Pandas基础课程

目录

Pandas 简介

Pandas的名称来自于面板数据（panel data）和python数据分析（data analysis）。

Pandas 最初由AQR Capital Management于2008年4月开发，并于2009年底开源出来，主要提供高性能易用数据类型和分析工具

AQR Capital Management

创始人*Cliff Asnester*

Pandas的优点有哪些？

- 可以轻易的处理浮点及非浮点数据类型的缺失值(NaN)
- 基于智能标签的切片，花式索引，轻易从大数据集中取出子集
- 可以灵活处理时间数据
- 灵活强大的分组功能，可对数据集进行拆分组合操作
- 方便的将其他Python和Pandas数据结构中不同类索引的数据转换为DataFrame对象
- 轴(axes)的分层标签（使每个元组有多个标签成为可能）

Pandas预备知识

- Pandas一般默认简写：pd
- Pandas有两种主要的数据类型：Series和DataFrame
  - Series可以理解成一种具有索引的数组
  - DateFrame是由共用相同索引的一组列组成的
- Pandas同样适用广播机制

	A	B	C
1	1	2	3
2	4	5	6
3	7	8	9

部分excel表

ndarray与Series的区别

	这	是	列	素	引
这	0.612501	0.592796	0.773207	0.504646	0.677181
是	0.756287	0.578716	0.651797	0.804316	NaN
行	0.310771	0.070035	0.050099	0.812016	0.298376
素	0.319321	0.600389	0.013632	0.441274	0.175991
引	0.279503	0.565244	0.154203	0.523429	0.852456

DateFrame数据示例

Series函数相关知识

Series函数 同样适用ndarray的操作，但是输出结果仍是Series类型

pandas.Series(*self, data=None, index=None, dtype=None, name=None, copy=False, fastpath=False***)**

Data： Series不改变数据的类型

Index： Series可以自定义索引，索引关键词可以使用符号或汉字

创建Series的几种方法

Python列表

Python字典

其他

标量值（必须有index）

ndarray

DateFrame函数练习

DataFrame函数

pandas.DataFrame(*data=None, index=None, columns=None, dtype=None, copy=False***)**

index： index定义行索引

columns： columns定义列索引

创建DataFrame的几种方法

二维ndarray

由一维ndarray、列表、字典、元组、或者Series构成的字典

Series类型

其他DataFrame类型

DataFrame查看功能

函数	描述
DataFrame()	创建一个DataFrame对象
DataFrame.values	返回ndarray类型的对象
DataFrame.index	获取行索引
DataFrame.columns	获取列索引
DataFrame.axes	获取行及列索引
DataFrame.T	行与列对调
DataFrame. info()	打印DataFrame对象的信息
DataFrame.head(i)	显示前 i 行数据
DataFrame.tail(i)	显示后 i 行数据
DataFrame.describe()	查看数据按列的统计信息

Series和DataFrame索引操作

pandas数据结构的索引与ndarray的索引类似，但是pandas数据的索引不仅仅可以是数字，还可以是字符串。与数组不同的是，Sreies单个索引提取值、多个索引提取Series数据，且索引提取左闭右闭。

pandas的索引可以支持更多的功能，每个索引都是一个不可变的集合，它让数据的传递更加的安全，索引支持一些集合的方法和属性，可以辅助解决常见问题。

方法	描述
in	判断索引是否存在
append	将额外的对象粘贴到原索引后，产生一个新索引
difference\union\intersection	计算两个索引的差集\并集\交集
isin	计算表示每一个值是否在传值容器的布尔数组
delete	将位置i的元素删除，并产生新的索引
drop	根据传入参数删除指定索引，并产生新的索引
insert	将位置i插入元素，并产生新的索引
unique	计算索引的唯一值序列

