Microarray Gene Expression Analysis with R

Example: Interferon Regulatory Factor 6 (*IRF6*)

Ahmed Moustafa

Feb 11, 2022

Objectives

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

The Central Dogma of Biology

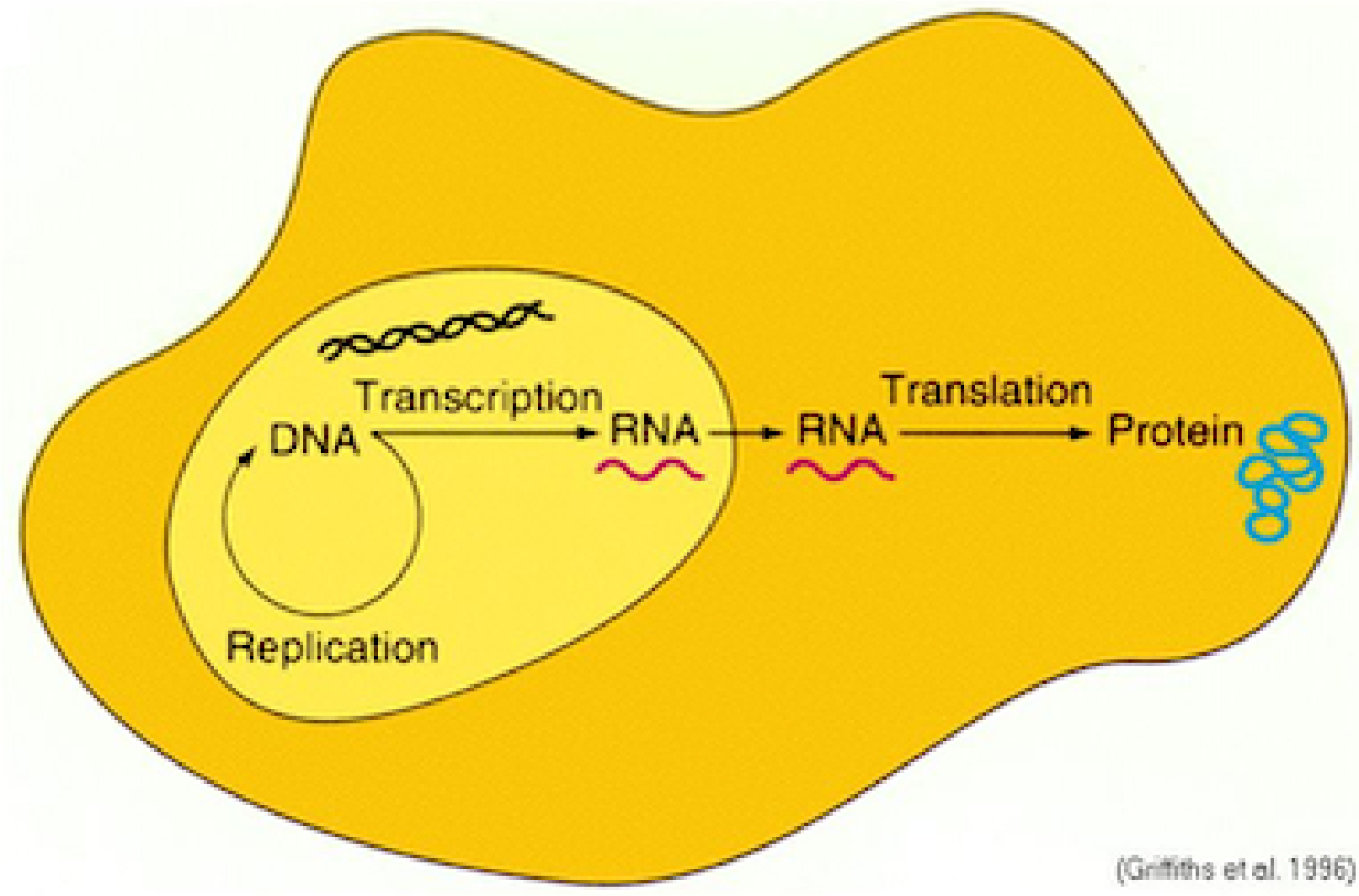
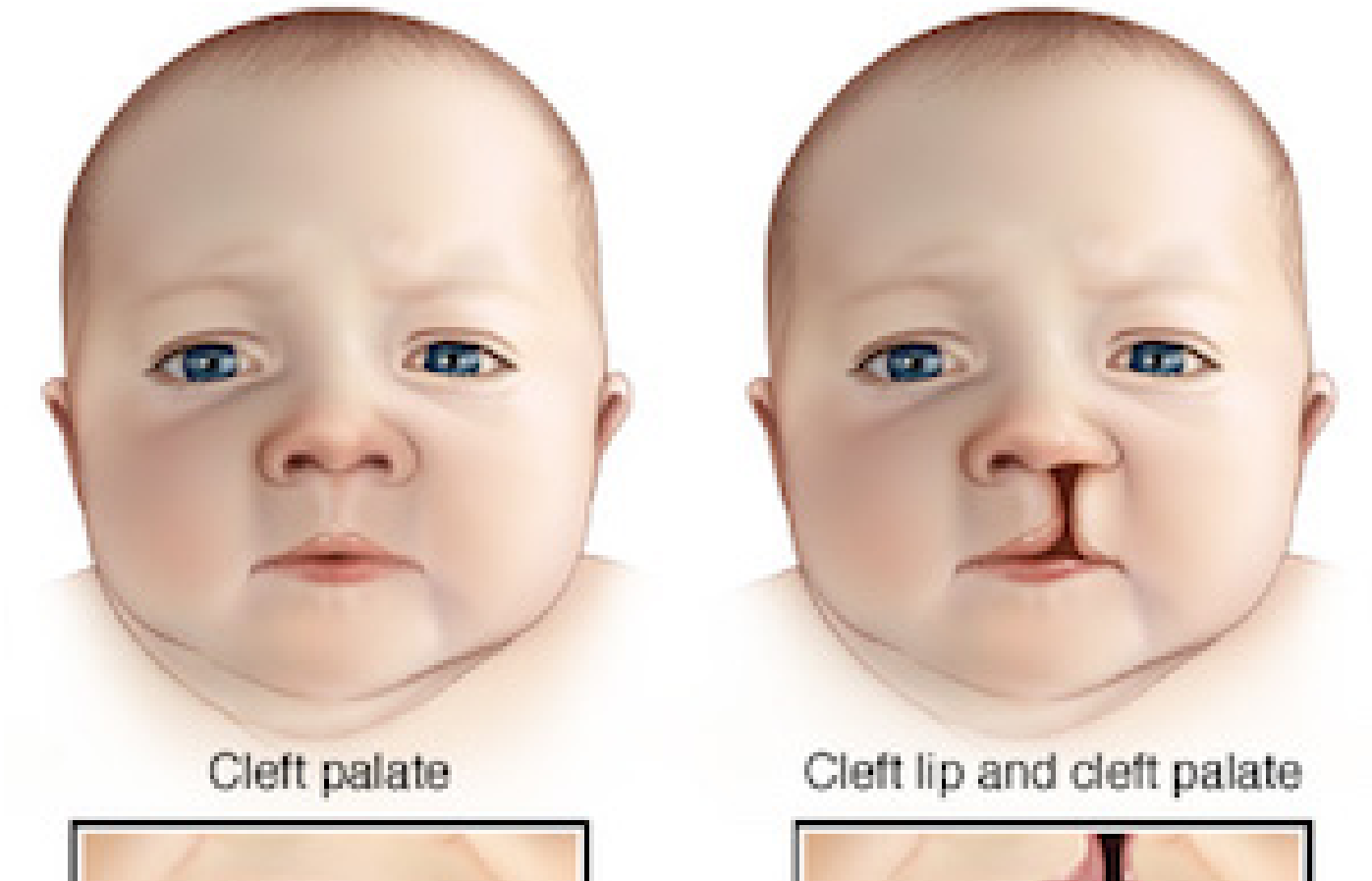


Figure 1: DNA makes RNA and RNA makes protein

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.



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

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

Question

Given:

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

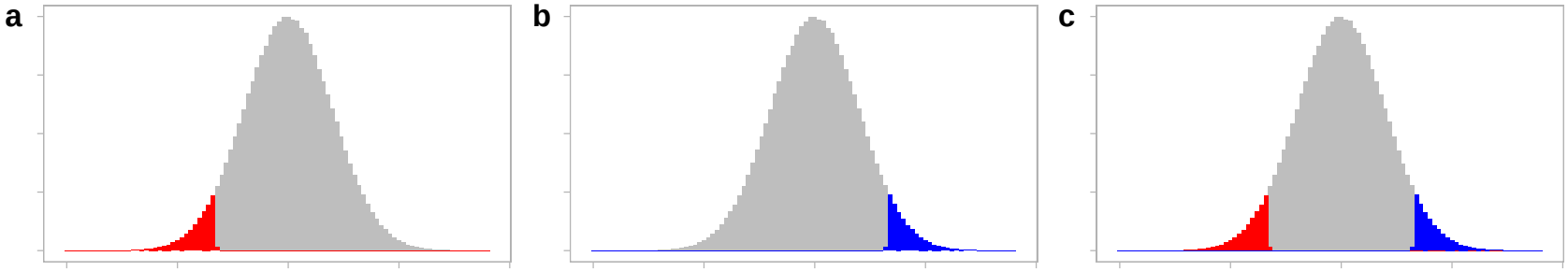
How can we identify other genes that might be involved in the remaining 30% of the VWS families?

