第126天：Seaborn-可视化统计关系

原创 吴刀钓鱼 Python技术 2月13日

大数据一直是近几年来比较火爆的方向，作为想接触大数据的你，就不得不了解 seaborn。它是当下 Python 非常流行的数据可视化库，可以绘制出美观且有价值的图形。用一句话总结就是对于大数据从业人员来说，具备数据可视化的能力非常重要的，因为我们面对的客户或上司将更多地依赖于视觉提示，而不是复杂的机器学习模型。

1 前言

1.1 什么是 seaborn？

seaborn 是基于 matplotlib 库封装而成的一个数据可视化库。相比于 matplotlib 库，seaborn 的操作更加简单方便，具有更高级的接口。它使我们能够创建放大的数据视觉效果，帮助我们理解数据，通过在可视上下文中显示数据来发现变量或趋势之间的任何隐藏相关性，而这些相关性最初可能并不明显。

1.2 seaborn 环境搭建

个人建议安装 annoconda 软件，清华镜像 annoconda 下载地址，安装好即可以使用 seaborn 等一些库了，简单方便。

1.3 本文内容

本文将介绍如何用 seaborn 进行绘制可视化图形，主要采用以下三个函数：

- replot()，这是在绘图过程中最常用的一个函数，默认是绘制散点图。
- scatterplot()，顾名思义，这个函数作用是绘制散点图，作用类似 replot(kind="scatter")。
- lineplot()，绘制线图，作用类似 replot(kind="line")。

使用这些函数时可以通过色调、大小和样式的语义映射最多三个额外的变量来增强绘制的二维图形。简单来说就是函数的功能简单灵活，可以表示复杂的数据集结构。下面将通过例子逐一展示。

使用 seaborn 时一般需要 numpy、pandas 以及 matplotlib 库配合使用：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="darkgrid") # 设置 seaborn 的绘图样式，可以有 “darkgrid、whitegrid、dark、white 和 ticks” 样
```

2 散点图绘制

散点图是统计学中主要的可视化图形之一，由点汇聚成云图，每个点代表统计数据集中的一个观测值，通过云图可以推断出许多信息，进而判断两个变量之间是否存在一些有意义的联系。

在以下每个示例中，我们采用的是库里面自带的小费数据集 “tips”，通过以下代码来了解一下数据集的内容，可以看出 "tips" 包含了7个变量，分别是总账单、小费、顾客性别、是否吸烟、日期、吃饭时间以及顾客人数。

```
tips= sns.load_dataset("tips")
print('tips 数据集前十行数据:\n', tips.head(5))
print('每一列的数据类型:\n', tips.dtypes)
#输出结果:
#tips 数据集前十行数据:
#    total_bill   tip     sex smoker  day    time  size
#0       16.99  1.01  Female     No  Sun  Dinner     2
#1       10.34  1.66    Male     No  Sun  Dinner     3
#2       21.01  3.50    Male     No  Sun  Dinner     3
#3       23.68  3.31    Male     No  Sun  Dinner     2
#4       24.59  3.61  Female     No  Sun  Dinner     4
#每一列的数据类型:
# total_bill     float64
#tip            float64
#sex           category
#smoker        category
#day           category
#time          category
#size             int64
#dtype: object
```

2.1 示例1

```
# 输出 tips 数据集中以 total_bill 变量为 x 轴，tip 变量为 y 轴的散点图，注意 replot() 绘制出的图默认就是散点图
sns.relplot(x="total_bill", y="tip", data=tips)
```

根据可视化的图形可以大致得出顾客给的小费与总账单之间的一些关系，比如消费高的给的小费也越高。

2.2 示例2

虽然示例1中这些点以二维的形式描述了两个变量之间的联系，但还可以通过第三个变量对点进行着色来将另一个维度添加到绘图中。在 seaborn 中，这被称为使用“色调语义”，因为该点的颜色获得了意义：

