

Matplotlib 3.0

Cookbook

Over 150 recipes to create highly detailed interactive visualizations using Python



Srinivasa Rao Poladi

Packt[®]
www.packt.com

Matplotlib 3.0 Cookbook

Over 150 recipes to create highly detailed interactive visualizations using Python

Srinivasa Rao Poladi



BIRMINGHAM - MUMBAI

Matplotlib 3.0 Cookbook

Copyright © 2018 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Sunith Shetty
Acquisition Editor: Namrata Patil
Content Development Editor: Unnati Guha
Technical Editor: Sayli Nikalje
Copy Editor: Safis Editing
Project Coordinator: Manthan Patel
Proofreader: Safis Editing
Indexer: Rekha Nair
Graphics: Jisha Chirayil
Production Coordinator: Arvindkumar Gupta

First published: October 2018

Production reference: 1191018

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

ISBN 978-1-78913-571-8

www.packtpub.com



mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

Packt.com

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.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.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.

Contributors

About the author

Srinivasa Rao Poladi has been in the IT services industry for over two decades, providing consulting and implementation services in data warehousing, business intelligence, and machine learning areas for global customers.

He has worked with Wipro Technologies for two decades and played key leadership roles in building large technology practices and growing them to multi-million \$ business.

He spoke at international conferences, published many blogs and white papers in the areas of big data, business intelligence, and analytics.

He is a co-founder of krtrimaIQ a consulting firm that provides cognitive solutions to create tomorrow's Intelligent Enterprises powered by automation, big data, machine learning, and deep learning.

About the reviewer

Nikhil Borkar holds a CQF designation and a post-graduate degree in quantitative finance. He also holds the Certified Financial Crime Examiner and Certified Anti-Money Laundering Professional qualifications. He is a registered research analyst with the **Securities and Exchange Board of India (SEBI)** and has a keen grasp of the Indian regulatory landscape pertaining to securities and investments. He is currently working as an independent FinTech and legal consultant. Prior to this, he worked with **Morgan Stanley Capital International (MSCI)** as a Global RFP Project Manager.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface	1
Chapter 1: Anatomy of Matplotlib	7
Introduction	7
Architecture of Matplotlib	7
Backend layer	8
Artist layer	8
Scripting layer	8
Elements of a figure	8
Figure	9
Axes	10
Axis	10
Label	10
Legend	10
Title	10
Ticklabels	11
Spines	11
Grid	11
Working in interactive mode	11
Getting ready	11
How to do it...	12
How it works...	13
There's more...	14
Working in non-interactive mode	15
How to do it...	15
How it works...	16
Reading from external files and plotting	17
Getting ready	17
How to do it...	17
Reading from a .txt file	18
Reading from a .csv file	19
Reading from an .xlsx file	19
Plotting the graph	20
How it works...	20
Changing and resetting default environment variables	21
Getting ready	21
How to do it...	22
How it works...	23
There's more...	23
Chapter 2: Getting Started with Basic Plots	24
Introduction	25

Line plot	25
Getting ready	25
How to do it...	25
How it works...	26
There's more...	27
Bar plot	29
Getting ready	29
How to do it...	29
How it works...	30
There's more...	31
Scatter plot	32
Getting ready	32
How to do it...	33
How it works...	33
There's more...	34
Bubble plot	36
Getting ready	36
How to do it...	37
How it works...	37
Stacked plot	38
Getting ready	38
How to do it...	39
How it works...	39
Pie plot	41
Getting ready	41
How to do it...	41
How it works...	42
Table chart	43
Getting ready	43
How to do it...	43
How it works...	45
Polar plot	46
Getting ready	46
How to do it...	47
How it works...	48
There's more...	49
Histogram	49
Getting ready	50
How to do it...	50
How it works...	51
There's more...	52
Box plot	53
Getting ready	54
How to do it...	54
How it works...	55

There's more...	55
Violin plot	56
Getting ready	56
How to do it...	56
How it works...	57
Reading and displaying images	58
Getting ready	58
How to do it...	58
How it works...	58
Heatmap	59
Getting ready	59
How to do it...	60
How it works...	60
Hinton diagram	62
Getting ready	62
How to do it...	62
How it works...	63
Contour plot	64
Getting ready	64
How to do it...	65
How it works...	66
There's more...	67
Triangulations	68
Getting ready	68
How to do it...	69
How it works...	69
There's more...	70
Stream plot	71
Getting ready	71
How to do it...	72
How it works...	72
There's more...	73
Path	74
Getting ready	74
How to do it...	75
How it works...	76
Chapter 3: Plotting Multiple Charts, Subplots, and Figures	78
Introduction	78
Plotting multiple graphs on the same axes	78
Getting ready	79
How to do it...	79
How it works...	80
Plotting subplots on the same figure	81
Getting ready	81

How to do it...	82
How it works...	83
There's more...	84
Plotting multiple figures in a session	87
Getting ready	87
How to do it...	87
How it works...	89
There's more...	91
Logarithmic scale	91
Getting ready	91
How to do it...	91
How it works...	92
There's more...	93
Using units of measurement	95
Getting ready	96
How to do it...	96
How it works...	97
There's more...	99
Chapter 4: Developing Visualizations for Publishing Quality	102
Introduction	102
Color, line style, and marker customization	103
Getting ready	103
How to do it...	103
How it works...	105
Working with standard colormaps	108
Getting ready	108
How to do it...	108
How it works...	109
There's more...	110
User-defined colors and colormaps	113
Getting ready	113
How to do it...	114
How it works...	115
There's more...	117
Working with legend	119
Getting ready	119
How to do it...	119
How it works...	122
There's more...	124
Customizing labels and titles	125
Getting ready	125
How to do it...	125
How it works...	127
There's more...	128

Using autoscale and axis limits	128
Getting ready	128
How to do it...	129
How it works...	130
Customizing ticks and ticklabels	131
Getting ready	131
How to do it...	131
How it works...	132
There's more...	133
Customizing spines	135
Getting ready	135
How to do it...	136
How it works...	137
Twin axes	138
Getting ready	138
How to do it...	139
How it works...	140
There's more...	143
Using hatch	145
Getting ready	145
How to do it...	145
How it works...	145
Using annotation	146
Getting ready	146
How to do it...	146
How it works...	147
Using style sheets	148
Getting ready	148
How to do it...	148
How it works...	149
There's more...	150
Chapter 5: Plotting with Object-Oriented API	154
Introduction	154
Plotting a correlation matrix using pyplot and object-oriented APIs	155
Getting ready	155
How to do it...	156
How it works...	158
Plotting patches using object-oriented API	160
Getting ready	161
How to do it...	161
How it works...	162
Plotting collections using object-oriented API	163
Getting ready	163
How to do it...	164