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*.

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?



Why Microarray?

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)

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

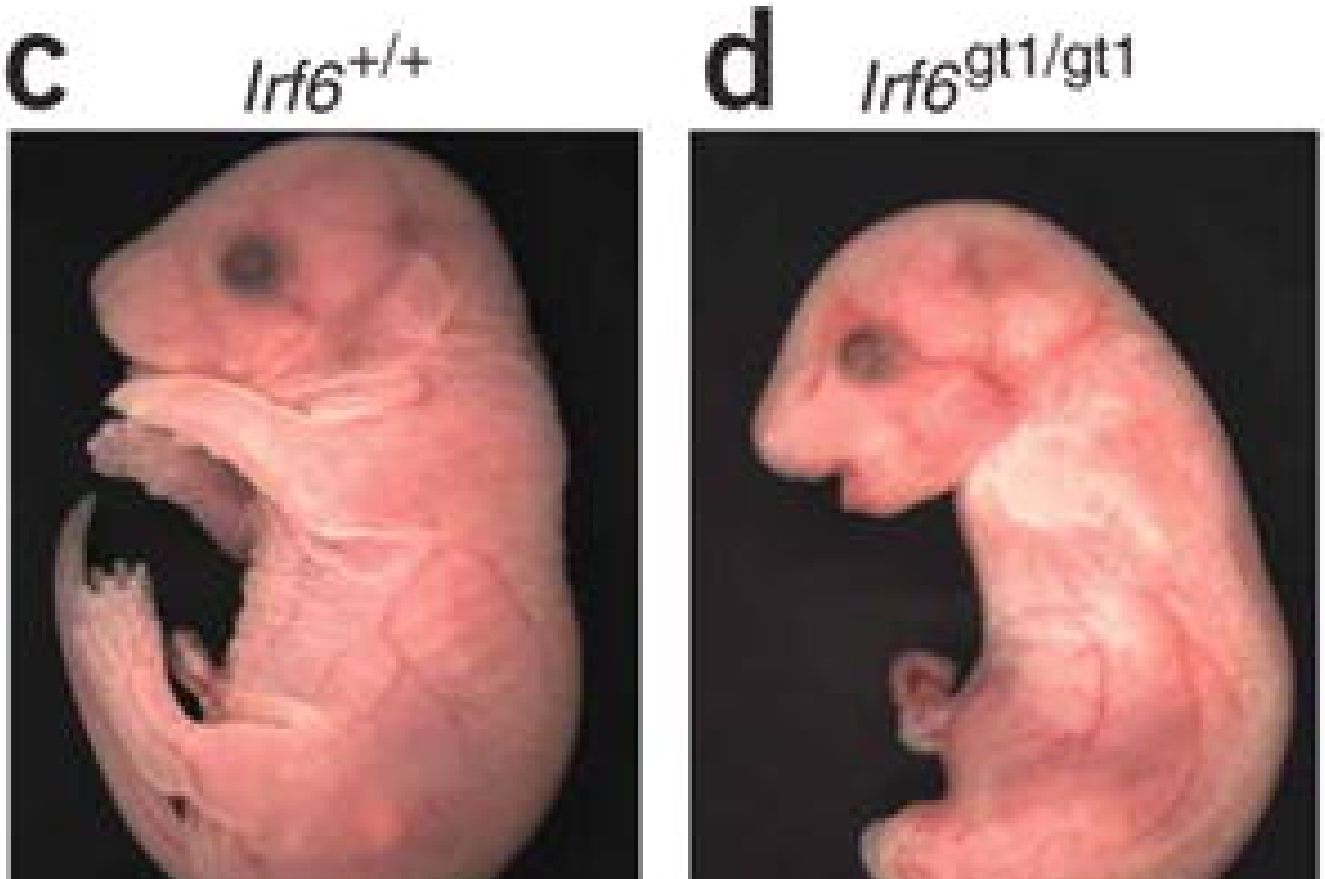
Transcription factor paralogs may share a common role in staged or overlapping expression in specific tissues, as in the Hox family. In other cases, family members have distinct roles in a range of embryologic, differentiation or response pathways (as in the Tbx and Pax families). For the interferon 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

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.



Dataset

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

ID	KO1	KO2	KO3	WT1	WT2	WT
1415670_at	6531.0	5562.8	6822.4	7732.1	7191.2	7551.
1415671_at	11486.3	10542.7	10641.4	10408.2	9484.5	7650.
1415672_at	14339.2	13526.1	14444.7	12936.6	13841.7	13285.
1415673_at	3156.8	2219.5	3264.4	2374.2	2201.8	2525.

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.

Note: `.tsv` stands for tab-separated values, which is simply a plain text file. The file itself can downloaded from here.

In the following code, `data` is just an arbitrary **variable** name to hold the result of `read.table`. See here for **valid** variable names in **R**. Also it is recommended to see here for generally Good Variable Names.

Loading

```
# Load the data from a file into a variable
data = read.table("https://media.githubusercontent.com/medi

# 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 code and they are not evaluated (or executed).

Checking

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

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

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

	KO1	KO2	KO3	WT1	WT2	W
1415670_at	6531.0	5562.8	6822.4	7732.1	7191.2	75
1415671_at	11486.3	10542.7	10641.4	10408.2	9484.5	76
1415672_at	14339.2	13526.1	14444.7	12936.6	13841.7	132
1415673_at	3156.8	2219.5	3264.4	2374.2	2201.8	25
1415674_a_at	4002.0	3306.9	3777.0	3760.6	3137.0	29
1415675_at	3468.4	3347.4	3332.9	3073.5	3046.0	29

Number of Genes and IDs

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

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

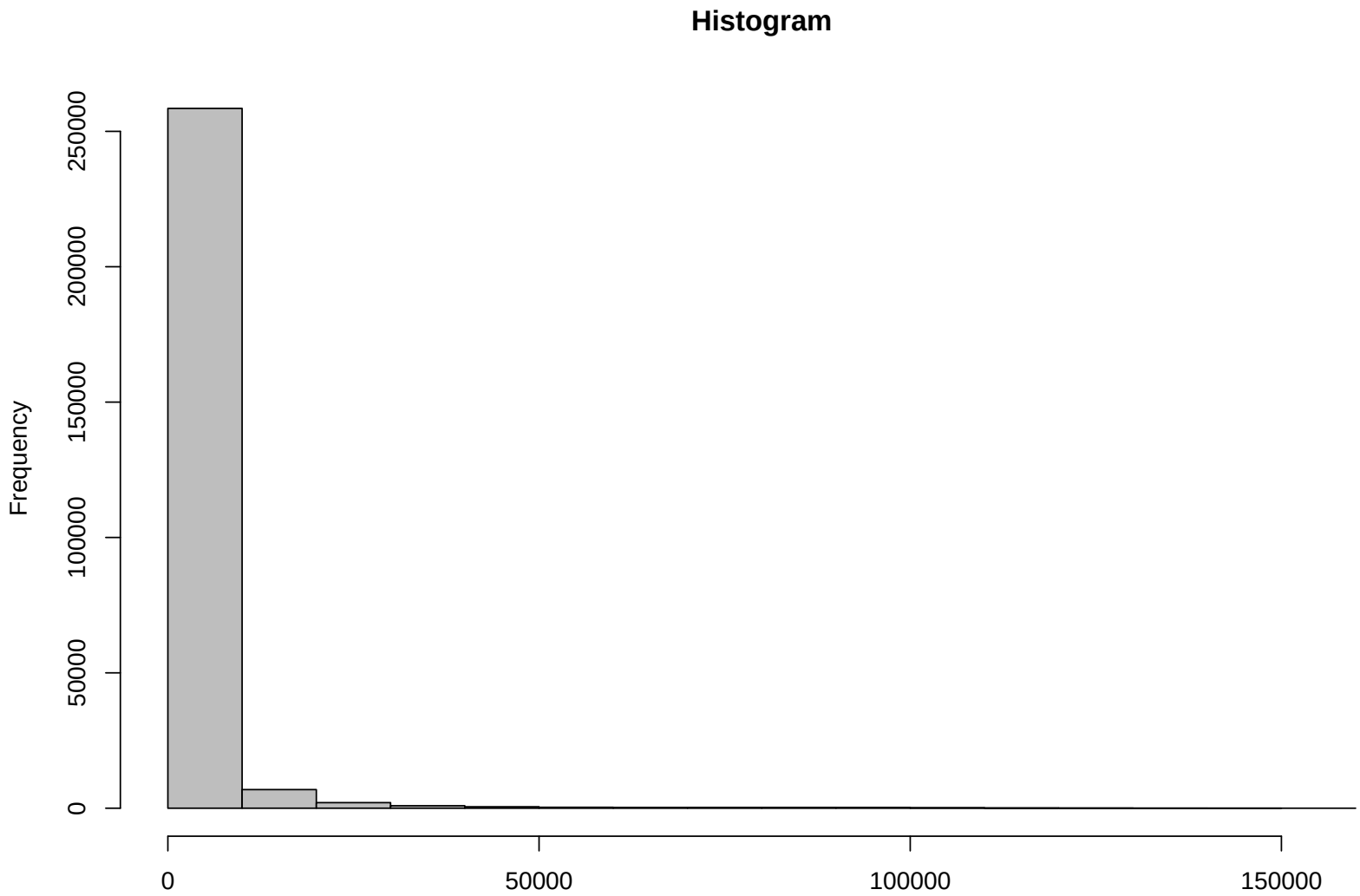
```
ids = row.names(data) # The ids of the genes are the names
head(ids)
```

