Visualization in R

Quantitative Applications for Data Analysis

Alexey Fedoseev

February 24, 2026

Visualization in R

Visualization is the product of almost all computations.

R has built-in capabilities to graphically present your data. However there are a number of high-quality visualization packages additionally available in R

- `ggplot2` has a powerful layer system that makes it easy to combine different sources of data;
- `Lattice` strong in visualizing multi-variate data;
- `Plotly` makes interactive plots;
- `highcharter` allows easy dynamic charting;
- `Leaflet` builds interactive maps, and others.

Installing ggplot2

To install ggplot2 package run the following command in your terminal

```
> install.packages("ggplot2")
Installing package into '/Users/alexey/R/x86_64-pc-linux-gnu-library/4.2'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
...
* DONE (ggplot2)
...
```

ggplot2 is imported using the following command

```
library(ggplot2)
```

Visualization using ggplot2

The **gg** in ggplot2 stands for Grammar of Graphics, a graphic concept which describes plots by using a "grammar". Basically, a grammar of graphics is a framework which follows a layered approach to describe and construct visualizations or graphics in a structured manner.

According to ggplot2 concept, a plot can be divided into different fundamental parts:

Plot = Data + Aesthetics + Geometry.

The principal components of every plot can be defined as follows:

- Data is a data frame
- Aesthetics is used to indicate x and y variables. It can also be used to control the color, the size or the shape of points, the height of bars, etc.
- Geometry corresponds to the type of graphics (histogram, box plot, line plot, density plot, scatter plot, etc.)

Visualization using ggplot2

The `ggplot2` package has two main functions `qplot()` and `ggplot()` that allows us to create plots of different complexity.

- `qplot()` is a **q**uick plot function which is easy to use for simple plots, and
- The `ggplot()` function is more flexible and robust than `qplot` for building a plot piece by piece.

The generated plot can be kept as a variable and then printed at any time using the function `print()`.

You can save your plot in the current working directory using the command ggsave

```
ggsave("plot.pdf", width = 5, height = 5)
```

Box plot

The box plot displays the distribution of a continuous variable. It visualizes the following statistics:

- the median;
- first and third quartiles (the 25th and 75th percentiles) and all “outlying” points individually;
- whiskers extend 1.5 * IQR from the first or third quartiles (where IQR is the inter-quartile range, or distance between the first and third quartiles), and
- outlying points.

Hands-on

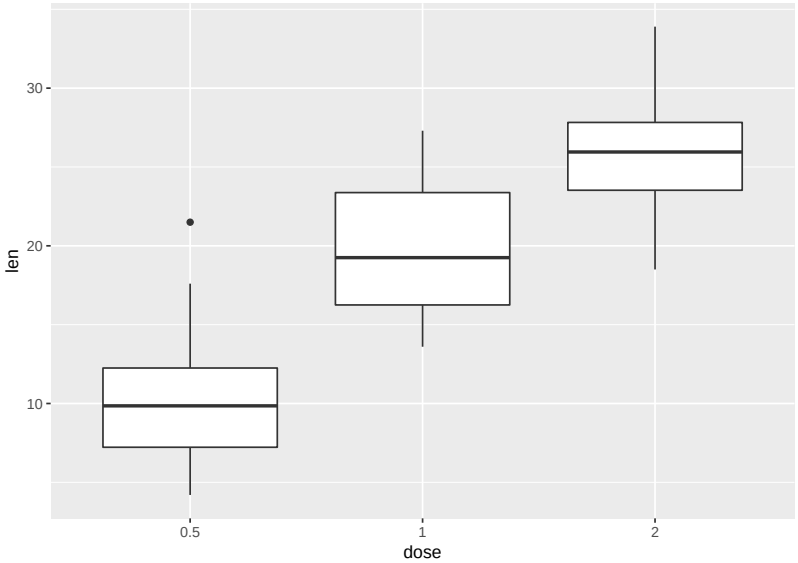
Create a new script `boxplot.R` with the following code.

```
library(ggplot2)
# ToothGrowth dataset comes with R
ToothGrowth$dose <- as.factor(ToothGrowth$dose)

ggplot(data = ToothGrowth, aes(x = dose, y = len)) +
  geom_boxplot()
```

Running the script does not display anything on the screen, however, if you check the directory you will see a new file `Rplots.pdf` with your plot. Note that every time you re-run your script R will overwrite the file `Rplots.pdf` with the new plot.

```
user@scinet ~ $ Rscript boxplot.R
user@scinet ~ $ ls
Rplots.pdf  boxplot.R
```

