R programming for beginners

Ni Shuai

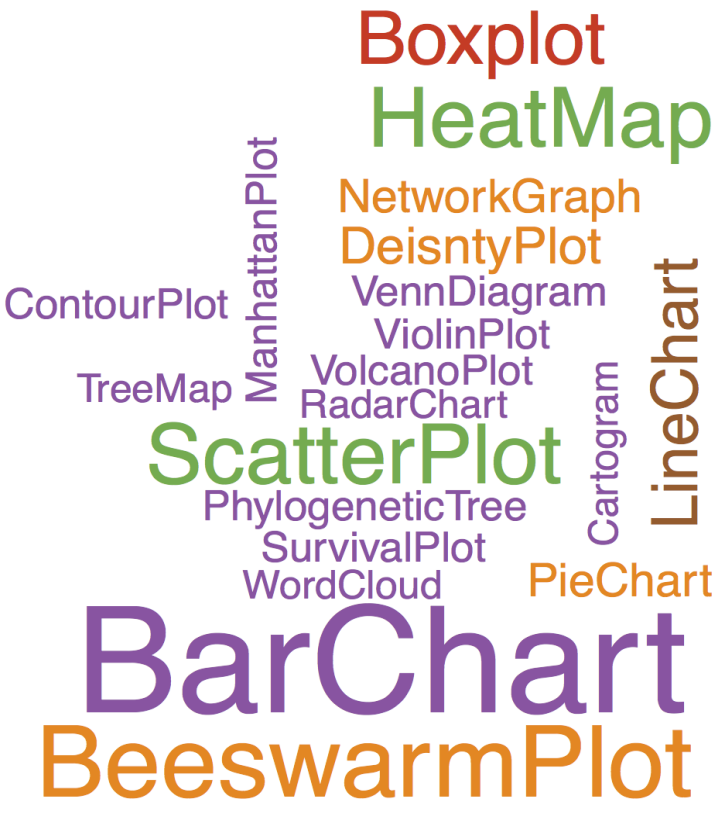
Computational Genome Biology
German Cancer Research Center (DKFZ)

November, 2016

Graphics in R

"A picture is worth a thousand words"

- Various inbuilt functions and packages to present your data
- Build elegant and complicated plots with little effort
- High quality and high resolution graphs for publication



