

## Capítulo

# 12

## Introdução a análise de dados com Python e Pandas

Abílio Soares Coelho

### *Abstract*

*Companies are investing in quantity and generation of information, but few companies are devoting themselves to data analysis and focusing on better positioning in the market. Leverage a large mass of information generated in the planning stages can aid in reducing losses and without increasing billing. Among the most diverse emerging areas of computing, a particular theme has drawn attention and aroused community interest, a data science, and the ability to extract valuable information from large masses of raw data. This chapter is an introduction to the use of Pandas, a Python library for data analysis, which plays a great role and has numerous functions for this purpose. They will be made available through the transparency portal of the federal government for demonstration and removal of important information.*

### *Resumo*

*As organizações estão investindo milhões na geração e armazenamento de informações, mas poucas empresas estão se dedicando com atenção à análise de dados e focando em um posicionamento melhor no mercado. Aproveitar a grande massa de informações geradas nas etapas de planejamento pode auxiliar na redução de perdas e no aumento de faturamento. Dentre as mais diversas áreas emergentes da computação, um tema em especial tem chamado atenção e despertado interesse da comunidade, a ciência de dados é capaz de extrair informações valiosas a partir de grandes massas de dados brutos. Este capítulo trata de uma introdução a utilização do Pandas, uma biblioteca Python para análise de dados, que desempenha um papel grandioso e possui inúmeras funções para esse propósito. Serão utilizados dados públicos disponibilizados pelo portal da transparência do governo federal para demonstração e retirada de informações importantes.*

## 12.1. Introdução

Estamos vivendo em um mundo globalizado e sedento por informação, hoje a maioria das atividades cotidianas estão sendo monitoradas e consequentemente gerando dados, porém segundo [Becker 2015] os dados são apenas a matéria-prima da informação, ou seja, os dados precisam ser interpretados e assim gerar, de fato, a informação.

Segundo [Grus 2016] o mundo está soterrado por dados, desde uma compra que um usuário realiza pela internet, um filme que é assistido online ou até os registros da localização de um smartphone geram dados que podem ser resposta para inúmeras questões como por exemplo um sistema de recomendação. É exatamente aí onde entra a área de ciência de dados, o cientista de dados é o profissional responsável por extrair informações de dados brutos, [Grus 2016] define em forma de piada que um cientista de dados é um profissional que sabe mais de estatística do que um cientista da computação e mais sobre ciência da computação do que um estatístico. Este capítulo trata de uma introdução a utilização do Pandas, uma biblioteca Python para análise de dados, que desempenha um papel grandioso e possui inúmeras funções para esse propósito.

## 12.2. Python

A linguagem de programação Python embora simples e clara é uma linguagem extremamente poderosa, segundo [Menezes 2016] é uma linguagem que vem crescendo em áreas da computação, como inteligência artificial, banco de dados, biotecnologia, animação 3D, aplicativos móveis, jogos e plataforma web. para [Oliphant 2007] Python se destaca como uma plataforma de computação científica, e cita as seguintes razões:

- Linguagem de código aberto, nenhuma permissão adicional é necessária;
- Multiplataforma, não há necessidade de se preocupar em escrever programas com portabilidade limitada;
- Possui sintaxe de fácil entendimento, mas que permite construções sofisticadas como procedural, orientada a objetos e funcional;
- Possui um poderoso interpretador interativo, que permite o aumento da produtividade no processo de desenvolvimento e teste de códigos;
- Possui um grande número de bibliotecas, dos mais variados propósitos, disponíveis para download;
- Possui uma comunidade atuante e participativa na resolução de dúvidas e problemas referentes a Python.

### 12.2.1. Instalação

O interpretador Python não vem instalado por padrão no Microsoft Windows, já no Mac OS X e Linux sua instalação é nativa, porém não conta com a sua versão mais recente. Pode-se contornar este problema e instalar a versão atual das seguintes formas:

- **Python.org:** O interpretador Python está atualmente na versão 3.6.2, porém a versão nativa no MAC OS X é a versão 2.7, uma maneira possível de realizar a instalação da versão atual é fazendo o download da versão corrente no Python Software Foundation<sup>1</sup>.
- **Anaconda<sup>2</sup>:** É uma plataforma de software que integra o Python com vários pacotes que podem ser úteis no desenvolvimento de aplicações científicas. Essa integração facilita a instalação e a manutenção desse software na sua máquina através do conda, um aplicativo que ajuda a gerenciar a instalação e manutenção de pacotes. No site do desenvolvedor está disponível o download para as principais plataformas. Após instalado, o Anaconda ocupa aproximadamente 4GB de espaço em disco.
- **Miniconda:** Como o propósito deste capítulo é uma introdução à análise de dados, é recomendado que o usuário instale o Miniconda<sup>3</sup>. Para quem não tem disponível muito espaço em disco, é uma ótima opção, ocupa aproximadamente 400MB. Além do interpretador Python e o ambiente de desenvolvimento IDLE, miniconda instala também o conda, tornando-o bem intuitivo no gerenciamento e instalação dos pacotes necessários.

### 12.3. Pandas

O Pandas<sup>4</sup> é uma biblioteca licenciada com código aberto que oferece estruturas de dados de alto desempenho e de fácil utilização voltado a análise de dados para a linguagem de programação Python. É amplamente utilizado para a preparação e visualização de dados e tem sido fundamental para impulsionar a utilização de Python na comunidade científica de dados. Contém estruturas de dados de alto nível e ferramentas de manipulação projetadas para tornar a análise de dados rápida e fácil em Python. O Pandas é construído em cima do NumPy, responsável pelo alto desempenho, e facilita a sua utilização em aplicativos baseados no NumPy.

