Python数据分析

pandas模块

pandas模块

介绍：

pandas介绍

Python Data Analysis Library

pandas是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入 了大量库和一些标准的数据模型，提供了高效地操作大型结构化数据集所需的工具。

pandas核心数据结构

数据结构是计算机存储、组织数据的方式。 通常情况下，精心选择的数据结构可以带来更高的运行或者存储效率。数据结构往往同高效的检索算法和索引技术有关。

pandas模块-Series

Series：

Series

Series可以理解为一个一维的数组，只是index名称可以自己改动。类似于定长的有序字典，有Index和 value。

pandas模块-Series

Series：

```
1  import pandas as pd
2
3  # 创建一个空的系列
4  s = pd.Series()
5  # 从ndarray创建一个系列
6  data = np.array(['a','b','c','d'])
7  s = pd.Series(data)
8  s = pd.Series(data,index=[100,101,102,103])
9  # 从字典创建一个系列
10 data = {'a' : 0., 'b' : 1., 'c' : 2.}
11 s = pd.Series(data)
12 # 从标量创建一个系列
13 s = pd.Series(5, index=[0, 1, 2, 3])
```

pandas模块-Series

Series：

访问Series中的数据：

```
1  # 使用索引检索元素
2  s = pd.Series([1,2,3,4,5],index =
   ['a','b','c','d','e'])
3  print(s[0], s[:3], s[-3:])
4  # 使用标签检索数据
5  print(s['a'], s[['a','c','d']])
```

pandas模块-Series

练习：

1、随机5个60-100的分数，A-E作为索引(学生编号)，测试Series

2、通过索引、切片、index获取元素

pandas模块-日期

日期处理：

pandas日期处理

```
1  # pandas识别的日期字符串格式
2  dates = pd.Series(['2011', '2011-02', '2011-03-01',
   '2011/04/01', '2011/05/01 01:01:01', '01 Jun 2011'])
3  # to_datetime() 转换日期数据类型
4  dates = pd.to_datetime(dates)
5  print(dates, dates.dtype, type(dates))
6  # datetime类型数据支持日期运算
7  delta = dates - pd.to_datetime('1970-01-01')
8  # 获取天数数值
9  print(delta.dt.days)
10
```

pandas模块-日期

日期处理：

Series.dt提供了很多日期相关操作，如下：

```
Series.dt.year  The year of the datetime.
Series.dt.month The month as January=1, December=12.
Series.dt.day   The days of the datetime.
Series.dt.hour  The hours of the datetime.
Series.dt.minute    The minutes of the datetime.
Series.dt.second    The seconds of the datetime.
Series.dt.microsecond   The microseconds of the
datetime.
Series.dt.week  The week ordinal of the year.
Series.dt.weekofyear    The week ordinal of the year.
Series.dt.dayofweek The day of the week with Monday=0,
```

pandas模块-日期

日期处理：

```
11 Series.dt.weekday   The day of the week with Monday=0,
   Sunday=6.
12 Series.dt.dayofyear The ordinal day of the year.
13 Series.dt.quarter   The quarter of the date.
14 Series.dt.is_month_start    Indicates whether the date
   is the first day of the month.
15 Series.dt.is_month_end  Indicates whether the date is
   the last day of the month.
16 Series.dt.is_quarter_start  Indicator for whether the
   date is the first day of a quarter.
17 Series.dt.is_quarter_end    Indicator for whether the
   date is the last day of a quarter.
```

pandas模块-日期

日期处理：

```
18 Series.dt.is_year_start Indicate whether the date is
   the first day of a year.
19 Series.dt.is_year_end   Indicate whether the date is
   the last day of the year.
20 Series.dt.is_leap_year  Boolean indicator if the date
   belongs to a leap year.
21 Series.dt.days_in_month The number of days in the
   month.
```

pandas模块- DataFrame

DataFrame：

DataFrame

DataFrame是一个类似于表格的数据类型，可以理解为一个二维数组，索引有两个维度，可更改。DataFrame具有以下特点：

- 列可以是不同的类型
- 大小可变
- 标记轴(行和列)
- 可以对行和列执行算术运算

pandas模块- DataFrame

DataFrame：

```
1  import pandas as pd
2
3  # 创建一个空的DataFrame
4  df = pd.DataFrame()
5  print(df)
6
```

