pplt Documentation

Release 1.0

International AudioLabs Erlangen

Aug 24, 2018

Contents

Paper PLT is a simple matplotlib renderer wrapper to build plots from the commandline.

Contents:

CHAPTER 1

Installation

```
pip install git+https://github.com/nils-werner/pplt.git
```

1.1 Setup

PPLT expects there to be a `pplt/` directory in your `$PYTHONPATH` or your current directory.

Inside this directory there must be an `__init__.py` file and all renderer modules you need. e.g.

```
|- paper.tex
`- pplt/
   |- __init__.py
   |- input_signal.py
   `- result_plots.py
```

With this structure you can then render your plots using

```
pplt input_signal.pdf
pplt result_plots.pdf
```

1.2 Makefiles

One key aspect is that each output file is represented by one renderer module Python file. This makes it possible to have a `Makefile` rule for each plot you need and only re-render the ones that were actually changed.

```
# Render plots automatically using PPLT
fig/%.pdf: pplt/%.py
        pplt $@

# Build plots when building paper.pdf
```

(continues on next page)

(continued from previous page)

```
paper.pdf: fig/input_signal.pdf fig/result_plots.pdf
    latex paper.tex
```

CHAPTER 2

Configuration

You may set the following values in a `pplt/conf.py` file:

2.1 Processing

`aliases`

A dictionary of aliases for your render modules. The key of each entry is the alias name, the value the actual linked-to module.

When using tuples here, the first value is the module and all following values are values passed to `main()`

```
aliases = {
    "alias":            "logspec",              # logspec.main()
    "logspec_real":     ("logspec", "real"),    # logspec.main("real")
    "logspec_synth":    ("logspec", "synth"),   # logspec.main("synth")
}
```

2.2 Styling

`stylesheet`

The Matplotlib style you wish to use. Use `plt.style.available` to see what styles you have available.

```
>>> plt.style.available
[u'dark_background', u'bmh', u'grayscale', u'ggplot', u'fivethirtyeight']
```

`columnwidth`

the width of your columns. You may resize the figure in LaTeX later on, but the resulting text size depends on a correct setting.

```
columnwidth = 244.6937  # Get this from LaTeX using \showthe\columnwidth
```

rc_params

A dictionary of values passed on to `plt.rcParams.update()`

```
rc_params = {
    'backend': 'ps',
    'axes.labelsize': 9,
    'legend.fontsize': 9,
    'xtick.labelsize': 8,
    'ytick.labelsize': 8,
    'text.usetex': True,
}
```

See also:

Defining per-module RC settings *RC Settings*

sns_params

A dictionary of values passed on to `sns.set()`

```
sns_params = {
    'font': 'serif',
}
```

See also:

Defining per-module RC settings *RC Settings*

tight_layout

Global setting do enable/disable tight layouts.

```
tight_layout = False
```

CHAPTER 3

Renderer Modules

Your code must at least implement a `main()` function that accepts the matplotlib instance as its only paramter and returns a figure

```
def main(plt):
    f, ax = plt.subplots(1, 1, figsize=(6, 2))

    ax.plot(...)

    return f
```

3.1 Artists

Your function may return the figure alongside a list/tuple of additional artists:

```
def main(plt):
    f, ax = plt.subplots(1, 1, figsize=(6, 2))

    ax.plot(...)
    lgd = ax.legend(...)

    return f, (lgd,)
```

3.2 RC Settings

Your module may define additional RC settings for Matplotlib and Seaborn as well as set a `stylesheet` and a `pre_hook` and a `post_hook` which will run before and after the plotting.

```
from contextlib import contextmanager

```

(continues on next page)

(continued from previous page)

```
def main(plt):
    f, ax = plt.subplots(1, 1, figsize=(6, 2))

    ax.plot(...)
    lgd = ax.legend(...)

    return f, (lgd,)


sns_params = {
    'font': 'serif',
}


rc_params = {
    'font.size': 9,
}


stylesheet = 'grayscale'


def pre_hook(plt):
    plt.style.use('grayscale')


def post_hook(plt):
    pass
```

See also:

Global RC settings in `conf.py`: *rc_params*, *sns_params*, *stylesheet*

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index