第十一章 Python在数据分析中的应用

11.1 NumPy简介

11.2 Pandas简介

11.3 数据分析应用

数据分析的四大任务

- **数据准备**（读写各种各样的文件格式和数据库）
- **数据处理**（对数据进行清洗、修整、整合等处理以便进行分析）
- **数据分析**（根据分析目的对数据集做合适的数学和统计运算）
- **数据可视化**（展示数据分析成果）

NumPy简介

- Numpy（Numerical Python的简称）是Python科学计算的基础包。它提供了以下功能（不限于此）：
  - 快速高效的多维数组对象ndarray
  - 用于对数组执行元素级计算以及直接对数组执行数学运算的函数
  - 用于读写硬盘上基于数组的数据集的工具
  - 线性代数运算、傅里叶变换，以及随机数生成
  - 用于将C|C++|Fortran代码集成到Python的工具

NumPy简介

- Numpy的安装
  - 下载符合自己Python版本的安装包https://www.numpy.org/


numpy-1.10.2-win32-superpack-pyth... 2015/12/18 13:46 应用程序

  - 双击进行安装

第十一章 Python在数据分析中的应用

11.1 NumPy简介

11.2 Pandas简介

11.3 数据分析应用

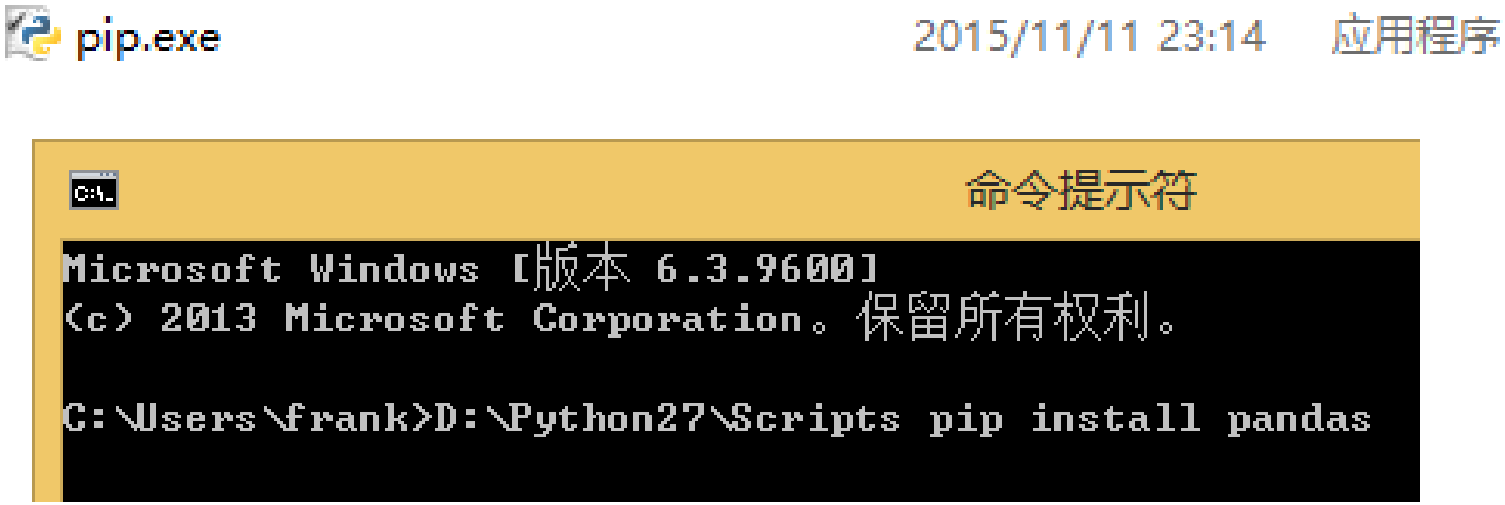
pandas简介

- pandas提供了使我们能够快速便捷地处理结构化数据的大量数据结构和函数，其兼具NumPy高性能的数组计算功能以及电子表格和关系型数据库（如SQL）灵活的数据处理功能。

- 它提供了复杂精细的索引功能，以便更为便捷地完成重塑、切片和切块、聚合以及选取数据子集等操作。

pandas安装

- 打开Python所在的文件夹下的Scripts文件夹可以看到可执行程序pip.exe
- 使用命令提示符输入pip.exe路径，然后输入pip install pandas，回车进行安装



pandas的数据结构—Series

- Series是一种类似于一维数组的对象，它是由一组**数据**（各种NumPy数据类型）以及一组与之相关的数据便签（即**索引**）组成。
- 仅由一组数据即可产生最简单的Series:

```
>>> from pandas import Series
>>> obj=Series([4,7,7,-3])
>>> obj
```

- Series的字符串表现形式为：索引在左边，值在右边。

pandas的数据结构—Series

- 如果数据被存放在一个Python字典中，可以直接通过这个字典来创建Series：

```
>>> sdata = {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah': 5000}
>>> obj3 = Series(sdata)
>>> obj3
```

- Series的索引可以通过赋值的方式就地修改

```
>>> states = ['California', 'Ohio', 'Oregon', 'Texas']
>>> obj4 = Series(sdata, index=states)
>>> obj4
```

pandas的数据结构—Series

- Series对象本身及其索引都有一个name属性，该属性跟pandas其他的关键功能关系非常密切：

```
>>> obj4.name = 'population'

>>> obj4.index.name = 'state'

>>> obj4
```

pandas的数据结构—DataFrame

- DataFrame是一个**表格型**的数据结构，它含有一组有序的列，每列可以是不同的值类型。
- DataFrame既有**行索引**也有**列索引**，它可以被看做由Series组成的字典。构建DataFrame的办法有很多，最常用的一种是直接传入一个由等长列表或NumPy数组组成的字典：

```
>>> data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
                'year': [2000, 2001, 2002, 2001, 2002],
                'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
>>> frame = DataFrame(data)
>>> frame
```

pandas的数据结构—DataFrame

- 如果指定了列序列，则DataFrame的列就会按照指定顺序进行排列：

```
>>> frame2=DataFrame(data, columns=['year', 'state', 'pop'],
                     index=['one','two','three','four','five'])
```

- 通过类似字典标记的方式或属性的方式，可以将DataFrame的列获取为一个Series：

```
>>> frame['state']
>>> frame2['state']
```

pandas的数据结构—DataFrame

- 行也可以通过位置或名称的方式进行获取，比如用索引字段ix：

```
>>> frame2.ix['two']
```

其它有关数据分析的Python库

- SciPy：是一组专门解决科学计算中各种标准问题域的包的集合。
- IPython：是Python科学计算标准工具集的组成部分，它将其他所有的东西联系到了一起。为交互式和探索式计算提供了一个强健而高效的环境。
- Matplotlib：是最流行的用于绘制数据图表的Python库。它最初由John D. Hunter创建，目前由一个庞大的开发人员团队维护。提供了一种非常好用的交互式数据绘图环境。绘制的图表也是交互式的。

第十一章 Python在数据分析中的应用

11.1 NumPy简介

11.2 Pandas简介

11.3 数据分析应用

MovieLens 1M数据集

- GroupLens Research采集了一组从20世纪90年代末到21世纪初由MovieLens用户提供的电影评分数据。这些数据中包括电影评分、电影元数据（风格类型和年代）以及关于用户的人口统计学数据（年龄、邮编、性别和职业等）

- MovieLens 1M数据集含有来自6000名用户对4000部电影的100万条评分数据。它分为3个表：评分、用户信息和电影信息。

MovieLens 1M数据集

- 将该数据从zip文件中解压出来之后，可以通过pandas.read_csv将各个表分别读到一个pandas DataFrame对象中：

```
>>> import pandas as pd
>>> import os        #引入操作系统有关操作
>>> encoding='latin1'
>>> upath=os.path.expanduser('movielens/users.dat')
#os.path.expanduser(path) 使用用户的主目录替换'-user'格式的路径名称。
>>> rpath=os.path.expanduser('movielens/ratings.dat')
>>> mpath=os.path.expanduser('movielens/movies.dat')
```