pandas模块- DataFrame

DataFrame：

```
7  # 从列表创建DataFrame
8  data = [['Alex',10],['Bob',12],['Clarke',13]]
9  df = pd.DataFrame(data,columns=['Name','Age'])
10 print(df)
11
12 data = [['Alex',10],['Bob',12],['Clarke',13]]
13 df = pd.DataFrame(data,columns=
   ['Name','Age'],dtype=float)
14 print(df)
15 data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
16 df = pd.DataFrame(data)
17 print(df)
```

pandas模块- DataFrame

DataFrame：

```
18
19  # 从字典来创建DataFrame
20  data = {'Name':['Tom', 'Jack', 'Steve',
    'Ricky'],'Age':[28,34,29,42]}
21  df = pd.DataFrame(data, index=['s1','s2','s3','s4'])
22  print(df)
23  data = {'one' : pd.Series([1, 2, 3], index=['a', 'b',
    'c']),
24          'two' : pd.Series([1, 2, 3, 4], index=['a',
    'b', 'c', 'd'])}
25  df = pd.DataFrame(data)
26  print(df)
```

pandas模块-列操作

访问

python

核心数据结构操作

列访问

DataFrame的单列数据为一个Series。根据DataFrame的定义可以 知晓DataFrame是一个带有标签的二维数组，每个标签相当每一列的列名。

pandas模块-列操作

访问：

```
1  import pandas as pd
2
3  d = {'one' : pd.Series([1, 2, 3], index=['a', 'b',
   'c']),
4       'two' : pd.Series([1, 2, 3, 4], index=['a', 'b',
   'c', 'd'])}
5
6  df = pd.DataFrame(d)
7  print(df['one'])
8  print(df[['one', 'two']])
```

pandas模块-列操作

添加

列添加

python

DataFrame添加一列的方法非常简单，只需要新建一个列索引。并对该索引下的数据进行赋值操作即可。

```python
import pandas as pd

data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['s1','s2','s3','s4'])
df['score']=pd.Series([90, 80, 70, 60], index=['s1','s2','s3','s4'])
print(df)
```

pandas模块-列操作

删除

列删除

删除某列数据需要用到pandas提供的方法pop，pop方法的用法如下：

pandas模块-列操作

删除

```
1  import pandas as pd
2
3  d = {'one' : pd.Series([1, 2, 3], index=['a', 'b',
   'c']),
4       'two' : pd.Series([1, 2, 3, 4], index=['a', 'b',
   'c', 'd']),
5       'three' : pd.Series([10, 20, 30], index=['a',
   'b', 'c'])}
6  df = pd.DataFrame(d)
7  print("dataframe is:")
8  print(df)
```

pandas模块-列操作

删除

```
8  # 删除一列： one
9  del(df['one'])
10 print(df)
11
12 #调用pop方法删除一列
13 df.pop('two')
14 print(df)
```

pandas模块-行操作

行

行访问

如果只是需要访问DataFrame某几行数据的实现方式则采用数组的选取方式，使用 ":" 即可：

```python
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print(df[2:4])

```

pandas模块-行操作

loc访问

loc方法是针对DataFrame索引名称的切片方法。loc方法使用方法如下：

```
1  import pandas as pd
2
3  d = {'one' : pd.Series([1, 2, 3], index=['a', 'b',
   'c']), 'two' : pd.Series([1, 2, 3, 4], index=['a', 'b',
   'c', 'd'])}
4
5  df = pd.DataFrame(d)
6  print(df.loc['b'])
7  print(df.loc[['a', 'b']])
```

pandas模块-行操作

iloc

iloc和loc区别是iloc接收的必须是行索引和列索引的下标。iloc方法的使用方法如下：

```
1  import pandas as pd
2
3  d = {'one' : pd.Series([1, 2, 3], index=['a', 'b',
   'c']),
4       'two' : pd.Series([1, 2, 3, 4], index=['a', 'b',
   'c', 'd'])}
5
6  df = pd.DataFrame(d)
7  print(df.iloc[2])
8  print(df.iloc[[2, 3]])
```

pandas模块-行操作

添加

行添加

```
1  import pandas as pd
2
3  df = pd.DataFrame([['zs', 12], ['ls', 4]], columns =
   ['Name','Age'])
4  df2 = pd.DataFrame([['ww', 16], ['zl', 8]], columns =
   ['Name','Age'])
5
6  df = df.append(df2)
7  print(df)
```

