

D16 pandas 時間序列

[簡報閱讀](#)[範例與作業](#)[問題討論](#)[學習心得\(完成\)](#)[重要知識點](#)[時間序列資料處理](#)[日期時間處理](#)[知識點回顧](#)[參考資料](#)[延伸閱讀](#)

重要知識點



- 日期時間處理

時間序列資料處理

- 所有資料中只要有時間關係就需要使用到時間序列的資料型態，因為資料之間是有時間關係的，資料之間的時間距離也不盡相同，例如右表，紅框內同樣差一個月，但是相差的天數不同
- 以每個月月底資料來檢視，這組資料並無缺值，但是以日資料來看，就缺了很多筆資料了
- 時間序列的資料非常注重時間的間隔

2020-01-31	1.296478
2020-02-29	1.484486
2020-03-31	-0.386317
2020-04-30	-0.013741
2020-05-31	-0.735426
2020-06-30	0.617767
2020-07-31	0.898438
2020-08-31	0.771031
2020-09-30	1.757624
2020-10-31	-0.656737

Freq: M, dtype: float64

既然時間間隔重要，那首先必須介紹控制時間長度的函數.to_period()，參數 freq 代表時間頻率(Y：年、M：月、D：日、H：小時)

2020	-0.735426	2020-05	-0.735426	2020-05-31	-0.735426	2020-05-31 00:00	-0.735426
2020	0.617767	2020-06	0.617767	2020-06-30	0.617767	2020-06-30 00:00	0.617767
2020	0.898438	2020-07	0.898438	2020-07-31	0.898438	2020-07-31 00:00	0.898438
2020	0.771031	2020-08	0.771031	2020-08-31	0.771031	2020-08-31 00:00	0.771031
2020	1.757624	2020-09	1.757624	2020-09-30	1.757624	2020-09-30 00:00	1.757624
2020	-0.656737	2020-10	-0.656737	2020-10-31	-0.656737	2020-10-31 00:00	-0.656737

Freq: A-DEC, dtype: float64 Freq: M, dtype: float64 Freq: D, dtype: float64 Freq: H, dtype: float64

更改時間頻率如果從年轉成季該怎麼做？

- 可以運用 `resample` 函數將年轉成季，如沒有值的填上 `nan`。

```
s = pd.Series([1, 2], index=pd.period_range('2018-01-01', freq='Y', periods=2))
s
```

```
2018    1
2019    2
Freq: A-DEC, dtype: int64
```

```
s.resample('Q', convention='start').asfreq()
```

```
2018Q1    1.0
2018Q2    NaN
2018Q3    NaN
2018Q4    NaN
2019Q1    2.0
2019Q2    NaN
2019Q3    NaN
2019Q4    NaN
Freq: Q-DEC, dtype: float64
```

- 可以用先前學到的索引操作找到特定時間點的資料。

```
ts['2020-03-31': '2020-07-31']
```

```
2020-03-31    -0.386317
2020-04-30    -0.013741
2020-05-31    -0.735426
2020-06-30     0.617767
2020-07-31     0.898438
Freq: M, dtype: float64
```

- 也可以用月的方式做索引操作

```
2020-02-29    1.484486
2020-03-31   -0.386317
2020-04-30   -0.013741
2020-05-31   -0.735426
Freq: M, dtype: float64
```

移動 (shifting) 指的是沿著時間軸將資料前移或後移。Series 和 DataFrame 都有一個 .shift() 方法用於執行單純的移動操作。

ts		ts.shift(2, freq='D')	
2020-01-31	1.057729	2020-02-02	1.057729
2020-02-29	-0.296776	2020-03-02	-0.296776
2020-03-31	-0.984358	2020-04-02	-0.984358
2020-04-30	0.205607	2020-05-02	0.205607
2020-05-31	-0.189151	2020-06-02	-0.189151
2020-06-30	-0.624924	2020-07-02	-0.624924
2020-07-31	-1.168424	2020-08-02	-1.168424
2020-08-31	-1.383008	2020-09-02	-1.383008
2020-09-30	-0.606416	2020-10-02	-0.606416
2020-10-31	-1.391943	2020-11-02	-1.391943
Freq: M, dtype: float64		dtype: float64	

