



Fork me on GitHub

Version 3.0.0

[home](#) | [examples](#) | [tutorials](#) | [API](#) | [docs](#) » [User's Guide](#) » [Tutorials](#) »

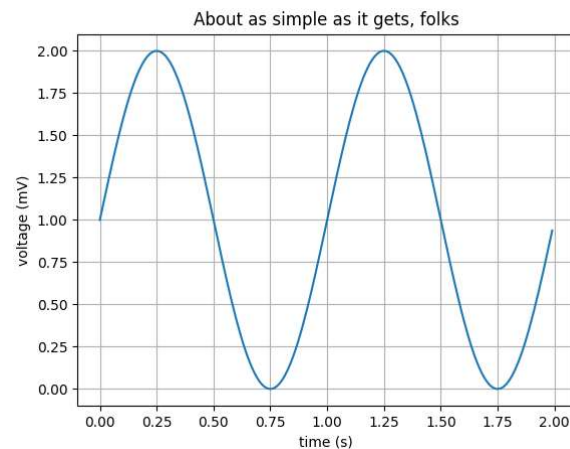
[previous](#) | [next](#) | [modules](#) | [index](#)

Sample plots in Matplotlib

Here you'll find a host of example plots with the code that generated them.

Line Plot

Here's how to create a line plot with text labels using `plot()`.



Simple Plot

Multiple subplots in one figure

Quick search

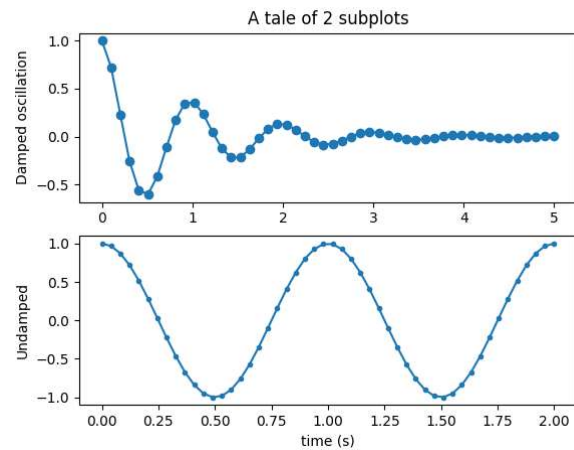
Go

Table Of Contents

Sample plots in Matplotlib

- [Line Plot](#)
- [Multiple subplots in one figure](#)
- [Images](#)
- [Contouring and pseudocolor](#)
- [Histograms](#)
- [Paths](#)
- [Three-dimensional plotting](#)
- [Streamplot](#)
- [Ellipses](#)
- [Bar charts](#)
- [Pie charts](#)
- [Tables](#)
- [Scatter plots](#)
- [GUI widgets](#)
- [Filled curves](#)
- [Date handling](#)
- [Log plots](#)
- [Polar plots](#)
- [Legends](#)
- [TeX-notation for text objects](#)

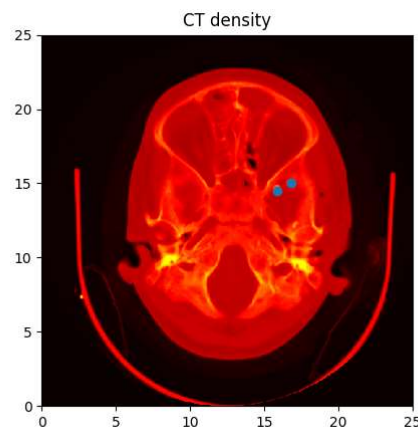
Multiple axes (i.e. subplots) are created with the `subplot()` function:



Subplot

Images

Matplotlib can display images (assuming equally spaced horizontal dimensions) using the `imshow()` function.



