2.4 实战练习

现希望对两个连续变量间的关联状况进行展示，但同时需要考虑展示另外3个分类变量的信息，请设计一个恰当的统计图框架。

现希望对两个连续变量的平均水平进行展示，但同时需要考虑展示另外4个分类变量的信息（即进行交叉分类），其中3个分类变量较为重要，另外1个相对较次要。请设计一个恰当的统计图框架。

现希望分3个分类变量的交叉组合，对一个连续变量和一个分类变量在各单元格中的均值/构成比分布状况进行展示，请设计一个恰当的统计图框架。

> 注意：上述问题中基于变量的取值范围等，可选的框架或许不止一个。

3 单变量信息的可视化

In [ ]:

```
# 准备ccss数据集
import pandas as pd

ccss = pd.read_excel("ccss_sample.xlsx")
ccss.head()
```

3.1 分类变量的可视化

3.1.1 简单条图

用matplotlib实现

DataFrame.plot.bar()：对matplotlib.pyplot.bar()的打包调用。

> 原始的`pyplot.bar()`需要同时定义类别变量和直条高度变量，打包调用简化了该操作

http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html)

In [ ]:

```
# 生成汇总数据
pd.value_counts(ccss.s5)
```

In [ ]:

```
pd.value_counts(ccss.s5).plot.bar()
```

In [ ]:

```
plt.bar(pd.value_counts(ccss.s5).index, pd.value_counts(ccss.s5))
```

用seaborn实现

seaborn和pandas有着更好的功能整合，因此支持下列两种数据格式：

类似于matplotlib，x/y轴等变量均以单独的数据序列形式出现
提供数据框名称，绘图用x/y轴等变量均为数据框中的变量列名称（long-form）
提供数据框名称，所有数值列均用于绘图（wide-form）

seaborn.barplot(

x, y, hue ： 绘图中所使用的分类/连续变量/颜色分组变量名
data ： 数据框名称

order, hue_order ： hue变量各类别取值的绘图顺序
orient ： “v” | “h”，条带绘制方向
saturation = 0.75 ： float，直条颜色的饱和度

)返回：**matplotlib的Axes对象，注意seaborn这里返回的是一个Axes对象！**

In [ ]:

```
# 按照长型数据格式绘图
pd.value_counts(ccss.s5)
```

In [ ]:

```
sns.barplot(x = pd.value_counts(ccss.s5).index,
            y = pd.value_counts(ccss.s5))
```

In [ ]:

```
# 按照宽型数据格式绘图
pd.DataFrame(pd.value_counts(ccss.s5)).T
```

In [ ]:

```
sns.barplot(data = pd.DataFrame(pd.value_counts(ccss.s5)).T)
```

In [ ]:

```
sns.barplot(data = pd.DataFrame(pd.value_counts(ccss.s5)).T,
            color = '#ff0000')
```

In [ ]:

```
# 自定义直条排列顺序
sns.barplot(x = pd.value_counts(ccss.s4).index,
            y = pd.value_counts(ccss.s4))
```

In [ ]:

```
sns.barplot(x = pd.value_counts(ccss.s4).index,
            y = pd.value_counts(ccss.s4),
            order = ['初中/技校或以下','高中/中专','大专'])
```

用countplot()直接绘图

所需参数基本上和barplot完全相同。

由于一个数轴已确定为频数，因此x和y参数不能同时出现。

In [ ]:

```
sns.countplot(ccss.s4)
```

百分条图

由于matplotlib中没有直接绘制百分条图的命令，因此将合并在复杂条图中进行介绍，此处略去。

3.1.2 饼图、半圆图与圆环图

3.1.2.1 普通饼图

DataFrame.plot.pie()：对matplotlib.pyplot.pie()的打包调用。

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.pie.html
(https://matplotlib.org/api/_as_gen/matplotlib.pyplot.pie.html)

In [ ]:

```
ccss.s5.value_counts().plot.pie()
```

In [ ]:

```
plt.pie(ccss.s5.value_counts(), labels = ccss.s5.value_counts().index)
```

In [ ]:

```
plt.pie(ccss.s5.value_counts(), labels = ccss.s5.value_counts().index,
       explode = [0,0.5,0,0.5,1,0,1,0,1,0,1])
```

In [ ]:

```
plt.pie(ccss.s5.value_counts(), labels = ccss.s5.value_counts().index,
        explode = [0,0.5,0,0.5,1,0,1,0,1,0,1],
        shadow = True, startangle = 90, radius = 2,
        counterclock = False, rotatelabels = True)
```

