**十分钟AI知识点】pandas最详细教程**

[庄蔚](https://www.zhihu.com/people/zhuangwei_1124)

[**庄蔚**](https://www.zhihu.com/people/zhuangwei_1124)

[**​**](https://www.zhihu.com/question/48510028)

上海财经大学 管理科学与工程博士在读

​关注她

18 人赞同了该文章

pandas 最有趣的地方在于里面隐藏了很多包。它是一个核心包，里面有很多其他包的功能。这点很棒，因为你只需要使用 pandas 就可以完成工作。

pandas 相当于 python 中 excel：它使用表（也就是 dataframe)，能在数据上做各种变换，但还有其他很多功能。

如果你早已熟知 python 的使用，可以直接跳到第三段。

让我们开始吧：

import pandas as pd

*别问为什么是「pd」而不是「p」，就是这样。用就行了:)*

**pandas 最基本的功能**

**读取数据**

data = pd.read\_csv( my\_file.csv )

data = pd.read\_csv( my\_file.csv , sep= ; , encoding= latin-1 , nrows=1000, skiprows=[2,5])

*sep 代表的是分隔符。如果你在使用法语数据，excel 中 csv 分隔符是「;」，因此你需要显式地指定它。编码设置为 latin-1 来读取法语字符。nrows=1000 表示读取前 1000 行数据。skiprows=[2,5] 表示你在读取文件的时候会移除第 2 行和第 5 行。*

* 最常用的功能：read\_csv, read\_excel
* 其他一些很棒的功能：read\_clipboard, read\_sql

**写数据**

data.to\_csv( my\_new\_file.csv , index=None)

*index=None 表示将会以数据本来的样子写入。如果没有写 index=None，你会多出一个第一列，内容是 1，2，3，...，一直到最后一行。*

我通常不会去使用其他的函数，像.to\_excel, .to\_json, .to\_pickle 等等，因为.to\_csv 就能很好地完成工作，并且 csv 是最常用的表格保存方式。

**检查数据**



Gives (#rows, #columns)

*给出行数和列数*

data.describe()

*计算基本的统计数据*

**查看数据**

data.head(3)

*打印出数据的前 3 行。与之类似，.tail() 对应的是数据的最后一行。*

data.loc[8]

*打印出第八行*

data.loc[8, column\_1 ]

*打印第八行名为「column\_1」的列*

data.loc[range(4,6)]

*第四到第六行（左闭右开）的数据子集*

**pandas 的基本函数**

**逻辑运算**

data[data[ column\_1 ]== french ]

data[(data[ column\_1 ]== french ) & (data[ year\_born ]==1990)]

data[(data[ column\_1 ]== french ) & (data[ year\_born ]==1990) & ~(data[ city ]== London )]

*通过逻辑运算来取数据子集。要使用 & (AND)、 ~ (NOT) 和 | (OR)，必须在逻辑运算前后加上「and」。*

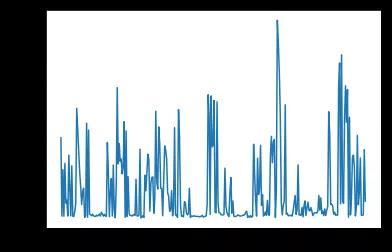
data[data[ column\_1 ].isin([ french , english ])]

*除了可以在同一列使用多个 OR，你还可以使用.isin() 函数。*

**基本绘图**

matplotlib 包使得这项功能成为可能。正如我们在介绍中所说，它可以直接在 pandas 中使用。

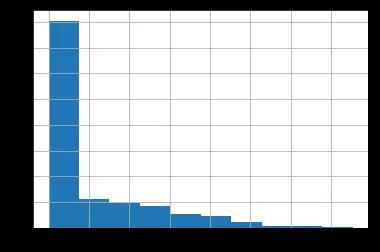
data[ column\_numerical ].plot()



*（）.plot() 输出的示例*

data[ column\_numerical ].hist()

*画出数据分布（直方图）*



*.hist() 输出的示例*

%matplotlib inline

*如果你在使用 Jupyter，不要忘记在画图之前加上以上代码。*

**更新数据**

data.loc[8, column\_1 ] = english

将第八行名为 column\_1 的列替换为「english」

data.loc[data[ column\_1 ]== french , column\_1 ] = French

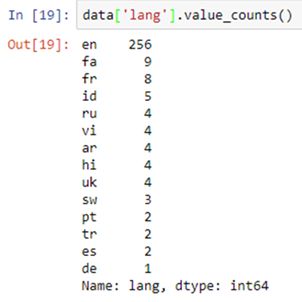
*在一行代码中改变多列的值*

好了，现在你可以做一些在 excel 中可以轻松访问的事情了。下面让我们深入研究 excel 中无法实现的一些令人惊奇的操作吧。

**中级函数**

**统计出现的次数**

data[ column\_1 ].value\_counts()