MovieLens 1M数据集

```
>>> unames=['user_id','gender','age','occupation','zip']
>>> rnames=['user_id','movie_id','rating','timestamp']
>>> mnames=['movie_id','title','genres']
>>> users = pd.read_csv(upath, sep='::', header=None, names=unames, encoding=encoding,engine='python')
  #读取upath这个路径的csv格式的文件,各个列之间使用::分隔,header=None是#指csv文件中不包含标题行，names属性使用unames，编码使用encoding
>>> ratings = pd.read_csv(rpath, sep='::', header=None, names=rnames, encoding=encoding,engine='python')
  >>> movies = pd.read_csv(mpath, sep='::', header=None, names=mnames, encoding=encoding,engine='python')
```

MovieLens 1M数据集

- 利用Python的切片语法，通过查看每个DataFrame的前几行即可验证数据加载工作是否一切顺利

```
>>> users[:5]

>>> ratings[:5]

>>> movies[:5]
```

数据分析应用

- 假设我们想要根据性别和年龄计算某部电影的平均得分，如果将所有数据都合并到一个表中的话问题就简单多了，我们先用pandas的merge函数（可根据一个或多个键将不同 DataFrame 中的行连接起来）将ratings跟users合并到一起，然后再将movies也合并进去：

```
>>> data = pd.merge(pd.merge(ratings, users), movies)

>>> data
```

补充：merge函数

left

	A	B	key1	key2
0	A0	B0	K0	K0
1	A1	B1	K0	K1
2	A2	B2	K1	K0
3	A3	B3	K2	K1

right

	C	D	key1	key2
0	C0	D0	K0	K0
1	C1	D1	K1	K0
2	C2	D2	K1	K0
3	C3	D3	K2	K0

Result

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A2	B2	K1	K0	C1	D1
2	A2	B2	K1	K0	C2	D2

merge函数可以将不同 DataFrame 中的行连接起来。pandas会根据列名的重叠情况推断出哪些列是连接键。

数据分析应用

- 为了按性别计算每部电影的平均得分，我们可以使用pivot_table方法：

```
>>> mean_ratings = data.pivot_table('rating', index='title',
                                    columns='gender', aggfunc='mean')
```

user_id	movie_id	rating	timestamp	gender	age	occupation	zip	title	genres
1	1193	5	978300760	F	1	10	48067	One Flew	Drama

```
>>> mean_ratings[:5]
```

gender	F	M
title		
$1,000,000 Duck (1971)	3.375000	2.761905
'Night Mother (1986)	3.388889	3.352941
'Til There Was You (1997)	2.675676	2.733333
'burbs, The (1989)	2.793478	2.962085

补充：pivot_table方法

Name	Rep	Manager	Product
Trantow-Barrows	Craig Booker	Debra Henley	CPU
Trantow-Barrows	Craig Booker	Debra Henley	Software
Trantow-Barrows	Craig Booker	Debra Henley	Maintenance
Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU
Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU

最简单的透视表必须有一个数据帧和一个索引。此外，你也可以有多个索引。实际上，大多数的pivot_table参数可以通过列表获取多个值。

```
1 pd.pivot_table(df,index=["Name"])
```

	Account	Price	Quantity
Name			
Barton LLC	740150	35000	1.000000
Fritsch, Russel and Anderson	737550	35000	1.000000
Herman LLC	141962	65000	2.000000

补充：pivot_table方法

参数名	说明
values	待聚合的列的名称。默认聚合所有数值列
rows	用于分组的列名或其他分组键，出现在结果透视表的行
cols	用于分组的列名或其他分组键，出现在结果透视表的列
aggfunc	聚合函数或函数列表，默认为“mean”

