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'])
```

In [ ]:

```
plt.plot([1,2,2,1])
plt.legend('Line Label') # 错误的设定方式
```

In [ ]:

```
# 直接指定图例坐标
plt.plot([1,2,2,1])
plt.legend(['Line Label'], loc = (0.5,0.5))
```

In [ ]:

```
# 此处赋值时需要以逗号结尾，否则返回的是对象列表
l1, = plt.plot([1, 2, 2, 1], label = 'sfsad')
l2, = plt.plot([1, 1.5, 1.5, 1])
plt.legend(handles = [l1, l2],
           labels = ['Line Label', 'line2'], loc = 'best')
```

In [ ]:

```
# 不加逗号则需要准确指定到列表的具体对象
l1 = plt.plot([1,2,2,1])
l2 = plt.plot([1,1.5,1.5,1])
plt.legend(handles = [l1[0], l2[0]],
           labels = ['Line Label', 'line2'], loc = 'best')
```

In [ ]:

```
# 不加逗号则需要准确指定到列表的具体对象
l1 = plt.plot([1,2,2,1])
l2 = plt.plot([1,1.5,1.5,1], label = 'asdasdf')
# 只要求显示某些图形元素的标签
plt.legend(handles = [l1[0]],
           labels = ['Line Label'], loc = 'best')
```

seaborn中的图例设定

图例也可以针对连续变量进行设定，此时可以修改图例的详细程度。

In [ ]:

```
# 显示简化图例
sns.lineplot('s4', 'index1', data = ccss, size='Qs9', ci = None)
```

In [ ]:

```
# 完整显示详细图例
sns.lineplot('s4', 'index1', data = ccss, size='Qs9',
             legend = 'full', ci = None)
```

面板图形中的图例显示位置

In [ ]:

```
# 面板图形中图例显示在外部
sns.lmplot(x = "s3", y="index1", hue = "s2", col = "s0", data = ccss)
```

In [ ]:

```
# 面板图形中图例显示在内部
sns.lmplot(x = "s3", y="index1", hue = "s2", col = "s0",
           data = ccss, legend_out = False)
```

6.2 混合图形

matplotlib中实现混合图形的基本原理非常简单，就是多个图形元素的直接叠加。

In [ ]:

```
ax1 = sns.kdeplot(ccss.s3)
```

In [ ]:

```
# 使用seaborn生成叠加图
ax1 = sns.kdeplot(ccss.s3)
sns.distplot(ccss.s3, kde = False, norm_hist = True, ax = ax1)
```

In [ ]:

```
# 混合使用seaborn和matplotlib生成叠加图
ax1 = sns.kdeplot(ccss.s3)
ax1.hist(ccss.s3, density = True)
```

In [ ]:

```
# 进一步的简化写法
sns.kdeplot(ccss.s3).hist(ccss.s3, density = True)
```

6.3 双轴图

使用pandas实现

matplotlib中可以直接设置双轴图，但更简便的方法是直接使用pandas.plot()中相应的功能实现。

In [ ]:

```
# 只使用第二y轴
ccss.plot.kde(y = 'index1', secondary_y = True)
```

In [ ]:

```
# 同时使用双轴，并指定使用第二y轴的变量
ccss.loc[:,['index1', 's3']].plot.kde(secondary_y = 's3')
```

用matplotlib实现

Axes.twinx()：建立一个新的Axes对象，该对象拥有独立的y轴，且该轴位于右侧。x轴则不可见（共用原有对象的x轴）

In [ ]:

```
# 建立一个空白双轴图
plt.gca().twinx()
```

In [ ]:

```
# 设置第一y轴对应的图形
ccss.index1.plot.kde(label = 'index1')
plt.gca().set_ylabel('$Index1$')

# 设置第二y轴对应的图形
ax2 = plt.gca().twinx()
sns.kdeplot(ccss.s3, legend = None, ax = ax2)
ax2.set_ylabel('$S3$')
```

6.4 使用行/列面板

行/列面板的本质就是指定具体的分组变量用于行/列拆分，除了lmplot等命令内置此功能之外，还可以使用专用的catplot()实现通用的行/列面板操作。

seaborn.catplot(# 0.9.0版之前为factorplot

```
    x, y, hue : names of variables in data
    data : DataFrame

    面板相关设定:
        row, col : 行/列面板对象名称
        col_wrap : 在指定的宽度折叠列面板至下一行，从而成为多行显示
        row_order, col_order : 行/列面板变量各类别的显示顺序
        legend = True : bool, optional
        legend_out = True : 将图例绘制在设定的图形区域外
        share{x,y} : 是否共用行/列

    kind : 可以绘制的图形种类
        “point”, “bar”, “strip”, “swarm”, “box”, “violin”, or “boxen”

    其余设定:
        estimator : 希望在每个类别中计算的统计量
        ci : float or “sd” or None, 希望绘制的可信区间宽度
        n_boot : 计算CI时的bootstrap抽样次数
        units : 用于确定抽样单元大小的变量
        order, hue_order : lists of strings, optional
        size : scalar, optional
        aspect : scalar, optional
        orient : “v” | “h”, optional
        color : matplotlib color, optional
        palette : palette name, list, or dict, optional
```

) 返回值：FacetGrid对象。

In [ ]:

```
sns.catplot(x = 'time', y = 'index1', col = 's0', data = ccss,
            kind = "box")
```

In [ ]:

```
sns.catplot(x = 'time', y = 'index1', col = 's0', data = ccss,
            kind = "boxen")
```

In [ ]:

```
sns.catplot(x = 'time', y = 'index1', col = 's0', data = ccss,
            kind = "violin")
```

In [ ]:

```
sns.catplot(x = 'time', y = 'index1', col = 's0', hue = 's2',
            data = ccss, kind = "bar")
```

In [ ]:

```
sns.catplot(x = 's0', y = 'index1', col = 'time', hue = 's2',
            data = ccss, kind="bar", col_wrap = 2, legend_out = False)
```

6.5 实战练习

尝试在同一张图形中给出分受教育程度的年龄均数和总信心指数均数的信息，要求：

年龄均数用线图表示。
总信心指数用带CI的条图表示。
对颜色、数轴等作必要的设定，使得图形显示效果达到最佳。

比较seaborn中的catplot()命令和pairplot()命令在使用目的、参数设定等各方面的异同，自行思考这两个命令应当适用于哪些数据可视化的需求。

7 子图与图形网格

7.1 图形叠加/图中图

在matplotlib中，所有的绘图操作实际上都是以Axes对象为独立的绘图区域进行，这里称为子图。

在一个Figure对象中可以有多个子图，这些子图对象可以叠加存在，从而形成图中图的效果。

matplotlib.figure.Figure.add_axes(# 在已有Figure对象中按照指定范围添加子图

```
rect : 代表插入子图对象大小的序列。
    [left, bottom, width, height]
projection : 子图使用的坐标体系。
    ['aitoff' | 'hammer' | 'lambert' |
    'mollweide' | 'polar' | 'rectilinear']
polar : boolean, 为True时等价于projection = 'polar'
```