Python para Ciência de Dados

SQLAlchemy, Web Scraping, Clean Data, Data Visualization

Luiz Alberto

Ciência da Computação

June 1, 2019

Contando Valores Distintos com SQLAlchemy

```
1
2  from sqlalchemy import create_engine, Table, MetaData, select,
       func
3
4  engine = create_engine('sqlite:///datasets/Northwind.sqlite')
5
6  metadata = MetaData()
7  orders   = Table('Orders', metadata, autoload=True,
       autoload_with=engine)
8
9  stmt   = select([func.count(orders.columns.ShipCountry.
       distinct())])
10
11 with engine.connect() as conn:
12   distinct_countries = conn.execute(stmt).scalar()
13
14 print(distinct_countries)
15 # 21
```

Contando Registros com SQLAlchemy (1)

```
1  import pandas as pd
2  from sqlalchemy import create_engine, Table, MetaData, select,
       func
3
4  engine = create_engine('sqlite:///datasets/Northwind.sqlite')
5
6  metadata  = MetaData()
7  customers = Table('Customers', metadata, autoload=True,
       autoload_with=engine)
8
9  stmt = select([customers.columns.Country
10   ,func.count(customers.columns.Id).label('Count')])
11 stmt = stmt.group_by(customers.columns.Country)
12
13 with engine.connect() as conn:
14   results   = conn.execute(stmt).fetchall()
15   countries = pd.DataFrame(results)
16   countries.columns = results[0].keys()
17
```

Contando Registros com SQLAlchemy (2)

```
18 print(countries)
19 #          Country    Count
20 # 0      Argentina         3
21 # 1        Austria         2
22 # 2        Belgium         2
23 # 3         Brazil         9
24 # 4         Canada         3
```

Somando Valores com SQLAlchemy (1)

```
1  import pandas as pd
2  from sqlalchemy import create_engine, Table, MetaData, select,
       func
3
4  engine = create_engine('sqlite:///datasets/Northwind.sqlite')
5
6  metadata  = MetaData()
7  details   = Table('OrderDetails', metadata, autoload=True,
       autoload_with=engine)
8
9  stmt = select([details.columns.ProductId
10   ,func.sum(details.columns.Id).label('Quantity')])
11 stmt = stmt.group_by(details.columns.ProductId)
12
13 with engine.connect() as conn:
14   results    = conn.execute(stmt).fetchall()
15   quantities = pd.DataFrame(results)
16   quantities.columns = results[0].keys()
17
```

Somando Valores com SQLAlchemy (2)

```
18 print(quantities)
19 #      ProductId  Quantity
20 # 0            1  406841.0
21 # 1            2  470965.0
22 # 2            3  128318.0
23 # 3            4  212436.0
24 # 4            5  106418.0
```