#### 12.3.1. Estrutura de Dados

Para trabalhar com Pandas, o usuário deve estar familiarizado com duas estruturas de dados que são:

##### 12.3.1.1. Séries

A estrutura Série, é um array unidimensional, semelhante a uma lista em Python, porém criado sobre o numpy. Além da velocidade de processamento, a principal característica que o difere de uma lista comum é que seus índices podem ser mutáveis, dependendo da necessidade do programa.

A Figura 12.1 apresenta a criação e manipulação de uma série. A primeira linha define a série *s* com 5 elementos, a segunda linha apresenta a série *s*, a quarta linha

<sup>1</sup>Disponível em: <http://www.Python.org>

<sup>2</sup>Disponível em: <https://www.anaconda.com/download/>

<sup>3</sup>Disponível em: <https://conda.io/miniconda>

<sup>4</sup>Disponível em: <http://Pandas.pydata.org>

```
In [20]: s = pd.Series([100,200,300,400,500])
print(s)
print('')
print(s[3])
print('')
s.index = ['A','B','C','D','E']
print(s)

0    100
1    200
2    300
3    400
4    500
dtype: int64

400

A    100
B    200
C    300
D    400
E    500
dtype: int64
```

**Figura 12.1. Criação de uma série**

apresenta o valor armazenado no índice 3 e a linha 6 altera os índices da série pelas letras A,B,C,D,E. Logo em seguida a série é impressa novamente com os novos índices.

### 12.3.1.2. Dataframes

Segundo [McKinney 2012], um Dataframe representa uma estrutura de dados tabular, semelhante a uma planilha, contendo uma coleção de colunas ordenadas, cada uma das quais pode ser um tipo de valor diferente (numérico, string, booleano, etc.) Para se criar um Dataframe a partir das estruturas de dados nativas em Python, pode-se passar um dicionário de listas para seu construtor. Usando-se o parâmetro *columns*, define no construtor como as colunas serão ordenadas. Por padrão, o construtor do Dataframe irá ordenar as colunas em ordem alfabética.

```
In [19]: estados = {
    'estado': ['Bahia','Ceará','Pernambuco','Maranhão', 'Alagoas', 'Rio Grande do Norte','Piauí','Paraíba','Sergipe'],
    'capital': ['Salvador','Fortaleza', 'Recife', 'São Luís', 'Maceió','Natal','Teresina','João Pessoa', 'Aracajú'],
    'sigla': ['BA','CE','PE','MA','AL','RN','PI','PB','SE'],
    'populacao':[2921087,2591188,1617183,1073893,1013773,869954,844245,791438,632744]
}
nordeste = pd.DataFrame(estados,columns=['estado','capital','sigla','populacao'])
nordeste
```

```
Out[19]:
```

	estado	capital	sigla	populacao
0	Bahia	Salvador	BA	2921087
1	Ceará	Fortaleza	CE	2591188
2	Pernambuco	Recife	PE	1617183
3	Maranhão	São Luís	MA	1073893
4	Alagoas	Maceió	AL	1013773
5	Rio Grande do Norte	Natal	RN	869954
6	Piauí	Teresina	PI	844245
7	Paraíba	João Pessoa	PB	791438
8	Sergipe	Aracajú	SE	632744

**Figura 12.2. Criação de um Dataframe**

A figura 12.2 ilustra a criação e apresentação de um Dataframe contendo informações referentes à região nordeste. As 6 primeiras linhas representam a criação de um dicionário chamado estados, a linha 7 representa, de fato, a criação do Dataframe nor-

deste, passando-se como parâmetro o dicionário e as colunas, a última linha apresenta na tela os dados armazenados no Dataframe.

A primeira coluna da figura 12.2 apresenta os índices padrões, porém como existem campos que possuem identificadores únicos como é o caso de sigla, pode-se optar por alterar os índices padrões pela coluna de siglas. A figura 12.3 ilustra essa modificação, onde a primeira linha substitui os índices padrões pelos valores referentes na coluna sigla, a segunda linha apaga a coluna sigla, para que não haja duplicidade de informações e por último apresenta na tela como ficou o Dataframe nordeste.

```
In [21]: nordeste.index = nordeste['sigla']
del nordeste['sigla']
nordeste
```

Out[21]:

	estado	capital	populacao
sigla			
BA	Bahia	Salvador	2921087
CE	Ceará	Fortaleza	2591188
PE	Pernambuco	Recife	1617183
MA	Maranhão	São Luís	1073893
AL	Alagoas	Maceió	1013773
RN	Rio Grande do Norte	Natal	869954
PI	Piauí	Teresina	844245
PB	Paraíba	João Pessoa	791438
SE	Sergipe	Aracajú	632744

**Figura 12.3. Alteração dos índices de um Dataframe**

### 12.3.2. Carregar Datasets

Geralmente é mais comum a necessidade de fazer a leitura de uma base de dados externa para um Dataframe, o Pandas oferece alguns métodos para esse fim e serão descritos a seguir:

- CSV: para ler um arquivo CSV basta chamar uma função nativa da biblioteca Pandas específica para esse fim, função `read_csv`.
- EXCEL: Pandas permite leitura e escrita no formato Excel. Ler a partir de arquivos Excel requer a biblioteca `xlrd`, pode-se instalar via `pip` (`pip install xlrd`).
- db.py: é uma maneira mais fácil de interagir com bancos de dados. Facilita a pesquisa de tabelas, colunas, visualizações, etc. Ele coloca a ênfase na interação do usuário, na exibição de informações e no fornecimento de funções auxiliares fáceis de usar. Suporta PostgreSQL, MySQL, SQLite, Redshift, MS SQL Server e Oracle.
- Clipboard: tendo dados tabulados, então pode-se copiar para o clipboard e colar direto no Dataframe. A função realiza um bom trabalho identificando qual é o delimitador, mas pode-se usar o parâmetro `sep` para ser explícito.
- URL: Pode-se usar a biblioteca `StringIO` de Python para ler diretamente dados de uma URL. `StringIO` permite tratar uma string como um objeto do tipo arquivo.

