MYPlotSpec Documentation

Release 0.1

Karl Debiec

January 15, 2015

CONTENTS

CHAPTER

ONE

INTRODUCTION

MYPlotSpec (Matplotlib YAML Plot Specification) is a Python package used to write Matplotlib-based plotting scripts that may be configured using YAML.

The intended purpose is to make it possible to very quickly write plotting scripts for particular types of data while detailed control over plot configuration. Essentially, the minimal 'quick & dirty' code needed to plot a certain type of data should be identical to the polished code used to produce a publication-quality figure with precisely chosen proportions, ticks, colors, and fonts. Settings may be routed to specific figures, subplots, or datasets. MyPlotSpec supports a system of defaults and presets that make it easy to prepare multiple versions of figures, such as for notebook, printing, and presentation.

This project has no affiliation with the developers of matplotlib or YAML.

1.1 Example Usage

For example: the following input file might be used to plot three datasets on three pages of a pdf file:

```
figures:
  all:
    outfile: test.pdf
  0:
    subplots:
      0:
        infile:  dataset_0.txt
  1:
    subplots:
      0:
        infile:  dataset_1.txt
  2:
    subplots:
      0:
        infile:  dataset_2.txt
```

However, the third dataset extends beyond the range we have set. we may adjust the range for that figure only:

```
figures:
  all:
    outfile: test.pdf
    subplots:
      all:
        xticks:  [1,2,3,4,5,6,7,8,9,10]
        yticks:  [0,1,2,3,4,5]
  0:
    subplots:
      0:
        infile:  dataset_0.txt
  1:
    subplots:
      0:
        infile:  dataset_1.txt
  2:
    subplots:
      0:
        yticks:  [0,1,2,3,4,5,6,7,8,9,10]
        infile:  dataset_2.txt
```

The package tries to allow some consistency in the names of arguments passed to matplotlib functions, particularly for the names of text strings and font properties. In matplotlib different text-related functions accept string arguments with names including 's', 't', 'title', and 'label'; and font properties arguments with names including fontproperties, font_properties, fp, and prop. MYPlotSpec should support the original argument names as well as a more consistent alternative such as title_fp, label_fp, tick_fp, or legend_fp.

1.2 Requirements

MYPlotSpec supports Python 2.7 and 3.4, and requires the following packages:

- matplotlib
- numpy
- six
- yaml

This package has been tested with Anaconda python 2.1.0 on Arch Linux, OSX Yosemite, and in Cygwin on Windows 8.1.

1.3 Installation

Put in your `$PYTHONPATH`:

:: export PYTHONPATH=/path/to/my/python/modules:$PYTHONPATH

where `/path/to/my/python/modules` contains `MYPlotSpec`. If writing a module that uses MYPlotSpec, it is recommended to include it as a submodule, as the module is still undergoing changes that may break compatibility.

1.4 License

Released under a 3-clause BSD licence

CHAPTER TWO

GETTING STARTED

For most users the first step for using MYPlotSpec is to create a subclass of `Figure_Manager` in which `draw_dataset` has been overridden to draw the desired type of data.

```
from MYPlotSpec.Figure_Manager import Figure_Manager
from MYPlotSpec.Method_Defaults_Presets import Method_Defaults_Presets
from MYPlotSpec.Manage_Kwargs import Manage_Kwargs

class Custom_Figure_Mangater(Figure_Manager)

    @Method_Defaults_Presets
    def draw_dataset(subplot, **kwargs):
```

CHAPTER

THREE

CLASSES

3.1 Figure_Manager

Class to manage the generation of figures using matplotlib

class `MYPlotSpec.Figure_Manager.`**`Figure_Manager`**
Class to manage the generation of figures using matplotlib

`Figure_Manager.`**`draw_report`**`(`**in_args*, ***in_kwargs*`)`
Draws a series of figures based on provided specifications

Arguments:

figures Figure specifications

`Figure_Manager.`**`draw_figure`**`(`**in_args*, ***in_kwargs*`)`
Draws a figure

Arguments:

outfile Output filename

title Figure title

shared_xlabel X label to be shared among subplots

shared_ylabel Y label to be shared among subplots

shared_legend Legend to be shared among subplots

`Figure_Manager.`**`draw_subplot`**`(`**in_args*, ***in_kwargs*`)`
Draws a subplot

