Data Visualization for Social Scientists

Using R!

Alfredo Hernandez Sanchez, PhD

2024-04-05

Table of contents

Welcome

This book offers a gentle introduction to data visualization using R and – occasionally – python. In this book you will learn how to make beautiful, informative, and reproducible visualizations with examples from different social sciences. It is inspired by a course on Data Visualization that I taught at the Barcelona Institute of International Studies. Thus, a special thanks goes out to the many graduate students whose efforts, questions, and feedback helped greatly improve the content of this book!

Contact

This book is in open review. If you have any questions, comments or suggestions; please contact me by email or report an issue on GitHub.

Build Information

This book was built using R version 4.4.0 (2024-04-24). For specific package versions please see the session information below:

```
R version 4.4.0 (2024-04-24)
Platform: aarch64-apple-darwin20
Running under: macOS Sonoma 14.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  L

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Europe/Vilnius
tzcode source: internal

attached base packages:
```

```
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.4.0    fastmap_1.1.1     cli_3.6.3         tools_4.4.0
 [5] htmltools_0.5.8.1 rstudioapi_0.16.0 rmarkdown_2.26    knitr_1.46
 [9] jsonlite_1.8.8    xfun_0.43         digest_0.6.35     rlang_1.1.4
[13] evaluate_0.23
```

License

1 Introduction

In an experiment conducted by researchers from University College London (Mcmanus and Gesiak 2014), 277 participants were asked to look at several pairs of paintings: one of the pairs was an original by abstract painter Piet Mondrian, and the other was fake version that closely resembled it.[1] The participants where asked:

> When looking at the pictures you should decide overall which you thinks looks better, in that it looks nicer, it looks better organised, or it looks better balanced.

The results suggested that people could identify the originals with some degree of accuracy (μ 54.7%, SE .40). This experiment "implies people know something about what makes a real Mondrian." In other words, we have an *intuition* of proportion and beauty.

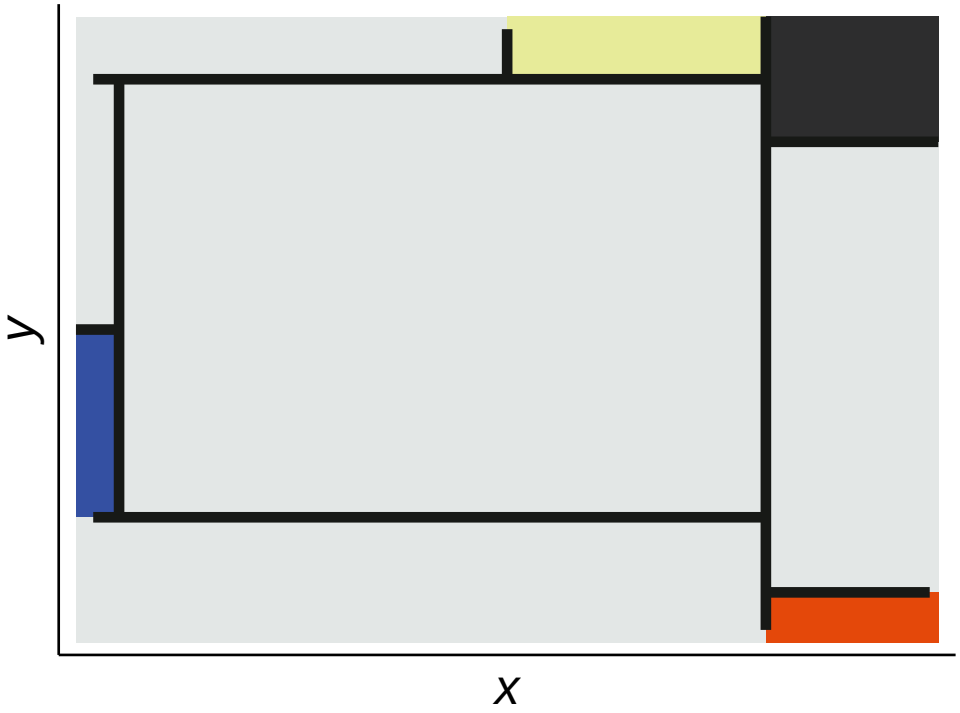

Figure 1.1: An *attempt* at a Mondrian using `ggplot2`

[1] The pseudo-Mondrians were created by jittering all the lines in the original but keeping the same relative positions.

One of the most fun things about data visualization is making things that are beautiful. In making them, you learn a lot about not just the patterns you are displaying, but the scope and limitations of the underlying data. Having taught courses on data viz for several years, I have found that there is a correlation between the effort put into making a plot beautiful and how accurately it conveys the intended message.

Why Visualize?

Data visualization plays numerous roles in the social sciences, from summarizing large amounts of information in a small space, to supporting claims about patterns and relationships among a vast array of indicators of human behavior.

Zinovyev (2010) identifies four types of visualizations in political science:

- *Statistical graphics and infographics* with extensive use of color, form, size, shape and style to superimpose many quantitative variables in the same chart or diagram
- *Geographical information systems (GIS)* to visualize geographically-linked data
- *Graph visualization or network maps* for representing relations between objects
- Projection of multidimensional data on low-dimensional screens with further visualization, *data cartography*

In addition, visualization is often the last part of a data analysis pipeline. To visualize a data set you must first answer some questions:

- What do I want to visualize?
- Where can I get the data?
- How should my data set look like?
- What is the data telling me?

In other words, to create visualizations you must also know what data is (or not) out there (literature review), develop an intuition about what patterns you might find in it (research question/hypothesis), clean and wrangle lots of often messy data sets (methodology), create the visualization (results) and share it with your target audience (discussion). Think of the process as writing a small research paper. A single `.png` often speaks more than a thousand lines of code!

About this Book

The examples used throughout this book are closely related to corruption since that is one of my main areas of interest. This also allows me to explore different types of data that social

scientists are likely to encounter such as surveys, text, networks, geo-locations, administrative records (*Big Data*), and regression models.

All used data is open source, though it often comes through APIs[2]. Sometimes, to access these data sets you need a *token* or *secret*. For obvious reasons, I will not share my own tokens here, but I will show you where the information you need should go in the code. When possible, I will also provide a clean version of the data used so you can experiment with it.

To keep things as simple as possible, the book follows this syntax:

- `packages` are placed inside a shaded box (e.g. `ggplot2`),
- common `functions()` are also inside a shaded box, and followed by parentheses (e.g. `mutate()` from `tidyverse` or `class()` from base R),
- less common `functions()` are the same, but the package is explicitly called `::` (e.g. `reshape2::melt()`),
- short R commands (e.g. `%in%`), are also shaded, non-R commands are in bold (e.g. **Ctrl + p**),
- the common *pipe* operator `%>%` will be used when possible in the code[3] (i.e., we will mostly use the `tidyverse` syntax over base R).

Recommended Readings

This book does not expect a lot of familiarity with R or programming, though some knowledge of statistics will be very helpful. The appendix covers the basics of working with R and RStudio during the first few sessions. Some tutorial videos on the basics of working with RStudio are available here. Similarly, you may also consult the following open-source books on R:[4]

- *R Cookbook* (Long and Teetor 2019)
- *R for Data Science* (Grolemund and Wickham 2016)
- *R Graphics Cookbook* (Chang 2018)
- *Efficient R Programming* (Gillespie and Lovelace 2016)
- *Hands-on Programming with R* (Grolemund 2014)
- *Fundamentals of Data Visualization* (Wilke 2019)
- *Text Mining with R* (Silge and Robinson 2017)
- *An Introduction to R* (Venables, Smith, and R Core Team 2021)
- *R Markdown: The Definitive Guide* (Xie, Allaire, and Grolemund 2018)
- *R Markdown Cookbook* (Xie, Dervieux, and Riederer 2020)

[2] API is short for *application programming interface* and it is a great way to get data directly from the source. Many data providers such as the World Bank offer APIs that connect directly to your R session.

[3] For Windows users, the `%>%` shortcut in RStudio is **Ctrl + Shift + M** and for Mac users it is **Cmd + Shift + M**.

[4] For a comprehensive list of R-related books, consult the *R-Project* Website

2 The Grammar of Graphics

2.1 The `tidyverse` packages

Throughout this course, we will be using tidy data principles[1] to create several types of visualizations. The main package we will use is the `tidyverse`, which includes several useful tools for data wrangling, analysis and visualization. The first step then is to install the package! You can do this from the packages vignette in *explorer pane* in RStudio, or by writing `install.packages("tidyverse")` into the *console pane*.

Once the package has been installed, the next step will be to load the library so that we can start using it! Simply write the command below in a script the *editor pane* and click *run*, or directly in the *console pane* and press *enter*.

After installing `tidyverse` packages, we will get access to two very important functions which we will be using extensively. The first is the the command `ggplot()` from the package `ggplot2` which will allow us to make plots based on the *grammar of graphics*. The second is the *pipe operator* or `%>%` from the `dplyr` package, which translates loosely to the phrase "and then", and which we will use to put several commands and functions together in a pipeline.[2]

At the top of every R script, you will see the libraries used. For almost every chapter of this book we will use the two libraries below. **IMPORTANT:** *every time you start a new R script or session, you must call on these libraries* so R knows which commands you want to use!

```
library(ggplot2)
library(dplyr)
```

2.2 The `ggplot2` Package

The `ggplot2` package is installed and loaded alongside the `tidyverse` package, though it can – and **should** – be called on separately. This is a very powerful tool to make print-quality graphs and all sorts of visual outputs. To do this, it draws on *the grammar of graphics*, which is a concept developed by Leland Wilkinson (Wilkinson 2005). The main idea behind this

[1] These principles are: a) each variable should have its own column, b) each observation should have its own row, and c) each value should have its own cell.

[2] For Windows users, you can use the RStudio short cut `ctrl` + `shift` + `m` to write this pipe `%>%` operator.

complex book is that plots can be divided into several elements, each with a specific role to play. `ggplot2` has 7 such elements:

- **Data**
- **Aesthetics**
- **Layers**
- Scales
- Coordinates
- Facets
- Themes

Throughout this chapter, we will focus on the first three (**Data**, **Aesthetics**, **Layers**) which are the minimum requirements to make a basic plot. The element **data** tells R which vector(s) from your environment are going to be used to draw the plot. The **aesthetics** element determines which variable(s) will be used and in what capacity. The **layers** element tells R which type of geometry you wish to draw and in which order.

```
df %>%
  ggplot(aes(x=var1,y=var2))+
  geom_point()
```

In the example above, we are telling R that there is an object `df` in our environment which has at least two vectors (columns), one called `var1` and another `var2`. We are also telling it that we want `var1` to be our **`x`** axis and `var2` to be our **`y`** axis, we define this inside the `aes()` command either globally for the plot (i.e. inside the `ggplot()` command) or specifically for a layer (i.e. inside `geom_point()`). Finally, we are telling R that we want to make a scatter plot by defining the layer `geom_point()`. Notice that after the `ggplot()` command and until the end of the graph, we use a + sign.

2.3 Example

To make our first `ggplot` plot, we will use the `mtcars` data set as an example.

```
data("mtcars")
```

The cars data set has 32 observations and 11 variables. This data set comes pre-loaded with R and it often used in examples. Let's see what is inside!

Once the data has been loaded, let's use the `pipe operator` to do some cleaning. In the code below, we are creating a new object called `df` - a common way of naming data frames - and filling it with the `mtcars` data with some modifications. We are asking R to a) take the `mtcars`

Table 2.1: The Motor Trend Car Road Tests Data Set

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

data, b) *and then* `%>%` *select* four variables, c) *and then* `%>%` give them new name. This pipeline is saved into the new object `df`.

```
df <- mtcars %>%
  select(cyl, mpg, hp, am) %>%
  rename(cylinders=cyl,
         mileage=mpg,
         horsepower=hp,
         transmission=am)
```

With this `df` stored in our environment, we can start making plots. Let's begin with a histogram that shows the distribution of mileage across the 32 variables in our data set. For this we will use `geom_histogram`.

```
df %>% #Our Data
  ggplot(aes(x= mileage))+ #Our Aesthetics
  geom_histogram() #Our Layer
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

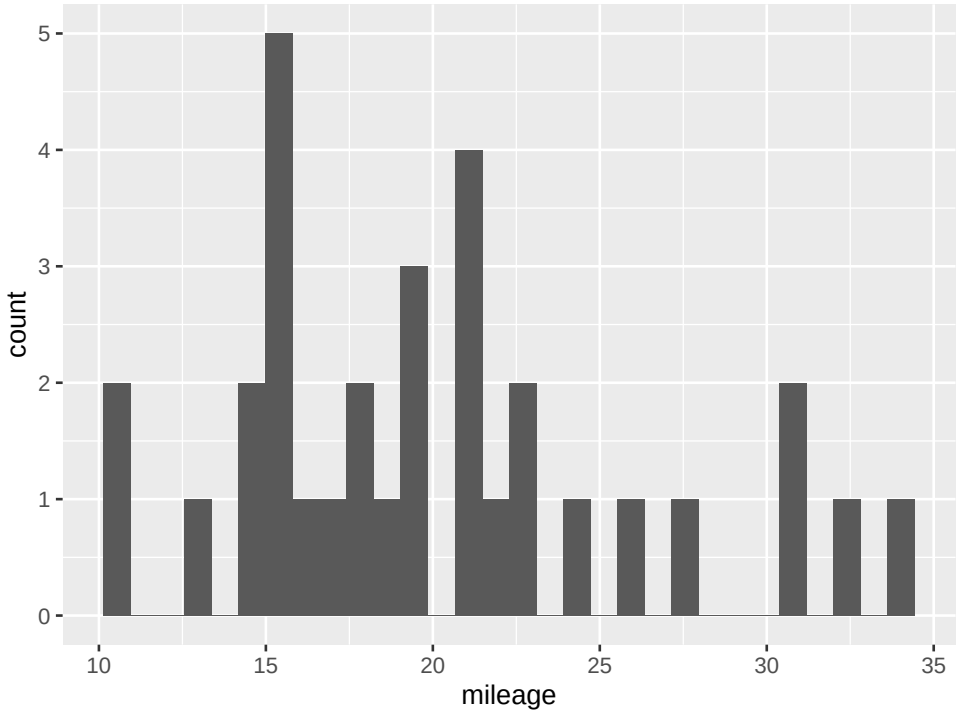

Figure 2.1: A histogram with default settings

Figure @ref(fig:hist-fig) shows us our very first **ggplot**, which shows the number of observations at each of the binned levels. From this plot we know that most cars in our data set do around 15 miles per gallon. However, it is not very nice looking! We can improve this by adding more parameters.

You might notice that below the code R is giving us a **warning**: `stat_bin() using bins = 30. Pick better value with binwidth.` Here the software is hinting that we might want to change the number of bars (`bins`) or their width (`binwidth`) in our plot to make it more informative.[3] In figure @ref(fig:hist2-fig) we change the number of bins to 5 inside our `geom_histogram` layer, and also declare the color of the column fill (darkgray) and the outline (black).

```
df %>%
  ggplot(aes(x= mileage))+
  geom_histogram(bins = 5, fill="darkgray", color="black")
```

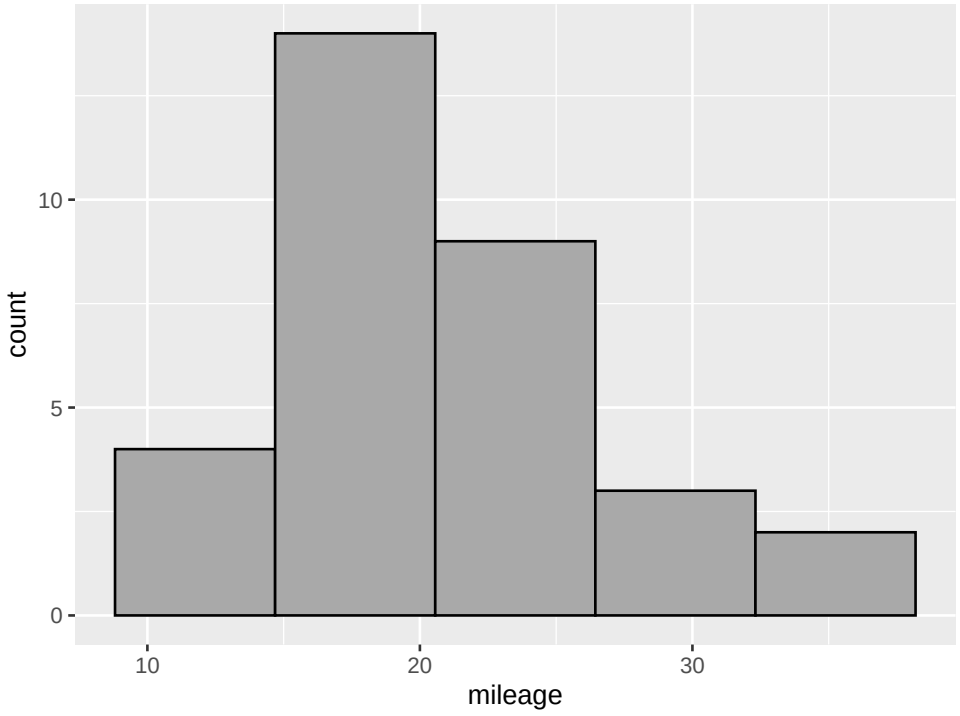

Figure 2.2: A nicer looking histogram

[3]Most other software will give you a default based on some parameter such as the Freedman-Diaconis rule, `ggplot` does not do this, forcing you to experiment with different parameters that best reflect your data.

References

Chang, Winston. 2018. *R Graphics Cookbook: Practical Recipes for Visualizing Data*. Second. Sebastopol, California: O'Reilly Media. https://r-graphics.org/.

Gillespie, Colin, and Robin Lovelace. 2016. *Efficient r Programming: A Practical Guide to Smarter Programming*. Sebastopol, California: O'Reilly Media. https://csgillespie.github.io/efficientR/.

Grolemund, Garrett. 2014. *Hands-on Programming with r: Write Your Own Functions and Simulations*. Sebastopol, California: O'Reilly Media. https://rstudio-education.github.io/hopr/.

Grolemund, Garrett, and Hadley Wickham. 2016. *R for Data Science*. Sebastopol, California: O'Reilly Media. https://r4ds.had.co.nz/.

Long, JD, and Paul Teetor. 2019. *R Cookbook: Proven Recipes for Data Analysis, Statistics, and Graphics*. Second. Sebastopol, California: O'Reilly Media. https://rc2e.com/.

Mcmanus, Ian, and Paul Gesiak. 2014. "Experimenting with Mondrian: Comparing the Method of Production with the Method of Choice." In. https://doi.org/10.13140/2.1.1561.2967.

Silge, Julia, and David Robinson. 2017. *Text Mining with r: A Tidy Approach*. Sebastopol, California: O'Reilly Media. https://www.tidytextmining.com/.

Venables, W. N., D. M. Smith, and the R Core Team. 2021. *An Introduction to r*. https://cran.r-project.org/doc/manuals/R-intro.pdf.

Wilke, Claus O. 2019. *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. Sebastopol, California: O'Reilly Media. https://clauswilke.com/dataviz/.

Wilkinson, Leland. 2005. *The Grammar of Graphics*. 2nd ed. New York: Springer-Verlag. https://www.springer.com/gp/book/9780387245447.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.

Zinovyev, iAndrei. 2010. "Data Visualization in Political and Social Sciences." In. https://arxiv.org/abs/1008.1188.