3.1.2.2 半圆图

饼块对应数值大于等于1时，会自动转换为所对应的构成比并加以显示。

当饼块对应数值总和小于1时，则按照原始数值绘制饼块大小，此时就可形成半圆图/扇区图。

另一种思路是绘制和底色相同的扇区，但在matplotlib中这样做显然画蛇添足了。

In [ ]:

```
plt.pie([0.1,0.2,0.1,0.3])
```

In [ ]:

```
plt.figure(figsize=(6, 6))
plt.pie(ccss.s5.value_counts(normalize = True)/2,
        labels = ccss.s5.value_counts().index)
```

3.1.2.3 圆环图

Axes对象可以叠加绘图，因此利用该功能在饼图中心叠加一个同底色的圆形即可。

In [ ]:

```
plt.pie([1])
```

In [ ]:

```
plt.pie([1], colors = ['#ffffff'])
```

In [ ]:

```
plt.figure(figsize=(6, 6))
plt.pie(ccss.s5.value_counts(),
        labels = ccss.s5.value_counts().index)
plt.pie([1], colors = ['#ffffff'], radius = 0.7)
```

In [ ]:

```
#半圆环图
plt.figure(figsize=(6, 6))
plt.pie(ccss.s5.value_counts(normalize = True)/2,
        labels = ccss.s5.value_counts().index)
plt.pie([1], colors = ['#ffffff'], radius = 0.7)
```

3.2 连续变量的可视化

3.2.1 条带图

seaborn.stripplot()命令

seaborn.stripplot(

x, y, hue : 绘图中所使用的分类/连续变量/颜色分组变量名
data : 数据框名称

绘制细节:
order, hue_order : hue变量各类别取值的绘图顺序
jitter = True : 是否对散点位置进行随机抖动以避免重叠
orient : "v" | "h", 条带绘制方向

其他设定:
color : matplotlib颜色设定
palette : 调色盘设定
size : float, optional
linewidth : float, optional
ax : matplotlib Axes, optional

)

In [ ]:

```
# 样本较多时散点通常会重叠，使得图形失去使用价值。
sns.stripplot(y = ccss.s3[:200], jitter = False)
```

In [ ]:

```
# 解决方案是使用一些随机的"抖动"来调整位置(仅沿着分类轴方向抖动)
sns.stripplot(y = ccss.s3[:200], jitter = True)
```

seaborn.swarmplot()命令

swarmplot()默认使用避免重叠点的算法来定位分类轴上的每个散点，呈现效果更好。

In [ ]:

```
sns.swarmplot(y = ccss.s3[:200])
```

3.2.2 直方图，KDE图与地毯图

用matplotlib实现

DataFrame.hist() : 直接打包调用matplotlib.pyplot.hist()

hist()命令的细节:

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.hist.html

In [ ]:

```
ccss.s3.plot.hist()
```

In [ ]:

```
plt.hist(ccss.s3)
```

用seaborn实现

seaborn.distplot() : 在功能上直接整合了matplotlib.pyplot.hist()，seaborn.kdeplot()和seaborn.rugplot()三个绘图函数，用于考察连续变量的分布特征更为方便。

seaborn.distplot(

基本设定:
a : 绘图用数据列，有变量名时会被用作轴标签
bins = None : 组段数，matplotlib.pyplot.hist()中使用的参数
默认按照Freedman-Diaconis自动计算

绘制图形种类:
hist = True : 直方图
kde = True : gaussian核密度估计曲线
rug = False : rugplot

图形细节:
vertical = False : 是否水平显示图形
norm_hist = False : 纵轴显示概率密度而不是原始频数

其他设定:
{hist, kde, rug, fit}_kws : 字典类型，自定义对应图形元素的各类属性
color : matplotlib颜色代码，指定绘图元素的颜色
axlabel : 轴标签
label : 图例中对应相应图形元素的标签
ax : 将图形绘制在指定的Axes对象内

)

In [ ]:

```
# 只绘制rugplot
sns.distplot(ccss.s3, hist = False, kde = False, rug = True)
```

In [ ]:

```
# 只绘制直方图
sns.distplot(ccss.s3, kde = False)
```

In [ ]:

```
# 默认的图形输出
sns.distplot(ccss.s3)
```

In [ ]:

```
# 和正态分布曲线相比较
from scipy.stats import norm
sns.distplot(ccss.s3, fit = norm)
```

In [ ]:

```
sns.distplot(ccss.s3, vertical = True)
```

In [ ]:

```
# 自定义图形元素细节
ax = sns.distplot(ccss.s3, rug = True,
                  rug_kws = {"color": "g"},
                  kde_kws = {"color": "k", "lw": 3, "label": "KDE"},
                  hist_kws = {"linewidth": 3, "color": "y",
                              "label" : "Hist"})
```

kdeplot命令

http://seaborn.pydata.org/generated/seaborn.kdeplot.html
(http://seaborn.pydata.org/generated/seaborn.kdeplot.html)

