Seminar III: R/Bioconductor

Microarray Data Analysis, Multitesting and SpeCond.

José Víctor Moreno Mayar
jmoreno@lcg.unam.mx

LCG - UNAM

August - December, 2009

Microarray Data Analysis, Multitesting and SpeCond.

1. Packages for this class.
2. From .CEL files to ExpressionSet objects.
3. Comparing Chips.
4. Initial Exploration.
5. t tests.
6. Annotated Lists of Interesting Genes.

Microarray Data Analysis, Multitesting and SpeCond.

7. The Multitesting Problem.
8. multtest.
9. SpeCond.

Packages for this class.

These are the packages that we will use this class.

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("affy")
> biocLite("limma")
> biocLite("genefilter")
> biocLite("annaffy")
> biocLite("KEGG.db")
> biocLite("GO.db")
> biocLite("hgu133a2.db")
> biocLite("SpeCond")
> install.packages("multtest")
```

Preprocessing Microarrays.

As we discussed last class, there is a whole ritual while working with microarrays. It goes like this:

1. Reading in probe level data.
2. Background correction.
3. Normalization.
4. Probe specific background correction, e.g. subtracting MM.
5. Summarizing the probe set values into one expression measure and, in some cases, a standard error for this summary.

rma.

- rma() does the work for us.
- It converts an AffyBatch object into expression measures.
- The data from the AffyBatch objet is passed directly to the ExpressionSet object.
- There are some other methods for doing this, such as expresso, threestep or mas5, explore them.

1. Probe specific correction through a signal+noise model.
2. Quantile normalization.
3. Calculation of expression measure.

Practice.

- Let us use the experiment that we used last class.
- Then, assign the phenoData as we learned last class.
- The file for the phenoData is called pdata1422rep.txt, download it.
- Use rma to extract the expression data from the AffyBatch object.

Practice.


```
> library(affy)
> library(ArrayExpress)
> Data = ArrayExpress("E-MEXP-1422",
+     save = T)
> pd <- read.AnnotatedDataFrame(filename = "pdata1422
+     header = T)
> pData(pd)
```

	Source.Name	Replicate
AF16.CEL	PROX1_siRNA-2	2
AF7.CEL	PROX1_siRNA-1	1
AF14.CEL	GFP_siRNA	2
AF8.CEL	PROX1_siRNA-2	1
AF15.CEL	PROX1_siRNA-1	2
AF6.CEL	GFP_siRNA	1

Practice.


```
> slot(Data, "phenoData") <- pd
> pData(slot(Data, "phenoData"))
```

	Source.Name	Replicate
AF16.CEL	PROX1_siRNA-2	2
AF7.CEL	PROX1_siRNA-1	1
AF14.CEL	GFP_siRNA	2
AF8.CEL	PROX1_siRNA-2	1
AF15.CEL	PROX1_siRNA-1	2
AF6.CEL	GFP_siRNA	1

```
> eset <- rma(Data)

Background correcting
Normalizing
Calculating Expression
```

Practice.

You can access the expression measure with the function exprs().

```
> e <- exprs(eset)
> head(e)
```

	AF16.CEL	AF7.CEL	AF14.CEL
1007_s_at	9.185240	9.279876	9.092066
1053_at	9.502950	9.273640	9.693810
117_at	4.806154	4.884860	4.882013
121_at	8.069389	8.299280	8.206973
1255_g_at	3.201186	3.074842	3.179423
1294_at	5.090224	5.082702	5.123301
	AF8.CEL	AF15.CEL	AF6.CEL
1007_s_at	9.157650	9.324122	9.070019
1053_at	9.441064	9.256205	9.664184

Practice.


```
117_at    4.700391 4.955092 4.724530
121_at    8.121853 8.316409 8.031186
1255_g_at 3.071355 3.089710 3.084504
1294_at   5.208284 5.058098 5.069656
```

Subsetting Experiments.

- As you know, microarray experiments are useful when comparing between two different conditions.
- Our dataset contains 3 conditions (two experimental siRNAs and a control siRNA.)
- We have to generate an index to know which samples come from which condition.
- This is where the phenoData finally becomes useful.

Subsetting Experiments.


