Microarray Gene Expression Analysis with R

Example: Interferon Regulatory Factor 6 (*IRF6*)

Contents

0.1 Objectives

- Load microarray dataset into R
- Explore the dataset with basic visualizations
- Identify differentially expressed genes (DEGs)
- Generate annotation of the DEGs (*Tentative*)

0.2 The Central Dogma of Biology

0.3 Cleft Lip and Palate 1/3

Cleft lip and cleft palate (**CLP**) are splits in the upper lip, the roof of the mouth (palate) or both. They result when facial structures that are developing in an unborn baby do not close completely. CLP is one of the most common birth defects with a frequency of 1/700 live births.

0.4 Cleft Lip and Palate 2/3

Children with cleft lip with or without cleft palate face a variety of challenges, depending on the type and severity of the cleft.

- **Difficulty feeding.** One of the most immediate concerns after birth is feeding.
- **Ear infections and hearing loss.** Babies with cleft palate are especially at risk of developing middle ear fluid and hearing loss.
- **Dental problems.** If the cleft extends through the upper gum, tooth development may be affected.
- **Speech difficulties.** Because the palate is used in forming sounds, the development of normal speech can be affected by a cleft palate. Speech may sound too nasal.

Reference: Mayo Foundation for Medical Education and Research

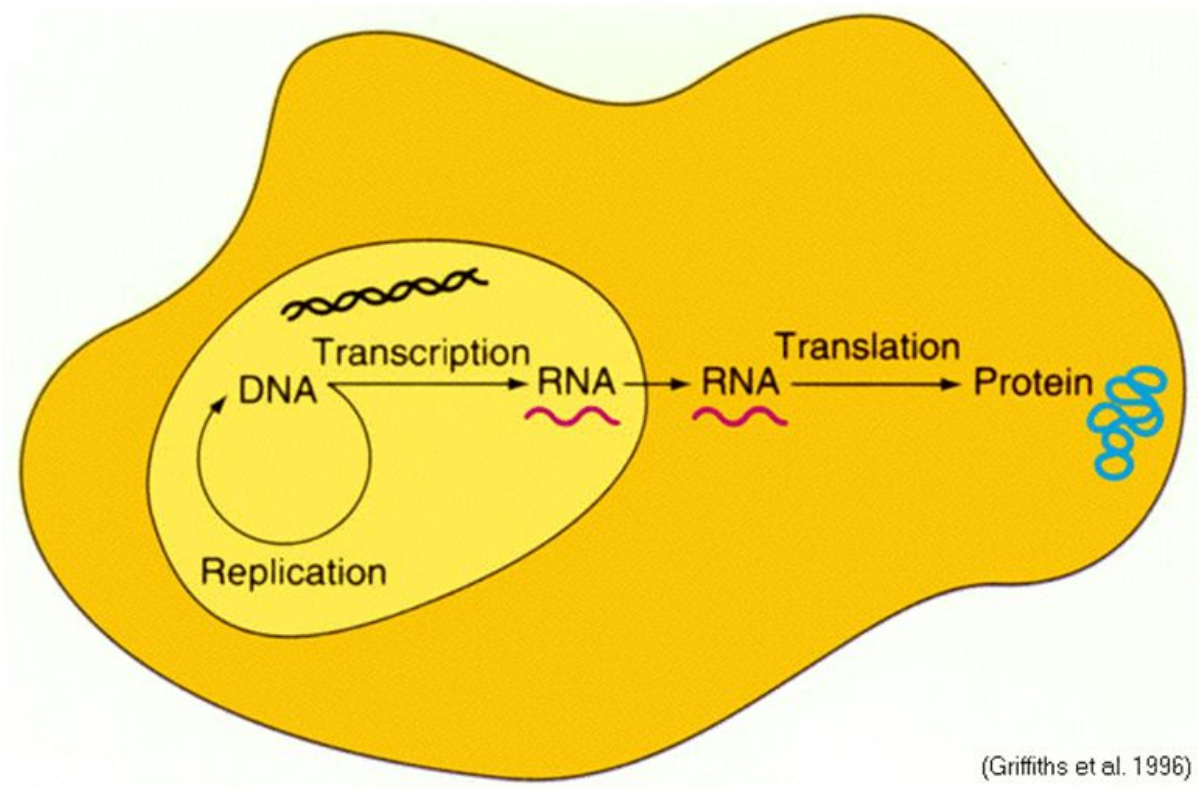


Figure 1: DNA makes RNA and RNA makes protein

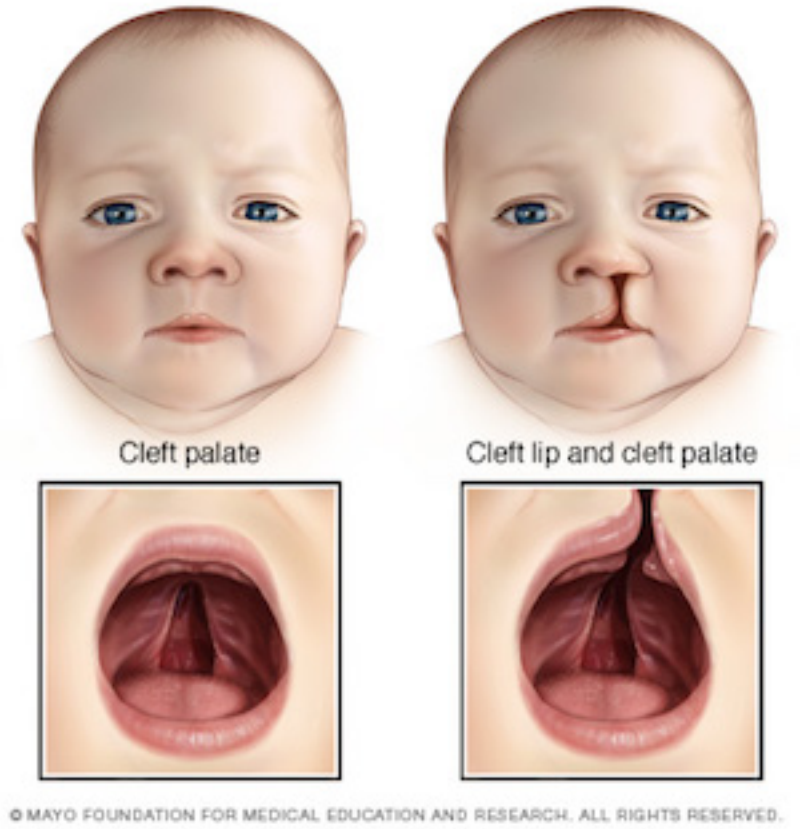


Figure 2: Cleft lip and palate

0.5 Cleft Lip and Palate 3/3

- DNA variation in Interferon Regulatory Factor 6 (**IRF6**) causes Van der Woude syndrome (**VWS**)
- VWS is the most common syndromic form of cleft lip and palate.
- However, the causing variant in IRF6 has been found in *only* 70% of VWS families!
- IRF6 is a **transcription factor** with a conserved helix-loop-helix DNA binding domain and a less well-conserved protein binding domain.

Reference: Hum Mol Genet. 2014 May 15; 23(10): 2711–2720

0.6 Question

Given:

1. The pathogenic variant in IRF6 exists in only 70% of the VWS families
2. IRF6 is a transcription factor

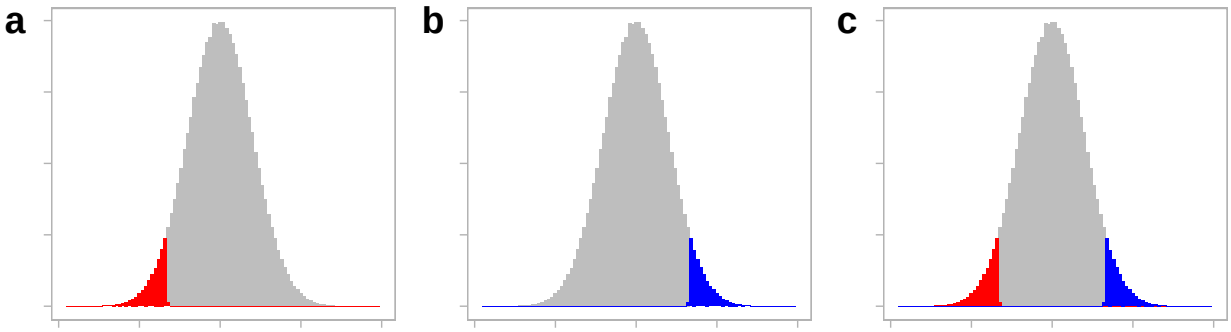
How can we identify other genes that might be involved in the remaining 30% of the VWS families?

0.7 Hint

- Usually, genes that are regulated by a transcription factor belong to the same biological process or pathway.
- Therefore, by comparing the gene expression patterns between wild-type (functional) *Irf6* and knockout (non-functional) *Irf6*, it could be possible to identify genes that are regulated (targeted) by *Irf6*.

0.8 Hypothesis

- $H_O : \mu_{WT} = \mu_{KO}$
- $H_A : \mu_{WT} \neq \mu_{KO}$
- Where μ is the *mean* of the gene expression values of a gene.
- **One**-sided or **Two**-sided testing?



0.9 Why Microarray?

0.10 Why Microarray?

- No need for candidate genes (or genes of interest)
- One experiment assesses the entire transcriptome
- One experiment generates many hypotheses
- Only small amount of RNA is required (~15–200 ng)

0.11 Original Paper

Nat Genet. 2006 Nov;38(11):1335-40. Epub 2006 Oct 15.

Abnormal skin, limb and craniofacial morphogenesis in mice deficient for interferon regulatory factor 6 (Irf6).

Ingraham CR[1], Kinoshita A, Kondo S, Yang B, Sajan S, Trout KJ, Malik MI, Dunnwald M, Goudy SL, Lovett M, Murray JC, Schutte BC.