数据分析应用

- 过滤掉评分数据不够250条的电影。

```
>>> ratings_by_title = data.groupby('title').size()
#使用 GroupBy 对象（不论是 DataFrameGroupBy 还是 SeriesGroupBy）的.size() 方法查看分组大小
 >>> ratings_by_title[:5]
>>> active_titles = ratings_by_title.index[ratings_by_title >= 250]
>>> active_titles[:10]
>>> mean_ratings = mean_ratings.ix[active_titles]
>>> mean_ratings
>>> data = pd.merge(pd.merge(ratings, users), movies)
>>> data
```

数据分析应用

- 为了了解女性观众最喜欢的电影，我们可以对F列降序排列：

```
>>> mean_ratings = mean_ratings.rename(index={'Seven Samurai
(The Magnificent Seven) (Shichinin no samurai) (1954)': 'Seven
Samurai
(Shichinin no samurai) (1954)'})
>>> top_female_ratings = mean_ratings.sort_values(by='F',
ascending=False)
#对F列降序排列
>>> top_female_ratings[:10]
```

数据分析应用

- 假设我们想要找出男性和女性观众分歧最大的电影。一个办法是给mean_ratings加上一个用于存放平均得分之差的列，并对其进行排序：

```
>>> mean_ratings['diff'] = mean_ratings['M'] - mean_ratings['F']
#按“diff”排序即可得到分歧最大且女性观众更喜欢的电影：
>>> sorted_by_diff = mean_ratings. sort_values(by='diff')
>>> sorted_by_diff[:15]
```

- 对排序结果反序并取出前15行，得到的则是男性观众更喜欢的电影：

```
>>> sorted_by_diff[::-1][:15]
```

数据分析应用

- 如果只是想要找出分歧最大的电影（不考虑性别因素），则可以计算得分数据的方差或标准差：

```
>>> rating_std_by_title = data.groupby('title')['rating'].std()
#根据active_titles进行过滤
>>> rating_std_by_title = rating_std_by_title.ix[active_titles]
#根据值对Series进行降序排列
>>> rating_std_by_title.order(ascending=False)[:10]
```

小结

- **用于数据分析的Python库简介**：介绍了NumPy、pandas两个重要的数据分析Python库
- **pandas的重要数据结构**：介绍了pandas的两个重要数据结构Series和DataFrame。
- **读取csv文件：**读取upath这个路径的csv格式的文件,各个列之间使用::分隔,header=None是指csv文件中不包含标题行，names属性使用unames，编码使用encoding

小结

- **连接两个DataFrame**：merge函数可以将不同DataFrame 中的行连接起来。pandas会根据列名的重叠情况推断出哪些列是连接键。
- **透视表**：pivot_table方法
- **查看分组大小**：使用 GroupBy 对象（不论是 DataFrameGroupBy 还是 SeriesGroupBy）的.size() 方法查看分组大小

本章的新函数

.pivot_table('rating', index='title',columns='gender', aggfunc='mean')

建立一个透视表：将原表中的"rating"对象以列'gender',行'title'为分组标准，并对其进行求均值操作。

习题

- 将MovieLens 1M数据集从zip文件中解压出来之后，可以通过pandas.read_table将各个表分别读到一个pandas DataFrame对象中，对数据进行如下分析操作：

  (1) 先用pandas的merge函数将ratings跟users合并到一起，然后再将movies也合并进去。

  (2) 使用pivot_table方法产生另一个DataFrame，其内容为电影平均得分，行标为电影名称，列标为性别。

  (3) 过滤掉评分数据不够500条的电影。

  (4) 了解男性观众最不喜欢的电影（对M列升序排列）

  (5) 得到分歧最大且使用两种方法得到男性观众更喜欢的电影

  (6) 通过计算得分数据的方差（var），找出分歧最大的电影（不考虑性别因素）