Example of using `imshow()` to display a CT scan

Contouring and pseudocolor

- [Native TeX rendering](#)
- [EEG GUI](#)
- [XKCD-style sketch plots](#)
- [Subplot example](#)

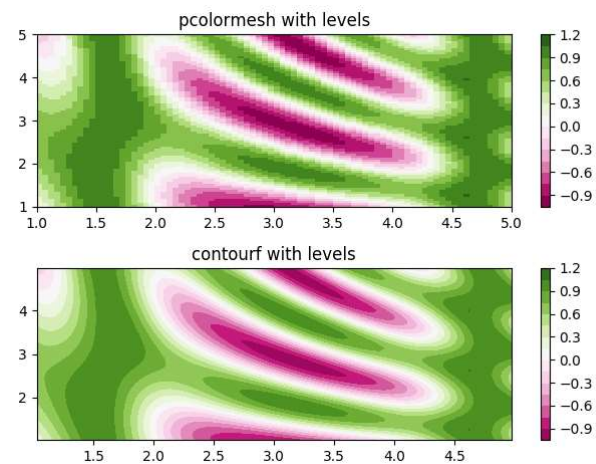
Related Topics

Documentation overview

- [User's Guide](#)
 - [Tutorials](#)
 - Previous: [Pyplot tutorial](#)
 - Next: [Image tutorial](#)

[Show Page Source](#)

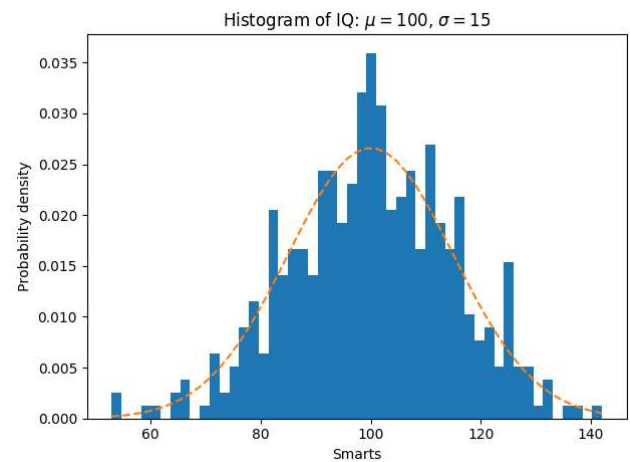
The `pcolormesh()` function can make a colored representation of a two-dimensional array, even if the horizontal dimensions are unevenly spaced. The `contour()` function is another way to represent the same data:



Example comparing `pcolormesh()` and `contour()` for plotting two-dimensional data

Histograms

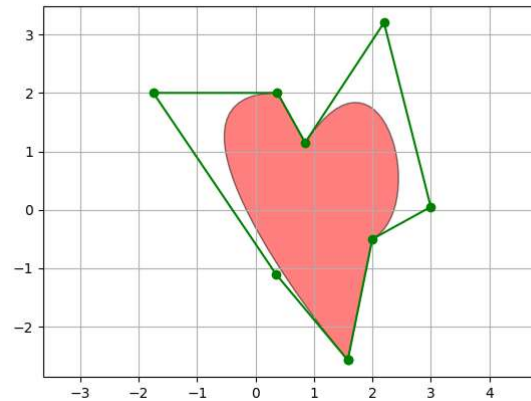
The `hist()` function automatically generates histograms and returns the bin counts or probabilities:



Histogram Features

Paths

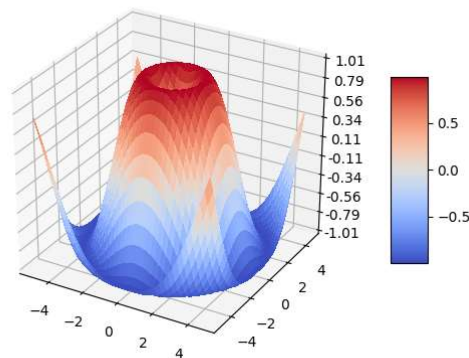
You can add arbitrary paths in Matplotlib using the `matplotlib.path` module:



Path Patch

Three-dimensional plotting

The `mplot3d` toolkit (see [Getting started](#) and [3D plotting](#)) has support for simple 3d graphs including surface, wireframe, scatter, and bar charts.

