Aims

These exercises include the following three sessions:

Session 1: Identifying differentially expressed genes
Session 2: Probe annotation information
Session 3: Gene set tests

Libraries needed:

```
# Do not run
#source("http://bioconductor.org/biocLite.R")
#biocLite()
#biocLite(c("ecoliLeucine", "estrogen", "hgu95av2cdf", "org.Hs.eg.db", "GOstats", "goTools",
#"gplots", "arrayQualityMetrics", "biomaRt"))
```

The following packages hgu95av2.db, KEGG.db, GO.db are also needed below, but they are part of the default biocLite() download and we therefore do not need to download them explicitly.

Session 1: Identifying differentially expressed genes

This case study illustrates more advanced linear modeling with Affymetrix single-channel microarrays.

1.1 Required data

The estrogen data set is required for this exercise. This data is from the estrogen data package on the Bioconductor website http://www.bioconductor.org/data/experimental.html:

```
library(estrogen)
datadir <- system.file("extdata", package = "estrogen")
setwd(datadir)
```

By typing dir() you should see nine CEL files and three text files:

```
dir()
```

To repeat this case study in full you will need to have the R packages limma, affy, hgu95av2cdf and hgu95av2.db installed.

1.2 Estrogen experiment

The data gives results from a 2x2 factorial experiment on MCF7 breast cancer cells using Affymetrix HGU95av2 arrays. The factors in this experiment were estrogen (present or absent) and length of

exposure (10 or 48 hours). The aim of the study is the identify genes which respond to estrogen and to classify these into early and late responders. Genes which respond early are putative direct-target genes while those which respond late are probably downstream targets in the molecular pathway.

1.3 Read the data

The first step in most analyses is to read the targets file which describes what RNA target has been hybridized to each array and, equally importantly, gives the names of the corresponding data files:

```
library(limma)
targets <- readTargets("targLimma.txt")
targets
```

Now read the CEL file data into an AffyBatch object and normalize using RMA:

```
library(affy)
library(hgu95av2cdf)

abatch <- ReadAffy(filenames=targets$FileName)
eset <- rma(abatch)
eset2 <- rma(abatch,normalize=F,background=F)
```

Here eset is a data object of class exprSet.

1.4 Quality assessment of data

It is usual and appropriate to check data quality before continuing your analysis. The arrayQualityMetrics function performs quality metrics on ExpressionSet, AffyBatch or BeadLevelList or NChannelSet containing microarray data from any platforms, one or two channels. The results, presented in a HTML report, are designated to allow the user to rapidly assess the quality of a set of arrays:

```
library(arrayQualityMetrics)

fname <- ?????  #Specify the output directory e.g. "C:/Projects/BioC course/QC"
arrayQualityMetrics( expressionset = eset, outdir = fname)
arrayQualityMetrics( expressionset = eset2, outdir = "C:/Projects/BioC course/QC2")
```

Task: Are there any quality problems with these arrays? Explain.

Solution: These plots show no significant quality problems with any arrays in this dataset? Perhaps an artifact. Compare boxplots before normalization.

1.5 Create a design matrix

We have four pairs of replicate arrays so we should estimate four parameters in the linear model. There are many valid ways to choose a design matrix, but perhaps the simplest is to make each column

correspond to a particular treatment combination. In this formulation, the four columns of the matrix correspond to absent10, present10, absent48 and present48, respectively. This design matrix given above can be computed in R as follows:

```
f <- paste(targets$estrogen,targets$time.h,sep="")
f <- factor(f)
f

design <- model.matrix(~0+f)
colnames(design) <- levels(f)
design
```

1.6 Fit the linear model

Now that we have defined our design matrix, fitting a linear model is as simple as:

```
fit <- lmFit(eset, design)
names(fit)
```

fit is an object of class MArrayLM.

The fitted coefficents fit$coef from the model fit are just the mean log-expression for each treatment combination for each probe set. For this reason, this choice of design matrix is called the *group means parameterization* in the Limma User's Guide.

