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

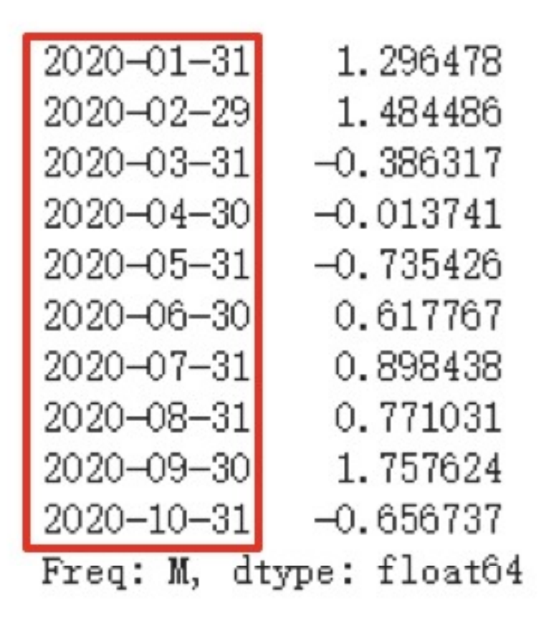
知識點回顧

重要知識點

- 時間序列資料處理
- 日期時間處理

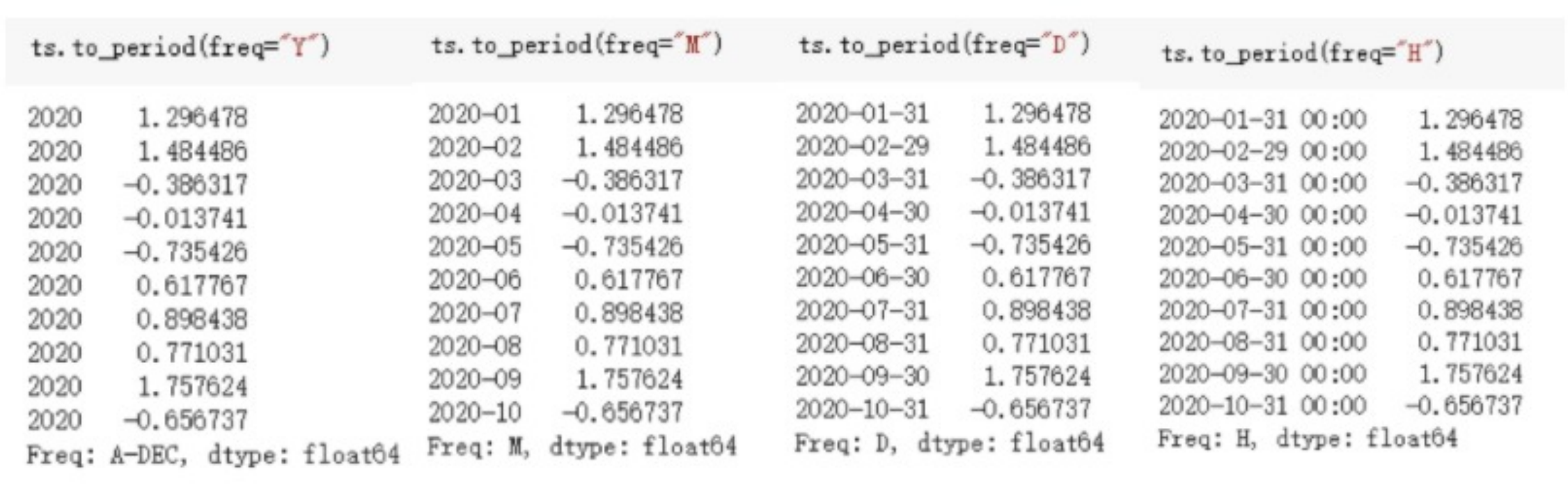
時間序列資料處理

- 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)



```
ts.to_period(freq="Y")

2020    1.296478
2020    1.484486
2020   -0.386317
2020   -0.013741
2020   -0.735426
2020    0.617767
2020    0.898438
2020    0.771031
2020    1.757624
2020   -0.656737
Freq: A-DEC, dtype: float64
```

```
ts.to_period(freq="M")

2020-01    1.296478
2020-02    1.484486
2020-03   -0.386317
2020-04   -0.013741
2020-05   -0.735426
2020-06    0.617767
2020-07    0.898438
2020-08    0.771031
2020-09    1.757624
2020-10   -0.656737
Freq: M, dtype: float64
```

```
ts.to_period(freq="D")

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: D, dtype: float64
```