Arguments:

subplot <Axes> on which to act

title Subplot's title

legend Subplot's legend

shared_handles Nascent OrderedDict of handles and labels shared among subplots of host figure

`Figure_Manager.`**`draw_dataset`**`(`**in_args*, ***in_kwargs*`)`
Draws a dataset

Arguments:

subplot <Axes> on which to act

infile Input file; first column is x, second is y

label Dataset label

handles Nascent list of dataset handles on subplot

CHAPTER

FOUR

DECORATORS

4.1 Method_Defaults_Presets

Decorator class to manage the passage of defaults and presets from a class to a method of that class.

class `MYPlotSpec.Method_Defaults_Presets.`**`Method_Defaults_Presets`**(*debug=False*, ***kwargs*)

Decorator class to manage the passage of defaults and presets from a class to a method of that class.

Defaults are accessed from the class's instance (or class)variable `self.defaults`, and may be a dictionary, a path to a yaml file, or a yaml string. The first level of keys are the names of methods of the class, and the values are the corresponding defaults for each argument of that method:

```
self.defaults = """
    method_1:
      method_1_argument_1: 1000
      method_1_argument_2: abcd
    method_2
      method_2_argument_1: 2000
      method_2_argument_2: efgh
    ...
"""
```

Presets are accessed from the instance variable `self.presets`. These are treated similarly to defaults, but contain an outer level of keys corresponding to names of the available presets:

```
self.presets = """
    preset_1:
        method_1:
          method_1_argument_1: 1001
          method_1_argument_2: abcde
        method_2
          method_2_argument_1: 2001
          method_2_argument_2: efghi
    preset_2:
        method_1:
          method_1_argument_1: 1002
          method_1_argument_2: abcdef
        method_2
          method_2_argument_1: 2002
          method_2_argument_2: efghij
"""
```

When this decorator is used to wrap a method of a class, it adds to the arguments being passed `defaults`, containing only the defaults specified for that method, and `presets`, containing only the presets containing arguments for that method.

```
@Method_Defaults_Presets()
def method_1(*args, **kwargs):
    ...

> kwargs = {
>   "defaults": {
>     "method_1_argument_1": 1000,
>     "method_1_argument_2": "asdf"
>   },
>   "presets": {
>     "preset_1": {
>       "method_1_argument_1": 1001,
>       "method_1_argument_2": "asde"
>     },
>     "preset_1": {
>       "method_1_argument_1": 1002,
>       "method_1_argument_2": "asdef"
>     }
>   },
>   ...
> }
```

4.2 Manage_Kwargs

Decorator class to manage the passage of keyword arguments to a wrapped function or method

class `MYPlotSpec.Manage_Kwargs.`**`Manage_Kwargs`**(*debug=False*, ***kwargs*)

Decorator class to manage the passage keyword arguments to a wrapped function or method

Accumulates keyword arguments from several sources, in order of increasing priority:

- *defaults* keyword argument at call:

```
my_function(
  defaults = {
    "width":  5.0
    "height": 5.0
    },
...)
```

  defaults may be a dictionary, path to a yaml file, or a yaml string.

- *preset* and *presets* keyword arguments at call:

```
my_function(
  preset  = "letter",
  presets = {
    "letter": {
      "width":   8.5
      "height": 11.0
    },
    "legal": {
      "width":   8.5
      "height": 14.0
    }
...)
```

preset defines the selected preset (or a list of selected presets), and *presets* the available presets; *preset* may be a string or list, and *presets* may be a dictionary, path to a yaml file, or yaml string.

•*yaml_dict* and *yaml_keys* keyword arguments at function call:

```
my_function(
  yaml_dict = """
    figures:
      all:
        width:   11.0
        height:  17.0
        outfile: plot.pdf
    figures:
      0:
        width:   12.0
  """
  yaml_keys = [["figures", "all"], ["figures", "0"]]
...)
```

yaml_dict defines the yaml file, and *yaml_keys* the paths within the yaml file from whih to load arguments, in order of priority. *yaml_dict* may be a dictionary, path to a yaml file, or yaml string if yaml_keys* is omitted, the complete yaml file will be used.

