Covers gnuplot version 5

gnuplot IN ACTION

Understanding data with graphs

SECOND EDITION

Philipp K. Janert

MANNING

Praise for the First Edition

Knee-deep in data? This is your guidebook to exploring it with gnuplot.

—Austin King
Mozilla

Sparkles with insight about visualization, image perception, and data exploration.

—Richard B. Kreckel
GiNaC.de

Incredibly useful for beginners—indispensable for advanced users.

—Mark Pruett
Dominion

Bridges the gap between gnuplot's reference manual and real-world problems.

—Mitchell Johnson
Border Stylo

A Swiss Army knife for plotting data.

—Nishanth Sastry
University of Cambridge / IBM

Plain and simple: if you use Gnuplot and would like to understand it better, this book is for you. If you are looking for an excellent plotting tool—one that is highly configurable and can easily handle millions of data points, then download Gnuplot and get this book.

—Amazon reviewer

Gnuplot in Action

SECOND EDITION

PHILIPP K. JANERT

MANNING
SHELTER ISLAND

Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964

Development editors:	Marina Michaels
Technical development editor:	Ravishankar Rajagopalan
Copyeditor:	Tiffany Taylor
Proofreader:	Toma Mulligan
Technical proofreader:	Clark Gaylord
Typesetter:	Dottie Marsico
Cover designer:	Marija Tudor

ISBN 9781633430181
Printed in the United States of America
1 2 3 4 5 6 7 8 9 10 – EBM – 21 20 19 18 17 16

The purpose of computing is insight, not numbers.
—R. W. Hamming

The purpose of computing is insight, not pictures.
—L. N. Trefethen

brief contents

contents

preface

On New Year's Day, 2015, the gnuplot development team released version 5.0—the first major new gnuplot release in over 10 years! I decided to take this opportunity to bring *Gnuplot in Action* up to date and to cover all the new features gnuplot has acquired since the first edition of this book was written (in 2007).

It quickly became apparent it wouldn't be sufficient to just add a couple of chapters explaining the new features. In fact, the book you're reading now has been almost entirely rewritten from scratch. Most of the material from the first edition has been retained, but it's been heavily rearranged to accommodate the addition of new topics and to reflect the changes in my own understanding and priorities.

Gnuplot 5 is largely backward compatible with previous versions, and hence most of the first edition remains valid. At the same time, new features have been added to all parts of gnuplot, either to add new functionality or to streamline and improve the existing usage. Although many of the new features are small by themselves, when taken together, their cumulative effect leads to a significantly different, more sophisticated user experience.

In the process, the book's page count has increased substantially from the first edition. To keep the physical dimensions of the printed book in check without having to sacrifice important and useful material, some topics of a more specialized nature have been relegated to the electronic (e-book) version. Access to the e-book is included in the purchase of a print copy of the book.

Today, gnuplot is still going strong. Despite increased competition, gnuplot's two most attractive features are still largely unmet by other tools:

- The ability to explore data graphically, with an absolute minimum of effort, protocol, overhead, or boilerplate

- The ability to create immaculate, very high-quality graphs, with text labels and other decorations, for presentation purposes

What's new is that gnuplot has arrived in the 21st century. Color is now the standard, font handling is up to date, and the graphing backend makes use of all contemporary technologies to create the best-looking graphs possible.

In the first edition, I wrote that gnuplot was "an indispensable part of my toolbox: one of the handful of programs I can't do without." Several years on, this is still true.

acknowledgments

During the preparation of this book, I enjoyed conversations and correspondence with Austin King, Richard Kreckel, Ethan Merritt, Dawid Weiss, Bastian Märkisch, Daniel Sebald, Petr Mikulik, Chris Mague, Luis Moux-Dominguez, and Lee Phillips. Christoph Bersch, Zoltán Vörös, and Clark Gaylord read drafts of this book and provided many detailed suggestions; Mojca Miklavec answered several specific questions with meticulous care. Others who read the draft manuscript include Ryan Balfanz, Martin Beer, Andrew Bovill, Vitaly Bragilevsky, Anthony Cramp, Wolfgang Ecker-Lala, Wesley R. Elsberry, Nitin Gode, David Kerns, Pavol Kral, Mathew Peet, Ravishankar Rajagopalan, Karl-Friedrich Ratzsch, Jonathan Rioux, Mike Shepard, and Arthur Zubarev.

I would also like to acknowledge the tremendous impact that Wikipedia has had on the way I work. When I prepared the first edition, obtaining even basic information on topics such as color spaces, Bézier curves, and the Mandelbrot set was a real challenge—difficult, time consuming, and not always successful. For all its faults and deficiencies, Wikipedia has made it tremendously much easier to obtain at least an initial introduction (and often quite a bit more) to an incredibly wide range of topics. It is a stunning achievement.

