

---

# **Matplotlib**

***Release 2.0.2***

**John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the m**

**May 10, 2017**



## CONTENTS

<b>I</b>	<b>User's Guide</b>	<b>1</b>
1	Introduction	3
2	Installing	5
3	Tutorials	11
4	Working with text	97
5	Colors	145
6	Customizing matplotlib	173
7	Interactive plots	189
8	Selected Examples	205
9	What's new in matplotlib	239
10	GitHub Stats	339
11	License	417
12	Credits	421
<b>II</b>	<b>The Matplotlib API</b>	<b>423</b>
13	Plotting commands summary	425
14	API Changes	433
15	The top level <code>matplotlib</code> module	483
16	<code>afm</code> (Adobe Font Metrics interface)	487
17	<code>animation</code> module	491

<b>18</b>	<b>artist Module</b>	<b>535</b>
<b>19</b>	<b>Axes class</b>	<b>555</b>
<b>20</b>	<b>axis and tick API</b>	<b>793</b>
<b>21</b>	<b>backends</b>	<b>913</b>
<b>22</b>	<b>cbook</b>	<b>953</b>
<b>23</b>	<b>cm (colormap)</b>	<b>971</b>
<b>24</b>	<b>collections</b>	<b>975</b>
<b>25</b>	<b>colorbar</b>	<b>1145</b>
<b>26</b>	<b>colors</b>	<b>1151</b>
<b>27</b>	<b>container</b>	<b>1165</b>
<b>28</b>	<b>dates</b>	<b>1167</b>
<b>29</b>	<b>dviread</b>	<b>1181</b>
<b>30</b>	<b>figure</b>	<b>1185</b>
<b>31</b>	<b>finance</b>	<b>1207</b>
<b>32</b>	<b>font_manager</b>	<b>1219</b>
<b>33</b>	<b>gridspec</b>	<b>1227</b>
<b>34</b>	<b>image</b>	<b>1231</b>
<b>35</b>	<b>legend and legend_handler</b>	<b>1237</b>
<b>36</b>	<b>lines</b>	<b>1245</b>
<b>37</b>	<b>markers</b>	<b>1255</b>
<b>38</b>	<b>mathtext</b>	<b>1259</b>
<b>39</b>	<b>mlab</b>	<b>1279</b>
<b>40</b>	<b>offsetbox</b>	<b>1313</b>
<b>41</b>	<b>patches</b>	<b>1325</b>
<b>42</b>	<b>path</b>	<b>1367</b>
<b>43</b>	<b>patheffects</b>	<b>1375</b>
<b>44</b>	<b>projections</b>	<b>1379</b>

<b>45</b>	<b>pyplot</b>	<b>1387</b>
<b>46</b>	<b>rcsetup</b>	<b>1593</b>
<b>47</b>	<b>sankey</b>	<b>1597</b>
<b>48</b>	<b>scale</b>	<b>1605</b>
<b>49</b>	<b>spines</b>	<b>1615</b>
<b>50</b>	<b>style</b>	<b>1619</b>
<b>51</b>	<b>text</b>	<b>1621</b>
<b>52</b>	<b>ticker</b>	<b>1635</b>
<b>53</b>	<b>tight_layout</b>	<b>1647</b>
<b>54</b>	<b>Working with transformations</b>	<b>1649</b>
<b>55</b>	<b>triangular grids</b>	<b>1671</b>
<b>56</b>	<b>type1font</b>	<b>1683</b>
<b>57</b>	<b>units</b>	<b>1685</b>
<b>58</b>	<b>widgets</b>	<b>1687</b>
<b>III</b>	<b>The Matplotlib FAQ</b>	<b>1703</b>
<b>59</b>	<b>Installation</b>	<b>1705</b>
<b>60</b>	<b>Usage</b>	<b>1713</b>
<b>61</b>	<b>How-To</b>	<b>1725</b>
<b>62</b>	<b>Troubleshooting</b>	<b>1741</b>
<b>63</b>	<b>Environment Variables</b>	<b>1745</b>
<b>64</b>	<b>Working with Matplotlib in Virtual environments</b>	<b>1747</b>
<b>65</b>	<b>Working with Matplotlib on OSX</b>	<b>1749</b>
<b>IV</b>	<b>Matplotlib AxesGrid Toolkit</b>	<b>1753</b>
<b>66</b>	<b>Overview of AxesGrid toolkit</b>	<b>1757</b>
<b>67</b>	<b>The Matplotlib AxesGrid Toolkit User's Guide</b>	<b>1779</b>

68	The Matplotlib AxesGrid Toolkit API	1795
69	The Matplotlib axes_grid1 Toolkit API	1805
V	mplot3d	1823
70	Matplotlib mplot3d toolkit	1825
VI	Toolkits	1879
71	Mapping Toolkits	1883
72	General Toolkits	1885
73	High-Level Plotting	1889
VII	External Resources	1893
74	Books, Chapters and Articles	1895
75	Videos	1897
76	Tutorials	1899
VIII	The Matplotlib Developers' Guide	1901
77	Contributing	1903
78	Developer's tips for testing	1911
79	Developer's tips for documenting matplotlib	1917
80	Developer's guide for creating scales and transformations	1929
81	Developer's tips for writing code for Python 2 and 3	1933
82	Working with <i>Matplotlib</i> source code	1937
83	Reviewers guideline	1957
84	Release Guide	1959
85	Matplotlib Enhancement Proposals	1965
86	Licenses	2017
87	Default Color changes	2019

<b>IX Matplotlib Examples</b>	<b>2023</b>
88 animation Examples	2025
89 api Examples	2029
90 axes_grid Examples	2123
91 color Examples	2175
92 event_handling Examples	2187
93 frontpage Examples	2221
94 images_contours_and_fields Examples	2225
95 lines_bars_and_markers Examples	2237
96 misc Examples	2251
97 mplot3d Examples	2275
98 pie_and_polar_charts Examples	2323
99 pylab_examples Examples	2329
100pyplots Examples	2769
101scales Examples	2791
102shapes_and_collections Examples	2793
103showcase Examples	2799
104specialty_plots Examples	2817
105statistics Examples	2825
106style_sheets Examples	2857
107subplots_axes_and_figures Examples	2873
108tests Examples	2877
109text_labels_and_annotations Examples	2889
110ticks_and_spines Examples	2895
111units Examples	2909
112user_interfaces Examples	2935
113widgets Examples	3001

<b>X</b>	<b>Glossary</b>	<b>3015</b>
	<b>Bibliography</b>	<b>3019</b>
	<b>Python Module Index</b>	<b>3021</b>
	<b>Index</b>	<b>3023</b>



# **Part I**

## **User's Guide**



## INTRODUCTION

Matplotlib is a library for making 2D plots of arrays in [Python](#). Although it has its origins in emulating the MATLAB®<sup>1</sup> graphics commands, it is independent of MATLAB, and can be used in a Pythonic, object oriented way. Although Matplotlib is written primarily in pure Python, it makes heavy use of [NumPy](#) and other extension code to provide good performance even for large arrays.

Matplotlib is designed with the philosophy that you should be able to create simple plots with just a few commands, or just one! If you want to see a histogram of your data, you shouldn't need to instantiate objects, call methods, set properties, and so on; it should just work.

For years, I used to use MATLAB exclusively for data analysis and visualization. MATLAB excels at making nice looking plots easy. When I began working with EEG data, I found that I needed to write applications to interact with my data, and developed an EEG analysis application in MATLAB. As the application grew in complexity, interacting with databases, http servers, manipulating complex data structures, I began to strain against the limitations of MATLAB as a programming language, and decided to start over in Python. Python more than makes up for all of MATLAB's deficiencies as a programming language, but I was having difficulty finding a 2D plotting package (for 3D [VTK](#) more than exceeds all of my needs).

When I went searching for a Python plotting package, I had several requirements:

- Plots should look great - publication quality. One important requirement for me is that the text looks good (antialiased, etc.)
- Postscript output for inclusion with TeX documents
- Embeddable in a graphical user interface for application development
- Code should be easy enough that I can understand it and extend it
- Making plots should be easy

Finding no package that suited me just right, I did what any self-respecting Python programmer would do: rolled up my sleeves and dived in. Not having any real experience with computer graphics, I decided to emulate MATLAB's plotting capabilities because that is something MATLAB does very well. This had the added advantage that many people have a lot of MATLAB experience, and thus they can quickly get up to steam plotting in python. From a developer's perspective, having a fixed user interface (the *pylab* interface) has been very useful, because the guts of the code base can be redesigned without affecting user code.

The Matplotlib code is conceptually divided into three parts: the *pylab interface* is the set of functions provided by `matplotlib.pylab` which allow the user to create plots with code quite similar to MATLAB

---

<sup>1</sup> MATLAB is a registered trademark of The MathWorks, Inc.

figure generating code (*Pyplot tutorial*). The *Matplotlib frontend* or *Matplotlib API* is the set of classes that do the heavy lifting, creating and managing figures, text, lines, plots and so on (*Artist tutorial*). This is an abstract interface that knows nothing about output. The *backends* are device-dependent drawing devices, aka renderers, that transform the frontend representation to hardcopy or a display device (*What is a backend?*). Example backends: PS creates **PostScript®** hardcopy, SVG creates **Scalable Vector Graphics** hardcopy, Agg creates PNG output using the high quality **Anti-Grain Geometry** library that ships with Matplotlib, GTK embeds Matplotlib in a **Gtk+** application, GTKAgg uses the Anti-Grain renderer to create a figure and embed it in a Gtk+ application, and so on for **PDF**, **WxWidgets**, **Tkinter**, etc.

Matplotlib is used by many people in many different contexts. Some people want to automatically generate PostScript files to send to a printer or publishers. Others deploy Matplotlib on a web application server to generate PNG output for inclusion in dynamically-generated web pages. Some use Matplotlib interactively from the Python shell in Tkinter on Windows™. My primary use is to embed Matplotlib in a Gtk+ EEG application that runs on Windows, Linux and Macintosh OS X.

## INSTALLING

There are many different ways to install matplotlib, and the best way depends on what operating system you are using, what you already have installed, and how you want to use it. To avoid wading through all the details (and potential complications) on this page, there are several convenient options.

### 2.1 Installing pre-built packages

#### 2.1.1 Most platforms : scientific Python distributions

The first option is to use one of the pre-packaged python distributions that already provide matplotlib built-in. The Continuum.io Python distribution ([Anaconda](#) or [miniconda](#)) and the Enthought distribution ([Canopy](#)) are both excellent choices that “just work” out of the box for Windows, OSX and common Linux platforms. Both of these distributions include matplotlib and *lots* of other useful tools.

#### 2.1.2 Linux : using your package manager

If you are on Linux, you might prefer to use your package manager. matplotlib is packaged for almost every major Linux distribution.

- Debian / Ubuntu : `sudo apt-get install python-matplotlib`
- Fedora / Redhat : `sudo yum install python-matplotlib`

#### 2.1.3 Mac OSX : using pip

If you are on Mac OSX you can probably install matplotlib binaries using the standard Python installation program [pip](#). See *Installing OSX binary wheels*.

#### 2.1.4 Windows

If you don't already have Python installed, we recommend using one of the [scipy-stack compatible Python distributions](#) such as WinPython, Python(x,y), Enthought Canopy, or Continuum Anaconda, which have matplotlib and many of its dependencies, plus other useful packages, preinstalled.

For [standard Python](#) installations, install matplotlib using [pip](#):

```
python -m pip install -U pip setuptools
python -m pip install matplotlib
```

In case Python 2.7 or 3.4 are not installed for all users, the Microsoft Visual C++ 2008 (64 bit or 32 bit for Python 2.7) or Microsoft Visual C++ 2010 (64 bit or 32 bit for Python 3.4) redistributable packages need to be installed.

Matplotlib depends on [Pillow](#) for reading and saving JPEG, BMP, and TIFF image files. Matplotlib requires [MiKTeX](#) and [GhostScript](#) for rendering text with LaTeX. [FFmpeg](#), [avconv](#), [mencoder](#), or [ImageMagick](#) are required for the animation module.

The following backends should work out of the box: agg, tkagg, ps, pdf and svg. For other backends you may need to install [pycairo](#), [PyQt4](#), [PyQt5](#), [PySide](#), [wxPython](#), [PyGTK](#), [Tornado](#), or [GhostScript](#).

TkAgg is probably the best backend for interactive use from the standard Python shell or IPython. It is enabled as the default backend for the official binaries. GTK3 is not supported on Windows.

The Windows wheels (\*.whl) on the [PyPI download page](#) do not contain test data or example code. If you want to try the many demos that come in the matplotlib source distribution, download the \*.tar.gz file and look in the examples subdirectory. To run the test suite, copy the lib\matplotlib\tests and lib\mpl\_toolkits\tests directories from the source distribution to sys.prefix\Lib\site-packages\matplotlib and sys.prefix\Lib\site-packages\mpl\_toolkits respectively, and install [nose](#), [mock](#), [Pillow](#), [MiKTeX](#), [GhostScript](#), [ffmpeg](#), [avconv](#), [mencoder](#), [ImageMagick](#), and [Inkscape](#).

## 2.2 Installing from source

If you are interested in contributing to matplotlib development, running the latest source code, or just like to build everything yourself, it is not difficult to build matplotlib from source. Grab the latest *tar.gz* release file from the [PyPI files page](#), or if you want to develop matplotlib or just need the latest bugfixed version, grab the latest git version *Source install from git*.

The standard environment variables CC, CXX, PKG\_CONFIG are respected. This means you can set them if your toolchain is prefixed. This may be used for cross compiling.

```
export CC=x86_64-pc-linux-gnu-gcc
export CXX=x86_64-pc-linux-gnu-g++
export PKG_CONFIG=x86_64-pc-linux-gnu-pkg-config
```

Once you have satisfied the requirements detailed below (mainly python, numpy, libpng and freetype), you can build matplotlib.

```
cd matplotlib
python setup.py build
python setup.py install
```

We provide a [setup.cfg](#) file that goes with [setup.py](#) which you can use to customize the build process. For example, which default backend to use, whether some of the optional libraries that matplotlib ships with are installed, and so on. This file will be particularly useful to those packaging matplotlib.

If you have installed prerequisites to nonstandard places and need to inform matplotlib where they are, edit `setuptools.py` and add the base dirs to the `basedir` dictionary entry for your `sys.platform`. e.g., if the header to some required library is in `/some/path/include/someheader.h`, put `/some/path` in the `basedir` list for your platform.

### 2.2.1 Build requirements

These are external packages which you will need to install before installing matplotlib. If you are building on OSX, see [Building on OSX](#). If you are building on Windows, see [Building on Windows](#). If you are installing dependencies with a package manager on Linux, you may need to install the development packages (look for a “-dev” postfix) in addition to the libraries themselves.

#### Required Dependencies

**python 2.7, 3.4, 3.5 or 3.6** [Download python](#).

**numpy 1.7.1 (or later)** array support for python ([download numpy](#))

**setuptools** Setuptools provides extensions for python package installation.

**dateutil 1.1 or later** Provides extensions to python datetime handling. If using pip, easy\_install or installing from source, the installer will attempt to download and install `python_dateutil` from PyPI.

**pyparsing** Required for matplotlib’s `mathtext` math rendering support. If using pip, easy\_install or installing from source, the installer will attempt to download and install `pyparsing` from PyPI.

**libpng 1.2 (or later)** library for loading and saving *PNG* files ([download](#)). libpng requires zlib.

**pytz** Used to manipulate time-zone aware datetimes. <https://pypi.python.org/pypi/pytz>

**FreeType 2.3 or later** Library for reading true type font files. If using pip, easy\_install or installing from source, the installer will attempt to locate FreeType in expected locations. If it cannot, try installing `pkg-config`, a tool used to find required non-python libraries.

**cycler 0.10.0 or later** Composable cycle class used for constructing style-cycles

**six** Required for compatibility between python 2 and python 3

#### Dependencies for python 2

**functools32** Required for compatibility if running on Python 2.7.

**subprocess32** Optional, unix only. Backport of the subprocess standard library from 3.2+ for Python 2.7. It provides better error messages and timeout support.

#### Optional GUI framework

These are optional packages which you may want to install to use matplotlib with a user interface toolkit. See [What is a backend?](#) for more details on the optional matplotlib backends and the capabilities they provide.

**tk 8.3 or later, not 8.6.0 or 8.6.1** The TCL/Tk widgets library used by the TkAgg backend.

Versions 8.6.0 and 8.6.1 are known to have issues that may result in segfaults when closing multiple windows in the wrong order.

**pyqt 4.4 or later** The Qt4 widgets library python wrappers for the Qt4Agg backend

**pygtk 2.4 or later** The python wrappers for the GTK widgets library for use with the GTK or GTKAgg backend

**wxpython 2.8 or later** The python wrappers for the wx widgets library for use with the WX or WXAgg backend

### Optional external programs

**ffmpeg/avconv or mencoder** Required for the animation module to be save out put to movie formats.

**ImageMagick** Required for the animation module to be able to save to animated gif.

### Optional dependencies

**Pillow** If Pillow is installed, matplotlib can read and write a larger selection of image file formats.

**pkg-config** A tool used to find required non-python libraries. This is not strictly required, but can make installation go more smoothly if the libraries and headers are not in the expected locations.

### Required libraries that ship with matplotlib

**agg 2.4** The antigrain C++ rendering engine. matplotlib links against the agg template source statically, so it will not affect anything on your system outside of matplotlib.

**qhull 2012.1** A library for computing Delaunay triangulations.

**ttconv** truetype font utility

### 2.2.2 Building on Linux

It is easiest to use your system package manager to install the dependencies.

If you are on Debian/Ubuntu, you can get all the dependencies required to build matplotlib with:

```
sudo apt-get build-dep python-matplotlib
```

If you are on Fedora/RedHat, you can get all the dependencies required to build matplotlib by first installing yum-builddep and then running:

```
su -c "yum-builddep python-matplotlib"
```

This does not build matplotlib, but it does get the install the build dependencies, which will make building from source easier.



### 2.2.3 Building on OSX

The build situation on OSX is complicated by the various places one can get the libpng and freetype requirements (darwinports, fink, /usr/X11R6) and the different architectures (e.g., x86, ppc, universal) and the different OSX version (e.g., 10.4 and 10.5). We recommend that you build the way we do for the OSX release: get the source from the tarball or the git repository and follow the instruction in `README.osx`.

### 2.2.4 Building on Windows

The Python shipped from <https://www.python.org> is compiled with Visual Studio 2008 for versions before 3.3, Visual Studio 2010 for 3.3 and 3.4, and Visual Studio 2015 for 3.5 and 3.6. Python extensions are recommended to be compiled with the same compiler.

Since there is no canonical Windows package manager, the methods for building freetype, zlib, and libpng from source code are documented as a build script at [matplotlib-winbuild](#).

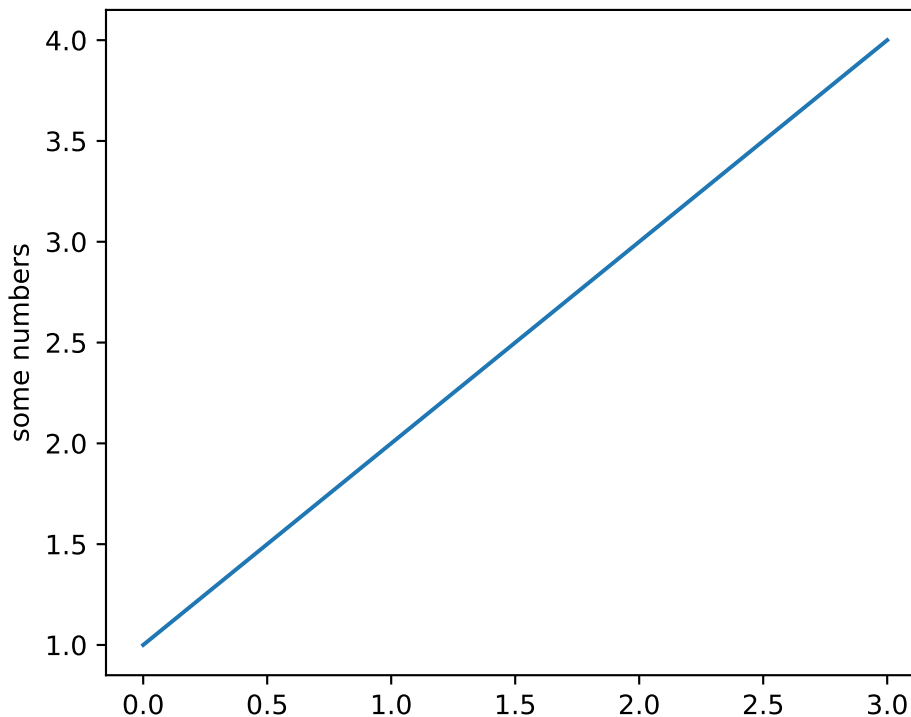


## 3.1 Introductory

### 3.1.1 Pyplot tutorial

*matplotlib.pyplot* is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In *matplotlib.pyplot* various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that “axes” here and in most places in the documentation refers to the *axes part of a figure* and not the strict mathematical term for more than one axis).

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```



You may be wondering why the x-axis ranges from 0-3 and the y-axis from 1-4. If you provide a single list or array to the `plot()` command, matplotlib assumes it is a sequence of y values, and automatically generates the x values for you. Since python ranges start with 0, the default x vector has the same length as y but starts with 0. Hence the x data are `[0, 1, 2, 3]`.

`plot()` is a versatile command, and will take an arbitrary number of arguments. For example, to plot x versus y, you can issue the command:

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
```

For every x, y pair of arguments, there is an optional third argument which is the format string that indicates the color and line type of the plot. The letters and symbols of the format string are from MATLAB, and you concatenate a color string with a line style string. The default format string is 'b-', which is a solid blue line. For example, to plot the above with red circles, you would issue

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```