```
## [1] "1415670_at"   "1415671_at"   "1415672_at"   "141567
## [6] "1415675_at"
```

Exploring

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

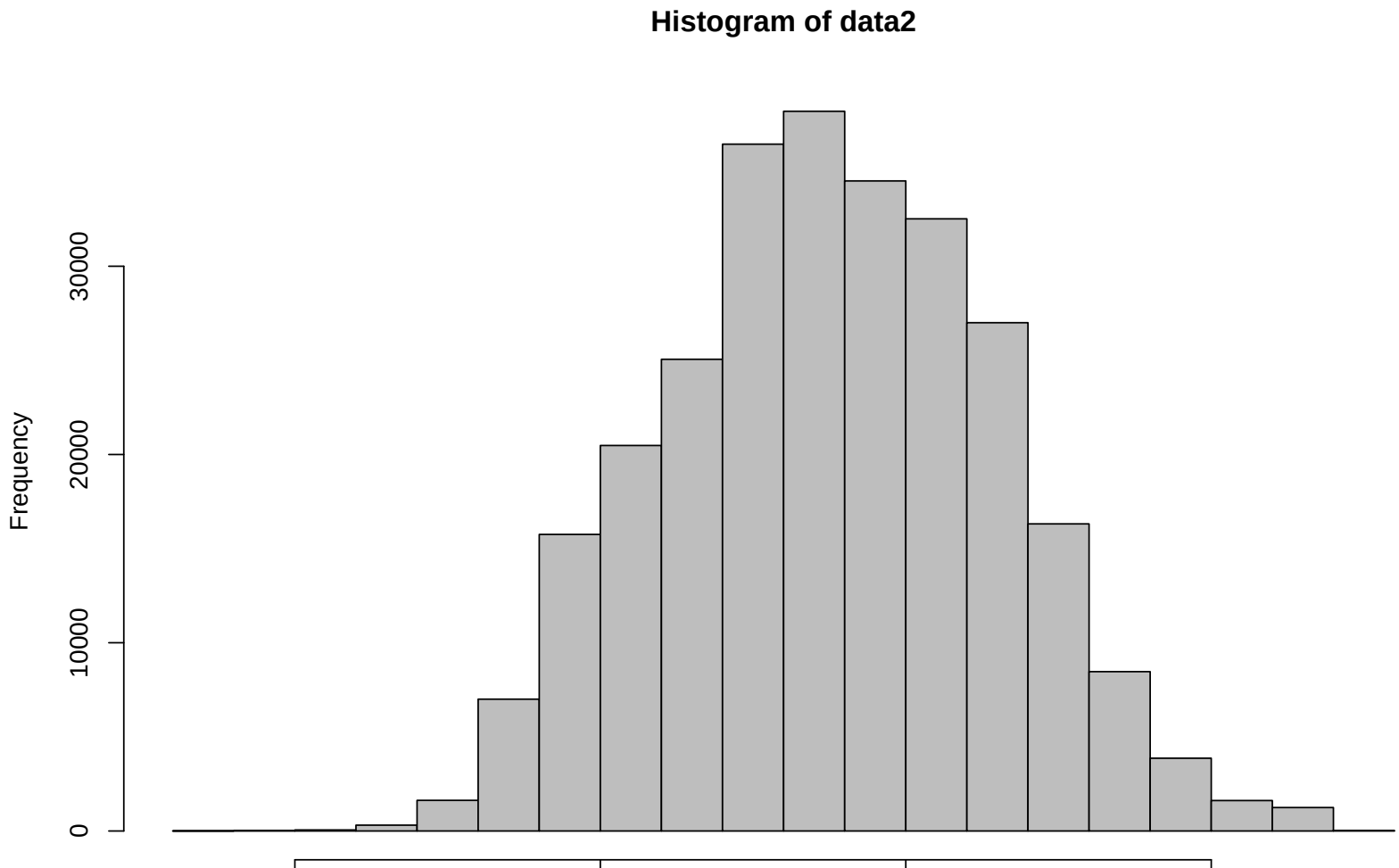
```
hist(data, col = "gray", main="Histogram")
```



Transforming

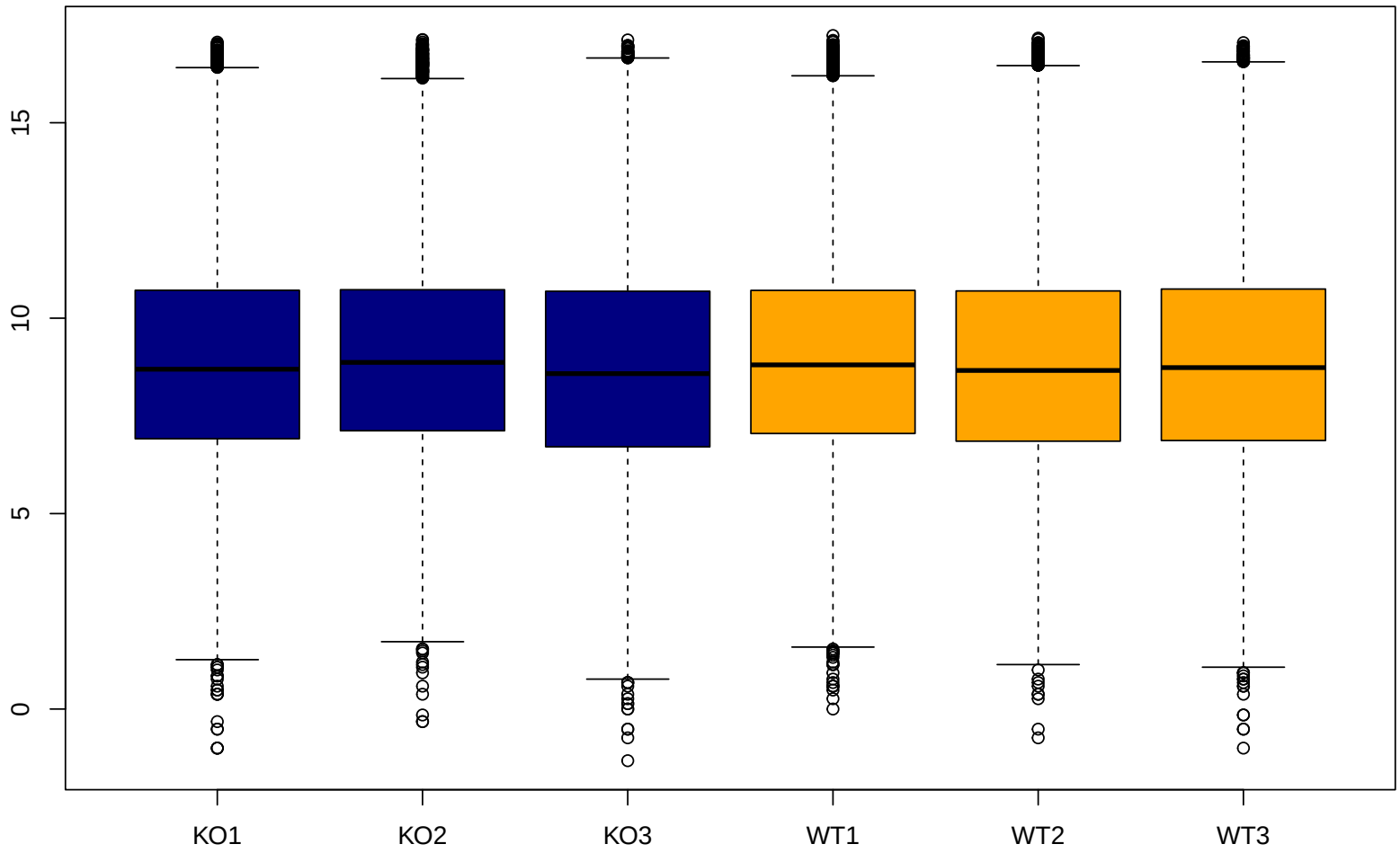
log_2 transformation (why?)

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



Boxplot

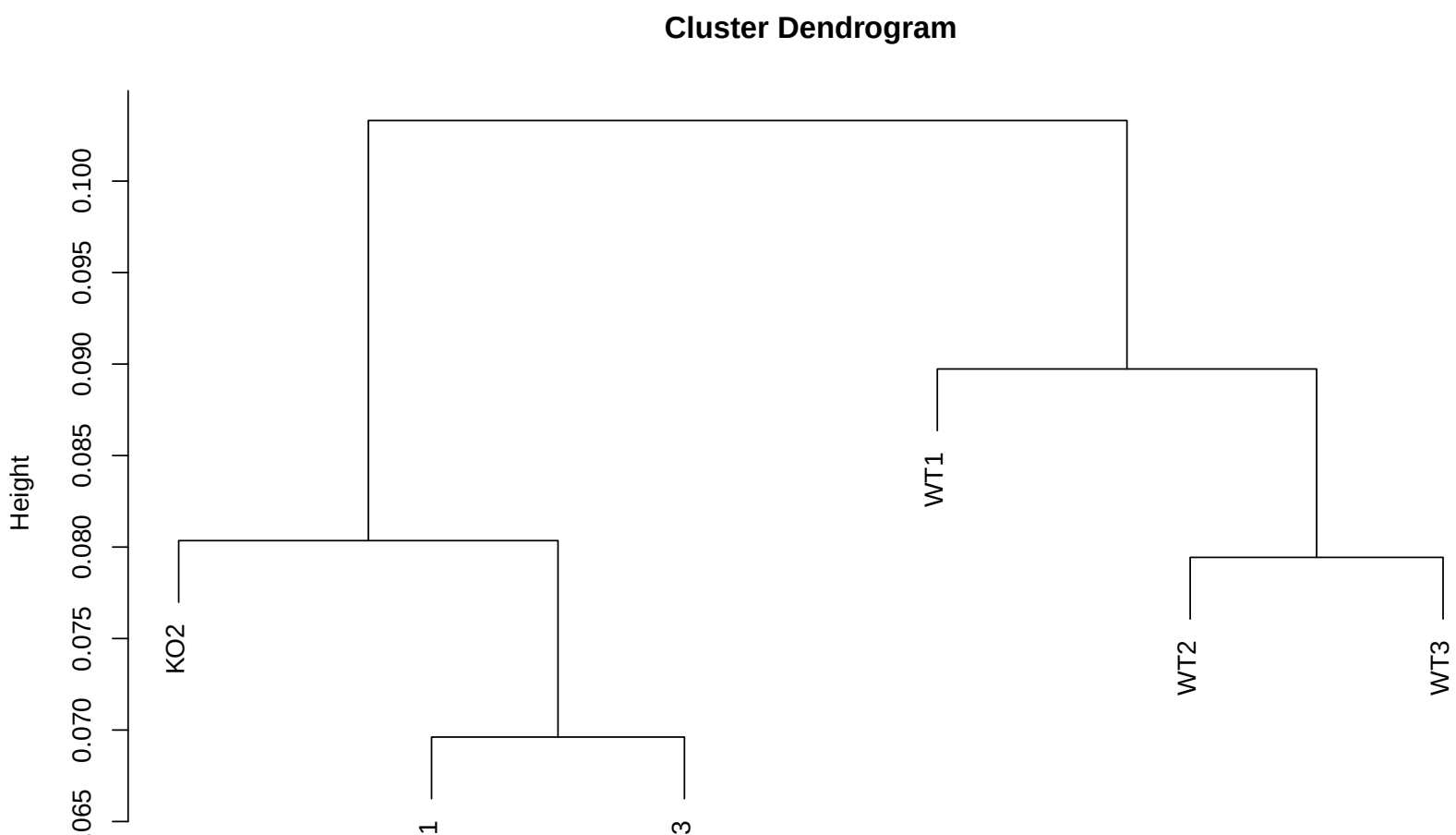
```
colors = c(rep("navy", 3), rep("orange", 3))
boxplot(data2, col = colors)
```



Clustering

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)
```