pandas模块-行操作

删除

行删除

使用索引标签从DataFrame中删除行。 如果标签重复，则会删除多行。

```
1  import pandas as pd
2
3  df = pd.DataFrame([['zs', 12], ['ls', 4]], columns =
   ['Name','Age'])
4  df2 = pd.DataFrame([['ww', 16], ['zl', 8]], columns =
   ['Name','Age'])
5  df = df.append(df2)
6  # 删除index为0的行
7  df = df.drop(0)
8  print(df)
```

pandas模块-行操作

修改

python

修改DataFrame中的数据

更改DataFrame中的数据，原理是将这部分数据提取出来，重新赋值为新的数据。

```python
import pandas as pd

df = pd.DataFrame([['zs', 12], ['ls', 4]], columns = ['Name','Age'])
df2 = pd.DataFrame([['ww', 16], ['zl', 8]], columns = ['Name','Age'])
df = df.append(df2)
df['Name'][0] = 'Tom'
print(df)

```

pandas模块-案例

要求:

1、生成2020-3-1 到2020-3-7日期作为序号（index）

2、随机生成 (7,4) 0-1之间的小数数据

3、ABCD 作为列名,

4、显示数据

5、查询第1、2列的数据

6、删除最后一列数据

7、查询出2-5行的数据

8、删除最后一行数据

pandas模块-DataFrame属性

常用属性

编号	属性或方法	描述
1	axes	返回 行/列 标签（index）列表。
2	dtype	返回对象的数据类型(dtype)。
3	empty	如果系列为空，则返回True。
4	ndim	返回底层数据的维数，默认定义：1。
5	size	返回基础数据中的元素数。
6	values	将系列作为ndarray返回。
7	head()	返回前n行。
8	tail()	返回最后n行。

pandas模块-DataFrame属性

常用属性

```
1  import pandas as pd
2
3  data = {'Name':['Tom', 'Jack', 'Steve',
   'Ricky'],'Age':[28,34,29,42]}
4  df = pd.DataFrame(data, index=['s1','s2','s3','s4'])
5  df['score']=pd.Series([90, 80, 70, 60], index=
   ['s1','s2','s3','s4'])
6  print(df)
7  print(df.axes)
8  print(df['Age'].dtype)
9  print(df.empty)
10 print(df.ndim)
11 print(df.size)
12 print(df.values)
13 print(df.head(3)) # df的前三行
14 print(df.tail(3)) # df的后三行
15
```

pandas模块

核心

pandas核心

pandas描述性统计

数值型数据的描述性统计主要包括了计算数值型数据的完整情况、最小值、均值、中位 数、最大值、四分位数、极差、标准差、方差、协方差等。在NumPy库中一些常用的统计学函数也可用于对数据框进行描述性统计。

pandas模块

numpy：

```
1  np.min   最小值
2  np.max   最大值
3  np.mean  均值
4  np.ptp   极差
5  np.median    中位数
6  np.std   标准差
7  np.var   方差
8  np.cov   协方差
```

pandas模块-pandas函数

pandas功能函数

python

pandas提供了统计相关函数：

1	count()	**非空观测数量**
2	sum()	所有值之和
3	mean()	所有值的平均值
4	median()	所有值的中位数
5	std()	值的标准偏差
6	min()	所有值中的最小值

pandas模块-pandas函数

panc

```
# 创建DF
d =
{'Name':pd.Series(['Tom','James','Ricky','Vin','Steve'
,'Minsu','Jack',
   'Lee','David','Gasper','Betina','Andres']),

'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46])
,

'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3
.78,2.98,4.80,4.10,3.65])}
```

pandas模块-排序

排序

pandas排序

*Pandas*有两种排序方式，它们分别是按标签与按实际值排序。

```
1 import pandas as pd
2 import numpy as np
3
4 unsorted_df=pd.DataFrame(np.random.randn(10,2),
5                          index=
  [1,4,6,2,3,5,9,8,0,7],columns=['col2','col1'])
6 print(unsorted_df)
```

