Importação/Manipulação de Dados com Python

Benilton Carvalho

Fatos sobre Python

- Linguagem interpretada, orientada a objeto;
- Possui estruturas de dados de alto nível;
- Usado para scripts ou, apenas, como conector entre diferentes ferramentas;
- Enfatiza a habilidade de leitura do código;
- Utiliza "pacotes", estimulando a modularidade de código;
- Ao contrário do R, não existe um foco (central da linguagem) na análise de dados;
- Gratuito e disponível na maior parte das plataformas.

Pandas

- Biblioteca do Python voltada para a estruturação de dados;
- Mais informações: https://pandas.pydata.org/ ;
- Alta performance, código aberto;
- Permite a realização de análise de dados e modelagem (torna possível a análise sem precisar mudar para R);
- Pandas não implementa estratégias avançadas de análise de dados;
- Estratégias mais elaboradas de análise de dados estão disponíveis em:
  - statsmodels: http://www.statsmodels.org
  - scikit-learn: https://scikit-learn.org

Instalação do Python e Pandas

No Linux,

```
sudo apt-get install python3 python3-pip
sudo pip3 install pandas
```

No MacOS,

```
brew install python3 python3-pip
pip3 install pandas
```

No Windows,

- Baixar o instalar em https://www.python.org/downloads/windows/
- Marcar a opção `add to PATH` no começo do processo
- `python` e `pip` ficarão disponíveis no Windows PowerShell.

- Executar:

  pip install pandas

Chamando Python do R

- Você pode, também, usar o Python a partir do R;
- Cenários assim são comuns quando você precisa conectar ferramentas disponíveis em cada uma das linguagens;
- Por exemplo, para criar as notas de aula em `RMarkdown`;
- Para acessar o Python, você deve utilizar o pacote `reticulate`;
- O código a ser executado em Python deve estar contido num `chunk` específico para `python`;

Chamando o Python a partir do R

```
library(reticulate)
py_discover_config() # Which version of Python is installed?
```

```
## python:         /usr/bin/python
## libpython:      /System/Library/Frameworks/Python.framework/Versions/2.7/l
## pythonhome:     /System/Library/Frameworks/Python.framework/Versions/2.7:/
## version:        2.7.10 (default, Feb 22 2019, 21:55:15)  [GCC 4.2.1 Compat
## numpy:          /System/Library/Frameworks/Python.framework/Versions/2.7/E
## numpy_version:  1.8.0
##
## python versions found:
##  /usr/bin/python
##  /usr/local/bin/python3
```

```
use_python("/usr/local/bin/python3") # Installation path
```

Referência: McKinney (2013) *Python for data analysis*, O'Reilly. O livro é do criador do pacote `pandas`, Wes McKinney.

Python como uma calculadora

- Tutorial: https://docs.python.org/3/tutorial/

```
# Comentários são precedidos por #
dir() # Lista variáveis no workspace
```

```
## ['R', '__annotations__', '__builtins__', '__doc__', '__loader__', '__name_
```

```
a = 1 # int
print(a) # Desnecessário em modo interativo
```

```
## 1
```

Os objetos carregam seus próprios métodos

```
b = "ME315" # str
print(b)
#help(b)  ## ajuda: mas nao dentro do RStudio
#help(matplotlib) ## ajuda: tambem para pacotes
```

```
## ME315
```

```
print(b.endswith("A"))
```

```
## False
```

```
print(b.endswith("5"))
```

```
## True
```

Índices

Arrays no Python começam no 0. Strings são Arrays de caracteres.

```
print(b)
```

```
## ME315
```

```
print(b[0]) # First letter
```

```
## M
```

```
print(b[1]) # Second letter
```

```
## E
```

```
print(b[-1]) # Last letter
```

```
## 5
```

Índices

Referência parcial a objetos e concatenação

```
print(b[:2])
```

```
## ME
```

```
print(b[2:])
```

```
## 315
```

```
print(b[:2] + b[2:])
```

```
## ME315
```

Arrays numéricos

Arrays em Python são guardados em listas

```
x = [1, 2, 4, 8, 16]
print(x)
```

```
## [1, 2, 4, 8, 16]
```

```
print(x[-2:]) # Últimos 2 elementos
```

```
## [8, 16]
```

```
x2 = x + [32, 64, 128] # Concatenação
print(x2)
```

```
## [1, 2, 4, 8, 16, 32, 64, 128]
```