Series和DataFrame基础操作（一）

Pandas需要通过改变索引进行增加和删除内部数据

添加列数据可以直接进行insert函数进行添加

Dataframe.insert(*loc, column, value, allow_duplicates=False***)**

loc： 使用整数定义_列数据_插入的位置，必须是0到columns列标签的长度

column： 可选字符串、数字或者object；列标签名

添加行数据需要使用reindex函数创建一个符合新索引的新对象。

DataFrame.reindex(*labels=None, index=None, columns=None, axis=None, method=None, copy=True, level=None, fill_value=nan, limit=None, tolerance=None***)**

index, columns： 新的行列自定义索引

fill_value： 重新索引中用于填充缺失位置的值

method： 填充方法，ffill当前值向前填充，bfill向后填充

Series和DataFrame基础操作（二）

对于已经建立了的Series和DataFrame数据可以通过drop函数删除指定行列数据

DataFrame.drop (*labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise'* **)**

labels： 就是要删除的行列的名字，用列表给定

axis： 默认为0，指删除行，因此删除columns时要指定axis=1

index： 直接指定要删除的行

columns： 直接指定要删除的列

inplace=False： 默认该删除操作不改变原数据，而是返回一个执行删除操作后的新dataframe

inplace=True： 则会直接在原数据上进行删除操作，删除后无法返回

Series和DataFrame基础操作（三）

Series和DataFrame数据直接进行计算时，数据会根据索引进行计算，对于不匹配的数据直接填充NAN。

	a	b	c
a	0.87	0.23	0.50
b	1.00	0.43	0.11

	b	c	d
b	0.21	0.57	0.34
c	0.06	0.21	0.74

	a	b	c	d
a	NaN	NaN	NaN	NaN
b	NaN	0.64	0.68	NaN
c	NaN	NaN	NaN	NaN

Series和DataFrame数据也支持根据条件判断进行数据选择。

	a	b	c
a	0.94	0.65	0.19
b	0.33	0.19	0.48
c	0.56	0.01	0.66

>0.5

	a	b	c
a	True	True	False
b	False	False	False
c	True	False	True

重新赋值

	a	b	c
a	6.00	6.00	0.19
b	0.33	0.19	0.48
c	6.00	0.01	6.00

Series和DataFrame基础操作（四）

索引排序

Series.sort_index(*axis=0, ascending=True, inplace=False,…***)**

axis： 轴直接排序。对于系列，只能为0

ascending： 升序与降序排序

inplace： 如果为True，则就地执行操作

值排序

DataFrame.sort_values(*by='##',axis=0,ascending=True, inplace=False, …***)**

by： 指定列名(axis=0或'index')或索引值(axis=1或'columns')

返回名次

Series.rank(*axis=0,method='average',na_option='keep',ascending=True***)**

method： {'average','min','max','first','dense'}指定排名时用于破坏平级关系的method选项

Series和DataFrame切片

Pandas的两种数据类型都可以使用numpy一样的数据格式进行索引，也可以使用loc和iloc进行更高级的索引与切片。

df.loc[]：获取具有特定标签的行(和/或列)，接受两个参数：行标和列标，当列标省略时，默认获取整行数据。两个参数都可以以字符，切片以及列表的形式传入。

df.iloc[]：用法和df.loc[]类似，最大的区别在于，loc是基于行列的标签进行检索，而iloc是基于位置进行检索。

标签索引与内置索引混用：

get_loc()是一个索引方法，意思是“获取标签在这个索引中的位置”。注意，因为用**iloc**切片不包含它的端点，必须在这个值上加上1。

数据读取

文本文件和CSV数据读取

pandas.read_csv(*filepath_or_buffer , sep= ' ' , header='infer' , names=None , index _col=None , usecols=None , dtype=None , skiprows=None , skipfooter=None , nrows=None , na_values=None , skip_blank_lines=True , parse_dates=False , comment=None , encoding=None***)**

filepath_or_buffer： 文件路径

sep： 指定分隔符。如果不指定参数，则会尝试使用逗号分隔

header： 指定行数用来作为列名，数据开始行数。如果文件中没有列名，则默认为0，否则设置为None

encoding： 指定字符集类型，通常指定为'utf-8'