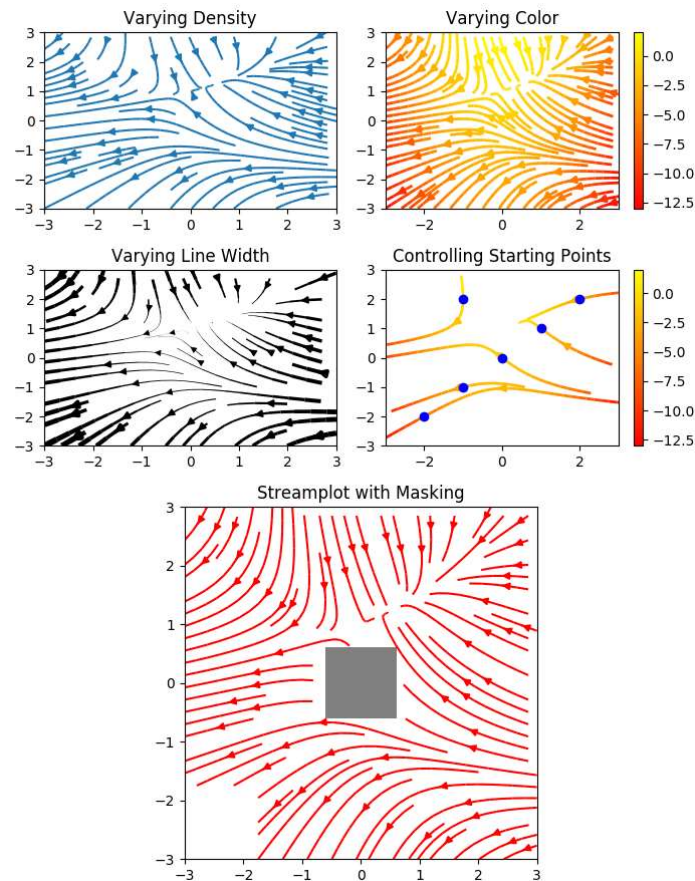


Surface3d

Thanks to John Porter, Jonathon Taylor, Reinier Heeres, and Ben Root for the `mplot3d` toolkit. This toolkit is included with all standard Matplotlib installs.

Streamplot

The `streamplot()` function plots the streamlines of a vector field. In addition to simply plotting the streamlines, it allows you to map the colors and/or line widths of streamlines to a separate parameter, such as the speed or local intensity of the vector field.

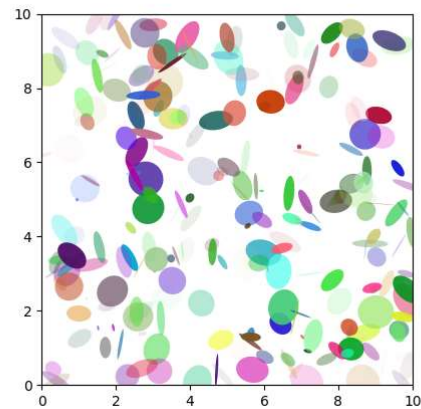


Streamplot with various plotting options.

This feature complements the `quiver()` function for plotting vector fields. Thanks to Tom Flannaghan and Tony Yu for adding the `streamplot` function.

Ellipses

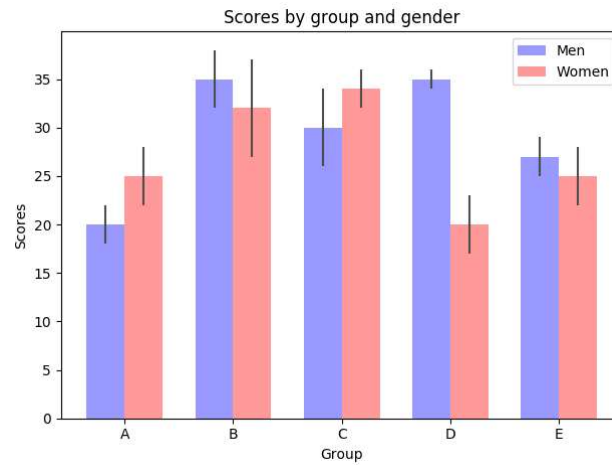
In support of the [Phoenix](#) mission to Mars (which used Matplotlib to display ground tracking of spacecraft), Michael Droettboom built on work by Charlie Moad to provide an extremely accurate 8-spline approximation to elliptical arcs (see [Arc](#)), which are insensitive to zoom level.



