FGT: Statistical analysis of microarray gene expression data

Getting Started

In this tutorial we will load an RMA normalised expression data set from our example MOE430v2 Affymetrix experiment. We will then perform a statistical analysis on the data using the Bioconductor package Limma. After Limma analysis we will generate a list of differentially expressed genes that will then be annotated.

Load the Tutorial Files

- Login to your account
- Create a directory for this week's files e.g. `mkdir tutorial4`
- Change working directory to this folder e.g. `cd tutorial4`
- Copy the tutorial data into this folder e.g. `cp /shared_files/fold_filtering.Rdata .`
- Start R

```
## Commands to download and install the package
## For the purposes of this tutorial, the package is already
## installed in your R environment
## Copy this code and uncomment the three commands below to
## install the required files on another machine
#load the required libraries for the tutorials...
library(affy)
library(limma)
library(mouse4302.db)
library(annotate)
```

Now load the R data objects from the saved file

```
#check the tutorial data has loaded into the current
#working directory.
dir()
#You now need to load the .Rdata file into
#your workspace to recover the saved files
load("fold_filtering.Rdata")
#Check that you have the required objects
ls()
```

Reminder: Affy and Limma Setup

The affy package contains methods for the processing of oligonucleotide arrays. It has been a part of the Bioconductor suite since its first release in 2002. The affy package provides a range of statistical analysis methods that cover all aspects of the analysis pipeline including data input, quality control, data normalization and plotting.

For a detailed documentation of the affy package,
please check the Reference Manual available on the Bioconductor website (http://www.bioconductor.org/packages/release/bioc/manuals/affy/man/affy.pdf).
In order to install the affy package, you need to install R and some related Bioconductor packages. For your reference, this can be easily done by the following commands. There is no need to do so presently, as the required packages have been already installed in the R environment of this tutorial. The Limma package is one of the most commonly used libraries for the statistical analysis of gene expression data. Current Limma documentation can be found:-
http://www.bioconductor.org/packages/release/bioc/html/limma.html

For this tutorial the examples should be run in sequential order. Data outputs generated in earlier sections are assumed to be present for later section examples.

Source of Tutorial Data

The affy package has one major class: AffyBatch. The AffyBatch class is a representation of the Affymetrix GeneChip probe level data and extends the container class eSet. To create an AffyBatch object which contains the expression values of all the CEL files in your working directory you can use the wrapper function ReadAffy().

Here we just use pre-saved example data from the experiment (GSE10806: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE10806).

We will stick with this dataset for the moment, but for your own reference, you can browse a collection of freely available datasets in the Gene Expression Omnibus (GEO, http://www.ncbi.nlm.nih.gov/geo/).

You do not need to download this data again-

Use of a Targets File

The targets file contains information about the different experiment conditions for each sample. We can create a targets file by annotating the samples according to the information available i.e. on GEO or your own experiment. For our specific dataset, the targets file looks as follows:

Name	Filename
ESC.1	GSM272753.CEL
ESC.2	GSM272836.CEL
ESC.3	GSM272837.CEL
iPSC2.2	GSM272839.CEL
iPSC2.3	GSM272846.CEL

NSC.1	GSM272847.CEL
NSC.2	GSM272848.CEL
iPSC2.1	GSM272890.CEL
iPSC4.1	GSM279200.CEL
iPSC4.2	GSM279201.CEL
iPSC4.3	GSM279202.CEL

Tidy-Up &Rename Samples

It is helpful to rename the samples in the Expression Set to make the analysis easier to follow. It is also helpful to reorder the samples into replicate order, rather than the order the samples load by default.

```
#Check original sample order
sampleNames(eset)
#Rename the samples
sampleNames(eset) <-
c("ESC.1","ESC.2","ESC.3","iPS2.2","iPS2.3","NSC.1","NSC.2","i
PS2.1","iPS4.1","iPS4.2","iPS.3")
#Check the samples have renamed
sampleNames(eset)
```

Annotate the Results with Gene Names

```
#establish annotation for MOE430v2
#which annotation do we need
#modified from
#http://gettinggeneticsdone.blogspot.co.uk/2012/01/annotating-limma-
#results-with-gene.html

eset@annotation

library(mouse4302.db)# load chip-specific annotation
#packages in the annotation package
ls("package:mouse4302.db")



#build an annotation table

ID <- featureNames(eset)

Symbol <- getSYMBOL(ID, "mouse4302.db")

Name <- as.character(lookUp(ID, "mouse4302.db", "GENENAME"))

tmp <- data.frame(ID=ID, Symbol=Symbol, Name=Name,
stringsAsFactors=F)

tmp[tmp=="NA"] <- NA #fix padding with NA characters

#assign as feature data of the current Eset

fData(eset) <- tmp
```

Statistical Analysis Using Limma

Limma uses linear models to analyse microarray data, requiring either one or two matrices to be specified. For the purposes of this tutorial we can view this approach as a flexible way to specify a microarray analysis.

Design Matrix

The Design Matrix indicates which samples have been applied to each array. It is created using the function model.matrix() using a 'factor' object. This sets up a matrix with a column for each 'level' (the different values that the factor takes) and a row for each element of the factor.

In the example, a design matrix has been made for an experiment with 11 chips of three sample types (three chips each, except for one group of two). Each element of the matrix takes a value of 1 if the element of the factor corresponding to the row of the matrix has the value corresponding to the column of the matrix.

Setting the column names to memorable names makes constructing the contrast matrix easier.

```
#Build the design matrix
design <- model.matrix(~-1+factor(c(1,1,1,2,2,3,3,2,4,4,4)))
colnames(design) <- c("ESC","iPS2","NSC","iPS4")
#Check it makes sense
sampleNames(eset)
#output the design matrix
design
```

This code produces the following design matrix- notice how the sample groups are represented.

```
     ESC  iPS2  NSC  iPS4
1     1     0    0     0
2     1     0    0     0
3     1     0    0     0
4     0     1    0     0
5     0     1    0     0
6     0     0    1     0
7     0     0    1     0
8     0     1    0     0
9     0     0    0     1
10    0     0    0     1
11    0     0    0     1
```

Understanding the Design Matrix

This section is for information only- do not modify your design matrix… continue with the R commands in the next section to make your Contrasts Matrix

NB It makes sense that in order to identify differentially expressed genes from the data set we need first to specify which sample belongs to each group of cell lines. This is specified by the

```
design <- model.matrix(~-1+factor(c(1,1,1,2,2,3,3,2,4,4,4)))
```

The "~-1+" term looks a little odd. In R notation the model formulae are specified by the general notation `response ~model`. In this case the response are the gene profiles and the model the formulae we use to predict this response. We are just specifying the model part of this formulae. Here we are looking at factors (different cell types) but Limma can also be used to model lots of other relationships- eg identify genes that change in response to a specific drug doses. For factors Limma's default behaviour which is to specify a design where Limma compares group to a global mean. This can be seen if we specify

`design <- model.matrix(~factor(c(1,1,1,2,2,3,3,2,4,4,4)))` but then the design looks like

```
1    1    0    0    0
2    1    0    0    0
3    1    0    0    0
4    1    1    0    0
5    1    1    0    0
6    1    0    1    0
7    1    0    1    0
8    1    1    0    0
9    1    0    0    1
10   1    0    0    1
11   1    0    0    1
```

Note that then the first grouping (after the sample numbers) includes all samples and so represents the mean level of each gene across all samples- not what we want here!

```
design <- model.matrix(~0+factor(c(1,1,1,2,2,3,3,2,4,4,4)))
```

Note also that R the above notation is equivalent notation to that we have used in class and so will also remove the global mean. This latter notation is used in the Limma userguide- an excellent source of further information.

Note that there is a very useful book chapter "Limma: Linear Models for Microarray Data" available online [http://www.statsci.org/smyth/pubs/limma-biocbook-reprint.pdf] that describes how to specify the design for other designs including two-colour and for experiments with multiple factors.

Contrasts Matrix

The Contrasts Matrix specifies which comparisons are to be made between the samples. This is not needed for the simplest experiments where only ‘test’ vs ‘control’ comparisons are to be drawn.

This example contrast matrix will tell Limma to look for differentially expressed genes between ES and the other cell types.

```
#This instructs Limma which comparisons to make
contrastmatrix <- makeContrasts(ESC-iPS2,ESC-NSC,ESC-iPS4,
levels=design)
contrastmatrix
```

The contrastmatrix object specifies the comparions that will be made.

```
Levels ESC - iPS2 ESC - NSC ESC - iPS4
  ESC            1         1          1
  iPS2          -1         0          0
  NSC            0        -1          0
  iPS4           0         0         -1
```

Fit the Linear Model

Once the design and contrast matrices have been created, a model is simply fitted. For expression matrix exprs, design matrix design and contrast matrix contrast.matrix

```
#issue these commands to fit the model
#and make the contrasts
fit <- lmFit(eset, design)

fit2 <- contrasts.fit(fit, contrastmatrix)

#this last part essentially moderates the t-statistic using
#the borrowed variance approach described in class
fit2 <- eBayes(fit2)
```

Writing a Summary Table

For expression matrix exprs, design matrix design and contrast matrix contrast.matrix. A list of the top differentially expressed genes for the first comparison specified by the contrast matrix can be obtained using the topTable() command:

```
topTable(fit2,coef=1,adjust="fdr")

myresults <-topTable(fit2,coef=1, adjust="fdr",
number=nrow(eset))

write.table(myresults,"myresults.txt")
```

Here `coef` specifies which column of the contrast matrix to use for the comparison and adjust specifies what statistical adjustments to use (in this case 'false discovery rate').

Classify each gene according to the pair-wise comparisons specified in the contrast matrix and display the results.

```
clas <- classifyTestsF(fit2)

vennDiagram(clas)
```

Limma Output Table

This section is for information to help with the interpretation of the output.

	logFC	AveExpr	t	P.Value	adj.P.Val	B
1421144_at	5.18	10.72	39.80	0.00	0.00	21.16
1437967_at	3.89	5.69	28.67	0.00	0.00	18.53
1425137_a_at	-3.65	5.60	-28.25	0.00	0.00	18.40
1425427_at	-3.43	4.07	-27.77	0.00	0.00	18.24
1425220_x_at	-3.55	6.18	-27.71	0.00	0.00	18.22

- The first column contain the row identifier – here the Affymetrix ProbeID
- logFC is the average fold change between the two compared conditions
- AvgExpr is the average expression between the two conditions
- t is the moderated t-statistic- adjusted from the model
- P.Value is the p-value
- Adj.P.Val is the adjusted p-value for Multiple Correction here we have used FDR correction
- B is the log odds that a gene is differentially expressed. So if B=1.5, odds of differential expression is exp(1.5)=4.8 ie about four and a half to one.

More details are provided in the Limma manual and from the course references.