

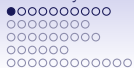


# Differential Expression Analysis of Microarray Data

Steven Buechler

Department of Mathematics  
276B Hurley Hall; 1-6233

Fall, 2007



# Outline

## Microarrays

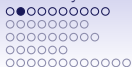
Introduction to Differential Expression Analysis

Multiple Testing Problem

Prefiltering the Gene List

Multtest Package

Differential Expression Example



## Most Expression Analysis is Comparative

Frequently the important investigations with microarrays are to identify the genes whose expression levels change between two sample groups. To understand the effect of a drug we may ask which genes are **up-regulated** (increased in expression) or **down-regulated** (decreased in expression) between treatment and control groups.

Other paradigms involve clustering genes that follow the same expression pattern across a set of samples, or clustering samples with similar expression patterns across genes. This is beyond the scope of the course (timewise).



## Most Expression Analysis is Comparative

Frequently the important investigations with microarrays are to identify the genes whose expression levels change between two sample groups. To understand the effect of a drug we may ask which genes are **up-regulated** (increased in expression) or **down-regulated** (decreased in expression) between treatment and control groups.

Other paradigms involve clustering genes that follow the same expression pattern across a set of samples, or clustering samples with similar expression patterns across genes. This is beyond the scope of the course (timewise).



## Differential Expression Compares Means

Each sample group will contain numerous replicates. The group expression level for a probe will be summarized as the mean of the expression levels in the group replicates. Thus, differential expression problems are a comparison of means. When there are two sample groups this is a t test of some kind.



## One Gene Example

As a very simple example we ask if *Mmp7* is differentially expressed between the vehicle and sulindac samples in the mouse expression data created previously.

```
> load("/Users/steve/Documents/Bio/Rcourse/Lect10/sulindacO
> library(affy)
> library(moe430a)
```



## Steps in Testing One Gene

- Find the probe(s) associated with the gene.
- For each probe create the vector of expression values for the two sample groups.
- Execute a t test for each probe.
- To be conservative we use the Mann-Whitney non-parametric t test.



## Steps in Testing One Gene

- Find the probe(s) associated with the gene.
- For each probe create the vector of expression values for the two sample groups.
- Execute a t test for each probe.
- To be conservative we use the Mann-Whitney non-parametric t test.





## Steps in Testing One Gene

- Find the probe(s) associated with the gene.
- For each probe create the vector of expression values for the two sample groups.
- Execute a t test for each probe.
- To be conservative we use the Mann-Whitney non-parametric t test.



## Steps in Testing One Gene

- Find the probe(s) associated with the gene.
- For each probe create the vector of expression values for the two sample groups.
- Execute a t test for each probe.
- To be conservative we use the Mann-Whitney non-parametric t test.



## Find the Probe(s) for Mmp7

The environment we need to query is `moe430aSYMBOL`, but we need to work backwards from the value to the key. A test like `moe430aSYMBOL == "Mmp7"` doesn't work with an environment. We extract the symbols into a long character vector with names the probes.

```
> prbs <- featureNames(sulEset1)
> moeSyms <- unlist(mget(prbs, envir = moe430aSYMBOL))
> moeSyms[1:3]
```

```
1415670_at 1415671_at 1415672_at
      "Copg" "Atp6v0d1"  "Golga7"
```



## Find the Probe(s) for Mmp7

moeSyms has many NA values, which complicates the matching process. (NA matches anything.)

```
> mmp7S1 <- moeSyms[moeSyms == "Mmp7"]  
> mmP7S <- mmp7S1[!is.na(mmp7S1)]  
> mmP7S
```

```
1449478_at  
  "Mmp7"
```



## Extract the Expression Vectors

for the Mmp7 probe

```
> sulExp <- exprs(sulEset1)
> pd <- phenoData(sulEset1)
> sMmp7 <- sulExp["1449478_at", pd$treatment ==
+   "S"]
> vMmp7 <- sulExp["1449478_at", pd$treatment ==
+   "V"]
> sMmp7
```

```
S1.cel S3.cel S4.cel S6.cel S8.cel S9.cel
11.138  9.711  9.648 11.212 10.715 10.090
```

```
> vMmp7
```

```
V2.cel V3.cel V4.cel V5.cel V6.cel V7.cel V8.cel
11.195  9.408 12.317 12.262 11.871 11.753 11.549
V9.cel
11.416
```



## Execute the t Test

Use the Mann-Whitney, also called the Wilcoxon Rank Sum test.

