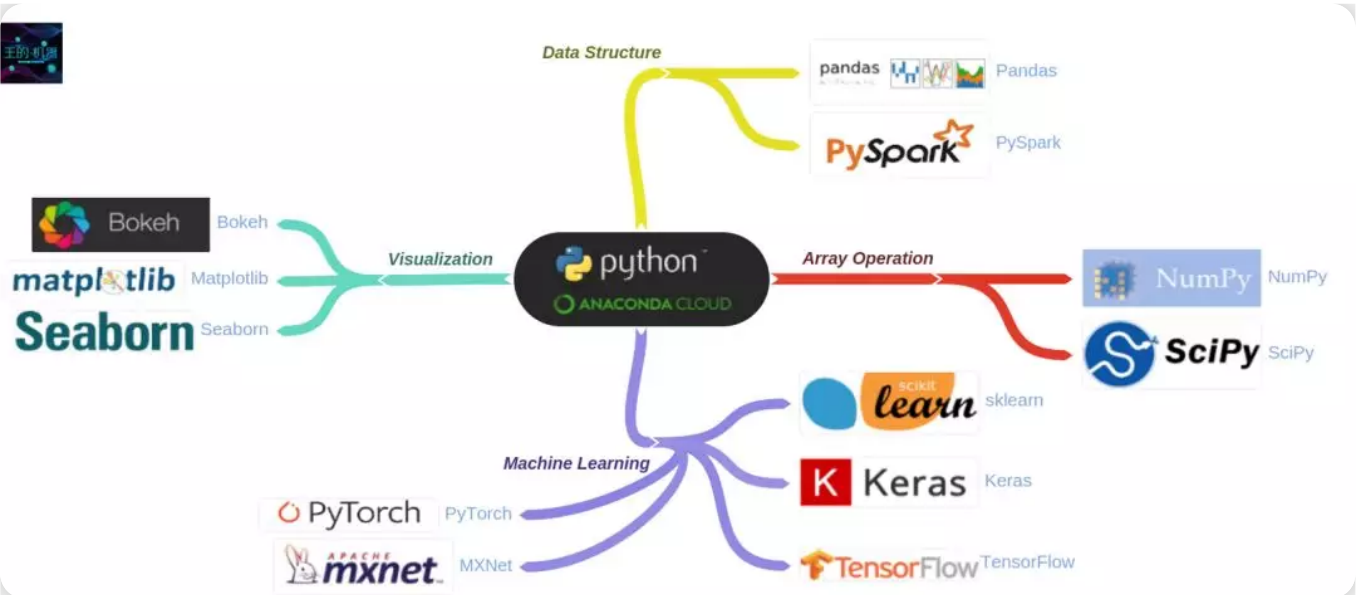


盘一盘 Python 系列 10 - Cufflinks

原创 王圣元 王的机器 4月12日

来自专辑
Python



本文含 3886 字，8 图表截屏
建议阅读 20 分钟

对在新加坡的读者
末尾有彩蛋

0 引言

本文是盘一盘 Python 系列的第十四篇，Cufflinks 篇

- Python 入门篇 (上)
- Python 入门篇 (下)
- 数组计算之 NumPy (上)
- 数组计算之 NumPy (下)
- 科学计算之 SciPy

- 数据结构之 Pandas (上)
- 数据结构之 Pandas (下)
- 基本可视化之 Matplotlib
- 统计可视化之 Seaborn
- 炫酷可视化之 PyEcharts (v0.5)
- 炫酷可视化之 PyEcharts (v1.0)
- **交互可视化之 Cufflinks**
- 机器学习之 Sklearn
- 机学可视化之 Scikit-Plot
- 深度学习之 Keras (上)

Cufflinks 是一个可视化的库，可以无缝衔接 **pandas** 和 **plotly**，**前者** 中的 dataframe 在数据分析中无处不在，**后者** 的交互式让可视化又上一个台阶。Cufflinks 连接了两者，必须要了解一下。

先引入可能需要的包。

```
1 import os
2 import pandas as pd
3 import numpy as np
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 %matplotlib inline
```

