Introdução ao Jupyter Notebook

Prof. Paulo Ribeiro

* Essa nota de aula é uma tradução direta de parte do tutorial *Jupyter Notebook for Beginners: A Tutorial*, disponível **aqui**.

Introdução

O Jupyter Notebook é uma ferramenta incrivelmente poderosa para desenvolver e apresentar projetos de forma interativa. Um notebook integra o código e sua saída em um único documento à visualizações, texto narrativo, equações matemáticas e outros tipos de elementos textuais. O fluxo de trabalho intuitivo promove um desenvolvimento iterativo e rápido, tornando os notebooks uma escolha cada vez mais popular em áreas como análise de dados, simulação, matemática computacional ou qualquer área cujo objetivo não seja desenvolver um *software*. Vale destacar que o Jupyter Notebook é totalmente gratuito.

O projeto Jupyter é o sucessor do IPython Notebook, e foi publicado pela primeira vez como protótipo em 2010. Embora seja possível usar muitas linguagens de programação diferentes nos Jupyter Notebooks, esta nota de aula focará no Python, ferramenta de nosso curso.

Instalação

A maneira mais fácil para um iniciante começar a usar o Jupyter Notebooks é instalar o Anaconda. O Anaconda é a distribuição Python mais amplamente utilizada para a ciência de dados e vem pré-carregada com mais de 1500 pacotes relacionando todas as bibliotecas e ferramentas mais populares. Assim como Jupyter, algumas das maiores bibliotecas de Python envolvidas no Anaconda incluem NumPy, Scipy, Pandas e Matplotlib, para citar alguns.

Essa instalação facilita o trabalho, uma vez que muito raramente haverá problemas de dependência de quaisquer módulos geralmente necessários para análise de dados e computação científica.

Para nossos propósitos, atente para:

1. baixar a versão mais recente do Anaconda for Python 3 (ignore Python 2.7);
2. instalar o Anaconda seguindo as instruções na página de download de acordo com o seu sistema operacional.

Criando o seu primeiro notebook

Considerando que a instalação foi corretamente feita, pode-se criar um novo notebook, procedendo assim:

- **no Windows** – menu iniciar → programas → anaconda3 → jupyter notebook;
- **no Linux** – abra um terminal e digite `jupyter notebook`.

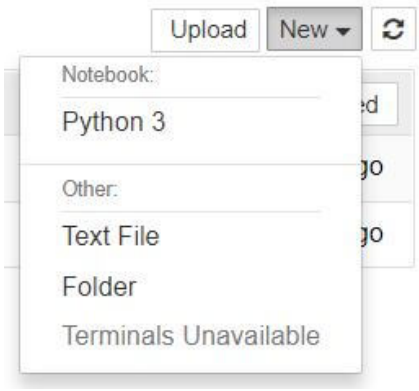
Esses comandos abrirão uma nova aba no seu navegador da Web padrão que deve ser parecida com a captura de tela a seguir

Este é o **Notebook Dashboard**, projetado especificamente para gerenciar seus notebooks Jupyter. Pense nisso como a plataforma de lançamento para explorar, editar e criar seus notebooks. Perceba que o painel lhe dará acesso apenas aos arquivos e subpastas contidos no diretório de inicialização do Jupyter; no entanto, o diretório de inicialização pode ser alterado .

Uma rápida observação pode ter levado-o a perceber que a URL do painel é algo como `http://localhost:8888/tree`. *Localhost* não é um site, mas indica que o conteúdo está sendo servido em sua máquina local: seu próprio computador. Os notebooks e o painel do Jupyter são aplicativos web, e o Jupyter inicia um servidor Python local (o anaconda instalado) para servir esses aplicativos ao seu navegador, tornando-o essencialmente independente de plataforma e abrindo as portas para um compartilhamento mais fácil na web. Em resumo, o notebook é uma interface de programação.

A interface do painel é em grande parte autoexplicativa - embora voltemos a ela brevemente mais tarde. Então, o que estamos esperando? Navegue até a pasta em que você gostaria de criar seu primeiro bloco de notas, clique no botão `Novo` no canto superior direito e selecione `Python 3` (ou a versão de sua escolha).

Seu primeiro Notebook Jupyter será aberto em uma nova aba - cada notebook usa sua própria aba, porque você pode abrir vários notebooks simultaneamente. Se você



voltar para o painel, verá o novo arquivo `Untitled.ipynb` e verá um texto em verde informando que seu bloco está em execução.

