ITMO UNIVERSITY

CT Lab ITMO UNIVERSITY

Microarrays-2

and introduction to gene expression studies

Kontantin Zaitsev

March 17th, 2020

Microarrays

Installing libraries for today

```r
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
if (!requireNamespace("GEOquery", quietly = TRUE)) BiocManager::install("GEOquery")
if (!requireNamespace("Biobase", quietly = TRUE)) BiocManager::install("Biobase")
if (!requireNamespace("ggplot2", quietly = TRUE)) install.packages("ggplot2")
if (!requireNamespace("reshape2", quietly = TRUE)) install.packages("reshape2")
if (!requireNamespace("limma", quietly = TRUE)) BiocManager::install("limma")
if (!requireNamespace("MASS", quietly = TRUE)) install.packages("MASS")

library(GEOquery)
library(Biobase)
library(ggplot2)
library(reshape2)
library(limma)
library(MASS)
```

Loading the dataset

```
GSE129260 <- getGEO("GSE129260", AnnotGPL = TRUE)[[1]]
```

```
## Warning: 64 parsing failures.
##   row              col           expected    actual        file
## 45038 Platform_SPOTID 1/0/T/F/TRUE/FALSE --Control literal data
## 45039 Platform_SPOTID 1/0/T/F/TRUE/FALSE --Control literal data
## 45040 Platform_SPOTID 1/0/T/F/TRUE/FALSE --Control literal data
## 45041 Platform_SPOTID 1/0/T/F/TRUE/FALSE --Control literal data
## 45042 Platform_SPOTID 1/0/T/F/TRUE/FALSE --Control literal data
## ..... ............... .................. ......... ............
## See problems(...) for more details.
```

fData and pData

- fData -- feature data, probe annotation
- pData -- phenotypcal data, sample annotaiton

Lets filter these objects and only keep things we need

Filtering pdata

This one is relatively straightforward

```
pData(GSE129260)$rep <- gsub(".*(rep\\d)$", "\\1", pData(GSE129260)$title)
pData(GSE129260) <- pData(GSE129260)[, c("characteristics_ch1.1", "characteristics_ch1.2"

colnames(pData(GSE129260)) <- c("Cell", "Treatment", "Replicate")
head(pData(GSE129260))
```

```
##                                           Cell              Treatment
## GSM3703675 cell type: IL-10 positive B cells stimulation: anti-CD40
## GSM3703676 cell type: IL-10 negative B cells stimulation: anti-CD40
## GSM3703677 cell type: IL-10 positive B cells       stimulation: LPS
## GSM3703678 cell type: IL-10 negative B cells       stimulation: LPS
## GSM3703679 cell type: IL-10 positive B cells stimulation: anti-CD40
## GSM3703680 cell type: IL-10 negative B cells stimulation: anti-CD40
##            Replicate
## GSM3703675      rep1
## GSM3703676      rep1
## GSM3703677      rep1
```

Filtering fdata

What do we want to keep?

```
colnames(fData(GSE129260))
```

```
##  [1] "ID"                    "Gene title"
##  [3] "Gene symbol"           "Gene ID"
##  [5] "UniGene title"         "UniGene symbol"
##  [7] "UniGene ID"            "Nucleotide Title"
##  [9] "GI"                    "GenBank Accession"
## [11] "Platform_CLONEID"      "Platform_ORF"
## [13] "Platform_SPOTID"       "Chromosome location"
## [15] "Chromosome annotation" "GO:Function"
## [17] "GO:Process"            "GO:Component"
## [19] "GO:Function ID"        "GO:Process ID"
## [21] "GO:Component ID"
```

Gene IDs: we are doomed

- **Gene symbol** - something meaningfull
- **Entrez ID** - https://www.ncbi.nlm.nih.gov/gene/ENTREZ_ID
- **ENSEMBL** -- just ENSEMBL, when you do RNA-seq, these IDs will show up
- RefSeq ID
- And many more

(I wanted to make a meme first, but gene id conversion is not funny at all)

