Introduction to R

Computing Practical II

22 April 2002

Practical Outline The purpose of this afternoon's practical is to introduce the use of scripts and functions in R and to give you some 'hands-on' experience with reading in, plotting and normalizing cDNA microarray data using the `sma` package (Statistics for Microarray Analysis) in R. We will also conduct some higher-level statistical analyses, including an empircial Bayes analysis of dye-swapped replicates.

We note here in pasing that `sma` is being modified and expanded as part of the Bioconductor Project (`http://www.bioconductor.org`).

1 Using scripts

A script is simply a text file containing R commands. Scripts usually have the extension `".R"`. There are two reasons for working with scripts.

First, a typical analysis consists of a sequence of R commands. When you are developing your analysis, it is very likely that you will decide several times that it would have been better to do some differently in the earlier stages. This can be very tedious even if you use the history mechanism. On the other hand, if you have your commands in a script file, it is simply a matter of changing the commands in the file and then running the script again.

Second, the script file provides an accurate audit of how your produced your output and graphs. If you want to repeat the analysis on different data some time in the future or just remind yourself exactly what you did (when you look at the output a few months later), the exact commands will be there in the script file.

To illustrate the use of script files, we will consider plotting the log intensity ratios for slides 3 and 4 of the mice data. These slides are dye-swapped replicates, so it is relevant to consider a scatter plot. The file, `myplot.R` is an R script containing the commands to obtain a plot similar to that shown in lectures.

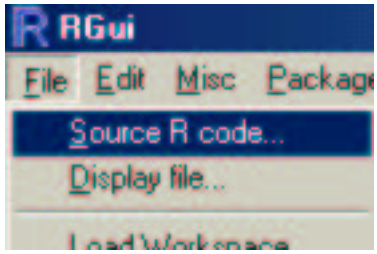
The contents of the file are shown below:

```
#Set up the axes and labels but don't show any points yet

plot(mice.lratio$M[,3],mice.lratio$M[,4],type="n",
xlab="Slide 3",ylab="Slide 4",main="Log Ratios")

s3<-sd(mice.lratio$M[,3])
s4<-sd(mice.lratio$M[,4])
radius<-sqrt(mice.lratio$M[,3]^2/s3^2+mice.lratio$M[,4]^2/s4^2)
```

```
threshold<-4.5

# Now add points with radius less than threshold
# No real evidence of differential expression so use a pale
# colour.
points(mice.lratio$M[radius<threshold,3], mice.lratio$M[radius<threshold,4],
col="yellow")

# Now add the points outside the radius using a darker colour.
points(mice.lratio$M[radius>=threshold,3], mice.lratio$M[radius>=threshold,4],
col="orange")

# The points with large radius and slide3 approximately -slide4
# are those that show consistent evidence from both slides
# Mark the in red.
threshold2<-1
absum<-abs(mice.lratio$M[,3]+mice.lratio$M[,4])
points(mice.lratio$M[radius>=threshold&absum<threshold2,3],
mice.lratio$M[radius>=threshold&absum<threshold2,4], col="red")

# Finally add a line with slope -1 and intercept zero to indicate
# points of exact agreement between the two slides.

abline(0,-1,col="pink")
```

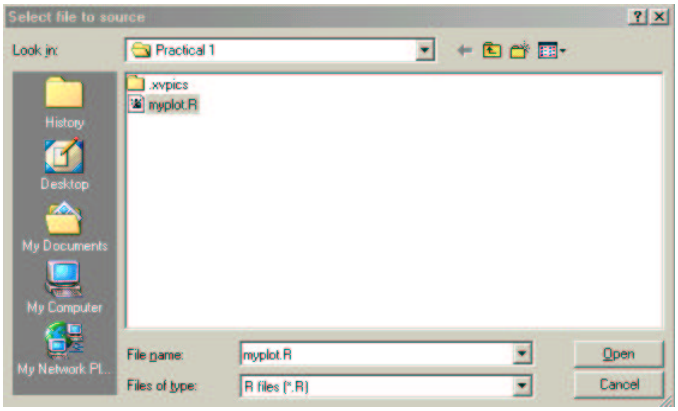
To run the script in R:

1. Choose `Source R code` from the `File` drop-down menu.



2. Select the appropriate file in the `Select file to source` dialog box and then click

on the `Open` button.



This will cause R to execute the commands in your script.

Exercise

1. Open the file `myplot.R` using `Notepad` and examine its contents.
2. Run the script in R and observe the graph produced.
3. Now change the threshold in your script file to 4.5 and make the line blue instead of pink. Run the script again and observe the effect on the plot.
4. Extension exercise (if you have time later). Modify the script to so that the points plotted red in the present version are plotted red when slide 3 is negative and and green when slide 3 is positive.

2 Functions in R

As was discussed in the lecture, you can also write your own functions in R. For example, a function to calculate the log ratios and average log intensities can be created in R as shown below.

```
> 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)
+ }
>
```


Exercise

1. Carefully type in the function shown above. Note that the ">"s and "+"s are prompts and should not be typed.
2. Type `rg2ma` to display the code for the function after it has been succesfully created.
3. Type `rg2ma(small.data)` and confirm that the output is correct.

2.1 Creating functions from scripts

If you make an error anywhere while typing a function, you will need to retype the entire function. For this reason it is much easier to put the commands for creating a function in a script file. Then, if there is an error you can simply correct it and run the script again.

The file `myfun.R` contains the code to create a function for plotting dye-swapped replicate slide. The contents of that file are shown below.

```
myplot<-function(x,y,threshold=4,threshold2=1){

# Note that x and y are required input parameters.
# The function will not work without them.
# threshold and threshold2 are optional parameters.
# If you do not specify them, R will assume you want
# threshold=4 and threshold2=1

# Note that the rest of the code is as in the previous script file
# except that we use x instead of "mice.lratio$M[,3]" and
# y instead of "mice.lratio$M[,4]"

plot(x,y,type="n",
xlab="Slide 1",ylab="Slide 2",main="Log Ratios")
sx<-sd(x)
sy<-sd(y)
radius<-sqrt(x^2/sx^2+y^2/sy^2)
points(x[radius<threshold], y[radius<threshold], col="yellow")
points(x[radius>=threshold], y[radius>=threshold], col="orange")
absum<-abs(x+y)
points(x[radius>=threshold&absum<threshold2],
y[radius>=threshold&absum<threshold2], col="red")
abline(0,-1,col="pink")
}
```

Exercise

1. Examine the script file `myfun.R` using notepad.
2. Run the script in R and confirm that a new function called myplot has been created.

3. Test the function by typing:

   ```
   > myplot(mice.lratio$M[,3],mice.lratio$M[,4])
   ```

4. Try using the optional arguments, eg by typing:

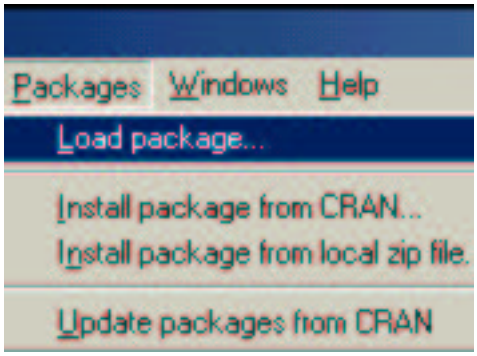
   ```
   > myplot(mice.lratio$M[,3],mice.lratio$M[,4],threshold=4.5,threshold2=0.5)
   ```

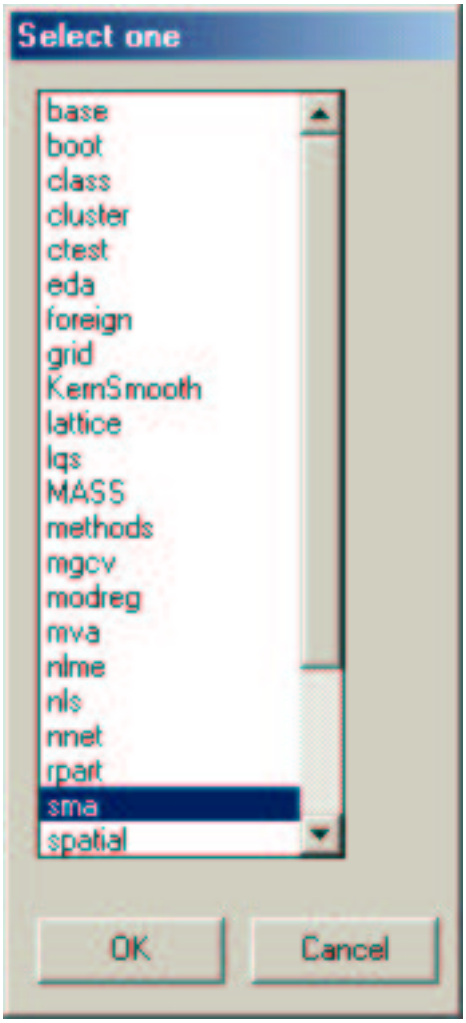
5. Extension excerise (if you have time later). Modify the code to include optional arguments, `col1="yellow"`, `col2="orange"` and `col3="red"` to control the colour coding of the points.

3 Using SMA

The package `SMA` has been installed on our system as described in the lecture. Before you can use its functions, it must be loaded in R as follows.

Choose the `Load package` item from the `Packages` drop-down menu. Then select select `sma` and click on the OK button.





The package will then be loaded and its functions available.
Sample microarray dataset: the data we will use to illustrate some of the key steps in the analysis of a microarray experiment are from a larger study designed to identify genes which play an important role in leukaemia. Two mutant mouse cell populations N and L were compared at time zero hours and at time 24 hours, so there are four possible population/time combinations: $N0$, $N24$, $L0$, $L24$. The cell population comparisons at time zero ($N0$ versus $L0$) and at time 24 ($N24$ versus $L24$) were repeated with the dye-assignment reversed, and these are the data we will use for this part of the practical.

The data have already been read into `sma` using the interactive initialization functions `init.grid()` and `init.data()`, which have produced the files `mice.setup` and `mice.data` you can see in your directory. [1]

`mice.setup` specifies the dimensions of the spot and grid matrices determined by the printing layout of the array.

`mice.data` is a list of four matrices, which store raw red and green foreground intensities(R and G), and red and green background intensities (Rb and Gb). The rows of the matrices correspond to the spots (i.e. genes) and the columns correspond to the slides (i.e. arrays).

1. **Reading in the data**: to begin, create the data structure which contains the unnormalized log ratios M and the average spot intensities A. We do this using the function `stat.ma`

```
> mice.MA  <- stat.ma(mice.data, mice.setup, norm="n")
>
```

which generates a list of two matrices, one for the log ratios M, and the other for the average log intensities A, where as before, the rows of the matrices correspond to the spots and the columns correspond to the slides. Note that `stat.ma` is used in the normalization steps which we will consider later; the `norm = "n"` option in the above command specifies that the log ratios are not to be normalized yet. Note also that `stat.ma` is a much more comprehensive version function rg2ma that we wrote previously.

Look at the first 10 lines of each of the two matrices in `mouse.MA` (note that the object name is case sensitive):

```
> mice.MA$M[1:10,]
> mice.MA$A[1:10,]
>
```

There are a total of eight slides in this experiment. We will analyse the first four.

[1] For people who are interested in these preliminary steps, see the attached Appendix A which contains a sample transript of a `Spot` session for setting up the data structures.

Slide 1: The combination $N0$ is labelled green and the combination $L0$ is labelled red.

Slide 2: The dye assignment is reversed for the same hybridization.

Slide 3: The combination $N24$ is labelled green and the combination $L24$ is labelled red.

Slide 4: the dye assignment is reversed for the same hybridization.

2. **Diagnostic plots**:

   (a) The function `plot.spatial` creates an image in which shades of grey or colour represent the value of a statistic for each spot on the array. Consider slide 3, the cell comparison at time 24 hours, before the data are normalized, and look at the spots having the highest and lowest pre-normalization log ratios:

```
> m3 <- mice.MA$M[,3]
> plot.spatial(m3, mice.setup, crit1=0.05)
>
```

   (b) Boxplots are useful for comparing the distributions of several sets of spots at the same time. Here we compare boxplots of the unnormalized log ratios by print-tip group for slide 1:

```
> m1 <- mice.MA$M[,1]
>  plot.scale.box(m1, mice.setup)
>
```

   Are the locations and spreads of the different print-tip groups comparable?

   If we want to look at boxplots to compare all eight slides (one boxplot per slide) type:

```
> plot.scale.box(mice.MA$M)
>
```

   How do the slides differ overall in terms of location and scale?

   (c) To construct an M versus A plot for a single slide (let's take slide 3), unnormalized log ratios, type

```
> plot.mva(mice.data, mice.setup, norm="n", image.id=3,
+ extra.type="p", plot.type="r",  col="blue")
>
```

   There are numerous options available for the arguments for this function; see `help(plot.mva)`. For instance, extreme log ratios can be highlighted using different plotting symbols or colours, or intensity-dependent lines can be added for a pre-specified proportion of extreme points. An example is


```
> plot.mva(mice.data, mice.setup, norm="n", image.id=3,
+ extra.type="tci", plot.type="r", crit1=0.005, col.ex="purple", pch=20)
> title("Mice slide 3")
>
```

which produces the same plot as above, highlighting the spots with the highest and lowest 0.5% log ratios using their row numbers in `mice.data`, and adding a title.

Try some different plotting options using a different slide.

3. **Normalization**:

   (a) Within-slide normalization. We can create a plot of the normalized log ratios using `plot.mva`. In the first instance, we will perform global lowess normalization:

```
> plot.mva(mice.data, mice.setup, image.id=3, extra.type="p",
+ plot.type="n" , col="blue")
>
```

   If we add the argument `plot.type="b"` we get two plots, one for the normalized log ratios and one for the unnormalized log ratios. Try this now: you will be prompted to see the plots, one after the other.

   To create the data structure (i.e. a list of two matrices) containing the global lowess normalized log ratios for all eight slides, type

```
> mice.lratio  <- stat.ma(mice.data, mice.setup, norm="l")
>
```

   Check `attributes(mice.lratio)` and have a look at the first 10 rows of each matrix in the list:

```
> mice.lratio$M[1:10,]
> mice.lratio$A[1:10,]
>
```

   (b) Within-slide normalization: print-tip location and scale. For cDNA microarrays, we usually need to substract spatial effects and normalize by print-tip groups. It is not always necessary to perform scale normalization as well as location normalization, so we will consider print-tip location normalization only first. The function `plot.print.tip.lowess` fits a lowess curve separately to each print-tip group, so let's do this for the unnormalized data for slide 4:

```
> plot.print.tip.lowess(mice.data, mice.setup, norm="n", image.id=1, pch=20)
>
```


Setting `norm="p"` produces the same plot for the log ratios normalized by print-tip groups, and `norm="s"` performs location and scale normalization.

Within-slide normalization needs to be performed for *each slide* in the experiment.

Comparing print-tip location normalization with print-tip scale and location normalization

```
> par(mfrow=c(1,2))
> mice.lratiop <- stat.ma(mice.data, mice.setup, norm="p")
> p1 <- mice.lratiop$M[,1]
> plot.scale.box(p1, mice.setup)
> mice.lratios <- stat.ma(mice.data, mice.setup, norm="s")
> s1 <- mice.lratiop$M[,1]
> plot.scale.box(s1, mice.setup)
>
```

suggests that location and scale normalization is better for these data. Note that the command `par(mfrow=c(1,2))` produces side by side graphs that are easy to compare but look somewhat crowded. The appearance is improved substantially by maximizing the graphics window.

Hence, we save an updated set of within-slide location and scale normalized log ratios by typing

```
> mice.lratio  <- stat.ma(mice.data, mice.setup, norm="s")
>
```

A Transcript of Spot commands for mice data

```
>  library(Spot)
IMPORTANT NOTICE

  COPYRIGHT Commonwealth Scientific and Industrial
Research Organisation ('CSIRO'), Australia 2001.

All right in this Software are reserved to CSIRO.
You are only permitted to have this Software in
your possession and to make use of it if you have
agreed to a Licence Agreement with CSIRO.

All enquiries for the use of this Software should
be referred to:

Dr Michael Buckley
CSIRO Mathematical and Information Sciences
```



```
Locked Bag 17, North Ryde, NSW 1670, Australia
email: Michael.Buckley@cmis.csiro.au
phone: +61 2 9325 3209
fax: +61 2 9325 3200

Attaching package 'Spot':


The following object(s) are masked from package:base :

 aperm as.array

Spot> SetTemplate("exp1", 2)
imview: no magic configuration file found (magic.mgk) [No such file
or directory].
Enter "Add point mode" in imview window and select
33 = 32 + 1 points as follows.

First select centre of top-left spot in each of the 32 grids.
Do first row, left-to-right,then second row, left-to-right, and so on.
Finally select centre of bottom-right spotin the last (i.e. bottom-right) grid.

When finished, press ENTER in this window to continue.
1:
Spot> spot.mice.data1 <- Spots("mice", 1)
Analysing images "mice1R.tif" and "mice1G.tif".
Reading image data ... done.
Combine R and G ... done.
Estimation of row positions.
Estimation of column positions.
imview: no magic configuration file found (magic.mgk) [No such file
or directory].
Get seeds ... done.
Seeded region growing ... done.
Getting spot and background values ... done.
Adding shape parameters ... done.

Spot> spot.mice.data2 <- Spots("mice", 2)
Analysing images "mice2R.tif" and "mice2G.tif".
Reading image data ... done.
Combine R and G ... done.
Estimation of row positions.
Estimation of column positions.
```



```
Get seeds ... done.
Seeded region growing ... done.
Getting spot and background values ... ls
done.
Adding shape parameters ... done.
Spot> spot.data3 <- Spots("mice",3)
Analysing images "slide3r.tif" and "slide3g.tif".
Reading image data ... done.
Combine R and G ... done.
Estimation of row positions.
Estimation of column positions.
Get seeds ... done.
Seeded region growing ... done.
Getting spot and background values ... done.
Adding shape parameters ... done.

(And so on ....)

Spot> mice.setup <- init.grid()
Enter number of rows of grids per image (ngrid.r):
Spot> 8
Enter number of columns of grids per image (ngrid.c):
Spot> 4
Enter number of rows of spots per grid (nspot.r):
Spot> 21
Enter number of columns of spots per grid (nspot.c):
Spot> 24
Initialization complete

Spot> mice.data <- init.data()
Are you creating a new data matrix or adding new array data
to a prexisting data matrix?
Enter "n" for creating  and "a" for adding new array data:
Spot> n
Do the names of all your datasets have the following format:
prefix1, prefix2, prefix3?, ... Here prefix can be any name,
but the suffixes must be integers 1,2, ..., # of arrays.
Enter "y" for yes, "n" for no:
Spot> y
Enter the prefix:
Spot> spot.mice.data
Enter the number of arrays to be processed:
Spot> 4
```



```
Enter the name of Cy3 raw data:
Spot> Gmean
Enter the name of Cy3 background:
Spot> morphG
Enter the name of Cy5 raw data:
Spot> Rmean
Enter the name of Cy5 background:
Spot> morphR
Finished creating the dataset.
```

