General Principles of Data Visualization

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Rutgers University – New Jersey Medical School
w.evan.johnson@rutgers.edu

2025-02-16

Introduction to Data Visualization

We will discuss some general principles we can use as a guide for effective data visualization.

Credit: These slides rely on a talk titled "Creating Effective Figures and Tables"[1] and includes some of his figures available on his GitHub repository[2]. We also use notes from Peter Aldhous' Introduction to Data Visualization course[3].

[1] https://www.biostat.wisc.edu/~kbroman/presentations/graphs2017.pdf
[2] https://github.com/kbroman/Talk_Graphs
[3] http://paldhous.github.io/ucb/2016/dataviz/index.html

Introduction to Data Visualization

We will discuss examples of plot styles we should avoid, explain how to improve them, and use these as motivation for a list of principles.

The principles are mostly based on research related to how humans detect patterns and make visual comparisons–we prefer approaches that best fit the way our brains process visual information.

Customizing to Data Visualization

When deciding on a visualization approach, keep your goal in mind. For example, comparing a viewable number of quantities, describing distributions, comparing the data from two groups, or describing the relationship between two variables.

For a data scientist, it is important to adapt and optimize graphs to the audience. An exploratory plot made for ourselves will be different than a chart intended to communicate a finding to a general audience.

Encoding data using visual cues

We start by describing some principles for encoding data. There are several approaches at our disposal including position, aligned lengths, angles, area, brightness, and color hue.

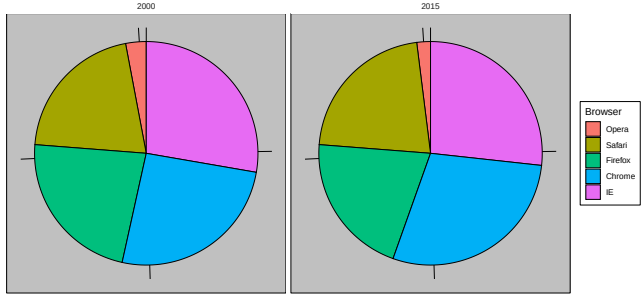
Consider the following data from two hypothetical polls regarding browser preference in 2000 and then 2015:

```
browsers <- data.frame(Browser = rep(c("Opera","Safari","Firefox","IE","Chrome"),2),
                       Year = rep(c(2000, 2015), each = 5),
                       Percentage = c(3,21,23,28,26, 2,22,21,27,29)) %>%
  mutate(Browser = reorder(Browser, Percentage))
browsers
```

```
##    Browser Year Percentage
## 1    Opera 2000          3
## 2   Safari 2000         21
## 3  Firefox 2000         23
## 4       IE 2000         28
## 5   Chrome 2000         26
## 6    Opera 2015          2
## 7   Safari 2015         22
## 8  Firefox 2015         21
## 9       IE 2015         27
```

Encoding data using visual cues

The ‘browsers’ data reports results from two hypothetical polls regarding browser preference in 2000 and then 2015.A widely used graphical approach, popularized by Microsoft Excel, is the pie chart:

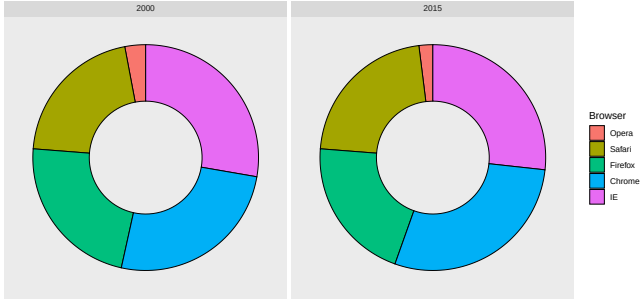


Encoding data using visual cues

In a pie chart we are representing quantities with both areas and angles, since both the angle and area of each pie slice are proportional to the quantity the slice represents. This turns out to be a sub-optimal choice since, as demonstrated by perception studies, humans are not good at precisely quantifying angles and are even worse when area is the only available visual cue.

Encoding data using visual cues

The donut chart is an example of a plot that uses only area:



Encoding data using visual cues

To see how hard it is to quantify angles and area, note that the rankings and all the percentages in the plots above changed from 2000 to 2015. Can you determine the actual percentages and rank the browsers' popularity? Can you see how the percentages changed from 2000 to 2015? It is not easy to tell from the plot. In fact, the `pie` R function help file states that:

"*Pie charts are a very bad way of displaying information. The eye is good at judging linear measures and bad at judging relative areas. A bar chart or dot chart is a preferable way of displaying this type of data.*"

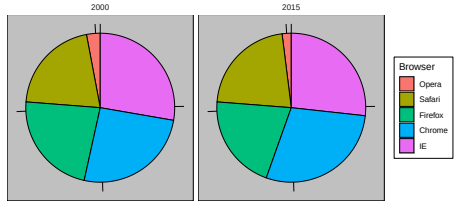
Encoding data using visual cues

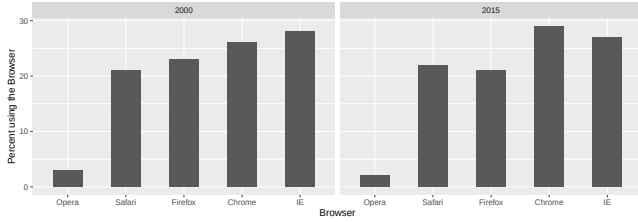
In this case, simply showing the numbers is not only clearer, but would also save on printing costs if printing a paper copy:

Browser	2000	2015
Opera	3	2
Safari	21	22
Firefox	23	21
Chrome	26	29
IE	28	27

Encoding data using visual cues

Compare and contrast the information we can extract from the two figures:





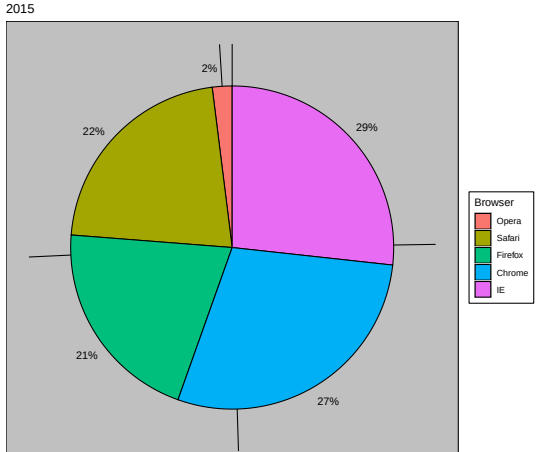
Encoding data using visual cues

The barplot uses this approach by using bars of length proportional to the quantities of interest. By adding horizontal lines at strategically chosen values, in this case at every multiple of 10, we ease the visual burden of quantifying through the position of the top of the bars.

Notice how much easier it is to see the differences in the barplot. In fact, we can now determine the actual percentages by following a horizontal line to the x-axis.

Encoding data using visual cues

If for some reason you need to make a pie chart, label each pie slice with its percentage:

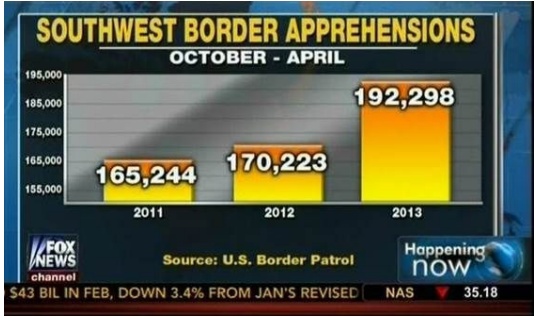


Encoding data using visual cues

In general, when displaying quantities, position and length are preferred over angles and/or area. Brightness and color are even harder to quantify than angles. But, as we will see later, they are sometimes useful when more than two dimensions must be displayed at once.

Know when to include 0

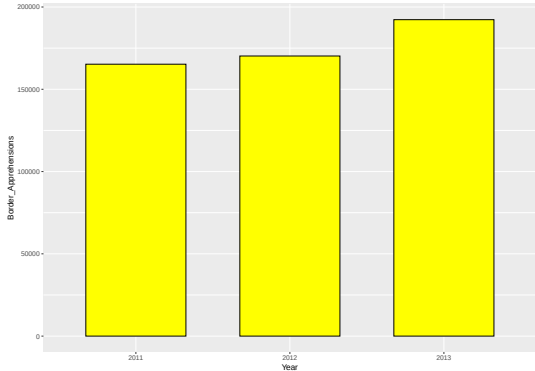
A barplot implies the length is proportional to the quantities being displayed. By avoiding 0, relatively small differences can be made to look much bigger:



(Source: http://mediamatters.org/blog/2013/04/05/fox-news-newest-dishonest-chart-immigration-enf/193507)

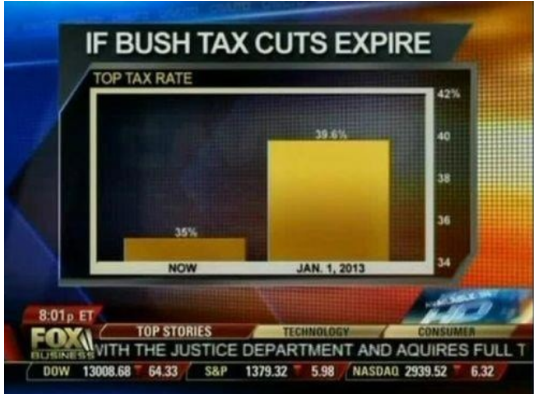
Know when to include 0

The plot appears that apprehensions have almost tripled when, in fact, they increased by about 16%. Starting the graph at 0 illustrates this clearly:



Know when to include 0

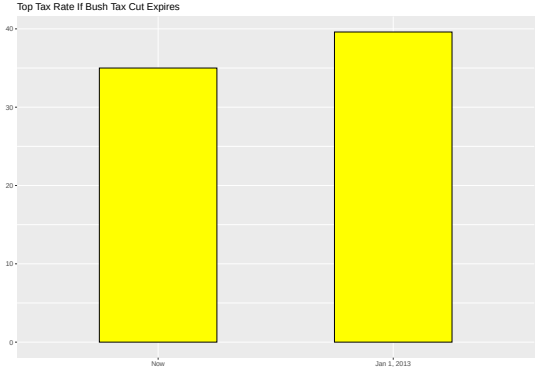
Here is another example:



(Source: http://flowingdata.com/2012/08/06/fox-news-continues-charting-excellence/

Know when to include 0

That plot makes a 13% increase look like a five fold change:



Know when to include 0

Finally, here is an extreme example that makes a very small difference of under 2% look like a 10-100 fold change:

(Source: https://www.pakistantoday.com.pk/2018/05/18/whats-at-stake-in-venezuelan-presidential-vote)

Do not distort quantities

During President Barack Obama's 2011 State of the Union Address, the following was used to compare the GDPs of competing nations:

(Source: https://www.youtube.com/watch?v=kl2g40GoRxg)

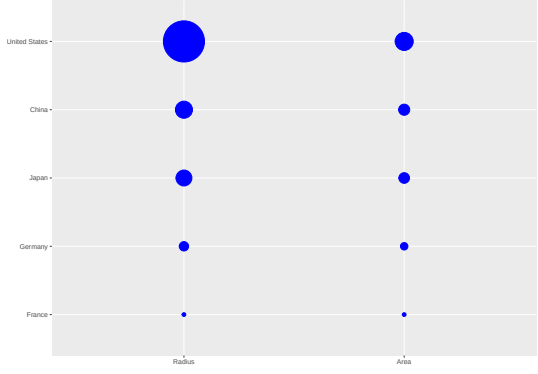
Do not distort quantities

Judging by the area of the circles, the US appears to have an economy over five times larger than China's and over 30 times larger than France's. However, if we look at the actual numbers, we see that this is not the case. The actual ratios are 2.6 and 5.8 times bigger than China and France, respectively.

The reason for this distortion is that the radius, rather than the area, was made to be proportional to the quantity, which implies that the proportion between the areas is squared: 2.6 turns into 6.5 and 5.8 turns into 34.1.

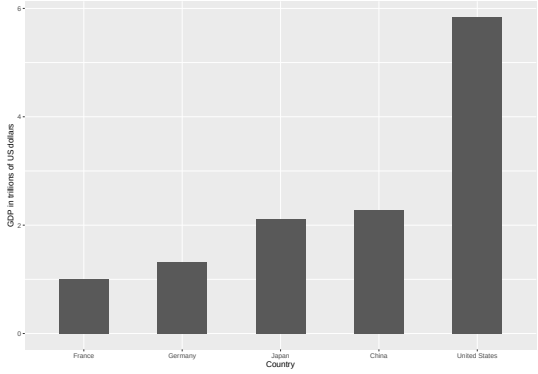
Do not distort quantities

Here is a comparison of the circles we get if we make the value proportional to the radius and to the area:



Do not distort quantities

Not surprisingly, **ggplot2** defaults to using area rather than radius. Of course, in this case, we really should not be using area at all since we can use position and length:



Order categories by a meaningful value

When one of the axes is used to show categories, as is done in barplots, the default **ggplot2** behavior is to order the categories alphabetically when they are defined by character strings. If they are defined by factors, they are ordered by the factor levels.

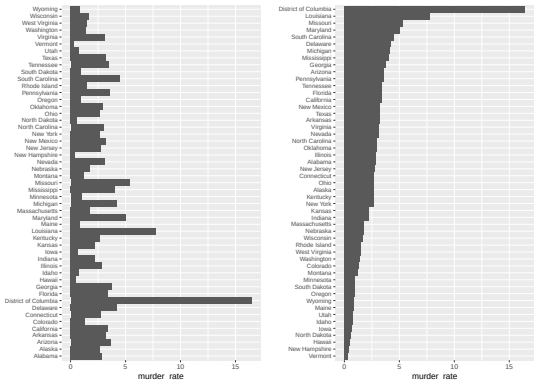
We rarely want to use alphabetical order. Instead, we should order by a meaningful quantity. In all the previous casese, the barplots were ordered by the values being displayed.

The exception was the graph showing barplots comparing browsers. In this case, we kept the order the same across the barplots to ease the comparison. Specifically, instead of ordering the browsers separately in the two years, we ordered both years by the average value of 2000 and 2015.

The **tidyverse** `reorder` function, helps us achieve this goal!

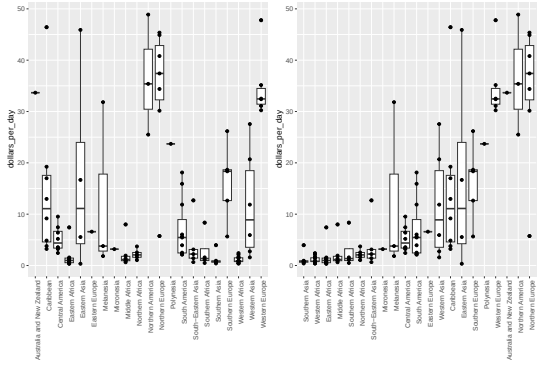
Order categories by a meaningful value

To appreciate how the right order can help convey a message, suppose we want to create a plot to compare the murder rate across states. Note the difference when we order by the actual rate:



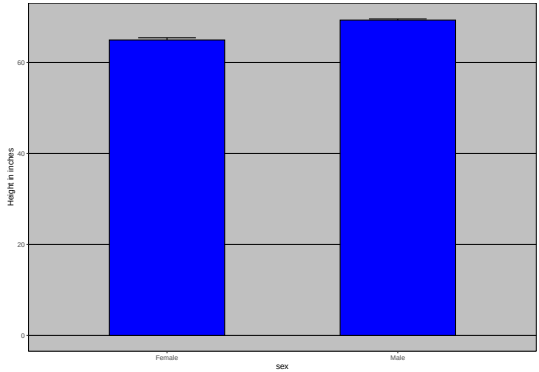
Order categories by a meaningful value

The `reorder` function lets us reorder groups as well. Here are the two versions plotted (second ordered by median):



Show the data

We now shift our attention to displaying data, with a focus on comparing groups. A commonly seen plot used for comparisons between groups, popularized by software such as Microsoft Excel, is the dynamite plot, which shows the average and standard errors:

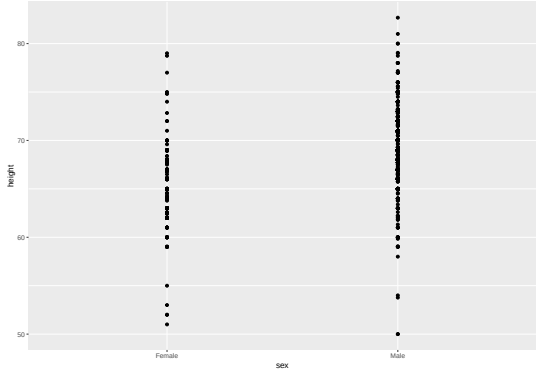


Show the data

The average of each group is represented by the top of each bar and the antennae extend out from the average to the average plus two standard errors. The bars go to 0: does this mean there are tiny humans measuring less than one foot? Are all males taller than the tallest females? Is there a range of heights?

Show the data

This brings us to our first principle: show the data. This simple **ggplot2** code already generates a more informative plot than the barplot by simply showing all the data points:

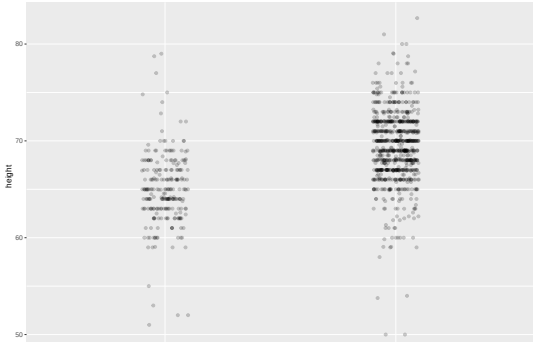


Show the data

For example, this plot gives us an idea of the range of the data. However, this plot has limitations as well, since we can't really see all the 238 and 812 points as many are plotted on top of each other. In the following slides we point out two ways we can improve a plot showing all the points.

Show the data

The first is to add *jitter*, which adds a small random shift to each point. A second improvement comes from using *alpha blending*: making the points somewhat transparent. The more points fall on top of each other, the darker the plot, which also helps us get a sense of how the points are distributed:

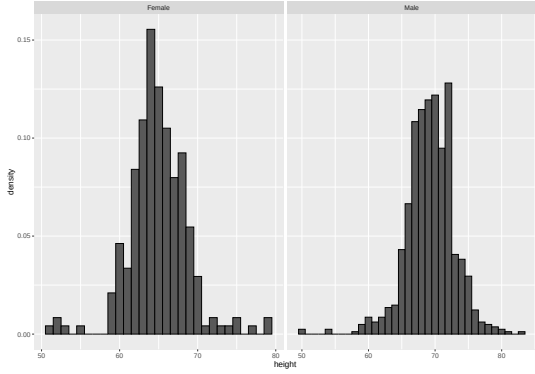


Show the data

Now we start getting a sense that, on average, males are taller than females. We also note dark horizontal bands of points, demonstrating that many report values that are rounded to the nearest integer.

Ease comparisons: Use common axes

Since there are so many points, it is more effective to show distributions rather than individual points. We therefore show histograms for each group:

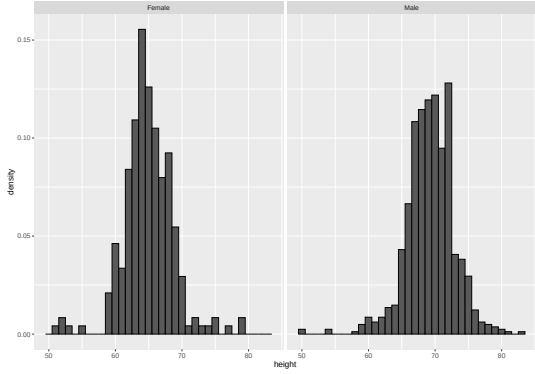


Use common axes

However, from this plot it is not immediately obvious that males are, on average, taller than females. We have to look carefully to notice that the x-axis has a higher range of values in the male histogram.