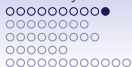
```
> wilcox.test(sMmp7, vMmp7)
```

```
Wilcoxon rank sum test
```

```
data: sMmp7 and vMmp7
```

```
W = 7, p-value = 0.02930
```

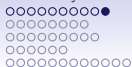
```
alternative hypothesis: true location shift is not equal to
```



## Conclusion About Mmp7

Conclude that Mmp7 expression is decreased in the sulindac treated samples.

However, if all we wanted to know about is one gene we'd just use RT-PCR, not an array with 22,000 features. The purpose of an array is to get a list of genes that are differentially expressed, after testing each one. This brings into play the multiple hypothesis testing problem.



## Conclusion About Mmp7

Conclude that Mmp7 expression is decreased in the sulindac treated samples.

However, if all we wanted to know about is one gene we'd just use RT-PCR, not an array with 22,000 features. The purpose of an array is to get a list of genes that are differentially expressed, after testing each one. This brings into play the multiple hypothesis testing problem.





# Outline

## Microarrays

Introduction to Differential Expression Analysis

**Multiple Testing Problem**

Prefiltering the Gene List

Multtest Package

Differential Expression Example



## Many Genes Tested at Once

In array-based differential expression analysis the problem is to generate a list of genes that are differentially expressed, being as complete as possible.

From a statistical point of view, for each gene we are testing the null hypothesis that there is no differential expression across the sample groups. This may represent thousands of tests.



## Many Genes Tested at Once

In array-based differential expression analysis the problem is to generate a list of genes that are differentially expressed, being as complete as possible.

From a statistical point of view, for each gene we are testing the null hypothesis that there is no differential expression across the sample groups. This may represent thousands of tests.



## Example of the Problem

Suppose that 1000 genes are represented on an array and we test each with a t test with a Type I error threshold of 0.05. We might expect 40 genes to be differentially expressed. Of the 960 non-differentially expressed genes we can expect 5% errors, or  $.05 \times 960 = 48$  false positives. In other words, there are more false positives than truly differentially expressed genes.

This illustrates the **multiple hypothesis testing problem**. With many independent tests the percentage of false positives may overwhelm the true positives and make the analysis effectively useless.

Most experimenters would state as the goal in such a multiple gene differential analysis is that the probability of *one* false positive is 0.05. That is, that the *experiment-wise* Type I error is 0.05.



## Example of the Problem

Suppose that 1000 genes are represented on an array and we test each with a t test with a Type I error threshold of 0.05. We might expect 40 genes to be differentially expressed. Of the 960 non-differentially expressed genes we can expect 5% errors, or  $.05 \times 960 = 48$  false positives. In other words, there are more false positives than truly differentially expressed genes.

This illustrates the **multiple hypothesis testing problem**. With many independent tests the percentage of false positives may overwhelm the true positives and make the analysis effectively useless.

Most experimenters would state as the goal in such a multiple gene differential analysis is that the probability of *one* false positive is 0.05. That is, that the *experiment-wise* Type I error is 0.05.



## Example of the Problem

Suppose that 1000 genes are represented on an array and we test each with a t test with a Type I error threshold of 0.05. We might expect 40 genes to be differentially expressed. Of the 960 non-differentially expressed genes we can expect 5% errors, or  $.05 \times 960 = 48$  false positives. In other words, there are more false positives than truly differentially expressed genes.

This illustrates the **multiple hypothesis testing problem**. With many independent tests the percentage of false positives may overwhelm the true positives and make the analysis effectively useless.

Most experimenters would state as the goal in such a multiple gene differential analysis is that the probability of *one* false positive is 0.05. That is, that the *experiment-wise* Type I error is 0.05.



