第128天：Seaborn-可视化数据集的分布

原创 吴刀钓鱼 Python技术 2月17日

上一篇我们介绍了介绍 seaborn 中分类数据的可视化图形表示。在处理数据过程中，我们通常想做的第一件事就是了解数据集中变量的分布情况，本篇将介绍 seaborn 中数据集分布情况的可视化图形表示。

1 前言

在这里我们将简要介绍 seaborn 中用于检查单变量和双变量分布的一些函数。

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
from scipy import stats
sns.set(color_codes=True)
```

2 绘制单变量分布

在 seaborn 中想要快速查看单变量分布的最方便的方法是使用 distplot() 函数，可以灵活绘制单变量观测值分布图。

2.1 示例1

默认情况下，distplot() 函数会绘制直方图并拟合内核密度估计(kernel density estimate，简称 KDE)。

```
x = np.random.normal(size=100)
sns.distplot(x)
```

2.2 示例2

直方图的绘制首先确定数据区间，然后观察数据落入这些区间中的数量来绘制柱形图以此来表征数据的分布情况。我们可以通过 kde=False 删除密度曲线，然后添加一个 rug=True，它会在横轴上为每一个观测值绘制垂直竖线。

```
x = np.random.normal(size=100)
sns.distplot(x, kde=False, rug=True)
```

```
# 可以通过 bins 来调整直方图中 bin 的数目，若填 None，则默认使用 Freedman-Diaconis 规则指定柱的数目
x = np.random.normal(size=100)
sns.distplot(x, bins=20, kde=False, rug=True)  # 尝试更多或更少的柱数目可能会揭示数据中的其他特性
```

2.3 示例3

核密度估计(KDE)是绘制分布形状的有力工具，它是在概率论中用来估计未知的密度函数。可以通过 hist=False 来禁用直方图的绘制，然后只绘制核密度估计。

```
x = np.random.normal(size=100)
sns.distplot(x, hist=False, rug=True)
```

下面介绍一下如何绘制核密度估计曲线，这比直方图绘制更复杂。首先每个观测值被一个以该值为中心的正态(高斯)曲线所取代。

```
x = np.random.normal(0, 1, size=30)  # 初始化一组服从正态分布的随机数
bandwidth = 1.06 * x.std() * x.size ** (-1 / 5.)  # 根据经验公式计算 KDE 的带宽
support = np.linspace(-4, 4, 200)

kernels = []
for x_i in x:

    kernel = stats.norm(x_i, bandwidth).pdf(support)  # 获取每一个观测值的核密度估计
    kernels.append(kernel)
    plt.plot(support, kernel, color="r") # 为每一个观测值绘制核密度估计曲线

sns.rugplot(x, color=".2", linewidth=3)
```

接下来，对这些曲线进行求和，计算支持网格(support grid)中每个点的密度值。然后对得到的曲线进行归一化，使曲线下的面积等于1。

```
from scipy.integrate import trapz
density = np.sum(kernels, axis=0)
density /= trapz(density, support)
plt.plot(support, density)
```

我们可以看到，如果在 seaborn 中使用 kdeplot() 函数， 我们可以得到相同的曲线。这个函数也被 distplot() 所使用, 但是当我们只想要核密度估计时，它提供了一个更直接的接口，可以更容易地访问其他选项。

```
sns.kdeplot(x, shade=True)
```

KDE 的带宽参数控制估计与数据的拟合程度，就像直方图中的 bin 大小一样。它对应于我们在上面绘制的内核的宽度。默认行为尝试使用常用参考规则猜测一个好的值，但尝试更大或更小的值可能会有所帮助。

```
sns.kdeplot(x)
sns.kdeplot(x, bw=.2, label="bw: 0.2")
sns.kdeplot(x, bw=2, label="bw: 2")
plt.legend()
```

正如我们在上面所看到的，高斯 KDE 过程的本质意味着估计超出了数据集中最大和最小的值，有可能控制超过极值多远的曲线是由 cut 参数控制的，然而这只影响曲线的绘制方式，而不影响曲线的拟合方式。

```
sns.kdeplot(x, shade=True, cut=0)
sns.rugplot(x)
```

2.4 示例4

可以使用 distplot() 函数将参数分布拟合到数据集上，并直观地评估其与观测数据的对应程度，这就是拟合参数分布。