Filtering fdata

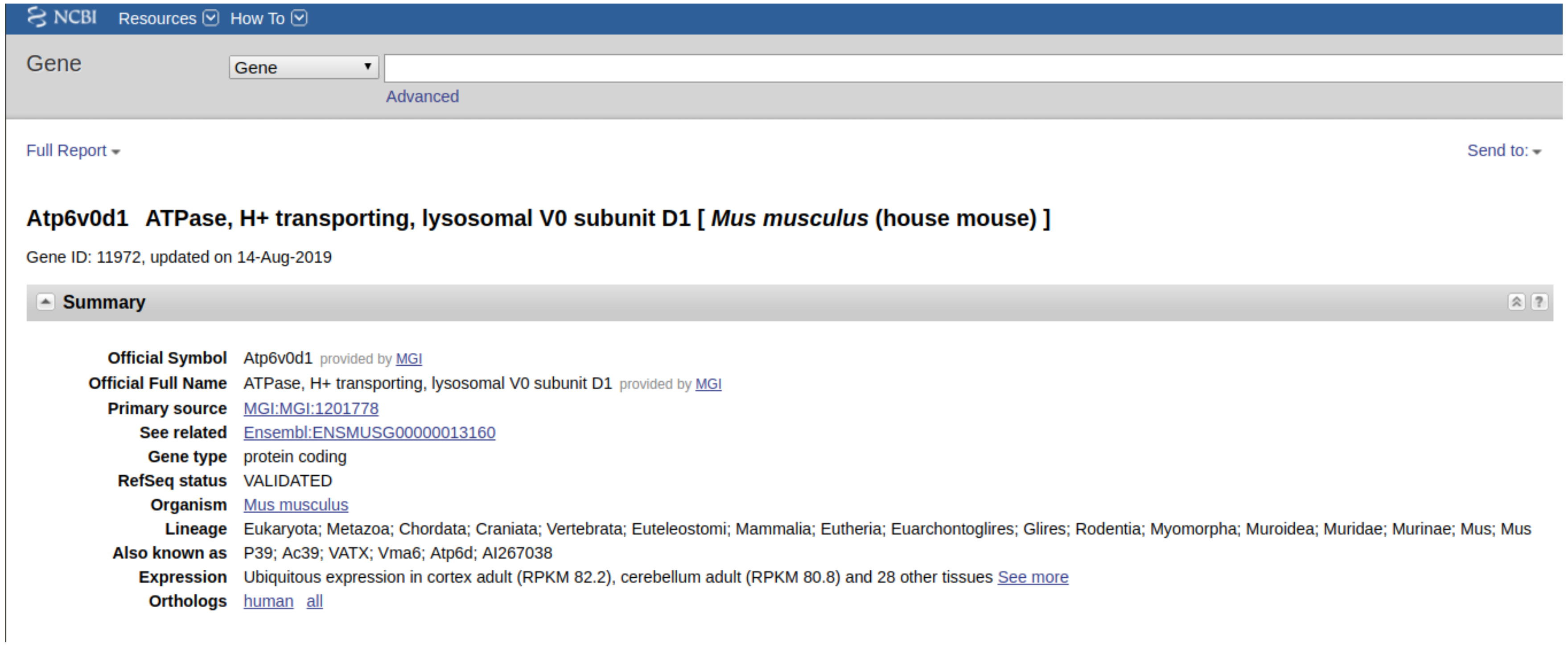
Lets keep ID, symbol, and entrez id

```
fData(GSE129260) <- fData(GSE129260)[, c("ID", "Gene symbol", "Gene ID")]
head(fData(GSE129260))
```

```
##                          ID Gene symbol Gene ID
## 1415670_at     1415670_at       Copg1   54161
## 1415671_at     1415671_at    Atp6v0d1   11972
## 1415672_at     1415672_at      Golga7   57437
## 1415673_at     1415673_at        Psph  100678
## 1415674_a_at 1415674_a_at     Trappc4   60409
## 1415675_at     1415675_at        Dpm2   13481
```

Entrez ID

Entrez ID: https://www.ncbi.nlm.nih.gov/gene/11972



Atp6v0d1 ATPase, H+ transporting, lysosomal V0 subunit D1 [*Mus musculus* (house mouse)]

Gene ID: 11972, updated on 14-Aug-2019

Summary

Official Symbol	Atp6v0d1 provided by MGI
Official Full Name	ATPase, H+ transporting, lysosomal V0 subunit D1 provided by MGI
Primary source	MGI:MGI:1201778
See related	Ensembl:ENSMUSG00000013160
Gene type	protein coding
RefSeq status	VALIDATED
Organism	Mus musculus
Lineage	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Glires; Rodentia; Myomorpha; Muroidea; Muridae; Murinae; Mus; Mus
Also known as	P39; Ac39; VATX; Vma6; Atp6d; AI267038
Expression	Ubiquitous expression in cortex adult (RPKM 82.2), cerebellum adult (RPKM 80.8) and 28 other tissues See more
Orthologs	human all