## Bonferroni Correction

Suppose that a collection of  $g$  null hypotheses are being tested; i.e., differential expression of  $g$  genes is being assessed. The **family-wise error rate** (FWER) is the probability of rejecting at least one null hypothesis, given that they are all true (one false positive). If  $\alpha$  is the desired FWER this can be achieved by testing each null hypothesis with a Type I error of  $\alpha/g$ . This is called the **Bonferroni Correction**.

If the desired FWER is 0.05 and there are 1000 genes tested, set the threshold for each at 0.00005.

Slightly less brutal is the Šidák correction that replaces  $\alpha$  by  $K(g, \alpha) = 1 - \sqrt[g]{1 - \alpha}$ , but this still places a very high threshold on satisfying each test.



## Bonferroni Correction

Suppose that a collection of  $g$  null hypotheses are being tested; i.e., differential expression of  $g$  genes is being assessed. The **family-wise error rate** (FWER) is the probability of rejecting at least one null hypothesis, given that they are all true (one false positive). If  $\alpha$  is the desired FWER this can be achieved by testing each null hypothesis with a Type I error of  $\alpha/g$ . This is called the **Bonferroni Correction**.

If the desired FWER is 0.05 and there are 1000 genes tested, set the threshold for each at 0.00005.

Slightly less brutal is the Šidák correction that replaces  $\alpha$  by  $K(g, \alpha) = 1 - \sqrt[g]{1 - \alpha}$ , but this still places a very high threshold on satisfying each test.





## Bonferroni Correction

Suppose that a collection of  $g$  null hypotheses are being tested; i.e., differential expression of  $g$  genes is being assessed. The **family-wise error rate** (FWER) is the probability of rejecting at least one null hypothesis, given that they are all true (one false positive). If  $\alpha$  is the desired FWER this can be achieved by testing each null hypothesis with a Type I error of  $\alpha/g$ . This is called the **Bonferroni Correction**.

If the desired FWER is 0.05 and there are 1000 genes tested, set the threshold for each at 0.00005.

Slightly less brutal is the Šidák correction that replaces  $\alpha$  by  $K(g, \alpha) = 1 - \sqrt[g]{1 - \alpha}$ , but this still places a very high threshold on satisfying each test.



# Step-Down Methods

## Westfall-Young

The methods above treat all tests as the same in that they set the same Type I error for all  $g$  tests. There is another method developed by Westfall-Young, called a **step-down** method, that is less restrictive but still gives an FWER of the desired  $\alpha$ .

There are, in fact, several other processes for controlling FWER. It's a big problem that's received a lot of attention.

The procedure we will use is a recent development using a **bootstrap** estimation of the joint null distribution for all tests. This is theoretically steep. It is implemented in the `multtest` package.



# Step-Down Methods

## Westfall-Young

The methods above treat all tests as the same in that they set the same Type I error for all  $g$  tests. There is another method developed by Westfall-Young, called a **step-down** method, that is less restrictive but still gives an FWER of the desired  $\alpha$ .

There are, in fact, several other processes for controlling FWER. It's a big problem that's received a lot of attention.

The procedure we will use is a recent development using a **bootstrap** estimation of the joint null distribution for all tests. This is theoretically steep. It is implemented in the `multtest` package.



# Step-Down Methods

## Westfall-Young

The methods above treat all tests as the same in that they set the same Type I error for all  $g$  tests. There is another method developed by Westfall-Young, called a **step-down** method, that is less restrictive but still gives an FWER of the desired  $\alpha$ .

There are, in fact, several other processes for controlling FWER. It's a big problem that's received a lot of attention.

The procedure we will use is a recent development using a **bootstrap** estimation of the joint null distribution for all tests. This is theoretically steep. It is implemented in the `multtest` package.



## False Discovery Rate

In some problems controlling the FWER is too lofty a goal. It leaves out false positives but hides too many true positives. A compromise is to control the **false discovery rate** (FDR). The FDR is the percentage of false positives among all the rejected hypotheses.

Given 1000 tests for differential expression, if we control the FDR at 0.05 and the procedure reports 40 genes as differentially expressed, we expect at most 2 of these to be false positives.



## False Discovery Rate

In some problems controlling the FWER is too lofty a goal. It leaves out false positives but hides too many true positives. A compromise is to control the **false discovery rate** (FDR). The FDR is the percentage of false positives among all the rejected hypotheses.

