

Pythondataframe多条件筛选过滤数据的方法及函数isin，query，cont。。。

1. 背景概述

日常的数据分析中，经常要根据各种不同的条件从数据集中筛选出相应的数据记录，再进行提取、替换、修改和分析等操作。因此筛选是数据分析中使用频率最高的操作之一。在刚开始做数据分析的时候，常常是使用for循环在数据集中进行条件筛选，导致代码比较冗长且效率不高。本文总结了在python中常用的并且使用效率比较高的几种数据筛选函数如：isin()、query()、contains()、loc()等，并且展示了它们单独使用或搭配一起使用的实践效果。

2. 筛选方法和函数简介

2.1 简单的筛选方法：

单一的筛选：条件范围可以是数值或字符串

df[df[“column_name”] == value]

多字段的筛选（又称为复合条件的筛选）：多个不同的特征列，并且条件可以对应不同的数值或字符串

df[(df[“column_name1”] <= value) & (df[“column_name2”] == str)]

2.2 isin函数：df[df[“column_name”].isin(li)] (# li = [20, 25, 27] 或 li = np.arange(20, 30))

根据从isin函数传入的列表(li)，筛选出与列表中包含的数值或字符串相同的数据记录, 用法有点类似sql中的"in"

2.3 query函数：df.query("(column_name1 == ‘str1’) & (column_name2 == ‘str2’)")

根据query中引入的不同字段（str1，str2等）和条件，筛选出同时能满足这些要求的数据记录

2.4 contrains函数：df[df[“column_name”].str.contians(“str”)]

筛选出所有含有(str)的数据记录, 用法类似于sql中的"contains"

2.5 loc函数：df.loc[df["column_name"] <= value]

根据特征属性（列名）或索引标签筛选数据：df.loc[columns 筛选条件] 或df.loc[index 筛选条件];

同时根据索引标签和特征属性（列名）筛选数据：df.loc[index 筛选条件, columns 筛选条件]

2.6 筛选函数之间还能根据各自的特点搭配使用

3. 函数的使用实践

3.1 数据准备和说明

数据准备：数据按行筛选（数据记录）并提取数据

```
import numpy as np
import pandas as pd
df = pd.DataFrame({"name": ["A001", "A002", "B001", "A001_K", "C002", "B001_K", "B001"],
                    "protein": [25, 28, 45, 22, 60, 40, 27],
                    "Qty": [85, 90, 75, 80, 30, 50, 30],
                    "rank": ["1st", "1st", "1st", "2nd", "1st", "1st", "2nd"]})
df
```



3.2 示例代码

3.2.1 简单的条件筛选方法

```
# 1 简单的条件筛选：单一条件筛选
data = df[df["protein"] <= 30]
data
```



代码描述：df.loc[df[“protein”] <= 30] 与 df[df[“protein”] <= 30]的运行结果是一样的。

```
# 2 多重条件筛选 -- 筛选的条件是数值
# 筛选并提取protein 在40-50之间的记录(符合条件)
data = df[(df["protein"] >= 40) & (df["protein"] <= 50)]
data
```



```
# 3 多重条件筛选 -- 筛选的条件有数值和字符串
# 筛选出蛋白质含量大于30并且产品评级为"1st"的数据
data = df[(df["protein"] >= 30) & (df["rank"] == "1st")]
data
```



3.2.2 isin 函数的使用

返回的结果是根据从isin函数传入的列表(li)，筛选出与列表中包含的数值或字符串相同的数据记录, 用法有点类似sql中的"in"

```
# 筛选出与列表中的数值或字符串相等的数据记录
# li = np.arange(20, 30)
li = [25, 60, 45, 40]
data = df[df["protein"].isin(li)]
data
```



3.2.3 query 函数的使用

返回的结果是根据query中引入的不同字段(str1, str2)和条件，筛选出同时能满足这些要求的数据记录

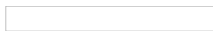
```
# 筛选出名称为"A001"或"B001", 并且级别都是"1st"的数据记录
data = df.query("(name=='A001' | name=='B001') & (rank == '1st')")
data
```