- `pydataset`: fornece acesso instantâneo a muitos conjuntos de dados diretamente do Python, a versão atual conta com 757 datasets prontos para serem importados e utilizados juntamente com o Pandas.

### 12.3.3. Filtros em Dataframes

Existem vários métodos para selecionar dados em um Dataframe. Quando se manipula grande quantidade de dados a escolha do método influencia bastante, visto que alguns métodos são mais eficientes que outros. [Pandas Library 2017] destaca os seguintes métodos:

- **Filtro booleano:** É possível filtrar qualquer coluna a partir de uma comparação relacional, por exemplo: pode-se recuperar apenas as capitais que tenham população acima de 1 milhão no exemplo visto na figura 12.2;
- **Método `loc`:** É utilizado para seleção através de índices rotulados, mas também aceita uma matriz booleana;
- **Método `iloc`:** É utilizado para seleção através de índices inteiros, mas também aceita uma matriz booleana;
- **Método `ix`:** Pode decidir indexar posicionalmente ou através de etiquetas dependendo do tipo de dados do índice. Está em desuso desde a versão Pandas 0.20.0.

### 12.3.4. Limpeza de Dados Perdidos

O problema de dados perdidos é de grande relevância para a análise de dados, pois deve-se tomar uma decisão importante para não prejudicar as estatísticas das colunas. O Pandas possui funções interessantes para tratar o problema de dados perdidos, as principais funções são:

- *`dropna`*: Utilizado para excluir linhas ou colunas que possuam dados perdidos;
- *`fillna`*: Utilizado para preencher dados perdidos com base em critérios. Exemplo: Em uma coluna de idade de um Dataframe pode-se utilizar essa função para onde tiver dado perdido substituir pela média das idades da coluna;
- *`isnull`*: Retorna objeto booleano indicando quais valores são *nan*<sup>5</sup>;
- *`notnull`*: Negação de *`isnull`*.

### 12.3.5. Operações no Dataframe

Séries e Dataframes formam o núcleo dos modelos de dados do Pandas em Python. Os conjuntos de dados são lidos primeiro nos Dataframes e, em seguida, várias operações (por exemplo, `group by`, agregações, etc.) podem ser aplicadas muito facilmente às suas colunas.

A tabela 12.1 apresenta um pequeno resumo de alguns comandos úteis para manipulação de informações em um Dataframe.

---

<sup>5</sup>acrônimo de not a number, equivalente a nulo

**Tabela 12.1. Principais comandos em um Dataframe**

Retorno	Comando
Quantidade de linhas e colunas do Dataframe	df.shape
Primeiros 5 registros do Dataframe	df.head()
Últimos 5 registros do Dataframe	df.tail()
Descrição do Index	df.Index
Colunas presentes no Dataframe	df.columns
Contagem de dados não-nulos	df.count()
Criando uma nova coluna em um Dataframe	df['Nova Coluna'] = 0
Soma dos valores de um Dataframe	df.sum()
Menor valor de um Dataframe	df.min()
Maior valor de um Dataframe	df.max()
Resumo estatístico do Dataframe	df.describe()

#### 12.3.5.1. Grouping

Segundo [McKinney 2012], categorizar um conjunto de dados e aplicar uma função a cada grupo, seja uma agregação ou transformação, é muitas vezes um componente crítico de um fluxo de trabalho de análise de dados. Após o carregamento, a fusão e a preparação de um conjunto de dados, uma tarefa familiar é calcular as estatísticas do grupo ou possivelmente tabelas dinâmicas para fins de relatórios ou de visualização. O Pandas fornece uma facilidade de agrupamento flexível e de alto desempenho.

Uma razão para a popularidade de bancos de dados relacionais e *SQL* (que significa "linguagem de consulta estruturada") é a facilidade com que os dados podem ser unidos, filtrados, transformados e agregados. No entanto, as linguagens de consulta como o *SQL* são bastante limitadas nos tipos de operações de grupo que podem ser executadas. Com a expressividade e o poder de Python e Pandas, pode-se realizar operações agrupadas muito mais complexas, utilizando qualquer função que aceite um objeto Pandas ou uma matriz NumPy.

O método *groupby* do Pandas parte da estratégia básica de separar-aplicar-combinar (*split-apply-combine*) em análise de dados. É uma boa representação de como você pensa sobre um problema de dados. Quando ataca-se um problema de análise de dados, geralmente quebra-se ele em pedaços gerenciáveis, realiza várias operações sobre cada um desses pedaços e depois reagrupa todos novamente (este é o essencial da estratégia *split-apply-combine*).

