

jupyter

June 11, 2020

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 Exportando o Jupyter Notebook
- 5 GitHub
- 6 Requirements
- 7 Erros
- 8 Referências

1 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 significado 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 processo 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 uma 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*.

```
import os
import time
import pandas as pd
```

```
os.makedirs('docs', exist_ok=True)
```

2 Get *Jupyter Notebook* filename

Testei diversos comandos para obter o nome do *Jupyter Notebook* em uma variável. A melhor opção que encontrei estava nesse [post](#) que tem diversas outras opções.

```
%%javascript

var kernel = IPython.notebook.kernel;
var body = document.body, attribs = body.attributes;
var command = 'ipynb_filename = ' + '"' + attribs['data-notebook-name'].value +
  '↵"';
kernel.execute(command);
```

```
ipynb_filename
```

```
%%javascript

var kernel = IPython.notebook.kernel;
var nb = IPython.notebook;
var command = 'ipynb_pathname = ' + '"' + nb.base_url + nb.notebook_path + '"';
kernel.execute(command);
```

```
ipynb_pathname
```

```
#!/run '../codes/files/get_jupyternotebook_name.py'
#name = get_jupyternotebook_name()
#name
```

3 Funções

3.1 Variáveis em *Markdowns*

Para inserir uma variável em uma célula markdow para eu inserir a variável entre colchetes duplos, por exemplo 10. Logo, se eu alterar o valor de a para qualquer um terei que **a=10**.

```
a = 10
```

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](#) que fornece várias opções a serem exploradas.

A

```
<th>B</th>    </tr>  </thead>  <tbody>    <tr>          <td>1.0000</td>    <td>4.12134</td>
```

```
df = pd.DataFrame({"A": [1.0, 2.2, 3.6666], "B": [4.12134, 5.674, 6.13215]})
```

```
df_html = df.to_html(index=False, decimal='.', notebook=True, justify='center')  
df_html = df_html.replace('\n', '')  
df_html
```

3.2 Linhas de Tabelas

Descobri que [nesse post](#) que é possível trabalhar para inserir também mais de uma tabela alinhada.

```
import pandas as pd  
import numpy as np  
from IPython.display import display, HTML  
  
CSS = """  
.output {  
    flex-direction: row;  
}  
"""  
  
HTML('<style>{}</style>'.format(CSS))
```

```
display(df)  
display(df)  
display(df)
```

3.3 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!

```
!ls
```

```
file_type = 'ipynb'  
  
!ls ./*$file_type
```

3.4 HTML

```
%%HTML  
This is <em>really</em> neat!
```

4 Exportando o *Jupyter Notebook*

O arquivo *.ipynb* pode ser exportado em formatos diversos. Abaixo carrego uma função que escrevi para facilitar o processo de exportação do arquivo em diferentes locais do PC para, posteriormente, atualizar os repositórios contidos no *GitHub*.

Com as funções para exportar o *Jupyter Notebook* e para obter o nome do arquivo *.ipynb* carregadas, basta exportar o arquivo, inicialmente para a pasta *docs* dentro do projeto e também, visando atualizar os *posts* do site, para a respectiva pasta.

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 individuais.

No *post* [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"}`

```
# Parâmetros
today = date.today().strftime('%Y-%m-%d')
today = '2020-06-11'

# Input/ Output
inp = 'jupyter.ipynb'
#out = os.path.join('docs', today + '-' + inp.split('.')[0])
out = os.path.join('docs', inp.split('.')[0])

# Exportar para
ext = 'pdf'                                # 'html', 'markdown', 'latex', 'pdf',
↳ 'python'

# Remover células taggeadas com:
tag = 'remove_cell'
```

```
!jupyter-nbconvert $inp\
--to $ext\
--TagRemovePreprocessor.enabled=True\
--TagRemovePreprocessor.remove_cell_tags='["$tag"]'\
--ClearOutputPreprocessor.enabled=True\
--TemplateExporter.exclude_markdown=False\
--TemplateExporter.exclude_code_cell=False\
--TemplateExporter.exclude_output=True\
--TemplateExporter.exclude_raw=False\
```

```
--TemplateExporter.exclude_input_prompt=True\  
--output $out
```

```
inp = 'jupyter.ipynb'  
out = os.path.join('docs', 'Arquivo')  
tag = "['remove_cell']"
```

-to pdf: markdown

```
!jupyter-nbconvert $inp\  
--to pdf\  
--TagRemovePreprocessor.enabled=True\  
--ClearOutputPreprocessor.enabled=True\  
--TemplateExporter.exclude_markdown=False\  
--TemplateExporter.exclude_code_cell=True\  
--TemplateExporter.exclude_output=True\  
--TemplateExporter.exclude_raw=False\  
--TemplateExporter.exclude_input_prompt=True\  
--TagRemovePreprocessor.remove_cell_tags='["remove_cell"]'\  
--output $out
```

```
out = os.path.join('docs', 'jupyter')  
tag = "['remove_cell']"  
  
os.system('jupyter-nbconvert jupyter.ipynb\  
    --to pdf\  
    --TagRemovePreprocessor.enabled=True\  
    --ClearOutputPreprocessor.enabled=True\  
    --TemplateExporter.exclude_markdown=False\  
    --TemplateExporter.exclude_code_cell=True\  
    --TemplateExporter.exclude_output=True\  
    --TemplateExporter.exclude_raw=False\  
    --TemplateExporter.exclude_input_prompt=True\  
    --TagRemovePreprocessor.remove_cell_tags="'+ tag +'\  
    --output '+ out)
```

```
%run '../codes/files/export_jupyter.py'
```

```
export_jupyter(ipynb_filename, 'docs', ['html', 'markdown', 'pdf', 'python'],  
↪False)
```

```
import nbformat
import nbconvert
from nbconvert import HTMLExporter, PythonExporter
from nbconvert.writers import FilesWriter
from pathlib import Path
```

```
#pip install notebooktoall
#https://github.com/notebooktoall/notebooktoall/blob/master/notebooktoall/
↳ transform.py
```

```
from notebooktoall.transform import transform_notebook

transform_notebook(ipynb_file=ipynb_filename, export_list=['html', 'py'])
```

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

```
file_wo_ext = ipynb_filename.split('.')[0]
file_md = file_wo_ext+'.md'
file_doc = file_wo_ext+'.docx'

file_doc = os.path.join('docs', file_doc)
file_md = os.path.join('docs', file_md)
```

```
!pandoc -o $file_doc -f markdown -t docx $file_md --reference-links
```

5 GitHub

A partir do *post* [How to Git Jupyter Notebooks the Right Way](#), compreendi que é considerada como *best practices* 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](#) é explicado detalhadamente como o comando atua.

```
!nbstripout --install --attributes .gitattributes
```

```
#os.system('nbstripout --install --attributes .gitattributes')
```

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.

```
%run '../codes/git/update_github.py'

git_full('.', '.', 'Atualizando')
```

Após as exportações dos arquivos nos formatos necessários, basta atualizar o repositório diretamente pelo *Jupyter Notebook*. Abaixo é atualizado o repositório desse projeto específico, bem como a

derivação desse projeto no *site*.

6 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*.

```
pip freeze > requirements.txt
```

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

7 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, alguns 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
```

8 Referências

Há muita informação na internet sobre funcionalidades do *Jupyter Notebook*. Apenas para exemplificar, usei parcialmente algumas das funções e truques apresentados em [Jupyter Notebook Extensions](#) e [28 Jupyter Notebook Tips, Tricks, and Shortcuts](#).