Splitting Data Matrix into two tables

KO

```
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

Splitting Data Matrix into two tables

WT

```
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

Gene (Row) Mean Expression

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

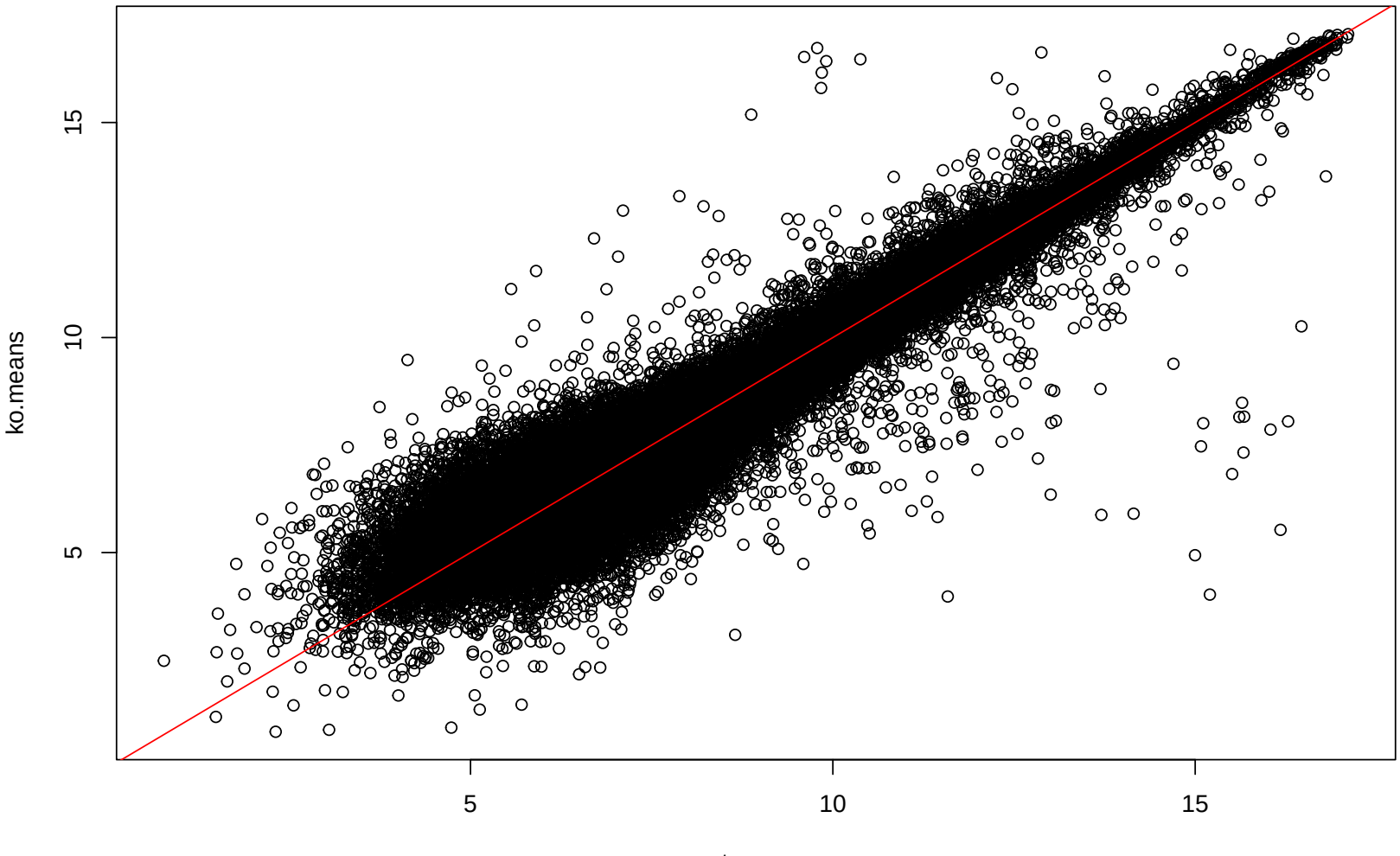
```
##   1415670_at   1415671_at   1415672_at   1415673_at 1415
##     12.61692     13.40966     13.78313     11.47096
```

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

```
##   1415670_at   1415671_at   1415672_at   1415673_at 1415
##     12.87043     13.15269     13.70450     11.20664
```

Scatter between the means

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



Differential Gene Expression Analysis

To identify Differentially Expressed Genes (DEGs), we will identify:

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

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

Biological Significance (fold-change)

