Feedback on RNA-seq data Analyses (Part 1)

Lijiao NING (lijiao.ning@cnrs.fr)
Mathilde BOISSEL (mathilde.boissel@cnrs.fr)

2019-05-28

RNA sequencing

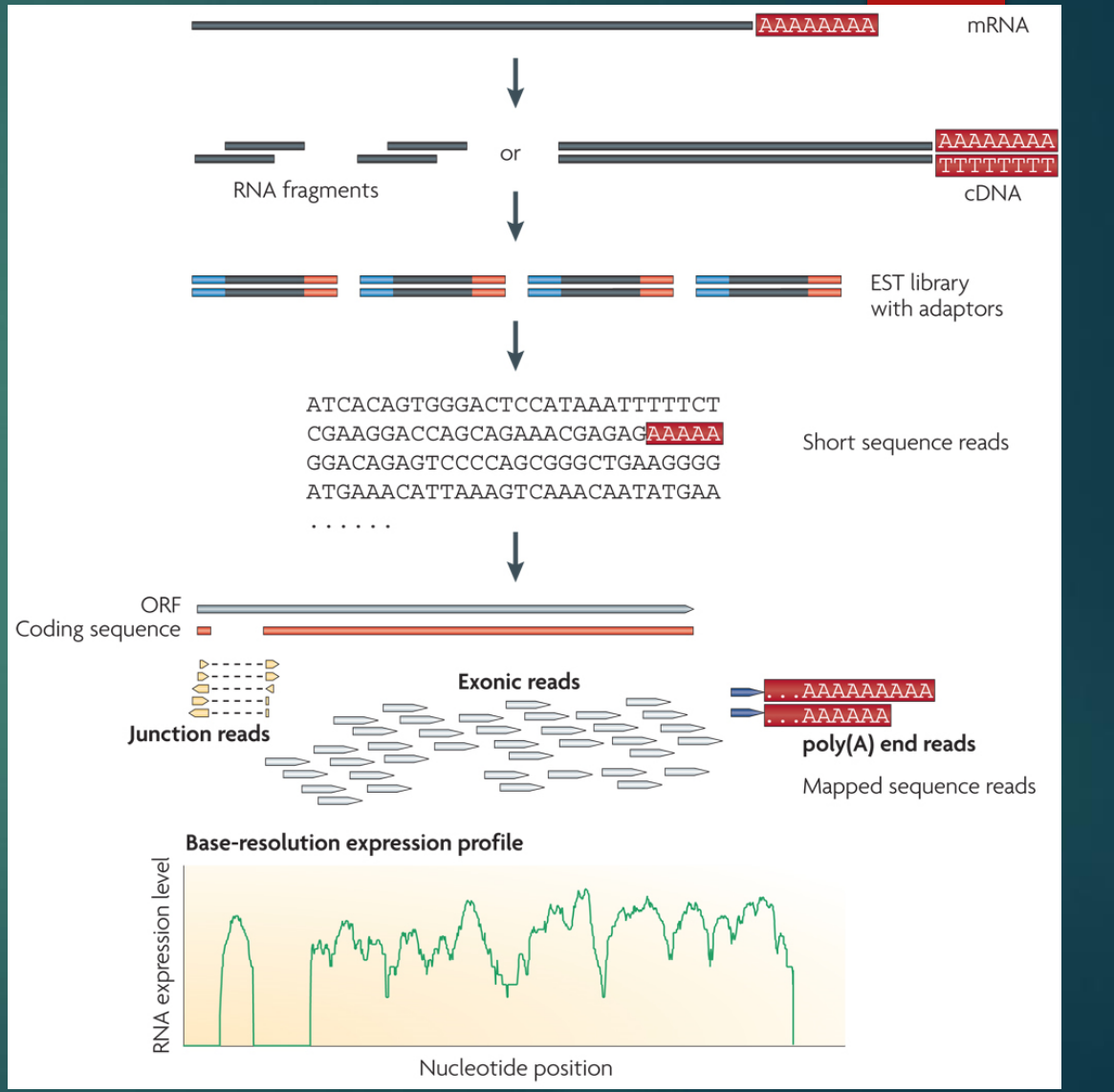
- RNA-Seq:

Identify and quantify of RNA in a biological sample at given moment

- Results (bioInfo)

RSEM : "a software package for estimating gene and isoform expression levels from RNA-Seq data".