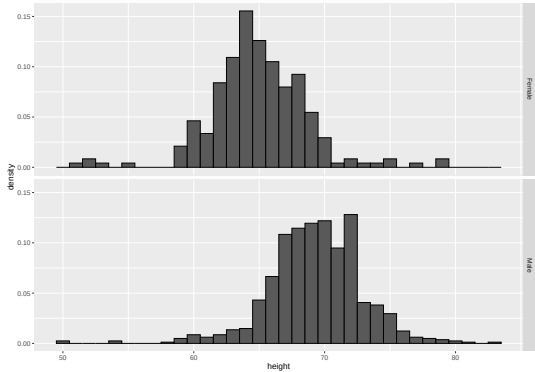
Use common axes

An important principle here is to **keep the axes the same** when comparing data across two plots. Below we see how the comparison becomes easier:



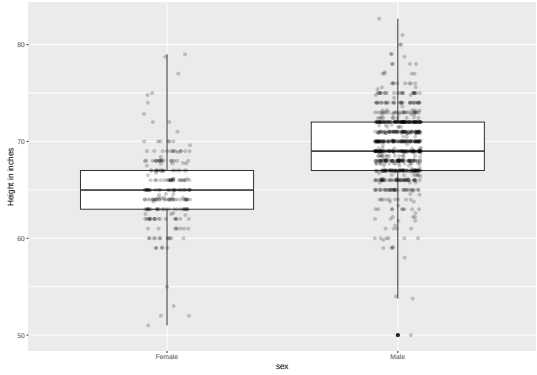
Plot Alignment

In these histograms, the visual cue related to decreases or increases in height are shifts to the left or right, respectively: horizontal changes. Aligning the plots vertically helps us see this change when the axes are fixed:



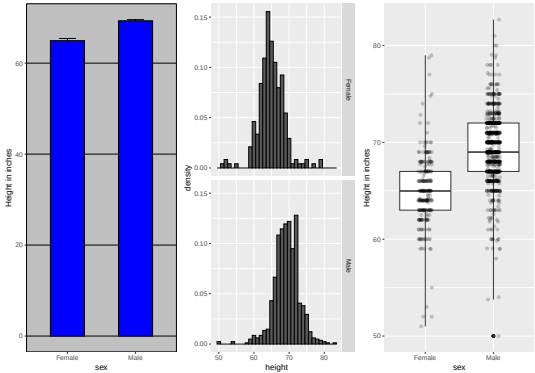
Plot Alignment

If we want the more compact summary provided by boxplots, we then align them horizontally. Following our *show the data* principle, we then overlay all the data points:



Plot Alignment

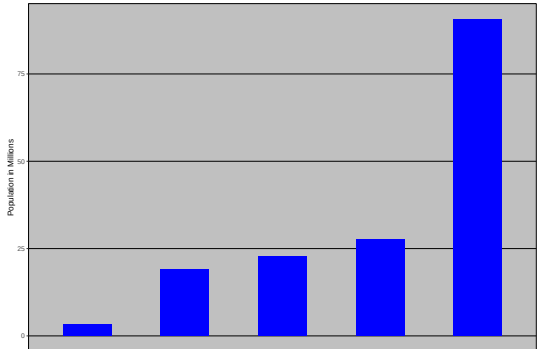
Now compare these three plots, based on exactly the same data:



Notice how much more we learn from the two plots on the right. Barplots are useful for showing one number, but not very useful when we want to describe distributions.

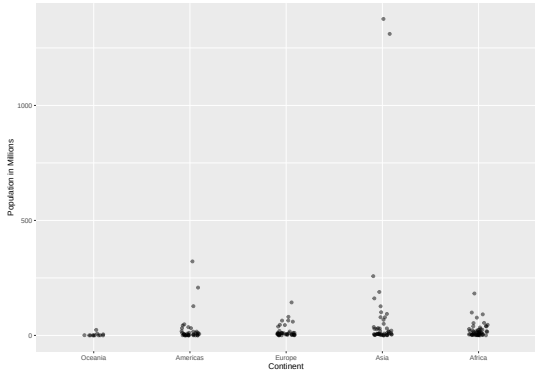
Consider transformations

The use of the log transformation is useful in cases where the changes are multiplicative. Population size is an example in which a log transformation can yield a more informative transformation. As an example, consider this barplot showing the average population sizes for each continent in 2015:



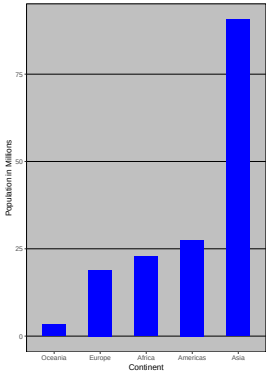
Consider transformations

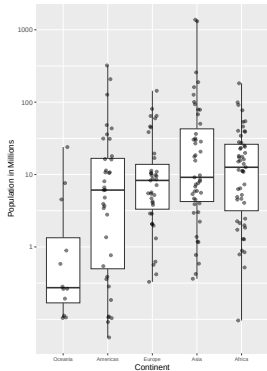
From this plot, one would conclude that countries in Asia are much more populous than in other continents. Following the *show the data* principle, we quickly notice that this is due to two very large countries, which we assume are India and China:



Consider transformations

Using a log transformation here provides a much more informative plot. We compare the original barplot to a boxplot using the log scale transformation for the y-axis:





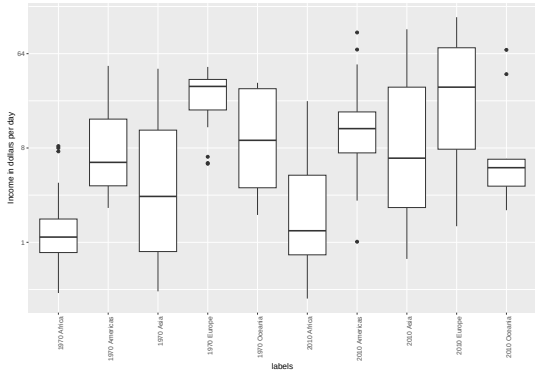
Consider transformations

With the new plot, we realize that countries in Africa actually have a larger median population size than those in Asia.

Other transformations you should consider are the logistic transformation (`logit`), useful to better see fold changes in odds, and the square root transformation (`sqrt`), useful for count data.

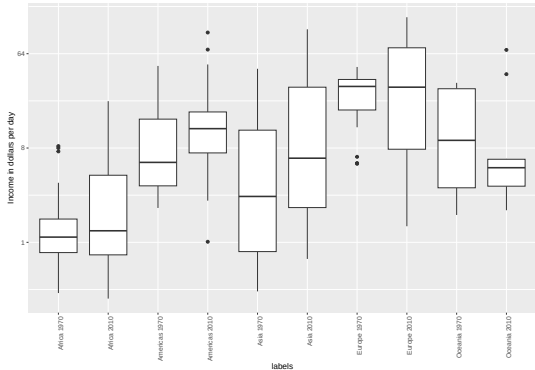
Visual cues to be compared should be adjacent

For each continent, let's compare income in 1970 versus 2010. When comparing income data across regions between 1970 and 2010, we made a figure similar to the one below, but this time we investigate continents rather than regions.



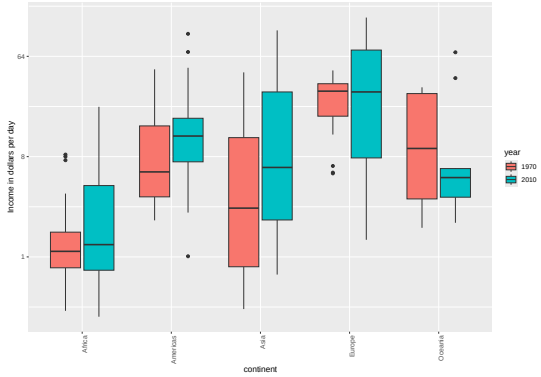
Visual cues to be compared should be adjacent

The default in **ggplot2** is to order labels alphabetically so the labels with 1970 come before the labels with 2010. It is much easier to make the comparison when the boxplots for that continent are next to each other:



Use color

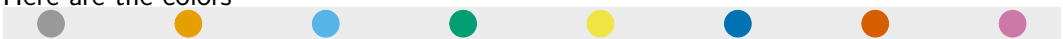
The comparison becomes even easier to make if we use color to denote the two things we want to compare:



Think of the color blind

About 10% of the population is color blind. Unfortunately, the default colors used in **ggplot2** are not optimal for this group. However, **ggplot2** does make it easy to change the color palette used in the plots. An example of how we can use a color blind friendly palette is described here: http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/#a-colorblind-friendly-palette:

Here are the colors



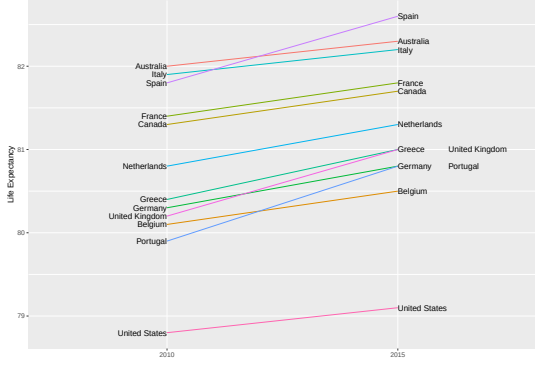
There are several resources that can help you select colors, for example this one: http://bconnelly.net/2013/10/creating-colorblind-friendly-figures/.

Plots for two variables

In general, you should use scatterplots to visualize the relationship between two variables. However, there are some exceptions and we describe two alternative plots here: the *slope chart* and the *Bland-Altman plot*.

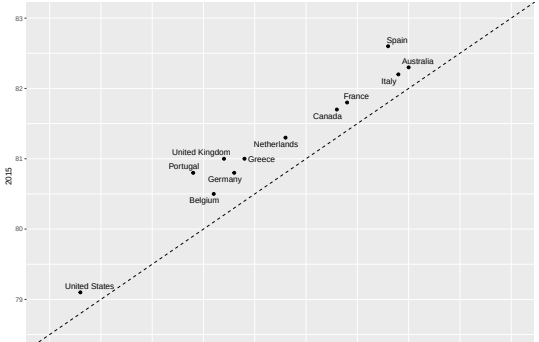
Slope charts

Slope charts are useful if you have paired data, such as same individuals at different time points (for a small number of comparisons). For example, comparing life expectancy between 2010 and 2015:



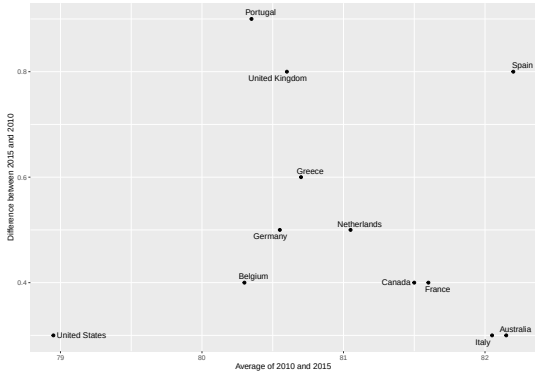
Slope charts

An advantage of the slope chart is that it permits us to quickly get an idea of changes based on the slope of the lines. Although we are using angle as the visual cue, we also have position to determine the exact values. Comparing the improvements is a bit harder with a scatterplot:



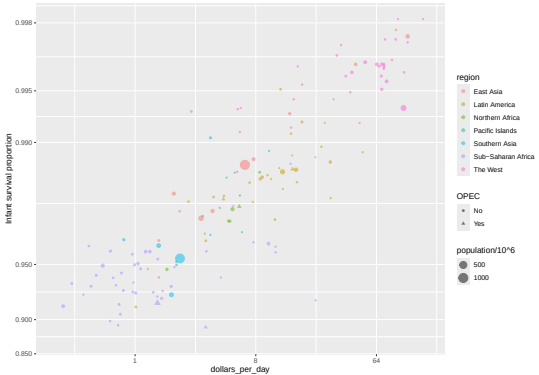
Bland-Altman plot

Since we are primarily interested in the difference, it makes sense to dedicate one of our axes to it. The Bland-Altman plot, also known as the Tukey mean-difference plot and the MA-plot, shows the difference versus the average:



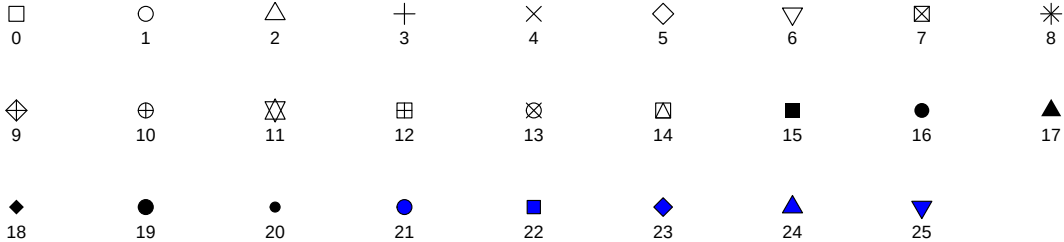
Encoding a third variable

An earlier scatterplot showed the relationship between infant survival and average income. Below is a version of this plot that encodes three variables: OPEC membership, region, and population.



Encoding a third variable

We encode categorical variables with color and shape. These shapes can be controlled with `shape` argument. Below are the shapes available for use in R. For the last five, the color goes inside.



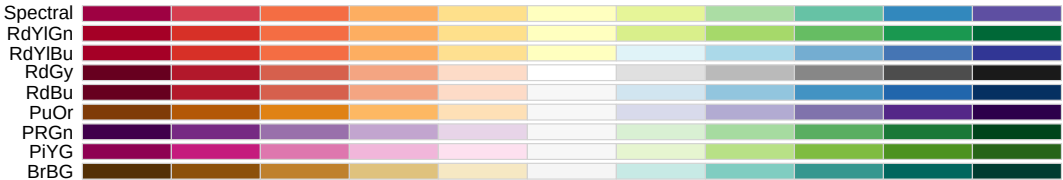
Encoding a third variable

For continuous variables, we can use color, intensity, or size. We now show an example of how we do this with a case study.

When selecting colors to quantify a numeric variable, we choose between two options: sequential and diverging. Sequential colors are suited for data that goes from high to low. High values are clearly distinguished from low values. Here are some examples offered by the package `RColorBrewer`:

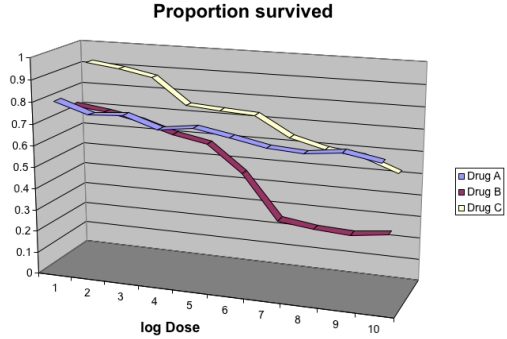
Encoding a third variable

Diverging colors are used to represent values that diverge from a center. We put equal emphasis on both ends of the data range: higher than the center and lower than the center. An example of when we would use a divergent pattern would be if we were to show height in standard deviations away from the average. Here are some examples of divergent patterns:



Avoid pseudo-three-dimensional plots

The figure below, taken from the scientific literature[4], shows three variables: dose, drug type and survival. Although your screen/book page is flat and two-dimensional, the plot tries to imitate three dimensions and assigned a dimension to each variable.



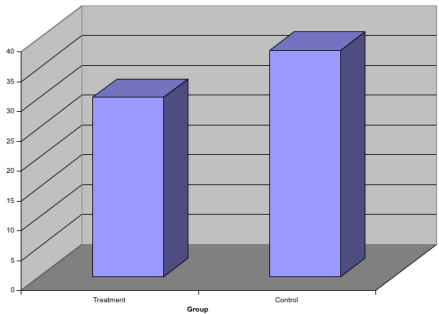
[4]https://projecteuclid.org/download/pdf_1/euclid.ss/1177010488

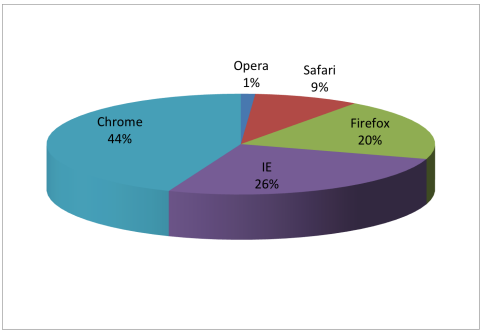
Avoid pseudo-three-dimensional plots

Humans are not good at seeing in three dimensions (which explains why it is hard to parallel park) and our limitation is even worse with regard to pseudo-three-dimensions. To see this, try to determine the values of the survival variable in the plot above. Can you tell when the purple ribbon intersects the red one? This is an example in which we can easily use color to represent the categorical variable instead of using a pseudo-3D:

Avoid pseudo-three-dimensional plots

Pseudo-3D is sometimes used completely gratuitously: plots are made to look 3D even when the 3rd dimension does not represent a quantity. This only adds confusion and makes it harder to relay your message. Here are two examples:





Avoid too many significant digits

By default, statistical software like R returns many significant digits. The default behavior in R is to show 7 significant digits. That many digits often adds no information and the added visual clutter can make it hard for the viewer to understand the message. As an example, here are the per 10,000 disease rates, computed from totals and population in R, for California across the five decades:

state	year	Measles	Pertussis	Polio
California	1940	37.8826320	18.3397861	0.8266512
California	1950	13.9124205	4.7467350	1.9742639
California	1960	14.1386471	NA	0.2640419
California	1970	0.9767889	NA	NA
California	1980	0.3743467	0.0515466	NA

Avoid too many significant digits

We are reporting precision up to 0.00001 cases per 10,000, a very small value in the context of the changes that are occurring across the dates. In this case, two significant figures is more than enough and clearly makes the point that rates are decreasing:

state	year	Measles	Pertussis	Polio
California	1940	37.9	18.3	0.8
California	1950	13.9	4.7	2.0
California	1960	14.1	NA	0.3
California	1970	1.0	NA	NA
California	1980	0.4	0.1	NA

Avoid too many significant digits

Useful ways to change the number of significant digits or to round numbers are `signif` and `round`. You can define the number of significant digits globally by setting options like this: `options(digits = 3)`.

Avoid too many significant digits

Another principle related to displaying tables is to place values being compared on columns rather than rows. Note that our table above is easier to read than this one:

state	disease	1940	1950	1960	1970	1980
California	Measles	37.9	13.9	14.1	1	0.4
California	Pertussis	18.3	4.7	NA	NA	0.1
California	Polio	0.8	2.0	0.3	NA	NA

Know your audience

Graphs can be used for 1) our own exploratory data analysis, 2) to convey a message to experts, or 3) to help tell a story to a general audience. Make sure that the intended audience understands each element of the plot.

As a simple example, consider that for your own exploration it may be more useful to log-transform data and then plot it. However, for a general audience that is unfamiliar with converting logged values back to the original measurements, using a log-scale for the axis instead of log-transformed values will be much easier to digest.

Session Info

```
## R version 4.4.2 (2024-10-31)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.2.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3.12.0
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Denver
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] RColorBrewer_1.1-3 ggrepel_0.9.6      scales_1.3.0       ggthemes_5.1.0
##  [5] gridExtra_2.3      dslabs_0.8.0       lubridate_1.9.4    forcats_1.0.0
##  [9] stringr_1.5.1      dplyr_1.1.4        purrr_1.0.2        readr_2.1.5
## [13] tidyr_1.3.1        tibble_3.2.1       ggplot2_3.5.1      tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] generics_0.1.3    xml2_1.3.6        stringi_1.8.4     hms_1.1.3
##  [5] digest_0.6.37     magrittr_2.0.3    evaluate_1.0.3    grid_4.4.2
##  [9] timechange_0.3.0  fastmap_1.2.0     tinytex_0.54      viridisLite_0.4.2
```