第11章、时间序列

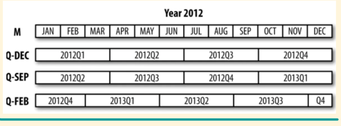
- 11.1 数据类型
  - datetime模块

类型	说明
date	以公历形式存储日历日期（年、月、日）
time	将时间存储为时、分、秒、毫秒
datetime	存储日期和时间
timedelta	表示两个datetime值之间的差（日、秒、毫秒）

  - 11.1.1 字符串与datetime转换
    - datetime转换为str
      - str(datetime对象)
      - datetime对象.strftime(格式)

代码	说明
%Y	4位数的年
%y	2位数的年
%m	2位数的月[01, 12]
%d	2位数的日[01, 31]
%H	时（24小时制）[00, 23]
%I	时（12小时制）[01, 12]
%M	2位数的分[00, 59]
%S	秒[00, 61]（秒60和61用于闰秒）
%w	用整数表示的星期几的（星期天）, 6]
%U	每年的第几周[00, 53]。星期天被认为是每周的第一天，每年第一个星期天之前的那几天被认为是“第0周”
%W	每年的第几周[00, 53]。星期一被认为是每周的第一天，每年第一个星期一之前的那几天被认为是“第0周”
%z	以+HHMM或－HHMM表示的UTC时区偏移量。如果时区为naive[译注3]，则返回空字符串
%F	%Y-%m-%d简写形式，例如2012-4-18[译注4]
%D	%m/%d/%y简写形式，例如04/18/12

    - str转datetime
      - datetime.strptime(str, 格式)
      - 不想写格式
        - from dateutil.parser import parse
        - parse(str)
      - pd.to_datetime(日期字符串数组)
- 11.2 时间序列
  - 把datetime对象当作索引，放入DatetimeIndex中
  - DatetimeIndex的标量值是Timestamp对象
  - 11.2.1 索引、选择、子集
    - stamp = ts.index[2]
      ts[stamp]
    - ts[日期字符串]
      - ts["20110110"]
    - ts[年/年月]
      - ts["2001"]
      - ts["2001-05"]
    - 切片
      - datetime对象
        - ts[datetime(2011, 1, 7):]
      - 字符串切片
        - ts['2011-06-01': '2011-11-01']
      - ts.truncate(before=日期字符串, after=, axis=)
    - df.loc["2001-05"]
  - 11.2.2 含重复索引
    - 聚合
      - grouped = dup_ts.groupby(level=0)
        grouped.mean()
- 11.3 日期范围、频率、移位
  - 时间序列频率不固定，转换为每日/月等使用resample
    - ts.resample('D')
  - 11.3.1 生成日期范围
    - pd.date_range(start=, end=, periods=, freq=, normalize=False标准化为零点)
      - 保留开始、结束的时间戳
  - 11.3.2 频率
    - freq = '4h'、'1h30min'、'90T'
    - 11.3.2.1 月中某星期(week of month)的日期
      - freq = 'WOM-3FRI'
        - 每月第三个周五
  - 11.3.3 移位
    - 将日期向前/向后移位
    - ts.shift(periods=1, axis=0, fill_value=None)
      - 迁移数据
      - 搭配freq参数，会迁移索引
    - ts.tshift(periods=, freq=, axis=)
      - 迁移索引
    - 11.3.3.1 偏置移位
      - from pandas.tseries.offsets import Day, MonthEnd
        offset = MonthEnd()
        offset.rollforward() # 向前滚动
        offset.rollback() # 向后滚动
      - ts.groupby(offset.rollforward).mean() ==
        ts.resample('M').mean()
- 11.4 时区处理
- 11.5 时间区间
  - 时间区间类型Period
    - pd.Period(value=, freq=)
      - p = pd.Period(2007, freq='A-DEC') # A-DEC表示以年为区间，12月份为年末
        p.start_time # Timestamp('2007-01-01 00:00:00')
        p.end_time # Timestamp('2007-12-31 23:59:59.999999999')



  - 构造规则区间序列
    - pd.period_range(start, end, periods, freq)
      - rng = pd.period_range('2000-01-01', '2000-06-30', freq='M')
        rng.start_time # DatetimeIndex序列
        rng.end_time # DatetimeIndex序列

```
DatetimeIndex(['2000-01-01', '2000-02-01', '2000-03-01', '2000-04-01',
               '2000-05-01', '2000-06-01'],
              dtype='datetime64[ns]', freq='MS')
```

    - 返回PeriodIndex类对象
  - 11.5.1 区间频率转化
    - 将区间和PeriodIndex对象转换成其他频率
      - p.asfreq('M', how='start')
        - 区间变成2007-01-01到2007-01-31
      - p.asfreq('M', how='end')
        - 区间变成2007-12-01到2007-12-31
  - 11.5.2 季度区间
    - p = pd.Period('2012Q4', freq='Q-JAN')
      (p.asfreq('B', 'e') - 1).asfreq('T', 's')+ 16 * 60
      # Period('2012-01-30 16:00', 'T')
  - 11.5.3 Period与Timestamp转换
    - Timestamp转换为Period
      - ts.to_period(freq=)
    - Period转换为Timestamp
      - p.to_timestamp(freq=, how='start'/'end')
  - 11.5.4 数组生成PeriodIndex
    - index = pd.PeriodIndex(year=, quarter=,
      month=, day=, hour=, minute=, second=, freq=)
- 11.6 重采样
  - 转换时间序列的频率
  - 高频率——>低频率：向下采样
    - 间隔1分钟的变为5分钟
  - 低频率——>高频率：向上采样
  - 11.6.1 向下采样
    - ts.resample(rule采样规则, closed='left'/'right'区间的闭合端,
      label='left'/'right'使用哪个边界标记区间, loffset=索引偏置，可用tshift实现)

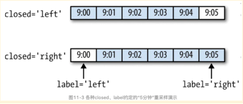


图11-3 各种closed、label的定时“5分钟”重采样图示

    - 11.6.1.1 OHLC（open、high、low、close）
      - ts.resample('5T').ohlc()
  - 11.6.2 向上采样
    - 在不聚合的情况下，将低频转换为高频
      - asfreq()
      - df.resample('D').asfreq()
    - 填充
      - ffill/bfill()
      - df.resample('D').ffill()
  - 11.6.3 使用区间进行重采样
    - df.resample('Q-DEC', convention='end').ffill()
- 11.7 窗口函数
  - s.rolling(window=移动窗口的大小, min_periods=窗口中最少包含的观察值个数, center=False用窗口的中心设为标签, on=df的列, axis=, closed='right', 'left', 'both' or 'neither')
    - 固定窗口大小，进行滑动计算
  - s.expanding(min_periods=1, center=False, axis=0)
    - 只设置最小的观测值数量，不固定窗口大小
    - 窗口大小从起始开始一直变化，类似cumsum
    - pd.DataFrame({'B': [0, 1, -2, np.nan, 1, 3, 4]}).expanding(2).sum()

```
     B
0  NaN
1  1.0
2 -1.0
3 -1.0
4  0.0
5  3.0
6  7.0
```

    - pd.DataFrame({'B': [0, 1, -2, np.nan, 1, 3, 4]}).rolling(2).sum()

```
     B
0  NaN
1  1.0
2 -1.0
3  NaN
4  NaN
5  4.0
6  7.0
```

  - 11.7.1 指数加权
    - s.ewm()
  - 11.7.2 二元移动窗口函数
    - 相关度、协方差
  - 11.7.3 自定义移动窗口函数
    - s.rolling().apply(自定义函数)
    - 自定义函数聚合每个数组，产生单值