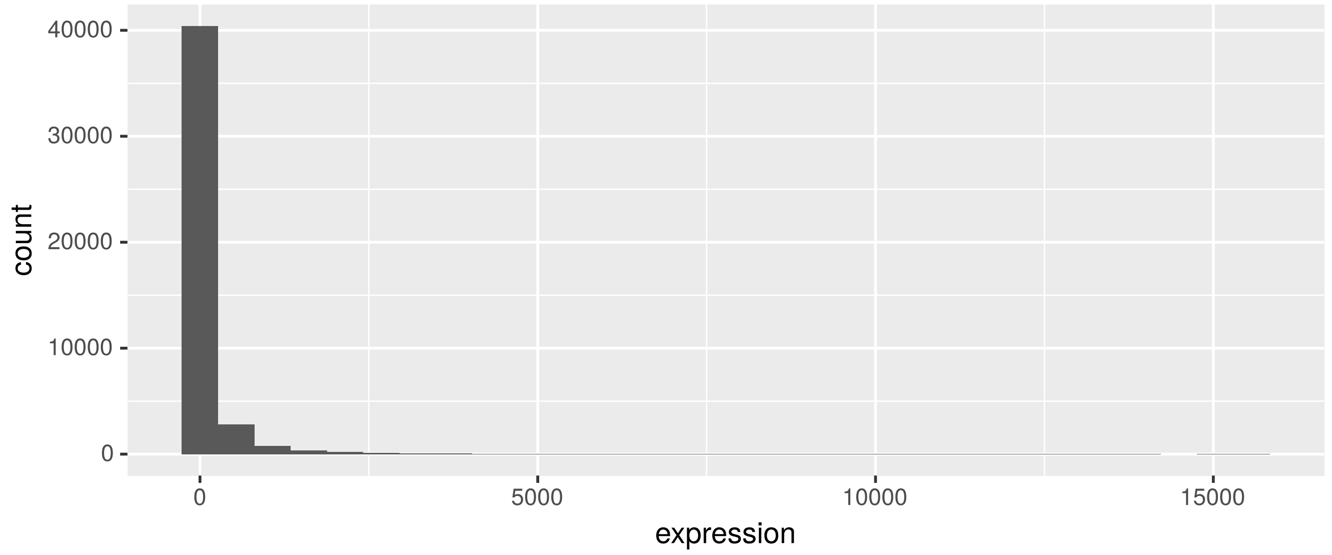
Figuring out expression space

- Expression levels for each gene/probe can be in different space: linear and logarithmical space.
- Usually we determine that by looking at values (especially) maximum values
- If maximum value is < 25 we think it is in log-space
- If it is > 1000 we think it is in linear space
- (If it is somewhere in between we are usually confused)
- **Common mistake is to get log (log (value)) instead of just log(value)**
- Don't apply log to your data if it is already log-transformed

Figuring out expression space

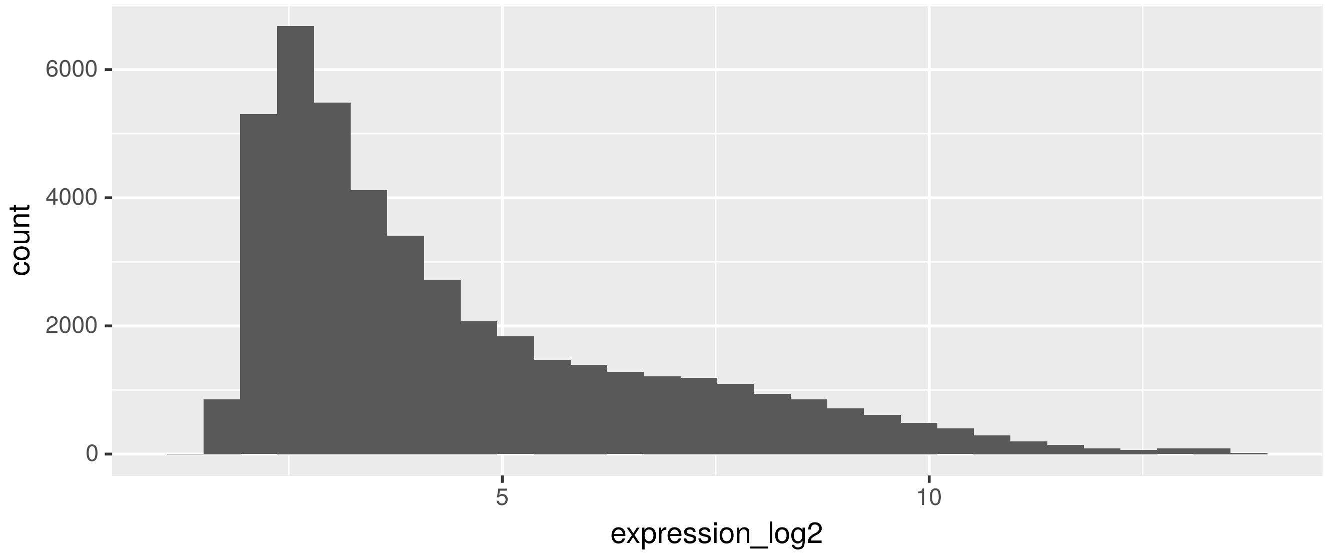
```r
ggplot(data=data.frame(expression=exprs(GSE129260)[, 1]),
       aes(x=expression)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Figuring out expression space

```
ggplot(data=data.frame(expression_log2=log2(exprs(GSE129260)[, 1])),
       aes(x=expression_log2)) +
  geom_histogram()
```



Observations

- In microarray we don't have true "zeroes"

