



温州大學
WENZHOU UNIVERSITY

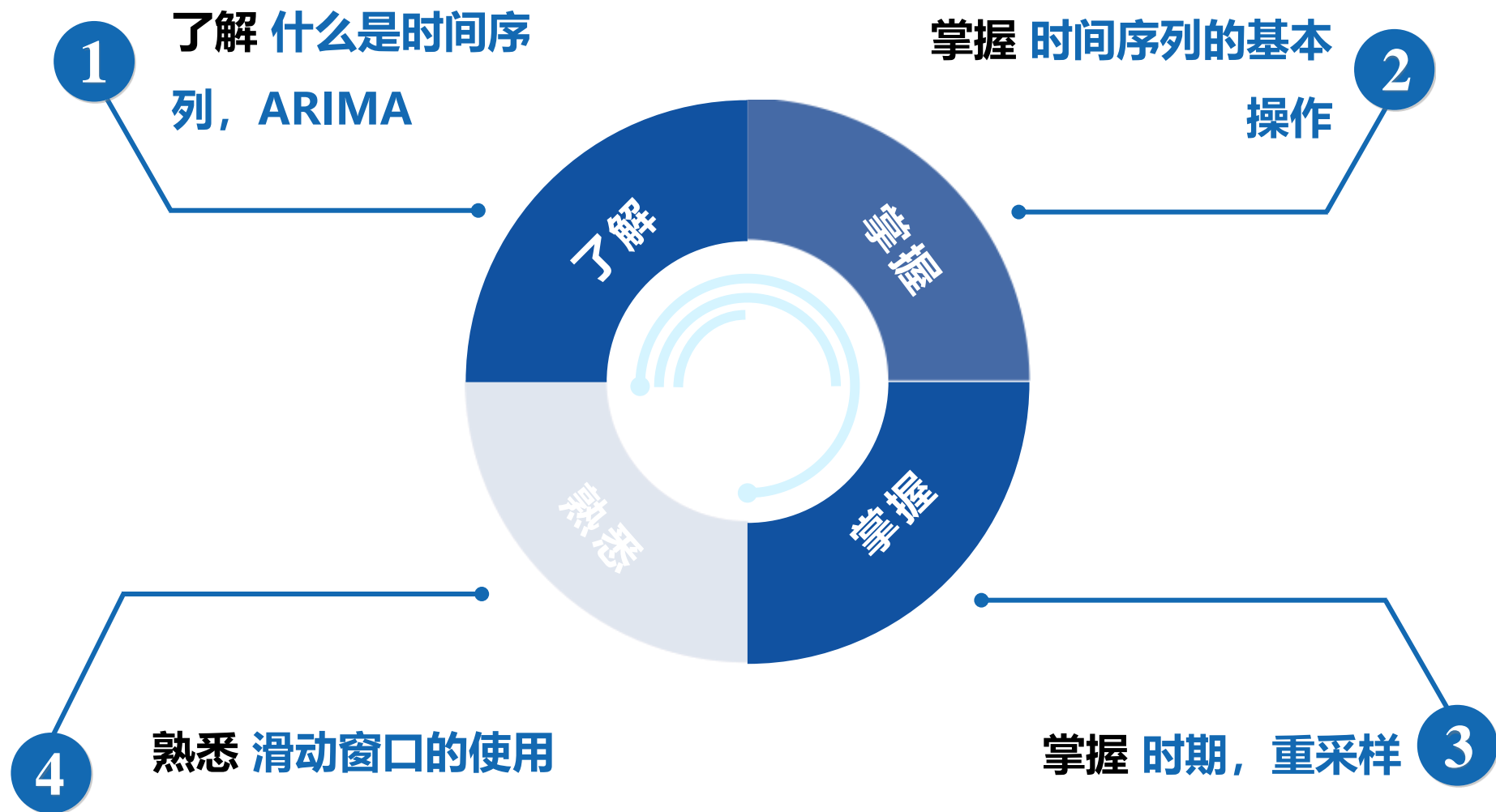
时间序列总结

黄海广 副教授

2022年01月

学习目标

2



目录

3



01 时间序列的基本操作

02 固定频率的时间序列

03 时间周期及计算

04 重采样

05 数据统计—滑动窗口

06 时序模型—**ARIMA**

1.时间序列的基本操作

4



01 时间序列的基本操作

02 固定频率的时间序列

03 时间周期及计算

04 重采样

05 数据统计—滑动窗口

06 时序模型—**ARIMA**

思考：
什么是时间序列？



时间序列的概念

6

时间序列是指多个时间点上形成的数值序列，它既可以是定期出现的，也可以是不定期出现的。



时间序列的数据种类

7

时间序列的数据主要有以下几种：

表示特定的时刻
， 比如现在

时间戳

比如2018年或者
2018年10月

时期

由起始时间戳和
结束时间戳表示

时间间隔

创建时间序列

8

Pandas中，时间戳使用Timestamp（Series派生的子类）对象表示。

该对象与datetime具有高度的兼容性，可以直接通过to_datetime()函数将datetime转换为TimeStamp对象。

```
pd.to_datetime('20180828')
```


创建时间序列

9

如果传入的是多个datetime组成的列表，则Pandas会将其强制转换为DatetimeIndex类对象。

```
date_index = pd.to_datetime(['20180820','20180828',  
                             '20180908'])
```

```
DatetimeIndex(['2018-08-20',  
                '2018-08-28', '2018-09-08'],  
               dtype='datetime64[ns]',  
               freq=None)
```

创建时间序列

10

在Pandas中，最基本的时间序列类型就是以时间戳为索引的Series对象。

```
date_ser = pd.Series([11, 22, 33], index=date_index)
```



2018-08-20	11
2018-08-28	22
2018-09-08	33

创建时间序列

11

还可以将包含多个datetime对象的列表传给index参数，同样能创建具有时间戳索引的Series对象。

```
date_list = [datetime(2018, 1, 1), datetime(2018, 1, 15)]  
time_se = pd.Series(np.arange(6), index=date_list)
```

创建时间序列

12

如果希望DataFrame对象具有时间戳索引，也可以采用上述方式进行创建。

```
data_demo = [[11, 22, 33], [44, 55, 66]]  
date_list = [datetime(2018, 1, 23), datetime(2018, 2, 15)]  
time_df = pd.DataFrame(data_demo, index=date_list)
