]:

```
# 使用jointplot绘制hexbin图
sns.jointplot(x = "s3", y = "index1", data = ccss, kind = 'hex')
```

In [ ]:

```
# 使用jointplot绘制hexbin图
plt.hexbin(x = ccss.s3, y = ccss.index1, gridsize = (40, 40))
```

5.3.3 等高线图

可以使用KDE估计来可视化双变量分布。在seaborn中，这种绘图以等高线图展示，并且可以作为jointplot()的一种类型参数使用。

In [ ]:

```
sns.jointplot(x = "s3", y = "index1", data = ccss, kind = 'kde')
```

In [ ]:

```
# 只绘制等高线图
sns.kdeplot(ccss.s3, ccss.index1)
```

In [ ]:

```
# 让曲线过渡尽量平滑
sns.kdeplot(ccss.s3, ccss.index1, shade = True, n_levels = 40)
```

In [ ]:

```
# 在图形边加绘数值条
sns.kdeplot(ccss.s3, ccss.index1, shade = True,
            cbar = True, n_levels = 40)
```

jointplot()在绘制后返回JointGrid对象，可以使用它添加更多图层或调整可视化的其他方面：

seaborn.JointGrid对象的方法：

```
annotate(func[, template, stat, loc])     添加指定的统计量作为注解。
plot(joint_func, marginal_func[, annot_func])     绘制完整图形。
plot_joint(func, **kwargs)     绘制x和y的双变量图。
plot_marginals(func, **kwargs)     分别绘制x和y的单变量图。
savefig(*args, **kwargs)     保存图形。
set_axis_labels([xlabel, ylabel])     设置轴标签。
```

In [ ]:

```
g = sns.jointplot(ccss.s3, ccss.index1, kind = 'kde',
                  n_levels = 40, color = "m")
# 在已有图形基础上加绘散点图
g.plot_joint(plt.scatter, c = "w", s = 30, linewidth = 1, marker = "+")
```

In [ ]:

```
# 在一条语句中完成上述操作
sns.jointplot(ccss.s3, ccss.index1, kind = 'kde',
              n_levels = 40, color = "m")\
    .plot_joint(plt.scatter, c = "w", s = 30, lw = 1, marker = "+")
```

5.4 散点图矩阵

如果希望考察数据集中多个数值变量两两间的数量关联，可以使用pairplot()函数。它会生成一个含有轴的矩阵，在默认状态下，会将数据集中所有列成对可视化。

seaborn.pairplot(

```
    data : 用于绘图的数据框。
    vars : 可选，需要分析的变量列表，默认分析全部数值变量列。
    {x, y}_vars : 用于分析的行/列变量，此时会生成非对称的矩阵图。
    kind = 'scatter' : 绘图种类，{‘scatter’, ‘reg’}。
    diag_kind = 'auto' : 主对角线绘图种类，{‘hist’, ‘kde’}

    散点分组
        hue : 变量名字符串，该变量将在绘图时用颜色将散点进行分组。
        hue_order : 分组变量各取值绘图时在调色板中的顺序，list格式。
        palette : hue变量所对应的调色盘设定，dict/seaborn调色盘格式。

    格式设定:
        markers : 散点标记，hue分组时以list格式提供。
        height = 2.5 : 面板高度，英寸。
        aspect = 1 : 面板宽度与高度之比，aspect * size即为面板宽度。
        dropna = True : 绘图前是否删除缺失值。
        {plot, diag, grid}_kws : dicts, keyword arguments
```

) 返回：PairGrid对象。

In [ ]:

```
sns.pairplot(data = ccss, vars = ['index1', 'index1a', 'index1b'])
```

In [ ]:

```
sns.pairplot(data = ccss, x_vars = 's3',
             y_vars = ['index1a', 'index1b'],
             kind = 'reg', hue = 's2', height = 4, aspect = 1.5)
```

seaborn.PairGrid对象的方法：

```
add_legend([legend_data, title, label_order])     绘制图例
map(func, **kwargs)     指定所有单元格绘制的图形种类
map_diag(func, **kwargs)     指定主对角线单元格绘制的单变量图形种类
map_lower(func, **kwargs)     指定下三角单元格绘制的双变量图形种类
map_offdiag(func, **kwargs)     指定非主对角线单元格绘制的双变量图形种类
map_upper(func, **kwargs)     指定上三角单元格绘制的双变量图形种类
savefig(*args, **kwargs)     保存图形
set(**kwargs)     设置每一个subplot Axes对象的属性
```

In [ ]:

```
sns.PairGrid(data = ccss,
             vars = ['index1', 'index1a', 'index1b']).map(sns.kdeplot)
```

In [ ]:

```
sns.PairGrid(data = ccss,
    vars = ['index1', 'index1a', 'index1b']).map_upper(sns.scatterplot)
```

5.5 3D散点图

matplotlib中提供的是比较基本的3D散点图功能，并未做特别的强化。

在单独窗口中可以做三维旋转观察，此外并无更多功能。

Axes3D.scatter(

```
xs, ys : 散点的x/y坐标
zs = 0 : 散点的z坐标，默认为0
zdir = 'z' : 当实际绘制2维图时设定哪一个维度为z轴
s = 20 : 散点大小，也可以为变量或者与x/y等长的数组
c :  散点颜色
depthshade = True : 是否给散点提供影深
```

)

In [ ]:

```
# from mpl_toolkits.mplot3d.axes3d import Axes3D
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = Axes3D(fig)

ax.scatter(ccss.s3, ccss.index1a, ccss.index1b)

plt.show()
```

5.6 实战练习

现希望分受教育程度考察年龄S3和总指数index1之间的关联情况，但使用图组的话，回归线重叠的会比较厉害，不便于观察，希望能够使用矩阵方式进行考察，请问如何能实现该需求?

自行思考怎样搭配使用seabron中的回归趋势考察功能和statsmodels中的回归建模功能，才能够使得这两个模块的优势得到充分的发挥。

6 纳入更多变量信息

6.1 设置图例

matplotlib中的图例设定

matplotlib.pyplot.legend(

```
handles : 需要设置图例标签的图形元素列表
labels : 相应图形元素的图例标签文字列表
loc = 'upper right' : 图例的显示位置, int/string
    '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
    也可以按照2-tuple格式直接指定从左下角起的(x, y)坐标值
```

)

完整参考信息：https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html)

常见用法：

```
legend() # 显示图例
legend(labels) # 按默认元素顺序设定标签并显示图例
legend(handles, labels) # 设定指定元素的标签并显示图例
```

In [ ]:

```
plt.plot([1,2,2,1], label='A line')
```

In [ ]:

```
plt.plot([1,2,2,1], label='A line')
plt.legend()
```

In [ ]:

```
plt.plot([1,2,2,1])
plt.legend(['Line Label'])
```