```
> Index1 <- which(eset$Source.Name ==
+     "PROX1_siRNA-1")
> Index2 <- which(eset$Source.Name ==
+     "PROX1_siRNA-2")
> Index3 <- which(eset$Source.Name ==
+     "GFP_siRNA")
```

Now, you can select a specific condition, just by specifying an index.

```
> e[, Index1]
```

MvA Plot.

- As you learned last class, MvA plots are very useful when comparing between replicate sets of arrays.
- Let us make an MvA plot for the arrays corresponding to Index1 (siRNA-1) and Index3 (control siRNA).
- What is plotted in an MvA plot?
- How do you get M?
- How do you get A?
- Plotting time. =)

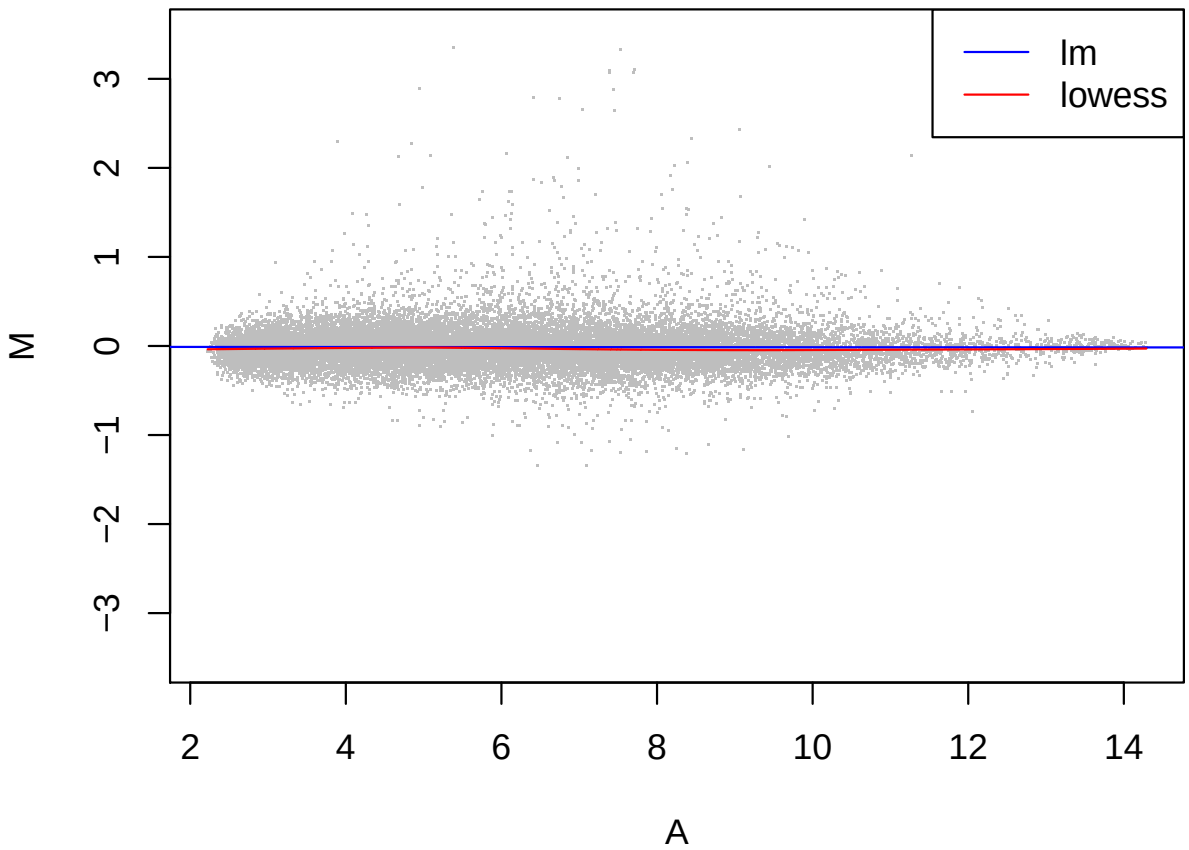
MvA Plot.