#### 12.3.5.2. Merge

Geralmente em uma análise, precisa-se fazer *merge/join* de dados que geralmente são armazenados de forma relacional. Como a cláusula *JOIN* no *SQL*, *pandas.merge* permite que 2 Dataframes possam ser agrupados por meio de uma ou mais chaves. A função provê uma série de parâmetros (*on*, *left\_on*, *right\_on*, *left\_index*, *right\_index*) permitindo que você possa especificar sobre quais colunas ou índices deseja-se realizar o join. Por padrão,

*pandas.merge* opera como um *inner join*, que pode ser alterado usando o parâmetro *how*. A partir do *docstring*: *how* : 'left', 'right', 'outer', 'inner', default 'inner'

- *left*: usa apenas chaves do frame da esquerda (*SQL*: *left outer join*)
- *right*: usa apenas chaves do frame da direita (*SQL*: *right outer join*)
- *outer*: usa a união das chaves de ambos frames (*SQL*: *full outer join*)
- *inner*: usa a interseção das chaves de ambos frames (*SQL*: *inner join*)

## 12.4. Matplotlib

É a biblioteca Python mais popular para a visualização de dados, geração de gráficos e permite que seja facilmente criado gráficos, histogramas e outras visualizações profissionais. Possui suporte a todos os sistemas operacionais e também pode exportar gráficos para vetor comum.

Segundo [Matplotlib 2017], tenta facilitar coisas fáceis e difíceis. Pode gerar gráficos, histogramas, espectros de poder, gráficos de barras, gráficos de erros, diagramas de dispersão, etc., com apenas algumas linhas de código. Para o usuário avançado, possui controle total de estilos de linha, propriedades de fontes, propriedades de eixos, etc., através de uma interface orientada a objetos ou através de um conjunto de funções familiares aos usuários do MATLAB.

## 12.5. Estudo de Caso

De acordo com [Brasil 2017], O Portal da Transparência do Governo Federal é uma iniciativa da Controladoria-Geral da União (CGU), lançada em novembro de 2004, para assegurar a boa e correta aplicação dos recursos públicos. O objetivo é aumentar a transparência da gestão pública, permitindo que o cidadão acompanhe como o dinheiro público está sendo utilizado e ajude a fiscalizar. Segundo o próprio site, em 2017, possui uma média mensal de 1.794.455 acessos, porém não é tão simples assim fazer consultas específicas no portal, para tanto existe um manual<sup>6</sup> contendo 241 páginas que foi criado com o intuito de facilitar a navegação no Portal. O objetivo é orientar o cidadão e os agentes públicos a encontrarem as informações desejadas para conhecer a aplicação dos recursos públicos federais e contribuir com o Governo em seu papel de fiscalização.

Esta seção apresenta um passo a passo, desde a instalação do ambiente virtual até a apresentação dos dados, de gastos federais com diárias dos campi do Instituto Federal de Educação, Ciência e Tecnologia do Maranhão - IFMA. O portal da transparência permite o download de arquivos em formato .CSV, esses arquivos serão importados e manipulados através de Python e biblioteca Pandas. Para todos efeitos, o sistema operacional adotado nessa demonstração é o macOS High Sierra.

### 12.5.1. Instalação do Python

Segundo [Miniconda 2017], a maneira mais rápida de obter o gerenciador de pacotes conda é instalar a Miniconda, uma mini versão da Anaconda que inclui o interpretador

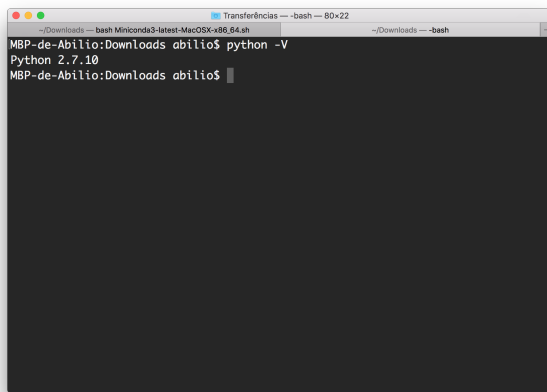
---

<sup>6</sup>Disponível em: <http://www.portaldatransparencia.gov.br/manual/manualCompleto.pdf>



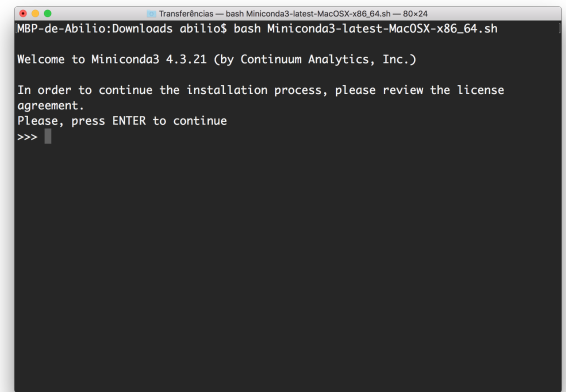
Python, o conda e suas dependências. Se você preferir ter o conda e mais de 720 pacotes de código aberto, recomenda-se a instalação da Anaconda.