*.value\_counts() 函数输出示例*

**在所有的行、列或者全数据上进行操作**

data[ column\_1 ].map(len)

*len() 函数被应用在了「column\_1」列中的每一个元素上*

.map() 运算给一列中的每一个元素应用一个函数

data[ column\_1 ].map(len).map(lambda x: x/100).plot()

*pandas 的一个很好的功能就是链式方法（[https://tomaugspurger.github.io/method-chaining](https://link.zhihu.com/?target=https%3A//tomaugspurger.github.io/method-chaining" \t "_blank)）。它可以帮助你在一行中更加简单、高效地执行多个操作（.map() 和.plot()）。*

data.apply(sum)

*.apply() 会给一个列应用一个函数。*

.applymap() 会给表 (DataFrame) 中的所有单元应用一个函数。

**tqdm, 唯一的**

在处理大规模数据集时，pandas 会花费一些时间来进行.map()、.apply()、.applymap() 等操作。tqdm 是一个可以用来帮助预测这些操作的执行何时完成的包（是的，我说谎了，我之前说我们只会使用到 pandas）。

from tqdm import tqdm\_notebook

tqdm\_notebook().pandas()

*用 pandas 设置 tqdm*

data[ column\_1 ].progress\_map(lambda x: x.count( e ))

*用 .progress\_map() 代替.map()、.apply() 和.applymap() 也是类似的。*

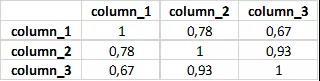
https://pic1.zhimg.com/v2-85f7c8c973b5a43d9c2de9d2247cc538_b.jpg

*在 Jupyter 中使用 tqdm 和 pandas 得到的进度条*

**相关性和散射矩阵**

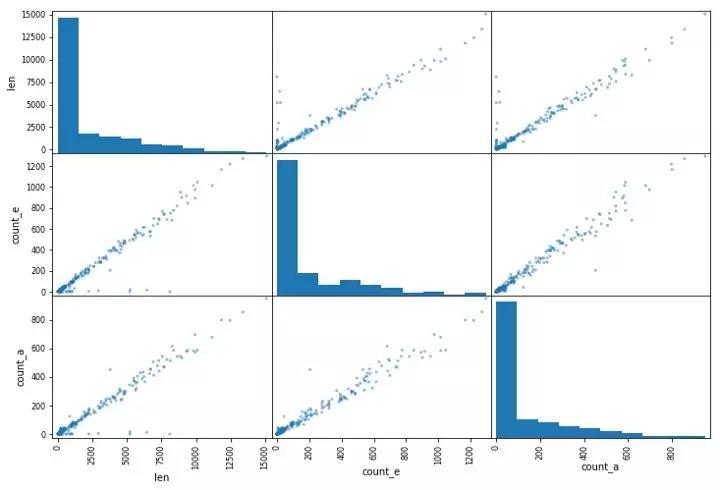
data.corr()

data.corr().applymap(lambda x: int(x\*100)/100)



*.corr() 会给出相关性矩阵*

pd.plotting.scatter\_matrix(data, figsize=(12,8))



*散点矩阵的例子。它在同一幅图中画出了两列的所有组合。*

**pandas 中的高级操作**

**The SQL 关联**

在 pandas 中实现关联是非常非常简单的

data.merge(other\_data, on=[ column\_1 , column\_2 , column\_3 ])

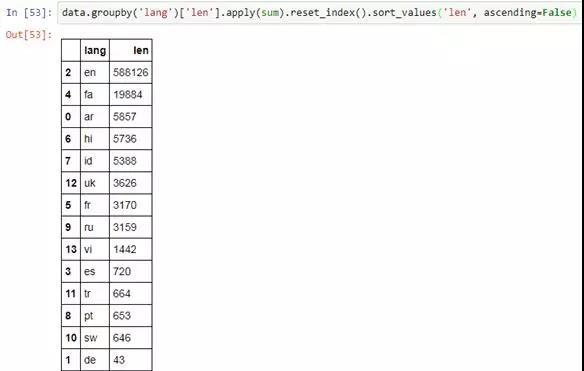
*关联三列只需要一行代码*

**分组**

一开始并不是那么简单，你首先需要掌握语法，然后你会发现你一直在使用这个功能。

data.groupby( column\_1 )[ column\_2 ].apply(sum).reset\_index()

*按一个列分组，选择另一个列来执行一个函数。.reset\_index() 会将数据重构成一个表。*



*正如前面解释过的，为了优化代码，在一行中将你的函数连接起来。*

**行迭代**

dictionary = {}

for i,row in data.iterrows():

dictionary[row[ column\_1 ]] = row[ column\_2 ]

*.iterrows() 使用两个变量一起循环：行索引和行的数据 (上面的 i 和 row)*