An Introduction to R

Beginners session + Q&A

R-Ladies London team | @RLadiesLondon

26th October 2016

What's R-ladies?

R-ladies is a world-wide organization to promote gender diversity in the R community: https://rladies.org/

- San Francisco, California, USA
- Twin Cities, Minnesota, USA
- London, England, UK
- Research Triangle Park, North Carolina, USA
- Istanbul, Turkey
- Paris, France
- Los Angeles, California, USA
- Melbourne, Australia
- Madrid, Spain
- Nashville, Tennessee, USA
- New York, New York, USA
- Barcelona, Spain
- Boston, Massachusetts, USA
- Columbus, Ohio, USA
- Izmir, Turkey
- Lima, Peru
- Berlin, Germany
- Taipei, Taiwan

What's R?

In origin was **S**, a programming language for statistical computing and interactive graphics. It was developed by John Chambers, Rick Becker and Allan Wilks of (NOKIA) Bell Laboratories in 1976.

S went through many version updates (1-4, 5 plus…) until in 1992 Ross Ihaka and Robert Gentleman (University of Auckland, New Zealand) worked on a further implementation and renamed it **R**.

R is currently developed by the **R Development Core Team**, of which Chambers is a member. It is a Free Software, available under a GNU General Public License. It compiles and runs on a wide variety of platforms (including Linux, Windows and MacOS).

(Wikipedia)

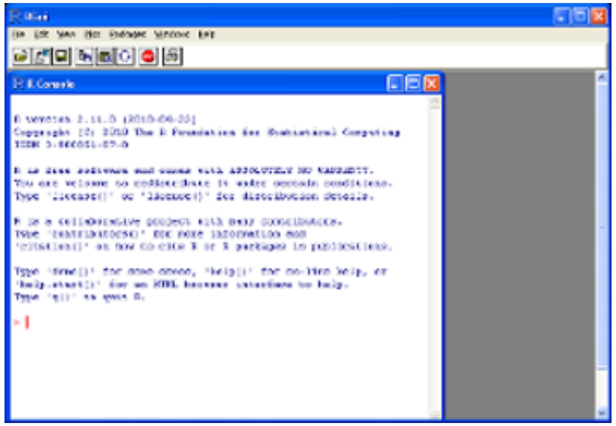
Let's install the software

Install R

Go to the R-project website, download and install R:

https://www.r-project.org/

The default Graphical User Interface (GUI) is basically a console!



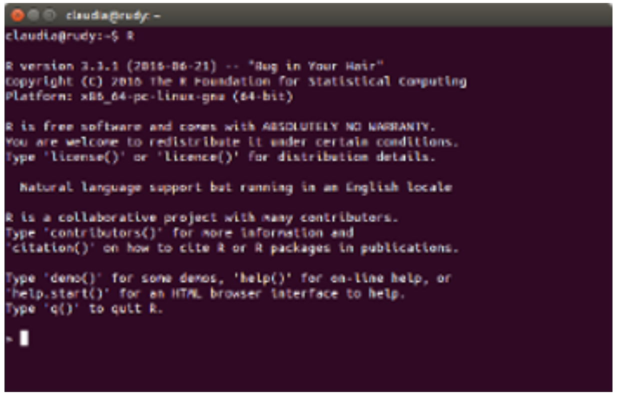
```
R version 2.11.0 (2010-04-22)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```



```
claudia@rudy:~$ R

R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Using the prompt/console

To start R from command line, type `R` then hit enter (ONLY FOR LINUX/MAC USERS!).

When R is waiting for us to tell it what to do, it begins the line with the symbol:

```
>
```

If we give it an incomplete command and it can not finish the task requested it provides:

```
+
```

To get out of R we use the command

```
q()
```

Your turn!

Try the basic GUI (console)

- Open an R instance
- Top line in the opening message: **R version number (release date) and name**
- Try the functions **license()**, **help()** and **q()**
- **For complex operations you might need to have an editor, check your environmental variables, visualise plots without switching windows... that's why we recommend to use RStudio!**

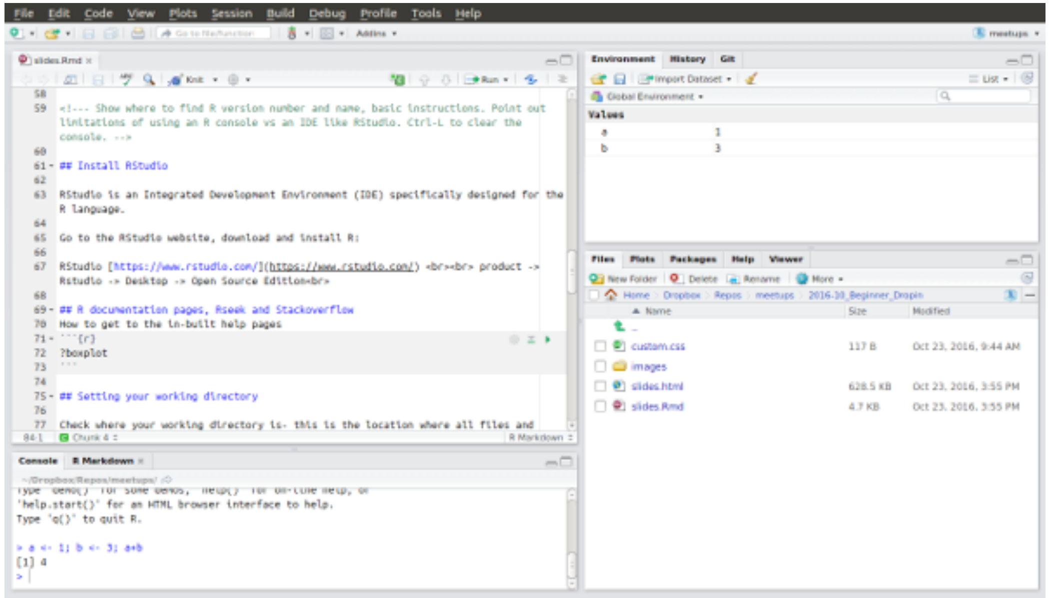
Install RStudio

RStudio is an Integrated Development Environment (IDE) specifically designed for the R language.

On the RStudio website (https://www.rstudio.com/)

go to **product -> Rstudio -> Desktop -> Open Source Edition**

download and install RStudio Open Source Edition:



File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
meetups
slides.Rmd
Knit
Run
Environment History Git
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home Dropbox Repos meetups 2016-10_Beginner_Dropin
Chunk 4
R Markdown
Console R Markdown
~/Dropbox/Repos/meetups/

~~~

<!--- Show where to find R version number and name, basic instructions. Point out limitations of using an R console vs an IDE like RStudio. Ctrl-L to clear the console. -->

Install RStudio

RStudio is an Integrated Development Environment (IDE) specifically designed for the R language.

Go to the RStudio website, download and install R:

RStudio https://www.rstudio.com/

 product -> Rstudio -> Desktop -> Open Source Edition

R documentation pages, Rseek and Stackoverflow
How to get to the in-built help pages
```{r}
?boxplot
~~~

## Setting your working directory

Check where your working directory is- this is the location where all files and
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> a <- 1; b <- 3; a+b
[1] 4
>
```

Values

a	1
b	3

Name	Size	Modified
..		
custom.css	117 B	Oct 23, 2016, 9:44 AM
images		
slides.html	628.5 KB	Oct 23, 2016, 3:55 PM
slides.Rmd	4.7 KB	Oct 23, 2016, 3:55 PM
```

# Basic operations, like a calculator

R can be used as a calculator.

You can enter an expression to evaluate and hit enter (if you use the command line) or click the Run button (if you use Rstudio) to compute the result.

- Use ***** for multiply.
- Use ^ for raised to the power of.
- Use **parentheses** to ensure that it understands what you are trying to compute.
- The order of doing arithmetic operations is (left [done first] to right [done last]): **^ / * - +**

# Your turn!

# Basic operations, like a calculator

```
1 + 2
```

```
## [1] 3
```

```
5 * 6 + 9 * (10 ^ (-2) + 2 * 3)
```

```
## [1] 84.09
```

Literal operations

Like in algebra, we may want to store a computation under some variable name. The result is assigned to a variable with the <- ('less than' symbol followed by a hyphen):

```
a <- 1
```

- When you want to know what is in a variable simply ask by typing the variable name.

```
a
```

```
## [1] 1
```

- We can store a computation under a new variable name or change the current value in an old variable.

```
b <- 2
b <- exp(b)
```

Something to remember

- Certain variable names are reserved for particular purposes (e.g. c q t C D F I T)
- Do not begin a variable name with a period or a number.
- Variable names are case (upper/lower) sensitive.

Data types and objects

R supports 5 basic data types:

- integer,
- numeric,
- character,
- logical and
- complex.

Missing values are labelled: NA.

Elements of basic type can be combined to form complex data objects such as:

- vectors,
- matrices,
- arrays,
- data frames and
- lists.

Objects

A **vector** is a sequence of data elements of the same basic type. To create a vector we can concatenate a list of elements using the function **c()**.

A **matrix** is a 2 dimensional table in which every element is of the same type. To create a matrix we use the function **matrix()**.

An **array** is similar to a matrix but can have more than 2 dimensions. To create an array we use the function **array()**.

A matrix in which each column can be of a different type is called **data frame** and can be created using the function **data.frame()**.

A **list** is like a data frame but each column can be of different length or even a tree of data objects. A list is generated using the function **list()**.

Your turn!

Let's experiment with data objects

```
a   <- 1                                                   # single variable

v1  <- c(1,2,3)                                            # vector
# Get the length of a vector
length(v1)
```

```
## [1] 3
```

```
m1  <- matrix(0, nrow = 3, ncol = 2)                       # matrix
# Count dimensions
dim(m1)
```

```
## [1] 3 2
```

```
df1 <- data.frame(v1, v1 * 10)                             # data frame

l1  <- list("a" = a, "v1" = v1, "m1" = m1, "df1" = df1)    # list
# What would you use to count the elements of a list?
```

Explore data objects

```
typeof(l1)
```

```
## [1] "list"
```

```
str(l1)
```

```
## List of 4
##  $ a  : num 1
##  $ v1 : num [1:3] 1 2 3
##  $ m1 : num [1:3, 1:2] 0 0 0 0 0 0
##  $ df1:'data.frame': 3 obs. of  2 variables:
##   ..$ v1     : num [1:3] 1 2 3
##   ..$ v1...10: num [1:3] 10 20 30
```

Extract elements from data objects

Data objects in R are indexed. These indices can be used to extract/subset vectors, matrices, data frames and lists. Alternatively, named dimensions can be extracted using the operator **$**

```
v1[2]                 # extract the second element of vector v1
l1[2]                 # extract the second element of list l1
m1[3,2]               # extract the element in the third row second column of m1
df1[3,2]              # extract the element in the third row second column of df1
```

```
l1$v1
```

```
## [1] 1 2 3
```

R Base + core packages

All the basic R functions and operators seen so far are automatically loaded under the **R Base Package** (base).

There are additional built-in functions that are loaded grouped in separated packages. Each package is used for a specific purpose.

Some examples are: **base**, **stats**, **graphics**, **datasets**, etc.

Built-in functions

```r
x <- c(1,2,3,4,5,6,7,8,9)

# Core functions from the "base" package
print("Hello world!")   # Print messages
sum(x)      # calculates the sum of the elements in the vector x
mean(v1)    # average
max(x)      # the largest value
min(x)      # the smallest value
log(b)      # natural logarithm (log10 computes logarith with base 10)
sort(x)     # re-arrange elements of x in ascending order
summary(v1) # summary statistics

# Core functions from the "stats" package
median(x)   # the sample median
var(x)      # the sample variance (has n-1 in denominator)
sd(x)       # the standard deviation
```

NAMESPACE

```
stats::cov
cov
```

But… be careful!

```
cov <- function(x){x+1}
cov
stats::cov              # the original cov function is not lost!
rm(cov)                 # remove the last definition
```

Setting your working directory

Check where your working directory is- this is the location where all files and functions will be read and written to

```
getwd()
```

```
## [1] "/home/claudia/Dropbox/Repos/meetups/2016-10_Beginner_Dropin"
```

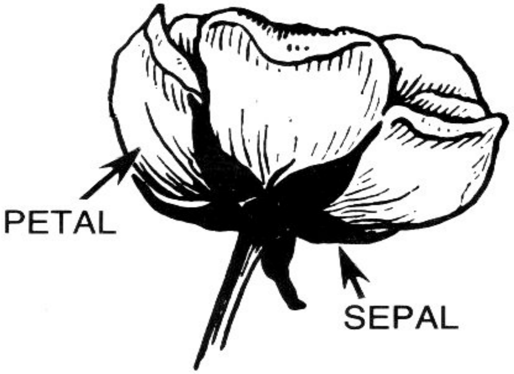
If this isn't where your files are then you can set it through (put your own path in the quotes!)

```
setwd("C:\Documents\mypath")
```

Or in Rstudio you can go to: Session > Set Working Directory > Choose Directory and navigate to your folder

Example data

R comes with a number of example datasets. To browse the complete list use the function **data()**, with no inputs.



For example, the **iris** dataset consists of 50 samples from three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Look at the in-built data set on trees

```
trees
```

```
##    Girth Height Volume
## 1    8.3     70   10.3
## 2    8.6     65   10.3
## 3    8.8     63   10.2
## 4   10.5     72   16.4
## 5   10.7     81   18.8
## 6   10.8     83   19.7
## 7   11.0     66   15.6
## 8   11.0     75   18.2
## 9   11.1     80   22.6
## 10  11.2     75   19.9
## 11  11.3     79   24.2
## 12  11.4     76   21.0
## 13  11.4     76   21.4
## 14  11.7     69   21.3
## 15  12.0     75   19.1
## 16  12.9     74   22.2
## 17  12.9     85   33.8
## 18  13.3     86   27.4
## 19  13.7     71   25.7
## 20  13.8     64   24.9
## 21  14.0     78   34.5
## 22  14.2     80   31.7
## 23  14.5     74   36.3
## 24  16.0     72   38.3
```

Explore the trees data set

The top of the data

```
head(trees)
```

The end of the data set

```
tail(trees)
```

The size and type of the data

```
str(trees)
```

Summary statistics on each of the fields

```
summary(trees)
```

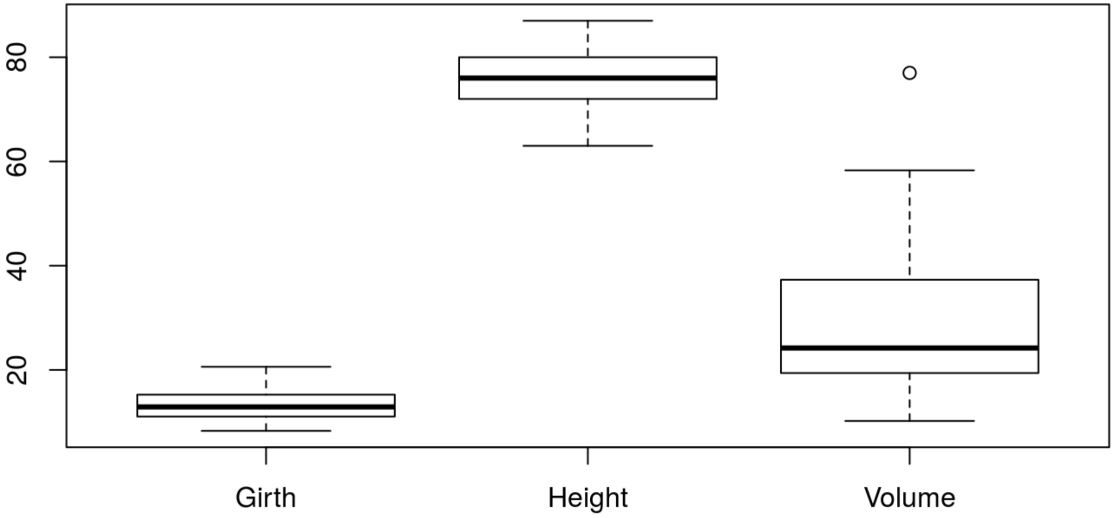
Pull out only one of the fields

```
summary(trees$Girth)
```

Basic plots

Create your first plot

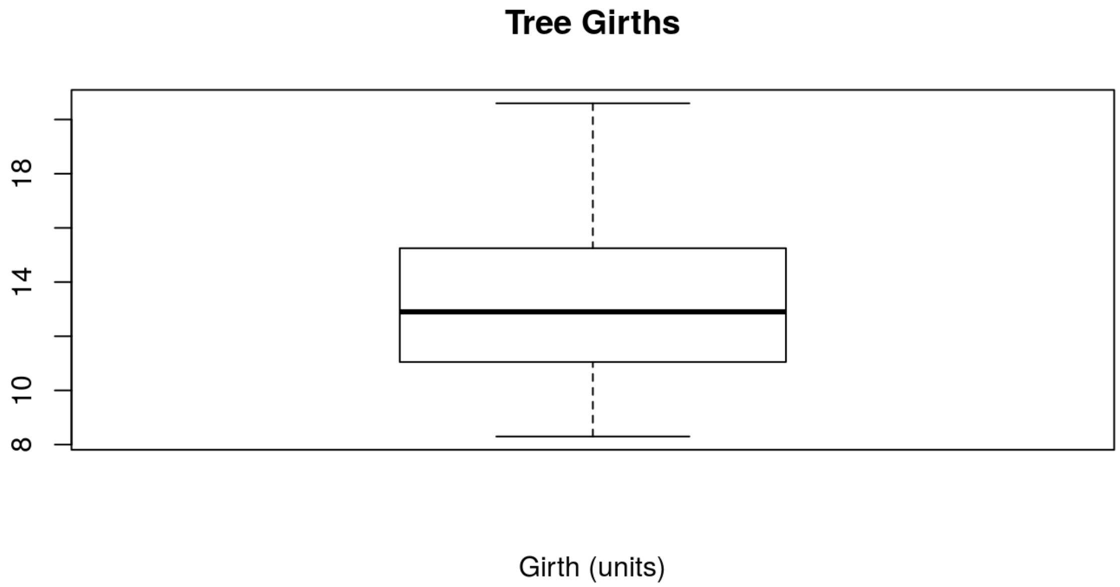
```
boxplot(trees)
```



Basic plots

Create another variable with only one of the fields and plot that

```
boxplot(trees$Girth, xlab = "Girth (units)", main = "Tree Girths")
```



Export/Import a table (data.frame) to/from text file

```
write.csv(x = iris, file = "iris.csv")
```

```
myTable <- read.csv(file = "iris.csv")
```

```
head(myTable)
```

Work with remote files

```
myTable2 <- read.csv(file = "http://www.ats.ucla.edu/stat/data/hsb2.csv")

head(myTable2)
```

```
##    id female race ses schtyp prog read write math science socst
## 1  70      0    4   1      1    1   57    52   41      47    57
## 2 121      1    4   2      1    3   68    59   53      63    61
## 3  86      0    4   3      1    1   44    33   54      58    31
## 4 141      0    4   3      1    3   63    44   47      53    56
## 5 172      0    4   2      1    2   47    52   57      53    61
## 6 113      0    4   2      1    2   44    52   51      63    61
```

R data formats

Rds stores a single R object, use readRDS() and saveRDS() to read in and save respectively.

```r
## save a single object to file
pippo <- c(1,2,4)
saveRDS(object = pippo, file = "pippo.rds")

## restore it under a different name
pippo2 <- readRDS(file = "pippo.rds")
```

RData (or rda) allows to save(), load(), attach() multiple data objects in one file.

```r
## save two objects to one file
pippo <- c(1,2,4)
pluto <- c(3,5,4)
save(pippo, pluto, file = "x.rda")

## restore the objects
load(file = "x.rda")
```

Other data formats

There are dedicated R packages to handle the most commonly used data formats.

Some examples are:

- Text files (e.g. csv)
- Spreadsheets (e.g. .xls)
- GIS files (e.g. .shp, ascii, .tif)
- Binaries (e.g. .nc)
- Markups (e.g. xml, gml)

R can also connect to databases (e.g. postgresql).

Install additional packages

There are ~8000 R packages available on the Comprehensive R Archive Network (CRAN).

```
# Install a new package for advanced graphics
install.packages("ggplot2")

# Load the package
library("ggplot2")
```

Custom functions

You can create a custom function as in the example below:

```
myFunction <- function(x){

   y <- x + 1

   return(y)
}

myFunction( x = 32 )
```

```
## [1] 33
```

Help

Each function comes with a documentation page that can be visualised typing **help(*name of the function*)** in the console. Alternatively, you can type **?** (question mark) before the name of the function.

```
help(print)
?print
```

You can also:

- Browse Rseek to find out which packages are available for a given topic (e.g., cluster analysis)
- Join the R users forum
- Google your problem
- Post a question on stack overflow

Cheat Sheets & Reference Guides

- R Reference Card
- Writing R extensions
- Google's R Style Guide
- RStudio website
  - Data Visualization
  - Package Development
  - Data Wrangling
  - R Markdown
  - R Markdown Reference Guide
  - Shiny

Where to go next

Great tutorials:

- edx MiT course: https://www.edx.org/course/analytics-edge-mitx-15-071x-2
- DataCamp: https://www.datacamp.com/
- Coursera: https://www.coursera.org/learn/r-programming
- Great Kaggle Tutorials: https://www.kaggle.com/mrisdal/titanic/exploring-survival-on-the-titanic