Author information

Abstract

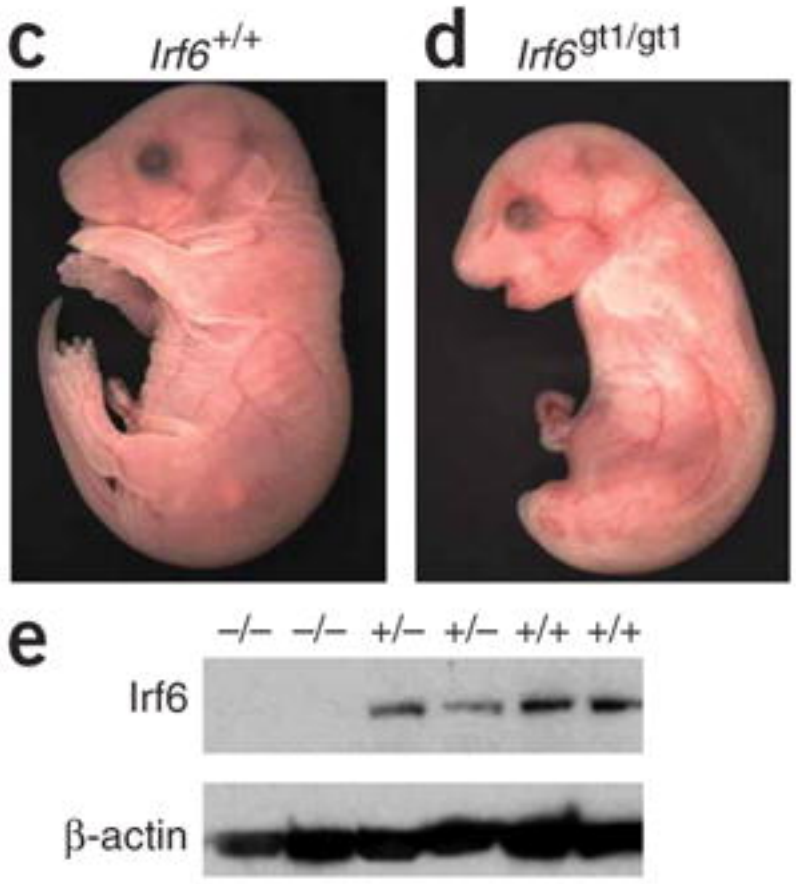
Transcription factor paralogs may share a common role in staged or overlapping expression in specific tissues, as in the Hox family. In other cases, family members have distinct roles in a range of embryologic, differentiation or response pathways (as in the Tbx and Pax families). For the interferon regulatory factor (IRF) family of transcription factors, mice deficient in Irf1, Irf2, Irf3, Irf4, Irf5, Irf7, Irf8 or Irf9 have defects in the immune response but show no embryologic abnormalities. Mice deficient for Irf6 have not been reported, but in humans, mutations in IRF6 cause two mendelian orofacial clefting syndromes, and genetic variation in IRF6 confers risk for isolated cleft lip and palate. Here we report that mice deficient for Irf6 have abnormal skin, limb and craniofacial development. Histological and gene expression analyses indicate that the primary defect is in keratinocyte differentiation and proliferation. This study describes a new role for an IRF family member in epidermal development.

PMID: 17041601 PMCID: PMC2082114 DOI: 10.1038/ng1903

Figure 3: PMID: 17041601

0.12 Experimental Design

- 3 IRF6 wild-type (+/+) and 3 knockout (-/-) mouse embryos.
- E17.5 embryos were removed from euthanized mothers.
- Skin was removed from embryos.
- Total RNA was isolated from the skin.
- Resultant RNA was hybridized to Affymetrix GeneChip Mouse Genome 430 2.0 arrays.



0.13 Dataset

- The original dataset can be obtained from NCBI GEO with accession GSE5800

ID	KO1	KO2	KO3	WT1	WT2	WT3
1415670_at	6531.0	5562.8	6822.4	7732.1	7191.2	7551.9
1415671_at	11486.3	10542.7	10641.4	10408.2	9484.5	7650.2
1415672_at	14339.2	13526.1	14444.7	12936.6	13841.7	13285.7
1415673_at	3156.8	2219.5	3264.4	2374.2	2201.8	2525.3

0.14 Loading

First, we are going to load the dataset from the `.tsv` file into `R` as a variable called `data` using the `read.table` function. `data` is just an arbitrary **varilable** name to hold the result of `read.table` and it can be called/named *almost* anything.

```
# Load the data from a file into a variable
data = read.table("https://raw.githubusercontent.com/ahmedmoustafa/AUCBIOT5206/master/microarray/dataset

# Convert the data.frame (table) in a matrix (numeric)
data = as.matrix(data)
```

Note: the hash sign (`#`) indicates that what comes after is a *comment*. Comments are for documentation

and readability of the R code and they are not evaluated (or executed).

0.15 Checking

```
dim(data) # Dimension of the dataset
```

```
## [1] 45101     6
```

```
head(data) # First few rows
```

	KO1	KO2	KO3	WT1	WT2	WT3
1415670_at	6531.0	5562.8	6822.4	7732.1	7191.2	7551.9
1415671_at	11486.3	10542.7	10641.4	10408.2	9484.5	7650.2
1415672_at	14339.2	13526.1	14444.7	12936.6	13841.7	13285.7
1415673_at	3156.8	2219.5	3264.4	2374.2	2201.8	2525.3
1415674_a_at	4002.0	3306.9	3777.0	3760.6	3137.0	2911.5
1415675_at	3468.4	3347.4	3332.9	3073.5	3046.0	2914.4

0.16 Number of Genes and IDs

```
number_of_genes = nrow(data) # number of genes = number of rows
number_of_genes
```

```
## [1] 45101
```

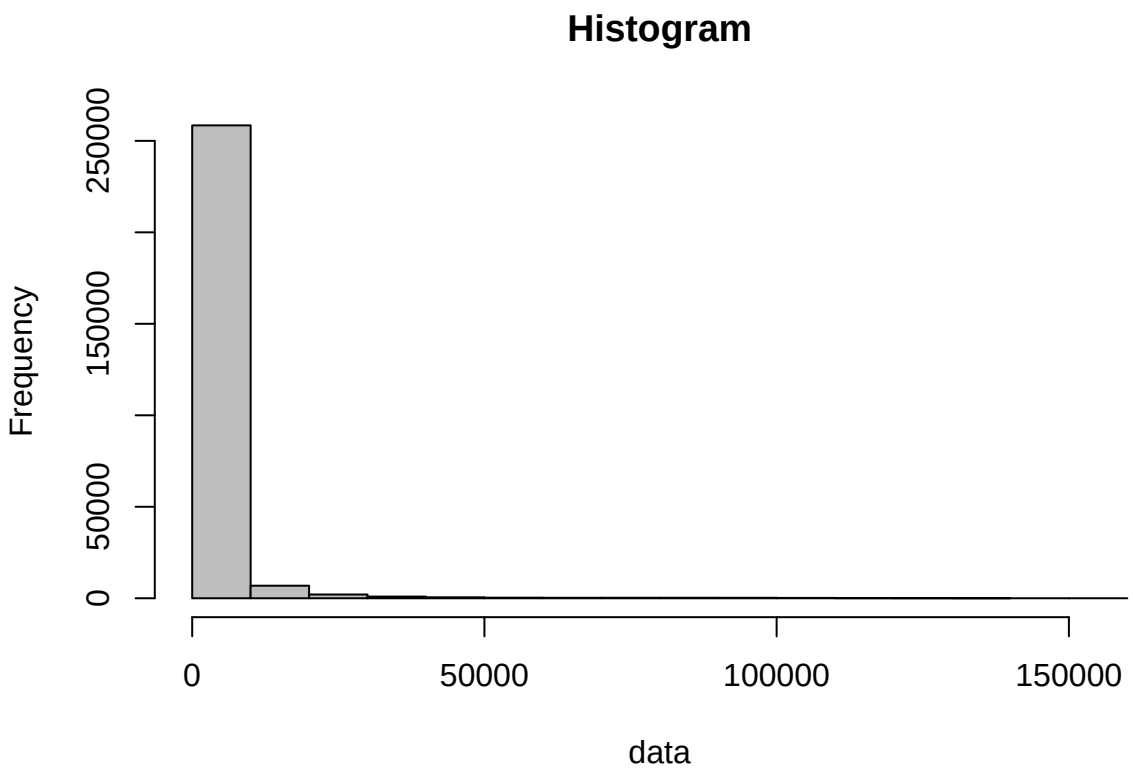
```
ids = row.names(data) # The ids of the genes are the names of the rows
head(ids)
```

```
## [1] "1415670_at"   "1415671_at"   "1415672_at"   "1415673_at"   "1415674_a_at"
## [6] "1415675_at"
```

0.17 Exploring

Check the behavior of the data (e.g., normal?, skewed?)