Ellipse Demo

Bar charts

Use the `bar()` function to make bar charts, which includes customizations such as error bars:

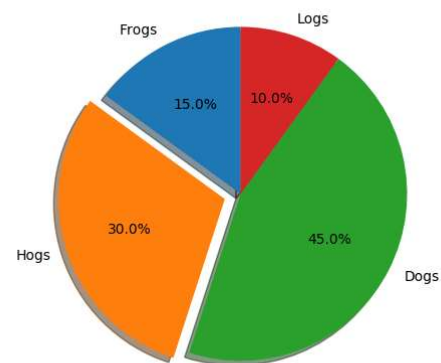


Bar chart Demo

You can also create stacked bars ([bar_stacked.py](#)), or horizontal bar charts ([barh.py](#)).

Pie charts

The `pie()` function allows you to create pie charts. Optional features include auto-labeling the percentage of area, exploding one or more wedges from the center of the pie, and a shadow effect. Take a close look at the attached code, which generates this figure in just a few lines of code.



Pie Features

Tables

The `table()` function adds a text table to an axes.

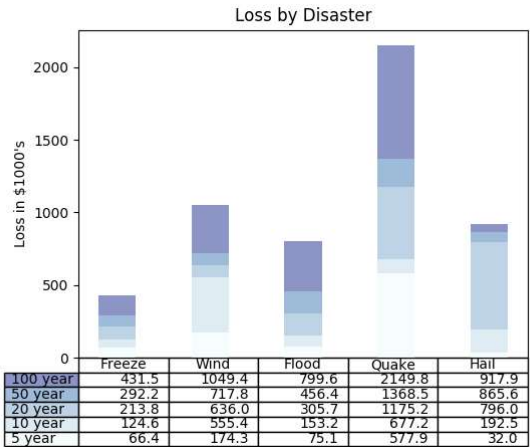
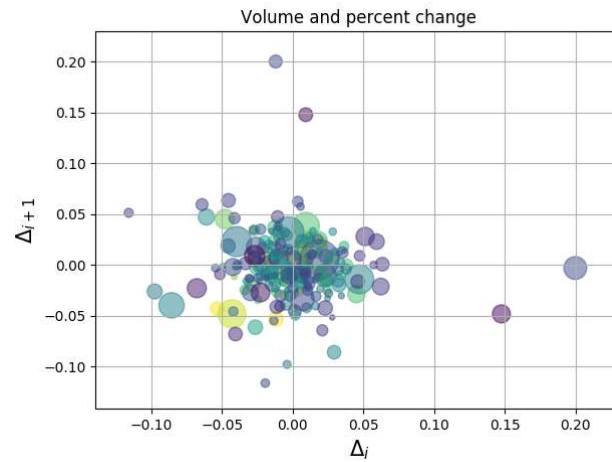


Table Demo

Scatter plots

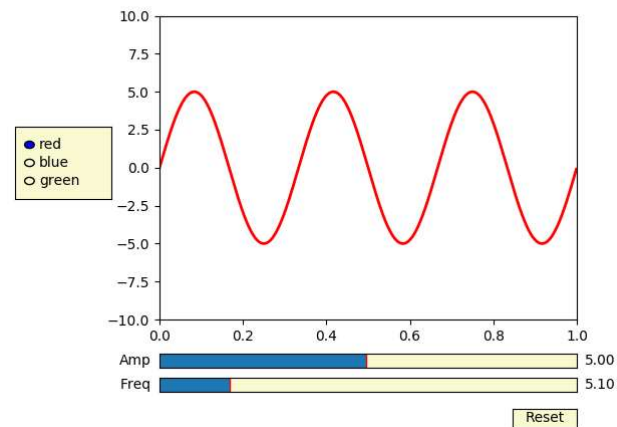
The `scatter()` function makes a scatter plot with (optional) size and color arguments. This example plots changes in Google's stock price, with marker sizes reflecting the trading volume and colors varying with time. Here, the alpha attribute is used to make semitransparent circle markers.