How it works...	165
Chapter 6: Plotting with Advanced Features	167
Using property cycler	167
Getting ready	168
How to do it...	168
How it works...	169
There's more...	170
Using Path effects	171
Getting ready	171
How to do it...	171
How it works...	172
There's more...	173
Using transforms	174
Transforming data co-ordinates to display co-ordinates	174
Getting ready	175
How to do it...	175
How it works...	176
There's more...	178
Using axes and blended co-ordinate system transforms	178
Getting ready	178
How to do it...	178
How it works...	180
Taking control of axes positions	182
Getting ready	182
How to do it...	182
How it works...	184
GridSpec for figure layout	185
Using GridSpec	185
Getting ready	186
How to do it...	186
How it works...	188
There's more...	189
GridSpec alignment	191
Getting ready	191
How to do it...	191
How it works ...	193
Constrained layout	196
Getting ready	196
How to do it...	196
How it works...	198
Using GridSpecFromSubplotSpec	200
Getting ready	200
How to do it...	200
How it works...	201
Using origin and extent for image orientation	202
Getting ready	203

How to do it...	203
How it works...	204
Geographical plotting using geopandas	205
Getting ready	206
How to do it...	206
How it works...	207
Chapter 7: Embedding Text and Expressions	209
Introduction	209
Using mathematical expressions with a font dictionary	210
Getting ready	210
How to do it...	210
How it works...	211
Annotating a point on a polar plot	212
Getting ready	212
How to do it...	212
How it works...	213
Using ConnectionPatch	215
Getting ready	215
How to do it...	215
How it works...	216
Using a text box	218
Getting ready	218
How to do it...	218
How it works...	219
There's more...	221
Plotting area under an integral curve	223
Getting ready	223
How to do it...	224
How it works...	225
Defining custom markers	226
Getting ready	226
How to do it...	227
How it works...	227
Fractions, regular mathematical expressions, and symbols	228
Getting ready	229
How to do it...	229
How it works...	231
There's more...	233
Word embeddings in two dimensions	235
Getting ready	236
How to do it...	236
How it works...	237
Chapter 8: Saving the Figure in Different Formats	240
Introduction	240

Saving the figure in various formats	240
Getting ready	241
How to do it...	241
How it works...	241
There's more...	243
Avoiding truncation while saving the figure	244
Getting ready	244
How to do it...	244
How it works...	245
Saving partial figures	250
Getting ready	250
How to do it...	251
How it works...	252
Managing image resolution	255
Getting ready	255
How to do it...	255
How it works...	256
Managing transparency for web applications	258
Getting ready	258
How to do it...	258
How it works...	260
Creating multi-page PDF reports	262
Getting ready	263
How to do it...	263
How it works...	266
Chapter 9: Developing Interactive Plots	271
Introduction	272
Events and callbacks	273
Exception handling	273
Getting ready	273
How to do it...	274
How it works...	275
There's more...	277
Key press and release events	277
Getting ready	278
How to do it...	278
How it works...	279
Mouse button press event	282
Getting ready	282
How to do it...	282
How it works...	283
Motion notify and mouse button press events	286
Getting ready	286
How to do it...	286
How it works...	288
Pick event	290

Getting ready	290
How to do it...	291
How it works...	292
Figure and axes, enter and leave events	295
Getting ready	295
How to do it...	295
How it works...	296
Using twin axes for plotting four temperature scales	299
Getting ready	300
How to do it...	300
How it works...	302
Widgets	304
Cursor	304
Getting ready	305
How to do it...	305
How it works...	305
Buttons	306
Getting ready	306
How to do it...	307
How it works...	308
Check buttons	312
Getting ready	312
How to do it...	313
How it works...	314
Radio buttons	317
Getting ready	317
How to do it...	318
How it works...	319
Textbox	321
Getting ready	322
How to do it...	322
How it works...	323
Animation	326
Animated sigmoid curve	327
Getting ready	327
How to do it...	327
How it works...	328
Saving the animation to an mp4 file	330
Getting ready	330
How to do it...	330
How it works...	331
Exponentially decaying tan function	331
Getting ready	331
How to do it...	331
How it works...	333
Animated bubble plot	334
Getting ready	334
How to do it...	335
How it works...	336

Animating multiple line plots	338
Getting ready	338
How to do it...	338
How it works...	339
Animation of images	340
Getting ready	340
How to do it...	341
How it works...	341
Chapter 10: Embedding Plots in a Graphical User Interface	342
Introduction	342
Interface between the Matplotlib and GUI applications	343
Using the Slider and Button Widgets of Matplotlib	345
Getting ready	345
How to do it...	345
How it works...	347
Using the Slider and Button widgets of Tkinter GUI	349
Getting ready	350
How to do it...	350
How it works...	351
Embedding Matplotlib in a Tkinter GUI application	354
Getting ready	355
How to do it...	355
How it works...	356
Using the Slider and Button widgets of WxPython GUI	359
Getting ready	360
How to do it...	360
How it works...	361
Embedding Matplotlib in to a wxPython GUI application	365
Getting ready	366
How to do it...	366
How it works...	368
Using the Slider and Button widgets of Qt's GUI	371
Getting ready	372
How to do it...	372
How it works...	374
Embedding Matplotlib in to a Qt GUI application	377
Getting ready	377
How to do it...	377
How it works...	379
Chapter 11: Plotting 3D Graphs Using the mplot3d Toolkit	383
Introduction	383
Line plot	384
Getting ready	384
How to do it...	384

How it works...	386
Scatter plot	387
Getting ready	388
How to do it...	388
How it works...	389
Bar plot	390
Getting ready	391
How to do it...	391
How it works...	392
Polygon plot	393
Getting ready	394
How to do it...	394
How it works...	396
There's more...	397
Contour plot	397
Getting ready	398
How to do it...	398
How it works...	399
There's more...	400
Surface plot	400
Getting ready	401
How to do it...	401
How it works...	402
Wireframe plot	403
Getting ready	403
How to do it...	403
How it works...	404
Triangular surface plot	406
Getting ready	406
How to do it...	406
How it works...	407
Plotting 2D data in 3D	409
Getting ready	409
How to do it...	409
How it works...	410
3D visualization of linearly non-separable data in 2D	411
Getting ready	412
How to do it...	412
How it works...	414
Word embeddings	415
Getting ready	415
How to do it...	416
How it works...	416
Chapter 12: Using the axisartist Toolkit	418

