Python时间序列分析之_时间重采样（降采样和升采样）

python万
发布时间：01-15 17:47

上篇文章中，我们学习了如何使用pandas库中的date_range()函数生成时间序列索引，而且我们知道我们可以生成不同频率的时间索引，比如按小时、按天、按周、按月等等，因此就会引出另外一个问题，如果我们相对数据做不同频率的转换，该怎么做，pandas库中是否有现成的方法可供使用呢？带着这个问题，我们本次就来学习下数据重采样的知识。

首先，简单解释什么是数据重采样，所谓数据重采样就是将数据原有的频率转换到另一个频率上，如果是从低频率转换到高频率，那么就是升采样，比如原来的数据是按月统计的，通过升采样可以转换为按日的序列数据;相反,如果原来的数据是按日的时间序列数据，通过转换函数转换为按月的时间序列数据，那么我们就将这个过程称为降采样。不管是升采样还是降采样，都通过pandas中的resample函数完成，下面我们就具体来学习这个函数。

pandas.DataFrame.resample

pandas.DataFrame.resample()这个函数主要是用来对时间序列做频率转换，函数原型如下：

DataFrame.resample(rule, how=None, axis=0, fill_method=None, closed=None, label=None, convention='start', kind=None, loffset=None, limit=None, base=0, on=None, level=None)，各参数含义如下列表：

参数	参数说明
rule	表示重采样频率，例如'M'、'5min'，Second(15)
how='mean'	用于产生聚合值的函数名或数组函数，例如'mean'、'ohlc'、np.max等，默认是'mean'，其他常用的值有：'first'、'last'、'median'、'max'、'min'
axis=0	默认是纵轴，横轴设置axis=1
fill_method = None	升采样时如何插值，比如'ffill'、'bfill'等
closed = 'right'	在降采样时，各时间段的哪一段是闭合的，'right'或'left'，默认'right'
label= 'right'	在降采样时，如何设置聚合值的标签，例如，9：30-9：35会被标记成9：30还是9：35,默认9：35
loffset = None	面元标签的时间校正值，比如'-1s'或Second(-1)用于将聚合标签调早1秒
limit=None	在向前或向后填充时，允许填充的最大时期数
kind = None	聚合到时期（'period'）或时间戳（'timestamp'），默认聚合到时间序列的索引类型
convention = None	重采样时期时，将低频率转换到高频率所采用的约定（start或end）。默认'en

重采样参数含义

降采样(高频数据到低频数据)

上面我们已经知道了重采样的函数和函数各参数的含义，下面我们就用一个例子演示降采样的过程。为了方便观察，这里我们生成10条日数据，如下：

- import pandas as pd
- import numpy as np
- index=pd.date_range('20190115','20190125',freq='D')
- data1=pd.Series(np.arange(len(index)),index=index)