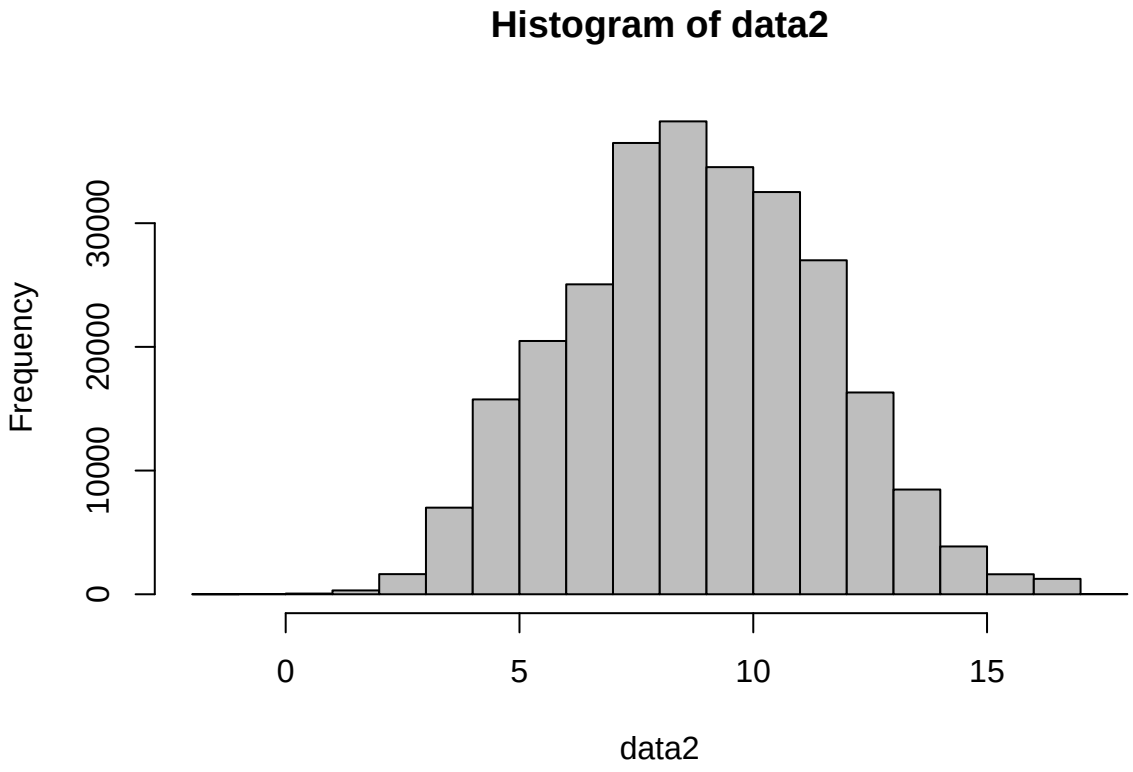
```
hist(data, col = "gray", main="Histogram")
```



0.18 Transforming

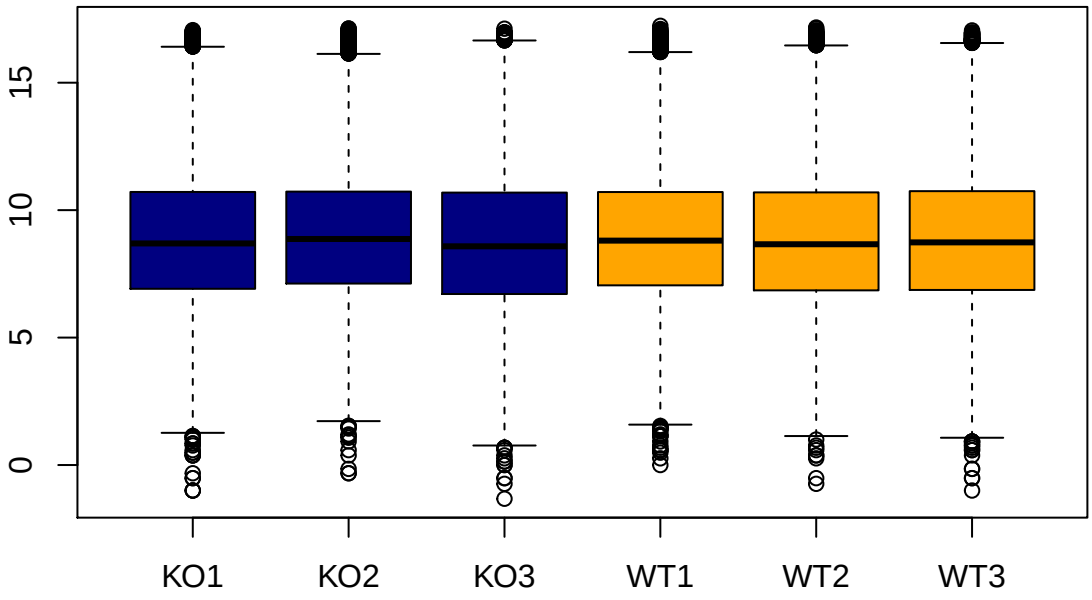
log_2 transformation (why?)

```
data2 = log2(data)
hist(data2, col = "gray")
```



0.19 Boxplot

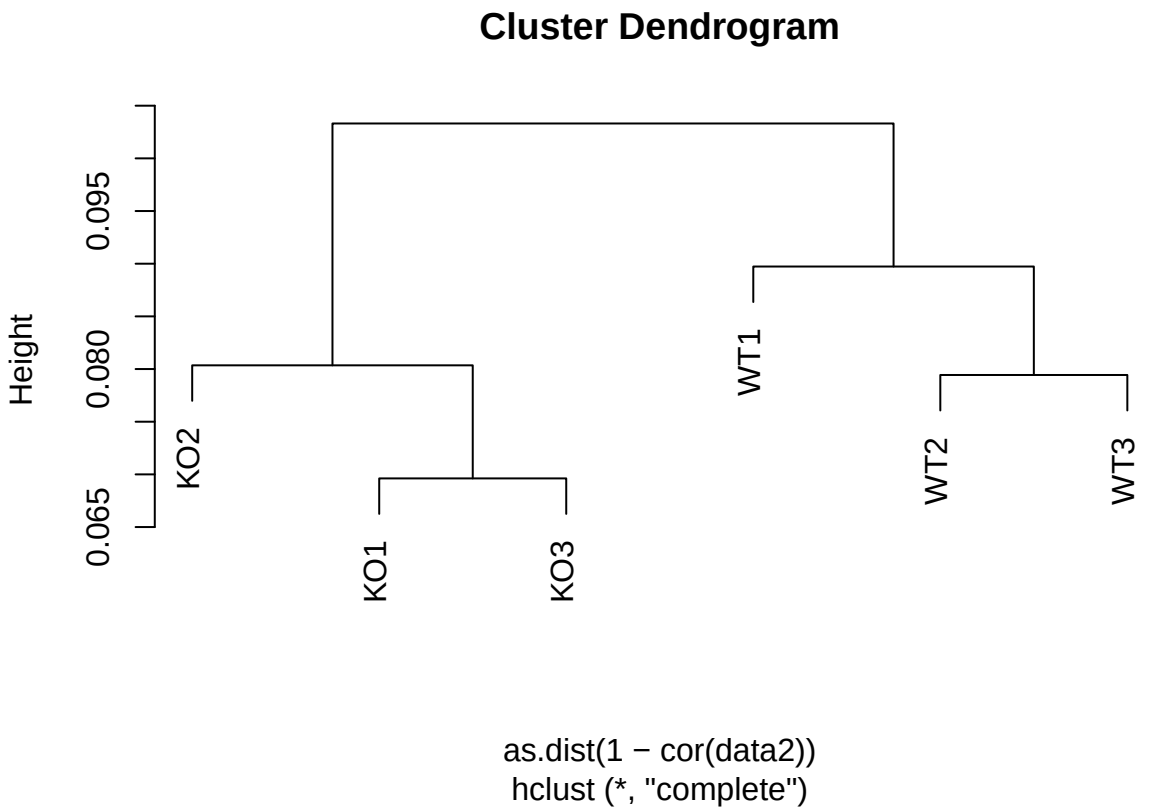
```
colors = c(rep("navy", 3), rep("orange", 3))
boxplot(data2, col = colors)
```



0.20 Clustering 1/2

Hierarchical clustering of the **samples** (i.e., columns) based on the correlation coefficients of the expression values

```
hc = hclust(as.dist(1 - cor(data2)))
plot(hc)
```



0.21 Clustering 2/2

To learn more about a function (e.g., `hclust`), you may type `?function` (e.g., `?hclust`) in the `console` to launch `R` documentation on that function:

0.22 Splitting Data Matrix into Two 1/2

```
ko = data2[, 1:3] # KO matrix
head(ko)
```

	KO1	KO2	KO3
1415670_at	12.67309	12.44160	12.73606
1415671_at	13.48763	13.36396	13.37740
1415672_at	13.80768	13.72346	13.81825
1415673_at	11.62425	11.11602	11.67260
1415674_a_at	11.96651	11.69126	11.88303
1415675_at	11.76005	11.70883	11.70256

0.23 Splitting Data Matrix into Two 2/2

```
wt = data2[, 4:6] # WT matrix
head(wt)
```

	WT1	WT2	WT3
1415670_at	12.91664	12.81202	12.88262
1415671_at	13.34543	13.21136	12.90128
1415672_at	13.65917	13.75673	13.69759
1415673_at	11.21323	11.10447	11.30224
1415674_a_at	11.87675	11.61517	11.50755
1415675_at	11.58567	11.57270	11.50898