Introduction	418
Understanding attributes in axisartist	418
Getting ready	419
How to do it...	419
How it works...	420
Defining curvilinear grids in rectangular boxes	424
Getting ready	424
How to do it...	424
How it works...	426
Defining polar axes in rectangular boxes	428
Getting ready	429
How to do it...	429
How it works...	431
Using floating axes for a rectangular plot	434
Getting ready	434
How to do it...	434
How it works...	437
Creating polar axes using floating axes	440
Getting ready	440
How to do it...	441
How it works...	442
Plotting planetary system data on floating polar axes	443
Getting ready	444
How to do it...	444
How it works...	447
Chapter 13: Using the axes_grid1 Toolkit	448
Introduction	448
Plotting twin axes using the axisartist and axesgrid1 toolkits	448
Getting ready	449
How to do it...	449
How it works...	451
There's more...	453
Using AxesDivider to plot a scatter plot and associated histograms	455
Getting ready	456
How to do it...	456
How it works...	457
Using AxesDivider to plot a colorbar	459
Getting ready	460
How to do it...	460
How it works...	461
Using ImageGrid to plot images with a colorbar in a grid	463
Getting ready	464
How to do it...	464
How it works...	465

Using inset_locator to zoom in on an image	468
Getting ready	468
How to do it...	469
How it works...	470
Using inset_locator to plot inset axes	472
Getting ready	473
How to do it...	473
How it works...	474
Chapter 14: Plotting Geographical Maps Using Cartopy Toolkit	477
Introduction	477
Plotting basic map features	478
Getting ready	478
How to do it...	478
How it works...	479
Plotting projections	481
Getting ready	481
How to do it...	482
How it works...	483
Using grid lines and labels	484
Getting ready	485
How to do it...	485
How it works...	488
Plotting locations on the map	492
Getting ready	493
How to do it...	493
How it works...	495
Plotting country maps with political boundaries	497
Getting ready	498
How to do it...	498
How it works...	501
Plotting country maps using GeoPandas and cartopy	504
Getting ready	505
How to do it...	505
How it works...	507
Plotting populated places of the world	509
Getting ready	509
How to do it...	509
How it works...	510
Plotting the top five and bottom five populated countries	511
Getting ready	511
How to do it...	512
How it works...	514
Plotting temperatures across the globe	516
Getting ready	516

How to do it...	516
How it works...	517
Plotting time zones	518
Getting ready	519
How to do it...	519
How it works...	520
Plotting an animated map	522
Getting ready	522
How to do it...	522
How it works...	523
Chapter 15: Exploratory Data Analysis Using the Seaborn Toolkit	525
Introduction	526
Snacks Sales dataset	526
Wine Quality	527
Semantic and facet variables	528
Relational plots	529
Line plots with one-to-one and one-to-many relationships	529
Getting ready	530
How to do it...	530
How it works...	531
There's more...	534
Line plots with a long-form dataset	537
Getting ready	538
How to do it...	539
How it works...	539
Scatter plots	541
Getting ready	542
How to do it...	542
How it works...	543
There's more...	545
Categorical plots	547
Strip and swarm plots	547
Getting ready	548
How to do it...	548
How it works...	549
Box and boxn plots	551
Getting ready	552
How to do it...	552
How it works...	553
Bar and count plots	556
Getting ready	556
How to do it...	557
How it works...	559
Violin plots	564
Getting ready	564
How to do it...	564
How it works...	566

Point plots	569
Getting ready	569
How to do it...	570
How it works...	571
Distribution plots	573
distplot()	574
Getting ready	574
How to do it...	574
How it works...	575
kdeplot()	578
Getting ready	579
How to do it...	579
How it works...	579
Regression plots	581
regplot() and residplot()	581
Getting ready	581
How to do it...	582
How it works...	583
lmpplot()	586
Getting ready	586
How to do it...	587
How it works...	587
Multi-plot grids	589
jointplot() and JointGrid()	590
Getting ready	590
jointplot()	591
How to do it...	591
How it works...	592
JointGrid()	598
How to do it...	598
How it works...	601
pairplot() and PairGrid()	609
Getting ready	609
pairplot()	610
How to do it...	610
How it works...	611
PairGrid()	615
How to do it...	615
How it works...	616
FacetGrid()	620
Getting ready	620
How to do it...	620
How it works...	621
Matrix plots	626
Heatmaps	627
Getting ready	627
How to do it...	627
How it works...	629
Clustermats	633

Table of Contents

Getting ready	633
How to do it...	633
How it works...	633
Other Books You May Enjoy	637
Index	640

Preface

In the era of big data, finding valuable business insights is akin to finding a needle in a haystack. Visualization plays a critical role in finding those nuggets from an ever-increasing volume and variety of data. Matplotlib, with its rich visualization functionality, makes the process of exploratory data analysis user friendly and more productive.

Matplotlib's core functionality is vast, and it is further enhanced by many in-house and third-party toolkits. Many of the books on the market cover only a small portion of its complete functionality. In this book, we have covered Matplotlib's complete core functionality and many of its popular toolkits.

Matplotlib is popular among machine learning practitioners and researchers who use the Python ecosystem. With its rich functionality, it can be used in business intelligence and operational reporting applications. In this book, we have made an attempt to present examples from these applications.

While a recipe-based cookbook approach makes this book a reference guide for quick solutions, we have covered sufficient theoretical background to make it easy for beginners as well.

Who this book is for

This book is for data analysts, business analysts, data scientists, and Python developers who are looking for quick solutions for a wide variety of visualization applications, such as ad hoc reports, professional dashboards, exploratory data analysis, interactive analysis, embedded visualizations in selected GUI toolkits and web applications, three-dimensional plots, and geographical maps.

Those who are interested in developing business intelligence, machine learning, scientific, or engineering applications will also benefit from the recipes that are relevant for each of these disciplines.

What this book covers

Chapter 1, *Anatomy of Matplotlib*, explains the architecture of Matplotlib, various elements of a figure, interactive and non-interactive modes of operation, and how to customize environmental parameters.