Scatter Demo2

GUI widgets

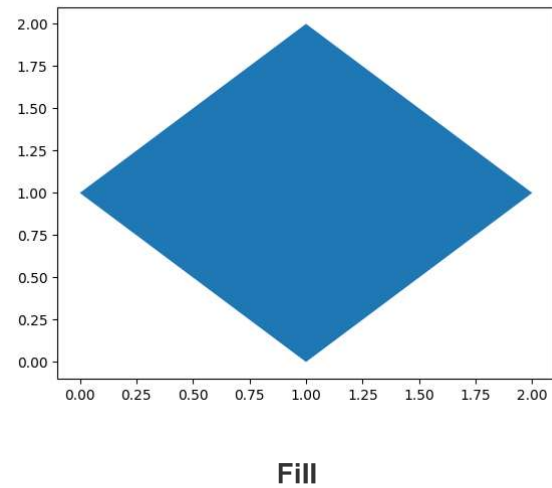
Matplotlib has basic GUI widgets that are independent of the graphical user interface you are using, allowing you to write cross GUI figures and widgets. See [matplotlib.widgets](#) and the [widget examples](#).



Slider and radio-button GUI.

Filled curves

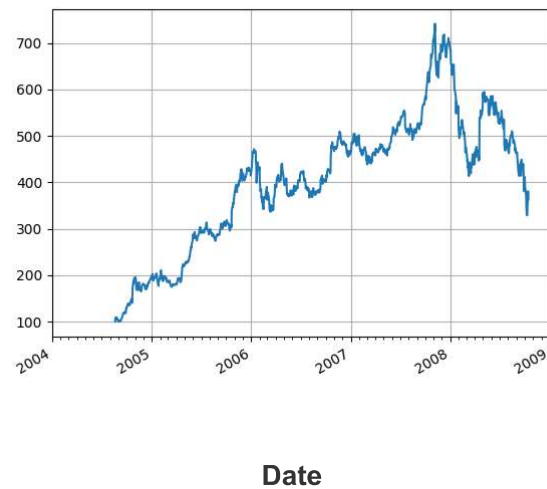
The `fill()` function lets you plot filled curves and polygons:



Thanks to Andrew Straw for adding this function.

Date handling

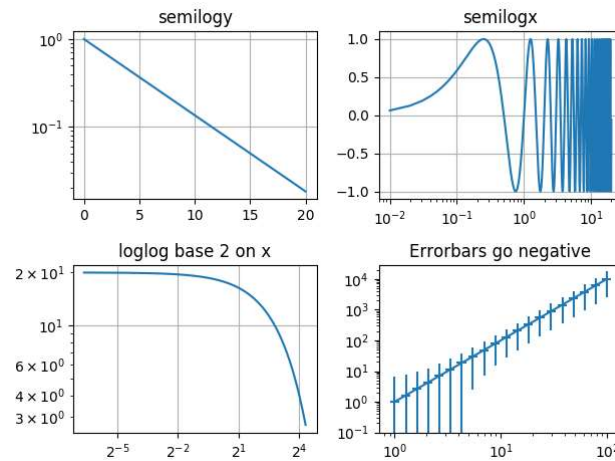
You can plot timeseries data with major and minor ticks and custom tick formatters for both.



See `matplotlib.ticker` and `matplotlib.dates` for details and usage.

Log plots

The `semilogx()`, `semilogy()` and `loglog()` functions simplify the creation of logarithmic plots.

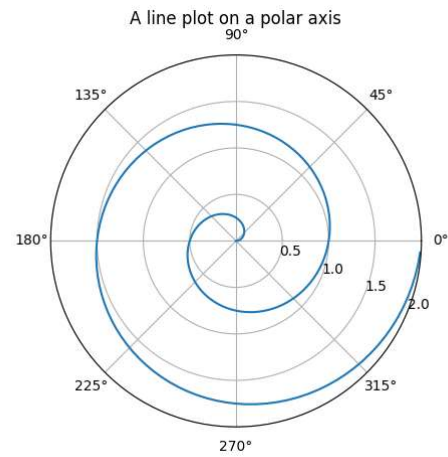


Log Demo

Thanks to Andrew Straw, Darren Dale and Gregory Lielens for contributions log-scaling infrastructure.