0.24 Gene (Row) Mean Expression

```
# Compute the means of the KO samples
ko.means = rowMeans(ko)
head(ko.means)
```

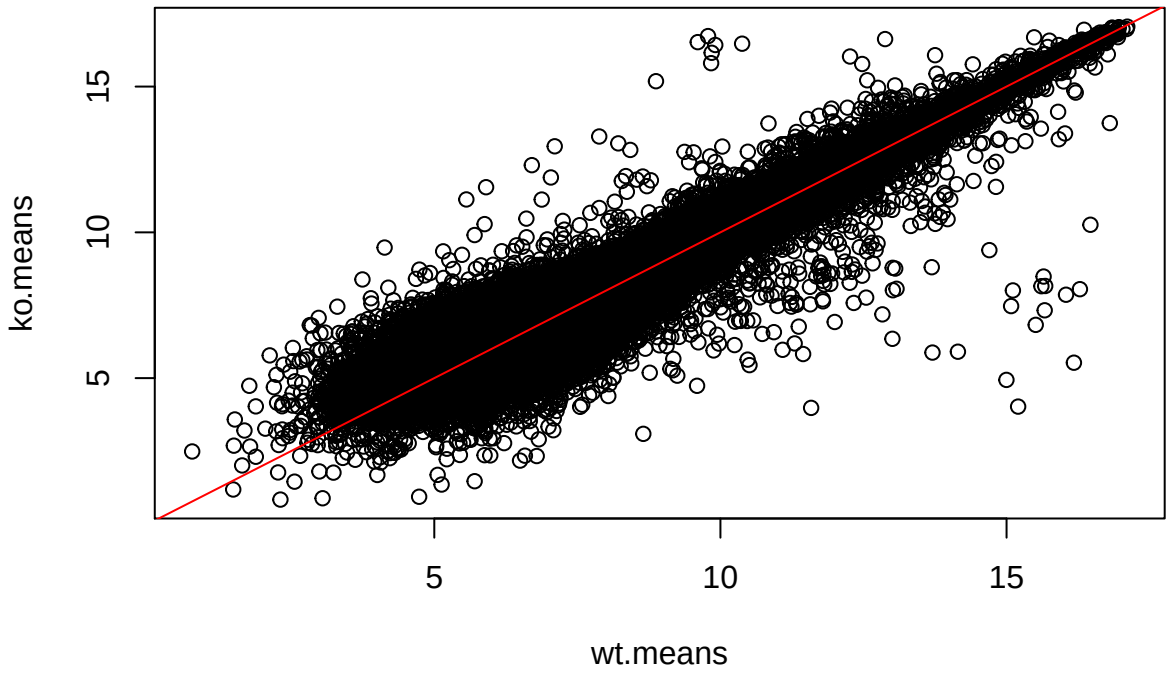
```
##   1415670_at   1415671_at   1415672_at   1415673_at 1415674_a_at   1415675_at
##     12.61692     13.40966     13.78313     11.47096     11.84693     11.72381
```

```
# Compute the means of the WT samples
wt.means = rowMeans(wt)
head(wt.means)
```

```
##   1415670_at   1415671_at   1415672_at   1415673_at 1415674_a_at   1415675_at
##     12.87043     13.15269     13.70450     11.20664     11.66649     11.55578
```

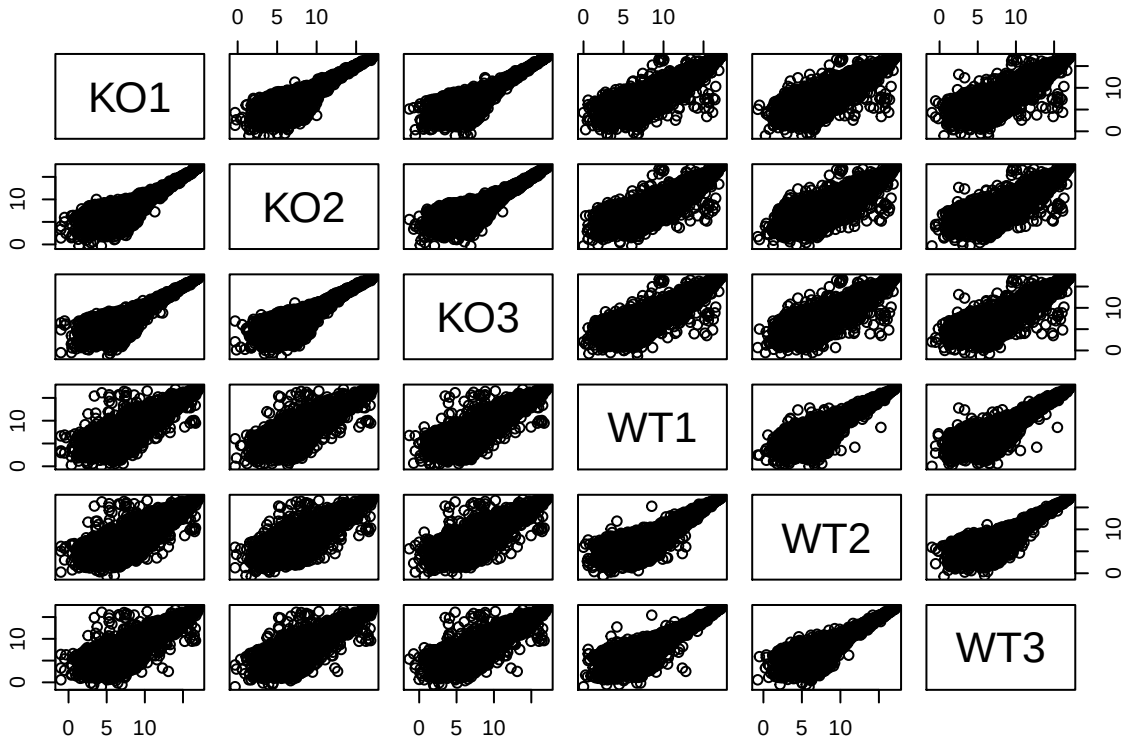
0.25 Scatter 1/2

```
plot(ko.means ~ wt.means) # The actual scatter plot
abline(0, 1, col = "red") # Only a diagonal line
```



0.26 Scatter 2/2

```
pairs(data2) # All pairwise comparisons
```

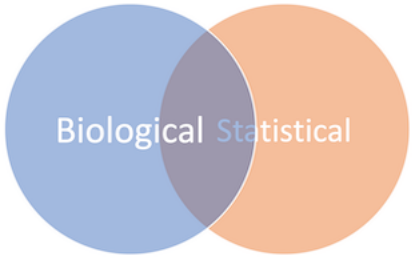


0.27 Differentially Expressed Genes (DEGs)

To identify DEGs, we will identify:

- **Biologically** significantly differentially expressed
- **Statistically** significantly differentially expressed

Then, we will take the **overlap** (**intersection**) of the two sets



0.28 Biological Significance (fold-change) 1/2

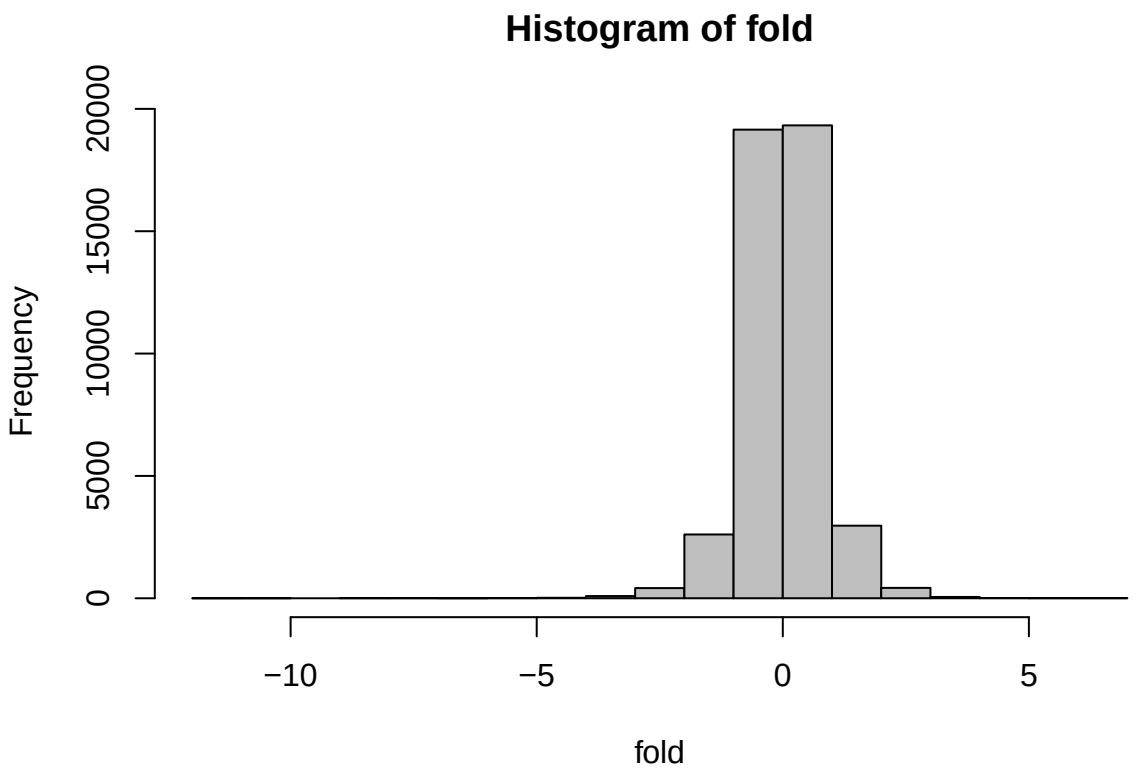
```
fold = ko.means - wt.means # Difference between means
head(fold)
```

```
##   1415670_at   1415671_at   1415672_at   1415673_at 1415674_a_at   1415675_at
##  -0.25351267   0.25697097   0.07863227   0.26431191   0.18044345   0.16803065
```

- What do the positive and negative values of the fold-change indicate? Considering the `WT` condition is the **reference** (or **control**)
- **+ve** fold-change → **Up**-regulation ↑
- **-ve** fold-change → **Down**-regulation ↓

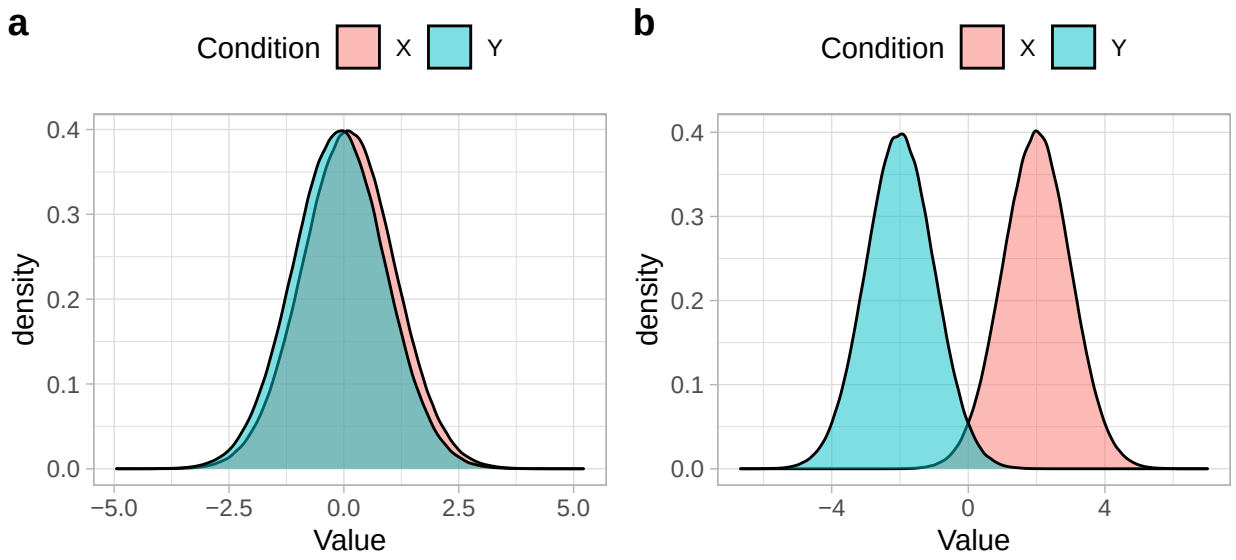
0.29 Biological Significance (fold-change) 2/2

```
hist(fold, col = "gray") # Histogram of the fold
```



0.30 Statistical Significance (p-value) 1/3

- To assess the statistical significance of the difference in the expression values for each gene between the two conditions (e.g., `WT` and `KO`), we are going to use t-test.



0.31 t-test

Let's say there are two samples x and y from the two populations, X and Y, respectively, to determine whether the means of two populations are significantly different, we can use `t.test`.

```
?t.test
```

```
## Student's t-Test
##
## Description:
##
##      Performs one and two sample t-tests on vectors of data.
##
## Usage:
##
##      t.test(x, ...)
##
##      ## Default S3 method:
##      t.test(x, y = NULL,
##             alternative = c("two.sided", "less", "greater"),
##             mu = 0, paired = FALSE, var.equal = FALSE,
##             conf.level = 0.95, ...)
##
##      ## S3 method for class 'formula'
##      t.test(formula, data, subset, na.action, ...)
##
## Arguments:
##
##        x: a (non-empty) numeric vector of data values.
##
##        y: an optional (non-empty) numeric vector of data values.
##
## alternative: a character string specifying the alternative hypothesis,
##           must be one of '"two.sided"' (default), '"greater"' or
##           '"less"'.  You can specify just the initial letter.
##
##       mu: a number indicating the true value of the mean (or difference
##           in means if you are performing a two sample test).
##
##   paired: a logical indicating whether you want a paired t-test.
##
## var.equal: a logical variable indicating whether to treat the two
##           variances as being equal. If 'TRUE' then the pooled variance
##           is used to estimate the variance otherwise the Welch (or
##           Satterthwaite) approximation to the degrees of freedom is
##           used.
##
## conf.level: confidence level of the interval.
##
##  formula: a formula of the form 'lhs ~ rhs' where 'lhs' is a numeric
##           variable giving the data values and 'rhs' either '1' for a
##           one-sample or paired test or a factor with two levels giving
##           the corresponding groups. If 'lhs' is of class '"Pair"' and
##           'rhs' is '1', a paired test is done
##
##     data: an optional matrix or data frame (or similar: see
##           'model.frame') containing the variables in the formula
##           'formula'.  By default the variables are taken from
##           'environment(formula)'.
```

```
##
##   subset: an optional vector specifying a subset of observations to be
##           used.
##
## na.action: a function which indicates what should happen when the data
##           contain 'NA's.  Defaults to 'getOption("na.action")'.
##
##      ...: further arguments to be passed to or from methods.
##
## Details:
##
##      'alternative = "greater"' is the alternative that 'x' has a larger
##      mean than 'y'. For the one-sample case: that the mean is positive.
##
##      If 'paired' is 'TRUE' then both 'x' and 'y' must be specified and
##      they must be the same length.  Missing values are silently removed
##      (in pairs if 'paired' is 'TRUE').  If 'var.equal' is 'TRUE' then
##      the pooled estimate of the variance is used.  By default, if
##      'var.equal' is 'FALSE' then the variance is estimated separately
##      for both groups and the Welch modification to the degrees of
##      freedom is used.
##
##      If the input data are effectively constant (compared to the larger
##      of the two means) an error is generated.
##
## Value:
##
##      A list with class '"htest"' containing the following components:
##
## statistic: the value of the t-statistic.
##
## parameter: the degrees of freedom for the t-statistic.
##
##  p.value: the p-value for the test.
##
## conf.int: a confidence interval for the mean appropriate to the
##           specified alternative hypothesis.
##
## estimate: the estimated mean or difference in means depending on
##           whether it was a one-sample test or a two-sample test.
##
## null.value: the specified hypothesized value of the mean or mean
##           difference depending on whether it was a one-sample test or a
##           two-sample test.
##
##   stderr: the standard error of the mean (difference), used as
##           denominator in the t-statistic formula.
##
## alternative: a character string describing the alternative hypothesis.
##
##   method: a character string indicating what type of t-test was
##           performed.
##
## data.name: a character string giving the name(s) of the data.
```

```
##
## See Also:
##
##      'prop.test'
##
## Examples:
##
##      require(graphics)
##
##      t.test(1:10, y = c(7:20))      # P = .00001855
##      t.test(1:10, y = c(7:20, 200)) # P = .1245    -- NOT significant anymore
##
##      ## Classical example: Student's sleep data
##      plot(extra ~ group, data = sleep)
##      ## Traditional interface
##      with(sleep, t.test(extra[group == 1], extra[group == 2]))
##
##      ## Formula interface
##      t.test(extra ~ group, data = sleep)
##
##      ## Formula interface to one-sample test
##      t.test(extra ~ 1, data = sleep)
##
##      ## Formula interface to paired test
##      ## The sleep data are actually paired, so could have been in wide format:
##      sleep2 <- reshape(sleep, direction = "wide",
##                        idvar = "ID", timevar = "group")
##      t.test(Pair(extra.1, extra.2) ~ 1, data = sleep2)
```

0.32 *t*-test : Example 1

```
x = c(4, 3, 10, 7, 9) ; y = c(7, 4, 3, 8, 10)
t.test(x, y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = 0.1066, df = 7.9743, p-value = 0.9177
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.12888  4.52888
## sample estimates:
## mean of x mean of y
##       6.6       6.4
```

```
t.test(x, y)$p.value
```

```
## [1] 0.917739
```

0.33 *t*-test : Example 2

```
x = c(6, 8, 10, 7, 9) ; y = c(3, 2, 1, 4, 5)
t.test(x, y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = 5, df = 8, p-value = 0.001053
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.693996 7.306004
## sample estimates:
## mean of x mean of y
##         8         3
```

```
t.test(x, y)$p.value
```

```
## [1] 0.001052826
```

0.34 Statistical Significance (*p*-value) 2/3

Let's compute the *p*-value for all genes using a `for`-loop of `t.test`, one gene at a time:

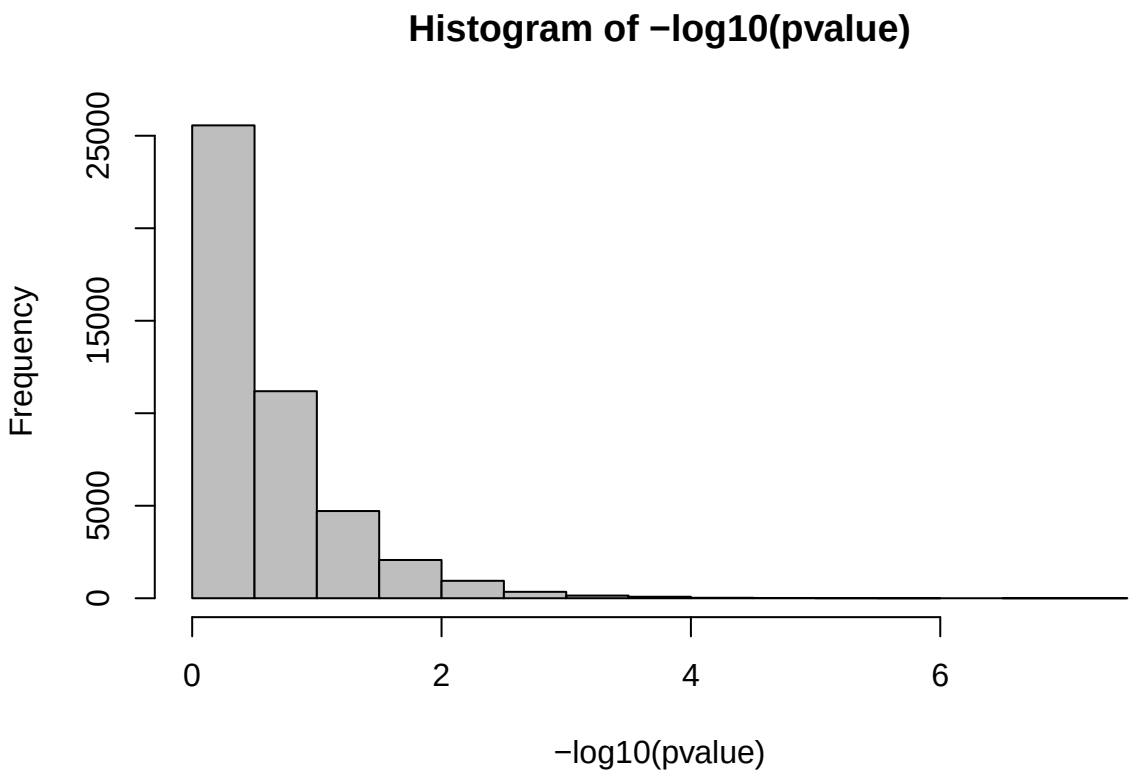
```
pvalue = NULL # Empty list for the p-values

for(i in 1 : number_of_genes) { # for each gene from to the number of genes
  x = wt[i, ] # wt values of gene number i
  y = ko[i, ] # ko values of gene number i
  t = t.test(x, y) # t-test between the two conditions
  pvalue[i] = t$p.value # Store p-value number i into the list of p-values
}
head(pvalue)
```

```
## [1] 0.092706280 0.182663337 0.129779075 0.272899180 0.262377176 0.005947807
```

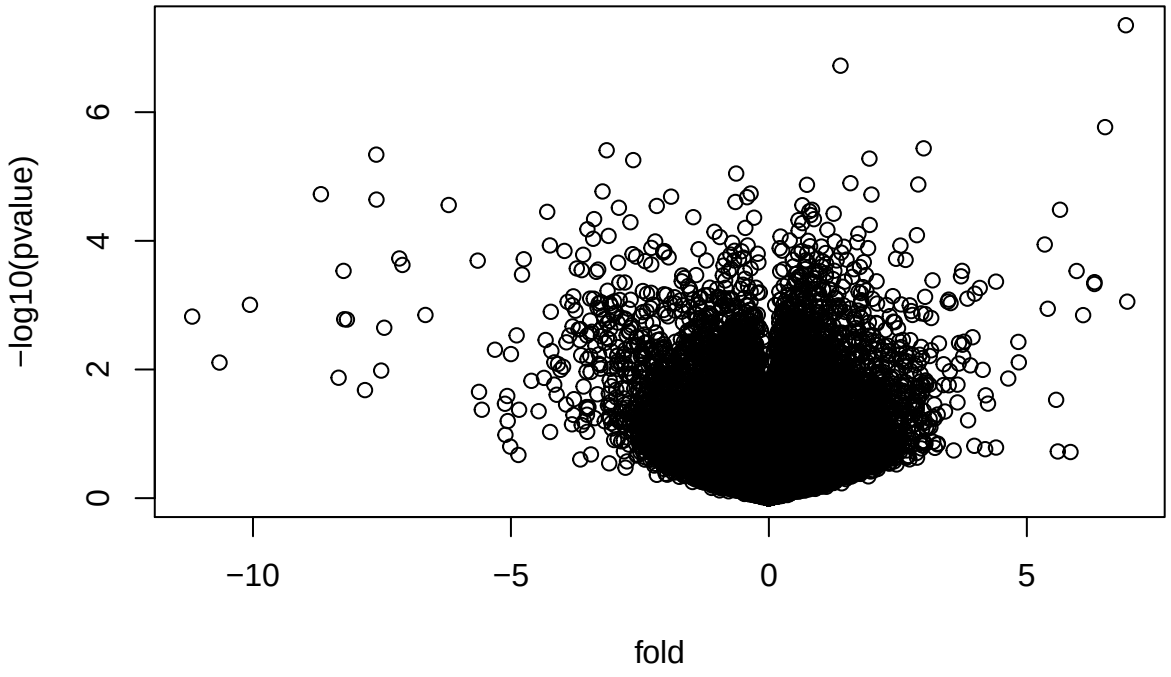
0.35 Statistical Significance (*p*-value) 3/3

```
hist(-log10(pvalue), col = "gray") # Histogram of p-values (-log10)
```



0.36 Volcano : Statistical & Biological 1/3

```
plot(-log10(pvalue) ~ fold)
```



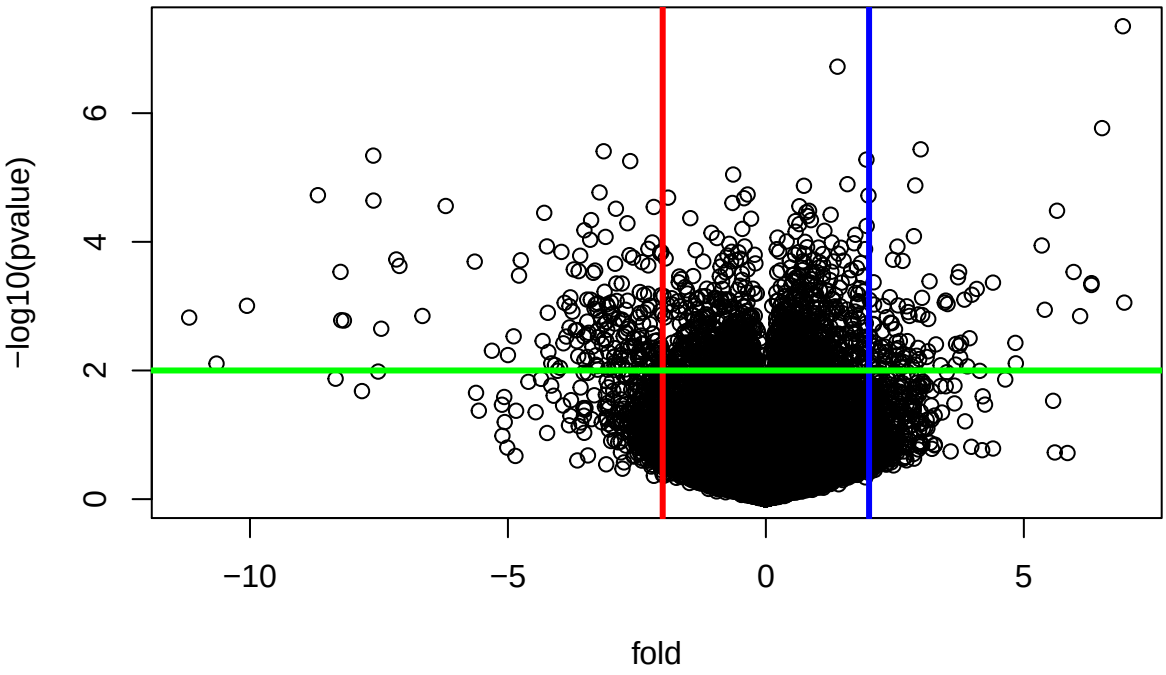
0.37 Volcano : Statistical & Biological 2/3

```
fold_cutoff = 2
pvalue_cutoff = 0.01

plot(-log10(pvalue) ~ fold)

abline(v = fold_cutoff, col = "blue", lwd = 3)
abline(v = -fold_cutoff, col = "red", lwd = 3)
abline(h = -log10(pvalue_cutoff), col = "green", lwd = 3)
```

0.38 Volcano : Statistical & Biological 3/3



0.39 Filtering for DEGs 1/3

```
filter_by_fold = abs(fold) >= fold_cutoff # Biological
sum(filter_by_fold) # Number of genes staisfy the condition
```

```
## [1] 1051
```

```
filter_by_pvalue = pvalue <= pvalue_cutoff # Statistical
sum(filter_by_pvalue)
```

```
## [1] 1564
```

```
filter_combined = filter_by_fold & filter_by_pvalue # Combined
sum(filter_combined)
```

```
## [1] 276
```

0.40 Filtering for DEGs 2/3

```
filtered = data2[filter_combined, ]
dim(filtered)
```

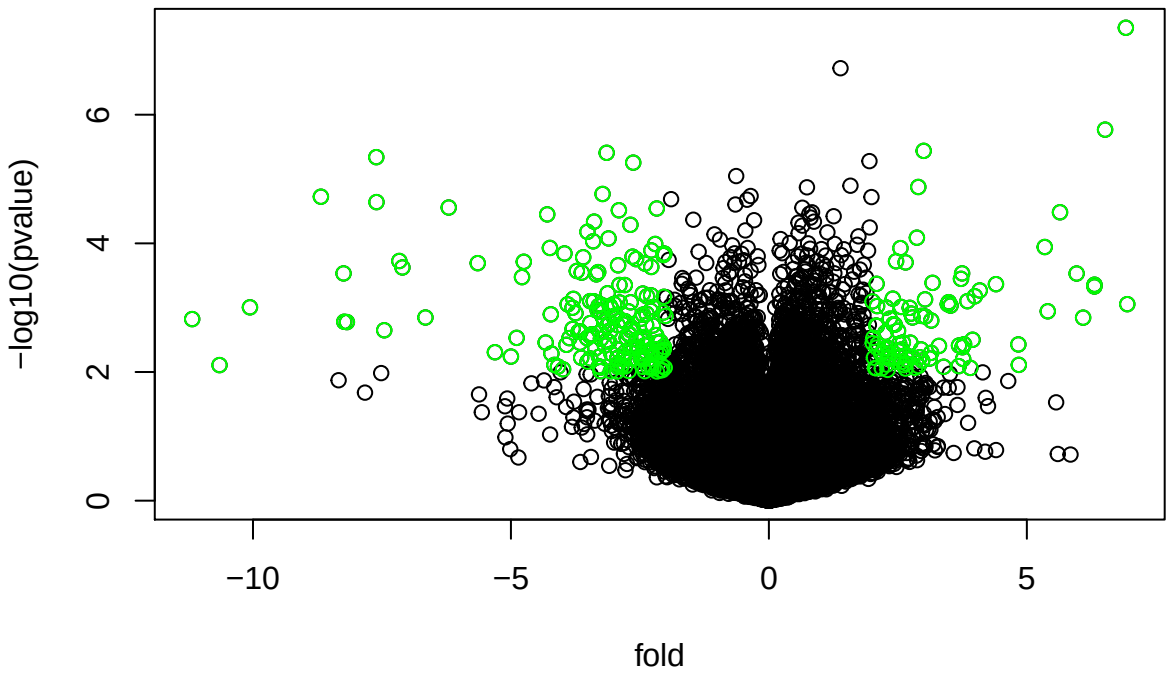
```
## [1] 276   6
```

```
head(filtered)
```

	KO1	KO2	KO3	WT1	WT2	WT3
1416200_at	13.312004	12.973357	12.868456	7.40429	8.558803	8.683696
1416236_a_at	14.148397	14.039236	14.130007	12.23604	12.022403	11.495055
1417808_at	5.321928	5.442944	4.053111	15.16978	15.070087	14.753274
1417932_at	10.602884	10.257152	10.496055	13.98445	14.203295	13.720960
1418050_at	10.622052	10.975490	10.795066	12.86513	13.012048	12.658122
1418100_at	9.117903	8.634811	9.057721	12.90358	12.842449	12.233769

0.41 Filtering for DEGs 3/3

```
plot(-log10(pvalue) ~ fold)
points(-log10(pvalue[filter_combined]) ~ fold[filter_combined],
       col = "green")
```



0.42 Exercise

On the volcano plot, highlight the up-regulated genes in red and the download-regulated genes in blue

0.43 Solution 1/2

- Up-regulated genes

```
# Screen for the up-regulated genes (+ve fold)
filter_up = filter_combined & fold > 0

head(filter_up)
```

```
##   1415670_at   1415671_at   1415672_at   1415673_at 1415674_a_at   1415675_at
##        FALSE        FALSE        FALSE        FALSE        FALSE        FALSE
```

```
# Number of filtered genes
sum(filter_up)
```

```
## [1] 95
```

- Down-regulated genes

```
# Screen for the down-regulated genes (-ve fold)
filter_down = filter_combined & fold < 0

head(filter_down)
```

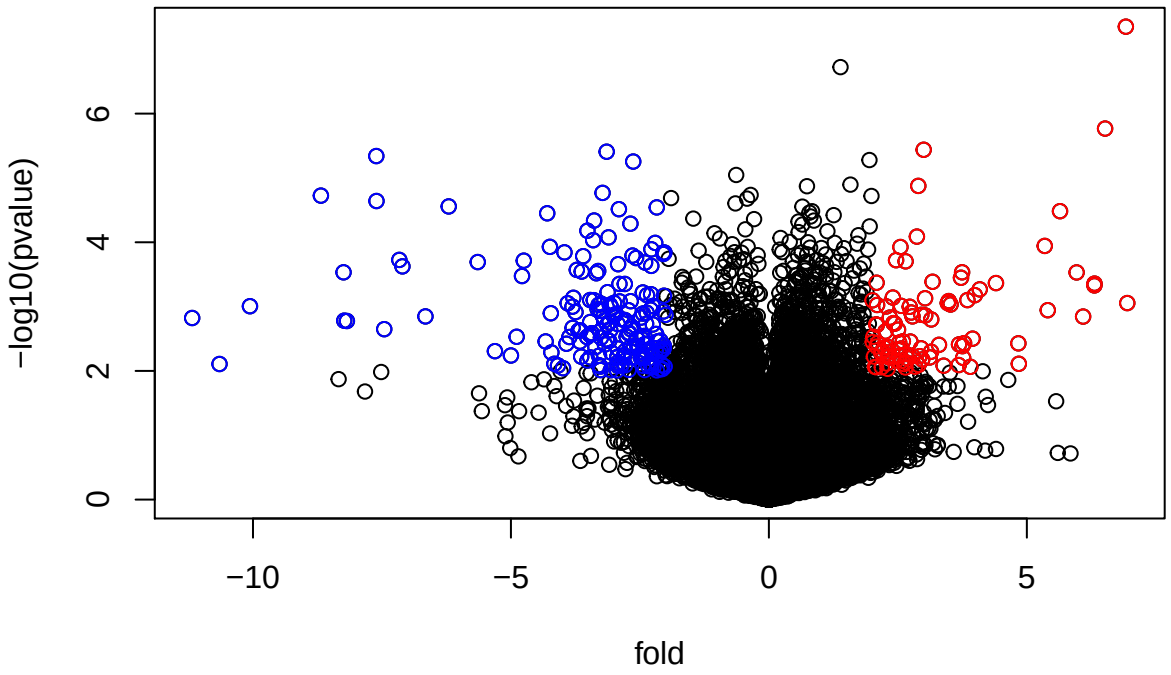
```
##   1415670_at   1415671_at   1415672_at   1415673_at 1415674_a_at   1415675_at
##        FALSE        FALSE        FALSE        FALSE        FALSE        FALSE
```

```
# Number of filtered genes
sum(filter_down)
```

```
## [1] 181
```

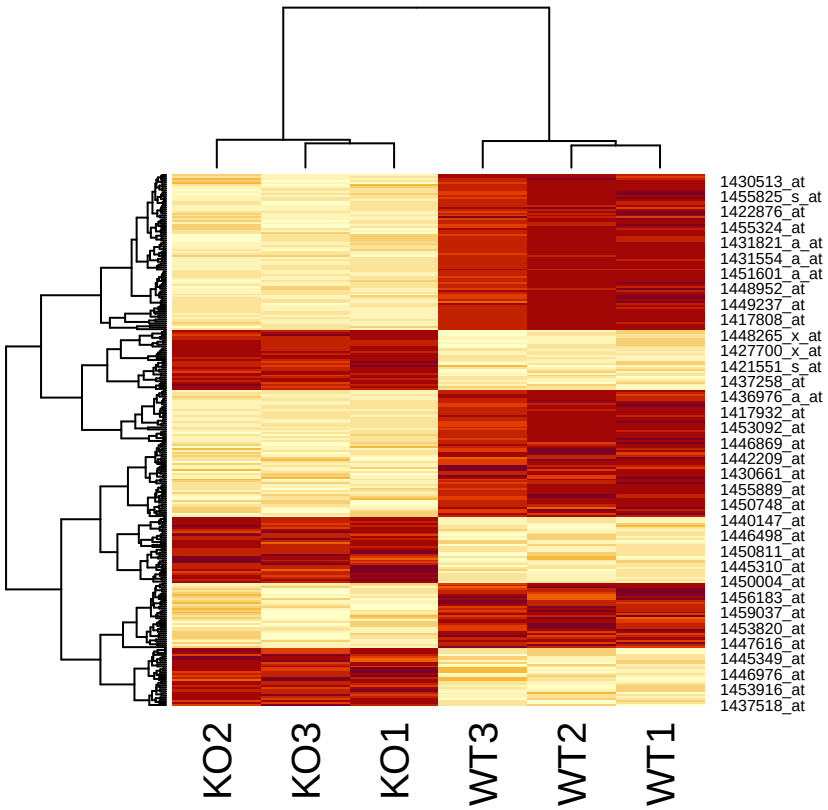
0.44 Solution 2/2

```
plot(-log10(pvalue) ~ fold)
points(-log10(pvalue[filter_up]) ~ fold[filter_up], col = "red")
points(-log10(pvalue[filter_down]) ~ fold[filter_down], col = "blue")
```



0.45 Heatmap 1/5

```
heatmap(filtered)
```



0.46 Heatmap 2/5

- By default, `heatmap` clusters genes (rows) and samples (columns) based on the Euclidean distance.
- In the context of gene expression, we need to cluster genes and samples based on the correlation to explore patterns of **co-regulation** (**co-expression**) - *Guilt by Association*.
- To let `heatmap` cluster the genes and/or samples, the genes and samples will be clustered (grouped) by correlation coefficients (using `cor`) among the genes and samples.

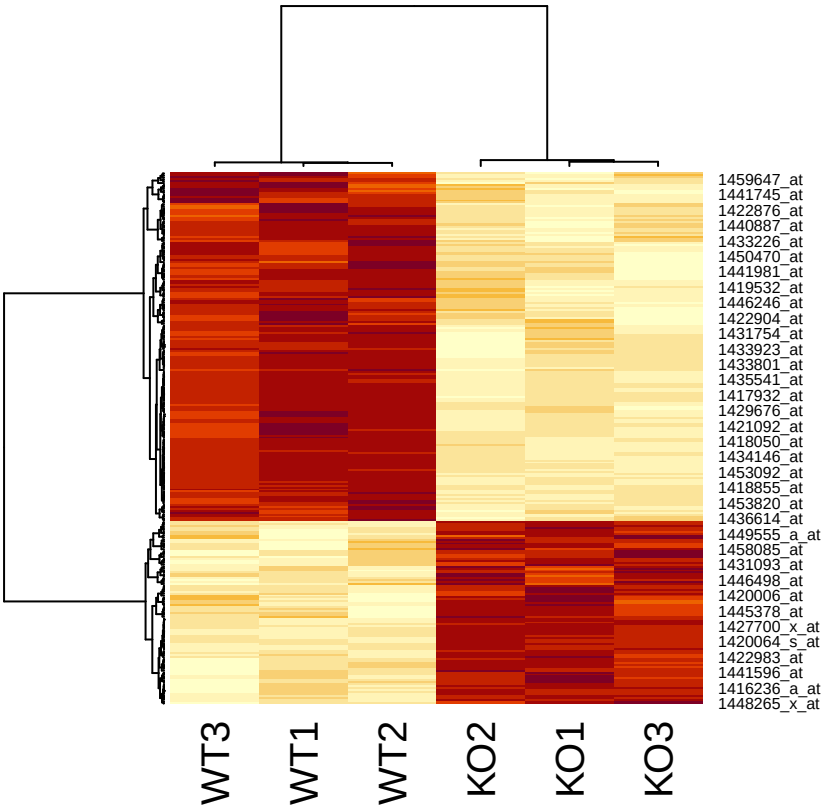
0.47 Heatmap 3/5

```
# Clustering of the columns (samples)
col_dendrogram = as.dendrogram(hclust(as.dist(1-cor(filtered))))

# Clustering of the rows (genes)
row_dendrogram = as.dendrogram(hclust(as.dist(1-cor(t(filtered)))))
```

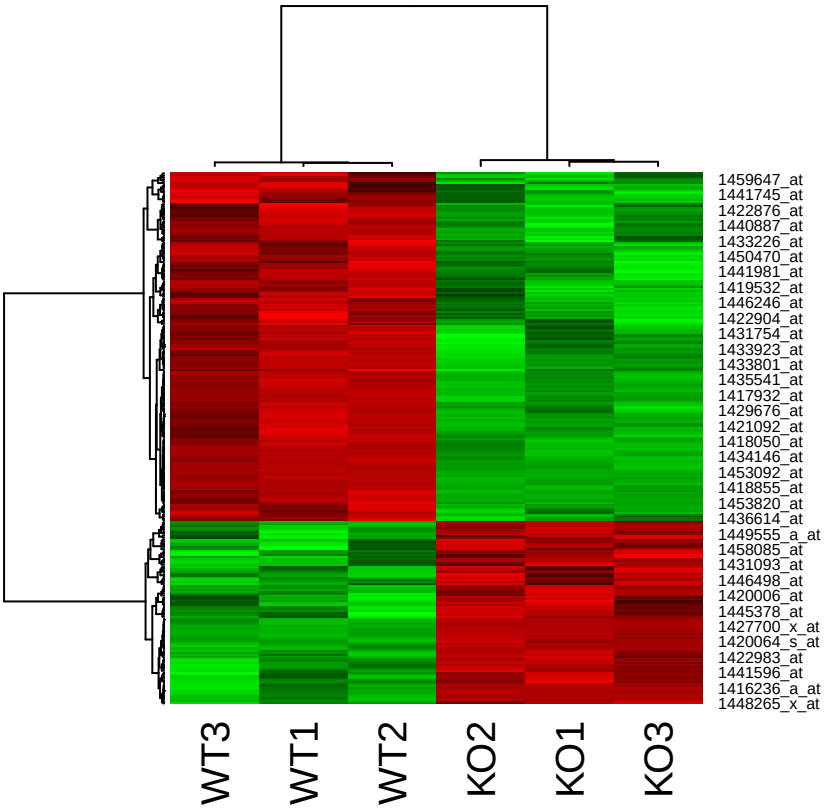
0.48 Heatmap 4/5

```
# Heatmap with the rows and columns clustered by correlation coefficients
heatmap(filtered, Rowv=row_dendrogram, Colv=col_dendrogram)
```



0.49 Heatmap 5/5

```
library(gplots) # Load the gplots library
heatmap(filtered, Rowv=row_dendrogram, Colv=col_dendrogram, col = rev(redgreen(1024)))
```



0.50 Annnotation 1/3

To obtain the functional annotation of the differentially expressed genes, we are going first to extract their probe ids:

```
filterd_ids = row.names(filtered) # ids of the filtered DE genes
length(filterd_ids)
```

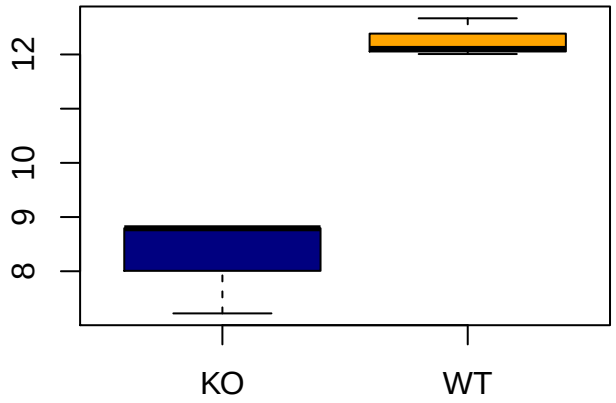
```
## [1] 276
```

```
head(filterd_ids)
```

```
## [1] "1416200_at"   "1416236_a_at" "1417808_at"   "1417932_at"   "1418050_at"
## [6] "1418100_at"
```

Figure 4: Down Regulation of Irf6

0.51 Sanity Check (Irf6)



0.52 Multiple Testing Correction 1/3

We conducted 10^6 statistical tests. The computed p-values need to be corrected for *multiple testing*. The correction can be performed using `p.adjust`, which simply takes the orignial p-values a vector and returns the adjusted (corrected) p-values:

```
adjusted.pvalues = p.adjust(pvalue, method = "fdr")
```

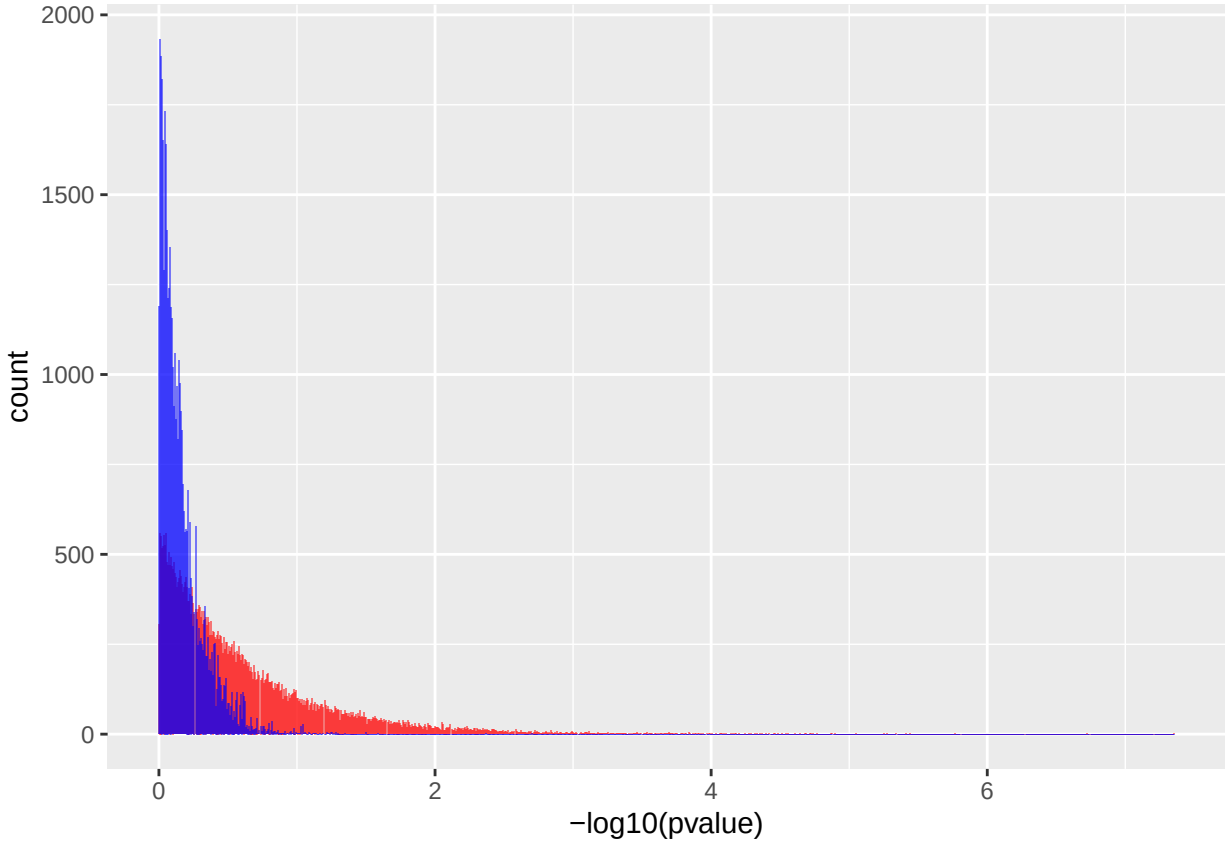
Number **original** p-values $\leq 0.05 = 5099$ while the number **adjusted** (**corrected**) p-values $< 0.05 \geq 9$

0.53 Multiple Testing Correction 2/3

Here is an example of the original p-values and corresponding adjusted p-values:

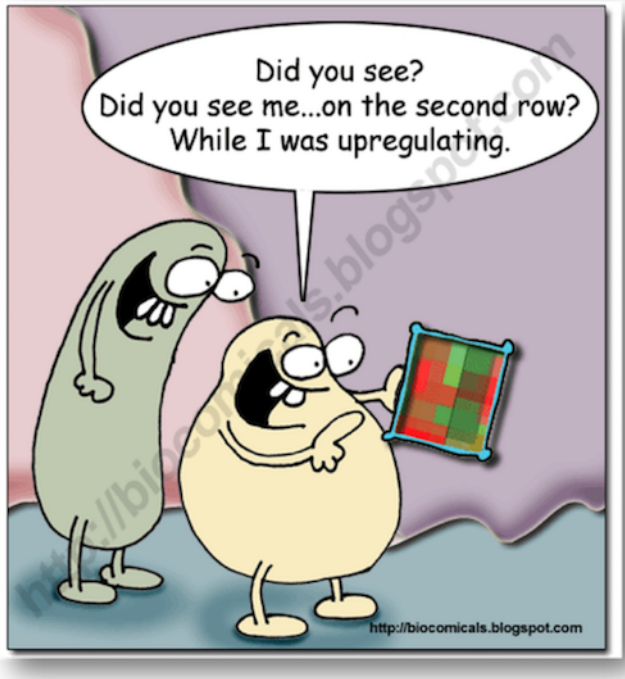
pvalue	adjusted.pvalue
0.0927063	0.5278755
0.1826633	0.6346918
0.1297791	0.5805456
0.2728992	0.7025472
0.2623772	0.6967834
0.0059478	0.2518079

0.54 Multiple Testing Correction 3/3



0.55 Homework

- Identify the top 10 *biologically* significant genes (i.e., by fold-change)
- Identify the top 10 *statistically* significant genes (i.e., by p-value)


Did you see?
Did you see me...on the second row?
While I was upregulating.
http://biocomicals.blogspot.com