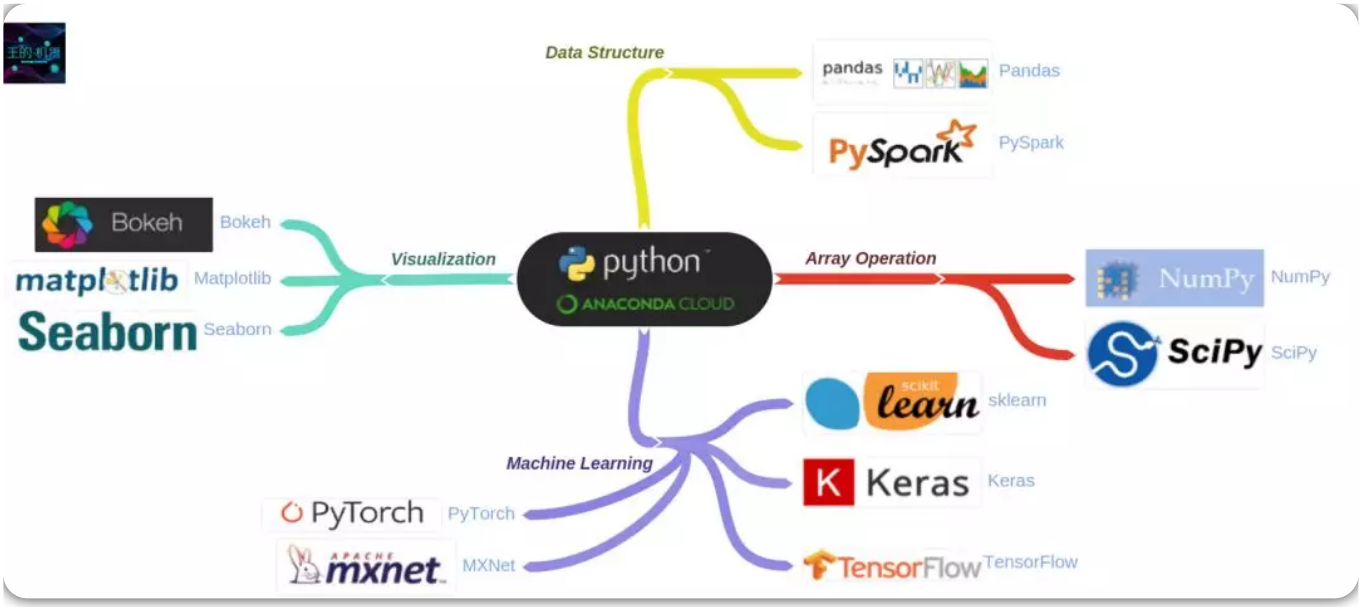
盘一盘 Python 系列特别篇 - Matplotlib Animation

原创 王圣元 王的机器 4月15日

来自专辑

Python



本文含 3400 字，15 图表截屏
建议阅读 20 分钟

在公众号对话框回复 MANI
获取完整 Jupyter Notebook

感谢 EasyShu 公众号主理人张杰博士的帮助

本文是 Python 系列的特别篇的第十三篇

- **特别篇 12 - Collection**
- **特别篇 13 - Matplotlib Animation**

0 引言

本帖我们目的只有一个，复现下面视频展示的内容，即中国（上证）和美国（标普 500）2016 年 3 月到 2020 年 4 月的股市走势对比。先点开视频看一看，配着 Fort Minor 的 Remember the Name 的前奏真带感。

00:19

做出该视频我用了四个工具：

1. **Matplotlib**（核心）
2. **ScreenToGif** 本地软件（用于录屏存成 gif）
3. **ezgif** 在线（用于快进 gif 播放速度被存成视频，用于压缩）
4. **腾讯微视 APP**（用于配乐）

不难发现，后三个都是锦上添花，而第一个才是雪中送炭。

首先引入可能需要的包，其中第 7 行引入的 `animation` 是为了画动态图的。第 12 行也比较重要，有时候动态图太大了，很容易突破默认 byte，如果不设置 `animation.emded_limit`，显示出来的图是不完整的，保险起见可以设一个比较大的数，比如 2^64。

```
import os
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import matplotlib.animation as animation

from datetime import datetime
from IPython.display import HTML

plt.rcParams['animation.embed_limit'] = 2**64
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显正负号
plt.rc('axes', axisbelow=True)
```

