Use R!

Hadley Wickham

ggplot2

Elegant Graphics for Data Analysis

Second Edition

Springer

Hadley Wickham

ggplot2

Elegant Graphics for Data Analysis

Second Edition

With contributions by Carson Sievert

Springer

Hadley Wickham
RStudio
Houston, Texas, USA

ISSN 2197-5736 ISSN 2197-5744 (electronic)
Use R!
ISBN 978-3-319-24275-0 ISBN 978-3-319-24277-4 (eBook)
DOI 10.1007/978-3-319-24277-4

Library of Congress Control Number: 2016937314

Printed on acid-free paper

This springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

*To my parents, Alison & Brian Wickham.
Without them, and their unconditional
love and support, none of this would have
been possible.*

Preface

Welcome to the second edition of “ggplot2: elegant graphics for data analysis”. I’m so excited to have an updated book that shows off all the latest and greatest ggplot2 features, as well as the great things that have been happening in R and in the ggplot2 community the last 5 years. The ggplot2 community is vibrant: the ggplot2 mailing list has over 7,000 members and there is a very active Stack Overflow community, with nearly 10,000 questions tagged with ggplot2. While most of my development effort is no longer going into ggplot2 (more on that below), there’s never been a better time to learn it and use it.

I am tremendously grateful for the success of ggplot2. It’s one of the most commonly downloaded R packages (over a million downloads in the last year!) and has influenced the design of graphics packages for other languages. Personally, ggplot2 has brought me many exciting opportunities to travel the world and meet interesting people. I love hearing how people are using R and ggplot2 to understand the data that they care about.

A big thanks for this edition goes to Carson Sievert, who helped me modernise the code, including converting the sources to R Markdown. He also updated many of the examples and helped me proofread the book.

Major Changes

I’ve spent a lot of effort ensuring that this edition is a true upgrade over the first. As well as updating the code everywhere to make sure it’s fully compatible with the latest version of ggplot2, I have:

- Shown much more code in the book, so it’s easier to use as a reference. Overall the book has a more “knitr”-ish sensibility: there are fewer floating figures and tables and more inline code. This makes the layout a little less pretty but keeps related items closer together.

- Published the complete source online at `https://github.com/hadley/ggplot2-book`.
- Switched from `qplot()` to `ggplot()` in the introduction, Chap. 2. Feedback indicated that `qplot()` was a crutch: it makes simple plots a little easier, but it doesn't help with mastering the grammar.
- Added practice exercises throughout the book so you can practise new techniques immediately after learning about them.
- Added pointers to the rich ecosystem of packages that have built up around ggplot2. You'll now see a number of other packages highlighted in the book and get pointers to other packages I think are particularly useful.
- Overhauled the toolbox chapter, Chap. 3, to cover all the new geoms. I've added a completely new section on text labels, Sect. 3.3, since it's important and not covered in detail elsewhere. The mapping section, Sect. 3.7, has been considerably expanded to talk more about the different types of map data and where you might find them.
- Completely rewritten the scales chapter, Chap. 6, to focus on the most important tasks. It also discusses the new features that give finer control over legend appearance, Sect. 6.4, and shows off some of the new scales added to ggplot2, Sect. 6.6.
- Split the data analysis chapter into three pieces: data tidying (with tidyr), Chap. 9; data manipulation (with dplyr), Chap. 10; and model visualisation (with broom), Chap. 11. I discuss the latest iteration of my data manipulation tools and introduce the fantastic broom package by David Robinson.

The book is accompanied by a new version of ggplot2: version 2.0.0. This includes a number of minor tweaks and improvements, and considerable improvements to the documentation. Coming back to ggplot2 development after a considerable pause has helped me to see many problems that previously escaped notice. ggplot2 2.0.0 (finally!) contains an official extension mechanism so that others can contribute new ggplot2 components in their own packages. This is documented in a new vignette, `vignette(‘‘extending-ggplot2")`.

The Future

ggplot2 is now stable and is unlikely to change much in the future. There will be bug fixes and there may be new geoms, but there will be no large changes to how ggplot2 works. The next iteration of ggplot2 is ggvis. ggvis is significantly more ambitious because it aims to provide a grammar of *interactive* graphics. ggvis is still young and lacks many of the features of ggplot2 (most notably it currently lacks facetting and has no way to make static graphics), but over the coming years the goal is to make ggvis better than ggplot2.

The syntax of ggvis is a little different to ggplot2. You won't be able to trivially convert your ggplot2 plots to ggvis, but we think the cost is worth it: the new syntax is considerably more consistent and will be easier for newcomers to learn. If you've mastered ggplot2, you'll find your skills transfer very well to ggvis and after struggling with the syntax for a while, it will start to feel quite natural. The important skills you learn when mastering ggplot2 are not the programmatic details of describing a plot in code, but the much harder challenge of thinking about how to turn data into effective visualisations.

Acknowledgements

Many people have contributed to this book with high-level structural insights, spelling and grammar corrections and bug reports. I'd particularly like to thank William E.J. Doane, Alexander Forrence, Devin Pastoor, David Robinson and Guangchuang Yu for their detailed technical reviews of the book.

Many others have contributed over the (now quite long!) lifetime of ggplot2. 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 advisors and supporting the development of ggplot2 during my PhD; 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.

Chief Scientist, RStudio — Hadley Wickham
Houston, TX, USA
September 2015

Contents

Part I
Getting Started

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 made up 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 iteratively. You can start with a layer showing the raw data then add 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 not only will help you 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, so most graphics packages are just a big collection of special cases. For example, in

H. Wickham, *ggplot2*, Use R!, DOI 10.1007/978-3-319-24277-4_1

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*.

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 eventually turn you into an expert who can build new components to extend the grammar.

1.2 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, 2010) 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. Facetting 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 the pieces and how they fit together.

All plots are composed of:

- **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.

- **Layers** made up of geometric elements and statistical transformation. 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.
- 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 plot.
- 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.
- A **theme** which 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 (Tufte, 1990, 1997, 2001).

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 Robbins (2013), Cleveland (1993), Chambers et al. (1983), and Tukey (1977).
- It does not describe interactivity: the grammar of graphics describes only static graphics and there is essentially no benefit to displaying them on a computer screen as opposed to a piece of paper. ggplot2 can only create static graphics, so for dynamic and interactive graphics you will have to look elsewhere (perhaps at ggvis, described below). 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).

1.3 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. (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. 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 (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, developed by Deepayan Sarkar, uses grid graphics to implement the trellis graphics system of Cleveland (1993) 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 (2008).
- 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.
- Work on ggvis, the successor to ggplot2, started in 2014. It takes the foundational ideas of ggplot2 but extends them to the web and interactive graphics. The syntax is similar, but it's been re-designed from scratch to take advantage of what I've learned in the 10 years since creating ggplot2. The most exciting thing about ggvis is that it's interactive and dynamic, so plots automatically re-draw themselves when the underlying data or plot specification changes. However, ggvis is work in progress and currently can create only a fraction of the plots in ggplot2 can. Stay tuned for updates!
- 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). htmlwidgets is to ggvis what the many specialised graphic packages are to ggplot2: it provides graphics honed for specific purposes.

Many other R packages, such as vcd (Meyer et al., 2006), plotrix (Lemon et al., 2006) and gplots (Warnes, 2015), 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, Chap. 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. Chapter 3 dives into more details, giving you a toolbox designed to solve a wide range of problems.

Chapter 4 describes the layered grammar of graphics which underlies ggplot2. The theory is illustrated in Chap. 5 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. 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 facetting control the position of elements of the plot. These are described in Chap. 7. Facetting 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 Chap. 8. There you will learn about how to control the theming system of ggplot2 and how to save plots to disk.

The book concludes with four chapters that show how to use ggplot2 as part of a data analysis pipeline. ggplot2 works best when your data is tidy, so Chap. 9 discusses what that means and how to make your messy data tidy. Chapter 10 teaches you how to use the dplyr package to perform the most common data manipulation operations. Chapter 11 shows how to integrate visualisation and modelling in two useful ways. Duplicated code is a big inhibitor of flexibility and reduces your ability to respond to changes in requirements. Chapter 12 covers useful techniques for reducing duplication in your code.

1.5 Installation

To use ggplot2, you must first install it. Make sure you have a recent version of R (at least version 3.2.0) from http://r-project.org and then run the following code to download and install ggplot2:

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

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, http://docs.ggplot2.org/, 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 (https://github.com/jennybc/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 R Markdown (http://rmarkdown.rstudio.com/) inside RStudio (http://www.rstudio.com/ide/). knitr (http://yihui.name/knitr/) and pandoc (http://johnmacfarlane.net/pandoc/) converted the raw Rmarkdown to html and pdf. The complete source is available from github (https://github.com/hadley/ggplot2-book). This version of the book was built with:

```
devtools::session_info(c("ggplot2", "dplyr", "broom"))
#> Session info -------------------------------------------------------
#>  setting  value
#>  version  R version 3.2.3 (2015-12-10)
#>  system   x86_64, darwin13.4.0
#>  ui       X11
#>  language (EN)
#>  collate  en_US.UTF-8
#>  tz       America/Chicago
#>  date     2016-02-27
#> Packages -----------------------------------------------------------
#>  package      * version  date       source
#>  assertthat     0.1      2013-12-06 CRAN (R 3.2.0)
#>  BH             1.58.0-1 2015-05-21 CRAN (R 3.2.0)
#>  broom          0.4.0    2015-11-30 CRAN (R 3.2.2)
#>  colorspace     1.2-6    2015-03-11 CRAN (R 3.2.0)
#>  DBI            0.3.1    2014-09-24 CRAN (R 3.2.0)
#>  dichromat      2.0-0    2013-01-24 CRAN (R 3.2.0)
#>  digest         0.6.9    2016-01-08 CRAN (R 3.2.3)
#>  dplyr        * 0.4.3    2015-09-01 CRAN (R 3.2.0)
#>  ggplot2      * 2.1.0    2016-02-26 local
#>  gtable         0.2.0    2016-02-26 CRAN (R 3.2.3)
#>  labeling       0.3      2014-08-23 CRAN (R 3.2.0)
#>  lattice        0.20-33  2015-07-14 CRAN (R 3.2.3)
#>  lazyeval       0.1.10   2015-01-02 CRAN (R 3.2.0)
#>  magrittr       1.5      2014-11-22 CRAN (R 3.2.0)
#>  MASS           7.3-45   2015-11-10 CRAN (R 3.2.3)
#>  mnormt         1.5-3    2015-05-25 CRAN (R 3.2.0)
#>  munsell        0.4.2    2013-07-11 CRAN (R 3.2.0)
#>  nlme           3.1-122  2015-08-19 CRAN (R 3.2.3)
#>  plyr           1.8.3    2015-06-12 CRAN (R 3.2.0)
#>  psych          1.5.8    2015-08-30 CRAN (R 3.2.0)
#>  R6             2.1.2    2016-01-26 CRAN (R 3.2.3)
#>  RColorBrewer   1.1-2    2014-12-07 CRAN (R 3.2.0)
#>  Rcpp           0.12.3   2016-01-10 CRAN (R 3.2.3)
#>  reshape2       1.4.1    2014-12-06 CRAN (R 3.2.0)
#>  scales         0.4.0    2016-02-26 CRAN (R 3.2.3)
#>  stringi        1.0-1    2015-10-22 CRAN (R 3.2.0)
#>  stringr        1.0.0    2015-04-30 CRAN (R 3.2.0)
#>  tidyr        * 0.4.1    2016-02-05 CRAN (R 3.2.3)
getOption("width")
#> [1] 67
```

References

Chambers J, William C, Beat K, Paul T (1983) Graphical methods for data analysis. Wadsworth, Belmont

Cleveland W (1993) Visualizing data. Hobart Press

Cook D, Deborah FS (2007) Interactive and dynamic graphics for data analysis: with examples using R and GGobi. Springer, New York

Lemon J (2006) Plotrix: a package in the red light district of R. R-News 6(4):8–12

Meyer D, Achim Z, Kurt H (2006) The strucplot framework: visualizing multi-way contingency tables with Vcd. J Stat Softw 17(3):1–48. http://www.jstatsoft.org/v17/i03/

Murrell P (1998) Investigations in graphical statistics. PhD Thesis, The University of Auckland

Robbins N (2013) Creating more effective graphs. Chart House, Wayne

Sarkar D (2008) Lattice: multivariate data visualization with R. Springer, New York

Tufte ER (1990) Envisioning information. Graphics Press, Cheshire

Tufte ER (1997) Visual explanations. Graphics Press, Cheshire

Tufte ER (2001) The visual display of quantitative information, 2nd edn. Graphics Press, Cheshire

Tukey JW (1977) Exploratory data analysis. Addison, Reading

Warnes GR, Bolker B, Bonebakker L, Gentleman R, Liaw WHA, Lumley T, Maechler M, Magnusson A, Moeller S, Schwartz M, Venables B (2015) gplots: various R programming tools for plotting data. R package version 2.17.0. https://CRAN.R-project.org/package=gplots

Wickham H (2010) A layered grammar of graphics. J Comput Graph Stat 19(1):3–28

Wickham H, Michael L, Duncan TL, Deborah FS (2008) An introduction to Rggobi. R-News 8(2):3–7. http://CRAN.R-project.org/doc/Rnews/Rnews_2008-2.pdf

Wilkinson L (2005) The grammar of graphics. Statistics and computing, 2nd edn. Springer, New York

Chapter 2
Getting Started with ggplot2

2.1 Introduction

The goal of this chapter is to teach you how to produce useful graphics with ggplot2 as quickly as possible. You'll learn the basics of `ggplot()` along with some useful "recipes" to make the most important plots. `ggplot()` allows you to make complex plots with just a few lines of code because it's based on a rich underlying theory, the grammar of graphics. Here we'll skip the theory and focus on the practice, and in later chapters you'll learn how to use the full expressive power of the grammar.

In this chapter you'll learn:

- About the `mpg` dataset included with ggplot2, Sect. 2.2.
- The three key components of every plot: data, aesthetics and geoms, Sect. 2.3.
- How to add additional variables to a plot with aesthetics, Sect. 2.4.
- How to display additional categorical variables in a plot using small multiples created by facetting, Sect. 2.5.
- A variety of different geoms that you can use to create different types of plots, Sect. 2.6.
- How to modify the axes, Sect. 2.7.
- Things you can do with a plot object other than display it, like save it to disk, Sect. 2.8.
- `qplot()`, a handy shortcut for when you just want to quickly bang out a simple plot without thinking about the grammar at all, Sect. 2.9.

H. Wickham, *ggplot2*, Use R!, DOI 10.1007/978-3-319-24277-4_2

2.2 Fuel Economy Data

In this chapter, we'll mostly use one data set that's bundled with ggplot2: `mpg`. It includes information about the fuel economy of popular car models in 1999 and 2008, collected by the US Environmental Protection Agency, http://fueleconomy.gov. You can access the data by loading ggplot2:

```
library(ggplot2)
mpg
#> Source: local data frame [234 x 11]
#>
#>    manufacturer model displ  year   cyl      trans   drv   cty
#>           (chr) (chr) (dbl) (int) (int)      (chr) (chr) (int)
#> 1          audi    a4   1.8  1999     4   auto(l5)     f    18
#> 2          audi    a4   1.8  1999     4 manual(m5)     f    21
#> 3          audi    a4   2.0  2008     4 manual(m6)     f    20
#> 4          audi    a4   2.0  2008     4   auto(av)     f    21
#> 5          audi    a4   2.8  1999     6   auto(l5)     f    16
#> 6          audi    a4   2.8  1999     6 manual(m5)     f    18
#> ..          ...   ...   ...   ...   ...        ...   ...   ...
#> Variables not shown: hwy (int), fl (chr), class (chr)
```

The variables are mostly self-explanatory:

- `cty` and `hwy` record miles per gallon (mpg) for city and highway driving.
- `displ` is the engine displacement in litres.
- `drv` is the drivetrain: front wheel (f), rear wheel (r) or four wheel (4).
- `model` is the model of car. There are 38 models, selected because they had a new edition every year between 1999 and 2008.
- `class` (not shown), is a categorical variable describing the "type" of car: two seater, SUV, compact, etc.

This dataset suggests many interesting questions. How are engine size and fuel economy related? Do certain manufacturers care more about fuel economy than others? Has fuel economy improved in the last 10 years? We will try to answer some of these questions, and in the process learn how to create some basic plots with ggplot2.

2.2.1 Exercises

1. List five functions that you could use to get more information about the `mpg` dataset.
2. How can you find out what other datasets are included with ggplot2?
3. Apart from the US, most countries use fuel consumption (fuel consumed over fixed distance) rather than fuel economy (distance travelled with fixed amount of fuel). How could you convert `cty` and `hwy` into the European standard of l/100 km?

4. Which manufacturer has the most the models in this dataset? Which model has the most variations? Does your answer change if you remove the redundant specification of drive train (e.g. "pathfinder 4wd", "a4 quattro") from the model name?

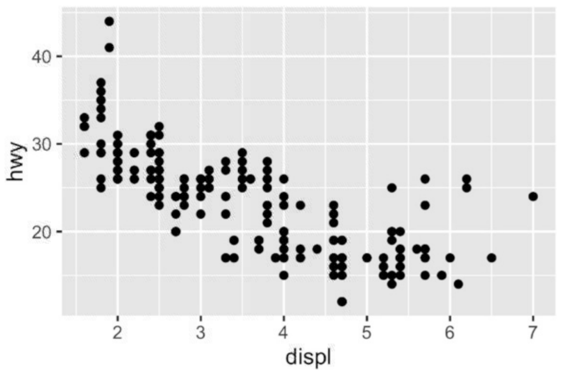
2.3 Key Components

Every ggplot2 plot has three key components:

1. **data**,
2. A set of **aesthetic mappings** between variables in the data and visual properties, and
3. At least one layer which describes how to render each observation. Layers are usually created with a **geom** function.

Here's a simple example:

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point()
```



This produces a scatterplot defined by:

1. Data: `mpg`.
2. Aesthetic mapping: engine size mapped to x position, fuel economy to y position.
3. Layer: points.

Pay attention to the structure of this function call: data and aesthetic mappings are supplied in `ggplot()`, then layers are added on with `+`. This is an important pattern, and as you learn more about ggplot2 you'll construct increasingly sophisticated plots by adding on more types of components.