⇒ Gene and isoform (transcript) quantification files



(Source Wang, Gerstein and Snyder, Nat Rev Genet. 2009 Jan; 10(1): 57–63.)

RSEM file

- Example of sample's isoforms count file

transcript_id	gene_id	length	effective_length	expected_count	TPM	FPKM	IsoPct
ENST00000373020	ENSG00000000003	2206	2047.38	1545.26	26.63	18.45	95.95
ENST00000494424	ENSG00000000003	820	661.38	0.00	0.00	0.00	0.00
ENST00000496771	ENSG00000000003	1025	866.38	1.95	0.08	0.05	0.29
ENST00000612152	ENSG00000000003	3796	3637.38	107.79	1.05	0.72	3.77
ENST00000614008	ENSG00000000003	900	741.38	0.00	0.00	0.00	0.00
ENST00000373031	ENSG00000000005	1339	1180.38	1.56	0.05	0.03	25.90
ENST00000485971	ENSG00000000005	542	383.45	1.44	0.13	0.09	74.10
…	…	…	…	…	…	…	…

Expression units: TPM

- Transcripts Per kilobase Million

(1) Divide the read counts by the length of each gene in kilobase

=> reads per kilobase (RPK)

(2) Count up all the RPK values in a sample and divide this number by 1,000,000 (here 10)

=> "per million" scaling factor

(3) Divide the RPK values by the "per million" scaling factor

N.B.: the sum of all TPMs in each sample are the same.

Gene (length)	Sample1	Sample2	Sample3
A (2kb)	100	12	300
B (4kb)	20	25	60
C (1kb)	5	8	15
D (10kb)	0	0	1
Total	**125**	**45**	**376**

Length normalized total counts / 10

Gene (length)	Sample1	Sample2	Sample3
A (2kb)	$8.33 = \frac{100}{2} / \frac{\frac{100}{2}+\frac{20}{4}+\frac{5}{1}+\frac{0}{10}}{10}$	2.96	8.32
B (4kb)	0.83	3.09	0.83
C (1kb)	0.83	3.95	0.83
D (10kb)	0	0	0.006
Total	**10**	**10**	**10**

Source https://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/

Expression units: RPKM

- Reads Per Kilobase Million (single-end)

(1) Count up the total reads in a sample and divide that number by 1,000,000 (here 10)

=> "per million" scaling factor

(2) Divide the read counts by the "per million" scaling factor

=> reads per million (RPM)

(3) Divide the RPM values by the length of the gene

N.B: FPKM (Fragments Per Kilobase Million) is made for paired-end RNA-seq

Gene (length)	Sample1	Sample2	Sample3
A (2kb)	100	12	300
B (4kb)	20	25	60
C (1kb)	5	8	15
D (10kb)	0	0	1
Total	**125**	**45**	**376**

Total counts / 10

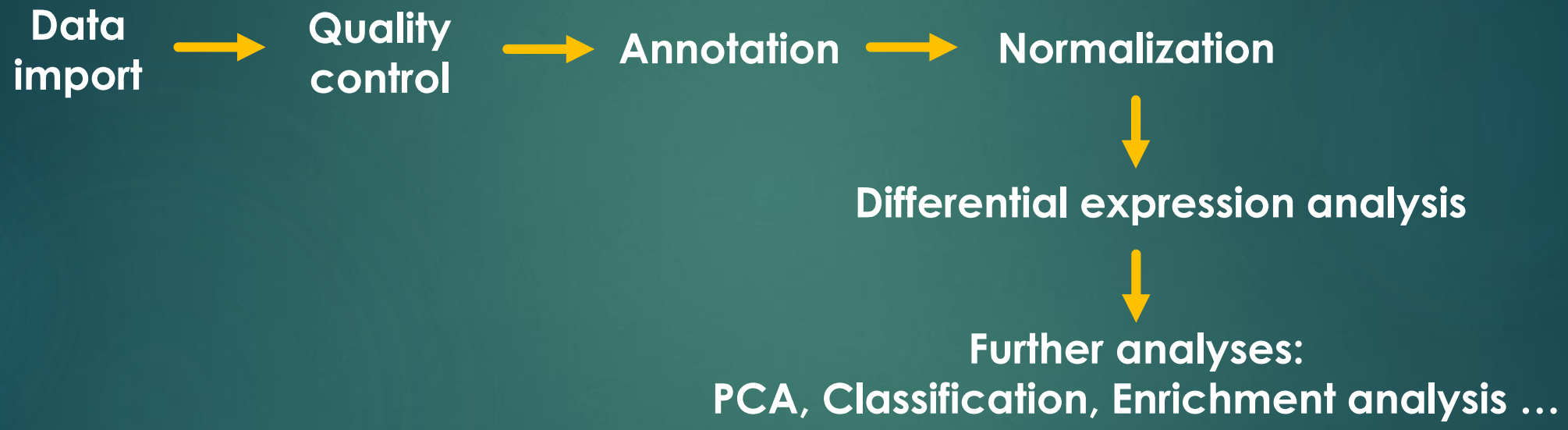
Gene (length)	Sample1	Sample2	Sample3
A (2kb)	4=100/((100+20+5+0)/10)/2	1.33	3.99
B (4kb)	0.4	1.39	0.40
C (1kb)	0.4	1.78	0.40
D (10kb)	0	0	0.0027
Total	**4.8**	**4.5**	**4.79**