引入 cufflinks 包，并起别名为 cf。

```
1 import cufflinks as cf
```

选择离线模式生成本地图片，即不会上传到它们的系统上。

```
1 cf.go_offline()
```












熟悉我的风格就知道我偏好的颜色，它们的 RGB 设定如下：

```
1 color = [ 'rgb(220,38,36)', 'rgb(43,71,80)', 'rgb(69,160,162)',
2           'rgb(232,122,89)', 'rgb(125,202,169)', 'rgb(100,158,125)',
```

```

3      'rgb(220,128,24)', 'rgb(200,159,145)', 'rgb(108,109,108)',
4      'rgb(79,98,104)', 'rgb(199,204,207)' ]

```

	Red:	RGB = 220, 38, 36	Hex = #dc2624
	Dark Teal:	RGB = 43, 71, 80	Hex = #2b4750
	Teal:	RGB = 69, 160, 162	Hex = #45a0a2
	Red:	RGB = 232, 122, 89	Hex = #e87a59
	Teal:	RGB = 125, 202, 169	Hex = #7dcaa9
	Green:	RGB = 100, 158, 125	Hex = #649e7d
	Orange:	RGB = 220, 128, 24	Hex = #dc8018
	Tan:	RGB = 200, 159, 145	Hex = #c89f91
	Grey-50:	RGB = 108, 109, 108	Hex = #6c6d6c
	Blue Grey:	RGB = 79, 98, 104	Hex = #4f6268
	Grey-25:	RGB = 199, 204, 207	Hex = #c7cccf

Cufflinks 里绘图函数就是 `df.iplot`，一招鲜吃遍天，但是 `iplot` 函数里的参数很多，一些参数说明如下：

- `kind`: 图的种类，如 `scatter`、`pie`、`histogram` 等
- `mode`: `lines`、`markers`、`lines+markers`，分别表示折线、点、折线和点
- `colors`: 轨迹对应的颜色
- `dash`: 轨迹对应的虚实线，`solid`、`dash`、`dashdot` 三种
- `width`: 轨迹的粗细
- `xTitle`: 横坐标名称
- `yTitle`: 纵坐标的名称
- `title`: 图表的标题

现在学习 Cufflinks（或学习任何新东西），我已经不会一开始就无脑看和其相关所有内容，没时间也没精力，我只会在处理具体任务需要用到 Cufflinks 再去查找相应知识点，完成任务就行了。

我们今天的任务就是可视化一个信用组合。

样本组合

考虑一个样本组合 (sample portfolio)，它包含 100 个不同的借贷人，有如下三个假设：

1. 组合的总规模为 1000，意味着平均每个借贷人的敞口 (exposure) 为 10。
2. **实际敞口**是根据韦伯分布 (Weibull) 模拟得出，范围从小于 1 到 50。
3. 借贷人的**无条件违约概率** (unconditional default probability) 根据卡方分布 (chi-square) 模拟得出，均值设为 1%。

我模拟好违约率和敞口存成两个 numpy 格式文件，加载存储成变量 p 和 c，N 为借贷人数，100。

```
1 dpFile = os.getcwd() + '\\defaultProbabilties.npy'
2 expFile = os.getcwd() + '\\exposures.npy'
3
4 c = np.load(expFile)
5 p = np.load(dpFile)
6 N = len(c)
```