1.7 Define a contrast matrix

The idea now is to use contrasts to make any comparisons of interest between the four treatment combinations. Contrasts are linear combinations of the coefficients from the linear model fit. We will estimate three contrasts (so our contrasts matrix will have three columns). The first contrast is an estrogen effect at time 10 hours, the second as an estrogen effect at time 48 hours and the third is the time effect in the absence of estrogen.
The makeContrast function is helpful for construction a contrast matrix:

```
?makeContrasts
```

Task: Make the contrast matrix

Solution:

```
cont.matrix <- makeContrasts(E10="present10-absent10",E48="present48-
absent48",Time="absent48-absent10",levels=design)
cont.matrix
```

1.8 Extract the linear model fit for the contrasts

```
fit2  <- contrasts.fit(fit, cont.matrix)
fit2  <- eBayes(fit2)
```

1.9 Assessing differential expression

We now use the function topTable to obtain a list of genes differentially expressed between Estrogen-Present and Estrogen-Absent at time 10 hours, followed by a list of genes differentially expressed between Estrogen-Present and Estrogen-Absent at time 48 hours:

```
colnames(fit2)
topTable(fit2,coef=1)
topTable(fit2,coef=2)
```

The function decideTests() provides a variety of ways to assign statistical significance to the contrasts while controlling for multiple testing:

```
results <- decideTests(fit2)
summary(results)
vennDiagram(results)
```

1.10 Create plots of the expression data

The exprs function extracts expression data from the eset object:

```
?exprs
```

Extracting expression values for all probes:

```
expr <- exprs(eset)
```

Extracting expression values for the first 10 probes:

```
expr2 <- exprs(eset[1:10,])
```

Task: Create a heatmap of the expression values for the 100 top genes. Use the heatmap function in the library stats or the heatmap.2 function in the library gplots:

Solution:

```
library(gplots)
```

```
top1 <- topTable(fit2,coef=1,number=100)
rws <- row.names(top1)
rws <- as.integer(rws)
expr <- exprs(eset)
heatmap(expr[rws,])
heatmap.2(expr[rws,])
```

Task: Make a density plot and histogram of the t-statistics and the un-adjusted p-values from the fit2 object.

```
hist(fit2$F.p.value)
hist(fit2$t)

plot( density(fit2$F.p.value))
plot( density(fit2$t) )
```

Session 2: Probe annotation information

The Bioconductor project provides software for associating microarray and other genomic data in real time to biological metadata from web databases such as GenBank, LocusLink and PubMed (annotate package). Functions are also provided for incorporating the results of statistical analysis in HTML reports with links to annotation WWW resources. Software tools are available for assembling and processing genomic annotation data, from databases such as GenBank, the Gene Ontology Consortium, LocusLink, UniGene, the UCSC Human Genome Project (AnnotationDbi package). Data packages are distributed to provide mappings between different probe identifiers (e.g. Affy IDs, LocusLink, PubMed). Customized annotation libraries can also be assembled.

Use of Bioconductor annotation for Affymetrix arrays is illustrated below. We will use alternative approaches to obtain probe annotation.

2.1 Probe annotation information using array specific annotation package

The purpose of the hgu95av2.db package is to provide detailed information about the hgu95av2 platform. First load the annotation package hgu95av2.db:

```
library(hgu95av2.db)
```

Try different options for displaying information about the content of the package:

```
library(hgu95av2.db)
ls("package:hgu95av2.db")

head(ls("package:hgu95av2.db"), n = 10)

hgu95av2()

?hgu95av2UNIGENE

head(toTable(hgu95av2UNIGENE))
```

We will now use some of the functions provided by the annotate package. The basic purpose of this package is to supply interface routines for getting data out of specific meta-data libraries (e.g. hgu95av2.db). First load the library:

```
library(annotate)
```

It is easy to get information about individual probes or a list of probes using the get/mget functions:

```
get("38187_at", hgu95av2GENENAME)
```

```
affyid <- c("38187_at", "38912_at", "33825_at", "36512_at", "38434_at")

mget(affyid, hgu95av2GENENAME)
```

Task: Try adding more annotation to the fit2 object generated in the linear model analysis described above. Add gene symbol and Entrez gene id:

Solution:

```
fit2$genes$Symbol <- getSYMBOL(fit2$genes$ID, "hgu95av2")
topTable(fit2, coef=2)
fit2$genes$EG <- getEG(fit2$genes$ID, "hgu95av2")
topTable(fit2, coef=2)
```

Task: How many probes do not have a gene symbol?

Solution:

```
sum(is.na(fit2$genes$Symbol))
sum(!is.na(fit2$genes$Symbol))
```

2.2 Creating weblinks

The annotate packages also provide functions that support querying the different web services provided by the National Library of Medicine (NLM) and the National Center for Biotechnology Information (NCBI). Using these functions it is possible to create hyperlinks to different web resources, from a gene list.

First load the vignette using the following command:

```
vignette("prettyOutput","annotate")
```

Task: By following the steps provided in the vignette for annotate try to create an html page that present results from your limma analyses.

Solution:

From the annotate package we can use the getQueryLink function to generate the web link. First read about the function:

```
? getQueryLink
```

Then generate a hyperlink for the gene with entrez gene id 5875:

getQueryLink("5875",repository="en")

the output of this function is a webaddress that link to PubMed entrez gene.

Task: The function also takes a vector of ids as input. Try using the entrez genes ids we added to the fit2 object in the previous section and create weblinks for all the probes on the array:

Solution:
entrezWeblink <- getQueryLink(fit2$genes$EG,repository="en")

It is also possible to create hyperlinks to different web resources using Excel. Below is a function that creates a hyperlink in Excel format. Copy and paste the function into your R prompt:

```
makeExcelHyperlink <- function( hyperlink=NULL, fullname=NULL, excelSep=";")
# excelSep=";" or excelSep=","
{
 if ( is.null(fullname) ) { fullname="hyperlink"}
 hyperlink <- paste( "=HYPERLINK(",' " ',hyperlink ,' " ',excelSep,' " ',fullname,' " ',')' ,sep="")
 return(hyperlink)
}
```

makeExcelHyperlink

Note: There were some problems getting this function to work during the exercises. The reason was that on the hand out the highlighted ' " ' looked like 2 double quotes. If you copy and paste the function into R it works.

entrezHyperlink <- makeExcelHyperlink(hyperlink=entrezWeblink, fullname= fit2$genes$EG)
head(entrezHyperlink)

Task: Add the hyperlink in excel format to the fit2 object created by limma:

Solution:
fit2$genes$entrezHyperlink <- entrezHyperlink

Task: Write the output from topTable to a tab delimited file:

Solution:
top1 <- topTable(fit2,coef=1, number=100)
filename <- "C:/Projects/BioC course/top1.xls"
write.table(top1, file=filename, quote=F, sep='\t', col.names=T, row.names=F)
write.table(top1, file="C:/Projects/BioC course/top2.xls", quote=F, col.names=T, row.names=F)

Open the file and test that the hyperlinks work (requires that you are online). If the hyperlink does not work then you should check if the excelSep argument (default is ";") is the correct one for your version of Excel.

2.3 Species specific annotation packages

```
library(org.Hs.eg.db)
library(KEGG.db)

caff <- get("Caffeine metabolism",
        revmap(KEGGPATHID2NAME))

get(caff, revmap(org.Hs.egPATH))
```

Task: Which gene symbols and gene names are associated with the following entrez gene Ids, 1544, 1548 and 1549?

Solution:
```
mget(c("1544", "1548", "1549"), org.Hs.egSYMBOL)
mget(c("1544", "1548", "1549"), org.Hs.egGENENAME)
```

2.4 The GO.db and KEGG.db annotation packages

The GO.db and KEGG.db annotation packages provides detailed information about the latest version of the Gene Ontologies and the KEGG pathway databases. You can learn what objects these packages supports with the following commands:

```
library(GO.db)
library(KEGG.db)

ls("package:GO.db")
ls("package:KEGG.db")
```

Each of these objects has their own manual page detailing where relevant data was obtained along with some examples of how to use it.

Retrieve the KEGG id for "caffeine metabolism" and identify which probes are associated with

```
get("Caffeine metabolism", revmap(KEGGPATHID2NAME))
```

Task: Create a heatmap of the expression values for the KEGG pathway 00232. Use the heatmap function in the library stats or the heatmap.2 function in the library gplots:

Solution:
affyids <- get("00232", revmap(hgu95av2PATH))
expr <- exprs(eset)[affyids,]
heatmap(expr)

2.5 Using biomaRt (this requires online access)

biomaRt provides an interface to a growing collection of databases implementing the BioMart software suite (http://www.biomart.org). The package enables retrieval of large amounts of data in a uniform way without the need to know the underlying database schemas or write complex SQL queries. Examples of BioMart databases are Ensembl, Uniprot, Gramene, Wormbase and HapMap. These major databases give biomaRt users direct access to a diverse set of data and enable a wide range of powerful online queries from gene annotation to database mining.

Selecting a BioMart database
Every analysis with biomaRt starts with selecting a BioMart database to use. A first step is to check which BioMart web services are available. The function listMarts will display all available BioMart web services:

```
library(biomaRt)
listMarts()
```

The useMart function can now be used to connect to a specified BioMart database, this must be a valid name given by listMarts. The first set of exercises is on the Ensembl BioMart so let's connect to Ensembl:

```
ensembl <- useMart("ensembl")
```

Selecting a dataset
BioMart databases can contain several datasets, for Ensembl every species is a different dataset. In a next step we look at which datasets are available in the selected BioMart by using the function listDatasets:

```
listDatasets(ensembl)
```

Let's use the Homo sapiens dataset of Ensembl. The dataset argument of the useMart function enables dataset selection:

```
ensembl <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
```

How to build a biomaRt query

The getBM function has three arguments that need to be introduced: filters, attributes and values. Filters define a restriction on the query. For example you want to restrict the output to all genes located on the human X chromosome then the filter chromosome name can be used with value 'X'. The listFilters function shows you all available filters in the selected dataset:

```
filters <- listFilters(ensembl)
filters[1:5, ]
```

Attributes define the values we are interested in to retrieve. For example we want to retrieve the gene symbols or chromosomal coordinates. The listAttributes function displays all available attributes in the selected dataset:

```
attributes <- listAttributes(ensembl)
attributes[1:5, ]
```

The getBM function is the main query function in biomaRt. It has four main arguments:
• attributes: is a vector of attributes that one wants to retrieve (= the output of the query).

• filters: is a vector of filters that one will use as input to the query.

• values: a vector of values for the filters. In case multple filters are in use, the values argument requires a list of values where each position in the list corresponds to the position of the filters in the filters argument (see examples below).

• mart: is an object of class Mart, which is created by the useMart function.

Now that we selected a BioMart database and dataset, and know about attributes, filters, and the values for filters; we can build a biomaRt query. Let's make an easy query for the following problem: We have a list of Ensembl gene identifiers and we want to retrieve the corresponding EntrezGene identifiers and the human gene symbols (hgnc). The ensembl gene id will be the filter for this query and as values for this filter we use our list of Ensembl gene identifiers. As output (attributes) for the query we want to retrieve the EntrezGene, the hgnc symbols and Ensembl gene identifiers so we get a mapping of these three identifiers as a result. The exact names that we will have to use to specify the attributes and filters can be retrieved with the listAttributes and listFilters function respectively.

Let's now run the query:

```
 ensemblids <- c("ENSG0000021302", "ENSG00000105722", "ENSG00000130303")

 getBM(attributes = c("ensembl_gene_id", "entrezgene", "hgnc_symbol"), filters = "ensembl_gene_id",
values = ensemblids, mart = ensembl)
```

Above we used the get function on the array specific package hgu95av2 to obtain the mappings between the KEGG id 00232 and the probe ids:

get("00232", revmap(hgu95av2PATH))

Task: If you are online do the same using biomaRt.

3. Gene Set Tests

In this lab, we move beyond the analysis of individual genes, and consider sets of genes in microarray experiments. We consider a particular approach called gene set enrichment. We begin with a known set of genes and then test whether this set as a whole is differentially expressed in a microarray experiment.

We will again use the Estrogen data set through the fit2 linear model fit object.

3.1 A single gene set from a previous study

The gene set test can be used to test whether previous author's lists of differentially expressed genes are also differentially expressed in a current experiment similar to that of the previous authors. The following gene set, is from Jin et al (2003), and contains genes which have been experimentally verified to be ER-regulated. These IDs are Entrez gene Ids.

myGset <-
```
c(183,9429,596,596,672,1312,1509,1103,2161,718,9167,1545,2202,3206,4057,3934,1728,5
016,5020,5241,27043,5757,5914,5272,5054,7015,7022,7031,7706,2353,3265)
```

This gene set should be differentially expressed between the breast cancer cells with estrogen reintroduced and the serum-starved breast cancer cells with no estrogen, because in the cells reintroduced to estrogen, the estrogen receptors (ERs) will bind the estrogen and as a result become activated, gaining the ability to regulate gene expression in the cells, hence resulting in differential expression between the cells with and without estrogen.

Now we want to add the entrez gene IDs to the fit2 object

```
library(hgu95av2.db)
library(annotate)

fit2$genes$EG <- getEG(fit2$genes$ID, "hgu95av2")
```

Then we identify the rows in the fit2 object that are associated with this set.

```
inSet <- fit2$genes$EG %in% myGset
```

Task: Have we picked the right rows/genes?
Solution:
fit2$genes$EG[inSet]

We will use the geneSetTest function in the Limma package for gene set test. Read more about the function using:

```
?geneSetTest
```

We first perform a gene set test for the first contrast which is an estrogen effect at time 10 hours. This can be done like this:

geneSetTest(inSet, fit2$t[,"E10"])

We can see the results graphically:

boxplot(fit2$t[,"E10"] ~ inSet)

barplot(fit2$t[inSet,"E10"])

Task: Do the gene set test for the other comparisons taking into account the importance of direction

Solution:
fit2$genes$EG[inSet]
geneSetTest(inSet, fit2$t[,"E10"], alt="up")
geneSetTest(inSet, fit2$t[,"E10"], alt="down")
geneSetTest(inSet, fit2$t[,"E10"], alt="mixed")
geneSetTest(inSet, fit2$t[,"E48"], alt="up")
geneSetTest(inSet, fit2$t[,"E48"], alt="down")
geneSetTest(inSet, fit2$t[,"E48"], alt="mixed")
geneSetTest(inSet, fit2$t[,"Time"], alt="up")
geneSetTest(inSet, fit2$t[,"Time"], alt="down")
geneSetTest(inSet, fit2$t[,"Time"], alt="mixed")

Any indications that direction is important?

3.2 Gene sets from GO and KEGG

Another approach is to form gene sets based on a priori knowledge of common biological features shared by the genes. Such sets can be defined using the GO and KEGG databases.

First we retrieve the gene sets based on gene ontology (GO) from the hgu95av2 annotation package

gset <- as.list(hgu95av2GO2PROBE)

and use the first gene set in the list for the analyses:

```
myGset <- gset[[1]]
inSet <- fit2$genes$ID %in% myGset
geneSetTest( inSet, fit2$t[,"E10"])
```

The annotation package provides access to data in the GO package. The data are assembled from publically available data from the Gene Ontology Consortium (GO), www.go.org. Given a list of GO identifiers they access the children (more specific terms), the parents (less specific terms) and the terms themselves.

Try:

? getGOTerm

Task: What are the GO ID, GO term and GO Ontology associated with the first gene set?

Solution:
names(gset)[1]
getGOTerm(names(gset)[1])
getGOOntology(names(gset)[1])

Instead of using a single gene set we can use the first 100 gene sets from GO in the gene set analyses. In order to do this we can simply subset the list

```
gset <- as.list(hgu95av2GO2PROBE)
gset <- gset[1:100]
```

and removes the gene sets with less than 5 probes and more than 500 probes

```
gset <- gset[lapply(gset,length)>5]
gset <- gset[lapply(gset,length)<500]
```

Task: How many gene sets are left?

Solution:
length(gset)

For the contrast in which is an estrogen effect at time 10 hours we will run the gene set analyses and save the results to tab delimited file. First we create a vector for storing the p value output from the geneSetTest function:

```
pE10 <- rep(NA,length=length(gset))
```

and run the geneSetTest function inside a loop:

```
for ( i in 1:length(gset) ) {
```

```
 myGset <- gset[[i]]
  inSet <- fit2$genes$ID %in% myGset
  pE10[i] <- geneSetTest( inSet, fit2$t[,"E10"], alt="mixed" )
}

goterm <- lapply( names(gset),function(x) { unlist(getGOTerm(x) )} )
goterm <- unlist(goterm)
goont <- lapply( names(gset),function(x) { unlist(getGOOntology (x) )} )
goont <- unlist(goont)

ngset <- unlist( lapply(gset,length) )
```

Above we used the annotate library to generate weblinks for Entrez gene IDs. It is also possible to make weblinks to GO website:

```
goWeblink <- getQueryLink(names(gset),repository="GO")

goHyperlink <- makeExcelHyperlink( hyperlink=goWeblink, fullname= names(gset))
```

Finally we join all information and write to file

```
gsea <- data.frame( goHyperlink, names(gset), pE10,ngset,goterm,goont)
fname <- "C:/Projects/BioC course/gsea.xls"
write.table(gsea, file=fname,quote=F,sep='\t', dec=".", row.names=F)
```