尝试使用不同的图形工具对总指数（index1）进行可视化描述。

尝试使用不同的图形工具对频数大于50的职业类别（s5）进行可视化描述。

使用年龄（s3），自行尝试绘制增强箱图和提琴图。

4 复杂条图，线图与面积图

4.1 复杂条图

4.1.1 带误差线的条图

当直条用于显示样本统计量时，往往需要加绘相应指标的可信区间。

seaborn.barplot(

```
    x, y, hue : names of variables in data
    data : DataFrame
    order, hue_order : 分类变量/hue变量各类别取值的绘图顺序

    可信区间计算:
        ci = 95 : float or "sd" or None, 希望绘制的可信区间宽度
        n_boot = 1000 : 计算CI时的bootstrap抽样次数。
        units : 用于确定抽样单元大小的变量。

    误差条格式:
        errcolor = '0.26' : CI线段的颜色, matplotlib color
        errwidth : CI线段的粗细, float
        capsize : 误差条顶端的宽度，占直条绘图区的比例, float
```

)

In [ ]:

```
sns.barplot(x = ccss.s4, y = ccss.s3)
```

In [ ]:

```
sns.barplot(x = ccss.s4, y = ccss.s3, color = 'c',
            errcolor = 'b', errwidth = '2', capsize = .1)
```

In [ ]:

```
# 如何不绘制CI
sns.barplot(x = ccss.s4, y = ccss.s3, ci = None)
plt.xlabel("S4: 教育程度")
plt.ylabel("S3: 年龄均值")
```

用直条表示中位数、标准差等特殊统计量

https://docs.scipy.org/doc/numpy-1.14.5/reference/routines.statistics.html (https://docs.scipy.org/doc/numpy-1.14.5/reference/routines.statistics.html)

In [ ]:

```
import numpy as np

sns.barplot(x = ccss.s4, y = ccss.s3, estimator=np.median)
```

In [ ]:

```
sns.barplot(x = ccss.s4, y = ccss.s3, estimator=np.std)
```

4.1.2 分组条图

In [ ]:

```
sns.barplot(x = ccss.s4, y = ccss.s3, hue = ccss.s2)
```

In [ ]:

```
# 存在嵌套分组
sns.barplot(x = ccss.Qs9, y = ccss.s3, hue = ccss.Ts9)
```

In [ ]:

```
# 存在嵌套分组时不调整直条宽度
sns.barplot(x = ccss.Qs9, y = ccss.s3, hue = ccss.Ts9, dodge = False)
```

4.1.3 堆积条图

堆积条图在matplotlib中没有命令直接实现，但可以通过重叠作图方式加以绘制：

方法一：先绘制全部类别的的累积直条，然后依次绘制n-1、n-2...个类别的累积直条，最终形成堆积条图的效果。
方法二：利用plt.plot()的bottom参数，依次将新类别的直条叠加在已有直条上方。

上述方法二更常用，且结合Pandas的汇总功能和seabon模块的绘图功能，实现堆积条图已经非常方便。

In [ ]:

```
# 利用Pandas的汇总功能生成所需汇总数据
tmpdf = pd.crosstab(index = ccss.s0, columns = ccss.s4)
tmpdf
```

In [ ]:

```
# 取出所需的汇总行
tmpdf.loc[['上海']]
```

In [ ]:

```
# 取出所需的汇总行（序列格式）
tmpdf.loc['上海']
```

In [ ]:

```
# 分三次使用seaborn绘制三个类别的直条，并依次叠加
sns.barplot(data = tmpdf.loc[['上海']], color = 'b', label = '上海')
sns.barplot(data = tmpdf.loc[['北京']], bottom = tmpdf.loc['上海'],
            color = 'c', label = '北京')
# 注意bottom参数需要累加已有的所有直条类别高度
sns.barplot(data = tmpdf.loc[['广州']],
            bottom = tmpdf.loc['北京'] + tmpdf.loc['上海'],
            color = 'y', label = '广州')
plt.legend()
```

In [ ]:

```
# 使用循环程序自动生成图形
tmpdf = pd.crosstab(index = ccss.s0, columns = ccss.s4)
colorstep0 = 1/len(tmpdf.index)
for i in range(len(tmpdf.index)):
    if i == 0:
        colorstep = colorstep0 / 2 # 避免最终出现纯白色直条
        sns.barplot(data = tmpdf.loc[[tmpdf.index[i]]],
                    color = str(colorstep), label = tmpdf.index[i])
        base = tmpdf.loc[tmpdf.index[i]]
    else:
        sns.barplot(data = tmpdf.loc[[tmpdf.index[i]]],
                    color = str(colorstep), bottom = base,
                    label = tmpdf.index[i])
        base = base + tmpdf.loc[tmpdf.index[i]]
    colorstep = colorstep + colorstep0
plt.legend()
```

4.1.4 百分条图

基本绘制思路和堆积条图相同，但需要将数据计算为相应的百分比。

python中没有为多选题专门提供分析/绘图功能，需要自行完成所需指标的汇总计算。

In [ ]:

```
pd.crosstab(index = ccss.s0, columns = ccss.s4, normalize = "columns")
```