Given 1000 tests for differential expression, if we control the FDR at 0.05 and the procedure reports 40 genes as differentially expressed, we expect at most 2 of these to be false positives.



## Adjusted P-Values

Recall that in a single hypothesis test, a **p-value** is often returned. This is the minimal Type I Error rate for which the null hypothesis would be rejected with the given data. Part of any multiple testing procedure is to create an **adjusted p-value** for each test (each gene). Just as we call the gene differentially expressed in a single test if the p-value is  $<$  the Type I Error, we call a gene differentially expressed in a multiple test procedure if the adjusted p-value is  $<$  the FWER, or FDR if that is what we're controlling. This makes it easy to process these multiple testing procedures as we would single tests.



## Power Versus Error Control

Differential expression analysis is a setting where the problem of minimizing error and maximizing power is at the forefront. It is provably impossible to have it both ways. Setting FWER or FDR at 0.05 increases the probability that there are many differentially expressed genes not on the list the procedure generates. It is only a sample of the differentially expressed genes. This must be kept in mind when drawing conclusions from an analysis.

Better procedures for comparing the “transcription states” of two sample groups are needed. This is now in progress.





## Power Versus Error Control

Differential expression analysis is a setting where the problem of minimizing error and maximizing power is at the forefront. It is provably impossible to have it both ways. Setting FWER or FDR at 0.05 increases the probability that there are many differentially expressed genes not on the list the procedure generates. It is only a sample of the differentially expressed genes. This must be kept in mind when drawing conclusions from an analysis.

Better procedures for comparing the “transcription states” of two sample groups are needed. This is now in progress.



# Outline

## Microarrays

Introduction to Differential Expression Analysis

Multiple Testing Problem

**Prefiltering the Gene List**

Multtest Package

Differential Expression Example



## Prefiltering the Gene List

Many genes are only expressed in specific tissues or at special times, such as in development. Of all the features on an array, only 50% or so will show any sign of expression in a given sample, and fewer still show any variation across the samples. In a search for differentially expressed genes it makes sense to exclude these from the list of genes in the multiple testing process. Making the list of genes shorter makes the multiple testing correction less severe.

This may seem backwards. How do we eliminate genes from a test for differential expression without testing them? We use very crude measures that select only genes that are certainly not of interest.



## Prefiltering the Gene List

Many genes are only expressed in specific tissues or at special times, such as in development. Of all the features on an array, only 50% or so will show any sign of expression in a given sample, and fewer still show any variation across the samples. In a search for differentially expressed genes it makes sense to exclude these from the list of genes in the multiple testing process. Making the list of genes shorter makes the multiple testing correction less severe.

This may seem backwards. How do we eliminate genes from a test for differential expression without testing them? We use very crude measures that select only genes that are certainly not of interest.



## The Genefilter Package

This package contains functions to aid in prefiltering `ExpressionSet` objects. Recall the use of `IQR` previously. Genes in a matrix of expression values were checked for an `IQR` value  $> .5$  and the matrix was restricted to those that meet this requirement. `Genefilter` tools accomplish roughly the same thing, but can be applied to `ExpressionSet` objects more smoothly.

To begin:

```
> library(genefilter)
```



## Filter Functions Are Applied to Expression Sets

**Filter functions** applied to an expression matrix amounts to testing each gene for a condition, returning a logical vector. This vector can then be used to subset the ExpressionSet object.

The method will be illustrated on an ExpressionSet object of breast cancer arrays.

```
> load("./eset3UPPS1")
```



## Show the ExpressionSet

```
> eset3UPPS1
```

```
ExpressionSet (storageMode: lockedEnvironment)
```

```
assayData: 22283 features, 123 samples
```

```
  element names: exprs, se.exprs
```

```
phenoData
```

```
  rowNames: GSM110625, GSM110627, ..., GSM110873 (123 total)
```

```
  varLabels and varMetadata:
```

```
    GSM.ID..A.B.chip.: arbitrary numbering
```

```
    Cohort: arbitrary numbering
```

```
    ...: arbitrary numbering
```

```
    tumor.size..mm.: arbitrary numbering
```

```
featureData
```

```
  featureNames: 1007_s_at, 1053_at, ..., AFX-r2-P1-cre-5_a
```

```
  varLabels and varMetadata: none
```

```
experimentData: use 'experimentData(object)'
```



