FGT: Data management, Complex Data Types and Plotting using R

Loading biological data- introduction

In this tutorial we will survey some of the common methods to import data into R. We will look at how to convert between complex biological object types used by R. This will allow data to be imported and connected with the downstream data analysis methods.

We will also look at some of the methods that can be used in R to generate useful data visualisations.

The tutorial exercise brings these two components together to look at how to import data and format it in a suitable form such that visualisations can be produced.

Downloading data from internet resources

Many data sets are available for download from web accessible databases, usually by FTP or by HTTP. Download can conveniently be performed using R.

Note: Before beginning the tutorial create a Unix directory to hold the files. Make this your current working directory. Copy `afile.txt` *and unzip the example* `example_CEL_files.zip` *from the* `/home/shared_files` *folder. Hint: use the* `Unix` *commands* `mkdir` *to create the folder* `cp` *to copy the files and* `unzip` *to unpack the zip archive.*

Once you have created the files, type `R` to start the R console.

```
dir() #list the current directory

#download the required file (here the University home page is used as
#a small example- but of course larger files can be downloaded.

 download.file(url="http://www.ed.ac.uk",destfile="ed.html")
 dir()# Check the file has been downloaded successfully
 file.show(<filename>) # <filename> is the name of a file here "ed.html")
```

A more general approach may also be taken to issue a Unix shell command (essentially a Unix command you could type on the command line). This uses the R command system.

```
dir() #check the contents of the working directory before

#issue the Unix command wget and download the University homepage

system("wget http://www.ed.ac.uk")

dir()# check for downloads!
```

Loading expression data from local files

The most commonly used approach to reading Affymetrix CEL files is to using `ReadAffy` which is part of the Bioconductor `affy` library. A list of files and a load path can also be specified. `ReadAffy` generates

an object of type AffyBatch in the R workspace (here called `mydata21`). In the overall Affymetrix analysis workflow AffyBatch files are commonly converted to ExpressionSet by normalisation with functions such as `rma` and `mas5`. For loading of large data sets (typically containing hundreds of arrays)

```
library(affy)

setwd(getwd())# typically this would be set to the current working directory
getwd()# check the directory is correct

mydata21 <-ReadAffy()# Read all CEL files in the local directory
```

It is also possible to load expression values from a text file and convert to and expression set afterwards

```
test <-read.table("afile.txt",sep="\t",header=TRUE,row.names=1,as.is=TRUE)

#turns test into a numerical matrix

mat1 <- as.matrix(test)

#Build expression set with numerical data

exprset1<- new("ExpressionSet",exprs=mat1)

ls() #Check objects created in the workspace
```

Commonly encountered data objects

A data frame is used to store tables and may contain elements of different types but all the vectors that make up the table are the same length.

```
A <- c(9, 8, 6)
B <- c("zz", "x", "vv")
C <- c(TRUE, TRUE, TRUE)
D <- c(1.2,1.3,0.2)
df<- data.frame(A,B,C,D)
df
```

Expression sets are complex objects that store both gene expression and phenotypic information in one composite object

```
#load an example ExpressionSet containing microarray gene expression data
#in data analysis this would be generated from imported and normalised
#microarray data data(sample.ExpressionSet) loads the sample data
```

```
data(sample.ExpressionSet)
#extract a numerical matrix from this object
mymatrix <-exprs(sample.ExpressionSet)
mymatrix[1:10,] #show the first 10 elements
#extract sample names
mysnames <-sampleNames(sample.ExpressionSet)
mysnames
```

Converting complex data types

A number of useful conversion functions are available that can be used to convert between different data types. Attempt to convert between another object (here a data frame) and a numerical matrix.

```
#attempt to convert type into a matrix

mtcars #this loads sample data that comes with R
matrix_mtcars<-as.matrix(mtcars)
matrix_mtcars
#to test if the conversion has been successful
is.matrix(mtcars)
is.matrix(matrix_mtcars)
matrix_mtcars <-as.matrix(mtcars)
mtcars
mtcars2<-data.frame(matrix_mtcars)
mtcars2
```

Simple Plots Scatter Plots

Here example data provided by R is used to generate a scatter plot from the x data matrix.

```
# convert example data to a matrix
x <- as.matrix(mtcars) #give your variables descriptive names not x!!

# plot the values of two columns against each other, say miles/gallon (mpg) vs
horsepower (hp)
plot(x[,'mpg'],x[,'hp'])
# customise plot appearance, e.g. change the colour and shape of the dots and
the axis labels:
plot(x[,'mpg'],x[,'hp'],col='blue',pch=16,xlab='MPG',ylab='HP')
# plot each column against every other column
pairs(x)
```

Histogram

Example simple histogram plots available in R, examples from mtcars package.

```
# load example data:
x <- as.matrix(mtcars)
# histogram of all values in x:
hist(x)
# or only of the miles-per-gallon (MPG)
hist(x[,'mpg'])
```

Many properties can be set to customise data displayed and the appearance of each histogram plot

```
# You can change the title, colour, etc. of these plots easily with
additional parameters:
hist(x[,'mpg'], main='MPG', col='grey')
# we might want to split the histogram by different categories, say the
number of cylinders:

par(mfrow=c(3,1))
for(cyl in unique(x[,'cyl']) ){
 selection <- which(x[,'cyl'] == cyl)
 hist(x[selection,'mpg'],main=cyl,col='grey',xlim=c(10,35))
}
```

More powerful tools to split and organise plots are provided by the lattice library.

```
# the lattice library provides an alternate way to plot data
#plots by an annotation column:

library(lattice)
histogram(~ mpg | factor(gear), data=as.data.frame(x))
```

Heatmaps

Useful for visualisation of complex two dimensional data such as microarray gene expression and RMA profiling data.

```
# load example data:
x<-as.matrix(mtcars)
# simple heatmap:
png("heatmap.png") #Here the heatmap is written to a file
heatmap(x)
dev.off()
```

```
# more functionality is provided by the heatmap.2 command in the gplots library:
library(gplots)
png("heatmap2.png")
heatmap.2(x)
dev.off()
# an alternative to heatmap.2 is pheatmap (= 'pretty' heatmap):
library(pheatmap)
png("heatmap3.png")
pheatmap(x)
dev.off()
```

Graphs

```
library(ggplot2) #loads library and diamonds example data
head(diamonds) #look at the first few elements in an object,
png("graph.png")
qplot(clarity, data=diamonds, geom="bar", group=cut, colour=cut)
dev.off()
```

Complex Heatmaps

Using heatmap.2, heatmaps can be customised in many ways, e.g.

-different colours (here: blue-white-red)
-no trace information (trace='none')
-clustering only by row (Colv=NA,dendrogram='row')
-scale colours by column instead of row (scale='column')

```
###################### ADVANCED: CUSTOMISED HEATMAPS #################
library(gplots)
png("advanced_heatmap.png")
heatmap.2(x, col=colorRampPalette(c('blue','white','red')),
trace='none', Colv=NA, dendrogram='row', scale='column')
dev.off()
```

Complex Scatter Plots

```
# load example data: this example is from the mtcars library
x <- as.matrix(mtcars)

# plot MPG vs HP (as before, but divide HP by 10 to make data more comparable)
# plot with 'worse than average MPG' in red (the rest in black):

png("scatter_adv.png")
plot(x[,'mpg'],x[,'hp']/10,col=ifelse(x[,'mpg']>=mean(x[,'mpg']),'red',
'black'),pch=16,xlab='MPG',ylab='HP')

# add a legend:
legend('topright',c('Better than average MPG','Worse than average
MPG'),fill=c('red','black'),bg='white')
dev.off()
```

Exercise Question

Load the example file "afile.txt" into R and convert this file to an expression set and a numerical matrix. Try to visualise this data using the code provided. **Exit to Unix and use `display <filename>` to view images or download the images and view on your local machine…**