# Introduction to R

Daniel Berglund

9 November 2017

# R

- R is available at the KTH computers
- If you want to install it yourself it is available at `https://cran.r-project.org/`
- Rstudio an IDE for R is available at `https://www.rstudio.com/products/RStudio/`
- There is an introduction to R starting at page 42 in the book
- There is a reference between MATLAB and R function names at `https://cran.r-project.org/doc/contrib/Hiebeler-matlabR.pdf`
- Google is your friend. There are plenty of packages available for R which is one reason for Rs popularity

# Basic Commands and Functions

- Can write commands directly into the console but recommened to create a script to save them
- Get information about a function with *help(function)* or *?function*

# Basic Commands and Functions

- Can write commands directly into the console but recommened to create a script to save them
- Get information about a function with *help(function)* or *?function*
- Comments are made with #
- Assignment is done with <- or =
- Vectors are created with the function *c*

```
x = c(1, 2, 3)
```

- We can then apply functions on the vector. For instance the length of the vector is given by:

```
length(x)
```

# Matrices

- Matrices are created with *matrix*
  The code below creates a matrix with 2 rows and 3 columns

```
x = matrix(data = c(1, 2, 3, 4, 5, 6), 2, 3)
```

- It will be filled row by row so the first row is 1,2,3

# Vector and Matrix operators

- Access a vector or matrix element by using [ ]. The index starts at 1

```
x = c(1, 2, 3)
x[1]
```

- By inputing vectors to the [ ] operator we can access several elements at once

```
X[ c(1 ,3), c(2 ,4) ]
```

- The first vector tells which rows to get and the second the columns
- So c(1,3), c(2,4) will return the positions 1,2 1,4 3,2 3,4

# Vector and Matrix operators

- Possible to add, subtract matrices and vectors etc

```
x = c(1, 2, 3)
y = c(4, 5, 6)
z = x+y
```

# File Reading

- To read a file containg data use *read.table()* or *read.csv()*
- These functions have plenty of options to set how to handle delimiters, missing values etc
- For instance if we want to read the Iris flower dataset we can use

```
irisData = read.csv('pathToIrisData.csv',
                    header = TRUE, sep = ",")
```

- The Iris data has now been stored in a *data frame*

# Data Frames

- Data Frames are a common data structure in R
- In some ways they behave similair to a matrix, we can for instance access column number two with

```
irisData[, 2]
```

- However we can also access it by using $ and its column name, in this case sepal_width

```
irisData$sepal_width
```

- Some functions need data frames as input while others need matrices. It can also depend on the data types inside the data frame

# Data Frame Example

- In this example we will do a t-test between the sepal width variable of the two classes Setosa and Virginica
- First we need to get the data. The rows below return two vectors with TRUE and FALSE based on the class

```
irisVirginicaBool = irisData$class=='Iris-virginica'
irisSetosaBool = irisData$class=='Iris-setosa'
```

- We can use the boolean vectors to get the rows containing the data of the two classes by using the index operator [ ]

```
irisVirginica = irisData[irisVirginicaBool,]
irisSetosa = irisData[irisSetosaBool,]
```

- The test itself is then done with:

```
t.test(irisVirginica$sepal_width,
irisSetosa$sepal_width, var.equal=FALSE,
paired=FALSE)
```

# Graphics

- Plots can be done with *plot*. It has various options for labels, colors, etc
- We can save the plot as pdf by using:

```
pdf('Figure.pdf')
plot(x,y)
dev.off()
```

- *dev.off()* tells R that we are done with the plot so it writes it to the file

# Functions

- We can also write our own functions
- They are formated as:

```
functionName = function(arguments)
{
  code
}
```

- The function below computes the square of the input argument

```
square = function(x)
{
  return x*x
}
```

# Packages and Libraries

- Packages and libraries for R can be found on the Cran webpage
- Installed with *install.packages()*
- They are loaded with *load()*
- They stay loaded until unloaded or the workspace is cleared

# Bootstrap

- Bootstrap can be done with help of the library *boot*

```
library("boot")
```

```
boot(data=irisData, statistic=func, R=1000)
```

- The statistic is which function to run on each bootstrap sample. R sets the number of bootstraps. In this case we will do 1000 bootstraps using the statistic from the function func on the dataset irisData
- The function takes two arguments. The bootstrapped data and the index of the current bootstrap

# Bootstrap

- To do the equivalent bootstrap of the t-test we can define the function as

```
func = function(data,b){
  d = data[b,]

  dVirginica = d[irisData$class=='Iris-virginica',]
  dSetosa = d[irisData$class=='Iris-setosa',]

  return(mean(dVirginica$sepal_width)-
  mean(dSetosa$sepal_width))
}
```

# Exercises

- The best way to familiarize yourself with a new language is to use it
- The book has applied exercises on R in pages 54 to 57