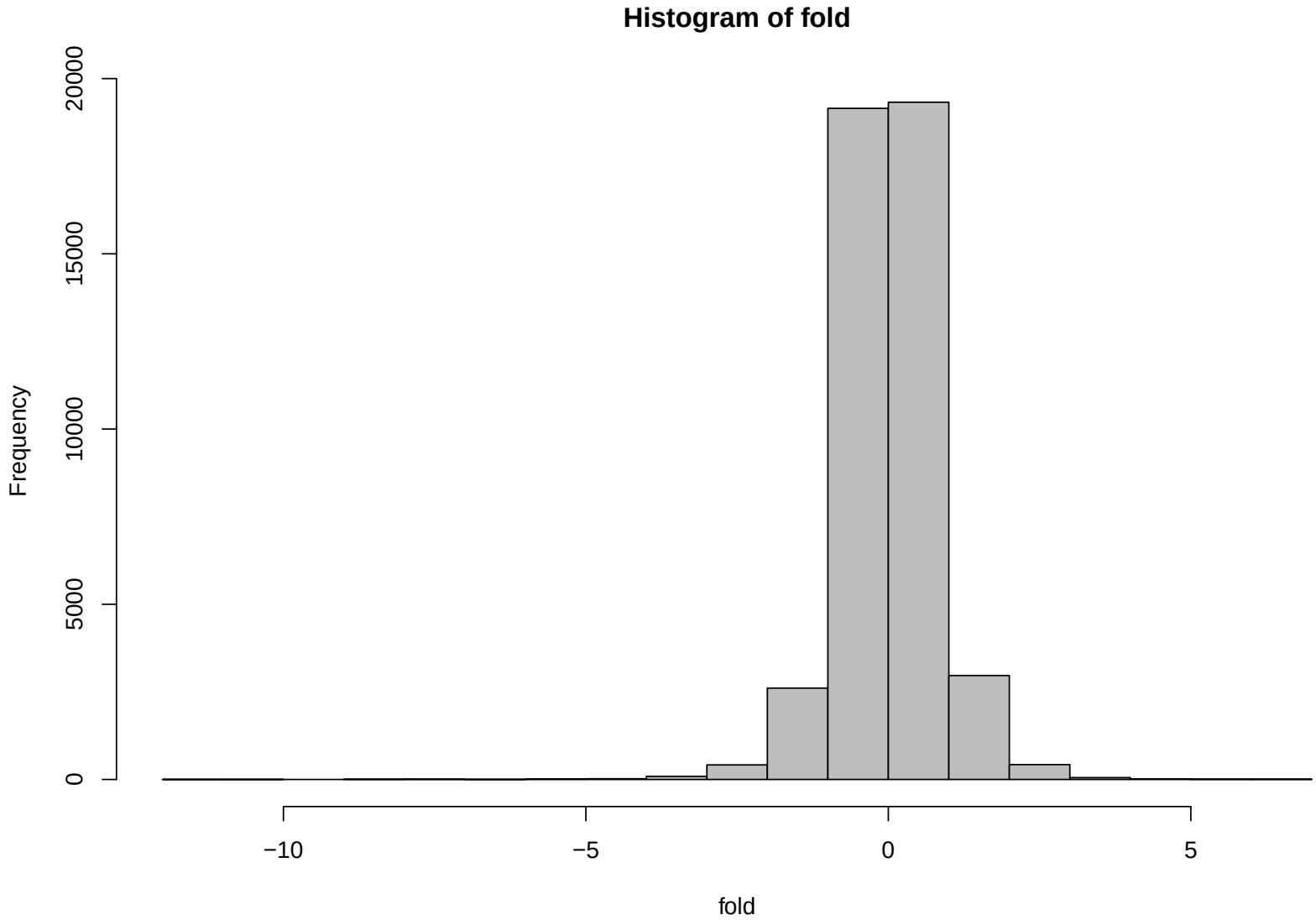
```
fold = ko.means - wt.means # Difference between means
head(fold)
```

```
##   1415670_at   1415671_at   1415672_at   1415673_at 1415
##  -0.25351267   0.25697097   0.07863227   0.26431191   0.
```

- What do the positive and negative values of the fold-change indicate? Considering the `WT` condition is the **reference** (or **control**)
- **+ve** fold-change $\rightarrow$ **Up**-regulation $\uparrow$
- **-ve** fold-change $\rightarrow$ **Down**-regulation $\downarrow$

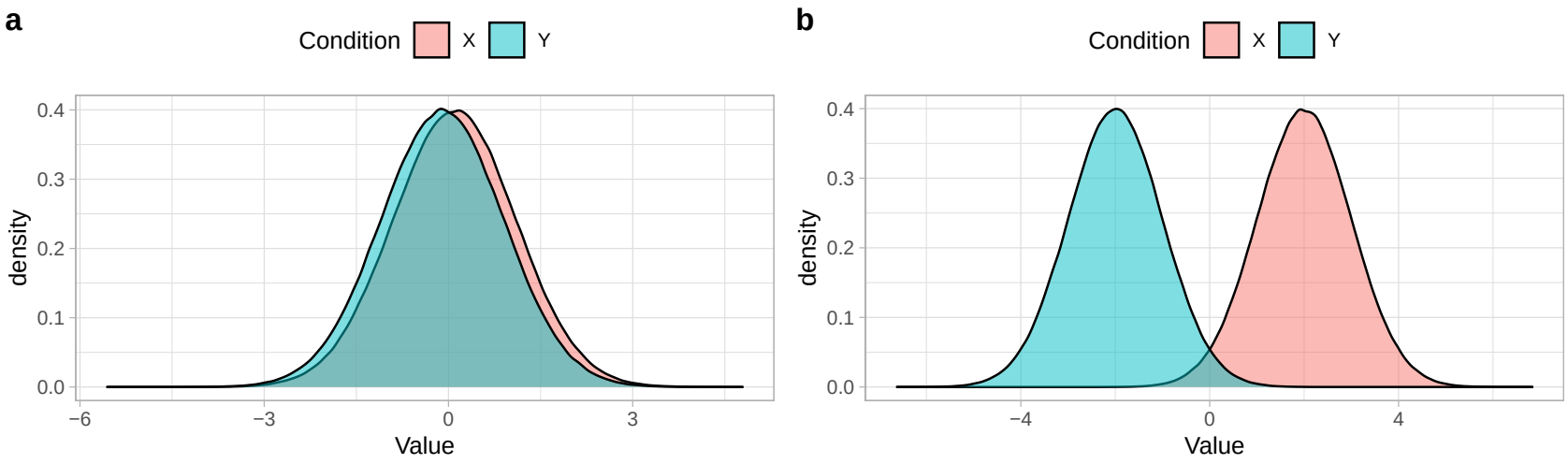
Biological Significance (fold-change)

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



Statistical Significance (p-value)

- 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.



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 : 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
## 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
```

t-test : Example 2

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

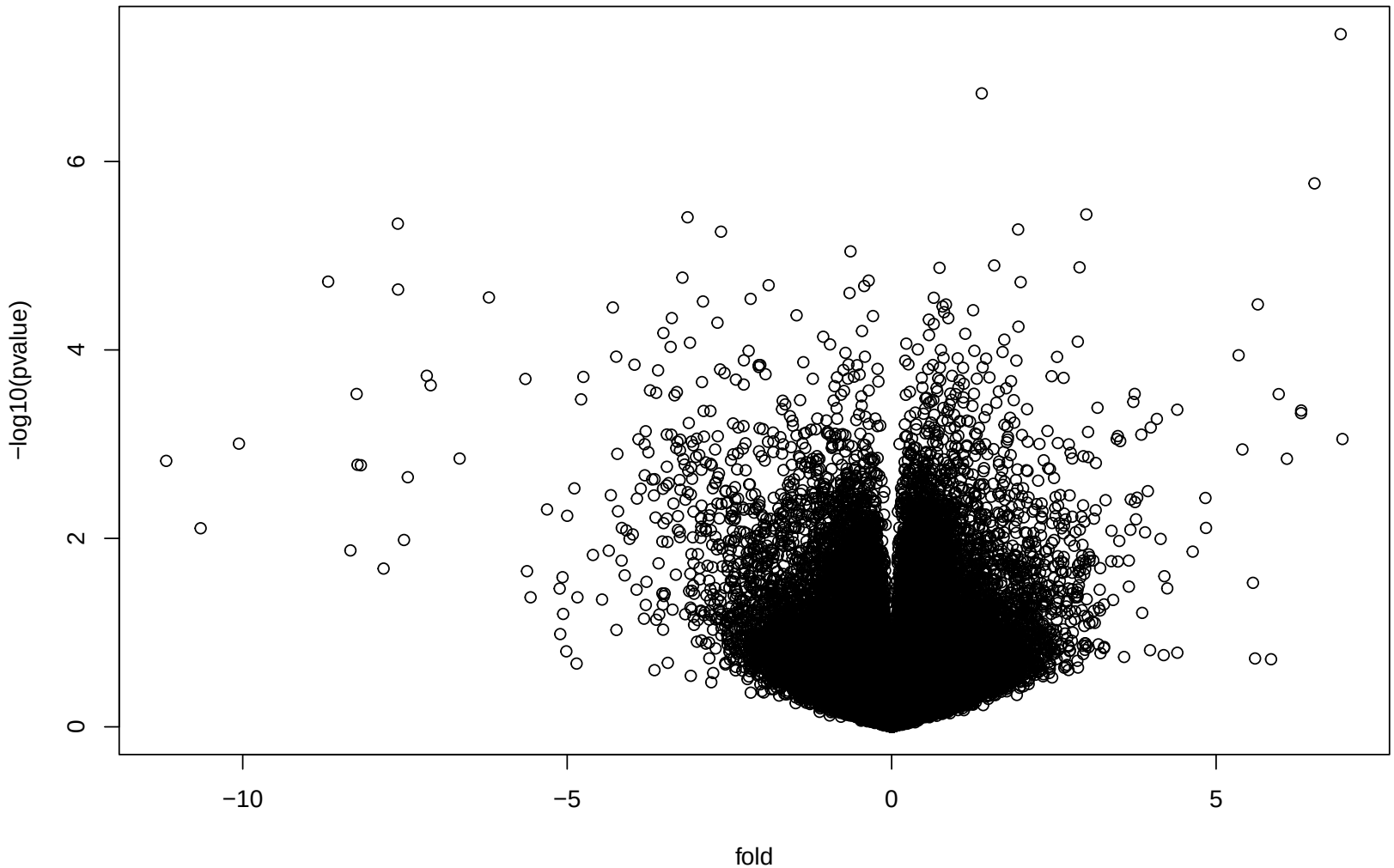
```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = 5, df = 8, p-value = 0.001053
## alternative hypothesis: true difference in means is not
## 95 percent confidence interval:
##  2.693996 7.306004
## sample estimates:
## mean of x mean of y
##         8         3
```

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

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

Volcano

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



Filtering for DEGs