温馨提示：在使用query函数时，所有的表达内容都必须用引号标识出来，并且字符串的引号与表达式的引号需要区分出来（即遵从使用双引号与单引号的套用规则）。另外，特征列名称是不需要使用引号标注的，这可以理解为是直接调用了列表，因此列名称不需要注释。

3.2.4 contains 函数的使用

```
# 1 筛选出所有名称中还有"K"的数据记录
data = df[df["name"].str.contains("K")]
data
```



```
# 2 筛选出级别中含有"st", 并且名字中含有"K"的数据记录
data = df[(df["rank"].str.contains("st") & df["name"].str.contains("K"))]
data
```



3.2.5 loc 函数的使用

```
# 5. loc() 函数 -- 根据标签和特征列名进行数据筛选
# 5.1 单一条件的筛选
data = df.loc[df["protein"] <= 30] # 与df[df["protein"] <= 30]的运行结果是一致的!
data
```



小结：这与之前说的 df[df["protein"] <=30] 的运行结果是一致的！

```
# 5.2 复合条件的筛选
```

```
# 筛选出名称为"B001"并且蛋白质含量低于30的数据记录
data = df.loc[(df["name"]=="B001") & (df["protein"]<=30)]
data
```

```
# 5.3 使用loc函数同时对索引标签和特征属性（列名）进行数据筛选
```

```
# 注意：当使用loc函数根据索引标签和特征列名进行筛选时，需要设定相应的索引标签。本测试会先将名称更换为索引值再进行筛选（使用set_index()函数将特征列名转换为索引值）
```

```
# 筛选出名称为"B001"的所有数据记录
test = df.copy().set_index("name")
test.loc["B001", :]
```

```
# 5.4 筛选出名称为"B001"和"A001"的蛋白质和评级数据
```

```
test_ = test.loc[["B001", "A001"], ["protein", "rank"]]
test_
```

3.2.6 筛选函数的搭配使用

上述介绍的筛选方法和函数是可以被搭配在一起使用，并且效果很不错！

```
# 1 筛选出蛋白质小于等于30，并且级别是含有"2nd"的数据记录
data = df[(df["protein"] <=30) & df["rank"].str.contains("2nd")]
data
```

```
# 2 筛选出蛋白质的含量是列表中的数值，并且名称中含有"K"的数据记录
```

```
li = [25, 60, 45, 40, 22]
data = df[(df["protein"].isin(li)) & (df["name"].str.contains("K"))]
data
```

4. 结束语

1. 单一条件的行或列的筛选可直接使用df[columns 筛选条件] 或df[indx 筛选条件]
2. isin 函数的使用很灵活，能将多个不用的数值范围要求或字段要求通过列表的形式传入函数中进行筛选。
3. query 函数能进行多字段的筛选，但要特别注意列名的引用，以及格式的书写与其他函数不一样的地方。
4. contains 函数其实是相当与SQL 中的contains的用法，能灵活地对字符串的数据进行筛选。
5. 筛选方法和函数是可以根据不同的需要被搭配在一起形成多重的条件筛选，并且使用的效果很不错！
6. 如果是同时对行和列进行筛选，可使用df.loc[index 筛选条件, columns 筛选条件]

基础概念：**loc**函数与**iloc**函数的区别

1. loc() 函数：指Selection by label的函数
是按照标签来提取数据，标签是由2个参数决定的。
第1个参数是index: “0” - “6” ； 第2个参数是column: “name” , “protein” , “Qty” , “rank” # df.loc[6, :], # df.loc[6, “protein”]
2. iloc(n, m) 函数：指Selection by Position的函数
是按位置选择数据，即第n行，第m列，只接受整数型 的参数 # df.iloc[:, 0]; #df.iloc[1, :]