Source Mortazavi et al. (2008) "Mapping and quantifying mammalian transcriptomes by RNA-Seq" Nat. Meth. 5
Source https://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/

Analysis tools

- Bioconductor Project
  - Collection of R packages for high throughput genomic data analysis
  - Free access
- Some packages :
  - { tximport }
    Import and summarize transcript-level estimates for transcript- and gene-level analysis
  - { DESeq2 }, { edgeR }, { limma }
    Differential gene expression analysis
  - { biomaRt }
    Interface to BioMart databases, e.g.: Ensembl
  - { ggplot2 }
    data visualization

Analyses Pipeline



Import Data

- Sample sheet

```
suppressMessages(library(tidyverse))
## read sample sheet and add file path
df_sample <- rbind(
  utils::read.csv("/disks/RUN/run_364/samplesheet.csv") %>%
    mutate(run = "364"),
  utils::read.csv("/disks/RUN/run_365/samplesheet.csv") %>%
    mutate(run = "365")
) %>%
  mutate(
    concentration = gsub(" mM glucose", "", Description) %>%
      as.factor(.) %>%
      relevel(., ref = "5"),
    gene_path = paste0(
      "/disks/RUN/run_", run, "/rnaseq_analysis_noTrim/Sample_", SampleID,
      "/RSEM/", SampleID, ".genes.results"
    ),
    isoform_path = gsub("genes", "isoforms", gene_path),
    FileExists_Genes = file.exists(gene_path),
    FileExists_Isoforms = file.exists(isoform_path)
  ) %>%
  as_tibble()
```

Import Data

- Sample sheet

```
head(df_sample)
## # A tibble: 6 x 16
##   FCID   Lane SampleID SampleRef Index Description Control Recipe Operator
##   <fct> <int> <fct>    <fct>     <fct> <fct>       <fct>   <fct>  <fct>
## 1 H5NC…     1 hum57-2  humain    GAAT… 5 mM gluco… Y       " 2x7… ed
## 2 H5NC…     1 hum57-3  humain    GAAT… 8 mM gluco… Y       multi… ed
## 3 H5NC…     1 hum57-4  humain    GAAT… 20 mM gluc… Y       multi… ed
## 4 HKFC…     1 mbe4-2   humain    ATTC… 5 mM gluco… Y       " 2x7… ed
## 5 HKFC…     1 mbe4-3   humain    ATTC… 8 mM gluco… Y       multi… ed
## 6 HKFC…     1 mbe4-4   humain    ATTC… 20 mM gluc… Y       multi… ed
## # … with 7 more variables: SampleProject <fct>, run <chr>,
## #   concentration <fct>, gene_path <chr>, isoform_path <chr>,
## #   FileExists_Genes <lgl>, FileExists_Isoforms <lgl>
```

Import Data

- Sample sheet
- RSEM files (by gene or transcript)

```
## import count data
my_files <- df_samplcount datae$gene_path
names(my_files) <- df_sample$SampleID ## to keep samples names
dta_raw <- tximport::tximport(
  files = my_files,
  type = "rsem",
  txIn = FALSE,
  txOut = FALSE,
  countsFromAbundance = "no",
  geneIdCol = "gene_id",
  abundanceCol = "TPM",
  countsCol = "expected_counts",
  lengthCol = "effective_length"
)
```