Polar plots

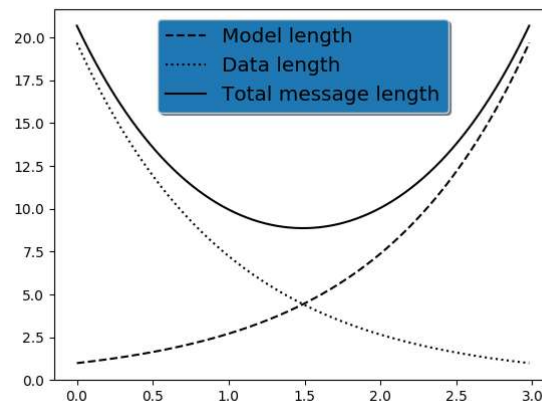
The `polar()` function generates polar plots.



Polar Demo

Legends

The `legend()` function automatically generates figure legends, with MATLAB-compatible legend-placement functions.



Legend

Thanks to Charles Twardy for input on the legend function.

TeX-notation for text objects

Below is a sampling of the many TeX expressions now supported by Matplotlib's internal mathtext engine. The mathtext module provides TeX style mathematical expressions using [FreeType](#) and the DejaVu, BaKoMa computer modern, or [STIX](#) fonts. See the [matplotlib.mathtext](#) module for additional details.

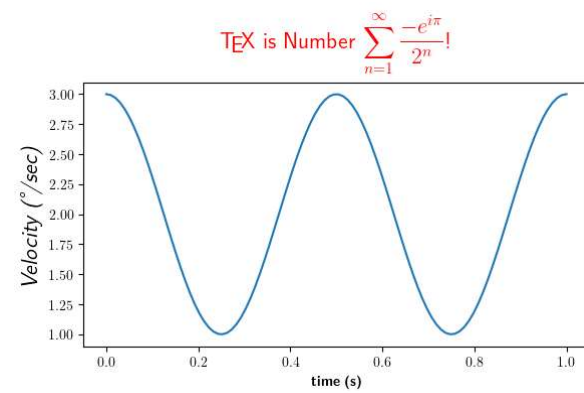
Matplotlib's math rendering engine
$W_{\delta_1 \rho_1 \sigma_2}^{3\beta} = U_{\delta_1 \rho_1}^{3\beta} + \frac{1}{8\pi^2} \int_{\alpha_2}^{\alpha_2'} d\alpha_2' \left[\frac{U_{\delta_1 \rho_1}^{2\beta} - \alpha_2' U_{\rho_1 \sigma_2}^{1\beta}}{U_{\rho_1 \sigma_2}^{0\beta}} \right]$
Subscripts and superscripts: $\alpha_i > \beta_i, \alpha_{i+1}^j = \sin(2\pi f_i t_i) e^{-5t_i/\tau}, \dots$
Fractions, binomials and stacked numbers: $\frac{3}{4}, \binom{3}{4}, \frac{3}{4}, \left(\frac{5-\frac{1}{x}}{4}\right), \dots$
Radicals: $\sqrt{2}, \sqrt[3]{x}, \dots$
Fonts: Roman , <i>Italic</i> , Typewriter or <i>CALLIGRAPHY</i>
Accents: $\acute{a}, \grave{a}, \check{a}, \hat{a}, \tilde{a}, \grave{\alpha}, \hat{\alpha}, \tilde{\alpha}, \widehat{xyz}, \overline{xyz}, \dots$
Greek, Hebrew: $\alpha, \beta, \chi, \delta, \lambda, \mu, \Delta, \Gamma, \Omega, \Phi, \Pi, \Upsilon, \nabla, \aleph, \beth, \daleth, \beth,$
Delimiters, functions and Symbols: $\sqcup, \int, \oint, \prod, \sum, \log, \sin, \approx, \oplus, *, \propto, \infty, \partial, \Re,$

Mathtext Examples

Matplotlib's mathtext infrastructure is an independent implementation and does not require TeX or any external packages installed on your computer. See the tutorial at [Writing mathematical expressions](#).