分時間資料以及字串差別，時間需要使用 pd.Timestamp() 做設定，並不是只使用字串就可以代表時間。

```
str_date = '2020-10-10'
date = pd.Timestamp(2020, 10, 10)
```

```
str_date, type(str_date)

('2020-10-10', str)
```

```
date, type(date)

(Timestamp('2020-10-10 00:00:00'), pandas._libs.tslibs.timestamps.Timestamp)
```

- 字串轉時間

```
date2str = date.strftime('%Y-%m-%d')  
date2str, type(date2str)
```

```
('2020-10-10', str)
```

```
str2date = pd.to_datetime(str_date)  
str2date, type(str2date)
```

```
(Timestamp('2020-10-10 00:00:00'), pandas._libs.tslibs.timestamps.Timestamp)
```

日期時間處理

接下來介紹 timestamps 的常用函數

- 直接呼叫出年月日，在 timestamps 後面加上回傳的 year, month, day 即可

```
date.year, date.month, date.day
```

```
(2020, 10, 10)
```

- 也可以呼叫星期與周數

```
date.day_name(), date.weekofyear
```

```
('Saturday', 41)
```

- Timestamps 可以直接加時間或是計算時間差距

```
date2 - date1
```

```
Timedelta('31 days 00:00:00')
```

```
date1 + pd.Timedelta(days=1)
```

```
Timestamp('2020-10-11 00:00:00')
```

- 也可以加工作日天數

```
two_business_days = 2 * pd.offsets.BDay()
date1_add_two_business_days = date1 + two_business_days
date1.day_name(), date1_add_two_business_days.day_name()

('Saturday', 'Tuesday')
```

參考網址：[時間序列與日期用法](#)

知識點回顧

- 時間序列的資料非常注重時間的間隔
- 時間序列的資料可以使用索引操作
- 時間資料可以加時間或是計算相差時間
- 時間資料可以呼叫年、月、日、第幾周、星期幾

參考資料

時間序列處理

網站：[Pandas庫基礎分析——詳解時間序列的處理](#)



慕課網
已認證的官方帳號

12人讀同了該文章

在使用Python進行數據分析時，經常會遇到時間日期格式處理和轉換，特別是分析和挖掘與時間相關的數據，比如量化交易就是從歷史數據中尋找股價的變化規律。Python中自帶的處理時間的模塊有datetime，NumPy庫也提供了相應的方法，Pandas作為Python環境下的數據分析庫，更是提供了強大的日期數據處理的功能，是處理時間序列的利器。

1、生成日期序列

主要提供pd.date_range()和pd.period_range()兩個方法，給定參數有起始時間、結束時間、生成時期的數目及時間頻率（freq='M'月，'D'天，'W'，週，'Y'年）等。

兩種主要區別在於pd.date_range()生成的是DatetimeIndex格式的日期序列；pd.period_range()生成的是PeriodIndex格式的日期序列。

以下通過生成月時間序列和周時間序列來對比下：

timestamp

網站：[pandas處理時間序列（1）](#)：
[pd.Timestamp\(\)](#)、[pd.Timedelta\(\)](#)、
[pd.datetime\(\)](#)、[pd.Period\(\)](#)、
[pd.to_timestamp\(\)](#)、[datetime.strptime\(\)](#)、
[pd.to_datetime\(\)](#)、[pd.to_period\(\)](#)

一、定義時間格式

1. pd.Timestamp()、pd.Timedelta()

(1) Timestamp時間戳

```
1 #定义timestamp
2 t1=pd.Timestamp('2019-01-10')
3 t2=pd.Timestamp('2018-12-10')
4 print(f't1= {t1}')
5 print(f't2= {t2}')
6 print(f't1与t2时间间隔: {(t1-t2).days}天')
```

```
t1= 2019-01-10 00:00:00
t2= 2018-12-10 00:00:00
t1与t2时间间隔: 31天
```

```
1 #获取当前时间
2 now=pd.datetime.now()
3 print(now)
4 print(now.strftime('%Y-%m-%d'))
```

```
2019-04-07 12:07:45.690932
2019-04-07
```