Excel文件读取

pandas.read_excel(*io， sheet_name = 0， header = 0， names = None， index_col = None， usecols = None， squeeze = False,dtype = None, encoding=None ...***)**

sheet_name： 字符串用于工作表名称,整数用于零索引工作表位置,字符串列表或整数列表用于请求多个工作表，为None时获取所有工作表。

数据存写

文本文件和CSV数据写入

DataFrame.to_csv(*path_or_buf=None, sep=',', na_rep='', columns=None, header=True, index=True, index_label=None,mode='w',encoding=None***)**

sep： 接收string。代表分隔符。默认为“,”

index_labels： 接收sequence。表示索引名。默认为None

na_rep： 接收string。代表缺失值。默认为“”

columns： 接收list。代表写出的列名。

encoding： 接收特定string。代表存储文件的编码格式。

header： 接收boolean，代表是否将列名写出。默认：Ture或者None

Excel文件写入

DataFrame.to_excel(*excel_writer, sheet_name='Sheet1', na_rep='', float_format=None, columns=None,header=True, index=True, index_label=None,…***)**

pandas统计功能

Pandas基于NumPy，同样可以使用NumPy的函数对数据框进行描述性统计

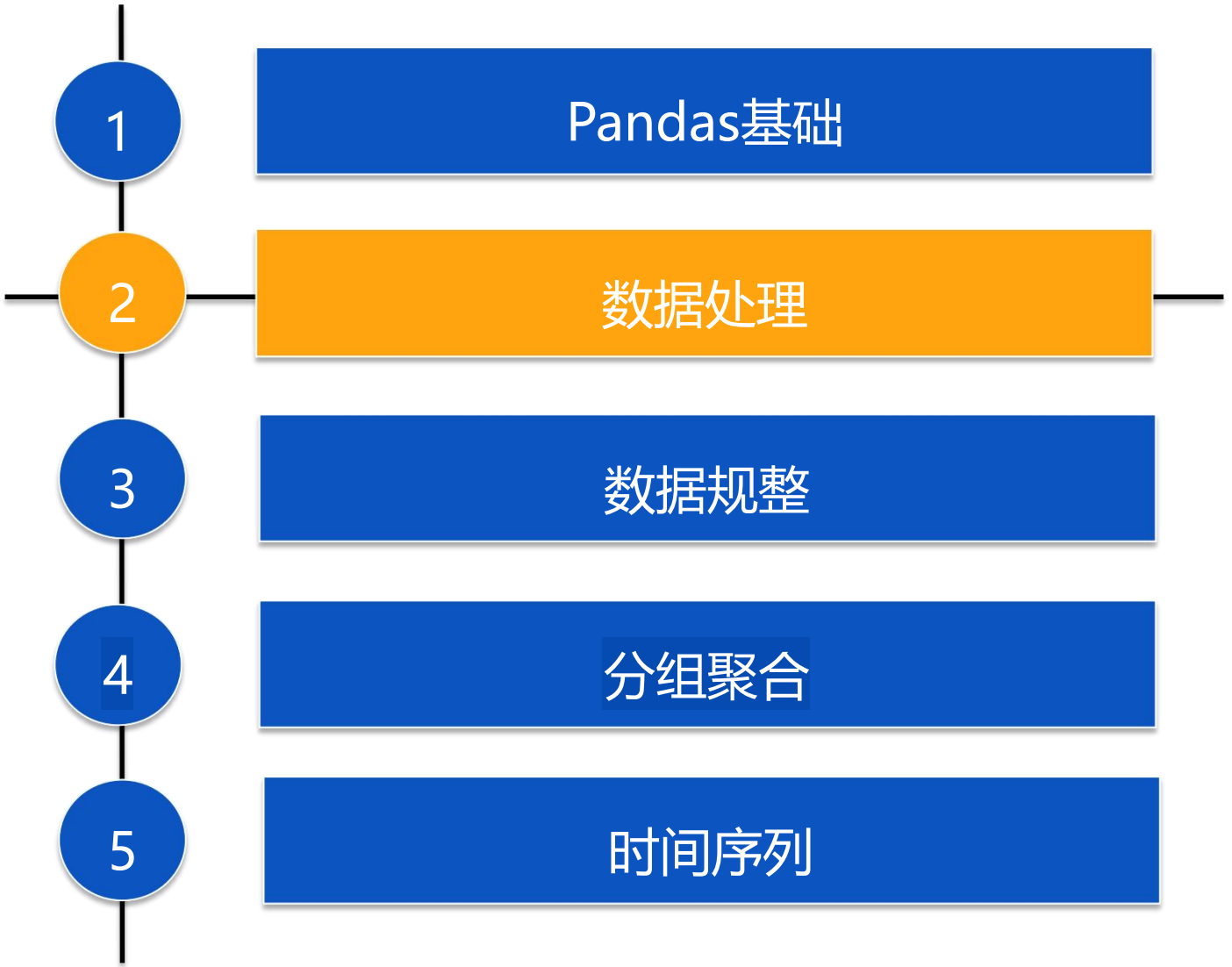
pandas还提供了更加便利的方法来计算均值，如detail['amounts'].mean()。