Finally, I want to thank the people at Manning who made this book possible: publisher Marjan Bace and everyone on the editorial and production teams, including Mary Piergies, Marina Michaels, Kevin Sullivan, Tiffany Taylor, Dottie Marsico, and many others who worked behind the scenes.

about this book

This book is intended to be a comprehensive introduction to gnuplot: from the basics to the power features and beyond. In addition to providing a tutorial on gnuplot itself, it demonstrates how to apply and use gnuplot to extract insight from data.

The gnuplot program has always had complete and detailed reference documentation, but what was often missing was a continuous presentation that tied all the different bits and pieces of gnuplot together and demonstrated how to use them to achieve certain tasks. This book attempts to fill that gap. It should also serve as a handy reference for more advanced gnuplot users and as an introduction to graphical ways of knowledge discovery.

And finally, this book tries to show you how to use gnuplot to achieve some surprisingly nifty effects that will make everyone say, "How did you do that?"

Contents of this book

This book is divided into four parts. Part 1 consists of chapters 1 through 3 and is intended as a tutorial introduction to get you started with gnuplot. These three chapters cover all the truly essential material so that by the end of chapter 3, you should be able to handle most basic plotting tasks in gnuplot.

Whereas part 1 only skims the surface, part 2 goes into depth. First, chapters 4 and 5 lay more groundwork by talking about the ins and outs of file formats, string handling, and other practical matters. Then, chapters 6 through 8 discuss the various ways to change the appearance of a plot: using different plotting styles; adding labels, arrows, or other decorations; and changing the axes and their subdivisions. These chapters cover the tactical aspects of working with gnuplot in detail.

Part 3 turns its attention away from individual graphs and addresses a variety of more technical aspects. First, in chapter 9, you'll learn more about color specification, point and line types, and other relatively low-level graph elements. Chapter 10 explains how to export plots to common graphics file formats. Finally, chapters 11 and 12 address ways to improve the overall workflow through scripting and configuration changes.

In the last part, I'll mostly take gnuplot's features for granted and concentrate on the things you can do with them. Chapter 13 presents various fundamental types of graphs and explains when and how to use them. Chapter 14 is more advanced and offers solutions to some recurring topics in graphical analysis, before we end the book with a reminder of what it's all about in chapter 15.

The book has several appendixes. Appendix A explains how to obtain, build, and install gnuplot. Appendix B provides pointers to some relevant resources.

Finally, some topics of a more specialized character have been relegated to a set of supplemental appendixes: appendixes C and D discuss three-dimensional surface plots and false-color plots (heatmaps), appendix E treats some special types of graphs, and appendix F covers more mathematical topics. To reduce the physical dimensions of the print book, these four appendixes are only available in the electronic (e-book) version of this book. The purchase of a hard copy includes access to the e-book as well—you can find instructions in the front of the print book.

TIP Appendixes C through F are only available in the e-book version of this book, which is included with the purchase of the hard-copy version. Check the front of the print book for instructions on how to obtain the e-book.

How to read this book

This book was written as if readers were going to read it sequentially, cover to cover. New material is presented in order, with later chapters relying only on topics introduced earlier and avoiding forward references as much as possible. I realize that this is not a realistic picture and that the need for technical information tends to arise in a much more disjointed manner. In this spirit, I offer a few different "trail maps" to the material presented here:

- If you're new to gnuplot, begin with chapters 2 and 3 and then dive into chapters 4–8 as required to pick up the skills you need to complete whatever task you want to accomplish.
- If you're comfortable creating day-to-day graphs with gnuplot, then the material in chapters 9–12 should help you achieve greater efficiency in your work and fine-tune the results.
- If you've been using gnuplot for a long time already, then make sure you read up on the new features in gnuplot 5. Chapters 5, 9, and 10, as well as parts of chapters 11 and 12 will probably be of the most immediate interest to you.

- If you're new to graphical analysis, you may want to begin with chapter 13 to learn some of the basic methods and concepts.

Finally, keep in mind that some interesting and useful material is only available in the e-book. Three-dimensional surface and contour plots are discussed in appendix C. False-color plots (heatmaps) are treated in appendix D, together with guidelines for how to construct effective color gradients for data visualization. Appendix E explains how to combine individual graphs into composites and also discusses some other specialized types of graphs. Appendix F treats topics of a more mathematical nature.

Whom this book is for

This book is intended for anyone who wants to plot and visualize data, either to explore data sets graphically, or to create attractive, high-quality graphs for presentation and publication purposes. I had two kinds of people in mind when writing this book—those who already know gnuplot, and those who don't:

- If you already know gnuplot, I hope you'll still find it a useful reference, in particular in regard to some of the more advanced topics later in the book. I've tried to provide exactly the big-picture explanations and examples that have always been missing from the standard gnuplot reference documentation.
- If you're new to gnuplot, I think you'll find it easy enough to pick up—in fact, I can promise you that by the end of chapter 2, you'll be productive with gnuplot; and by the end of chapter 3, you'll be well equipped for most day-to-day data graphing tasks that may come your way.