Mais sobre listas

```
print(x2)
```

```
## [1, 2, 4, 8, 16, 32, 64, 128]
```

```
x2[0] = 3 # Modificar valores
print(x2)
```

```
## [3, 2, 4, 8, 16, 32, 64, 128]
```

```
del x2[0] # Remover valores
print(x2)
```

```
## [2, 4, 8, 16, 32, 64, 128]
```

Listas de listas

```
a = ["a","b","c"]
n = [12, 15, 7]
x = [a,n] # Listas de listas
print(x)
```

```
## [['a', 'b', 'c'], [12, 15, 7]]
```

Note: no R, `data.frames` são listas de listas; é errado, do ponto de vista de programação, pensar em `data.frames` como planilhas (i.e. acessar linhas é *lento*, acessar colunas é relativamente rápido).

```
print(x[1]) # Acesso à segunda lista
```

```
## [12, 15, 7]
```

```
print(x[1][0])
```

```
## 12
```

Controle de fluxo

O python naturalmente possui as ferramentas usuais de controle de fluxo, tais como

`if, for, while`

Estrutura de programação é um tópico complexo. Iremos apenas ilustrar o uso das ferramentas com alguns exemplos.

IF

Importante: Identação é feita com tabulações (`\t`) ou dois espaços, e é parte da sintaxe!

```
x = 2
if x < 0:
  x = 0
  print('Negative changed to zero')
elif x == 0:
  print('Zero')
elif x == 1:
  print('Single')
else:
  print('More')
```

```
## More
```

FOR

Importante: Identação é feita com tabulações (`\t`) ou dois espaços, e é parte da sintaxe!

```
words = ['Matemática', 'Estatística', 'Ciência de Dados']
for w in words:
  print(w, len(w))
```

```
## Matemática 10
## Estatística 11
## Ciência de Dados 16
```

Ainda sobre FOR

```
words = ['Matemática', 'Estatística', 'Ciência de Dados', 'IMECC']
for w in words:
  if len(w) >= 6:
    print(w[:3] + '...')
  else:
    print(w)
```

```
## Mat...
## Est...
## Ciê...
## IMECC
```

DEF

A função `def` permite definir funções. Note o escopo!

```
def wins(arr, howMany): # Média Winsorized
  """Linha com documentacao"""
  if len(arr) < 2*howMany:
    return NaN # error
  else:
    arr.sort()
    for i in range(0, howMany, 1):
      arr[i] = arr[howMany]
      arr[-(i+1)] = arr[-(howMany+1)]
    result = sum(arr)/len(arr)
    return result

a = [4,7,3,4,5,2,1,6,999]
print(wins(a, 2))
```

```
## 4.444444444444445
```

```
a = [4,7,3,4,5,2,1,6,999] # Escopo!!
print(sum(a)/len(a))
```

Lendo CSV em Python, pandas

Primeiramente, `import` evoca pacotes e tem função similar a `library()` e/ou `require()` no `R`.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
print(pd.__version__) # Checks pandas version, we need 0.18 at least
```

```
## 0.25.0
```

Baby names

Dados são da SSA (Social Security Agency), mas eu só consegui baixá-los de https://github.com/hadley/data-baby-names/blob/master/baby-names.csv.

```
babyNames = pd.read_csv("../dados/baby-names.csv", header = 0)
print(babyNames)
```

```
##         year      name   percent   sex
## 0       1880      John  0.081541   boy
## 1       1880   William  0.080511   boy
## 2       1880     James  0.050057   boy
## 3       1880   Charles  0.045167   boy
## 4       1880    George  0.043292   boy
## ...      ...       ...       ...   ...
## 257995  2008  Carleigh  0.000128  girl
## 257996  2008     Iyana  0.000128  girl
## 257997  2008    Kenley  0.000127  girl
## 257998  2008    Sloane  0.000127  girl
## 257999  2008   Elianna  0.000127  girl
##
## [258000 rows x 4 columns]
```

Transparência com R

```
print(babyNames.head())
```

```
##    year     name   percent  sex
## 0  1880     John  0.081541  boy
## 1  1880  William  0.080511  boy
## 2  1880    James  0.050057  boy
## 3  1880  Charles  0.045167  boy
## 4  1880   George  0.043292  boy
```

Transparência com R

Somente os 1000 nomes mais comuns de cada ano....