```
> M <- rowMeans(e[, Index1]) - rowMeans(e[,
+     Index3])
> A <- rowMeans(e)

> plot(A, M, pch = ".", ylim = c(-3.5,
+     3.5))
```

Fit an lm() and a lowess() curve, which one fits the best?

MvA plot.



Student's t tests.

- As you learned in the limma class, a way to get DEGs is making *t* tests.
- What does a *t* test tests?
- The genefilter package gives the possibility of making multivariate tests.
- We will perform a two-sample *t* test between arrays from Index1 and Index3(control).
- First of all, let us subset our experiment, and then apply the test.

Student's *t* tests.


```
> weset <- eset[, c(Index1, Index3)]
> weset$Source.Name <- factor(weset$Source.Name)
> library(genefilter)
> tt <- rowttests(weset, "Source.Name")
> head(tt)
```

```
           statistic          dm
1007_s_at -8.9394256 -0.22095639
1053_at   24.0911153  0.41407447
117_at    -1.3536214 -0.11670495
121_at    -2.1375252 -0.18876475
1255_g_at  1.0343322  0.04968753
1294_at    0.8837524  0.02607872
               p.value
1007_s_at 0.012283467
```

Student's t tests.


```
1053_at    0.001718563
117_at     0.308538555
121_at     0.166009410
1255_g_at 0.409660335
1294_at    0.470057568
```

Student's t tests.

- Why do we made Source.Name a factor?[1]
- Which other tests can be performed by the genefilter package?
- What does the p-value represents?[2]
- Now, let us make a volcano plot.

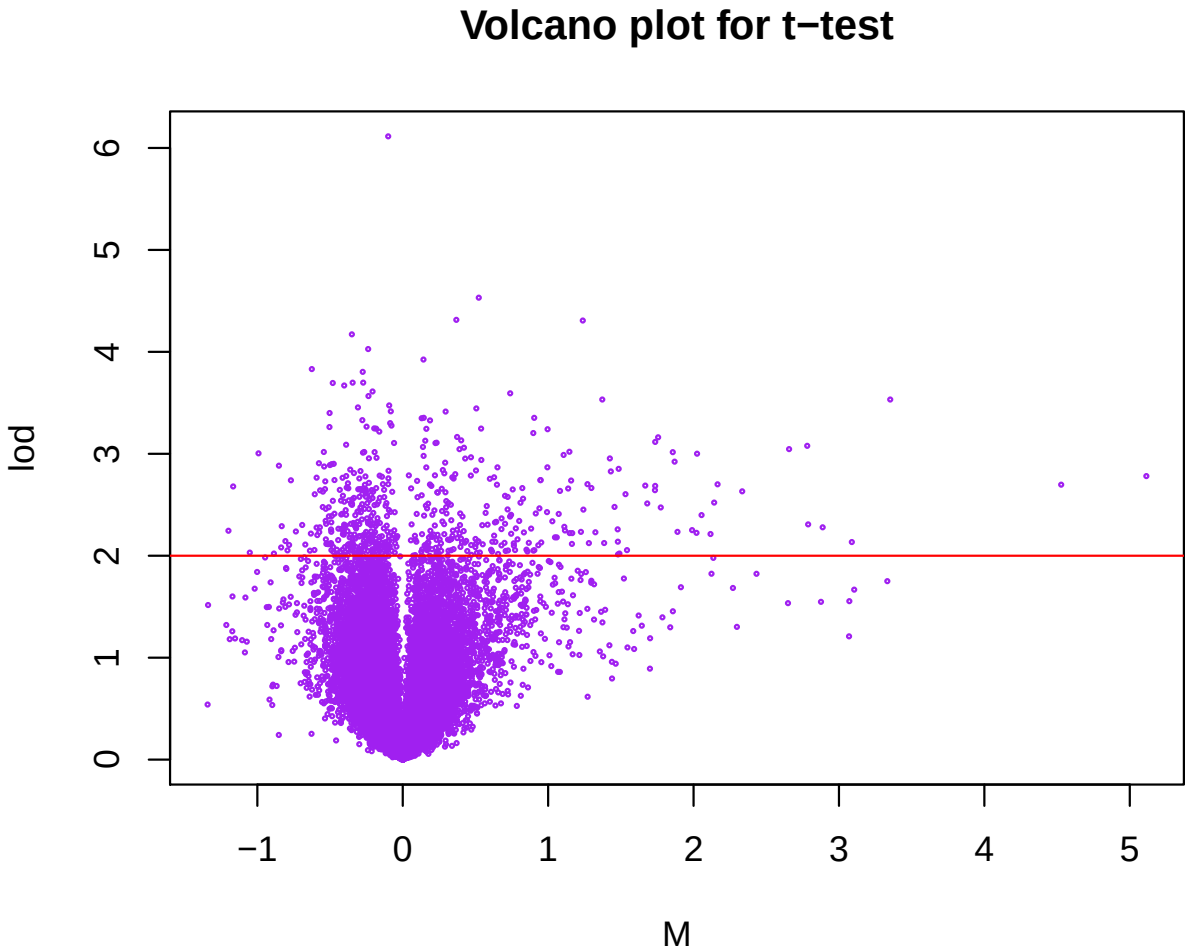
[1] See the rowttests help.

[2] Think of false positives.

Volcano Plot.


```
> lod <- -log10(tt$p.value)
> plot(M, lod, cex = 0.25, main = "Volcano plot for t
+      col = "purple")
> abline(h = 2, col = "red")
```

Volcano Plot.



Volcano Plot.

- Why is the cutoff set at 2?
- Which points would you believe to be worth validating experimentally.

eBayes.

- When there is a small number of replicates, t tests are not so appropriate.
- As you learned in the limma class, there is an alternative to this.
- You can use a moderated $t-$statistic to solve this problem.[3]
- Compare the volcano plots.

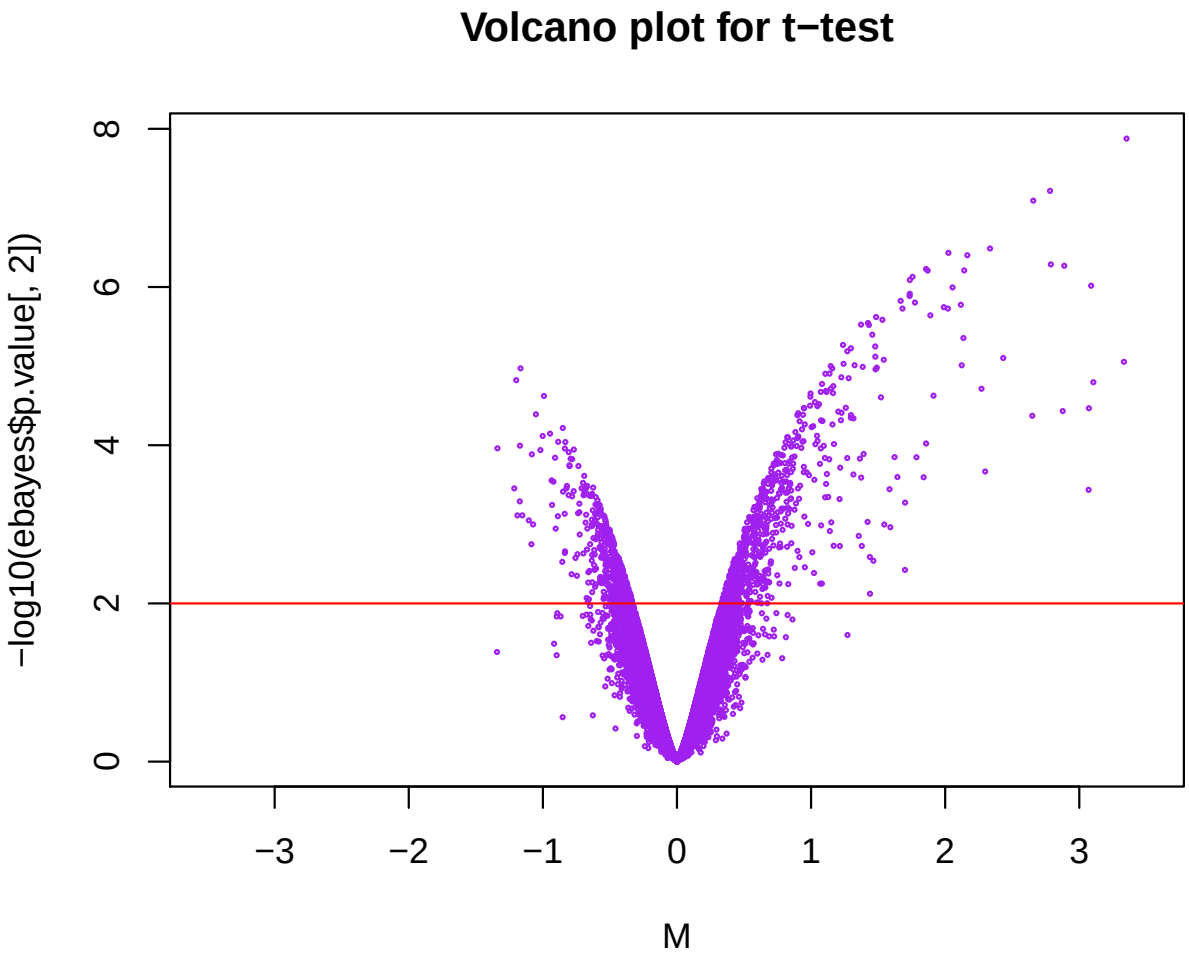
[3] eBayes.

eBayes.


```
> library("limma")
> design <- model.matrix(~weset$Source.Name)
> fit <- lmFit(weset, design)
> ebayes <- eBayes(fit)

> plot(M, -log10(ebayes$p.value[,
+     2]), xlim = c(-1, 1), cex = 0.25,
+     main = "Volcano plot for t-test",
+     col = "purple")
> abline(h = 2, col = "red")
```

Volcano Plot for eBayes.



Annotation.

- Another important aspect of a microarray experiment, is the annotation of the chip.
- Actually, there are custom chips for different organisms and given experiments.
- So, the probe annotation is given by the environment you are using.
- For each AffyBatch object, you can know which annotation is used by entering to the Annotation slot.
- You can also know the annotation used for an ExpressionSet by using the annotation() function.
- Which is the environment for our Data object?[4]

Annotation.


```
> library(affy)
> slot(Data, "annotation")
> library("annotate")
> annotation(weset)
```

[4] You can download different environments.

HTML Reports.

- Now that we have a set of p-values, we can get a table of interesting genes with some functions you already know.
- We will use our ebayes object and the function topTable.
- We will also obtain the gene names as they will be useful later.

```
> tab <- topTable(ebayes, coef = 2,
+     adjust.method = "BH", n = 15)
> genenames <- as.character(tab$ID)
```

HTML Reports.

- Now that we have a list of 15 interesting genes, it would be interesting to know which ones are them.
- We will generate two reports, one containing the stats of our interesting genes and one containing some other interesting facts about them, such as GOs, chromosome location, and pathways in which they participate.
- There are many functions which will helps you to get information on the probes for a determined chip.
- Some of them are getLL() and getSYMBOL()

HTML reports.


```
> library("hgu133a2.db")
> library("annotate")
> ll <- getLL(genenames, "hgu133a2")
> sym <- getSYMBOL(genenames, "hgu133a2")
> tab <- data.frame(sym, tab[, -1])
> htmlpage(as.data.frame(ll), filename = "GeneList1.h
+     title = "HTML report", othernames = tab,
+     table.head = c("Locus ID",
+         colnames(tab)), table.center = TRUE)
```

annaffy.

A great way to get a more detailed report is by using the annaffy package.

```
> library("annaffy")
> library("KEGG.db")
> library("GO.db")
> anntab <- aafTableAnn(genenames,
+     "hgu133a2.db", aaf.handler())
> saveHTML(anntab, file = "GeneList2.html")
```

- What do you think aaf.handler() is for?

Multitesting.

- We are done with $t-$tests, but there must be a criterion to measure the reliability of the obtained $p-$values.
- This is because we are analyzing huge sets of observations.
- How many genes are we testing in our experiment?[5]
- If we set a cutoff at a $p-$value of .01, how many false positives will we get?
- =S

[5] Use dim().

Multitesting Corrections.

- Do not worry, there are several ways to adjust the $p-$values.
- Actually, we have already tried one, which is the default parameter adjust.method="BH" of the topTable() function of the limma package.
- Let us see some other ones.

Important Concepts.

- Family-Wise Error Rate (FWER).
  - A family of hypotheses is defined.
  - So the FWER is defined as the probability of finding at least one false positive in the family of tests

$$FWER = 1 - (1 - pval)^m \tag{1}$$

- False Discovery Rate
  - It is defined as how many false positives will be discovered in the whole set of comparisons.
  - The same as we did before.

Multitesting Corrections.

- The Bonferroni correction is achieved by setting a new threshold by muliplying the $p-$values by the number of observations.
- Other methods that control the FWER are the Holm, Hochberg, and Hommel corrections.
- More powerful corrections are the Benjamini and Hochberg and the Benjamini and Yekutieli corrections which take control over the FDR, being less conservative.

multtest.

- multtest is a package which performs Multiple Testing Procedures.
- The main function is called MTP().
- This function gives you the oportunity to decide which test is to be performed, which type I error rate is to be controlled, how should the null distribution will be built, and which one is the rejection threshold to be used.
- The arguments that control these parameters are test, typeone, nulldist and B, and alpha.
- Do not forget that a factor indicating how are the two samples composed has to be passed in Y.

Some MTP practice.

- For this practice, we will use a bigger dataset called golub.
- Apply MTP to golub with the default $t-$test, an α of .01, an fdr type I error rate, and 50 bootstrap distributions.

Some MTP practice.


```
> library(multtest)
> data(golub)
> t.multi.test.golub <- MTP(golub,
+     Y = golub.cl, typeone = "fdr",
+     B = 50, alpha = 0.01)

running bootstrap...
iteration = 50

> slotNames(t.multi.test.golub)
```

Some MTP practice.


```
 [1] "statistic"     "estimate"
 [3] "sampsize"      "rawp"
 [5] "adjp"          "conf.reg"
 [7] "cutoff"        "reject"
 [9] "rawdist"       "nulldist"
[11] "nulldist.type" "marg.null"
[13] "marg.par"      "label"
[15] "index"         "call"
[17] "seed"
```

```
> summary(t.multi.test.golub)
```

Some MTP practice.


```
MTP:  ss.maxT
Type I error rate:  fdr (conservative)

  Level Rejections
1  0.01         33

             Min. 1st Qu.   Median
adjp        0.000  1.0000  1.00000
rawp        0.000  0.0000  0.10000
statistic -7.548 -1.6740 -0.06980
estimate  -2.160 -0.2559 -0.01275
                 Mean 3rd Qu.   Max.
adjp        8.866e-01  1.0000  1.000
rawp        2.583e-01  0.4600  1.000
statistic -1.929e-01  1.3520 10.580
```

Some MTP practice.


```
estimate   1.928e-07  0.2187  2.892
          NA's
adjp             0
rawp             0
statistic        0
estimate         0
```

Some MTP practice.

- What is a rejection in this case?
- Where can you find the new $p-$values?
- The gene names for this experiment are contained in golub.gnames, how would you know the names of the DEGs in this test?

Some MTP practice.


```
> head(golub.gnames[, 2][slot(t.multi.test.golub,
+     "reject")])

[1] "CYSTATIN A"
[2] "Macmarcks"
[3] "SPTAN1 Spectrin, alpha, non-erythrocytic 1 (alph
[4] "IEF SSP 9502 mRNA"
[5] "RB1 Retinoblastoma 1 (including osteosarcoma)"
[6] "Inducible protein mRNA"
```

This is it for affy and mtp, now, let us move on to SpeCond.

SpeCond Overwiew.

- The SpeCond package finds specific DEGs for different conditions.
- The process is made by fitting a null distribution to the gene expression measures.
- The model is made by a mixture of normal distributions.
- As soon as there is a null distribution, significantly DEGs are identified.
- Adjusted $p-$values are used, as you may imagine

Important Parameters.

There are some important parameters used for the definition of condition specific DEGs.

- lambda is involved in the model choosing process by choosing 1, 2 or 3 normal distributions.
- beta is involved in the distributions' variance.
- per is how many specific conditions may be found per gene.
- md is the median distance between two normal components.[6]
- mlk (minimum log-likelihood)is used to cluster separate different conditions, so a gene may be defined as an outlier.

Important Parameters.

- rsd (standard deviation ratio) is used to compare against the standard deviation of the null distribution, so outliers can be found.

[6] To find outliers.

SpeCond Important Functions.

- There are 3 imporant functions which are considered to be the most important.
- SpeCond is the most important function, it does the fitting and the detection.
- getFullHtmlSpeCondResult saves the results for all genes in an HTML report.
- getGeneHtmlPage makes an HTML report for each gene.
- Time for some practice.

SpeCond Demo.

- The package already includes a dataset
- expSetSpeCondExample contains 64 chips, 32 duplicated conditions, with 220 features each.
- How would you check this if expSetSpeCondExample is an ExpressionSet object?[7]

[7] You should know by now.

SpeCond Demo.