•Additional keyword arguments at call

```
my_wrapped_function(
  width = 6.0,
...)
```

All of the above will override defaults provided in the function declaration itself.

4.3 Figure_Output

Decorator class to manage the output of matplotlib figures by a wrapped function or method

class `MYPlotSpec.Figure_Output.`**`Figure_Output`**(*debug=False*, *verbose=True*)

Decorator class to manage the output of matplotlib figures by a wrapped function or method

Saves figure returned by wrapped function to a file named *outfile*; passing additional keyword arguments *savefig_kw* to savefig. For pdf output, additional argument *outfiles* may be provided; this contains a dictionary whose keys are the absolute paths to output pdf files, and whose values are references to open PdfPages objects representing those files. The purpose of this is to allow figures output from multiple calls to the wrapped function (or other analogously wrapped functions) to be output to sequential pages of the same pdf file. Typically *outfiles* will be initialized before calling this wrapped function; and once calls to the function is complete the close() method of each outfile in *outfiles* should be run.

CHAPTER

FIVE

FUNCTIONS

5.1 Formatting

5.1.1 Axes

Functions for formatting axes

MYPlotSpec.axes.**set_xaxis**(*subplot*, *xticks=None*, *xtick_kw=None*, *xticklabels=None*, *xticklabel_fp=None*, *ticklabel_fp=None*, *xticklabel_kw=None*, *xlabel=None*, *xlabel_fp=None*, *label_fp=None*, *xlabel_kw=None*, *xtick_params=None*, *tick_params=None*, *xtick_pad=None*, *tick_pad=None*, *xlw=None*, *lw=None*, ***kwargs*)

Formats the x-axis of a subplot using provided keyword arguments

Arguments:

subplot <Axes> on which to act

xticks Ticks; first and last are used as upper and lower boundaries

xtick_kw Keyword arguments passed to set_xticks()

xticklabels Tick label text

[x]ticklabel_fp Tick label font

xticklabel_kw Keyword arguments passed to set_xticklabels()

xlabel Label text

[x]label_fp Label font

xlabel_kw Keyword arguments passed to set_xlabel()

[x]tick_params Keyword arguments passed to set_tick_params(); only affect x axis

xaxis_kw Additional keyword arguments

[x]tick_pad Padding between ticks and labels

[x]lw Line width

MYPlotSpec.axes.**set_yaxis**(*subplot*, *yticks=None*, *ytick_kw=None*, *yticklabels=None*, *yticklabel_fp=None*, *ticklabel_fp=None*, *yticklabel_kw=None*, *ylabel=None*, *ylabel_fp=None*, *label_fp=None*, *ylabel_kw=None*, *ytick_params=None*, *tick_params=None*, *ytick_pad=None*, *tick_pad=None*, *ylw=None*, *lw=None*, ***kwargs*)

Formats the y-axis of a subplot using provided keyword arguments

Arguments:

subplot <Axes> on which to act

yticks Ticks; first and last are used as upper and lower boundaries

ytick_kw Keyword arguments passed to set_yticks()

yticklabels Tick label text

[y]ticklabel_fp Tick label font

yticklabel_kw Keyword arguments passed to set_yticklabels()

ylabel Label text

[y]label_fp Label font

ylabel_kw Keyword arguments passed to set_ylabel()

[y]tick_params Keyword arguments passed to set_tick_params(); only affect y axis

yaxis_kw Additional keyword arguments

[y]tick_pad Padding between ticks and labels

[y]lw Line width

5.1.2 Text

Functions for formatting text

`MYPlotSpec.text.`**`set_title`**(*figure_or_subplot*, *title=None*, *title_fp=None*, **args*, ***kwargs*)

Draw a title on *figure_or_subplot*

Arguments:

figure_or_subplot <Figure> or <Axes> on which to act

title Title text

title_fp Title font

top Distance between top of figure and title (inches); Figure title only

title_kw Keyword arguments passed to figure.suptitle() or subplot.set_title()

Additional title_kw Arguments:

top Distance between top of figure and title

Returns:

title <Text>

`MYPlotSpec.text.`**`set_shared_xlabel`**(*figure_or_subplots*, *xlabel=None*, *xlabel_fp=None*, *label_fp=None*, **args*, ***kwargs*)

