Computing Practical I

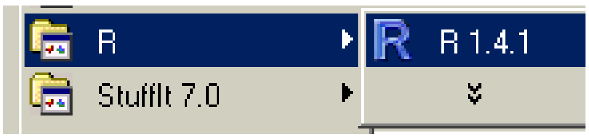
Introduction to R

22 April 2002

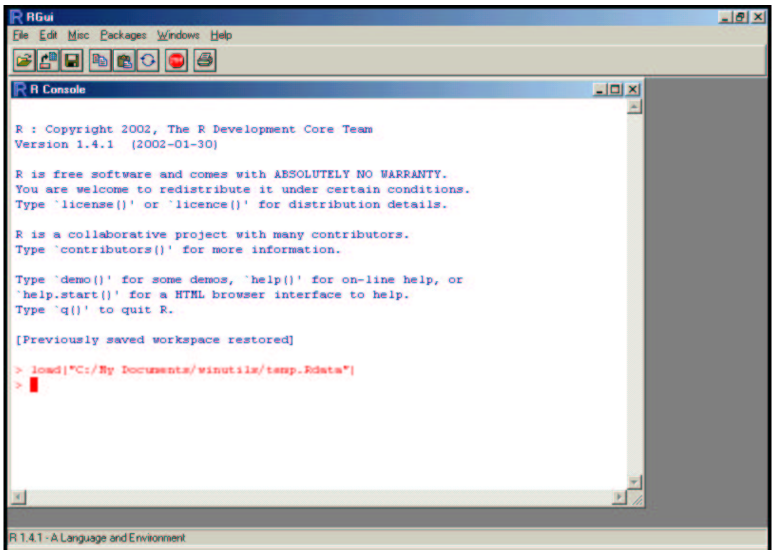
Practical Outline The purpose of this morning's practical is to provide an introduction to using R and to illustrate its general data handling and plotting capabilities.

1 Running R for Windows

To start R for windows, click on the `Start` button and then under `Programs>R`, choose `R1.4.1`.

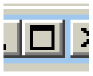


The `RGui` application should then appear as shown below.


RGui
File Edit Misc Packages Windows Help
R Console
R 1.4.1 - A Language and Environment

```
R : Copyright 2002, The R Development Core Team
Version 1.4.1  (2002-01-30)

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.

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

[Previously saved workspace restored]

> load("C:/My Documents/winutils/temp.Rdata")
> 
```

If the `RGui` window does not occupy the full screen, then click on the `maximize` button on the top right corner of the window.

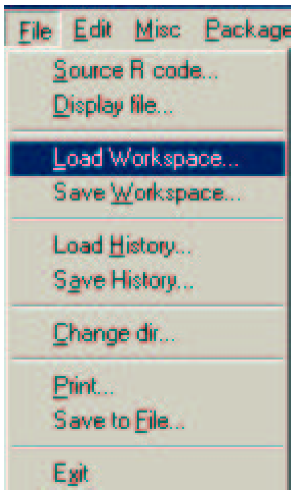


1.1 Loading a workspace image file

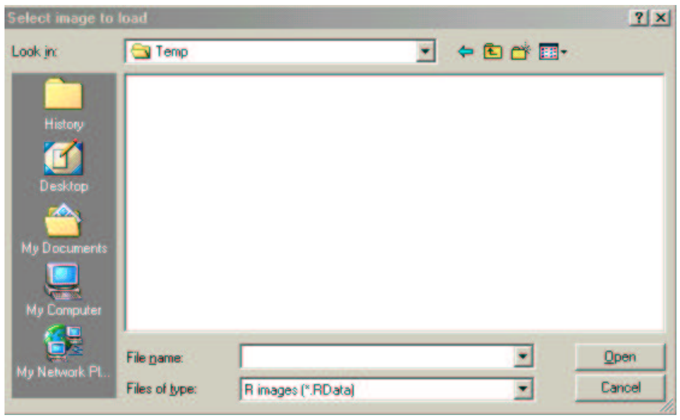
For this practical, a workspace image has been specially prepared and is available on the temporary partition, `D:\`. The file is called `.Rdata`.

To load the workspace image:

1. Choose `Load Workspace` from the `File` drop-down menu.



2. Select the workspace file in the `Select image to load` dialog box and then click on `OK`.



1.2 Using Help

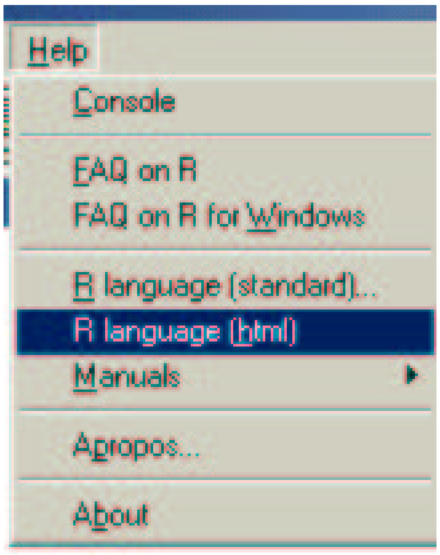
Various help facilities are available in R for windows.

- If you know the name of the function, but need more information, you can type `help` in the R console. For example, to get help on the `ls()` function, you can type

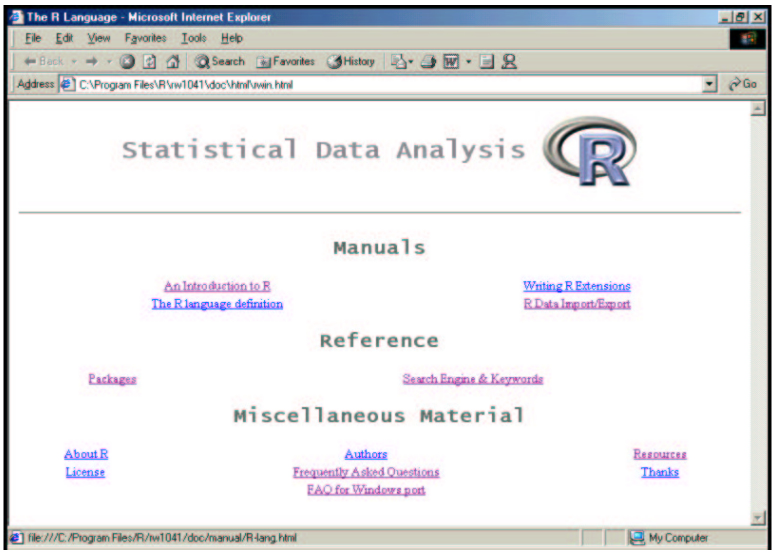
```
> help(ls)
>
```

This will open a window containing information about the functions. Unfortunately you need to know the name of the function to be able to use this facility.

- A large amount of documentation is also available via the `Help` drop-down menu.



- The `R Language(html)` entry opens the main page in the html version of the help system.


The R Language - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address C:\Program Files\R\rw1041\doc\html\rwin.html
Go
Statistical Data Analysis
Manuals
An Introduction to R
The R language definition
Writing R Extensions
R Data Import/Export
Reference
Packages
Search Engine & Keywords
Miscellaneous Material
About R
License
Authors
Frequently Asked Questions
FAQ for Windows port
Resources
Thanks
file:///C:/Program Files/R/rw1041/doc/manual/R-lang.html
My Computer

This page contains links to many useful resources including a search facility, an introduction to R, the manuals and, if you are connected to the internet, the FAQ sites.

2 Some basic operations in R

2.1 Examining Data Objects

The first step in examining the available data objects is to obtain a listing of all objects currently accessible. This is done using the `ls()` function.

```
> ls()
 [1] "exp1.data"            "exp1.norm.lratio"     "mice.data"
 [4] "mice.lratio"          "mice.norm.lratio"     "mice.not.norm.lratio"
 [7] "mice.setup"           "small.data"           "small.G"
[10] "small.R"
>
```

Typing the name of an object causes it to be displayed as output in the R console. For example,

```
> small.R
      [,1]  [,2] [,3]
 [1,]  983  2190 2517
 [2,]  921  2092 2516
 [3,] 5289 12785 7592
 [4,] 5569 11044 7874
 [5,] 5320  9404 8113
 [6,] 5533 12689 8507
 [7,] 5563 12655 8643
 [8,] 5670 12989 8521
 [9,] 4746  7998 7125
[10,] 4946  9012 7453
>
```

allows us to see immediately that `small.R` is a matrix with 10 rows and 3 columns. However, for large objects typically associated with microarray data, this is not feasible. Try typing `exp1.data`.

We can determine the `type` of an object by using commands `is.vector` etc.

```
> is.vector(small.R)
[1] FALSE
> is.matrix(small.R)
[1] TRUE
> is.list(small.R)
[1] FALSE
> is.data.frame(small.R)
[1] FALSE
```

The dimensions of a matrix can be displayed using the `dim` function.

```
> dim(small.R)
[1] 10  3
```

The object `small.data` is a list containing four components each of which is a matrix. It can be displayed in the usual way.

```
> small.data
$R
      [,1]  [,2] [,3]
 [1,]  983  2190 2517
 [2,]  921  2092 2516
 [3,] 5289 12785 7592
 [4,] 5569 11044 7874
 [5,] 5320  9404 8113
 [6,] 5533 12689 8507
 [7,] 5563 12655 8643
 [8,] 5670 12989 8521
 [9,] 4746  7998 7125
[10,] 4946  9012 7453

$G
      [,1]  [,2]  [,3]
 [1,]  481  1423  1740
 [2,]  453  1356  1803
 [3,] 3185 10965 11051
 [4,] 3350 10376 11638
 [5,] 3242  9405 12053
 [6,] 3412 11454 12225
 [7,] 3447 11580 12943
 [8,] 3557 11939 12812
 [9,] 2879  7820 11371
[10,] 3282  8443 11483

$Rb
      [,1] [,2] [,3]
 [1,]   14    7    3
 [2,]   14    7    3
 [3,]   14    9    5
 [4,]   16    9    5
 [5,]   16    9    5
 [6,]   16    9    9
 [7,]   12   14    9
 [8,]    8   14    9
 [9,]   12   14    8
[10,]   14    2    8

$Gb
      [,1] [,2] [,3]
 [1,]  128  268  230
 [2,]  128  282  226
 [3,]  128  282  234
 [4,]  123  282  234
 [5,]  123  282  234
 [6,]  118  275  234
 [7,]  133  275  224
 [8,]  133  264  225
 [9,]  133  271  225
```

```
[10,]  126  271  225

> length(small.data)
[1] 4
> attributes(small.data)
$names
[1] "R"  "G"  "Rb" "Gb"
```

When the commponents of a list are named, as is the case with `small.data`, the names can be displayed using the `attributes` function as shown above. In this case the individual components can be referred to by name using the "`$`" symbol. For example:

```
> small.data$Rb
      [,1] [,2] [,3]
 [1,]   14    7    3
 [2,]   14    7    3
 [3,]   14    9    5
 [4,]   16    9    5
 [5,]   16    9    5
 [6,]   16    9    9
 [7,]   12   14    9
 [8,]    8   14    9
 [9,]   12   14    8
[10,]   14    2    8
> is.matrix(small.data$Rb)
[1] TRUE
> dim(small.data$Rb)
[1] 10  3
>
```

Excerise

Check that `exp1.data` and `mice.lratio` are both lists. Find the names of their components. Check that each component is a matrix and find its dimension.

Note: for this exercise the command line editing feature in the R command window is very useful.

- Previous commands can be recalled using the $\uparrow$ key.
- The cursor can be positioned within the command using the $\leftarrow$ and $\rightarrow$ keys as necesary.
- The `delete` and `backspace` keys can be used to delete unwanted characters.
- New characters can be typed from the keyboard as usual.
- When the command has been edited, press the enter key to execute it.

For example, if you have previously typed the command `dim(small.data$R)` you can enter the command `dim(small.data$G)` simply by recalling the previous command and changing the `R` to a `G`. Cut and paste functions are also provided in the `Edit` drop-down menu.

2.2 Basic Data Manipulation

2.2.1 Arithmetic

Arithmetic expressions typed into R are evaluated and printed out. For example,

```
> x<-1
> y<-2
> x+y
[1] 3
> log(x/(1+3*y))
[1] -1.94591
>
```

Often we want to save the result of a calculation as a new R object and this is done using the assignment operator `<-`. For example,

```
> w<-x+y
> z<-log(x/(1+3*y))
> w
[1] 3
> z
[1] -1.94591
>
```

The data objects in an arithmetic expression can be a combination of scalars (single values), vectors or matrices. If they are all matrices of the same dimension, then the expression is evaluated. for each component. For example, `small.data$R` and `small.data$Rb` are matrices of red foreground and background intensities. To obtained the background adjusted intensities we subtract the backgrounds.

```
> small.data$R
      [,1]  [,2] [,3]
 [1,]  983  2190 2517
 [2,]  921  2092 2516
 [3,] 5289 12785 7592
 [4,] 5569 11044 7874
 [5,] 5320  9404 8113
 [6,] 5533 12689 8507
 [7,] 5563 12655 8643
 [8,] 5670 12989 8521
 [9,] 4746  7998 7125
[10,] 4946  9012 7453
> small.data$Rb
      [,1] [,2] [,3]
 [1,]   14    7    3
 [2,]   14    7    3
 [3,]   14    9    5
 [4,]   16    9    5
 [5,]   16    9    5
 [6,]   16    9    9
 [7,]   12   14    9
 [8,]    8   14    9
 [9,]   12   14    8
[10,]   14    2    8
> R<-small.data$R-small.data$Rb
> R
      [,1]  [,2] [,3]
 [1,]  969  2183 2514
 [2,]  907  2085 2513
 [3,] 5275 12776 7587
 [4,] 5553 11035 7869
 [5,] 5304  9395 8108
 [6,] 5517 12680 8498
 [7,] 5551 12641 8634
 [8,] 5662 12975 8512
 [9,] 4734  7984 7117
[10,] 4932  9010 7445
>
```

With the small example, we can obtain the green background adjusted intensities and the log intensity ratios, `M` as follows.

```
> R<-small.data$R-small.data$Rb
> G<-small.data$G-small.data$Gb
> M<-log(R/G,base=2)
> M
           [,1]       [,2]       [,3]
 [1,] 1.4568285 0.91841928  0.7354361
 [2,] 1.4806628 0.95705339  0.6722280
 [3,] 0.7870545 0.25811938 -0.5116990
 [4,] 0.7830738 0.12858864 -0.5352877
 [5,] 0.7659972 0.04238484 -0.5436900
 [6,] 0.7440434 0.18176361 -0.4967567
 [7,] 0.7441741 0.16114960 -0.5588842
 [8,] 0.7256290 0.15231199 -0.5643644
 [9,] 0.7857281 0.08082617 -0.6471849
[10,] 0.6440756 0.14083790 -0.5966068
>
```

Exercise

Calculate the matrices R, G and M from the full `exp1.data` object by adapting the above commands. Calculate also the matrix `A` of average log intensities defined by

$$A = (\log_2 R + \log_2 G)/2.$$

2.2.2 Selecting from matrices

Individual elements of a matrix `A` are addressed using the notation `A[i,j]` where `i` is a number specifying the row and `j` is a number specifying the columns. Submatrices, row and columns can also be addressed this way as illustrated below.

```
> small.R   # The whole matrix
      [,1]  [,2] [,3]
 [1,]  983  2190 2517
 [2,]  921  2092 2516
 [3,] 5289 12785 7592
 [4,] 5569 11044 7874
 [5,] 5320  9404 8113
 [6,] 5533 12689 8507
 [7,] 5563 12655 8643
 [8,] 5670 12989 8521
 [9,] 4746  7998 7125
[10,] 4946  9012 7453
> small.R[1,1]  # The 1,1 entry.
[1] 983
> small.R[1:3,1:2]   # The first 3 rows and 2 columns
     [,1]  [,2]
[1,]  983  2190
[2,]  921  2092
[3,] 5289 12785
> small.R[2,]  # The second row.
[1]  921 2092 2516
> small.R[,1]  # The first column.
 [1]  983  921 5289 5569 5320 5533 5563 5670 4746 4946
>
```

3 Plotting the data

Data sets arising from microarray experiments are generally too large to dislay as text and better described using a combination of suitable graphs and summary statistics.

3.1 Histograms

Histograms are produced in R using the `hist` function. In particular, if `x` is a vector containing the data, then `hist(x)` produces a histogram of the data. Keeping in mind how columns of matrices are extracted as vectors, we can now look at histrograms of various quantities on a slide by slide basis. For example, the matrix, `R` created in the previous exercise contains the red background adjusted intensities for the three slides from experiment 1. A histogram for slide 1, together with the mean and standard deviation, can then be obtained as follows:

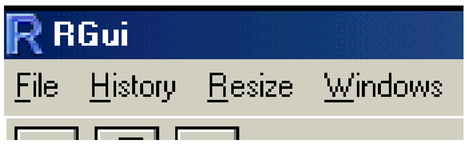
```
> hist(log(R[,1],base=2))
> hist(R[,1])
> mean(R[,1])
[1] 1126.764
> sd(R[,1])
[1] 2697.163
> # The histogram is highly skewed so look at the
> # log-transformed values.
> hist(log(R[,1],base=2))
> mean(log(R[,1],base=2))
[1] 9.10023
> sd(log(R[,1],base=2))
[1] 1.486119
>
```

3.2 Saving your graphs

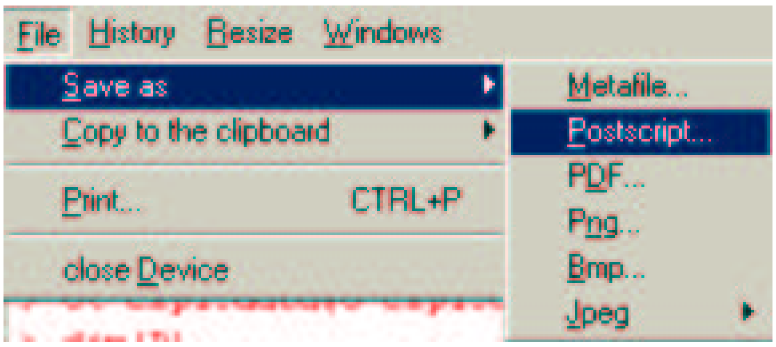
Histograms and other graphs can exported from R in various formats (such as postscript or JPEG) to be used in other documents. This can be done as follows.

1. Create the desired graphs by typing commands in the R console.

2. When you are happy with graph, click anywhere on the `R Graphics` window to bring it to the foreground. The menu bar will now appear as follows.



3. Select the `Save as` item from the `File` drop-down menu and choose the required format.



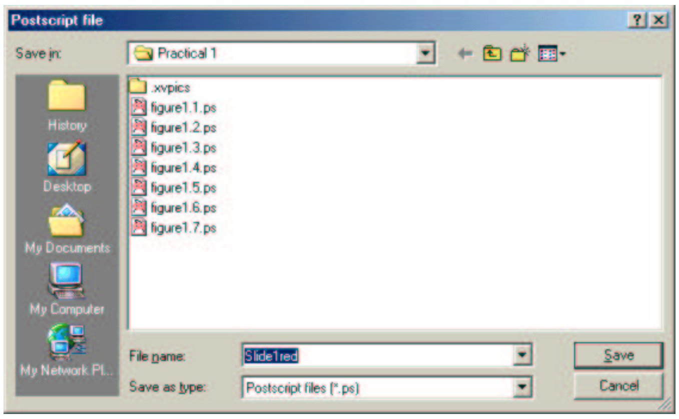
   (a) Postscript files are suitable when high quality graphics are required but tend to be large and may require special software to view.

   (b) JPEG files will not produce the same quality but tend to be smaller and easier to handle.

   (c) Both formats can be inserted directly into Word and resized as necessary, but postscript figures are usually not visible on the screen.

   (d) Other formats are also available.

4. Choose a name and folder for the new file and click on the `Save` button.



5. A new file containing your graph in the specified format should then have been created.

Exercise

Save the histogram of the red log intensities for slide 1 as a JPEG file on the `Temp` partition `"D:\"`. Check that the file has been created.

3.3 Customizing histograms

R provides extremely powerful facilities for producing customized graphics and a systematic discussion is beyond the scope of this workshop. To give some idea of what can be acheived, suppose we wish to compare the distributions of red and green background values for slide 1 of experiment 1. To do this effectively, we would like to produce clearly labelled histograms one above the other with the same scale on the horizontal axis. Shown below is a sequence of commands that achieve this.

```
> # First set up the graphics windows for multiple plots
> par(mfrow=c(2,1)) # multiple figures, by row, 2 rows one column
> # Now find the total range of the values to be plotted
> range(exp1.data$Rb[,1])
[1]  1 31
> range(exp1.data$Gb[,1])
[1]  67 146
> # We will use the range 0-150 for both graphs
> hist(exp1.data$Rb[,1],col="red",main="Red Background",
+ xlab="intensity",xlim=c(0,150),breaks=c(0:50)*3)
> hist(exp1.data$Gb[,1],col="green",main="Green Background",
+ xlab="intensity",xlim=c(0,150),breaks=c(0:50)*3)
>
```

The result is shown in Figure 1. In generating the histograms, the following optional arguments were used:

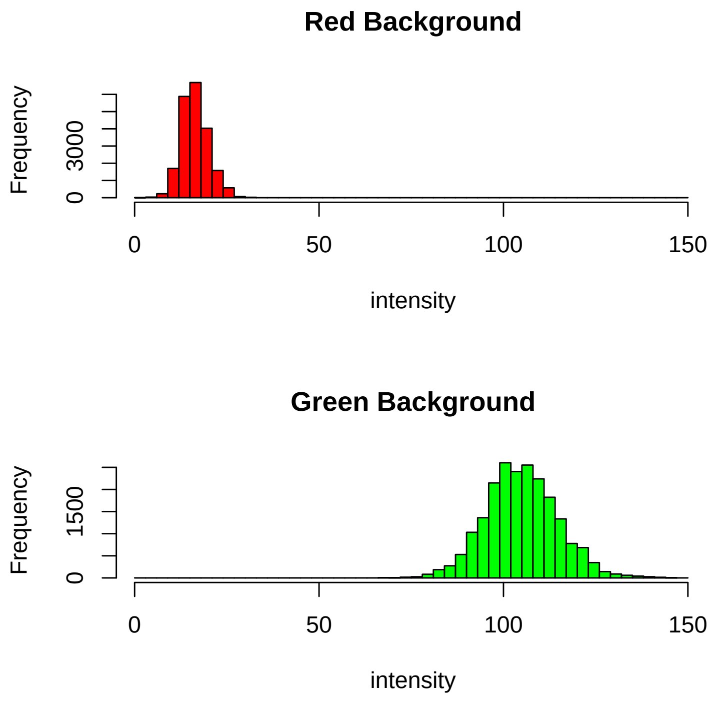


Figure 1: Red and green background values for slide 1

col: Specify the color of the bars.

main: Specify the title for the histogram.

xlab: Specify the label for the x-axis. (ylab is also available)

xlim: Set the lower and upper limits of the x-axis.

breaks: Specify the boundaries for the class intervals. In the case,

$$0 - 3, 3 - 6, \ldots, 147 - 150.$$

As a second example, shown below is the command to generate a histogram of the log ratios M from the first slide of experiment 1. In this case the bars are colored on a scale where $M = -6$ corresponds to pure green and $M = +6$ pure red. Detailed explanation of this command will not be given, but the result is shown in Figure 2.

```
> hist(exp1.norm.lratio[,1],nclass=20,col=rainbow(67)[24:5],
+ main="Slide 1",xlab="Log Intensity Ratio")
>
```

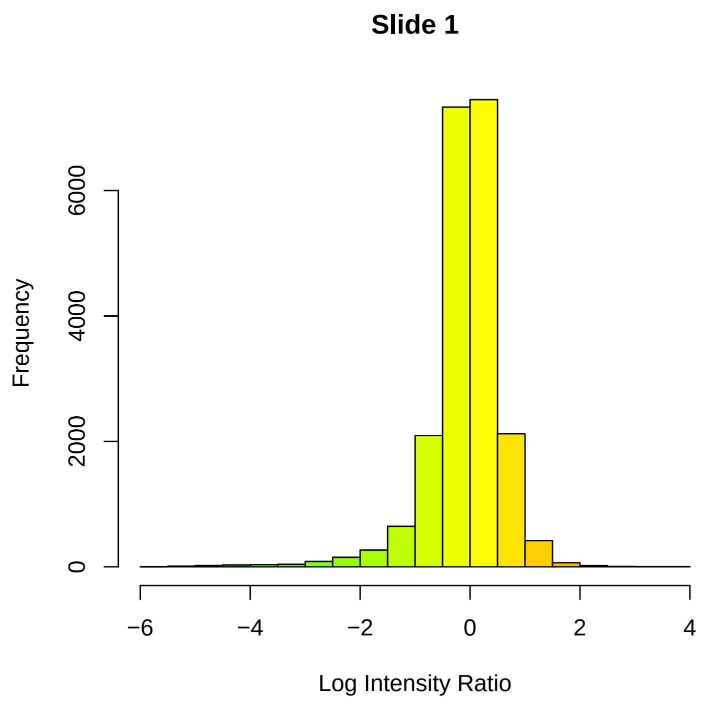


Figure 2: Normalised log ratios from slide 1, experiment 1.

3.4 Scatter Plots

Another commonly used plot is the scatter plot. If `x` and `y` are vectors of the same length, the command `plot(x,y)` produces a scatter plot of `y` vs `x`. That is `y` on the vertical and `x` on the horizontal axis. For example, to determine the extent to which normalisation is required we can plot the log ratios vs average intensity from experiment 1 using the `M` and `A` matrices that you created previously. The M vs A plot for slide 1 can be obtained using the following commands and the result is shown in Figure 3.

```
> # Basic version
> plot(A[,1],M[,1])
> # Can also specify options as for the histogram
> plot(A[,1],M[,1],main="M vs A, Slide 1", col="blue",
+ xlab="Average log intensity",ylab="Log ratio")
> # and add a line at M=0 to the plot
> abline(0,0,col="red")
>
```

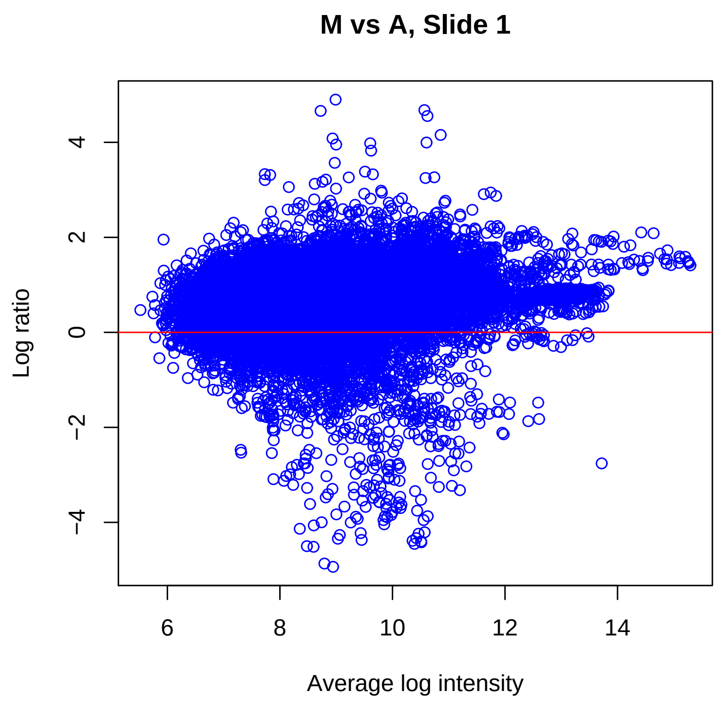


Figure 3: M vs A plot from slide 1.

4 Sorting and Selecting

Graphs enable use to visually identify interesting aspects of the data. However, it is often also necessary to be able to examine the data in other ways. For example, we might want to identify all genes with `M` above a certain threshold or perhaps the top ten values. The most important functions for this purpose are `sort` and `order`.

sort: If `x` is a vector then `sort(x)` produces a vector of the same length as `x` with the components sorted in ascending order.

order: If `x` is a vector then `order(x)` produces a vector of same length as `x` containing the indices of the sorted components.

To illustrate, we consider the green matrix from the small data set.

```
> > small.G
      [,1]  [,2]  [,3]
 [1,]  481  1423  1740
 [2,]  453  1356  1803
 [3,] 3185 10965 11051
 [4,] 3350 10376 11638
 [5,] 3242  9405 12053
 [6,] 3412 11454 12225
 [7,] 3447 11580 12943
 [8,] 3557 11939 12812
 [9,] 2879  7820 11371
[10,] 3282  8443 11483
> small.G[,1]
 [1]  481  453 3185 3350 3242 3412 3447 3557 2879 3282
> sort(small.G[,1]
+ )
 [1]  453  481 2879 3185 3242 3282 3350 3412 3447 3557
> order(small.G[,1])
 [1]  2  1  9  3  5 10  4  6  7  8
> # the smallest value occurs as position 2 of the original vector
> # the second smallest is in position 1
> # the largest is in position 6, etc.
> index<-order(small.G[,1])
> small.G[index,1]
 [1]  453  481 2879 3185 3242 3282 3350 3412 3447 3557
> # produces the same result as sort()
> # can also order the rows of the whole matrix
> small.G[index,]
      [,1]  [,2]  [,3]
 [1,]  453  1356  1803
 [2,]  481  1423  1740
 [3,] 2879  7820 11371
 [4,] 3185 10965 11051
 [5,] 3242  9405 12053
 [6,] 3282  8443 11483
 [7,] 3350 10376 11638
 [8,] 3412 11454 12225
 [9,] 3447 11580 12943
[10,] 3557 11939 12812
> # and choose the three with the smallest values of column 1.
> small.G[index[1:3],]
     [,1] [,2]  [,3]
[1,]  453 1356  1803
[2,]  481 1423  1740
[3,] 2879 7820 11371
>
```