Por questões de espaço em disco optou-se pela instalação do Miniconda, que ocupa pouco mais de 400Mb de espaço em disco após instalado.



```
MBP-de-Abilio:Downloads abilio$ python -V
Python 2.7.10
MBP-de-Abilio:Downloads abilio$
```

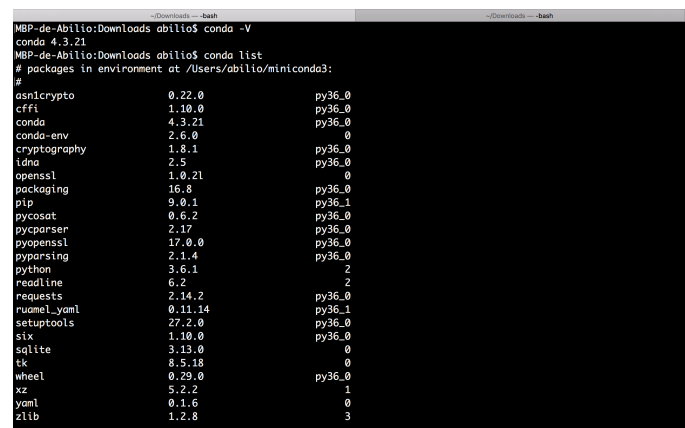
(a) Versão nativa do Python



```
MBP-de-Abilio:Downloads abilio$ bash Miniconda3-latest-MacOSX-x86_64.sh
Welcome to Miniconda3 4.3.21 (by Continuum Analytics, Inc.)

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

(b) Instalação do Miniconda



```
MBP-de-Abilio:Downloads abilio$ conda -V
conda 4.3.21
MBP-de-Abilio:Downloads abilio$ conda list
# packages in environment at /Users/abilio/miniconda3:
#
asn1crypto      0.22.0      py36_0
cffi             1.10.0      py36_0
conda           4.3.21      py36_0
conda-env       2.6.0              0
cryptography    1.8.1      py36_0
idna            2.5         py36_0
openssl         1.0.21      0
packaging       16.8        py36_0
pip             9.0.1       py36_1
pycosat         0.6.2       py36_0
pycparser       2.17        py36_0
pyopenssl       17.0.0      py36_0
pyparsing       2.1.4       py36_0
python          3.6.1              2
readline        6.2         py36_0
requests        2.14.2      py36_1
ruamel_yaml     0.11.14     py36_1
setuptools      27.2.0      py36_0
six             1.10.0      py36_0
sqlite          3.13.0      0
tk              8.5.18      0
wheel           0.29.0      py36_0
xz              5.2.2              1
yaml            0.1.6              0
zlib            1.2.8              3
```

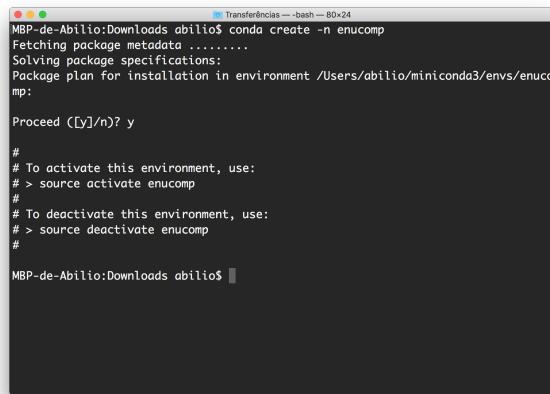
(c) Versão do conda e os pacotes instalados por padrão

**Figura 12.4. Processo de instalação Miniconda (a), (b) e (c)**

A figura 12.4 apresenta um conjunto de imagens que ilustram a instalação do Miniconda. Primeiramente a imagem 12.4(a) mostra que a versão do Python instalada por padrão no macOS High Sierra é a versão 2.7.10, versão antiga do interpretador. Após download no site do desenvolvedor, a imagem 12.4(b) apresenta, de fato a instalação do Miniconda, onde após essa tela é apresentado os termos de licença e o restante da instalação é bem intuitiva. Após a instalação do Miniconda, a imagem 12.4(c) mostra a versão do gerenciador de pacotes conda e posteriormente os pacotes que já são instalados por padrão, pode-se notar que a versão do interpretador Python foi atualizado para a versão 3.6.1.

Após a instalação do miniconda é necessária a criação do ambiente virtual, que é uma técnica bastante utilizada por programadores onde cria-se ambientes com total

independência dos outros. Isso permite que cada ambiente tenha autonomia para instalar plug-ins e bibliotecas de forma que a configuração de um não impacte nos restantes.

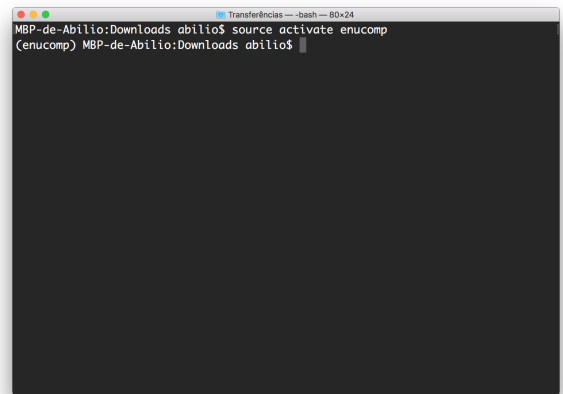


```
MBP-de-Abilio:Downloads abilio$ conda create -n enucomp
Fetching package metadata .....
Solving package specifications:
Package plan for installation in environment /Users/abilio/miniconda3/envs/enucomp:

Proceed ([y]/n)? y

#
# To activate this environment, use:
# > source activate enucomp
#
# To deactivate this environment, use:
# > source deactivate enucomp
#
MBP-de-Abilio:Downloads abilio$
```

(a) Criação do ambiente virtual