Draws an x-axis label shared by multiple subplots

Arguments:

figure_or_subplots <Figure> or OrderedDict of <Axes> on which to act; if Figure, position is relative to all subplots, if OrderedDict, position is relative to subplots in OrderedDict only

xlabel Label text

[x]label_fp Label font

xlabel_kw Keyword arguments passed to set_text()

Additional xlabel_kw Arguments:

top Distance between top of figure and label; if negative, distance between topmost plot and label; overrides *bottom*

bottom Distance between bottom of figure and label; if negative, distance between bottommost plot and label

Returns:

label <Text>

`MYPlotSpec.text.`**`set_shared_ylabel`**(*figure_or_subplots*, *ylabel=None*, *ylabel_fp=None*, *label_fp=None*, **args*, ***kwargs*)

Draws a y-axis label shared by multiple subplots

Arguments:

figure_or_subplots <Figure> or OrderedDict of <Axes> on which to act; if Figure, position is relative to all subplots, if OrderDict, position is relative to subplots in OrderedDict

ylabel Label text

[y]label_fp Label font

ylabel_kw Keyword arguments passed to set_text()

Additional ylabel_kw Arguments:

left Distance between left side of figure and label; if negative, distance between leftmost plot and label

right Distance between right side of figure and label; if negative, distance between rightmost plot and label; overrides *left*

rotation Label rotation; default: 'vertical'

Returns:

text <Text>

`MYPlotSpec.text.`**`set_inset`**(*subplot*, *inset=None*, *inset_fp=None*, **args*, ***kwargs*)

Draws an inset on a subplot

Arguments:

subplot <Axes> on which to act

inset Inset text

inset_fp Inset font

inset_kw Keyword arguments passed to set_text()

Additional inset_kw Arguments:

x Horizontal position of inset in subplot reference frame (subplot coordinate); overrides *xpro*

y Vertical position of inset in subplot reference frame (subplot coordinate), overrides *ypro*

xpro Horizontal position of inset in subplot reference frame (proportion)

ypro Vertical position of inset in subplot reference frame (proportion)

Returns:

text <Text>

`MYPlotSpec.text.`**`set_text`**(*figure_or_subplot*, *text=None*, *text_fp=None*, **args*, ***kwargs*)

Prints text on a figure or subplot

Arguments:

figure_or_subplot <Figure> or <Axes> on which to act

text Text

text_fp Text Font

text_kw Keyword arguments passed to text()

Returns:

text <Text>

5.1.3 Legend

Functions for formatting legends

Note: Acceptable values of *loc* and their meanings, for reference:

```
0 = Best
+---------+
|2   9   1|
|6  10   7|
|3   8   4|
+---------+
```

`MYPlotSpec.legend.`**`set_legend`**(*subplot*, *handles=None*, *legend_lw=None*, *legend_fp=None*, ***kwargs*)

Draws and formats a legend on *subplot*

By default includes all series; may alternatively accept manual OrderedDict of handles and labels

Arguments:

subplot <Axes> on which to act

handles OrderedDict; keys are series labels and values are handles

legend_lw Legend handle linewidth

legend_fp Legend font

legend_kw Keyword arguments passed to *subplot*.legend()

Returns:

legend <Legend>

`MYPlotSpec.legend.`**`set_shared_legend`**(*figure*, *subplots*, ***kwargs*)

Adds a subplot to *figure*, draws a legend on it and hides subplot borders

Useful when several plots on the same figure share the same source.

Arguments:

figure Figure

subplots OrderedDict of subplots

Returns:

legend new legend

5.2 Auxiliary

5.2.1 General

General functions

`MYPlotSpec.`**`merge_dicts`**(*dict1*, *dict2*)

Recursively merges two dictionaries.

Arguments:

dict1 First dictionary

dict2 Second dictionary; values for keys shared by both dictionaries are drawn from *dict2*

Returns:

merged Merged dictionary

`MYPlotSpec.`**`multi_kw`**(*keys*, *dictionary*)

Scans *dictionary* for *keys*, returns first matching value (or None if none are present), and deletes *keys* from *dictionary*

This is not really ideal, but is appropriate here due to the inconsistency of the names of some of matplotlib's arguments, in particular fontproperties, font_properties, fp, and sometimes prop.

