



Visualizing Data in R 4

Graphics Using the base, graphics,
stats, and ggplot2 Packages

Margot Tollefson

Apress®

Visualizing Data in R 4

Graphics Using the base, graphics,
stats, and ggplot2 Packages

Margot Tollefson

Apress®

Visualizing Data in R 4: Graphics Using the base, graphics, stats, and ggplot2 Packages

Margot Tollefson
Stratford, IA, USA

ISBN-13 (pbk): 978-1-4842-6830-8
<https://doi.org/10.1007/978-1-4842-6831-5>

ISBN-13 (electronic): 978-1-4842-6831-5

Copyright © 2021 by Margot Tollefson

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Steve Anglin
Development Editor: Matthew Moodie
Coordinating Editor: Mark Powers

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484268308. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

For Clay.

Table of Contents

About the Author xi

About the Technical Reviewer xiii

Acknowledgments xv

Part I: An Overview of plot()..... 1

Chapter 1: Introduction: plot(), qplot(), and ggplot(), Plus Some 3

1.1 plot(), par(), layout(), and split.screen() 3

1.2 qplot() and ggplot()..... 4

1.3 The Appendixes 4

1.4 Software Versions and Hardware Used in This Book 5

1.5 Graphics Devices 7

Chapter 2: The plot() Function..... 9

2.1 Arguments and Default Values 9

2.2 Ancillary Functions..... 10

2.3 Methods 10

2.4 The Graphics Devices and the Functions par(), layout(), and split.screen()..... 10

2.5 An Example 11

Chapter 3: The Arguments of plot()..... 13

3.1 The Dataset 13

3.2 Changing the Overall Appearance in plot() 15

3.2.1 Labels and Axis Limits 16

3.2.2 Box Type, Aspect Ratio, Annotation, and Expanded Plotting 19

3.3 Points and Lines..... 24

3.3.1 Types of Plots..... 25

3.3.2 The Arguments pch and lty 28

TABLE OF CONTENTS

3.4 Details	31
3.4.1 Colors	31
3.4.2 Fonts and Font Families	38
3.4.3 Character Size in plot().....	42
3.4.4 Line Details: lwd, lend, ljoin, and lmitre.....	46
3.4.5 Making Changes to the Axes	49
3.4.6 Working with Log Scales	58
Chapter 4: Ancillary Functions for plot().....	61
4.1 Functions That Affect Overall Appearance	61
4.1.1 The title() Function	62
4.1.2 The axis() and axTicks() Functions	67
4.1.3 The box(), grid(), clip(), and rug() Functions	79
4.2 Functions Defined at Points	88
4.2.1 The points() Function	88
4.2.2 The text() Function	90
4.2.3 The symbols() Function	99
4.2.4 The image() and rasterImage() Functions.....	104
4.3 Functions That Use Lines	114
4.3.1 The lines() and abline() Functions.....	115
4.3.2 The curve() Function	118
4.3.3 The segments() and arrows() Functions	121
4.3.4 Functions That Plot Lines That Close on Themselves: rect(), polygon(), and polypath()	124
4.3.5 The contour() Function	131
4.4 Functions to Provide Information About or to Interact with a Plot	136
4.4.1 The legend() and mtext() Functions	136
4.4.2 The Interactive Functions: identify() and locator()	151

Chapter 5: The Methods of plot()	159
5.1 Methods	159
5.2 The Methods for plot() in the graphics Package	160
5.2.1 The data.frame Method	161
5.2.2 The factor Method	163
5.2.3 The formula Method	166
5.2.4 The function Method.....	168
5.2.5 The histogram Method.....	170
5.2.6 The raster Method	174
5.2.7 The table Method.....	181
5.3 The Methods for plot() in the stats Package	184
5.3.1 The acf Method	186
5.3.2 The decomposed.ts Method	191
5.3.3 The dendrogram Method	193
5.3.4 The density Method	198
5.3.5 The ecdf Method.....	200
5.3.6 The hclust Method	203
5.3.7 The HoltWinters Method	206
5.3.8 The isoreg Method.....	209
5.3.9 The lm Method.....	212
5.3.10 The ppr Method	219
5.3.11 The prcomp and princomp Methods.....	222
5.3.12 The profile.nls Method	225
5.3.13 The spec, spec.coherency, and spec.phase Methods	229
5.3.14 The stepfun Method.....	234
5.3.15 The stl Method	237
5.3.16 The ts Method.....	240
5.3.17 The tskernel Method.....	247