This book doesn't require a strong background in mathematical methods or any in statistics, but I occasionally do expect you to have at least a fleeting familiarity with simple programming concepts. A few sections naturally require some special preliminaries (for instance, some of the discussions in chapter 10 require knowledge of LaTeX, and some sections in chapter 11 use Perl or Python code), but you can safely skip those sections if their material doesn't apply to you.

Conventions

I spell the name of the program in all lowercase (gnuplot), except at the beginning of a sentence, when I capitalize it normally. This is in accordance with the usage recommended in the gnuplot FAQ.

The gnuplot documentation is extensive, and I refer to it occasionally for additional details on topics covered only briefly or not at all here. Traditionally, the gnuplot documentation has been called the online help or online documentation, owing to the fact that it's available online during a gnuplot session. But since the advent of the internet, the word *online* seems to suggest network connectivity—falsely, in this context. To avoid confusion, I'll always refer to it as the *standard gnuplot reference documentation.*

Code examples

Gnuplot commands are shown using a monospace font, like this: `plot sin(x)`. Gnuplot commands can be entered at the gnuplot command prompt as shown in the text; the prompt itself has been suppressed to save space.

Single command lines can be long; to make them fit on a page, I occasionally had to break them across multiple lines. If so, a gray arrow (➥) has been placed at the beginning of the next line, to indicate that it is the continuation of the previous one:

```
plot "data" using 1:2 smooth csplines title "data" with lines,
➥    sin(x) title "model"
```

The break in the original line isn't indicated separately. When using gnuplot in an interactive session, your terminal program should automatically wrap a line that's too long. Alternatively, you can break lines by escaping the newline with a backslash as usual. This is useful in command files for batch processing (and you'll see some examples in chapter 12 in the context of string macros).

Some code snippets are only intended to demonstrate the syntax and don't have a graph associated with them. In this case, I use the generic name "data" as a placeholder for the actual filename. No file named *data* exists in the downloads (in the same way that no key named *any* can be found on a computer keyboard). It's just a generic placeholder.

Occasionally, I show Unix commands that need to be entered in a Unix shell; to emphasize that these aren't gnuplot commands, I prefix them with a generic shell prompt, like this: `shell>`. Similarly, Python commands to be entered in a Python session are prefixed with `python>>>`.

Downloads

The code for all numbered listings is available for download from www.manning.com/books/gnuplot-in-action-second-edition, and so are the data sets. The only exception to this are publicly available data sets: for these, I provide the URL where they can be found.

Gnuplot searches for data files in the current directory, so the easiest way to run the supplied command files is as follows:

1. Change into the data directory of the downloaded bundle.
2. Start gnuplot.
3. Issue `plot` commands at the gnuplot prompt the way they're shown in the text (for example, `plot "marathon" using 1:2`), or give the full pathname to the gnuplot command file that you wish to run (for example, `load "../gnuplot/shapes.gp"`).

Command synopses

Gnuplot has a large number of options, and keeping all of them and their suboptions and optional parameters straight is a major theme running through this book. Frequently, I'll display all available options to a command in a command synopsis before discussing the options in detail. To distinguish a synopsis of available options from actual gnuplot code, a synopsis uses an italic font, like so:

```
set datafile commentschar ["{str:chars}"]
```

Within these summaries, I use a few syntactic conventions. My intent here is to stay close to the usage familiar from the standard gnuplot reference documentation, but also to follow more general conventions (such as those used for Unix man pages):

`[ ... ]`	**Square brackets for optional parts**
`[ | ]`	**Vertical bars to separate alternatives**
`{ ... }`	**Curly braces for user-supplied input**

For parameters supplied by the user, it's often not clear from the context what kind of information the command expects: is it a string or a number? If it's a number, is it a value selected from a fixed range of integers or a numerical factor? And so on. I've tried to clarify this situation by prefixing each user-supplied input parameter with a type indicator, terminated by a colon. I summarize the prefixes and their meanings in table 1.

Table 1 Type indicators for user-supplied parameters

Prefix	Description
`str:`	A string
`int:`	An integer number
`flt:`	A floating-point number
`idx:`	An integer number, which is interpreted as a selection from a fixed range of values
`clr:`	A color specification—for example, `rgbcolor "red"` or `rgb "#FFFF00"`
`pos:`	A pair of comma-separated coordinates, optionally containing coordinate system specifiers—for example, `0,0` or `first 1.1, screen 0.9`
`enum:`	A gnuplot keyword as unquoted string

Abbreviations

Many gnuplot commands have abbreviated forms, which I use frequently. The essential `plot` command, in particular, takes a large number of keyword directives, which I usually abbreviate to save space and keystrokes. I strongly recommend that you quickly become familiar with these shorthands and use them yourself. Table 2 lists both the abbreviated and the full forms. The `plot` command also understands a large number of appearance options (controlling aspects such as line width, style, and color), which are generally also abbreviated. A comprehensive summary of appearance options, together with their shorthands, can be found in table 6.1.