Chapter 2, *Getting Started with Basic Plots*, introduces many types of graph that are commonly used in business intelligence and machine learning applications, including line, scatter, bar, stacked, histogram, box, violin, contour plots, heatmaps, and Hinton diagrams.

Chapter 3, *Plotting Multiple Graphs, Subplots, and Figures*, shows how to organize graphs into subplots and figures.

Chapter 4, *Developing Visualizations for Publishing Quality*, illustrates how to customize various attributes of a figure, including color, fonts, labels, titles, legend, spines, styles, markers, and annotation.

Chapter 5, *Plotting with the Object-Oriented API*, introduces the object-oriented API and compares it with the pyplot API. The object-oriented API gives flexibility in designing complex dashboards as required, but requires Python programming experience if you want to write code. The pyplot API comes with pre-packaged graphs that require simple commands to plot, without needing to write much Python code.

Chapter 6, *Plotting with Advanced Features*, covers how to develop complex visualization applications by using the advanced customization of legends, artist, and layout, as well as cycling object properties, origin and extent in images, transforms, animations, event handling, and path effects.

Chapter 7, *Embedding Text and Expressions*, covers how to add text to plots with regular text, annotations and mathematical expressions.

Chapter 8, *Saving the Figure in Different Formats*, explains how to save figures to external output files in PNG, PDF, SVG, and PS formats.

Chapter 9, *Developing Interactive Plots*, explains how to develop interactive plots using event handling, animations, and widgets. These features enable the users to perform interactive analysis.

Chapter 10, *Embedding Plots in Graphical User Interface*, explains how to embed Matplotlib plots into other graphical user interfaces used for developing applications.

Chapter 11, *Plotting 3D Graphs Using the mplot3d Toolkit*, covers how to use the mplot3D toolkit to plot 3D graphs, and the next two chapters cover two more toolkits.

Chapter 12, *Using the axisartist Toolkit*, explains that while the standard Matplotlib axes uses a traditional Cartesian coordinate system, it can't handle special features such as curved or floating axes that are useful in plotting geographical or planetary systems. This chapter explains how to create special applications using the axisartist toolkit.

Chapter 13, *Using the axes_grid1 Toolkit*, covers the `axes_grid1` toolkit. This toolkit enables you to plot images in a grid with an associated color bar that aligns well with the image and also enables anchor images as legends, zoom in/out effects, and more.

Chapter 14, *Plotting Geographical Maps Using the Cartopy Toolkit*, explains wide variety of features that cater to many different user communities. We will cover most of the features typically used in business applications.

Chapter 15, *Exploratory Data Analysis Using the Seaborn Toolkit*, explains the process of exploratory data analysis using exhaustive features of seaborn toolkit.

To get the most out of this book

Basic knowledge of Python is enough to understand the content in this book, except for Chapters 9, *Developing Interactive Plots* and Chapter 10, *Embedding Plots in a Graphical User Interface*. These two chapters deal with interactive plotting and embedded applications that need medium-level Python programming experience.

Many Python distributions automatically include Matplotlib, along with all its dependencies. If you have not installed any standard Python distributions, you can follow the installation process at <https://matplotlib.org/users/installing.html> to install Matplotlib and its associated dependencies.

Download the example code files

You can download the example code files for this book from your account at www.packt.com. If you purchased this book elsewhere, you can visit www.packt.com/support and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at www.packt.com.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at <https://github.com/PacktPublishing/Matplotlib-3.0-Cookbook>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: https://www.packtpub.com/sites/default/files/downloads/9781789135718_ColorImages.pdf.

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "We will follow the order of `.txt`, `.csv`, and `.xlsx` files, in three separate sections."

A block of code is set as follows:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from matplotlib import cm
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "When you run the program and click **Next** and **Next**, you will see the following three figures, representing each of the clusters, as shown in the header of each figure."



Warnings or important notes appear like this.



Tips and tricks appear like this.

Sections

In this book, you will find several headings that appear frequently (*Getting ready*, *How to do it...*, *How it works...*, *There's more...*, and *See also*).

To give clear instructions on how to complete a recipe, use these sections as follows:

Getting ready

This section tells you what to expect in the recipe and describes how to set up any software or any preliminary settings required for the recipe.

How to do it...

This section contains the steps required to follow the recipe.

How it works...

This section usually consists of a detailed explanation of what happened in the previous section.

There's more...

This section consists of additional information about the recipe in order to make you more knowledgeable about the recipe.

See also

This section provides helpful links to other useful information for the recipe.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packt.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packt.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packt.com.

1 Anatomy of Matplotlib

This chapter begins with an introduction to Matplotlib, including the architecture of Matplotlib and the elements of a figure, followed by the recipes. The following are the recipes that will be covered in this chapter:

- Working in interactive mode
- Working in non-interactive mode
- Reading from external files and plotting
- How to change and reset default environment variables

Introduction

Matplotlib is a cross-platform Python library for plotting two-dimensional graphs (also called **plots**). It can be used in a variety of user interfaces such as Python scripts, IPython shells, Jupyter Notebooks, web applications, and GUI toolkits. It can be used to develop professional reporting applications, interactive analytical applications, complex dashboard applications or embed into web/GUI applications. It supports saving figures into various hard-copy formats as well. It also has limited support for three-dimensional figures. It also supports many third-party toolkits to extend its functionality.



Please note that all the examples in this book are tested with Matplotlib 3.0 and Jupyter Notebook 5.1.0.

Architecture of Matplotlib

Matplotlib has a three-layer architecture: **backend**, **artist**, and **scripting**, organized logically as a stack. Scripting is an API that developers use to create the graphs. Artist does the actual job of creating the graph internally. Backend is where the graph is displayed.

Backend layer

This is the bottom-most layer where the graphs are displayed on to an output device. This can be any of the user interfaces that Matplotlib supports. There are two types of backends: **user interface backends** (for use in `pygtk`, `wxpython`, `tkinter`, `qt4`, or `macosx`, and so on, also referred to as **interactive backends**) and **hard-copy** backends to make image files (`.png`, `.svg`, `.pdf`, and `.ps`, also referred to as **non-interactive backends**). We will learn how to configure these backends in later Chapter 9, *Developing Interactive Plots* and Chapter 10, *Embedding Plots in a Graphical User Interface*.

Artist layer

This is the middle layer of the stack. Matplotlib uses the `artist` object to draw various elements of the graph. So, every element (see elements of a figure) we see in the graph is an artist. This layer provides an **object-oriented API** for plotting graphs with maximum flexibility. This interface is meant for seasoned Python programmers, who can create complex dashboard applications.

