时间索引临时补充材料

胡俊峰 2024-03-19

原始数据表:

- 索引

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	82.45	82.55	80.81	82.06	11715200	IBM
1	2006-01-04	82.20	82.50	81.33	81.95	9840600	IBM
2	2006-01-05	81.40	82.90	81.00	82.50	7213500	IBM
3	2006-01-06	83.95	85.03	83.41	84.95	8197400	IBM
4	2006-01-09	84.10	84.25	83.38	83.73	6858200	IBM
5	2006-01-10	83.15	84.12	83.12	84.07	5701000	IBM
6	2006-01-11	84.37	84.81	83.40	84.17	5776500	IBM
7	2006-01-12	83.82	83.96	83.40	83.57	4926500	IBM
8	2006-01-13	83.00	83.45	82.50	83.17	6921700	IBM
9	2006-01-17	82.80	83.16	82.54	83.00	8761700	IBM

查找有空值的行?

```
1  IBM_df.isnull().any()   # 这样出来的是所有有空值的列
```

```
Date      False
Open       True
High      False
Low        True
Close     False
Volume    False
Name      False
dtype: bool
```

先转置一下，查到有空值的行的布尔序列

```
1  IBM_df.T.isnull().any()    # 每一行是否有空值的布尔索引
```

```
0       False
1       False
2       False
3       False
4       False
       ...
3015    False
3016    False
3017    False
3018    False
3019    False
Length: 3020, dtype: bool
```

显示表中有NULL值的所有行:

```
1  IBM_df.T.isnull().any()    # 每一行是否有空值的布尔索引
```

```
0       False
1       False
2       False
3       False
4       False
        ...
3015    False
3016    False
3017    False
3018    False
3019    False
Length: 3020, dtype: bool
```

布尔序列做行索引:

```
1  IBM_df[IBM_df.T.isnull().any()] # 直接做行索引取值
```

	Date	Open	High	Low	Close	Volume	Name
2913	2017-07-31	NaN	144.93	NaN	144.67	4355718	IBM

使用drop操作清理

```
1  IBM_df[IBM_df.isnull().T.any()]  # 效果同上
```

	Date	Open	High	Low	Close	Volume	Name
2913	2017-07-31	NaN	144.93	NaN	144.67	4355718	IBM

生成时间序列:

```
1  IBM_dr = pd.date_range("2006-01-03", "2017-12-31")
2  IBM_dr
```

```
DatetimeIndex(['2006-01-03', '2006-01-04', '2006-01-05', '2006-01-06',
               '2006-01-07', '2006-01-08', '2006-01-09', '2006-01-10',
               '2006-01-11', '2006-01-12',
               ...
               '2017-12-22', '2017-12-23', '2017-12-24', '2017-12-25',
               '2017-12-26', '2017-12-27', '2017-12-28', '2017-12-29',
               '2017-12-30', '2017-12-31'],
              dtype='datetime64[ns]', length=4381, freq='D')
```

这里是按天

注意长得像时间并不意味着是时间类型

```
idx1 = pd.DatetimeIndex(['2017/8/1','2018/8/2','2018-8-3','2018/8/4/','2018/8/5']) # 自动进行打包转换为日期-时间对象
data1 = pd.Series([1,2,3,4,5], index = idx1)
data1
```

```
2017-08-01    1
2018-08-02    2
2018-08-03    3
2018-08-04    4
2018-08-05    5
2019-09-12    6
2020-10-13    7
dtype: int64
```

```
idx2 = pd.Index(['2017/8/11/','2018/8/2','2018/8/08','2018-8-4','2018/8/5']) # 这里还不
data2 = pd.Series([1,2,3,4,5], index = idx2)
data2
```

```
2017/8/11/    1
2018/8/2      2
2018/8/08     3
2018-8-4      4
2018/8/5      5
dtype: int64
```

方案一：将表格中的时间数据转换为时间类型的数据

- 这样两个表对相同类型的字段就可以merge或join了
- 另一个思路是把原表中的日期字段指定为index，然后转成日期类型，然后再重新用新的日期索引加入：df.reindex(data_index)

```
1  df = pd.DataFrame({'Date': idx2})
2  df.Date = pd.to_datetime(df.Date)
3  df
```

	Date
0	2017-08-11
1	2018-08-02
2	2018-08-08
3	2018-08-04
4	2018-08-05

- 也可以将表格中的时间数据转换为字符串，然后跟日期串进行匹配，将时间序列数据转换为字符串要注意原时间格式能完全匹配

作业里用到的按月份聚合：

- 一种方案是用：groupby的pd.Grouper指定freq参数
- 另一种我看到的方法是在时间索引上指定变换为：index.strftime(“%Y-%m”)，这样就可以得到一个只有年月的新索引，然后group by就行了。

- 本次作业技术上有难点，有问题可以直接在群里交流，不要自己死磕