```
> library(SpeCond)

by using mclust, you accept the license agreement in t
and at http://www.stat.washington.edu/mclust/license.t

> data(expressionSpeCondExample)
> data(expSetSpeCondExample)
> expSetSpeCondExample

ExpressionSet (storageMode: lockedEnvironment)
assayData: 220 features, 64 samples
  element names: exprs
phenoData
  sampleNames: S_1, S_2, ..., S_64  (64 total)
  varLabels and varMetadata description:
    Tissue: Tissue names
```

SpeCond Demo.


```
    Exp: Experience number
featureData
  featureNames: 200606_at, 200607_s_at,
  ..., 217856_at  (220 total)
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
Annotation:
```

```
> Mexp = expressionSpeCondExample
> MexpS = getMatrixFromExpressionSet(expSetSpeCondExa
+     condition.factor = expSetSpeCondExample$Tissue,
+     condition.method = "mean")
> generalResult = SpeCond(expSetSpeCondExample,
+     param.detection = NULL, multitest.correction.me
+     prefix.file = "E", print.hist.pv = TRUE,
```

SpeCond Demo.


```
+      fit1 = NULL, fit2 = NULL, specificOutlierStep1
+      condition.factor = expSetSpeCondExample$Tissue,
+      condition.method = "mean")
[1] "The expressionMatrix argument that you entered h
[1] "Step1"
[1] "Step1, fitting"
[1] "start: get null distributions"
[1] "end: get null distributions"
[1] "start: specific detection from p-values"
[1] "end: specific detection from p-values"
[1] "Step2"
[1] "Step2, fitting"
[1] "start: get null distributions"
[1] "end: get null distributions"
```

SpeCond Demo.


```
[1] "start: specific detection from p-values"
[1] "end: specific detection from p-values"
```

```
> specificResult = generalResult$specificResult
> getFullHtmlSpeCondResult(SpeCondResult = generalRes
+     param.detection = specificResult$param.detectio
+     page.name = "Example_SpeCond_results",
+     page.title = "Tissue specific results",
+     sort.condition = "all", heatmap.profile = TRUE,
+     heatmap.expression = FALSE,
+     heatmap.unique.profile = FALSE,
+     expressionMatrix = Mexp)
```

SpeCond Demo.


```
[1] "force=TRUE, Delete all files in E_General_Result
[1] "The following files are created in the directory
[1] "/Users/Mayar/microarrayslecture/E_General_Result
[1] "E_barplot_nb_tissue_nb_genes.png"
[1] "E_nb_specific_gene_in_condition.png"
[1] "E_profile_heatmap.png"
[2] "E_profile_heatmap.pdf"
[1] "E_result_specific_probeset.txt"

> genePageInfo = getGeneHtmlPage(Mexp,
+     specificResult, name.index.html = "index_exampl
+     gene.html.ids = c(1:20))

[1] "force=TRUE, Delete all files in E_Single_result_
[1] "The gene html page(s) will be created in the E_S
```

SpeCond Pros.

- What step that we did in the weset example does getMatrixFromExpressionSet summarize?[8]
- What is the difference between this approach and the $t-$test approach that we followed earlier?[9]
- How would you solve this?[10]
- The reports are already in HTML.

[8] Remember the subsetting step?

[9] How many tissues (conditions) are tested?

[10] Tell me a statistical test for several data groups.