

昵称: [Treant](#)
园龄: 5年4个月
粉丝: 197
关注: 78
[+加关注](#)

搜索

谷歌搜索

我的标签

[LeetCode](#)(5)
[Kylin](#)(4)
[概率图模型](#)(4)
[集成学习](#)(3)
[Hive](#)(3)
[Pig](#)(3)
[Spark](#)(3)
[Kaggle](#)(2)
[Elasticsearch](#)(2)
[深度学习](#)(2)
[更多](#)

随笔分类(94)

[Java](#)(5)
[JavaScript](#)(1)
[Linux](#)(1)
[Python](#)(6)
[Scala](#)(2)
[大数据](#)(7)
[机器学习](#)(14)
[计算机视觉](#)(1)
[数据仓库](#)(8)
[数据结构](#)(4)
[数据库](#)(3)
[数据挖掘](#)(3)
[搜索引擎](#)(1)
[算法](#)(15)
[推荐系统](#)
[信息论与编码](#)(4)
[一点微小的感想](#)(3)
[自然语言处理](#)(15)
[总想搞个大轮子](#)(1)

随笔档案(95)

[2017年8月](#) (2)
[2017年7月](#) (1)
[2017年6月](#) (3)
[2017年5月](#) (2)
[2017年4月](#) (3)
[2017年3月](#) (4)
[2017年2月](#) (4)
[2017年1月](#) (7)
[2016年12月](#) (8)
[2016年11月](#) (5)
[2016年10月](#) (4)
[2016年9月](#) (5)
[2016年8月](#) (2)
[2016年7月](#) (4)
[2016年6月](#) (4)
[2016年5月](#) (3)
[2016年4月](#) (5)
[2016年3月](#) (5)
[2016年1月](#) (7)
[2015年12月](#) (5)
[2015年11月](#) (4)
[2015年3月](#) (1)
[2014年9月](#) (4)
[2014年8月](#) (3)

积分与排名

积分 - 146198
排名 - 1609

【Python实战】Pandas : 让你像写SQL一样做数据分析 (一)

1. 引言

Pandas是一个开源的Python数据分析库。Pandas把结构化数据分为了三类：

- Series，1维序列，可视作为没有column名的、只有一个column的DataFrame；
- DataFrame，同Spark SQL中的DataFrame一样，其概念来自于R语言，为多column并schema化的2维结构化数据，可视作为Series的容器（container）；
- Panel，为3维的结构化数据，可视作为DataFrame的容器；

DataFrame较为常见，因此本文主要讨论内容将为DataFrame。DataFrame的生成可通过读取纯文本、Json等数据来生成，亦可以通过Python对象来生成：

```
import pandas as pd
import numpy as np

df = pd.DataFrame({'total_bill': [16.99, 10.34, 23.68, 23.68, 24.59],
                   'tip': [1.01, 1.66, 3.50, 3.31, 3.61],
                   'sex': ['Female', 'Male', 'Male', 'Male', 'Female']})
```

对于DataFrame，我们可以看到其固有属性：

```
# data type of columns
df.dtypes
# indexes
df.index
# return pandas.Index
df.columns
# each row, return array[array]
df.values
# a tuple representing the dimensionality of df
df.shape
```

- .index，为行索引
- .columns，为列名称（label）
- .dtype，为列数据类型

2. SQL操作

[官方Doc](#)给出了部分SQL的Pandas实现，在此基础上本文给出了一些扩充说明。以下内容基于Python 2.7 + Pandas 0.18.1的版本。

select

Coder 无他, 但手熟尔

1. 【Python实战】Pandas: 让你像写SQL一样做数据分析 (一) (20327)
2. Java实时读取日志文件(19504)
3. 【图论】求无向连通图的割点(16490)
4. Apache Kylin 部署之不完全指南(13377)
5. 【动态规划】最长公共子序列与最长公共子串(13078)
6. Kylin的cube模型(9479)
7. 【十大经典数据挖掘算法】CART(9001)
8. 【十大经典数据挖掘算法】C4.5(7610)
9. 连续子数组最大和(7393)
10. 【图论】有向无环图的拓扑排序(6992)

评论排行榜

1. 中文分词工具thulac4j发布(10)
2. 最长回文子串(9)
3. Java中的逆变与协变(8)
4. Node.js大众点评爬虫(8)
5. 【中文分词】二阶隐马尔可夫模型2-HMM(6)
6. 一点做用户画像的人生经验: ID强打通(5)
7. 【Kylin实战】邮件报表生成(5)
8. 连续子数组最大和(5)
9. 【图论】求无向连通图的割点(4)
10. 【JDK源码分析】String的存储区与不可变性(4)

