

(Brief) Introduction to R

Ivan G. Costa

IZKF Computational Biology Research Group

University Hospital Aachen

RWTH Aachen

R Language

- Script based Programming language
- Focus of statistical data analysis
- Open source
- Contributing packages
 - Bioconductor (bioinformatics tools)
 - ggplot (plotting functions)
 - ...



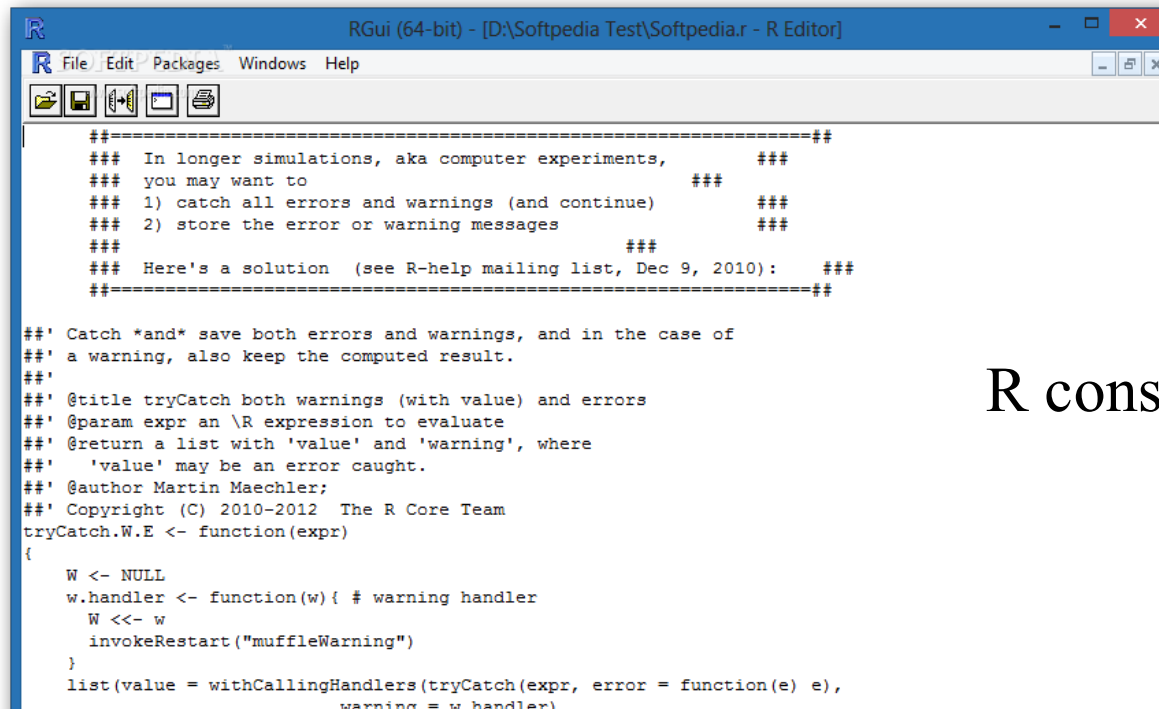
<http://www.r-project.org/>

Getting Started

- Install R

<http://cran.r-project.org/>

- Run R



```
RGui (64-bit) - [D:\Softpedia Test\Softpedia.r - R Editor]
File Edit Packages Windows Help
#####
### In longer simulations, aka computer experiments,      ###
### you may want to                                       ###
### 1) catch all errors and warnings (and continue)       ###
### 2) store the error or warning messages                ###
### Here's a solution (see R-help mailing list, Dec 9, 2010): ###
#####

##' Catch *and* save both errors and warnings, and in the case of
##' a warning, also keep the computed result.
##'
##' @title tryCatch both warnings (with value) and errors
##' @param expr an \R expression to evaluate
##' @return a list with 'value' and 'warning', where
##'   'value' may be an error caught.
##' @author Martin Maechler;
##' Copyright (C) 2010-2012 The R Core Team
tryCatch.W.E <- function(expr)
{
  W <- NULL
  w.handler <- function(w){ # warning handler
    W <<- w
    invokeRestart("muffleWarning")
  }
  list(value = withCallingHandlers(tryCatch(expr, error = function(e) e),
    warning = w.handler)
```

R console in windows

Variables and Data Types

Single data can be stored in variables

- Data Types: "numeric", "character", "logical", ...

