???????????

August 9, 2019

```
In [2]: import pandas as pd
        import numpy as np
        from datetime import datetime
        from pandas import Series, DataFrame

In [3]: ts1=Series(np.random.randn(3),index=pd.date_range('2012-6-13',periods=3, freq='W-WED'))

In [4]: ts1

Out[4]: 2012-06-13    0.137145
        2012-06-20   -0.579045
        2012-06-27   -0.234178
        Freq: W-WED, dtype: float64

In [5]: ts1.resample('B')

Out[5]: DatetimeIndexResampler [freq=<BusinessDay>, axis=0, closed=left, label=left, conventio

In [6]: ts1.resample('B',fill_method='ffill')

/Users/yuyangli/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning:
the new syntax is .resample(...).ffill()
  """Entry point for launching an IPython kernel.


Out[6]: 2012-06-13    0.137145
        2012-06-14    0.137145
        2012-06-15    0.137145
        2012-06-18    0.137145
        2012-06-19    0.137145
        2012-06-20   -0.579045
        2012-06-21   -0.579045
        2012-06-22   -0.579045
        2012-06-25   -0.579045
        2012-06-26   -0.579045
        2012-06-27   -0.234178
        Freq: B, dtype: float64

In [7]: rng=pd.date_range('2012-06-01 09:30', '2012-06-01 15:59', freq='T')
```

```
In [8]: rng=rng.append([rng+pd.offsets.BDay(i) for i in range(1,4)])

In [9]: ts=Series(np.arange(len(rng),dtype=float),index=rng)

In [10]: ts

Out[10]: 2012-06-01 09:30:00       0.0
         2012-06-01 09:31:00       1.0
         2012-06-01 09:32:00       2.0
         2012-06-01 09:33:00       3.0
         2012-06-01 09:34:00       4.0
         2012-06-01 09:35:00       5.0
         2012-06-01 09:36:00       6.0
         2012-06-01 09:37:00       7.0
         2012-06-01 09:38:00       8.0
         2012-06-01 09:39:00       9.0
         2012-06-01 09:40:00      10.0
         2012-06-01 09:41:00      11.0
         2012-06-01 09:42:00      12.0
         2012-06-01 09:43:00      13.0
         2012-06-01 09:44:00      14.0
         2012-06-01 09:45:00      15.0
         2012-06-01 09:46:00      16.0
         2012-06-01 09:47:00      17.0
         2012-06-01 09:48:00      18.0
         2012-06-01 09:49:00      19.0
         2012-06-01 09:50:00      20.0
         2012-06-01 09:51:00      21.0
         2012-06-01 09:52:00      22.0
         2012-06-01 09:53:00      23.0
         2012-06-01 09:54:00      24.0
         2012-06-01 09:55:00      25.0
         2012-06-01 09:56:00      26.0
         2012-06-01 09:57:00      27.0
         2012-06-01 09:58:00      28.0
         2012-06-01 09:59:00      29.0
                                  ...
         2012-06-06 15:30:00    1530.0
         2012-06-06 15:31:00    1531.0
         2012-06-06 15:32:00    1532.0
         2012-06-06 15:33:00    1533.0
         2012-06-06 15:34:00    1534.0
         2012-06-06 15:35:00    1535.0
         2012-06-06 15:36:00    1536.0
         2012-06-06 15:37:00    1537.0
         2012-06-06 15:38:00    1538.0
         2012-06-06 15:39:00    1539.0
         2012-06-06 15:40:00    1540.0
```

```
2012-06-06 15:41:00    1541.0
2012-06-06 15:42:00    1542.0
2012-06-06 15:43:00    1543.0
2012-06-06 15:44:00    1544.0
2012-06-06 15:45:00    1545.0
2012-06-06 15:46:00    1546.0
2012-06-06 15:47:00    1547.0
2012-06-06 15:48:00    1548.0
2012-06-06 15:49:00    1549.0
2012-06-06 15:50:00    1550.0
2012-06-06 15:51:00    1551.0
2012-06-06 15:52:00    1552.0
2012-06-06 15:53:00    1553.0
2012-06-06 15:54:00    1554.0
2012-06-06 15:55:00    1555.0
2012-06-06 15:56:00    1556.0
2012-06-06 15:57:00    1557.0
2012-06-06 15:58:00    1558.0
2012-06-06 15:59:00    1559.0
Length: 1560, dtype: float64
```

```
In [16]: #
```

```
In [19]: data1=DataFrame(np.ones((6,3),dtype=float),columns=['a','b','c'],index=pd.date_range(
```

```
In [21]: data2=DataFrame(np.ones((6,3),dtype=float)*2,columns=['a','b','c'],index=pd.date_range
```

```
In [24]: spliced=pd.concat([data1.ix[:'2012-06-14'],data2.ix['2012-06-15':]])
```

```
/Users/yuyangli/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: DeprecationWarni
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
```

```
In [25]: spliced
```

```
Out[25]:
            a    b    c
2012-06-12  1.0  1.0  1.0
2012-06-13  1.0  1.0  1.0
2012-06-14  1.0  1.0  1.0
2012-06-15  2.0  2.0  2.0
2012-06-16  2.0  2.0  2.0
2012-06-17  2.0  2.0  2.0
2012-06-18  2.0  2.0  2.0
```