```

通过时间戳索引选取子集

13

最简单的选取子集的方式，是直接使用位置索引来获取具体的数据。

```
# 根据位置索引获取数据  
time_se[3]
```

通过时间戳索引选取子集

14

还可以使用datetime构建的日期获取其对应的数据。

```
date_time = datetime(2015, 6, 1)
date_se[date_time]
```

通过时间戳索引选取子集

15

还可以在操作索引时，直接使用一个日期字符串（符合可以被解析的格式）进行获取。

```
date_se['20150530']
```

```
date_se['2018/01/23']
```


通过时间戳索引选取子集

16

如果希望获取某年或某个月的数据，则可以直接用指定的年份或者月份操作索引。

```
date_se['2015']
```

通过时间戳索引选取子集

17

除了使用索引的方式以外，还可以通过 `truncate()` 方法截取 `Series` 或 `DataFrame` 对象。

```
truncate(before = None, after = None,  
axis = None, copy = True)
```

- `before` -- 表示截断此索引值之前的所有行。
- `after` -- 表示截断此索引值之后的所有行。
- `axis` -- 表示截断的轴，默认为行索引方向

2.固定频率的时间序列

18



01 时间序列的基本操作

02 固定频率的时间序列

03 时间周期及计算

04 重采样

05 数据统计—滑动窗口

06 时序模型—**ARIMA**

创建固定频率的时间序列

19

Pandas中提供了一个date_range()函数，主要用于生成一个具有固定频率的DatetimeIndex对象。

```
date_range(start = None, end = None, periods = None,  
freq = None, tz = None, normalize = False, name = None,  
closed = None, ** kwargs)
```

- start: 表示起始日期，默认为None。
- end: 表示终止日期，默认为None。
- periods: 表示产生多少个时间戳索引值。
- freq: 用来指定计时单位。

创建固定频率的时间序列

20



start、end、periods、freq这四个参数
至少要指定三个参数，否则会出现错误。

创建固定频率的时间序列

21

当调用`date_range()`函数创建`DatetimeIndex`对象时，如果只是传入了开始日期（`start`参数）与结束日期（`end`参数），则默认生成的时间戳是按天计算的，即`freq`参数为`D`。

```
pd.date_range('2018/08/10', '2018/08/20')
```

创建固定频率的时间序列

22

如果只是传入了开始日期或结束日期，则还需要用periods参数指定产生多少个时间戳。

```
pd.date_range(start='2018/08/10', periods=5)
```

```
pd.date_range(end='2018/08/10', periods=5)
```


