Automatic pdf report from a python script with pandoc

ChristianV

14 Oct 2022

Abstract

Export a pdf report (code, outputs) commented with the Markdown embedded in the python script, a bash preprocessing and pandoc. Everything is launched whithin the python script.

Contents

1 Which problem does this solve?

To have the possibility to make a report linked to a python script is an interesting function. It allows to keep notes about the script (why, how), to show and share the results. The best location of the report is inside the script itself. A pdf file as documentation has in addition a confortable readability. Tools like Pweave offer this functionalities, but my knowledge is inadequate to solve the problems that may occur... or occurred like the last time I updated my Manjaro laptop...

So I came to this solution, probably with a lower performance but that uses standard tools, explicit rather simple steps.

2 How it works?

2.1 The features

The expected features are :

- To export a pdf file that includes :
  - The python code cut in chunks
  - All the needed text to explain the goal and outputs of the script
  - Math and formulas

    - The outputs of the script, to the console or graphs (matplotlib)
- The script shall remain directly executable with no modifications
- The only maintained file is the script

2.2 The proposed solution

The script shall include :

- Some codes to store the outputs
- Lines of markdown starting by a mark
- Some lines to include external text files

The mark is a unique sequence of characters starting by "#" (comment in python)

Then the script is preprocessed to :

- remove the 2 first lines (python3 and coding specification)
- remove the mark
- insert the external files
- store the result into an intermediate markdown file

The markdown file is then transformed in pdf with pandoc

2.3 The sources of this solution

The idea of a mark "#%%" comes from pweave, spyder to embedded markdown in the python script.

The first test shown that with, at least, a French keyboard this sequence is hard to use (need AltGr then shift) and is visually intrusive in editor. Several "#" are not allowed (used by markdown) so I changed to "#~~" as "~" is next key on the left of "#".

The script "filename.py" is copied into an image markdown file "filename.md"

A call to "sed" can remove the 2 first lines and the mark.

```
sed -i '1,2d' filename.md
sed -i -e 's/abc/XYZ/g' filename.md
```

A "magic" call to perl can insert external text files. It is proposed by stackoverflow ie Embedding one markdown document in another

```
#!/usr/bin/env bash
perl -ne 's/^!\[\[(.+?)\]\].*/`cat $1`/e;print' source.md > result.md
```

All the found ![[textfilepath]] in source.md will be replaced by the textfilepath file content in result.md

This method can be applied to subpart of the file like the header to get more compact script or any other text file to share like a bash file.

Finally the markdown image is exported to pdf with pandoc

```
pandoc -s -o filename.pdf filename.md
```

There are probably more interesting options of pandoc to use... This will be another task

-[] TODO explore pandoc to pdf options.

2.4 What is missing for a full magic

Some code lines are added to prepare the structure : a directory "py2pdf_files" is created to store the output files.

A local print function allows to print in the console and in a logfile. The logfile name is changed when necessary to separate the output. Inspiration for this comes from stackoverflow print on console and text file simultaneously python

Some code is also needed to store the desired matplotlib graphs.

2.4.1 Resources

- pandoc : pandoc.org
- python, bash, perl : standard of Linux distribution?

3 python code

3.1 Import modules

```
# import
import os # for py2pdf
import matplotlib.pyplot as plt # for the demo
import numpy as np # for the demo
```

3.2 Functions

This is a demo file. In other applications, for a better readability, those specific functions might be included in a library to import in the above section.

```
def print_twice(log,*args,**kwargs):
    # the double print function (to console and to log)
    print(*args,**kwargs)
    with open(log,"a") as f:  # appends to file and closes it when finished
        print(file=f,*args,**kwargs)

def outputdir():
    # create a directory to save outputs
    directory = "py2pdf_files"
    if not os.path.exists(directory):
        # print("create directory")
        os.makedirs(directory)
    else:
        pass
        # print("existing directory")

def clearlog(log):
    # clear logfile
    file_to_delete = open(log,'w')
    file_to_delete.close()
```

3.3 Main code

3.3.1 Code section 1

```
# lines to be included for py2pdf export
scriptname = os.path.basename(__file__).split('.')[0] # get this script file name without extension
logfile = "./py2pdf_files/log1.txt" # define the logfile
outputdir() # create the directory to store the output
clearlog(logfile) # clear the logfile (in case script is ran several times)

# code for console output demo
path = os.getcwd() # get the path of the current directory
print_twice(logfile, "Path of the current directory : " + path)
print_twice(logfile, "python script name : ",scriptname)
print_twice(logfile, "Directory content :")
print_twice(logfile, "\n".join(os.listdir(path)))
print_twice(logfile, "Logfile : ", logfile)
```