In [26]:
```
#
# price[]/price[]-1
```

In [33]:
```
import random; random.seed(0)
```

In [34]:
```
import string
```

In [35]:
```
N=1000
def rands(n):
    choices = string.ascii_uppercase
    return ''.join([random.choice(choices) for _ in range(n)])
tickers=np.array([rands(5) for _ in range(N)])
```

In [36]:
```
M=500
df=DataFrame({'Momentum':np.random.randn(M)/200+0.03,
             'Value':np.random.randn(M)/200+0.08,
             'ShortInterest':np.random.randn(M)/200-0.02},
            index=tickers[:M])
```

In [37]:
```
ind_names=np.array(['FINANCIAL','TECH'])
sampler=np.random.randint(0, len(ind_names), N)
industries=Series(ind_names[sampler],index=tickers,name='industry')
```

In [38]:
```
by_industry=df.groupby(industries)
```

In [39]:
```
by_industry.mean()
```

Out[39]:
```
           Momentum     Value  ShortInterest
industry
FINANCIAL  0.030135  0.080097      -0.020274
TECH       0.029424  0.079972      -0.019990
```

In [40]:
```
by_industry.describe()
```

Out[40]:
```
          Momentum                                                        \
             count      mean      std       min       25%       50%       75%
industry
FINANCIAL    251.0  0.030135  0.00511  0.017602  0.026444  0.030331  0.033271
TECH         249.0  0.029424  0.00514  0.016540  0.026037  0.029207  0.032574

                       Value            ...                     ShortInterest  \
                max    count      mean  ...       75%       max          count
industry                                ...
FINANCIAL  0.043429    251.0  0.080097  ...  0.083140  0.095624          251.0
TECH       0.041857    249.0  0.079972  ...  0.083288  0.091457          249.0

                                                                        \
               mean       std       min       25%       50%       75%
industry
```

```
FINANCIAL -0.020274  0.004976 -0.033299 -0.023922 -0.020003 -0.017447
TECH      -0.019990  0.005383 -0.033550 -0.023573 -0.019829 -0.016190


                max
industry
FINANCIAL -0.005325
TECH      -0.006957

[2 rows x 24 columns]
```

In [41]:
```
def zscore(group):
    return (group-group.mean())/group.std()
```

In [42]:
```
df_stand=by_industry.apply(zscore)
```

In [43]:
```
df_stand.groupby(industries).agg(['mean','std'])
```

Out[43]:

	Momentum		Value		ShortInterest	
	mean	std	mean	std	mean	std
industry						
FINANCIAL	-4.359948e-15	1.0	4.488884e-15	1.0	-2.435745e-15	1.0
TECH	-9.647571e-16	1.0	-4.091328e-15	1.0	-1.243873e-15	1.0

In [44]:
```
ind_rank=by_industry.rank(ascending=False)
```

In [45]:
```
ind_rank.groupby(industries).agg(['min','max'])
```

Out[45]:

	Momentum		Value		ShortInterest	
	min	max	min	max	min	max
industry						
FINANCIAL	1.0	251.0	1.0	251.0	1.0	251.0
TECH	1.0	249.0	1.0	249.0	1.0	249.0

In [46]:
```
by_industry.apply(lambda x: zscore(x.rank()))
```

Out[46]:

	Momentum	Value	ShortInterest
MYNBI	1.527262	1.374536	1.402304
QPMZJ	-0.374873	-0.305452	1.291231
PLSGQ	1.235694	1.596683	-1.041315
EJEYD	0.606047	-0.881523	0.826427
TZIRW	-0.791400	-0.583137	-0.388758
ZTEJD	0.936618	1.019261	0.096417
XCVKP	1.541146	0.249916	-0.069421
RDLNK	-0.111074	-0.874705	1.166273
TUGRP	-0.385666	0.247928	-1.156998
OQIBZ	-1.666629	-1.377379	1.019261
RACXM	-0.902473	-1.638336	0.388758
WZVUA	1.487569	0.633594	-1.432474

TPKHX	1.088129	0.482083	0.261702
KWCGS	1.680403	-0.853975	1.239641
HHZEZ	1.542665	-0.137738	0.399440
ROCCK	-0.812654	1.363605	1.336058
QPDJR	0.485947	-1.013547	-1.249578
JWDRK	0.234154	1.652855	0.977939
RGZTR	-0.867749	1.487569	-0.041321
SJOCT	1.294736	0.537178	0.165285
ZMKSH	0.564725	1.556438	-1.280963
JFGFB	-0.916357	0.097189	-0.985778
TVIPC	-1.707757	0.999663	-0.694210
CVYEE	1.721724	-1.184546	-0.578499
BCWRV	-1.249578	0.624789	1.138505
MWQIQ	-0.971894	1.652220	-1.166273
ZHGVS	1.374536	-0.472063	-0.152726
NSIOP	1.583986	0.413214	1.501343
VUWZL	1.418700	-0.619821	0.330571
CKTDP	0.888589	-1.041315	1.207926
...	...	...	...
CCPMS	-0.964165	-0.192833	1.721724
NRCIP	0.222147	0.360989	1.513378
HDJEL	0.624789	-1.679989	0.777515
ZDEVB	0.482083	-0.440761	0.027548
VESRG	1.129451	0.661142	-1.363605
ABMRW	0.138842	1.235694	-0.902473
YTPWW	1.360652	0.735863	1.360652
DPRLY	-1.239641	-1.501343	-0.743785
KDVAH	-1.170772	-1.322284	-1.267189
HPJIH	1.291231	1.638336	0.805284
APLQK	-0.192833	-1.336058	1.473796
CCJSN	1.582799	-0.749747	1.666104
HXLMY	-0.330571	1.239641	0.840201
EHJGX	-0.347105	-1.707757	-0.291568
YPVZL	1.391153	0.606047	1.005487
JMTEY	-1.363605	1.143225	-0.936618
DMLQP	0.537178	0.716237	-0.909070
HUWLU	-0.220381	-0.468309	0.743785
LNILM	1.194041	-0.430410	-0.374873
YYWJD	-0.527600	-0.124958	0.874705
PJDOE	-0.399440	-0.330571	1.129451
LHXFK	-0.096417	-0.674916	1.253415
PHDVM	1.707757	1.555031	-1.610568
MOQOS	0.550952	-0.606047	-1.473796
THVMQ	1.239641	-0.261702	-1.143225
JPHKQ	0.137738	-1.088129	-1.005487
VACPK	1.336058	-1.418700	0.619821
MHNBS	0.206607	-0.110190	1.487569
YBNCI	-0.152726	-1.277347	-0.111074

```
         GXKFD -0.647368  0.564725       0.192833

         [500 rows x 3 columns]
```

```
In [48]: # !!!!!
```

```
In [49]: from numpy.random import rand
```

```
In [50]: fac1, fac2, fac3=np.random.rand(3,1000)
```

```
In [51]: ticker_subset=tickers.take(np.random.permutation(N)[:1000])
```

```
In [66]: port=Series(0.7*fac1-1.2*fac2+0.3*fac3+rand(1000), index=ticker_subset)
```

```
In [67]: factors=DataFrame({'f1':fac1, 'f2':fac2, 'f3':fac3}, index=ticker_subset)
```

```
In [68]: factors.corrwith(port)
```

```
Out[68]: f1    0.393140
         f2   -0.663512
         f3    0.186882
         dtype: float64
```

```
In [ ]:
```

```
In [ ]:
```