1 正文

数据预处理

用 Pandas 从 'data.csv' 中加载数据（2006 年 1 月到 2020 年 4 月 10 日上证和标普 500 的日收盘价），csv 数据的截屏如下：

	A	B	C
1	Date	China	US
2	10/04/2020	2796.63	2789.82
3	09/04/2020	2825.9	2749.98
4	08/04/2020	2815.37	2659.41
5	07/04/2020	2820.76	2663.68
6	03/04/2020	2763.99	2488.65
7	02/04/2020	2780.64	2526.9
8	01/04/2020	2734.52	2470.5
9	31/03/2020	2750.3	2584.59
10	30/03/2020	2747.21	2626.65
11	27/03/2020	2772.2	2541.47
12	26/03/2020	2764.91	2630.07
13	25/03/2020	2781.59	2475.56
14	24/03/2020	2722.44	2447.33
15	23/03/2020	2660.17	2237.4
16	20/03/2020	2745.62	2304.92

下列代码注意三个细节：

1. 将第一列日期作为 DataFrame 的即行标签 (设置 `index_col=0`)
2. 并用列表解析式（list comprehension）将日期字符串转成 `datetime` 对象
3. 用 `df.iloc[::-1]` 将日期逆排，第一行对应着是最旧的日期

打印 DataFrame 的首尾三行看看。

```
df = pd.read_csv('data.csv', index_col=0)
df.index = [datetime.strptime(d, '%d/%m/%Y').date() for d in df.index]
df = df.iloc[::-1]
df.head(3).append(df.tail(3))
```

	China	US
2006-01-04	1180.96	1250.56
2006-01-05	1197.27	1239.20
2006-01-06	1209.42	1246.00
2020-04-08	2815.37	2659.41
2020-04-09	2825.90	2749.98
2020-04-10	2796.63	2789.82

在本例中， 我们只看最近 1000 天的数据，数据太多生成动图太慢。想生成完整的图的同学可用 `df1 = df`。

```
df1 = df.iloc[-1000:,:]
df1.head(3).append(df1.tail(3))
```

	China	US
2016-03-07	2897.34	2091.48
2016-03-08	2901.39	2091.58
2016-03-09	2862.56	2087.79
2020-04-08	2815.37	2659.41
2020-04-09	2825.90	2749.98
2020-04-10	2796.63	2789.82

选取了起始日后（本例是 2016 年 3 月 7 日，读者可以随意选定），为了公平比较，我们计算出每天相对起始日的收益（`df1/df1.iloc[0,:]`），而起始日的收益为 0，换句话就是说从起始日开始投资指数，得到的每天累积收益。收益的单位用 % 来表示，因此乘上个 100。

```
df1 = df1 / df1.iloc[0,:]*100 - 100
df1.head(3).append(df1.tail(3))
```

	China	US
2016-03-07	0.000000	0.000000
2016-03-08	0.139783	0.004781
2016-03-09	-1.200411	-0.176430
2020-04-08	-2.829147	27.154455
2020-04-09	-2.465710	31.484882
2020-04-10	-3.475947	33.389753

数据可视化

要做动图，步骤分三步：

1. 写一个**静态画图函数**，假设叫 `animate(i)`，其中 `i` 可看成是 df1 的变化的 index。不同的 `i` 就会切片得到 `df1.iloc[:i,:]`。

2. 使用 `animation` 库里的 `FuncAnimation()`，其调用形式为

```
FuncAnimation( fig,
               animate,
               frames,
               interval )
```