作者最新文章

时间序列的平稳性及使用差分法处理非平稳时间序列

python时间序列分析之_用pandas中的rolling函数计算时间窗口数据

Python时间序列分析之_时间重采样（降采样和升采样）

相关文章

SpringBoot 跨系统单点登陆的实现丨CSDN 博文精选

Java 服务端乱象大盘点

Jupyter Notebook 使用小技巧

为什么说Python是学习人工智能的第一语言？

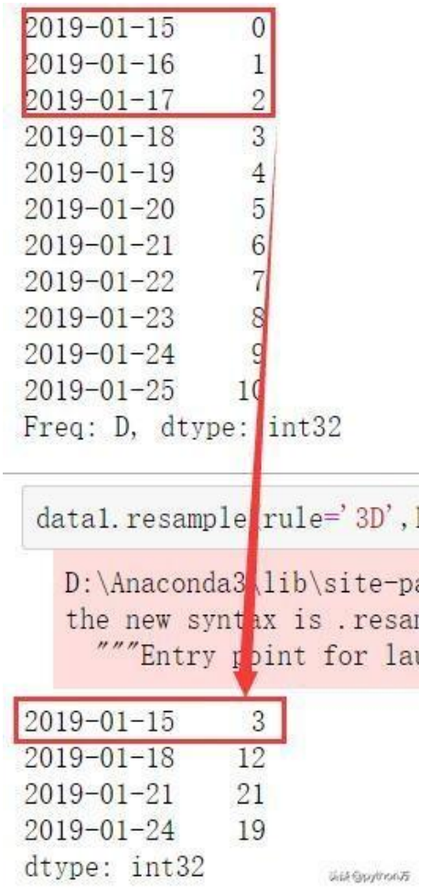
Spring中对LookUp注解的处理

```
Out[25]: 2019-01-15     0
         2019-01-16     1
         2019-01-17     2
         2019-01-18     3
         2019-01-19     4
         2019-01-20     5
         2019-01-21     6
         2019-01-22     7
         2019-01-23     8
         2019-01-24     9
         2019-01-25    10
         Freq: D, dtype: int32
```


10条日数据

如上图，这里有10条日数据，现在我们将这按日统计的数据通过降采样的方法转化为按3日求和统计的数据，如下：

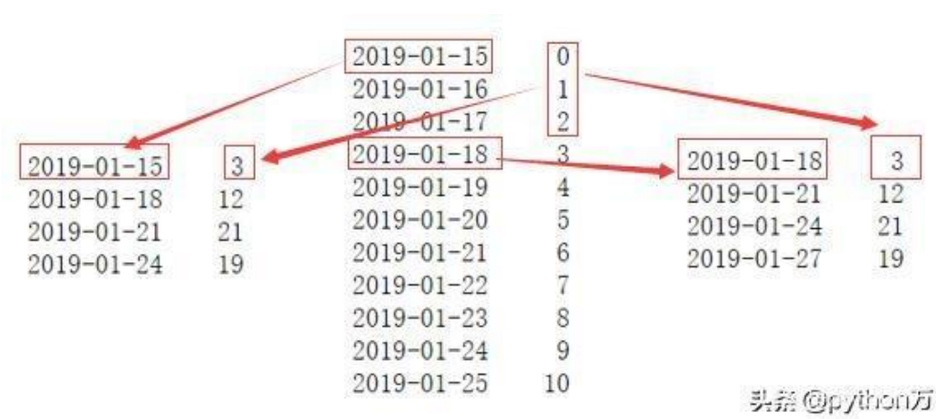
- data1.resample(rule='3D',how='sum')



降采样

可以看到，原来10条按日统计的数据经过降采样转换变为4条（最后一条是2日的数据和）按每3天统计的数据，这就是降采样的过程，这里需要注意的是起算的节点。上面的例子中label这个参数默认的是left,现在我们改为right,看看有什么区别，如下：

- data1.resample(rule='3D',how='sum',label='right')



通过上面的对比，可以得出label这个参数控制了分组后聚合标签的取值。在label为right的情况下，将取分箱右侧的值作为新的标签。上面这个例子基本上演示了降采样的过程，当然我们可以调整how这个参数值进行不同的取值，如下：

- data1.resample(rule='3D',how='mean')#取3天的平均值

```
data1.resample(rule='3D',how='mean')
D:\Anaconda3\lib\site-packages\ipyk
the new syntax is .resample(...).me
  """Entry point for launching an I
2019-01-15    1.0
2019-01-18    4.0
2019-01-21    7.0
2019-01-24    9.5
```


how=mean

也可以将聚合方法写到外面，如下：

- data1.resample(rule='3D').mean()

```
data1.resample(rule='3D').mean()
2019-01-15    1.0
2019-01-18    4.0
2019-01-21    7.0
2019-01-24    9.5
dtype: float64
```


取3天的平均值

取3天的最小值作为新标签值：

```
data1.resample(rule='3D').min()
2019-01-15    0
2019-01-18    3
2019-01-21    6
2019-01-24    9
dtype: int32
```


取3天的最小值

升采样(低频数据到高频数据)

