Aula 8 - Carregando e manipulando dados com Pandas

Karl Jan Clinckspoor

15 de julho de 2018

Sumário

1 Introdução

Pandas é um pacote com base no numpy utilizado bastante para ciência de dados. Aqui, usaremos uma pequena fração de suas capacidades. A sintaxe do pandas é um pouquinho diferente do já utilizado, e pode ser um pouco frustrante de início. Com o tempo, as coisas começam a fazer um pouco mais de sentido.

2 Abrindo um arquivo

Nas aulas passadas, foram feitas funções para conseguir extrair o conteúdo de alguns arquivos. O processo foi um tanto trabalhoso. O pandas permite abrir dados .csv e .xls(x) com muito mais facilidade. Vamos reabrir o arquivo Consolidação1.txt e algum dos arquivos da pasta dados-1. Para isso, utilizamos a função **read_csv**, colocamos o resultado na variável df (de *dataframe*, o tipo de objeto no pandas) e visualizamos o começo do *dataframe* com a função **head**.

```
In [5]: import pandas as pd
        df = pd.read_csv('Consolidação1.txt')
        df.head()
```

```
Out[5]:              x y
        0  0.0000 4.9332
        1  0.3344 2.6304
        2  0.6689 0.2616
        3  1.0033 9.6285
        4  1.3378 9.8156
```

Notamos que ele aparentemente não conseguiu diferenciar direito as duas colunas. Para nos assegurarmos, vamos ver as informações que o pandas nos dá para esse dataframe.

```
In [7]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 1 columns):
x y    300 non-null object
dtypes: object(1)
memory usage: 2.4+ KB
```

Vemos então que temos, de fato, somente 1 coluna, cujo nome é 'x y', que contém 300 'objetos'. Geralmente quando o pandas fala que há objetos numa coluna, ele quer dizer *strings*.

O ponto é que, por padrão, o pandas tenta abrir arquivos csv assumindo que o separador é uma vírgula (afinal, o nome do arquivo é *comma separated values*). Então, é necessário informar o separador com a keyword *sep*.

```
In [10]: df = pd.read_csv('Consolidação1.txt', sep=' ')
         df.head()

Out[10]:          x       y
         0  0.0000  4.9332
         1  0.3344  2.6304
         2  0.6689  0.2616
         3  1.0033  9.6285
         4  1.3378  9.8156

In [11]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 2 columns):
x    300 non-null float64
y    300 non-null float64
dtypes: float64(2)
memory usage: 4.8 KB
```

Agora vemos que ele, de fato, conseguiu separar em duas colunas, com os nomes x e y, e que contém floats. Para acessar uma dessas colunas, utilizamos uma notação parecida com a de dicionários.

```
In [13]: df['x'].head()

Out[13]: 0    0.0000
         1    0.3344
         2    0.6689
         3    1.0033
         4    1.3378
         Name: x, dtype: float64
```

3 Operações sobre colunas

E também, parecido com o numpy, podemos aplicar operações nas colunas, abrangendo-as por inteiro. Vamos agora criar uma coluna 'x1' que contém os valores de 'x' somados a 1, e vamos calcular as médias dessas duas colunas.

```
In [16]: df['x1'] = df['x'] + 1
         print(f"Média de x: {df['x'].mean():.0f}\nMédia de x1: {df['x1'].mean():.0f}")

Média de x: 50
Média de x1: 51
```

Agora que essa operação foi feita, podemos ver que há uma nova coluna no *dataframe*.

```
In [17]: df.head()

Out[17]:         x       y      x1
         0  0.0000  4.9332  1.0000
         1  0.3344  2.6304  1.3344
         2  0.6689  0.2616  1.6689
         3  1.0033  9.6285  2.0033
         4  1.3378  9.8156  2.3378
```

4 Seccionando dataframes

Suponha que desejamos somente os pontos após o vigésimo ponto do dataframe. Fazemos isso utilizando a localização com base no índice, ou *iloc*. Note que é necessário usar colchetes ao invés de parênteses, pois iloc não é uma função.

```
In [19]: df1 = df.iloc[20:, :]
         df1.head()

Out[19]:          x       y      x1
         20  6.6890  2.5082  7.6890
         21  7.0234  9.0577  8.0234
         22  7.3579  7.6524  8.3579
         23  7.6923  4.4824  8.6923
         24  8.0268  8.7327  9.0268
```

O mesmo pode ser feito utilizando dois colchetes juntos, como se estivéssemos aplicando uma operação sobre arrays/listas 2D.

```
In [21]: df1 = df[:][20:]
         df1.head()
```

```
Out[21]:          x       y      x1
         20  6.6890  2.5082  7.6890
         21  7.0234  9.0577  8.0234
         22  7.3579  7.6524  8.3579
         23  7.6923  4.4824  8.6923
         24  8.0268  8.7327  9.0268
```

Podemos também selecionar somente algumas colunas para essa transição. Isso é feito passando uma lista com os nomes das colunas que serão copiadas.

```
In [22]: cols = ['x', 'x1']
         df1 = df[cols]
         df1.head()
```

```
Out[22]:          x      x1
         0  0.0000  1.0000
         1  0.3344  1.3344
         2  0.6689  1.6689
         3  1.0033  2.0033
         4  1.3378  2.3378
```

É possível utilizar *loc* ao invés de *iloc*, que aceita os nomes das seções (*labels*) ao invés de seus índices. É necessário colocar na ordem *linha,coluna*.

```
In [27]: df.loc[:5, 'x':'y']
```

```
Out[27]:          x       y
         0  0.0000  4.9332
         1  0.3344  2.6304
         2  0.6689  0.2616
         3  1.0033  9.6285
         4  1.3378  9.8156
         5  1.6722  3.4563
```

```
In [28]: df.loc[2:7, ['x', 'x1']]
```

```
Out[28]:          x      x1
         2  0.6689  1.6689
         3  1.0033  2.0033
         4  1.3378  2.3378
         5  1.6722  2.6722
         6  2.0067  3.0067
         7  2.3411  3.3411
```

5 Máscaras lógicas

Suponha agora que desejemos selecionar somente os valores que obedecem uma regra. Se você se lembra, a coluna 'y' possui valores aleatórios entre 0 e 10, com média em torno de 5. Quais são os pontos de x onde o y correspondente é menor que a média?