```
min(exprs(GSE129260))
```

```
## [1] 2.538441
```

- This means that even for non-expressed probes we detect some light intensity (background fluorescence)
- In log-scale distribution of expression values usually looks "more normal"

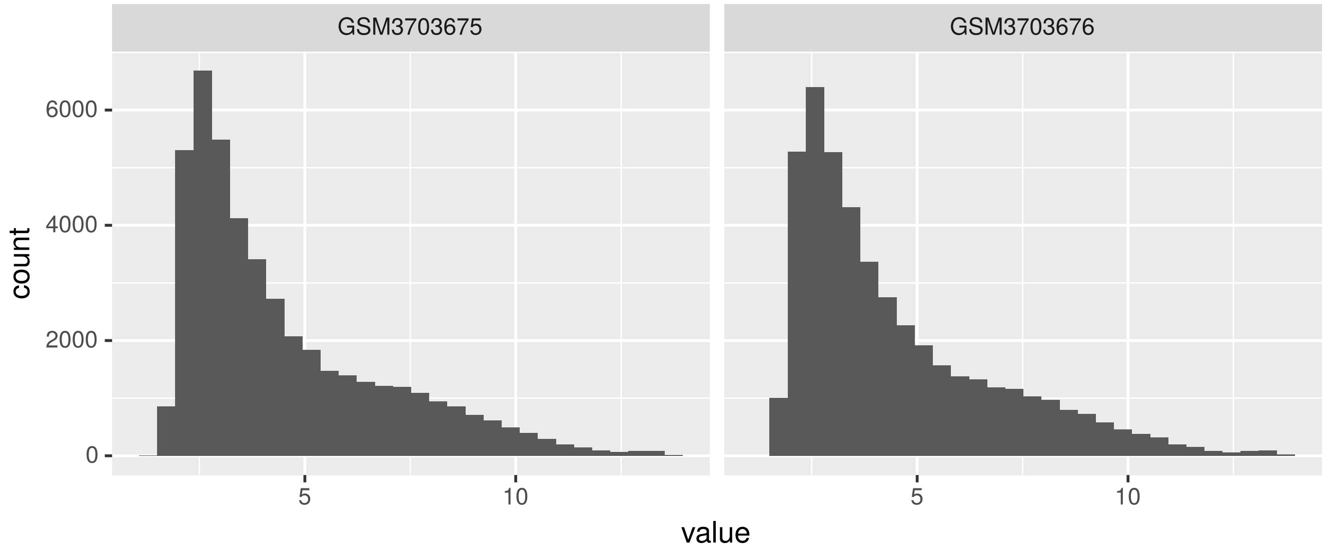
About expression space

- Most of gene expression studies are done in log-space (we believe that error is normally distributed in log-space)
- There are some exceptions (like gene expression deconvolution, usually is done in linear space)

Comparing distributions between samples

```r
twoSamples <- melt(exprs(GSE129260[, 1:2]))
twoSamples$value <- log2(twoSamples$value)

ggplot(data=twoSamples, aes(x=value)) +
  facet_grid(~Var2) + geom_histogram()
```



Comparing distributions between samples

```
colSums(exprs(GSE129260))
```

```
## GSM3703675 GSM3703676 GSM3703677 GSM3703678 GSM3703679 GSM3703680
##    7423000    7373798    7272281    7464280    7669166    7689773
## GSM3703681 GSM3703682
##    7710919    7758146
```

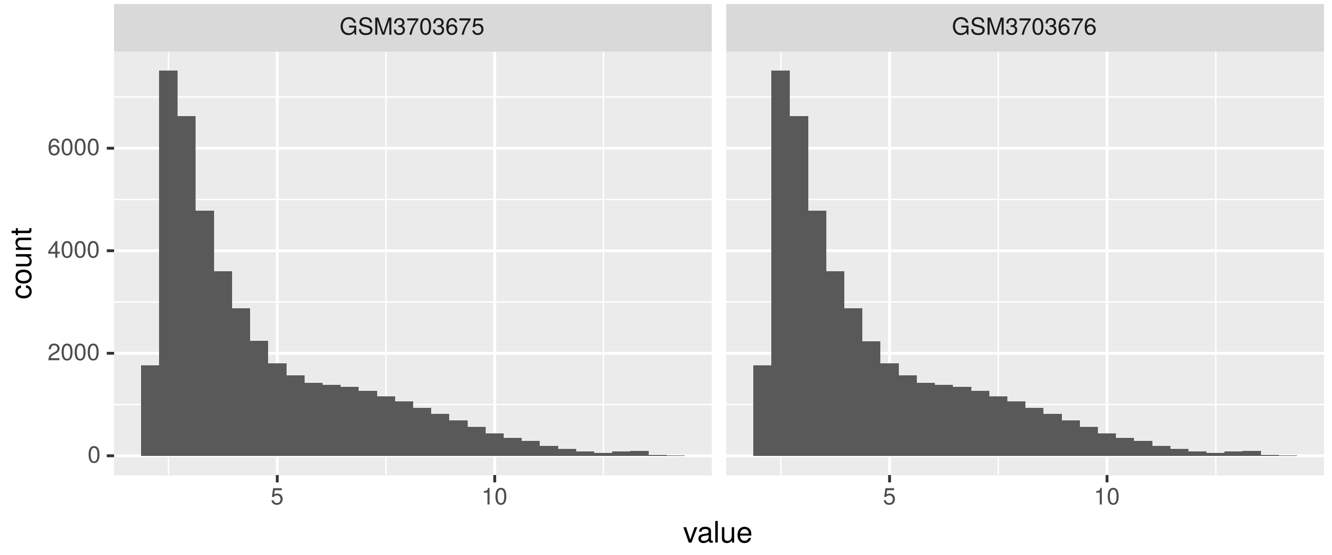
Quantile normalization

- Distributions are similar yet different
- Better safe than sorry: we apply quantile normalization anyway
- https://en.wikipedia.org/wiki/Quantile_normalization

Quantile normalization