Table 2 Abbreviations for frequently used directives to the `plot` command

Abbreviation	Full
`i`	`index`
`ev`	`every`
`u`	`using`
`s`	`smooth`
`s acs`	`smooth acsplines`
`s f`	`smooth frequency`
`s kdens`	`smooth kdensity`
`t`	`title`
`w`	`with`
`w l`	`with lines`
`w linesp` or `w lp`	`with linespoints`
`w p`	`with points`
`w vec`	`with vectors`

Table 3 lists three frequently occurring commands that are also usually abbreviated.

Table 3 Abbreviations for frequently occurring commands

Abbreviation	Full
`set t`	`set terminal`
`set o`	`set output`
`set logsc`	`set logscale`

The figures in this book

The graphs in this book were generated with gnuplot; some special cases were handled using pic. All graphs were originally prepared in color, using my own set of preferred colors instead of one of gnuplot's default color schemes. The color versions of the graphs are used in the electronic (e-book) version of this book. For the print book, I prepared black-and-white versions through the application of an appropriate stylesheet (see chapter 12). A handful of graphs required manual touch-ups in addition to the monochrome stylesheet to yield an optimal appearance.

You'll find the line-type definitions of both the color and the black-and-white stylesheets in table 4. The same colors and dash patterns are discussed in listings 12.7 and 12.9.

In particular in the latter part of the book, I frequently use point types (point shapes) that aren't the default, because the visual appearance of the graphs can often be improved greatly this way. If so, the point type is usually chosen explicitly in the appropriate code examples and listings.

The final version of each figure was generated using the `pdfcairo` terminal, using a (non-default) aspect ratio of √2 to 1 and Helvetica as the requested font.

Table 4 Colors and dash patterns used for the color and monochrome figures in this book

Color	Monochrome
`set linetype 1 lc rgb '0xee0000'`	`set linetype 1 lc black dt solid`
`set linetype 2 lc rgb '0x008b00'`	`set linetype 2 lc black dt (8,6)`
`set linetype 3 lc rgb '0x0000cd'`	`set linetype 3 lc black dt (4,3)`
`set linetype 4 lc rgb '0xff3fb3'`	`set linetype 4 lc black dt (3,6)`
`set linetype 5 lc rgb '0x00cdcd'`	`set linetype 5 lc black dt (12,5,2,5,2,5)`
`set linetype 6 lc rgb '0xcd9b1d'`	`set linetype 6 lc black dt (16,8)`
`set linetype 7 lc rgb '0x8968cd'`	`set linetype 7 lc black dt (20,6,2,6)`
`set linetype 8 lc rgb '0x8b8b83'`	`set linetype 8 lc black dt (30,10)`

Hardware and software requirements

This book describes gnuplot version 5.0 or higher, which was initially released in early 2015. Not all examples in this book will work with earlier gnuplot versions. If you have an earlier version of gnuplot, you should upgrade to a more current version—appendix A tells you how.

I assume you have access to a reasonably modern computer running any flavor of Unix/Linux, a recent release of MS Windows, or Mac OS X. Gnuplot has been ported to many other platforms but is actively supported primarily on the three operating systems just mentioned, and so I concentrate on them in this book.

Reference materials

Command and option references are distributed throughout the book, wherever the material is first introduced. The following pointers are intended to help you find these summaries more easily.

Graphical styles and specifications

Appearance specifiers and line options	Table 6.1	page 104
Graph locations	Figure 7.2	page 128
Graph layers	Figure 7.3	page 129
Explicit color-specification formats	Table 9.1	page 184
Point types	Figure 9.7	page 196
Dash patterns	Table 9.2	page 197

File access

Column-access methods and functions	Table 4.2	page 67
Pseudofiles	Table 4.3	page 71
Metadata in data files	Table 4.4	page 76

String handling and formatting

String functions	Table 5.1	page 81
General conversion specifiers	Table 8.2	page 160
Accuracy specifiers	Table 8.3	page 160
Time-series conversions, sorted alphabetically	Table 8.4	page 172
Time-series conversions, sorted by topic	Table 8.5	page 173
Time functions	Table 8.6	page 176
Enhanced text mode	Table 10.1	page 218

Operators and mathematical functions

Unary operators	Table 3.1	page 35
Binary operators	Table 3.2	page 35
Mathematical functions	Table F.1	page F11
Complex numbers	Table F.2	page F13

Programming constructs

Inline loops	Listing 5.3	page 92
General loops and conditionals	Table 11.1	page 238

About the author

PHILIPP K. JANERT was born and raised in Germany. He obtained a Ph.D. in theoretical physics from the University of Washington in 1997 and has been working in the tech industry ever since, including four years at Amazon.com, where he initiated and led several projects to improve Amazon's order-fulfillment process. He's the author of several books on data analysis and applied math, including the best-selling *Data Analysis with Open Source Tools* (O'Reilly, 2010). He has contributed to CPAN and is an occasional committer on the gnuplot project. Visit his company website at www.principal-value.com.