pandas还提供了一个方法叫作describe，能够一次性得出数据框所有数值型特征的非空值数目、均值、四分位数、标准差。

方法名称	说明	方法名称	说明
min	最小值	max	最大值
mean	均值	ptp	极差
median	中位数	std	标准差
var	方差	cov	协方差
sem	标准误差	mode	众数
skew	样本偏度	kurt	样本峰度
quantile	四分位数	count	非空值数目
describe	描述统计	mad	平均绝对离差

函数名称	说明
np.min/max	最小/大值
np.mean	均值
np.median	中位数
np.var	方差
np.ptp	极差
np.std	标准差
np.cov	协方差

目录



处理缺失值

pandas支持多种方法处理缺失值，可以直接调用函数进行处理

方法	描述
dropna	根据每个标签的值是否缺失数据来筛选标签，并根据允许缺失的数据来确定阈值
fillna	用某些值填充缺失的数据或使用插值方法
isnull	返回表明哪些值是缺失值的布尔值
notnull	isnull的反函数

DataFrame.dropna(*self, axis=0, how='any', thresh=None, subset=None, inplace=False***)**

how： 选择方式是全为空还是部分为空

DataFrame.fillna(*value=None, method=None, axis=None, inplace=False, limit=None, **kwargs***)**

value： 它是一个用于填充空值的值

method： 一种用于填充重新索引的Series中的空值的方法

处理重复值

DataFrame.drop_duplicates(*subset=['A','B'],keep='first',inplace=True***)**

subset： 列名，表示只考虑这两列，将这两列对应值相同的行进行去重。默认为subset=None表示考虑所有列

keep='first'： 保留第一次出现的重复行，是默认值。keep另外两个取值为“last”和False，分别表示保留最后一次出现的重复行和去除所有重复行

inplace=True： 表示直接在原来的DataFrame上删除重复项，而默认值False表示生成一个副本

数据离散分箱

连续数值经常需要离散化，或者分离成箱子进行分析。如果相对数据进行分组，需要借用cut函数和qcut函数。

pandas.cut(*x, bins, right=True, labels=None, retbins=False, precision=3, duplicates='raise'***)**

pandas.qcut(*x, bins, labels=None, retbins=False, precision=3, duplicates='raise'***)**

x： 被切分的类数组数据，必须是1维的（不能用DataFrame）

bins： bins是被切割后的区间，有3中形式：一个int型的标量、标量序列（数组）或pandas.IntervalIndex

right： bool型参数，默认为True，表示是否包含区间右部。比如如果bins=[1,2,3]，right=True，则区间为(1,2],(2,3]；right=False，则区间为(1,2),(2,3)

labels： 给分割后的bins打标签，比如把年龄x分割成年龄段bins后，可以给年龄段打上诸如青年、中年的标签

retbins： 是否返回bins

precision： 数据精度

duplicates： 是否允许重复区间。有两种选择：raise：不允许，drop：允许

哑变量处理

将分类变量转换成‘虚拟’或‘指标’矩阵式是另一种用于统计建模或机器学习的转换操作。pandas有一个函数get_dummies函数用于实现该功能。

pandas.get_dummies(*data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False***)**