In [ ]:

```
# sns.set()

# 注意此处使用了Axes对象的叠加绘图功能
sns.kdeplot(ccss.s3, bw = .5, label = "bw: 0.5")
sns.kdeplot(ccss.s3, bw = 1, label = "bw: 1")
sns.kdeplot(ccss.s3, bw = 2, label = "bw: 2")
```

3.2.3 箱图

3.2.3.1 标准箱图

用matplotlib实现

In [ ]:

```
ccss.s3.plot.box()
```

In [ ]:

```
plt.boxplot(ccss.s3)
```

用seaborn实现

http://seaborn.pydata.org/generated/seaborn.boxplot.html
(http://seaborn.pydata.org/generated/seaborn.boxplot.html)

seaborn.boxplot(

```
    x, y, hue : names of variables in data or vector data, optional
    data : DataFrame, array, or list of arrays, optional
    orient : "v" | "h", optional
    color : matplotlib color, optional
    palette : palette name, list, or dict, optional

    saturation = 0.75 : float, 箱体颜色的饱和度
    width = 0.8 : float, 箱体宽度所占比例
    fliersize = 5 : float, 离群值散点大小
    linewidth = None : float, 框线宽度
    whis = 1.5 : float, 离群值确定标准，距离IQR上下界的倍数

)
```

In [ ]:

```
sns.boxplot(y = ccss.s3)
```

In [ ]:

```
sns.boxplot(y = ccss.index1, saturation = 0.3,
            width = 0.5, linewidth = 3, fliersize = 10)
```

3.2.3.2 增强箱图

对于大样本数据，箱图只显示IRQ和离群值，显然提供的信息不够丰富，因此可以考虑使用增强箱图，以提供更为丰富的百分位数信息。

增强箱图是在中位数两侧，除了绘制原有的四分位数以外，还继续向外绘制8分位数、16分位数、32分位数。。。以此类推，直至达到停止绘制的标准为止。
各箱体的宽度/颜色深度对应了相应的样本数量。
离群值的比例由用户自行设定。

seaborn.boxenplot(# 0.9以前版本中为lvplot，注意此处lv指的是letter-value！！！

```
    x, y, hue等参数

    k_depth = 'proportion' : 具体的箱体位置算法
        "proportion" | "tukey" | "trustworthy"
    scale = 'exponential' : 箱体宽度和样本量的对应关系
        "linear" 等比例减少
        "exponential" 对应箱体外侧未覆盖的样本比例
        "area" 对应箱体本身覆盖的样本比例
    outlier_prop = 0.007 : 离群值在样本中的比例

)
```

In [ ]:

```
# 函数有bug，不分组显示不完整
sns.boxenplot(x = ccss.time, y = ccss.index1)
```

In [ ]:

```
# 函数有bug，不分组显示不完整
sns.boxenplot(x = ccss.time, y = ccss.index1, outlier_prop = 0.1)
```

In [ ]:

```
# 函数有bug，不分组显示不完整
sns.boxenplot(x = ccss.time, y = ccss.index1,
              scale = 'area', outlier_prop = 0.1)
```

3.2.4 提琴图

提琴图结合了箱图、条带图和分布描述的KDE图，可根据数据特征做更清晰的信息呈现。

http://seaborn.pydata.org/generated/seaborn.violinplot.html
(http://seaborn.pydata.org/generated/seaborn.violinplot.html)

seaborn.violinplot(

```
    x=None, y=None, hue=None, data=None, order=None, hue_order=None

    bw = 'scott' : KDE图的具体计算方法或者核宽度
        {‘scott’, ‘silverman’, float}, optional
    cut = 2 : 绘图时是否超越两端的极值，设为0时严格限定在观测值范围内
    scale = 'area' : 各KDE图形设置为相同面积/按照频数设定宽度/相同宽度
        {“area”, “count”, “width”}, optional
    gridsize = 100 : 用于形成KDE曲线的散点数，越多越光滑
    width = 0.8 : 图形占的宽度比例
    inner = 'box' : 提琴图内部呈现的图形种类，箱图/分位数/原始数据。
        {“box”, “quartile”, “point”, “stick”, None}, optional
    saturation = 0.75 : 图形色彩饱和度。
```

)

In [ ]:

```
sns.violinplot(y = ccss.s3)
```

In [ ]:

```
sns.violinplot(y = ccss.s3, inner = "quartile")
```

In [ ]:

```
sns.violinplot(y = ccss.s3, inner = None)
```

In [ ]:

```
sns.violinplot(y = ccss.s3, inner = None)
sns.swarmplot(y = ccss.s3, size=2, color = "k")
```

3.3 实战练习

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)