Arguments:

keys List of acceptable keyword arguments in order of priority; first match is used and other are deleted

dictionary Dictionary of keyword arguments to be tested

default Value to return if not found

Returns:

value Value from *dictionary* of first matching keyword in *keys*, or None if none are present

`MYPlotSpec.`**`pad_zero`**(*ticks*, *digits=None*, ***kwargs*)

Returns a list of tick labels, each with the same number of digits after the decimal

Arguments:

ticks List or numpy array of ticks

digits Number of digits to include after the decimal

Returns:

tick_labels Tick labels, each with the same number of digits after the decimal

5.2.2 Matplotlib

`MYPlotSpec.`**`get_edges`**(*figure_or_subplots*, ***kwargs*)

Finds the outermost edges of a set of subplots on a figure

Arguments:

figure_or_subplots <Figure> or list or dictionary of <Axes> on which to act

Returns:

edges dictionary of edges; keys are 'left', 'right', 'top', and 'bottom'

`MYPlotSpec.`**`get_color`**(*color*)

Generates a color

Arguments:

> ***color*** May be a string "red", "blue", etc. corresponding to a default color; a string "pastel.red", "pastel.blue" corresponding to a palette and color, a list of three floating point numbers corresponding to red, green, and blue values, or a single floating point number corresponding to a grayscale color

`MYPlotSpec.`**`get_font`**(*fp=None*, ***kwargs*)

Arguments:

> ***fp*** Font properties

Behavior:

If *fp* is <FontProperties>, acts as a pass-through, returns
fp argument
If *fp* is a String of form '##L', makes new <FontProperties>
object for which '##' = size; 'L' = {'r': regular,
'b' bold}
If *fp* is a Dict, makes new <FontProperties> using given
keyword arguments

Returns:

> ***fp*** <FontProperties> object to given specifications

`MYPlotSpec.`**`get_figure_subplots`**(*figure=None*, *subplots=None*, *nrows=None*, *ncols=None*, *nsubplots=None*, *left=None*, *sub_width=None*, *wspace=None*, *right=None*, *top=None*, *sub_height=None*, *hspace=None*, *bottom=None*, *fig_width=None*, *fig_height=None*, *figsize=None*, *verbose=False*, *debug=False*, ***kwargs*)

Generates a figure and subplots to specifications

Differs from matplotlib's built-in functions in that it:

- Accepts subplot dimensions is inches rather than proportional figure coordinates
- Optionally calculates figure dimensions from provided subplot dimensions, rather than the reverse
- Returns subplots in an OrderedDict
- Smoothly adds additional subplots to a previously-generated figure (i.e. can be called multiple times)

Arguments:

> ***figure*** Figure, if adding subplots to a previously-existing figure
>
> ***subplots*** OrderedDict of subplots, if adding subplots to a previously-existing figure
>
> ***nrows*** Number of rows of subplots
>
> ***ncols*** Number of columns of subplots
>
> ***nsubplots*** Number of subplots to add; if less than nrows*ncols (e.g. 2 cols and 2 rows but only three subplots)
>
> ***sub_width*** Width of subplot(s)
>
> ***sub_height*** Height of subplot(s)
>
> ***left*** Margin between left side of figure and leftmost subplots

right Margin between right side of figure and rightmost subplot

top Margin between top of figure and highest subplot

bottom Margin between bottom of figure and lowest subplot

wspace Horizontal margin between adjacent subplots

hspace Vertical margin between adjacent subplots

fig_width Width of figure; may be determined from above

fig_height Height of figure, may be determined from above

figsize Equivalent to [fig_width, fig_height]

figure_kw Keyword arguments passed to figure()

subplot_kw Keyword arguments passed to Axes()

axes_kw Alias to subplot_kw

verbose Enable verbose output

debug Enable debug output

Returns:

figure <Figure>

subplots OrderedDict of subplots

`MYPlotSpec.`**`identify`**`(`*subplots*, ***kwargs*`)`

Identifies key of each subplot with inset text

Arguments:

subplots OrderedDict of subplots

PYTHON MODULE INDEX