Best Practices for Data Visualisation

Insights, advice, and examples (with code) to make data outputs more readable, accessible, and impactful

Andreas Krause, Nicola Rennie, & Brian Tarran

Table of contents

1 Introduction

Statistics is "the science of collecting, analyzing, presenting, and interpreting data" (Williams, Anderson, and Sweeney 2023). Presentation of data is a key means to support and guide interpretation and subsequent decision making. Techniques exist for effective display. This is what this guide is all about.

Good data visualisation requires appreciation and careful consideration of the technical aspects of data presentation. But it also involves a creative element. Authorial choices are made about the "story" we want to tell, and design decisions are driven by the need to convey that story most effectively to our audience. Software systems use default settings for most graphical elements. However, each visualisation has its own story to tell, and so we must actively consider and choose settings for the visualisation under construction.

This guide covers both aspects of data visualisation: the art and the science. It is written primarily for contributors to Royal Statistical Society publications – chiefly, *Significance* magazine, the *Journal of the Royal Statistical Society Series A*, and Real World Data Science – but we trust you will find the information and advice within to be of broad relevance and use to any data visualisation task.

The overarching aim of this guide is to equip the reader with the fundamentals for creating data visualisations that are high quality, readable, effective at conveying information, accurate in display and interpretation, and fulfil their intended purpose. It begins with an overview of why we visualise data, and then discusses the core principles and elements of data visualisations – including the structure of charts and tables, and how those structures can be refined to aid readability. Concrete advice, examples, and code are presented to help improve the styling of charts, with a particular focus on accessibility. There's a dedicated section on styling charts for RSS publications, and readers will also find links to resources for choosing the right type of chart for the data at hand.

In constructing this guide, the authors draw on many exceptional textbooks, papers, and other material created by experts in the field. References can be found throughout (they are also collected on a dedicated page, and readers are encouraged to seek out the original sources to deepen their understanding of, and appreciation for, the art and science of data visualisation.

2 How to use this guide

We designed this guide to be read from beginning to end, so that each section builds on the foundations of those that came before. If this is your first time reading the guide, we would encourage you to read it in this way. However, those looking for guidance on specific aspects of data visualisation can make use of the search and navigation functions of this website.

If you would prefer to read offline, you can download the guide as a PDF. Individual sections of the guide can also be downloaded as separate PDFs using the "Other Formats > PDF" option when shown in the right-hand navigation menu.

Throughout the text you will find examples of charts and graphs. For many of these, we provide code in R to allow you to reproduce the visualisations, and we encourage you to do so and to adapt the code for your own purposes.

2.1 How to contribute to this guide

To make a suggestion or ask a question

Please open up a discussion in our GitHub repository's Discussions section.

If you spot a bug or an error

Please either:

1. Raise an issue in our GitHub repository's Issues section. You can do this using one of the 'Report an issue' links on our website; or
2. Fork our repository, edit the relevant file(s), and make a pull request against the `main` branch of our repository. You can do this using one of the 'Edit this page' links on our website.

If you want to add a new feature or section to the guide

1. Raise an issue in our GitHub repository's Issues section, tag it as an 'enhancement' and describe your proposed contribution.
2. Fork the repository and create a new branch named, e.g., `my-new-feature-or-section`.
3. Add content, code and files to your branch.
4. Make a pull request against the `main` branch of our repository.

Further information about the contribution process can be found in our repository's README file.

2.2 How to engage with other users of this guide

Readers are encouraged to make full use of our GitHub repository's Discussions section. Please note that use of, and all contributions to, our website and repository are governed by our Code of Conduct.

2.3 Copyright, re-use, distribution, and disclaimer

> **How to cite:** Krause, Andreas, Nicola Rennie and Brian Tarran. 2023. "Best Practices for Data Visualisation." Royal Statistical Society, July 2023. https://royal-statistical-society.github.io/datavisguide/.

3 Why we visualise data

3.1 Motivation

Data visualisations can be a very efficient means of identifying patterns in data and conveying a message. The scientific aim of any visualisation is to allow the reader to understand data and extract information:

- intuitively;
- efficiently; and
- accurately.

It is important, when creating a visualisation, to consider the background of the reader or intended audience (Krause 2013). Interpretation is in the eye of the beholder, and a visualisation will only succeed at conveying its message if designed with its audience in mind.

A successful data visualisation will:

- **Grab attention** In a sea of text, a visualisation will stand out. If a reader is short on time or uncertain about whether a document is of interest, an attention-grabbing visualisation may entice them to start reading.
- **Improve access to information** Textual descriptions can be lengthy and hard to read, while skilfully created visualisations permit the extraction of key information more efficiently, making information extraction a fun task.
- **Increase precision** Textual descriptions are frequently less precise than a visual depiction showing data points and corresponding axes, while a text with too many precise numbers can make it hard to follow a line of argument.

- **Bolster credibility** While a textual summary provides a story, a visualisation of the data can add credibility to otherwise unsubstantiated claims: readers can see the numbers for themselves and arrive at the (same?) conclusions.
- **Summarise content** Visual displays allow for summarising complex textual content, aiding the reader in memorising key points.

For these reasons, data visualisations are key elements in almost any kind of publication – scientific papers, media reports, conference presentations, social media posts, video summaries, etc.

Tables, too, are a way to visualise data or statistics, and can be similarly important components of a publication. A table may in some cases visualise data better than a graphic. For example, five numbers are probably better displayed in a table than in a complex pie chart that uses colours, angles, and possibly shading and more than two dimensions.

3.2 A brief history of data visualisation

Data visualisation has been, for a long time, both a topic of scientific research and an evolving art form with a variety of high-impact applications.

In 1859, Florence Nightingale, the founder of modern nursing, published her findings on the sanitary status of the British army during the war with Russia. She showed raw data as well as summary statistics in tables and charts (Nightingale, 1859). One chart in particular continues to be celebrated today: a polar area chart on the "causes of mortality in the army in the East".

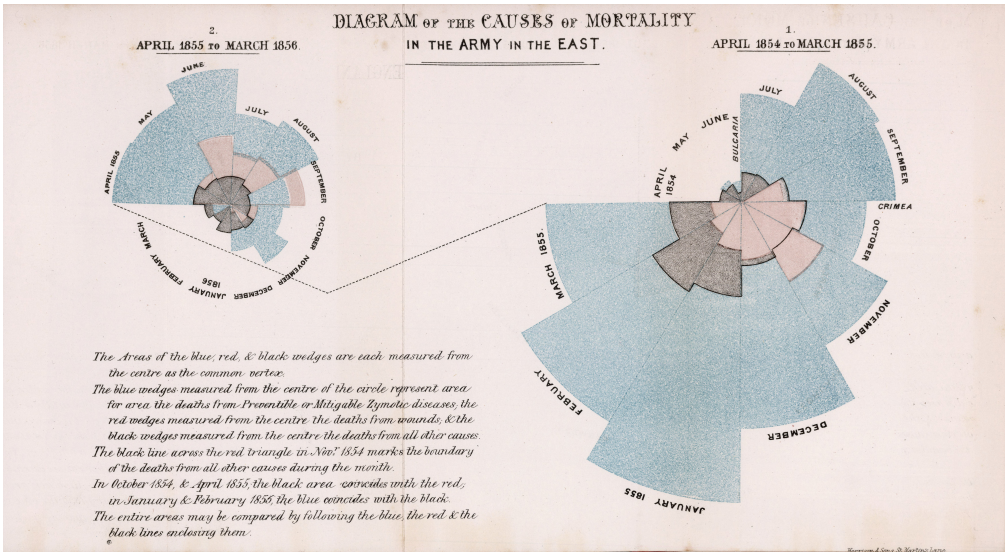

Figure 1: Florence Nightingale's polar area chart, "Diagram of the causes of mortality in the army in the East". Source: Wikimedia Commons.

What made Nightingale's graphs "particularly iconic was their powerful use of visual rhetoric to make an argument about data" (Hedley 2020). This quality is also evident in other visualisations produced by Nightingale's contemporaries.

A simplistic but rather impactful visualisation of the water pumps in London associated with transmission of cholera paved the way for root cause identification. Medical doctor John Snow collected data on cholera deaths and created a visualisation where the number of deaths was represented by the height of a bar at the corresponding address in London. This visualisation

showed that the deaths clustered around Broad Street, which helped identify the cause of the cholera transmission, the Broad Street water pump (Snow 1854; Wikipedia contributors 2023).

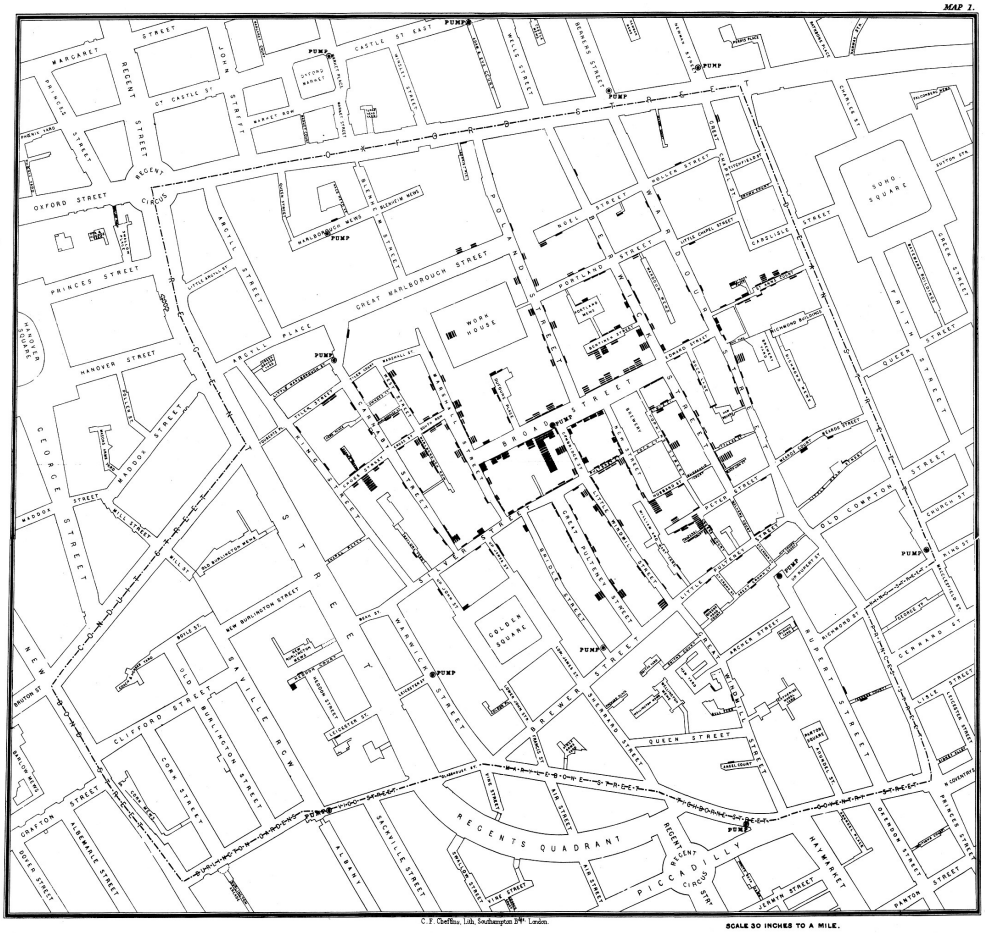

Figure 2: Map by John Snow showing clusters of cholera cases in the London epidemic of 1854. Source: Wikimedia Commons.

An early complex visualisation was created by Minard in 1861, depicting data from Napoleon's march on Moscow in 1812/13 and his subsequent retreat.

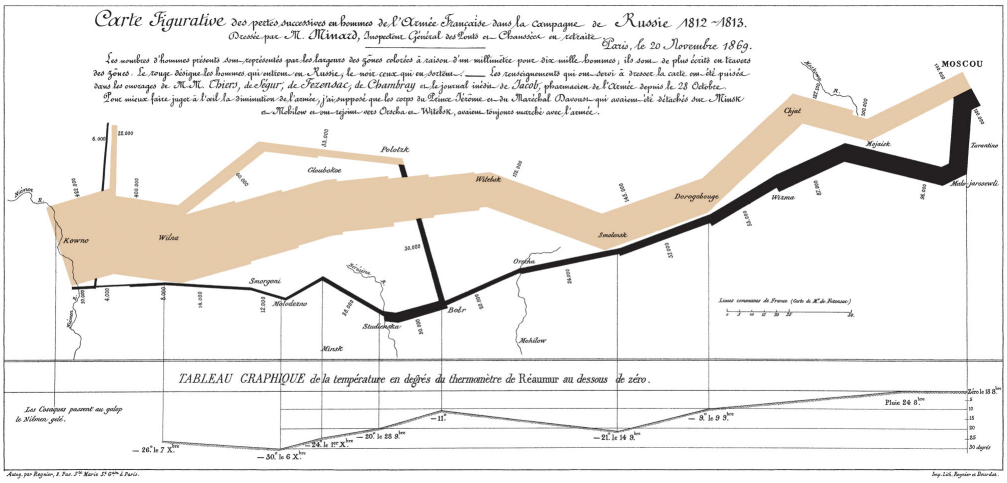

Figure 3: Charles Minard's 1869 map of "the successive losses in men of the French Army in the Russian campaign 1812–1813". Source: Wikimedia Commons.

The map shows latitude and longitude of the army as it moved. The line shows the direction of movement, and the line width represents the size of the army (the surviving soldiers). Particular locations were marked by the date of the army presence, and the temperature is shown, too. Six variables were elegantly woven into a single display (Tufte 2001; Corbett 2001; Robinson 1967).

The Paris Exposition in 1900 featured W. E. B. Du Bois exhibiting graphs, charts, and maps of how Black Americans were living (Du Bois 1900; Battle-Baptiste and Rusert 2018).

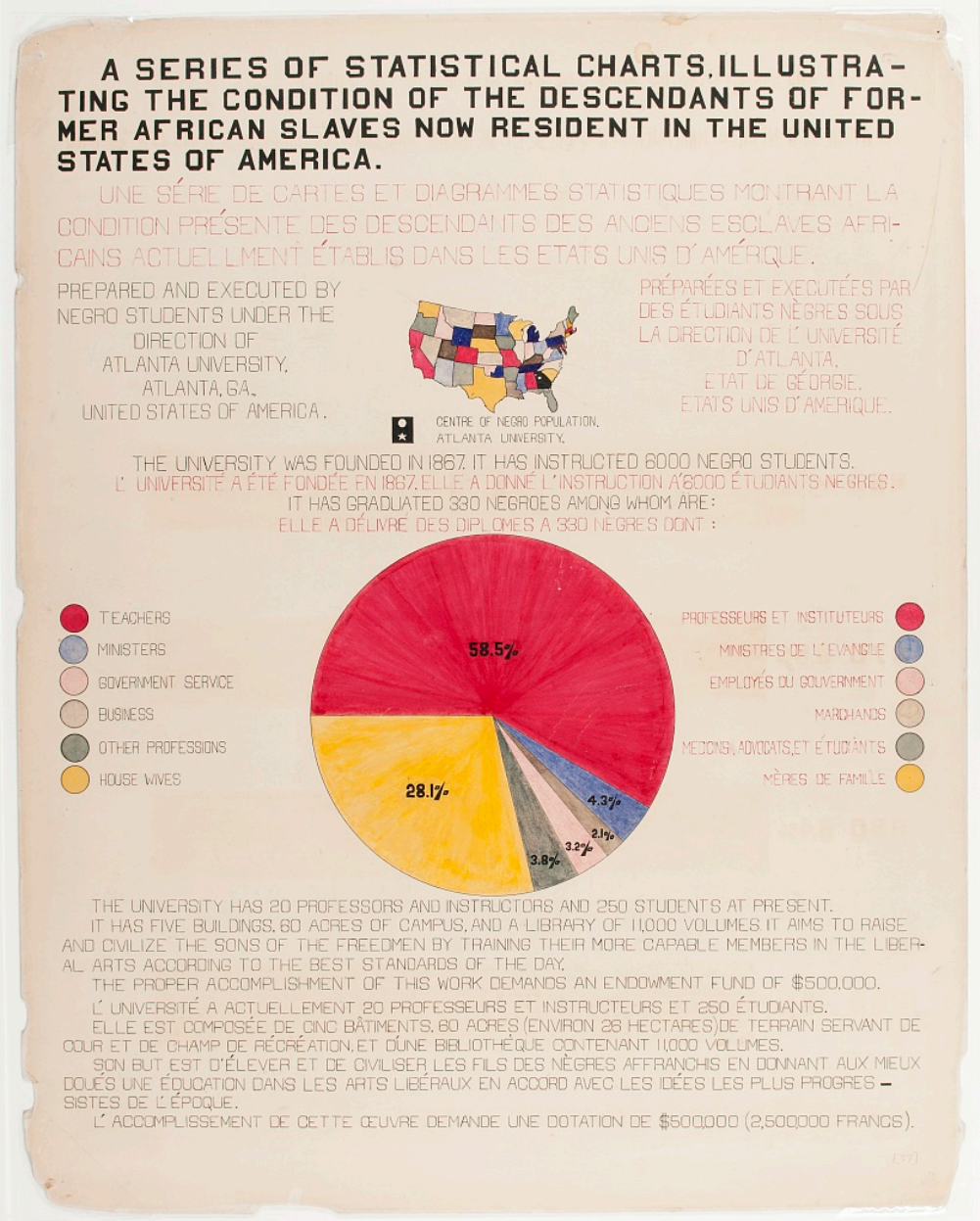

Figure 4: A series of statistical charts illustrating the condition of the descendants of former African slaves now in residence in the United States of America. Drawing, ca. 1900. //hdl.loc.gov/loc.pnp/ppmsca.33899. Source: Library of Congress.

Groundbreaking work in modern visualisation was provided by Tukey with his book *Exploratory Data Analysis* (Tukey 1977), and Edward Tufte (Tufte 1990, 2004, 2006).

The historical development of visualisations is provided in various publications by Michael Friendly (Friendly and Denis 2005; Friendly 2018, 2022).

Data were visualised by hand until computers came along. The first monitors and printers worked in text mode only, with a resolution of 25 rows and 80 columns or similar, which did not permit much detail or precision. Graphics terminals and dot matrix printers followed, and resolutions kept increasing with the development of laser printers.

Statistical systems such as SAS enabled creation of data visualisations early on ("Documentation" n.d.).

Arguably the most consistent implementation of a graphics system was realised with ggplot (Wickham 2011, 2016a) based on Wilkinson's *The Grammar of Graphics* (Wilkinson 2005).

A key component of ggplot2 is the feature of creating conditional displays. These allow for subsetting the data by the values of one or more variables. The concept was introduced by Cleveland (Cleveland 1993, 1994), showcasing a widely known barley data set that was analysed in textbooks for decades until a visualisation strongly suggested an error in the data set – two years of crop yield of one variety at one of six farms were accidentally swapped. It took a visualisation to reveal what numerous numerical analyses had missed!

Cleveland named the display type "trellis", being inspired by a trellis in his garden (Cleveland 1993). The trellis concept was first implemented in S (Becker and Chambers 1984) and S-PLUS (Becker and Cleveland 1996). When R (R Core Team 2021) was developed by Robert Gentleman and Ross Ihaka in Auckland NZ in the 1980s, a package named `lattice` was developed (Sarkar 2008) since "trellis" carried a trademark (by Lucent and later MathSoft). With the advent of ggplot, the term faceting (with functions facet_grid and facet_wrap) replaced the "lattice" with its "panels".

4 Principles and elements of visualisations

Data visualisations must serve a purpose. By understanding the purpose of a visualisation, we – as author or reader – are in a position to assess whether a visualisation succeeds in its aims or requires improvement!

Arguably the most common purpose of a visualisation is a comparison of groups of data, such as data on patients receiving different treatments. A good choice of axes, axis limits, labels and symbols can facilitate substantially the identification of patterns in the data, whereas poor choices for any of these elements can substantially hamper the extraction of information.

4.1 Elements of charts

Various elements of a visualisation can contribute to the efficacy with which information can be distilled. All visualisation software packages will output graphics in a default style, but these will rarely, if ever, be the optimum choices for the data visualisation you are creating. When designing a data visualisation, one needs to consider the range of options available. Some of these options are discussed in what follows.

Layout (panels, facets)

The layout (arrangement of multiple panels, facets, or subplots) is highly relevant for efficient comparison. If data on the y-axis are to be compared, a single y-axis with all panels aligned horizontally facilitates comparison, whereas for efficient comparison of x-axis values, panels should be stacked. Matrix layouts (multiple rows and columns in a single figure) should only be used if the data shown in individual panels are not related or space does not permit a single row or a single column (e.g., if there are too many panels to fit on a single row).

Example: The figure below shows two different layouts of exactly the same data with exactly the same type of visualisation except for the layout. Note how difficult it is to compare the data across panels on the left-hand side, while it is easy with the layout on the right-hand side. The key difference is that the panels on the right share a common y-axis, which is key for comparison of y-values across panels.

```r
# ---
# Data set creation.

set.seed(93384)

time <- c(0, 0.5, 1, 2, 4, 8, 12, 16, 24)
n <- 32 # no of subjects

data <- expand.grid(ID = 1:n, time = time)

bw <- data.frame(
  ID = sort(unique(data$ID)),
  bw = rlnorm(n, log(75), sdlog = 0.25)
)

bw$bw.category <- cut(bw$bw,
  breaks = quantile(bw$bw, c(0, 0.33, 0.66, 1)),
  labels = paste(c("low", "medium", "high"), "body weight"),
  include.lowest = TRUE
)

data <- merge(data, bw)

data <- data[order(data$ID, data$time), ]

# Simulate drug concentrations as a function of body weight.
data$conc <- 100 / (data$bw^1.0) * exp(-0.085 * data$time) *
  rlnorm(nrow(data), sdlog = 0.25) + # res. error
  (data$ID - mean(data$ID)) / mean(data$ID) / 4 # r. eff

# ---
# Visualisation.
library(ggplot2)

gg <- list()

data$ID <- factor(data$ID)

gg[["3x1"]] <- ggplot(data, aes(x = time, y = conc, group = ID, color = ID)) +
  geom_line()
gg[["3x1"]] <- gg[["3x1"]] + scale_x_continuous(breaks = seq(0, 24, by = 4))
gg[["3x1"]] <- gg[["3x1"]] + theme_bw() +
  xlab("time [h]") +
  ylab("drug concentration [ng/mL]")
gg[["3x1"]] <- gg[["3x1"]] + facet_grid(bw.category ~ .)
gg[["3x1"]] <- gg[["3x1"]] + theme(legend.position = "none")

gg[["1x3"]] <- gg[["3x1"]] + facet_grid(. ~ bw.category)

# Add space to the rhs of the first figure for better separation in the cowplot.
gg[["3x1"]] <- gg[["3x1"]] +
```

```
  theme(plot.margin = unit(c(0.5, 4, 0.5, 0.5), "lines"))

# Both figures into a single output figure.
library(cowplot)
plot_grid(gg[[1]], gg[[2]], rel_widths = c(1.5, 2))
```

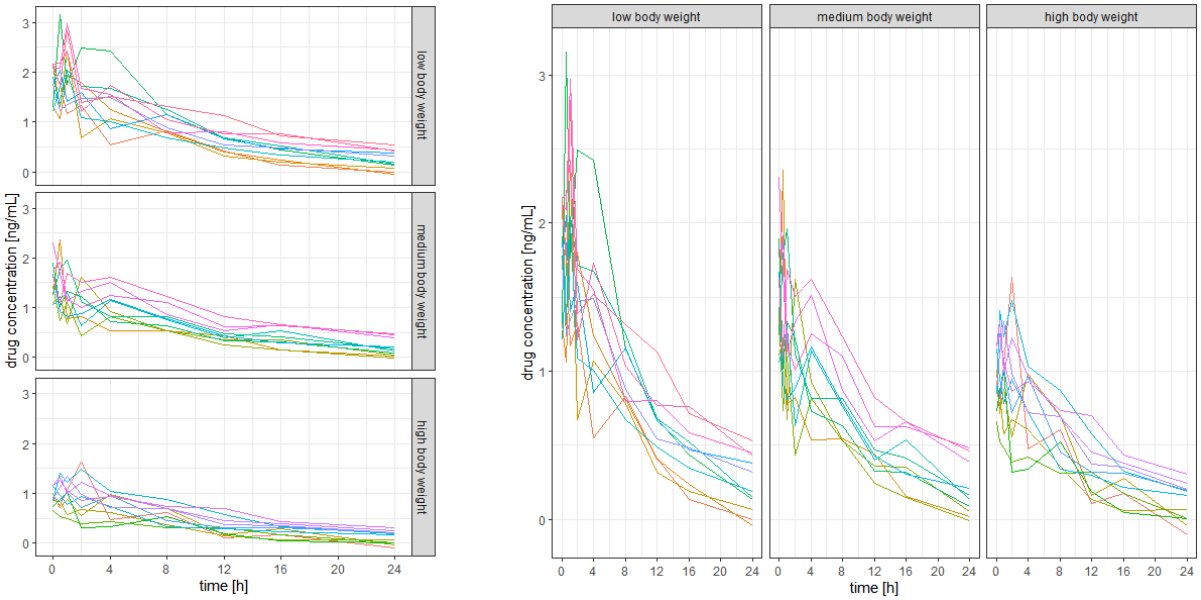

Figure 5: Line plots showing how choice of layout determines ease of comparison across panels.

Aspect ratio

Our visual perception of data in a display must not depend on the choice of x- and y-axes. In many cases, a square figure avoids visual bias. A square figure should be considered in particular if the two axes share a communality such as a measurement before and after some event, observed data vs model-predicted values or, more generally, the same units (such as kg or metres). Generally, a 1:1 aspect ratio (the physical length of 1 measurement unit in the x- and the y-direction) is a good default. If the figure has identical ranges for the x- and the y-axis, a 1:1 aspect ratio yields a square figure.

Example: The three panels below all show the same data, aimed at enabling judgement about the goodness of a model fit (predicted vs observed values). Note the difference in visual perception between a stretched x-axis, a stretched y-axis, and a 1:1 aspect ratio with identical axis limits.

```
# Observed vs predicted (any data with comparable x and y will do).

# ---
# Data set.
# Old Faithful Geyser (Yellowstone) data set with eruption duration
#   and waiting time to the next eruption (both in minutes).
data <- data.frame(
  x = faithful$eruptions,
  y = faithful$waiting
```

```
)

# ---
# Regression model fit.
fit <- lm(y ~ x, data = data)

# Addition of predicted values to the data set.
data$pred <- predict(fit)

# Range of y and y predicted combined.
r <- range(unlist(data[c("y", "pred")]))

# ---
# Plotting.

library(ggplot2)

gg <- ggplot(data, aes(x = pred, y = y))

# Adding the line of identity, y = x
# (note: plotting it first will add points on top).
gg <- gg + geom_abline(intercept = 0, slope = 1, color = "black", linewidth = 1)

# Adding points, removing grey background.
gg <- gg + geom_point() + theme_bw()

# Adding regression fit (local smoother, loess) of y~x.
gg <- gg + geom_smooth(method = "loess", color = "firebrick", se = FALSE)

# Adding axis labels.
gg <- gg + xlab("predicted") + ylab("observed")

# Aspect ratios are commonly not fixed but adapted to figure size.
#   With dynamic displays, the point of different perception
#   might not be obvious depending on the figure/screen size.
#   To make that point independent of figure height and width,
#     the aspect ratio is fixed in this example.
gg <- gg + coord_fixed(ratio=0.5)

# Copy the figure and fix the aspect ratio to 2, i.e.,
#   one pixel in x corresponds to 2 pixels in y.
gg2 <- gg + coord_fixed(ratio=2)

# Setting the aspect ratio to 1 (1 unit in x and y
#   corresponds to the same number of pixels) and
#   setting axis limits to be identical.
gg3 <- gg + coord_fixed(ratio=1, xlim=r, ylim=r)

# Cow (column-wise) plot, combine all figures into one.
library(cowplot)
plot_grid(gg, gg2, gg3, rel_widths = c(4, 2, 3), nrow = 1)
```

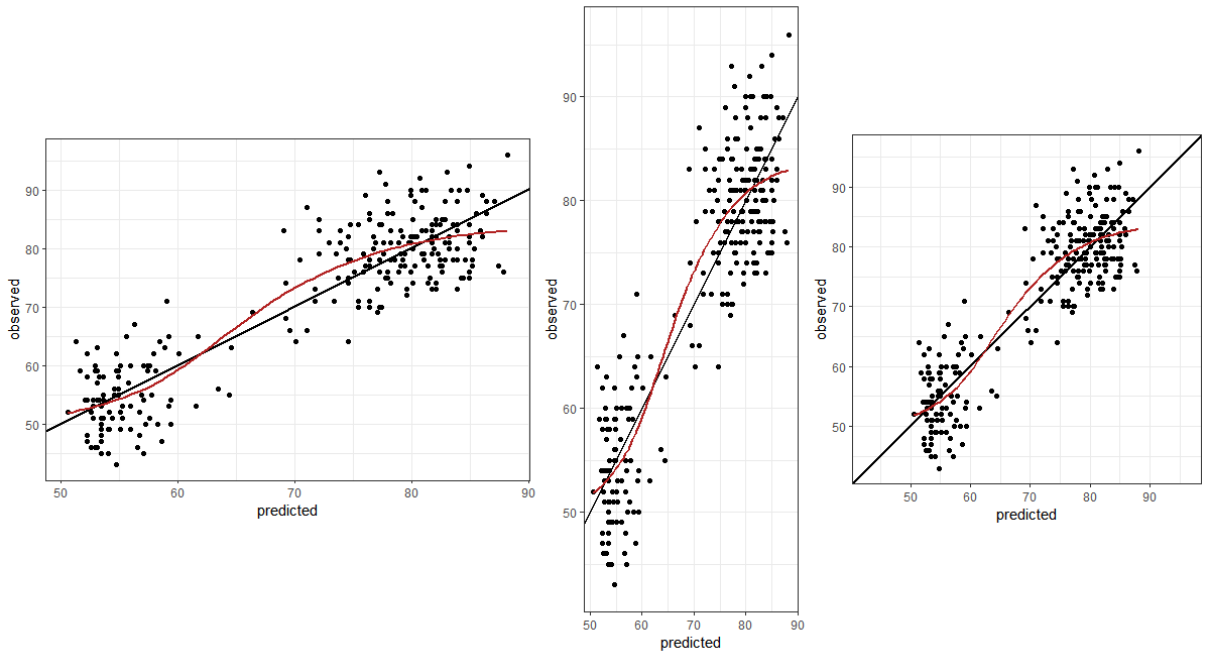

Figure 6: Scatter plots showing how visual perception of model fit can be influenced by aspect ratio.

Lines

Lines introduce an order, a sequence. If there is no order, data should not be interconnected by lines. If different line types indicate different groups that have an inherent order, it is helpful if the chosen line styles have an order, too (example: line thickness, dash density, or darkness of colour increases with values from the lowest to the highest group).

Points

Point symbols should be chosen such that the visualisation displays the data in a meaningful way. If thousands of data points are to be shown, open circles or smaller filled symbols are better than overlapping filled symbols. If the data are discrete and observations overlay, consider jittering the data gently for a better visualisation.

Colours

Colours should serve a purpose such as helping to identify different groups. Colours for pure decoration are discouraged. Tufte pointed out that because "they do have a natural visual hierarchy, varying shades of gray show varying quantities better than color", and "the shades of gray provide an easily comprehended order to the data measures. This is the key" (Tufte 2001, 154).

Axes

Origins and limits

Axis ranges should generally start at 0 unless there is a good reason for choosing other ranges. If the data do not contain negative values, the axis should not stretch into negative values and in particular not have tick marks at negative values.

If relative changes or ratios are displayed, the corresponding axis should be logarithmic (such that, e.g., 1/4 has the same distance to the reference point, 1, as 4) and symmetric around the point of no change. An auxiliary line at the point of no change can facilitate interpretation. Axis tickmarks and auxiliary grey lines will facilitate reading off values, avoiding wrong linear interpolation by the viewer. Axis tick mark labels should indicate the ratio (e..g, "1/4" instead of 0.25).

If the data displayed as x and as y are comparable, axis limits should be identical and the figure square, such that distances are consistent in the x- and in the y-direction.

Example: The figure below illustrates that the deliberate choice of axis limits (here, y-axis limits) can make a big difference to perception, and therefore interpretation, by the reader. The only difference between the two panels is the y-axis range.

```
library(ggplot2)
plot_data <- data.frame(
  type = factor(
    c("Our product", "Competitor"),
    levels = c("Our product", "Competitor")
  ),
  value = c(220, 210)
)

# Original plot
ggplot(plot_data) +
  geom_col(
    mapping = aes(x = type, y = value),
    fill = "lightblue",
    colour = "black"
  ) +
  scale_y_continuous(breaks = seq(0, 220, by = 20), expand = c(0, 0)) +
  labs(x = "", y = "") +
  theme_minimal()

# Offset the y axis
offset <- 208
ggplot(plot_data) +
  geom_col(
    mapping = aes(x = type, y = value - offset),
    fill = "lightblue",
    colour = "black"
  ) +
  scale_y_continuous(
    breaks = seq(0, 14, by = 2),
    labels = seq(0 + offset, 14 + offset, by = 2),
```

```
    expand = c(0, 0)
  ) +
  labs(x = "", y = "") +
  theme_minimal()
```

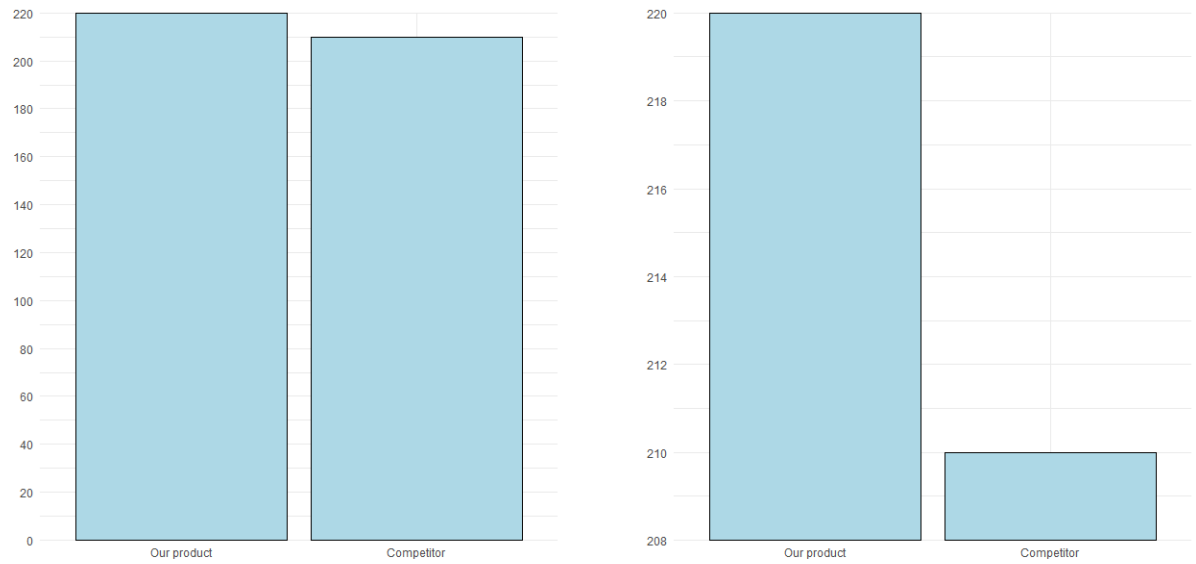

Figure 7: Bar charts showing how visual impression can be driven by the choice of axis limits.

Linear and logarithmic axes

Example: Let's say that you want to know whether some variable, y, changes (compared to a reference or baseline measurement) depending on another variable, x. For example, does blood pressure change with treatment?

The figure below shows data with, on average, no change compared to a reference (baseline) measurement (red horizontal line, top two panels) – approximately half the data are below the point of no change, and the other half are above. However, this is not easily apparent from the top-left panel, which uses a linear axis, and so space is allocated asymmetrically in the vertical direction – both up and down – away from the reference line. In the top-right panel, which uses a logarithmic axis, the relatively even spread of the data points around the reference line is clear to see. In the bottom row, the same data are shown in histogram form, again using linear and logarithmic axes.

```
# Naïve plot of y vs x. If there is no change (on average),
# half the data are below the line of no change.
# Asymmetric view, and it depends on y/x or x/y.

set.seed(33838)
x <- data.frame(
  x = rlnorm(200, 2, 0.2),
  y = rlnorm(200, 0.2, 0.75)
)
# Add an outlier manually.
x <- rbind(x, data.frame(x = quantile(x$x, 0.8), y = max(x$y) * 1.5))
```

```r
# ---
# Plotting.

library(ggplot2)
# Scatterplot of y vs x.
gg <- ggplot(x, aes(x = x, y = y)) +
  geom_point() +
  theme_bw()
gg <- gg + geom_hline(yintercept = 1, color = "firebrick", linewidth = 2)
gg <- gg + xlab("x-variable") + ylab("Fold-change")
gg

# Logarithmic axes, symmetric range (!):
xbr <- c(1 / 10, 1 / 5, 1 / 2, 1, 2, 5, 10)
gg <- gg + scale_y_continuous(
  breaks = xbr, trans = "log10",
  limits = max(abs(x$y))^c(-1, 1)
)
gg

# Second axis:
gg <- gg + scale_y_continuous(
  breaks = xbr,
  labels = paste(100 * xbr, "%", sep = ""),
  trans = "log10",
  limits = max(abs(x$y))^c(-1, 1),
  sec.axis = sec_axis(
    trans = ~ . * 1, breaks = xbr,
    labels = ifelse(xbr < 1, paste("1/", 1 / xbr, sep = ""), xbr)
  )
)

# ---
# Univariate distribution (histogram).

gg <- ggplot(x, aes(x = y)) +
  theme_bw() +
  xlab("Fold-change")
gg <- gg + geom_histogram(color = "firebrick", fill = "gray")
gg

# Symmetric range, log scale.
gg <- gg + scale_x_continuous(
  breaks = xbr,
  labels = ifelse(xbr < 1, paste("1/", 1 / xbr), xbr),
  trans = "log10",
  limits = max(abs(x$x))^c(-1, 1)
)
gg
```

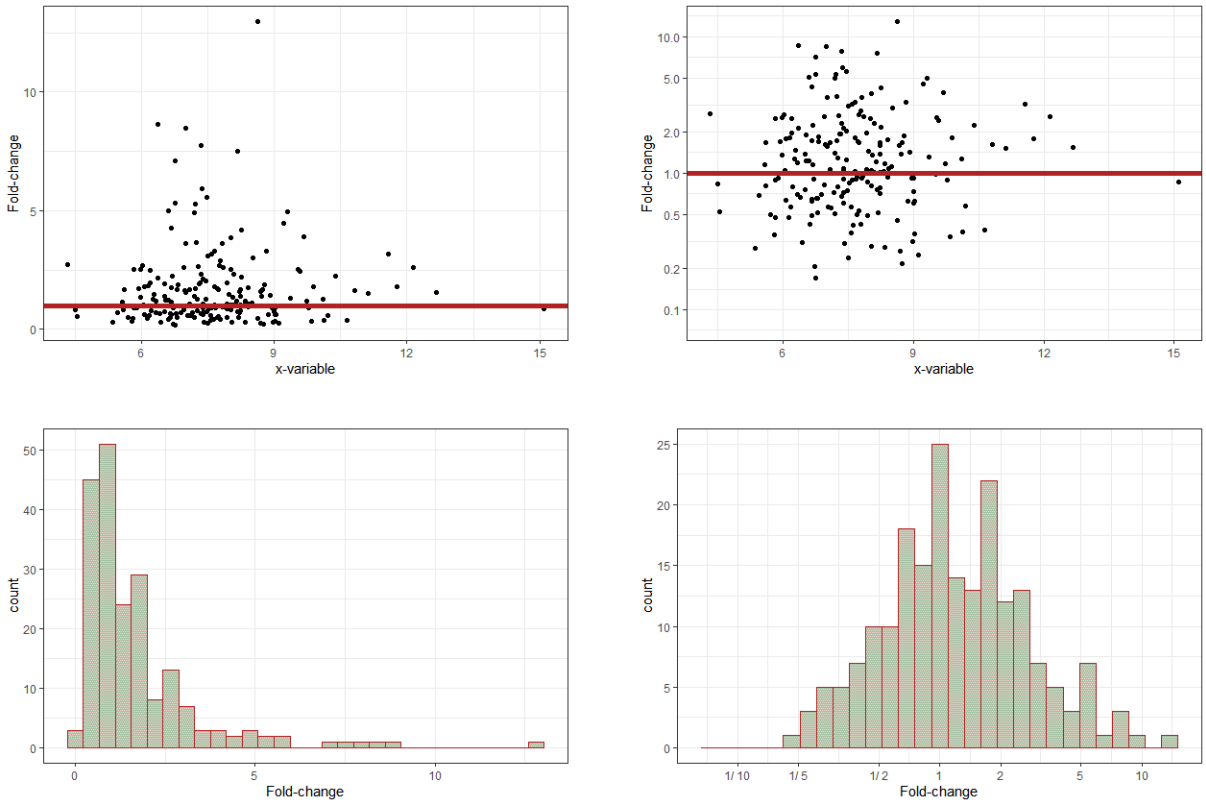

Figure 8: Switching from linear axes (left) to logarithmic axes (right) helps clarify changes in values relative to some baseline measure.

Symbols

Symbols should be intuitive (for example, "+" for a positive outcome, "–" for a negative outcome, "O" for a neutral outcome). In an ideal case, symbols require only a single look at the legend to subsequently focus on the visualisation.

Symbols such as triangles, circles, and squares do not have an intuitive connotation. However, if there is an order in the data, it might be reflected in the order of the symbols, e.g., the number of vertices (circle, dash, triangle, square, pentagon, etc.).

Legends

Legends should not attract too much attention and certainly not cover up data. They should be placed in the margins of the figure or can be captured in a small text below the figure.

If legend entries refer to single objects (e.g., one line per group), placing the legend next to the corresponding object makes it easier for the reader to map legend entries to display objects.

Example: The figure below shows that ease of reading can be improved by adding the legend directly into the figure, next to the corresponding data. Note that labels must not overlay and label positions might require adaptation depending on the data.

```
# ---
# EU stock markets, year and indices DAX, SMI, CAC, and FTSE.
```

```r
# Store graphics into a list.
gg.list <- list()

# Prepare the data set (reformat EuStockMarkets that comes with R).
x <- EuStockMarkets
df <- data.frame(
  time = as.vector(time(x)),
  index = rep(colnames(x), rep(nrow(x), ncol(x))),
  value = as.vector(x),
  stringsAsFactors = TRUE
)
df$index2 <- df$index # For use with labels later.

library(ggplot2)

# Standard layout and legend.
gg <- ggplot(df, aes(x = time, y = value, group = index, color = index, label = index2))
gg <- gg + geom_line() + theme_bw()

# Nicer axis tick mark settings.
ax <- pretty(df$time, n = 10)
gg <- gg + scale_x_continuous(limits = range(ax), breaks = ax)
gg <- gg + xlab("year") + ylab("Stock index")

gg.list[[1]] <- gg

# Use the last element of each time series for x,y of the label.
# Use that the last element is the first element of the reversed order,
#   and extract the first element per index by using !duplicated.
y <- df[rev(order(df$time)), ] # descending in time.
y <- y[!duplicated(y$index), ] # first entry per index
y$index2 <- y$index # Create a copy that contains formatted strings.
levels(y$index2)[levels(y$index2) == "FTSE"] <- "\n\nFTSE"
# Add a newline to separate FTSE from DAX.
# Note that the factor level is modified, not the data.

# Drop the legend, move labels into figure.
gg <- gg + geom_text(data = y, hjust = "left", nudge_x = 0.1)
# aes as before, nudge adds space on the lhs.
gg <- gg + theme(legend.position = "none")
gg.list[[2]] <- gg

# ---
# Both figures into a single output figure.

library(cowplot)
plot_grid(gg.list[[1]], gg.list[[2]], rel_widths = c(2.25, 2))
```

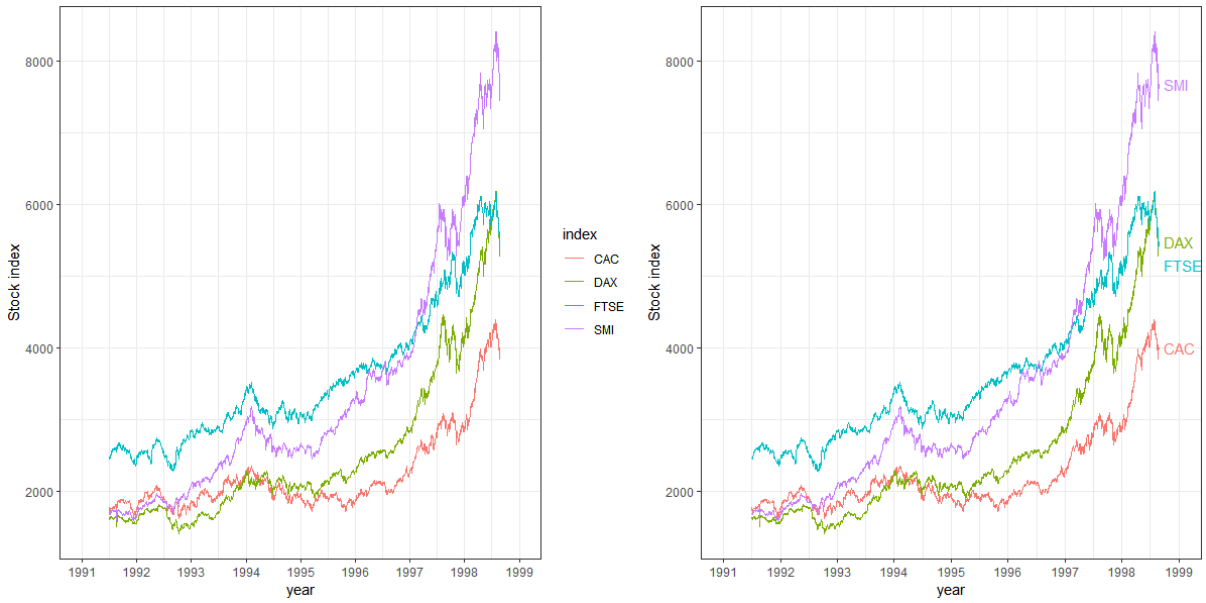

Figure 9: Line plots showing how direct labelling of figure elements can improve ease of reading.

Orientation

If there is an order in the data that is to be visualised, e.g., as a barplot, showing the data as horizontal bars and sorting values from top to bottom (highest value to lowest) is more intuitive than showing vertical bars with a left-to-right orientation (Few 2004, 182). One common exception to this is where data are to be ordered according to units of time. Time is often visually interpreted as flowing left to right, from past to present to future.

In the case of boxplots, for example, a horizontal orientation allows for a more accurate visual comparison than a vertical orientation, since it is easier for the human eye to follow an imaginary vertical line than an imaginary horizontal line, as can be seen from the figure below. Note, also, that the long x-axis labels are not readable in the panel on the left since they overlap, but they become readable with a horizontal orientation in the panel on the right.

```
# Store figures into a list.
gg.list <- list()

library(ggplot2)

x <- mpg # miles per gallon data set.
x$car <- paste(x$manufacturer, x$model)

gg <- ggplot(x, aes(x = car, y = hwy, group = car))
gg <- gg + geom_boxplot() + theme_bw() + xlab("Miles per gallon (highway)")
gg.list[["vertical orientation"]] <- gg

gg.list[["horizontal orientation"]] <- gg + coord_flip()

# ---
# Both figures into a single output figure.
```

```
library(cowplot)
plot_grid(gg.list[[1]], gg.list[[2]], rel_widths = c(2, 2.5))
```

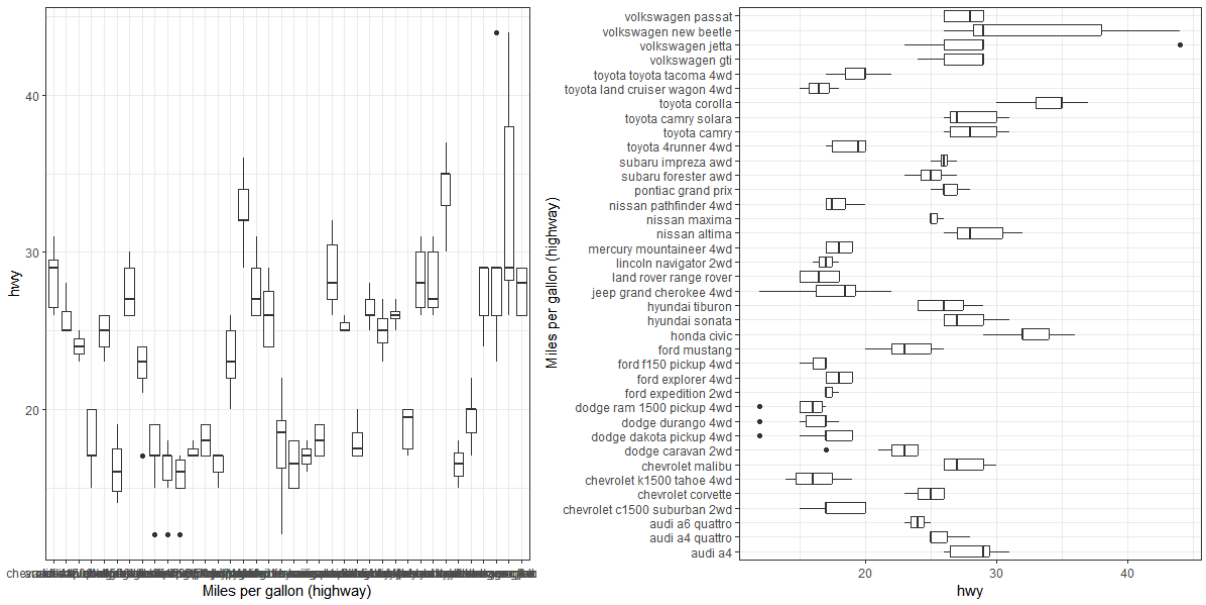

Figure 10: Changes to orientation can improve both readability of labels and the visual interpretation of data.

Auxiliary elements

Generally speaking, when designing a data visualisation, we need to be wary of what Tufte calls "chart junk" – any element of a chart that does not add to or enhance the information on display. Data are, of course, the star of the show, so should receive the most space. Meanwhile, all additional elements (lines, colours, symbols, etc.) should be assessed for their contribution to the interpretation of the data. If there is no contribution, they can be removed. A typical example of a redundant element in a data visualisation is a figure that has 5 panels (sub-figures or facets), each with x- and y-axis. To allow for efficient comparison of the data shown in the different panels, a single y-axis suffices if the panels are all aligned on a single row. This creates more space for the data and synchronises y-axis limits and data positions as a side benefit.

Some helpful additional elements can include lines at relevant points. Examples include a vertical line at x=0 (e.g., for time 0), a horizontal line at y=0 (e.g., to indicate the point of no change, or a diagonal (at y=x, e.g., to indicate no difference between x and y). Consider whether the auxiliary element is better plotted under or over the data (which generally depends on the sequence of adding graphics elements).

A local scatterplot smoother (loess, lowess, polynomial) can be particularly helpful to identify relationships with limited model assumptions. The confidence band (pointwise confidence intervals) should only be shown if relevant.

Example: The figure below displays changes from a starting point (time 0) over time. A plain figure (left panel) requires the reader to read the y-axis labels to identify the point of no change (y=0). The addition of a line (middle panel) indicating the point of no change makes reading easier, and choosing an axis range symmetric around the point of no change (right panel) allocates increases (changes upward) to the upper half of the visualisation and decreases (changes downward) to the lower half, making reading and interpretation more intuitive.

```r
# Function for data set generation.
make.data <- function(
    x = c(0, 0.5, 1, 2, 4, 8, 12, 16, 24),
    y = exp(-0.2 * x) - exp(-0.21 * x),
    sd = 0.25, # std dev of y
    seed = 4384590,
    n = 50) {
  # Setting the random number seed for reproducibility.
  set.seed(seed)
  # Creation of x- and y-variables.
  x2 <- rep(x, n)
  y2 <- NULL
  for (i in 1:n) {
    y2 <- c(y2, y * (2 * (n / 4 - i)) + rlnorm(length(y), sd = sd))
  }

  # Creation of an identifier for each profile.
  ID <- factor(rep(1:n, rep(length(y), n)))

  # Composition of the data set.
  df <- data.frame(PD = 100 * y2, time = x, ID = ID)

  # Addition of a baseline variable.
  BL <- df[df$time == 0, c("ID", "PD")]
  names(BL) <- c("ID", "BL")
  df <- merge(df, BL)

  # Addition of change from baseline.
  df$Change <- df$PD - df$BL

  # Definition of treatment.
  df$trt <- ifelse(df$BL > mean(df$BL), "active", " placebo")

  return(df)
}

# Generate the data.
x <- make.data()

# ---
# Figures.

library(ggplot2)
gg <- ggplot(x, aes(x = time, y = Change, group = ID, color = ID))
gg <- gg + theme_bw()
gg <- gg + xlab("Time [h]") + ylab("Change from baseline")
gg <- gg + geom_line(linewidth = 1.1) + theme(legend.position = "none")
gg <- gg + facet_grid(. ~ trt)

# Addition of an auxiliary line at y=0.
gg2 <- gg + geom_hline(yintercept = 0, linewidth = 1.2)
```

```
# Symmetric y-axis limits.
gg3 <- gg2 + ylim(c(-1, 1) * max(abs(x$Change)))

# Arranging all plots into one figure.
library(cowplot)
plot_grid(gg, gg2, gg3, nrow = 1)
```

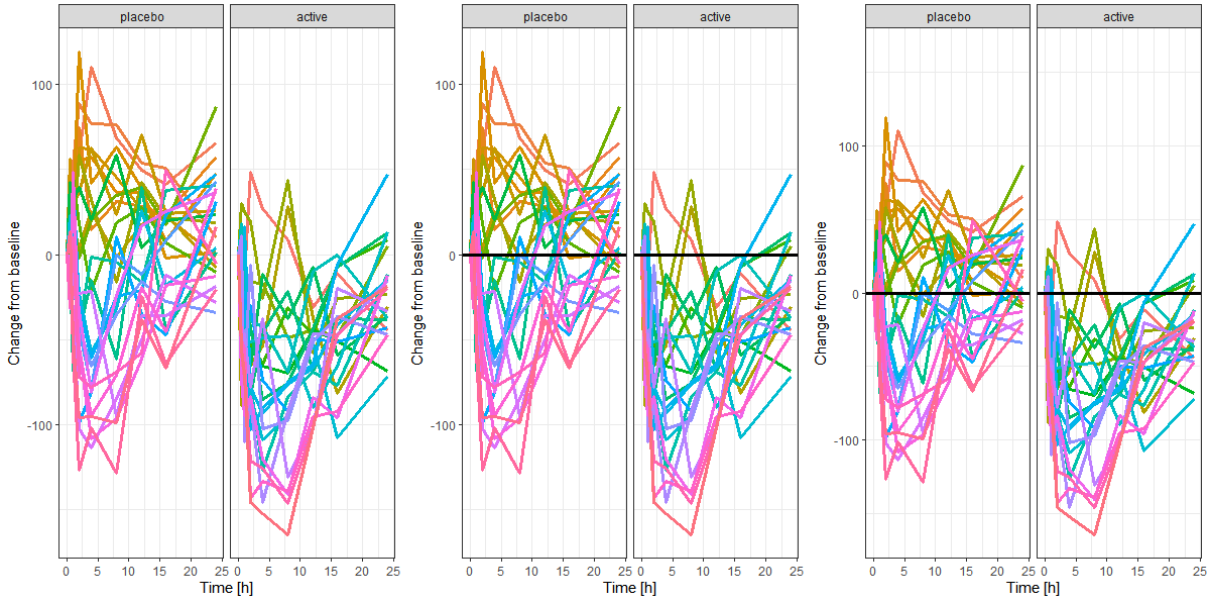

Figure 11: Adding elements, such as lines to indicate the point of no change, can improve readability.

Three-dimensional charts

Three-dimensional displays of data may look striking, in some respects, but accurate reading and interpretation is not straightforward with such presentations.

Example: The left panel of the figure below displays a single number, 10, as a 3D bar chart. The correct identification of the value is indicated by the red "X": the height of the bar must be projected against the rear wall (the axis) from the viewing position. The panel on the right positions bars on a tilted surface, resulting in perceived height differences when there are none: the same four numbers, 10, 20, 30, and 40, are shown in each row and column.

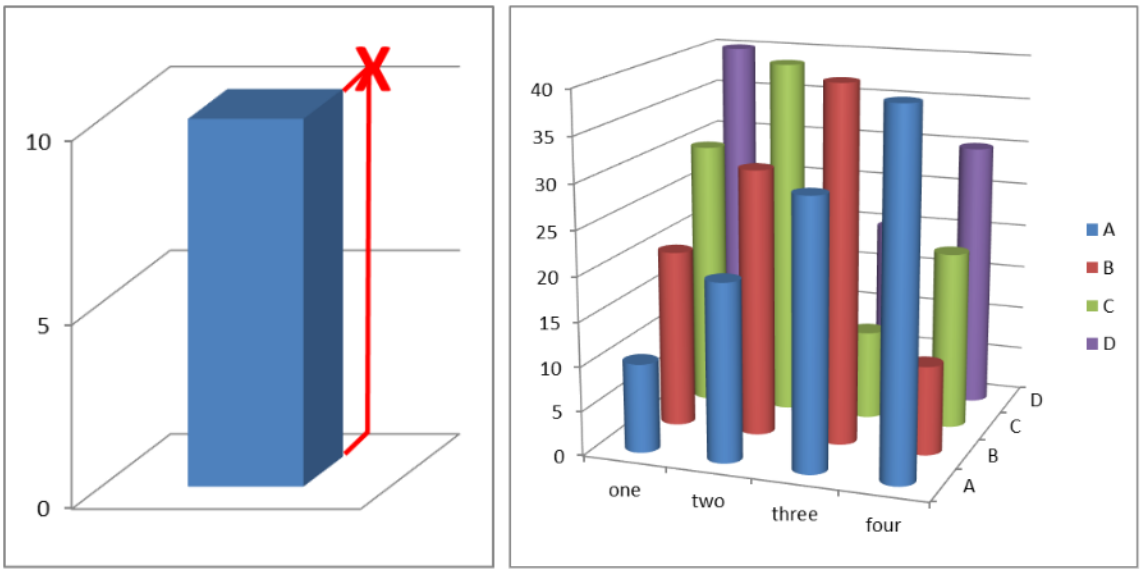

Figure 12: Reading off data values from 3D charts is difficult.

4.2 Elements of tables

Tables can be an efficient means of visualisation, just like graphics – and they also can be misleading or cumbersome to read if not properly presented. Tables should in particular be considered for a small number of data points or if precise numbers are to be shown. For example if numbers need to be overlaid on graphics to aid readability and interpretation (for example, appending digits to the end of the bars in a barchart), one might consider omitting the graphic entirely and displaying the numbers as a table.

Tables have design elements just like graphics. Some of them are discussed in the following.

Layout

One design element for tables is the layout, i.e., rows and columns. The layout should be chosen actively with the reader in mind. It is generally easier to compare numbers vertically than horizontally.

Digits

The number of digits should be consistent within a table row or column (i.e., the same variable) but can differ across variables (e.g., if the table contains small and large numbers). Unnecessary precision should be avoided, a lower number of digits might facilitate reading and comparison.

Alignment

Numbers should be right-aligned to allow for easy identification of large and small numbers. In this way, large numbers literally stick out.

Multiple numbers in table cells

If table cells contain more than a single value (e.g., means and confidence intervals), it should be considered if there is a better way to display the data for easy reading. Options include splitting the numbers into multiple table cells, rows, or columns, or reducing the font size for the less relevant numbers (e.g., means might be more important than confidence intervals).

Orientation

Landscape orientation is discouraged unless it cannot be avoided. Rotating a document in order to read a part of it is cumbersome for the reader, whether in print or on screen. Splitting a table into two might be a remedy to avoid landscape orientation – but one will need to consider whether splitting the table makes important comparisons more difficult to achieve.

Fonts and colours

Different fonts and colours can be used if these serve a purpose, such as highlighting a particular value. Use of different fonts or colours for decorative purposes is discouraged.

5 Choosing a visualisation type

5.1 Goals and audience

The process of deciding what type of data visualisation to create begins with a simple question: Why am I doing this? Data visualisations must serve a purpose so all decisions about chart type, design, layout, and more, should flow from a clear understanding of the intended purpose of a graphic.

Christian Hennig, a statistics professor at the University of Bologna, suggests working through the following questions:

1. Is the aim of the graph to find something out ("analysis graph"), or to make a point to others?
2. What do you want to find out?
3. Who is the audience for the graph? (It may be yourself.)

Identifying the target audience at an early stage is crucial, as different groups of people are likely to have different levels of graph literacy depending on their education level, technical expertise, prior exposure to data visualisation formats, and other factors. Design decisions that do not properly account for the needs of the intended audience will fail to achieve their aims.

Checklist

Before any design decisions are made:

- Be clear about the intended purpose of a data visualisation.
- Understand your target audience, including their needs and familiarity with different visualisation types.

5.2 Data

Software tools generally allow users the flexibility to visualise data in whatever format they choose. However, decisions about chart type must be informed either by the kind of data at hand or the data relationship that is of interest. Format choice must then be further guided by audience needs, as previously discussed.

There are a range of online tools to help decide on a visualisation type. We recommend the following:

Type

From Data to Viz presents users with a series of decision trees, each leading to different recommended chart formats depending on the type of data selected (numeric, categoric, etc.).

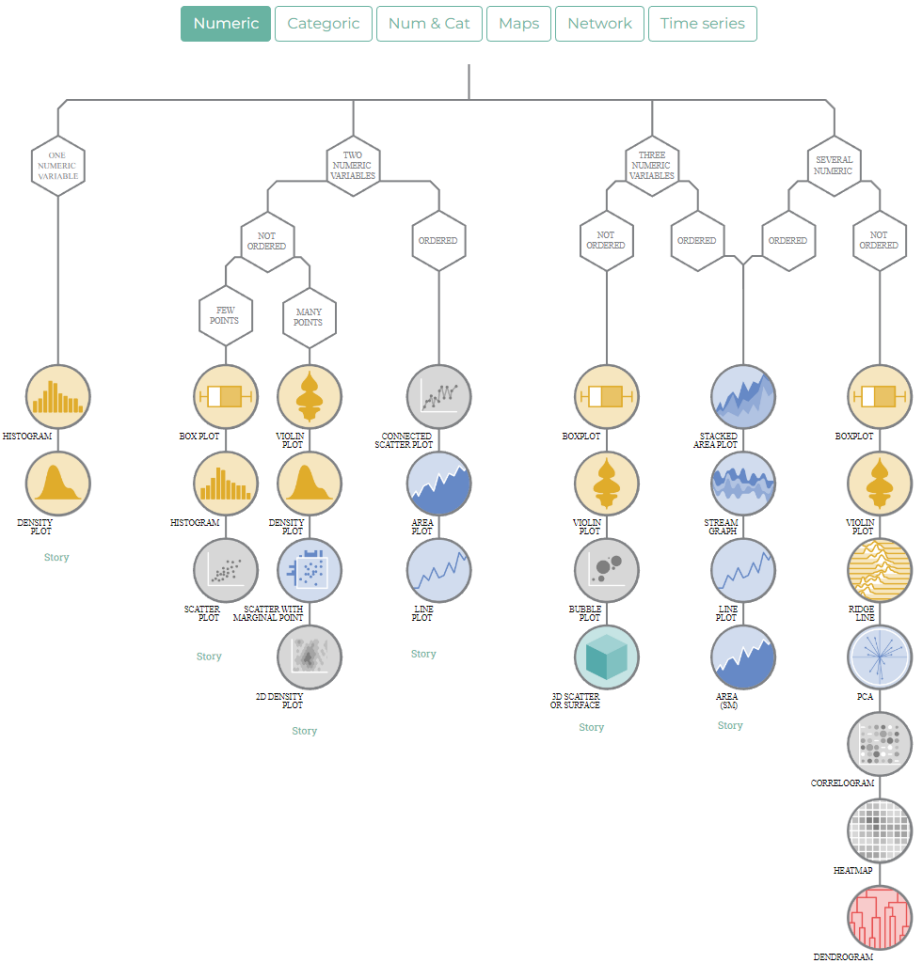

Figure 13: Screenshot of From Data to Viz.

Relationship

Visual Vocabulary is a site developed by the data visualisation team at the *Financial Times*. It allows users to narrow down the choice of chart type based on the data relationship that

is “most important in your story”. Relationship options include deviation, correlation, change versus time, ranking, and more.

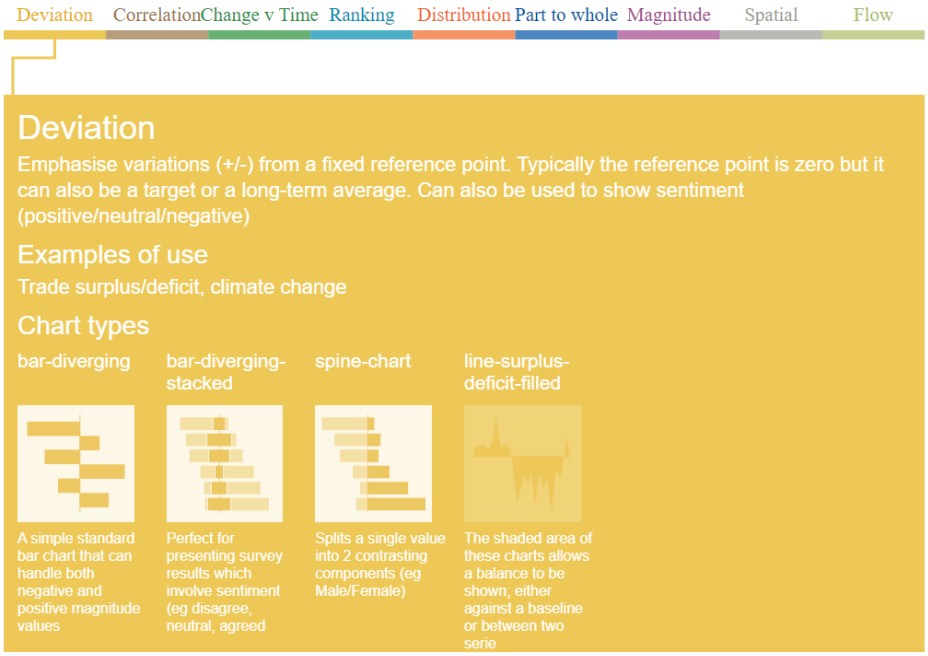

Figure 14: Screenshot of Visual Vocabulary.

Checklist

- Choose a chart type based on either data type or data relationship.
- When selecting from a range of suitable chart types, keep in mind the needs of the target audience and their likely levels of graph literacy.

> **Note**
>
> Further general guidance is provided by Few’s *Show Me the Numbers* (Few 2004) and Robbins’ *Creating Better Graphs* (Robbins 2006).

6 Styling for accessibility

6.1 Principles of styling charts

The two charts below show the same data using the same type of chart – guinea pig tooth growth data visualised on a horizontal bar chart. However, the clarity of the two charts is distinctly different. The choice of colours, addition of text annotations, change of font size, and more informative labels make the chart on the right-hand side much easier to interpret.[1] In this section, we’ll explore each of these elements in detail, and discuss how to style different elements of charts to improve accessibility and interpretability of your data visualisations.

[1] For an alternative presentation of the same tooth growth data, see this discussion (with code) in our GitHub repository.

```r
library(ggplot2)
library(dplyr)
library(ggtext)
plot_data <- ToothGrowth %>%
  mutate(dose = factor(dose)) %>%
  group_by(dose, supp) %>%
  summarise(len = mean(len)) %>%
  ungroup()

# Unstyled plot
ggplot(
  data = plot_data,
  mapping = aes(x = len, y = dose, fill = supp)
) +
  geom_col(position = "dodge")

# Styled plot
ggplot(
  data = plot_data,
  mapping = aes(x = len, y = dose, fill = supp)
) +
  geom_col(
    position = position_dodge(width = 0.7),
    width = 0.7
  ) +
  scale_x_continuous(
    limits = c(0, 30),
    name = "Tooth length"
  ) +
  geom_text(
    mapping = aes(label = round(len, 0)),
    position = position_dodge(width = 0.7),
    hjust = 1.5,
    size = 6,
    fontface = "bold",
    colour = "white"
  ) +
  scale_fill_manual(values = c("#9B1D20", "#3D5A80")) +
  labs(
    title = "Tooth Growth",
    subtitle = "Each of 60 guinea pigs received one of three dose levels of
       vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods:
       <span style='color: #9B1D20'>**orange juice**</span> or
    <span style='color: #3D5A80'>**ascorbic acid**</span>.",
    y = "Dosage (mg/day)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    plot.title = element_textbox_simple(face = "bold"),
    plot.subtitle = element_textbox_simple(
```

```
      margin = margin(t = 10),
      lineheight = 1.5
    ),
    plot.title.position = "plot",
    plot.margin = margin(15, 10, 10, 15),
    panel.grid = element_blank(),
    axis.text.x = element_blank()
  )
```

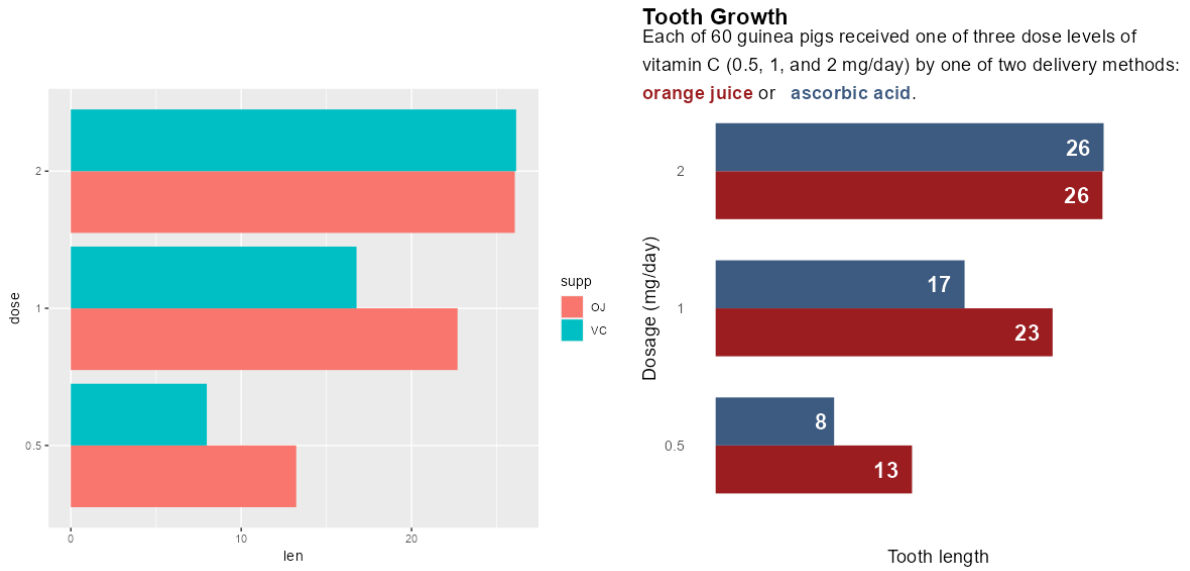

Figure 15: Before and after bar charts visualising tooth growth of guinea pigs.

6.2 Colours

Colour choices can strongly impact the accessibility of your data visualisation. The correct use of colour can also help to emphasise the story you are trying to tell.

Before you get into choosing colours, ask yourself the question: do I really need to use colour here? Beecham et al. (2021) showed that the use of colour is one of the least effective methods for visually communicating differences between variables.

```
library(dplyr)
library(ggplot2)
plot_data <- mtcars %>%
  mutate(car = rownames(mtcars))

# Colour all bars
ggplot(
  data = plot_data,
  mapping = aes(
    y = reorder(car, disp),
    x = disp,
    fill = car
  )
```

```
) +
  geom_col() +
  labs(
    x = "Variable 1",
    y = ""
  ) +
  coord_cartesian(expand = FALSE) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    legend.title = element_blank(),
    plot.title = element_text(
      face = "bold",
      margin = margin(b = 10)
    ),
    plot.title.position = "plot",
    plot.margin = margin(15, 10, 10, 15)
  )

# Highlight one bar
ggplot(
  data = plot_data,
  mapping = aes(
    y = reorder(car, disp),
    x = disp,
    fill = (car == "Maserati Bora")
  )
) +
  geom_col() +
  scale_fill_manual(values = c("#AFE1AF", "#7a9d7a")) +
  labs(
    x = "Variable 1",
    y = ""
  ) +
  coord_cartesian(expand = FALSE) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    legend.title = element_blank(),
    plot.title = element_text(
      face = "bold",
      margin = margin(b = 10)
    ),
    plot.title.position = "plot",
    plot.margin = margin(15, 10, 10, 15)
  )
```

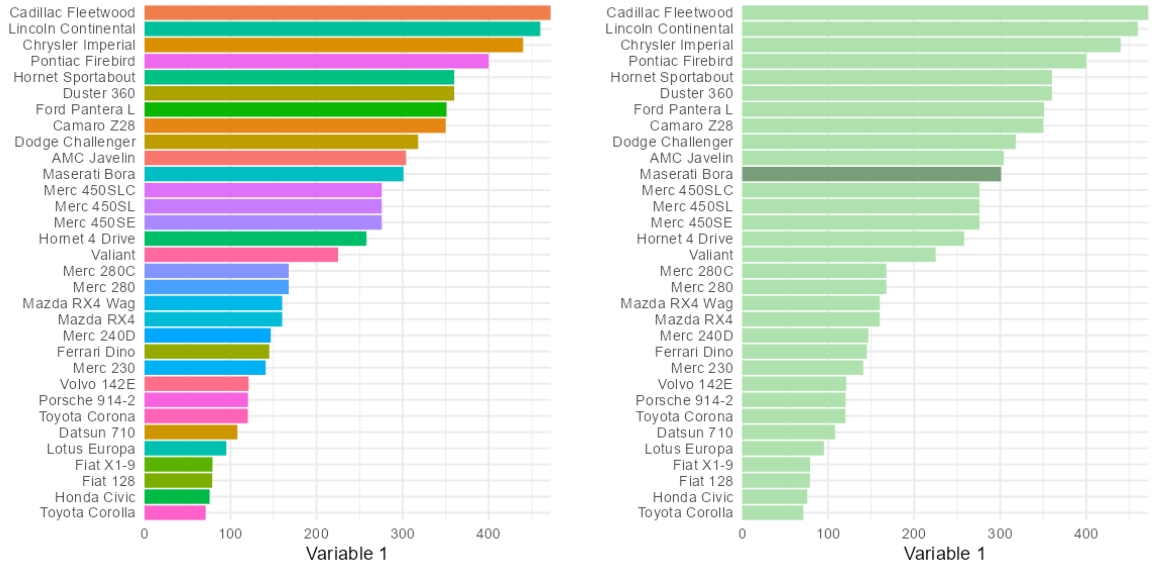

Figure 16: Before and after bar charts visualising car data showing colouring vs highlighting.

In the bar chart example above, rather than duplicating the information on the y-axis and colouring each bar a different colour, a better approach is to use colour to highlight a specific element of the data, e.g., a specific car, which focuses the reader's attention straight to the point you are trying to make.

It's useful not to rely on colour as the only factor which distinguishes data points in different groups. For example, in scatter plots, as well as colouring points in two groups either green or orange, the points could be encoded as circles and triangles.

```
library(readr)
library(dplyr)
library(tidyr)
library(ggplot2)
wheels <- read_csv(
  "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2022/2022-08-09
  )
plot_data <- wheels %>%
  select(country, height, diameter) %>%
  drop_na() %>%
  filter(country %in% c("USA", "Japan"))

# Colour only
ggplot(
  data = plot_data,
  mapping = aes(
    x = diameter,
    y = height,
    colour = country
  )
) +
  geom_point(size = 3, alpha = 0.8) +
  scale_x_continuous(limits = c(0, 800)) +
```

```r
  scale_y_continuous(limits = c(0, 800)) +
  scale_colour_brewer(palette = "Dark2") +
  coord_cartesian(expand = FALSE) +
  labs(
    title = "Ferris wheels",
    x = "Diameter (ft)",
    y = "Height (ft)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "top",
    legend.title = element_blank(),
    plot.title = element_text(
      face = "bold",
      margin = margin(b = 10)
    ),
    plot.title.position = "plot",
    plot.margin = margin(15, 10, 10, 15)
  )

# Shapes and colours
ggplot(
  data = plot_data,
  mapping = aes(
    x = diameter,
    y = height,
    colour = country,
    shape = country
  )
) +
  geom_point(size = 3, alpha = 0.8) +
  scale_x_continuous(limits = c(0, 800)) +
  scale_y_continuous(limits = c(0, 800)) +
  scale_colour_brewer(palette = "Dark2") +
  coord_cartesian(expand = FALSE) +
  labs(
    title = "Ferris wheels",
    x = "Diameter (ft)",
    y = "Height (ft)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "top",
    legend.title = element_blank(),
    plot.title = element_text(
      face = "bold",
      margin = margin(b = 10)
    ),
    plot.title.position = "plot",
    plot.margin = margin(15, 10, 10, 15)
  )
```

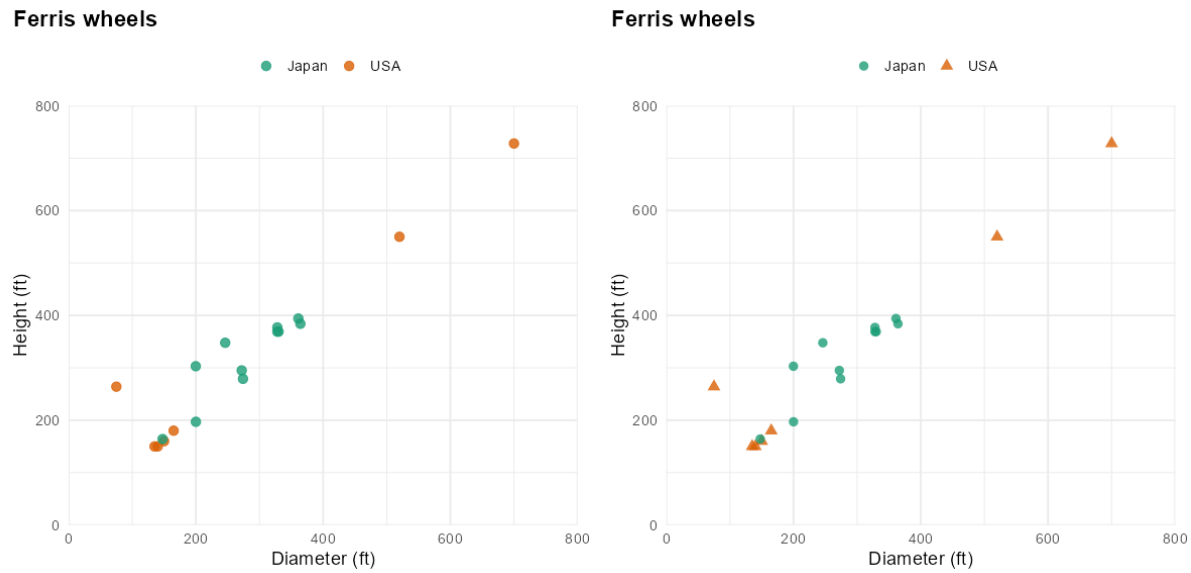

Figure 17: Before and after scatterplots visualising Ferris wheel data showing only dots, vs, dots and triangles to differentiate groups.

Despite the difficulties that can arise from ineffective use, colour remains one of the most common methods for either distinguishing between categories, or showing changes in a value. If you decide to use colours, you then need to decide which colours to use.

Types of colour palette

Colour palettes generally fall into one of three categories, and the type of colour palette you choose should reflect the type of data you are visualising.

- **Sequential**: used to visualise data that is ordered from low to high (or vice versa), e.g., temperature.
- **Diverging**: used to visualise data that is ordered and diverges from an important midpoint, e.g., days of above- or below-average temperature, in two directions.
- **Qualitative**: used to visualise categorical data, where the magnitude of difference between, and ordering of, categories is not meaningful, e.g., distinct railway lines.

```
library(ggplot2)
library(PrettyCols)

# Sequential
ggplot(
  data = data.frame(x = 1:7, y = 1),
  mapping = aes(x = x, y = y, fill = x)
) +
  geom_tile() +
  labs(title = "Sequential") +
  scale_fill_pretty_c("Teals") +
  theme_void() +
  theme(
    legend.position = "none",
```

```
    plot.title = element_text(size = 20, face = "bold"),
    plot.margin = margin(15, 10, 10, 15)
  )

# Diverging
ggplot(
  data = data.frame(x = 1:7, y = 1),
  mapping = aes(x = x, y = y, fill = x - mean(x))
) +
  geom_tile() +
  labs(title = "Diverging") +
  scale_fill_gradient2(low = "#f1a340", high = "#998ec3") +
  theme_void() +
  theme(
    legend.position = "none",
    plot.title = element_text(size = 20, face = "bold"),
    plot.margin = margin(15, 10, 10, 15)
  )

# Qualitative
ggplot(
  data = data.frame(x = 1:7, y = 1),
  mapping = aes(x = x, y = y, fill = factor(x))
) +
  geom_tile() +
  labs(title = "Qualitative") +
  scale_fill_brewer(palette = "Dark2") +
  theme_void() +
  theme(
    legend.position = "none",
    plot.title = element_text(size = 20, face = "bold"),
    plot.margin = margin(15, 10, 10, 15)
  )
```

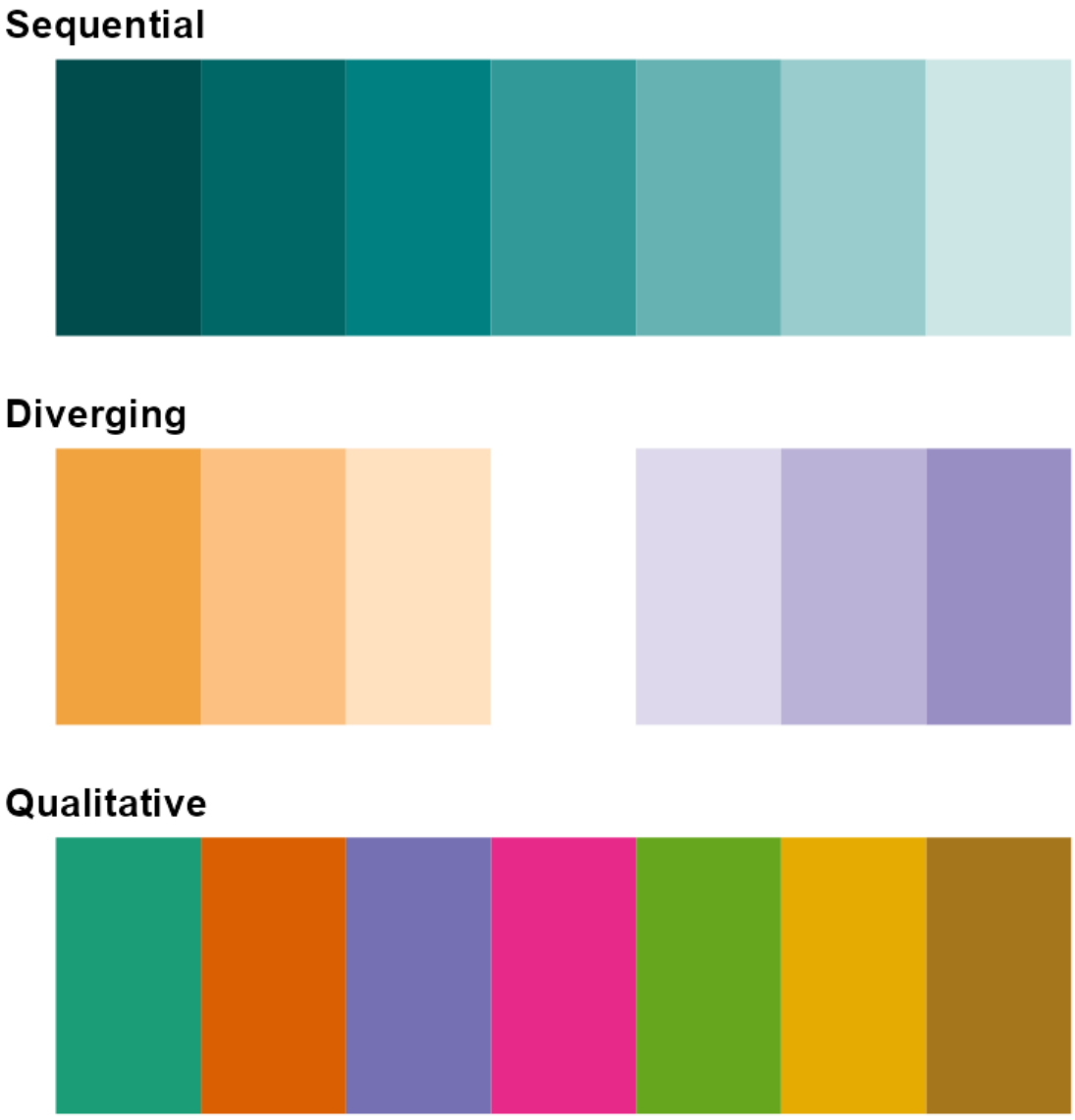

Figure 18: Different types of colour palettes.

Accessible colour palettes

When choosing a particular colour palette, there are several important factors to consider.

Colours should be distinct from each other, for all readers. There are several different forms of colour blindness which can cause some colours to appear indistinguishable. The use of a colour blind checker ("Coblis — Color Blindness Simulator" n.d.) can show you what your plots may look like under different types of colour blindness. This is particularly important for diverging and qualitative palettes where the distinction between hues is used to tell apart different values. In contrast, sequential palettes often use luminosity to show how values change. Tennekes and Puts (2023) note that single hue palettes are a safer choice as they are typically more likely to be colourblind friendly. Tennekes and Puts (2023) also note that the larger the variability across luminosity and chroma (purity of colour), the likelier it is that the colours are more distinguishable by those with colour blindness. Paul Tol (2021) has some very useful resources on choices of colours, and several suggested palettes.

```r
library(ggplot2)
library(dplyr)
library(ggtext)
library(colourblindr)
plot_data <- ToothGrowth %>%
  mutate(dose = factor(dose)) %>%
  group_by(dose, supp) %>%
  summarise(len = mean(len)) %>%
  ungroup()

g <- ggplot(
  data = plot_data,
  mapping = aes(x = len, y = dose, fill = supp)
) +
  geom_col(
    position = position_dodge(width = 0.7),
    width = 0.7
  ) +
  scale_x_continuous(
    limits = c(0, 30),
    name = "Tooth length"
  ) +
  geom_text(
    mapping = aes(label = round(len, 0)),
    position = position_dodge(width = 0.7),
    hjust = 1.5,
    size = 6,
    fontface = "bold",
    colour = "white"
  ) +
  scale_fill_manual(values = c("#9B1D20", "#3D5A80")) +
  labs(
    title = "Tooth Growth",
    subtitle = "Each of 60 guinea pigs received one of three dose levels of
       vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods:
       <span style='color: #9B1D20'>**orange juice**</span> or
    <span style='color: #3D5A80'>**ascorbic acid**</span>.",
    y = "Dosage (mg/day)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    plot.title = element_textbox_simple(face = "bold"),
    plot.subtitle = element_textbox_simple(
      margin = margin(t = 10),
      lineheight = 1.5
    ),
    plot.title.position = "plot",
    plot.margin = margin(15, 10, 10, 15),
    panel.grid = element_blank(),
    axis.text.x = element_blank()
```

```
  )

cvd_grid(g)
```

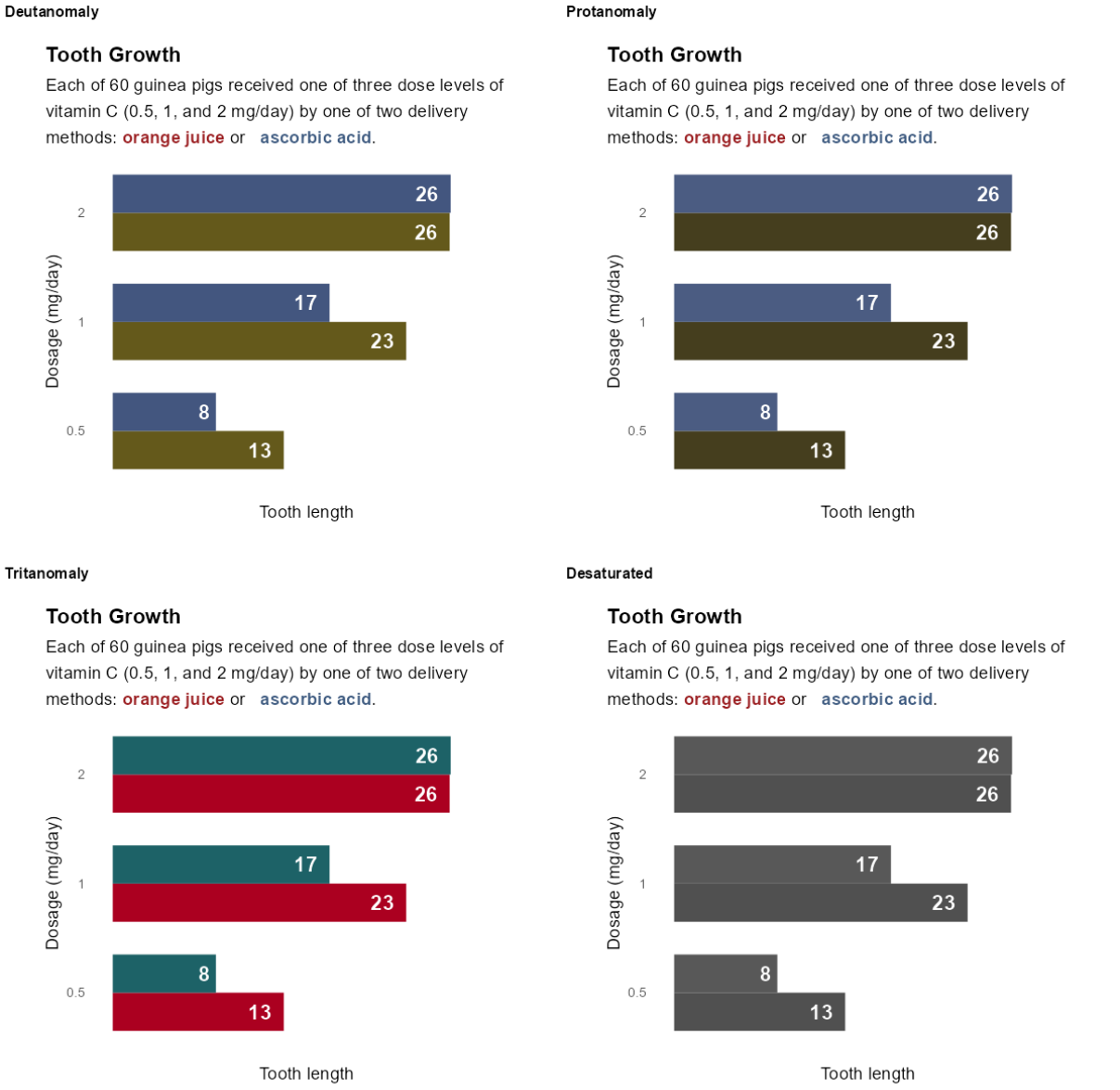

Figure 19: The simulated appearance of the same bar chart with different types of colour blindness.

Colours should appear distinct from each other when printed in black and white. Readers, or indeed academic journals, may choose to print your visualisations in monochrome. Colours that seem visually distinct that have similar luminosity (a measure of how light or dark a hue is) may appear indistinguishable when printed in black and white.

Colours should appropriately match your data. It's important not to play into stereotypes, e.g., choosing pink and blue for data relating to women and men. Don't confuse your readers by flipping common colour associations around, e.g., red for good and green for bad. Muth (2018) discusses the use of colour to display gender data, and many of the points can be extended to other types of data.

Checklist

- Minimise the number of different colours used.
- Check that your colours are colour blind friendly, and have sufficient contrast when viewed in monochrome.
- Check that the type of colour palette matches the type of data visualised.
- Use non-colour elements to distinguish between different groups where possible.
- Avoid stereotypes and confusing associations with different colours.

6.3 Annotations

When talking about adding annotations to data visualisations, we simply mean adding text that adds a comment or clarification in order to help the reader understand the point made in the graphic. Annotations can be used to add details or explanation, highlight an interesting data point, or clarify how the chart should be interpreted. Although additional text can be extremely useful to a reader, it's important not to overload a data visualisation with text as it can be distracting and make it more difficult to focus the point.

A common use of annotation that enhances clarity of data visualisations is directly labelling data points on line graphs or bar charts. In line graphs, it's common to label the value at the end of the line as it's often far away from the y-axis labels (on the left by default). In bar charts, adding the values for each bar at, or near, the end of each bar also means readers don't have to look up the values themselves. This can reduce the amount of eye movement required for readers to find the exact values and allows them to make better, more accurate comparisons. This isn't always required if, for example, these values are additionally provided in a table.

```
library(ggplot2)
library(dplyr)
plot_data <- ToothGrowth %>%
  mutate(dose = factor(dose)) %>%
  group_by(dose, supp) %>%
  summarise(len = mean(len)) %>%
  ungroup()

# Not annotated
ggplot(
  data = plot_data,
  mapping = aes(
    x = len,
    y = dose,
    fill = supp
  )
) +
  geom_col(
    position = position_dodge(width = 0.7),
    width = 0.7
  ) +
  scale_x_continuous(
    limits = c(0, 30),
    name = "Tooth length"
  ) +
```

```r
  scale_fill_manual(
    name = "Supplement: ",
    values = c("#9B1D20", "#3D5A80")
  ) +
  labs(
    title = "Tooth Growth",
    y = "Dose"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "top",
    plot.title = element_text(face = "bold"),
    plot.title.position = "plot",
    plot.margin = margin(15, 10, 10, 15)
  )

# Annotated
ggplot(
  data = plot_data,
  mapping = aes(
    x = len,
    y = dose,
    fill = supp
  )
) +
  geom_col(
    position = position_dodge(width = 0.7),
    width = 0.7
  ) +
  scale_x_continuous(
    limits = c(0, 30),
    name = "Tooth length"
  ) +
  geom_text(
    mapping = aes(label = round(len, 0)),
    position = position_dodge(width = 0.7),
    hjust = 1.5,
    size = 6,
    fontface = "bold",
    colour = "white"
  ) +
  scale_fill_manual(
    name = "Supplement: ",
    values = c("#9B1D20", "#3D5A80")
  ) +
  labs(
    title = "Tooth Growth",
    y = "Dose"
  ) +
  theme_minimal(base_size = 14) +
  theme(
```

```
  legend.position = "top",
  plot.title = element_text(face = "bold"),
  plot.title.position = "plot",
  plot.margin = margin(15, 10, 10, 15)
)
```

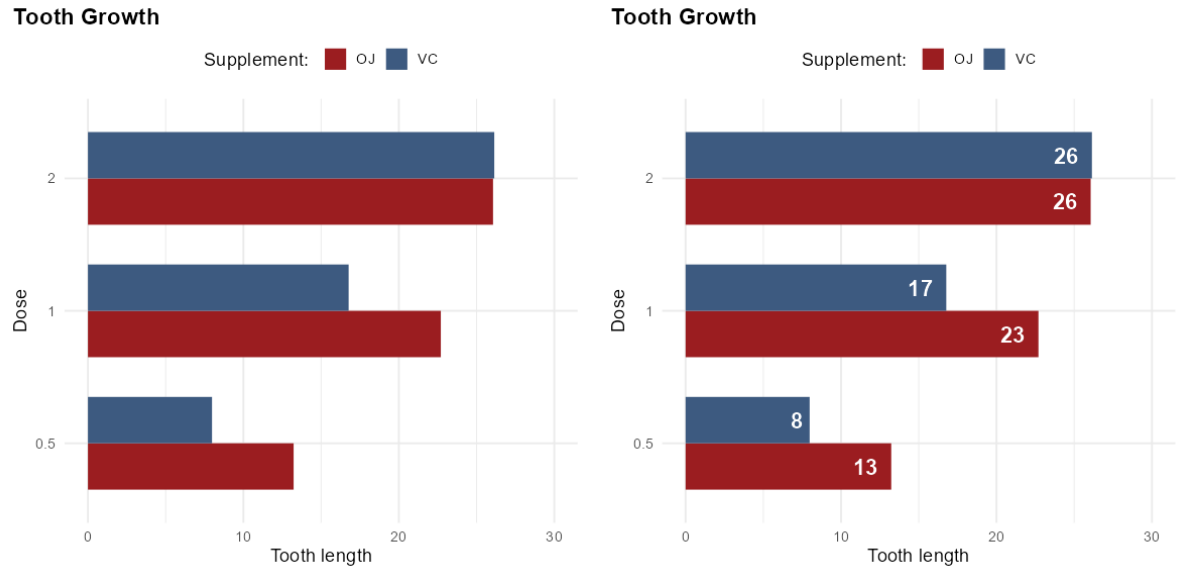

Figure 20: Before and after adding annotations to label a bar chart.

The placement of annotations is also key:

- In a scatter plot, directly labelling every data point isn't usually an option as there are too many points to label without clutter. Instead, you may choose to highlight only one or two interesting points, such as outliers.
- When adding more extensive text to explain an outlier or point of interest, place the text in a relevant position. For example, if you're annotating a line chart to explain a decrease in value in 2008, position the text near to 2008.
- Don't fill every white space with text. "Chart Titles and Text" (n.d.) recommends a maximum of 3 or 4 annotations per chart to avoid overwhelming readers. This also means keeping annotations brief and to the point. This is especially true for data visualisations that accompany journal publications or magazine articles where further textual explanation is likely included already.

Checklist

- Check if there are outliers that need further explanation, or values that could be directly labelled.
- Labels are positioned sensibly, and the data visualisation does not feel crowded with text.
- Make sure there is sufficient contrast between the text and the background.

6.4 Fonts

Font choice is a key component of making your data visualisations easy to understand. Some fonts are easier to read than others, and this is particularly true for those with visual impairments, or a learning disability such as dyslexia. A poor choice of font may make your visualisation inaccessible to a significant number of people in your audience.

Try to minimise the number of different types of font you use: use a fixed set of sizes, a maximum of two different font families, and minimise the use of mixing font faces. This all helps to reduce unnecessary work from a reader when they look at your visualisation. Previous advice relating to choice of colour is also relevant here - it's particularly important to check there is sufficient contrast between the background colour and the text colour.

Font size

Larger fonts are easier to read. It's generally recommended that font size is at least 12pt for printed materials or websites. If you're creating a presentation, fonts should be at least 36pt to make sure they're visible to people nearer the back of the room.

Font family

So what makes a good choice of font family? There are three main types of font:

- **Serif**: serifs are the small strokes on the ends of some longer strokes that occur in some fonts. Fonts that have these serifs are called serif fonts.
- **Sans serif**: Fonts without serifs are called sans serif fonts.
- **Monospace**: monospace fonts are those whose characters each take up an equal width.

There is no consensus as to which type of font (serif, sans serif, or monospace) is more accessible. The simpler characters of sans serif fonts may increase readability for visually impaired readers, while those with dyslexia may find the characters more difficult to tell apart. Serif fonts (such as Times New Roman) can be more difficult to read as the decorative lines distract from the shape of the letter. This is especially true in digital publications where on-screen pixelation can further distort the letters. We'd recommend avoiding serif fonts for any text in images that will appear online. Common sans serif fonts such as Arial, Calibri, and Verdana are often considered to be accessible.

Dyslexie and OpenDyslexic are fonts specifically designed to aid readability for those with dyslexia, as they increase font weight at the bottom of the letters which reduces how much the letters appear to move around. Atkinson Hyperlegible is a font developed by the Braille Institute of America, which is designed to maximise distinction between different characters for low vision readers. It's freely available, and can be downloaded from the Braille Institute or used through Google Fonts. There is conflicting evidence about whether or not specially designed fonts are actually more accessible Wery and Diliberto (2017). Therefore, sticking with a sans serif font is likely a safer choice.

Some fonts, particularly sans serif fonts, have characters which appear very similar to other characters. For example, a capital I, lowercase l, and number 1, can sometimes be indistinguishable. Choose a font face with a distinguishable feature for each letter and number. Source Sans Pro and Verdana both meet this condition.

Font face

Font faces, e.g., bold or italic effects, can be used to emphasise particular parts of text. For example, a plot annotation may highlight a specific value placed in amongst some explanatory text. Italicised text is generally considered more difficult to read (Rello and Baeza-Yates 2016), as is capitalised text when used for the purpose of emphasis. Instead, use bold text to highlight (sparingly!).

Checklist

As you can see, there is no single font recommendation that will ensure your visualisation is accessible to all. However, there are some things you can do to improve accessibility.

- Ensure the font size is at least 12 pt.
- Ensure that a capital I, lowercase l, and number 1 are distinguishable.
- Minimise the number of different fonts used (no more than two).
- Minimise the use of italics and uppercase text, and instead use (sparingly) bold text for emphasis.

6.5 Alt Text

Alt text (short for alternative text) is text that describes the visual aspects and purpose of an image – including charts. Though alt text has various uses, its primary purpose is to aid visually impaired users in interpreting images when the alt text is read aloud by screen readers.

In Green (2023), Mine Dogucu discusses the importance of adding alt text to your data visualisations, to ensure those who are blind or visually impaired don't miss out on the content in your charts. Cesal (2020) provides a simple structure to aid you in writing alt text for data visualisations.

7 Styling for RSS Publications

7.1 Styling charts with different tools

In this section, we discuss the technical details of how to change the style of charts built with some of the most common types of software used to create data visualisations.

R

R (R Core Team 2021) is a programming language which is popular for statistical computing and graphics. There are many packages with R that can be used to create data visualisations, and we don't aim to cover them all here. Instead, we address the most common methods: built-in base R graphics, and the {ggplot2} package (Wickham 2016b).

To help authors with styling their charts to fit in with the guidance in this document, we have developed an R package, {RSSthemes}. We include some examples of using the package here, but encourage readers to check the GitHub repository for any updates.

You can install the {RSSthemes} package from CRAN using:

```
install.packages("RSSthemes")
```

You can install the development version from GitHub (although we can't promise it will be as stable as the CRAN version):

```
remotes::install_github("nrennie/RSSthemes")
```

You can then load the package using:

```
library(RSSthemes)
```

Base R

R has built-in graphics capabilities that allow users to make a wide range of data visualisations without installing any additional packages. This blog post from Jumping Rivers ("Styling Base r Graphics" 2018) provides instructions on how to style graphics created in base R.

Example: changing bar chart colours in base R.

If all of the bars, lines, points, etc. should have the same colour, you can set the `col` argument to have one of the RSS colours. The options are: `signif_red`, `signif_blue`, `signif_green`, `signif_orange`, or `signif_yellow`.

```
barplot(table(mtcars$gear), col = signif_blue)
```

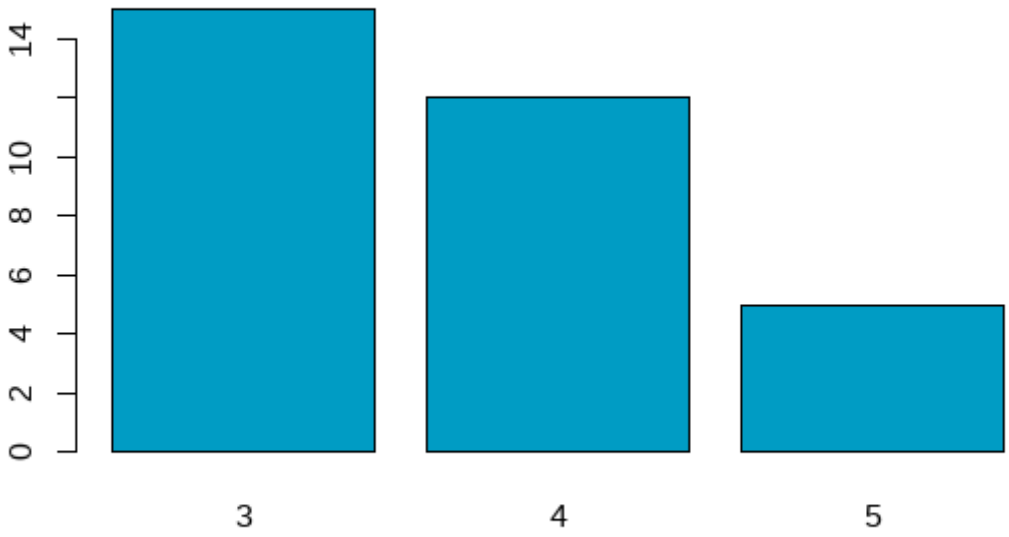

Figure 21: Bar chart with bars coloured in *Significance* blue.

If the colours in your plot are based on values in your data, you can set the default colours using the `palette()` function. Within {RSSthemes}, the `set_rss_palette()` function changes the default colours used. There are currently three palettes available in {RSSthemes}, although we hope to add more in the future. The options are `signif_qual`, `signif_div`, or `signif_seq`.

```
set_rss_palette("signif_qual")
plot(1:4, 1:4, col=1:4, pch=19, cex=3, xlab="", ylab="")
```

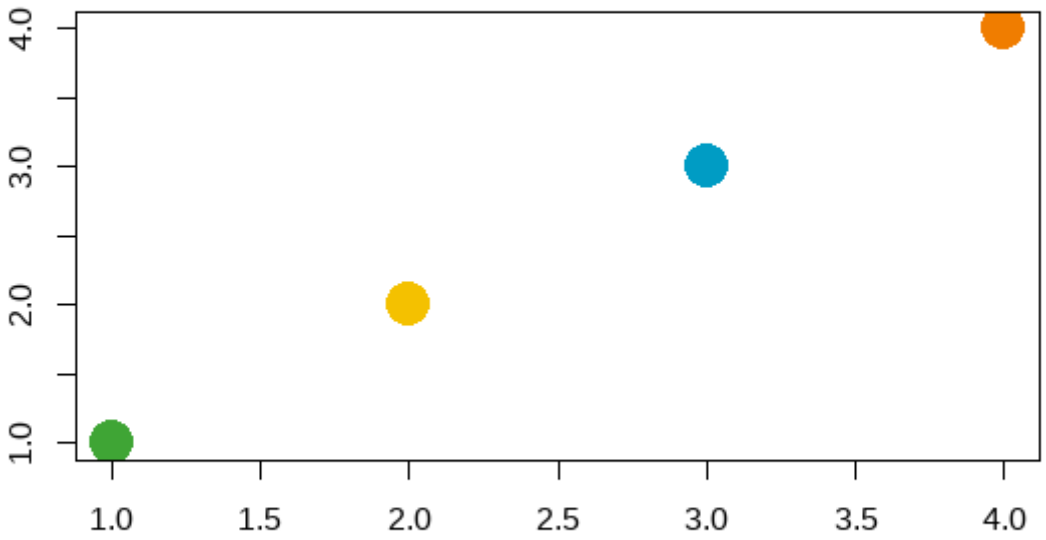

Figure 22: Scatter plot showing the colours from the `signif_qual` palette.

Run `palette("default")` to reset to original base R colours.

Example: changing the styling of base R plots.

Within the `plot()` function (and related base R plotting functions such as `barplot()`, and `hist()`) , there are arguments to control how the non-data elements of the plot look. For example, the `family` argument changes which font family is used. You can also set many of these arguments globally by calling the `par()` function. Within {RSSthemes}, there is a function `set_signif_par()` which sets some default options, including the text alignment and font for all base R plots. We also recommend adding reference lines using the `abline()` function.

```
set_signif_par()
plot(1:4, 1:4, col=1:4, pch=19, cex=3, xlab="", ylab="",
     main = "My Significance Plot",
     sub = "Source: data source")
abline(h=1:4, v=1:4, col = "lightgrey")
```

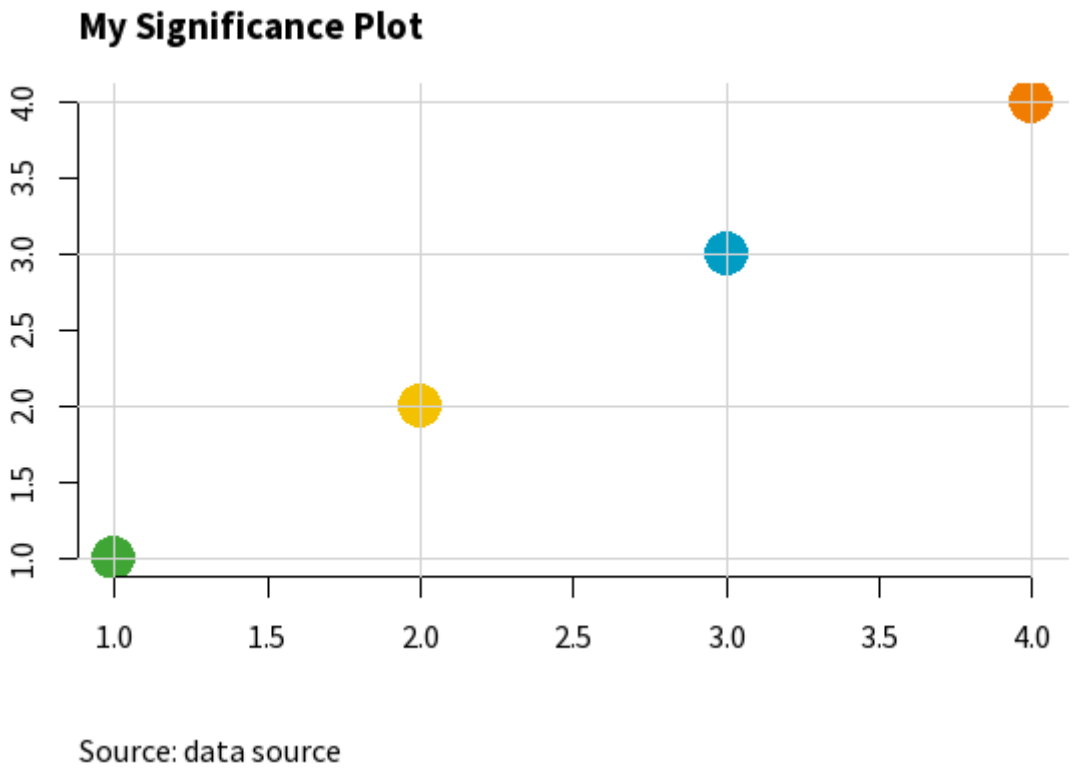

Figure 23: Scatter plot showing the base R styling implemented by `set_signif_par()`.

{ggplot2}

{ggplot2} is an R package within the {tidyverse} framework specifically for creating data visualisations. The package documentation provides guidance on how to create different types of charts. Advice on changing the colours and styles of {ggplot2} visualisations, can be found in the ggplot2: Elegant Graphics for Data Analysis book by Hadley Wickham (Wickham 2016b).

Let's set up a basic data set to make some plots with {ggplot2}.

```
library(ggplot2)
plot_df <- data.frame(x = LETTERS[1:4],
                      y = 1:4)
```

Example: changing the non-mapped colours in {ggplot2}.

In {ggplot2}, the `colour` (or `color`) argument changes the colour that outlines an element, and `fill` changes the colour that fills the element. If all of the, e.g., bars, lines, or points should have the same colour, you can set either the `fill` or `colour` arguments to have one of the RSS colours. The options are: `signif_red`, `signif_blue`, `signif_green`, `signif_orange`, or `signif_yellow`.

```
ggplot(data = plot_df,
       mapping = aes(x = x, y = y)) +
  geom_col(fill = signif_yellow)
```

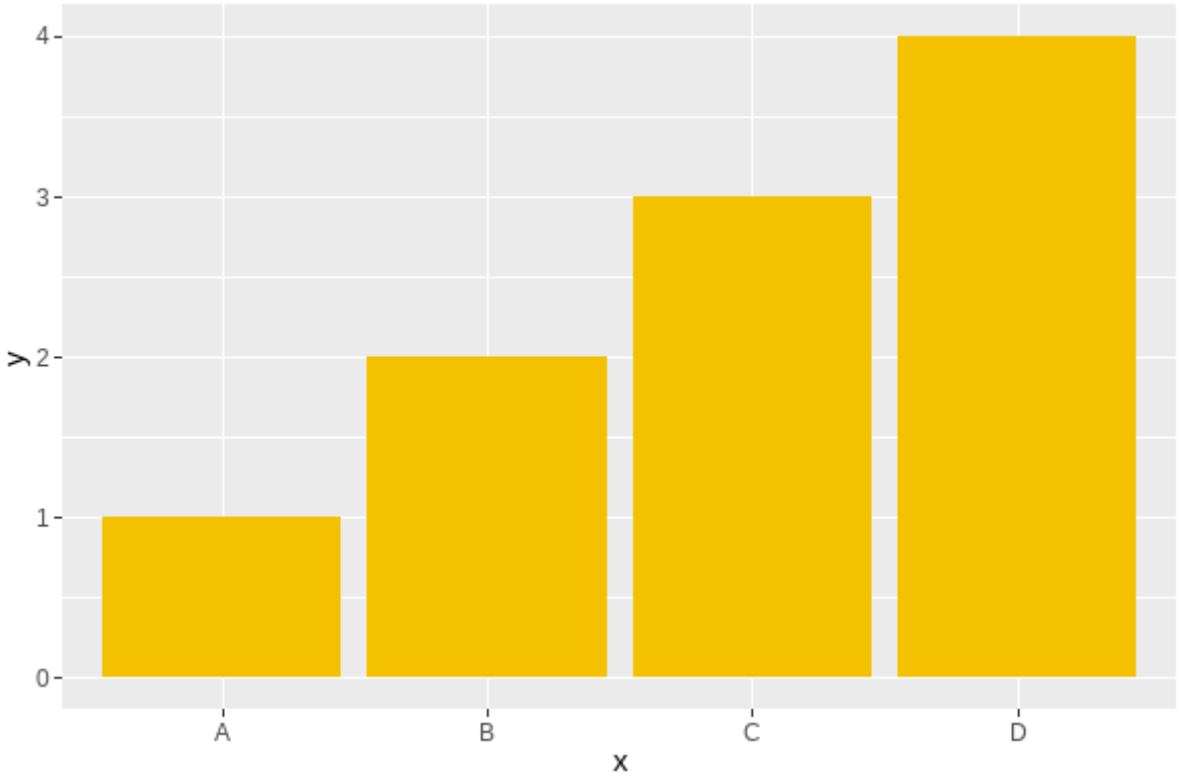

Figure 24: Bar chart with bars coloured yellow.

Example: using a discrete colour scale in {ggplot2}.

For working with qualitative (discrete) data, the best palette to use is `"signif_qual"`. This palette currently only contains four colours.

- Discrete (fill) scale: `scale_fill_rss_d()`

```
ggplot(data = plot_df,
       mapping = aes(x = x, y = y, fill = x)) +
  geom_col() +
  scale_fill_rss_d(palette = "signif_qual")
```

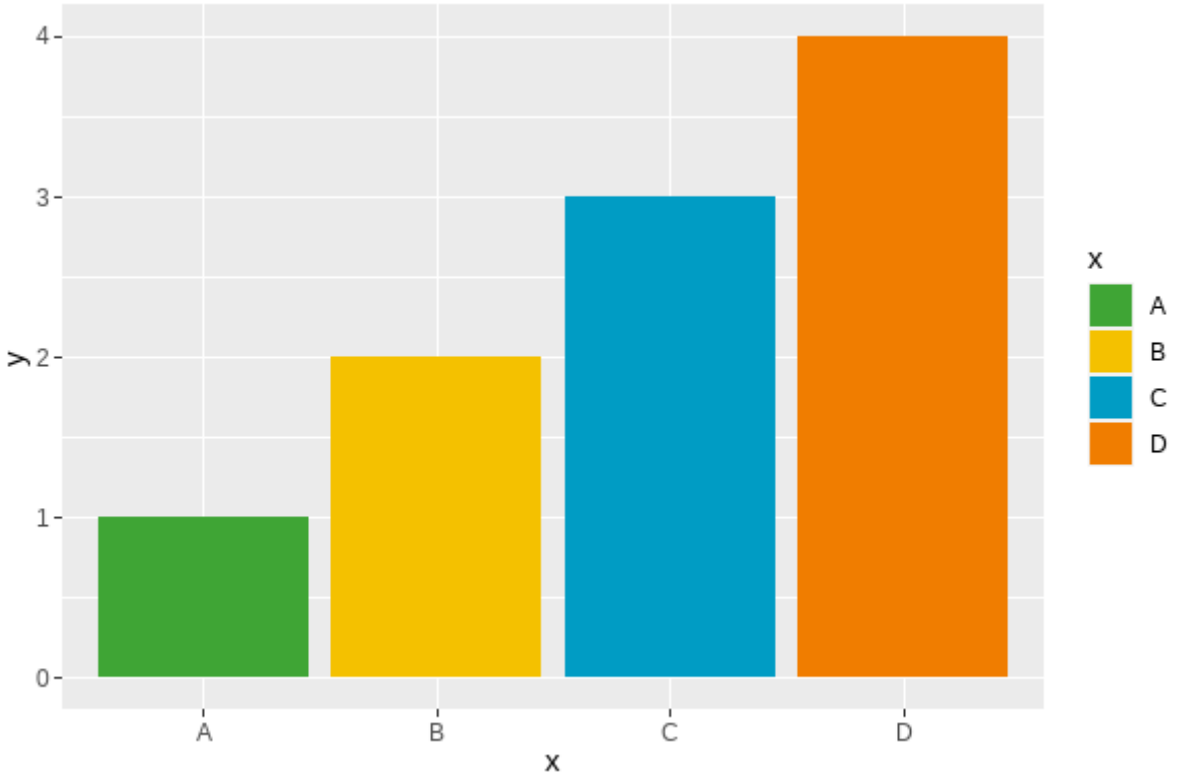

Figure 25: Bar chart with colours from `signif_qual`.

- Discrete (colour) scale: `scale_colour_rss_d()`

```
ggplot(data = plot_df,
       mapping = aes(x = x, y = y, colour = x)) +
  geom_point(size = 4) +
  scale_colour_rss_d(palette = "signif_qual")
```

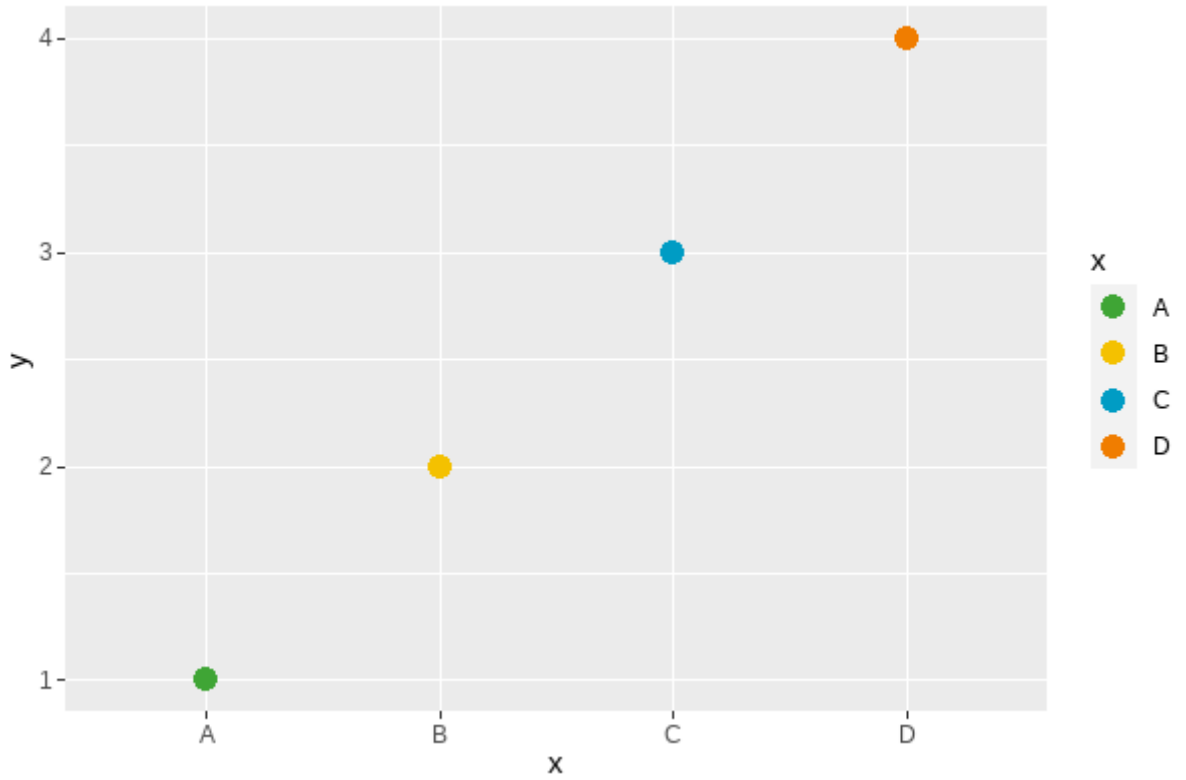

Figure 26: Scatter plot with colours from `signif_qual`.

Example: using a continuous colour scale in {ggplot2}.

Continuous colour scales may be sequential or diverging. For working with sequential (continuous) data, the best palette to use is `"signif_seq"`.

- Continuous (fill) scale: `scale_fill_rss_c()`

```
ggplot(data = plot_df,
       mapping = aes(x = x, y = y, fill = y)) +
  geom_col() +
  scale_fill_rss_c(palette = "signif_seq")
```

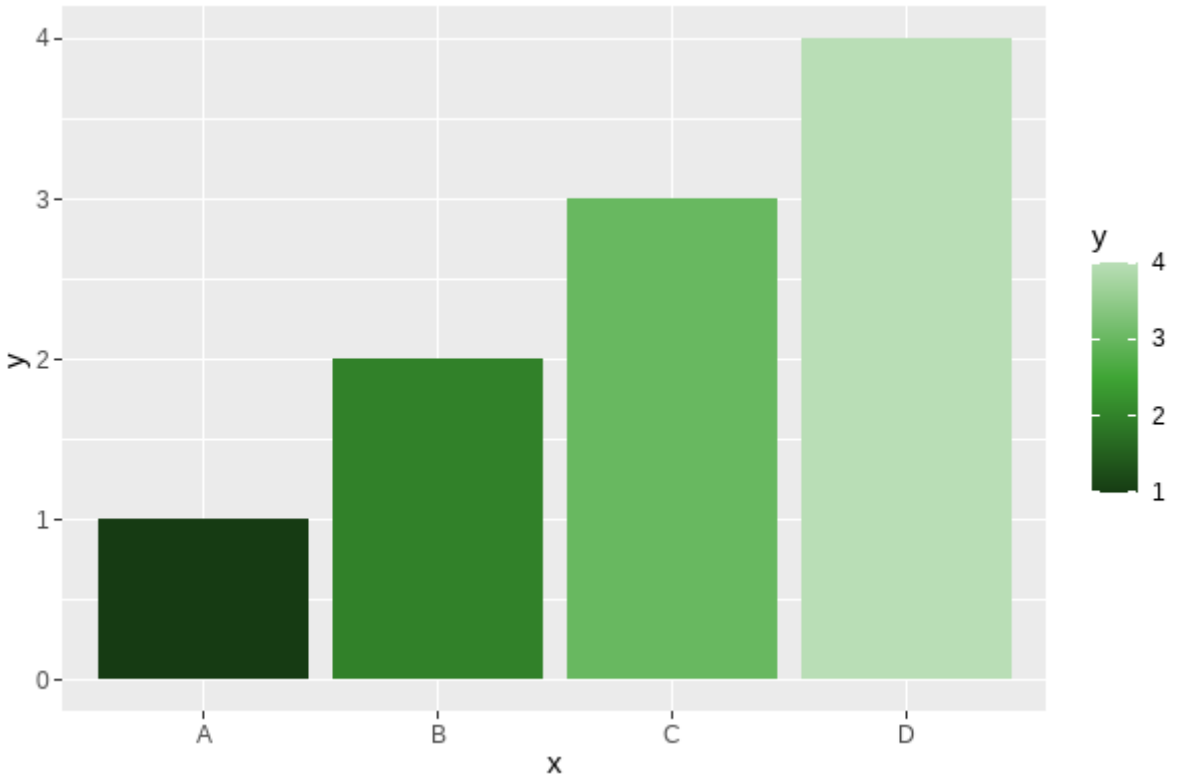

Figure 27: Bar chart showing sequential green colour palette.

- Continuous (colour) scale: `scale_colour_rss_c()`

```
ggplot(data = plot_df,
       mapping = aes(x = x, y = y, colour = y)) +
  geom_point(size = 4) +
  scale_colour_rss_c(palette = "signif_seq")
```

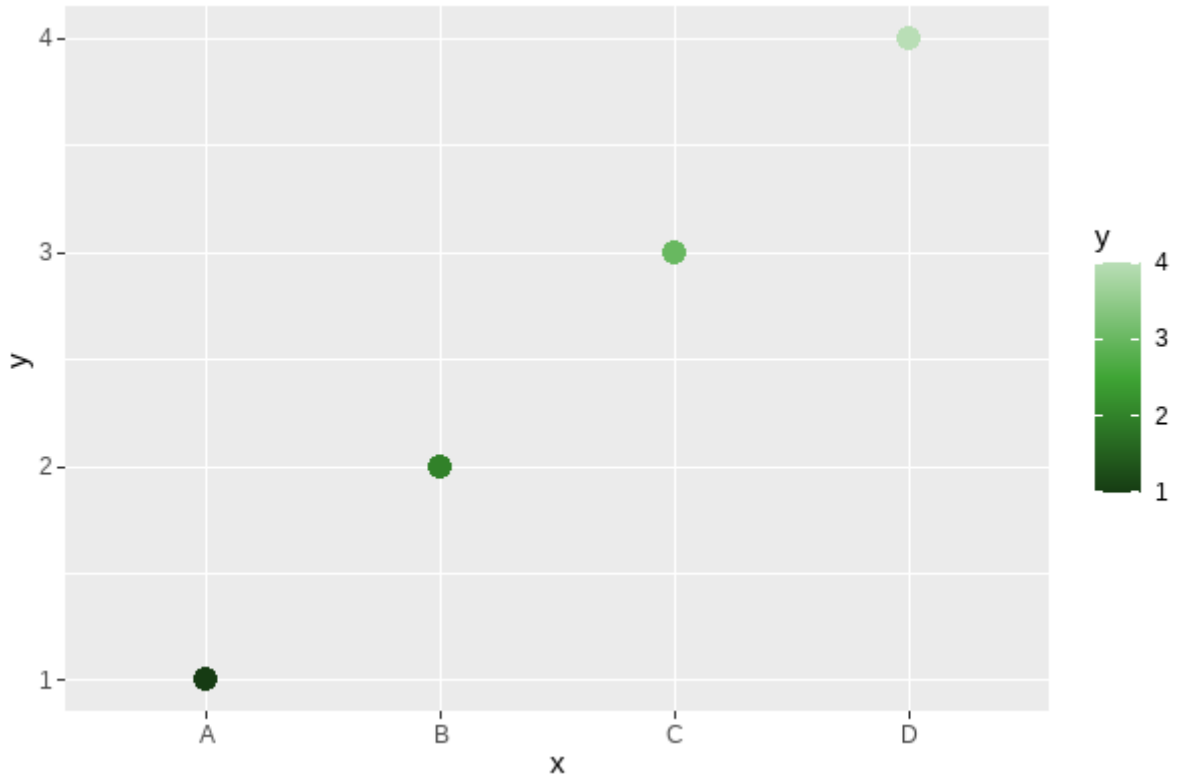

Figure 28: Scatter plot showing sequential green colour palette.

For working with diverging (continuous) data, the best palette to use is `"signif_div"`.

- Continuous (fill) scale: `scale_fill_rss_c()`

```
ggplot(data = plot_df,
       mapping = aes(x = x, y = y, fill = y)) +
  geom_col() +
  scale_fill_rss_c(palette = "signif_div")
```

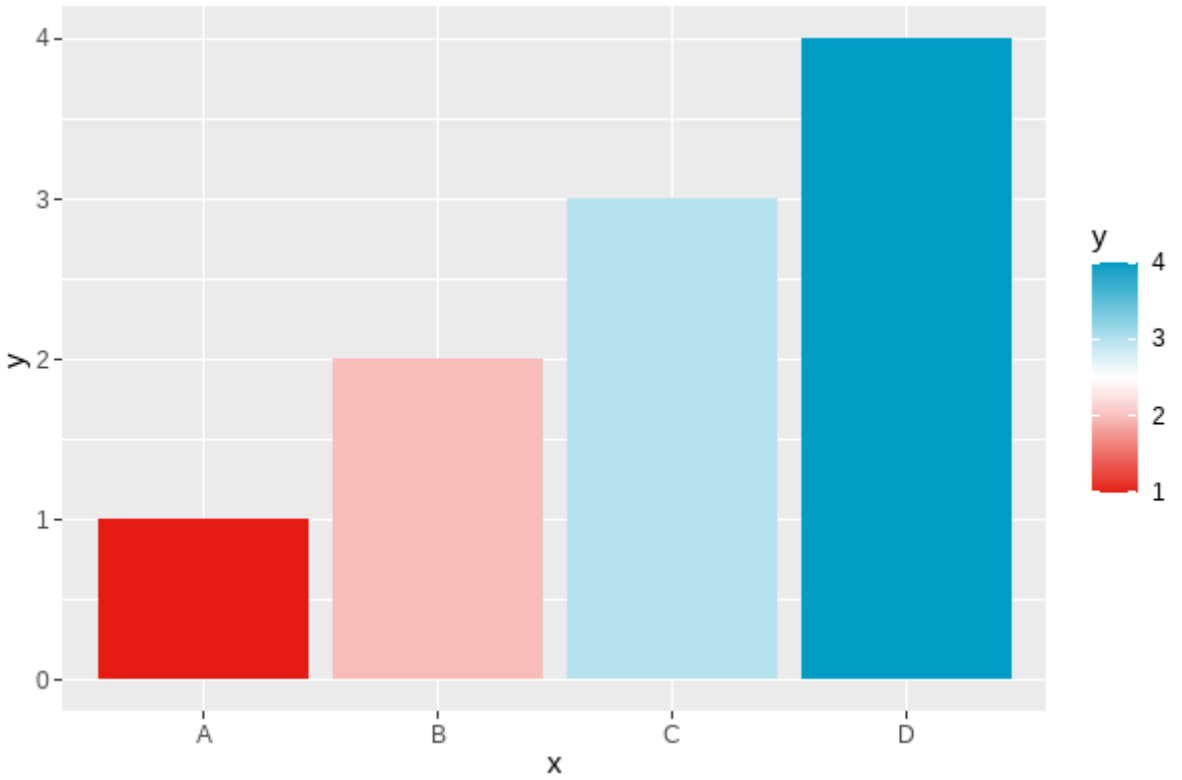

Figure 29: Bar chart showing diverging red to blue palette.

If you want to centre the diverging scale around a different value, you can alternatively pass the pre-defined colours from {RSSthemes} into `scale_fill_gradient2()` in {ggplot2}:

```
ggplot(data = plot_df,
       mapping = aes(x = x, y = y, fill = y)) +
  geom_col() +
  scale_fill_gradient2(low = signif_red, high = signif_blue, midpoint = 2)
```

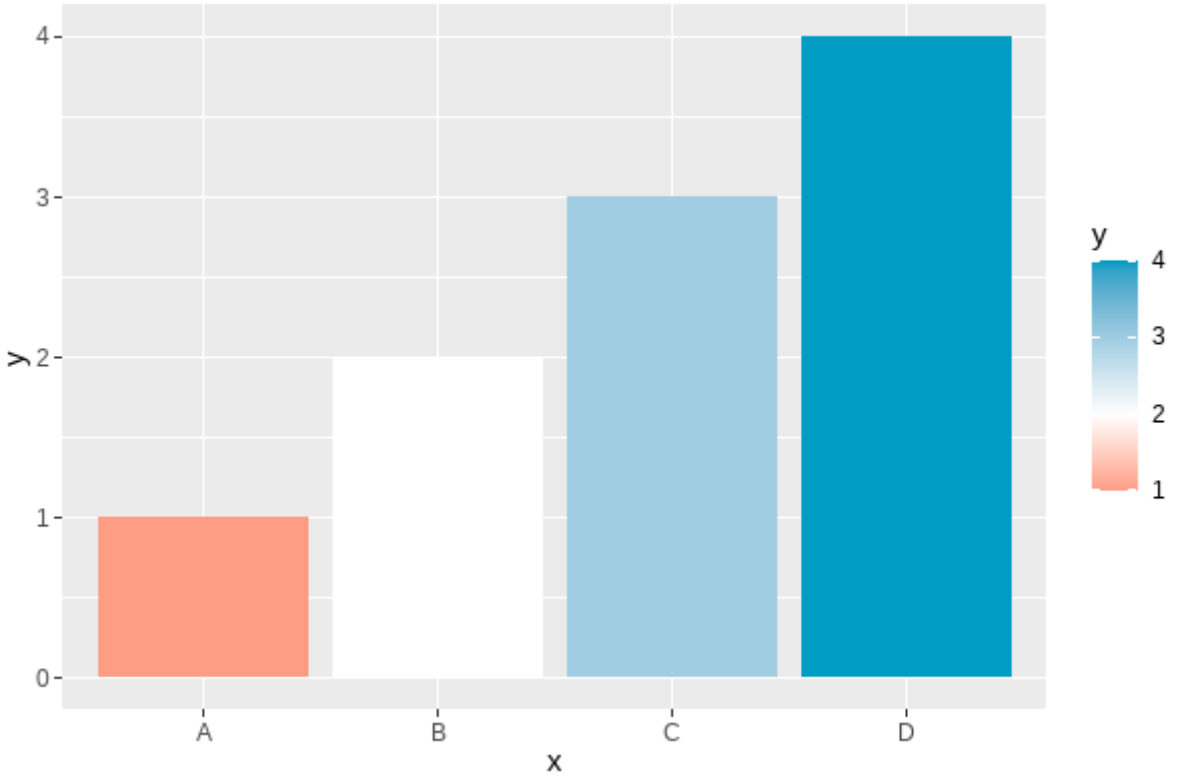

Figure 30: Bar chart showing diverging red to blue palette centred at 2.

Example: changing the theme in {ggplot2}.

Within {ggplot2}, themes allow you to control the appearance of the non-data elements of your plot. The default theme is `theme_grey()` which has a darker background. We recommend using a white or transparent background, such as those created with `theme_minimal()` or `theme_bw()`.

You can also use `theme_significance()` from {RSSthemes} which additionally sets the plot font to one of those used in *Significance* magazine. Check that you have already run `library(RSSthemes)` to ensure the fonts load correctly.

```
ggplot(data = plot_df,
       mapping = aes(x = x, y = y, fill = x)) +
  geom_col() +
  labs(title = "My Significance Plot",
       subtitle = "Some longer sentence explaining what is happening in the chart.",
       caption = "Source: name of data source") +
  scale_fill_rss_d(palette = "signif_qual") +
  theme_significance()
```

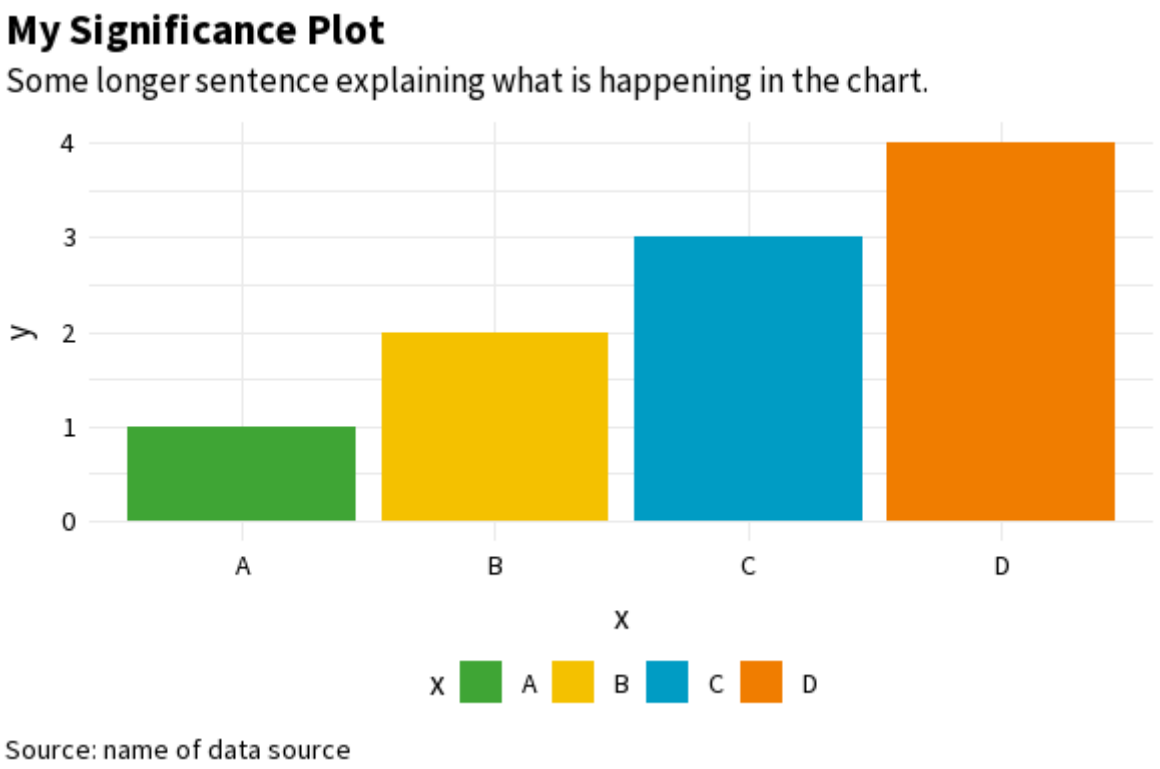

Figure 31: Bar chart styled with `theme_significance()`.

If you find a bug in the {RSSthemes} package, or something that isn't working quite as you expected, please submit a GitHub issue.

Exporting images from R

There are different ways to export and save images from R. Using the *Export* button on the *Plots* pane in RStudio doesn't usually result in images of high enough resolution for publication quality graphics. The minimum image resolution for images published in print is 300 dpi. If you use `ggsave()` from {ggplot2}, 300 dpi is the default resolution. We recommend saving images in PDF or EPS file format as this makes it easier for them to be resized.

Further information on specific image sizes for different RSS publications is given in the Publication specifications section.

As an example, suppose you were creating a plot for the Features section of *Significance* magazine, and you wanted the plot to span two of the three columns. From the table below, the width of the image should be 124 mm. To use the `pdf()` function to save an image, the width and height should be in inches (124 mm ~ 4.88 in). If we want a 2:1 aspect ratio, we make the height equal to half the width:

```
pdf(file = "significance-feature-plot.pdf", # name of file
    width = 4.88,                           # width of plot
    height = 4.88 / 2                       # height of plot
    )
plot(1:4, 1:4)
dev.off()
```

Python

Python is a general-purpose programming language, with libraries available that provide capabilities for data analysis and visualisation.

A work-in-progress Python package for implementing RSS colour palettes can be found at github.com/nrennie/RSSpythemes. If you'd like to contribute to this Python package development, please raise an issue or create a pull request on GitHub.

Matplotlib

Matplotlib is a Python library for creating static, animated, and interactive data visualisations.

Example: changing bar chart colours in matplotlib.

You can change the colour of chart elements in matplotlib using the `color` argument:

````
```{python}
#| message: false
#| eval: false
import matplotlib.pyplot as plt
# generate data
x_vals = ['A', 'B', 'C', 'D']
y_vals = [1, 2, 3, 4]
# create barchart
plt.bar(x_vals, y_vals, color = "#009cc4")
plt.show()
```
````

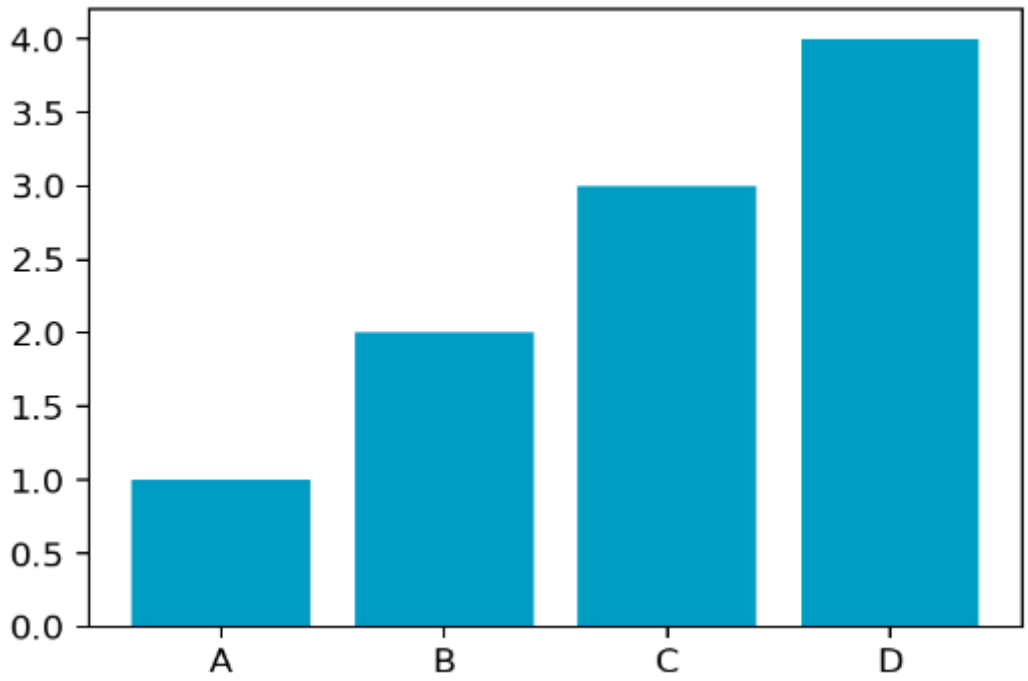

Figure 32: Bar chart with bars coloured in *Significance* blue.

If the colours in your plot are based on values in your data, you can also change the colours used by providing a list of colours:

```
```{python}
#| message: false
#| eval: false
# define colour palette
signif_qual = ['#3fa535', '#f4c100', '#009cc4', '#f07d00']
# create barchart
plt.bar(x_vals, y_vals, label = x_vals, color = signif_qual)
plt.show()
```
```

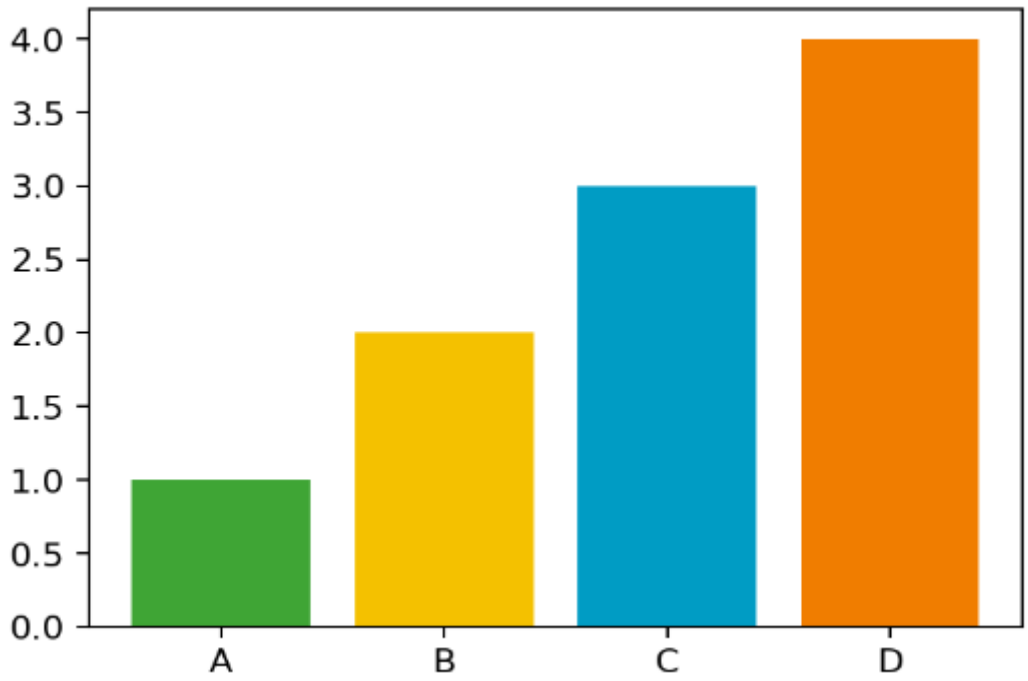

Figure 33: Bar chart showing the colours from the `signif_qual` palette.

Example: changing the font family in matplotlib.

You can change the font used in all elements of the plot using `rcParams`. Good practice when setting a custom font family is to add a generic font family (such as sans serif) as a back up. If you're using a font that isn't pre-installed on your system, you can load it in using `font_manager`:

```
```{python}
#| message: false
#| eval: false
from matplotlib import font_manager
font_manager.fontManager.addfont("SourceSans3-Regular.ttf")
```
```

You can specify a different font family, weight, or size using `fontdic` for individual elements.

```
```{python}
#| message: false
#| eval: false
# define fonts
from matplotlib import rcParams
rcParams['font.family'] = ['Source Sans 3', 'sans-serif']

# create barchart
fig, ax = plt.subplots(1, 1)
plt.bar(x_vals, y_vals, color = signif_qual, label = x_vals)
```
```

```
plt.title('My Significance Plot', fontdict = {'fontsize':14}, loc = 'left')
# add grid lines lines
ax.set_axisbelow(True)
ax.xaxis.grid(color = 'lightgrey')
ax.yaxis.grid(color = 'lightgrey')
# add legend below plot
ax.legend(ncol = 4, loc = 'lower center',
          bbox_to_anchor = (0.5, -0.15), frameon = False)
plt.show()
```

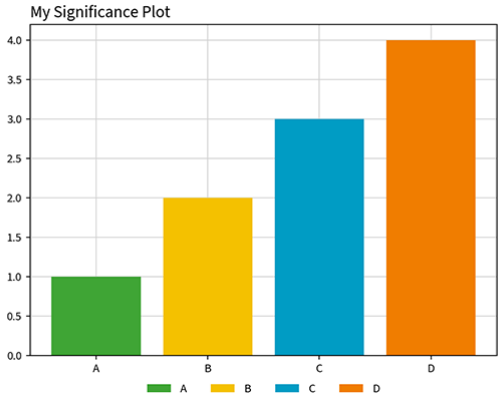

Figure 34: Barchart showing the use of Source Sans font.

Julia

Julia is a general-purpose programming language, focused in high performance, while providing easy syntax. Julia has libraries available that provide capabilities for data analysis and visualisation.

Makie

Makie is a high-level plotting library for the Julia programming language.

Example: changing bar chart colours in Makie.

You can change the colour of chart elements in Makie using the `color` argument and custom tick labels using the `xticks` argument inside `axis`:

```
```{julia}
#| message: false
#| eval: false
using CairoMakie
# generate data
x_vals = [1, 2, 3, 4]
y_vals = [1, 2, 3, 4]
# create barchart
barplot(x_vals, y_vals, color="#009cc4", axis=(; xticks=(1:4, ["A", "B", "C", "D"])))
```
```

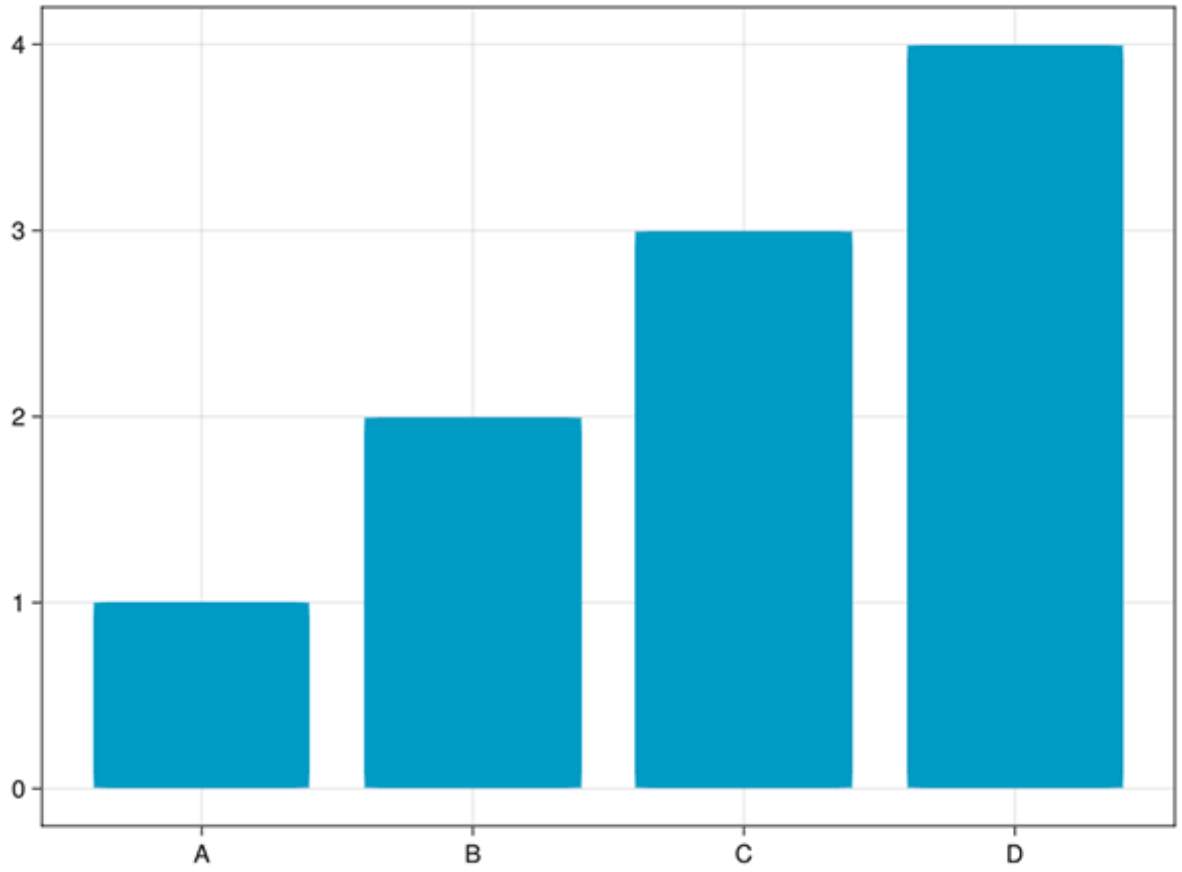

Figure 35: Bar chart with bars coloured in *Significance* blue.

If the colours in your plot are based on values in your data, you can also change the colours used by providing a list of colours:

```
```{julia}
#| message: false
#| eval: false
# define colour palette
signif_qual = ["#3fa535", "#f4c100", "#009cc4", "#f07d00"]
# create barchart
barplot(x_vals, y_vals, color=signif_qual, axis=(; xticks=(1:4, ["A", "B", "C", "D"])))
```
```

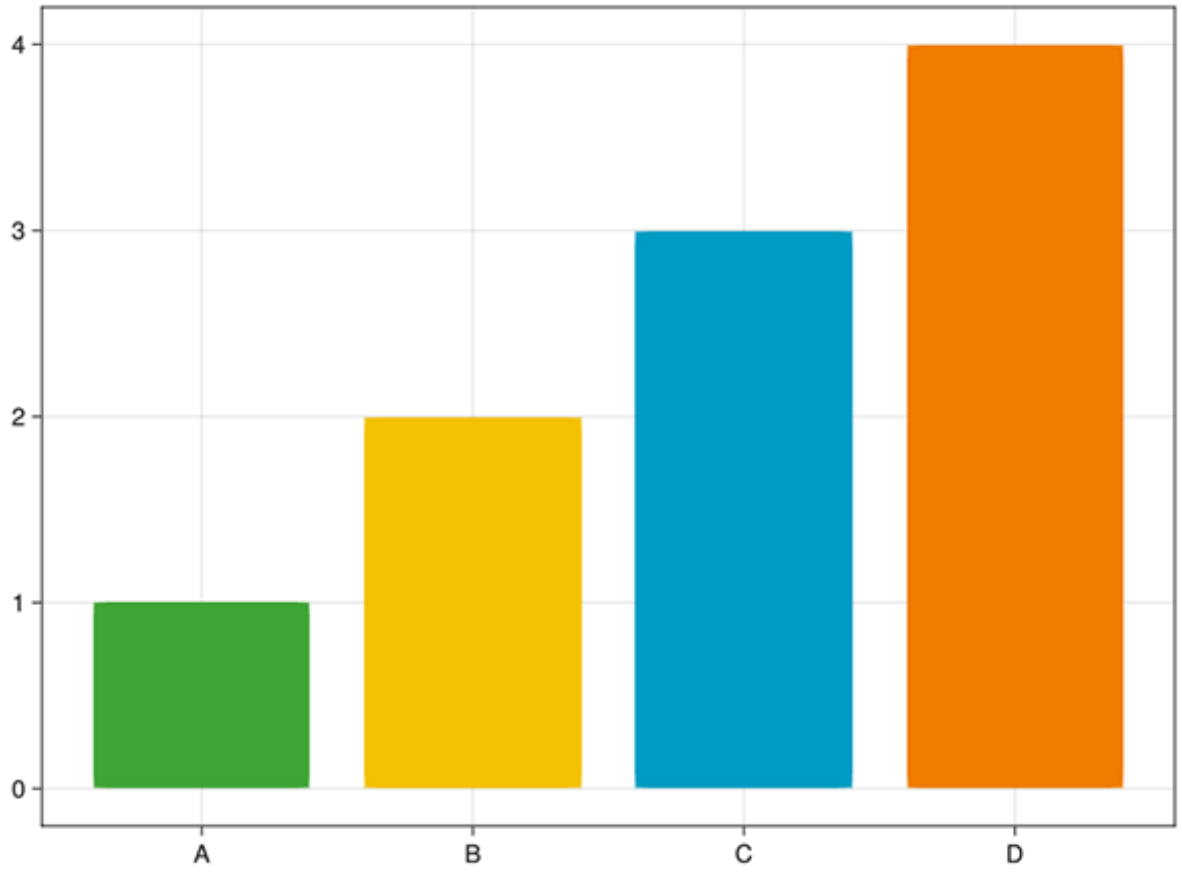

Figure 36: Bar chart showing the colours from the `signif_qual` palette.

You can specify custom labels and titles using the `axis` argument:

```
```{julia}
#| message: false
#| eval: false
# define labels and title
title = "My Significance Plot"
subtitle = "Some longer sentence explaining what is happening in the chart."
xlabel = "X-axis label"
ylabel = "Y-axis label"

# create barchart
barplot(x_vals,
        y_vals;
        color=signif_qual,
        axis=(;
            xticks=(1:4, ["A", "B", "C", "D"]),
            title=title,
            subtitle=subtitle,
            titlealign=:left,
            xlabel=xlabel,
            ylabel=ylabel,
        ),
```
```

```
)
```

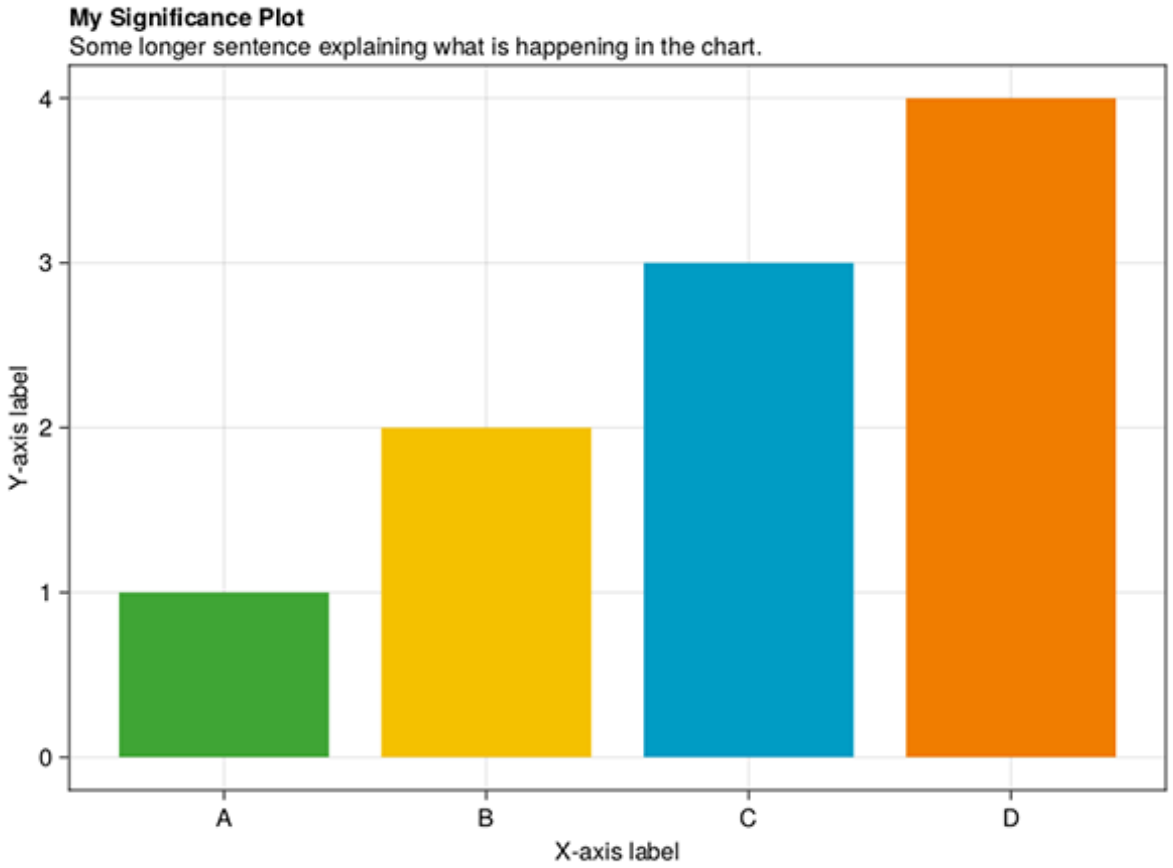

Figure 37: Barchart showing the use of custom labels and titles.

AlgebraOfGraphics

AlgebraOfGraphics is a high-level plotting library for the Julia programming language that uses Makie as a plotting backend. It follows the grammar of graphics and is similar to R's `ggplot2`.

Example: changing bar chart colours in AlgebraOfGraphics.

You can change the colour of chart elements in Makie using the `color` argument and custom tick labels using the `xticks` argument inside `axis`:

```
```{julia}
#| message: false
#| eval: false
using AlgebraOfGraphics
using CairoMakie
# generate data
x_vals = [1, 2, 3, 4]
y_vals = [1, 2, 3, 4]
# create barchart
plt = data((; x_vals, y_vals)) * mapping(:x_vals, :y_vals) * visual(BarPlot; color="#009cc4'
```
```

```
draw(plt; axis=(; xticks=(1:4, ["A", "B", "C", "D"])))
```

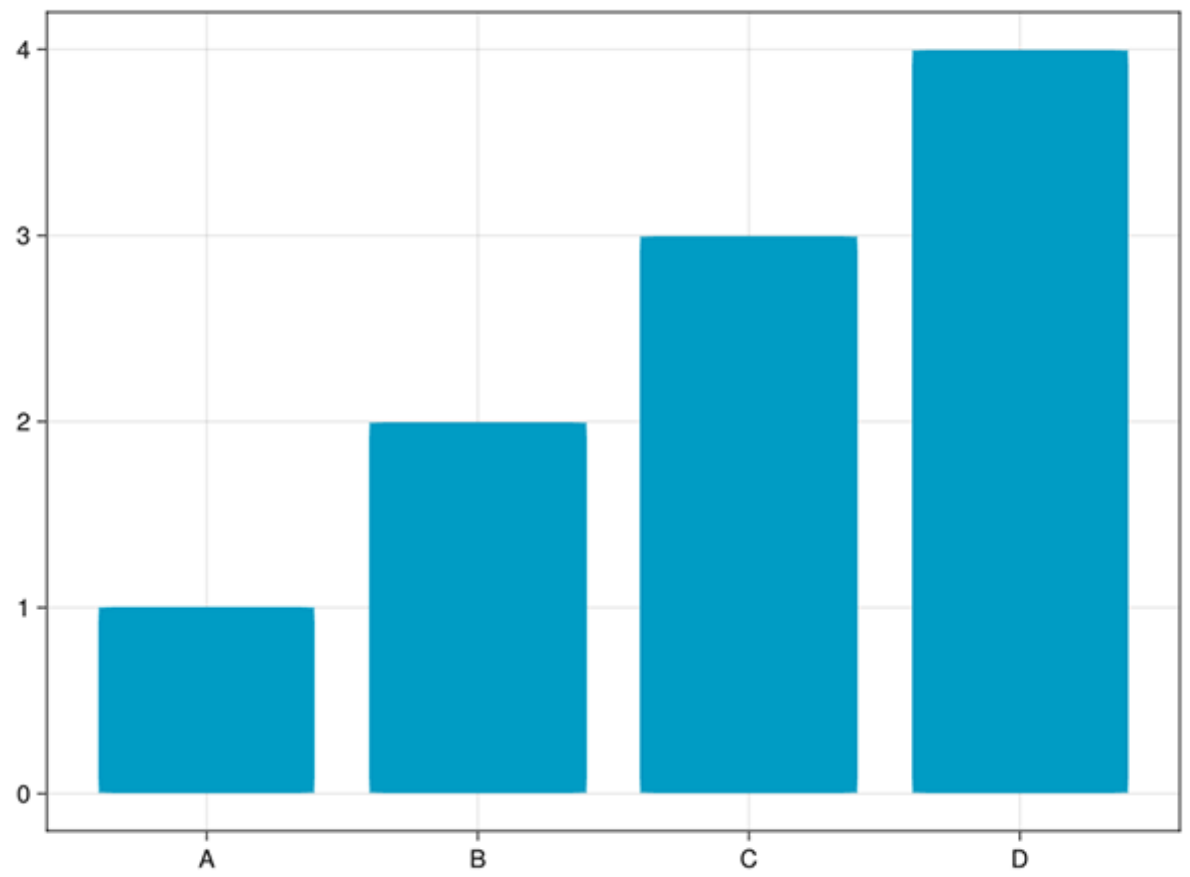

Figure 38: Bar chart with bars coloured in *Significance* blue.

If the colours in your plot are based on values in your data, you can also change the colours used by providing a list of colours:

```
```{julia}
#| message: false
#| eval: false
# define colour palette
signif_qual = ["#3fa535", "#f4c100", "#009cc4", "#f07d00"]
# create barchart
plt = data((; x_vals, y_vals)) * mapping(:x_vals, :y_vals) * visual(BarPlot; color=signif_qu
draw(plt; axis=(; xticks=(1:4, ["A", "B", "C", "D"])))
```
```

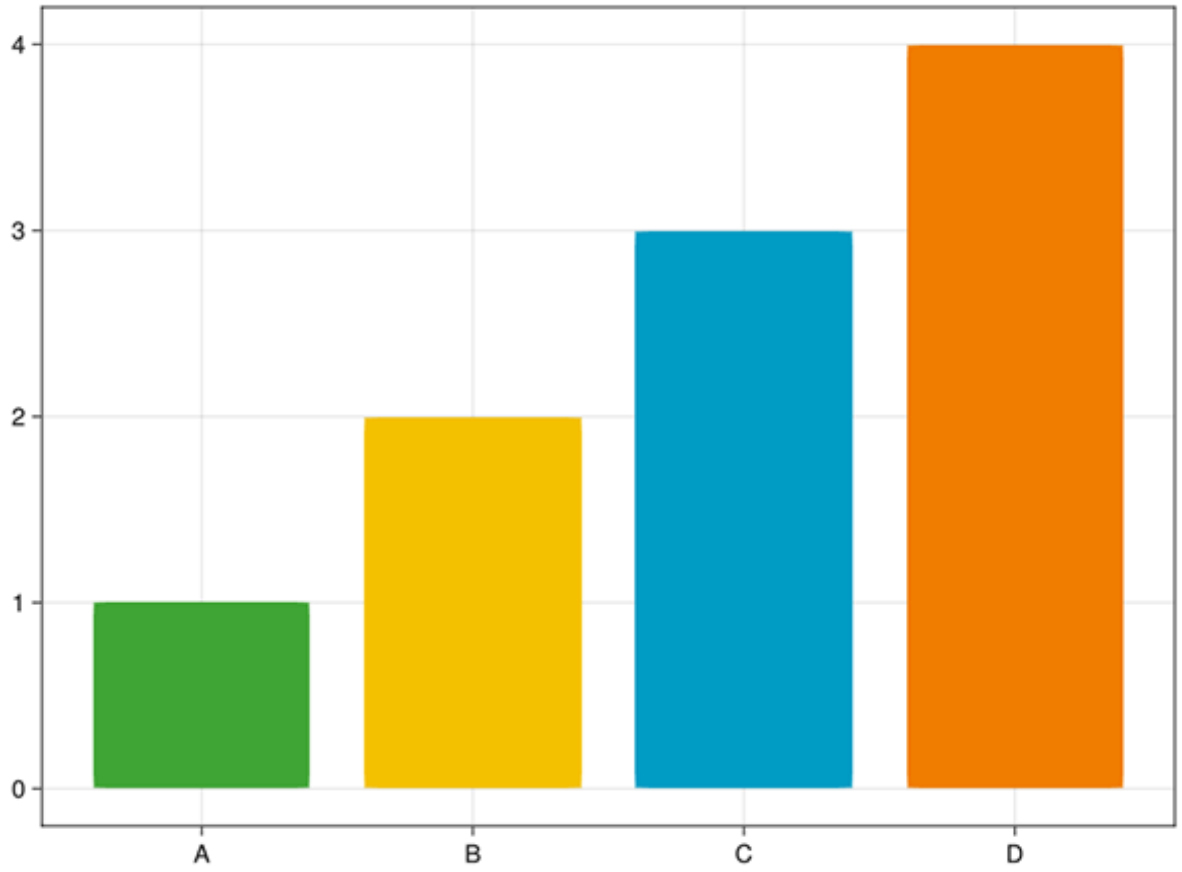

Figure 39: Bar chart showing the colours from the `signif_qual` palette.

You can specify custom labels and titles using the `axis` argument:

```
```{julia}
#| message: false
#| eval: false
# define labels and title
title = "My Significance Plot"
subtitle = "Some longer sentence explaining what is happening in the chart."
xlabel = "X-axis label"
ylabel = "Y-axis label"

# create barchart
draw(plt;
     axis=(;
         xticks=(1:4, ["A", "B", "C", "D"]),
         title=title,
         subtitle=subtitle,
         titlealign=:left,
         xlabel=xlabel,
         ylabel=ylabel,
     ),
)
```
```

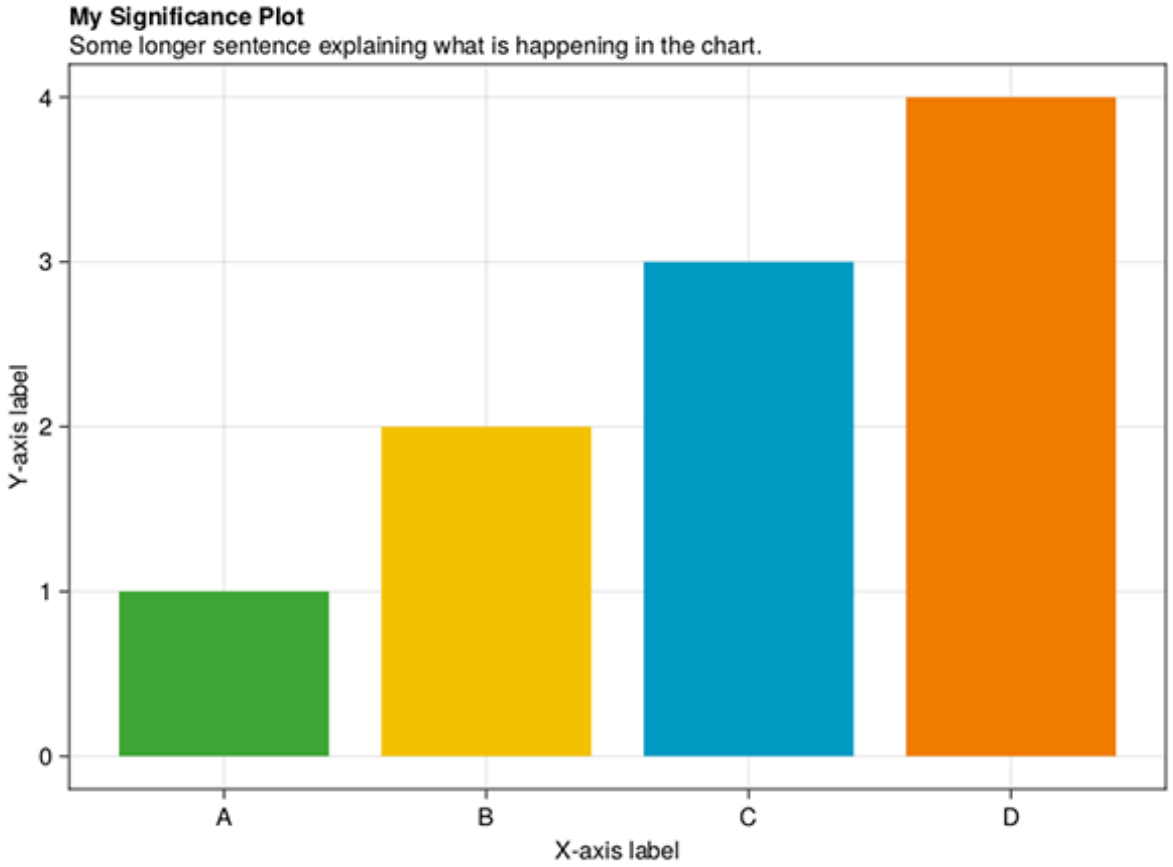

Figure 40: Barchart showing the use of custom labels and titles.

7.2 Publication specifications

The following information should be used to design graphs and charts that meet RSS publication requirements. Details include page sizes and column widths, font types and sizes, and image resolutions and file formats.

Significance Magazine

Page size	(W) 212.55 mm x (H) 263.65 mm
Text area	(W) 188 mm x (H) 212 mm
Image resolution	300 dpi (print quality)
Recommended image file formats	jpeg, png

Notebook section

Uses four-column layout.

1x column width	45 mm
2x column width	93 mm
3x column width	140 mm
4x column width	188 mm

Body font	Meta Serif OT, Book
Font size	8.5 pt
Section colour	Red: (in `RSSthemes` package use `signif_red`) • CMYK = 0, 96, 98, 1 • RGB = 228, 27, 18 • Hex code = #e41b12

Features section

Uses three-column layout.

1x column width	60 mm
2x column width	124 mm
3x column width	188 mm
Body font	Source Sans Pro, Regular
Font size	9 pt
Section colour	Green: (in `RSSthemes` package use `signif_green`) • CMYK = 75, 5, 100, 0 • RGB = 63, 165, 53 • Hex code = #3fa535

Profiles / Perspectives / Statscom section

Uses three-column layout.

1x column width	60 mm
2x column width	124 mm
3x column width	188 mm
Body font	Meta Serif OT, Book
Font size	8.5 pt
Section colours:	

Profiles	Blue: (in `RSSthemes` package use `signif_blue`) • CMYK = 78, 19, 15, 1 • RGB = 0, 156, 196 • Hex code = #009cc4
Perspectives	Yellow: (in `RSSthemes` package use `signif_yellow`) • CMYK = 5, 24, 95, 1 • RGB = 244, 193, 0 • Hex code = #f4c100
Statscomm	Orange: (in `RSSthemes` package use `signif_orange`) • CMYK = 0, 60, 99, 0 • RGB = 240, 125, 0 • Hex code = #f07d00

Journal of the Royal Statistical Society Series A

Uses a single-column layout.

Page size	(W) 189 mm x (H) 246 mm
Text area	(W) 136 mm x (H) 217 mm
Body font	Sabon LT Std Roman
Font size	9.25 pt
Image resolution	300 dpi (print quality)
Recommended image file formats	jpeg, png

8 References

8.1 Texts referenced in the Guide

Battle-Baptiste, W., and B. Rusert. 2018. *W.e.b. Du Bois's Data Portraits: Visualizing Black America: The Color Line at the Turn of the Twentieth Century.* The W.E.B. Du Bois Center at the University of Massachusetts.

Becker, Richard A., and John M. Chambers. 1984. *S: An Interactive Environment for Data Analysis and Graphics.* Pacific Grove, CA: Wadsworth & Brooks/Cole.

Becker, Richard A., and William S. Cleveland. 1996. *S-PLUS Trellis Graphics User's Manual*. Seattle, WA: MathSoft.

Beecham, Roger, Jason Dykes, Layik Hama, and Nik Lomax. 2021. "On the Use of 'Glyphmaps' for Analysing the Scale and Temporal Spread of COVID-19 Reported Cases." *ISPRS International Journal of Geo-Information* 10 (4). https://doi.org/10.3390/ijgi10040213.

Cesal, Amy. 2020. "Writing Alt Text for Data Visualization." Nightingale. 2020. https://medium.com/nightingale/writing-alt-text-for-data-visualization-2a218ef43f81.

"Chart Titles and Text." n.d. Office for National Statistics. Accessed July 10, 2023. https://style.ons.gov.uk/data-visualisation/titles-and-text/annotation-and-footnotes/.

Cleveland, William S. 1993. *Visualizing Data*. Summit, NJ: Hobart Press.

———. 1994. *The Elements of Graphing Data*. Summit, NJ: Hobart Press.

"Coblis — Color Blindness Simulator." n.d. Colblindor. Accessed July 10, 2023. https://www.color-blindness.com/coblis-color-blindness-simulator/.

Corbett, J. 2001. "Charles Joseph Minard, Mapping Napoleon's March, 1861." CSISS Class 2001. 2001. https://escholarship.org/uc/item/4qj8h064.

D'Ignazio, C., and L. F. Klein. 2020. *Data Feminism*. MIT Press. https://mitpress.mit.edu/9780262547185/data-feminism/.

"Documentation." n.d. Statistical Analysis System. Accessed June 19, 2023. https://support.sas.com/en/documentation.html.

Du Bois, W. E. B. 1900. *The Exhibit of American Negroes*. Paris.

Few, Stephen. 2004. *Show Me the Numbers*. Burlingame, CA: Analytics Press.

Friendly, M. 2018. "A Very Brief History of Visualization: Visions, Stories and Pictures." Chicago, IL. 2018. http://datavis.ca/papers/CHF-2x2.pdf.

———. 2022. "Remembrances of Things EDA." 2022. https://www.researchgate.net/publication/361191335_Rememberances_of_Things_EDA.

Friendly, M., and D. Denis. 2005. "The Early Origins and Development of the Scatterplot." *J. Hist. Behav. Sci.* 41 (2): 103–30. https://doi.org/10.1002/jhbs.20078.

Garland, Kevin. 1994. *Mr Beck's Underground Map*. Capital Transport.

Green, Nathan. 2023. "Why your data viz needs alt text." *Significance* 20 (1): 38–39. https://doi.org/10.1093/jrssig/qmad011.

Hedley, Alison. 2020. "Florence Nightingale and Victorian Data Visualisation." *Significance* 17 (2): 26–30. https://doi.org/10.1111/1740-9713.01376.

Kent, AJ. 2021. "When Topology Trumped Topography: Celebrating 90 Years of Beck's Underground Map." *The Cartographic Journal* 58 (1): 1–12. https://doi.org/10.1080/00087041.2021.1953765.

Krause, Andreas. 2013. "Concepts and Principles of Clinical Data Graphics." In *A Picture Is Worth a Thousand Tables: Graphics in Life Sciences*. Springer.

Krause, Andreas, and Michael O'Connell, eds. 2013. *A Picture Is Worth a Thousand Tables: Graphics in Life Sciences*. Springer.

Muth, Lisa. 2018. "An Alternative to Pink & Blue: Colors for Gender Data." Datawrapper. 2018. https://blog.datawrapper.de/gendercolor/.

Nightingale, Florence. 1859. "A Contribution to the Sanitary History of the British Army During the Late War with Russia." London, UK: Harrison; Sons. 1859. https://iiif.lib.harvard.edu/manifests/view/drs:7420433$24b.

Norman, Donald A. 1990. *The Design of Everyday Things*. New York, NY: Currency Doubleday.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rello, Luz, and Ricardo Baeza-Yates. 2016. "The Effect of Font Type on Screen Readability by People with Dyslexia." *ACM Trans. Access. Comput.* 8 (4). https://doi.org/10.1145/2897736.

Robbins, Naomi B. 2006. *Creating More Effective Graphs*. Hoboken, NJ: Wiley.

Robinson, A. H. 1967. "The Thematic Maps of Charles Joseph Minard." *Imago Mundi* 21: 95–108.
Sarkar, Deepan. 2008. *Lattice: Multivariate Data Visualization with r*. New York, NY: Springer.
Schwabish, Jonathan. 2021. *Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks*. Columbia University Press. http://www.jstor.org/stable/10.7312/schw19310.
Setlur, V., and B. Cogley. 2022. *Functional Aesthetics for Data Visualization*. Wiley. https://www.functionalaestheticsbook.com/.
Snow, John. 1854. "Mode of Communication of Cholera." Piccadilly (London), UK: John Churchill. 1854. https://archive.org/details/b28985266/page/52/mode/2up?view=theater.
"Styling Base r Graphics." 2018. Jumping Rivers. 2018. https://www.jumpingrivers.com/blog/styling-base-r-graphics/.
Tennekes, Martijn, and Marco J. H. Puts. 2023. "cols4all: a Color Palette Analysis Tool." In *EuroVis 2023 - Short Papers*, edited by Thomas Hoellt, Wolfgang Aigner, and Bei Wang. The Eurographics Association. https://doi.org/10.2312/evs.20231040.
Tol, Paul. 2021. "Introduction to Colour Schemes." 2021. https://personal.sron.nl/~pault/.
Tufte, Edward R. 1990. *Envisioning Information*. Graphics Press.
———. 2001. *The Visual Display of Quantitative Information*. 2nd ed. Cheshire, CT: Graphics Press.
———. 2004. *Visual Explanations: Images and Quantities, Evidence and Narrative*. Cheshire, CT: Graphics Press.
———. 2006. *Beautiful Evidence*. Cheshire, CT: Graphics Press.
Tukey, John W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison-Wesley.
van Weert, Julia C. M., Monique C. Alblas, Liset van Dijk, and Jesse Jansen. 2021. "Preference for and Understanding of Graphs Presenting Health Risk Information. The Role of Age, Health Literacy, Numeracy and Graph Literacy." *Patient Education and Counseling* 104 (1): 109–17. https://doi.org/10.1016/j.pec.2020.06.031.
Wery, J. J., and J. A. Diliberto. 2017. "The Effect of a Specialized Dyslexia Font, OpenDyslexic, on Reading Rate and Accuracy." *Ann. Of Dyslexia.* 67: 114–27. https://doi.org/10.1007/s11881-016-0127-1.
Wickham, Hadley. 2011. "Ggplot2." *Wiley Interdisciplinary Reviews: Computational Statistics* 3: 180–85. https://onlinelibrary.wiley.com/doi/10.1002/wics.147.
———. 2016a. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. Vol. 2. Use r! Springer International Publishing. https://doi.org/10.1007/978-3-319-24277-4.
———. 2016b. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.
Wikipedia contributors. 2023. "1854 Broad Street Cholera Outbreak." 2023. https://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak.
Wilkinson, Leland. 2005. *The Grammar of Graphics*. Statistics and Computing. New York: Springer-Verlag. https://doi.org/10.1007/0-387-28695-0.
Williams, T. A., David R. Anderson, and Dennis J. Sweeney. 2023. "Statistics." *Encyclopedia Britannica*. https://www.britannica.com/science/statistics.

8.2 Further reading

Functional Aesthetics for Data Visualization.

Setlur and Cogley (2022)

Why we recommend: *This book connects research and practice, combining ideas from cognitive psychology, data literacy, and visual design. Its ideas go beyond simple charts, and can be extending to complex visual displays including interactive dashboards and visualisations.*

Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks.

Schwabish (2021)

Why we recommend: *This book guides you towards which types of data visualisations may best depending on what you're trying to show. Not only that, but it uses visual perception theory to explain why some charts are easier to read than others.*

Data Feminism.

D'Ignazio and Klein (2020)

Why we recommend: *We often hear the term "storytelling" when we're talking about data visualisation, and we should remember the role of the storyteller in this. In this book, the authors make several critical points, including that when we work with data we must also think about why it was collected, and what it could be used for.*

Preference for and understanding of graphs presenting health risk information. The role of age, health literacy, numeracy and graph literacy.

van Weert et al. (2021)

Why we recommend: *Patients are often presented with data about health risk information in the form in charts and tables. This article examines how well patients understand the information presented to them across a range of different chart types. It finds that the types of charts that patients prefer, is not always the type of chart that they understand best.*

8.3 Additional resources

Training courses

Presenting Data (RSS)

This online course is the foundation to all presentations of statistical information. The basic principles of presenting information in tables, charts, maps and text are explained. These are illustrated and then reinforced through practical exercises.

Data Exploration in Tableau (RSS)

Tableau is more than just a simple data visualisation tool. It also gives people the capability to manipulate multiple data sources, create custom charts, build predictive models, and turn their plots into interactive dashboards and presentations.

This virtual course will run over two afternoons. It will guide you through the process of wrangling data, performing data analysis, and visually communicating outputs. By the end of this course you will be able to manipulate your own data to build custom charts. You will learn how to work with geographic data in Tableau, use predictive models to make forecasts, and create interactive dashboards and stories to share your work.

Power BI for Data Exploration and Basic Statistics (RSS)

Power BI is rapidly becoming a standard tool for producing interactive reports and dashboards and this course will provide a systematic introduction to Power BI, focusing on the workflow, from data preparation through to building and sharing visuals. There will be a focus on design issues, helping to achieve the right level of functionality and usability based on the intended audience. Particular attention will also be paid to the use of basic statistics.

Publishing Quality Charts in R with ggplot2 (RSS)

This tutor-led virtual course will introduce how the tidyverse and ggplot2 can be used to reproducibly create publication quality charts from R.

Low-code/no-code visualisation tools

Datawrapper

https://www.datawrapper.de/

Flourish

https://flourish.studio/

9 About the authors

9.1 Andreas Krause

Andreas Krause holds a PhD and an MSc in statistics and computer science. His current position is senior director, data science, modeling and simulation, at Idorsia Pharmaceuticals Ltd in Basel, Switzerland. His group provides clinical drug development teams with statistics and model-based analysis and predictions to support decision making. His works include more than 100 peer-reviewed scientific publications.

Andreas' scientific activities include positions as co-editor of the American Statistical Association's section on Statistical Computing and Graphics, book review editor and board member of *Computational Statistics*, associate editor and editor-in-chief of *Pharmaceutical Statistics*, and, currently, advisory board member of *Pharmaceutical Statistics* and the *Journal of Pharmacokinetics and Pharmacodynamics*.

Andreas started using the S language in 1987, became European S-PLUS distributor and supporter in 1988, and started using R in the 1990s. He is author of *The Basics of S and S-Plus* (Springer, 4 editions, more than 20,000 copies sold) and editor of *A picture is worth a thousand tables: graphics in life sciences* (Springer, 2012).

Graphics and visualisation are long-standing topics for him, including numerous presentations and workshops on "Graphics of clinical data: the good, the bad, and the ugly" at various conferences and seminars.

- LinkedIn: linkedin.com/in/andreaskrause
- ORCID: orcid.org/0000-0002-4686-976X

9.2 Nicola Rennie

Nicola Rennie is an academic, data scientist, and educator with a passion for effectively communicating data. She is a Lecturer in Health Data Science at Lancaster Medical School, where her research focuses on the use of statistical models to improve health outcomes. Her teaching experience covers topics including data visualisation, programming in R and Python, and how to effectively communicate the results of statistical analyses. Nicola is the author and maintainer of multiple R packages, and an active member of the R community – regularly presenting at R User Group Meetups and mentoring new members of the community.

- Website: nrennie.rbind.io
- Twitter: @nrennie35
- LinkedIn: linkedin.com/in/nicola-rennie
- Mastodon: @fosstodon.org/@nrennie
- ORCID: orcid.org/0000-0003-4797-557X

9.3 Brian Tarran

Brian Tarran is a writer and editor with 20 years of experience covering the research and data space. He has worked for the Royal Statistical Society (RSS) for the past 8 years, and was editor of *Significance* Magazine (a joint publication of the RSS, the American Statistical Association and the Statistical Society of Australia) prior to the launch of Real World Data Science. Brian is a former editor of Research-Live.com and was launch editor of Impact magazine, both published by the Market Research Society.

- Twitter: @brtarran
- LinkedIn: linkedin.com/in/brian-tarran-58b0261
- Mastodon: @mastodon.social/@brtarran
- GitHub: github.com/brtarran

9.4 Contributors

- Jose Storopoli added Julia examples to Styling for RSS publications.

9.5 Acknowledgements

Thanks to Kelly Zou, Marco Geraci, Jordi Prats-Rodriguez, Marc Vandemeulebroecke, and Peter Bonate for comments on an early version of this guide. Thanks also to:

- Philip Shirk for styling fixes.
- Brent Riechelman for bug hunting.
- MMJansen for spotting an error.

10 Terms and Conditions

10.1 Legal disclaimer

Statements of fact and opinion published on this website are those of the respective authors and contributors and not necessarily those of the Royal Statistical Society (RSS).

The authors have prepared the content of this website responsibly and carefully. However, the authors and the RSS disclaim all warranties, express or implied, as to the accuracy of the information contained in any of the materials on this website or on other linked websites or on any subsequent links. This includes, but is not by way of limitation:

- any implied warranties of merchantability and fitness for a particular purpose.
- any liability for damage to your computer hardware, data, information, materials and business resulting from the information or the lack of information available.
- any errors, omissions, or inaccuracies in the information.
- any decision made or action taken or not taken in reliance upon the information.

The authors and the RSS make no warranty as to the content, accuracy, timeliness or completeness of the information or that the information may be relied upon for any reason and bear no responsibility for the accuracy, content or legality of any linked site or for that of any subsequent links. The authors and the RSS make no warranty that the website service will be uninterrupted or error-free or that any defects can be corrected.

The authors and the RSS shall not be liable for any losses or damages (including without limitation consequential loss or damage) whatsoever from the use of, or reliance on, the information in this website, or from the use of the internet generally. Links to other websites or the publication of advertisements do not constitute an endorsement or an approval by the authors and the RSS.

These disclaimers and exclusions shall be governed by and construed in accordance with the laws of England and Wales under the exclusive jurisdiction of the courts of England and Wales. Those who choose to access this site from outside the United Kingdom are responsible for compliance with local laws if and to the extent local laws are applicable.

By using this site, you agree to these terms and conditions of use.

10.2 Site content

10.3 What websites do we link to?

Authors and contributors recommend external web links on the basis of their suitability and usefulness for our users. Selection and addition of links to our website is entirely a matter for the authors and the authors alone.

It is not our policy to enter into agreements for reciprocal links.

The inclusion of a link to an organisation's or individual's website does not constitute an endorsement or an approval by the authors and the RSS of any product, service, policy or opinion of the organisation or individual. The authors and the RSS are not responsible for the content of external websites.

10.4 What websites will we not link to?

We will not link to websites that contain racist, sexual or misleading content; that promote violence; that are in breach of any UK law; which are otherwise offensive to individuals or to groups of people.

The decision of the authors is final and no correspondence will be entered into.

If you wish to report a concern, please email b.tarran@rss.org.uk

10.5 Software and services

Source code and files for this site are available from GitHub. Use of our GitHub repository is governed by the Contributor Covenant Code of Conduct.

This site is built using Quarto, an open-source scientific and technical publishing system developed by Posit. Quarto source code and software licences are available from GitHub.

This site is hosted by GitHub Pages.

This site uses Google Analytics 4 for web analytics reporting.

10.6 Notice and Takedown policy

If you are a rights holder and are concerned that you have found material on our site for which you have not given permission, or is not covered by a limitation or exception in national law, please contact us in writing stating the following:

1. Your contact details.
2. The full bibliographic details of the material.
3. The exact and full url where you found the material.
4. Proof that you are the rights holder and a statement that, under penalty of perjury, you are the rights holder or are an authorised representative.

Contact details:
Notice and Takedown,
Licensing,
12 Errol Street,
London EC1Y 8LX
web@rss.org.uk

Upon receipt of notification, the 'Notice and Takedown' procedure is then invoked as follows:

1. We will acknowledge receipt of your complaint by email or letter and will make an initial assessment of the validity and plausibility of the complaint.
2. Upon receipt of a valid complaint the material will be temporarily removed from our website pending an agreed solution.
3. We will contact the contributor who deposited the material, if relevant. The contributor will be notified that the material is subject to a complaint, under what allegations, and will be encouraged to assuage the complaints concerned.
4. The complainant and the contributor will be encouraged to resolve the issue swiftly and amicably and to the satisfaction of both parties, with the following possible outcomes:

    - The material is replaced on our website unchanged.
    - The material is replaced on our website with changes.
    - The material is permanently removed from our website.

If the contributor and the complainant are unable to agree a solution, the material will remain unavailable through the website until a time when a resolution has been reached.

10.7 Contributor Covenant Code of Conduct

Our pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, caste, color, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

Our standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Enforcement responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

Scope

This Code of Conduct applies within all community spaces (encompassing this site and our GitHub repository). It also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

Note that unless prior permission is agreed in writing with the RSS, only the lead authors (Andreas Krause, Nicola Rennie, and Brian Tarran) may officially represent the community. Comment to the media must only be given by appointed representatives and must be approved by the RSS press office.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at b.tarran@rss.org.uk. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

Enforcement guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.1, available at https://www.contributor-covenant.org/version/2/1/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.