```
exprs(GSE129260) <- normalizeBetweenArrays(log2(exprs(GSE129260)+1), method="quantile")
twoSamples <- melt(exprs(GSE129260[, 1:2]))

ggplot(data=twoSamples, aes(x=value)) +
  facet_grid(~Var2) + geom_histogram()
```



Moving to gene expression

- We were mostly looking at "probe-level" expression
- We would like to move to gene-level expression
- But what can happen to a probe?

Moving to gene expression

Please run

```
head(fData(GSE129260), 1000)
```

Moving to gene expression

- Let's remove **probes that map to several genes** (they are not measuring anything specific)
- Let's remove **probes that don't map to any gene** (they are not measuring anything useful)
- If several probes are mapped to the same gene we only take the probe with the highest average expression
- Let's only keep 12000 top expressed genes

Moving to gene expression

```r
GSE129260 <- GSE129260[!grepl("///", fData(GSE129260)$`Gene symbol`), ]
GSE129260 <- GSE129260[fData(GSE129260)$`Gene symbol` != "", ]

fData(GSE129260)$mean_expression <- apply(exprs(GSE129260), 1, mean)
GSE129260 <- GSE129260[order(fData(GSE129260)$mean_expression, decreasing = TRUE), ]
GSE129260 <- GSE129260[!duplicated(fData(GSE129260)$`Gene ID`), ]
GSE129260 <- GSE129260[seq_len(12000), ]
dim(GSE129260)
```

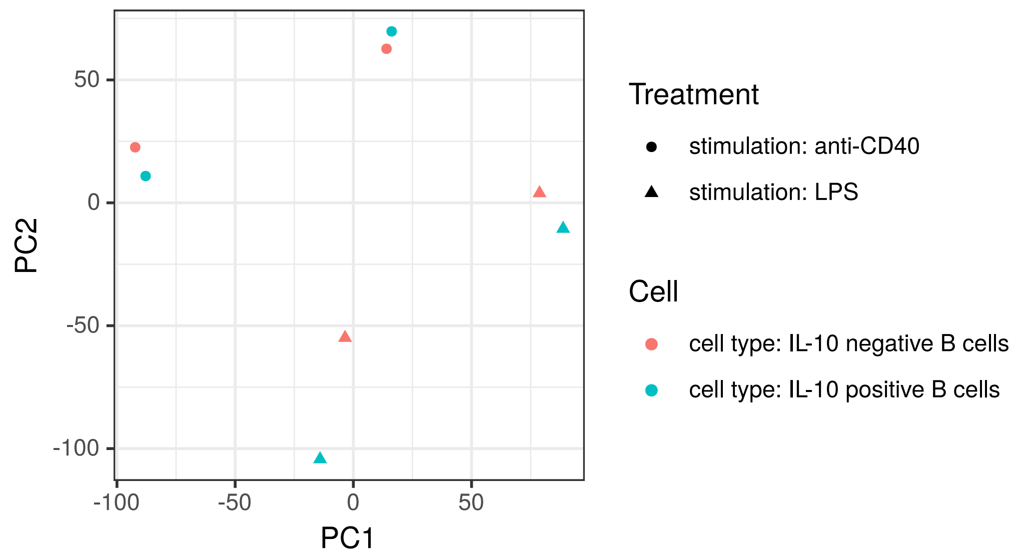
```
## Features  Samples
##    12000        8
```

Whoray

- This matrix is finally something that we can analyze
- Let's do a PCA first and see how our samples are grouped

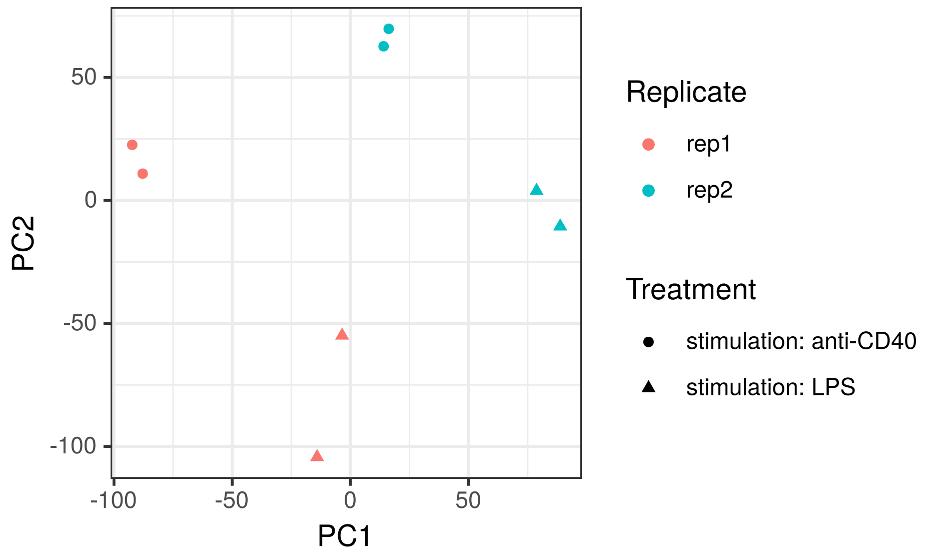
PCA plot