Import Data

- Sample sheet
- RSEM files (by gene or transcript)

```
str(dta_raw)
## List of 4
##  $ abundance           : num [1:58137, 1:14] 7.07 0 35.01 7.77 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:58137] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" ...
##   .. ..$ : chr [1:14] "hum57-2" "hum57-3" "hum57-4" "mbe4-2" ...
##  $ counts              : num [1:58137, 1:14] 388 0 856 667 119 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:58137] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" ...
##   .. ..$ : chr [1:14] "hum57-2" "hum57-3" "hum57-4" "mbe4-2" ...
##  $ length              : num [1:58137, 1:14] 2094 794 933 3274 2042 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:58137] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" ...
##   .. ..$ : chr [1:14] "hum57-2" "hum57-3" "hum57-4" "mbe4-2" ...
##  $ countsFromAbundance: chr "no"
```

Construct DESeq dataset

- From a tximport object

```
dta_raw$length[dta_raw$length == 0] <- 1
dds <- DESeq2::DESeqDataSetFromTximport(
  txi = dta_raw,
  colData = df_sample,
  design = ~ concentration
)
```

- From a matrix object

```
count_mat <- apply(dta_raw$counts, 2, as.integer)
rownames(count_mat) <- rownames(dta_raw$counts)
dds_fromMat <- DESeq2::DESeqDataSetFromMatrix(
  countData = count_mat,
  colData = df_sample,
  design = ~ concentration
)
```

```
model.matrix(~ concentration, data = df_sample)
##    (Intercept) concentration2 concentration20 concentration8
## 1            1               0                0               0
## 2            1               0                0               1
## 3            1               0                1               0
## 4            1               0                0               0
## 5            1               0                0               1
## 6            1               0                1               0
## 7            1               1                0               0
## 8            1               0                0               0
## 9            1               0                0               1
## 10           1               0                1               0
## 11           1               0                0               0
## 12           1               0                0               1
## 13           1               0                1               0
## 14           1               1                0               0
## attr(,"assign")
## [1] 0 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$concentration
## [1] "contr.treatment"
```

Quality control

- Gene (or transcript) level
  - Remove genes with 0 read count in all samples

```
keep <- rowSums(DESeq2::counts(dds)) > 0 # remove zero count genes
dds <- dds[keep,]
```

  - Recommend : remove low count genes, e.g.:
    - Keep genes with at least 1 count per sample

```
low_reads <- rowSums(DESeq2::counts(dds)>1) < nrow(col_dta) # tag low reads genes
```

  - Check the distribution of the average counts and remove extremes
- Sample level
  - Duplicated samples (technical replicates)?
    - Select the best one according to sample's quality, no missing phenotype data , etc.