```
MBP-de-Abilio:Downloads abilio$ source activate enucomp
(enucomp) MBP-de-Abilio:Downloads abilio$
```

(b) Ativação do ambiente enucomp

**Figura 12.5. Criação e ativação do ambiente virtual (a) e (b)**

A figura 12.5 apresenta duas imagens que ilustram a criação e ativação do ambiente virtual. A imagem 12.5(a) mostra o comando para fazer a criação de um novo ambiente, o parâmetro *-n* refere-se a *name*, onde passa-se como argumento o nome do ambiente a ser criado. Após criado, para efeito de instalações e execuções referentes a um ambiente, é necessário que o mesmo esteja ativo. A imagem 12.5(b) apresenta a ativação do ambiente recentemente criado e após comando verifica-se que o ambiente está ativo, representado pelo nome do ambiente entre parênteses antes do cursor, a partir desse momento, qualquer processo de instalação ou execução realizado no terminal será executado pelo ambiente ativo.

### 12.5.2. Instalação das bibliotecas necessárias

Após o Miniconda instalado, ambiente virtual criado e ativo, é necessária a instalação das bibliotecas que serão utilizadas nessa demonstração.

A figura 12.6, apresenta duas imagens que ilustram a instalação e teste dos pacotes instalados. A imagem 12.6(a) apresenta a utilização do gerenciador de pacotes conda para instalar as bibliotecas necessárias para a realização da proposta. Nota-se que além da biblioteca *Pandas*, é solicitada a instalação do *matplotlib* e *seaborn*, que são bibliotecas responsáveis pela geração de gráficos e do *jupyter* que segundo [Grus 2016], fornece um *shell* com muito mais funcionalidades que o padrão do Python, acrescenta várias novas funções, além do mais, permite que se crie *notebooks*, combinando texto e código Python. A imagem 12.6(b), mostra a realização de testes para verificação se tudo ocorreu como previsto, como não é apresentado nenhum erro de importação então conclui-se que o teste foi bem sucedido.

```
(enucomp) MBP-de-Abilio:Downloads abilio$ conda install pandas matplotlib seaborn jupyter
n jupyter
Fetching package metadata .....
Solving package specifications: .

Package plan for installation in environment /Users/abilio/miniconda3/envs/enucomp:

The following NEW packages will be INSTALLED:

appnope:          0.1.0-py36_0
bleach:           1.5.0-py36_0
certifi:          2016.2.28-py36_0
cycler:           0.10.0-py36_0
decorator:        4.1.2-py36_0
entrypoints:      0.2.3-py36_0
freetype:         2.5.5-2
html5lib:         0.999-py36_0
icu:              54.1-0
ipykernel:        4.6.1-py36_0
ipython:          6.1.0-py36_0
ipython_genutils: 0.2.0-py36_0
ipywidgets:       6.0.0-py36_0
jedi:             0.10.2-py36_2
```

(a) Instalando as bibliotecas