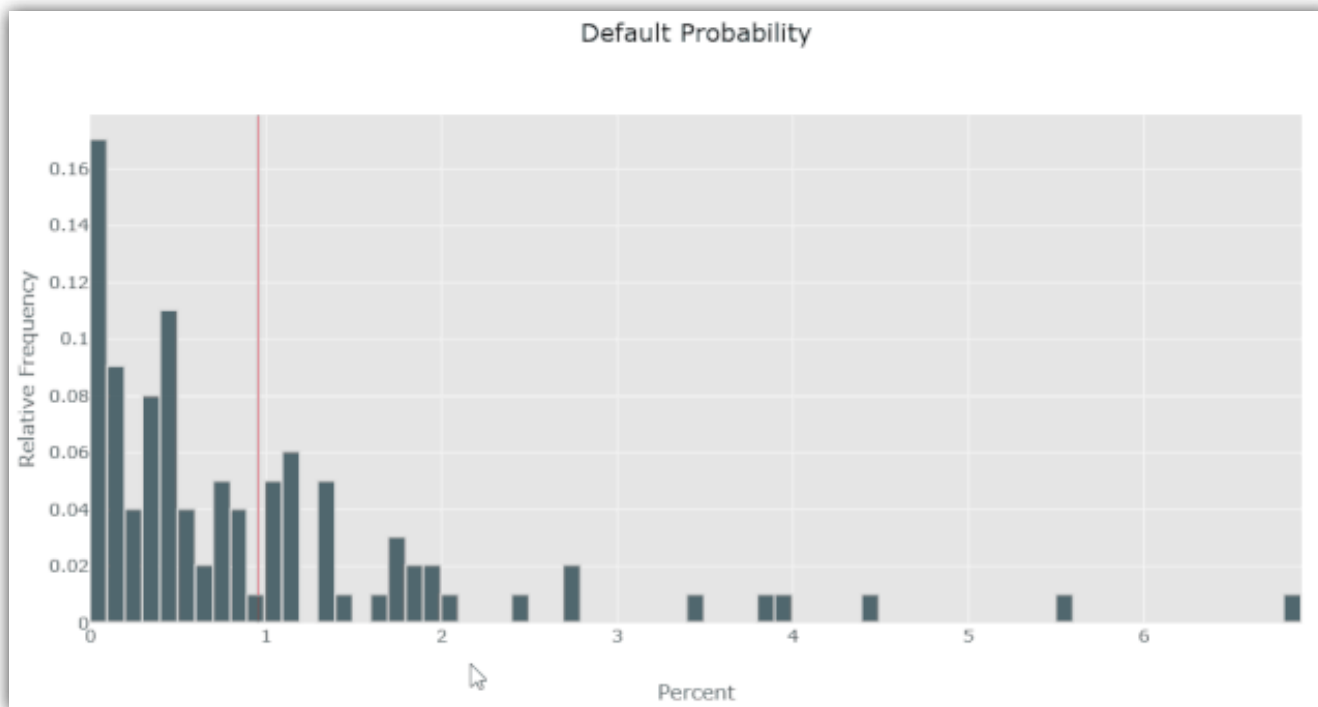
违约率的分布

在 Cufflinks 中我们是在 DataFrame 上做东西，因此先将 p 转成 DataFrame df，然后用 df.iplot() 函数。

我们想看违约率的分布函数，因此 kind 设为 histogram，分 100 个箱。此外加一个垂直线 vline，作为 p 的均值，我们发现值接近 1%，和之前卡方分布的均值非常相近。之后设置一些坐标名称、图名称、颜色和主题 (ggplot 美如画，用过的都说好) 很简单，就不细讲了。

```
1 df = pd.DataFrame(p*100)
2 df.iplot( kind='histogram',
3           histnorm='probability',
4           bins=100,
```

```
5     vline=df.mean().tolist(),
6     title='Default Probability',
7     xTitle='Percent',
8     yTitle='Relative Frequency',
9     color='rgb(43,71,80)',
10    theme='ggplot' )
```

