PSViewer Documentation

Release 1.0

Duncan Forel

Jun 09, 2016

CONTENTS

CHAPTER

ONE

INTRODUCTION TO PSVIEWER

PSViewer is an Python module designed to enhance a Jupyter notebook. It creates multiple magics to deal with the PostScript and Latex code directly in the cells (including files written in these languages).

1.1 Installation

1.1.1 Dependencies

This module use multiple external tools to convert the files to the correct format.

As the compilation of the code/file is not mandatory, the installation of the following dependencies isn't required if you project to use an already generated notebook with the associated images.

The dependencies are :

- ghostScript in order to read the PostScript files (download from the ghostScript website);
- ImageMagick in order to convert the files (download from the ImageMagick website);
- latex/dvips if you project to use the latex macros of the module (the distribution of your choice);

1.1.2 Configuration of ImageMagick on Windows

ImageMagick provides a lot of utility functions accessible by command-line invocation. However, some of them override native functionality of the OS (notably convert on Windows). In order to keep the system clear, it will be needed to rename this function.

For that, go to the installation folder of ImageMagick (usually C:\Programmes\ImageMagick-[version]) and rename the executable convert.exe by convert_img.exe. If another program needs the convert.exe, copy the file and rename the copy as convert_img.exe.

1.1.3 Loading the module

The module can be loaded in a document with the magic

```
%load_ext PSViewer
```

More details on how to load a module in a notebook can be found here .

1.2 Optional compilation and PNG

1.2.1 Compiling or loading ?

Every magic of this module (excepting %PSViewer) allow to provide a path to a PNG file as argument. The default behaviour is to check for the existence of PNG images at this emplacement using the method described below and display this image without executing the magic.

In the case of forced compilation (or if no file is found), the magic will be executed and the PNG image(s) resulting of the compilation will be saved at the given path, then displayed.

1.2.2 Format of the search for PNG files

A source of unexpected error is the way PSViewer searches for images. Multiple magics allow to specify a path which will be used to scan for PNG file and if found, display this/these files.

The precise path to a file will display the corresponding image, but if no image is found the module will scan the folder for a file in the format path-x.png (where x is an integer and path is the name of the file without extension) and display all the successive files in crescent value of x. The search starts with a x value of 0 and ends when no file of the searched format is found. Take care, files in the format path-00.png are not checked.

For example, if you give a path like myImage.png, but the folder contains only myImage-0.png, myImage-1.png and myImage-3.png, the program will display myImage-0.png and myImage-1.png.

It allows to write only one path to retrieve multiple PNG, but can induce unexpected behaviour when the page's number of a compiled document decrease (as the images from the previous compilation aren't deleted). In the same way, going from single-page document to multi-page will cause an apparent error, even if the compilation is correct.

CHAPTER

TWO

MAGICS ADDED BY PSVIEWER

PSViewer adds multiple line and cell magics to your notebook. They are a total of 5, one of them is only for configuration and utility.

2.1 %PSViewer

This is the utility/configuration magic. It contains multiple options.

It mainly contains utility functionalities in order to modify slightly the behaviour of the module or retrieve some informations. The magic is expected to be followed by an option from:

-reset : delete the files created during the compilations, expect if they have been stored in a specified path.

-path : display the path to the folder where the intermediate files are created.

-forceCompile : toggle the forced compilation of the code/files given.

-compilationState : display if the compilation is forced or not.

Any other option (even an empty line) will be interpreted as -help and will display the help.

Form of the magic : %PSViewer [option]

2.2 %PSFile

The expected arguments are the path to the PS file to display and optionally the path of the PNG file corresponding (if not found or compilation forced, it will be the converted file's path).

In case of incorrect use, this function displays an error message.

Form of the magic : %PSFile path [pngPath]

2.3 %LatexFile

The expected arguments are the path to the Latex file to display and optionally the path of the PNG file corresponding (if not found or compilation forced, it will be the converted file's path).