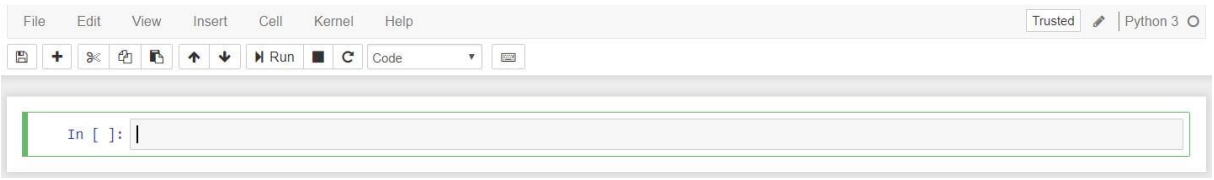
Cada arquivo `.ipynb` é um arquivo de texto que descreve o conteúdo do seu notebook em um formato chamado JSON, como mostra a figura abaixo. Cada célula e seu conteúdo, incluindo anexos de imagem que foram convertidos em strings de texto, são listados junto com alguns metadados. Você pode editar isso sozinho - se você sabe o que está fazendo! - selecionando "*Edit* → *Edit Notebook Metadata*" na barra de menu do notebook.


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "!pip install jupyterlab_latex"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.6.5"
  }
```


Se você der dois clicks no nome ao lado do logo do Jupyter Notebook, pode renomear o notebook recém criado. Ao renomeá-lo, também o fará no arquivo `.ipynb` salvo.

A interface do notebook

Ao criar um novo notebook, a interface visualizada é, basicamente, a mostrada a seguir.



Há dois termos bastante importantes que você deve notar, que provavelmente são novos para você: as células e os kernels são fundamentais para entender o Jupyter e o que o torna mais do que apenas um processador de texto. Felizmente, esses conceitos não são difíceis de entender:

- um kernel é um "mecanismo computacional" que executa o código contido em um documento do notebook;
- uma célula é um recipiente para o texto a ser exibido no bloco de notas ou código a ser executado pelo kernel do notebook.

Kernels

Você deve ter notado que o Jupyter lhe dá a opção de mudar o kernel, e na verdade existem muitas opções diferentes para escolher. Quando você criou um novo bloco de notas a partir do painel, selecionando uma versão do Python, você estava realmente escolhendo qual kernel usar.

Não só existem kernels para diferentes versões do Python, mas também para mais de 100 linguagens, incluindo Java, C e até Fortran. Cientistas de dados podem estar particularmente interessados nos kernels para R e Julia, bem como no imatlab e no Calysto MATLAB Kernel for Matlab. O kernel do SoS fornece suporte a vários idiomas em um único notebook. Cada kernel tem suas próprias instruções de instalação, mas provavelmente exigirá que você execute alguns comandos em seu computador.

Não nos deteremos em mostrar outros kernels, uma vez que o Python é nossa ferramenta de estudo.

Células

Células formam o corpo de um notebook. Na captura de tela de um novo notebook na seção acima, essa caixa com o contorno verde é uma célula vazia. Existem dois tipos de células principais que abordaremos:

- **célula de código** contém código a ser executado no kernel e exibe sua saída abaixo;
- **célula de texto**, contém texto formatado usando Markdown e exibe sua saída no local quando é executada.

A primeira célula de um novo notebook é sempre uma célula de código. Vamos testá-lo com um exemplo clássico de hello world. Digite `print('Hello World!')` na célula e clique no botão de execução (`RUN`) na barra de ferramentas ou pressione `Ctrl + Enter`. O resultado deve ficar assim:

```
In [1]: print('Hello World!')
        Hello World!
```

Quando você executou a célula, sua saída será exibida abaixo e o rótulo à sua esquerda será alterado de `In [ ]` para `In [1]`. A saída de uma célula de código também faz parte do documento, e é por isso que você pode vê-lo neste artigo. Você sempre pode identificar a diferença entre as células de código e Markdown, porque as células de código têm esse rótulo à esquerda e as células de Markdown não. A parte `In` do rótulo é simplesmente abreviação de "*Input*", enquanto o número do rótulo indica quando a célula foi executada no kernel - nesse caso, a célula foi executada primeiro. Execute a célula novamente e o rótulo mudará para `In [2]` porque agora a célula foi a segunda a ser executada no kernel.

O importante aqui é perceber que a numeração mostrada é referente a ordem de execução dos comandos no kernel, não de posição. Por exemplo, podemos ter duas linhas sequenciais, cuja numeração não corresponde a ordem de posição, como mostrado abaixo

```
In [3]: print('Hello World!')
        Hello World!