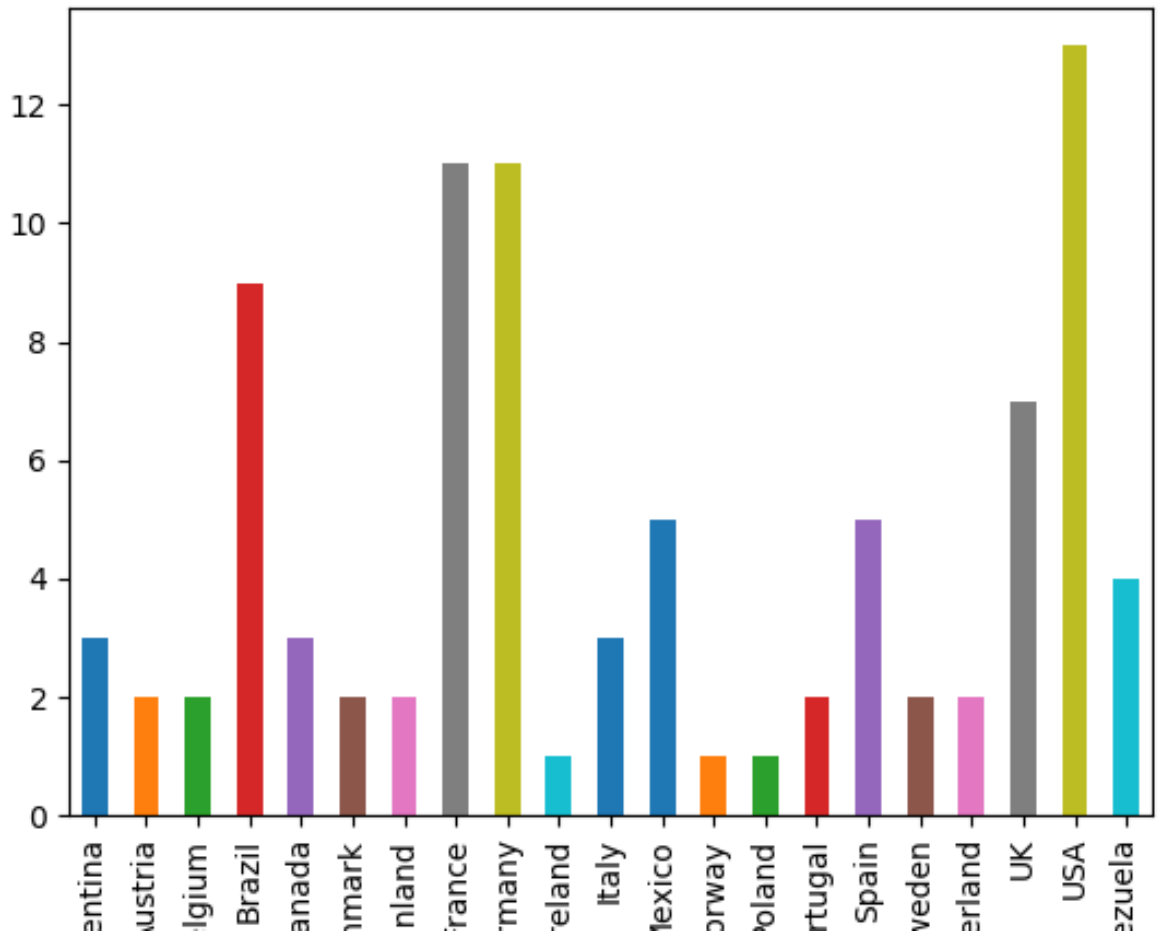
SQLAlchemy, Pandas e Matplotlib (1)

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  from sqlalchemy import create_engine, Table, MetaData, select,
       func
4
5  engine = create_engine('sqlite:///datasets/Northwind.sqlite')
6
7  metadata  = MetaData()
8  customers = Table('Customers', metadata, autoload=True,
       autoload_with=engine)
9
10 stmt = select([customers.columns.Country
11   ,func.count(customers.columns.Id).label('Count')])
12 stmt = stmt.group_by(customers.columns.Country)
13
14 with engine.connect() as conn:
15   results   = conn.execute(stmt).fetchall()
16   countries = pd.DataFrame(results)
17   countries.columns = results[0].keys()
```

SQLAlchemy, Pandas e Matplotlib (2)

```
18
19 countries.plot(kind="bar", x='Country', y='Count', legend=None
       )
20 plt.show()
```

SQLAlchemy, Pandas e Matplotlib (3)



Hora de colocar as mãos na massa

- Na pasta `python-para-ciencia-de-dados/notebooks`, abra o arquivo `aula-4-maos-na-massa-1.ipynb`
- Faça todos os exercícios neste notebook.

Importando dados da web

- O pacote **urllib** fornece uma interface para ler dados da web
- `urlopen` - aceita uma url ao invés de um nome de arquivo

Importando dados da web

```
1
2 import pandas as pd
3 from urllib.request import urlretrieve
4 url = "https://archive.ics.uci.edu/ml/machine-learning-
      databases/wine-quality/winequality-red.csv"
5 urlretrieve(url, 'winequality-red.csv')
6 df = pd.read_csv('winequality-red.csv', sep=';')
7 print(df.head())
```

URL

- Uniform/Universal Resource Locator
- Referencia recursos na web
- Foco: endereço web
- Ingredientes:
  - identificador do protocolo - http:
  - nome do recurso - `python.org`

HTTP

- HyperText Transfer Protocol
- Fundação da comunicação de dados na Web
- HTTPS - forma mais segura do HTTP
- `urlretrieve()` executa uma requisição GET

Requisição GET utilizando **urllib**

```
1
2 import pandas as pd
3 from urllib.request import urlopen, Request
4 url = "https://www.wikipedia.org"
5 request  = Request(url)
6 response = urlopen(request)
7 html = response.read()
8 response.close()
```

Requisição GET utilizando **requests**