data： 输入的数据

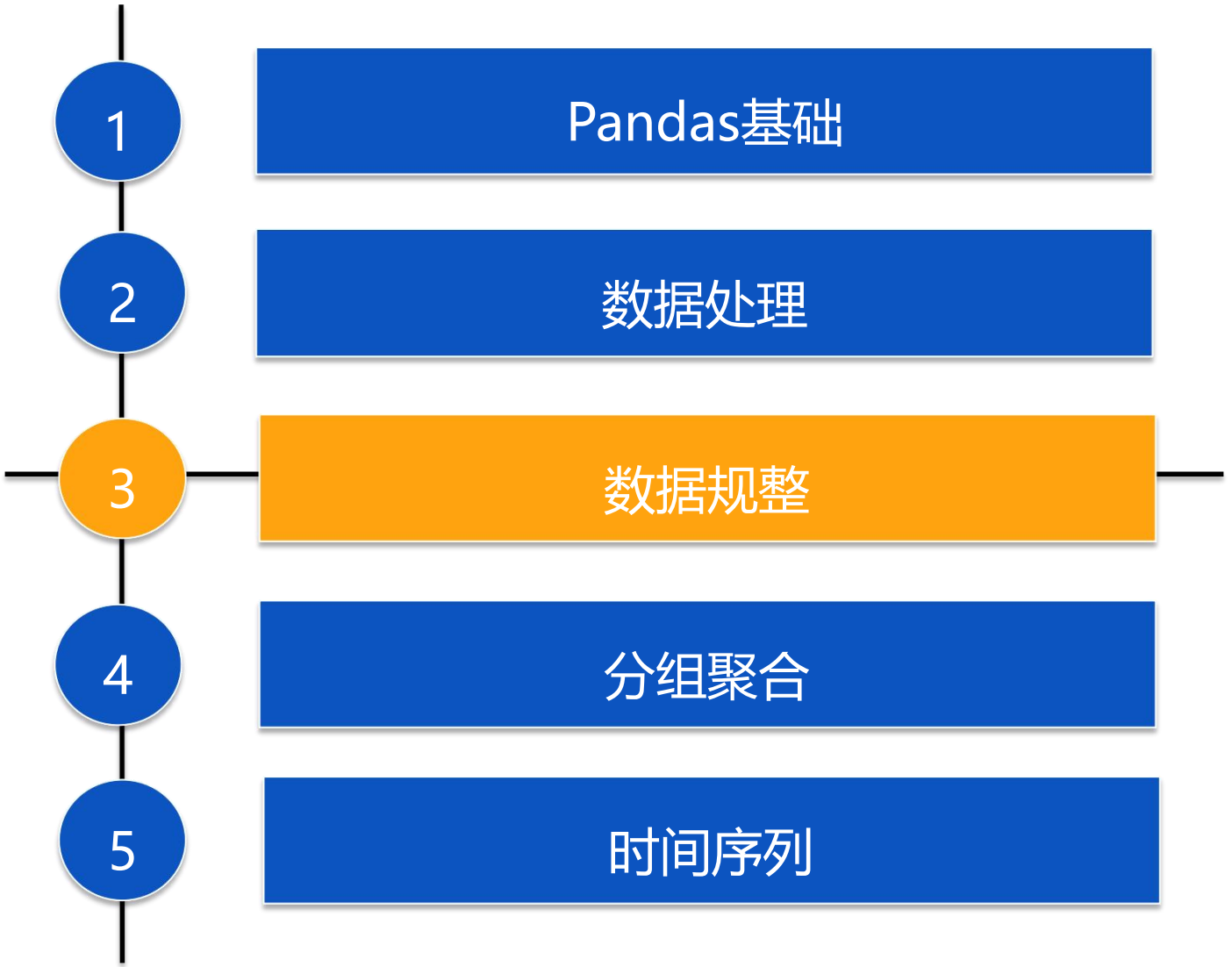
prefix： 转换后，列名的前缀

columns： 指定需要实现类别转换的列名

dummy_na： 增加一列表示空缺值，如果False就忽略空缺值

drop_first： 获得k中的k-1个类别值，去除第一个，默认False

目录



Pandas按键合并数据

pandas.concat(*objs, axis=0, join='outer', join_axes=None, ignore_index=False, keys=None, levels=None, names=None, verify_integrity=False, sort=None, copy=True***)**