Para isso, utilizamos as máscaras lógicas. A criação de uma máscara é parecida com uma simples operação de comparação.

```
In [35]: media = df['y'].mean()
         filtro_menor_media = df['y'] < media
         filtro_menor_media.head(n=10)
```

```
Out[35]: 0     True
         1     True
         2     True
         3    False
         4    False
         5     True
         6     True
         7     True
         8     True
         9    False
         Name: y, dtype: bool
```

Agora colocamos esse filtro lógico para selecionar os valores desejados.

```
In [37]: df_filtrado = df[filtro_menor_media]
         df_filtrado.head(n=10)
```

```
Out[37]:          x       y      x1
         0   0.0000  4.9332  1.0000
         1   0.3344  2.6304  1.3344
         2   0.6689  0.2616  1.6689
         5   1.6722  3.4563  2.6722
         6   2.0067  4.2851  3.0067
         7   2.3411  2.9700  3.3411
         8   2.6756  2.6107  3.6756
         10  3.3445  0.4337  4.3445
         11  3.6789  3.1854  4.6789
         14  4.6823  1.1816  5.6823
```

Note que as linhas com os índices 3, 4 e 9 não estão presentes no dataframe filtrado, mostrando que o filtro ocorreu.

Essas funções por si só já permitem que você faça muitas operações complexas.

6 Exercícios

Os exercícios a seguir utilizam dados reais e problemas que eu encontrei durante meu doutorado. O foco dos exercícios é no carregamento dos dados. O tratamento é algo muito específico e fora do escopo desse curso.

6.1 Calorimetria

Dificuldade: [U+2605] [U+2605] [U+2605]
Conceitos utilizados:

- Nomear colunas
- Separadores de colunas
- Separadores decimais
- Cabeçalho (*header*)

Nome do arquivo: 'Calorimetria.csv'
Tarefa: Carregar o dataframe inteiro, depois separar somente as colunas Xt e ΔH.
Dicas:

1. Observe o formato do arquivo antes de começar a importar. Veja quais são os separadores decimais e de coluna. Conte quantos ';' existem na primeira linha e nas linhas subsequentes.
2. Quando você for tratar os seus dados, talvez seja mais fácil fazer pequenas alterações em todos eles, utilizando uma ferramenta como o Notepad++, que permite realizar substituições em todos os arquivos abertos. Veja se isso será útil neste caso.

```
In [ ]: %load ./respostas/Pandas-ITC.py
```

6.2 DSC

Dificuldade: [U+2605] [U+2605]
Conceitos utilizados:

- Linhas de cabeçalho
- Nomear colunas manualmente
- Separador de colunas

Nome do arquivo: 'DSC.txt'
Descrição do experimento:
Com o passar do tempo (t), a temperatura (T) de uma amostra é aumentada. O equipamento fornece uma energia (Q) para que a amostra fique nessa temperatura. Há um fluxo de gás inerte (f). As colunas nos dados estão nessa ordem.
Tarefa: Carregar o dataframe e separar somente a temperatura e a taxa de calor, que são as colunas centrais.

```
In [ ]: %load ./respostas/Pandas-DSC.py
```

6.3 Reologia

Dificuldade: [U+2605] [U+2605] [U+2605] [U+2605]
Conceitos:

- Nomear colunas
- Codificação
- Cabeçalho
- Separador de colunas

- Separador decimal
- Valores que não significam um número (NA)
- Funções de string em dataframes, semelhantes à funções de string normais.
- Extração de dados por filtros

Nome do arquivo: 'Reologia.txt'
Descrição do experimento:
São feitos três experimentos em sequência.

1. No primeiro, chamado de varredura de tensão, é aplicada uma tensão (Tau) e mede-se os parâmetros G' e G" (melhor simbolizar por G1 e G2 por conveniencia).
2. No segundo, chamado de varredura de frequência, é aplicada uma tensão constante variando-se a frequência, omega (simbolizar por w é conveniente), e mede-se os parâmetros G' e G" (simbolize por G1 e G2)
3. No terceiro, chamado de curva de fluxo, é aplicada uma taxa de cisalhamento (GP) e é medida a viscosidade (Eta).

Os três experimentos podem ser separados pela primeira coluna, que contém valores do tipo 1|2, 3|7, etc. O primeiro número é o número do experimento, sendo 1 o oscilatório de tensão, 2 o oscilatório de frequência e 3 a curva de fluxo. Os valores após | se referem ao número do ponto, e não são muito relevantes.

Tarefa: Carregar o arquivo, tomando cuidado com os nomes das colunas, os separadores e a codificação do arquivo (dica: é latin1). Separar o dataframe em três dataframes referentes aos três experimentos, possivelmente pelo uso de filtros, que contém somente as colunas de interesse para aquele experimento. Dica: utilize o fato de que a diferença mais marcante entre eles é a presença de um 1, 2 ou 3 no primeiro valor da primeira coluna. Utilize a função .str.startswith('1').

```
In [ ]: %load ./respostas/Pandas-reologia.py
```