```r
pcas <- prcomp(t(exprs(GSE129260)), scale. = T)
plotData <- cbind(pcas$x[, 1:2], pData(GSE129260))
ggplot(plotData, aes(x=PC1, y=PC2, color=Cell, shape=Treatment)) +
  geom_point() + theme_bw() + theme(aspect.ratio = 1)
```



PCA plot

```
ggplot(plotData, aes(x=PC1, y=PC2, color=Replicate, shape=Treatment)) +
  geom_point() + theme_bw() + theme(aspect.ratio = 1)
```



Sanity check:

Maybe something is just misslabeled in GEO? Can we check Il10 expression?

```
fData(GSE129260)[fData(GSE129260)$`Gene symbol` == "Il10", ]
```

```
##                      ID Gene symbol Gene ID mean_expression
## 1450330_at 1450330_at        Il10   16153        10.62271
```

```
exprs(GSE129260)["1450330_at", ]
```

```
## GSM3703675 GSM3703676 GSM3703677 GSM3703678 GSM3703679 GSM3703680
##  12.655218   9.000712  11.919731   9.508046  12.107867   8.721228
## GSM3703681 GSM3703682
##  11.585481   9.483412
```

Variance expained:

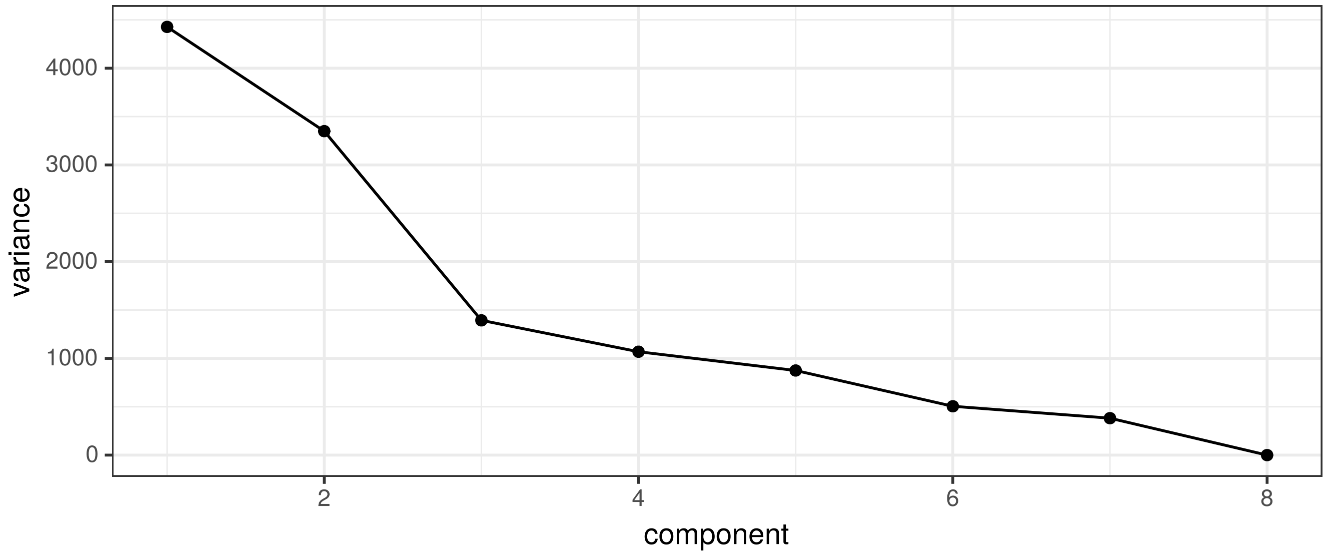
Usually we show variance explained by components

$$Var = \sigma^2$$

prcomp calculates standard deviation

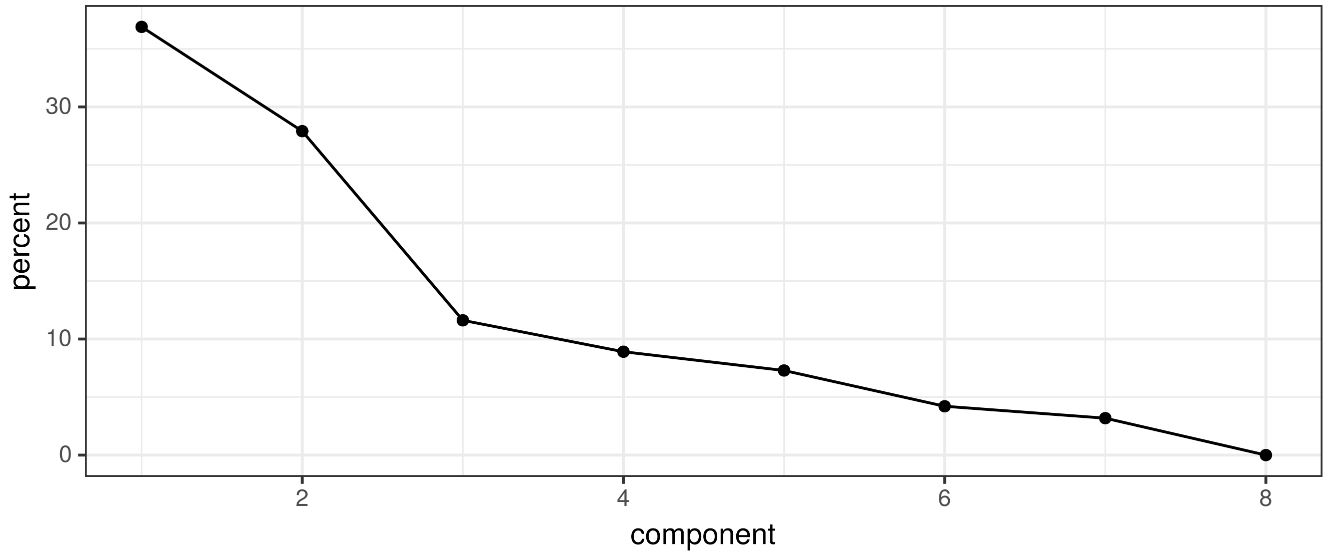
Variance explained