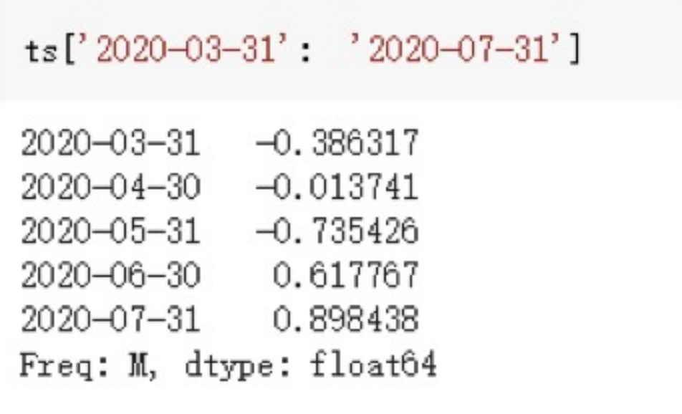
```
ts.to_period(freq="H")

2020-01-31 00:00    1.296478
2020-02-29 00:00    1.484486
2020-03-31 00:00   -0.386317
2020-04-30 00:00   -0.013741
2020-05-31 00:00   -0.735426
2020-06-30 00:00    0.617767
2020-07-31 00:00    0.898438
2020-08-31 00:00    0.771031
2020-09-30 00:00    1.757624
2020-10-31 00:00   -0.656737
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。

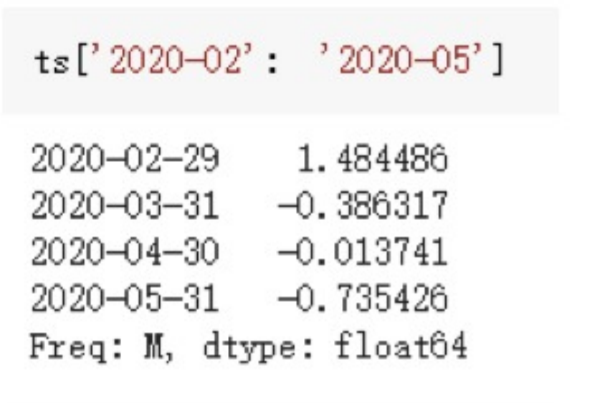
```
s = pd.Series([1, 2], index=pd.period_range('2018-01-01',freq='Y',periods=2))
s

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。



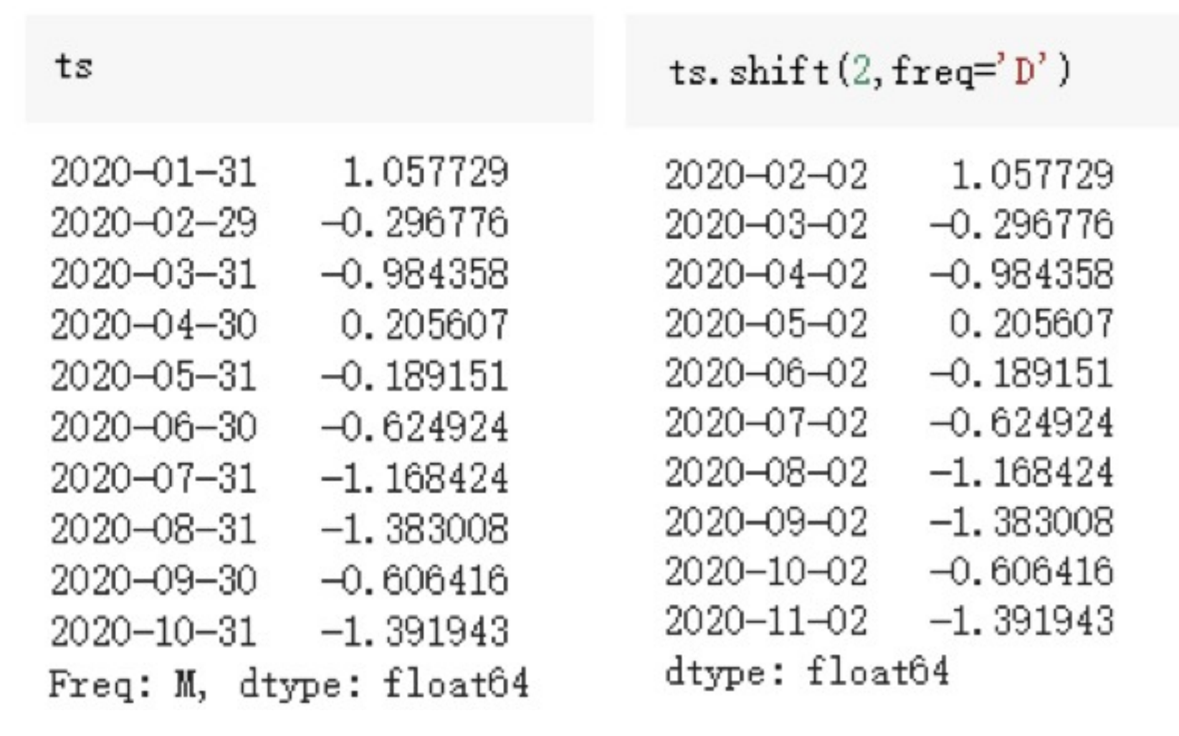
```
ts['2020-03-31': '2020-07-31']

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



