ITMO UNIVERSITY

CT Lab
ITMO UNIVERSITY

Secondary analysis

in gene expression studies

Kontantin Zaitsev

March 17th, 2020

Secondary analysis

Getting an expression matrix

- Assume bulk gene expression
- Microarray: itensities -> probe/gene symbol mapping -> gene expression matrix
- RNA-seq: raw reads -> alignments -> quantification -> gene expression matrix

Gene expression matrix

- Rows are genes
- Columns are samples
- We assume columns to be somewhat normalized, so gene expression levels are representative across all samples

	GSM3703675	GSM3703676	GSM3703677	GSM3703678	GSM3703679	GSM3703680	GSM3703681	GSM3703682
Dhx36	6.868925	7.802883	7.265303	7.711423	7.925366	8.064470	7.782933	7.878214
Arl6ip4	8.610726	8.349129	8.961090	8.863572	8.210400	8.298973	8.612580	8.380108
Tram1	8.117650	7.725020	9.782122	9.205673	8.632618	8.403167	10.311316	9.872407
Mir425	4.124838	4.674299	5.128062	4.936080	4.404071	4.785895	4.397090	4.670425
Pex6	8.295669	8.071793	7.823910	8.110941	7.758333	7.989627	7.529114	7.686020
Nans	8.768137	8.620869	9.358366	9.096089	8.752107	8.696490	9.641165	9.123614
...								

Once we have an expression matrix

Conceptual analysis steps are the same:

- **Quality controls: PCA + outlier/batch removal if needed**
- Differential expression design
- Performing differential expression
- DE genes: looking for possible biological pathways, transcriptional factors, regulators...

Sources of variance

It is important to identify sources of gene expression variance

- Variation included by design: cell type, treatment, cases vs controls
- Biological sources of variation: cell cycle, sex of mice/donor, cell types present in the sample
- Unwanted variaton: batch effect, donor effect
- Technical variation: microarray variation, sequencing variation

libraries

```
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
if (!requireNamespace("ggplot2", quietly = TRUE)) install.packages("ggplot2")
if (!requireNamespace("pheatmap", quietly = TRUE)) install.packages("pheatmap")
if (!requireNamespace("RColorBrewer", quietly = TRUE)) install.packages("RColorBrewer")
if (!requireNamespace("ggrepel", quietly = TRUE)) install.packages("ggrepel")
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr")

if (!requireNamespace("limma", quietly = TRUE)) BiocManager::install("limma")
if (!requireNamespace("Biobase", quietly = TRUE)) BiocManager::install("Biobase")
if (!requireNamespace("sva", quietly = TRUE)) BiocManager::install("sva")
if (!requireNamespace("fgsea", quietly = TRUE)) BiocManager::install("fgsea")
```

Understanding variance

```
library(Biobase)
library(limma)
library(sva)
library(ggplot2)
library(pheatmap)
library(RColorBrewer)
library(ggrepel)
library(dplyr)
library(fgsea)

blueWhiteRed <- colorRampPalette(c("#3859A8", "#EEEEEE", "#EE2930"))(10)

load("gse129260.Rdata")
```

Understanding variance

```
head(exprs(gse129260))
```

```
##        GSM3703675 GSM3703676 GSM3703677 GSM3703678 GSM3703679 GSM3703680
## Rps29     13.76204   13.83782   13.72959   13.76204   13.97115   13.76204
## Rpl37a    13.54983   13.70194   13.64128   13.64128   13.72959   13.79243
## Rplp2     13.62928   13.68329   13.65993   13.68329   13.65993   13.68329
## Tpt1      13.64128   13.65993   13.59760   13.62928   13.76204   13.65993
## Rpl41     13.53492   13.50823   13.61335   13.56125   13.52919   13.54983
## Eef1a1    13.68329   13.66878   13.46390   13.54537   13.53492   13.58597
##        GSM3703681 GSM3703682
## Rps29     13.68329   13.76204
## Rpl37a    13.72959   13.79243
## Rplp2     13.65993   13.65993
## Tpt1      13.64128   13.62928
## Rpl41     13.59760   13.64128
## Eef1a1    13.49293   13.47191
```

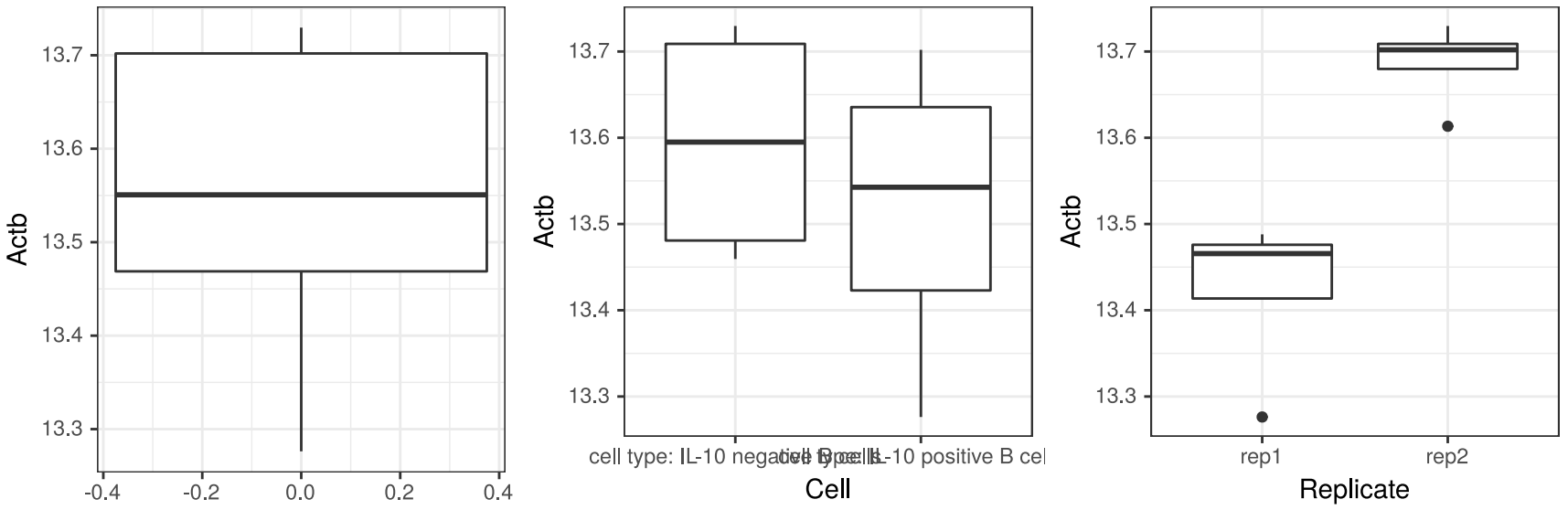
Understanding variance

```
someGenes <- exprs(gse129260)[c("Actb", "Ddx3y", "Il10"), ]
plotData <- t(someGenes)
plotData <- as.data.frame(plotData)
plotData <- cbind(plotData, pData(gse129260))

head(plotData, 4)
```

```
##                Actb    Ddx3y      Il10                                Cell
## GSM3703675 13.47191 7.674220 12.655218 cell type: IL-10 positive B cells
## GSM3703676 13.45945 6.702041  9.000712 cell type: IL-10 negative B cells
## GSM3703677 13.27625 7.122010 11.919731 cell type: IL-10 positive B cells
## GSM3703678 13.48798 6.043041  9.508046 cell type: IL-10 negative B cells
##                          Treatment Replicate
## GSM3703675 stimulation: anti-CD40      rep1
## GSM3703676 stimulation: anti-CD40      rep1
## GSM3703677       stimulation: LPS      rep1
## GSM3703678       stimulation: LPS      rep1
```

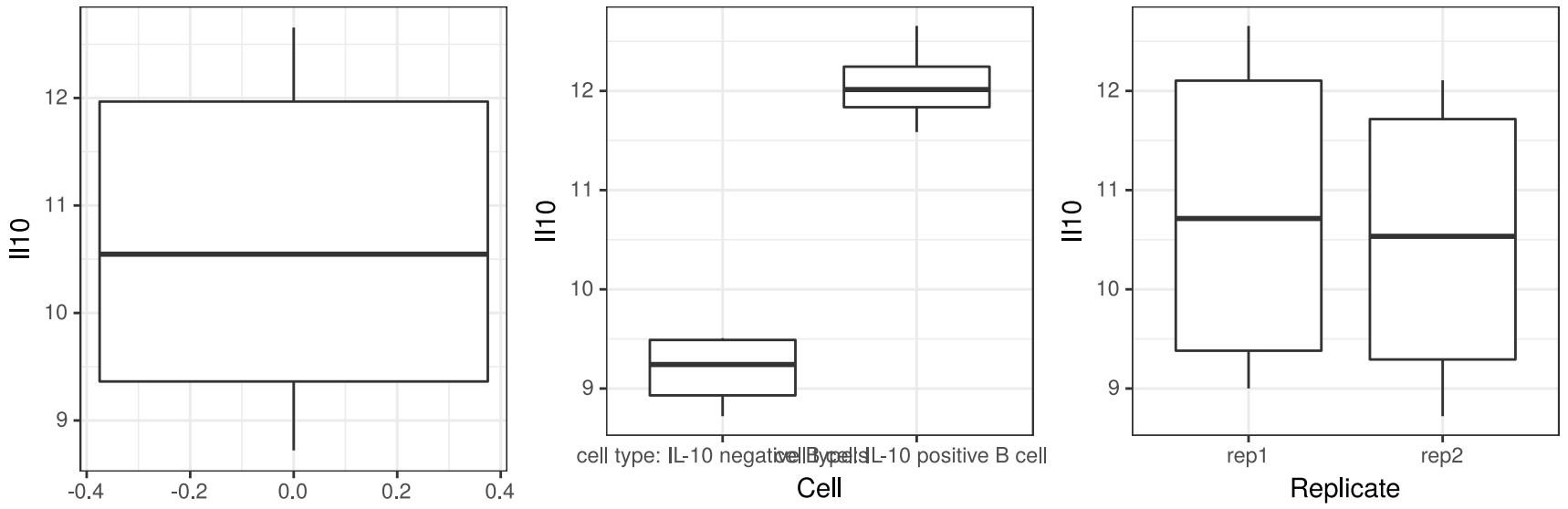
Understanding variance

```
ggplot(plotData, aes(y=Actb)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Cell, y=Actb)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Replicate, y=Actb)) +
  geom_boxplot() + theme_bw()
```



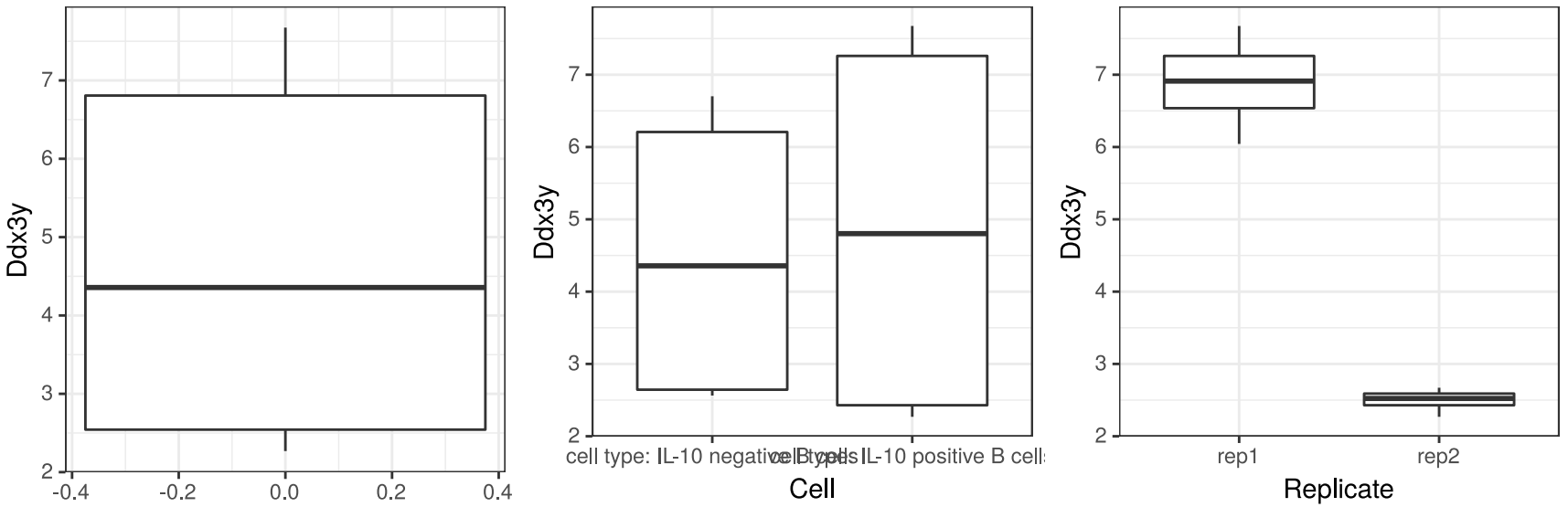
Understanding variance

```
ggplot(plotData, aes(y=Il10)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Cell, y=Il10)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Replicate, y=Il10)) +
  geom_boxplot() + theme_bw()
```



Understanding variance

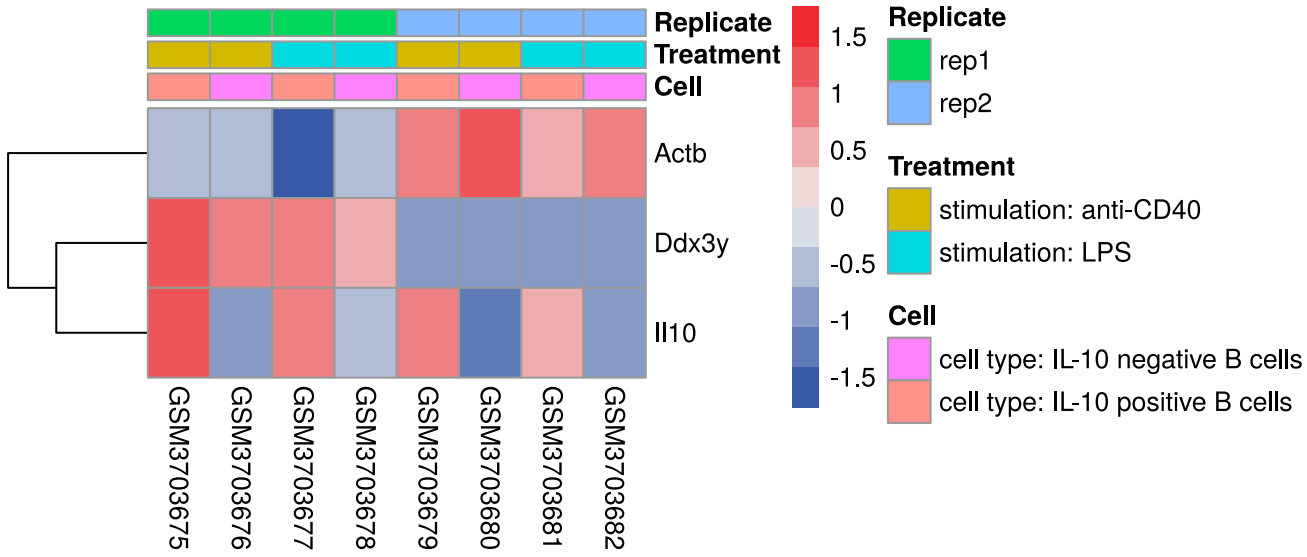
```
ggplot(plotData, aes(y=Ddx3y)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Cell, y=Ddx3y)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Replicate, y=Ddx3y)) +
  geom_boxplot() + theme_bw()
```



Heatmaps:

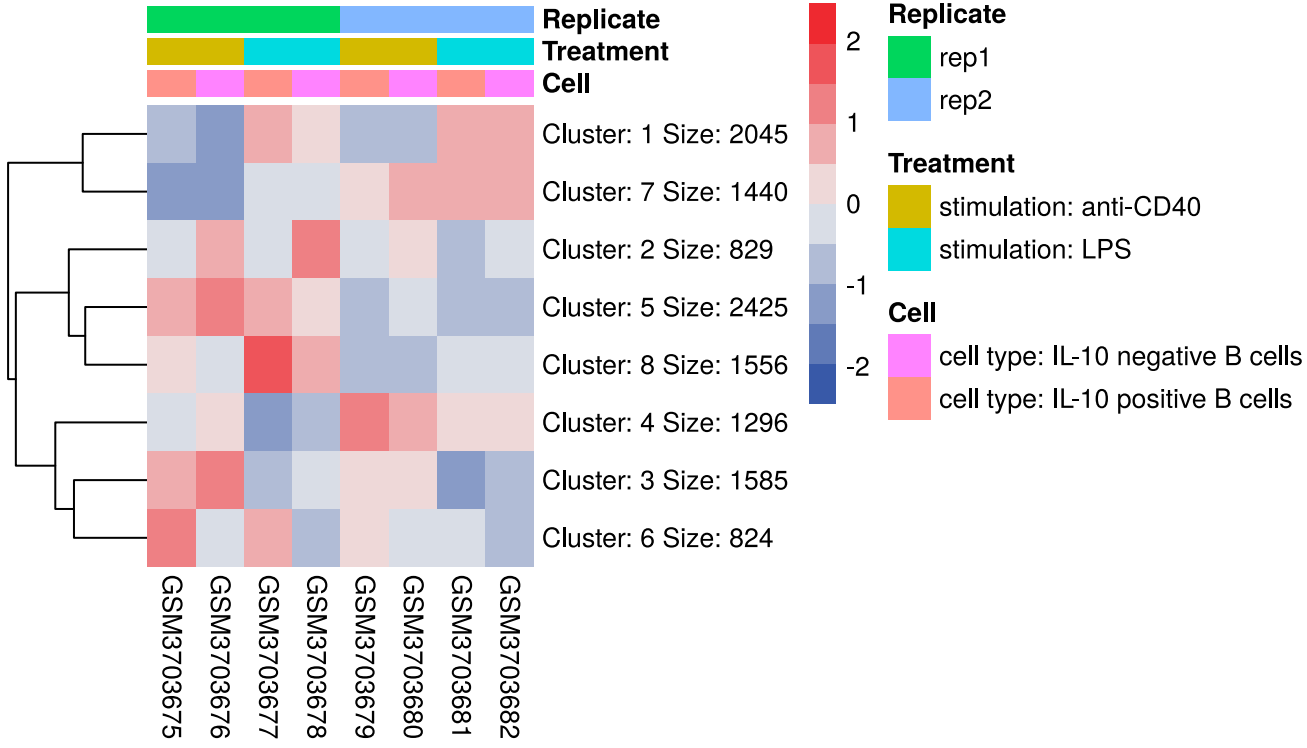
In gene set expression studies we usually use heatmaps to visualize expression levels:

```
pheatmap(someGenes, scale="row", color=blueWhiteRed, annotation_col = pData(gse129260), c
```



Heatmaps: clustered

```
pheatmap(exprs(gse129260), scale="row", color=blueWhiteRed, border_color = NA, kmeans_k =
         annotation_col = pData(gse129260), cluster_cols = F)
```

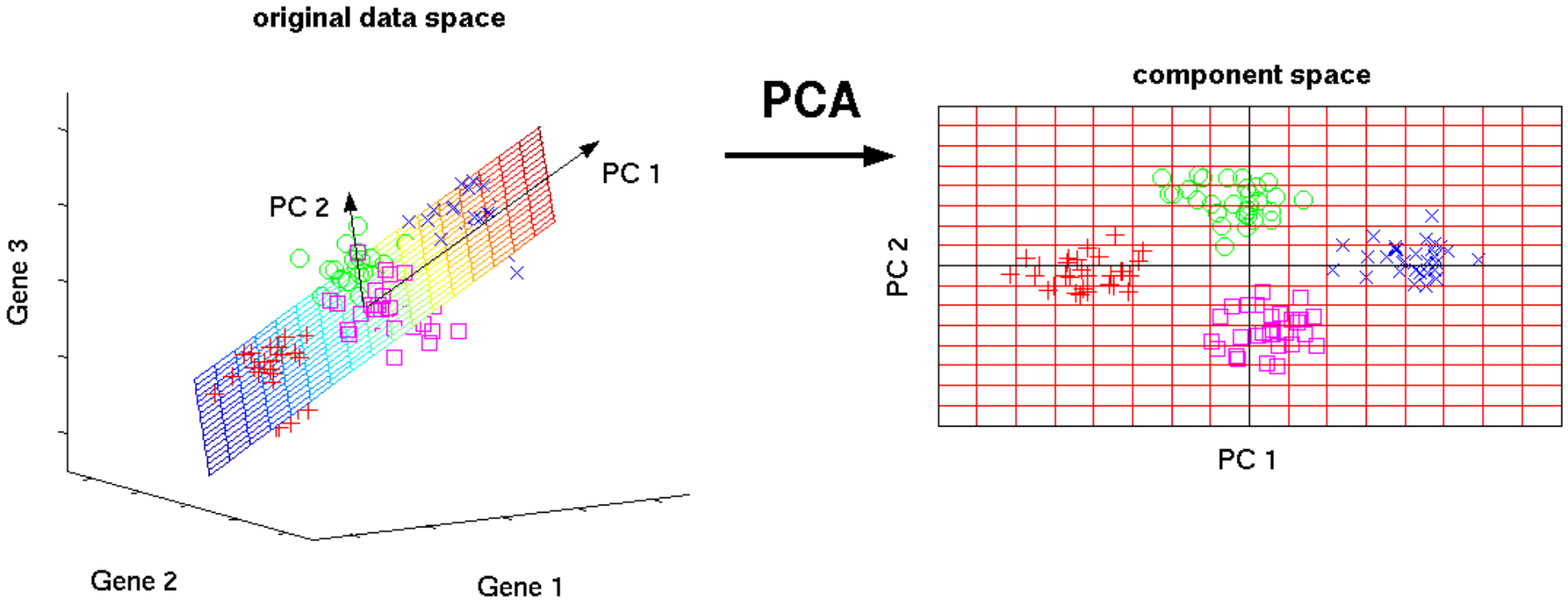


PCA: exploring the variance

PCA (principal compononent analysis):

- PCA puts samples in new space
- Principal components (new coordinates) explain as much variance as possible in original space
- Samples in original space are far from each other if a lot of genes are differentially expressed between them

PCA: exploring the variance

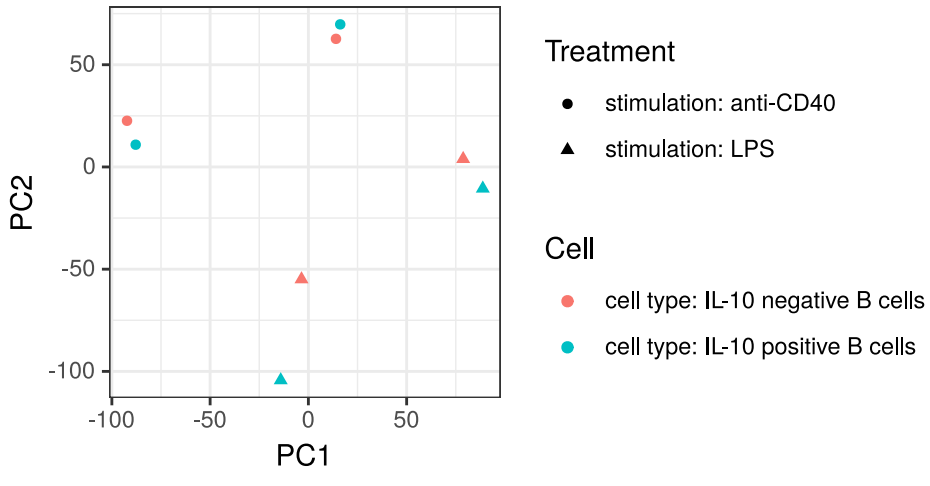


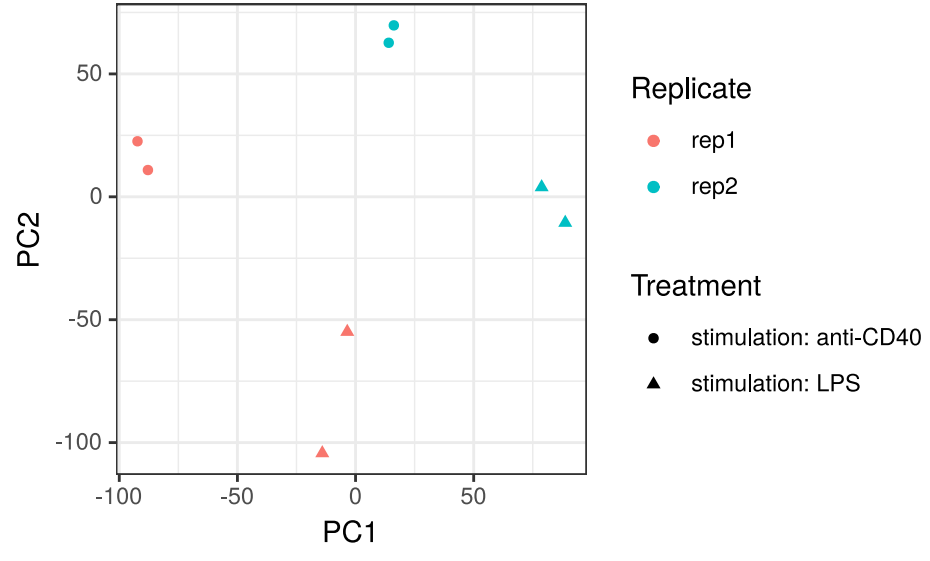
PCA: exploring the variance

```
pcas <- prcomp(t(exprs(gse129260)), scale. = T)
plotData <- cbind(pcas$x[, 1:2], pData(gse129260))
```

PCA: exploring the variance

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





PCA: exploring the variance

```
rotation <- pcas$rotation
PC1GenesDown <- head(rownames(rotation[order(rotation[, 1]), ]), 10)
PC1GenesUp <- tail(rownames(rotation[order(rotation[, 1]), ]), 10)
print(PC1GenesDown)
```

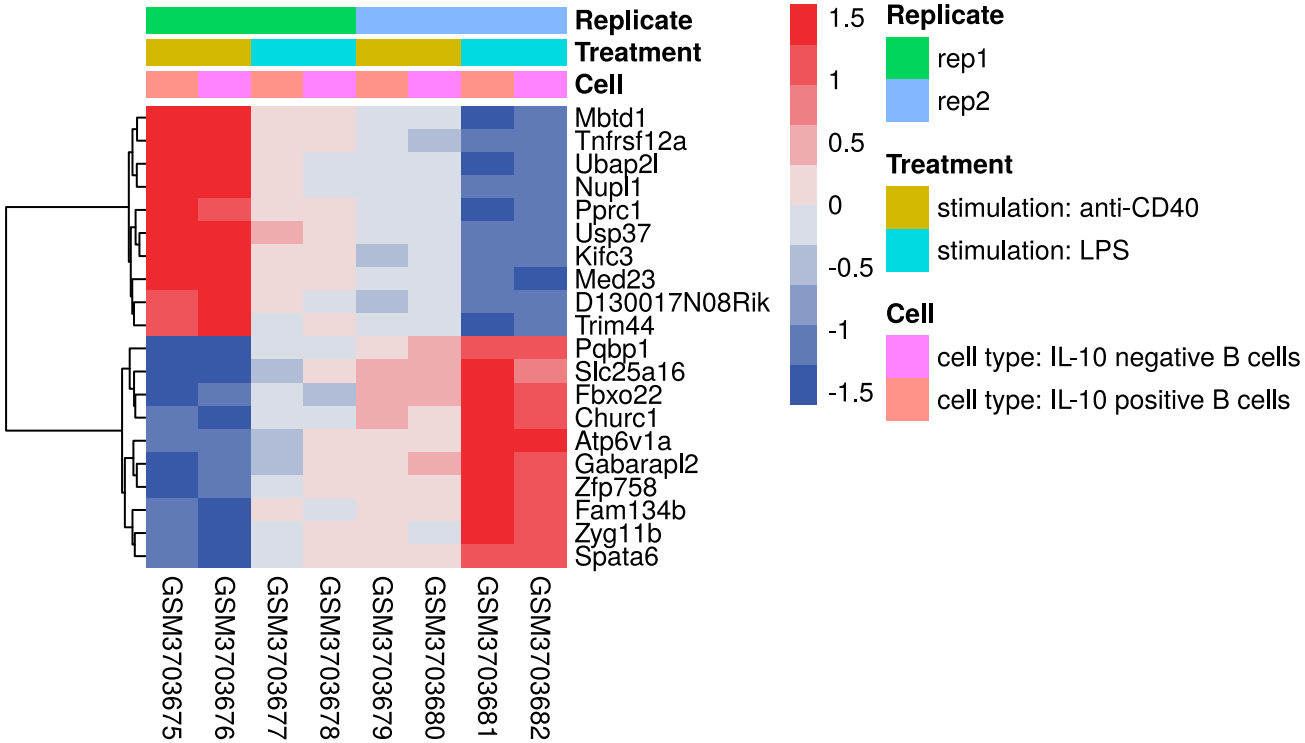
```
##  [1] "Usp37"         "Kifc3"         "Pprc1"         "Mbtd1"
##  [5] "Tnfrsf12a"     "D130017N08Rik" "Ubap2l"        "Nupl1"
##  [9] "Med23"         "Trim44"
```

```
print(PC1GenesUp)
```

```
##  [1] "Zyg11b"    "Atp6v1a"   "Fbxo22"    "Pqbp1"     "Slc25a16"
##  [6] "Gabarapl2" "Fam134b"   "Churc1"    "Zfp758"    "Spata6"
```

Heatmaps: PC1 genes

```
pheatmap(exprs(gse129260)[c(PC1GenesDown, PC1GenesUp), ],
         scale="row", color=blueWhiteRed, border_color = NA,
         annotation_col = pData(gse129260), cluster_cols = F)
```



PCA: exploring the variance

```
rotation <- pcas$rotation
PC2GenesDown <- head(rownames(rotation[order(rotation[, 2]), ]), 10)
PC2GenesUp <- tail(rownames(rotation[order(rotation[, 2]), ]), 10)
print(PC2GenesDown)
```

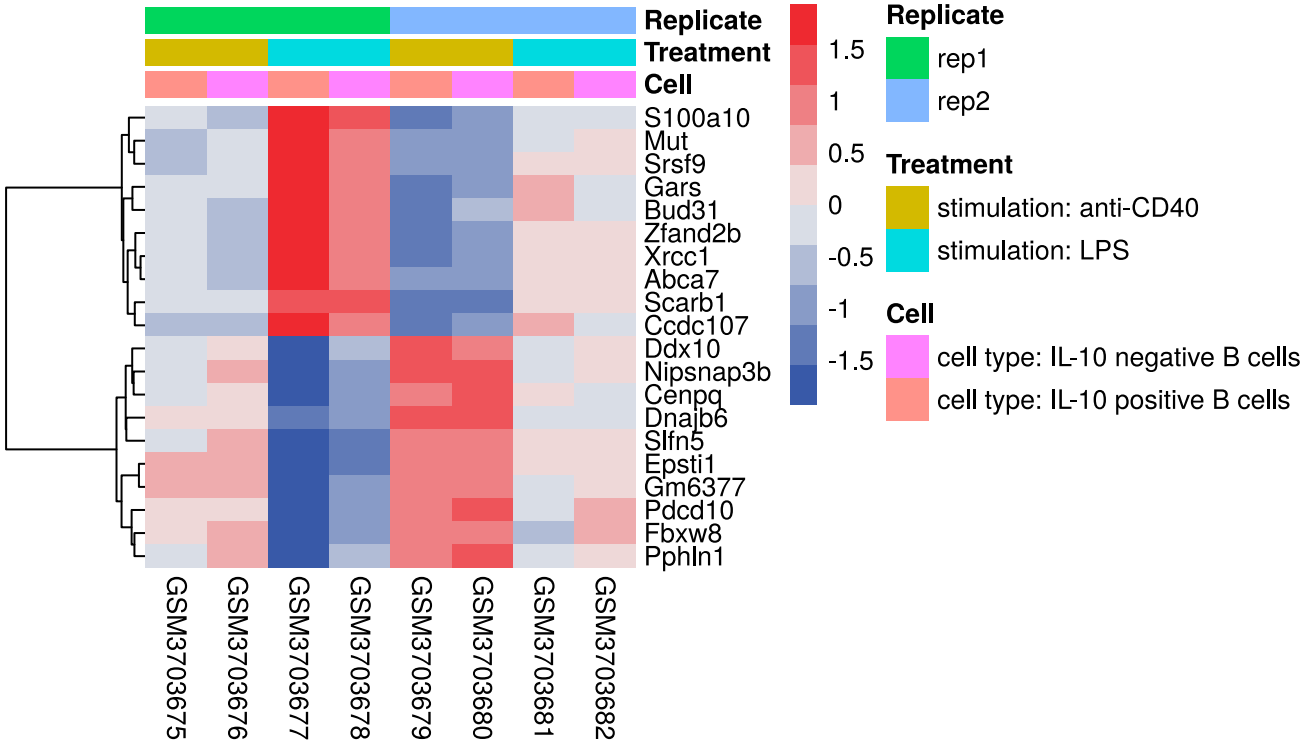
```
##  [1] "Xrcc1"   "Abca7"   "Mut"     "S100a10" "Gars"    "Scarb1"  "Zfand2b"
##  [8] "Bud31"   "Ccdc107" "Srsf9"
```

```
print(PC2GenesUp)
```

```
##  [1] "Epsti1"    "Pdcd10"    "Fbxw8"     "Dnajb6"    "Gm6377"
##  [6] "Slfn5"     "Ddx10"     "Nipsnap3b" "Pphln1"    "Cenpq"
```

Heatmaps: PC2 genes

```
pheatmap(exprs(gse129260)[c(PC2GenesDown, PC2GenesUp), ],
         scale="row", color=blueWhiteRed, border_color = NA,
         annotation_col = pData(gse129260), cluster_cols = F)
```



Can we remove unwanted sources of variance ?

In our experiment we wanted to get variance from Treatment + Cell type

- Can we identify sources of unwanted variance?
- If we know source of variance, can we remove it?

Batch correction: ComBat from SVA

SVA package:

- Allows to identify latent variables
- Allows to remove unwanted variance (ComBat)
- "Adjusting batch effects in microarray expression data using empirical Bayes methods"

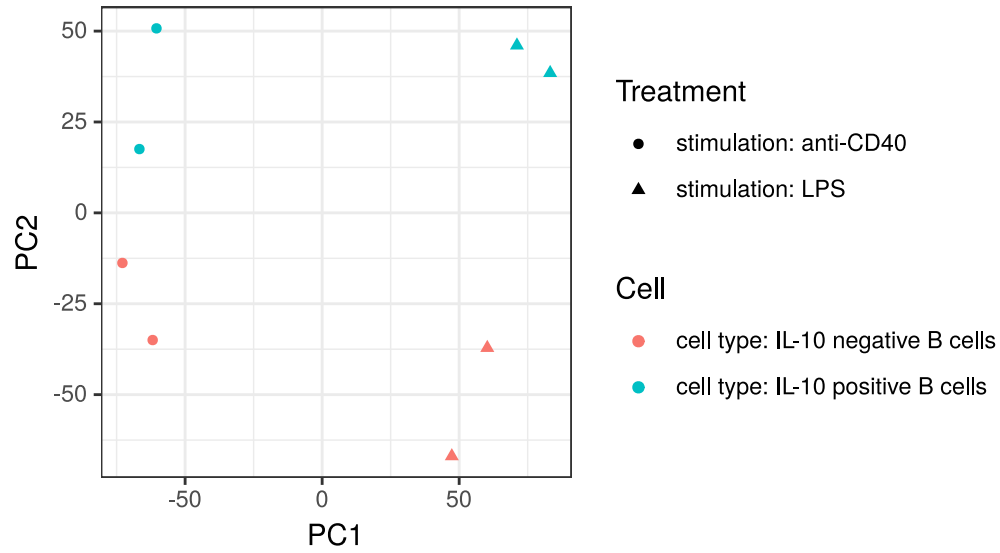
Batch correction: ComBat

```
batch <- pData(gse129260)$Replicate
modcombat <- model.matrix(~1, data=pData(gse129260))
combat_gse129260 = ComBat(dat=exprs(gse129260), batch=batch, mod=modcombat)
```

```
## Found2batches

## Adjusting for0covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding parametric adjustments

## Adjusting the Data
```

Batch correction: ComBat

```
pcas <- prcomp(t(combat_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)
```



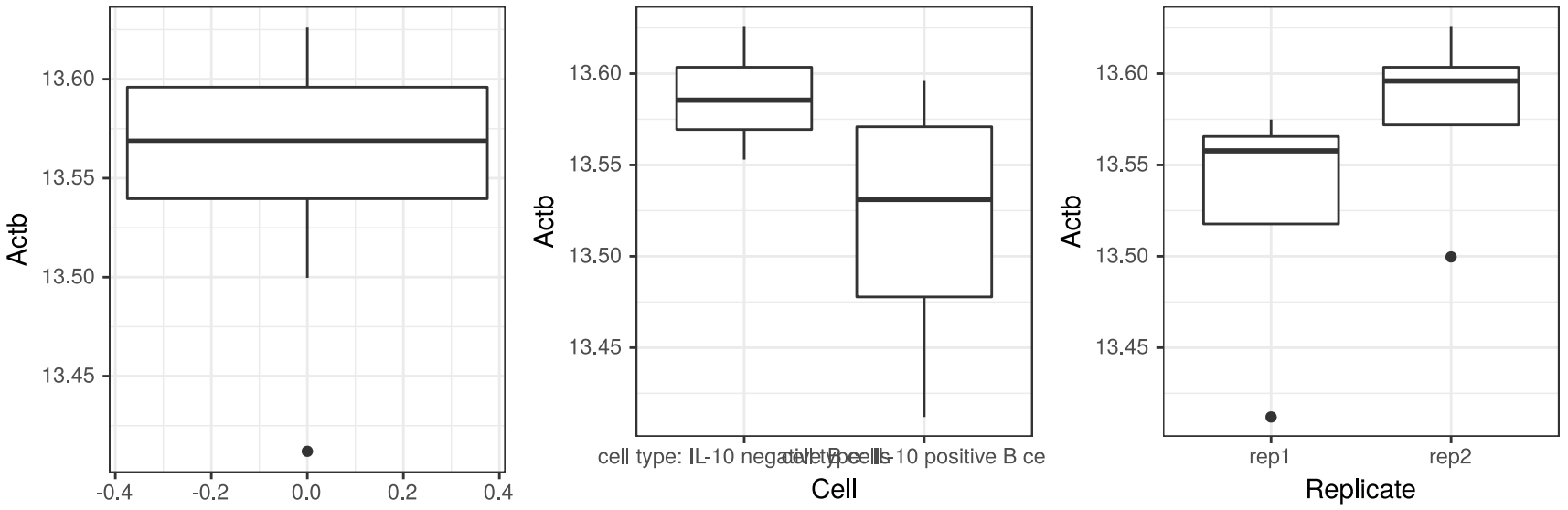
ComBat: removes batch variance

```
someGenes <- combat_gse129260[c("Actb", "Ddx3y", "Il10"), ]
plotData <- t(someGenes)
plotData <- as.data.frame(plotData)
plotData <- cbind(plotData, pData(gse129260))

head(plotData)
```

```
##                 Actb    Ddx3y      Il10                                Cell
## GSM3703675 13.56253 5.660009 12.108932 cell type: IL-10 positive B cells
## GSM3703676 13.55294 5.001689  9.132109 cell type: IL-10 negative B cells
## GSM3703677 13.41205 5.286075 11.509832 cell type: IL-10 positive B cells
## GSM3703678 13.57488 4.555441  9.545364 cell type: IL-10 negative B cells
## GSM3703679 13.59598 4.334017 12.307073 cell type: IL-10 positive B cells
## GSM3703680 13.62604 4.532145  8.903688 cell type: IL-10 negative B cells
##                          Treatment Replicate
## GSM3703675 stimulation: anti-CD40      rep1
## GSM3703676 stimulation: anti-CD40      rep1
## GSM3703677       stimulation: LPS      rep1
## GSM3703678       stimulation: LPS      rep1
## GSM3703679 stimulation: anti-CD40      rep2
## GSM3703680 stimulation: anti-CD40      rep2
```

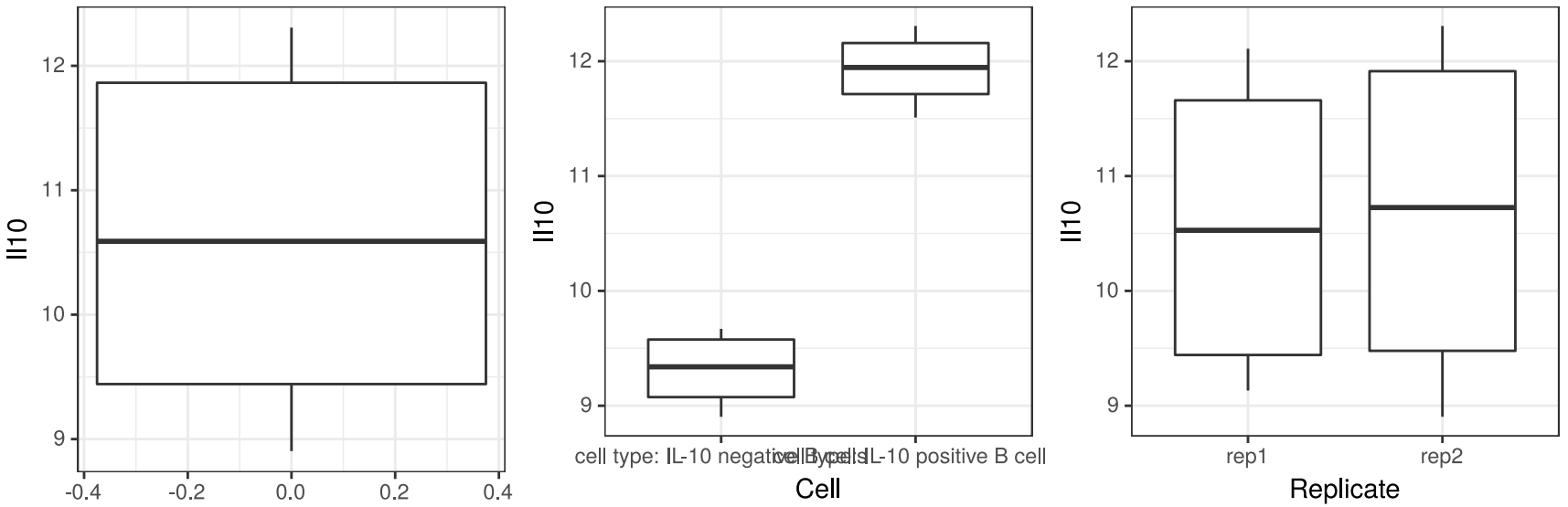
ComBat: removes batch variance

```
ggplot(plotData, aes(y=Actb)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Cell, y=Actb)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Replicate, y=Actb)) +
  geom_boxplot() + theme_bw()
```



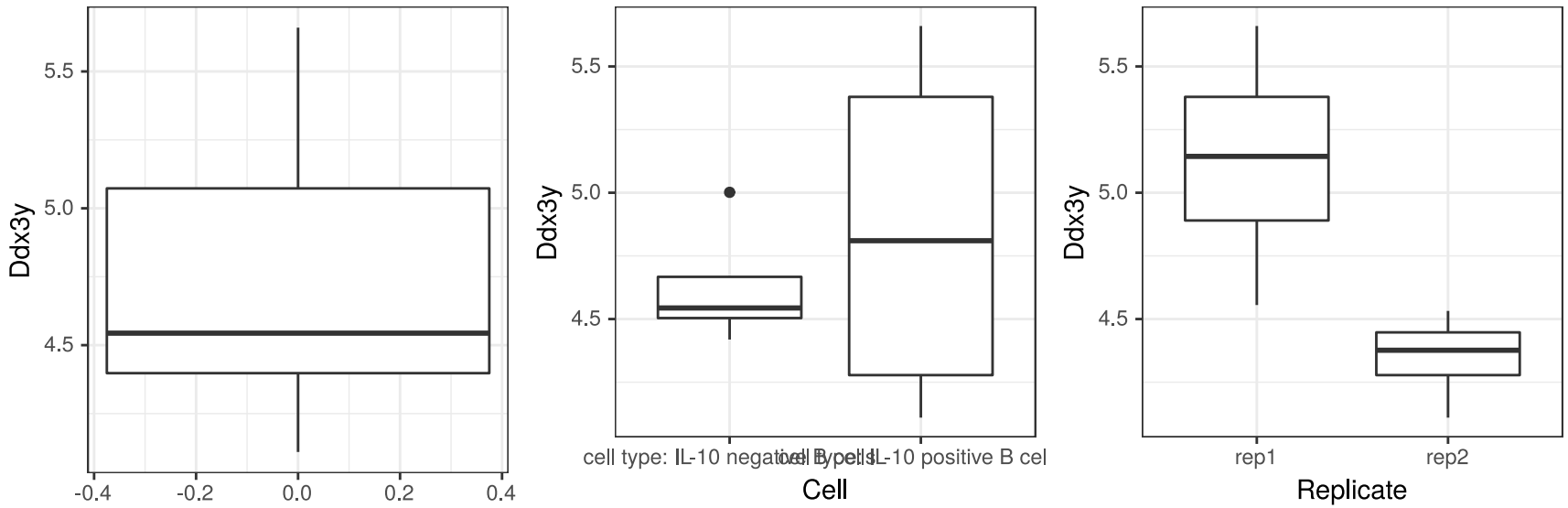
ComBat: removes batch variance

```
ggplot(plotData, aes(y=Il10)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Cell, y=Il10)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Replicate, y=Il10)) +
  geom_boxplot() + theme_bw()
```



ComBat: removes batch variance

```
ggplot(plotData, aes(y=Ddx3y)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Cell, y=Ddx3y)) +
  geom_boxplot() + theme_bw()
ggplot(plotData, aes(x=Replicate, y=Ddx3y)) +
  geom_boxplot() + theme_bw()
```



Variance

- Varince in transcriptional data comes from both signal and noise
- In ideal scenario the only source of variance is included by your experimental design
- In reality: batch effect, donor effect (super common for human data)
- We can remove unwanted sources of variance if they introduce too much variance

Once we have an expression matrix

Conceptual analysis steps are the same:

- Quality controls: PCA + outlier/batch removal if needed
- **Differential expression design**
- Performing differential expression
- DE genes: looking for possible biological pathways, transcriptional factors, regulators...

Linear models

The most simple linear models are:

$$y = kx + b$$

- We know both y and x and we try to predict k and b
- Usually both x and y are numeric

Linear models: x can be factor

Let's look at expression of gene Il10

$$y = k_{pos}x_{pos} + k_{neg}x_{neg}$$

where

- $x_{pos} = 1$ and $x_{neg} = 0$ if sample is *Il10*-positive
- $x_{pos} = 0$ and $x_{neg} = 1$ if sample is *Il10*-negative.

Linear models:

```
model_simple <- model.matrix(~0 + Cell, data = pData(gse129260))
colnames(model_simple) <- c("Negative", "Positive")
model_simple
```

```
##            Negative Positive
## GSM3703675        0        1
## GSM3703676        1        0
## GSM3703677        0        1
## GSM3703678        1        0
## GSM3703679        0        1
## GSM3703680        1        0
## GSM3703681        0        1
## GSM3703682        1        0
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Cell
## [1] "contr.treatment"
```

Linear models:

```
exprs(gse129260)["Il10", ]
```

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

```
linear_fit <- lm.fit(model_simple, exprs(gse129260)["Il10", ])
linear_fit$coef
```

```
##  Negative  Positive
##  9.178349 12.067074
```

Linear models:

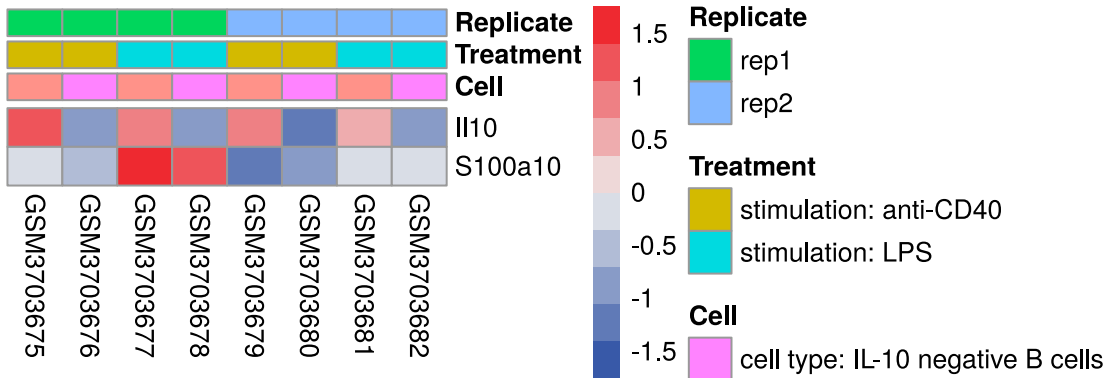
We can model expression of Il10 gene as:

- $Il10 = 9.178349$ if sample is Il10 negative
- $Il10 = 12.067074$ if sample is Il10 positive

More complicated linear models

- Il10 can be well-modeled with single factor variable
- Let's look at S100a10 gene

```
pheatmap(exprs(gse129260)[c("Il10", "S100a10"), ], scale="row", cluster_cols = F, cluster
```



Linear models: Treatment

```
treatment_model <- model.matrix(~0 + Treatment, data = pData(gse129260))
colnames(treatment_model) <- c("aCD-40", "LPS")
treatment_model
```

```
##            aCD-40 LPS
## GSM3703675      1   0
## GSM3703676      1   0
## GSM3703677      0   1
## GSM3703678      0   1
## GSM3703679      1   0
## GSM3703680      1   0
## GSM3703681      0   1
## GSM3703682      0   1
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Treatment
## [1] "contr.treatment"
```

Linear models: Treatment + Replicate

```
treatment_rep_model <- model.matrix(~0 + Treatment + Replicate, data = pData(gse129260))
colnames(treatment_rep_model) <- c("aCD-40", "LPS", "Rep2")
treatment_rep_model
```

```
##             aCD-40 LPS Rep2
## GSM3703675      1   0    0
## GSM3703676      1   0    0
## GSM3703677      0   1    0
## GSM3703678      0   1    0
## GSM3703679      1   0    1
## GSM3703680      1   0    1
## GSM3703681      0   1    1
## GSM3703682      0   1    1
## attr(,"assign")
## [1] 1 1 2
## attr(,"contrasts")
## attr(,"contrasts")$Treatment
## [1] "contr.treatment"
##
## attr(,"contrasts")$Replicate
## [1] "contr.treatment"
```

Linear models:

```
exprs(gse129260)["S100a10", ]
```

```
## GSM3703675 GSM3703676 GSM3703677 GSM3703678 GSM3703679 GSM3703680
##   9.514830   9.427012  10.549706  10.256628   8.977444   9.178550
## GSM3703681 GSM3703682
##   9.569274   9.572045
```

```
linear_fit <- lm.fit(treatment_rep_model, exprs(gse129260)["S100a10", ])
linear_fit$coef
```

```
##    aCD-40       LPS      Rep2
##  9.580817 10.293271 -0.612716
```

Linear models: Treatment + Replicate

- Including several variables in the design allows us to calculate effects for each variable
- First variable is usually a target for differential expression
- Only the first variable will have both 0/1 effect calculated

Linear models: full model

```
full_model <- model.matrix(~0 + Treatment + Cell + Replicate, data = pData(gse129260))
colnames(full_model) <- c("aCD-40", "LPS", "Il10pos", "Rep2")
full_model
```

```
##            aCD-40 LPS Il10pos Rep2
## GSM3703675      1   0       1    0
## GSM3703676      1   0       0    0
## GSM3703677      0   1       1    0
## GSM3703678      0   1       0    0
## GSM3703679      1   0       1    1
## GSM3703680      1   0       0    1
## GSM3703681      0   1       1    1
## GSM3703682      0   1       0    1
## attr(,"assign")
## [1] 1 1 2 3
## attr(,"contrasts")
## attr(,"contrasts")$Treatment
## [1] "contr.treatment"
##
## attr(,"contrasts")$Cell
## [1] "contr.treatment"
```

Linear models: full model

```
linear_fit <- lm.fit(full_model, exprs(gse129260)["Il10", ])
linear_fit$coef
```

```
##     aCD-40        LPS    Il10pos       Rep2
##  9.3251086  9.3280201  2.8887248 -0.2964297
```

```
linear_fit <- lm.fit(full_model, exprs(gse129260)["S100a10", ])
linear_fit$coef
```

```
##     aCD-40        LPS    Il10pos       Rep2
##  9.5586898 10.2711438  0.0442547 -0.6127160
```

Linear models: full model with 1

```
full_model <- model.matrix(~1 + Treatment + Cell + Replicate, data = pData(gse129260))
colnames(full_model) <- c("Intercept", "LPS", "Il10pos", "Rep2")
full_model
```

```
##             Intercept LPS Il10pos Rep2
## GSM3703675          1   0       1    0
## GSM3703676          1   0       0    0
## GSM3703677          1   1       1    0
## GSM3703678          1   1       0    0
## GSM3703679          1   0       1    1
## GSM3703680          1   0       0    1
## GSM3703681          1   1       1    1
## GSM3703682          1   1       0    1
## attr(,"assign")
## [1] 0 1 2 3
## attr(,"contrasts")
## attr(,"contrasts")$Treatment
## [1] "contr.treatment"
##
## attr(,"contrasts")$Cell
## [1] "contr.treatment"
```

Linear models: full model with 1

```
linear_fit <- lm.fit(full_model, exprs(gse129260)["Il10", ])
linear_fit$coef
```

```
##    Intercept          LPS      Il10pos         Rep2
##  9.325108606  0.002911487  2.888724826 -0.296429707
```

```
linear_fit <- lm.fit(full_model, exprs(gse129260)["S100a10", ])
linear_fit$coef
```

```
##  Intercept        LPS    Il10pos       Rep2
##  9.5586898  0.7124539  0.0442547 -0.6127160
```

Linear models

- Linear models are usefull for calculating effects of variables
- `~ 0 + Annotation1 + Annotation2 ...` will calculate means for both factors in Annotation1, and calculate effects for other annotations (excluding Annotation1)
- `~ 1 + Annotation1 + Annotation2 ...` will calculate means for one the Annotation1 factors, and calculate effects for all annotations (including remaining factor in Annotation1)

Once we have an expression matrix

Conceptual analysis steps are the same:

- Quality controls: PCA + outlier/batch removal if needed
- Differential expression design
- **Performing differential expression**
- DE genes: looking for possible biological pathways, transcriptional factors, regulators…

Performing DE (differential expression)

- Performing DE is usually much easier than designing proper DE :)
- We will use limma in this example
- Let's check which design get more results

Performing limma

```
cell_full_model <- model.matrix(~0 + Cell + Treatment + Replicate, data=pData(gse129260))
colnames(cell_full_model) <- c("il10neg", "il10pos", "LPS", "rep2")

fit <- lmFit(gse129260, cell_full_model)

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

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

Differential expression

```
head(de)
```

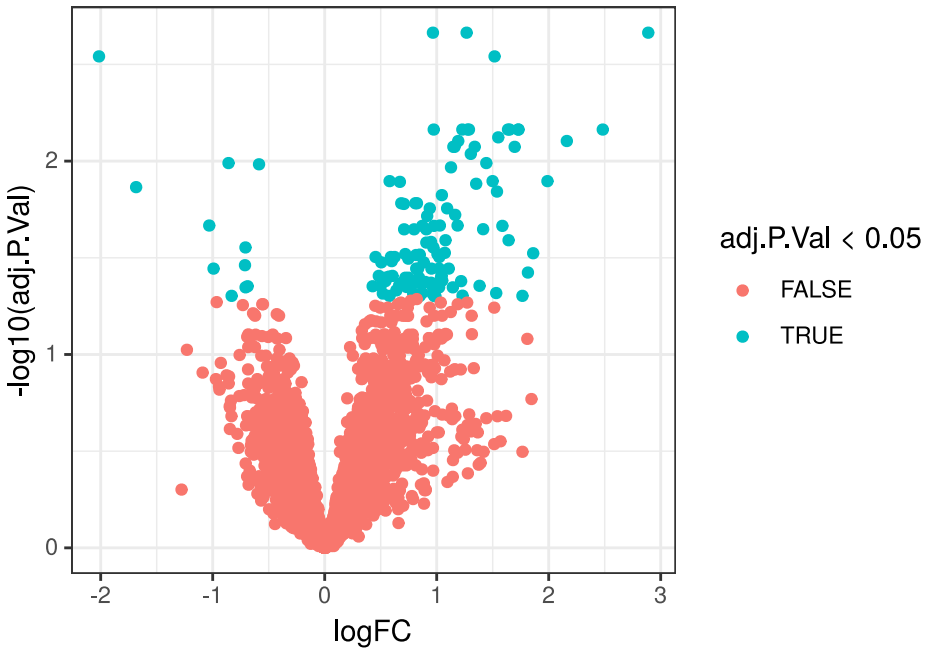
```
##                    ID Gene.symbol Gene.ID mean_expression      logFC
## Cd9         1416066_at         Cd9   12527        9.408102  1.2672152
## Il10        1450330_at        Il10   16153       10.622712  2.8887248
## Mzb1        1428947_at        Mzb1   69816       10.836028  0.9663412
## Pon3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

Volcano plot

- Volcano plot is the usual way to display DE results
- X axis is log fold change showing the direction of the change
- Y axis is -log10(p adjusted) - showing the significane of DE

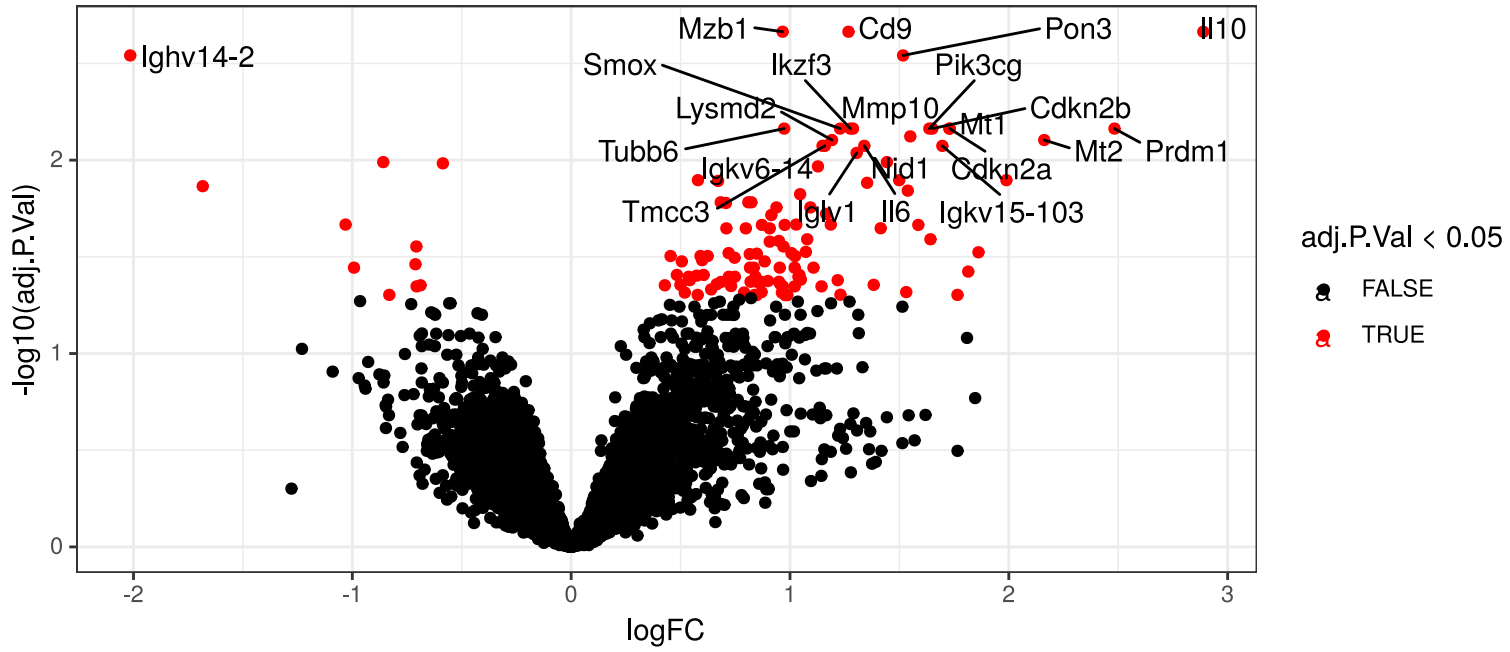
DE: Volcano plot

```
ggplot(de, aes(x=logFC, y=-log10(adj.P.Val), color=adj.P.Val < 0.05)) +
  geom_point() + theme_bw()
```



DE: Volcano plot (fancy)

```
ggplot(de, aes(x=logFC, y=-log10(adj.P.Val), color=adj.P.Val < 0.05)) +
  geom_point() + theme_bw() + scale_color_manual(values=c("black", "red")) +
  geom_text_repel(data=de %>% dplyr::filter(adj.P.Val < 0.01), aes(label=Gene.symbol, col
```



Performing limma: bad model

```
cell_bad_model <- model.matrix(~0 + Cell, data=pData(gse129260))
colnames(cell_bad_model) <- c("il10neg", "il10pos")

fit_bad <- lmFit(gse129260, cell_bad_model)

fit_bad2 <- contrasts.fit(fit_bad, makeContrasts(il10pos - il10neg, levels=cell_bad_model
fit_bad2 <- eBayes(fit_bad2, trend = T)

de_bad <- topTable(fit_bad2, adjust.method="BH", number=Inf, sort.by = "P")
```

Comparing results

```
de %>% filter(adj.P.Val < 0.05) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   124
```

```
de_bad %>% filter(adj.P.Val < 0.05) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1     1
```

Differential expression

- Good design empowers you to find differences that you want to find in your data
- Design with a smaller number of variables ignores additional information and assumes samples in a group to be the same
- Design with a smaller number of variables is still something you might want to do

Once we have an expression matrix

Conceptual analysis steps are the same:

- Quality controls: PCA + outlier/batch removal if needed
- Differential expression design
- Performing differential expression
- **DE genes: looking for possible biological pathways, transcriptional factors, regulators…**

Pathway enrichment:

- In most cases gene expression changes are not coming one gene by one
- Genes that are changed are regulated by biological processes (pathways)
- We have bunch of databases that describe gene sets: sets of genes that regulate or regulated by biological process
- Technically speaking, for us pathway is just a set of genes

Pathway enrichment:

- Pathways
- GO terms
- Targets of transcriptional factors
- Gene sets produced by other datasets

Pathway enrichment:

Let's didcuss terms firts:

- Universe: genes that are expressed in the dataset (in our case size of the universe 12000):

$$U = \{g_1, g_2, \ldots, g_n\}, \quad |U| = n \approx 12000$$

- We have N pathways:

$$P_i = \{g_{i,1}, g_{i,2}, \ldots, g_{i,m_i}\}$$
$$|P_i| = m_i, \quad g_{i,j} \in U$$

Pathway enrichment:

- We have results of our DE
- "Pathway behaves non-random" = "Genes from pathways are not changing randomly"
- We want to identify pathways that behave non-random in our DE results

Simple implementation: exact Fisher test

Exact Fisher test (or hypergeometric test)

- We only choose significantly expressed genes
- We test overlaps of these genes with pathway
- Identify if overlap is random

Loading kegg Pathways

```
load("keggSymbolMouse.rdata")
upRegulatedGenes <- de %>% filter(adj.P.Val < 0.05 & logFC > 0) %>% pull("Gene.symbol")
length(upRegulatedGenes)
```

```
## [1] 113
```

```
randomGeneSet <- keggSymbolMouse[["Cardiac muscle contraction - Mus musculus (mouse)"]]
randomGeneSet <- randomGeneSet[randomGeneSet %in% rownames(de)]
length(randomGeneSet)
```

```
## [1] 41
```

```
length(intersect(randomGeneSet, upRegulatedGenes))
```

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

Hypergeometric test

https://en.wikipedia.org/wiki/Hypergeometric_distribution

- N = 12000: total number of genes (TOTAL)
- K = 41: number of genes in pathway (SUCCESSES)
- n = 113: number of DE genes (DRAWS)
- k = 1: overlap (SUCCESSFUL DRAWS)

Null hypothesis -- genes are drawn from 12000 genes at random with respect to the pathway

Running hypergeometric test

```
N <- nrow(de)
K <- length(randomGeneSet)
n <- length(upRegulatedGenes)
k <- length(intersect(upRegulatedGenes, randomGeneSet))
phyper(k - 1, K, N - K, n, lower.tail = F)
```

```
## [1] 0.32197
```

Non-random set

```
nonRandomGeneSet <- keggSymbolMouse[["Cytokine-cytokine receptor interaction - Mus muscul
nonRandomGeneSet <- nonRandomGeneSet[nonRandomGeneSet %in% rownames(de)]


N <- nrow(de)
K <- length(nonRandomGeneSet)
n <- length(upRegulatedGenes)
k <- length(intersect(upRegulatedGenes, nonRandomGeneSet))
print(c(N, K, n, k))
```

```
## [1] 12000   135   113     7
```

```
phyper(k - 1, K, N - K, n, lower.tail = F)
```

```
## [1] 0.000279903
```

Hypergeometric tests

- Require you to define gene set to test:
  - Setting arbitrary threshold (< 0.01 or < 0.05)
  - Only work with decent amount of genes (hard to calculate overlaps for 20 genes)
- Very robust with large number of genes
- Many databases offer you hypergeometric test with FDR correction (multiple pathways tests)
- http://software.broadinstitute.org/gsea/msigdb/annotate.jsp (you can enter my email for now: zayats1812@mail.ru)

Hypergeometric tests

- http://software.broadinstitute.org/gsea/msigdb/annotate.jsp (you can enter my email for now: zayats1812@mail.ru)

```
cat(upRegulatedGenes)
```

```
## Cd9 Il10 Mzb1 Pon3 Mt1 Tubb6 Smox Mmp10 Ikzf3 Pik3cg Prdm1 Cdkn2a Cdkn2b Nid1 Lysmd2 Mt2 Il6 Igkv6
```

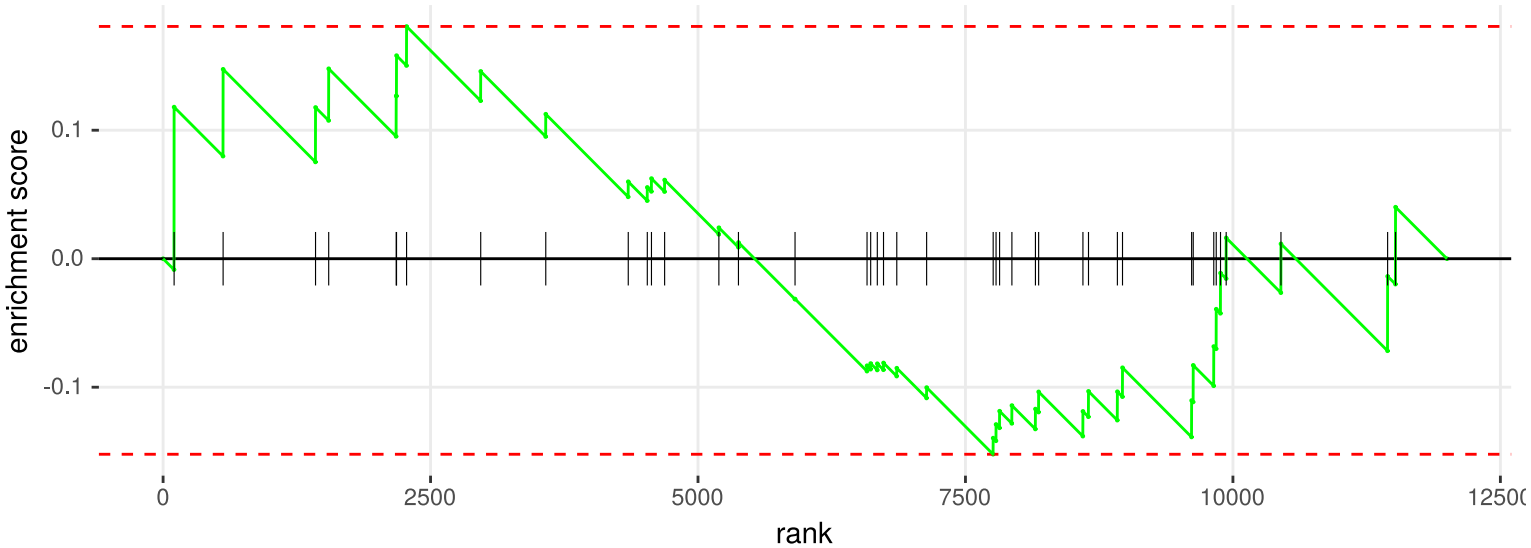
GSEA (gene-set enrichment analysis)

- GSEA uses information about all genes in DE to score pathways
- Genes are ranked by their difference in DE (usually by t statistic)
- We try to identify pathways for which genes are distributed at random

GSEA (gene-set enrichment analysis)

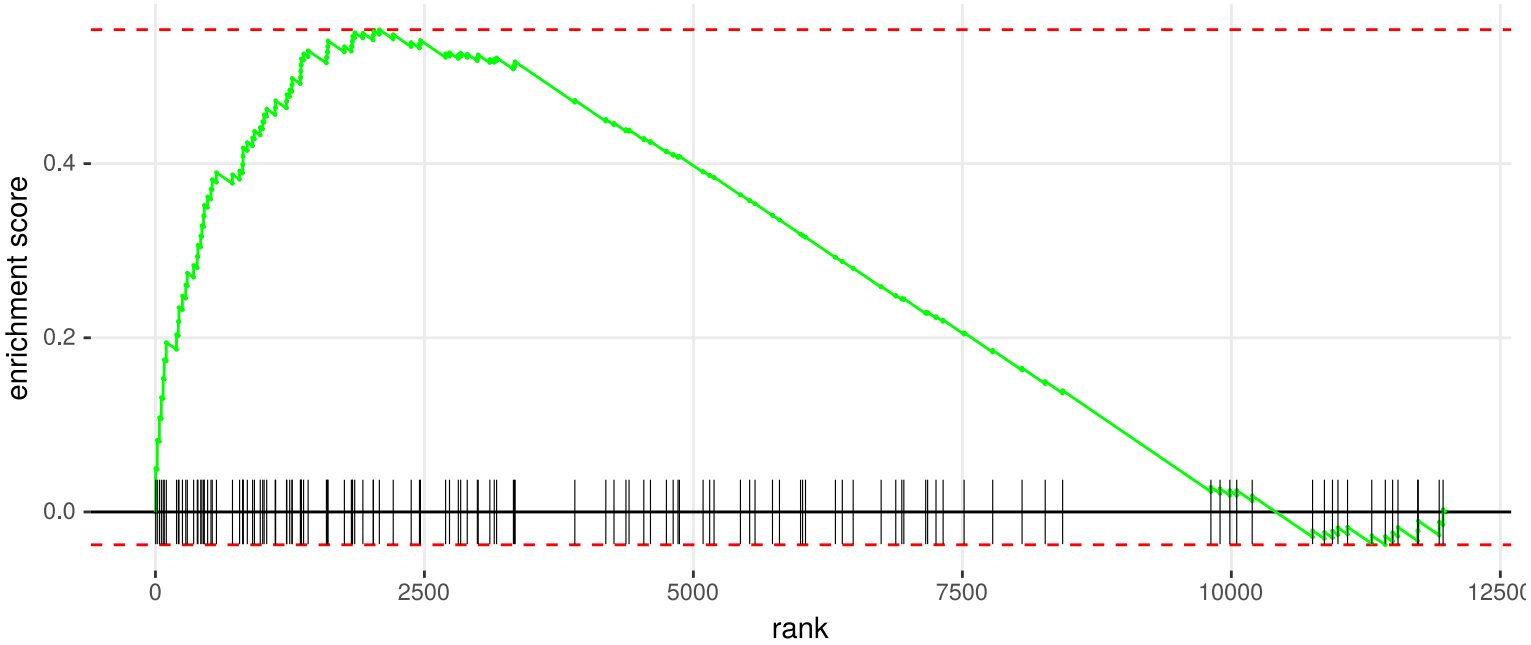
We will use fgsea package (the guys in our lab are amazing):

```
stats <- de$t
names(stats) <- de$Gene.symbol
plotEnrichment(randomGeneSet, stats)
```



GSEA (gene-set enrichment analysis)

```
plotEnrichment(nonRandomGeneSet, stats)
```



FGSEA

- Based on enrichment score we can calculate p value for each pathway
- fgsea allows us to do it quickly (f for FAST) for all the pathways in the same time

FGSEA

```
fgseaResults <- fgseaMultilevel(keggSymbolMouse, stats, minSize = 15, maxSize = 500)
head(fgseaResults, 3)
```

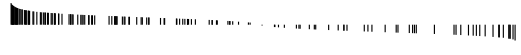
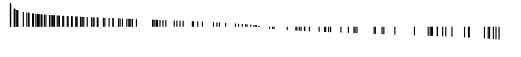
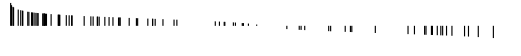
```
##                                                                    pathway
## 1:                                    ABC transporters - Mus musculus (mouse)
## 2: AGE-RAGE signaling pathway in diabetic complications - Mus musculus (mouse)
## 3:                                AMPK signaling pathway - Mus musculus (mouse)
##            pval       padj   log2err        ES       NES size
## 1: 0.142384106 0.22833549 0.1501629 0.4237276 1.2876147   25
## 2: 0.001864951 0.00925438 0.4550599 0.4582477 1.6770374   66
## 3: 0.941176471 0.94571807 0.0343128 0.1846762 0.7202221   91
##                                leadingEdge
## 1:    Abcg1,Abcc1,Abca1,Abcb1b,Abca7,Abcd1,...
## 2:          Il6,Prkcz,Foxo1,Pim1,Jak2,Ccl2,...
## 3: Foxo1,Igf1r,Rab2a,Creb3l2,Cpt1a,Ppp2r5a,...
```

FGSEA

```
topPathwaysUp <- fgseaResults[ES > 0, ][head(order(pval), n=5), pathway]
topPathwaysDown <- fgseaResults[ES < 0, ][head(order(pval), n=5), pathway]
topPathways <- c(topPathwaysUp, rev(topPathwaysDown))
```

FGSEA

```
plotGseaTable(keggSymbolMouse[topPathways], stats, fgseaResults, gseaParam = 0.5)
```

Pathway	Gene ranks	NES	pval	padj
ine-cytokine receptor interaction - Mus musculus (mouse)		2.31	3.4e-11	4.4e-09
essing in endoplasmic reticulum - Mus musculus (mouse)		2.18	3.6e-09	3.1e-07
associated herpesvirus infection - Mus musculus (mouse)		2.06	9.1e-08	6.0e-06
Pathways in cancer - Mus musculus (mouse)		1.75	1.5e-06	4.8e-05
TNF signaling pathway - Mus musculus (mouse)		2.09	1.8e-06	5.2e-05
Proteasome - Mus musculus (mouse)		-2.20	7.8e-06	1.5e-04
Nucleotide excision repair - Mus musculus (mouse)		-2.31	7.5e-07	2.8e-05
RNA transport - Mus musculus (mouse)		-1.98	3.1e-07	1.4e-05
osome biogenesis in eukaryotes - Mus musculus (mouse)		-2.28	2.7e-07	1.4e-05
Spliceosome - Mus musculus (mouse)		-2.75	5.8e-16	1.5e-13

FGSEA:

- Does not require a priori threshold to define DE genes
- Can detect a lot of small changes
- Detects up/down pathways in the same time

Conclusion

Secondary analysis is about making sense of the data:

- Making sense of the variance
- Making sense of differential expression design
- Making sense of differential expression results

Once we have differential expression we can try to guess what's going on:

- Biological pathways
- TF targets

Questions?