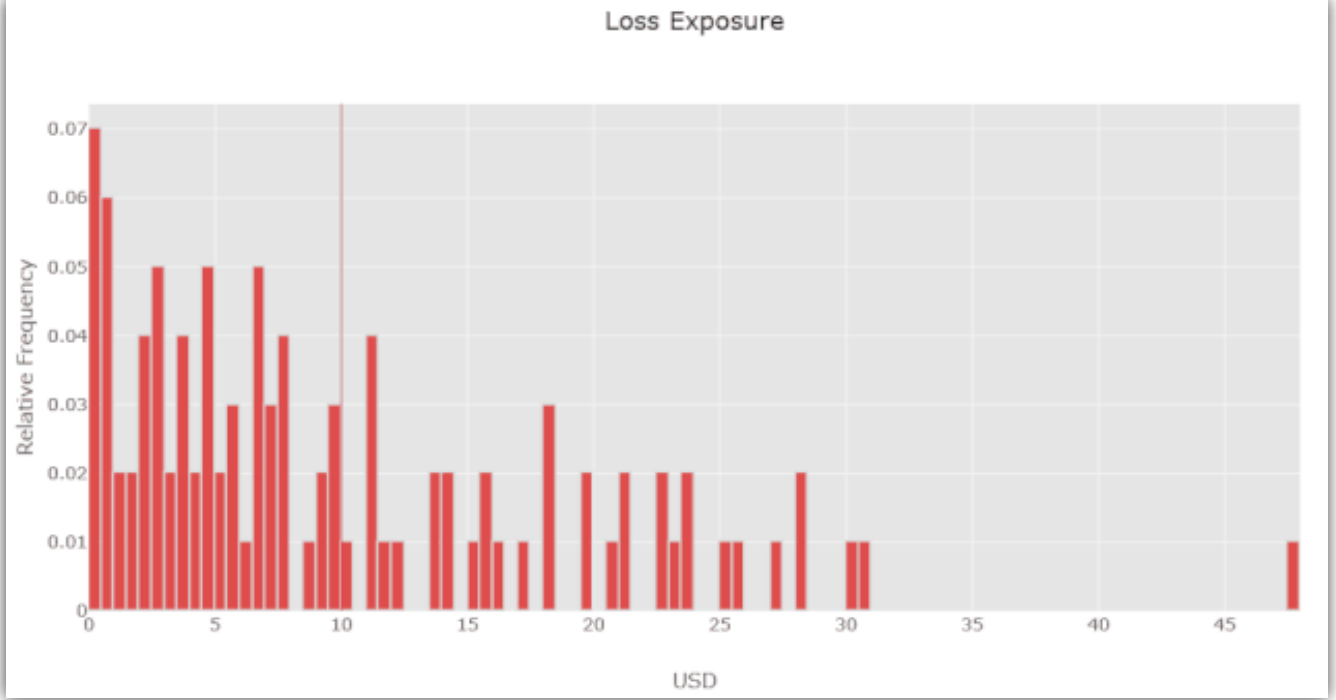


从上图可看出，**违约概率**的值域从 0% 到 7%，而且向右偏斜（skew to the right）。

损失分布

按同样的方法，画出损失分布图。

```
1 df = pd.DataFrame(c)
2 df.iplot( kind='histogram',
3           histnorm='probability',
4           bins=100,
5           vline=df.mean().tolist(),
6           title='Loss Exposure',
7           xTitle='USD',
8           yTitle='Relative Frequency',
9           color='rgb(220,38,36)',
10          theme='ggplot' )
```