```
ts['2020-02': '2020-05']

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。



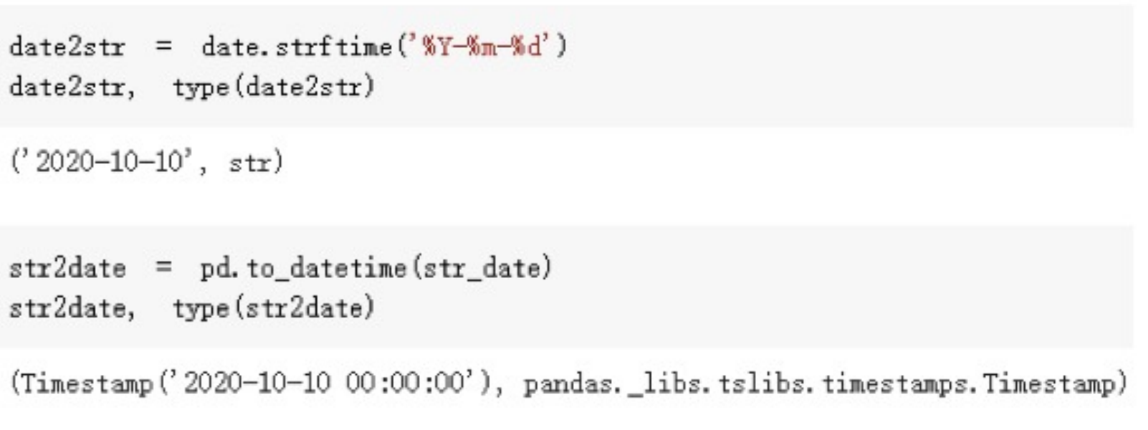
```
ts

2020-01-31    1.057729
2020-02-29   -0.296776
2020-03-31   -0.984358
2020-04-30    0.205607
2020-05-31   -0.189151
2020-06-30   -0.624924
2020-07-31   -1.168424
2020-08-31   -1.383008
2020-09-30   -0.606416
2020-10-31   -1.391943
Freq: M, dtype: float64
```

```
ts.shift(2,freq='D')

2020-02-02    1.057729
2020-03-02   -0.296776
2020-04-02   -0.984358
2020-05-02    0.205607
2020-06-02   -0.189151
2020-07-02   -0.624924
2020-08-02   -1.168424
2020-09-02   -1.383008
2020-10-02   -0.606416
2020-11-02   -1.391943
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)
```

