On this page

1 Introduction

1.1 Welcome to ggplot2

1.2 What is the grammar of graphics?

1.3 How does ggplot2 fit in with other R graphics?

1.4 About this book

1.5 Prerequisites

1.6 Other resources

1.7 Colophon

View source

Edit this page

1 Introduction

1.1 Welcome to ggplot2

ggplot2 is an R package for producing statistical, or data, graphics. Unlike most other graphics packages, ggplot2 has an underlying grammar, based on the Grammar of Graphics,[1] that allows you to compose graphs by combining independent components. This makes ggplot2 powerful. Rather than being limited to sets of pre-defined graphics, you can create novel graphics that are tailored to your specific problem. While the idea of having to learn a grammar may sound overwhelming, ggplot2 is actually easy to learn: there is a simple set of core principles and there are very few special cases. The hard part is that it may take a little time to forget all the preconceptions that you bring over from using other graphics tools.

ggplot2 provides beautiful, hassle-free plots that take care of fiddly details like drawing legends. In fact, its carefully chosen defaults mean that you can produce publication-quality graphics in seconds. However, if you do have special formatting requirements, ggplot2's comprehensive theming system makes it easy to do what you want. Ultimately, this means that rather than spending your time making your graph look pretty, you can instead focus on creating the graph that best reveals the message in your data.

ggplot2 is designed to work iteratively. You start with a layer that shows the raw data. Then you add layers of annotations and statistical summaries. This allows you to produce graphics using the same structured thinking that you would use to design an analysis. This reduces the distance between the plot in your head and the one on the page. This is especially helpful for students who have not yet developed the structured approach to analysis used by experts.

Learning the grammar will not only help you create graphics that you're familiar with, but will also help you to create newer, better graphics. Without a grammar, there is no underlying theory, so most graphics packages are just a big collection of special cases. For example, in base R, if you design a new graphic, it's composed of raw plot elements like lines and points so it's hard to design new components that combine with existing plots. In ggplot2, the expressions used to create a new graphic are composed of higher-level elements, like representations of the raw data and statistical transformations, that can easily be combined with new datasets and other plots.

This book provides a hands-on introduction to ggplot2 with lots of example code and graphics. It also explains the grammar on which ggplot2 is based. Like other formal systems, ggplot2 is useful even when you don't understand the underlying model. However, the more you learn about it, the more effectively you'll be able to use ggplot2.

This book will introduce you to ggplot2 assuming that you're a novice, unfamiliar with the grammar; teach you the basics so that you can re-create plots you are already familiar with; show you how to use the grammar to create new types of graphics; and eventually turn you into an expert who can build new components to extend the grammar.

1.2 What is the grammar of graphics?

Wilkinson[2] created the grammar of graphics to describe the fundamental features that underlie all statistical graphics. The grammar of graphics is an answer to the question of what is a statistical graphic? ggplot2[3] builds on Wilkinson's grammar by focussing on the primacy of layers and adapting it for use in R. In brief, the grammar tells us that a graphic maps the data to the aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also include statistical transformations of the data and information about the plot's coordinate system. Facetting can be used to plot for different subsets of the data. The combination of these independent components are what make up a graphic.

As the book progresses, the formal grammar will be explained in greater detail. The first description of the components follows below. It introduces some of the terminology that will be used throughout the book and outlines the basic function of each component. Don't worry if it doesn't make sense right away: you'll have many more opportunities to learn about the components and how they work together.

All plots are composed of the **data**, the information you want to visualise, and a **mapping**, the description of how the data's variables are mapped to aesthetic attributes. There are five mapping components:

- A **layer** is a collection of geometric elements and statistical transformations. Geometric elements, **geom**s for short, represent what you actually see in the plot: points, lines, polygons, etc. Statistical transformations, **stat**s for short, summarise the data: for example, binning and counting observations to create a histogram, or fitting a linear model.

- **Scale**s map values in the data space to values in the aesthetic space. This includes the use of colour, shape or size. Scales also draw the legend and axes, which make it possible to read the original data values from the plot (an inverse mapping).

- A **coord**, or coordinate system, describes how data coordinates are mapped to the plane of the graphic. It also provides axes and gridlines to help read the graph. We normally use the Cartesian coordinate system, but a number of others are available, including polar coordinates and map projections.

- A **facet** specifies how to break up and display subsets of data as small multiples. This is also known as conditioning or latticing/trellising.