## First Establish a Floor

Genes that are barely expressed in most of the samples are likely to have little biological interest. `genefilter` has a function `pOverA` that tests if a certain fraction of samples have expression values over a threshold. The following creates a filter function that tests a gene for expression value over 3.5 in 25% of the samples.

```
> f1 <- pOverA(0.25, 3.5)
> ffun1 <- filterfun(f1)
> flrGene <- genefilter(exprs(eset3UPPS1),
+   ffun1)
```





## Filter the ExpressionSet

```
> flrGene[1:5]
```

1007_s_at	1053_at	117_at	121_at	1255_g_at
TRUE	TRUE	TRUE	FALSE	FALSE

```
> sum(flrGene)
```

```
[1] 13484
```

This removes almost 10,000 probes. Use this logical vector to subset the ExpressionSet.

```
> flset3UPPS1 <- eset3UPPS1[flrGene, ]
```



## Further Filter by IQR

Retain only genes with an IQR  $> .75$ , a reasonable spread with a good outcome.

```
> f2 <- function(x) {  
+   IQR(x) > 0.75  
+ }  
> ffun2 <- filterfun(f2)  
> sprdGns <- genefilter(exprs(f1eset3UPPS1),  
+   ffun2)  
> sum(sprdGns)  
  
[1] 5390  
  
> f3eset3UPPS1 <- f1eset3UPPS1[sprdGns,  
+   ]
```



## Prefiltering Result

```
> f3eset3UPPS1
```

```
ExpressionSet (storageMode: lockedEnvironment)
```

```
assayData: 5390 features, 123 samples
```

```
  element names: exprs, se.exprs
```

```
phenoData
```

```
  rowNames: GSM110625, GSM110627, ..., GSM110873 (123 total)
```

```
  varLabels and varMetadata:
```

```
    GSM.ID..A.B.chip.: arbitrary numbering
```

```
    Cohort: arbitrary numbering
```

```
    ...: arbitrary numbering
```

```
    tumor.size..mm.: arbitrary numbering
```

```
featureData
```

```
  rowNames: 1007_s_at, 1294_at, ..., AFX-r2-Hs28SrRNA-3_at
```

```
  varLabels and varMetadata: none
```

```
experimentData: use 'experimentData(object)'
```



# Outline

## Microarrays

Introduction to Differential Expression Analysis

Multiple Testing Problem

Prefiltering the Gene List

**Multtest Package**

Differential Expression Example



## A Versatile Package for Multiple Testing

The `multtest` package is useful for performing differential expression analysis. A single function `MTP` handles numerous testing problems simply by setting optional parameters. First load the library.

```
> library(multtest)
```

**Problem.** Identify a set of genes differentially expressed in the breast cancer expression set between the samples with normal p53 status and mutated p53.

The factor in the `phenoData` defining the p53 status is named `p53` with the following distribution. `p53+` denotes the mutant.

```
> table(phenoData(f3eset3UPPS1)$p53)
```

```
p53+ p53-  
  27   96
```



## MTP Parameters

The function performing the analysis is called MTP. Below are the most important options found from ?MTP.

In any differential expression problem there is a need to specify the matrix of expression values and the factor defining the two sample groups. The matrix of expression values are assigned to the parameter  $X$ . The factor defining the sample groups is assigned to  $Y$ .

Note that it is possible to do an ANOVA test when the factor has more than two levels.



## MTP Options

for setting test statistic

The parameter `test` is used to set the test statistic. The default value is `t.twosamp.unequalvar`. For an ANOVA use `f`.

To use a nonparametric version of a test, like the Wilcoxon rank sum test, set the parameter `robust` equal to `TRUE`. The default is `FALSE`.

`alternative` is a character string that describes the form of the alternative hypothesis. The default is `two.sided`. Other possibilities are `less` or `greater`.

`typeone` sets the type of control over multiple testing. The default is `fwere`, but `fdr` is a good option. Others are possible.

`alpha` sets the Type I error rate, after multiple testing correction. The default is `0.05`. It is possible to give a vector if you want to test multiple levels.