In case of incorrect use, this function displays an error message.

Form of the magic : %LatexFile path [pngPath]

2.4 %%PSCode

The line argument contains the options of the magic. If a path is specified as first option, the magic will try to retrieve PNG images from this emplacement and display them without any compilation. If the compilation is forced or if no file are found, the following options will be evaluated and the conversion of the cell's code will be execute.

The current options are :

> -noSave which allow the override of the intermediate files. Caution : this can cause unexpected behaviour in some cases.

Any other command will be ignored.

Form of the magic : %%PSCode [pngPath] [option]*

The code should be the content of the cell, starting the line after the magic.

2.5 %%LatexCode

The line argument contains the options of the magic. If a path is specified as first option, the magic will try to retrieve PNG images from this emplacement and display them without any compilation. If the compilation is forced or if no file are found, the following options will be evaluated and the conversion of the cell's code will be execute.

The current options are :

> -noSave which allow the override of the intermediate files. Caution : this can cause unexpected behaviour in some cases.
>
> -noImport generates the documentclass, begin{document} and end{document} of the file automatically.
>
> -doubleCompilation indicates that a double compilation is required. This option is useful in case of internal links.

Any other command will be ignored.

Form of the magic : %%LatexCode [pngPath] [option]*

The code should be the content of the cell, starting the line after the magic.

CHAPTER

THREE

NON-MAGICS FUNCTIONS

This section doesn't contain information about how to use the module. The purpose is to describe the methods actually present but not directly registered as magics.

3.1 Low-level functions

3.1.1 clearVersion

clearVersion()

Utility function used to clear the documents created by the compilation. It retrieves the current index for the compilation and remove all the previous folders. If the value stored for the index is incorrect, nothing is done. It also removes the file containing the current index.

The default name of the file containing the index is "version.txt".

3.1.2 loadVersion

loadVersion()

Utility function loading the current index for the folder storing a compiled document.

If the document storing the index is missing or the value contained is incorrect (typically, if it is a string) the index will be set to 0. Then, the index incremented is stored in the version file and the folder corresponding to the index is created.

The default name of the file containing the index is "version.txt".

3.1.3 writeTeXFile

writeTeXFile(*data*, *version=0*)

Write the *data* in a TeX file corresponding to the folder of index *version*.

Keyword arguments:

- data – the string containing the text to write in the document.
- version – the index of the folder to store the file in (default 0)

3.1.4 writePSFile

writePSFile(*data*, *version=0*)

Almost the same function than writeTeXFile, only the extension of the written file changes to *.ps*.

Keyword arguments:

- data – the string containing the text to write in the document.
- version – the index of the folder to store the file in (default 0)

3.1.5 convertLaTeXFile

convertLaTeXFile(*path*, *doubleCompilation=False*)

Convert the TeX file at the specified path to a PS file, using the latex and dvips commands. If an error is encountered during the conversion, the process is stopped and an error message is displayed. This function also clears the .log, .aux and .dvi files created in the process. The double compilation is supported if needed.

Caution : this error message uses IPython functionalities.

The process return the path to the PS file if the compilation was successful, and None otherwise.

Keyword arguments:

- path – the path of the TeX file.
- doubleCompilation – boolean containing if a double compilation should be done (default False)

3.1.6 convertPSFile

convertPSFile(*path*, *pngPath=None*)

Convert the PS file at the specified path to one or multiple PNG files, one PNG image per page. The path to save the PNG can be specified, is must be a file and not a directory. The conversion uses a command from ImageMagick, convert (renamed convert_img on Windows). If no PNG path is specified, the name and the emplacement of the postScript file will be used instead. If a path different from a PNG file is given, a missing extension will be assumed.

This function returns the path of the PNG image (or the path of the common prefix of the files if multiple images are created).

