Quick answers to common problems

matplotlib Plotting Cookbook

Learn how to create professional scientific plots using matplotlib, with more than 60 recipes that cover common use cases

Alexandre Devert

[PACKT] open source* PUBLISHING
community experience distilled

matplotlib Plotting Cookbook

Learn how to create professional scientific plots using matplotlib, with more than 60 recipes that cover common use cases

Alexandre Devert

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

matplotlib Plotting Cookbook

First published: March 2014

Production Reference: 1200314

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84951-326-5

www.packtpub.com

Cover Image by Artie Ng (artherng@yahoo.com.au)

Credits

Author

Alexandre Devert

Reviewers

Francesco Benincasa

Valerio Maggio

Jonathan Street

Dr. Allen Chi-Shing Yu

Acquisition Editor

Rebecca Youe

Commissioning Editor

Usha Iyer

Content Development Editor

Ankita Shashi

Technical Editors

Shubhangi Dhamgaye

Pratik More

Humera Shaikh

Copy Editors

Dipti Kapadia

Aditya Nair

Kirti Pai

Project Coordinator

Sanchita Mandal

Proofreaders

Ameesha Green

Paul Hindle

Indexer

Tejal Soni

Production Coordinator

Manu Joseph

Cover Work

Manu Joseph

About the Author

Alexandre Devert is a scientist, currently busy solving problems and making tools for molecular biologists. Before this, he used to teach data mining, software engineering, and research in numerical optimization. He is an enthusiastic Python coder as well and never gets enough of it!

I would like to thank Xiang, my amazing, wonderful wife, for her patience, support, and encouragement, as well as my parents for their support and encouragement.

About the Reviewers

Francesco Benincasa, Master of Science in Software Engineering, is a designer and developer. He is a GNU/Linux and Python expert and has vast experience in many languages and applications. He has been using Python as the primary language for more than 10 years, together with JavaScript and framewoks such as Plone or Django.

He is interested in advanced web and network developing as well as scientific data manipulation and visualization. Over the last few years, he has been using graphical Python libraries such as Matplotlib/Basemap and scientific libraries such as NumPy/SciPy, as well as scientific applications such as GrADS, NCO, and CDO.

Currently, he is working at the Earth Science Department of the Barcelona Supercomputing Center (`www.bsc.es`) as a Research Support Engineer for the World Meteorological Organization Sand and Dust Storms Warning Advisory and Assessment System (`sds-was.aemet.es`).

Valerio Maggio has a PhD in Computational Science from the University of Naples "Federico II" and is currently a Postdoc researcher at the University of Salerno.

His research interests are mainly focused on unsupervised machine learning and software engineering, recently combined with semantic web technologies for linked data and Big Data analysis.

Valerio started developing open source software in 2004, when he was studying for his Bachelor's degree. In 2006, he started working on Python, and has since contributed to several open source projects in this language. Currently, he applies Python as the mainstream language for his machine learning code, making intensive use of matplotlib to analyze experimental data.

Valerio is also a member of the Italian Python community and enjoys playing chess and drinking tea.

I wish to sincerely thank Valeria for her true love and constant support and for being the sweetest girl I've ever met.

Jonathan Street is a well-known researcher in the fields of physiology and biomarker discovery. He began using Python in 2006 and extensively used matplotlib for many figures in his PhD thesis. He shares his interest in Python data tools by giving lectures and guiding educational sessions for regional groups, as well as writing on his blog at `http://jonathanstreet.com`.

Dr. Allen Chi-Shing Yu is a postdoctoral researcher working in the field of cancer genetics. He obtained his BSc degree in Molecular Biotechnology from the Chinese University of Hong Kong in 2009, and obtained a PhD in Biochemistry from the same university in 2013. Allen's PhD research primarily involved genomic and transcriptomic characterization of novel bacterial strains that can use toxic fluoro-tryptophans but not canonical tryptophan for propagation, under the supervision of Prof. Jeffrey Tze-Fei Wong and Prof. Ting-fung Chan. The findings demonstrated that the genetic code is not an immutable construct, and a small number of analogue-sensitive proteins are stabilizing the assignment of canonical amino acids to the genetic code.