```
# hue 参数用来输入定义色调语义的变量，即增加新的维度 smoker 变量
sns.relplot(x="total_bill", y="tip", hue="smoker", data=tips)
```

根据图上所示可知有3个维度的信息：total_bill、tip 以及 smoker(Yes:蓝色，No:橙色)。

2.3 示例3

为了强调类别之间的差异并提高辨识度，可以为每个类别使用不同的标记样式，引入 style 参数：

```
# style 参数用来输入定义样式的变量，这里仍然输入 smoker 变量
sns.relplot(x="total_bill", y="tip", hue="smoker", style="smoker", data=tips)
```

根据图上所示可知有3个维度的信息：total_bill、tip 以及 smoker(Yes:蓝色圆圈，No:橙色叉叉)。

2.4 示例4

可视化图形可以表示四个维度的信息，通过单独改变每个点的色调和样式来表示四个变量。但是人眼对形状的敏感度远低于对颜色的敏感度，建议谨慎使用:

```
# style 参数输入 time 变量
sns.relplot(x="total_bill", y="tip", hue="smoker", style="time", data=tips)
```

图中表示了四个维度的信息：total_bill、tip、smoker(Yes:蓝色，No:橙色)以及 time(Lunch:圆圈，Dinner:叉叉)。

2.5 示例5

在上面的例子中，因为 smoker 变量表示的是类别，所以色调语义 hue 参数在可视化图形中以两种颜色区分表示类别，即使用了默认的定性调色板。如果色调语义 hue 参数填入的变量表示的是数值，则默认的颜色会切换到顺序调色板:

```
# size 变量表示的是数值大小，不再是类别
sns.relplot(x="total_bill", y="tip", hue="size", data=tips)
```

图中通过颜色深浅表示每个点代表的顾客人数大小，颜色越深表示的顾客人数越多。

2.6 示例6

通过引入 size 参数，来改变每个点的大小：

```
#
sns.relplot(x="total_bill", y="tip", size="size", data=tips)
```

图中点的形状越大代表的顾客人数越多。

我们还可以通过引入 sizes 参数，来表示每个点的大小变化范围：

```
# sizes 中表示每个点的面积大小，最小值15，最大值200
sns.relplot(x="total_bill", y="tip", size="size", sizes=(15, 200), data=tips)
```

3 线图绘制

虽然散点图是一种非常有效的数据可视化图形，但他并不是对所有数据都通用的。绘制可视化图形时，应该根据数据集的内容以及试图用图形回答的问题。针对某些数据集，我们可能希望了解一个变量的随时间的变化关系。在这种情况下，根据数据集来绘制线图是一个不错的选择。在 seaborn 中，我们可以通过 lineplot() 函数直接绘制线图，也可以通过设置 relplot() 的参数 kind="line" 来实现。

3.1 示例1

```
# 定义一个数据集 df，绘制 value 变量关于 time 变量的函数
df = pd.DataFrame(dict(time=np.arange(500), value=np.random.randn(500).cumsum()))
sns.relplot(x="time", y="value", kind="line", data=df)
```

```
# 想要将 y 绘制为 x 的函数，默认行为是在绘制之前按数字 x 对数据进行排序。但是，可以通过设置 sort 参数值来禁用：
df = pd.DataFrame(np.random.randn(500, 2).cumsum(axis=0), columns=["x", "y"])
sns.relplot(x="x", y="y", sort=False, kind="line", data=df)
```

3.2 示例2

有一些复杂的数据集将对 x 变量的相同值有多个观测值。seaborn 的默认行为是通过绘制平均值及 95% 的置信区间，在每个 x 周围聚合多个测量值。

示例中我们采用的是库里面自带的数据集 "fmri"，通过以下代码来了解一下数据集的内容，可以看出 "fmri" 包含了5个变量，分别是subject、timepoint、event、region 和 signal。

```
fmri = sns.load_dataset("fmri")
print(fmri)