Author Online

Purchase of *Gnuplot in Action, Second Edition* includes free access to a private web forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the lead author and from other users. To access the forum and subscribe to it, point your web browser to www.manning.com/books/gnuplot-in-action-second-edition. This page provides information on how to get on the forum once you are registered, what kind of help is available, and the rules of conduct on the forum.

Manning's commitment to our readers is to provide a venue where a meaningful dialog between individual readers and between readers and the author can take place. It is not a commitment to any specific amount of participation on the part of the author, whose contribution to Author Online remains voluntary (and unpaid). We suggest you try asking the author some challenging questions lest his interest stray! The Author Online forum and the archives of previous discussions will be accessible from the publisher's website as long as the book is in print.

About the cover

The figure on the cover of *Gnuplot in Action, Second Edition* is captioned "A peer of France." The title of Peer in France was held by the highest-ranking members of the French nobility. It was an extraordinary honor granted only to a few dukes, counts, and princes of the church. The illustration is taken from a 19th-century edition of Sylvain Maréchal's four-volume compendium of regional dress customs published in France. Each illustration is finely drawn and colored by hand.

The rich variety of Maréchal's collection reminds us vividly of how culturally apart the world's towns and regions were just 200 years ago. Isolated from each other, people spoke different dialects and languages. In the streets or in the countryside, it was easy to identify where they lived and what their trade or station in life was just by their dress.

Dress codes have changed since then, and the diversity by region, so rich at the time, has faded away. It's now hard to tell apart the inhabitants of different continents, let along different towns or regions. Perhaps we have traded cultural diversity for a more varied personal life—certainly for a more varied and fast-paced technological life.

At a time when it's hard to tell one computer book from another, Manning celebrates the inventiveness and initiative of the computer business with book covers based on the rich diversity of regional life of two centuries ago, brought back to life by Maréchal's pictures.

Part 1

Getting started

Gnuplot is a tool for visualizing data and mathematical functions. The chapters in this first part will give a first introduction to gnuplot and its most important features. Chapter 1 introduces gnuplot and describes the kinds of problems it's designed to solve. Chapter 2 provides a quick tutorial to gnuplot. By the end of this chapter, you'll be able to prepare simple plots with gnuplot and to save and export your work. Chapter 3 takes a detailed look at the all-important `plot` command, which is used to generate most graphs in gnuplot. You'll also learn about inline transformations and built-in smoothing methods.

1 Prelude: understanding data with gnuplot

This chapter covers

- Warmup examples
- What is graphical analysis?
- What is gnuplot?

NOTE TO PRINT BOOK READERS Some material of a more specialized nature is only available in the e-book version of this book. The e-book also shows all the graphs in color. To get your free e-book in PDF, ePub, or Kindle format, go to www.manning.com/books/gnuplot-in-action-second-edition to register your print book.

Gnuplot has long been one of the most popular open source programs for plotting and visualizing data. In this book, I want to show you how to use gnuplot to make plots and graphs of your data: both quick and easy graphs for your own use and highly polished graphs for presentations and publications.

But I also want to show you something else: how to solve data-analysis problems using graphical methods. The art of discovering relationships in data and extracting information from it by visual means is called *graphical analysis*, and I believe gnuplot to be an excellent tool for it.

As a teaser, let's look at some problems and how you might be able to approach them using graphical methods. The graphs here and in the rest of the book (with very few exceptions) have been, of course, generated with gnuplot.

1.1 A busy weekend

To get a feeling for the kinds of problems you may be dealing with and for the kinds of solutions gnuplot can help you find, let's look at two examples. Both take place during a long, busy weekend.

1.1.1 Planning a marathon

Imagine you're in charge of organizing the local city marathon. There will be more than 2,000 starters, traffic closed around the city, plenty of spectators—and a major Finish Line Festival to celebrate the victors. The big question is: when should the Finish Line crew be ready to deal with the majority of runners? At what point do you expect the big influx of the masses?

You have the results from last year's event. Assuming that the starters haven't improved dramatically over the last year (probably a safe assumption), you do a quick average of the completion times and find that last year's average was 282 minutes. To be on the safe side, you calculate the standard deviation as well, which comes out to about 50 minutes. So you tell your crew to be ready for the big rush starting three and a half hours (210 minutes) after the start, and you feel reasonably well prepared for the event.