Box plot



Improving the plot

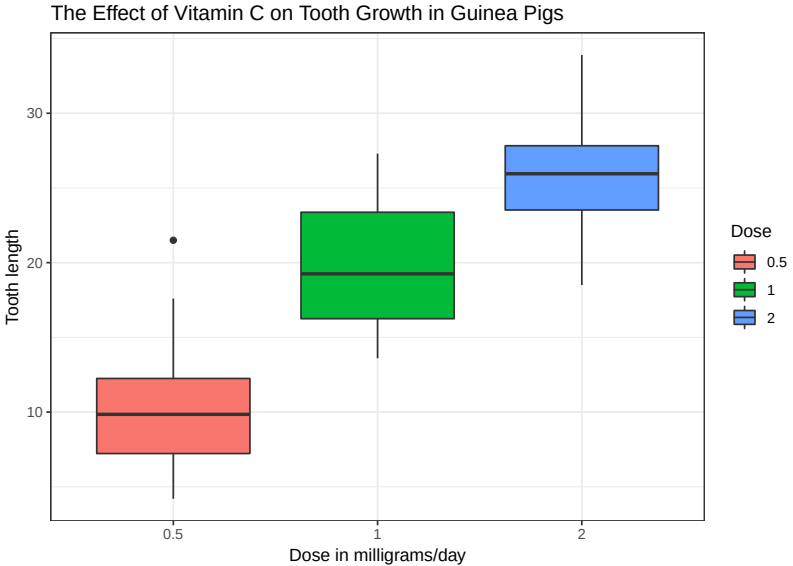
Our plot shows the essential information, however, it is not very descriptive. Let us label it properly and change its outlook.

```
ggplot(data = ToothGrowth, aes(x = dose, y = len, fill = dose)) +
  geom_boxplot() +
  labs(x = "Dose in milligrams/day",
       y = "Tooth length",
       title = "The Effect of Vitamin C on Tooth Growth in Guinea Pigs",
       fill="Dose") +
  theme_bw()
```

The aesthetic mappings, such as `color`, `shape`, and `fill` map to categorical variables which means they subset the data into groups that are displayed in different colors or shapes.

Using the command `labs` you can specify labels on your plot, where `x` and `y` correspond to the axis labels, `title` is responsible for the main title and `fill` for the title of the legend (since we used `fill` in aesthetics).

Box plot

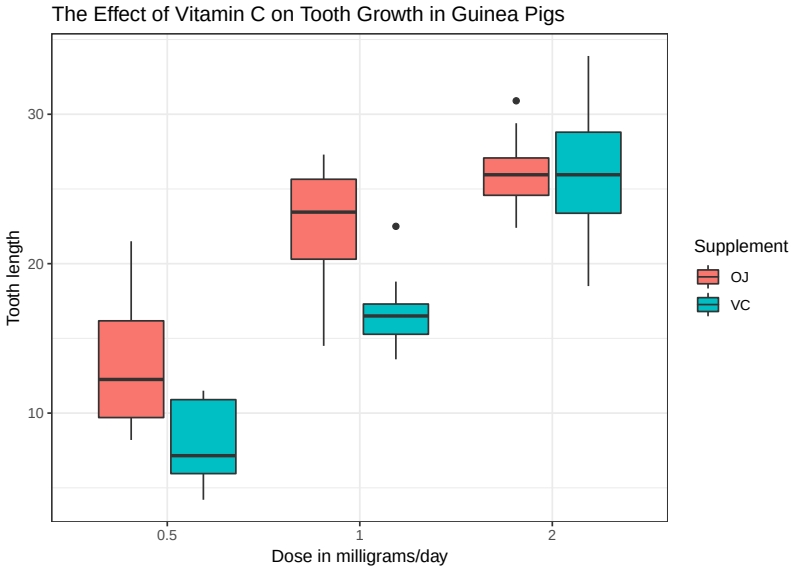

The Effect of Vitamin C on Tooth Growth in Guinea Pigs
Tooth length
30
20
10
0.5
1
2
Dose in milligrams/day
Dose
0.5
1
2

Box plot

We can add additional information on our plot by specifying that we want to fill our boxes based on the supplement type.

```
ggplot(data = ToothGrowth, aes(x = dose, y = len, fill = supp)) +
  geom_boxplot() +
  labs(x = "Dose in milligrams/day",
       y = "Tooth length",
       title = "The Effect of Vitamin C on Tooth Growth in Guinea Pigs",
       fill="Supplement") +
  theme_bw()
```