# 输出结果：
```

```
#     subject  timepoint event    region    signal
#0        s13         18  stim  parietal -0.017552
#1         s5         14  stim  parietal -0.080883
#2        s12         18  stim  parietal -0.081033
#3        s11         18  stim  parietal -0.046134
#4        s10         18  stim  parietal -0.037970
#     ...        ...   ...       ...       ...
#1059      s0          8   cue   frontal  0.018165
#1060     s13          7   cue   frontal -0.029130
#1061     s12          7   cue   frontal -0.004939
#1062     s11          7   cue   frontal -0.025367
#1063      s0          0   cue  parietal -0.006899
```

```
# 绘制 signal 变量关于 timepoint 变量的函数，包括 95% 的置信区间
sns.relplot(x="timepoint", y="signal", kind="line", data=fmri)
```

```
# 可以设置 ci 的参数禁用显示置信区间
sns.relplot(x="timepoint", y="signal", ci=None, kind="line", data=fmri)
```

```
# 可以通过绘制标准差，而不是置信区间来表示分布在每个时间点的分布范围
sns.relplot(x="timepoint", y="signal", kind="line", ci="sd", data=fmri)
```

```
# 可以通过设置estimator参数为None，来完全停用标准差与置信区间的显示。当数据在每个点上有多个观察值时，会产生奇怪可
sns.relplot(x="timepoint", y="signal", estimator=None, kind="line", data=fmri)
```

```
# 添加色调语义参数 hue，每个曲线都着色以指示它们对应于 event。
sns.relplot(x="timepoint", y="signal", hue="event", kind="line", data=fmri)
```

```
# 在线条图中添加 style 参数，默认情况下会用短划线的形式来区分不同的 style
sns.relplot(x="timepoint", y="signal", hue="region", style="event", kind="line", data=fmri)
```

```
# 可以通过设置参数 dashes 和 markers，来确定是否启用短划线或者标记来辨别不同的 style
sns.relplot(x="timepoint", y="signal", hue="region", style="event", dashes=False, markers=True, kind=
```

```
# 与散点图一样，要谨慎使用多个参数来制作线图。虽然做出来的图形提供了丰富的信息，但它们也很难解析和解释。如果我们只需
sns.relplot(x="timepoint", y="signal", hue="event", style="event", kind="line", data=fmri)
```

```
# 当一个 x 变量对应多个采样数据时，可以单独给每个采样单位绘制曲线，采用 units 参数
sns.relplot(x="timepoint", y="signal", hue="region", units="subject", estimator=None, kind="line", da
```

3.3 示例3

本示例中我们采用的是库里面自带的数据集 “dots”，通过以下代码来了解一下数据集的内容，可以看出 "dots" 包含了5个变量，分别是align、choice、time、coherence 和 firing_rate。

```
dots = sns.load_dataset("dots").query("align == 'dots'")
print(dots)

# 输出结果：
#     align choice  time  coherence  firing_rate
# 0    dots     T1   -80        0.0    33.189967
# 1    dots     T1   -80        3.2    31.691726
# 2    dots     T1   -80        6.4    34.279840
# 3    dots     T1   -80       12.8    32.631874
# 4    dots     T1   -80       25.6    35.060487
# ..    ...    ...   ...        ...          ...
# 389  dots     T2   680        3.2    37.806267
# 390  dots     T2   700        0.0    43.464959
# 391  dots     T2   700        3.2    38.994559
# 392  dots     T2   720        0.0    41.987121
# 393  dots     T2   720        3.2    41.716057
```

```
# 之前的示例中我们添加的参数 hue 中定义的变量都是类别，接下来我们通过一个例子来看当该参数定义的变量是数值时会产生什
sns.relplot(x="time", y="firing_rate",
            hue="coherence", style="choice",
            kind="line", data=dots)
```

如图所示，coherence 中以四个层级深浅不一的颜色表示该变量数值大小。