## MTP Options

for setting test statistic

The parameter `test` is used to set the test statistic. The default value is `t.twosamp.unequalvar`. For an ANOVA use `f`.

To use a nonparametric version of a test, like the Wilcoxon rank sum test, set the parameter `robust` equal to `TRUE`. The default is `FALSE`.

`alternative` is a character string that describes the form of the alternative hypothesis. The default is `two.sided`. Other possibilities are `less` or `greater`.

`typeone` sets the type of control over multiple testing. The default is `fwere`, but `fdr` is a good option. Others are possible.

`alpha` sets the Type I error rate, after multiple testing correction. The default is `0.05`. It is possible to give a vector if you want to test multiple levels.





## MTP Options

for setting test statistic

The parameter `test` is used to set the test statistic. The default value is `t.twosamp.unequalvar`. For an ANOVA use `f`.

To use a nonparametric version of a test, like the Wilcoxon rank sum test, set the parameter `robust` equal to `TRUE`. The default is `FALSE`.

`alternative` is a character string that describes the form of the alternative hypothesis. The default is `two.sided`. Other possibilities are `less` or `greater`.

`typeone` sets the type of control over multiple testing. The default is `fwer`, but `fdr` is a good option. Others are possible.

`alpha` sets the Type I error rate, after multiple testing correction. The default is `0.05`. It is possible to give a vector if you want to test multiple levels.



## MTP Options

for setting test statistic

The parameter `test` is used to set the test statistic. The default value is `t.twosamp.unequalvar`. For an ANOVA use `f`.

To use a nonparametric version of a test, like the Wilcoxon rank sum test, set the parameter `robust` equal to `TRUE`. The default is `FALSE`.

`alternative` is a character string that describes the form of the alternative hypothesis. The default is `two.sided`. Other possibilities are `less` or `greater`.

`typeone` sets the type of control over multiple testing. The default is `fwer`, but `fdr` is a good option. Others are possible.

`alpha` sets the Type I error rate, after multiple testing correction. The default is `0.05`. It is possible to give a vector if you want to test multiple levels.



## MTP Options

for setting test statistic

The parameter `test` is used to set the test statistic. The default value is `t.twosamp.unequalvar`. For an ANOVA use `f`.

To use a nonparametric version of a test, like the Wilcoxon rank sum test, set the parameter `robust` equal to `TRUE`. The default is `FALSE`.

`alternative` is a character string that describes the form of the alternative hypothesis. The default is `two.sided`. Other possibilities are `less` or `greater`.

`typeone` sets the type of control over multiple testing. The default is `fwer`, but `fdr` is a good option. Others are possible.

`alpha` sets the Type I error rate, after multiple testing correction. The default is `0.05`. It is possible to give a vector if you want to test multiple levels.



## MTP Output

a new object class

The MTP function returns an object of class MTP. There are numerous slots. The important ones for reading off the differentially expressed genes and understanding the amount of differential expression are as follows.

`statistic` : The vector of values of the test statistic, one for each null hypothesis; i.e., probe.

`adjp` : The vector of adjusted p-values. When this is  $< \alpha$  we reject the null.

`reject` : This is a logical matrix reporting if a null hypothesis is rejected for a given gene (rows) and value of  $\alpha$  (columns). (Normally this has one column.)



## MTP Output

a new object class

The MTP function returns an object of class MTP. There are numerous slots. The important ones for reading off the differentially expressed genes and understanding the amount of differential expression are as follows.

**statistic** : The vector of values of the test statistic, one for each null hypothesis; i.e., probe.

**adjp** : The vector of adjusted p-values. When this is  $< \alpha$  we reject the null.

**reject** : This is a logical matrix reporting if a null hypothesis is rejected for a given gene (rows) and value of  $\alpha$  (columns). (Normally this has one column.)



## MTP Output

a new object class

The MTP function returns an object of class MTP. There are numerous slots. The important ones for reading off the differentially expressed genes and understanding the amount of differential expression are as follows.

**statistic** : The vector of values of the test statistic, one for each null hypothesis; i.e., probe.

**adjp** : The vector of adjusted p-values. When this is  $< \alpha$  we reject the null.

**reject** : This is a logical matrix reporting if a null hypothesis is rejected for a given gene (rows) and value of alpha (columns). (Normally this has one column.)



