Importação/Manipulação de Dados com Python

Parte 2

Lendo CSV em Python, pandas

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

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

```
## 1.1.4
```

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.

```
#R
babyNamesR = readr::read_csv("../dados/baby-names.csv") %>% as.data.f
babyNamesR %>% head(n=3)
```

```
##   year    name  percent sex
## 1 1880    John 0.081541 boy
## 2 1880 William 0.080511 boy
## 3 1880   James 0.050057 boy
```

```
#python
babyNamesPY = pd.read_csv("../dados/baby-names.csv", header = 0)
babyNamesPY.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
```

Em R

```
class(babyNamesR)
```

```
## [1] "data.frame"
```

```
class(py$babyNamesPY)
```

```
## [1] "data.frame"
```

Em python

```
type(babyNamesPY)
```

```
## <class 'pandas.core.frame.DataFrame'>
```

```
type(r.babyNamesR)
```

```
## <class 'pandas.core.frame.DataFrame'>
```

Em python

```
babyNamesPY.shape
```

```
## (258000, 4)
```

```
babyNamesPY.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 258000 entries, 0 to 257999
## Data columns (total 4 columns):
##  #   Column   Non-Null Count   Dtype  
## ---  ------   --------------   -----  
##  0   year     258000 non-null  int64  
##  1   name     258000 non-null  object 
##  2   percent  258000 non-null  float64
##  3   sex      258000 non-null  object 
## dtypes: float64(1), int64(1), object(2)
## memory usage: 7.9+ MB
```

Selecionando colunas

```
print(babyNamesPY.year) #ou print(babyNamesPY['year'])
```

```
## 0         1880
## 1         1880
## 2         1880
## 3         1880
## 4         1880
##           ...
## 257995    2008
## 257996    2008
## 257997    2008
## 257998    2008
## 257999    2008
## Name: year, Length: 258000, dtype: int64
```

Apply functions on DataFrames

```
df = pd.DataFrame(
  {'A': [1, 2, 3],
   'B':[4, 5, 6]
  })
df.apply(np.mean,axis=0)
```

```
## A    2.0
## B    5.0
## dtype: float64
```

```
df.apply(np.mean,axis=1)
```

```
## 0    2.5
## 1    3.5
## 2    4.5
## dtype: float64
```

Dados do SSA contêm somente os 1000 nomes mais comuns de cada ano....

```
print(babyNamesPY.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
```

Alguns verbos coincidem com `dplyr`

```
print(babyNamesPY.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 (linhas)

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

```
print(babyNamesPY.iloc[0]) # ou print(babyNamesPY.iloc[0,:])
```

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

```
print(babyNamesPY.iloc[0:3])
```

```
##    year     name   percent  sex
## 0  1880     John  0.081541  boy
## 1  1880  William  0.080511  boy
## 2  1880    James  0.050057  boy
```

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

```
print(babyNamesPY.loc[babyNamesPY.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]
```

Indexando (linhas e colunas)

```
print(babyNamesPY.iloc[0,0])
```

```
## 1880
```

```
print(babyNamesPY.loc[0,'name'])
```

```
## John
```

```
print(babyNamesPY.loc[[0,10,100],['name','year']])
```

```
##         name  year
## 0       John  1880
## 10    Edward  1880
## 100    Perry  1880
```

```
print(babyNamesPY.index)
```

```
## RangeIndex(start=0, stop=258000, step=1)
```

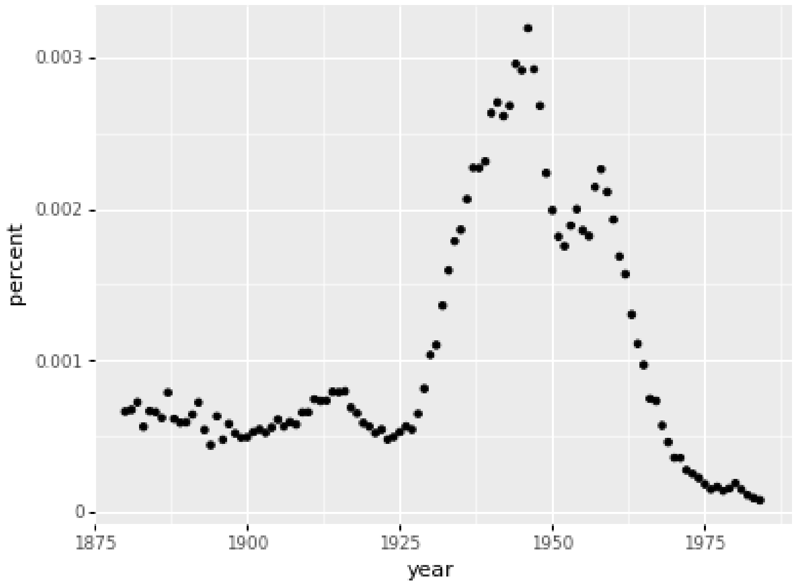
Plot usando matplotlib

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

Plot usando ggplot

```python
from plotnine import *
ggplot(babySue) + geom_point(aes(x = 'year', y = 'percent'))
```

```
## <ggplot: (-9223372036806471017)>
```



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

```python
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

```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
```

```python
conn.close()
```

MongoDB, Pandas e Python

```
from pymongo import MongoClient
import pprint
myurl = "mongodb+srv://fernandaBD:mongo123@cluster0.2ph3s.mongodb.net
client = MongoClient(myurl)
db = client['me315mongodb']
collection = db['diamantes']
collection
```

```
## Collection(Database(MongoClient(host=['cluster0-shard-00-01.2ph3s.mongodb.
```

MongoDB

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

```
## {'_id': ObjectId('5fd034c6e17a0000d50063aa'),
##  'carat': 0.22,
##  'clarity': 'VS2',
##  'color': 'E',
##  'cut': 'Fair',
##  'depth': 65.1,
##  'price': 337,
##  'table': 61.0,
##  'x': 3.87,
##  'y': 3.78,
##  'z': 2.49}
```

MongoDB

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

```
## {'_id': ObjectId('5fd034c6e17a0000d50063ae'),
##  'carat': 0.22,
##  'clarity': 'SI1',
##  'color': 'F',
##  'cut': 'Premium',
##  'depth': 60.4,
##  'price': 342,
##  'table': 61.0,
##  'x': 3.88,
##  'y': 3.84,
##  'z': 2.33}
```

MongoDB

```
doc = collection.find({"cut":"Premium"}).limit(5)
for x in doc:
  pprint.pprint(x,width=10)
```

```
## {'_id': ObjectId('5fd034c6e17a0000d50063ae'),
##  'carat': 0.22,
##  'clarity': 'SI1',
##  'color': 'F',
##  'cut': 'Premium',
##  'depth': 60.4,
##  'price': 342,
##  'table': 61.0,
##  'x': 3.88,
##  'y': 3.84,
##  'z': 2.33}
## {'_id': ObjectId('5fd034c6e17a0000d50063da'),
##  'carat': 0.3,
##  'clarity': 'SI2',
##  'color': 'J',
##  'cut': 'Premium',
##  'depth': 59.3,
##  'price': 405,
```