其中

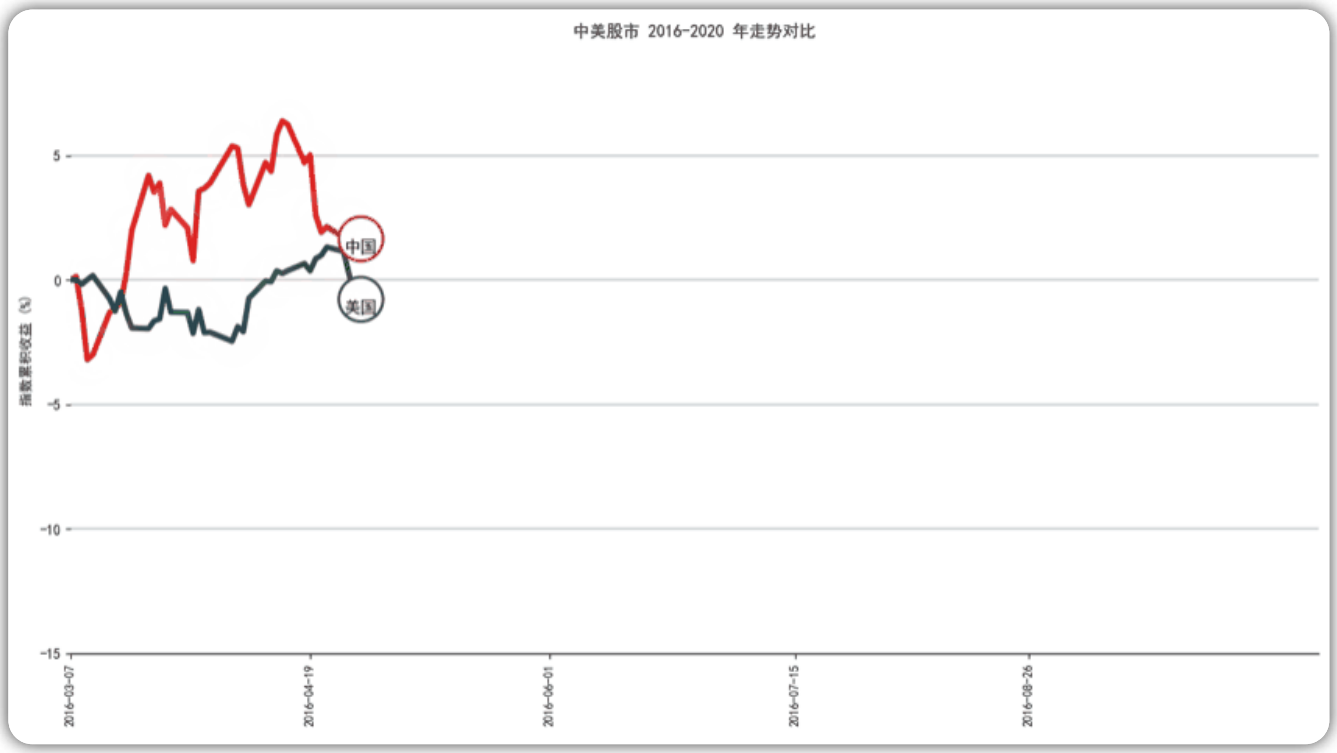
- `fig` 是图对象
- `animate` 是第一步定义的静态画图函数，还记得 Python 里面函数是可以作为参数传递到另外一个高阶函数吗？
- `frames` 设定动画应含多少帧，也就是说，通过该参数定义调用 `animate(i)` 的频率，这里设定为 `np.arange(1,df1.shape[0],1)`，即该动画为 `df1.shape[0]` 帧。
- `interval` 是每一帧的时间间隔，默认是 200ms。

该函数的返回对象起名为 animator。

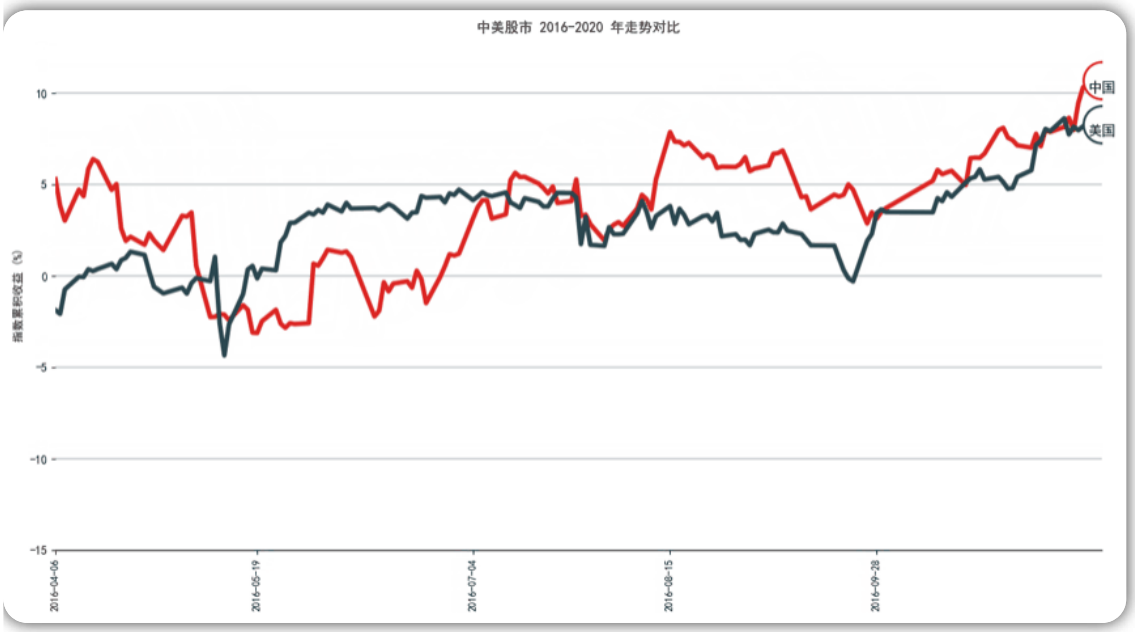
3. 用 `HTML(animator.to_jshtml())` 将动图在 Jupyter Notebook 里展示。

