Introduction: Statistical Computing with R

Part 1: basics and vectors

Complied by: Birgit Erni (amended by Allan Clark)

Department of Statistical Sciences, University of Cape Town

01 February 2022

Objectives: Introduction to statistical computing with R

- manipulate/organize data and data files
- use R for summary statistics, fitting statistical models
- use R to create quality graphics
- use the R language for programming/writing your own custom functions
- use simulations to solve some statistical problems (Monte Carlo and bootstrap)
- optimize functions using R
- do parallel computing in R

Today

- R and RStudio
- basic operations
- script files, R Markdown
- data types
- data objects: vectors, matrices and data frames

What is R?

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or in print, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.
  (R is an objected orientated language: more on this later)

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

Why R?

- R is free
- R creates better graphics than Excel
- R has a broad range of statistical packages for doing specialist work
- R has certain data structures such as data frames that can make analysis more straightforward than Excel
- R is better for doing complex jobs

RStudio

We will use R within the Integrated Development Environment (IDE) RStudio.

IDE:
A software application that provides comprehensive facilities to computer programmers for software development. (*wikipedia*)

IDEs typically have tools for

- automatic code completion
- code documentation
- code compilation and testing

RStudio is a good general purpose IDE. There are also specialized IDEs... eg. Rattle is a good IDE for data mining.

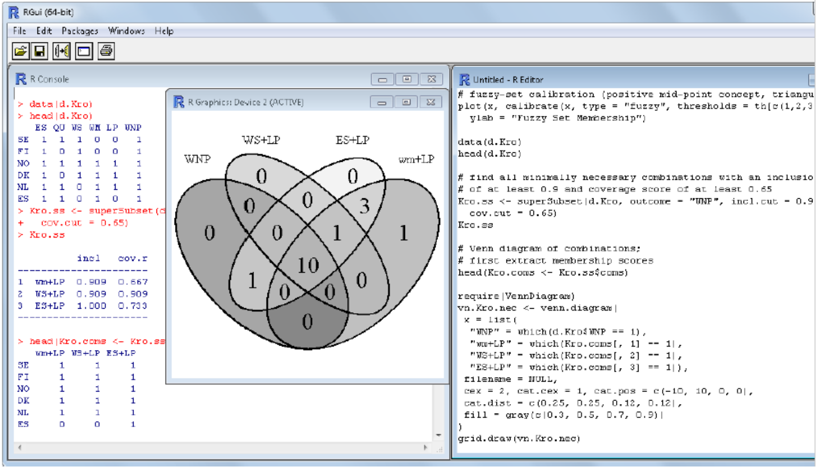
Downloads

- R - http://cran.r-project.org/
- Download R for Windows
- Select the “base” installation
- Follow instructions to install R.
- Download RStudio as well https://www.rstudio.com/

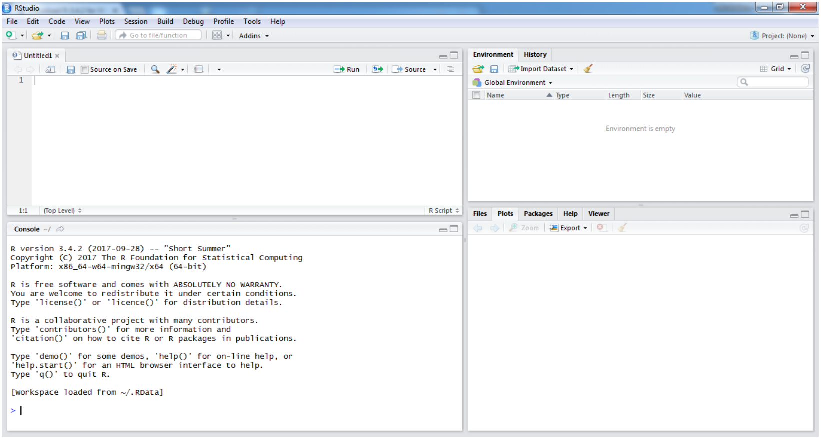
Resources

- Getting Started with R - FAQ.pdf (Vula)
- Roger Peng's "R Programming for Data Science" is an easy book to follow.
- Some programming tutorials.
  - http://zoonek2.free.fr/UNIX/48_R/02.html is accessible.
  - http://cran.r-project.org/doc/manuals/R-intro.html provides examples to work through.
  - Lumley's course http://faculty.washington.edu/tlumley/Rcourse/ is also good.
- Some other resources
  - Harry R Erwin, PhD, "Programming in R"
  - Hung Chen, "R programming"
  - Google "Hadley Wickam", "Hastie", . . . , there are many good books available.

The R Graphical User Interface (GUI)



The RStudio GUI



RStudio GUI main features

Feature	Description
Script window	Used to develop programs and display R script
Integrated R console	Commands typed here can be executed directly by pressing *enter* (Console window).
Object browser	A window listing objects defined in a current R session (Environment window).
History browser	Window showing script of commands used in the past.
File browser	Browser similar in function and form to windows file browser.
Graphics browser	Window allowing one to scroll through, zoom, manipulate and export graphics interactively.
Packages browser	List of packages in user library, browser also facilitates installation of new packages.
Help browser	Window that allows search and display help information on functions.
Viewer	

Object orientation

- R stores data, functions, and output from data analysis (as well as everything else) as objects.
- Things are assigned to and stored in objects using the <- or = operator.
- A list of all objects in the current session (local level environment) can be obtained with ls().

Nice short and easy explanation of Object-Orientated-Languages;

https://www.techopedia.com/definition/8678/object-oriented-language-ool

Data types (types of values)

data types in R are:

- numeric (integer, double, complex)
- character
- logical (FALSE, TRUE, F, T) - don't assign objects with these object names!

Out of these, vectors, arrays, lists can be built (data objects)

and other things: dates

Before we start further work

Comments in R are created using #. e.g

```
# generate 1000 values from N(0,1)
x <- rnorm(1000)          # note round brackets
```

Set your working directory before you start working:

```
#use the setwd function
#help for a function is ?name of the function

#?setwd

#setwd(" add your path in here")
```

We normally save all workspaces, R scripts and data files to be used in the working directory.

Style guide

Google's R Style Guide:
https://google.github.io/styleguide/Rguide.xml

R nuts and bolts

The <- symbol is the assignment operator. Its the same as =.

```
x <- 1
print(x)
```

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

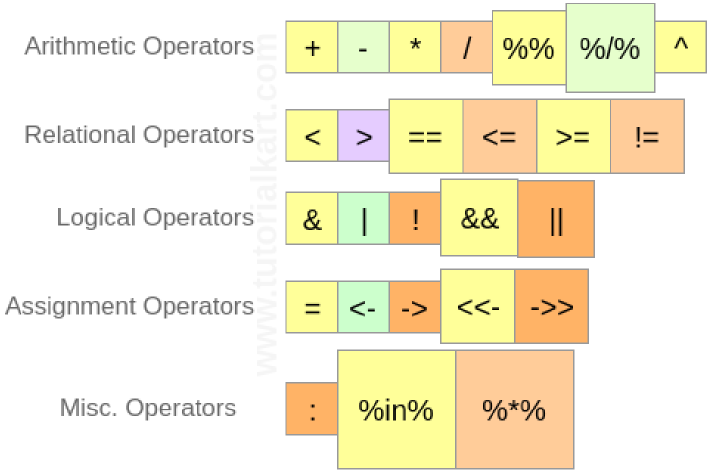
```
x
```

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

```
msg <- "hello"
msg
```

```
## [1] "hello"
```

R Operators


Arithmetic Operators
+
-
*
/
%%
%/%
^
Relational Operators
<
>
==
<=
>=
!=
Logical Operators
&
|
!
&&
||
Assignment Operators
=
<-
->
<<-
->>
Misc. Operators
:
%in%
%*%
www.tutorialkart.com

Variables/objects and the assignment operator <-

```r
x1 <- 49
sqrt(x1)
```

```
## [1] 7
```

```r
x2 <- "South Africa"
x2
```

```
## [1] "South Africa"
```

```r
x3 <- x1 == 5      # does x1 equal 5?
x3
```

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

```r
x4 <- date()
```

- Of what data type are the above?
- What does `str()` do?

Constants

```r
pi
```

```
## [1] 3.141593
```

```r
FALSE                 # logical constants
```

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

```r
val1 <- TRUE
val1
```

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

```r
val2 <- F
```

Functions

```r
sqrt(9)
```

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

```r
x.plus1 <- function(x) {
    return(x + 1)
}
x.plus1(3)
```

```
## [1] 4
```

Round brackets around the *arguments*, curly brackets around the function instructions.

Miscellaneous

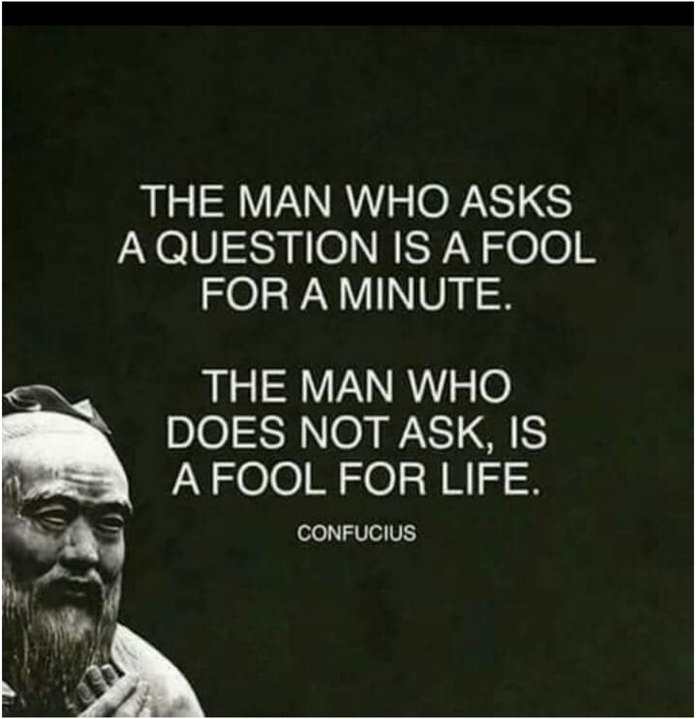
- case sensitive (SAM, sam, Sam are 3 different things)
- names of R ojects must start with a letter or '.', can contain any number, letter, '.', '_', e.g. model.residuals, model2, x, X, x2, house.number.bedrooms are all valid names.

```
is.integer(2)                # double
is.integer(2L)               # integer
```

- NA: missing value (not available)
- NaN: not a number (arithmetically undefined)

Help

```
?rnorm
?"&"
```


THE MAN WHO ASKS
A QUESTION IS A FOOL
FOR A MINUTE.
THE MAN WHO
DOES NOT ASK, IS
A FOOL FOR LIFE.
CONFUCIUS

Object classes (a little more details)

- R stores both data and output from data analysis (as well as everything else) in objects.
- Things are assigned to and stored in objects using the <- or = operator.
- A list of all objects in the current session can be obtained with ls().

The basic classes of objects are:

- numeric (integer, double, complex)
- character
- logical
- function

Object classes

- a numeric object

```
#create object named 'a'.
#Set equal to the value 49.
a<-49
```

- a character string

```
a = "The dog ate my homework"
a #this evaluates 'a' and prints the output to screen
```

```
## [1] "The dog ate my homework"
```

Object classes

- a logical statement

```
a<-49 #Set equal to the value 49.
a==49 #Test if 'a is equal to 49'. a=49 thus FALSE
```

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

```
a > 5 #Test if 'a is greater than 5'. a=49 thus TRUE.
```

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

```
#Result is 'TRUE' since a=49.
```

```
a != 5 #Test if 'a is not equal to 5'
```

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

Object classes

- a function

```
#a simple function
Function1<-function(x)
{
    return( x+1 )
}

Function1
```

```
## function(x)
## {
##  return( x+1 )
## }
```

- The name of the function is Function1.
- The function has one argument named x.
- The function returns x+1.

Object classes

```r
#evaluating the function
Function1(3) #3+1
```

```
## [1] 4
```

```r
Function1(a) #a+1
```

```
## [1] 50
```

```r
Function1(a) + Function1(3) # a+1 + 3+1
```

```
## [1] 54
```

Object classes

- A function can have more than one argument.

```
Function2<-function(x, y)
{
    return( x+y )
}

Function2( 1, 5 )
```

```
## [1] 6
```

Object classes

```
Hypotenuse1<-function(x, y)
{
  sqrt(x^2 + y^2) #the result is returned
}

Hypotenuse1(3, 4)
```

```
## [1] 5
```

Object classes

```r
Hypotenuse2<-function(x, y)
{
  hyp <- sqrt(x^2 + y^2)
  #the result is NOT returned
  #take note
}

Hypotenuse2(3, 4)  #here nothing is outputted

temp <- Hypotenuse2(3, 4)
temp #here something is outputted! but don't do this.
```

```
## [1] 5
```

Classes

```r
a <- 49

Function1<-function(x)
{
    return( x+1 )
}

class(a)
```

```
## [1] "numeric"
```

```r
class(Function1)
```

```
## [1] "function"
```

```r
class("a") #Not the same as class(a)
```

```
## [1] "character"
```

R Markdown

R Markdown is a fantastic way to integrate data analysis with report writing.

Create your own handbook for this course:

- Look at the top of SC1_basics_vectors.Rmd
- **output:** beamer_presentation, html_document, pdf_document, word_document ...
- **Help:** Markdown Quick Reference, Cheatsheet
- **R chunks**
- **markdown:** text to HTML (typesetting with LaTeX formulae, headings, ...)
- document with R code, output and your own comments and annotations
- **reproducible**
- write blogs/websites

Yihui Xie et al. *R Markdown: The Definitive Guide.* https://bookdown.org/yihui/rmarkdown/

Basic Computations

- Open Basics1_1.pdf and work through the file. Later look at the script file; Basics1.r.

Basic Computations

```
2 + 3 * 5      # Note the order of operations.
log(10)       # Natural logarithm with base e=2.718282
4^2            # 4 raised to the second power
15/4           # Division
sqrt(16)      # Square root
abs(3-7)      # Absolute value of 3-7
pi             # The mysterious number
exp(2)         # exponential function
15 %/% 4       # This is the integer divide operation
?"%/%"         # Help on the function
# This is a comment line
```

Data Structures: Vectors

$$x_1 = \begin{pmatrix} 3 & 5 & 6 & 7 & 1 \end{pmatrix}$$

In statistics: typically measurements on a variable are stored in a vector, e.g. here x_1 could be age of 5 children.

Creating vectors in R

```
x1 <- c(3, 5, 6, 7, 1)   # c for combine/concatenate
x2 <- 1:10

## sequence from 0.5 to 2.5, step size 0.5
x3 <- seq(0.5, 2.5, 0.5)
x3b <- seq(from = 0.5, to = 2.5, by = 0.5)   # equivalent

## sequence from 0.5 to 2.5, length 100
x4 <- seq(0.5, 2.5, length = 100)
x5 <- rep(0, times = 4)

## Repeat each of 1 and 2 three times. Note that the first
## argument has to be a single object, hence the use of
## c() to build a vector.
x6 <- rep(c(1, 2), each = 3)
```

Note the way in which R prints vectors to the screen.

Vector calculations

```
#Vectors

x<-c(1,3,2,10,5) #create vector x with 5 components
x

y<-1:5 #create vector of consecutive integers
y

y+2 #scalar addition

2*y #scalar multiplication
```

Basic Computations

```
y #y itself has not been changed

y<-y*2
y #it is now changed
```

(more) Vector calculations

```
#two statements are separated by semicolons
x<-c(1,3,2,10,5); y<-1:5

x+y

x*y

x/y #what is being done here?

x^y #what is being done here?
```

Referencing elements of a vector

```
x #x
x[2] #the second element in x
x[6] #NA since there are only 5 elements in the object 'x'

x[1]+x[5] #referencing elements of a vector
```

Extracting elements of a vector

```
#x = c(1, 3, 2, 10, 5)

length(x) #number of elements in x

x[3] #third element of x

x[3:5] #third - fifth element of x, inclusive

x[-2] #all except the second element

x[x>3] #list of elements in x greater than 3
```

Character vectors

```
colours<-c("green", "blue", "orange", "yellow", "red")
colours

colours[2]
colours[5]

#colours[1] + colours[2] #can't perform numeric operations
```

Good functions to know about!

```
str(x1)       # summary of data type, length, structure
class(x1)
typeof(x1)
is.numeric(x1)
is.logical(x1)
is.integer(x1)
```

Some more calculations with vectors

```
x1 + x1                    # elementwise
x1^2
x1 == 5                    # returns logical vector

x1 + x2                    # recycles shorter vector

length(x1)
sum(x1)
max(x1)
min(x1)
prod(x1)
```

Indexing/subsetting a vector

```r
x1[1]          # first element of vector
x1[-1]         # all but first element
x1[1:3]        # first 3 elements
x1[c(4, 5)]    # 4th and 5th

## which elements = 5, returns indeces/positions
ind <- which(x1 == 5); ind
x1[ind]

x1[x1 > 5]     # only keep elements > 5
```

Prac 1:

1. Open *SC1 basics vectors.Rmd* (in RStudio). Go to Prac 1.
2. Calculate log(i + 1) for i = 0 to 100. The average of these values should = 3.647074.
3. Generate 10000 random values as follows:

```
set.seed(20190128)   # set starting point for random numbe

## generate 10000 values from an exponential distribution,
y <- rexp(10000)
```

4. Find the largest number and its position. (Answer: 10.23488, 6150)
5. How many values are > 2 (absolute and %)? (Answer: 1291, 12.91%)
6. y2: select every 2nd element of y, starting from 1st.
7. y3: replace values > 3 in y2 with 3. The average of these values should = 0.9423062.
8. variance of y3, check with `var()`: 0.6949041.