So it comes as a surprise when on the big day, plenty of runners start showing up at the finish line after only two hours—a good 90 minutes earlier than the expected onset of the rush. In terms of event management, the number of runners who show up early isn't overwhelming, but it's a bit strange. The next day you wonder: what went wrong?

Let's look at the data to see what you can learn about it. So far, all you know are the mean and the standard deviation.

The mean is convenient: it's easy to calculate, and it summarizes the entire data set in a single number. But in forming the mean, you lost a lot of information. To understand the entire data set, you have to *look* at it. And because you can't understand data by looking at more than 2,000 individual finish times, this means you have to *plot* it.

It will be convenient to group the runners by completion time and to count the number of participants who finished during each five-minute interval. The resulting file might start like this:

```
# Minutes Runners
135       1
140       2
145       4
150       7
155       11
160       13
165       35
170       29
...
```

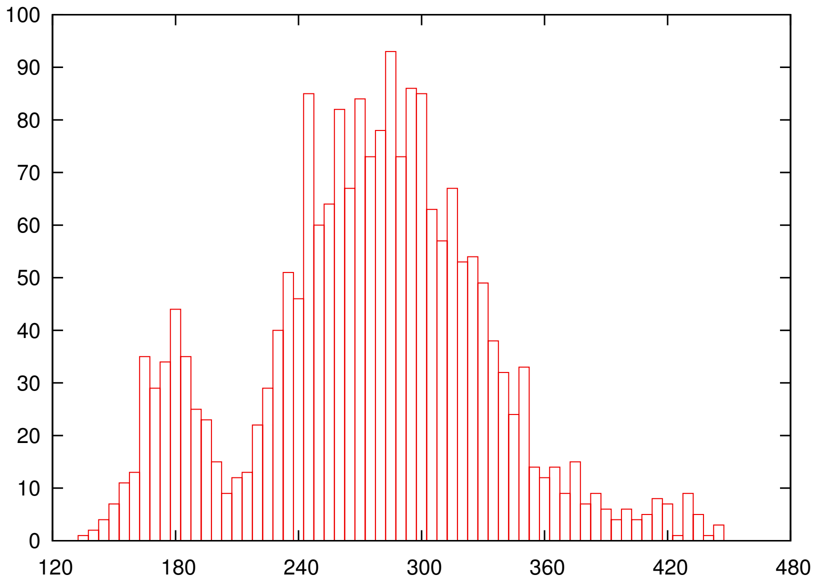

Figure 1.1 Number of finishers vs. time to complete (in minutes)

Now you plot the number of runners against the completion time (see figure 1.1). It's immediately obvious where you went wrong: the data is *bimodal*, meaning it has two peaks. There is an early peak at around 180 minutes and a later main peak at 300 minutes.

Actually, this makes sense: a major sporting event such as a city marathon attracts two very different groups of people: athletes, who train and compete throughout the year and are in it to win, and a much larger group of amateurs, who come out once a year for a big event and are mostly there to participate. The problem is that for such data, the mean and standard deviation are obviously bad representations—so much so. that at the time when you expected the big rush (200 minutes), there's a lull at the finish line!

The take-home message here is that it's usually not a good idea to rely on summary statistics (such as the mean) for unknown data sets. You *always* should investigate what the data looks like. Once you've confirmed the basic shape, you can choose how to summarize your findings best.

And of course, there is always more to learn. In this example, for instance, you see that after about 400 minutes, almost everybody has made it, and you can start winding down the operation. The actual "tail" of the distribution is quite small—surprisingly so. (I would've expected to see a greater number of stragglers, but possibly many runners who are *really* slow drop out of the race when they realize they'll place badly.)

USING GNUPLOT

Let's look at the gnuplot command that was used to generate figure 1.1. Gnuplot is command-line oriented: after you start gnuplot, it drops you into an interactive command session, and all commands are typed at the interactive gnuplot prompt.

Gnuplot reads data from simple text files, with the data arranged in columns as shown previously. To plot a data file takes only a single command, `plot`, like this:[1]

```
plot "marathon" using 1:2 with boxes
```

The `plot` command requires the name of the data file as argument in quotes. By default, gnuplot looks for the data file in the current working directory—normally the directory from which you started gnuplot. The filename provided to the `plot` command may contain path information to refer to a file that doesn't reside in the current directory.

The rest of the command line specifies which columns to use for the plot and in which way to represent the data. The `using 1:2` declaration tells gnuplot to use the first and second columns in the file called marathon. The final part of the command, `with boxes`, selects a box style, which is often suitable to display counts of events.

Gnuplot handles most everything else by itself: it sizes the graph and selects the most interesting plot range, it draws the border, and it draws the tic marks and their labels. All these details can be customized, but gnuplot typically does a good job at anticipating what the user wants.

