More on the structure of R

Typing an expression causes R to evaluate and print the expression.

- If you want to save the result, you need to assign it using `"<-"`.
- For example, `sort(x)` will print the values of `x` in ascending order.
- Typing `y<-sort(x)` creates a new object `y` containing the sorted values of `x`.
- Typing `x<-sort(x)` replaces `x` with a new object containing the original values sorted in ascending order.

Functions are objects.

- Typing the name of a function will cause R to print the code.
- To execute the function you need to use brackets.

  `sort` prints out the code for the sort function.

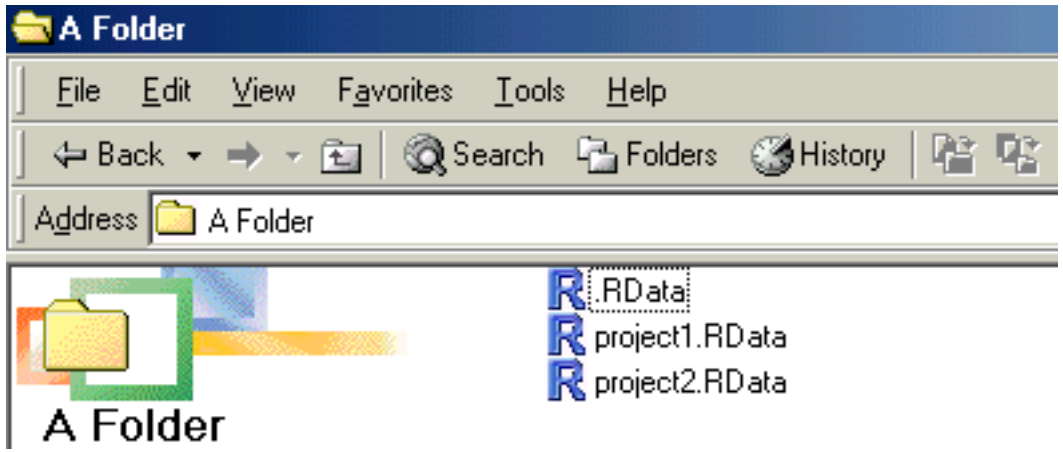
  `sort(x)` sorts the vector `x`.
- When no arguments are needed, you still need the brackets.

```
> quit
function (save = "default", status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
> quit()
```

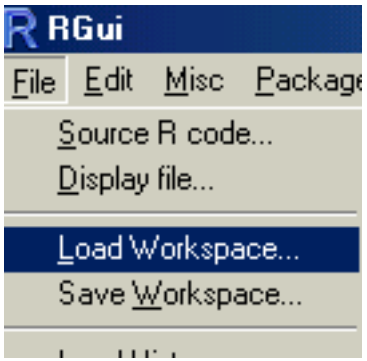
Organising your files

- R permanently stores your data objects in the workspace image, `.RData`.
- This feature can be very useful in the context of a single project.
  - But the workspace will become very cluttered if you do not remove objects that are no longer needed.
  - It is also helpful if objects have meaningful names.

- If you are working with several different experiments it is sensible to keep the data separate.
- A natural way to organise your files is to have a separate folder (directory) for each project.
  - You can have a separate `.Rdata` file within each folder.
- You can also have several workspace images in the same folder.

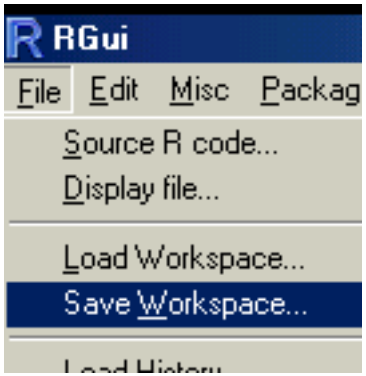


- Workspace image files have extension `.Rdata`.
- You can start R by double clicking on the icon of the required `.Rdata` file.
- When R is running you can load an existing workspace image:
  - Using the `Load Workspace` item on the `File` menu.



  - Using the `Load Workspace` button on the toolbar.

- New workspace image files can be created using either the `Save Workspace` item on the `File` menu



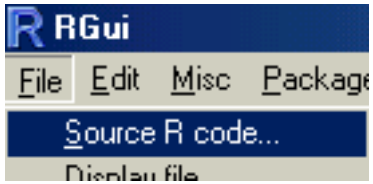
  or the `Save Workspace` button on the toolbar.

- The workspace image files can be copied and moved like any other files.

- They are also binary compatible between different platforms.

Using scripts

- To obtain useful output from R usually involves a sequence of several commands.
- These commands (including comments) can typed into a file using eg `Notepad`.
- The commands can then be run in R:
  - Using the `Source R Code` item from the `File` menu.



  - Using the `Source R Code` button on the toolbar.

- Scripts can also be run by typing the command

  `source("filename.R")`

  in the R Console.

- Files containing R code are text files.

- They usually have the extension `.R`.

- The main reasons for using scripts are:
  - It is useful for development.
  - You can re-use the scripts for different data.
  - They provide an audit mechanism.

Functions in R

- The R environment allows you to write and use your own functions.
- Functions are most useful when you need to perform the same sequence of operations with several different objects.
- Functions can be used:
  - To generate numerical output such as summary statistics,
  - To generate new data objects,
  - To generate plots and other graphics

Example 1

The function `rg2ma` takes a list containing matrices the log-ratios and averages. of red and green foreground and background values and generates the log ratios and average log intensities.

```
> rg2ma
function(x){
g<-x$G-x$Gb
r<-x$R-x$Rb
m<-log(r,2)-log(g,2)
a<-(log(r,2)+log(g,2))/2
list(M=m,A=a)
}
>
```

- The input argument `x` is assumed to be a list with matrix components `R, G, Rb, Gb`,
  - A more serious version would actually check this.
- The value of the final expression `list(M=m,A=a)` is returned by the function.
- The intermediate quantities `g, r, a, m` do not exist outside of the function.
- Comments may also be included.

```
> rg2ma
function(x){
g<-x$G-x$Gb                 # subtract green backgrounds
r<-x$R-x$Rb                 # subtract red backgrounds
m<-log(r,2)-log(g,2)        # calculate log ratio
a<-(log(r,2)+log(g,2))/2  # calculate average log intensity
list(M=m,A=a)          #create list with components a and m
}
>
```

The function (without comments) was created in R as shown below:

Note that the prompt changes from ">" to "+" until the final } is entered.

```
> rg2ma<-function(x){
+ g<-x$G-x$Gb
+ r<-x$R-x$Rb
+ m<-log(r,2)-log(g,2)
+ a<-(log(r,2)+log(g,2))/2
+ list(M=m,A=a)
+ }
>
```

We will test it on small.data shown below.

```
> small.data
$R
     [,1]  [,2] [,3]
[1,]  983  2190 2517
[2,]  921  2092 2516
[3,] 5289 12785 7592
$G
     [,1]  [,2]  [,3]
[1,]  481  1423  1740
[2,]  453  1356  1803
[3,] 3185 10965 11051
$Rb
     [,1] [,2] [,3]
[1,]   14    7    3
[2,]   14    7    3
[3,]   14    9    5
$Gb
     [,1] [,2] [,3]
[1,]  128  268  230
[2,]  128  282  226
[3,]  128  282  234
>
```

```
> rg2ma(small.data)
$M
          [,1]      [,2]       [,3]
[1,] 1.4568285 0.9184193  0.7354361
[2,] 1.4806628 0.9570534  0.6722280
[3,] 0.7870545 0.2581194 -0.5116990

$A
          [,1]     [,2]     [,3]
[1,]  9.191939 10.63289 10.92805
[2,]  9.084627 10.54730 10.95908
[3,] 11.971428 13.51209 13.14516

>
```

In practice we would probably use the function to create a new object:

```
> small.lratio<-rg2ma(small.data)
> small.lratio
$M
[1,] 1.4568285 0.9184193  0.7354361
[2,] 1.4806628 0.9570534  0.6722280
[3,] 0.7870545 0.2581194 -0.5116990

$A
[1,]  9.191939 10.63289 10.92805
[2,]  9.084627 10.54730 10.95908
[3,] 11.971428 13.51209 13.14516
```

Example 2

- The function `ma.plot` is a function for generating M vs A plots from a list `x` with matrix components `M` and `A`.

```
> ma.plot
function(x,i){
plot(x$A[,i],x$M[,i],main="M vs A plot",xlab="A",ylab="M")
abline(0,0,col="red")
}
> ma.plot(small.lratio,1)  # M vs A plot for slide 1
> ma.plot(rg2ma(small.data),1) # Equivalent to above
> ma.plot(small.lratio,2)  # M vs A plot for slide 2
```

- It automates the tedious task of getting separate plots for each column of `M` and `A`.

Scripts vs Functions

- Scripts are just a way of saving the commands that you would otherwise type into R.
  - It is useful for development and auditing.
  - A script can contain function definitions.
  - Scripts do not have input parameters and do not return values.
- Functions are most useful when you have a task that needs to be repeated on several objects of the same type.
- Unlike a script, a function is like a built in part of the R language.

Using Packages from CRAN

- The Comprehensive R Archive Network (CRAN) is located at

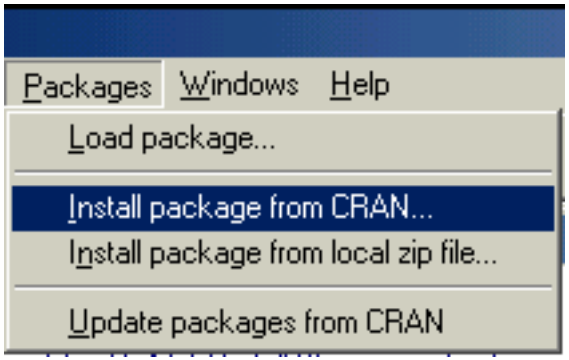
  `www.cran.r-project.org`

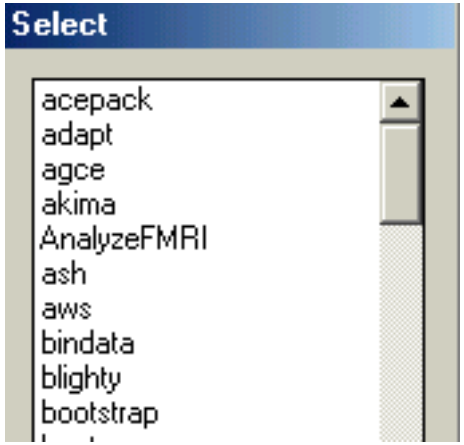
- It contains all of the R distribution files and several contributed packages.

- Packages are libraries of R functions that have been contributed by various authors.

Downloading and installing packages

If your computer is connected to the internet, packages can be installed directly from the CRAN archive by choosing `Install package from CRAN` on the `Packages` drop-down menu and then selecting the required package from the `Select` dialog box.





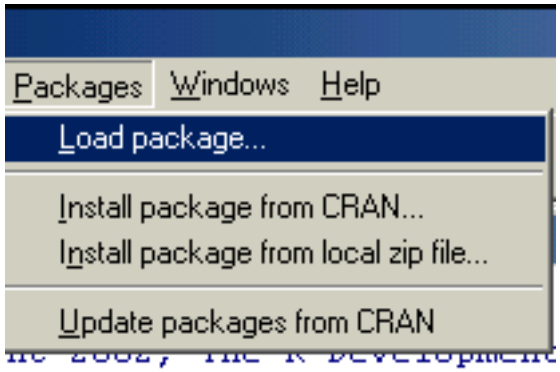
This will download, unzip and install the selected package.

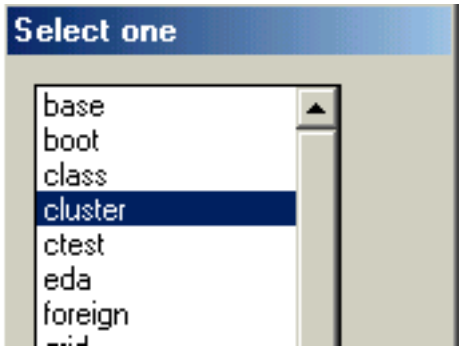
Keeping packages up to date

- Packages can be kept up to date by choosing `Update packages from CRAN` on the `Packages` menu.
- This checks whether there are newer versions of any of your packages available and installs them if desired.

Loading Packages

- Once a package has been installed it needs to be `loaded` before it can be used.
- Packages can be loaded by choosing the `Load Packages` item from the `Packages` menu and then choosing the required package from the `Select one` dialog box.





- Packages can also be loaded from the R console using the `library` function. eg `library(sma)`
- A package needs to be downloaded and installed only once.
- But it needs to be loaded every time you start a new R session.