Native TeX rendering

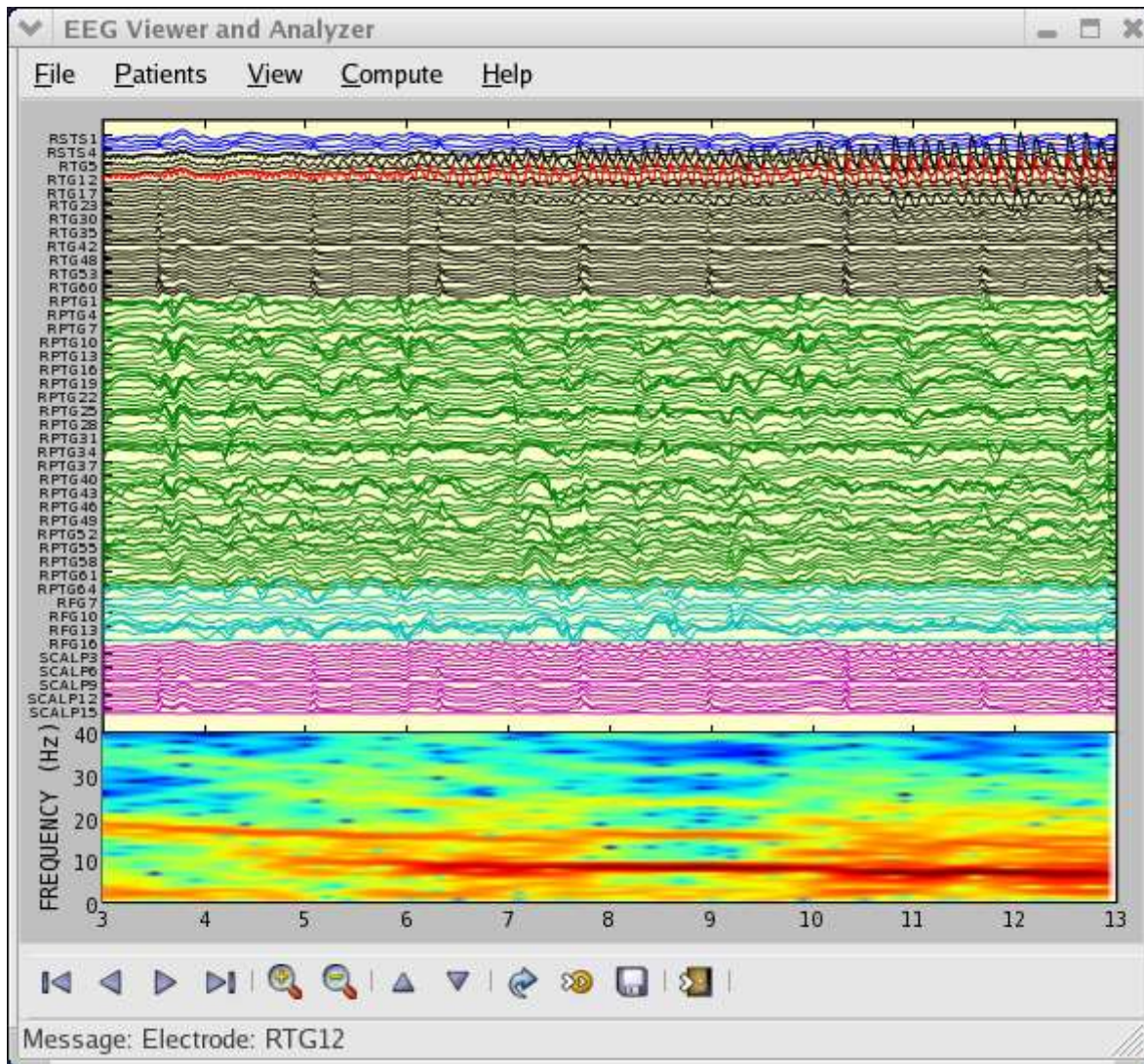
Although Matplotlib's internal math rendering engine is quite powerful, sometimes you need TeX. Matplotlib supports external TeX rendering of strings with the *usetex* option.



Tex Demo

EEG GUI

You can embed Matplotlib into pygtk, wx, Tk, or Qt applications. Here is a screenshot of an EEG viewer called [pbrain](#).