从上图可看出，**损失**的值域从 0 到 50，而且也向右偏斜 (skew to the right)。

组合可视化

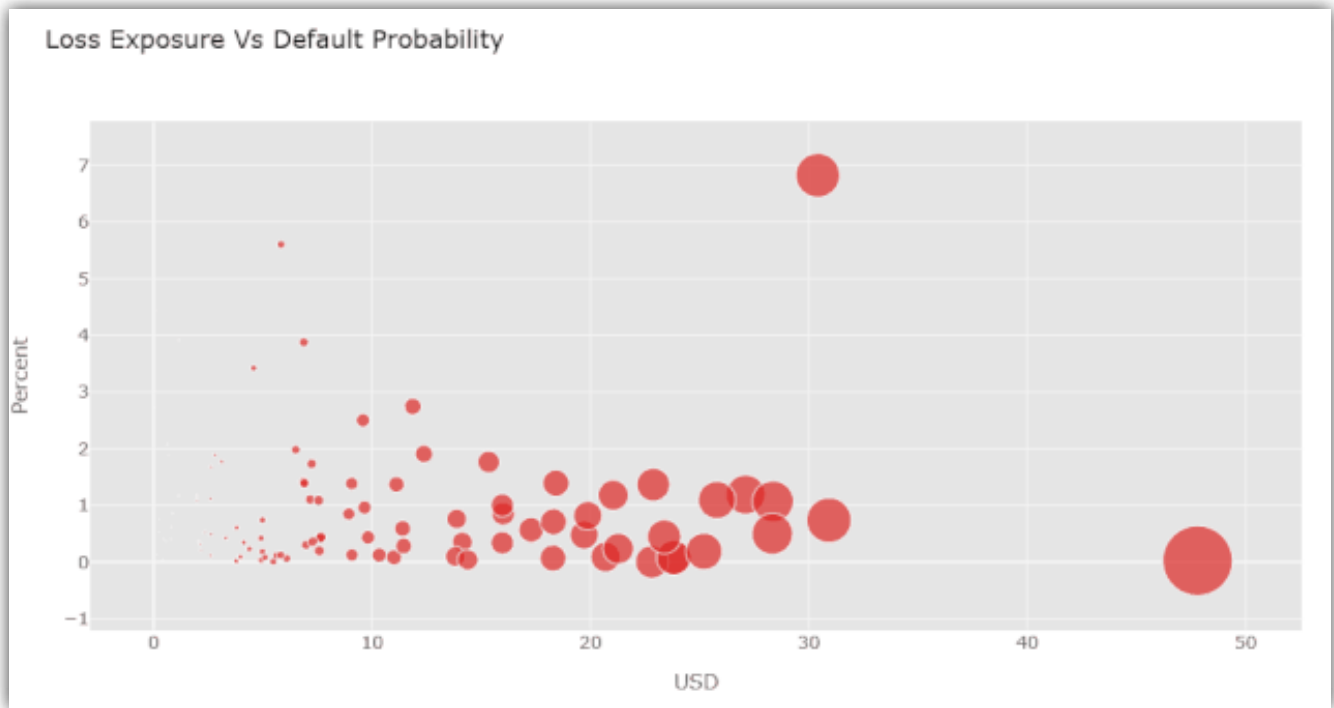
信用组合有 p 和 c 两层信息，组合成新的 DataFrame。这时我们想显示了每个借贷人的**损失敞口**（横轴）和其**无条件违约概率**（纵轴）的散点图 (scatter plot)，因此 kind 设置 scatter。

为了看起来更加直观，我们用每个散点的面积来代表损失敞口，面积越大损失敞口越大，那么点随着向右侧移动而变大，因此 size 设置成 c，而 c 是损失敞口的值。

```
1 df = pd.DataFrame( np.vstack((p*100,c)).T,
2                     columns=['Default Probability', 'Loss Exposure'] )
3
4 df.iplot( kind='scatter',
5           x='Loss Exposure',
6           y='Default Probability',
7           mode='markers',
8           size=c,
9           title='Loss Exposure Vs Default Probability',
10          xTitle='USD',
11          yTitle='Percent',
12          color=color,
```

13

theme='ggplot')



我们可以做一些观察：

1. 大部分大头寸的违约概率都不高，但有几个例外的点。比如有个损失敞口为 30 的违约率为 7 %。
2. 没有其他高违约的借贷人的损失敞口超过 10。
3. 最大的单一头寸（接近 50）的违约概率极小，几乎为 0。