```
# 如上个示例所示，即使 hue 变量是数值，它也很难用线性色标表示全。这时，我们可以通过传递列表或字典为每一数值提供特定
palette = sns.cubehelix_palette(n_colors=6) # 数据集中 coherence 变量有6个数值，所以 n_colors=6
```

```
palette = sns.cubehelix_palette(n_colors=6)  # 数据集中 coherence 变量有6个数值，所以 n_colors=6
sns.relplot(x="time", y="firing_rate", hue="coherence", style="choice", palette=palette, kind="line",
```

```
# 当 hue 参数的变量是数值时，可以通过添加 hue_norm 参数用于将 colormap 标准化
from matplotlib.colors import LogNorm
sns.relplot(x="time", y="firing_rate",
            hue="coherence", style="choice",
            hue_norm=LogNorm(),
            kind="line", data=dots)
```

```
# 可以通过添加 size 参数来改变线的粗细大小
sns.relplot(x="time", y="firing_rate",
            size="coherence", style="choice",
            kind="line", data=dots)
```

```
# 虽然 size 变量通常是数值型的，但是也可以用线宽映射为类别变量。在这样做的时候要小心，因为除了“粗”线和“细”线之外，
sns.relplot(x="time", y="firing_rate",
           hue="coherence", size="choice",
           kind="line", data=dots)
```

3.4 示例4

线图通常用于可视化与实际日期和时间相关的数据。

```
df = pd.DataFrame(dict(time=pd.date_range("2017-1-1", periods=500),
                       value=np.random.randn(500).cumsum()))
g = sns.relplot(x="time", y="value", kind="line", data=df)
```

如下图所示，x 坐标产生乱码，影响了图形的可读性。

```
# 我们可以通过添加 g.fig.autofmt_xdate() 来解决 x 坐标的乱码问题
g.fig.autofmt_xdate()
```

4 显示多图-格点图

因为 relplot() 是基于 FacetGrid，所以很容易显示附加变量的影响，而不是将其分配给图中的一个语义角色。这意味着可以创建多个轴并在每个轴上绘制数据的子集。

4.1 示例

```
# 以 tips 数据集为例，添加 col 参数为 time 变量，在行上按照 time 变量进行分列
tips = sns.load_dataset("tips")
sns.relplot(x="total_bill", y="tip", hue="smoker",
            col="time", data=tips)
```

```
# 以 fmri 数据集为例，显示两个变量的影响：一个是通过 col，而另一个是通过 row。同时可以通过 height 参数改变图形的
fmri = sns.load_dataset("fmri")
sns.relplot(x="timepoint", y="signal", hue="subject",
            col="region", row="event", height=3,
            kind="line", estimator=None, data=fmri)
```

```
# 以 fmri 数据集为例，在行上按照 subject 变量进行分列，通过 col_wrap=5 设置每行五个分类，aspect 用来设置图形的
fmri = sns.load_dataset("fmri")
sns.relplot(x="timepoint", y="signal", hue="event", style="event",
            col="subject", col_wrap=5,
            height=3, aspect=.75, linewidth=2.5,
            kind="line", data=fmri.query("region == 'frontal'"))
```

总结

本节给大家介绍了 seaborn 中关于散点图、线图以及格点图的绘制方式，助你掌握统计数据中各变量间关系可视化表示的一些基本方法。

参考资料

[1] https://seaborn.pydata.org/tutorial/relational.html

[2] https://www.cntofu.com/book/172/docs/3.md

> 示例代码：Python-100-days

系列文章

第125天：Flask 项目结构
第124天：Web 开发 Django 模板
第123天：Web 开发 Django 管理工具
第122天：Flask 单元测试
第121天：机器学习之决策树
从 0 学习 Python 0 - 120 大合集总结

PS：公号内回复 ：Python，即可进入Python 新手学习交流群，一起**100天计划！**

-END-

Python 技术

关于 Python 都在这里