Sumário

1  Introdução

2  Get Jupyter Notebook filename

3  Funções

3.1  Variáveis em Markdowns

3.2  Linhas de Tabelas

3.3  Comandos do Sistema

3.4  HTML

4  Export

5  GitHub

6  Requirements

7  Erros

8  Referêcias

# Introdução

O *Jupyter Notebook* é a maneira que optei para escrever os códigos na linguagem *Python*, visto que além de rodar os códigos, é possível: 1. Documentar os *scripts*, escrevendo o singnificado e objetivo de cada conjunto de comandos; 2. Atualizar os meus repositórios na plataforma **GitHub**; 3. Trabalhar com uma diversidade de opções de exportação do arquivo em formatos diversos, adaptados até mesmo para as simples leitura, como PDFs e Markdowns.

É no procesos de exportação dos arquivos que eu me ative nessa publicação, pois um dos objetivos de longo prazo que busco é exportar relatórios padronizados, para distribuição geral e irrestrita, ou seja, quero algo que não seja inteligível apenas por pessoas que conhecem de programação.

Para isso foram aqui apresentados um diversidade de opções para exportação de um arquivo *.ipynb*, sendo possível: - Incluir apenas campos determinados; - Incluir apenas as células que tenham determinada *tag*; - Incluir apenas as células de *markdown*; - Excluir as células de *outputs*.

# Get *Jupyter Notebook* filename

Testei diversos comandos para obter o nome do *Jupyter Notebook* em uma variável. A melhor opçãoque encontrei estava nesse [*post*](https://stackoverflow.com/questions/12544056/how-do-i-get-the-current-ipython-jupyter-notebook-name) que tem diversas outras opções.

# Funções

## Variáveis em *Markdowns*

Para inserir uma variável em uma célula markdow para eu inserir a variável entre colchetes duplos, por exemplo { {a} }. Logo, se eu alterar o valor de a para qualquer um terei que **a={{a}}**.

O mesmo pode ser feito com tabelas. Em tentativa de inserir tabelas diretamente do Pandas não obtive sucesso… Depois temos dataframe modificado pelo *.to\_html()*, [função](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_html.html) que fornece várias opções a serem exploradas. {{df\_html}}

## Linhas de Tabelas

Descobri que [nesse *post*](https://stackoverflow.com/questions/38783027/jupyter-notebook-display-two-pandas-tables-side-by-side) que é possível trabalhar para inserir também mais de uma tabela alinhada.

## Comandos do Sistema

Praticamente qualquer comando do sistema pode ser acessado usando previamente **!**, o qual passa qualquer comando subsequente diretamente para o sistema operacional. Você pode até usar variáveis python em comandos enviados para o sistema operacional!

## HTML

# *Export*

Os arquivos Jupyter Notebook podem ser exportados em diversos formatos, seja através do menu de opções, ou através dos comandos. Ao exportar, é possível definir diversas opções que limitam o que será exportado, podendo escolher determinados tipos de células ou, até mesmo, células invidivuais.

No *post* [Jupyter Notebook nbconvert without Magic Commands/ w/o Markdown](https://stackoverflow.com/questions/57701538/jupyter-notebook-nbconvert-without-magic-commands-w-o-markdown) é apresentado algumas opções de exportação. Incorporei várias delas no script *../codes/files/export\_jupyter.py*. Ainda existem outras opções que não estudei a finalidade, listadas a seguir:

1. *–stdout*
2. *–TemplateExporter.exclude\_input\_prompt=True*
3. *–TagRemovePreprocessor.remove\_input\_tags = {“hide”}*

Usando pandoc descobri que dá pra exportar para **.doc**! Não ficou tão bom, mas ajuda!

# GitHub

A partir do *post* [**How to Git Jupyter Notebooks the Right Way**](http://mateos.io/blog/jupyter-notebook-in-git), compreendi que é considerada como *best pratices* no git de projetos escritos em *Jupyter Notebook* a aplicação de um determinado código usando o package *nbstripout*, conforme apresentado abaixo. No vídeo [**nbstripout: strip output from Jupyter and IPython notebooks**](https://www.youtube.com/watch?v=BEMP4xacrVc) é explicado detalhadamente como o comando atua.

Criei uma função para exportar o *Jupyter Notebook* em diversos formatos. Aproveitei para incorporar o comando do nbstripout na função que faz o *commit*, visando simplificar as coisas.

# *Requirements*

O comando pip freeze é o mais difundido na internet para se obter os *requirements.txt*, ou seja, o arquivo com o qual é possível indicar quais os *packages* necessários para rodar um determinado *script*.

Tentei usar também o package pipreqs, porém ele não funciona em *Juptyter Notebook*. Descobri ainda que o comando conda env export > environment.yml pode auxiliar na criação destes parâmetros.

# Erros

Em uma tentativa de exportar o *Jupyter Notebook* para PDF tive problemas. O arquivo não era exportado e apresentava a seguinte mensagem de erro: - *nbconvert failed: xelatex not found on PATH, if you have not installed xelatex you may need to do so. Find further instructions at https://nbconvert.readthedocs.io/en/latest/install.html#installing-tex.*

Para solucionar, descobri que é necessário instalar, no Linux, akguns pacotes de aplicativos com os seguintes comandos, sendo o primeiro uma instalação mais compacta e o segundo uma instalação completa.

sudo apt-get install texlive-xetex texlive-fonts-recommended texlive-generic-recommended

sudo apt-get install texlive-full

# Referêcias

Há muita informação na internet sobre funcionalidades do *Jupyter Notebook*. Apenas para exemplificar, usei particialmente algumas das funções e truques apresentados em [**Jupyter Notebook Extensions**](https://towardsdatascience.com/jupyter-notebook-extensions-517fa69d2231) e [**28 Jupyter Notebook Tips, Tricks, and Shortcuts**](https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts).