Notes on Tufte's principles and visualisation

Sam Clifford

2021-06-16

Introduction

Here we provide a little more detail on Tufte's principles (Tufte 1983, 1997) and the use of ggplot2 to create effective visualisations.

For further guidance on the use of ggplot2 in R, there are online versions of Chang (2017) and Wickham and Grolemund (2020) (specifically chapter 3) as well as the RStudio cheatsheets (RStudio 2012). The BBC has used ggplot2 to develop an in-house style of data visualisation for their data journalism (BBC data team 2019).

Tufte's principles for visualising data

A quick tour

- Visual representations of data must tell the truth
- Good graphical representations maximize data-ink and erase as much non-data-ink as possible
- Avoid chartjunk: the excessive and unnecessary use of graphical effects in graphs
- Produce high data density graphs

Examples of visualisation for critique

The following visualisations are found in Tufte (1983) and we discuss them here with a brief critique each.

Bar plot

In Figure 1, we see an example of a bar plot with additional chart junk. Specifically, we have: a 3D effect; too many additional labels, such as the total budget for each year; and arrows for labelling the estimate for the current year and recommendation for the next year.

By removing the elements in the left hand panel of Figure 2, we retain the key information and are more easily able to make comparisons without being distracted by 3D effects.

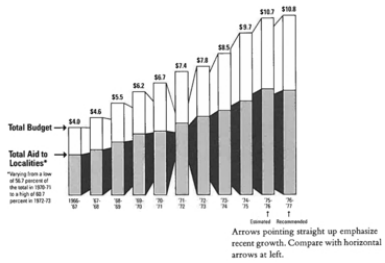


Figure 1: Total aid to localities as a fraction of total budget, p83 Tufte (1983)

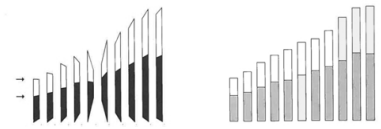


Figure 2: Elements of bar plot

Line plot

In Figure 3 we see thermal conductivity as a function of temperature. Each experiment is a point, and data from the same lab are joined with a line. The curve varies across labs due to impurities in copper, and this grouping enables comparisons across labs *and* against the overall behaviour. The scales are logarithmic on both the x axis (temperature in Kelvin) and y axis (Watts per metre-Kelvin). While this may have been an effective plot when it was created, it is difficult to tell which lab a given data point and curve comes from; today this might be drawn as an interactive plot.

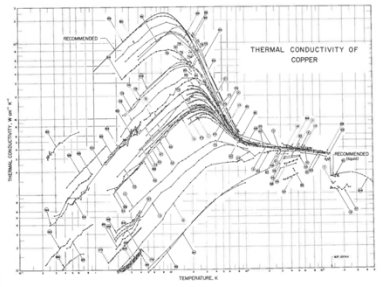


Figure 3: Thermal conductivity of copper across laboratories, p49 Tufte (1983) citing Ho et al. (1974).

Ribbon plot

The coloured ribbon plot shows the age split of college enrolments between 1972-1976, for under-25s and 25 and over. This is called a ribbon plot because the plot is a ribbon, a shaded area, between some minimum and maximum lines.

One issue with this ribbon plot is that the two ribbons sum to 100% but not in an obvious way, as the ribbons are separate rather than stacked and a doubly broken axis is used without showing the zero point or the top of the axis. Four colours are used, to show each age group and to give a visually distracting 3D effect. No colouring is necessary given that only one set of numbers can be used to represent these mutually exclusive and completely exhaustive age groups. That is, this could be a line plot or even a table, e.g.

Year	1972	1973	1974	1975	1976
Under 25	72.0%	70.8%	67.2%	66.4%	67.0%

N.B. Guidance on the layout of tables is available in Mori (2007) and Fear (2005).

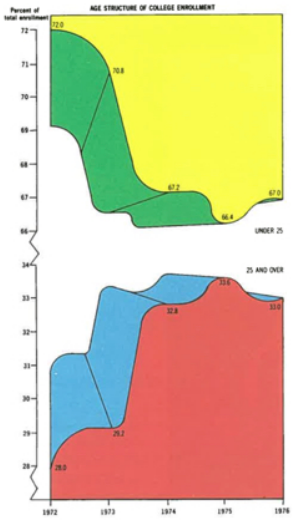


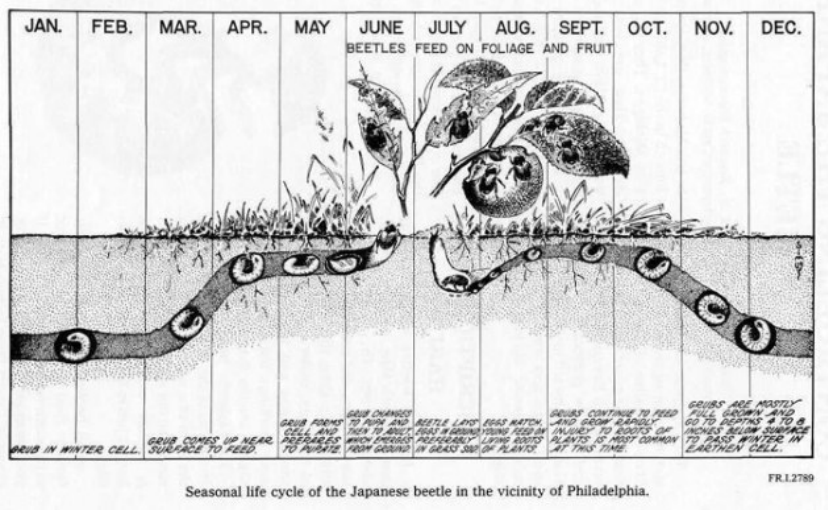
Figure 4: Proportion of college enrolments under or over 25 years old. p43 Tufte (1983)

Artist collaboration

The diagram of the life cycle of the Japanese beetle (Figure 5) is rich with information and visually appealing - a highly impactful graph. The result of a collaboration with an artist, we see the depth of the grub, the life stage of grubs and larvae over time and which foliage is available as a food source. As this is an annual cycle, the graphs wraps around at each end and annotations provide additional detail which may not be easily captured graphically but in such a way that they do not hijack the viewer's attention.

Graphical integrity and ggplot2 code

Visual representations of data must tell the truth



Seasonal life cycle of the Japanese beetle in the vicinity of Philadelphia.

Figure 5: Life cycle of Japanes beetle, p43 Tufte (1983)

1. The visual representation of numbers, as physically measured on the graph, should be directly proportional to the numerical quantities represented
    - Lie Factor = the size of the effect shown in the graphic divided by the size of the effect in the data.
    - Values over 1 overstate the effect & under 1 understate.
    - Don't make points bigger to emphasise them as important
2. Clear, detailed and thorough labelling helps avoid graphical distortion and ambiguity. Write explanations of data on the graph. Label important events from data.
3. Show variation through data, not through design.
4. Ensure appropriate standardization & comparisons are used, e.g. CPI-adjusted or seasonally-adjusted.
5. The number of information carrying (variable) dimensions depicted should not exceed the number of dimensions in the data.
    - In practice, this means not mapping the same variable to two graphical elements (e.g. Figure 7)
6. Graphics must not quote data out of context.

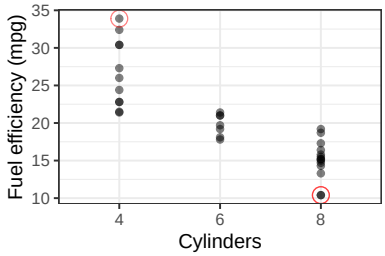


Figure 6: Improper emphasis of most and least fuel efficient cars. We only need look at vertical range to see these are the min and max

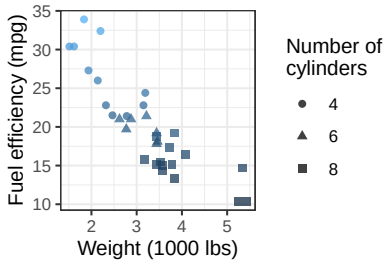


Figure 7: Fuel efficiency mapped to colour and y coordinate. Gradated pattern of colour implies there's another relationship with fuel efficiency being looked at.

Maximising `data:ink` ratio

Good graphical representations maximize data-ink and erase as much non-data-ink as possible

1. Above all else, show data
2. Maximise the data-ink ratio
    - ink on a graph that represents data
    - data-ink ratio = 1 minus the proportion of the graph that can be erased without loss of data-information
3. Erase non-data-ink
4. Erase redundant data-ink
5. Revise and edit

For example, the boxplot is a simple five number summary of a continuous variable that draws multiple rectangles, lines and points to show the quantiles of the data and any outliers (Frigge, Hoaglin, and Iglewicz 1989). Tufte considers that the outliers are not important in showing the typical spread of the data and that the box is unnecessary as the quantiles are shown with the 'whiskers' of the plot and the median line. A standard boxplot from ggplot2 is shown in Figure 8 and a version that removes extraneous lines and points in Figure 9.

A lot of the elements of the boxplot can be removed without reducing our ability to distinguish the relevant information.

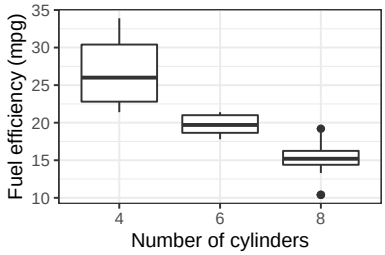


Figure 8: Standard boxplot

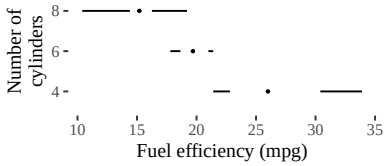


Figure 9: Tufte's interpretation of boxplot that maximises data ink

Avoid Chartjunk

If an element of a graph is to be included, it should be because it aids understanding and reveals information, not because it looks pretty

- Examples of chartjunk include but are not restricted to:
    - hatching (patterns instead of colours)
    - heavy grids
    - equally spaced lines, too close together, as in bar charts, histograms, boxplots
    - self-promoting graphs that demonstrate the graphical ability of the designer rather than displaying the data
    - 3D graphics that distort perspective (e.g. 3D pie charts)

Maximising data density

We must ensure that when we are plotting we don't end up with large amounts of white space when avoidable. Small multiples can be attractive, but leaving empty facets leads to unnecessary amounts of white space that reduce the amount of space in the figure dedicated to showing the data in existing combinations of faceting variables.

1. Maximise data density and the amount of data shown, within reason
2. Apply the shrink principle

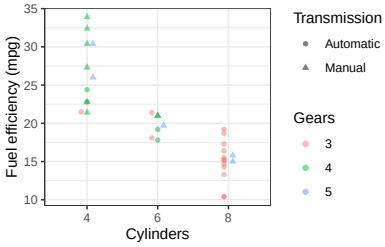


Figure 10: High dimensional data set with points coloured by number of forward gears and shape mapped to transmission type

- most graphs can be shrunk down very far without losing legibility or information
- this is because most graphs are quite sparse

3. Exploit **small multiples** to provide for comparisons across groups
   - series of the same small graph repeated in one visual
   - can compare a main relationship across one or more grouping variables
   - a great way to visualise large quantities of data, or when there are a high number of dimensions

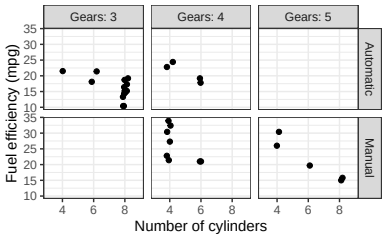


Figure 11: High dimensional data set with small multiples faceting by number of forward gears and transmission type. Two of the six facets are empty, indicating that the graph should be redesigned.

Aesthetics in ggplot2

Inheriting aesthetics

In the lecture we show how the following code takes the variables `my.x.variable` and `my.y.variable` from the data frame `my.data.frame`, maps them to the x and y coordinates on a set of axes and then draws the pairs of x and y as points.

```
ggplot(data = my.data.frame,
       aes(x = my.x.variable,
           y = my.y.variable)) +
  geom_point()
```

In addition to default settings about how to draw points, the `geom_point()` call has inherited the `x` and `y` aesthetics from the `ggplot()` call that forms the basis of the plot. We could have also written this plot as:

```
ggplot(data = my.data.frame) +
  geom_point(aes(x = my.x.variable,
                 y = my.y.variable))
```

or even

```
ggplot() +
  geom_point(data = my.data.frame,
             aes(x = my.x.variable,
                 y = my.y.variable))
```

and we get the exact same plot in each case.

However, as we add more and more geometries to our plot it becomes tiresome, and easy to make a mistake, to continue putting the exact same aesthetics in the call for every single geometry. In practice, it's easiest to think about which aesthetics will be common to all (or nearly all) geometries in a plot and put those in the initial `ggplot()` call. Within the geometry we can add additional aesthetic options, e.g.

```
ggplot(data = my.data.frame,
       aes(x = my.x.variable,
           y = my.y.variable)) +
  geom_point(aes(color = my.color.variable))
```

or overwrite what is being inherited with something else

```
ggplot(data = my.data.frame,
       aes(x = my.x.variable,
           y = my.y.variable)) +
  geom_point() +
  geom_line(aes(y = my.prediction.from.a.linear.model))
```

Other plotting aesthetics

We can pass many other arguments in `ggplot2` to change things about the geometries we use to show the x and y variables. If we put these aesthetics outside the `aes()` statement, this will make this option constant for the entire geometry, e.g. `geom_point(color = 'red')` will make all of these points red. Alternatively, we can map a variable in the data frame to one of the following aesthetics:

- **group** to repeat geometry for a grouping variable, an alternative to calling `lines()` in base plotting within a loop that subsets the data frame
- **size** of point or thickness of boundary
- **shape** of points
- **colour** of boundary (for most point shapes, the entire point is the boundary)
- **alpha** transparency
- **fill** colour of internals of geometry

Below we look at some examples of changing one or more of these.

Group

Instead of splitting all data up with small multiples, we could use grouping to show each each group on a common set of axes. This is especially useful when we have many, many groups in the data set such as the 261 participants in the spinal bone density data set in Figure 12.

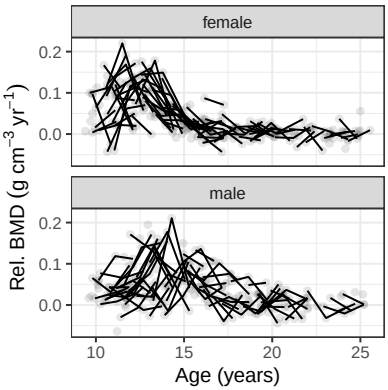


Figure 12: Rate of change of spinal bone mineral density

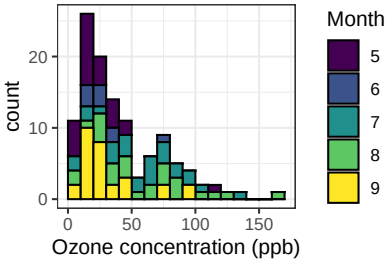


Figure 13: Histogram bars stacked and coloured by month

Colour and fill

We can set the colour common to all instances of the geometry by setting it constant, outside the `aes()` statement and then set the fill to vary according to some variable, e.g. Figure 13 shows how common each range of ozone concentration is (bins of width 10ppb) and which months those observations were measured in. For advice on the use of colour, see Neuwirth (2014), Stone (2006) and Stone (2006).

Size and shape

We must be careful when using shape that we do not choose to use a variable that has so many levels that it becomes too taxing for the reader to tell which is which. Thankfully, ggplot2 gives us some warning when try to do something stupid and use too many.

Alpha transparency

Alpha refers to the transparency (1 = solid, 0 = fully transparent). This is useful when you've got lots of things stacked on top of each other in a plot and want to make it clear how many of them there are, or you're using an overlay.

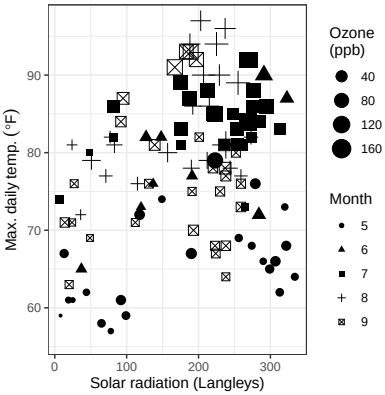


Figure 14: Inappropriate use of plot marker shape to show how relationship between solar radiation and Ozone concentration varies by month.

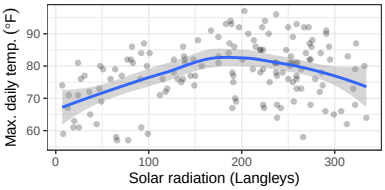


Figure 15: Use of alpha transparency can help reduce overplotting when many points/lines/polygons overlap

Changing the default options

Many `scale_*` functions allow us to set options for the relevant aesthetic and corresponding legend name, e.g.

- `scale_color_gradient()` makes a color gradient for when we use `aes(color=...)`
- `scale_fill_brewer()` sets a color palette for `aes(fill=...)` using colour schemes at `http://colorbrewer2.org/`
- `scale_x_log10()` changes the x axis to have a logarithmic scale in increasing powers of 10.
- Find more at the `ggplot2 documentation page`

Behind the scenes

All geometries provide visual summaries of the data. Sometimes it's a direct plot of the data, such as points and line plots; other times there is a calculation of summary statistics happening behind the scenes.

For example: a bar plot counts the number of times each level of the categorical variable occurs and then draws uses `geom_col()` to draw the result; a histogram bins the continuous variable values, counts how many observations are in each bin and then draws the columns with no gap between them; a density plot performs a kernel density estimation with automatic bandwidth selection and then draws a ribbon between the estimated density and x axis.

The boxplot is arguably one of the more complex examples of this as it performs a five number summary and then draws a series of polygons and line segments between the quantiles and then draws the points as outliers.

You can define your own geometries and summary statistic functions (Wickham et al. 2019), but it's too complex to go into here. For the most part, the exploratory data analysis and model visualisation you'll be doing is able to be performed using the built-in geometries.

References

BBC data team. 2019. "BBC Visual and Data Journalism cookbook for R graphics." BBC. https://bbc.github.io/rcookbook/.

Chang, Winston. 2017. *R Graphics Cookbook: Practical Recipes for Visualizing Data*. 2nd ed. O'Reilly Media. http://www.cookbook-r.com/Graphs/.

Fear, Simon. 2005. "Publication Quality Tables in LaTeX."

Frigge, Michael, David C. Hoaglin, and Boris Iglewicz. 1989. "Some Implementations of the Boxplot." *The American Statistician* 43 (1). JSTOR:50. https://doi.org/10.2307/2685173.

Ho, C. Y., R. W. Powell, and P. E. Liley. 1974. "Thermal Conductivity of the Elements: A Comprehensive Review." *Journal of Physical and Chemical Reference Data, Supplement 1* 3 (December):1–244.

Mori, Lapo Filippo. 2007. "Tables in LaTeX2ε: Packages and Methods." *The PracTeX Journal* 1:2007–1.

Neuwirth, Erich. 2014. *RColorBrewer: ColorBrewer Palettes*. https://CRAN.R-project.org/package=RColorBrewer.

RStudio. 2012. "RStudio Cheat Sheets." 2012. https://www.rstudio.com/resources/cheatsheets/.

Stone, Maureen. 2006. "Choosing Colors for Data Visualization." *Business Intelligence Network* 2. http://www.perceptualedge.com/articles/b-eye/choosing_colors.pdf.

Tufte, Edward R. 1983. *The Visual Display of Quantitative Information*. Graphics Press.

———. 1997. "Visual and Statistical Thinking." In *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press. https://www.edwardtufte.com/tufte/books_textb.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, and Hiroaki Yutani. 2019. "Extending ggplot2." https://ggplot2.tidyverse.org/articles/extending-ggplot2.html.

Wickham, Hadley, and Garrett Grolemund. 2020. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. http://r4ds.had.co.nz.