3.3.2 Code Output (section 1)

```
Path of the current directory : /home/christian/pweave
```

```
python script name :  scriptpython2pdf
Directory content :
final.md
py2md.sh
parseMd.sh
DMLmaterial4.py
scriptpython2pdf.py
scriptpython2pdf.pdf
main.md
scriptpython2pdf
py2pdf_files
Logfile :  ./py2pdf_files/log1.txt
```

3.3.3 Code section 2

From matplotlib site : Simple plot

```
# code for matplotlib output demo
# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)
fig, ax = plt.subplots()
ax.plot(t, s)
ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()
fig.savefig("./py2pdf_files/simple_plot.png") # this is the key line to store the matplotlib output
plt.show()
```

3.3.4 Code Output (section 2)

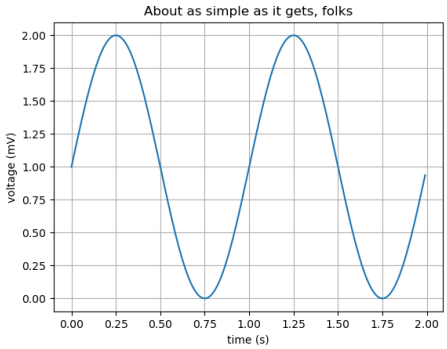

Figure 1: matplotlib output

3.3.5 Code last section

```
# lines to be included for py2pdf export
# create a second logfile
logfile = "./py2pdf_files/loglast.txt" # define the logfile
clearlog(logfile) # clear the logfile (in case script is ran several times)
# prepare the command to launch the report creation
cmd = "./py2md.sh " + scriptname

# code for console output demo
print_twice(logfile, "launch command : ",cmd) # for demo
print_twice(logfile, "pdf should be available soon") # for demo
# launch the report creation
os.system(cmd) # to launch the bash file (copy, preprocessing, pandoc to pdf)
```

/!\WARNING : no output after this line can be recorded automatically

```
print("Hurra the pdf is created!")
```

3.3.6 Code output (last section)

```
launch command :  ./py2md.sh scriptpython2pdf
pdf should be available soon
```

3.3.7 Code of the bash file

```
#!/bin/bash
ext1=".py"
ext2=".md"
ext3=".pdf"
scriptfile=$1$ext1
mdcopy=$1$ext2
pdfout=$1$ext3
cp $scriptfile temp.md
sed -i '1,2d' temp.md
sed -i -e 's/#~~ //g' temp.md
perl -ne 's/^!\[\[(.+?)\]\].*/`cat $1`/e;print' temp.md > $mdcopy
pandoc -s -o $pdfout $mdcopy
rm temp.md
rm $mdcopy
```

3.4 Additional output

Some pictures or snapshots might be added. It is a manual operation to store them.

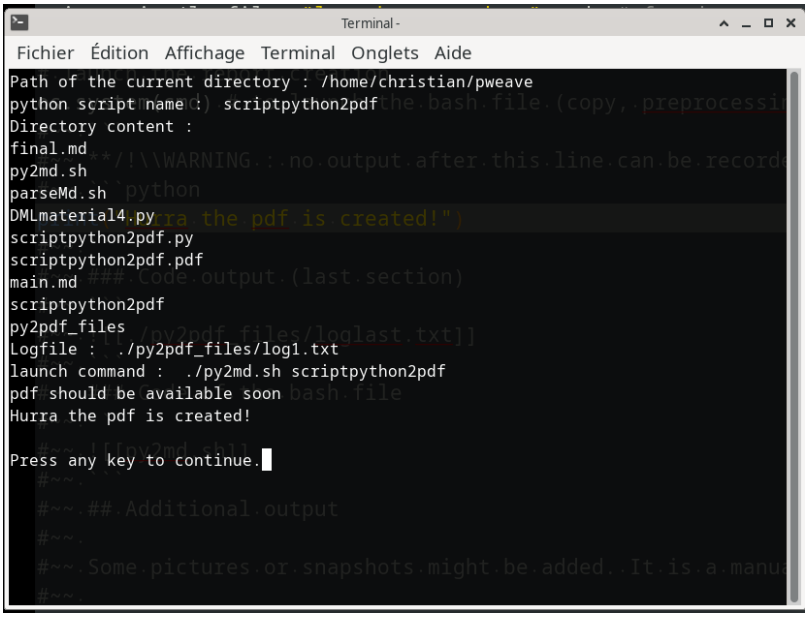

Figure 2: terminal

End of the python script