后面 2 和 3 两步非常标准化，真正的细节都体现在第 1 步的 `animate(i)` 中。

看了上面视频，我们发现一开始坐标轴是静止的，任由这两条折线向右运动，如图所示。



过了一段时间，坐标轴变成动态，随着折线也开始运动，如下图所示。因为数据太多了，如果不弄成动态坐标轴最后发现图会越来越小。



为了处理这个坐标轴从静态变动态这个细节，我们要写个 if-else 条件语句，而技巧就是定义一个 num_of_span（比如设定为 150），当 num_of_date（也就是 `animiate(i)` 里的 i）小于 num_of_span 坐标轴为静态，大于等于 num_of_span 坐标轴为动态。

现在静态坐标轴时的代码。

```
1  def animate( num_of_date ):
2      num_of_span = 150
3      ax.clear()
4      if num_of_date < num_of_span:
5          df_temp = df1.iloc[0:num_of_date,:]
6          df_span = df1.iloc[0:num_of_span,:]
7          idx = df_temp.index
8
9          plt.plot( idx, df_temp['China'],
10                   color='#dc2624', linewidth=4, zorder=2 )
11         plt.scatter( idx[-1], df_temp['China'][-1],
12                      color='white', s=1000, edgecolor ='#dc2624', linewidth=2, zorder=3 )
13         plt.text( idx[-1], df_temp['China'][-1],
14                   s='中国', size=12, ha='center', va='top' )
15
16         plt.plot( idx, df_temp['US'],
17                   color='#2b4750', linewidth=4, zorder=2 )
18         plt.scatter( idx[-1], df_temp['US'][-1],
19                      color='white', s=1000, edgecolor ='#2b4750', linewidth=2, zorder=3 )
20         plt.text( idx[-1], df_temp['US'][-1],
21                   s='美国',size=12, ha='center', va='top' )
22
23         plt.ylim( -15, df_span.values.max()*1.2 )
24         plt.xlim( df_span.index[0], df_span.index[-1] )
25         plt.xticks( ticks=df_span.index[0:num_of_span+1:30],
26                     labels=df_span.index.values[0:num_of_span+1:30],
27                     rotation=90,
28                     fontsize=9 )
```

核心代码在第 **5-28** 行，关于 matplotlib 的知识可参考【**盘一盘 matplotlib**】一贴 。

第 5-7 行：切片两个 DataFrame，df_temp 用于画折线和散点，df_span 用于标注横轴标签（第 25-28 行的 xticks）。获取 df_temp 的日期起名为 idx。

第 9-14, 16-21 行：画中美两个股市的**折线图**（用 `plot` 函数），**散点图**（用 `scatter` 函数）和**文字**（用 `text` 函数），我们就以中国举例。

折线图：这个太简单了，前两个参数就是 x 和 y，而后面三个参数都是美化折现，颜色选我个人喜好的那个红色，线宽为 4，zorder = 2 是和下面散点 zorder = 3 对应，就是先画折现后画散点，散点要盖住折线。这样才能出来图中散点加在折线（而不是折线加在散点）的效果。

散点图：这个也简单，但是我们只需要一个散点，最后一个数据的散点，因此 x 和 y 有 [-1] 的索引。其他美化散点的参数就不提了，也是慢慢试出来的，比如散点大小 s 我从 500 试到 1000。