Keyword arguments:

- path : the path of the PS file.
- pngPath : the path where to save the PNG file(s) (default None)

3.1.7 findPNGFiles

findPNGFiles(*pngPath*, *expectations=False*)

Utility function to retrieve one or multiple PNG files from an emplacement. The given path can design a file or the prefix of multiple PNG files (in the format prefix-x.png, where x is an integer). If a file is found, the prefix case won't be evaluated. All the path found are stored in a table and returned by the function. If the table is empty and it was expected to find results, an error message is displayed.

Keyword arguments:

- pngPath – the file's emplacement or the common prefix.

- expectations – if the function displays an error message in case of empty result (default False).

3.2 Intermediate functions

For all this functions, if a pngPath is provided, the corresponding image(s) will be loaded and displayed (using checkPNGFiles). If the compilation is forced or if no file is provided/found, the magic will be executed.

3.2.1 displayPSCode

displayPSCode(*data*, *pngPath=None*)

This function writes the given text in an PS file and calls displayPSFile on it, transmitting the PNG path.

Keyword arguments:

- data – the string containing the PS code.
- pngPath – the path where to fetch/save the PNG file (default None).

3.2.2 displayPSFile

displayPSFile(*path*, *pngPath=None*)

This function converts the PS file given in argument and displays the images resulting of the conversion.

Keyword arguments:

- path – the path of the PS file to convert.
- pngPath – the path where to fetch/save the PNG file (default None).

3.2.3 displayLatexCode

displayLatexCode(*data*, *pngPath=None*, *doubleCompilation=False*)

This function writes the given text in an TeX file and calls displayLatexFile on it, transmitting the PNG path and the indication for the double compilation.

Keyword arguments:

- data – the string containing the LaTeX code.
- pngPath – the path where to fetch/save the PNG file (default None).
- doubleCompilation – whether the compilation should be executed twice (default False)

3.2.4 displayLatexFile

displayLatexFile(*path*, *pngPath=None*, *doubleCompilation=False*)

This function converts the LaTeX file given in argument, using the double compilation if specified, generating a PS file. The result is then converted and displayed using displayPSFile (with the specified pngPath transmitted).

Keyword arguments:

- path – the path of the LaTeX file to convert.

- pngPath – the path where to fetch/save the PNG file (default None).
- doubleCompilation – whether the compilation should be executed twice (default False)

3.2.5 checkPNGFiles

checkPNGFiles (*path*, *expectations=False*)

Retrieve and display the PNG images corresponding to the specified path. If no file is found can display an error message, depending on the value of *expectations*. The algorithm used for retrieving files is given in findPNGFiles. This function returns whether files were found.

Keyword arguments:

- path – the path of the PNG file or the common prefix of multiple files.
- expectations – if the function displays an error message in case of empty result (default False).

3.3 IPython utiliy

load_ipython_extension (*ipython*)

The function used by the notebook to register the magics and their types.

CHAPTER

FOUR

EXAMPLES OF THE MODULE

4.1 Displaying files

To display a file, you can use both

```
%LatexFile <your file>

%PSFile <your file>
```

depending on the type of the file. If you want to save the result to a specific emplacement, use the following expressions :

```
%LatexFile <your file> <save path>

%PSFile <your file> <save path>
```

4.2 Writing code

There is no check in the notebook, however the error in compilation will display an error message.

In the case of LaTeX code, a link to the log document will appear and you'll be able to find what error caused the compilation fail. However, for PS code no error log is displayed. It is consequently recommended to use LaTeX compilation to create PS code.

Format of writing :

```
%%LatexCode <optional save path> <option*>
<multi-line code>

%%PSCode <optional save path> <option*>
<multi-line code>
```

4.3 Generating notebook for another user

If you want to share your notebook with persons not using latex, you can ! First, enable forced compilation with

```
%PSViewer -forceCompile
```

Then, write your notebook as usual, specifying a save path for each figure using the magics. Take care at giving distinct names for each figure !

If you want to share, you just have to give the notebook with the PNG images. The receiver just has to execute the cells without activating the forced compilation and the figures will be retrieved from the already generated images.