NOTE The little markers along the edge that define the *scale* of the corresponding axis are called *tick marks* (or *tic marks*). The gnuplot standard reference documentation uses the spelling *tic* mark; the relevant commands are called `set xtics`, `set ytics`, and so on. In order to avoid confusion, I use the same spelling (*tic*) throughout this book.

1.1.2 Determining the future

The same weekend when 2,000 runners are running through the city, a diligent graduate student is working on his research topic. He studies diffusion limited aggregation (DLA), a process wherein a particle performs a random walk until it comes into contact with a growing cluster of particles. At the moment of contact, the particle sticks to the cluster at the location where the contact occurred and becomes part of the cluster. Then a new random walker is released to perform a random walk, until *it* sticks to the cluster. And so on. Clusters grown through this process have a remarkably open, tenuous structure (as shown in figure 1.2): they're *fractals*.[2]

The DLA process is simple, so it seems straightforward to write a program to grow such clusters in a computer, and this is what the busy graduate student has done. Initially, all seems well; but as the simulation progresses, the cluster appears to grow more and more slowly—excruciatingly slowly, in fact. The goal was to grow a DLA cluster in excess of 100,000 particles. Will the program ever finish?

---

[1] Depending on your gnuplot setup and initialization, your graphs may look slightly different from the figures shown in this chapter. We'll discuss user-defined appearance options starting with chapter 6.

[2] The original paper on DLA was "Diffusion Limited Aggregation, A Kinetic Critical Phenomenon" by T. A. Witten and L. M. Sander, *Physical Review Letters* 41 (1981): 1400. It's one of the most-quoted papers from that journal of all time. If you want to learn more about DLA and similar processes, check out *Fractals, Scaling, and Growth Far From Equilibrium* by Paul Meakin (Cambridge University Press, 1998).

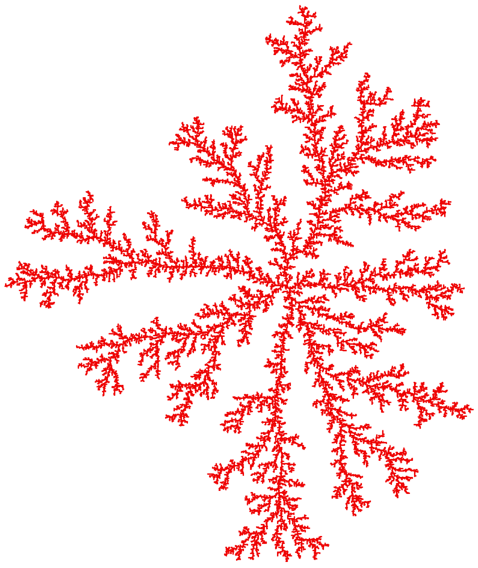

Figure 1.2 A DLA cluster of *N*=50,000 particles, drawn with gnuplot

Luckily, the simulation program periodically writes information about its progress to a log file: for each new particle added to the cluster, the time (in seconds) since the start of the simulation is recorded. The grad student should be able to predict the completion time from this data, but an initial plot (figure 1.3) isn't helpful; there are too many ways this curve can be extrapolated to larger cluster sizes.