Chapter 6: Graphics Devices and Laying Out Plots..... 251

6.1 Graphics Devices and Working with Graphics Devices 251

6.1.1 The Graphics Devices 251

6.1.2 Working with Graphics Devices 253

6.2 The par(), layout(), and split.screen() Functions 257

6.2.1 The par() Function 257

6.2.2 The layout(), layout.show(), and lcm() Functions..... 269

6.2.3 The split.screen(), screen(), erase.screen(), and close.screen() Functions 272

Part II: A Look at the ggplot2 Package 279

Chapter 7: Graphics with the ggplot2 Package: An Introduction 281

7.1 The Language and Syntax Used in the ggplot2 Package 282

7.2 The qplot() Function..... 282

7.3 An Overview of the ggplot() Function 290

Chapter 8: Working with the ggplot() Function: The Theme and the Aesthetics 295

8.1 The Theme Functions 295

8.1.1 The theme() Function 295

8.1.2 The Preset Theme Functions 304

8.1.3 Working with Themes 307

8.2 The Aesthetic Functions 315

Chapter 9: The Geometry, Statistic, Annotation, and borders() Functions 323

9.1 The Geometry Functions 323

9.2 The Statistic Functions..... 325

9.3 The Annotation Functions..... 330

9.3.1 The annotate() Function 330

9.3.2 The annotation_custom() and ggplotGrob() Functions 334

9.3.3 The annotation_logticks() Function..... 337

9.3.4 The annotation_map() and annotation_raster() Functions 340

9.4 The borders() Function..... 343

Chapter 10: Formatting and Plot Management Tools	347
10.1 Working with the <code>scale_</code> , <code>coord_</code> , and <code>guide_</code> Functions	347
10.1.1 The Scale Functions That Affect Color, Size, Shape, and Line Type.....	348
10.1.2 Setting the Order of Evaluation.....	360
10.1.3 Formatting Axes with the Scale and Coordinate Functions, Plus Some	362
10.1.4 The Guide and Draw Key Functions.....	365
10.2 Functions That Cut, Summarize, and Facet.....	368
10.2.1 The Cut Functions.....	368
10.2.2 The Summary Functions and the <code>resolution()</code> Function.....	369
10.2.3 The Facet Functions	372
10.3 Working with Plots, Automatic Plots, and Prototypes.....	376
10.3.1 The <code>ggsave()</code> Function and the <code>plot()</code> and <code>print()</code> Functions Applied to ggplot Objects	377
10.3.2 The <code>autoplot()</code> and <code>autolayer()</code> Functions	378
10.3.3 Prototype Functions in the <code>ggplot2</code> Package.....	379
Part III: Appendixes	381
Appendix A: Plots for Contingency Tables and Discrete Data.....	383
Appendix B: Plots for Continuous Variables	385
Appendix C: Functions That Plot Multiple Plots.....	387
Appendix D: Smoothers	389
Appendix E: Plots for Time Series.....	391
Appendix F: Miscellaneous Plotting Functions.....	393
Index.....	395

About the Author

Margot Tollefson, PhD, is a semiretired freelance statistician, with her own consulting business, Vanward Statistics. She received her PhD in statistics from Iowa State University and has many years of experience applying R to statistical research problems. Dr. Tollefson has chosen to write this book because she often creates graphics using R and would like to share her knowledge and experience. Her professional blog is on WordPress at [vanwardstat](http://vanwardstat.com). She has social media accounts on LinkedIn, Facebook, and Twitter. Her social media name is @vanstat on Twitter.

About the Technical Reviewer

Tom Barker is an engineer and technology leader, a professor, and an author. He has authored several books on web development, data visualization, and technical leadership, including *High Performance Responsive Design*, *Intelligent Caching*, *Pro JavaScript Performance: Monitoring and Visualization*, and *Pro Data Visualization Using R and JavaScript*.

Acknowledgments

I would like to acknowledge the Comprehensive R Archive Network (CRAN). Without the R documentation, for which the Comprehensive R Archive Network is responsible, this book could not have been written. Also, my husband, Clay, for bearing with me while I wrote.

PART I

An Overview of plot()

CHAPTER 1

Introduction: `plot()`, `qplot()`, and `ggplot()`, Plus Some

R provides many ways to visualize data. Graphics in the R language are generated by functions. Some functions create useful visualizations almost instantly. Other functions combine together to create highly coded sophisticated images. This book shows you how to generate both types of objects.

In the first part of this book, we look in detail at the `plot()` function – the most basic and versatile of the plotting functions. The functions `par()`, `layout()`, and `split.screen()`, which set global plotting parameters and layout options, are also described, as well as graphics devices.

The second part covers the functions in the `ggplot2` package, starting with `qplot()` and `ggplot()`. The functions `qplot()` and `ggplot()` are simpler to use in many ways than the earlier R functions. Truthfully, the syntax used in the `ggplot2` package requires long strings of names and arguments – but the autocomplete function in RStudio makes code entry quite easy.

The third part of the book includes six appendixes containing the canned plotting functions in the `graphics` and `stats` packages. The appendixes are sorted by the type of object to be displayed.

1.1 `plot()`, `par()`, `layout()`, and `split.screen()`

The function `plot()` takes an object or objects and creates an image based on the class of the object(s). There are many arguments to `plot()` that affect the appearance of the image. The arguments can be given values within the call to `plot()`, or many can be assigned globally using the function `par()`. After the initial call to `plot()`, there are several ancillary functions that can be used to add to the image. If having more than one plot on a page is the goal of the user, the functions `par()`, `layout()`, or `split.screen()` can be used to set up the format for multiple plots.

In Chapter 2, we describe the basics of the `plot()` function. In Chapter 3, the various arguments to `plot()` are categorized. The ancillary functions for `plot()` are introduced in Chapter 4. In Chapter 5, the methods (types of plots) for which `plot()` is defined are presented. In Chapter 6, the possible arguments to the function `par()` are described, and a way to set up multiple plots using `par()`, `layout()` or `split.screen()` is given. Also, graphics devices are covered.

1.2 `qplot()` and `ggplot()`

The functions `qplot()` and `ggplot()` in the `ggplot2` package provide alternatives to `plot()`. Default plots generated by `qplot()` and `ggplot()` tend to look nicer than default plots generated by `plot()`. For simple plots, `qplot()` is sufficient to give an elegant-looking graphic. The function `ggplot()` provides for more plotting options than `qplot()`. Some of the syntax used in the `ggplot2` package is not used in standard R.

In `ggplot2`, a theme for a graphic can be defined using `theme()` or by a canned theme function beginning with “`theme_`”. The objects to be plotted are “mappings” and are assigned in the function `aes()`. The type of image to be displayed is a geometry or statistic and is assigned by functions beginning with “`geom_`” or “`stat_`”. The appearance of the graphic can be changed by using `aes()`, “`aes_`” functions, “`geom_`” functions, “`stat_`” functions, and/or other formatting functions. More than one aesthetic, geometry, and statistic can be used to create an image. The appearance of the image can be changed by running a formatting function from within another function or by running a formatting function separately.

In Chapter 7, we look at `qplot()`, `ggplot()`, and the syntax of the `ggplot2` package. Chapter 8 introduces the `theme()`, “`theme_`”, and “`element_`” functions, as well as the `aes()` and “`aes_`” functions. In Chapter 9, geometry, statistic, annotate, and the `borders()` functions are described. Chapter 10 goes over various functions in the `ggplot2` package that also change the appearance of the image.

1.3 The Appendixes

Other than `plot()`, `qplot()`, and `ggplot()`, which are in the base and `ggplot2` packages, there are many plotting functions in the graphics and stats packages. The functions are useful for data cleaning, data exploration, and/or model fitting. For many of the functions, the graphical arguments used by `plot()` can be assigned. The specialized

plotting functions are given in the appendixes at the end of this book, along with brief descriptions of what the functions do and how to use them.

Appendix [A](#) lists functions used with contingency tables. Appendix [B](#) gives functions for continuous variables. Appendix [C](#) lists functions that generate multiple plots. Appendix [D](#) gives functions that smooth data. Appendix [E](#) gives plotting functions used in time series analysis. Appendix [F](#) lists the plotting functions that are in the stats and graphics packages and not covered in the first five appendixes.

1.4 Software Versions and Hardware Used in This Book

The versions of R used in this book are R 4.0.1 and R 4.0.3; the versions of RStudio are 1.3.595 and 1.3.1093. Since R and RStudio are constantly changing, the Comprehensive R Archive Network (CRAN) provides news on changes to R. The news for R 4.0.1 and R 4.0.0 can be found at <https://cran.r-project.org/doc/manuals/r-release/NEWS.pdf>. The beginning of that news pdf is shown in Figure [1-1](#).

NEWS for R version 4.0.1 (2020-06-06)

NEWS

R News



CHANGES IN R 4.0.1

NEW FEATURES:

- `paste()` and `paste0()` gain a new optional argument `recycle0`. When set to true, zero-length arguments are recycled leading to `character(0)` after the `sep`-concatenation, i.e., to the empty string `""` if `collapse` is a string and to the zero-length value `character(0)` when `collapse = NULL`.
A package whose code uses this should depend on 'R (>= 4.0.1)'.
- The `summary(<warnings>)` method now maps the counts correctly to the warning messages.

BUG FIXES:

- `aov(frm1, ...)` now also works where the formula deparses to more than 500 characters, thanks to a report and patch proposal by Jan Hauffa.
- Fix a dozen places (code, examples) as `Sys.setlocale()` returns the new rather than the previous setting.
- Fix for adding two complex `grid` units via `sum()`. Thanks to Gu Zuguang for the report and Thomas Lin Pedersen for the patch.
- Fix `parallel::mclapply(..., mc.preschedule=FALSE)` to handle raw vector results correctly. [PR#17779](#)
- Computing the `base` value, i.e., 2, "everywhere", now uses `FLT_RADIX`, as the original 'machar' code looped indefinitely on the ppc64 architecture for the `longdouble` case.
- In R 4.0.0, `sort.list(x)` when `is.object(x)` was true, e.g., for `x <- I(letters)`, was accidentally using `method = "radix"`. Consequently, e.g., `merge(<data.frame>)` was much slower than previously; reported in [PR#17794](#).
- `plot(y ~ x, ylab = quote{y[i]})` now works, as e.g., for `xlab`; related to [PR#10525](#).
- `parallel::detect.cores(all.tests = TRUE)` tries a matching OS name before the other tests (which were intended only for unknown OSes).

Figure 1-1. R NEWS screenshot

The computer used for the examples is a MacBook Air running macOS Catalina version 10.15.5.

1.5 Graphics Devices

R opens graphics objects in a graphics device. By default, R opens the graphics object on the computer screen, but the object can be written to a file using one of several image formats. Section 6.1 covers graphics devices.

R and RStudio provide ways to save graphics objects to image files by selecting links in the menus of the two programs. R automatically opens and closes the relevant devices when a link is used.

To work with graphics devices through code, see Section 6.1 or the R help pages for “device” and `dev.cur()`. Both Section 6.1.1 and the help page for “device” have a list of the functions on your device that open a graphics device to create a specific type of image file. Section 6.1.2 and the help page for `dev.cur()` give a list of the functions that manage graphics devices.

CHAPTER 2

The `plot()` Function

The `plot()` function in R creates a graphic from objects of specific R classes. A figure resulting from a call to `plot()` can contain text, lines, points, and/or images, and the areas of the graphic can be filled by colors or patterns. The kind of graphic displayed depends on the class of the object(s) to be displayed. For example, a single time series (an object of class `ts`) gives a line plot that is plotted over time.

2.1 Arguments and Default Values

By default, a graphic usually has black lines and text – with preset line, point, and text sizes and weights. Arguments that change the graphical properties of the graphic can be set in `plot()`. Some of the arguments are used to make changes to line, point, or text color; to line width or style; to point or text size; to the plotting character; to the style and font weight of text; and to fill colors or patterns.

Other arguments set alternative text for the axis labels or give a main title and a subtitle to the figure. The orientation, style, and weight of the text in the titles or labels can be changed.

Axes can be included or not included in the graphic created by `plot()`. If axes are initially included, the axis color and the color of tick marks can be changed by arguments within `plot()`. Axis line width, tick mark length, and tick mark spacing can be changed.

Axis tick labels can be assigned in the call to `plot()`. The color, size, and orientation of the axis tick labels can be changed. If necessary, a blank graphics object can be generated with `plot()`. In Chapter 3, we look closely at the arguments that are available to `plot()`.

2.2 Ancillary Functions

After the initial call to `plot()`, graphical information can be added to the original graphic by using ancillary functions. These functions are used to overlay other plots over the original plot and to add annotation to a plot. For example, a regression line can be added to a scatterplot, or a legend can be included in a plot.

There are several ancillary functions. Titles and axis labels can be added by using the ancillary function `title()` – instead of including titles and axis labels in the original call to `plot()`. Axes can be added with the function `axis()` if axes are suppressed in a call to `plot()`. Regression lines can be added to a graphic (in a few ways).

Most of the ancillary functions are eponymous, such as `text()`, `points()`, `lines()`, `segments()`, and `arrows()`. Others are not, such as `polypath()` or `clip()`. In Chapter 4, we list the ancillary functions in the graphics and stats packages and show how each function is used.

2.3 Methods

The methods of a function are those classes of objects for which a function is defined. In the graphics and stats packages, there are 29 methods defined for `plot()`.

In Chapters 3 and 4, we usually use the version of `plot()` that takes an `x`, and possibly a `y`, as the object(s) to be plotted and which plots a scatterplot of `x` against index values or of `y` against `x`. The actual name of the function is `plot.default()`.

However, since R automatically determines the method to use when running `plot()`, the “default” extension is not necessary in the call to `plot()`. For `plot.default()`, the `x` and `y` would need to be equal-length vectors that can be coerced to numeric. The methods for `plot()` are given in Chapter 5, along with what each method creates.

2.4 The Graphics Devices and the Functions `par()`, `layout()`, and `split.screen()`

R plots are created in graphics devices. A graphics device can be opened on the screen of a computer or in a file external to R. (Some graphics devices are specific to a given operating system.) The arguments used by the plotting functions that are covered in Part 1 can be assigned in the `plot()` function and in the ancillary functions. When assigned in a plotting function, the arguments are used in the specific function in which the arguments are assigned. To globally assign arguments, the arguments can be assigned in the `par()` function.

The `par()` function contains the default values that many of the arguments to `plot()` and the ancillary functions use. The default arguments of `par()` can be changed – for a given R session or within a function call – by a call to `par()`. Most of the arguments in `par()` are the same as those arguments in `plot()` that affect the appearance of the graphic. Some arguments in `par()` can only be set in `par()`.

R allows multiple plots to be put in one graphic. A grid for multiple plots can be created using one of two arguments of `par()`. The grid created by `par()` has the same number of columns in each row. Alternatively, the function `layout()` can be used to create a more flexible design, with differing numbers of columns in each row. The `split.screen()` function allows for placing plots at different locations on a graphics device.

In Chapter 6, the types of graphics devices are listed, as well as ways to work with graphics devices. More on `par()`, `layout()`, and `split.screen()` is also found in Chapter 6.

2.5 An Example

In Figure 2-1, an example is given of a plot of the `sunspot.year` time series (from the `datasets` package) using default arguments and the same plot done with some arguments set. The two plots are plotted in one figure.

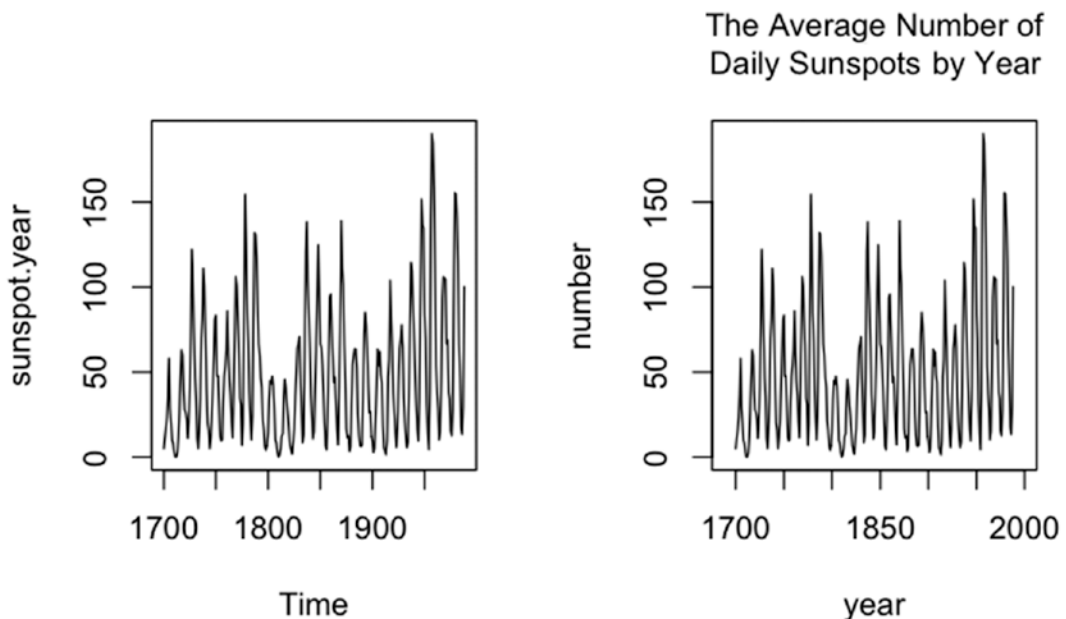


Figure 2-1. Plots of the time series `sunspot.year` (found in the `datasets` package in R). The first plot uses the default argument settings in `plot()`. In the second plot, some of the arguments are set.

The class of `sunspot.year` is `ts`, so the values in the time series are plotted against time. The default axis labels are Time on the x axis and the name of the object that is plotted on the y axis. By default, no title or subtitle is plotted.

For the preceding figure, the function `par()` is used to put two plots in a row into one figure. After `plot()` is run twice, the number of plots per figure is changed back to one by running `par()` again. Changes to `par()` remain in effect throughout an R session, unless changed.

CHAPTER 3

The Arguments of `plot()`

Many arguments can be used in `plot()`. In this chapter, we go over the arguments. The arguments are grouped by categories – arguments that affect overall appearance, the appearance of lines and points, and details. We look at the effects on the appearance of graphics by using examples, with data from an R dataset. In this chapter, the function `plot.default()`, which plots scatterplots, is used for the examples. While most of the arguments in this chapter can be used in all or most versions of `plot()`, a few are specific to `plot.default()`.

3.1 The Dataset

In this section, we work with the `LifeCycleSavings` dataset from the `datasets` package. Since the `datasets` package is loaded by default in RStudio, for most users the dataset is available to access. To bring the dataset into the workspace in order to look at the contents, enter the following at the R prompt:

```
data( "LifeCycleSavings" )
```

To look at the dataset in RStudio, double-click “LifeCycleSavings” (in the Data section of the Environment window in the upper-right pane). The Console (lower-left window) will display

```
> force(LifeCycleSavings)
      sr pop15 pop75      dpi  ddpi
Australia  11.43 29.35  2.87 2329.68  2.87
Austria    12.07 23.32  4.41 1507.99  3.93
```

(Here, only the first two observations are displayed.) The dataset will also appear in the Source (upper-left) window.

A description of the dataset from the R documentation follows:

Intercountry Life-Cycle Savings Data

Description

Data on the savings ratio 1960–1970.

Usage

LifeCycleSavings

Format

A data frame with 50 observations on 5 variables.

[,1]	sr	numeric	aggregate personal savings
[,2]	pop15	numeric	% of population under 15
[,3]	pop75	numeric	% of population over 75
[,4]	dpi	numeric	real per-capita disposable income
[,5]	ddpi	numeric	% growth rate of dpi

Details

Under the life-cycle savings hypothesis as developed by Franco Modigliani, the savings ratio (aggregate personal saving divided by disposable income) is explained by per-capita disposable income, the percentage rate of change in per-capita disposable income, and two demographic variables: the percentage of population less than 15 years old and the percentage of the population over 75 years old. The data are averaged over the decade 1960–1970 to remove the business cycle or other short-term fluctuations.

Source

The data were obtained from Belsley, Kuh and Welsch (1980). They in turn obtained the data from Sterling (1977).

References

Sterling, Arnie (1977) Unpublished BS Thesis. Massachusetts Institute of Technology.

Belsley, D. A., Kuh. E. and Welsch, R. E. (1980) Regression Diagnostics. New York: Wiley.

—R documentation for the datasets package

The five variables in the dataset are aggregate personal savings (sr), the percentage of the population under 15 years of age (pop15), the percentage of the population over 75 years of age (pop75), real per-capita disposable income (dpi), and the percentage growth rate of real per-capita disposable income (ddpi). The values of the variables are averages taken over the years 1960–1970, and the averages were found for 50 countries.

3.2 Changing the Overall Appearance in plot()

The first argument to plot() is always x – an R object of an appropriate class. If x (or x and y) is a vector object of the numeric class, plot() plots a scatterplot. (For scatterplots, a numeric vector y of the same length as x can be included but is not necessary.) The first example is a scatterplot using the default values of the arguments of plot().

In Listing 3-1, code that plots a default scatterplot is given. The scatterplot is a plot of the percentage of the population under 15 (y) against the percentage of the population over 75 (x), both from the LifeCycleSavings dataset.

For data frames, a variable in the data frame can be accessed by the data frame name followed by a dollar sign followed by the variable name. For example, LifeCycleSavings\$pop75 accesses the pop75 variable in the LifeCycleSavings data frame.

Listing 3-1. The code for a default scatterplot

```
plot(
  LifeCycleSavings$pop75,
  LifeCycleSavings$pop15
)
```

The plot is in Figure 3-1.

Example of a Default Plot

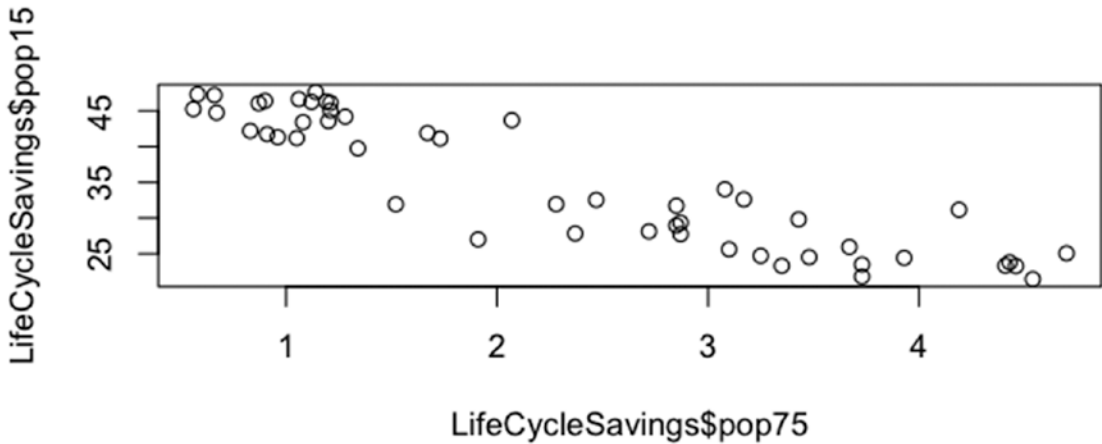


Figure 3-1. Default scatterplot of two variables from the *LifeCycleSavings* dataset

Note that the first argument is on the x axis (horizontal axis) and the second on the y axis (vertical axis). Also, the graphic does not contain a title or subtitle, and the axis labels contain the names of the variables in the dataset. The points are scattered so that the most extreme points are close to the axes. In this chapter, we will show how to change the appearance of, mainly, this plot by setting arguments.

3.2.1 Labels and Axis Limits

A title and subtitle can be added to the plot, and the axis labels can be changed. To add a title, we use the argument `main`. The title will display above the plot. To add a subtitle, we use the argument `sub`. The subtitle will display below the x axis label. To specify the x and y axis labels, we use the arguments `xlab` and `ylab`. Setting either to `""` will suppress the label. For all of the four arguments, the value is a character vector that is usually one element long.

Line breaks can be added to a character string by including “\n” in the character string at the location of the line break. Or a character vector with more than one element puts each element on a separate line (except with the argument `sub`).

The axis limits are the values from which the axis starts and at which the axis ends. Having the start value greater than the end value is an accepted option. To set the x axis and y axis limits of the plot, we use the arguments `xlim` and `ylim`. The format for the two arguments is a numeric vector of length two, where the first number is the beginning limit and the second number is the ending limit. By default, points outside the limits do not plot.

An example of setting the labels and axis arguments is seen in Listing 3-2.

Listing 3-2. Code to plot a scatterplot where the title, subtitle, x axis label, y axis label, x limits, and y limits have been set

```
plot(
  LifeCycleSavings$pop75,
  LifeCycleSavings$pop15,
  main="Percentage under 15 versus Percentage over 75\nfor 50 Countries",
  sub="More Under 15 Means Less Over 75 on Average",
  xlab="Percentage over 75",
  ylab="Percentage under 15",
  xlim=c( 0, 5.25 ),
  ylim=c( 18, 52 )
)
```

In Figure 3-2, the code has been run.

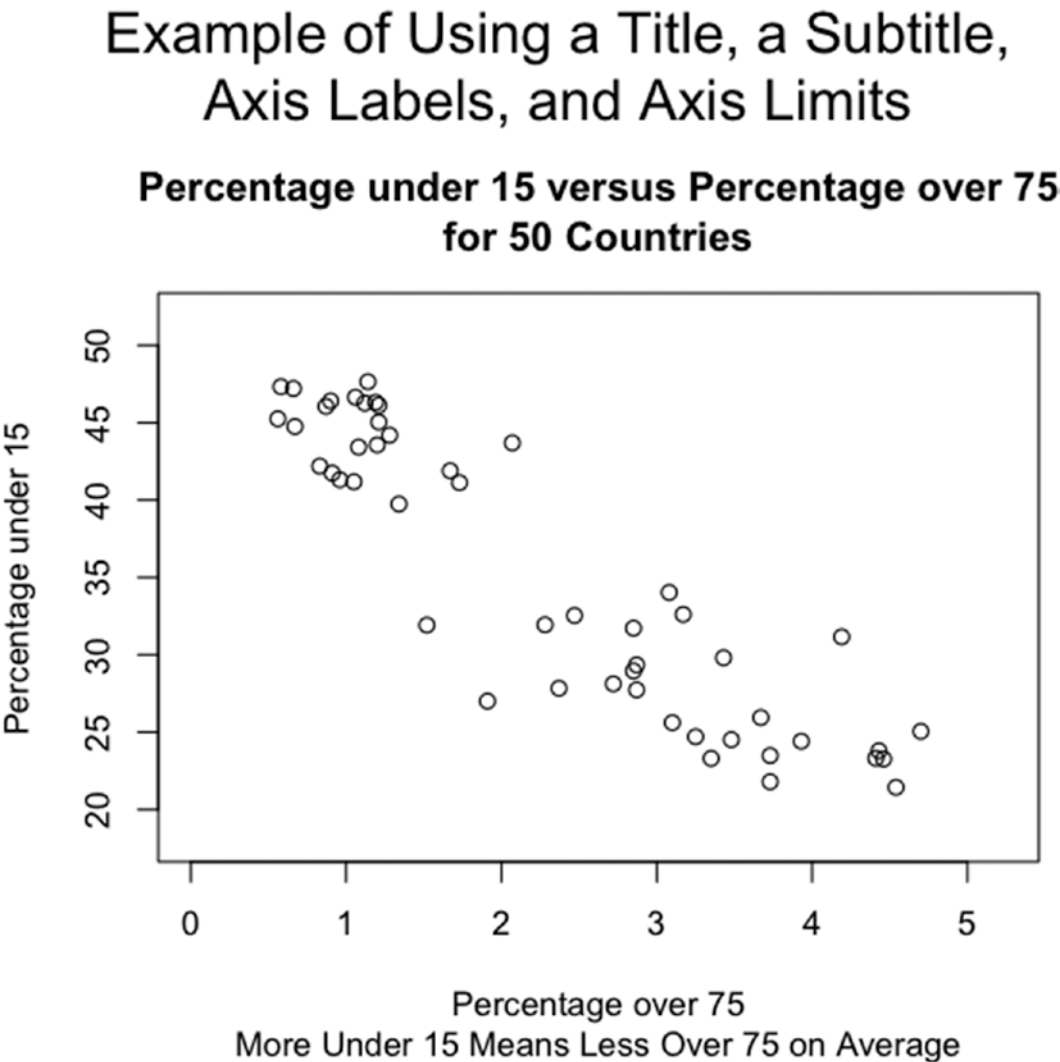


Figure 3-2. Scatterplot plotted using the arguments *main*, *sub*, *xlab*, *ylab*, *xlim*, and *ylim*

3.2.2 Box Type, Aspect Ratio, Annotation, and Expanded Plotting

Whether to include a border, the type of border to include, and the aspect ratio of a plot can be changed. Also, annotation can be shut off, and points can be plotted outside the limits of the plot. The relevant arguments are `frame.plot` for including a border, `bty` for the box type, `asp` for the aspect ratio, `ann` for annotation, and `xpd` for expanded plotting.

The argument `frame.plot` is a logical single-value argument that tells R whether to put a frame around the plot. If the argument is a vector of length greater than one, only the first value is used, and a warning is given. The default value is `TRUE`.

If `frame.plot` is `TRUE`, the argument `bty` describes the type of box. The argument takes character values and can take on the values “o”, “l”, “7”, “c”, “u”, “]”, and “n”. The value “o” is the default and indicates a four-sided box. The first six values (the capital letter for the letters) look like the shape of the box that the value creates.

The value “l” plots the left and bottom axes; the value “7” plots the right and top axes; the value “c” plots the top, left, and bottom axes; the value “u” plots the left, bottom, and right axes; and the value “]” plots the bottom, right, and top axes. The value “n” indicates to not draw a box. Figure 3-3 shows the six options other than the default, “o”.

The following code created the six plots in Figure 3-3.

Listing 3-3. Plotting with `bty` set to six different values

```
plot( LifeCycleSavings$pop75, LifeCycleSavings$pop15, bty="l", main="bty = l" )
plot( LifeCycleSavings$pop75, LifeCycleSavings$pop15, bty="7", main="bty = 7" )
plot( LifeCycleSavings$pop75, LifeCycleSavings$pop15, bty="c", main="bty = c" )
plot( LifeCycleSavings$pop75, LifeCycleSavings$pop15, bty="u", main="bty = u" )
plot( LifeCycleSavings$pop75, LifeCycleSavings$pop15, bty="]", main="bty = ]" )
plot( LifeCycleSavings$pop75, LifeCycleSavings$pop15, bty="n", main="bty = n" )
```

In Figure 3-3 are the plots generated by the code in Listing 3-3.

Examples of Changing bty

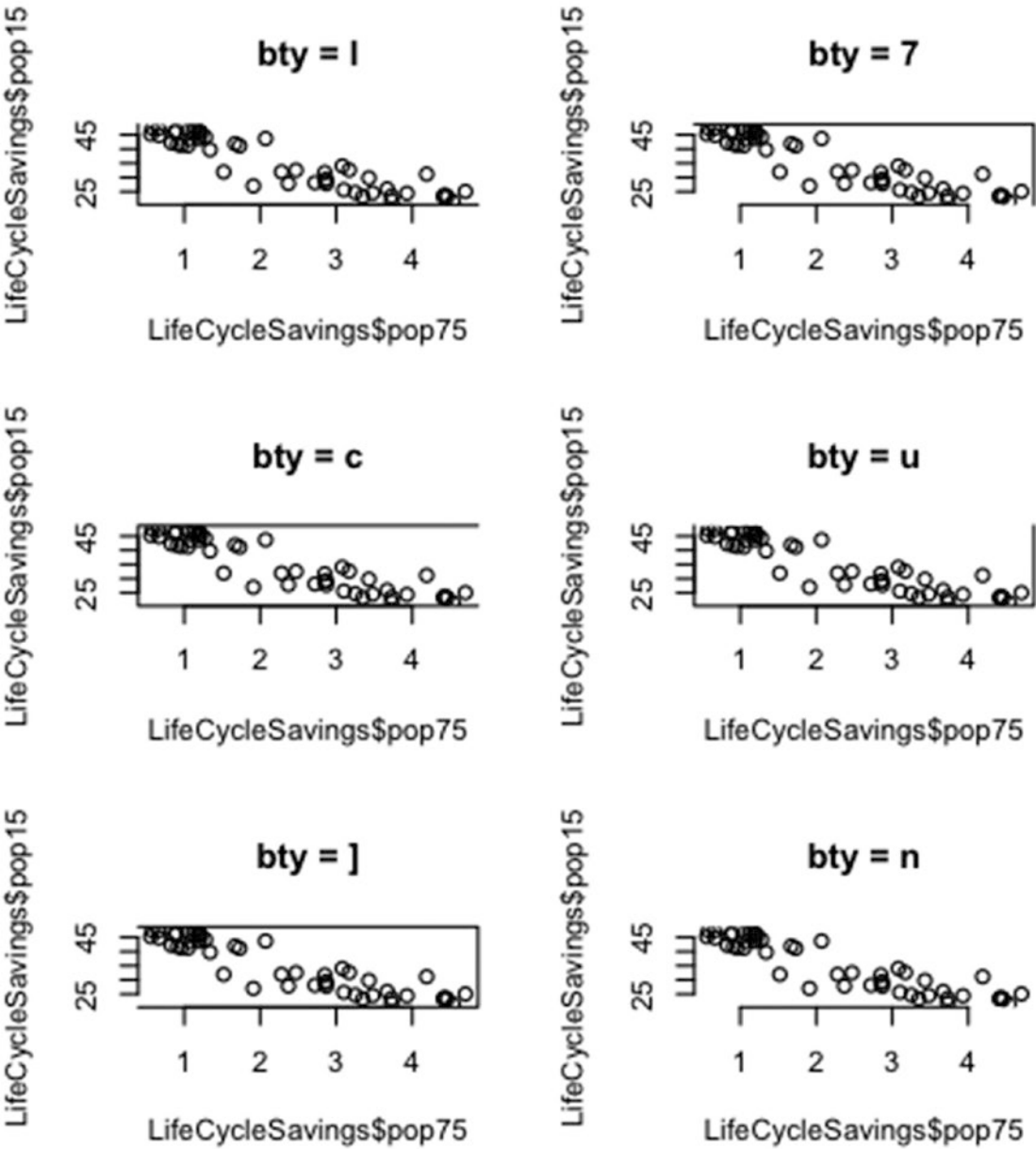


Figure 3-3. Box style examples for bty equal to “l”, “7”, “c”, “u”, “]”, and “n”