The lower axes uses `specgram()` to plot the spectrogram of one of the EEG channels.

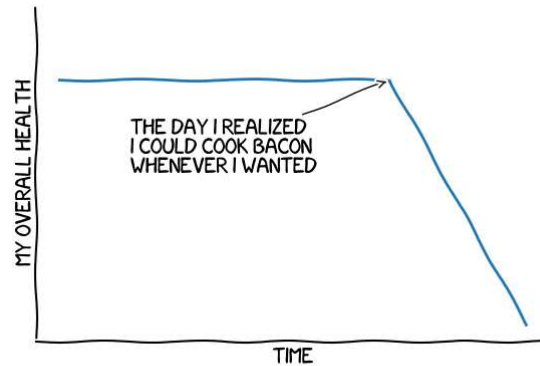
For examples of how to embed Matplotlib in different toolkits, see:

- [Embedding In GTK3](#)
- [Embedding In Wx2](#)
- [Matplotlib With Glade 3](#)
- [Embedding in Qt](#)

- [Embedding In Tk](#)

XKCD-style sketch plots

Just for fun, Matplotlib supports plotting in the style of xkcd.



"STOVE OWNERSHIP" FROM XKCD BY RANDALL MONROE

Xkcd

Subplot example

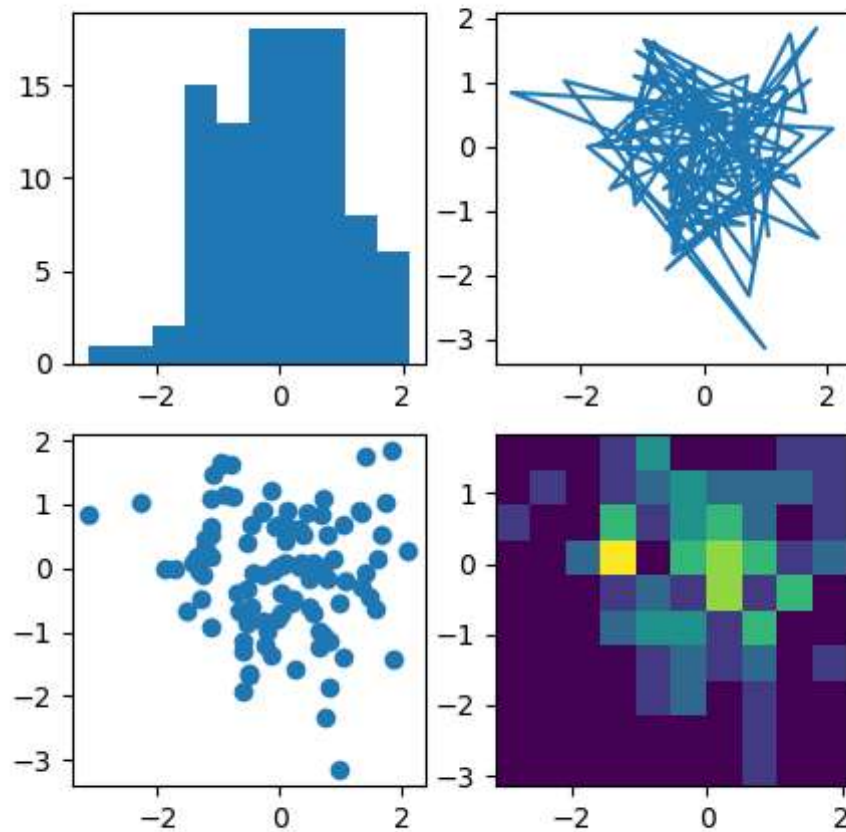
Many plot types can be combined in one figure to create powerful and flexible representations of data.

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(19680801)
data = np.random.randn(2, 100)

fig, axs = plt.subplots(2, 2, figsize=(5, 5))
axs[0, 0].hist(data[0])
axs[1, 0].scatter(data[0], data[1])
axs[0, 1].plot(data[0], data[1])
axs[1, 1].hist2d(data[0], data[1])

plt.show()
```

Download Python source code: [sample_plots.py](#)

Download Jupyter notebook: [sample_plots.ipynb](#)