上面演示了降采样的过程，下面我们演示升采样的过程，根据升采样的定义，我们只需在resample函数中改变频率即可，但与降采样不同的是升采样后新增频率的数为空值，为此，rasample也提供了3种方式进行填充，下面我们通过代码来演示。

data1.resample(rule='6H').asfreq()

```
2019-01-15 00:00:00    0.0
2019-01-15 06:00:00    NaN
2019-01-15 12:00:00    NaN
2019-01-15 18:00:00    NaN
2019-01-16 00:00:00    1.0
2019-01-16 06:00:00    NaN
2019-01-16 12:00:00    NaN
2019-01-16 18:00:00    NaN
2019-01-17 00:00:00    2.0
2019-01-17 06:00:00    NaN
2019-01-17 12:00:00    NaN
2019-01-17 18:00:00    NaN
2019-01-18 00:00:00    3.0
2019-01-18 06:00:00    NaN
2019-01-18 12:00:00    NaN
2019-01-18 18:00:00    NaN
2019-01-19 00:00:00    4.0
```


按6小时进行升采样

可以看到，将原来的按日的数据进行升采样为6小时时，会产生很多空值，对于这种空值resample提供了3种方式，分别为ffill（取前面的值）、bfill（取后面的值）、interpolate(线性取值)，这里我们分别进行测试，如下：

- data1.resample(rule='6H').ffill()

```
data1.resample(rule='6H').ffill()
2019-01-15 00:00:00    0
2019-01-15 06:00:00    0
2019-01-15 12:00:00    0
2019-01-15 18:00:00    0
2019-01-16 00:00:00    1
2019-01-16 06:00:00    1
2019-01-16 12:00:00    1
2019-01-16 18:00:00    1
2019-01-17 00:00:00    2
2019-01-17 06:00:00    2
2019-01-17 12:00:00    2
2019-01-17 18:00:00    2
```


按前面的值进行填充

可以看到，在ffill不带任何数字的情况下，填充了所有的空值，这里我们可以输入个数，从而指定要填充的空值个数，如下:

- data1.resample(rule='6H').ffill(2)

```
data1.resample(rule='6H').ffill(2)
2019-01-15 00:00:00    0.0
2019-01-15 06:00:00    0.0
2019-01-15 12:00:00    0.0
2019-01-15 18:00:00    NaN
2019-01-16 00:00:00    1.0
2019-01-16 06:00:00    1.0
2019-01-16 12:00:00    1.0
2019-01-16 18:00:00    NaN
2019-01-17 00:00:00    2.0
2019-01-17 06:00:00    2.0
2019-01-17 12:00:00    2.0
2019-01-17 18:00:00    NaN
```


填充2个

```
data1.resample(rule='6H').bfill()
```

```
2019-01-15 00:00:00    0
2019-01-15 06:00:00    1
2019-01-15 12:00:00    1
2019-01-15 18:00:00    1
2019-01-16 00:00:00    1
2019-01-16 06:00:00    2
2019-01-16 12:00:00    2
2019-01-16 18:00:00    2
2019-01-17 00:00:00    2
2019-01-17 06:00:00    3
2019-01-17 12:00:00    3
```


用后面的值填充

- data1.resample(rule='6H').interpolate()

```
data1.resample(rule='6H').interpolate()
```

```
2019-01-15 00:00:00    0.00
2019-01-15 06:00:00    0.25
2019-01-15 12:00:00    0.50
2019-01-15 18:00:00    0.75
2019-01-16 00:00:00    1.00
2019-01-16 06:00:00    1.25
2019-01-16 12:00:00    1.50
2019-01-16 18:00:00    1.75
2019-01-17 00:00:00    2.00
2019-01-17 06:00:00    2.25
2019-01-17 12:00:00    2.50
2019-01-17 18:00:00    2.75
2019-01-18 00:00:00    3.00
```

线性填充

线性填充

好了，今天的内容到此为止，下篇讲时间滑动窗口。喜欢的小伙伴请收藏和关注！