- A **theme** controls the finer points of display, like the font size and background colour. While the defaults in ggplot2 have been chosen with care, you may need to consult other references to create an attractive plot. A good starting place is Tufte's early works.[4]

It's also important to note what the grammar doesn't do:

- It doesn't suggest which graphics to use. While this book endeavours to promote a sensible process for producing plots, the focus is on how to produce the plots you want, not on which plot to produce. For more advice on choosing or creating plots to answer the question you're interested in, you may want to consult Naomi Robbins[5], William Cleveland[6], John Chambers et al.[7], and John W. Tukey[8].

- It doesn't describe interactive graphics, only static ones. There is essentially no difference between displaying ggplot2 graphs on a computer screen and printing them on a piece of paper. For dynamic and interactive graphics, you'll have to look elsewhere (perhaps at ggvis, described below). Dianne Cook and Deborah F. Swayne[9] provides an excellent introduction to the interactive graphics package GGobi. GGobi can be connected to R with the rggobi package.[10]

1.3 How does ggplot2 fit in with other R graphics?

There are a number of other graphics systems available in R: base graphics, grid graphics and trellis/lattice graphics. How does ggplot2 differ from them?

- Base graphics were written by Ross Ihaka based on experience implementing the S graphics driver and partly looking at Chambers et al.[11] Base graphics has a pen on paper model: you can only draw on top of the plot, you cannot modify or delete existing content. There is no (user accessible) representation of the graphics, apart from their appearance on the screen. Base graphics includes both tools for drawing primitives and entire plots. Base graphics functions are generally fast, but have limited scope. If you've created a single scatterplot, or histogram, or a set of boxplots in the past, you've probably used base graphics.

- The development of "grid" graphics, a much richer system of graphical primitives, started in 2000. Grid is developed by Paul Murrell, growing out of his PhD work.[12] Grid grobs (graphical objects) can be represented independently of the plot and modified later. A system of viewports (each containing its own coordinate system) makes it easier to lay out complex graphics. Grid provides drawing primitives, but no tools for producing statistical graphics.

- The lattice package, developed by Deepayan Sarkar, uses grid graphics to implement the trellis graphics system of Cleveland[13] and is a considerable improvement over base graphics. You can easily produce conditioned plots and some plotting details (e.g., legends) are taken care of automatically. However, lattice graphics lacks a formal model, which can make it hard to extend. Lattice graphics are explained in depth in Deepayan Sarkar[14].

- ggplot2, started in 2005, is an attempt to take the good things about base and lattice graphics and improve on them with a strong underlying model which supports the production of any kind of statistical graphic, based on the principles outlined above. The solid underlying model of ggplot2 makes it easy to describe a wide range of graphics with a compact syntax, and independent components make extension easy. Like lattice, ggplot2 uses grid to draw the graphics, which means you can exercise much low-level control over the appearance of the plot.

- htmlwidgets, http://www.htmlwidgets.org, provides a common framework for accessing web visualisation tools from R. Packages built on top of htmlwidgets include leaflet (https://rstudio.github.io/leaflet/, maps), dygraph (http://rstudio.github.io/dygraphs/, time series) and networkD3 (http://christophergandrud.github.io/networkD3/, networks).

- plotly, https://plotly-r.com, is a popular javascript visualisation toolkit with an R interface. It's a great tool if you want to make interactive graphics for HTML documents, and even comes with a `ggplotly()` function that can convert many ggplot2 graphics into their interactive equivalents.

Many other R packages, such as vcd,[15] plotrix[16] and gplots,[17] implement specialist graphics, but no others provide a framework for producing statistical graphics. A comprehensive list of all graphical tools available in other packages can be found in the graphics task view at http://cran.r-project.org/web/views/Graphics.html.

1.4 About this book

The first chapter, Chapter 2, describes how to quickly get started using ggplot2 to make useful graphics. This chapter introduces several important ggplot2 concepts: geoms, aesthetic mappings and facetting.

Chapters 3 to 9 explore how to use the basic toolbox to solve a wide range of visualisation problems that you're likely to encounter in practice.

Then Chapters 10 to 12 show you how to control the most important scales, allowing you to tweak the details of axes and legends.

Chapter 13 describes the layered grammar of graphics which underlies ggplot2. The theory is illustrated in Chapter 14 which demonstrates how to add additional layers to your plot, exercising full control over the geoms and stats used within them.

Understanding how scales work is crucial for fine-tuning the perceptual properties of your plot. Customising scales gives fine control over the exact appearance of the plot and helps to support the story that you are telling. Chapters 10, 11 and 12 will show you what scales are available, how to adjust their parameters, and how to control the appearance of axes and legends.