```
(enucomp) MBP-de-Abilio:Downloads abilio$ python
Python 3.6.2 |Continuum Analytics, Inc.| (default, Jul 20 2017, 13:14:59)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> import pandas as pd
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>>
```

(b) Testando as bibliotecas

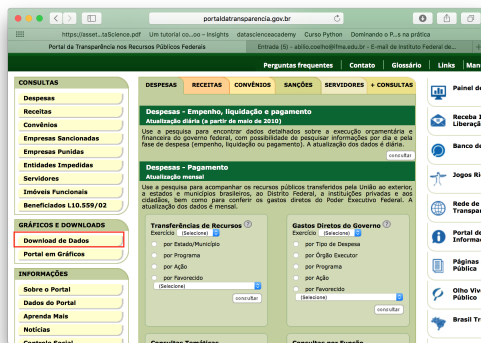
**Figura 12.6. Instalação e testes das bibliotecas necessárias (a) e (b)**

### 12.5.3. Download e importação dos dados externos

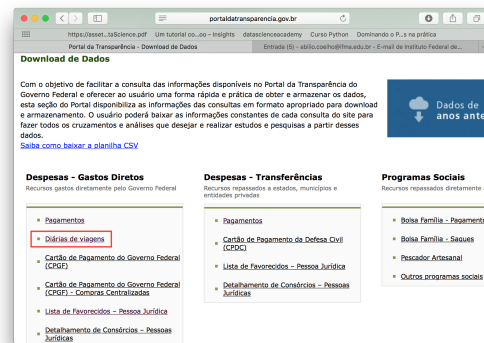
Com o objetivo de facilitar a consulta das informações disponíveis no Portal da Transparência do Governo Federal e oferecer ao usuário uma forma rápida e prática de obter e armazenar os dados, existe uma seção do Portal que disponibiliza as informações das consultas em formato apropriado para download e armazenamento. O usuário poderá baixar as informações constantes de cada consulta do site para fazer todos os cruzamentos e análises que desejar e realizar estudos e pesquisas a partir desses dados.

Os arquivos são gerados com a mesma periodicidade em que as respectivas informações são atualizadas no Portal. Assim, alguns arquivos têm atualização mensal, outros, semanal; e algumas diárias. Outra diferença entre os arquivos é que alguns estão com dados separados por ano e outros trazem a informação consolidado de um determinado período. Essa diferença também reflete a forma como as informações estão disponíveis nas consultas do Portal

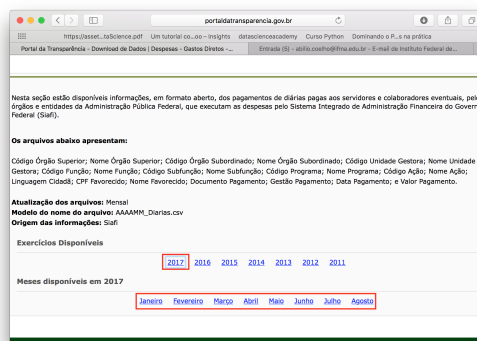
A figura 12.7 apresenta o caminho a ser percorrido para a obtenção dos dados. Pode-se identificar na imagem 12.7(a), um pequeno retângulo vermelho sobre o link download de dados, após acessar esse link o usuário será redirecionado para a página identificada pela imagem 12.7(b), como a proposta é fazer um acompanhamento das diárias, o link a ser acessado está marcado com o retângulo vermelho, porém existem vários outros tipos de acompanhamento que podem ser realizados. A imagem 12.7(c) apresenta a identificação do arquivo a ser baixado, juntamente com o formato do mesmo, mostrando todas as colunas existentes. Como é um arquivo atualizado mensalmente dispõe do ano corrente e opções por exercícios de anos anteriores.



(a) Tela inicial do Portal da Transparência



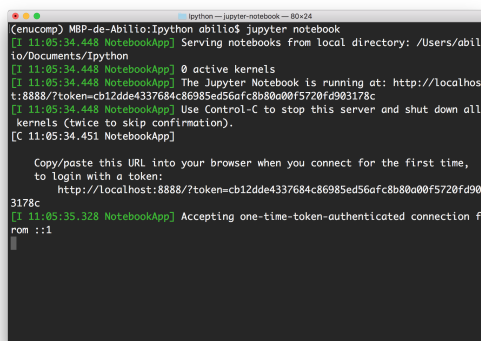
(b) Tela de downloads de dados



(c) Tela de download do arquivo de diárias

Figura 12.7. Download de dados externos (a), (b) e (c)

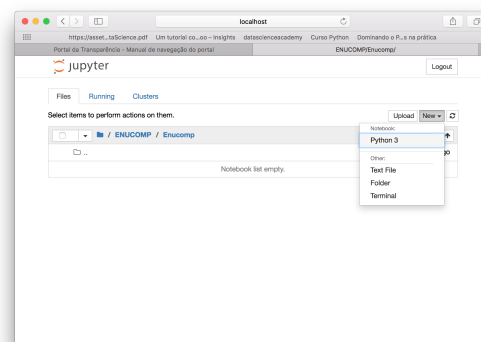
A figura 12.8 Apresenta um conjunto de três imagens referentes à importação dos dados externos. A imagem 12.8(a) mostra o comando para iniciar o ambiente *jupyter*, após esse comando o browser principal é automaticamente aberto, a tela inicial do *jupyter* pode ser observada na imagem 12.8(b). Para se criar um *notebook*, é necessário clicar no botão *new* e logo em seguida clicar na versão do interpretador Python a ser iniciado, neste caso *Python3*. A imagem 12.8(c) apresenta, de fato, a criação de um *Dataframe* chamado *jan* e utiliza a função *pd.read\_csv* para importar os dados do arquivo passado como parâmetro. Geralmente arquivos CSV são delimitados por "vírgulas", porém nesse arquivo específico do Portal da Transparência é delimitado por tabulação, por esse motivo foi passado o argumento `\t` para o parâmetro *delimiter*. O último parâmetro passado, é referente ao formato de texto que será importado, nesse caso, como os textos possuem acentos o formato recomendado é o *ISO-8859-1*, se esse argumento não fosse passado, os textos não seriam importados com acentos. Na próxima célula é possível verificar a utilização da função *columns*, essa função retorna o cabeçalho de todas as colunas do *Dataframe*.



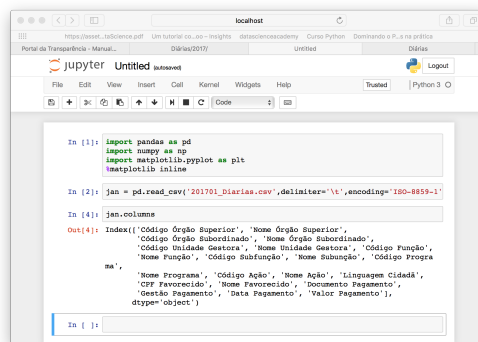
```
(enucomp) MBP-de-Abilio:ipython abilio$ jupyter notebook
[11:05:34.448 NotebookApp] Serving notebooks from local directory: /Users/abilio/Documents/ipython
[11:05:34.448 NotebookApp] 0 active kernels
[11:05:34.448 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?token=cb12dde4337684c86985ed56afc8b88a00f5720fd903178c
[11:05:34.448 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 11:05:34.451 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time, to login with a token:
http://localhost:8888/?token=cb12dde4337684c86985ed56afc8b88a00f5720fd903178c
[11:05:35.328 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

(a) Iniciar jupyter



(b) Tela inicial do jupyter



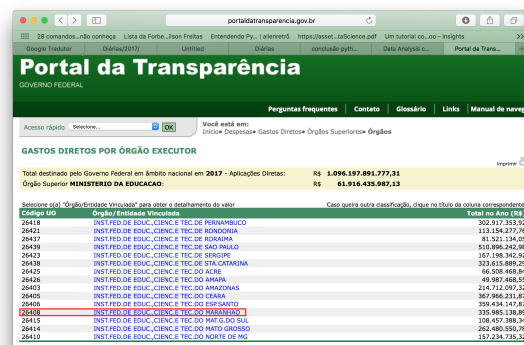
(c) Importação do arquivo de diárias

Figura 12.8. Importação de dados externos (a), (b) e (c)