推荐排行榜

1. 【图论】求无向连通图的割点(9)
2. Java中的逆变与协变(5)
3. Node.js大众点评爬虫(4)
4. 【Python实战】Scrapy豌豆荚应用市场爬虫(4)
5. 工作流引擎Oozie (二): coordinator(4)
6. 【动态规划】最长公共子序列与最长公共子串(4)
7. 【十大经典数据挖掘算法】kNN(4)
8. 我的博文目录整理(4)
9. 【数据压缩】LZ78算法原理及实现(3)
10. 开源中文分词工具探析 (三): Ansj(3)

名称选取, 还可以根据列所在的position选取。相关函数如下:

- loc, 基于列label, 可选取特定行 (根据行index);
- iloc, 基于行/列的position;

```
print df.loc[1:3, ['total_bill', 'tip']]
print df.loc[1:3, 'tip': 'total_bill']
print df.iloc[1:3, [1, 2]]
print df.iloc[1:3, 1: 3]
```

- at, 根据指定行index及列label, 快速定位DataFrame的元素;
- iat, 与at类似, 不同的是根据position来定位的;

```
print df.at[3, 'tip']
print df.iat[3, 1]
```

- ix, 为loc与iloc的混合体, 既支持label也支持position;

```
print df.ix[1:3, [1, 2]]
print df.ix[1:3, ['total_bill', 'tip']]
```

此外, 有更为简洁的行/列选取方式:

```
print df[1: 3]
print df[['total_bill', 'tip']]
# print df[1:2, ['total_bill', 'tip']] # TypeError: unhashable type
```

where

Pandas实现where filter, 较为常用的办法为

`df[df[column] boolean expr]`, 比如:

```
print df[df['sex'] == 'Female']
print df[df['total_bill'] > 20]

# or
print df.query('total_bill > 20')
```

在where子句中常常会搭配and, or, in, not关键词, Pandas中也有对应的实现:

```
# and
print df[(df['sex'] == 'Female') & (df['total_bill'] > 20)]
# or
print df[(df['sex'] == 'Female') | (df['total_bill'] > 20)]
# in
print df[df['total_bill'].isin([21.01, 23.68, 24.59])]
# not
print df[~(df['sex'] == 'Male')]
print df[~df['total_bill'].isin([21.01, 23.68, 24.59])]
# string function
print df = df[~df['app'].isin(sys_app) & (~df.app.str.contains('^微信\d+$'))]
```

对where条件筛选后只有一行的dataframe取其中某一列的值, 其两种实现方式如下:

```
total = df.loc[df['tip'] == 1.66, 'total_bill'].values[0]
total = df.get_value(df.loc[df['tip'] == 1.66].index.values[0], 'total_bill')
```

distinct

```
df.drop_duplicates(subset=['sex'], keep='first', inplace=True)
```

包含参数：

- subset, 为选定的列做distinct, 默认为所有列；
- keep, 值选项{'first', 'last', False}, 保留重复元素中的第一个、最后一个, 或全部删除；
- inplace, 默认为False, 返回一个新的dataframe; 若为True, 则返回去重后的原dataframe

group

group一般会配合**合计函数** (Aggregate functions) 使用, 比如: count、avg等。Pandas对合计函数的支持有限, 有count和size函数实现SQL的count:

```
print df.groupby('sex').size()
print df.groupby('sex').count()
print df.groupby('sex')['tip'].count()
```

对于多合计函数,

```
select sex, max(tip), sum(total_bill) as total
from tips_tb
group by sex;
```

实现在agg()中指定dict:

```
print df.groupby('sex').agg({'tip': np.max, 'total_bill': np.sum})

# count(distinct **)
print df.groupby('tip').agg({'sex': pd.Series.nunique})
```

as

SQL中使用as修改列的别名, Pandas也支持这种修改:

```
# first implementation
df.columns = ['total', 'pit', 'xes']
# second implementation
df.rename(columns={'total_bill': 'total', 'tip': 'pit', 'sex': 'xes'}, inplace=True)
```

其中, 第一种方法的修改是有问题的, 因为其是按照列position逐一替换的。因此, 我推荐第二种方法。

join

Pandas中join的实现也有两种:

```
# 1.
df.join(df2, how='left'...)

# 2.
pd.merge(df1, df2, how='left', left_on='app', right_on='app')
```

第一种方法是按DataFrame的index进行join的, 而第二种方法才是按on指定的列做join。Pandas满足left、right、inner、full outer四种join方式。

order

出度 :

```
print df.sort_values(['total_bill', 'tip'], ascending=[False, True])
```

top

对于全局的top :

```
print df.nlargest(3, columns=['total_bill'])
```

对于分组top, MySQL的实现 (采用自join的方式) :

```
select a.sex, a.tip
from tips_tb a
where (
    select count(*)
    from tips_tb b
    where b.sex = a.sex and b.tip > a.tip
) < 2
order by a.sex, a.tip desc;
```

Pandas的等价实现, 思路与上类似 :

```
# 1.
df.assign(rn=df.sort_values(['total_bill'], ascending=False)
        .groupby('sex')
        .cumcount()+1)\
    .query('rn < 3')\
    .sort_values(['sex', 'rn'])

# 2.
df.assign(rn=df.groupby('sex')['total_bill']
        .rank(method='first', ascending=False)) \
    .query('rn < 3') \
    .sort_values(['sex', 'rn'])
```

replace

[replace](#)函数提供对dataframe全局修改, 亦可通过where条件进行过滤修改 (搭配loc) :

```
# overall replace
df.replace(to_replace='Female', value='Sansa', inplace=True)

# dict replace
df.replace({'sex': {'Female': 'Sansa', 'Male': 'Leone'}}),
inplace=True)

# replace on where condition
df.loc[df.sex == 'Male', 'sex'] = 'Leone'
```

自定义

除了上述SQL操作外, Pandas提供对每列/每一元素做自定义操作, 为此而设计以下三个函数 :

- map(func), 为Series的函数, DataFrame不能直接调用, 需取列后再调用;
- apply(func), 对DataFrame中的某一行/列进行func操作;
- applymap(func), 为element-wise函数, 对每一个元素做func操作

```
print df['tip'].map(lambda x: x - 1)
print df[['total_bill', 'tip']].apply(sum)
```

3. 实战

环比增长

现有两个月APP的UV数据, 要得到月UV环比增长; 该操作等价于两个Dataframe left join后按指定列做减操作:

```
def chain(current, last):
    df1 = pd.read_csv(current, names=['app', 'tag', 'uv'], sep='\t')
    df2 = pd.read_csv(last, names=['app', 'tag', 'uv'], sep='\t')
    df3 = pd.merge(df1, df2, how='left', on='app')
    df3['uv_y'] = df3['uv_y'].map(lambda x: 0.0 if pd.isnull(x) else x)
    df3['growth'] = df3['uv_x'] - df3['uv_y']
    return df3[['app', 'growth', 'uv_x', 'uv_y']].sort_values(by='growth', ascending=False)
```

差集

对于给定的列, 一个Dataframe过滤另一个Dataframe该列的值; 相当于集合的差集操作:

```
def difference(left, right, on):
    """
    difference of two dataframes
    :param left: left dataframe
    :param right: right dataframe
    :param on: join key
    :return: difference dataframe
    """
    df = pd.merge(left, right, how='left', on=on)
    left_columns = left.columns
    col_y = df.columns[left_columns.size]
    df = df[df[col_y].isnull()]
    df = df.ix[:, 0:left_columns.size]
    df.columns = left_columns
    return df
```

如需转载, 请注明作者及出处.

作者: [Treant](#)

出处: <http://www.cnblogs.com/en-heng/>

分类: [Python](#)

好文要顶

关注我

收藏该文



Treant

关注 - 78

粉丝 - 197

[+加关注](#)

2

0

« 上一篇: [【Python实战】Scrapy豌豆荚应用市场爬虫](#)

» 下一篇: [Scala比较器: Ordered与Ordering](#)

posted @ 2016-06-30 18:08 Treant 阅读(20330) 评论(4) 编辑 收藏

评论列表

#1楼 2016-06-30 18:22 会长

您是做大数据分析工作的吗? 我也打算学一些呢

[支持\(0\)](#) [反对\(0\)](#)

#2楼[楼主] 2016-06-30 19:04 Treant

3170

您是做大数据分析工作的吗？我也打算学一些呢

是的

支持(0) 反对(0)

#3楼 2016-07-01 10:11 DelphiSeattle

后面学的东西太多了，数据分析这个门槛太高了！

支持(1) 反对(0)

#4楼 2016-07-20 07:53 智浪淘沙

不错的笔记，非常感谢

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【推荐】极光开发者服务平台，五大功能一站集齐

【推荐】阿里云“全民云计算”优惠升级

【推荐】一小时搭建人工智能应用，让技术更容易入门

 滴滴

 Google

联合打造

零基础成为机器学习工程师

仅限300席 | [立即查看](#)

成为人工智能、数据分析领域稀缺人才

最新IT新闻:

- [苹果CEO库克写备忘录谴责白人至上主义](#)
 - [小猿搜题回应百度声明《即使树大招风，依然谢绝碰瓷》：呵呵](#)
 - [李彦宏为女儿吉他伴奏是假弹？其实不是的](#)
 - [京东看呆！天猫送货全面大提速：这次放出绝招](#)
 - [转眼竟已22年！全新未拆封Windows 95重现人间](#)
- » [更多新闻...](#)

 JIGUANG | 极光

移动开发首选 极光

»»» 推送 IM 短信 统计 分享



最新知识库文章:

- [做到这一点，你也可以成为优秀的程序员](#)
 - [写给立志做码农的大学生](#)
 - [架构腐化之谜](#)
 - [学会思考，而不只是编程](#)
 - [编写Shell脚本的最佳实践](#)
- » [更多知识库文章...](#)