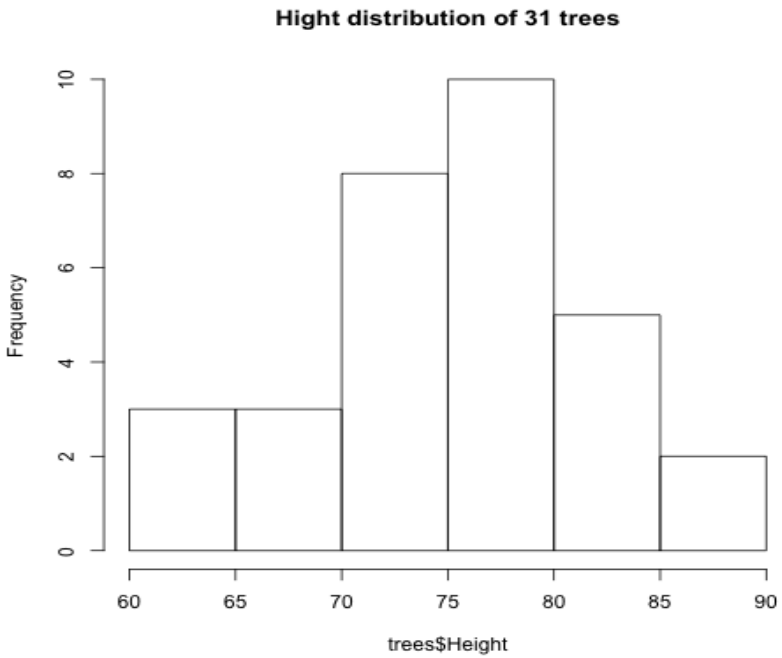
Graphics in R

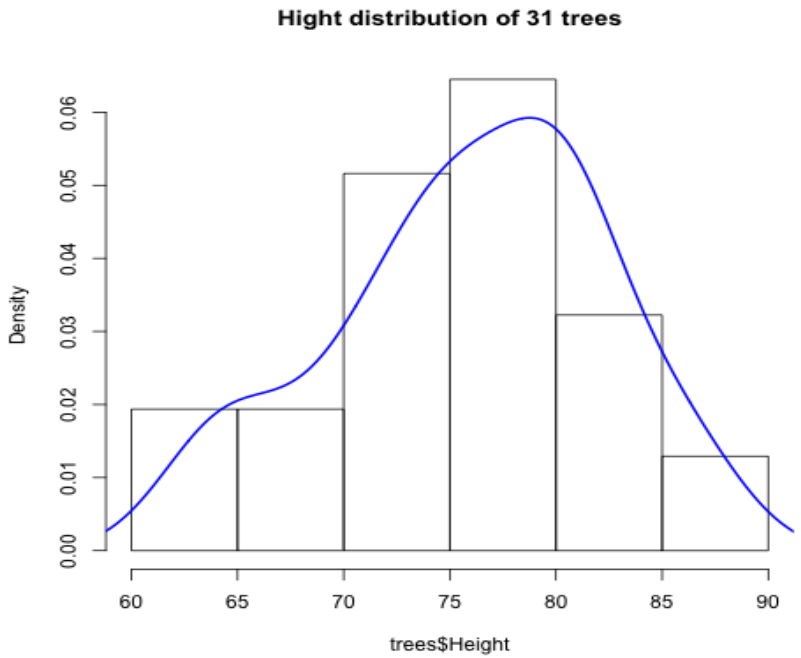
Let's look at the demos of the bacis plots that can be generated in R

```
#demo(graphics)
```

There are two kinds of graphical functions:

- High-level functions
  Initializing graphical devide and create a new plot
- Low-level functions
  Plotting functions add more information to an existing plot





Hig-level graphics functions

Frequently used high-level graphics functions:

plot()	Depend on input data type
boxplot()	Show the distribution of a vector
barplot()	Show in bars values in a vector
pie()	Illustrate numerical proportion
hist()	Show histogram of a numeric vector
pairs()	Scatter plot for each column in matrix
curve()	Draw a curve corresponding to a function
qqplot()	Scatter plot of qualtiles for two vectors

Usually calling a highly-level graphical function will erase the current plot. Some functions can detect the type of its input object and generate plot accordingly, for example let's try:

```
plot(iris$Sepal.Length)
plot(iris$Species)
plot(iris$Species, iris$Sepal.Length)
plot(iris$Sepal.Width, iris$Sepal.Length)
```

Low-level graphics functions

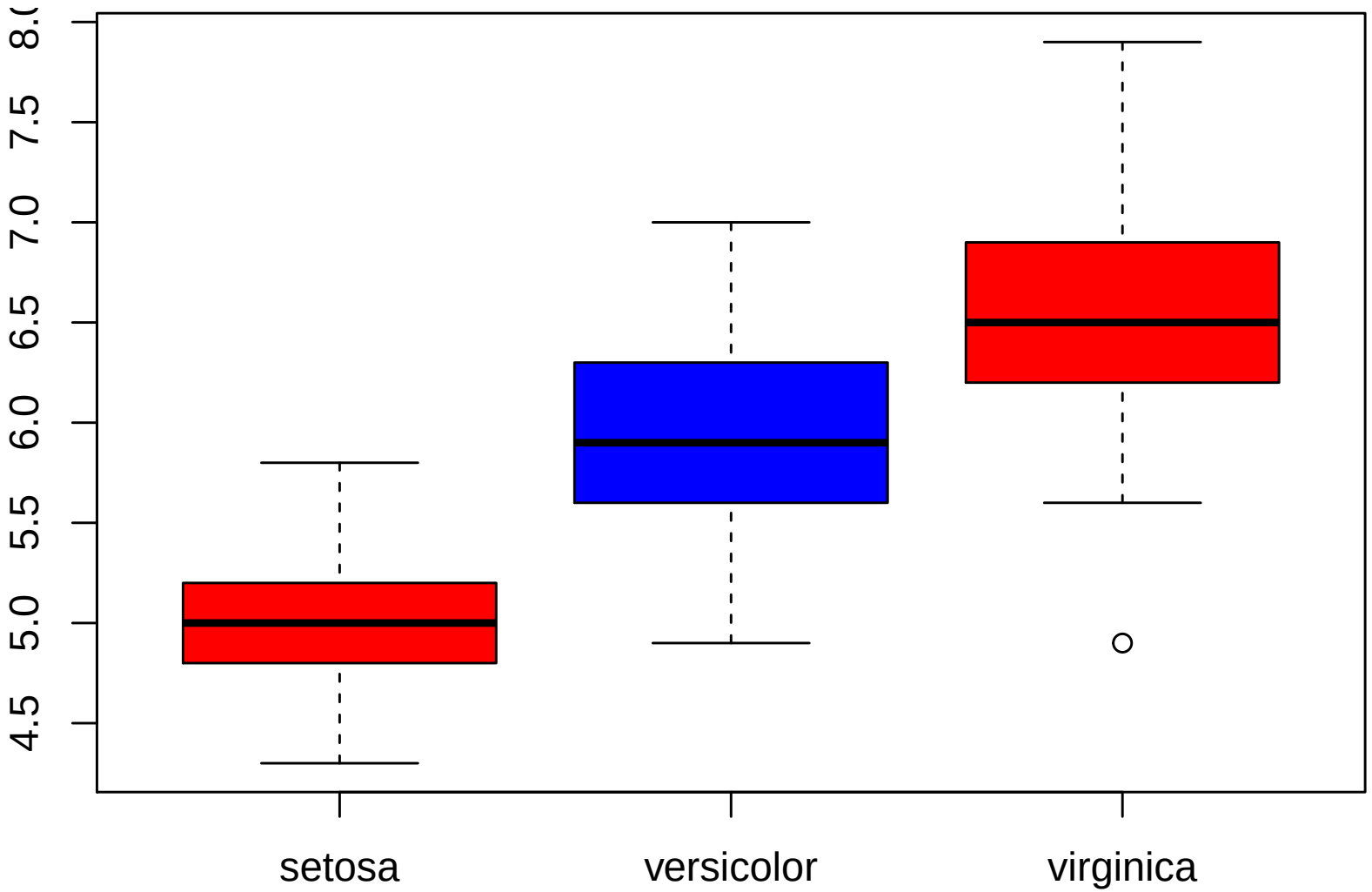
Sometimes high-level plotting functions cannot produce the exact kind of plot you desire. In this case, low-level plotting commands can be used to add extra pices (such as points, lines or text) to the current plot. Some of the most used low-level plotting functions are:

points(x, y)	Add points to the current plot
lines(x, y)	Add connected lines to the current plot
abline(a, b)	Add a line of slope b and intercept a to the current plot
abline(h=y)	Add a horizontal linen
abline(v=x)	Add a vertical line
legend(...)	Add a legend to the current plot
title(...)	Add a title to the current plot
axis(...)	Add an axis to the current plot

Graphics in R

Example: Box plot with boxplot()

```
boxplot(iris$Sepal.Length ~ iris$Species, col=(c("red","blue")))
```



Graphics in R

Arguments to function boxplot():

```
?boxplot
```

```
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
        notch = FALSE, outline = TRUE, names, plot = TRUE,
        border = par("fg"), col = NULL, log = "",
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
        horizontal = FALSE, add = FALSE, at = NULL)
```

Exercises:

- Plot the same bar plot with three colors: gold, darkgreen and skyblue.
- Name the plot 'Sepal length in 3 species'
- Add a horizontal line to the plot shows the average sepal length

Save your plot

3 steps to save your plot to file:

1. Choose the format that you want to use and initialize the device, e.g. png("c:/MyFirstPlot.png", res=300, width = 480, height = 480), here you can make adjustment to the size and resolution
2. Draw a plot with high-level and low-level graphical functions, you will not actually see the plot, because its saved in the choosen divice
3. Close the device: dev.off()

Some useful devices:

- screen graphics: when you use interactive R
- pdf: Adobe Portable Document Format
- png: PNG graphics
- jpeg: small size, not recommended

Graphics

Exercises:

- Draw the function

$$E = mc^2$$

  when 0< m <10 (c = 299792458 m/s)

- Draw a histgram of 1000 radom numbers from a normal distribution, give it a title
- Suppose your monthly expenditure is as follows,

Housing	Food	Cloths	Entainment	Other
600	200	250	350	200

  Present it in a pie chart.

- In the scatter plot Sepal.Length VS. Sepal.Width in iris, extend the x axis to include a new observation, Sepal.Length=3.2, Sepal.Width=2.5

- Save the scatter plot as 'lengthVSwidth.pdf'

Global Parameters

Arguments given to a plot function only changes the plot settings locally, what if we would like some setting to be applied to all plots? This way we use function "par()".

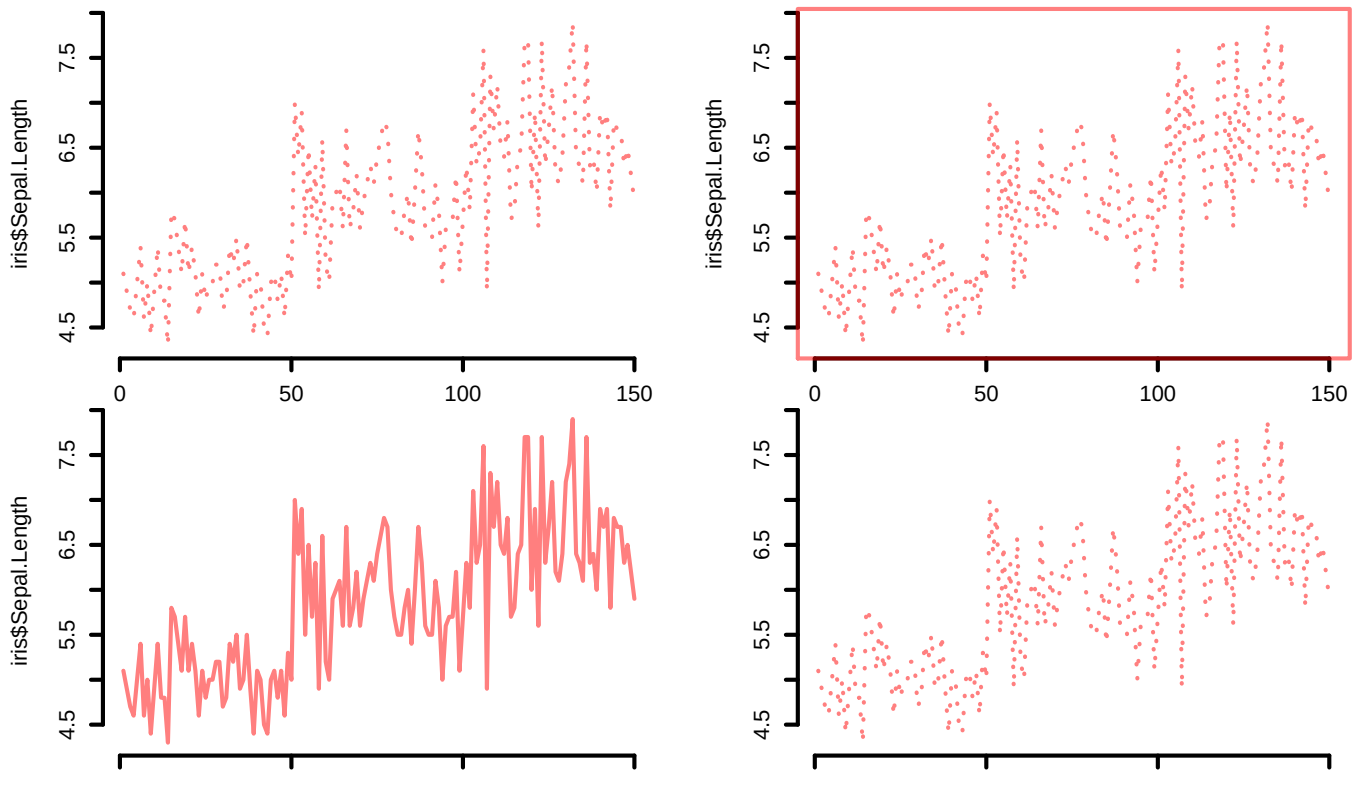
par() returns all current graphical parameters as a named list. Setting in each high-level plotting function will override the defaults set in par()

There are some parameters that can only be set by a call to par()

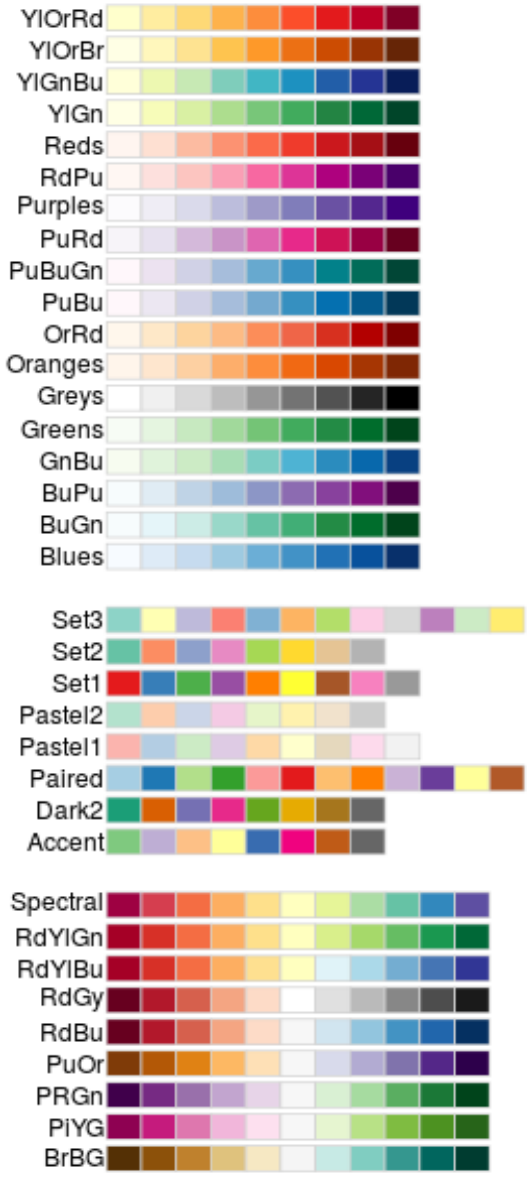
ask	Ask the user before a new figure is drawn
mar	c(bottom, left, top, right), margins on 4 sides of the plot
mfcol	c(nr, nc), number of rows and columns of a multi-paneled plot
mfrow	Figuers will be drawn by row instead by columns
xlog	A local value indicating should a logarithmic scale to be used
bg	background color of the plot

Global Parameters

```
par(mfcol=c(2,2), lty=3, col=rgb(1,0,0, alpha=0.5), bty='n', cex=0.3)
plot(iris$Sepal.Length, type='l')
plot(iris$Sepal.Length, type='l', lty=1)
plot(iris$Sepal.Length, type='l', bty='o')
plot(iris$Sepal.Length, type='l', ann=FALSE)
```

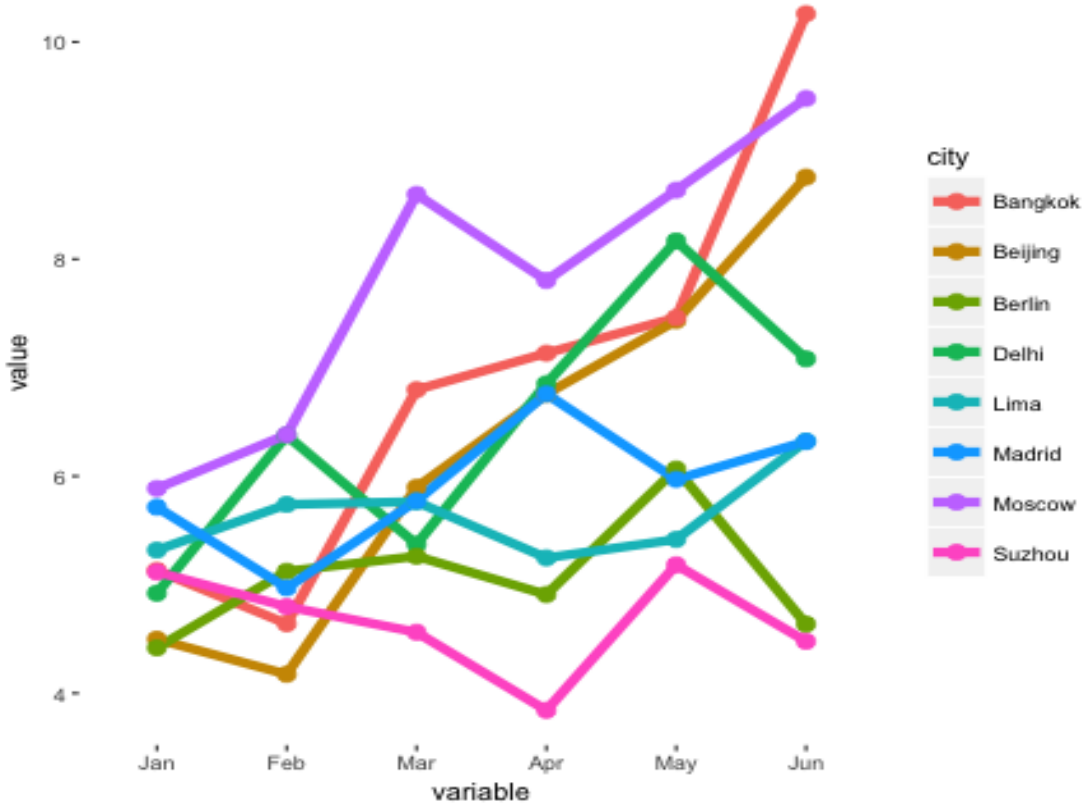


Choose your colors



What if you are drawing 8 lines in a plot and having trouble choosing a sensible colour scheme? RColorBrewer helps you to do this

The main function to extract the colors:
brewer.pal(number, "Setname")



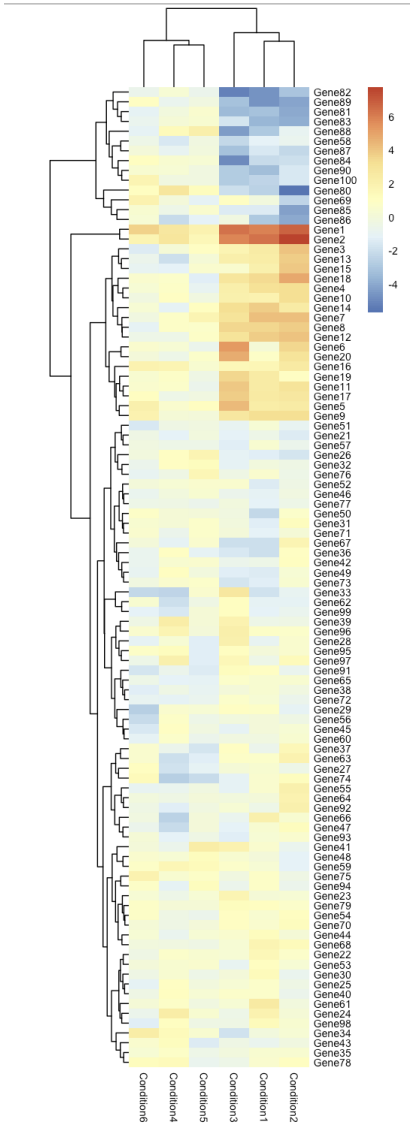
Heatmap

Build a heatmap with Pheatmap package

Three steps twards a nice heatmap:

- Formatting the data into a matrix
- Annotating the rows and columns
- Draw your heatmap using pheatmap() and save it

```
library(pheatmap)
genes <- matrix(rnorm(600), nrow=100)
rownames(genes) <- paste0("Gene", 1:100)
colnames(genes) <- paste0("Condition", 1:6)
genes[1:2,]= genes[1:2,]+3
genes[1:20, 1:3]= genes[1:20, 1:3]+3
genes[80:90, 1:3]= genes[80:90, 1:3]-3
pheatmap(genes, cellwidth = 20, fontsize = 7)
pheatmap(genes, filename = 'Myheatmap.pdf')
```



Heatmap

Need even more colors?

Usually 8 colors are not enough to represent the tinest changes of values in a heatmap, function colorRampPalette() can help us to convert hand-designed color schemes into color ramps with finer scales

```
library(grDevices)
Finer_color = colorRampPalette(brewer.pal(n = 7, name = "RdBu"))(100)
Mymatrix=matrix(1:5000, nrow=10)
pheatmap(Mymatrix, cellheight = 10, color=brewer.pal(n = 7, name='RdBu'),
         cluster_rows = FALSE, cluster_cols = FALSE,
         main = 'Heatmap with 7 individual colors')
pheatmap(Mymatrix, color = Finer_color, cellheight = 10,
         cluster_rows = FALSE, cluster_cols = FALSE,
         main = 'Heatmap with 847 individual colors')
```