延伸閱讀

Pandas 必備技能之「時間序列資料處理」

網站：[mdeditor](#)

```
在[ 9 ]中: data.info ()
<類 'Pandas' 核心。框架。Dataframe '>
RangeIndex: 209 個條目, 0 到 208個
數據列 (總共 2 列):
日期 209 非null 對象
UNRATE 209 非null float64
dtypes: float64 (1), object (1)
內存使用量: 3.3 + KB
```

設定引數parse_dates = ['date']，將資料型別轉換成日期，再設定 index_col = 'date'，將這一列用作索引，結果如下。

```
在[11]中: data = pd.read_csv('unemployment.csv', parse_dates = [ 'date' ], index_col = 'date')
在[12]中: data.info ()
< class'pandas.core.frame.DataFrame' >
DatetimeIndex: 209個條目, 從2000-01-01至2017-05-01
數據列 (共1列):
UNRATE 209非null 浮點數64
dtypes: float 64 (1)
內存使用量: 13.3 KB
```

這時，索引變成了日期'20000101'-'2017-05-01'，資料型別是datetime。

第二種方法是在已經匯入資料的情況下，用pd.to_datetime()【2】將列轉換成日期型別，再用 df.set_index()【3】將其設定為索引，完成轉換。

以tushare.pro上面的日線行情資料為例，我們把'trade_date'列轉換成日期型別，並設定成索引。

Pandas 日期功能、Pandas 時間差 (Timedelta)

● [易百教程：Pandas 日期功能](#)

Pandas描述性統計	<h3>創建一個日期範圍</h3> <p>通過指定週期和頻率，使用 <code>date.range()</code> 函數就可以創建日期序列。默認情況下，範圍的頻率是天。參考以下示例代碼-</p> <pre>import pandas as pd datelist = pd.date_range('2020/11/21', periods=5) print(datelist)</pre> <p>執行上面示例代碼，得到以下結果 -</p> <pre>DatetimeIndex(['2020-11-21', '2020-11-22', '2020-11-23', '2020-11-24', '2020-11-25'], dtype='datetime64[ns]', freq='D')</pre> <h3>更改日期頻率</h3> <pre>import pandas as pd datelist = pd.date_range('2020/11/21', periods=5, freq='M') print(datelist)</pre> <p>執行上面示例代碼，得到以下結果 -</p> <pre>DatetimeIndex(['2020-11-30', '2020-12-31', '2021-01-31', '2021-02-28', '2021-03-31'], dtype='datetime64[ns]', freq='M')</pre> <h3>bdate_range()函數</h3> <p><code>bdate_range()</code> 用來表示商業日期範圍。不同於 <code>date_range()</code>，它不包括星期六和星期天。</p> <pre>import pandas as pd datelist = pd.bdate_range('2011/11/03', periods=5) print(datelist)</pre> <p>執行上面示例代碼，得到以下結果 -</p>
Pandas函數應用	
Pandas重建索引	
Pandas迭代	
Pandas排序	
Pandas字符串和文本數據	
Pandas選項和自定義	
Pandas索引和選擇數據	
Pandas統計函數	
Pandas窗口函數	
Pandas聚合	
Pandas缺失數據	
Pandas分組 (GroupBy)	
Pandas合併/連接	
Pandas級聯	
Pandas日期功能	
Pandas時間差	
Pandas分類數據	
Pandas可視化	
Pandas IO工具	
Pandas稀疏數據	
Pandas注意事項 & 竅門	

● [易百教程：Pandas 時間差](#)

可以使用各種參數創建 `Timedelta` 對象，如下所示--

弦樂

通過傳遞字符串，可以創建一個 `timedelta` 對象。參考以下示例代碼-

```
import pandas as pd

timediff = pd.Timedelta('2 days 2 hours 15 minutes 30 seconds')
print(timediff)
```

執行上面救命代碼，得到以下結果-

```
2 days 02:15:30
```

整體

通過傳遞一個整數值與指定單位，這樣的一個參數也可以用來創建 `Timedelta` 對象。

```
import pandas as pd

timediff = pd.Timedelta(6,unit='h')
print(timediff)
```

[下一步：閱讀範例與完成作業](#)