In [2]: print('Hello World de novo!')
        Hello World de novo!
```

Nesse caso, a segunda célula foi executada antes da primeira, como mostra a numeração a esquerda.

Na barra de menus, clique em `Insert` e selecione *Insert cell above* para criar uma nova célula de código abaixo da primeira e experimentar o código a seguir para ver o que acontece. Alternativamente, você pode executar a célula usando `Alt+Enter`, que executa a célula atual e cria uma nova logo abaixo.

Considere agora o seguinte código

```
In [*]: import time

        time.sleep(10)
```

Essa célula não produz saída, mas leva 10 segundos para ser executada. Observe como o Jupyter mostra que a célula está em execução alterando seu rótulo para `In [*]`.

Ou seja, enquanto tivermos um asterisco no lugar de um número no rótulo da célula de código, significa que esta célula está em execução. Vale salientar que as células seguintes só serão executas após a atual terminar o processamento. Ainda estamos falando de uma linguagem interpretada!

Caso deseje parar essa execução, pode interromper o kernel indo em `kernel` → `Interrupt` ou, no menu de ferramentas, clicando no quadrado.

Ainda sobre saídas, é importante percebermos que, nos exemplos executados até agora, usamos basicamente a função `print( )` para gerar uma saída.

Porém, para visualizar resultados de operações ou saídas de funções, não é necessário invocar essa função. Veja por exemplo, a figura abaixo

```
In [1]: a = 1
        b = 2
        print(a+b)

        3

In [2]: a = 1
        b = 2
        a+b

Out[2]: 3
```

Quando usamos `print`, a saída aparece diretamente. Quando não usamos, como no caso da segunda célula, a saída recebe um rótulo, `Out[ ]:`, com a mesma numeração do rótulo da célula. Em outras palavras, não há obrigação de usar a função `print` para visualizar os resultados de operações.

Principais atalhos

Uma última coisa que você pode ter observado ao executar suas células é que a borda delas ficou azul, enquanto era verde enquanto você estava editando. Há sempre uma célula *ativa* realçada com uma borda cuja cor indica seu modo atual, em que verde significa *modo de edição* e azul é *modo de comando*.

Até agora, vimos como executar uma célula com `Ctrl + Enter`, mas há muito mais. Os atalhos de teclado são um aspecto muito popular do ambiente Jupyter porque facilitam um fluxo de trabalho rápido baseado em células. Muitas delas são ações que você pode executar na célula ativa quando está no modo de comando (azul).

Abaixo, é mostrada uma lista de alguns atalhos de teclado do Jupyter. Você não deve buscá-las imediatamente, mas a lista deve dar uma boa ideia do que é possível. Caso queira ver todos os atalhos, você pode ir em *Help* → *Keyboard shortcuts* ou usar o atalho `Ctrl+Shift+p`.

- Alterne entre o modo de edição e comando com `Esc` e `Enter`, respectivamente;
- Uma vez no modo de comando:
  - Role para cima e para baixo nas células com as teclas de deslocamento;
  - Pressione `A` ou `B` para inserir uma nova célula acima ou abaixo da célula ativa;
  - `M` transformará a célula ativa em uma célula de texto;
  - `Y` irá definir a célula ativa para uma célula de código;
  - `D + D` (`D` duas vezes) apagará a célula ativa;
  - `C` copiará a célula ativa;
  - `X` recortará a célula ativa;
  - `V` colará a célula copiada logo abaixo da célula ativa;
  - `Z` irá desfazer ações sobre a célula;
  - Segure a `Shift` e pressione `Up` ou para `Down` para selecionar várias células de uma só vez;
  - Com várias células selecionadas, `Shift + M` irá mesclar sua seleção;
  - `Ctrl + Shift + -`, no modo de edição, dividirá a célula ativa no cursor.

Markdown

O Markdown é uma linguagem de marcação leve e fácil de aprender para formatar texto simples. Sua sintaxe tem uma correspondência de um-para-um com *tags* HTML, portanto, algum conhecimento anterior aqui seria útil, mas definitivamente não é um pré-requisito.

Um excelente tutorial para usar a linguagem Markdown pode ser encontrado nesse link, que mostra, inclusive, a correspondência dos comandos com o html básico.