创建固定频率的时间序列

23

如果希望时间序列中的时间戳都是每周固定的星期日，则可以在创建DatetimeIndex时将freq参数设为“W-SUN”。

```
dates_index = pd.date_range('2018-01-01',  
                             periods=5, freq='W-SUN')
```

创建固定频率的时间序列

24

如果日期中带有与时间相关的信息，且想产生一组被规范化到当天午夜的时间戳，可以将`normalize`参数的值设为`True`。

```
pd.date_range(start='2018/8/1 12:13:30',  
periods=5, normalize=True, tz='Asia/Hong_Kong')
```

时间序列的频率、偏移量

25

默认生成的时间序列数据是按天计算的，即频率为 “D” 。

- “D” 是一个基础频率，通过用一个字符串的别名表示，比如 “D” 是 “day” 的别名。
- 频率是由一个基础频率和一个乘数组成的，比如，“5D” 表示每5天。

时间序列的频率、偏移量

26

通过一张表来列举时间序列的基础频率。

别名	说明
D	每日历日
B	每工作日
H	每小时
T 或 min	每分
S	每秒
L 或 ms	每毫秒
U	每微秒
M	每月最后一个日历日
BM	每月最后一个工作日
MS	每月第一个日历日
BMS	每月第一个工作日
W-MON、W-TUE...	从指定的星期几（MON、TUE、WED、THU、FRI、SAT、SUN）开始算起，每周几
WOM-1MON、WOM-2MON...	每月第一周、第二周、第三周或第四周的星期几。

时间序列的频率、偏移量

27

通过一张表来列举时间序列的基础频率。

别名	说明
Q-JAN、Q-FEB...	对于以指定月份（JAN、FEB、MAR、APR、MAY、JUN、JUL、AUG、SEP、OCT、NOV、DEC）结束的年度，每季度最后一月的最后一个日历日
BQ-JAN、BQ-FEB...	对于以指定月份结束的年度，每季度最后一月的最后一个工作日
QS-JAN、QS-FEB...	对于以指定月份结束的年度，每季度最后一月的最后一个日历日
BQS-JAN、BQS-FEB...	对于以指定月份结束的年度，每季度最后一月的第一个工作日
A-JAN、A-FEB...	每年指定月份的最后一个日历日
BA-JAN、BA-FEB...	每年指定月份的最后一个工作日
AS-JAN、AS-FEB...	每年指定月份的第一个日历日
BAS-JAN、BAS-FEB...	每年指定月份的第一个工作日

时间序列的频率、偏移量

28

每个基础频率还可以跟着一个被称为日期偏移量的DateOffset对象。如果想要创建一个DateOffset对象，则需要先导入pd.tseries.offsets模块后才行。

```
from pandas.tseries.offsets import *  
DateOffset(months=4, days=5)
```

还可以使用offsets模块中提供的偏移量类型进行创建。

例如，创建14天10小时的偏移量，可以换算为两周零十个小时，其中“周”使用Week类型表示的，“小时”使用Hour类型表示，它们之间可以使用加号连接。

$$\text{Week}(2) + \text{Hour}(10)$$

时间序列的移动

30

移动是指沿着时间轴方向将数据进行前移或后移。

数据移动前

2018-01-01	1
2018-01-02	2
2018-01-03	3
2018-01-04	4
2018-01-05	5

数据移动后

2018-01-01	2
2018-01-02	3
2018-01-03	4
2018-01-04	5
2018-01-05	NaN

向前移动

数据移动后

2018-01-01	NaN
2018-01-02	1
2018-01-03	2
2018-01-04	3
2018-01-05	4

向后移动

时间序列的移动

31

Pandas对象中提供了一个shift()方法，用来前移或后移数据，但数据索引保持不变。

```
shift(periods=1, freq=None, axis=0)
```

- `periods` -- 表示移动的幅度，可以为正数，也可以为负数，默认值是1，代表移动一次。

3.时间周期及计算

32



01 时间序列的基本操作

02 固定频率的时间序列

03 时间周期及计算

04 重采样

05 数据统计—滑动窗口

06 时序模型—**ARIMA**

创建时期对象

33

Period类表示一个标准的时间段或时期，比如某年、某月、某日、某小时等。

创建Period类对象的方式比较简单，只需要在构造方法中以字符串或整数的形式传入一个日期即可。

```
pd.Period(2018)
```

```
pd.Period('2017/6')
```