Scripting layer

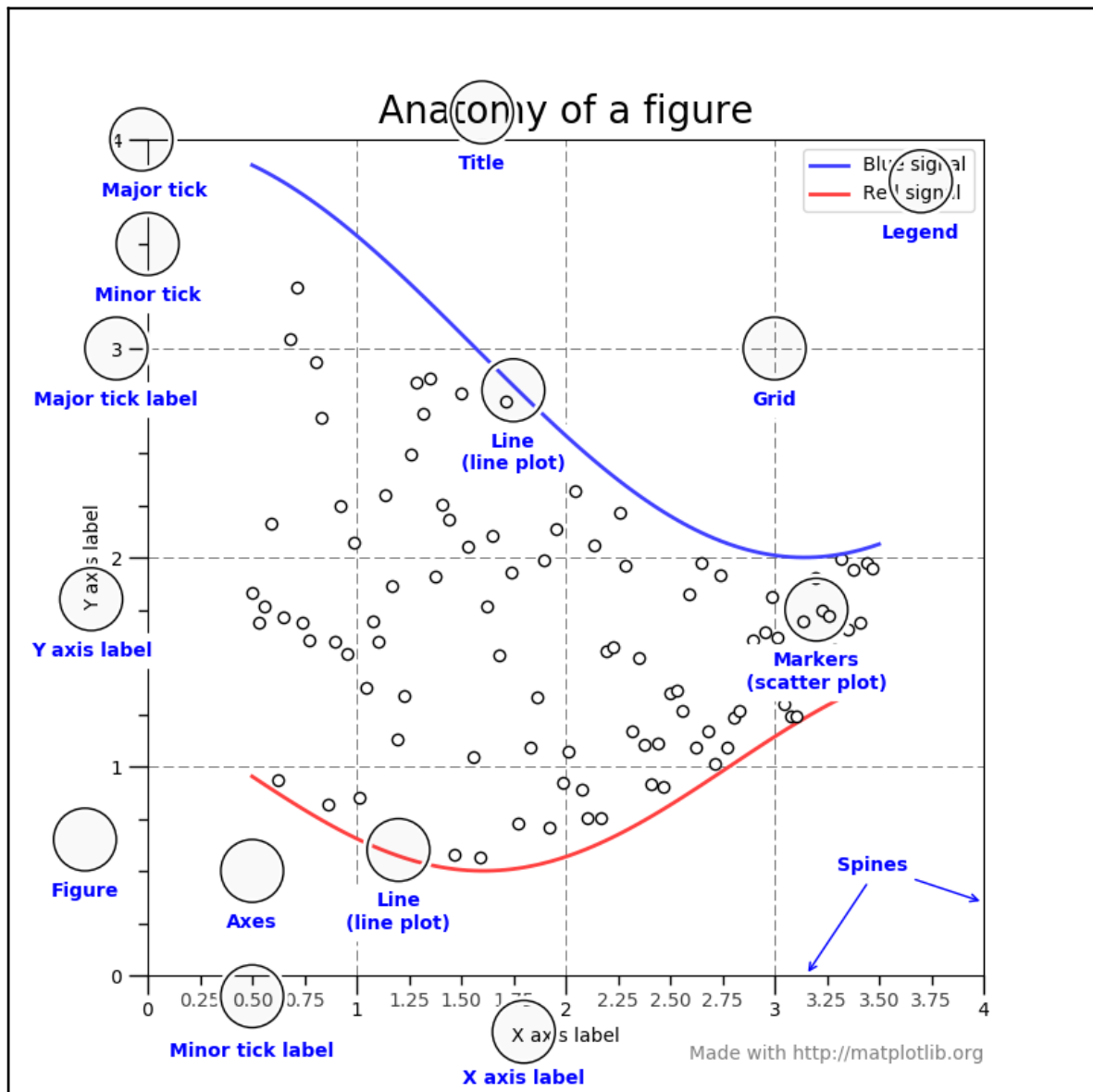
This is the topmost layer of the stack. This layer provides a simple interface for creating graphs. This is meant for use by end users who don't have much programming expertise. This is called a `pypplot` API.

Elements of a figure

The high-level Matplotlib object that contains all the elements of the output graph is called a `figure`. Multiple graphs can be arranged in different ways to form a figure. Each of the figure's elements is customizable.

Figure

The following diagram is the anatomy of a figure, containing all its elements:



Anatomy of a figure (Source : <http://diagramss.us/plotting-a-graph-in-matlab.html>)

Axes

`axes` is a sub-section of the figure, where a graph is plotted. `axes` has a **title**, an **x-label** and a **y-label**. A `figure` can have many such `axes`, each representing one or more graphs. In the preceding figure, there is only one `axes`, two line graphs in blue and red colors.

Axis

These are number lines representing the scale of the graphs being plotted. Two-dimensional graphs have an x axis and a y axis, and three-dimensional graphs have an x axis, a y axis, and a z axis.



Don't get confused between `axes` and `axis`. `Axis` is an element of `axes`. Grammatically, `axes` is also the plural for `axis`, so interpret the meaning of `axes` depending on the context, whether multiple `axis` elements are being referred to or an `axes` object is being referred to.

Label

This is the name given to various elements of the figure, for example, x axis label, y axis label, graph label (blue signal/red signal in the preceding figure *Anatomy of a figure*), and so on.

Legend

When there are multiple graphs in the `axes` (as in the preceding figure *Anatomy of a figure*), each of them has its own label, and all these labels are represented as a legend. In the preceding figure, the legend is placed at the top-right corner of the figure.

Title

It is the name given to each of the `axes`. The `figure` also can have its own title, when the figure has multiple `axes` with their own titles. The preceding figure has only one `axes`, so there is only one title for the `axes` as well as the figure.

Ticklabels

Each axis (x , y , or z) will have a range of values that are divided into many equal bins. Bins are chosen at two levels. In the preceding figure *Anatomy of a figure*, the x axis scale ranges from 0 to 4, divided into four major bins (0-1, 1-2, 2-3, and 3-4) and each of the major bins is further divided into four minor bins (0-0.25, 0.25-0.5, and 0.5-0.75). Ticks on both sides of major bins are called **major ticks** and minor bins are called **minor ticks**, and the names given to them are **major ticklabels** and **minor ticklabels**.

Spines

Boundaries of the figure are called **spines**. There are four spines for each axes (top, bottom, left, and right).

Grid

For easier readability of the coordinates of various points on the graph, the area of the graph is divided into a grid. Usually, this grid is drawn along major ticks of the x and y axis. In the preceding figure, the grid is shown in dashed lines.

Working in interactive mode

Matplotlib can be used in an **interactive** or **non-interactive** modes. In the interactive mode, the graph display gets updated after each statement. In the non-interactive mode, the graph does not get displayed until explicitly asked to do so.

Getting ready

You need working installations of Python, NumPy, and Matplotlib packages.

Using the following commands, interactive mode can be set on or off, and also checked for current mode at any point in time:

- `matplotlib.pyplot.ion()` to set the interactive mode ON
- `matplotlib.pyplot.ioff()` to switch OFF the interactive mode
- `matplotlib.is_interactive()` to check whether the interactive mode is ON (True) or OFF (False)

How to do it...

Let's see how simple it is to work in interactive mode:

1. Set the screen output as the backend:

```
%matplotlib inline
```

2. Import the `matplotlib` and `pyplot` libraries. It is common practice in Python to import libraries with crisp synonyms. Note `plt` is the synonym for the `matplotlib.pyplot` package:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

3. Set the interactive mode to ON:

```
plt.ion()
```

4. Check the status of interactive mode:

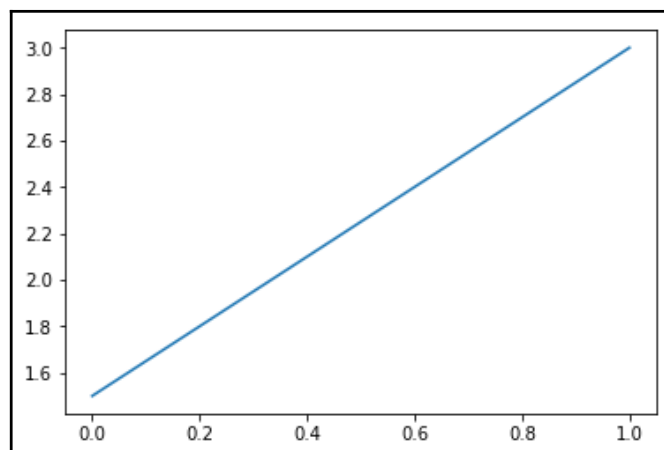
```
mpl.is_interactive()
```

5. You should get the output as `True`.

6. Plot a line graph:

```
plt.plot([1.5, 3.0])
```

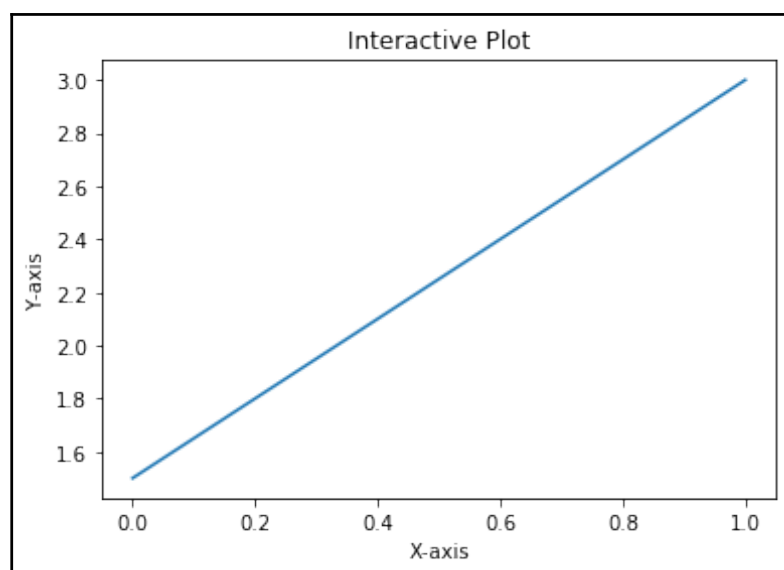
You should see the following graph as the output:



7. Now add the axis labels and a title to the graph with the help of the following code:

```
# Add labels and title
plt.title("Interactive Plot") #Prints the title on top of graph
plt.xlabel("X-axis")         # Prints X axis label as "X-axis"
plt.ylabel("Y-axis")         # Prints Y axis label as "Y-axis"
```

After executing the preceding three statements, your graph should look as follows:



How it works...

So, this is how the explanation goes:

- `plt.plot([1.5, 3.0])` plots a line graph connecting two points (0, 1.5) and (1.0, 3.0).
- The `plot` command expects two arguments (Python list, NumPy array or pandas DataFrame) for the x and y axis respectively.
- If only one argument is passed, it takes it as y axis co-ordinates and for x axis co-ordinates it takes the length of the argument provided.
- In this example, we are passing only one list of two points, which will be taken as y axis coordinates.

- For the x axis, it takes the default values in the range of 0 to 1, since the length of the list `[1.5, 3.0]` is 2.
- If we had three coordinates in the list for y , then for x , it would take the range of 0 to 2.
- You should see the graph like the one shown in *step 6*.
- `plt.title("Interactive Plot")`, prints the title on top of the graph as **Interactive Plot**.
- `plt.xlabel("X-axis")`, prints the x axis label as **X-axis**.
- `plt.ylabel("Y-axis")`, prints the y axis label as **Y-axis**.
- After executing preceding three statements, you should see the graph as shown in *step 7*.

If you are using Python shell, after executing each of the code statements, you should see the graph getting updated with title first, then the x axis label, and finally the y axis label.

If you are using Jupyter Notebook, you can see the output only after all the statements in a given cell are executed, so you have to put each of these three statements in separate cells and execute one after the other, to see the graph getting updated after each code statement.



In older versions of Matplotlib or certain backends (such as `macosx`), the graph may not be updated immediately. In such cases, you need to call `plt.draw()` explicitly at the end, so that the graph gets displayed.

There's more...

You can add one more line graph to the same plot and go on until you complete your interactive session:

1. Plot a line graph:

```
plt.plot([1.5, 3.0])
```

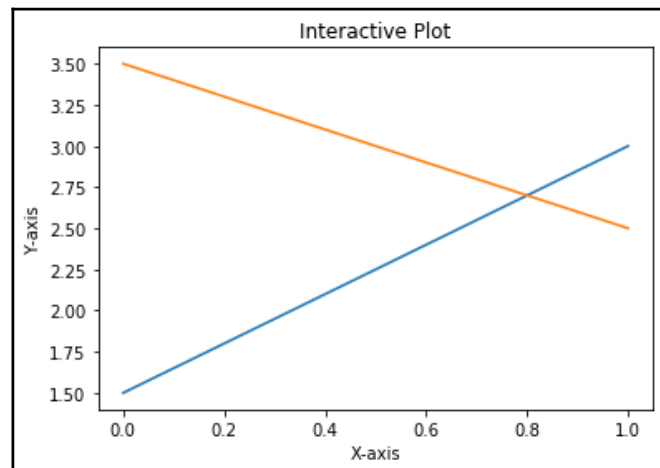
2. Add labels and title:

```
plt.title("Interactive Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
```

3. Add one more line graph:

```
plt.plot([3.5, 2.5])
```

The following graph is the output obtained after executing the code:



Hence, we have now worked in interactive mode.

Working in non-interactive mode

In the interactive mode, we have seen the graph getting built step by step with each instruction. In non-interactive mode, you give all instructions to build the graph and then display the graph with a command explicitly.

How to do it...

Working on non-interactive mode won't be difficult either:

1. Start the kernel afresh, and import the `matplotlib` and `pyplot` libraries:

```
import matplotlib
import matplotlib.pyplot as plt
```

2. Set the interactive mode to OFF:

```
plt.ioff()
```

3. Check the status of interactive mode:

```
matplotlib.is_interactive()
```

4. You should get the output `False`.
5. Execute the following code; you will not see the plot on your screen:

```
# Plot a line graph
plt.plot([1.5, 3.0])

# Plot the title, X and Y axis labels
plt.title("Non Interactive Mode")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
```

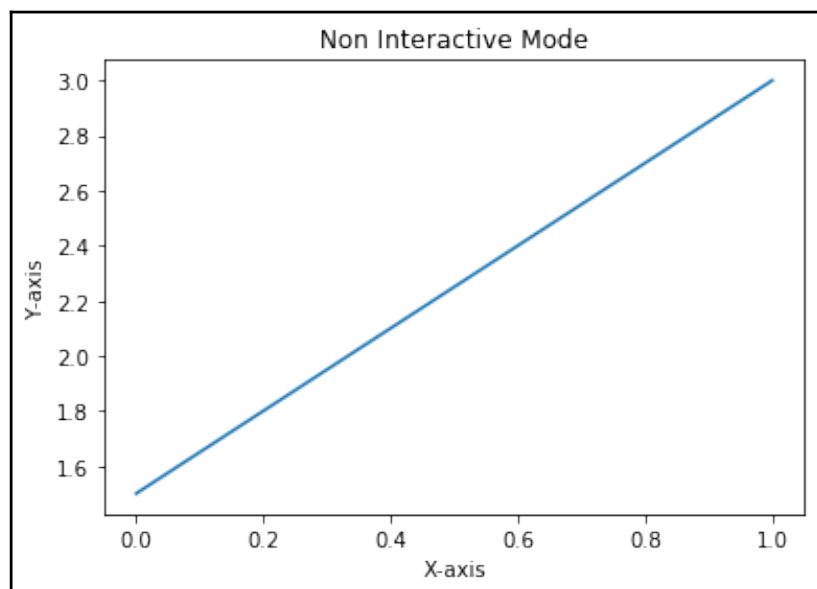
6. Execute the following statement, and then you will see the plot on your screen:

```
# Display the graph on the screen
plt.show()
```

How it works...

Each of the preceding code statements is self-explanatory. The important thing to note is in non-interactive mode, you write complete code for the graph you want to display, and call `plt.show()` explicitly to display the graph on the screen.

The following is the output obtained:





The latest versions of Jupyter Notebook seem to display the figure without calling `plt.show()` command explicitly. However, in Python shell or embedded applications, `plt.show()` or `plt.draw()` is required to display the figure on the screen.

Reading from external files and plotting

By default, Matplotlib accepts input data as a Python list, NumPy array, or pandas DataFrame. So all external data needs to be read and converted to one of these formats before feeding it to Matplotlib for plotting the graph. From a performance perspective, NumPy format is more efficient, but for default labels, pandas format is convenient.

If the data is a `.txt` file, you can use NumPy function to read the data and put it in NumPy arrays. If the data is in `.csv` or `.xlsx` formats, you can use pandas to read the data. Here we will demonstrate how to read `.txt`, `.csv`, and `.xlsx` formats and then plot the graph.

Getting ready

Import the `matplotlib.pyplot`, `numpy`, and `pandas` packages that are required to read the input files:

1. Import the `pyplot` library with the `plt` synonym:

```
import matplotlib.pyplot as plt
```

2. Import the `numpy` library with the `np` synonym. The `numpy` library can manage n-dimensional arrays, supporting all mathematical operations on these arrays:

```
import numpy as np
```

3. Import the `pandas` package with `pd` as a synonym:

```
import pandas as pd
```

How to do it...

We will follow the order of `.txt`, `.csv`, and `.xlsx` files, in three separate sections.

Reading from a .txt file

Here are some steps to follow:

1. Read the text file into the `txt` variable:

```
txt = np.loadtxt('test.txt', delimiter = ',')
txt
```

Here is the explanation for the preceding code block:

- The `test.txt` text file has 10 numbers separated by a comma, representing the x and y coordinates of five points (1, 1), (2, 4), (3, 9), (4, 16), and (5, 25) in a two-dimensional space.
- The `loadtxt()` function loads text data into a NumPy array.

You should get the following output:

```
array([ 1.,  1.,  2.,  4.,  3.,  9.,  4., 16.,  5., 25.])
```

2. Convert the flat array into five points in 2D space:

```
txt = txt.reshape(5,2)
txt
```

After executing preceding code, you should see the following output:

```
array([[ 1.,  1.], [ 2.,  4.], [ 3.,  9.], [ 4., 16.], [ 5., 25.]])
```

3. Split the `.txt` variable into x and y axis co-ordinates:

```
x = txt[:,0]
y = txt[:,1]
print(x, y)
```

Here is the explanation for the preceding code block:

- Separate the x and y axis points from the `txt` variable.
- x is the first column in `txt` and y is the second column.
- The Python indexing starts from 0.