Soon after his microbial studies, Allen was involved in the identification and characterization of a novel mutation marker causing Spinocerebellar Ataxia—a group of genetically diverse neurodegenerative disorders. Through the development of a tool for detecting viral integration events in human cancer samples (ViralFusionSeq), he has entered the field of cancer genetics. As the postdoctoral researcher in Prof. Nathalie Wong's lab, he is now responsible for the high-throughput sequencing analysis of hepatocellular carcinoma, as well as the maintenance of several Linux-based computing clusters.

Allen is proficient in both wet-lab techniques and computer programming. He is also committed to developing and promoting open source technologies, through a collection of tutorials and documentations on his blog at `http://www.allenyu.info`. Readers wishing to contact Dr. Yu can do so via the contact details on his website.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

matplotlib is a Python module for plotting, and it is a component of the ScientificPython modules suite. matplotlib allows you to easily prepare professional-grade figures with a comprehensive API to customize every aspect of the figures. In this book, we will cover the different types of figures and how to adjust a figure to suit your needs. The recipes are orthogonal and you will be able to compose your own solutions very quickly.

What this book covers

Chapter 1, *First Steps*, introduces the basics of working with matplotlib. The basic figure types are introduced with minimal examples.

Chapter 2, *Customizing the Color and Styles*, covers how to control the color and style of a figure—this includes markers, line thickness, line patterns, and using color maps to color a figure several items.

Chapter 3, *Working with Annotations*, covers how to annotate a figure—this includes adding an axis legend, arrows, text boxes, and shapes.

Chapter 4, *Working with Figures*, covers how to prepare a complex figure—this includes compositing several figures, controlling the aspect ratio, axis range, and the coordinate system.

Chapter 5, *Working with a File Output*, covers output to files, either in bitmap or vector formats. Issues like transparency, resolution, and multiple pages are studied in detail.

Chapter 6, *Working with Maps*, covers plotting matrix-like data—this includes maps, quiver plots, and stream plots.

Chapter 7, *Working with 3D Figures*, covers 3D plots—this includes scatter plots, line plots, surface plots, and bar charts.

Chapter 8, *User Interface*, covers a set of user interface integration solutions, ranging from simple and minimalist to sophisticated.

What you need for this book

The examples in this book are written for Matplotlib 1.2 and Python 2.7 or 3.

Most examples rely on NumPy and SciPy. Some examples require SymPy, while some other examples require LaTeX.

Who this book is for

The book is intended for readers who have some notions of Python and a science background.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "We can include other contexts through the use of the `include` directive."

A block of code is set as follows:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,Voicemail(u100)
exten => s,102,Voicemail(b100)
exten => i,1,Voicemail(s0)
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,Voicemail(u100)
exten => s,102,Voicemail(b100)
exten => i,1,Voicemail(s0)
```

Any command-line input or output is written as follows:

```
# cp /usr/src/asterisk-addons/configs/cdr_mysql.conf.sample

  /etc/asterisk/cdr_mysql.conf
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Clicking on the **Next** button moves you to the next screen".

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Downloading the color images of this book

We also provide you a PDF file that has color images of the screenshots/diagrams used in *Chapter 1*, *First Steps*, of this book. The color images will help you better understand the changes in the output. You can download this file from `https://www.packtpub.com/sites/default/files/downloads/3265OS_Graphics.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
First Steps

In this chapter, we will cover:

- Installing matplotlib
- Plotting one curve
- Using NumPy
- Plotting multiple curves
- Plotting curves from file data
- Plotting points
- Plotting bar charts
- Plotting multiple bar charts
- Plotting stacked bar charts
- Plotting back-to-back bar charts
- Plotting pie charts
- Plotting histograms
- Plotting boxplots
- Plotting triangulations

Introduction

matplotlib makes scientific plotting very straightforward. matplotlib is not the first attempt at making the plotting of graphs easy. What matplotlib brings is a modern solution to the balance between ease of use and power. matplotlib is a module for Python, a programming language. In this chapter, we will provide a quick overview of what using matplotlib feels like. Minimalistic recipes are used to introduce the principles matplotlib is built upon.

Installing matplotlib

Before experimenting with matplotlib, you need to install it. Here we introduce some tips to get matplotlib up and running without too much trouble.

How to do it...

We have three likely scenarios: you might be using Linux, OS X, or Windows.

Linux

Most Linux distributions have Python installed by default, and provide matplotlib in their standard package list. So all you have to do is use the package manager of your distribution to install matplotlib automatically. In addition to matplotlib, we highly recommend that you install NumPy, SciPy, and SymPy, as they are supposed to work together. The following list consists of commands to enable the default packages available in different versions of Linux:

- **Ubuntu**: The default Python packages are compiled for Python 2.7. In a command terminal, enter the following command:

  ```
  sudo apt-get install python-matplotlib python-numpy python-scipy
  python-sympy
  ```

- **ArchLinux**: The default Python packages are compiled for Python 3. In a command terminal, enter the following command:

  ```
  sudo pacman -S python-matplotlib python-numpy python-scipy python-
  sympy
  ```

  If you prefer using Python 2.7, replace `python` by `python2` in the package names

- **Fedora**: The default Python packages are compiled for Python 2.7. In a command terminal, enter the following command:

  ```
  sudo yum install python-matplotlib numpy scipy sympy
  ```

There are other ways to install these packages; in this chapter, we propose the most simple and seamless ways to do it.

Windows and OS X

Windows and OS X do not have a standard package system for software installation. We have two options—using a ready-made self-installing package or compiling matplotlib from the code source. The second option involves much more work; it is worth the effort to have the latest, bleeding edge version of matplotlib installed. Therefore, in most cases, using a ready-made package is a more pragmatic choice.

You have several choices for ready-made packages: Anaconda, Enthought Canopy, Algorete Loopy, and more! All these packages provide Python, SciPy, NumPy, matplotlib, and more (a text editor and fancy interactive shells) in one go. Indeed, all these systems install their own package manager and from there you install/uninstall additional packages as you would do on a typical Linux distribution. For the sake of brevity, we will provide instructions only for Enthought Canopy. All the other systems have extensive documentation online, so installing them should not be too much of a problem.

So, let's install Enthought Canopy by performing the following steps:

1. Download the Enthought Canopy installer from `https://www.enthought.com/products/canopy`. You can choose the free Express edition. The website can guess your operating system and propose the right installer for you.
2. Run the Enthought Canopy installer. You do not need to be an administrator to install the package if you do not want to share the installed software with other users.
3. When installing, just click on **Next** to keep the defaults. You can find additional information about the installation process at `http://docs.enthought.com/canopy/quick-start.html`.

That's it! You will have Python 2.7, NumPy, SciPy, and matplotlib installed and ready to run.

Plotting one curve

The initial example of Hello World! for a plotting software is often about showing a simple curve. We will keep up with that tradition. It will also give you a rough idea about how matplotlib works.

Getting ready

You need to have Python (either v2.7 or v3) and matplotlib installed. You also need to have a text editor (any text editor will do) and a command terminal to type and run commands.

How to do it...

Let's get started with one of the most common and basic graph that any plotting software offers—**curves**. In a text file saved as `plot.py`, we have the following code:

```
import matplotlib.pyplot as plt

X = range(100)
Y = [value ** 2 for value in X]

plt.plot(X, Y)
plt.show()
```