objs： Series，DataFrame或Panel对象的序列或映射

axis： {0,1，...}，默认为0。沿着连接的轴

join： {‘inner’，‘outer’}，默认为“outer” 处理其他轴上的索引。outer为并集，inner为交集

Pandas值合并函数：merge

pandas.merge(*left,right,how="inner",on=None,left_on=None,right_on=None,left_index=False,right_index=False,sort=True,suffixes=("_x", "_y"),copy=True,indicator=False,validate=None,...***)**

left： 参与合并的左侧DataFrame

right： 参与合并的右侧DataFrame

how： 连接方式： ‘inner’ （默认）；还有， ‘outer’ 、 ‘left’ 、 ‘right’

on： 用于连接的列名，必须同时存在于左右两个DataFrame对象中，如果未指定，则以left和right列名的交集作为连接键

left_on： 左侧DataFrame中用作连接键的列

right_on： 右侧DataFrame中用作连接键的列

left_index： {True or False,默认False}将左侧的行索引用作其连接键

right_index： {True or False ,默认False}将右侧的行索引用作其连接键

suffixes： 字符串值元组，用于追加到重叠列名的末尾，默认为（ ‘_x’ ，‘_y’ ）

Pandas合并重叠数据

在处理数据的过程中，当一个DataFrame对象中出现了缺失数据，而对于这些缺失数据，我们希望可以使用其他DataFrame对象中的数据填充，这时可以通过combine_first()方法填充缺失数据。

DataFrame.combine_first(*other*)

other： 接收填充缺失值的DataFrame对象。

```
In [42]: left
```

Out[42]:

	A	B	key
0	NaN	NaN	K0
1	A0	B1	K1
2	A1	NaN	K2
3	A2	B3	K3

```
In [37]: right
```

Out[37]:

	A	B
1	C0	D0
0	C1	D1
2	C2	D2

```
In [43]: left.combine_first(right)
```

Out[43]:

	A	B	key
0	C1	D1	K0
1	A0	B1	K1
2	A1	D2	K2
3	A2	B3	K3

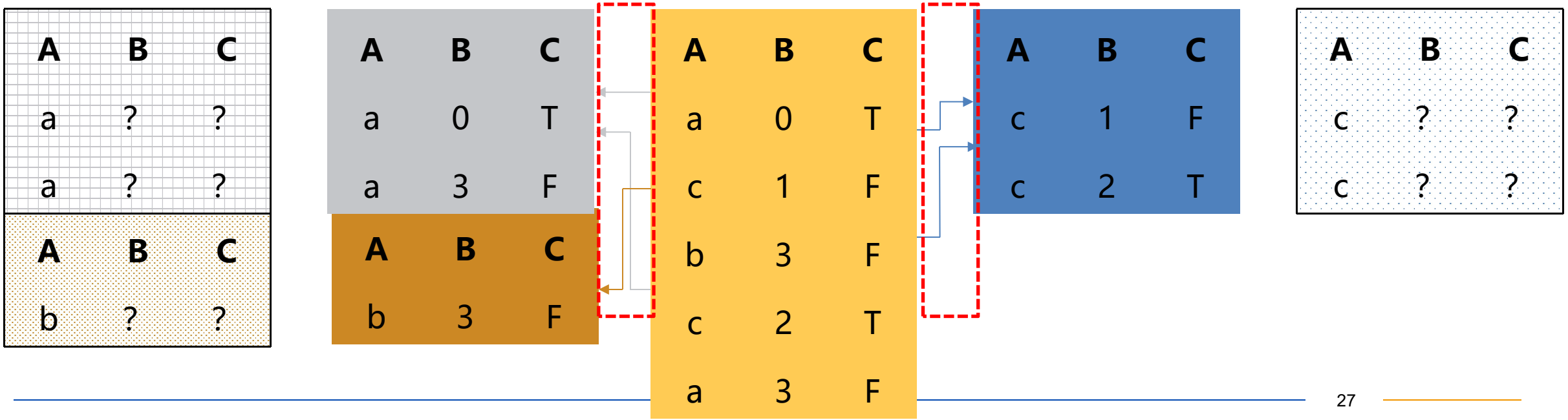
目录

groupby技术

groupby就是按xx分组, 比如, 将一个数据集按A进行分组, 效果如下

DataFrame.groupby(*by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs***)**