创建时期对象

34

Period对象能够参与数学运算。如果Period对象加上或者减去一个整数，则会根据具体的时间单位进行位移操作，

```
period = pd.Period('2017/6')  
period + 1
```

Period('2017-07', 'M')

创建时期对象

35

如果具有相同频率的两个Period对象进行数学运算，那么计算结果为它们的单位数量。

```
pd.Period('2017/6')  
other_period = pd.Period(201201, freq='M' )  
period - other_period
```

65

创建时期对象

36

如果希望创建多个Period对象，且它们是固定出现的，则可以通过period_range()函数实现。

```
period_index = pd.period_range('2012.1.8',  
                                '2012.3.31', freq='M')
```

```
PeriodIndex(['2012-01', '2012-02', '2012-03'],  
            dtype='period[M]', freq='M')
```


创建时期对象

37

上述示例返回了一个PeriodIndex对象，它是由一组时期对象构成的索引。

Period对象1
Period对象2
...
Period对象N

除了使用上述方式创建PeriodIndex外，还可以直接在PeriodIndex的构造方法中传入一组日期字符串。

```
str_list = ['2010', '2011', '2012']  
pd.PeriodIndex(str_list, freq='A-DEC')
```

创建时期对象

39



DatetimeIndex是用来指代一系列时间点的一种索引结构，而PeriodIndex则是用来指代一系列时间段的索引结构。

时期的频率转换

40

Pandas中提供了一个`asfreq()`方法来转换时期的频率。

```
asfreq (freq, method = None, how = None, normalize  
= False, fill_value = None )
```

- `freq` -- 表示计时单位。
- `how` -- 可以取值为`start`或`end`，默认为`end`。
- `normalize` -- 表示是否将时间索引重置为午夜。
- `fill_value` -- 用于填充缺失值的值。

4.重采样

41



01 时间序列的基本操作

02 固定频率的时间序列

03 时间周期及计算

04 重采样

05 数据统计—滑动窗口

06 时序模型—**ARIMA**

重采样方法 (resample)

42

Pandas中的resample()是一个对常规时间序列数据重新采样和频率转换的便捷的方法。

```
resample(rule, how=None, axis=0, fill_method=None, closed=None, label=None, ...)
```

- rule -- 表示重采样频率的字符串或DateOffset。
- fill_method -- 表示升采样时如何插值。
- closed -- 设置降采样哪一端是闭合的。

重采样方法 (resample)

43

例如，通过resample()方法对数据重新采样。

```
time_ser.resample('W-MON').mean()
```

how参数不再建议使用，而是采用新的方式
“`.resample(...).mean()`”求平均值。

重采样方法 (resample)