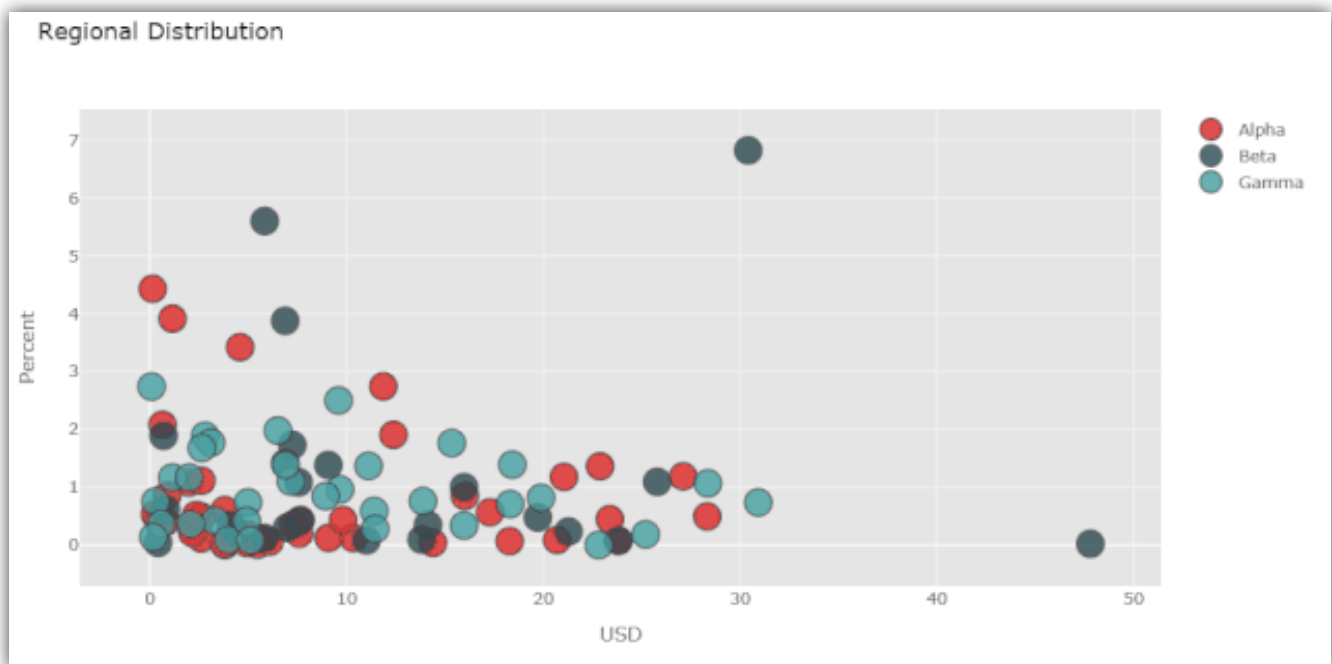
接下来我们来看信用组合在不同区域的分布。广义来讲，区域可按地理，行业或者借贷人规模来分类。

在该案例中，我们将 100 个借贷人随机分配到三个虚构的区域，分别为 Alpha，Beta 和 Gamma。我们同样随机生成些区域数据并存成 regions.npy。

```
1 rgnFile = os.getcwd() + '\\regions.npy'
2 region = np.load(rgnFile)
```

在画图中，每个散点都以根据区域分类而用不同的颜色来显示（在代码将 category 设置为 'Region'）。在实际环境中，每个散点还会包含借贷人 ID 或名称，可帮助我们能够锁定某些特定的借贷人。

```
1 df = pd.DataFrame( np.vstack((p*100,c,region)).T,
2                     columns=['Default Probability', 'Loss Exposure', 'Region'])
3 df.loc[df['Region']==1, 'Region'] = 'Alpha'
4 df.loc[df['Region']==2, 'Region'] = 'Beta'
5 df.loc[df['Region']==3, 'Region'] = 'Gamma'
6
7 df.iplot( kind='scatter',
8           x='Loss Exposure',
9           y='Default Probability',
10          mode='markers',
11          categories='Region',
12          size=20,
13          title='Regional Distribution',
14          xTitle='USD',
15          yTitle='Percent',
16          color=color,
17          theme='ggplot' )
```



我们可以做一些观察：

1. 数据按区域划分，没有显示出任何模式。
2. **最上面**那个深青色点 (30.41, 6.82) 对应的借贷人是高风险头寸 (损失敞口大，违约率高) 。

3. **最右边**那个深青色点 (47.80, 0.02) 对应的借贷人头寸最大, 但是违约风险不大。
4. **底部和中部**有一组借贷人 (1 个深青色, 2 个红色, 2 个青色), 虽然损失敞口大, 但违约率不高, 约为 1%。
5. **左上角**有一组借贷人 (2 个深青色, 3 个红色) 虽然违约率高, 但是损失敞口小。

将信贷组合可视化一下, 我们就可以迅速看到组合里的一些特性。下篇我们来计算该组合的一些有用指标, 如**预期损失** (expected loss, EL), **损失波动** (loss volatility, LV), **风险价值** (value-at-risk, VaR) 和**期望损失** (expected shortfall, ES)。

Stay Tuned!

END

喜大普奔

我的新书《快乐机器学习》
在新加坡终于有买了

扫以下二维码进 Lazada 购买

新加坡全岛包邮哦