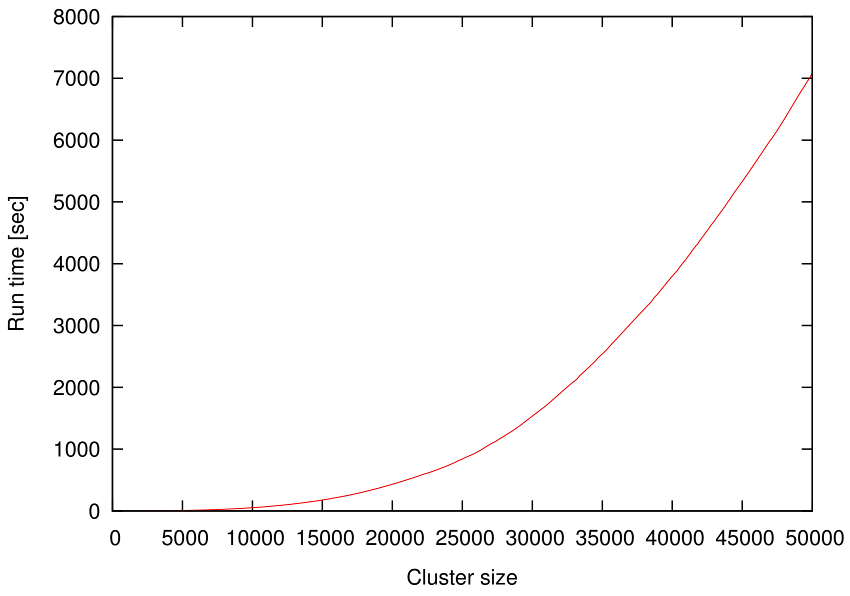

Figure 1.3 Time required to grow a DLA cluster

The time consumed by many computer algorithms grows as a simple power of the size of the problem. In this case, this would be the number *N* of particles in the cluster $T \sim N^k$, for some value of *k*. The research student therefore plots the running time of his simulation program on a double-logarithmic plot versus the cluster size (see figure 1.4). The data points fall on a straight line, indicating a power law. (I'll explain later how and why this works.) Through a little trial and error, he also finds an equation that approximates the data quite well. The equation can be extended to any cluster size desired and will give the time required. For *N*=100,000 (which was the original goal), he can read off almost *T*=100,000 seconds (or more), corresponding to more than 24 hours, so there is no point in your friend spending the weekend in the lab—he should go out (maybe run a marathon) and come back on Monday, or perhaps work on a better algorithm. (For simulations of DLA cluster growth, dramatic speedups over the naive implementation are possible. Try it if you like.)

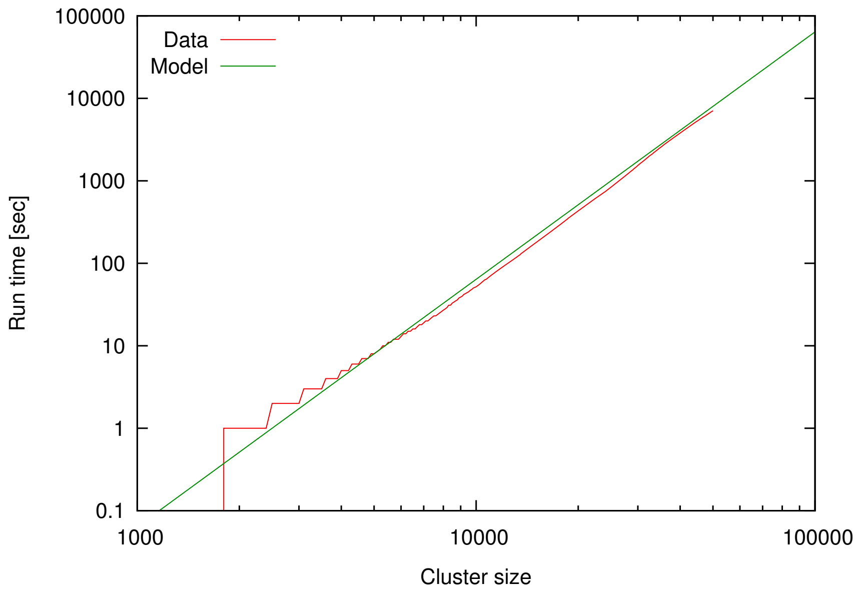

Figure 1.4 Time required to grow a DLA cluster in a double-logarithmic plot, together with an approximate mathematical model

USING GNUPLOT

Again, let's see how the graphs in this section were created. The easiest to understand is figure 1.3. Given a file containing two columns, one listing the cluster size and the other listing the completion time, the command is just

```
plot "runtime" using 1:2 with lines
```

The only difference compared to figure 1.1 is the style: rather than boxes, I use line segments to connect consecutive data points: `with lines`.

Did you notice that figure 1.3 and figure 1.4 contain more than just data? Both axes are now labelled! Details such as labels and other helpful decorations often make the difference between a mediocre and a high-quality graph, because they provide the observer with the necessary context to fully understand the graph.

In gnuplot, all details of a graph's appearance are handled by setting the appropriate options. To place the labels on the x and y axes in figure 1.3, I used

```
set xlabel "Cluster size"
set ylabel "Run time [sec]"
```

Figure 1.4 is drawn using double-logarithmic axes. This is another option, which is set as follows:

```
set logscale
```

Figure 1.4 shows two curves: the data together with a best "fit." Plotting several data sets or mathematical functions together in one plot is easy—you list them one after another on the command line for the `plot` command:

```
plot "runtime" using 1:2 title "Data" with lines,
➥     (x/2500)**3 title "Model"
```

This command introduces a further gnuplot feature: the `title` directive. It takes a string as argument, which is displayed together with a line sample in the plot's key or legend (visible at upper left in figure 1.4).

Finally, we come to figure 1.2. It's a somewhat different beast. Notice that the border and the tic marks are missing. The aspect ratio (the ratio of the graph's width to its height) has been constrained to 1, and a single dot has been placed at the position of each particle in the cluster. Here are the most important commands that I used:

```
unset border
unset xtics
unset ytics

set size square

plot "cluster" using 1:2 with dots
```

You can see that gnuplot is simple to use. In the next section, I talk more about using graphical methods to understand a data set, before coming back to gnuplot and discussing why it's my favorite tool for this kind of activity.

1.2 What is graphical analysis?

The previous two examples should have given you an idea of what graphical analysis is and how it works. The basic steps are always the same: