



## ***Using Our Discipline to Enhance Human Welfare August 7 - 11, 2005***

### **Lab 7 - Gene ontologies and gene sets in analysis of differential expression**

**James Wettenhall. August 6-7, 2005**

## **1. Software required for this lab.**

You will need R 2.1.0 or 2.1.1 (<http://www.R-Project.org>) for this lab. This section lists all of the R packages you need to have installed and also lists some additional R packages which are recommended. Note that most of the R packages can be installed from the Bioconductor site automatically using:

```
source("http://www.bioconductor.org/getBioC.R")
getBioC()
```

but you are expected to use a more recent version of the `limmaGUI`, `limma`, `marray` and `arrayQuality` packages for this workshop, which is the reason for the alternate (non-Bioconductor) URLs for these packages below.

### **1.1 Required R packages**

It is highly desirable to have these R packages in a directory in which you have write permission. You can use `.libPaths("C:/Custom/R/library/directory")` or `.libPaths("C:\\Custom\\R\\library\\directory")` before you run `install.packages()` (or equivalent) to install the package(s) in a customized directory location.

Package	Windows	MacOS X	Source
<b>affy_1.6.7</b>	<a href="#">affy_1.6.7.zip</a>	<a href="#">affy_1.6.7.tgz</a>	<a href="#">affy_1.6.7.tar.gz</a>
<b>annotate_1.5.16</b>	<a href="#">annotate_1.5.16.zip</a>	<a href="#">annotate_1.5.16.tgz</a>	<a href="#">annotate_1.5.16.tar.gz</a>
<b>Biobase_1.5.12</b>	<a href="#">Biobase_1.5.12.zip</a>	<a href="#">Biobase_1.5.12.tgz</a>	<a href="#">Biobase_1.5.12.tar.gz</a>
<b>estrogen_1.5.0</b>	<a href="#">estrogen_1.5.0.zip</a>	<a href="#">estrogen_1.5.0.tar.gz</a>	<a href="#">estrogen_1.5.0.tar.gz</a>
<b>genefilter_1.6.3</b>	<a href="#">genefilter_1.6.3.zip</a>	<a href="#">genefilter_1.6.3.tgz</a>	<a href="#">genefilter_1.6.3.tar.gz</a>
<b>GO_1.8.2</b>	<a href="#">GO_1.8.2.zip</a>	<a href="#">GO_1.8.2.tgz</a>	<a href="#">GO_1.8.2.tar.gz</a>

<b>GOstats_1.1.3</b>	<a href="#">GOstats 1.1.3.zip</a>	<a href="#">GOstats 1.1.3.tgz</a>	<a href="#">GOstats 1.1.3.tar.gz</a>
<b>goTools_1.0.6</b>	<a href="#">goTools 1.0.6.zip</a>	<a href="#">goTools 1.0.6.tgz</a>	<a href="#">goTools 1.0.6.tar.gz</a>
<b>graph_1.5.9</b>	<a href="#">graph 1.5.9.zip</a>	<a href="#">graph 1.5.9.tgz</a>	<a href="#">graph 1.5.9.tar.gz</a>
<b>hgu95av2_1.8.4</b>	<a href="#">hgu95av2 1.8.4.zip</a>	<a href="#">hgu95av2 1.8.4.tgz</a>	<a href="#">hgu95av2 1.8.4.tar.gz</a>
<b>hgu95av2cdf_1.5.1</b>	<a href="#">hgu95av2cdf 1.5.1.zip</a>	<a href="#">hgu95av2cdf 1.5.1.tgz</a>	<a href="#">hgu95av2cdf 1.5.1.tar.gz</a>
<b>hgu133acdf_1.5.1</b>	<a href="#">hgu133acdf 1.5.1.zip</a>	<a href="#">hgu133acdf 1.5.1.tgz</a>	<a href="#">hgu133acdf 1.5.1.tar.gz</a>
<b>hgu133a_1.8.4</b>	<a href="#">hgu133a 1.8.4.zip</a>	<a href="#">hgu133a 1.8.4.tgz</a>	<a href="#">hgu133a 1.8.4.tar.gz</a>
<b>limma_2.0.2</b>	<a href="#">limma 2.0.2.zip</a>	<a href="#">limma 2.0.2.tgz</a>	<a href="#">limma 2.0.2.tar.gz</a>
<b>multtest_1.6.0</b>	<a href="#">multtest 1.6.0.zip</a>	<a href="#">multtest 1.6.0.tgz</a>	<a href="#">multtest 1.6.0.tar.gz</a>
<b>RBGL_1.3.13</b>	<a href="#">RBGL 1.3.13.zip</a>	<a href="#">RBGL 1.3.13.tgz</a>	<a href="#">RBGL 1.3.13.tar.gz</a>
<b>Ruuid_1.5.3</b>	<a href="#">Ruuid 1.5.3.zip</a>	<a href="#">Ruuid 1.5.3.tgz</a>	<a href="#">Ruuid 1.5.3.tar.gz</a>
reposTools_1.5.19	<a href="#">reposTools 1.5.19.zip</a>	<a href="#">reposTools 1.5.19.tgz</a>	<a href="#">reposTools 1.5.19.tar.gz</a>
<b>statmod_1.1.1</b>	<a href="#">statmod 1.1.1.zip</a>	<a href="#">statmod 1.1.1.tgz</a>	<a href="#">statmod 1.1.1.tar.gz</a>
<b>xtable_1.2-5</b>	<a href="#">xtable 1.2-5.zip</a>	<a href="#">xtable 1.2-5.tgz</a>	<a href="#">xtable 1.2-5.tar.gz</a>

The GOstats package used in this lab required two packages not listed above - `cluster` and `survival`. These packages are not listed above because they are included in the standard R distribution as part of the "Recommended Packages". If you do not have these packages installed, you can install them from CRAN, with the following R commands:

```
install.packages("cluster")
install.packages("survival").
```

## 1.2 Required data

The Estrogen dataset is required for this lab, and can be downloaded as an R package:

- [Windows](#)
- [Source \(Mac, Linux\)](#)

## 2. Introduction

In this lab, we move beyond the analysis of individual genes, and consider sets of genes in microarray experiments. We have already seen in lab 6 how sets of genes from a list of differentially expressed genes can be clustered together based on common behaviour in one or more microarray experiment. Another approach is to form gene sets based on a priori knowledge of common biological features shared by the genes (as described in the Gene Ontology (GO) database [2] for example).

The GO analysis can be formalized to include statistical tests to determine whether a particular Gene Ontology (GO) is over-represented (or under-represented) in the list of differentially expressed genes from the microarray experiment. For example if 10% of the most differentially expressed genes were associated with the GO term [apoptosis \(GO:0006915\)](#), this would seem to be an unusually large proportion of the gene list, given that apoptosis is such a specific type of biological process. To determine how much larger than "usual" this proportion is, it must be compared to the proportion of apoptosis-related genes in a reference gene list, which could be the entire set of genes on the microarray, or in fact the entire set of genes in the GO database. We will use the GOstats package [GOstats] to determine which gene ontologies (GOs) found in gene lists are statistically over-represented or under-represented. Another useful software tool for statistical analysis of GOs is <http://qostat.wehi.edu.au/> ([1]).

The second approach to gene set analysis we will examine in this list is quite different. We begin with a known set of genes and then test whether this set as a whole is differentially expressed in a microarray experiment. This type of test is useful when comparing one's microarray data with that of previous authors who have performed similar microarray experiments, because the lists of most differentially expressed genes reported by the previous authors can be regarded as

a "gene set" and tested to determine whether the genes are also differentially expressed in the current context.

Gene set testing was introduced by Mootha et al [5] and Lamb et al [6] in 2003. Mootha et al define the concept of a gene set enrichment test. For a given set of genes, one can test whether the set as a whole is up-regulated, down-regulated or differentially expressed with individual genes possibly going in either direction. Sometimes performing the traditional differential expression analysis of individual genes will yield no statistically significant results, but there may be stronger evidence for differential expression of gene sets.

## 3. Gene Ontologies

### 3.1 Introduction to Gene Ontologies

From [http://en.wikipedia.org/wiki/Gene\\_Ontology](http://en.wikipedia.org/wiki/Gene_Ontology) :

*"The **Gene Ontology**, or **GO**, is a trio of controlled vocabularies that are being developed to aid the description of the molecular functions of gene products, their placement in and as cellular components, and their participation in biological processes. Terms in each of the vocabularies are related to one another within a vocabulary in a polyhierarchical (or **Directed Acyclic Graph**) manner; terms are mutually exclusive across the three vocabularies. Each term consists of a definition, an alphanumeric identifier, and a phrase (the term itself)."*

In practice, one thinks of the **Directed Acyclic Graph (DAG)** as an (upside-down) tree, with the top three nodes being *molecular function*, *cellular component* and *biological process*. Then one can test whether a gene list contains a larger than usual proportion of genes found under a certain node of the GO tree. Technically it is a **DAG**, not a tree, because branches can merge together, but the concept of a tree may be helpful in understanding how GOs work. The reader is referred to the Gene Ontology consortium website for more information:

<http://www.geneontology.org/> [3].

### 3.2 GOstats example

We will use the Estrogen data set from lab 4. Whilst it would have been possible to save an .RData file at the end of lab 4 using the `save.image()` command and then load it here, we will start from scratch, so that this lab can be done without having done lab 4. We will not give any explanation for the R commands already discussed in lab 4.

As described in lab 4, we read in the Estrogen data, normalize it and fit a linear model in order to find differentially expressed genes.

```
library(limma)
library(affy)
library(hgu95av2cdf)
datadir <- system.file("extdata",package="estrogen")

# Please ensure that either "EstrogenTargets.txt" from
# http://bioinf.wehi.edu.au/marray/jsm2005/Data/EstrogenTargets.txt
# is in the current working directory as given by getwd(), or that
you
# provide the read.phenoData function with a full path to
# "EstrogenTargets.txt", e.g. "C:/JSM/EstrogenTargets.txt"
# or "C:\\JSM\\EstrogenTargets.txt".

pd <-
read.phenoData("EstrogenTargets.txt",header=TRUE,row.names=1,as.is=TRUE)
abatch <- ReadAffy(filenamees=pData(pd)$FileName,
celfile.path=datadir)
```

```

abatch@phenoData <- pd
eset <- rma(abatch)
targets <- pData(eset)
design <- model.matrix(~ -1 +
factor(targets$Target, levels=unique(targets$Target)))
colnames(design) <- unique(targets$Target)
numParameters <- ncol(design)
parameterNames <- colnames(design)
fit <- lmFit(eset, design=design)
contrastNames <- c(paste(parameterNames[2], parameterNames[1], sep="-"),
paste(parameterNames[4], parameterNames[3], sep="-"),
paste(parameterNames[3], parameterNames[1], sep="-"))
contrastsMatrix <- matrix(c(-1, 1, 0, 0, 0, 0, -1, 1, -1, 0, 1, 0), nrow=ncol(design))
rownames(contrastsMatrix) <- parameterNames
colnames(contrastsMatrix) <- contrastNames
fit2 <- contrasts.fit(fit, contrasts=contrastsMatrix)
fit2 <- eBayes(fit2)
numGenes <- nrow(eset@exprs)
completeTableEst10 <- topTable(fit2, coef="EstPresent10-EstAbsent10", number=numGenes)
save.image("EstrogenLinearModelFit.RData")

```

We will consider the comparison of "Estrogen Present" vs "Estrogen Absent" at time 10 hours, and take the top 50 genes with most evidence of differential expression according to the B statistic (also known as lod score or log odds of differential expression).

```

ord.Est10 <- order(fit2$lods[, "EstPresent10-EstAbsent10"], decreasing=TRUE)
top50.Est10 <- ord.Est10[1:50]

```

We now load the annotation package for the hgu95av2 chip used in the Estrogen experiment, and obtain Locus Link IDs for the top 50 most differentially expressed genes in the "Estrogen Present" vs "Estrogen Absent" (time 10hrs) comparison.

```

library(hgu95av2)
geneIDs <- ls(hgu95av2cdf)
LocusLinkIDs <-
as.character(unlist(lapply(mget(geneIDs, env=hgu95av2LOCUSID),
function(LL.ID) { return(paste(LL.ID, collapse="; ")) })))
topLocusLinkIDs <- LocusLinkIDs[top50.Est10]
topLocusLinkIDs <- topLocusLinkIDs[topLocusLinkIDs!="NA"]

```

We now load the GOstats package and use the GOHyperG function to search for statistically over-represented GOs within the top 50 most differentially expressed genes from the Estrogen set, compared with the GOs represented by all of the genes on the hgu95av2 chip. The test for statistical significance is based on a hypergeometric test. For more information, we refer to the help for the GOHyperG function which can be obtained by typing ?GOHyperG. The latter part of the help file, describing the statistical test is reproduced below:

*The test performed is a Hypergeometric test, using 'phyper', where at each GO node we determine how many LLIDs from the chip were annotated there, how many of the supplied LLIDs were annotated there and compute a p-value. This is the equivalent of using Fisher's exact test.*

```

library(GOstats)

```

```

goHyperG <- GOHyperG(unique(topLocusLinkIDs), lib="hgu95av2",
what="MF")
# MF is an abbreviation for "Molecular Function".
# See ?GOHyperG for more information.
names(goHyperG)
bestGOs <- goHyperG$pvalues[goHyperG$pvalues < 0.2]
bestGOs[1:5] # Inspect some of the p-values and corresponding GOs.
bestGOs <- sort(bestGOs) # order p-values
bestGOs.terms <- getGOTerm(names(bestGOs)) [["MF"]]
bestGOstring <- unlist(bestGOs.terms)
numCh <- nchar(bestGOstring)
bestGOstring2 <- substr(bestGOstring, 1, 25) # Limit names of GO
terms to 25 characters
bestGOstring3 <- paste(bestGOstring2, ifelse(numCh > 17, "...", ""),
sep="")
##get counts
bestGOs.counts <- goHyperG$goCounts[names(bestGOs)]
bestGOsTable <- data.frame("GO
Term"=I(bestGOstring3), "p.value"=round(bestGOs, 3),
"No. of Genes"=bestGOs.counts)
bestGOsTable

```

We now look at the GO annotations associated with the hgu95av2 chip (as used above), and consider how long the GOs are, i.e. how many nodes they are away from the basic Molecular Function (MF), Biological Process (BP) and Cellular Component (CC) nodes. We will then look at how many probes (genes) on the chip have multiple GOs associated with them. Finally we will look at information about where the GO information was obtained from for the genes on the hgu95av2 chip.

Firstly, we look at the lengths of the GOs associated with the hgu95av2 chip. A gene with a GO of length zero is classified as MF, BP or CC, but is not classified any more specifically than that. The longest GO length on this chip is 14. The majority of genes on the chip of GO annotations with lengths between 0 and 4, with only 1000 or so having more specific GO annotations (with longer lengths). Of course describing how specific a GO annotation is cannot really be done quantitatively, so one must not read too much into the meaning of the GO lengths.

```

affyGO <- eapply(hgu95av2GO, getOntology)
table(sapply(affyGO, length))

```

For each GO annotation on the hgu95av2chip, we can find out where this annotation information was obtained from, i.e. the evidence for this annotation.

```

affyEv <- eapply(hgu95av2GO, getEvidence)
table(unlist(affyEv, use.names = FALSE))

```

where:

Symbol	Meaning
IMP	inferred from mutant phenotype
IGI	inferred from genetic interaction
IPI	inferred from physical interaction
ISS	inferred from sequence similarity
IDA	inferred from direct assay
IEP	inferred from expression pattern
IEA	inferred from electronic annotation
TAS	traceable author statement
NAS	non-traceable author statement

ND	no biological data available
IC	inferred by curator

### 3.3 goTools

Another R package to consider when looking at GO annotations for genes on a microarray chip is the `goTools` package. This does not have some of the sophisticated statistical features of the `GOstats` package, but it is very easy to use when performing simple GO analysis. Rather than considering GOs of all possible lengths separately, it allows the user to ask what proportion of the genes in the list are under a certain node in the tree, or directed acyclic graph (DAG) to be precise. It is particularly useful for comparing gene lists and asking what proportion of the genes in each list fall within one branch of the GO tree (DAG).

The `goTools` algorithm is reproduced from the vignette of that package:

1. Read in a list of sets of probe id you want to compare.
2. Map each probe id to corresponding ontologies in the GO tree, if any.
3. Create the set of GO ids of interest used to compare your datasets (`endnode`). The function `EndNodeList` will create a set of nodes of the DAG located one level under MF, BP or CC, but you can use any sets of GO ids.
4. For each GO id, go up the GO tree until reaching the nodes in `endnode`. Search may be limited to MF, BP or CC if specified in `goType`.
5. Compute the percentage of direct children found under each node in `endnode`.
6. Return the results. Plot them if `plot=TRUE`.

The default end node list contains the direct children of MF (molecular function), BP (biological process), and CC (cellular component), giving 31 different ontologies (nodes) all together. The output obtained from `goTools` is the proportion of genes in the list found under each these ontologies in the DAG.

### 3.4 goTools Example

Firstly, let's load the `goTools` package, load the `probeID` data set included in the `data` subdirectory of the `goTools` package, which can be found using the R command, `system.file("data", package="goTools")`. The `ls()` command lists all of the R objects present in memory after loading the `probeID` data set. We see that there is a list object, `affylist` which contains three vectors of 100 Affy probe set IDs (i.e. gene IDs). The probe IDs in the `affylist` object are randomly chosen from the Affymetrix hgu133a chip.

```
library(GO)
library(annotate)
library(goTools)
data(probeID)
ls()
length(affylist)
length(affylist[[1]])
affylist[[1]][1:5]
```

Now we use the `ontoCompare` function from the `goTools` package to compare the gene ontologies represented by the genes in the three gene lists. This function returns the percentage of probe set IDs associated with the immediate children of the nodes contained in `endnode`. `endnode` will be discussed shortly. By default, it is the set of gene ontologies (GOs) which are immediate children of the nodes Biological Process (BP), Molecular Function (MF) and Cellular Component (CC).

```
res <- ontoCompare(affylist, probeType="hgu133a")
res[,1:5]
```

Now we plot a bar chart (with three bars for each ontology), comparing the number of genes in each gene list which represent that Gene Ontology (GO).

```
res <- ontoCompare(affylist, probeType="hgu133a", plot=TRUE)
```

The default end nodes list is defined by a call to the function `EndNodeList`. It contains all immediate children of Molecular Function(GO:0003674), Biological Process(GO:0008150) and Cellular Component (GO:0005575).

```
EndNodeList()
```

If you want to use more ontologies to describe your set of genes, you can use the function `CustomEndNodeList(id,rank)` to create a bigger set of end nodes. It returns all GO IDs children of `id` up to `rank` levels below `id`.

```
MFendnode <- CustomEndNodeList("GO:0003674", rank=2)
```

Finally, the code below shows you how to use a custom end node list, and also how to modify the `goType` argument to select only Molecular Function (MF) ontologies.

```
res <- ontoCompare(affylist, probeType="hgu133a",  
                  endnode=MFendnode, goType="MF")
```

You can also create a list of GO ids of nodes of interest and pass it directly to the `endnode` argument in the `ontoCompare` function. GO IDs must be in the following format: "GO:XXXXXXX"

## 4. Gene Set Analysis

### 4.1 Introduction to Gene Set Analysis

Now we turn our attention to tests for differential expression involving a set of genes. Mootha et al. [5] and Lamb et al. [6] made this methods popular in 2003. We will use a "gene set enrichment test", which is closely based on the one defined by Mootha et al. 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. Another possible application is to try to find differential expression in microarray experiments which show no strong differential expression when testing for individual differentially expressed genes, but they might show more evidence of differential expression when testing a predefined set of genes. Defining a useful gene set for this sort of analysis is not always trivial. One possibility is to use a set of genes which share common gene ontologies, i.e. choose a set of genes which are all associated with GOs below a certain node in the GO DAG (Directed Acyclic Graph). We will begin with some artificial examples to illustrate the concept of gene set tests with a small number of made-up t-statistics. Then we will use two sets of genes thought to be regulated by the Estrogen Receptor (ERalpha) to demonstrate testing for differential expression of gene sets in the Estrogen data set.

The `geneSetTest` function in the `limma` package [8] is described in its help file, reproduced below:

*This is essentially a stream-lined approach to Gene Set Enrichment Analysis introduced by Mootha et al (2003). Usually, 'statistics' is intended to hold t-like statistics, meaning that the genewise null hypotheses would be rejected for large positive or large negative values. Then 'alternative="greater"' can be used to test whether genes in the set tend to be up-regulated, 'alternative="less"' can be used to test whether the gene set is down-regulated, while 'alternative="two.sided"' tests whether the gene set holds highly ranked genes without regard to direction of change. Important note: if 'statistics' is an F-like statistic for which only large values are relevant for rejecting the null hypothesis, then you must use 'alternative="greater"' to get meaningful results.*

We now demonstrate the use of the `geneSetTest` function in `limma` using a miniscule set of artificial made-up t-statistics, where as usual, a positive t statistic means that a gene is up-regulated (i.e. expressed more highly in a condition of interest), whilst a negative value means that the gene is down-regulated. A t-statistic close to zero means that the gene is not differentially expressed.

In the first example we use, the artificial t-statistics will range from -9 to 9, and we will select a set of three genes for the test, those with t-statistics of -8, -6 and -5, i.e. we will use the 2nd, 4th and 5th t-statistics from our vector of artificial t-statistics. If these t-statistics represented real genes, all three genes would show strong evidence of differential expression (down-regulation) individually, so they should certainly show strong evidence of differential expression as a set as well. The value returned by `limma`'s `geneSetTest` function is a p-value.

```
library(limma)
sel <- c(2,4,5)
stat <- -9:9
stat[sel]
geneSetTest(sel, stat, nsim=100)
geneSetTest(sel, stat, ranks.only=TRUE)
```

If we did a test for up-regulation of the set, we would expect a large p-value (low evidence of up-regulation):

```
geneSetTest(sel, stat, alternative="greater", nsim=100)
```

On the otherhand, a test for down-regulation should give a small p-value:

```
geneSetTest(sel, stat, alternative="less", nsim=100)
```

## 4.2 Gene Set Analysis Example

We will again use the Estrogen data set, included in the `estrogen` R package. If you still have the R session open which you used to run through the `GOstats` exercises, you can use the `fit2` linear model fit object (and other objects previously computed). If you have closed your R session, but saved your data with the `save.image` command, then you can load it with:

```
load("EstrogenLinearModelFit.RData")
```

If you have trouble with this, you can just run the `estrogen` linear modelling commands again or ask someone else in the class for their copy of `EstrogenLinearModelFit.RData`.

We will use two sets of genes which are thought to be ER-regulated, i.e. regulated by the Estrogen Receptor alpha. The first set (Jin et al [4]) contains genes which have been experimentally verified to be ER-regulated and the second set (O'lone et al [7]) contains a large list of genes which are predicted to be ER-regulated by a model (so they may or may not be ER-regulated).

These gene sets (particularly the first one) 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.

The data required for this exercise is available at <http://bioinf.wehi.edu.au/marray/jsm2005/Data/knownERgenes.txt> and <http://bioinf.wehi.edu.au/marray/jsm2005/Data/predictedERgenes.txt>.

Now, having loaded the Estrogen linear modelling data or rerun the analysis (above), ensure that the two gene list files are in your current working directory, as given by the R command `getwd()`. Then read the gene lists into R:

```
known <-  
read.table("knownERgenes.txt", header=TRUE, as.is=TRUE, sep="\t")  
knownERgenes <- known$UGCluster  
predicted <-  
read.table("predictedERgenes.txt", header=TRUE, as.is=TRUE, sep="\t")  
predictedERgenes <- predicted$UGCluster  
library(hgu95av2)  
unigene <- unlist(as.list(hgu95av2UNIGENE))  
knownERgenesOnChip <- match(knownERgenes, unigene)  
knownERgenesOnChip <- knownERgenesOnChip[!is.na(knownERgenesOnChip)]  
predictedERgenesOnChip <- match(predictedERgenes, unigene)  
predictedERgenesOnChip <-  
predictedERgenesOnChip[!is.na(predictedERgenesOnChip)]
```

Now we will use the moderated t-statistics calculated previously for the comparison of estrogen present and estrogen absent 10 hours after the estrogen was reintroduced into the cells. We try all three types of tests - "two-sided", "greater" (for genes up-regulated in the sample with estrogen present), and "less" (for genes down-regulated in the sample with estrogen present). We expect to get better (smaller) p-values for the known ER-regulated genes than for the predicted ER-regulated genes. The recent review of Estrogen-repressed genes in breast cancer cells by Zubairy and Oesterreich [9] suggests that the majority of genes regulated by estrogen receptors are actually repressed (down-regulated), so we should expect a lower p-value for the "less" test (at least for the known ER-regulated genes).

```
geneSetTest(knownERgenesOnChip, completeTableEst10$t, "two.sided")  
geneSetTest(knownERgenesOnChip, completeTableEst10$t, "greater")  
geneSetTest(knownERgenesOnChip, completeTableEst10$t, "less")  
  
set.seed(0)  
geneSet <- sample(predictedERgenesOnChip, length(knownERgenesOnChip))  
geneSetTest(geneSet, completeTableEst10$t, "two.sided")  
geneSetTest(geneSet, completeTableEst10$t, "greater")  
geneSetTest(geneSet, completeTableEst10$t, "less")  
  
geneSet <- sample(predictedERgenesOnChip, length(knownERgenesOnChip))  
geneSetTest(geneSet, completeTableEst10$t, "two.sided")  
geneSetTest(geneSet, completeTableEst10$t, "greater")  
geneSetTest(geneSet, completeTableEst10$t, "less")  
  
geneSet <- sample(predictedERgenesOnChip, length(knownERgenesOnChip))  
geneSetTest(geneSet, completeTableEst10$t, "two.sided")  
geneSetTest(geneSet, completeTableEst10$t, "greater")  
geneSetTest(geneSet, completeTableEst10$t, "less")
```

## 5. Acknowledgements

Some material from the GOstats section was based on material from the GOstats vignette by Robert Gentleman. The goTools example was taken from the goTools vignette by Jean Yee Hwa Yang. The first gene set enrichment test example was taken from the limma package documentation by Gordon Smyth. Thanks to Hui Tang and Terry Speed for finding the known and predicted ER-regulated gene sets (Jin et al. [4] and O'loné et al. [7]).

## 6. References

1. Beissbarth T, Speed TP (2004). Gostat: find statistically overrepresented Gene Ontologies within a group of genes. *BIOINFORMATICS* 20 (9): 1464-1465.
2. Camon E, Magrane M, Barrell D, Lee V, Dimmer E, Binns D, Maslen J, Harte N, Lopez R, and Apweiler R (2004). The Gene Ontology annotation (GOA) database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Research*, 32:D262-D266.
3. The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25-29, 2000.
4. Jin VX, Leu YW, Liyanarachchi S, Sun H, Fan1 M, Nephew1 KP, Huang T.H. and Davuluri DV (2004). Identifying estrogen receptor {alpha} target genes using integrated computational genomics and chromatin immunoprecipitation microarray. *Nucleic Acids Research* 32(22): 6627-6635
5. Lamb J, Ramaswamy S, Ford HL, Contreras B, Martinez RV, Kittrell FS, Zahnow CA, Patterson N, Golub TR, Ewen ME (2003). A mechanism of cyclin D1 action encoded in the patterns of gene expression in human cancer. *Cell*; 114(3):323-34.
6. Mootha VK, Lindgren CM, Eriksson KF, Subramanian A, Sihag S, Lehar J, Puigserver P, Carlsson E, Ridderstråle, M, Laurila E, Houstis N, Daly MJ, Patterson N, Mesirov JP, Golub TR, Tamayo P, Spiegelman BM, Lander ES, Hirschhorn JN, Altshuler D, and Groop LC (2003). PGC-1alpha Responsive Genes Involved in Oxidative Phosphorylation are Coordinately Downregulated in Human Diabetes, *Nature Genetics* 34(3):267-73.
7. O'lonc R, Frith MC, Karlsson EK, Hansen U (2004). Genomic Targets of Nuclear Estrogen Receptors. *Molecular Endocrinology* 18(8): 1859-1875
8. Smyth GK (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology* 3, No. 1, Article 3.
9. Zubairy S, Oesterreich S (2005). Estrogen-repressed genes -- key mediators of estrogen action? *Breast Cancer Res.* 2005;7(4):163-4.