```
x = np.random.gamma(6, size=200)
sns.distplot(x, kde=False, fit=stats.gamma)
```

3 绘制二元分布

对于可视化两个变量的二元分布也很有用。在 seaborn 中，最简单的方法就是使用jointplot()函数，它创建了一个多面板图形，显示了两个变量之间的二元(或联合)关系，以及每个变量在单独轴上的一元(或边际)分布。

接下来以下面的数据集为例展示例子：

```
mean, cov = [0, 1], [(1, .5), (.5, 1)]
data = np.random.multivariate_normal(mean, cov, 200)
df = pd.DataFrame(data, columns=["x", "y"])
```

```
df = pd.DataFrame(data, columns=["x", "y"])
```

3.1 示例1-散点图

可视化二元分布最常见的方法是散点图，其中每个观察点都以 x 和 y 值表示。这类似于二维 rugplot。您可以使用 matplotlib 的plt.scatter 函数绘制散点图, 它也是 jointplot() 函数显示的默认类型的图。

```
sns.jointplot(x="x", y="y", data=df)
```

3.2 示例2-六边形“桶”(Hexbin)图

类似于单变量的直方图，用于描绘二元变量关系的图称为 “hexbin” 图,因为它显示了落入六边形“桶”内的观察计数。此图对于相对较大的数据集最有效。它可以通过调用 matplotlib 中的 plt.hexbin 函数获得并且在 jointplot() 作为一种样式。当使用白色作为背景色时效果最佳。

```
x, y = np.random.multivariate_normal(mean, cov, 1000).T
with sns.axes_style("white"):
    sns.jointplot(x=x, y=y, kind="hex", color="k")
```

3.3 示例3-核密度估计

也可以使用上面描述的核密度估计过程来可视化二元分布。在 seaborn 中，这种图用等高线图表示， 在 jointplot() 中被当作一种样式。

```
sns.jointplot(x="x", y="y", data=df, kind="kde")
```

还可以使用 kdeplot() 函数绘制二维核密度图。这允许您在一个特定的(可能已经存在的) matplotlib 轴上绘制这种图，而 jointplot() 函数能够管理它自己的图。

```
f, ax = plt.subplots(figsize=(6, 6))
sns.kdeplot(df.x, df.y, ax=ax)
sns.rugplot(df.x, color="g", ax=ax)
sns.rugplot(df.y, vertical=True, ax=ax)
```

如果希望更连续地显示双变量密度，可以简单地增加轮廓层的数量。

```
f, ax = plt.subplots(figsize=(6, 6))
cmap = sns.cubehelix_palette(as_cmap=True, dark=0, light=1, reverse=True)
sns.kdeplot(df.x, df.y, cmap=cmap, n_levels=60, shade=True)
```

jointplot() 函数使用 JointGrid 来管理图形。为了获得更大的灵活性，您可能想直接使用 JointGrid 来绘制图形。jointplot() 在绘图后返回 JointGrid 对象，您可以使用它添加更多图层或调整可视化的其他方面。

```
g = sns.jointplot(x="x", y="y", data=df, kind="kde", color="m")
g.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")
g.ax_joint.collections[0].set_alpha(0)
g.set_axis_labels("$X$", "$Y$")
```

4 可视化数据集中的成对关系

要在数据集中绘制多个成对的双变量分布，我们可以使用 pairplot() 函数。这将创建一个轴矩阵并显示 DataFrame 中每对列的关系，默认情况下，它还绘制对角轴上每个变量的单变量分布。

```
iris = sns.load_dataset("iris")
sns.pairplot(iris)
```

与 jointplot() 和 JointGrid 之间的关系非常类似， pairplot() 函数构建在 PairGrid 对象之上, 可以直接使用它来获得更大的灵活性。

```
g = sns.PairGrid(iris)
g.map_diag(sns.kdeplot)
g.map_offdiag(sns.kdeplot, n_levels=6)
```

总结

本节给大家介绍了 seaborn 中关于可视化数据集分布的一些方法，主要包含了三个方面的内容，分别是绘制绘制单变量分布、绘制二元分布以及可视化数据集中的成对关系。

参考资料

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

[2] https://github.com/apachecn/seaborn-doc-zh/blob/master/docs/5.md

> 示例代码：https://github.com/JustDoPython/python-100-day

系列文章

第127天：Seaborn-可视化分类数据
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