

# Introduction to R and BioConductor

Sini Rautio (Originally by Leo Lahti), edited by Viivi Halla-aho

November 2, 2018

## 1 Starting with R/BioConductor

### Why use R?

See <http://monkeysuncle.stanford.edu/?p=367>

### Introductory material

R reference card: a useful and brief 4-page intro: <http://cran.r-project.org/doc/contrib/Short-refcard.pdf>

Recommended book with example source codes online: <https://www.bioconductor.org/help/publications/books/bioinformatics-and-computational-biology-solutions/>

The project pages <http://www.r-project.org> and <http://www.bioconductor.org> provide extensive resources.

### R-Matlab dictionary

If you already know Matlab this may help: <http://www.math.umaine.edu/~hiebler/comp/matlabR.pdf>

### Search engine for R

Seeking R-related information from www is challenging. This solves the problem: <http://www.rseek.org/>

### Graphics

R has excellent graphics capabilities with publication quality. See graphics galleries for *examples with source code* (!):

<https://www.r-graph-gallery.com>

### R and L<sup>A</sup>T<sub>E</sub>X

Automated L<sup>A</sup>T<sub>E</sub>X document generation with parameters and plots:

<https://stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf>

This document was produced with Sweave. The source (.Rnw) is provided at course home page, and can be converted into L<sup>A</sup>T<sub>E</sub>X code in R with the command 'Sweave('handout.Rnw')'.

### Graphical User Interfaces (GUIs) for R?

Miscellaneous GUIs for R, in case you need one:

- RStudio (Recommended)
- Tinn-R
- R Commander (basic-statistics GUI)
- Rattle (data mining)
- RKWard (KDE libraries)
- JGR (universal Java GUI)
- PMG

## 1.1 Installing packages in Aalto computer system

If you prefer to use your own laptop, you have to install R (<https://www.r-project.org>) and BioConductor (<https://bioconductor.org/install/>).

R can be started from command line by typing 'R'.

It is often necessary to install new packages with specific functionality. In Aalto computer system, the quickest option is to install to your home directory (if you have enough disk space). For example the following commands can be used to install packages within R:

```
> install.packages("MASS")
```

or from source tarball

```
> install.packages("mypackage.tar.gz", repos=NULL)
```

To get more information on any specific named function:

```
> ?install.packages
```

**If you are using the Aalto Windows machines, there will be no need to install packages needed in this course yourself as we have requested Aalto IT to do the installations.**

## 2 Hands-on exercises

```
> print("Hello, world! :)")
```

### 2.1 Numbers and vectors

Set up a vector called x

```
> x <- c(1, 1.3, 5, 4.8, 6)
> length(x)
> x[2]
```

Calculate a mean of a vector x

```
> mean(x)
```

Different types of vector

```
> a <- c(1, 2, 5.3, 6, -2, 4) # numeric vector
> b <- c("one", "two", "three") # character vector
> c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE) #logical vector
```

Make a sequence

```
> s<-seq(-5, 5, by=.2)
> s
```

Generate 5 x 4 numeric matrix

```
> y<-matrix(1:20, nrow=5,ncol=4)
> y
> y[,4]
> y[3,4]
```

## 2.2 Random samples and plotting

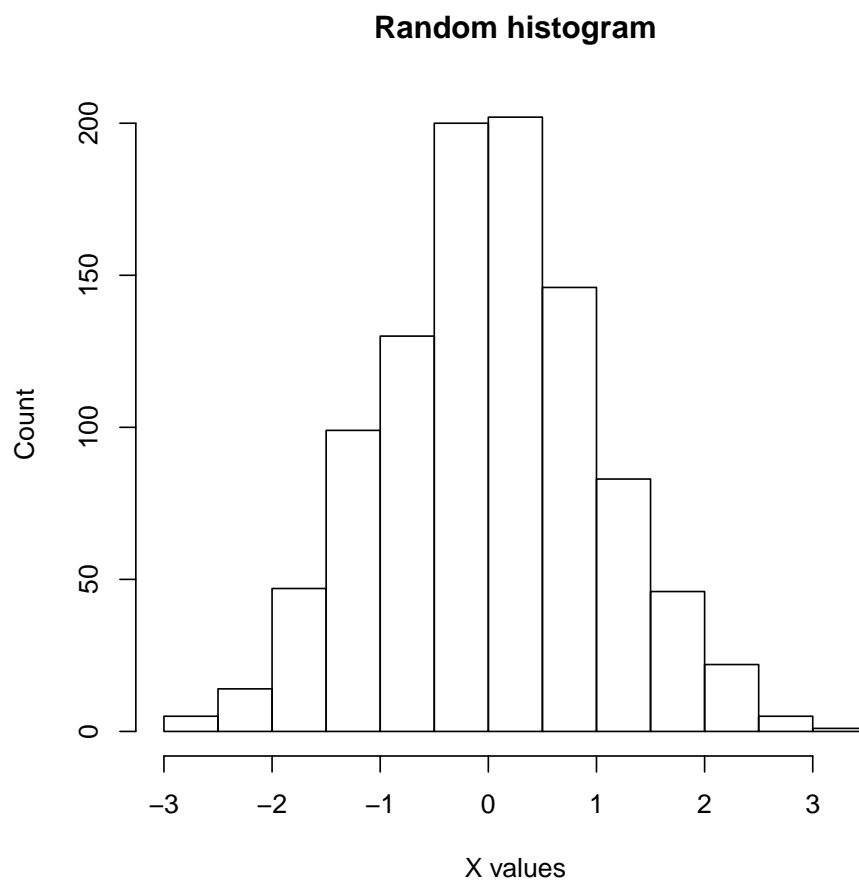
Random samples from normal distribution and a histogram:

```
> set.seed(123) # set random seed
> random.data <- rnorm(1000,mean=0,sd=1)
> # Plot historam of the data
> hist(random.data)
```

(Note: *rnorm* generates random samples from normal distribution. If you don't set random seed (*set.seed*), the code will draw different random data at each run. By setting random seed, you can retrieve the same random data later on, if necessary. This is standard procedure which allows exact reproducibility of the results.)

Define title and axes:

```
> hist(random.data,main="Random histogram",xlab="X values",ylab="Count")
```



See histogram function in more detail: type `'?hist'` or `'help(hist)'`.

## Example data set

Investigate example data:

```
> # Look at the data
> iris
> dim(iris)
> dimnames(iris)
> summary(iris)
```

Boxplot of the first four columns.

```
> # Boxplot
> boxplot(iris[,1:4])
```

Compare sepal length between two flower species with t-test:

```
> setosaRows = (iris[,"Species"]=="setosa")
> virginicaRows = (iris[,"Species"]=="virginica")
> x = iris[setosaRows,"Sepal.Length"]
> y = iris[virginicaRows,"Sepal.Length"]
> t.test(x,y)
```

Welch Two Sample t-test

```
data:  x and y
t = -15.386, df = 76.516, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.78676 -1.37724
sample estimates:
mean of x mean of y
   5.006    6.588
```

## Principal component analysis (PCA)

Investigate variable contents with `'names'`, call subvariables with `'$'` or `'@'`:

```
> # Principal components analysis
> iris.pca <- prcomp(iris[,1:4])
> names(iris.pca) # check result contents

[1] "sdev"      "rotation" "center"   "scale"    "x"

>
```

Plot variance along each principal component

```
> barplot(iris.pca$sdev)
```

Plot first principal component

```
> barplot(iris.pca$rotation[, "PC1"], main="PCA")
```

Plot data for two random features, and then along the most important principal components:

```
> par(mfrow=c(2,1))
> plot(iris[, 1:2], main="Two of the original features")
> plot(iris.pca$x[, 1:2], main="Two PCA features")
```

### Plots with different colors

```
> plot(iris[iris[, "Species"] == "setosa", 1:2], col="red", xlim=range(iris[, 1:2]), ylim=range(iris[, 1:2]))
> points(iris[iris[, "Species"] == "virginica", 1:2], col="blue")
```

Order and plot correlations between 20 random samples using first 4 features:

```
> random.samples <- sample(nrow(iris), 20)
> flower.correlations <- cor(t(iris[random.samples, 1:4]))
> heatmap(flower.correlations, scale = "none")
```

### Produce PDF

This will save 'myFigure.pdf' in your working directory (check with 'getwd()')

```
> pdf("myFigure.pdf")
> heatmap(flower.correlations, scale="none", col = grey(seq(0, 1, length=100)))
> dev.off()
```

Remember to close the output stream with 'dev.off()'

### K-means

Based on the plot it seems that there are two groups of flowers. Use K-means to detect these clusters. It turns out that the clusters are explainable by flower species:

```
> km <- kmeans(iris[random.samples, 1:4], centers = 2)
> names(km)
> iris[names(which(km$cluster == 1)), "Species"]
> iris[names(which(km$cluster == 2)), "Species"]
```

### Hierarchical clustering

```
> d <- dist(iris[, 1:4])           # distance matrix
> hc <- hclust(d, method = "ave") # hierarchical clustering
> plot(hc, hang = -1)             # plot
```

Check source code of the function with 'hclust'.

## 3 FAQ

### How to create multidimensional matrices?

```
> a <- array(0, dim=c(10, 4, 2))
```

### How to run R scripts from source file?

```
> source("myscript.R")
```

## How to run R scripts on command line (batch mode)?

Something along these lines:

```
#!/bin/sh  
/your.R.path/R CMD BATCH --no-save myscript.R myscript.Rlog
```

## Why my figure does not pop up?

Typically the previous output stream has been *opened* but *not closed*. Close output stream(s) using `dev.off()`, this may need to be applied several times.

## 4 Advanced tips, tricks & hacks

If you already know R, check the following packages:

- Matrix (improved matrix operations capabilities)
- plyr (efficient data manipulation)
- ggplot2 (high-quality graphics tools)
- tikzdevice (R and L<sup>A</sup>T<sub>E</sub>Xgraphics combined)
- <http://cran.r-project.org/web/views/HighPerformanceComputing.html> (parallel computing etc.)