by： 用来指定作用于index的函数、字典、Series对象或指定列名作为分组依据



groupby常见分组示例

根据不同的键进行分组

单键分组: DataFrame.groupby(*'key'*).method()

多键分组: DataFrame.groupby(*['key1',…, 'keyn']*).method()

分组后对**指定列**的值进行计算(有多种方法，这里举一种作为示例)

DataFrame.groupby(*['key1',…'keyn']*)[*'val1',…, 'valn'*].method()

groupby支持的方法

方法	描述	方法	描述
count	分组中的非NA值	sum	非NA值得累和
mean	非NA值得均值	median	非NA值的算数中位数
std、var	无偏标准差和方差	min、max	非NA的最小值和最大值
prod	非NA值的乘积	first、last	非NA的第一个值和最后一个值
quantile(n)	计算Series或DataFrame列的样本分位数，n∈（0，1）		

groupby逐列多函数应用

如果已经存在分好组的数据，可以借助agg函数对一列数据应用多个函数，或者对不同列应用不同函数

对一列数据应用多个函数

已分组数据：**grouped=DataFrame.groupby(*'key1'*)[*'col1'*]**

对分组后数据应用多个函数：**grouped.agg(*['fun1','fun2', ···,'funn']*)**

这里直接输入函数名的字符串即可，函数可以是自定义函数。

已分组数据：**grouped=DataFrame.groupby(*'key1'*)[*'col1', 'col2'*]**

对分组后数据应用多个函数：**grouped.agg(*{ 'col1':['fun1','fun2', ···,'funn'],*
'col2':['fun1','fun2', ···,'funm']})**

apply函数

apply函数也可以对groupby的结果进行操作，函数具体的使用参数如下

DataFrame.apply(*func, axis=0, broadcast=False, raw=False, reduce=None, args=(), **kwds***)**

func： apply应用的函数

axis： 函数应用的轴向，0是纵向，1是横向

apply函数和agg函数辨析(不同点)

apply函数

1. 对整个分组下的整体数据应用，可以返回多个值
2. 一次只能应用一个方法
3. 可选方向

agg函数

1. 单独对分组的某列数据进行计算，只能返回一个结果
2. 一次能使用多个方法
3. 只能对列计算

数据透视表

透视表根据一个或者多个键聚合一张表的数据，将数据在矩形格式中排列，其中一部分分组键沿着行，另一部分沿着列。实现该功能可以借助pivot_table函数实现。

pandas.pivot_table(*data, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All'***)**

data： 要处理的数据

index/ columns： 在结果透视表行/列上进行分组的列名或者其他分组键

values： 需要聚合的列名，默认聚合所有列

aggfunc： 值的聚合方式(默认的是求平均)

margins： 总计

fill_value： 当出现nan值时，用什么填充

dropna： 是否去掉nan值

交叉表

交叉表是透视表的一个特殊情况，计算的是分组中的频率。主要使用crosstable函数实现

pandas.crosstab(*index, columns, values=None, rownames=None, colnames=None, aggfunc=None,margins=False, dropna=True, normalize=False***)**

index：在结果交叉表行上进行分组的列名或者其他分组键

columns：在结果交叉表列上进行分组的列名或者其他分组键

values： 需要聚合的列名

aggfunc： 值的聚合方式(默认的是求平均)