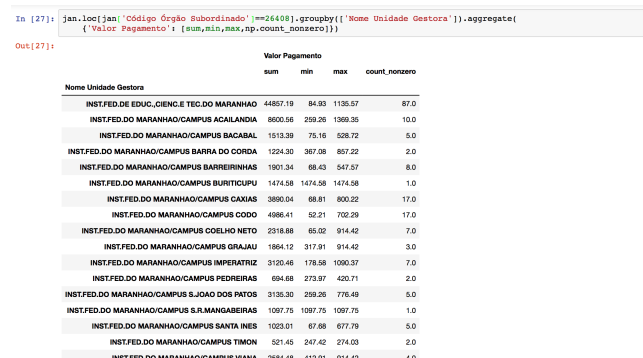
#### 12.5.4. Manipulação do Dataframe

Sabendo-se que o código do IFMA junto ao Portal da Transparência é 26408, primeiro é necessário fazer uma seleção de todos os registros que aparecem com esse código, utiliza-se a função `loc` e será retornado uma matriz booleana de todos os registros que possuem esse valor no campo Código Órgão Subordinado. Após a seleção faz-se o agrupamento, nesse caso utiliza-se apenas o campo Unidade Gestora, pois os valores de agregação serão calculados sobre cada campus. Em seguida é, de fato, realizada a agregação, utilizando a coluna Valor Pagamento, calcula-se a soma, mínimo, máximo e quantidade de diárias por campus.

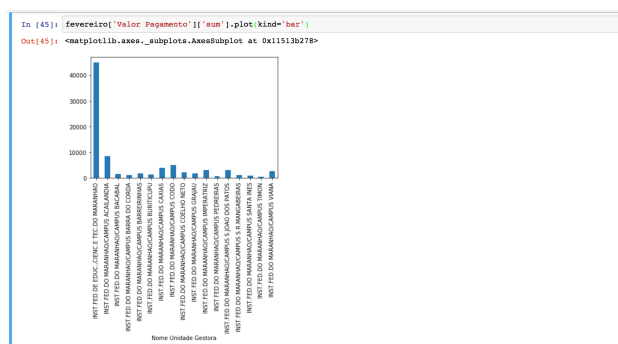
A figura 12.9 apresenta a manipulação dos dados para retirada da informação de quanto cada campus do IFMA gastou com diárias e passagens. A imagem 12.9(a) identifica qual é o código do IFMA junto ao portal da transparência. A imagem 12.9(b) apresenta o comando a ser executado para a geração da informação, onde primeiro faz-se uma seleção de todas as linhas em que o Código do Órgão Subordinado seja igual a 26408, gerando assim uma matriz booleana, que é passada como parâmetro para a função `loc`. Após isso é feito o agrupamento através da coluna Nome da Unidade Gestora, que de fato, são os executores da despesa. Para obter informações do agrupamento são realizadas algumas funções *aggregate*, onde a primeira coluna calculada é a soma de todos os gastos por



(a) Código IFMA



(b) Agrupamento de dados



(c) Geração do gráfico

Figura 12.9. Manipulação do Dataframe (a), (b) e (c)

campus, a segunda seria o valor mínimo de diária por campus, a terceira refere-se ao valor máximo que um campus recebeu e a última coluna é referente a quantidade de diárias que o campus recebeu no mês. A imagem 12.9(c) mostra o quanto é simples gerar um gráfico utilizando o matplotlib. O próprio Pandas possui uma função que utiliza os recursos da biblioteca de gráfico, ainda pode-se optar por qual tipo de gráfico utilizar através do parâmetro *kind*. Usuários mais experientes podem utilizar recursos da biblioteca *seaborn* para deixar os gráficos com aspecto mais profissional.

## 12.6. Considerações finais

Neste capítulo foram apresentados conceitos gerais sobre a linguagem Python e a biblioteca Pandas, demonstrado o quanto é simples e rápida a sua utilização. Foi apresentado um breve estudo de caso utilizando a tecnologia sobre dados abertos disponibilizados pelo governo, porém Várias outras organizações fornecem dados como: dados eleitorais, financeiros, estatísticos, etc. Muitas outras informações poderiam ter sido extraídas desse exemplo, como agrupamento por servidor ou por e funções de agregações diversas, porém como é apenas uma introdução ficará para trabalhos futuros.

## Referências

- [Becker 2015] Becker, J. L. (2015). *Estatística básica: transformando dados em informação*. Bookman Editora.
- [Brasil 2017] Brasil (2017). Portal da transparência. Disponível em: <http://www.portaldatransparencia.gov.br>. Acesso em: 18 Outubro 2017.
- [Grus 2016] Grus, J. (2016). *Data Science do Zero: Primeiras regras com Python*. Alta Books Editora.
- [Matplotlib 2017] Matplotlib (2017). Matplotlib. Disponível em: <https://matplotlib.org>. Acesso em: 18 Outubro 2017.
- [McKinney 2012] McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. "O'Reilly Media, Inc."
- [Menezes 2016] Menezes, N. N. C. (2016). *Introdução à programação com Python—2ª edição: Algoritmos e lógica de programação para iniciantes*. Novatec Editora.
- [Miniconda 2017] Miniconda (2017). Miniconda. Disponível em: <https://conda.io/docs/user-guide/install/index.html>. Acesso em: 18 Outubro 2017.
- [Oliphant 2007] Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3).
- [Pandas Library 2017] Pandas Library (2017). Python data analysis library. Disponível em: <http://pandas.pydata.org/pandas-docs/stable/>. Acesso em: 18 Outubro 2017.