- 時間跟字串可以互相轉換
  - 時間轉字串
  - 字串轉時間

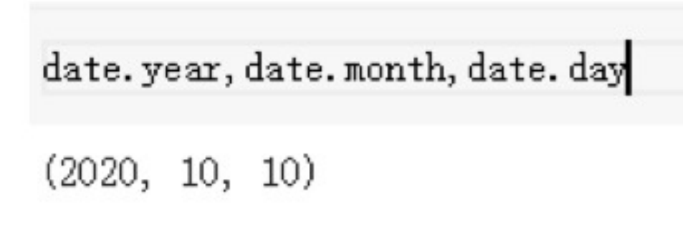


```
date2str = date.strftime('%Y-%m-%d')
date2str, type(date2str)

('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

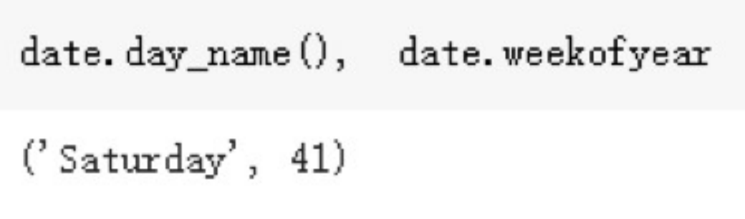
接下來介紹 timestamps 的常用函數

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



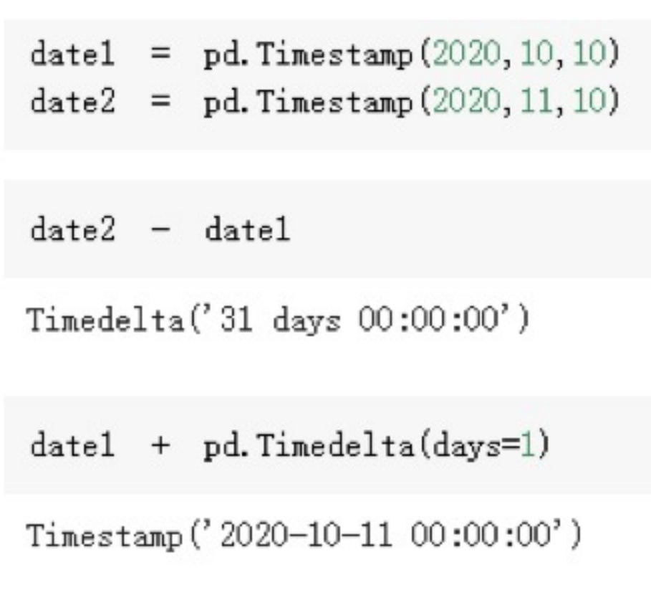
```
date.year,date.month,date.day

(2020, 10, 10)
```

- 也可以呼叫星期與周數



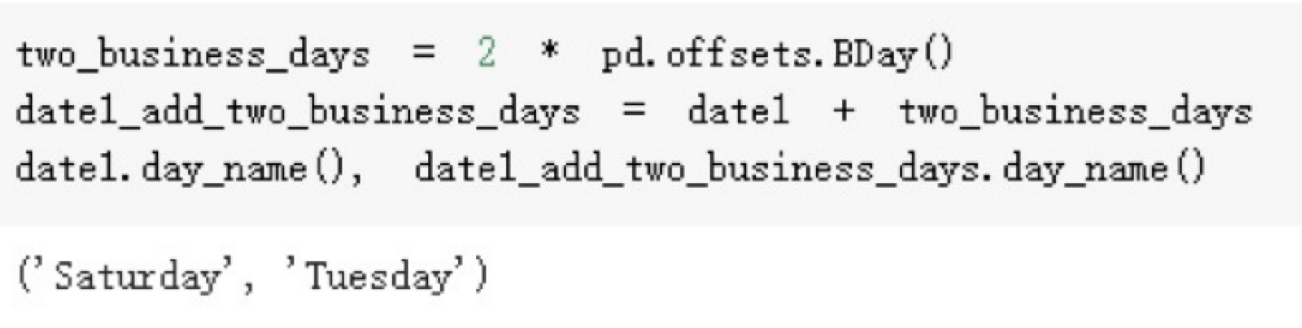
```
date.day_name(), date.weekofyear

('Saturday', 41)
```

- Timestamps 可以直接加時間或是計算時間差距



```
date1 = pd.Timestamp(2020,10,10)
date2 = pd.Timestamp(2020,11,10)

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

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人讚同了該文章

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.data_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.strftime()、pd.to_datetime( )、pd.to_period()

一、定義時間格式

1. pd.Timestamp()、pd.Timedelta()

(1) Timestamp時間戳

```
#定义timestamp
t1=pd.Timestamp('2019-01-10')
t2=pd.Timestamp('2018-12-10')
print(f't1= {t1}')
print(f't2= {t2}')
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
```

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

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

下一步：閱讀範例與完成作業