margins： 总计

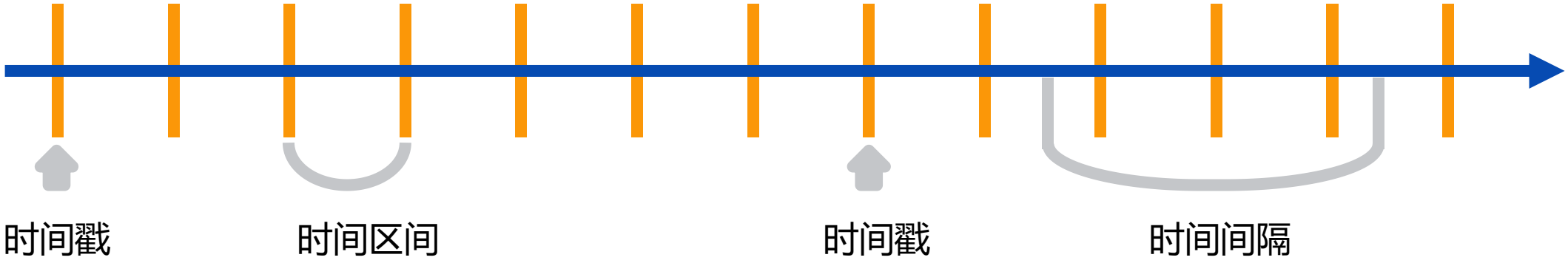
fill_value： 当出现nan值时，用什么填充

dropna： 是否去掉nan值

目录

时间数据

时间序列在很多领域都是常用的一种数据结构形式，例如金融、经济、物理等。在不用的时间点观测的数据形成了时间序列。



时间戳：具体时刻

时间区间：固定的时间间隔

时间间隔：由开始和结束时间戳表示，时间区间被认为是时间间隔的特殊情况

*Python中包含时间格式的模块有很多，这里只讲解Pandas中的用法

时间戳

许多文件中时间数据会被存储成字符串的格式，可以借助函数讲字符串转换成时间数据格式。

时间戳函数

pandas.Timestamp(*ts_input,freq=None, tz=None, unit=None, year=None, month=None, day=None, hour=None, minute=None, second=None, microsecond=None, nanosecond=None, tzinfo=None, *, fold=None***)**

ts_input： 要转换为Timestamp的值

unit： 可以为['D', 'h' ,'m', 'ms' ,'s', 'ns']

整列数据转换成时间

pandas.to_datetime(*arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True, unit=None, infer_datetime_format=False, origin='unix', cache=True***)**

dayfirst/ yearfirst： 表示传入数据的前两位数为天/年

format： 自定义输出格式，如“%Y-%m-%d”

时间索引

Pandas支持将时间序列数据设置为索引，常见的索引有DatetimeIndex和PeriodIndex两种。二者区别在日常使用的过程中相对较小，其中DatetimeIndex是用来指代一系列时间点的一种数据结构，而PeriodIndex则是用来指代一系列时间段的数据结构。

pandas.DatetimeIndex(*data=None, freq=<no_default>, tz=None, normalize=False, closed=None, ambiguous='raise', dayfirst=False, yearfirst=False, dtype=None, copy=False, name=None***)**

data： 接收array。表示DatetimeIndex的值。无默认

freq： 接收string。表示时间的间隔频率。无默认

start： 接收string。表示生成规则时间数据的起始点。无默认

periods： 表示需要生成的周期数目。无默认

end： 接收string。表示生成规则时间数据的终结点。无默认

tz： 接收timezone。表示数据的时区。默认为None

name： 接收int，string。默认为空。指定DatetimeIndex的名字

时间索引操作

将时间序列设置为索引，可以执行多种数据处理的简便操作。

功能	函数示例
取指定一段时间的数据	df[data1:data2]
取指定年、月、日的数据	df[df.index.mouth==3]
取指定时间段的数据	df[(df.index.hour > 8) & (df.index.hour <12)] df.between_time('08:00','12:00')

面对大量的时间数据时可以借用resample函数对时间序列进行采样。

DataFrame.resample(*rule*)

rule：表示目标转换的偏移字符串或对象，一般是时间参数，比如“M”，“A”，“Q”，“BM”，“BA”，“BQ”和“W”；

采样后的数据是一个resample.DatetimeIndexResampler的数据，无法直接查看

Thank you!