Annotation

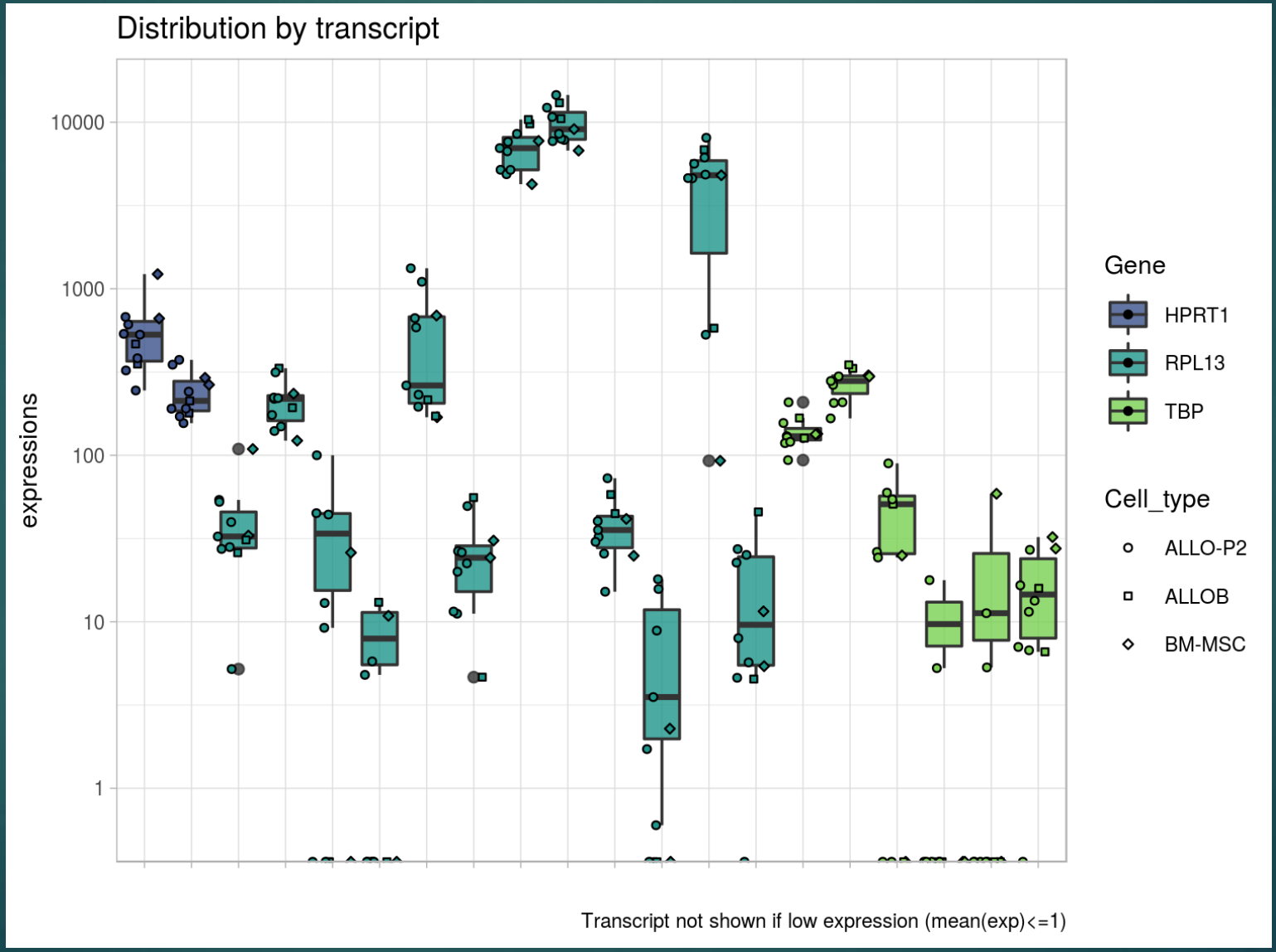
- **biomaRt** based on Ensembl

```
# list_dataset <- biomaRt::listDatasets(biomaRt::useMart('ensembl'))
ensembl <- biomaRt::listEnsemblArchives() %>%
  as_tibble() %>%
  filter(version == "94") %>%
  .[["url"]] %>%
  biomaRt::useMart(
    biomart = "ensembl",
    dataset = "hsapiens_gene_ensembl",
    host = .
)
# list_feature <- biomaRt::listAttributes(ensembl, page="feature_page")
list_gene_ids <- rownames(dta_raw$counts)
annot <- biomaRt::getBM(
  attributes = c(
    "ensembl_gene_id", "external_gene_name", "chromosome_name",
    "start_position", "end_position", "description", "gene_biotype"
  ),
  filters = "ensembl_gene_id",
  values = list_gene_ids,
  mart = ensembl
)
```

Normalization

- Aim : ↓ technical bias / remove non biological variation
- With house keeping genes:
  - Those genes should have steady level of expression across all samples,
    So we can use them to calibrate our counting data
  - Example : HPRT1 seems to be a good candidate in the role of house keeping gene, with a low variability at both levels (gene and transcripts).
    In analysis based on genes results we propose to normalize data on HPRT1.
    In analysis based on transcripts results we propose to normalize data on the geometric mean of the HPRT1's transcripts.
    But, notice that this normalization can not be so robust being based on one gene only.

Normalization



Normalization


```r
#### Normalisation by 1 gene ----
count_gene <- matrix(
  data = c(
    2,3,2,3,
    10,50,10,60,
    0,90,20,80,
    25,29,15,20
  ), nrow = 4, ncol = 4, byrow = TRUE)
row.names(count_gene) <- c("gene_HPRT1","gene_B","gene_C","gene_D")
colnames(count_gene) <- c("sample1", "sample2","sample3", "sample4")

vec_norm_bySample <- count_gene[grep("_HPRT1$",rownames(count_gene)),] ## vector of normalisation
count_gene_norm <- sweep(x = count_gene, MARGIN = 2, STATS = vec_norm_bySample, FUN = "/")

## ready to be analysed
# count_gene_norm <- count_gene_norm[-grep("_HPRT1$",rownames(count_gene)), ]
```

```
> count_gene ## raw count
          sample1 sample2 sample3 sample4
gene_HPRT1       2       3       2       3
gene_B          10      50      10      60
gene_C           0      90      20      80
gene_D          25      29      15      20
```

Normalization


```
#### Normalisation by transcripts : use geometric mean ----
count_iso <- matrix(
  data = c(
    2,3,2,3,
    4,3,4,3,
    4,4,4,5,
    10,50,10,60,
    20,45,10,60,
    0,90,20,80,
    25,29,15,20
  ), nrow = 7, ncol = 4, byrow = TRUE)

row.names(count_iso) <- c(
  "isoform1_HPRT1", "isoform2_HPRT1", "isoform3_HPRT1",
  "isoform1_B", "isoform2_B","isoform1_C","isoform1_D«
)
colnames(count_iso) <- c("sample1", "sample2","sample3", "sample4")
```

```
> count_iso  ## raw count
              sample1 sample2 sample3 sample4
isoform1_HPRT1       2       3       2       3
isoform2_HPRT1       4       3       4       3
isoform3_HPRT1       4       4       4       5
isoform1_B          10      50      10      60
isoform2_B          20      45      10      60
isoform1_C           0      90      20      80
isoform1_D          25      29      15      20
```

Normalization


```
#### Normalisation by transcripts : use geometric mean ----
tab_norm_iso <- count_iso[grep("_HPRT1$", rownames(count_iso)), , drop = FALSE] %>%
  t() %>%
  as.data.frame() %>%
  rownames_to_column(var = "SampleID") %>%
  select(names(.)[c(
      TRUE, # keep SampleID
      colMeans(x = .[, -c(1)]) > 1 ## to remove iso with low expression
    )]
  )

tab_norm_iso[, "geo_fac"] <- apply(X = tab_norm_iso[, -c(1)], MARGIN = 1, FUN = function(x) {
  x[x <= 0] <- 1
  return(exp(mean(log(x))))
})

vec_normTr_bySample <- tab_norm_iso[, "geo_fac"] ## vecteur de normalisation
names(vec_normTr_bySample) <- tab_norm_iso[, "SampleID"] ## avec les memes noms de samples
count_iso_norm <- sweep(x = count_iso, MARGIN = 2, STATS = vec_normTr_bySample, FUN = "/")

## ready to be analysed
# count_iso_norm <- count_iso_norm[-grep("_HPRT1$",rownames(count_iso_norm)), ]
```

Normalization


```
> count_gene ## raw count
           sample1 sample2 sample3 sample4
gene_HPRT1       2       3       2       3
gene_B          10      50      10      60
gene_C           0      90      20      80
gene_D          25      29      15      20
> count_gene_norm ## normalized count
           sample1   sample2 sample3   sample4
gene_HPRT1     1.0  1.000000     1.0  1.000000
gene_B         5.0 16.666667     5.0 20.000000
gene_C         0.0 30.000000    10.0 26.666667
gene_D        12.5  9.666667     7.5  6.666667
>
```

```
> count_iso  ## raw count
                sample1 sample2 sample3 sample4
isoform1_HPRT1       2       3       2       3
isoform2_HPRT1       4       3       4       3
isoform3_HPRT1       4       4       4       5
isoform1_B          10      50      10      60
isoform2_B          20      45      10      60
isoform1_C           0      90      20      80
isoform1_D          25      29      15      20
> count_iso_norm ## normalized count
                 sample1    sample2   sample3    sample4
isoform1_HPRT1 0.6299605  0.9085603 0.6299605  0.8434327
isoform2_HPRT1 1.2599210  0.9085603 1.2599210  0.8434327
isoform3_HPRT1 1.2599210  1.2114137 1.2599210  1.4057211
isoform1_B     3.1498026 15.1426716 3.1498026 16.8686533
isoform2_B     6.2996052 13.6284044 3.1498026 16.8686533
isoform1_C     0.0000000 27.2568089 6.2996052 22.4915377
isoform1_D     7.8745066  8.7827495 4.7247039  5.6228844
>
```

Normalization

- Aim : ↓ technical bias / remove non biological variation
- With house keeping genes:
  - Those genes should have steady level of expression across all samples,
    So we can use them to calibrate our counting data
- Without house keeping genes, e.g.:
  - Quantile normalization
  - Median of ratio(implemented in {DESeq2}, based on a sample-specific normalization factor and estimated by the median of the ratios of RSEM counts.
  - Trimmed Mean of M value (TMM, implemented in {edgeR})
  - Counts TPM

Differential expression analysis

- Aim: Identify over/under-expressed genes or transcripts

- Model in DESeq2: generalized linear model (GLM)
  - Statistic test
    - Null hypothesis (H0): the gene expression level between two (or more) conditions are equal
    - p-value: probability of getting the same value of the test if the null hypothesis was true

- Fold change (FC)

$$FC = \frac{mean(cond1) - mean(cond2)}{min(mean(cond1), mean(cond2))}$$

Log2(FC) > 0: over-expressed

Log2(FC) = 0: no change

Log2(FC) < 0: under-expressed

Differential expression analysis

- Run DESeq2::DESeq()

```
## differential expression analysis
dds <- DESeq2::DESeq(object = dds)

### extract results
res <- DESeq2::results(
  object = dds,
  pAdjustMethod = "fdr",
  contrast = c("concentration", "2", "5"),
  lfcThreshold = 0,
  cooksCutoff = 0.99,
  independentFiltering = FALSE,
  ...
)
```

```
> res[order(res$pvalue), ]
log2 fold change (MLE): concentration 2 vs 5
Wald test p-value: concentration 2 vs 5
DataFrame with 33760 rows and 6 columns
                     baseMean     log2FoldChange             lfcSE                stat
                    <numeric>          <numeric>         <numeric>           <numeric>
ENSG00000176868  20.9622529048082   8.44178026314738  1.24162148000951   6.79899663388773
ENSG00000233728  57.7149158393483  -6.40781085456706  1.14986671738487  -5.57265529794644
ENSG00000204816   11.200722191153   7.54269585949615  1.40471683598291   5.36954898402591
ENSG00000165194  33.9505530160418   5.36316905268844  1.03029890582241   5.20544962474497
ENSG00000121351  31247.3746833213  -1.43749363524137 0.306427869376118  -4.69113216812846
...                           ...                ...               ...                ...
ENSG00000284240 0.409004164513431  -1.07935127319798  4.32799504117685 -0.249388287862845
ENSG00000284337 0.727291130336306  -2.54873929733279   4.2627784161611 -0.597905649439805
ENSG00000284413  1.57127832517687  -3.62660434808228  3.26963285171273  -1.10917785346528
ENSG00000284526  9.18116967151051  0.350253808721456  1.02992763091943  0.340076135649241
ENSG00000284540  1.18624640876061   1.62225933355514  2.82653578181334  0.573939075526011
                          pvalue               padj
                       <numeric>          <numeric>
ENSG00000176868 1.05350278872081e-11 3.25648247021491e-07
ENSG00000233728 2.50885672495995e-08 0.000387756351126185
ENSG00000204816 7.89337957477222e-08  0.00081330752011928
ENSG00000165194 1.93527443335338e-07  0.00149553170023465
ENSG00000121351 2.71697380098908e-06   0.0167968754324747
...                              ...                ...
ENSG00000284240                   NA                 NA
ENSG00000284337                   NA                 NA
ENSG00000284413                   NA                 NA
ENSG00000284526                   NA                 NA
ENSG00000284540                   NA                 NA
```

```
> dta_raw$counts[c("ENSG00000176868", "ENSG00000233728"), ]
                 hum57-2 mbe4-2 hum54-1 hum54-2 hum56-2 hum57-1
ENSG00000176868    0.00    0.00   81.47    0.00     0.0    43.7
ENSG00000233728  111.63  129.08    1.91   36.11    80.4     0.0
```

Multiple testing correction

- 4 results
  false positive: Type-I error α
  false negative: Type-II error β
  True positive and True negative.

- By doing n independent tests, reject $H0_i$ when $pv_i \leq \alpha$.
- But if we consider the whole of *n* tests, the number of false positive obtain by chance increases with *n*.
- To minimize this inflation of false positive, the idea is to choose a threshold more stringent, not based on the Type-I error of each test (usually $\alpha = 0{,}05$), but based on the global amount of errors made over the *n* tests.

- Examples

Bonferroni (fwer Family Wise Error Rate) : very stringent.
Control the fact of wrongly reject at least once the null hypothesis.
Benjamini Hochberg (fdr False Discovery Rate) :
Control the expected proportion of false positive.

