Matplotlib 时间序列

Guangyao Zhao

2023-05-01

Contents

Matplotlib 中画折线图用 ax.plot(x, y)，当横坐标 x 是时间数组时，例如 datetime 或 np.datetime64 构成的列表，x 和 y 的组合即一条时间序列。Matplotlib 能直接画出时间序列，并自动设置刻度。下面以一条长三年的气温时间序列为例：

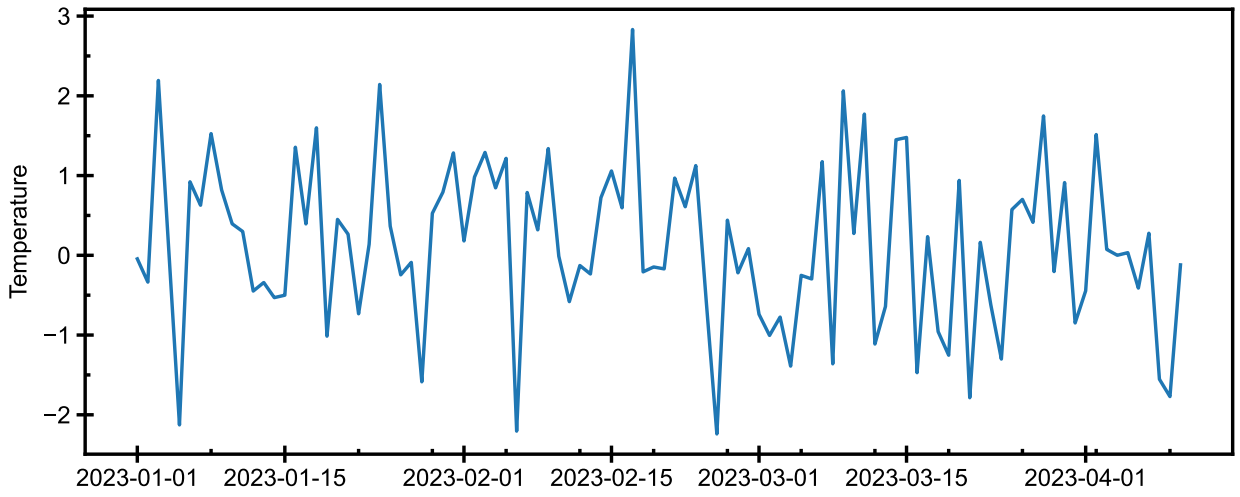
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

tmp = 100
series = pd.Series(np.random.randn(tmp),
                   index=pd.date_range(start="2023/01/01",
                                       periods=tmp,
                                       freq="D"))

fig, ax = plt.subplots(figsize=(10, 4))
ax.plot(series.index, series)
ax.set_ylabel("Temperature")
ax.tick_params(axis="both", which="major", direction="inout")

print(ax.xaxis.get_major_locator())
print(ax.xaxis.get_major_formatter())
```

```
<matplotlib.dates.AutoDateLocator object at 0x1284825e0>
<matplotlib.dates.AutoDateFormatter object at 0x12848d5b0>
```



打印 x 轴的属性发现，Matplotlib 默认为时间序列设置了 `AutoDateLocator` 和 `AutoDateFormatter`，前者会自动根据 `ax` 的时间范围在 x 轴上选出位置、数量和间隔都比较合适的刻度，后者会自动根据主刻度的间隔，将刻度标签格式化为合适的样式。

虽然自动刻度很方便，但如果想像上图一样调整刻度间隔，追加小刻度，并修改刻度标签格式，就需要手动设置刻度。本文的目的就是介绍手动修改时间刻度的方法，内容主要分为三点：

- 了解 Matplotlib 处理时间的机制。
- 运用 `matplotlib.dates` 模块里提供的工具设置刻度。
- 解决 Pandas 时间序列图的问题。

matplotlib 处理时间的机制

`matplotlib.dates`（后简称 `mdates`）模块里有两个函数：`date2num` 和 `num2date`。前者能将一个 `datetime` 或 `np.datetime64` 对象转换成该对象离 `1970-01-01T00:00:00` 以来的天数（注意不是秒数），后者则是反过来转换。当 `ax.plot` 接受时间类型的 x 时，会在内部创建一个 `mdates.DateConverter` 对象，对 x 的每个元素调用 `date2num`，将其转换成表示天数的浮点型一维数组。Matplotlib 在内部便是以这种浮点数的形式存储时间的。下面验证一下这点：

```
x0, x1 = ax.get_xlim()
origin = "1970-01-01 00:00"
t0 = pd.to_datetime(x0, unit="D", origin=origin)
```

```
t1 = pd.to_datetime(x1, unit="D", origin=origin)
print("x0 =", x0, "t0 =", t0)
print("x1 =", x1, "t1 =", t1)
```

```
x0 = 19353.05 t0 = 2022-12-27 01:12:00
x1 = 19461.95 t1 = 2023-04-14 22:48:00
```

其中 pd.to_datetime 可以直接换成 num2date。所以后续在 ax 上画新线条时，使用时间类型或浮点类型的 x 都可以。

使用 matplotlib.dates 提供的工具

除引言里提到的 AutoDateLocator 和 AutoDateFormatter 外，mdates 还提供其它规则的 Locator 和 Formatter。以设置月份刻度的 MonthLocator 为例：

```
dates.MonthLocator(bymonth=None, bymonthday=1, interval=1, tz=None)
```

其中 bymonth 参数可以是表示月份的整数，或整数构成的列表，默认值是 1 - 12 月。MonthLocator 会在 ax 的 x 轴显示范围间生成一系列间隔为 interval 个月的 datetime 对象，它们的日由 bymonthday 指定，时分秒都为 0。从中挑选出月份跟 bymonth 匹配的对象，调用 date2num 函数作为最后的刻度值。因为内部实现用的是 dateutil.rrule.rrule，所以参数也是与之同名的。例如 MonthLocator() 的效果就是在每年每月 1 号 00:00:00 的位置设置一个刻度，那么一年就会有 12 个刻度。MonthLocator(bymonth=[1，4，7，10]) 就是在每年 1、4、7 和 10 月设置刻度。

除此之外 mdates 里还有 YearLocator，DayLocator，WeekDayLocator，HourLocator 等，原理和参数跟 MonthLocator 类似，就不多介绍了。

接着以 DateFormatter 为例：class matplotlib.dates.DateFormatter(fmt, tz=None, *, usetex=None)。原理非常简单，就是对刻度值 x 调用 num2date(x).strftime(fmt)，得到刻度标签。例如取 DateFormatter(fmt='%Y-%m')，就能让刻度标签呈 YYYY-MM 的格式。此外我们知道，如果直接向 ax.xaxis.get_major_formatter 传入一个参数为 x 和 pos 的函数，就相当于用这个函数构造了一个 FuncFormatter。所以可以简单自制一个只在每年 1 月标出年份的 Formatter：

```
def format_func(x, pos=None):
    x = mdates.num2date(x)
    if x.month == 1:
        fmt = "%m\n%Y"
```

```
    else:
        fmt = "%m"
    label = x.strftime(fmt)

    return label
```

```
import matplotlib.dates as mdates

ax.xaxis.set_major_locator(mdates.MonthLocator([1, 4, 7, 10]))
ax.xaxis.set_minor_locator(mdates.MonthLocator())
ax.xaxis.set_major_formatter(format_func)
```