ROSLIN

R basics

THE UNIVERSITY of EDINBURGH

BBSRC
20 Years of Pioneering
Great British Bioscience

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

[Home]

Download

CRAN

R Project

About R
Contributors
What's New?
Mailing Lists
Bug Tracking
Conferences
Search

R Foundation

Foundation
Board
Members
Donors
Donate

Documentation

Manuals
FAQs
The R Journal
Books
Certification
Other

Links

Bioconductor
Related Projects

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

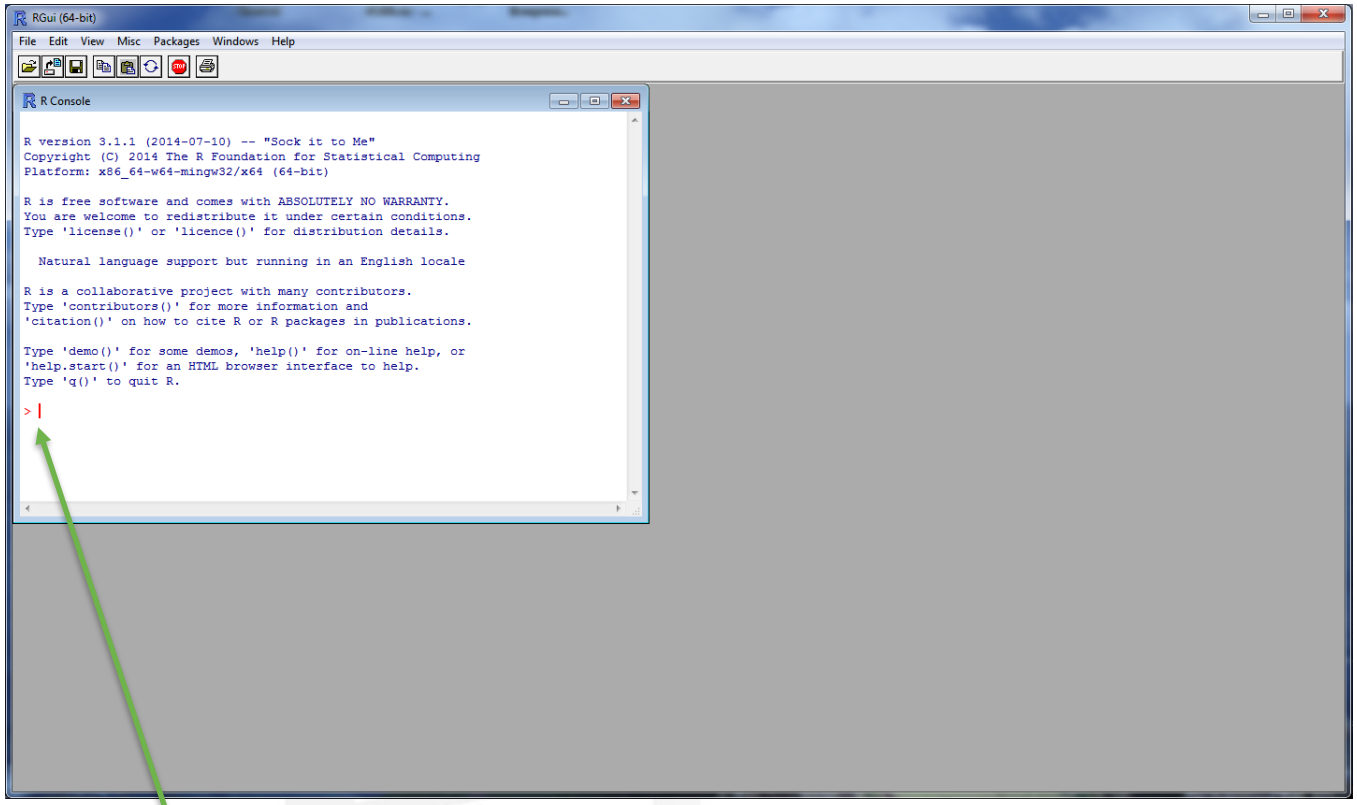
News

- **The R Journal Volume 7/2** is available.
- **R version 3.2.3 (Wooden Christmas-Tree)** has been released on 2015-12-10.>
- **R version 3.1.3 (Smooth Sidewalk)** has been released on 2015-03-09.
- **useR! 2015**, took place at the University of Aalborg, Denmark, June 30 - July 3, 2015.