```
print(babyNames.groupby(['year','sex']).name.count())
```

```
## year  sex
## 1880  boy     1000
##       girl    1000
## 1881  boy     1000
##       girl    1000
## 1882  boy     1000
##               ...
## 2006  girl    1000
## 2007  boy     1000
##       girl    1000
## 2008  boy     1000
##       girl    1000
## Name: name, Length: 258, dtype: int64
```

Transparência com R

Alguns verbos coincidem com `dplyr`

```
print(babyNames.groupby(['year','sex']).percent.sum())
```

```
## year  sex
## 1880  boy     0.930746
##       girl    0.934546
## 1881  boy     0.930439
##       girl    0.932690
## 1882  boy     0.927532
##                 ...
## 2006  girl    0.684830
## 2007  boy     0.801105
##       girl    0.677453
## 2008  boy     0.795414
##       girl    0.672516
## Name: percent, Length: 258, dtype: float64
```

Indexando

Não é possível indexar diretamente um DataFrame, você precisa acessar o atributo `iloc`

```
print(babyNames.iloc[0,:])
```

```
## year           1880
## name           John
## percent    0.081541
## sex             boy
## Name: 0, dtype: object
```

No (significant number of) boys named Sue...

```
print(babyNames.loc[babyNames.name == "Sue",:])
```

```
##         year name   percent   sex
## 129189  1880  Sue  0.000666  girl
## 130185  1881  Sue  0.000678  girl
## 131171  1882  Sue  0.000726  girl
## 132216  1883  Sue  0.000566  girl
## 133194  1884  Sue  0.000669  girl
## ...      ...  ...       ...   ...
## 229543  1980  Sue  0.000193  girl
## 230654  1981  Sue  0.000152  girl
## 231777  1982  Sue  0.000116  girl
## 232885  1983  Sue  0.000096  girl
## 233984  1984  Sue  0.000082  girl
##
## [105 rows x 4 columns]
```

No (significant number of) boys named Sue...

```
babySue = babyNames.loc[babyNames.name == "Sue",]
babySue.plot(kind = 'scatter', x = 'year', y = 'percent')
plt.show() # from matplotlib
```

Formatos Suportados pelo Pandas

Format Type	Data Description	Reader	Writer
text	CSV (*)	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	OpenDocument	read_excel	

Observações:

- `read_csv` possui o argumento `delimiter`, que pode ser ajustado para outros tipos de arquivos em texto plano retangulares;
- `read_csv` também possui o argumento `chunksize`, que pode ser usado para leitura por partes.

Formatos Suportados pelo Pandas

Format Type	Data Description	Reader	Writer
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

Pandas e Chunks

```
reader = pd.read_csv("../dados/baby-names.csv",
        header = 0, chunksize=1000)
soma = 0
for df in reader:
  soma += df.loc[df.name == "Sue", :].percent.sum()
print(soma)
```

```
## 0.10973800000000006
```

SQLite, Pandas e Python

```
import pandas as pd
import sqlite3
conn = sqlite3.connect("../dados/disco.db")
pd.read_sql_query("SELECT * FROM artists LIMIT 5", conn)
```

```
##    ArtistId                Name
## 0         1              AC/DC
## 1         2              Accept
## 2         3           Aerosmith
## 3         4  Alanis Morissette
## 4         5    Alice In Chains
```

MongoDB, Pandas e Python

```
from pymongo import MongoClient
import pprint
myurl = 'mongodb+srv://benilton:123mudei@cluster0-s8gg0.mongodb.net'
client = MongoClient(myurl)
db = client['me315mongodb']
collection = db['diamantes']
collection
```

```
## Collection(Database(MongoClient(host=['cluster0-shard-00-02-s8gg0.mongodb.
```

MongoDB

```
doc = collection.find_one()
pprint.pprint(doc)
```

```
## {'_id': ObjectId('5dc555a016f20211cb54edd2'),
##  'carat': 0.23,
##  'clarity': 'SI2',
##  'color': 'E',
##  'cut': 'Ideal',
##  'depth': 61.5,
##  'price': 326,
##  'table': 55.0,
##  'x': 3.95,
##  'y': 3.98,
##  'z': 2.43}
```

MongoDB

```
doc = collection.find_one({"cut":"Premium"})
pprint.pprint(doc)
```

```
## {'_id': ObjectId('5dc555a016f20211cb54edd3'),
##  'carat': 0.21,
##  'clarity': 'SI1',
##  'color': 'E',
##  'cut': 'Premium',
##  'depth': 59.8,
##  'price': 326,
##  'table': 61.0,
##  'x': 3.89,
##  'y': 3.84,
##  'z': 2.31}
```