文字：这个也不难，同理我们也只需一个文字，即散点出坐标下写文字“中国”。其他都是美化文字的参数，也不提了。

第 23-28 行：分别设置横轴和纵轴的上下界。对于横轴的上下界，我们用 df_span 的首尾日期，由于 df_num 在这种情况一直小于 df_span，那么当 df_num 动时，df_span 是静止的，因此横轴是静止的。关于 xticks, 我们用 df_span 每隔 30 天显示日期标签，rotation = 90 是为了防止日期太拥挤，转成纵向。

好了，静态横轴的代码详细解释完了，我相信你们可以看懂动态横轴的代码了。最大的变化就是所有数据都是用 [-1] 来索引，因为每次我们都只画最新的数据。

```
29      else:
30          df_temp = df1.iloc[num_of_date-num_of_span:num_of_date,:]
31          idx = df_temp.index
32
33          plt.plot( idx[:-1], df_temp['China'][:-1],
34                    color='#dc2624', linewidth=4, zorder=2 )
35          plt.scatter( idx[-1], df_temp['China'][-1],
36                       color='white', s=1000, edgecolor ='#dc2624', linewidth=2, zorder=3 )
37          plt.text( idx[-1], df_temp['China'][-1],
38                    s='中国', size=12, ha='center', va='top' )
39
40          plt.plot( idx[:-1], df_temp['US'][:-1],
41                    color='#2b4750', linewidth=4, zorder=2 )
42          plt.scatter( idx[-1], df_temp['US'][-1],
43                       color='white', s=1000, edgecolor ='#2b4750', linewidth=2, zorder=3 )
44          plt.text( idx[-1], df_temp['US'][-1],
45                    s='美国', size=12, ha='center', va='top' )
46
47          plt.ylim( -15, df_temp.values.max()*1.2 )
48          plt.xlim( df_temp.index[0], df_temp.index[-1] )
49          plt.xticks( ticks=df_temp.index[0:num_of_span+1:30],
50                      labels=df_temp.index[0:num_of_span+1:30],
51                      rotation=90,
52                      fontsize=9 )
```

最后将图的上边、左边和右边的框去掉，加上横向网格线，标注纵轴标签和图标题。

```
53
54      plt.margins(x=0.2)
55      ax.spines['top'].set_color('none')
56      ax.spines['right'].set_color('none')
57      ax.spines['left'].set_color('none')
58      plt.grid( axis='y', c='#c7cccf', linewidth=1.5 )
59      plt.ylabel('指数累积收益 (%)')
60      plt.title('中美股市 2016-2020 年走势对比')
```

之后用 `FuncAnimation()` 来调用 `animate` 赋予其动态“魔力”。

```
fig = plt.figure(figsize=(16,8), dpi=100)
ax = fig.gca()
animator = animation.FuncAnimation( fig,
                                    animate,
                                    frames=np.arange(1,df1.shape[0],1),
                                    interval=200)
HTML(animator.to_jshtml())
```

最后你可以用 `animator.save()` 来存成视频或者 html 形式，但我发现文件太大，因此我手动用 **ScreenToGif** 做成动图（gif 还是很大，大概 17MB，根本传不上公众号模板中），然后在 **ezgif** 网页上压缩并快播存成视频（20 秒视频才 1.8 MB），再用**微视 APP** 给其配音乐。

这些后期制造大家可以按自己的需求和喜好来做，核心还是用 matplotlib 做出动态图。

2 总结

由于我刚接触这个用 matplotlib 画动图，就是有天一个读者在微信群给我看了这样的视频，我觉的很酷而且记得 matplotlib 可以画动图就是试着实现。整个实现文前视频花了 4 个小时（当然在咨询了可视化专家张杰博士的情况下），单纯就是为实现而学新知识的，因此我只是根据自己的理解把画图原理解释了下，有些地方可能不太准确，大家看了有什么不对的地方欢迎指出。

两点心得：

非技术：根据兴趣学东西非常快，为了完成某个目的，找东西也更有针对性，结果导向很有效。

技术：在运行动图时，由于非常费时，因此建议先把静态函数 `animate(i)` 调试好，然后选取不同的 `i` 值，看看画出来的图是否正确是否符合直觉，再用 `FuncAnimation()` 和 `HTML()` 将它动态化并展示出来，要不然大量时间花在等待制作动图上了。

Stay Tuned!

END