```
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 # Comb
sum(filter_combined)
```

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

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

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

Exercise

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

Click here for the solution

- 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 1415
##        FALSE        FALSE        FALSE        FALSE
```

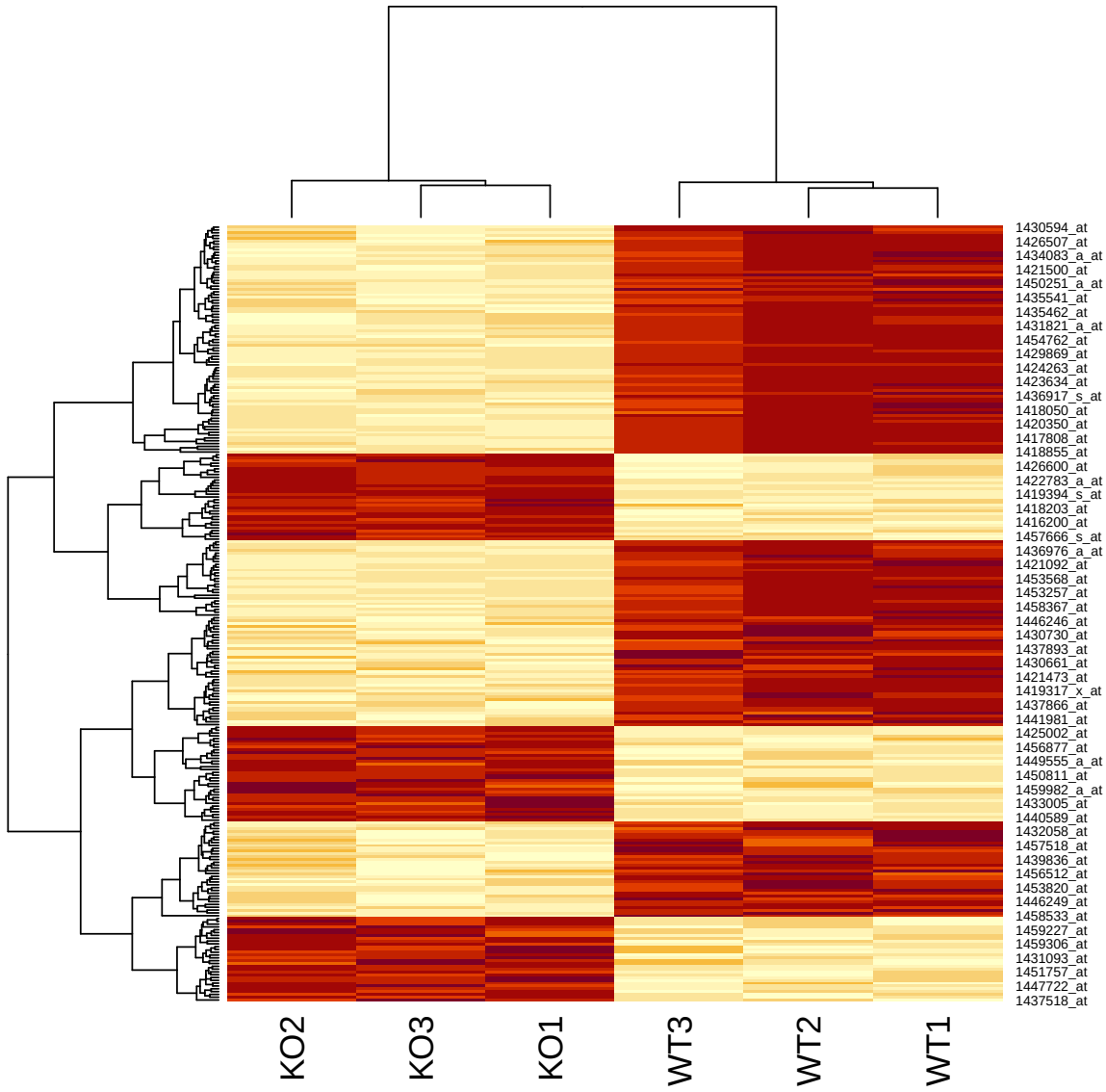
```
# Number of filtered genes
sum(filter_up)
```

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

- Down-regulated

Heatmap

```
heatmap(filtered)
```



Heatmap

- 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.