Coordinate systems and facetting control the position of elements of the plot. These are described in Chapter 14.7. Faceting is a very powerful graphical tool as it allows you to rapidly compare different subsets of your data. Different coordinate systems are less commonly needed, but are very important for certain types of data.

To polish your plots for publication, you will need to learn about the tools described in Chapter 18. There you will learn about how to control the theming system of ggplot2 and how to save plots to disk.

1.5 Prerequisites

Before we continue, make sure you have all the software you need for this book:

- **R**: If you don't have R installed already, you may be reading the wrong book; I assume a basic familiarity with R throughout this book. If you'd like to learn how to use R, I'd recommend my *R for Data Science* which is designed to get you up and running with R with a minimum of fuss.

- **RStudio**: RStudio is a free and open source integrated development environment (IDE) for R. While you can write and use R code with any R environment (including R GUI and ESS), RStudio has some nice features specifically for authoring and debugging your code. We recommend giving it a try, but it's not required to be successful with ggplot2 or with this book. You can download RStudio Desktop from https://www.rstudio.com/products/rstudio/download

- **R packages**: This book uses a bunch of R packages. You can install them all at once by running:

```
install.packages(c(
  "colorBlindness", "directlabels", "dplyr", "ggforce", "gghighlight",
  "ggnewscale", "ggplot2", "ggraph", "ggrepel", "ggtext", "ggthemes",
  "hexbin", "Hmisc", "mapproj", "maps", "munsell", "ozmaps",
  "paletteer", "patchwork", "rmapshaper", "scico", "seriation", "sf",
  "stars", "tidygraph", "tidyr", "wesanderson"
))
```

1.6 Other resources

This book teaches you the elements of ggplot2's grammar and how they fit together, but it does not document every function in complete detail. You will need additional documentation as your use of ggplot2 becomes more complex and varied.

The best resource for specific details of ggplot2 functions and their arguments will always be the built-in documentation. This is accessible online, https://ggplot2.tidyverse.org/reference/index.html, and from within R using the usual help syntax. The advantage of the online documentation is that you can see all the example plots and navigate between topics more easily.

If you use ggplot2 regularly, it's a good idea to sign up for the ggplot2 mailing list, http://groups.google.com/group/ggplot2. The list has relatively low traffic and is very friendly to new users. Another useful resource is stackoverflow, http://stackoverflow.com. There is an active ggplot2 community on stackoverflow, and many common questions have already been asked and answered. In either place, you're much more likely to get help if you create a minimal reproducible example. The reprex package by Jenny Bryan provides a convenient way to do this, and also include advice on creating a good example. The more information you provide, the easier it is for the community to help you.

The number of functions in ggplot2 can be overwhelming, but RStudio provides some great cheatsheets to jog your memory at http://www.rstudio.com/resources/cheatsheets/.

Finally, the complete source code for the book is available online at https://github.com/hadley/ggplot2-book. This contains the complete text for the book, as well as all the code and data needed to recreate all the plots.

1.7 Colophon

This book was written in RStudio using bookdown. The website is hosted with netlify, and automatically updated after every commit by Github Actions. The complete source is available from GitHub.

This version of the book was built with R version 4.2.2 (2022-10-31) and the following packages:

package	version	source
colorBlindness	0.1.9	RSPM
directlabels	2021.1.13	RSPM
dplyr	1.0.10	RSPM
ggforce	0.4.1	RSPM
gghighlight	0.4.0	RSPM
ggnewscale	0.4.8	RSPM
ggplot2	3.3.6	RSPM
ggraph	2.1.0	RSPM
ggrepel	0.9.1	RSPM
ggtext	0.1.2	RSPM
ggthemes	4.2.4	RSPM
hexbin	1.28.2	RSPM

package	version	source
Hmisc	4.7-1	RSPM
mapproj	1.2.9	RSPM
maps	3.4.1	RSPM
munsell	0.5.0	RSPM
ozmaps	0.4.5	RSPM
paletteer	1.5.0	RSPM
patchwork	1.1.2	RSPM
rmapshaper	0.4.6	RSPM
scico	1.3.1	RSPM
seriation	1.4.0	RSPM
sf	1.0-8	RSPM
stars	0.5-6	RSPM
tidygraph	1.2.2	RSPM
tidyr	1.2.1	RSPM
wesanderson	0.3.6	RSPM

« Preface to the second edition

2 First steps »