After executing the preceding code, you should see the following output:

```
[ 1.  2.  3.  4.  5.] [ 1.  4.  9. 16. 25.]
```


Reading from a .csv file

The `.csv` file has a relational database structure of rows and columns, and the `test.csv` file has x, y co-ordinates for five points in 2D space. Each point is a row in the file, with two columns: x and y . The same NumPy `loadtxt()` function is used to load data:

```
x, y = np.loadtxt('test.csv', unpack = True, usecols = (0,1), delimiter =
',')
print(x)
print(y)
```

On execution of the preceding code, you should see the following output:

```
[ 1.  2.  3.  4.  5.] [ 1.  4.  9. 16. 25.]
```

Reading from an .xlsx file

Now let's read the same data from an `.xlsx` file and create the x and y NumPy arrays. The `.xlsx` file format is not supported by the NumPy `loadtxt()` function. A Python data processing package, `pandas` can be used:

1. Read the `.xlsx` file into `pandas DataFrame`. This file has the same five points in 2D space, each in a separate row with x, y columns:

```
df = pd.read_excel('test.xlsx', 'sheet', header=None)
```

2. Convert the `pandas DataFrame` to a NumPy array:

```
data_array = np.array(df)
print(data_array)
```

You should see the following output:

```
[[ 1  1] [ 2  4] [ 3  9] [ 4 16] [ 5 25]]
```

3. Now extract the x and y coordinates from the NumPy array:

```
x, y = data_array[:,0], data_array[:,1]
print(x,y)
```

You should see the following output:

```
[1 2 3 4 5] [ 1  4  9 16 25]
```

Plotting the graph

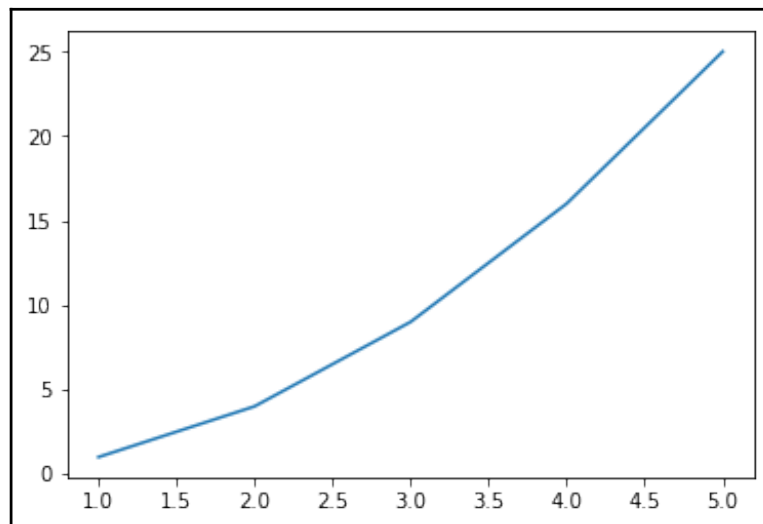
After reading the data from any of the three formats (`.txt`, `.csv`, `.xlsx`) and format it to `x` and `y` variables, then we plot the graph using these variables as follows:

```
plt.plot(x, y)
```

Display the graph on the screen:

```
plt.show()
```

The following is the output obtained:



How it works...

Depending on the format and the structure of the data, we will have to use the Python, NumPy, or pandas functions to read the data and reformat it into an appropriate structure that can be fed into the `matplotlib.pyplot` function. After that, follow the usual plotting instructions to plot the graph that you want.

Changing and resetting default environment variables

Matplotlib uses the `matplotlibrc` file to store default values for various environment and figure parameters used across matplotlib functionality. Hence, this file is very long. These default values are customizable to apply for all the plots within a session.

You can use the `print(matplotlib.rcParams)` command to get all the default parameter settings from this file.

The `matplotlib.rcParams` command is used to change these default values to any other supported values, one parameter at a time. The `matplotlib.rc` command is used to set default values for multiple parameters within a specific group, for example, lines, font, text, and so on. Finally, the `matplotlib.rcdefaults()` command is used to restore default parameters.



The `matplotlib.rcsetup()` command is used internally by Matplotlib to validate that the parameters being changed are acceptable values.

Getting ready

The following code block provides the path to the file containing all configuration the parameters:

```
# Get the location of matplotlibrc file
import matplotlib
matplotlib.matplotlib_fname()
```

You should see the directory path like the one that follows. The exact directory path depends on your installation:

```
'C:\\Anaconda3\\envs\\keras35\\lib\\site-packages\\matplotlib\\mpl-
data\\matplotlibrc'
```

How to do it...

The following block of code along with comments helps you to understand the process of changing and resetting default environment variables:

1. Import the `matplotlib.pyplot` package with the `plt` synonym:

```
import matplotlib.pyplot as plt
```

2. Load `x` and `y` variables from same `test.csv` file that we used in the preceding recipe:

```
x, y = np.loadtxt ('test.csv', unpack = True, usecols = (0,1),  
                  delimiter = ',')
```

3. Change the default values for multiple parameters within the group `'lines'`:

```
matplotlib.rc('lines', linewidth=4, linestyle='-', marker='*')
```

4. Change the default values for parameters individually:

```
matplotlib.rcParams['lines.markersize'] = 20  
matplotlib.rcParams['font.size'] = '15.0'
```

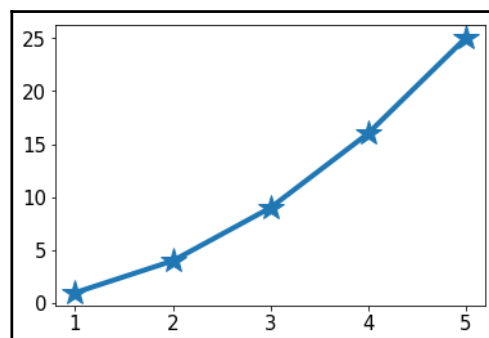
5. Plot the graph:

```
plt.plot(x,y)
```

6. Display the graph:

```
plt.show()
```

The following is the output that will be obtained:



How it works...

The `matplotlib.rc` and `matplotlib.rcParams` commands overwrite the default values for specified parameters as arguments in these commands. These new values will be used by the `pyplot` tool while plotting the graph.



It should be noted that these values will be active for all plots in the session. If you want different settings for each plot in the same session, then you should use the attributes available with the `plot` command.

There's more...

You can reset all the parameters to their default values, using the `rcdefaults()` command, as shown in the following block:

```
# To restore all default parameters
matplotlib.rcdefaults()
plt.plot(x,y)
plt.show()
```

The graph will look as follows:

