

## Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.

[Sign up](#)

[Dismiss](#)

### Custom Jupyter Notebook Themes

#jupyter #jupyter-notebook #theme #css #syntax-highlighting #jupyter-themes

529 commits

1 branch

86 releases

14 contributors

MIT

Branch: master ▾

New pull request

[Find file](#)

[Clone or download ▾](#)

 dunovank	bump version v0.18.2; fixed soft-selectd cells border style; other mi...	...	Latest commit 3d954e5 13 days ago
 <a href="#">jupyterthemes</a>	bump version v0.18.2; fixed soft-selectd cells border style; other mi...	13 days ago	
 <a href="#">screens</a>	Merge branch 'master' of https://github.com/dunovank/jupyter-themes	a month ago	
 <a href="#">tests</a>	remove straggler from test_themes	9 months ago	
 <a href="#">.gitignore</a>	bump version v0.16.8; attempt to fix lesscpy dependency	3 months ago	
 <a href="#">.travis.yml</a>	bump version v0.16.3; removed broken matplotlib dependency	5 months ago	
 <a href="#">CHANGES.md</a>	bump version v0.18.2; fixed soft-selectd cells border style; other mi...	13 days ago	
 <a href="#">LICENSE.txt</a>	reorganized files	a year ago	
 <a href="#">MANIFEST.in</a>	bump version v0.17.3, attempt to Windows bug errors on remove tempfil...	3 months ago	
 <a href="#">README.md</a>	minor changes	20 days ago	
 <a href="#">index.ipynb</a>	bump version v0.16.2; added matplotlib and seaborn dependencies, upda...	5 months ago	
 <a href="#">requirements.txt</a>	bump version v0.17.7, removed dependency checker	a month ago	
 <a href="#">setup.cfg</a>	minor patch to setupcfg	a year ago	
 <a href="#">setup.py</a>	bump version v0.18.2; fixed soft-selectd cells border style; other mi...	13 days ago	

 [README.md](#)

## jupyterthemes

**Theme-ify your Jupyter Notebooks!**

Author	Version	Status	Demo
Kyle Dunovan	<a href="#">pypi v0.18.2</a>	<a href="#">build passing</a>	<a href="#">launch binder</a>

*plotting style*



*markdown/equations*

**Fitting Flat Models**

- All models are initially fit by optimizing the full set of parameters to the "flattened" data (flat meaning the average data collapsing across all conditions of interest).

**Steps in fitting routine:**

- Global optimization on flat data (average values collapsing across any/all conditions)
- Local optimization using parameters passed from global optimizer as starting values

- Flat model fits are performed by identifying the full set of parameter values that minimize the following cost-function:

$$\chi^2 = \sum [\omega * (\hat{Y} - Y)]^2$$

- $Y$  is an array of observed data (e.g., accuracy, RT quantiles, etc.)
- $\hat{Y}$  is an equal length array of corresponding model-predicted values, given by the parameterized model  $f(\theta)$
- The error  $\chi^2$  between the predicted and the observed data ( $\hat{Y} - Y$ ) is weighted by an equal length array of scalars  $\omega$  proportional to the inverse of the variance in each value of  $Y$ .

**Accessing flat weights ( $\omega$ ) and data ( $Y$ ) vectors**

```
flat_data = model.observedDF.mean()
flat_wts = model.wtsDF.mean()
```

pandas dataframes

**Building a model**

```
In [7]: model = build.Model(data=data, kind='xdpm', depends_on={'v': 'Cond'})
model.observedDF.head()
```

idx	Cond	acc	200	250	300	350	400	c10	c20	...	c90	e10	e20	e30	e40	
0	28	bsl	0.9917	1.0	1.0	0.95	0.60	0.00	0.5051	0.5252	...	0.5982	0.4961	0.5185	0.5317	0.531
1	28	pnl	0.9752	1.0	1.0	0.95	0.80	0.10	0.5177	0.5318	...	0.6119	0.5198	0.5324	0.5452	0.554
2	29	bsl	0.9917	1.0	1.0	1.00	0.90	0.00	0.5250	0.5377	...	0.5984	0.5268	0.5450	0.5451	0.548
3	29	pnl	0.9669	1.0	1.0	1.00	0.75	0.35	0.5314	0.5452	...	0.6250	0.5314	0.5322	0.5448	0.545
4	30	bsl	0.9421	1.0	1.0	1.00	0.80	0.25	0.5298	0.5585	...	0.6384	0.5361	0.5452	0.5606	0.584

5 rows × 26 columns

**Header of observed dataframe (model.observedDF)**

- idx: subject ID
- Cond: Baseline(bsl)/Caution(pnl) (could be any experimental condition of interest)
- Acc: Accuracy on "go" trials
- sacc: Mean accuracy on "stop" trials (mean condition SSD used during simulations)
- c10 - c90: 10th - 90th RT quantiles for correct responses
- e10 - e90: 10th - 90th RT quantiles for error responses

command palette

A screenshot of a Jupyter Notebook interface. The top menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. The title bar on the right says "Python 2". A search dialog box is open in the center, containing the text "find and replace". Below the search bar, there is a dropdown menu with the following options:

- C confirm restart kernel (command) ⌘, ⌘
- confirm restart kernel and clear output
- confirm restart kernel and run all cells
- find and replace (command) F

The main code editor area contains Python code. The "find and replace" dialog is overlaid on the code, obscuring some of the text.

```
#!/usr/local/bin/env python
from __future__ import division
import os
import
from
import
from
def
    confirm restart kernel and clear output
**kwargs
    confirm restart kernel and run all cells
1
1
1
z=theta['zz']; ssd=theta['ssd'];
ssv=-abs(theta['ssv']); a=theta['a'];

nSS=int(ntrials*(1-theta['pGo']))

dx=np.sqrt(si*tau)
Pg=0.5*(1 + mu*dx/si)
Ps=0.5*(1 + ssv*dx/si)
```

## *oceans16 syntax*

## Trialwise Analysis Functions

```
In [14]: @jit(float64(float64[:,], float64, float64, float64), nopython=True)
def lower_cross_jit_single(trace, onset, lower, dt):
    return onset + (np.argmax(trace <= 0) * dt)

@jit(float64(float64[:,], float64, float64, float64), nopython=True)
def upper_cross_jit_single(trace, onset, upper, dt):
    return onset + (np.argmax(trace >= upper) * dt)
```

## Component/uFuncs

```
In [15]: @jit(nopython=True)
def slice_theta_array(theta_array, index_arrays, n_vals, shape):
    i, ix = 0, 0
    param_array = np.empty(shape)
    for nlvls in n_vals:
        arr_ix = index_arrays[i]
        param_array[:, i] = theta_array[ix:ix+nlvls][arr_ix]
        i += 1
        ix += nlvls
    return param_array.T
```

## *grade3 syntax*

## Trialwise Analysis Functions

```
In [14]: @jit(float64(float64[:,], float64, float64, float64), nopython=True)
def lower_cross_jit_single(trace, onset, lower, dt):
    return onset + (np.argmax(trace <= 0) * dt)

@jit(float64(float64[:,], float64, float64, float64), nopython=True)
def upper_cross_jit_single(trace, onset, upper, dt):
    return onset + (np.argmax(trace >= upper) * dt)
```

## Component/uFuncs

```
In [15]: @jit(nopython=True)
def slice_theta_array(theta_array, index_arrays, n_vals, shape):
    i, ix = 0, 0
    param_array = np.empty(shape)
    for nlvls in n_vals:
        arr_ix = index_arrays[i]
        param_array[:, i] = theta_array[ix:ix+nlvls][arr_ix]
        i += 1
        ix += nlvls
    return param_array.T
```

*onedork* syntax

## Trialwise Analysis Functions

```
In [14]: @jit(float64(float64[:,], float64, float64, float64), nopython=True)
def lower_cross_jit_single(trace, onset, lower, dt):
    return onset + (np.argmax(trace <= 0) * dt)

@jit(float64(float64[:,], float64, float64, float64), nopython=True)
def upper_cross_jit_single(trace, onset, upper, dt):
    return onset + (np.argmax(trace >= upper) * dt)
```

## Component/uFuncs

```
In [15]: @jit(nopython=True)
def slice_theta_array(theta_array, index_arrays, n_vals, shape):
    i, ix = 0, 0
    param_array = np.empty(shape)
    for nlvls in n_vals:
        arr_ix = index_arrays[i]
        param_array[:, i] = theta_array[ix:ix+nlvls][arr_ix]
        i += 1
        ix += nlvls
    return param_array.T
```

*chesterish* syntax

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell, titled 'Trialwise Analysis Functions', contains code for calculating onset times based on trace data. The second cell, titled 'Component/uFuncs', contains code for slicing theta arrays.

```
In [14]:
```

```
@jit(float64(float64[:, :], float64, float64, float64), nopython=True)
def lower_cross_jit_single(trace, onset, lower, dt):
    return onset + (np.argmax(trace <= 0) * dt)

@jit(float64(float64[:, :], float64, float64, float64), nopython=True)
def upper_cross_jit_single(trace, onset, upper, dt):
    return onset + (np.argmax(trace >= upper) * dt)
```

  

```
In [15]:
```

```
@jit(nopython=True)
def slice_theta_array(theta_array, index_arrays, n_vals, shape):
    i, ix = 0, 0
    param_array = np.empty(shape)
    for nlvls in n_vals:
        arr_ix = index_arrays[i]
        param_array[:, i] = theta_array[ix:ix+nlvls][arr_ix]
        i += 1
        ix += nlvls
    return param_array.T
```

## Links

- [jupyterthemes on PyPI](#)
- [jupyterthemes on GitHub](#)

## Requirements

- Python 2.7, 3.4, 3.5, 3.6
- Jupyter ([Anaconda](#) recommended)
- matplotlib

## Install with pip

```
# install jupyterthemes
pip install jupyterthemes

# upgrade to latest version
pip install --upgrade jupyterthemes
```

## Known issues

- **refreshing / removing / resetting:** depending on your system, browser, etc., you may need to empty your browser cache after installing a new theme ( `-t` ) or attempting to restore the default ( `-r` ) in order for those changes to take effect. (see discussion [here](#)). At the very least you'll need to refresh your browser window (usually `cmd+r` or `ctrl+r` ).
- **install issue:** if you get an error saying `jt` is not recognized, try [this](#) fix.
- **slow render when scrolling:** fix available [here](#)
- **for best results:** use `notebook>=5.0` ( `pip install --upgrade notebook` )

## Command Line Usage

```
jt [-h] [-l] [-t THEME] [-f MONOFONT] [-fs MONOSIZE] [-nf NBFONT]
[-nfs NBFONTSIZE] [-tf TCFONT] [-tfs TCFONTSIZE] [-dfs DFFONTSIZE]
[-m MARGINS] [-cursw CURSORWIDTH] [-cursc CURSORCOLOR] [-vim]
[-cellw CELLWIDTH] [-lineh LINEHEIGHT] [-altp] [-altdmd] [-altout]
[-P] [-T] [-N] [-r] [-dfonts]
```

### Description of Command Line options

cl options	arg	default
Usage help	-h	--
List Themes	-l	--
Theme Name to Install	-t	--
Code Font	-f	--
Code Font-Size	-fs	11
Notebook Font	-nf	--
Notebook Font Size	-nfs	13
Text/MD Cell Font	-tf	--
Text/MD Cell Fontsize	-tfs	13
Pandas DF Fontsize	-dfs	9
Output Area Fontsize	-ofs	8.5
Mathjax Fontsize (%)	-mathfs	100
Intro Page Margins	-m	auto
Cell Width	-cellw	980
Line Height	-lineh	170
Cursor Width	-cursw	2
Cursor Color	-cursc	--
Alt Prompt Layout	-altp	--
Alt Markdown BG Color	-altdmd	--
Alt Output BG Color	-altout	--
Style Vim NBExt*	-vim	--
Toolbar Visible	-T	--
Name & Logo Visible	-N	--
Reset Default Theme	-r	--
Force Default Fonts	-dfonts	--

### Command Line Examples

```

# list available themes
# onedork | grade3 | oceans16 | chesterish | monokai | solarizedl | solarizedd
jt -l

# select theme...
jt -t chesterish

# restore default theme
# NOTE: Need to delete browser cache after running jt -r
# If this doesn't work, try starting a new notebook session.
jt -r

# toggle toolbar ON and notebook name ON
jt -t grade3 -T -N

# set code font to 'Roboto Mono' 12pt
# (see monospace font table below)
jt -t onedork -f roboto -fs 12

# set code font to Fira Mono, 11.5pt
# 3digit font-sizes get converted into float (115-->11.5)
# 2digit font-sizes > 25 get converted into float (85-->8.5)
jt -t solarizedd -f fira -fs 115

# set font/font-size of markdown (text cells) and notebook (interface)
# see sans-serif & serif font tables below
jt -t oceans16 -tf merriserif -tfs 10 -nf ptsans -nfs 13

# adjust cell width (% screen width) and line height
jt -t chesterish -cellw 90% -lineh 170

# or set the cell width in pixels by leaving off the '%' sign
jt -t solarizedl -cellw 860

# fix the container-margins on the intro page (defaults to 'auto')
jt -t monokai -m 200

# adjust cursor width (in px) and make cursor red
# options: b (blue), o (orange), r (red), p (purple), g (green), x (font color)
jt -t oceans16 -cursc r -cursw 5

# choose alternate prompt layout (narrower/no numbers)
jt -t grade3 -altp

# my two go-to styles
# dark
jt -t onedork -fs 95 -altp -tfs 11 -nfs 115 -cellw 88% -T
# light
jt -t grade3 -fs 95 -altp -tfs 11 -nfs 115 -cellw 88% -T

```

## Set Plotting Style (from within notebook)

`jtplot.style()` makes changes to matplotlib's rcParams dictionary so that figure aesthetics match those of a chosen jupyterthemes style. In addition to setting the color scheme, `jtplot.style()` allows you to control various figure properties (spines, grid, font scale, etc.) as well as the plotting "context" (borrowed from [seaborn](#)).

Note, these commands do not need to be re-run every time you generate a new plot, just once at the beginning of your notebook or whenever style changes are desired after that.

**Pro-tip:** Include the following two lines in `~/.ipython/profile_default/startup/startup.ipynb` file to set plotting style automatically whenever you start a notebook:

```
# import jtplot submodule from jupyterthemes
from jupyterthemes import jtplot

# currently installed theme will be used to
# set plot style if no arguments provided
jtplot.style()
```

## jtplot.style() Examples

```
# import jtplot module in notebook
from jupyterthemes import jtplot

# choose which theme to inherit plotting style from
# onedork | grade3 | oceans16 | chesterish | monokai | solarizedl | solarizedd
jtplot.style(theme='onedork')

# set "context" (paper, notebook, talk, poster)
# scale font-size of ticklabels, legend, etc.
# remove spines from x and y axes and make grid dashed
jtplot.style(context='talk', fscale=1.4, spines=False, gridlines='--')

# turn on X- and Y-axis tick marks (default=False)
# turn off the axis grid lines (default=True)
# and set the default figure size
jtplot.style(ticks=True, grid=False, figsize=(6, 4.5))

# reset default matplotlib rcParams
jtplot.reset()
```

## Monospace Fonts (code cells)

<b>-f arg</b>	<b>Monospace Font</b>
anka	Anka/Coder
anonymous	Anonymous Pro
aurulent	Aurulent Sans Mono
bitstream	Bitstream Vera Sans Mono
bpmono	BPmono
code	Code New Roman
consolamono	Consolamono
cousine	Cousine
dejavu	DejaVu Sans Mono
droidmono	Droid Sans Mono
fira	Fira Mono
firacode	Fira Code
generic	Generic Mono
hack	Hack
hasklig	Hasklig

<b>-f arg</b>	<b>Monospace Font</b>
inconsolata	Inconsolata-g
inputmono	Input Mono
liberation	Liberation Mono
meslo	Meslo
office	Office Code Pro
oxygen	Oxygen Mono
roboto	Roboto Mono
saxmono	saxMono
source	Source Code Pro
sourcemed	Source Code Pro Medium
ptmono	PT Mono
ubuntu	Ubuntu Mono

### Sans-Serif Fonts

<b>-nf/-tf arg</b>	<b>Sans-Serif Font</b>
opensans	Open Sans
droidsans	Droid Sans
exosans	Exo_2
latosans	Lato
ptsans	PT Sans
robotosans	Roboto
sourcesans	Source Sans Pro

### Serif Fonts

<b>-nf/-tf arg</b>	<b>Serif Font</b>
loraserif	Lora
ptserif	PT Serif
georgiaserif	Georgia
cardoserif	Cardo
crimsonserif	Crimson Text
ebserif	EB Garamond
merriserif	Merriweather
neutonserif	Neuton
goudyserif	Sorts Mill Goudy