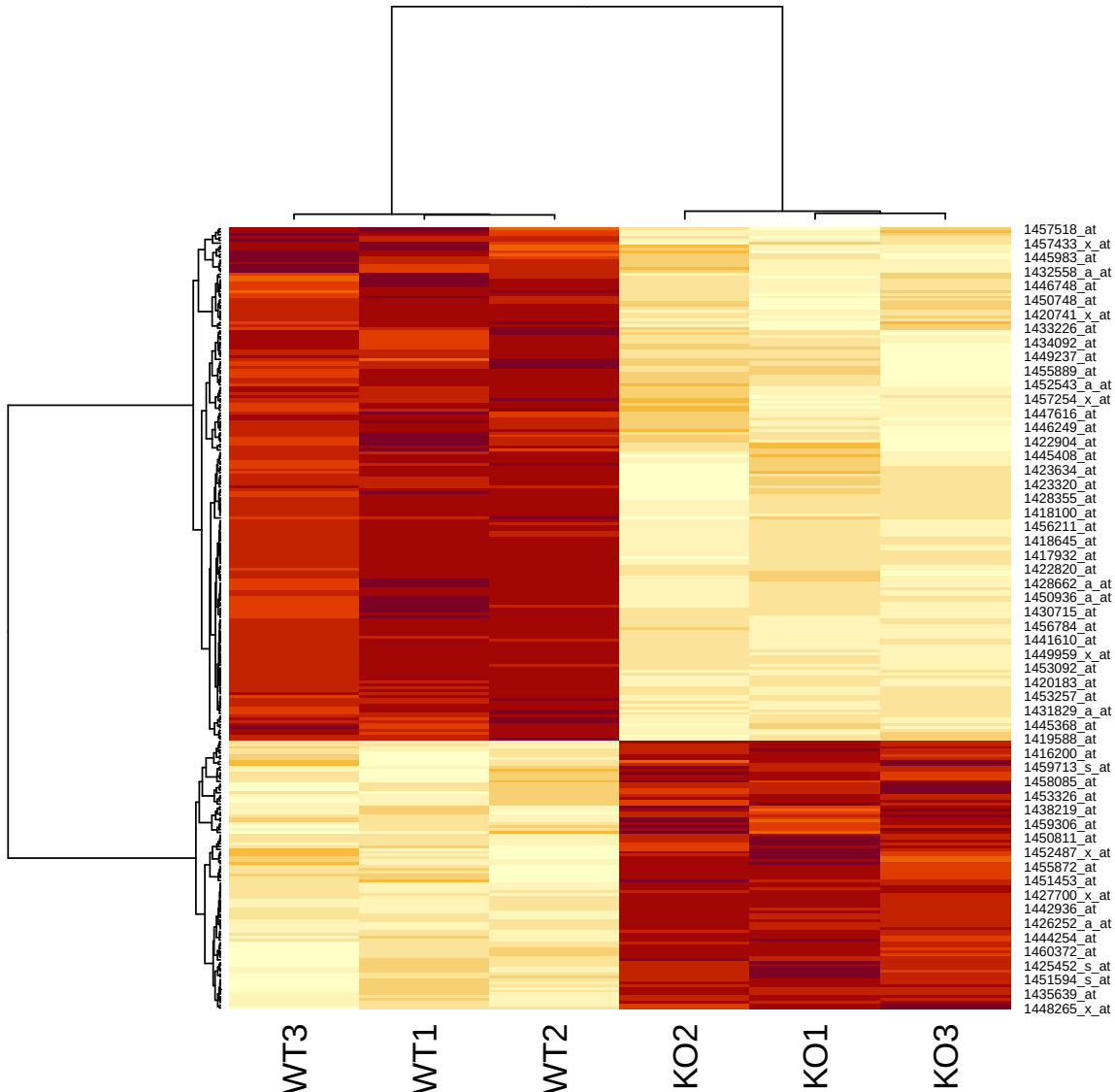
Heatmap

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

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

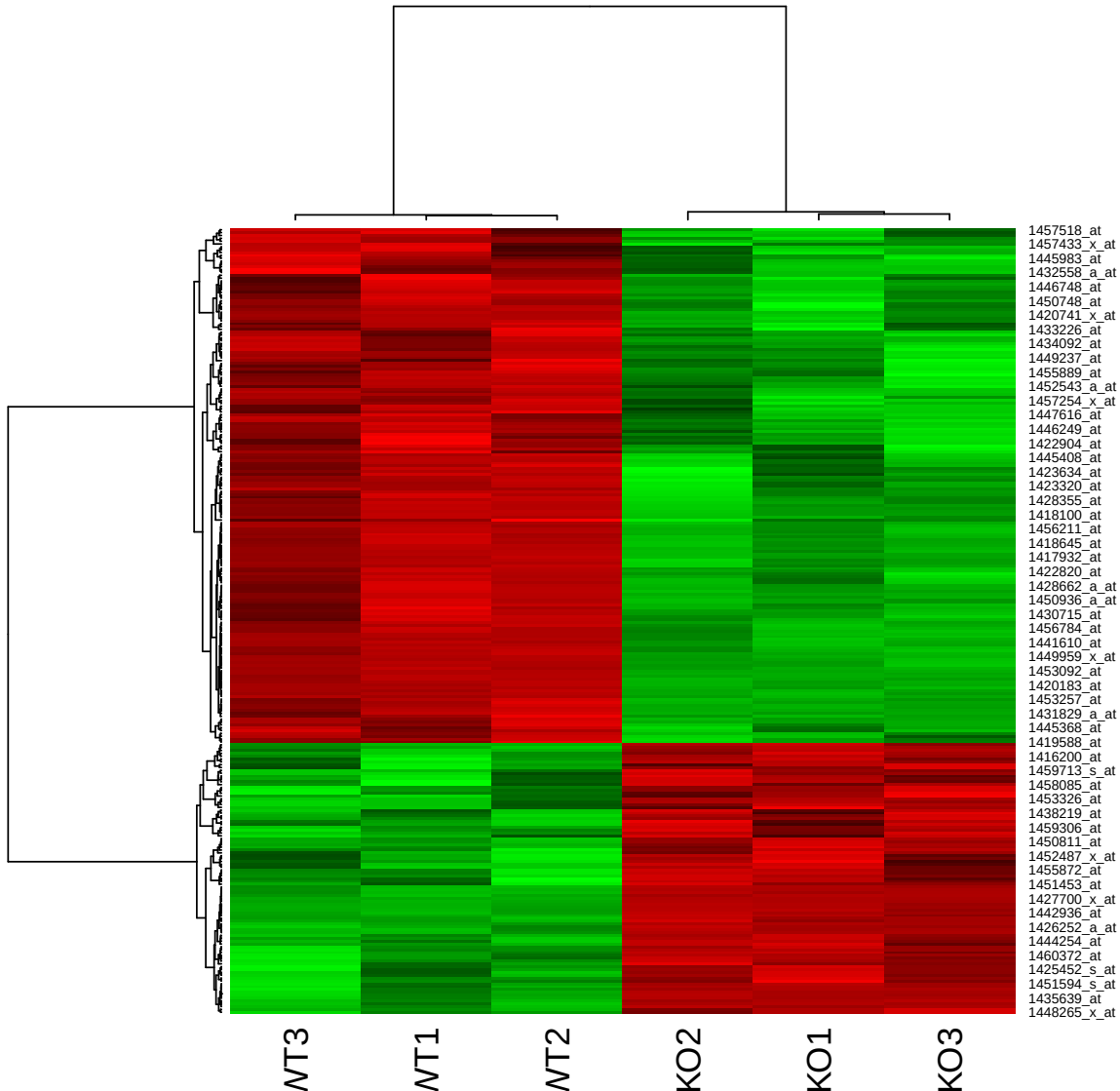
Heatmap

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



Heatmap

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



Annnotation

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

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

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

```
head(filtered_ids)
```

```
## [1] "1416200_at"   "1416236_a_at" "1417808_at"   "141793
## [6] "1418100_at"
```

```
write.table (filtered_ids, "results/filtered_ids.txt", row
```

Using DAVID annotation database

We can obtain the annotation online via the Database for Annotation, Visualization and Integrated Discovery (DAVID) by basically copying and pasting the probe ids to DAVID and running the annotation analysis online.

Using BioConductor annotation packages

Alternatively, we can generate a comprehensive functional annotation via BioConductor packages `annaffy` and `mouse4302.db`.

If the packages are not already installed, they can be installed using the following code:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install(c("annaffy", "mouse4302.db"))
```

Annnotation

Next the annotation can be obtained by loading the library:

```
library(annaffy)
```

```
## Loading required package: Biobase

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
```

Annnotation

Then generating and exporting the annotation into HTML format:

```
annotation_table = aafTableAnn(filtered_ids, "mouse4302.db"
```

```
## Loading required package: mouse4302.db
```

```
## Loading required package: org.Mm.eg.db
```

```
##
```

```
##
```

```
saveHTML(annotation_table, file="results/filtered_annotatio
browseURL("results/filtered_annotation.html")
```

Annnotation

As an HTML file, the annotation table can now be opened in any standard browser.

For a further focused functional analysis, let's split the DEGs according to the regulation pattern (up-regulated vs. down-regulated) to determine which functions/pathways/processes turned on or off from WT to KO:

Up-regulated Genes

```
filtered_up_ids = ids[filter_up]
length (filtered_up_ids)
```

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

```
write.table (filtered_up_ids, "results/filtered_up_ids.txt"
annotation_table = aafTableAnn(filtered_up_ids, "mouse4302
saveHTML(annotation_table, file="results/filtered_up_annota
browseURL("results/filtered_up_annotation.html")
```

Here is the functional annotation of the up-regulated genes

Down-regulated Genes

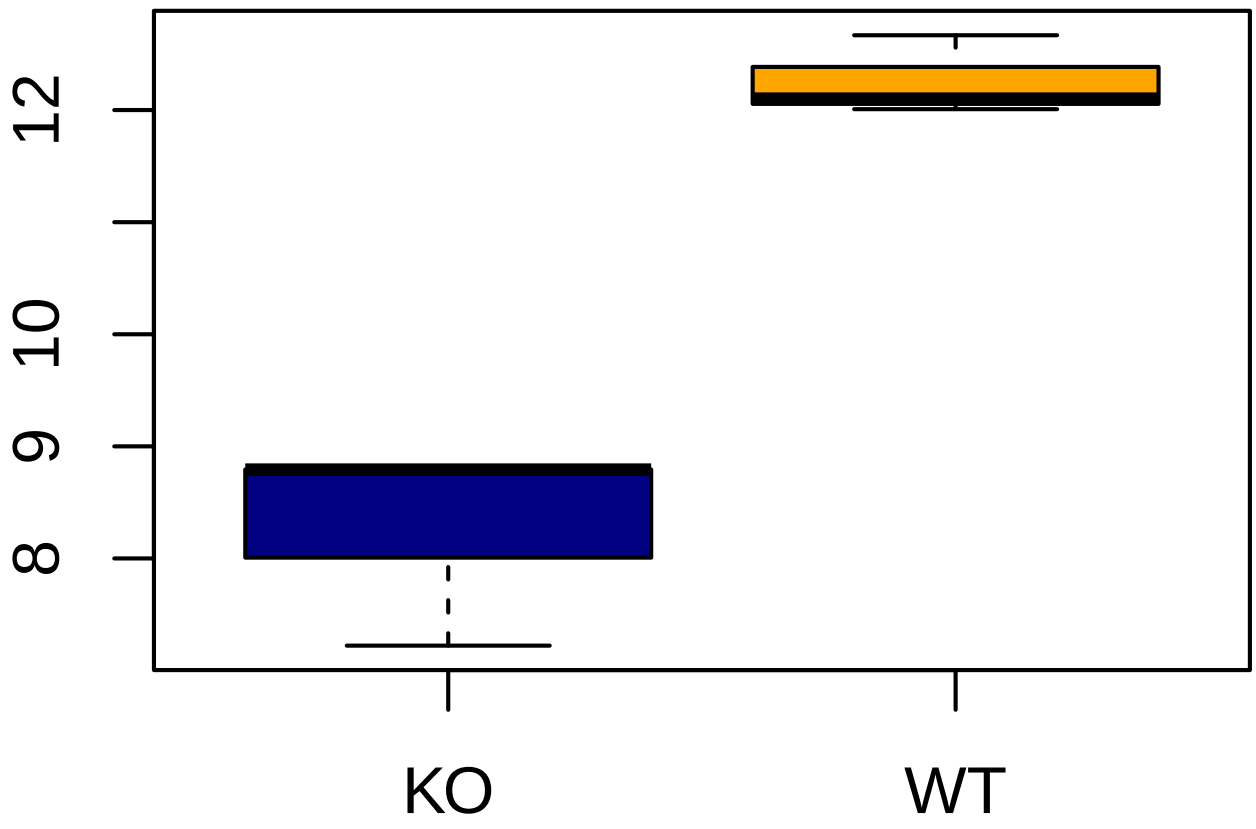
```
filtered_down_ids = ids[filter_down]
length (filtered_down_ids)
```

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