pandas模块-排序

行标签

按行标签排序

使用`sort_index()`方法，通过传递`axis`参数和排序顺序，可以对`DataFrame`进行排序。 默认情况下，按照升序对行标签进行排序。

```
1  import pandas as pd
2  import numpy as np
3
4  # 按照行标进行排序
5  sorted_df=unsorted_df.sort_index()
6  print (sorted_df)
7  # 控制排序顺序
8  sorted_df = unsorted_df.sort_index(ascending=False)
9  print (sorted_df)
```

pandas模块-排序

行标签

按列标签排序

```
1  unsorted_df = pd.DataFrame(d)
2  # 按照列标签进行排序
3  sorted_df=unsorted_df.sort_index(axis=1)
4  print (sorted_df)
```

pandas模块-排序

按某列值排序

像索引排序一样，`sort_values()`是按值排序的方法。它接受一个`by`参数，排序值的`DataFrame`的列名称。

值排序

```
# 按照年龄进行排序
sorted_df = unsorted_df.sort_values(by='Age')
print (sorted_df)
# 先按Age进行升序排序，然后按Rating降序排序
sorted_df = unsorted_df.sort_values(by=['Age',
'Rating'], ascending=[True, False])
print (sorted_df)
```

pandas模块-分组

分组

pandas分组

在许多情况下，我们将数据分成多个集合，并在每个子集上应用一些函数。在应用函数中，可以执行以下操作：

- *聚合* - 计算汇总统计
- *转换* - 执行一些特定于组的操作
- *过滤* - 在某些情况下丢弃数据

pandas模块-分组

分组

```
1  import pandas as pd
2
3  ipl_data = {'Team': ['Riders', 'Riders', 'Devils',
   'Devils', 'Kings',
4           'kings', 'Kings', 'Kings', 'Riders',
   'Royals', 'Royals', 'Riders'],
5           'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
6           'Year':
   [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,201
   5,2017],
7           'Points':
   [876,789,863,673,741,812,756,788,694,701,804,690]}
8  df = pd.DataFrame(ipl_data)
9  print(df)
```

pandas模块-分组

分

将数据拆分成组

```
1  # 按照年份Year字段分组
2  print (df.groupby('Year'))
3  # 查看分组结果
4  print (df.groupby('Year').groups)
5
```

pandas模块-分组

分组

python

迭代遍历分组

groupby返回可迭代对象，可以使用for循环遍历：

```python
print (df.groupby('Year').groups)
# 遍历每个分组
for year,group in grouped:
    print (year)
    print (group)

```

pandas模块-分组

分组

获得一个分组细节

```
1 grouped = df.groupby('Year')
2 print (grouped.get_group(2014))
```

pandas模块-分组

分组

分组聚合

聚合函数为每个组返回聚合值。当创建了分组(*group by*)对象，就可以对每个分组数据执行求和、求标准差等操作。

```
# 聚合每一年的平均的分
grouped = df.groupby('Year')
print (grouped['Points'].agg(np.mean))
# 聚合每一年的分数之和、平均分、标准差
grouped = df.groupby('Year')
agg = grouped['Points'].agg([np.sum, np.mean, np.std])
print (agg)
```

pandas模块-数据表关联

关联

pandas数据表关联操作

Pandas具有功能全面的高性能内存中连接操作，与SQL等关系数据库非常相似。

Pandas提供了一个单独的`merge()`函数，作为DataFrame对象之间所有标准数据库连接操作的入口。

pandas模块-数据表关联

DataFrame合并

```
1  import pandas as pd
2  left = pd.DataFrame({
3          'student_id':[1,2,3,4,5,6,7,8,9,10],
4          'student_name': ['Alex', 'Amy', 'Allen',
   'Alice', 'Ayoung', 'Billy', 'Brian', 'Bran', 'Bryce',
   'Betty'],
5          'class_id':[1,1,1,2,2,2,3,3,3,4]})
6  right = pd.DataFrame(
7          {'class_id':[1,2,3,5],
8          'class_name': ['ClassA', 'ClassB', 'ClassC',
   'ClassE']})
9  print (left)
10 print("=========================================")
11 print (right)
12 print("=========================================")
```

pandas模块-数据表关联

DataFrame关联

使用“how”参数合并DataFrame：

python

```python
# 合并两个DataFrame（左连接）
print("========================================")
rs = pd.merge(left, right, how='left')
print(rs)
```

pandas模块-数据表关联

DataFrame关联

合并方法	SQL等效	描述
left	LEFT OUTER JOIN	使用左侧对象的键
right	RIGHT OUTER JOIN	使用右侧对象的键
outer	FULL OUTER JOIN	使用键的联合
inner	INNER JOIN	使用键的交集