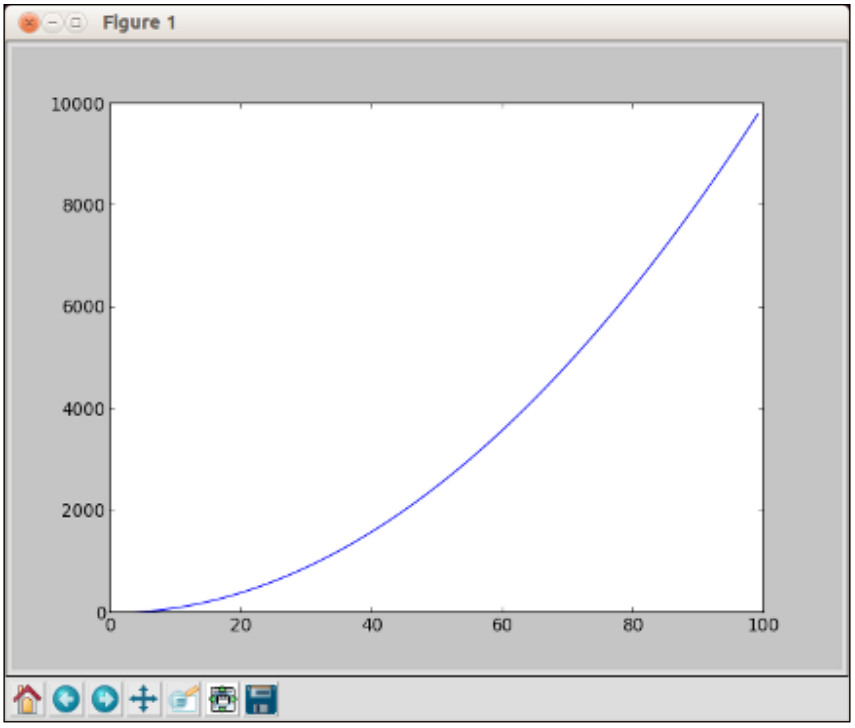
Downloading the example code

You can download the sample code files for all Packt books that you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Assuming that you installed Python and matplotlib, you can now use Python to interpret this script. If you are not familiar with Python, this is indeed a Python script we have there! In a command terminal, run the script in the directory where you saved `plot.py` with the following command:

```
python plot.py
```

Doing so will open a window as shown in the following screenshot:



The window shows the curve *Y = X ** 2* with `X` in the [0, 99] range. As you might have noticed, the window has several icons, some of which are as follows:

-  : This icon opens a dialog, allowing you to save the graph as a picture file. You can save it as a bitmap picture or a vector picture.

-  : This icon allows you to translate and scale the graphics. Click on it and then move the mouse over the graph. Clicking on the left button of the mouse will translate the graph according to the mouse movements. Clicking on the right button of the mouse will modify the scale of the graphics.
-  : This icon will restore the graph to its initial state, canceling any translation or scaling you might have applied before.

How it works...

Assuming that you are not very familiar with Python yet, let's analyze the script demonstrated in the previous section.

The first line tells Python that we are using the `matplotlib.pyplot` module. To save on a bit of typing, we make the name `plt` equivalent to `matplotlib.pyplot`. This is a very common practice that you will see in matplotlib code.

The second line creates a list named `X`, with all the integer values from 0 to 99. The range function is used to generate consecutive numbers. You can run the interactive Python interpreter and type the command `range(100)` if you use Python 2, or the command `list(range(100))` if you use Python 3. This will display the list of all the integer values from 0 to 99. In both versions, `sum(range(100))` will compute the sum of the integers from 0 to 99.

The third line creates a list named `Y`, with all the values from the list `X` squared. Building a new list by applying a function to each member of another list is a Python idiom, named **list comprehension**. The list `Y` will contain the squared values of the list `X` in the same order. So `Y` will contain 0, 1, 4, 9, 16, 25, and so on.

The fourth line plots a curve, where the x coordinates of the curve's points are given in the list `X`, and the y coordinates of the curve's points are given in the list `Y`. Note that the names of the lists can be anything you like.

The last line shows a result, which you will see on the window while running the script.

There's more...

So what we have learned so far? Unlike plotting packages like gnuplot, matplotlib is not a command interpreter specialized for the purpose of plotting. Unlike Matlab, matplotlib is not an integrated environment for plotting either. matplotlib is a Python module for plotting. Figures are described with Python scripts, relying on a (fairly large) set of functions provided by matplotlib.

Thus, the philosophy behind matplotlib is to take advantage of an existing language, Python. The rationale is that Python is a complete, well-designed, general purpose programming language. Combining matplotlib with other packages does not involve tricks and hacks, just Python code. This is because there are numerous packages for Python for pretty much any task. For instance, to plot data stored in a database, you would use a database package to read the data and feed it to matplotlib. To generate a large batch of statistical graphics, you would use a scientific computing package such as SciPy and Python's I/O modules.

Thus, unlike many plotting packages, matplotlib is very orthogonal—it does plotting and only plotting. If you want to read inputs from a file or do some simple intermediary calculations, you will have to use Python modules and some glue code to make it happen. Fortunately, Python is a very popular language, easy to master and with a large user base. Little by little, we will demonstrate the power of this approach.

Using NumPy

NumPy is not required to use matplotlib. However, many matplotlib tricks, code samples, and examples use NumPy. A short introduction to NumPy usage will show you the reason.

Getting ready

Along with having Python and matplotlib installed, you also have NumPy installed. You have a text editor and a command terminal.

How to do it...

Let's plot another curve, `sin(x)`, with x in the *[0, 2 * pi]* interval. The only difference with the preceding script is the part where we generate the point coordinates. Type and save the following script as `sin-1.py`:

```
import math
import matplotlib.pyplot as plt

T = range(100)
X = [(2 * math.pi * t) / len(T) for t in T]
Y = [math.sin(value) for value in X]

plt.plot(X, Y)
plt.show()
```

Then, type and save the following script as `sin-2.py`:

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