```
1 import requests
2 url = "https://www.wikipedia.org"
3 r = requests.get(url)
4 texto = r.text
```

Web Scraping com Python
HTML

- Mistura dados estruturados e não-estruturados
- Dados estruturados:
  - tem um modelo de dados pré-definido
  - organizado de uma forma definida
- Dados não estruturados: nenhum propriedade definida

Parser e Extração Usando **BeautifulSoup**

HTML

- Mistura dados estruturados e não-estruturados
- Dados estruturados:
  - tem um modelo de dados pré-definido
  - organizado de uma forma definida
- Dados não estruturados: nenhum propriedade definida

Parser e Extração Usando **BeautifulSoup**

```
1  from bs4 import BeautifulSoup
2  import requests
3  url='https://www.crummy.com/software/BeautifulSoup/'
4  r = requests.get(url)
5  html_doc = r.text
6  soup = BeautifulSoup(html_doc)
7  print(soup.prettify())
8  # <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 ...
9  # <html>
10 # <head>
11 #  <meta content="text/html; charset=utf-8" http-equiv="
       Content-Type"/>
12 #  <title>
13 #   Beautiful Soup: We called him Tortoise because he taught
       us.
14 #  </title>
15 # ...
```

Explorando o **BeautifulSoup** (1)

- Fornece vários métodos tais como:

```
1  from bs4 import BeautifulSoup
2  import requests
3  url='https://www.crummy.com/software/BeautifulSoup/'
4  r = requests.get(url)
5  html_doc = r.text
6  soup = BeautifulSoup(html_doc)
7  print(soup.title)
8  #  <title>
9  #   Beautiful Soup: We called him Tortoise because he taught
       us.
10 #  </title>
```

Explorando o **BeautifulSoup** (2)

```
1  from bs4 import BeautifulSoup
2  import requests
3  url='https://www.crummy.com/software/BeautifulSoup/'
4  r = requests.get(url)
5  html_doc = r.text
6  soup = BeautifulSoup(html_doc)
7  print(soup.get_text())
8  # Beautiful Soup: We called him Tortoise because he taught us.
9  #
10 #
11 # You didn't write that awful page. You're just trying to get
       some
12 # data out of it. Beautiful Soup is here to help. Since 2004,
       it's been
13 # saving programmers hours or days of work on quick-turnaround
```

Explorando o **BeautifulSoup** (3)

```
1  from bs4 import BeautifulSoup
2  import requests
3  url='https://www.crummy.com/software/BeautifulSoup/'
4  r = requests.get(url)
5  html_doc = r.text
6  soup = BeautifulSoup(html_doc)
7  for link in soup.find_all('a'):
8    print(link.get('href'))
9  # bs4/download/
10 # #Download
11 # bs4/doc/
12 # #HallOfFame
13 # https://code.launchpad.net/beautifulsoup
14 # https://bazaar.launchpad.net/%7Eleonardr/beautifulsoup/bs4/
       view/head:/CHANGELOG
```

Hora de colocar as mãos na massa

- Na pasta `python-para-ciencia-de-dados/notebooks`, abra o arquivo `aula-4-maos-na-massa-2.ipynb`
- Faça todos os exercícios neste notebook.

Entendendo Tidy Data

- "Tidy Data": artigo escrito por Hadley Wickham
- Formaliza a forma de descrição dos dados
- Fornece uma meta para formatação de dados
- "Maneira padrão de organizar valores de dados em um dataframe"

Motivação para Tidy Data

	name	treatment a	treatment b
0	Daniel	-	42
1	John	12	31
2	Jane	24	27

	0	1	2
name	Daniel	John	Jane
treatment a	-	12	24
treatment b	42	31	27

Princípios do Tidy Data

- Colunas representam variáveis separadamente
- Linhas representam observações individuais
- Colunas e observações foram uma tabela

	name	treatment a	treatment b
0	Daniel	-	42
1	John	12	31
2	Jane	24	27

Convertendo para Tidy Data

- Tidy data facilita a correção de problemas comuns em dados
- Problema: na tabela acima a colunas contém valores, ao invés de variáveis
- Solução: `pd.melt()`

	name	treatment a	treatment b
0	Daniel	-	42
1	John	12	31
2	Jane	24	27

	name	treatment	value
0	Daniel	treatment a	-
1	John	treatment a	12
2	Jane	treatment a	24
3	Daniel	treatment b	42
4	John	treatment b	31
5	Jane	treatment b	27

Melting

```
1  import pandas as pd
2  tratamentos = pd.read_csv('tratamento.csv')
3  tratamentos = pd.melt(frame=tratamentos, id_vars='nome',
4    value_vars=['tratamento_a', 'tratamento_b'])
5  print(tratamentos)
6  #       nome       variable  value
7  # 0   Daniel  tratamento_a     20
8  # 1     John  tratamento_a     12
9  # 2     Jane  tratamento_a     24
10 # 3   Daniel  tratamento_b     42
11 # 4     John  tratamento_b     31
12 # 5     Jane  tratamento_b     27
```

Melting

```
1  import pandas as pd
2  tratamentos = pd.read_csv('tratamento.csv')
3  tratamentos = pd.melt(frame=tratamentos, id_vars='nome',
4    value_vars=['tratamento_a', 'tratamento_b'],
5    var_name='tratamento', value_name='resultado')
6  print(tratamentos)
7  #       nome    tratamento  resultado
8  # 0   Daniel  tratamento_a         20
9  # 1     John  tratamento_a         12
10 # 2     Jane  tratamento_a         24
11 # 3   Daniel  tratamento_b         42
12 # 4     John  tratamento_b         31
13 # 5     Jane  tratamento_b         27
```

Pivot: un-melting data

- Oposto da operação de melting
- No melting, convertemos colunas em linhas
- No pivoting, converte valores únicos em colunas separadas
- Torna a forma de relatórios mais amigável
  - Viola o princípio de tidy data: linhas contém observações
  - Múltiplas variáveis em uma mesma coluna

Pivot: un-melting data

```
1 import pandas as pd
2 temperaturas = pd.read_csv('temperaturas.csv')
3 temperaturas = temperaturas.pivot(index='data',
4   columns='temperatura', values='valor')
5 print(temperaturas)
6 # temperatura   tmax   tmin
7 # data
8 # 2010-01-30    27.8   14.5
9 # 2010-02-02    27.3   14.4
```

Pivot: un-melting data

	date	element	value
0	2010-01-30	tmax	27.8
1	2010-01-30	tmin	14.5
2	2010-02-02	tmax	27.3
3	2010-02-02	tmin	14.4

	date	element	value
0	2010-01-30	tmax	27.8
1	2010-01-30	tmin	14.5
2	2010-02-02	tmax	27.3
3	2010-02-02	tmin	14.4
4	2010-02-02	tmin	16.4

Pivot: dados duplicados

```
1 import pandas as pd
2 temperaturas = pd.read_csv('temperaturas_com_repeticao.csv')
3 temperaturas = temperaturas.pivot(index='data',
4   columns='temperatura', values='valor')
5 # ...
6 # ValueError: Index contains duplicate entries, cannot reshape
```

Pivot: dados duplicados

- O método **pivot_table** tem um parâmetro que especifica como o método deve tratar valores duplicados.
- Exemplo: o método pode agregar os valores e calcular a média

Pivot Table: dados duplicados

```
1  import numpy  as np
2  import pandas as pd
3  temperaturas = pd.read_csv('temperaturas_com_repeticao.csv')
4  temperaturas = temperaturas.pivot_table(index='data',
5                                          columns='temperatura',
6                                          values='valor',
7                                          aggfunc=np.mean)
8  print(temperaturas)
9  # temperatura   tmax   tmin
10 # data
11 # 2010-01-30    27.8   14.5
12 # 2010-02-02    27.3   14.2
```

Além do Melt e do Pivot

- Melt e pivot são ferramentas básicas
- Um outro problema comum:
  - colunas que contém mais um dado

Coluna com mais de um dado

	country	year	m014	m1524
0	AD	2000	0	0
1	AE	2000	2	4
2	AF	2000	52	228

Coluna com mais de um dado

```
1  import numpy  as np
2  import pandas as pd
3  pesquisas = pd.read_csv('pesquisa.csv')
4  pesquisas = pd.melt(frame=pesquisas, id_vars=['pais', 'ano'])
5  pesquisas['sexo'] = pesquisas.variable.str[0]
6  print(pesquisas)
7  #    pais   ano variable  value sexo
8  # 0    AD  2000     m014      0    m
9  # 1    AE  2000     m014      2    m
10 # 2    AF  2000     m014     52    m
11 # 3    AD  2000    m1524      0    m
12 # 4    AE  2000    m1524      4    m
13 # 5    AF  2000    m1524    228    m
```

Hora de colocar as mãos na massa

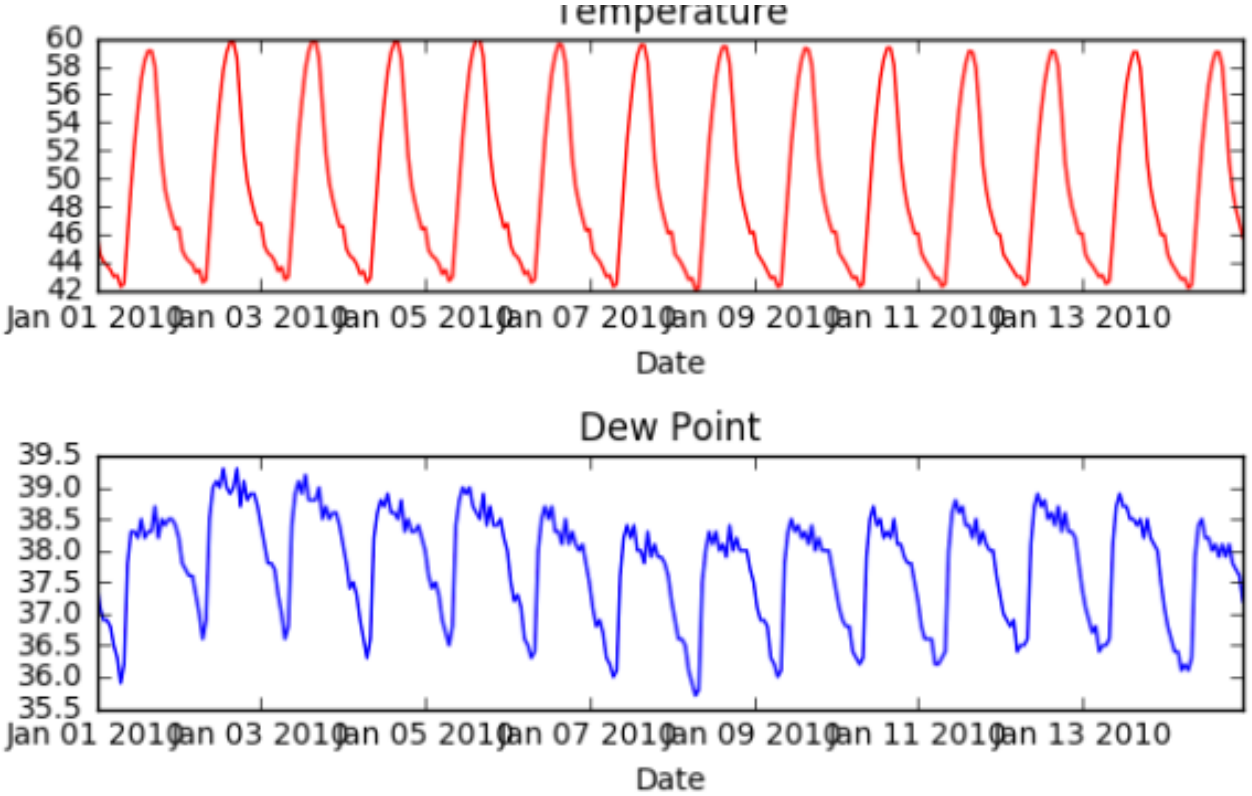
- Na pasta `python-para-ciencia-de-dados/notebooks`, abra o arquivo `aula-4-maos-na-massa-3.ipynb`
- Faça todos os exercícios neste notebook.

Plotando múltiplos gráficos com **subplot**

- `subplot(nrows, ncols, nsubplot)`

```
1  import matplotlib.pyplot as plt
2  plt.subplot(2, 1, 1)
3  plt.plot(t, temperature, 'red')
4  plt.xlabel('Date')
5  plt.title('Temperature')
6  plt.subplot(2, 1, 2)
7  plt.plot(t, dewpoint, 'red')
8  plt.xlabel('Date')
9  plt.title('Dew Point')
10 plt.tight_layout()
11 plt.show()
```

Plotando múltiplos gráficos com **subplot**



Hora de colocar as mãos na massa

- Na pasta `python-para-ciencia-de-dados/notebooks`, abra o arquivo `aula-4-maos-na-massa-4.ipynb`
- Faça todos os exercícios neste notebook.