```
write.table (filtered_down_ids, "results/filtered_down_ids
annotation_table = aafTableAnn(filtered_down_ids, "mouse43
saveHTML(annotation_table, file="results/filtered_down_ann
browseURL("results/filtered_down_annotation.html")
```

Here is the functional annotation of the down-regulated genes

Sanity Check (Irf6)



Multiple Testing Correction

We conducted 45,101 statistical tests. The computed p-values should be corrected for **multiple testing**. The correction can be performed using `p.adjust`, which simply takes the original p-values as a vector and returns the adjusted (corrected) p-values using the False Discovery Rate adjustment method:

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

The number of the **original** p-values ≤ 0.05 is 5,099 while the number of **adjusted** p-values ≤ 0.05 is 9.

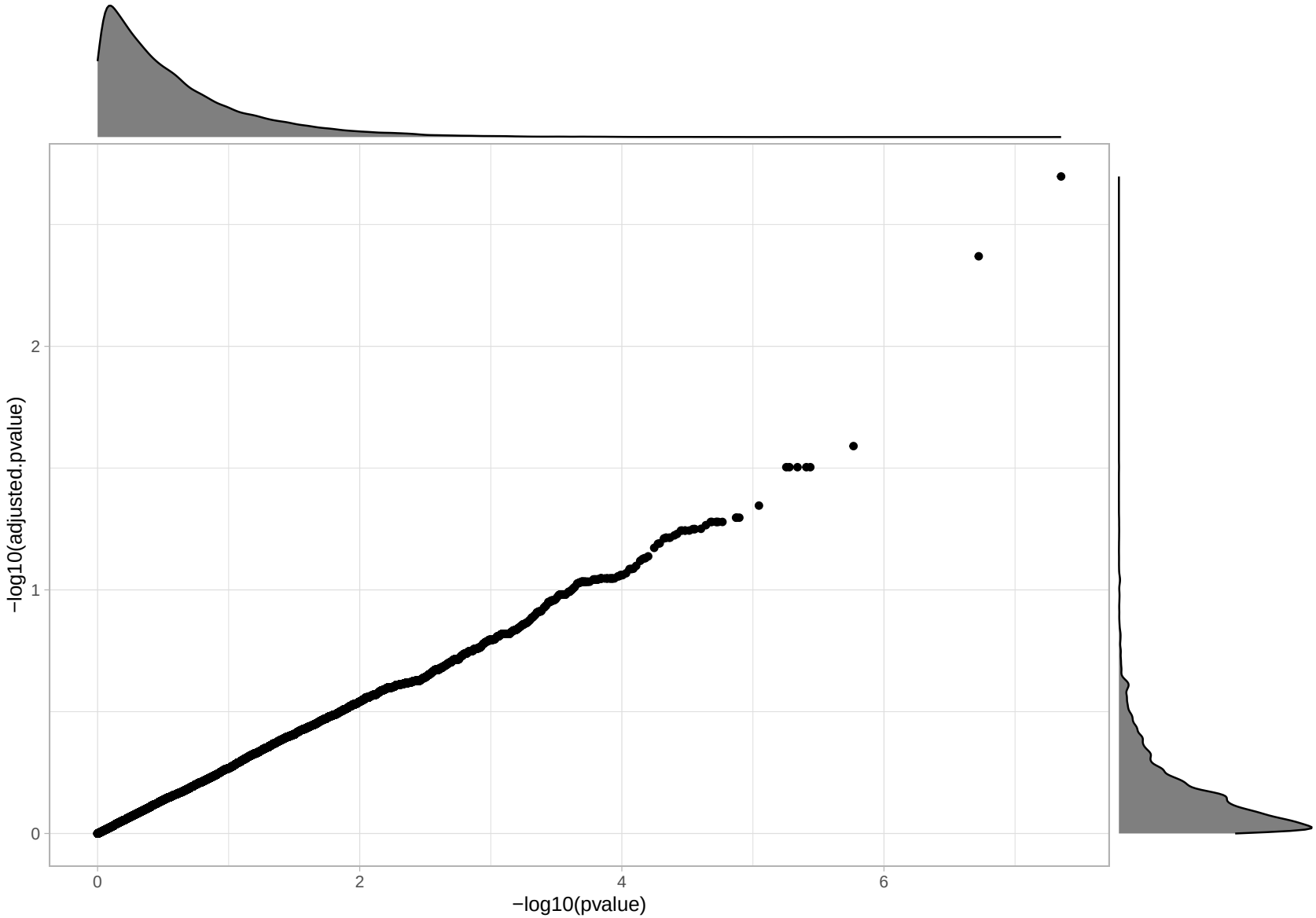
Multiple Testing Correction

Here is a sample of the original p-values and their corresponding adjusted p-values:

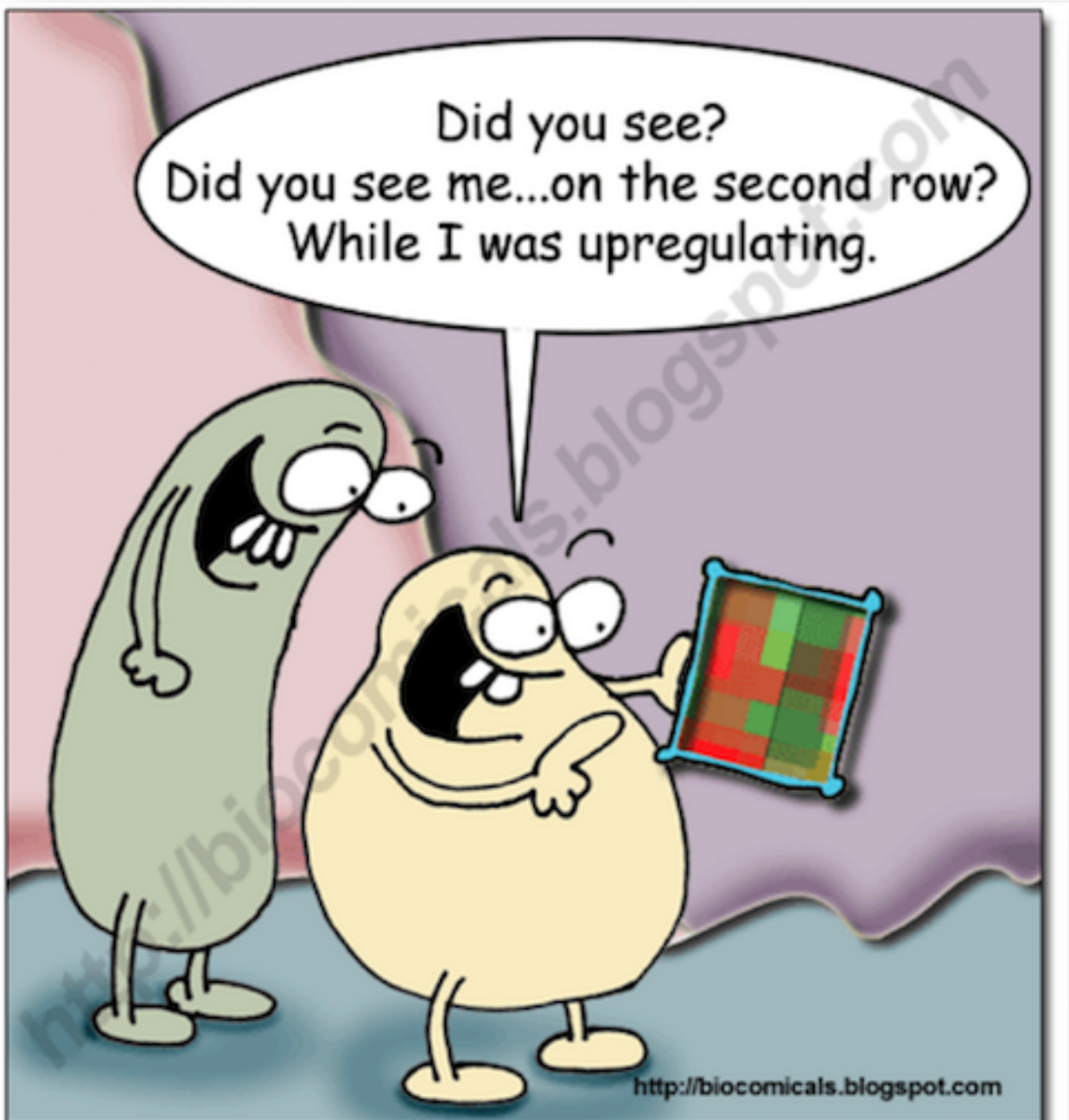
pvalue	adjusted.pvalue
0.0927063	0.5278755
0.1826633	0.6346918
0.1297791	0.5805456
0.2728992	0.7025472
0.2623772	0.6967834
0.0059478	0.2518079

Multiple Testing Correction

And here is the overall relationship between the original and the adjusted p-values:



Thank you!


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