第127天：Seaborn-可视化分类数据

原创 吴刀钓鱼 Python技术 2月14日

上一篇我们介绍了可视化表示数据集中各变量间关系的基本方法。在示例中，我们专注于两个数值变量之间的主要关系。如果其中一个主要变量是“可分类的”（能被分为不同的组），那么我们可以使用更专业的可视化方法，即本篇将介绍 seaborn 中分类数据的可视化图形表示。

1 前言

在 seaborn 中，有几种不同的方法可以对分类数据进行可视化。在这里我们将不同的分类图类型视为三个不同的家族，下面我们将详细讨论，它们是分类散点图、分类分布图以及分类估计图，我们应该根据实际情况来决定到底要使用哪个。它们有统一的 API，所以我们可以轻松地在不同类型之间进行切换，并从多个角度来观察数据。

在文中，我们主要关注 catplot() 函数。这个函数是接下来介绍的每个函数更高级别的接口，因此当我们显示每种绘图时都会引用它们。以下各个小节示例演示前均首先进行以下声明：

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="darkgrid")
```

2 分类散点图

在 catplot() 中，数据默认使用散点图表示。在 seaborn 中有两种不同的分类散点图，它们采用不同的方法来表示分类数据，分别是：

- stripplot() (with kind="strip"; the default)
- swarmplot() (with kind="swarm")

2.1 示例-stripplot()

stripplot() 函数是 catplot() 中 kind 的默认参数，它的作用是将属于一个类别的所有点，沿着与分类变量对应的轴落在相同位置，同时用少量随机“抖动”调整分类轴上的点的位置。

```
tips = sns.load_dataset("tips")
sns.catplot(x="day", y="total_bill", data=tips)
```

```
# 我们可以通过 jitter 参数控制抖动的大小，也可以完全禁用它
sns.catplot(x="day", y="total_bill", jitter=False, data=tips)
```

```
2
```

2.2 示例-swarmplot()

同样可以使用 seaborn 中的 swarmplot() 函数来绘制分类散点图，在 catplot() 中设置 kind="swarm" 来激活该功能，该函数的作用是防止每个点的重叠，以用它更好地表示观测分布，但是只适用于相对较小的数据集。

```
sns.catplot(x="day", y="total_bill", kind="swarm", data=tips)
```

```
# 可以通过使用 hue 参数向分类图添加另一个变量，丰富图形的内容
sns.catplot(x="day", y="total_bill", hue="sex", kind="swarm", data=tips)
```

```
# 如果传递给分类轴的变量看起来是数字，则将对数字进行排序
sns.catplot(x="size", y="total_bill", kind="swarm", data=tips)
```

```
# 可以使用 order 参数在特定图表的基础上控制排序顺序
sns.catplot(x="size", y="total_bill", order=[6, 5, 4, 2, 1, 3], kind="swarm", data=tips)
```

```
# 在以上示例中，它始终对应于水平轴。但是也可以将分类变量放在垂直轴上
sns.catplot(x="total_bill", y="day", hue="time", kind="swarm", data=tips)
```

3 分类分布图

随着数据集的大小增加，分类散点图中每个类别可以提供的值分布信息受到限制。当发生这种情况时，有几种方法可以总结分布信息，以便于我们可以跨分类级别进行简单比较，分别是：

- boxplot() (with kind="box")
- boxenplot() (with kind="boxen")
- violinplot() (with kind="violin")

3.1 示例-boxplot()

boxplot()，这就是箱型图，绘制方法是找出一组数据的上边缘、下边缘、中位数和两个四分位数；然后，连接两个四分位数画出箱体；再将上边缘和下边缘与箱体相连接，中位数在箱体中间。“胡须”延伸到位于，超出下四分位数和上四分位数的1.5 IQR 内的点会独立显示，称之为异常值。

```
sns.catplot(x="day", y="total_bill", kind="box", data=tips)
```

```
# 可以添加 hue 参数，添加的变量每个类别的箱型图沿着分类轴移动，它们将不会重叠，这个行为称为 "dodging"，默认开启的
sns.catplot(x="day", y="total_bill", hue="smoker", kind="box", data=tips)
```

