Chapter 1

Introduction

1.1 Welcome to ggplot2

`ggplot2` is an R package for producing statistical, or data, graphics, but it is unlike most other graphics packages because it has a deep underlying grammar. This grammar, based on the Grammar of Graphics (Wilkinson, 2005), is composed of a set of independent components that can be composed in many different ways. This makes `ggplot2` very powerful, because you are not limited to a set of pre-specified graphics, but you can create new graphics that are precisely tailored for your problem. This may sound overwhelming, but because there is a simple set of core principles and very few special cases, `ggplot2` is also easy to learn (although it may take a little time to forget your preconceptions from other graphics tools).

Practically, `ggplot2` provides beautiful, hassle-free plots, that take care of fiddly details like drawing legends. The plots can be built up iteratively and edited later. A carefully chosen set of defaults means that most of the time you can produce a publication-quality graphic in seconds, but if you do have special formatting requirements, a comprehensive theming system makes it easy to do what you want. Instead of spending time making your graph look pretty, you can focus on creating a graph that best reveals the messages in your data.

`ggplot2` is designed to work in a layered fashion, starting with a layer showing the raw data then adding layers of annotations and statistical summaries. It allows you to produce graphics using the same structured thinking that you use to design an analysis, reducing the distance between a plot in your head and one on the page. It is especially helpful for students who have not yet developed the structured approach to analysis used by experts.

Learning the grammar will help you not only create graphics that you know about now, but will also help you to think about new graphics that would be even better. Without the grammar, there is no underlying theory and existing 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

points and lines, and 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, and 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 assumes some basic familiarity with R, to the level described in the first chapter of Dalgaard's *Introductory Statistics with R*. You should know how to get your data into R and how to do basic data manipulations. If you don't, you might want to get a copy of Phil Spector's *Data Manipulation with R*.

This book will introduce you to ggplot2 as 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 even turn you into an expert who can build new components to extend the grammar.

1.2 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. Furthermore, ggplot2 will almost certainly continue to evolve. For these reasons, you will need additional documentation as your use of ggplot2 becomes more complex and varied.

The best resource for low-level details will always be the built-in documentation. This is accessible online, http://had.co.nz/ggplot2, 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.

The website also lists talks and papers related to ggplot2 and training opportunities if you'd like some hands-on practice. The CRAN website, http://cran.r-project.org/web/packages/ggplot2/, is another useful resource. This page links to what is new and different in each release. 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.

Finally, the book website, http://had.co.nz/ggplot2/book, provides updates to this book, as well as pdfs containing all graphics used in the book, with the code and data needed to reproduce them.

1.3 What is the grammar of graphics?

Wilkinson (2005) created the grammar of graphics to describe the deep features that underlie all statistical graphics. The grammar of graphics is an answer to a question: what is a statistical graphic? The layered grammar of graphics (Wickham, 2009) builds on Wilkinson's grammar, focussing on the primacy of layers and adapting it for embedding within R. In brief, the grammar tells us that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system. Faceting can be used to generate the same plot for different subsets of the dataset. It is the combination of these independent components that make up a graphic.

As the book progresses, the formal grammar will be explained in increasing 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 responsibilities of each component. Don't worry if it doesn't all make sense right away: you will have many more opportunities to learn about all of the pieces and how they fit together.

- The **data** that you want to visualise and a set of aesthetic **mapping**s describing how variables in the data are mapped to aesthetic attributes that you can perceive.
- Geometric objects, **geom**s for short, represent what you actually see on the plot: points, lines, polygons, etc.
- Statistical transformations, **stat**s for short, summarise data in many useful ways. For example, binning and counting observations to create a histogram, or summarising a 2d relationship with a linear model. Stats are optional, but very useful.
- The **scale**s map values in the data space to values in an aesthetic space, whether it be colour, or size, or shape. Scales draw a legend or axes, which provide an inverse mapping to make it possible to read the original data values from the graph.
- A coordinate system, **coord** for short, describes how data coordinates are mapped to the plane of the graphic. It also provides axes and gridlines to make it possible to read the graph. We normally use a Cartesian coordinate system, but a number of others are available, including polar coordinates and map projections.
- A **facet**ing specification describes how to break up the data into subsets and how to display those subsets as small multiples. This is also known as conditioning or latticing/trellising.

It is also important to talk about what the grammar doesn't do:

- It doesn't suggest what graphics you should use to answer the questions you are interested in. While this book endeavours to promote a sensible

process for producing plots of data, the focus of the book is on how to produce the plots you want, not knowing what plots to produce. For more advice on this topic, you may want to consult Chambers et al. (1983); Cleveland (1993a); Robbins (2004); Tukey (1977).

- Ironically, the grammar doesn't specify what a graphic should look like. The finer points of display, for example, font size or background colour, are not specified by the grammar. In practice, a useful plotting system will need to describe these, as `ggplot2` does with its theming system. Similarly, the grammar does not specify how to make an attractive graphic and while the defaults in `ggplot2` have been chosen with care, you may need to consult other references to create an attractive plot: Tufte (1990, 1997, 2001, 2006).
- It does not describe interaction: the grammar of graphics describes only static graphics and there is essentially no benefit to displaying on a computer screen as opposed to on a piece of paper. `ggplot2` can only create static graphics, so for dynamic and interactive graphics you will have to look elsewhere. Cook and Swayne (2007) provides an excellent introduction to the interactive graphics package GGobi. GGobi can be connected to R with the `rggobi` package (Wickham et al., 2008).

1.4 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 S graphics driver and partly looking at Chambers et al. (1983). 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. When 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 (Murrell, 1998). 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 (Sarkar, 2008a), developed by Deepayan Sarkar, uses grid graphics to implement the trellis graphics system of Cleveland (1993a, 1985) 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 (Sarkar, 2008b).

- `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 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.

Many other R packages, such as `vcd` (Meyer et al., 2006), `plotrix` (Lemon et al., 2008) and `gplots` (Warnes, 2007), implement specialist graphics, but no others provide a framework for producing statistical graphics. A comprehensive resource listing all graphics functionality available in other contributed packages is the graphics task view at `http://cran.r-project.org/web/views/Graphics.html`.

1.5 About this book

Chapter 2 describes how to quickly get started using `qplot` to make graphics, just like you can using `plot`. This chapter introduces several important `ggplot2` concepts: geoms, aesthetic mappings and faceting.

While `qplot` is a quick way to get started, you are not using the full power of the grammar. Chapter 3 describes the layered grammar of graphics which underlies `ggplot2`. The theory is illustrated in Chapter 4 which demonstrates how to add additional layers to your plot, exercising full control over the geoms and stats used within them. Chapter 5 describes how to assemble and combine geoms and stats to solve particular plotting problems.

Understanding how scales works 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. Chapter 6 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 faceting control the position of elements of the plot. These are described in Chapter 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 fine tune your plots for publication, you will need to learn about the tools described in Chapter 8. There you will learn about how to control the theming system of `ggplot2`, how to change the defaults for geoms, stats and scales, how to save plots to disk, and how to lay out multiple plots on a page.

The book concludes with two chapters that discuss high-level concerns about data structure and code duplication. Chapter 9 discusses some techniques that will enable you to get your data into the form required for `ggplot2`, and tools that enable you to perform more advanced aggregation and manipulation than is available in the plotting code. You will also learn about the `ggplot2` philosophy behind visualising other types of objects, and how you can extend `ggplot2` with your own methods.

Duplicated code is a big inhibitor of flexibility and reduces your ability to respond to changes in requirements. Chapter 10 covers three useful techniques for reducing duplication in your code: iteration, plot templates and plot functions.

Three appendices provide additional useful information. Appendix B describes how colours, shapes, line types and sizes can be specified by hand. Appendix A shows how to translate the syntax of base graphics, lattice graphics, and Wilkinson's GPL to `ggplot2` syntax. Appendix C describes the high-level organisation of grid objects and viewports used to draw a `ggplot2` plot. This will be useful if you are familiar with grid, and want to make changes to the underlying objects used to draw the plots.

1.6 Installation

To use `ggplot2`, you must first install it. Make sure you have a recent version of R (at least version 2.8) from `http://r-project.org` and then run the following line of code to download and install the `ggplot2` package.

```
install.packages("ggplot2")
```

`ggplot2` isn't perfect, so from time to time you may encounter something that doesn't work the way it should. If this happens, please email me at hadley@rice.edu with a reproducible example of your problem, as well as a description of what you think should have happened. The more information you provide, the easier it is for me to help you. .

1.7 Acknowledgements

Many people have contributed to this book with high-level structural insights, spelling and grammar corrections and bug reports. In particular, I would like to thank: Leland Wilkinson, for discussions and comments that cemented my understanding of the grammar; Gabor Grothendieck, for early helpful comments; Heike Hofmann and Di Cook, for being great major professors; Charlotte Wickham; the students of stat480 and stat503 at ISU, for trying it out when it was very young; Debby Swayne, for masses of helpful feedback and advice; Bob Muenchen, Reinhold Kliegl, Philipp Pagel, Richard Stahlhut,

Baptiste Auguie, Jean-Olivier Irisson, Thierry Onkelinx and the many others who have read draft versions of the book and given me feedback; and last, but not least, the members of R-help and the `ggplot2` mailing list, for providing the many interesting and challenging graphics problems that have helped motivate this book.