```
variance <- pcas$sdev^2
ggplot(data=data.frame(component=1:8, variance=variance),
       aes(x=component, y=variance)) +
  geom_point() + geom_line() + theme_bw()
```



Variance explained: ratio

```
variance <- variance / sum(variance)
ggplot(data=data.frame(component=1:8, percent=variance * 100),
       aes(x=component, y=percent)) +
  geom_point() + geom_line() + theme_bw()
```



Differential expression

```r
GSE129260.design <- model.matrix(~0+Cell+Treatment+Replicate, data=pData(GSE129260))
colnames(GSE129260.design) <- c("il10neg", "il10pos", "LPS", "rep2")

fit <- lmFit(GSE129260, GSE129260.design)

fit2 <- contrasts.fit(fit, makeContrasts(il10pos - il10neg, levels=GSE129260.design))
fit2 <- eBayes(fit2, trend = T)

de <- topTable(fit2, adjust.method="BH", number=Inf, sort.by = "P")
```

Differential expression

```
head(de)
```

```
##                              ID Gene.symbol Gene.ID mean_expression      logFC
## 1416066_at     1416066_at         Cd9   12527        9.408102  1.2672152
## 1450330_at     1450330_at        Il10   16153       10.622712  2.8887248
## 1428947_at     1428947_at        Mzb1   69816       10.836028  0.9663412
## 1419298_at     1419298_at        Pon3  269823        5.266634  1.5167428
## 1455530_at     1455530_at    Ighv14-2  668421        7.709365 -2.0150553
## 1422557_s_at 1422557_s_at         Mt1   17748        9.510878  1.7284823
##                AveExpr          t      P.Value   adj.P.Val        B
## 1416066_at    9.408102  13.164815 3.503802e-07 0.002167242 6.623000
## 1450330_at   10.622712  12.644593 4.950996e-07 0.002167242 6.368938
## 1428947_at   10.836028  12.512033 5.418104e-07 0.002167242 6.301640
## 1419298_at    5.266634  11.570981 1.054446e-06 0.002874045 5.791615
## 1455530_at    7.709365 -11.398509 1.197519e-06 0.002874045 5.691623
## 1422557_s_at  9.510878   9.946067 3.758758e-06 0.006867570 4.758682
```

Differential expression

```r
GSE129260.design <- model.matrix(~0+Replicate+Treatment+Cell, data=pData(GSE129260))
colnames(GSE129260.design) <- c("rep1", "rep2", "LPS", "pos")

fit <- lmFit(GSE129260, GSE129260.design)

fit2 <- contrasts.fit(fit, makeContrasts(rep2-rep1, levels=GSE129260.design))
fit2 <- eBayes(fit2, trend = T)

de <- topTable(fit2, adjust.method="BH", number=Inf, sort.by = "P")
```

Differential expression

```
head(de)
```

```
##                          ID Gene.symbol Gene.ID mean_expression     logFC
## 1417210_at       1417210_at     Eif2s3y   26908        4.441604 -4.094437
## 1452406_x_at 1452406_x_at       Erdr1  170942       11.196021 -2.201366
## 1426438_at       1426438_at       Ddx3y   26900        4.690768 -4.389121
## 1446928_at       1446928_at     Dnajc17   69408       10.279861 -1.147371
## 1439465_x_at 1439465_x_at       Agbl5  231093        7.096873 -2.576527
## 1420450_at       1420450_at       Mmp10   17384        8.281950 -2.096449
##                 AveExpr         t      P.Value    adj.P.Val         B
## 1417210_at     4.441604 -24.39940 1.575663e-09 1.890796e-05 11.442885
## 1452406_x_at 11.196021 -18.39299 1.911655e-08 1.146728e-04  9.677991
## 1426438_at     4.690768 -17.56275 2.866819e-08 1.146728e-04  9.359591
## 1446928_at    10.279861 -16.56499 4.782405e-08 1.282766e-04  8.945990
## 1439465_x_at   7.096873 -16.35540 5.344860e-08 1.282766e-04  8.854482
## 1420450_at     8.281950 -15.30989 9.502704e-08 1.696957e-04  8.371977
```

Let's dig a bit into theory

Spoiler alert: i am just a prorammer / bioinformatician :) You might want to consult a proper statistician

Simple t-test

- DE is about comparing means of several groups
- Let's know forget about previous dataset
- Lets assume we have 3 A samples and 3 B samples
- We would like to compare A vs B

Linear models: simulations

Lets assume we know true A and true B:

```
trueAB <- matrix(c(
  1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
  10, 9, 8, 7, 6, 5, 4, 3, 2, 1
), ncol=2)
colnames(trueAB) <- c("A", "B")
rownames(trueAB) <- paste0("Gene ", 1:10)
head(trueAB)
```

```
##         A  B
## Gene 1 1 10
## Gene 2 2  9
## Gene 3 3  8
## Gene 4 4  7
## Gene 5 5  6
## Gene 6 6  5
```

ITsMOre than a UNIVERSITY

Linear models: simulations

Let's create noisy replicates (we assume noise is equal)

```
set.seed(1)
observed <- trueAB[, c(1, 1, 1, 2, 2, 2)]
colnames(observed) <- c("A1", "A2", "A3", "B1", "B2", "B3")
rownames(observed) <- paste0("Gene ", 1:10)
observed <- observed + rnorm(60)
head(observed)
```

```
##               A1       A2       A3        B1       B2        B3
## Gene 1 0.3735462 2.511781 1.918977 11.358680 9.835476 10.398106
## Gene 2 2.1836433 2.389843 2.782136  8.897212 8.746638  8.387974
## Gene 3 2.1643714 2.378759 3.074565  8.387672 8.696963  8.341120
## Gene 4 5.5952808 1.785300 2.010648  6.946195 7.556663  5.870637
## Gene 5 5.3295078 6.124931 5.619826  4.622940 5.311244  7.433024
## Gene 6 5.1795316 5.955066 5.943871  4.585005 4.292505  6.980400
```

Linear models: simulations

Let's define linear model

```r
modelMatrix <- matrix(
  c(1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1),
  ncol = 2
)
colnames(modelMatrix) <- c("A", "B")
rownames(modelMatrix) <- colnames(observed)
head(modelMatrix)
```

```
##    A B
## A1 1 0
## A2 1 0
## A3 1 0
## B1 0 1
## B2 0 1
## B3 0 1
```

Linear models: simulations

If we multiply matrices $trueAB$ and $modelMatrix^T$, we will get desired matrix. But without noise.

```
trueAB %*% t(modelMatrix)
```

```
##          A1 A2 A3 B1 B2 B3
## Gene 1    1  1  1 10 10 10
## Gene 2    2  2  2  9  9  9
## Gene 3    3  3  3  8  8  8
## Gene 4    4  4  4  7  7  7
## Gene 5    5  5  5  6  6  6
## Gene 6    6  6  6  5  5  5
## Gene 7    7  7  7  4  4  4
## Gene 8    8  8  8  3  3  3
## Gene 9    9  9  9  2  2  2
## Gene 10  10 10 10  1  1  1
```

Linear models: simulations

In reality we don't know true answer, and we would like to estimate the opposite: from model and noisy matrixes get estimates to mean values.

$$observed = true \times model^T + noise$$

$$observed \times (model^T)^{-1} = true + noise$$

Linear models: simulations

Linear models allow us to quickly find means

```
means <- observed %*% ginv(t(modelMatrix))
head(means)
```

```
##              [,1]      [,2]
## Gene 1 1.601435 10.530754
## Gene 2 2.451874  8.677275
## Gene 3 2.539232  8.475252
## Gene 4 3.130410  6.791165
## Gene 5 5.691421  5.789069
## Gene 6 5.692823  5.285970
```

Comparing means: T-test

We could simply use T-test to test if means are different.

We have equal size samples, equal error variance. Good.

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s^2_{X_1} + s^2_{X_2}}{n}}}$$

and degrees of freedom for testing are

$$d.f. = 2n - 2$$

Taken from https://en.wikipedia.org/wiki/Student%27s_t-test

Simple T-test: gene 4

```r
t.test(observed[4, 1:3], observed[4, 4:6], var.equal=TRUE)
```

```
##
##      Two Sample t-test
##
## data:  observed[4, 1:3] and observed[4, 4:6]
## t = -2.7547, df = 4, p-value = 0.05113
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -7.35042845  0.02891789
## sample estimates:
## mean of x mean of y
##  3.130410  6.791165
```

Simple t-test

- Even for true different (4 vs 7 with sd=1 error) genes we couldn't get P significant T-test p value
- Can we somehow empower T-test ?

eBayes

- Empirical Bayes Statistics for Differential Expression

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s^2_{X_1} + s^2_{X_2}}{n}}}$$

- Idea is that deviations depend on n, t-statistic depends on n and degrees of freedom depend on n

eBayes

https://konsolerr.github.io/gene_expression_2019/microarray/smyth2004.pdf

- Since we calculate DE for many-many genes we can infer additional sample size from aggregating their deviations too