In [ ]:

```
tmpdf = pd.crosstab(index = ccss.s0, columns = ccss.s4,
                    normalize = "columns")
colorstep0 = 1/len(tmpdf.index)
for i in range(len(tmpdf.index)):
    if i == 0:
        colorstep = colorstep0 / 2
        sns.barplot(data = tmpdf.loc[[tmpdf.index[i]]],
                    color = str(colorstep), label = tmpdf.index[i])
        base = tmpdf.loc[tmpdf.index[i]]
    else:
        sns.barplot(data = tmpdf.loc[[tmpdf.index[i]]],
                    color = str(colorstep), bottom = base,
                    label = tmpdf.index[i])
        base = base + tmpdf.loc[tmpdf.index[i]]
    colorstep = colorstep + colorstep0
plt.legend()
```

4.2 线图、误差图与面积图

4.2.1 线图

df.plot()：默认绘制的就是线图，实际上是对matplotlib.pyplot.plot()的打包调用。

matplotlib.pyplot.plot()：默认绘制的也是线图，事先计算好汇总数值即可。

seaborn.lineplot(): seaborn 0.9版本新增，绘制各类线图。

seaborn.pointplot(): 绘制x轴为有序分类变量的线图，可叠加绘制误差图。

用matplotlib绘制

In [ ]:

```
sumdata = ccss.groupby('time').index1.mean()
sumdata
```

In [ ]:

```
sumdata.plot()
```

In [ ]:

```
plt.plot(sumdata)
```

In [ ]:

```
# 将数值索引转换为字符串格式
sumdata.index.astype('str')
```

In [ ]:

```
plt.plot(sumdata.index.astype('str'), sumdata, 'bo-')
```

用lineplot()函数绘制

seaborn.lineplot()针对的x轴为理论上可以连续取值的变量（如时间），因此默认计算出的CI为连贯的条带。

如果x轴是有序分类变量，则应当慎用该函数

seaborn.lineplot(

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

estimator = 'mean' : 对y变量的汇总方式，为None时绘制所有原始值
sort = True: 绘图前数据是否按照x和y轴变量排序，否则将按照数据原始顺序绘制
    绝大多数情况下，按照原始数据绘制的图形并无意义
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
    dashes = True : 针对style变量的线形绘制设定
        boolean, list, or dictionary
    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

可信区间计算:
    ci = 95 : float or “sd” or None, 希望绘制的可信区间宽度
    n_boot = 1000 : 计算CI时的bootstrap抽样次数
    units : 用于确定抽样单元大小的变量
    err_style = 'band' : “band” or “bars”, 代表CI的误差条的显示方式
    err_band : dict of keyword arguments, 进一步控制误差条的显示方式
        具体参数来自于ax.fill_between或ax.errorbar
```

)

In [ ]:

```
# 错误地使用lineplot
sns.lineplot(ccss.time, ccss.index1)
```

In [ ]:

```
# 正确的使用lineplot，x轴变量设定为分类，误差修改为离散条状
sns.lineplot(ccss.time.astype('str'), ccss.index1, err_style ='bars')
```

In [ ]:

```
# 只绘制线图
sns.lineplot(ccss.s3, ccss.index1, ci = None)
```

In [ ]:

```
# 绘制分组线图
sns.lineplot(ccss.s3, ccss.index1, hue = ccss.s2, ci = None)
```

In [ ]:

```
# 使用线形
sns.lineplot('s4', 'index1', data = ccss,
             style = 's0', size = 's0', ci = None)
```

In [ ]:

```
# 在图例中同时包含颜色和线形
sns.lineplot('s4', 'index1', 's2', data = ccss,
             style = 's0', ci = None)
```

4.2.2 误差图

seaborn.pointplot()针对的x轴为有序分类变量，在线图的基础上着重显示点估计值和对应点的可信区间范围。

http://seaborn.pydata.org/generated/seaborn.pointplot.html
在不绘制连线的情况下，即构成标准的误差图

In [ ]:

```
sns.pointplot(x = ccss.time, y = ccss.index1, capsize = .1)
```

In [ ]:

```
# 不绘制连线
sns.pointplot(x = ccss.time, y = ccss.index1,
              capsize = .1, join = False)
```

In [ ]:

```
sns.pointplot(x = ccss.time, y = ccss.index1, hue = ccss.s0, ci = None)
```

In [ ]:

```
# 增加线形修饰
sns.pointplot(x = ccss.time, y = ccss.index1, hue = ccss.s0,
              ci = None, linestyles = ['-','--',':'])
```

In [ ]:

```
# 稍微错开CI的位置以便于观察
sns.pointplot(x = ccss.time, y = ccss.index1,
              hue = ccss.s0, dodge = True)
```

4.2.3 面积图

可以近似地理解为将线图下方填充满颜色即可。

分组面积图可以有是否叠加（stacked）的选项。

也可按照面积图的基本原理，直接使用区域填色方式实现面积图。

In [ ]:

```
ccss.a3.value_counts()
```

In [ ]:

```
ccss.groupby('a3').index1.mean().plot.area()
```

In [ ]:

```
ccss.groupby('s3').index1.mean().plot.area()
```

In [ ]:

```
tmpdf = pd.crosstab(index = ccss.s4, columns = ccss.O1)
tmpdf
```

In [ ]:

```
tmpdf.plot.area()
```

In [ ]:

```
tmpdf.plot.area(stacked = False)
```

In [ ]:

```
# 按照百分比填充面积
pd.crosstab(index = ccss.s4, columns = ccss.O1,
            normalize = 'index').plot.area()
```

4.3 实战练习

请使用适当的图形工具对总指数（index1）进行考察：

总指数分城市、月份交叉的变化规律。
总指数在不同教育程度、学历、职业间，以及上述指标交叉后的的变化规律。
总指数按照年龄5岁一组段时的变化规律。