Box plot

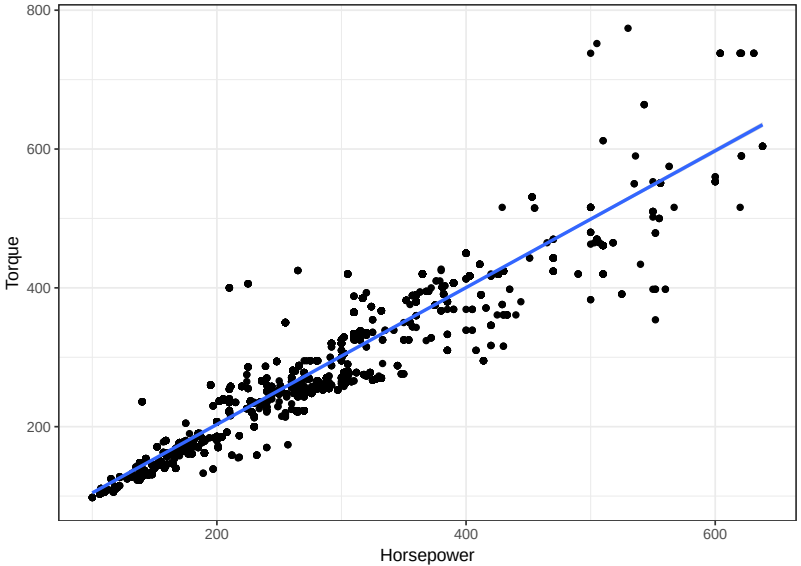

The Effect of Vitamin C on Tooth Growth in Guinea Pigs
Tooth length
30
20
10
0.5
1
2
Dose in milligrams/day
Supplement
OJ
VC

Example plot using qplot

The function `qplot()` provides a simpler syntax while the function `ggplot()` allows maximum features and flexibility. Additionally, `qplot()` lets you work with vectors.

```
library(ggplot2)

cars <- read.csv("cars.csv")

cars <- subset(cars, Highway.mpg < 200) # Remove the outlier

# plotting with qplot

qplot(x = Horsepower, y = Torque, data = cars, geom = "point") +
        geom_smooth(method="lm") +
        theme_bw()
```

Example plot using `qplot`



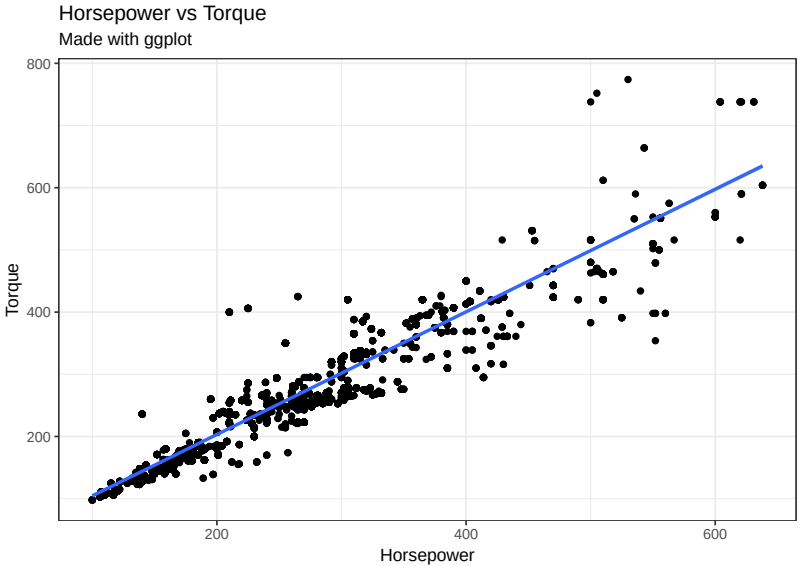
Example plot using ggplot

The previous chart we can re-create using ggplot instead qplot.

Notice how I add more features to the plot.

```
ggplot(data = cars, aes(x = Horsepower, y = Torque)) +
        geom_point() +
        geom_smooth(method="lm") +
        ggtitle("Horsepower vs Torque", subtitle = "Made with ggplot") +
        theme_bw()
```

Example plot using ggplot



Scatterplot

The most frequently used plot for data analysis is the scatterplot. It allows you to understand the nature of relationship between two variables.

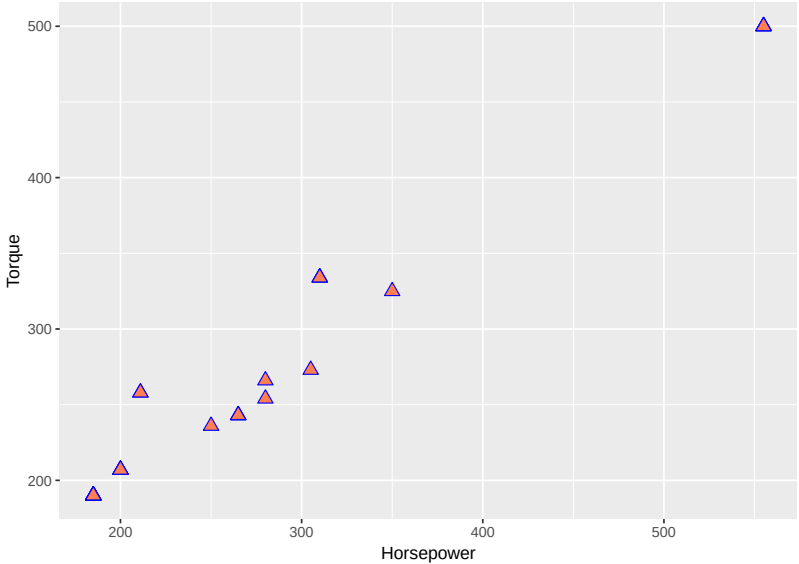
Simple scatter plots are created using the R code below.

```
geom_point(size, color, shape)
```

The color, the size and the shape of points can be changed using the function `geom_point()` as follow :

```
ggplot(data = cars[1:30,], aes(x = Horsepower, y = Torque)) +
        geom_point(size = 3, color = "blue", fill = "coral", shape = 24)
```

Scatterplot



Scatterplot

Scatterplot allows us to view multidimensional data. For example, we can use different colors to distinguish several groups in our data.

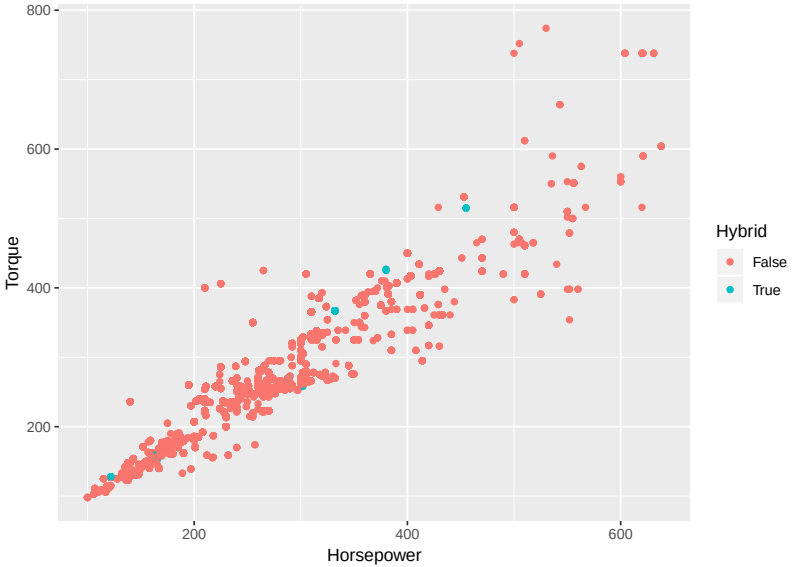
We already know how to change the color for all points in the scatterplot, but `ggplot` provides us with the ability to use different colors within our plot. To do so, specify color as the column name inside the aesthetic parameter.

```
ggplot(data = cars, aes(x = Horsepower, y = Torque)) +
        geom_point(aes(color = Hybrid))
```

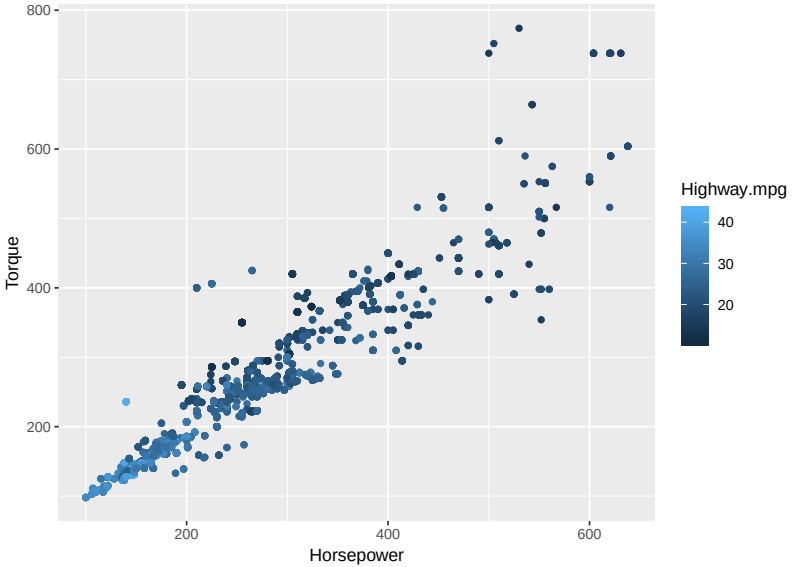
You can also specify a continuous feature

```
ggplot(data = cars, aes(x = Horsepower, y = Torque)) +
        geom_point(aes(color = Highway.mpg))
```

Scatterplot (colorize by categories)



Scatterplot (colorize by a continuous feature)



Scatterplot

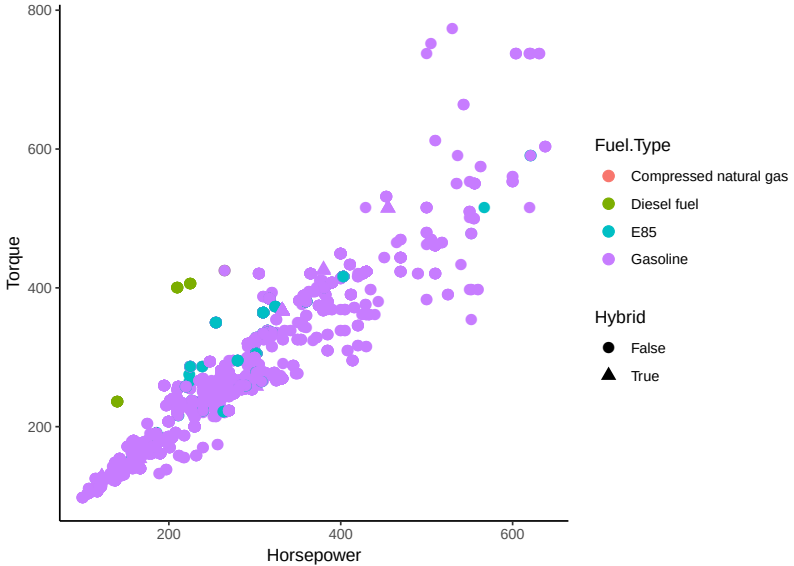
Using scatterplot we can visualize even 4 different features on one plot. The shape of the point could help us to differentiate between different categories. Note that the `size` parameter is specified outside of the **aes**thetic which means it is applied to all points.

```
ggplot(data = cars, aes(x = Horsepower, y = Torque)) +
  geom_point(size = 3, aes(shape = Hybrid, color = Fuel.Type)) +
  theme_classic()
```

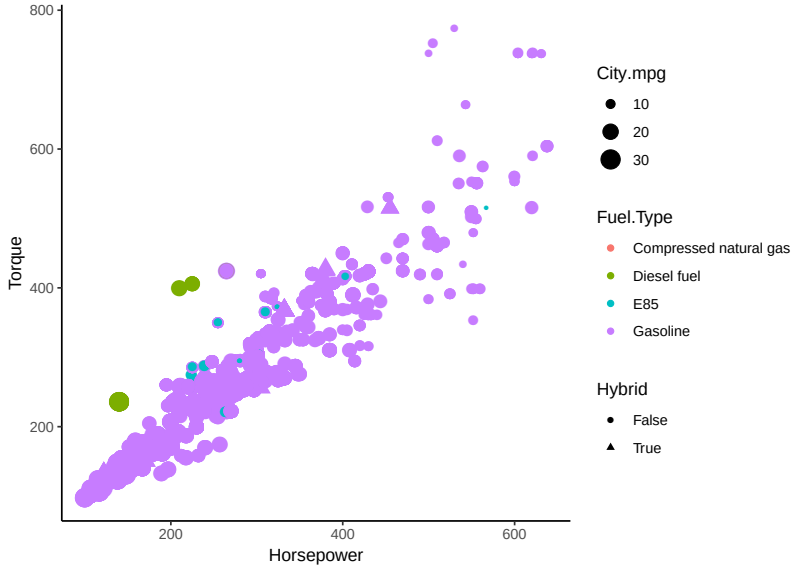
We can use `size` as an additional feature to distinguish different values.

```
ggplot(data = cars, aes(x = Horsepower, y = Torque)) +
  geom_point(aes(shape = Hybrid, color = Fuel.Type, size = City.mpg)) +
  theme_classic()
```

Scatterplot (4 features)



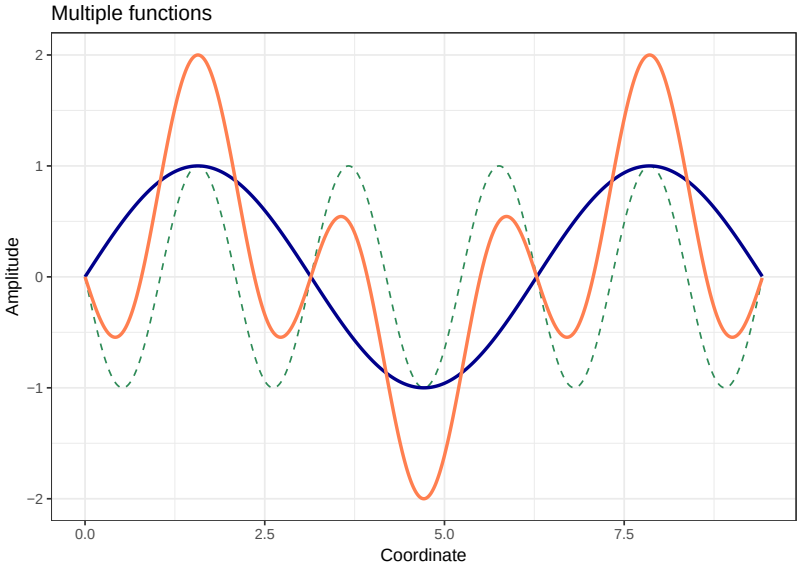
Scatterplot (5 features)



Line plot

```r
library(ggplot2)

x <- seq(0, 3*pi, by=0.01)
fast.wave <- sin(3*x - pi)
slow.wave <- sin(x)
total.wave <- fast.wave + slow.wave

funcs <- data.frame(x, fast.wave, slow.wave, total.wave)

ggplot(data = funcs, aes(x = x)) +
  geom_line(aes(y = fast.wave), col = "seagreen", linetype = 2) +
  geom_line(aes(y = slow.wave), col = "darkblue", linetype = 1, size = 1) +
  geom_line(aes(y = total.wave), col = "coral", size = 1) +
  labs(x = "Coordinate", y = "Amplitude", title = "Multiple functions") +
  theme_bw()
```

Line plot



Other plots

Scatterplot is one of the most commonly used plot type to visualizing data. The `ggplot2` package also supports many other types of plots. You can use the following functions combined with the command `ggplot`:

- use `geom_histogram()` to plot histograms;
- use `geom_bar()` to display bar plots;
- use `geom_line()` to draw lines;
- and others.

You can find most of `ggplot2` commands on the following cheat sheet:

https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf

Plotting for professional scientists

Scientists must make their work presentable. That means making a serious effort to make your results easy for your audience to understand.

Suggestions:

- DO label everything: axes, lines, data.
- DO put units on your axis labels, including color bars.
- DO use legends, where appropriate.
- DO adjust the font size of axis and tick labels so that they can actually be read.
- DO NOT put a title label over your plot if use the caption (for talks and papers).
- DO NOT use colors that cannot be read on a white background (yellow, orange, light green, cyan). This is especially important for figures used in talks.
- DO make your data fill the plot.

More plotting for professional scientists

It's also important to consider file types and sizes.

More suggestions:

- DO set the image size and resolution to that requested by the journal you're submitting to.
- DO NOT use bitmap or stroke fonts for your plot. These cannot be rescaled properly, which is often needed for publication. Use vector (the default for R) or TrueType fonts.
- If possible, DO NOT use image file types that cannot be scaled (bitmap, jpeg). Use EPS or PDF.
- DO NOT leave a bunch of white space around the outside of your plots.
- DO make a script (in R or whatever language), whose sole purpose is to make that plot for your paper.