R console

```
>x = 3;  
>x;  
[1] 3  
>class(z);  
"numeric"  
>y = "Bioinformatics"  
>y;  
"Bioinformatics"
```

```
>class(y);  
"character"  
>z= TRUE;  
>z;  
TRUE  
>class(z);  
"logical"
```

Operations

We can apply arithmetic/logical functions to variables

+ (addition), - (subtraction) , / (division), * (multiplication)

R console

```
> x = 3;
> y = 4;
> x+y;
[1] 8
> x*y;
[1] 12
> x/y
[1] 1
```

```
> x>y;
[1] FALSE
> z=TRUE;
> z & (x>y) # logical and
[1] FALSE
> Z | (x>y) # logical or
[1] TRUE
```

Complex Data Structures

- Vector – variable containing a array of items of the same type
- Matrix – two dimensional vector with items of the same type
- Data Frame – complex data structure for two dimensional data where columns can be of distinct type (as an excell sheet).
- ...

Vector

- Creating, accessing and updating vector

```
> v=c(3.2,4.1,1.9)
> v
[1] 3.2 4.1 1.9
> v[2]          # access 2nd position of vector
[1] 4.1
> v[3] = 10.4  #update 3rd position of vector
> v
3.2 4.1 10.4
> u=c(1,2,3)
> u+v
>v
[1] 4.2 6.1 13.4
```

Vector

- Operations, functions and access

```
> z = u+v          # arithmetic operation in vector
> z
[1] 4.2 6.1 13.4
> length(z)        # function indicating size of vector
[1] 3
> z[c(1,3)]        #subsetting vector (1st and 3rd pos.)
[1] 4.2 13.4
> z>6              #logical operator
[1] FALSE TRUE TRUE
> z[z>6]           # return all values greater than 6
[1] 6.1 13.4
```


Matrix

- Matrix – two dimensional vector / same type

```
> m = matrix(1:12, 4, 3) # 4 by 3 matrix
>dim(m) # size of matrix
4 3
>m[1,] # show first row of matrix
[1] 1 5 9
>m[3,1] #show element at 3rd row / 1st collumn
[1]
>m
 [,1] [,2] [,3]
[1,]  1  5  9
[2,]  2  6 10
[3,]  3  7 11
[4,]  4  8 12
```

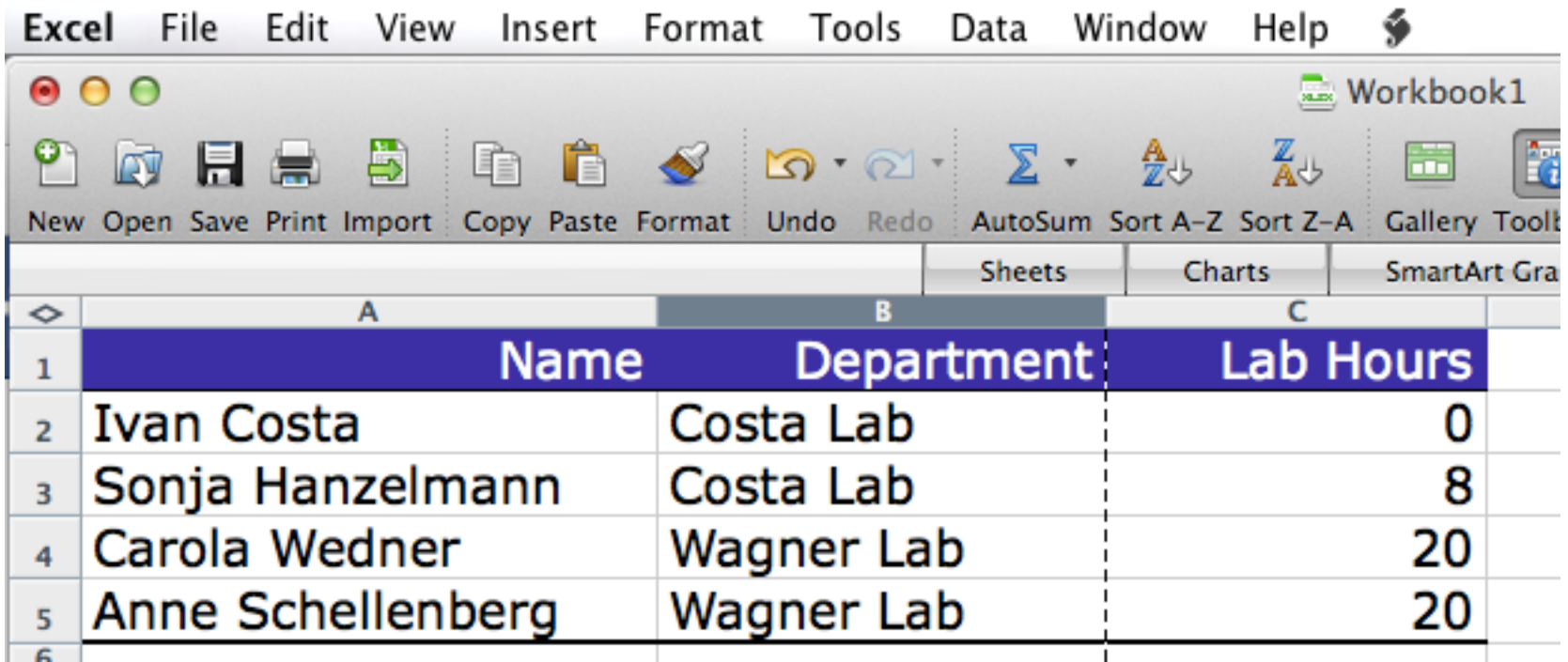
Matrix

- Matrix – two dimensional vector / same type

```
> m = matrix(1:12, 4, 3) # 4 by 3 matrix
>dim(m) # size of matrix
4 3
>m[1,] # show first row of matrix
[1] 1 5 9
>m[3,1] #show element at 3rd row / 1st collumn
[1]
>m
 [,1] [,2] [,3]
[1,]  1  5  9
[2,]  2  6 10
[3,]  3  7 11
[4,]  4  8 12
```

Data Frames

- Data frames hold a spreadsheet like table. The observations are the rows and the covariates are the columns. Columns share the same type.
- Data frames can be operated as matrices and be indexed with two subscripts.



The image shows a screenshot of the Microsoft Excel application window. The title bar reads "Excel File Edit View Insert Format Tools Data Window Help". The menu bar includes "File", "Edit", "View", "Insert", "Format", "Tools", "Data", "Window", and "Help". The toolbar contains icons for New, Open, Save, Print, Import, Copy, Paste, Format, Undo, Redo, AutoSum, Sort A-Z, Sort Z-A, and Gallery Tools. The spreadsheet area shows a table with three columns: "Name", "Department", and "Lab Hours". The data is as follows:

	A	B	C
1	Name	Department	Lab Hours
2	Ivan Costa	Costa Lab	0
3	Sonja Hanzelmann	Costa Lab	8
4	Carola Wedner	Wagner Lab	20
5	Anne Schellenberg	Wagner Lab	20
6			

Data Frames

- Creation and manipulation

```
> data = data.frame(name=c("Ivan", "Sonja", "Carola", "Anne"),
+ department=c("Costa", "Costa", "Wagner", "Wagner"),
+ labhour=c(0,8,20,20))
> data
  name department labhour
1  Ivan      Costa      0
2  Sonja     Costa      8
3  Carola    Wagner     20
4   Anne    Wagner     20
> data$department
[1] Costa Costa Wagner Wagner
Levels: Costa Wagner
```

Data Frames

- Creation and manipulation

```
>data[1,]
 name department labhour
1 Ivan      Costa      0
> data$labhour >8          # lab hours exceeding 8
[1] FALSE FALSE  TRUE  TRUE
> data[data$labhour >8,] # data from members with more than 8 hours
 name department labhour
3 Carola      Wagner    20
4 Anne       Wagner    20
> data[data$department=="Costa",] # data from members of Costa dept.
 name department labhour
1 Ivan      Costa      0
2 Sonja     Costa      8
```

Functions

- A section of a program that perform a specific task
 - Takes values as input parameter and returns some new value (or perform a operation)
- R and defines several types of functions
 - math: `log`, `exp`, `abs`, `sqrt`,...
 - array/matrix manipulation: `length`, `dim`, `array`, `repmat`,...
 - Read/write files: `read.table`, `write.table`, ...
- Can be created by user (not seen here) or defined in contributing packages

Example of Functions

```
>log2(4)
```

```
[1] 2
```

```
>dim(data)
```

size of the data frame

```
[1] 4 3
```

```
>summary(data)
```

statistics of a data frame columns

```
name  department  labhour
```

```
Anne :1  Costa :2  Min.   :0
```

```
Carola:1  Wagner:2  1st Qu.: 6
```

```
Ivan  :1                Median :14
```

```
Sonja :1                Mean   :12
```

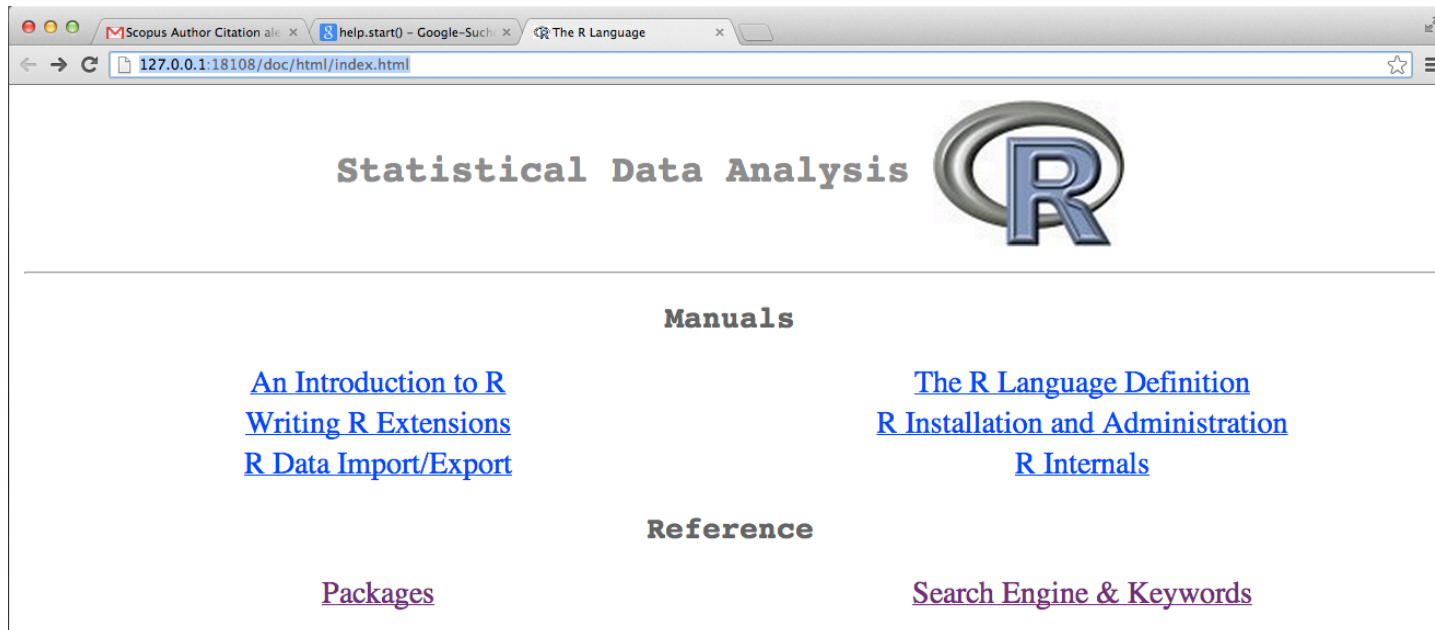
```
3rd Qu.:20
```

```
Max.   :20
```

```
>write.data("mydata.txt",data) # write data in a .txt file
```

General Commands & Links

- >quit() # end R session
- >ls() #show all variables currently defined
- [1] “x”, “y”, “data”
- >help(matrix) #shows help function for the function passed as param.
- >help.start() #opens a page with manual, tutorials and help search



The screenshot shows a web browser window with the URL `127.0.0.1:18108/doc/html/index.html`. The page content includes the text "Statistical Data Analysis" and the R logo. Below this, there are two main sections: "Manuals" and "Reference".

Manuals

- [An Introduction to R](#)
- [Writing R Extensions](#)
- [R Data Import/Export](#)
- [The R Language Definition](#)
- [R Installation and Administration](#)
- [R Internals](#)

Reference

- [Packages](#)
- [Search Engine & Keywords](#)

Packages

- In R the primary mechanism for distributing software is via packages
- CRAN is the major repository for packages.
- You should use `install.packages` or `update.packages` to install and update packages.
 - `>install.packages("packagename")` # install a new package
- In addition, on Windows and other GUIs, there are menu items that facilitate package downloading and updating.
- Bioconductor packages are installed with `biocLite` command.
 - `>source("http://bioconductor.org/biocLite.R")`
 - `>biocLite("packagename")`

Exercise 1

Create a vector representing the radius of three circles with lengths 5, 10, and 20. Use `*` and the built-in constant `pi` to compute the areas of `t`.

Exercise 1 - Solution

```
> circles <- c(5, 10, 20)
> # areas are:
> pi * circles * circles
[1] 78.53982 314.15927 1256.63706
> # you could also use ^
> pi * circles^2
[1] 78.53982 314.15927 1256.63706
> # reduce radii by 2.1
> pi * (circles - 2.1)^2
[1] 26.42079 196.06680 1006.59770
```

Exercise 2

Creating regular numeric sequences is a common task in statistical computing. You can use the `seq` function to create sequences.

1. Read the help page for `seq` by entering `help(seq)`.
2. Generate a decreasing sequence from 50 to 1, then another sequence from 1 to 50.
3. Use `seq` to generate a sequence of the even integers between one and ten.

Exercise 2 - Solution

```
> seq(50, 1) #sequences 50 to 1
[1] 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
[19] 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15
[37] 14 13 12 11 10 9 8 7 6 5 4 3 2 1
> seq(1, 50) # sequences 1 to 50
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
[37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50
> seq(2, 10, 2) #even integers
[1] 2 4 6 8 10
```

Exercise 3

- Create an integer vector i that can be used to subset v such that it will output the elements of v in decreasing order. For the general case, read the help pages for `order` and `sort`.

```
> v = c(1.1, 2, 100, 50, 60)
```

Exercise 3 - Solution

```
> i = c(3, 5, 4, 2, 1)           #create a vector with right order
> i = order(v, decreasing = TRUE) # or use sort function to get order
> i;
[1] 3 5 4 2 1
> v[i]                           # reorder elements from v
[1] 100.0 60.0 50.0 2.0 1.1
```

- More exercises at ...

http://www.bioconductor.org/help/course-materials/2010/BioC2010/First_Steps_With_R_SOLUTIONS.pdf