pandas模块-透视表和交叉表

准备数据

```
import pandas as pd
left = pd.DataFrame({
        'student_id':[1,2,3,4,5,6,7,8,9,10],
        'student_name': ['Alex', 'Amy', 'Allen',
'Alice', 'Ayoung', 'Billy', 'Brian', 'Bran', 'Bryce',
'Betty'],
        'gender':
['M','M','M','F','M','F','M','F','M',''']
        'class_id':[1,1,1,2,2,2,3,3,3,4]})
right = pd.DataFrame(
        {'class_id':[1,2,3,5],
        'class_name': ['ClassA', 'ClassB', 'ClassC',
'ClassE']})
# 合并两个DataFrame
data = pd.merge(left,right)
print(data)
```

pandas模块-透视表和交叉表

透视表

透视表(pivot table)是各种电子表格程序和其他数据分析软件中一种常见的数据汇总工具。**它根据一个或多个键对数据进行分组聚合，并根据每个分组进行数据汇总。**

pandas模块-透视表和交叉表

透视表

```
# 以class_id与gender做分组汇总数据，默认聚合统计所有列
print(data.pivot_table(index=['class_id', 'gender']))

# 以class_id与gender做分组汇总数据，聚合统计score列
print(data.pivot_table(index=['class_id', 'gender'],
values=['score']))

# 以class_id与gender做分组汇总数据，聚合统计score列，针对age的每个值列级分组统计
print(data.pivot_table(index=['class_id', 'gender'],
values=['score'],
                       columns=['age']))

```

pandas模块-透视表和交叉表

透视表

```
14  # 以class_id与gender做分组汇总数据，聚合统计score列，针对age
    的每个值列级分组统计，添加行、列小计
15  print(data.pivot_table(index=['class_id', 'gender'],
    values=['score'],
16                         columns=['age'], margins=True,
    aggfunc='max'))
17
```

附加

pandas官网：

pandas:https://pandas.pydata.org/

附加

安装Jupyter notebook

```
pip3 install jupyter

# 启动命令： 在某个目录下
jupyter notebook
```

数据读取与存储

数据读取与存储

CSV

数据读取与存储

读取与存储csv：

```
# filepath 文件路径。该字符串可以是一个URL。有效的URL方案包括http，ftp和file
# sep 分隔符。read_csv默认为“,”，read_table默认为制表符“[Tab]”。
# header 接收int或sequence。表示将某行数据作为列名。默认为infer，表示自动识别。
# names 接收array。表示列名。
# index_col 表示索引列的位置，取值为sequence则代表多重索引。
# dtype 代表写入的数据类型（列名为key，数据格式为values）。
# engine 接收c或者python。代表数据解析引擎。默认为c。
# nrows 接收int。表示读取前n行。
```

数据读取与存储

读

```
9
10 pd.read_table(
11     filepath_or_buffer, sep='\t', header='infer',
   names=None,
12     index_col=None, dtype=None, engine=None,
   nrows=None)
13 pd.read_csv(
14     filepath_or_buffer, sep=',', header='infer',
   names=None,
15     index_col=None, dtype=None, engine=None,
   nrows=None)
```

数据读取与存储

写入csv

```
DataFrame.to_csv(excel_writer=None, sheetname=None,
header=True, index=True, index_label=None, mode='w',
encoding=None)
```

数据读取与存储

读取、存储execl

读取与存储excel：

```python
# io 表示文件路径。
# sheetname 代表excel表内数据的分表位置。默认为0。
# header 接收int或sequence。表示将某行数据作为列名。默认为infer，表示自动识别。
# names 表示索引列的位置，取值为sequence则代表多重索引。
# index_col 表示索引列的位置，取值为sequence则代表多重索引。
# dtype 接收dict。数据类型。
pandas.read_excel(io, sheetname=0, header=0, index_col=None, names=None, dtype=None)
```

python

```python
DataFrame.to_excel(excel_writer=None, sheetname=None, header=True, index=True, index_label=None, mode='w', encoding=None)
```

数据读取与存储

读取、存储json

读取与存储JSON：

```
1  # 通过json模块转换为字典，再转换为DataFrame
2  pd.read_json('../ratings.json')
```