44

如果重采样时传入closed参数为left，则表示采样的范围是左闭右开型的。

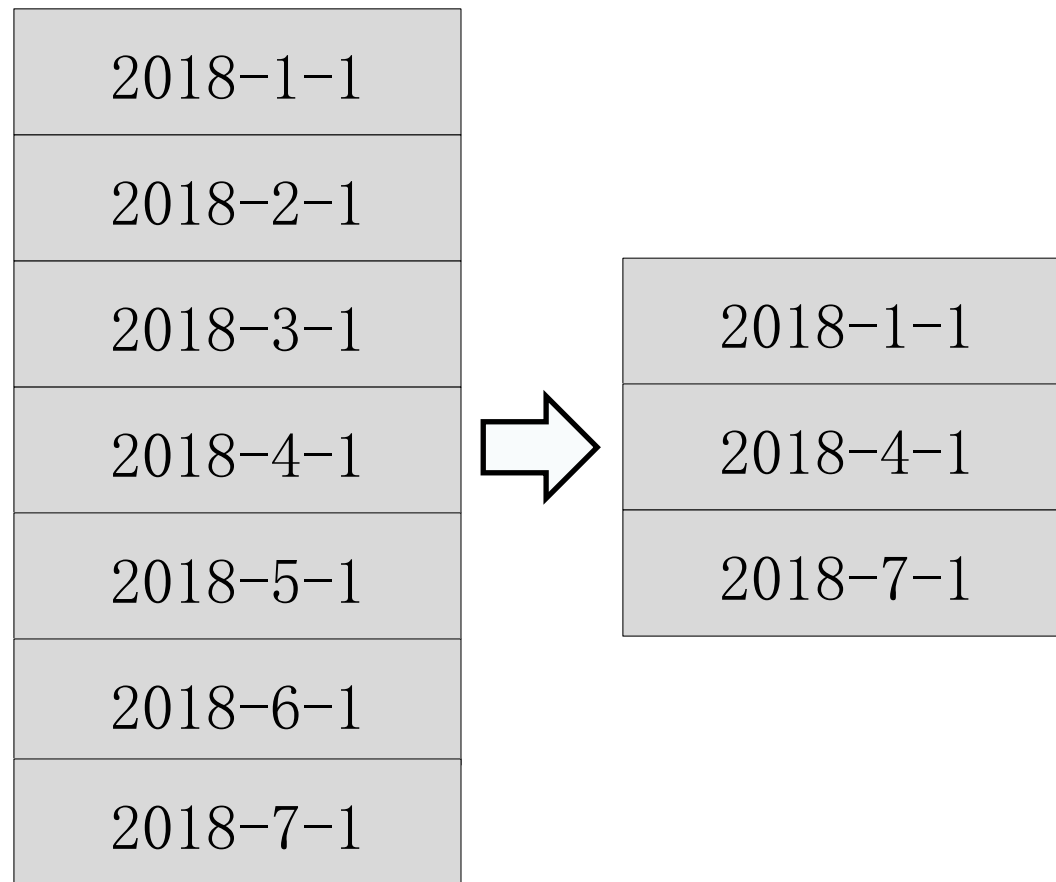
换句话说位于某范围的时间序列中，开头的时间戳包含在内，结尾的时间戳是不包含在内的。

```
time_ser.resample('W-MON', closed='left').mean()
```


降采样

45

降采样时间颗粒会变大，数据量是减少的。为了避免有些时间戳对应的数据闲置，可以利用内置方法聚合数据。



股票数据比较常见的是OHLC重采样，包括开盘价、最高价、最低价和收盘价。

Pandas 中专门提供了一个 `ohlc()` 方法。

```
date_index = pd.date_range('2018/06/01', periods=30)
shares_data = np.random.rand(30)
time_ser = pd.Series(shares_data, index=date_index)
time_ser.resample('7D').ohlc()
```

降采样

47

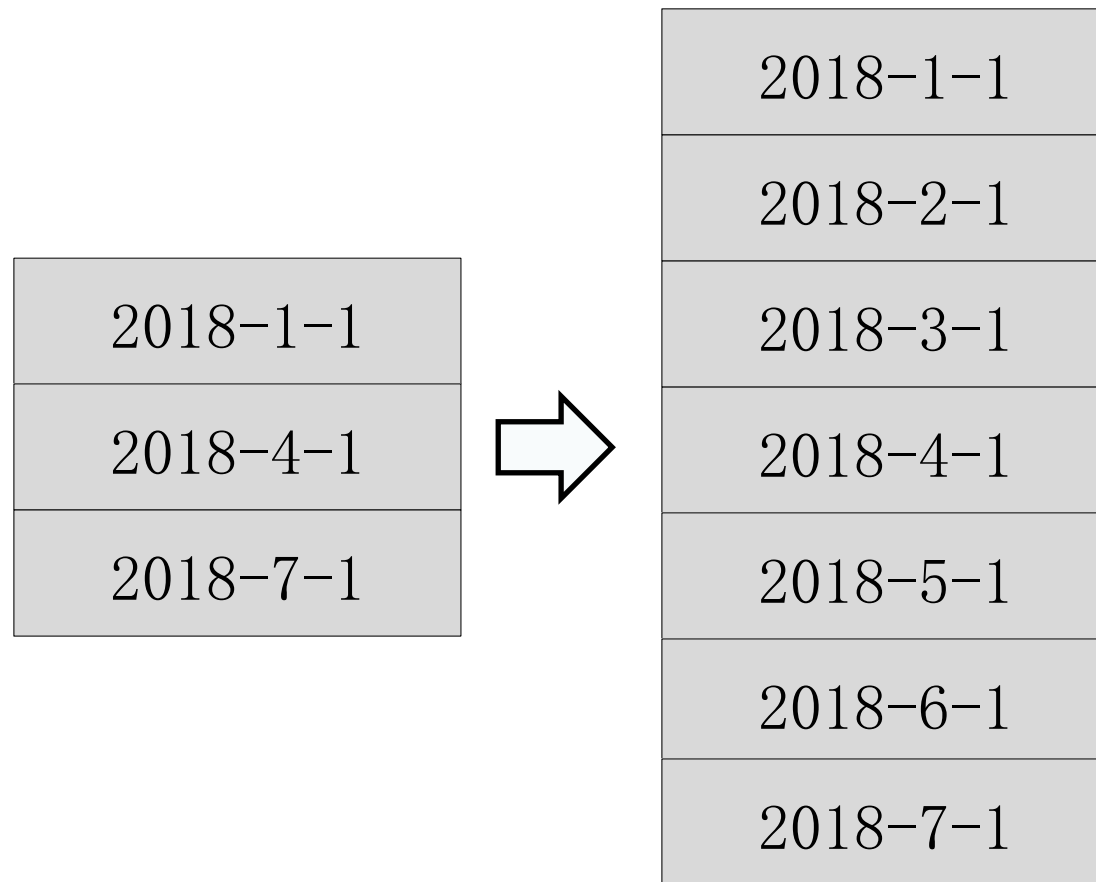
降采样相当于另外一种形式的分组操作，它会按照日期将时间序列进行分组，之后对每个分组应用聚合方法得出一个结果。

```
time_ser.groupby(lambda x: x.week).mean()
```

升采样

48

升采样的时间颗粒是变小的，数据量会增多，这很有可能导致某些时间戳没有相应的数据。



遇到这种情况，常用的解决办法就是插值，具体有如下几种方式：

- 通过`ffill(limit)`或`bfill(limit)`方法，取空值前面或后面的值填充，`limit`可以限制填充的个数。
- 通过`fillna('ffill')`或`fillna('bfill')`进行填充，传入`ffill`则表示用NaN前面的值填充，传入`bfill`则表示用后面的值填充。
- 使用`interpolate()`方法根据插值算法补全数据。

5.数据统计—滑动窗口

50



01 时间序列的基本操作

02 固定频率的时间序列

03 时间周期及计算

04 重采样

05 数据统计—滑动窗口

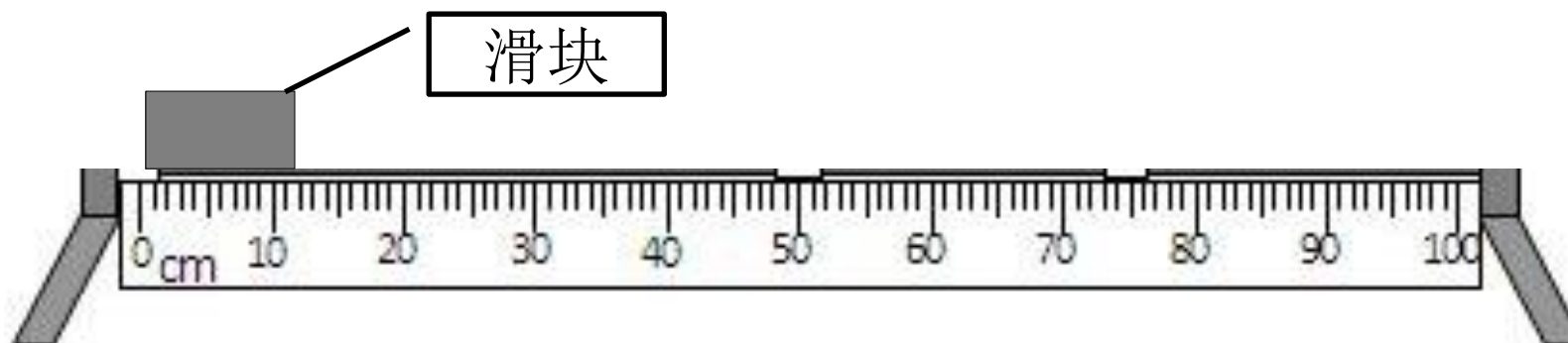
06 时序模型—ARIMA

数据统计—滑动窗口

51

滑动窗口指的是根据指定的单位长度来框住时间序列，从而计算框内的统计指标。

相当于一个长度指定的滑块在刻度尺上面滑动，每滑动一个单位即可反馈滑块内的数据。

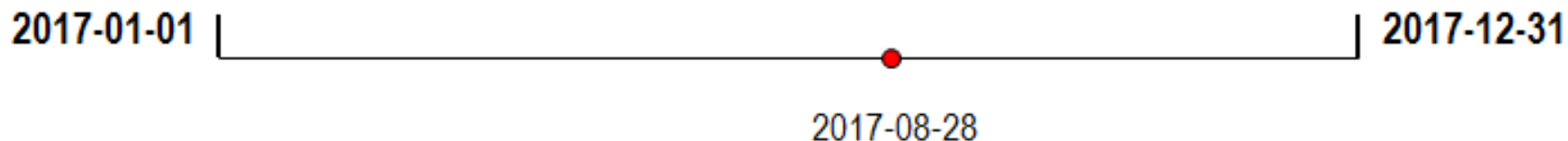


数据统计—滑动窗口

52

滑动窗口的概念比较抽象，下面我们来举个例子描述一下。

某分店按天统计了2017年全年的销售数据，现在总经理想抽查分店8月28日（七夕）的销售情况，如果只是单独拎出来当天的数据，则这个数据比较绝对，无法很好地反映出这个日期前后销售的整体情况。

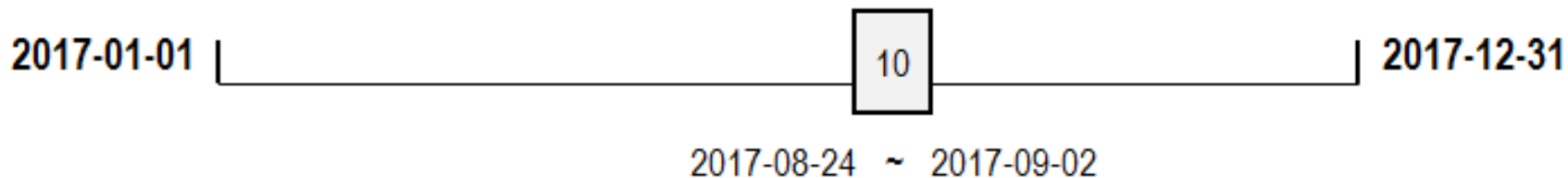


数据统计—滑动窗口

53

为了提升数据的准确性，可以将某个点的取值扩大到包含这个点的一段区间，用区间内的数据进行判断。

例如，我们可以将8月24日到9月2日的数据拿出来，求此区间的平均值作为抽查结果。



数据统计—滑动窗口

54



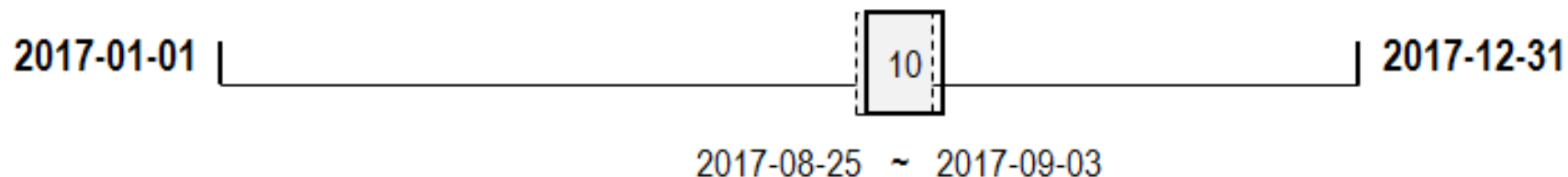
这个区间就是窗口，它的单位长度为10，数据是按天统计的，所以统计的是10天的平均指标，这样显得更加合理，可以很好地反映了七夕活动的整体情况。

6.数据统计—滑动窗口

55

移动窗口就是窗口向一端滑行，每次滑行并不是区间整块的滑行，而是一个单位一个单位的滑行。

例如，窗口向右边滑行一个单位，此时窗口框住的时间区间范围为2017-08-25到2017-09-03。



数据统计—滑动窗口

56



每次窗口移动，一次只会移动一个单位的长度，并且窗口的长度始终为10个单位长度，直至移动到末端。

由此可知，通过滑动窗口统计的指标会更加平稳一些，数据上下浮动的范围会比较小。

Pandas中提供了一个窗口方法rolling()。

```
rolling(window, min_periods=None, center=False, win_type=None, on=None, axis=0, closed=None)
```

- window -- 表示窗口的大小。
- min_periods -- 每个窗口最少包含的观测值数量。
- center -- 是否把窗口的标签设置为居中。
- win_type -- 表示窗口的类型。
- closed -- 用于定义区间的开闭。

时序模型—ARIMA

58



01 时间序列的基本操作

02 固定频率的时间序列

03 时间周期及计算

04 重采样

05 数据统计—滑动窗口

06 时序模型—ARIMA

时序模型—ARIMA

59

思考：

什么是ARIMA模型？



时序模型—ARIMA

60

ARIMA的全称叫做差分整合移动平均自回归模型，又称作整合移动平均自回归模型，是一种用于时间序列预测的常见统计模型。

记作：

$$\text{ARIMA}(p,d,q)$$

时序模型—ARIMA

61

ARIMA模型主要由AR、I与MA模型三个部分组成。

AR(p) 模型

I 模型

MA(q) 模型

时序模型—ARIMA

62

ARIMA(p,d,q)模型可以表示为：

$$\left(1 - \sum_{i=1}^p \phi_i L^i\right) (1 - L)^d X_t = \left(1 + \sum_{i=1}^q \theta_i L^i\right) \varepsilon_t$$

- p--代表预测模型中采用的时序数据本身的滞后数，即自回归项数。
- d--代表时序数据需要进行几阶差分化，才是稳定的，即差分的阶数。
- q--代表预测模型中采用的预测误差的滞后数，即滑动平均项数。

时序模型—ARIMA

63

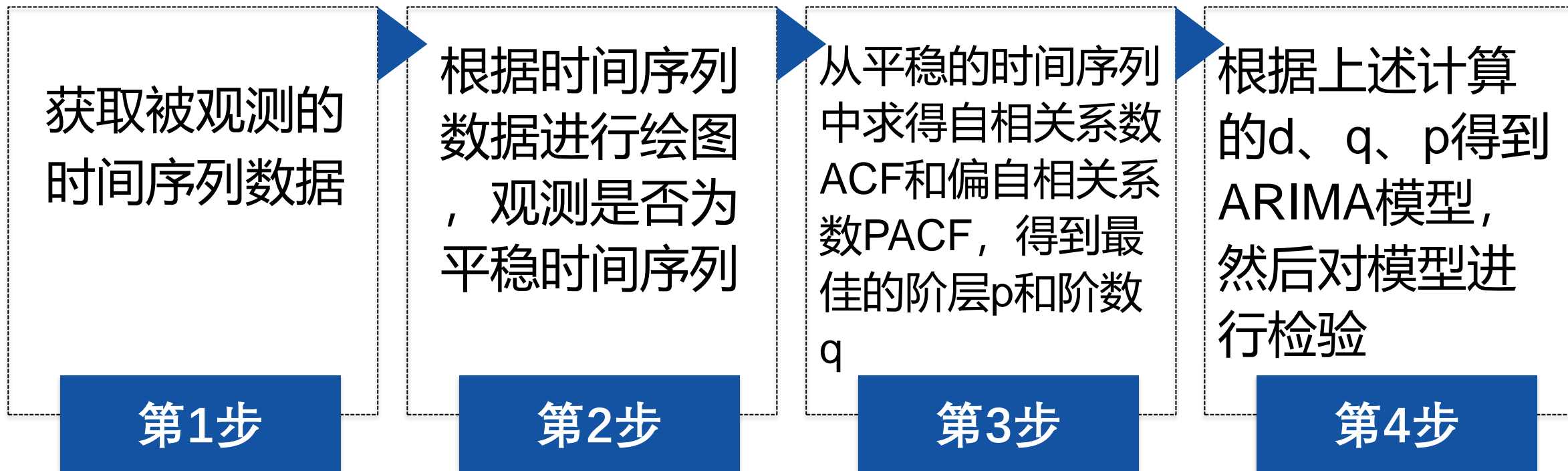


ARIMA模型的基本思想是：将预测对象随时间推移而形成的数据序列视为一个随机序列，用一定的数学模型来近似描述这个序列，这个模型一旦被识别后，就可以从时间序列的过去值及现在值来**预测未来值**。

时序模型—ARIMA

64

ARIMA模型建立的基本步骤如下：



时序模型—ARIMA

65



对于一个时间序列来说，如果它的均值没有系统的变化（无趋势），方差没有系统变化，并且严格消除了周期性的变化，就称为是平稳的。

本章小结

66

- 本章主要介绍了Pandas中用于处理时间序列的相关内容，包括创建时间序列、时间戳索引和切片操作、固定频率的时间序列、时期及计算、重采样、滑动窗口和时序模型，最后开发了一个股票预测分析的案例。
- 通过对本章内容的学习，读者应该掌握处理时间序列数据的一些技巧，并灵活加以运用。

谢谢!