```
# 如果把 "dodging" 功能关闭，箱型图就会产生重叠现象
sns.catplot(x="day", y="total_bill", hue="smoker", kind="box", dodge=False, data=tips)
```

```
# 有语义变量嵌套在主分类变量中，我们才会开启 "dodging"，否则可以禁用 "dodging"
tips["weekend"] = tips["day"].isin(["Sat", "Sun"])  # 新增变量 "weekend"，该变量并未嵌套在主分类 "day" 变
sns.catplot(x="day", y="total_bill", hue="weekend",
            kind="box", dodge=False, data=tips)
```

3.2 示例-boxenplot()

boxenplot() 函数可以绘制一个与 boxplot() 函数类似的图，但是它为了显示更多信息而对分布的形状进行了优化，比较适合于较大的数据集。

我们采用 Python 自带的数据集 "diamonds"，收录50225颗钻石的价格和其他属性的数据集。

```
diamonds = sns.load_dataset("diamonds")
print(diamonds)
# 输出结果
#        carat        cut color clarity  depth  table  price     x     y     z
# 0       0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
# 1       0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
# 2       0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
# 3       0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
# 4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
#      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
# 50220   0.70  Very Good     F     SI1   61.7   61.0   2230  5.67  5.70  3.51
# 50221   0.70  Very Good     F     SI1   61.6   55.0   2230  5.66  5.70  3.50
# 50222   0.52      Ideal     E    VVS2   62.0   54.0   2230  5.16  5.19  3.21
# 50223   0.53      Ideal     E    VVS2   62.0   58.0   2230  5.16  5.20  3.21
```

```
# 50224   0.56      Ideal     G     VS1   61.0   56.0   2230  5.00   NaN   NaN
```

```
sns.catplot(x="color", y="price", kind="boxen",
            data=diamonds.sort_values("color"))
```

3.3 示例-violinplot()

最后我们来看一下 violinplot() 函数绘制的图形，我们可以称之为小提琴图。小提琴图是用来展示多组数据的分布状态以及核密度。跟箱型图类似，但是在密度层面展示的更好。在数据量非常大且不方便一个一个展示的时候小提琴图特别适用。

```
sns.catplot(x="total_bill", y="day", hue="time",
            kind="violin", data=tips)
```

```
# 小提琴图中显示了来自箱型图的四分位数和上下边缘值，缺点是我们需要调整一些额外参数，与箱形图相比增加了一些复杂性
sns.catplot(x="total_bill", y="day", hue="time",
            kind="violin", bw=.15, cut=0,  # bw:用来计算核密度的带宽，cut:设置为0将小提琴图范围限制在观察数
            data=tips)
```

```
# 当 hue 参数添加的变量只有两个类别时，将 split 设置为 True 则会为每种颜色绘制对应半边小提琴，这样可以更容易直接
sns.catplot(x="day", y="total_bill", hue="sex",
            kind="violin", split=True, data=tips)
```

```
# 将 inner 设置为 stick 则显示具体数据点或数据线，而不是摘要箱型图值
sns.catplot(x="day", y="total_bill", hue="sex",
            kind="violin", inner="stick", split=True,
            palette="pastel", data=tips)
```

```
# 可以将 swarmplot() 或 striplot() 与箱型图或小提琴图结合起来，展示每次观察以及分布摘要
g = sns.catplot(x="day", y="total_bill", kind="violin", inner=None, data=tips)
sns.swarmplot(x="day", y="total_bill", color="k", size=3, data=tips, ax=g.ax)  # ax:绘图时使用 g.ax 作
```

4 分类估计图

对于一些数据集，你可能希望显示值的集中趋势估计，而不是显示每个类别中的分布。Seaborn 有以下三种方式来显示这些信息：

- barplot() (with kind="bar")
- countplot() (with kind="count")
- pointplot() (with kind="point")

接下来我们使用 Python 自带的数据集 "titanic"，如下所示。

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

# 输出结果：
#      survived  pclass     sex   age  ...  deck  embark_town  alive  alone
# 0           0       3    male  22.0  ...   NaN  Southampton     no  False
# 1           1       1  female  38.0  ...     C    Cherbourg    yes  False
# 2           1       3  female  26.0  ...   NaN  Southampton    yes   True
# 3           1       1  female  35.0  ...     C  Southampton    yes  False
# 4           0       3    male  35.0  ...   NaN  Southampton     no   True
# ..        ...     ...     ...   ...  ...   ...          ...    ...    ...
# 886         0       2    male  27.0  ...   NaN  Southampton     no   True
# 887         1       1  female  19.0  ...     B  Southampton    yes   True
# 888         0       3  female   NaN  ...   NaN  Southampton     no  False
# 889         1       1    male  26.0  ...     C    Cherbourg    yes   True
# 890         0       3    male  32.0  ...   NaN   Queenstown     no   True
```