With R you can

- Load data
- Calculate statistics
- Plot Graphs
- Create custom functions and scripts
- Create and run models
- Use advanced algorithms
- Perform data analysis

Basic R



R, with command prompt

RStudio

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Go to file/function

Project: (None)

simpleExamples.R

Source on Save | Run | Source

```
# An example script in R
# S J Lycett
# 25 Jan 2016

##############################################################################
# functions (do not change)

mySimpleFunction <- function( x, y ) {
  z <- x + y
  return( z )
}

myCustomFunction <- function(k=1, a=1, npts=100) {
  t <- 0:(npts-1)
  x <- k*exp(a*t)
  return( list(t=t, x=x) )
}

plotMyCustomFunction <- function( myOutput ) {
  plot( myOutput$t, myOutput$x, type="l", col="blue", main="My Custom Function")
}

##############################################################################
# Example commands

# use mySimpleFunction to add 8 & 14
mySimpleFunction(8, 14)

# plot myCustomFunction
myOutput <- myCustomFunction()
plotMyCustomFunction( myOutput )
```

31:33 (Untitled) R Script

View and Edit Scripts

Console //ris-fas1a.roslin.ed.ac.uk/slycett2/

```
R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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.

>
```

Command prompt

Environment | History

Import Dataset | Clear | List

Global Environment

Environment is empty

Data, variables
custom functions
("objects")

Files | Plots | Packages | Help | Viewer

Zoom | Export | Clear All

Plots etc

RSTUDIO

simpleExamples.R

```r
# An example script in R
# S J Lycett
# 25 Jan 2016

##############################################################################
# functions (do not change)

mySimpleFunction <- function( x, y ) {
  z <- x + y
  return( z )
}

myCustomFunction <- function(k=1, a=1, npts=100) {
  t <- 0:(npts-1)
  x <- k*exp(a*t)
  return( list(t=t, x=x) )
}

plotMyCustomFunction <- function( myOutput ) {
  plot( myOutput$t, myOutput$x, type="l", col="blue", main="My Custom Function")
}

##############################################################################
# Example commands

# use mySimpleFunction to add 8 & 14
mySimpleFunction(8, 14)

# plot myCustomFunction
myOutput <- myCustomFunction()
plotMyCustomFunction( myOutput )
```

Structure of the script

1. 3 lines of comment
2. Custom functions
3. Actual commands to run

```
R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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.

>
```

You can type commands into the prompt
Or highlight them in the script and press Ctrl+R

R has many in-built functions
But R must know about any custom functions first

What is a function ? (non rigorous def!)

- Functions have inputs and outputs
- Inputs are the variables (or parameters)
- Outputs are usually other variables, but can also be print to screen commands or plot commands

- In-built functions include:
  - sum
  - mean
  - sd (standard deviation)

Type:
help(mean)
To see the help pages for mean

- Functions can be imported from extra R packages (more later)
- Functions can also be defined in a script

What is a variable ? (non rigorous def!)

- In R variables can be:

Type	Example	Comment
Single value	x=1	Integer (1) or double (1.0) automatic conversion
Array	t=c(1,2,3)	Creates a 1D array containing 1, 2, 3 To see the 2nd element of the array, use t[2]
Array	t=0:9	Creates a 1D array containing 0 to 9 (10 elements)
Array	t=array(0, 20)	Creates a 1D array of 20 zeros
Matrix	M=matrix(1,3,3)	Creates a 3 x 3 matrix of 1's M[row,column] gives the element
List	L=list(x=1,t=0:9)	In this example List element L$x=1 and L$t = 0,1,2,3,4,5,6,7,9

What is a function ? (non rigorous def!)

- An example custom function (how to define one)

In R <- meaning assignment is used rather than =

Function name

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

Input variables, these can be single values, arrays or matrices in this example

The calculation to perform

The result to return to the command line (or another function)

different types of brackets are important for correct syntax (be aware, but you don't need to worry just yet)

Using a function

- An example custom function (how to use one)
- In the command line you would type:

  mySimpleFunction(4, 6)

- And in this case you would get the answer:

  10

Inputs to functions

- In this example, the inputs can be arrays as well as single values:
- Define arrays a & b:

```
a=c(1,2,3)
b=c(4,5,6)
```

- Perform mySimpleFunction on a and b (3 elements each so that is OK):

```
mySimpleFunction(a,b)
```

- The answer will be:

```
[1] 5 7 9
```

The [1] means the numbers printed on this line start from element 1 - if there were 100 numbers then it would be
[1] …
[10] …
[20] … etc

Default values in functions

- A function with named parameters

```
myCustomFunction <- function(k=1, a=1, npts=100) {
        …..
}
```

- The named parameters have default values, so the function can be used like this (just use defaults):

```
result <- myCustomFunction()
```

- Or like this - specify a different value for k and a but not npts:

```
result <- myCustomFunction(k=2, a=-1)
```

Custom Functions in R

RStudio

File Edit Code View Plots Session Build Debug Tools Help

simpleRExamples.R

```r
# An example script in R
# S J Lycett
# 25 Jan 2016

##########################################################################
# functions (do not change)

mySimpleFunction <- function( x, y ) {
  z <- x + y
  return( z )
}

myCustomFunction <- function(k=1, a=1, npts=100) {
  t <- 0:(npts-1)
  x <- k*exp(a*t)
  return( list(t=t, x=x) )
}

plotMyCustomFunction <- function( myOutput ) {
  plot( myOutput$t, myOutput$x, type="l", col="blue", main="My Custom Function")
}

##########################################################################
# Example commands

# use mySimpleFunction to add 8 & 14
mySimpleFunction(8, 14)

# plot myCustomFunction
myOutput <- myCustomFunction()
plotMyCustomFunction( myOutput )
```

Console //ris-fas1a.roslin.ed.ac.uk/slycett2/

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' for 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.

> mySimpleFunction <- function( x, y ) {
+   z <- x + y
+   return( z )
+ }
>
> myCustomFunction <- function(k=1, a=1, npts=100) {
+   t <- 0:(npts-1)
+   x <- k*exp(a*t)
+   return( list(t=t, x=x) )
+ }
>
> plotMyCustomFunction <- function( myOutput ) {
+   plot( myOutput$t, myOutput$x, type="l", col="blue", main="My Custom Function")
+ }
>
```

Environment History

Global Environment

Functions

myCustomFunction	function (k = 1, a = 1, npts = 100)
mySimpleFunction	function (x, y)
plotMyCustomFunction	function (myOutput)

3. Custom functions now exist in R

Files Plots Packages Help Viewer

1. Highlight the part of script you want to run

2. Functions "run" (defined) via command line

Running something

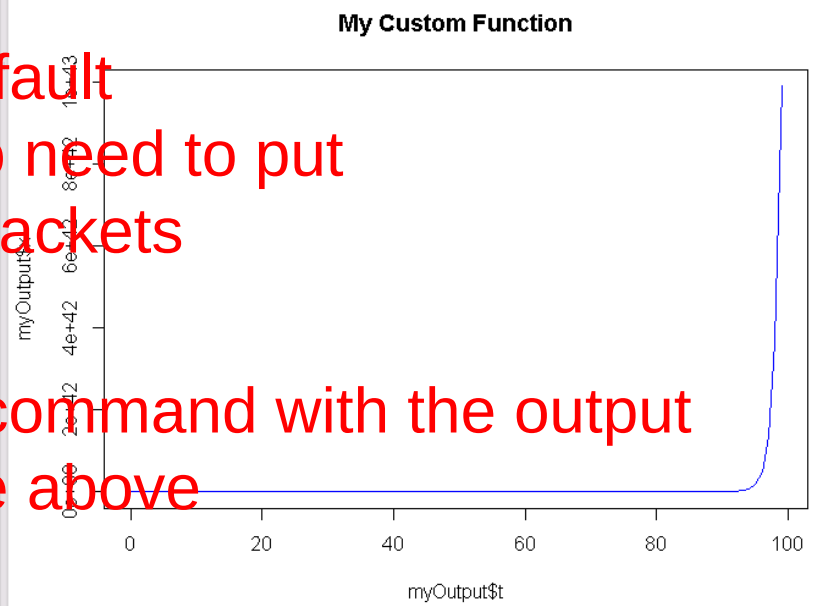
```r
# An example script in R
# S J Lycett
# 25 Jan 2016

##########################################################################
# functions (do not change)

mySimpleFunction <- function( x, y ) {
  z <- x + y
  return( z )
}

myCustomFunction <- function(k=1, a=1, npts=100) {
  t <- 0:(npts-1)
  x <- k*exp(a*t)
  return( list(t=t, x=x) )
}

plotMyCustomFunction <- function( myOutput ) {
  plot( myOutput$t, myOutput$x, type="l", col="blue", main="My Custom Function")
}

##########################################################################
# Example commands

# use mySimpleFunction to add 8 & 14
mySimpleFunction(8, 14)

# plot myCustomFunction
myOutput <- myCustomFunction()
plotMyCustomFunction( myOutput )
```

```
'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.

> mySimpleFunction <- function( x, y ) {
+   z <- x + y
+   return( z )
+ }
>
> myCustomFunction <- function(k=1, a=1, npts=100) {
+   t <- 0:(npts-1)
+   x <- k*exp(a*t)
+   return( list(t=t, x=x) )
+ }
>
> plotMyCustomFunction <- function( myOutput ) {
+   plot( myOutput$t, myOutput$x, type="l", col="blue", main="My Custom Function")
+ }
> myOutput <- myCustomFunction()
> plotMyCustomFunction( myOutput )
```

Commands run in command window

Just using the default parameters so no need to put anything in the brackets

Now do the plot command with the output generated on line above



Running something

RStudio

File Edit Code View Plots Session Build Debug Tools Help

simpleRExamples.R

```r
# An example script in R
# S J Lycett
# 25 Jan 2016

##########################################################################
# functions (do not change)

mySimpleFunction <- function( x, y ) {
  z <- x + y
  return( z )
}

myCustomFunction <- function(k=1, a=1, npts=100) {
  t <- 0:(npts-1)
  x <- k*exp(a*t)
  return( list(t=t, x=x) )
}

plotMyCustomFunction <- function( myOutput ) {
  plot( myOutput$t, myOutput$x, type="l", col="blue", main="My Custom Function")
}

##########################################################################
# Example commands

# use mySimpleFunction to add 8 & 14
mySimpleFunction(8, 14)

# plot myCustomFunction
myOutput <- myCustomFunction()
plotMyCustomFunction( myOutput )
```

Environment | History

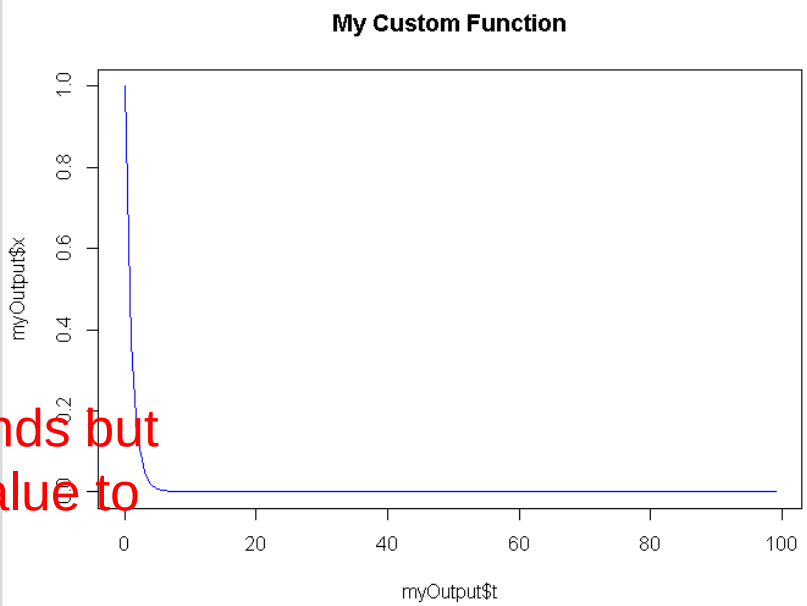
Global Environment

Values	
myOutput	List of 2
Functions	
myCustomFunction	function (k = 1, a = 1, npts = 100)
mySimpleFunction	function (x, y)
plotMyCustomFunction	function (myOutput)

Console //ris-fas1a.roslin.ed.ac.uk/slycett2/

```
+   return( z )
+ }
>
> myCustomFunction <- function(k=1, a=1, npts=100) {
+   t <- 0:(npts-1)
+   x <- k*exp(a*t)
+   return( list(t=t, x=x) )
+ }
>
> plotMyCustomFunction <- function( myOutput ) {
+   plot( myOutput$t, myOutput$x, type="l", col="blue", main="My Custom Function")
+ }
> myOutput <- myCustomFunction()
> plotMyCustomFunction( myOutput )
>
>
>
>
>
>
> myOutput <- myCustomFunction(a=-1)
> plotMyCustomFunction( myOutput )
>
```

Files | Plots | Packages | Help | Viewer



Repeating the commands but with a different input value to the function