## MTP Output

a new object class

The MTP function returns an object of class MTP. There are numerous slots. The important ones for reading off the differentially expressed genes and understanding the amount of differential expression are as follows.

`statistic` : The vector of values of the test statistic, one for each null hypothesis; i.e., probe.

`adjp` : The vector of adjusted p-values. When this is  $< \alpha$  we reject the null.

`reject` : This is a logical matrix reporting if a null hypothesis is rejected for a given gene (rows) and value of  $\alpha$  (columns). (Normally this has one column.)



## Getting Results

from the MTP object

Given an MTP object, `mtpObj`, `summary(mtpObj)`, reports simple results like the number of rejections, however we need to know **which** probes are rejected.

`mtpObj@reject` is a logical matrix. When `alpha` is a single number, `mtpObj@reject` has one column and the vector of rejects can be obtained by

```
rejs <- rownames(mtpObj@reject)[mtpObj@reject]
```

`rejs` is then a vector of probes. We can use other slots in `mtpObj` to find corresponding adjusted p-values and statistic values.

Annotation information is used to identify gene names and symbols from the probes. Normally, they are reported in a table sorted by adjusted p-value. The function `mtpOut` (written by myself) produces such a table.





# Outline

## Microarrays

Introduction to Differential Expression Analysis

Multiple Testing Problem

Prefiltering the Gene List

Multtest Package

Differential Expression Example



## Example Problem from Breast Cancer

**Problem.** Identify a set of genes differentially expressed in the breast cancer expression set between the samples with normal p53 status and mutated p53.

The ExpressionSet object following prefiltering is `f3eset3UPPS1` and the factor defining p53 status is `phenoData(f3eset3UPPS1)$p53`.

Do differential expression using MTP with all defaults: a two-sided, two sample t test with unequal variances, FWER as MTP adjustment, and  $\alpha = .05$ .



## Example Problem from Breast Cancer

**Problem.** Identify a set of genes differentially expressed in the breast cancer expression set between the samples with normal p53 status and mutated p53.

The ExpressionSet object following prefiltering is `f3eset3UPPS1` and the factor defining p53 status is `phenoData(f3eset3UPPS1)$p53`.

Do differential expression using MTP with all defaults: a two-sided, two sample t test with unequal variances, FWER as MTP adjustment, and  $\alpha = .05$ .



## Example Problem from Breast Cancer

**Problem.** Identify a set of genes differentially expressed in the breast cancer expression set between the samples with normal p53 status and mutated p53.

The ExpressionSet object following prefiltering is `f3eset3UPPS1` and the factor defining p53 status is `phenoData(f3eset3UPPS1)$p53`.

Do differential expression using MTP with all defaults: a two-sided, two sample t test with unequal variances, FWER as MTP adjustment, and  $\alpha = .05$ .



## Execute MTP on this Set

```
> X3 <- exprs(f3eset3UPPS1)
> Fp <- f3eset3UPPS1$p53
> names(Fp) <- colnames(X3)

> mtp531 <- MTP(X = X3, Y = Fp)

  running bootstrap...
iteration = 100 200 300 400 500 600 700 800 900 1000
```



## MTP Object Summary

```
> summary(mtp531)
```

```
MTP:  ss.maxT
```

```
Type I error rate:  fwer
```

### Level Rejections

```
alpha=0.05  0.05          89
```

	Min.	1st Qu.	Median	Mean	3rd Qu.
adjp	0.00	0.993	1.0000	0.9100	1.000
rawp	0.00	0.010	0.0995	0.2440	0.423
statistic	-9.59	-1.290	0.3570	0.2450	1.890
estimate	-3.02	-0.269	0.0719	0.0638	0.408

Max.

```
adjp      1.00
```

```
rawp      1.00
```



## A Table of Rejects

Now use the `mtpOut` function or one you write to create a table of results. The function writes the

```
> mtpP53out <- mtpOut(mtp531, X3, Fp, hgu133aSYMBOL,  
+   hgu133aGENENAME, "mtp53OutTbl1")
```

The file written into the working directory is `mtp53OutTbl1.csv`.