4.1 示例-barplot()

使用 barplot() 函数可以绘制条形图。在 seaborn 中，barplot() 函数在完整数据集上运行并应用函数来获取估计值（默认情况下取平均值）。当每个类别中有多个观察值时，它还使用自举来计算估计值周围的置信区间，并使用误差条绘制。

```
sns.catplot(x="sex", y="survived", hue="class", kind="bar", data=titanic)
```

4.2 示例-countplot()

条形图的一个特例是，当你想要显示每个类别中的观察数量而不是计算第二个变量的统计数据时。在 seaborn 中，使用 countplot() 函数很容易实现。

```
sns.catplot(x="deck", kind="count", palette="ch:.25", data=titanic)
```

```
# 也可以将 count 放置在 x 轴上
sns.catplot(y="deck", hue="class", kind="count",
            palette="pastel", edgecolor=".6",
            data=titanic)
```

4.3 示例-pointplot()

pointplot() 函数提供了另一种可视化相同信息的样式。此函数还对另一个轴上的高度估计值进行编码，但不是显示一个完整的条形图，而是绘制点估计值和置信区间。另外，pointplot() 连接来自相同 hue 类别的点。我们可以很容易的看出主要关系如何随着色调语义的变化而变化，因为人类的眼睛很擅长观察斜率的差异。

```
sns.catplot(x="sex", y="survived", hue="class", kind="point", data=titanic)
```

```
# 可以调整 markers 和 linestyles 与色调一起改变，以制作最大可访问的图形
sns.catplot(x="class", y="survived", hue="sex",
            palette={"male": "g", "female": "m"},
            markers=["^", "o"], linestyles=["-", "--"],
            kind="point", data=titanic)
```

5 绘制“宽格式”数据

虽然优选使用“长形式”或“整齐”数据，但这些函数也可以应用于各种“宽格式”的数据，包括 pandas DataFrames 或二维 numpy 数组。这些对象应该直接传递给 data 参数。

```
iris = sns.load_dataset("iris")
sns.catplot(data=iris, orient="h", kind="box")
```

```
# 这些函数接受 Pandas 或 numpy 对象的向量，而不是 DataFrame 中的变量
sns.violinplot(x=iris.species, y=iris.sepal_length)
```

```
# 为了控制由上述功能制作的图形的大小和形状，您必须使用matplotlib命令自己设置图形。当然，这也意味着这些图块可以和其
```

```
f, ax = plt.subplots(figsize=(7, 3))
sns.countplot(y="deck", data=titanic, color="c")
```

6 显示多图

就像 relplot() 一样, catplot() 也是建立在 FacetGrid 上，这意味着很容易添加层面变量来可视化高维关系。

```
sns.catplot(x="day", y="total_bill", hue="smoker",
            col="time", aspect=.6,
            kind="swarm", data=tips)
```

```
# 要进一步自定义绘图，我们可以使用它返回的 FacetGrid 对象上的方法
g = sns.catplot(x="fare", y="survived", row="class",
                kind="box", orient="h", height=1.5, aspect=4,
                data=titanic.query("fare > 0"))
g.set(xscale="log")
```

总结

本节给大家介绍了 seaborn 中关于分类数据可视化的一些方法，主要包含了三个方面的内容，分别是分类散点图、分类分布图以及分类估计图。

参考资料

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

[2] https://github.com/apachecn/seaborn-doc-zh/blob/master/docs/4.md

> 示例代码：Python-100-days

系列文章

第126天：Seaborn-可视化统计关系
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