## Interpreting the Gene List

A check of the values for an individual gene shows that a negative t statistic in the table means the probe is up-regulated in the mutant (p53+).

Many of the genes on the list are involved in cell cycle regulation. Cell cycle acceleration is a major factor in the aggressiveness of breast cancer. This gives some insight into the way p53 mutation promotes tumor growth.

Interpretation of this information can be aided by a facility for marking up KEGG pathway diagrams. The function `prepColors3` produces a data.frame of coloring codes and Entrez Ids in the format KEGG needs.





## Interpreting the Gene List

A check of the values for an individual gene shows that a negative  $t$  statistic in the table means the probe is up-regulated in the mutant (p53+).

Many of the genes on the list are involved in cell cycle regulation. Cell cycle acceleration is a major factor in the aggressiveness of breast cancer. This gives some insight into the way p53 mutation promotes tumor growth.

Interpretation of this information can be aided by a facility for marking up KEGG pathway diagrams. The function `prepColors3` produces a data.frame of coloring codes and Entrez Ids in the format KEGG needs.



## Interpreting the Gene List

A check of the values for an individual gene shows that a negative t statistic in the table means the probe is up-regulated in the mutant (p53+).

Many of the genes on the list are involved in cell cycle regulation. Cell cycle acceleration is a major factor in the aggressiveness of breast cancer. This gives some insight into the way p53 mutation promotes tumor growth.

Interpretation of this information can be aided by a facility for marking up KEGG pathway diagrams. The function `prepColors3` produces a data.frame of coloring codes and Entrez Ids in the format KEGG needs.



## Mark-up KEGG Pathways

```
> colTbl1 <- prepColors3(mtpP53out, hgu133aENTREZID)
> write.csv(colTbl1, file = "colTbl1.csv")
```

As this mark-up shows, RRM2, significantly up-regulated in the mutant samples, is involved in DNA repair with p53.



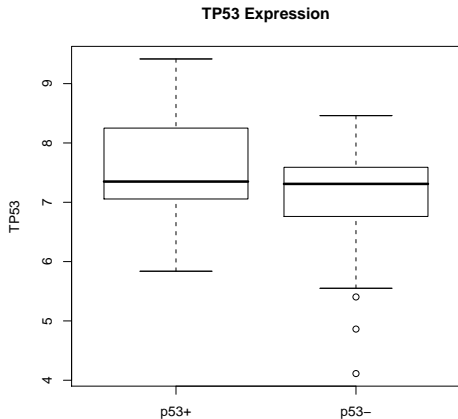
## Where is p53?

We are comparing samples with normal p53 to those with a mutant p53. Why isn't p53 at the top of the list of genes differentially expressed here?

Plot the expression values of the p53 gene for the two sample groups and compare. The gene symbol is TP53. A search of the symbols reveals two probes for this gene. In one probe the expression values are uniformly very low. For the other the plot follows.



## Plot of TP53 Expression





## Nature of the Mutation of p53

The mutation of TP53 does not prohibit transcription of the mRNA or translation of the mRNA into a protein. However, the mutant protein structure prohibits its normal binding to DNA. That is, it cannot act as a transcription factor like the wild-type p53 protein. Thus, the effect of the mutation is found “down-stream” in the other genes. The mutation in the mRNA is likely not in the 25mer comprising the probe so there is no significant change in expression value measurement.



## How Complete is the Picture?

We generated a list of 89 differentially expressed genes, corrected for multiple testing. How many have we missed? To gain some insight, we execute a t test on each probe and consider it **probably differentially expressed** if the p-value of this test is  $< 0.05$ . Leave out the multiple testing correction.

The `genefilter` package provides a function `rowttests` for executing a t test on each row of an expression matrix based on some factor.



## Testing the Power

```
> ttests <- rowttests(X3, Fp)
> probDiff <- ttests$p.value < 0.05
> sum(probDiff)
```

```
[1] 2296
```

This is the sobering reality of problems in which the number of variables significantly exceeds the number of samples. What is gained through these basic methods is still valuable, but we have only begun to understand the expression-based differences between cell types. Methods are under development and significant improvements aren't unreasonable.