		Reality	
		H_o Is True	H_1 Is True
Conclusion	Do Not Reject H_o	Correct Conclusion	Type II Error
	Reject H_o	Type I Error	Correct Conclusion

(Source Méthodes statistiques pour l'analyse des données génétiques d'association à grande échelle, Mickael Guedj)

Graph

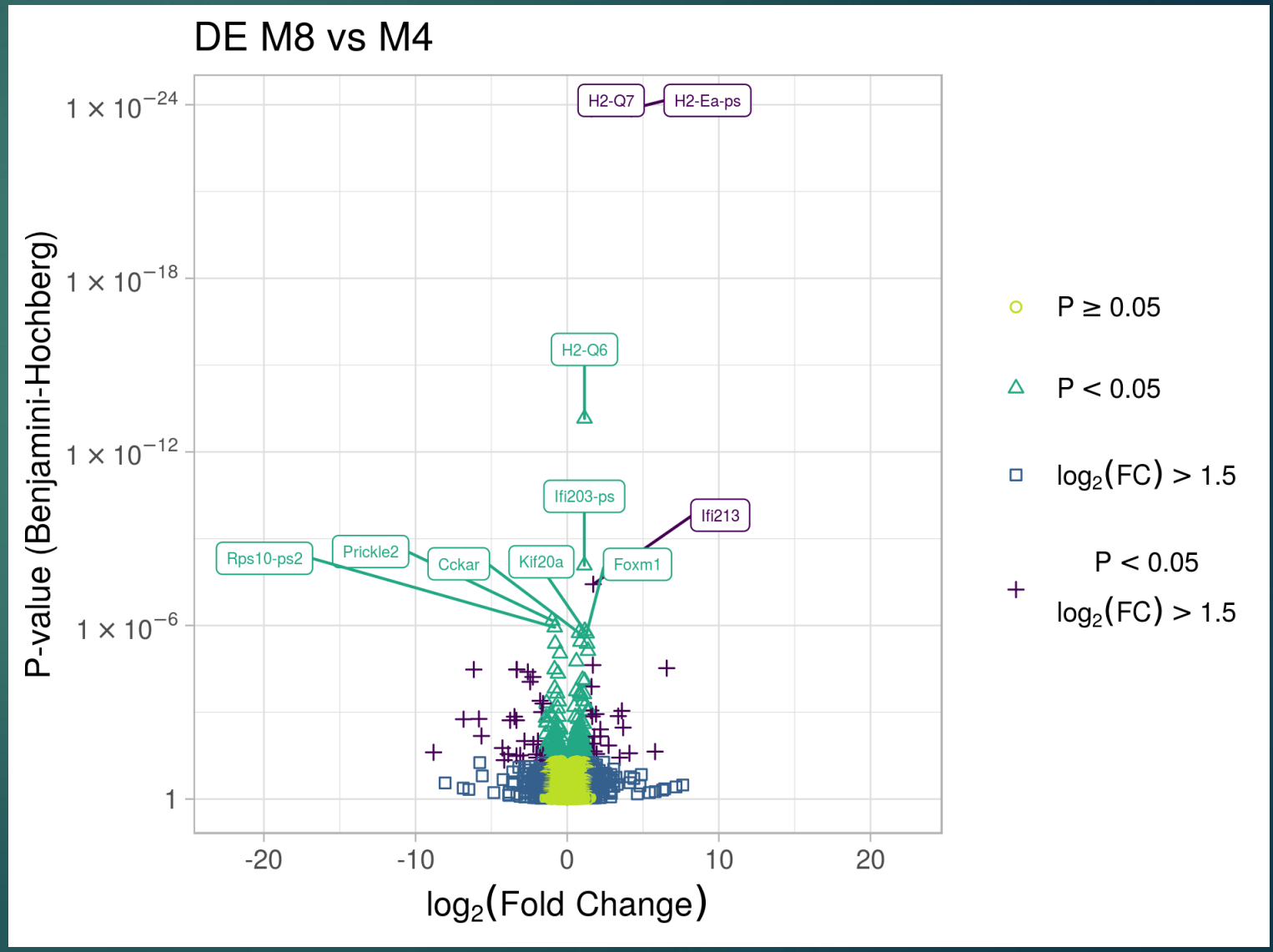
Volcano plot

- The Fold Change (FC) or ratio measure the variation of the gene expression between 2 conditions
- Values are log2 transformed. More interpretable.

Log2(FC) > 0: gene over-expressed

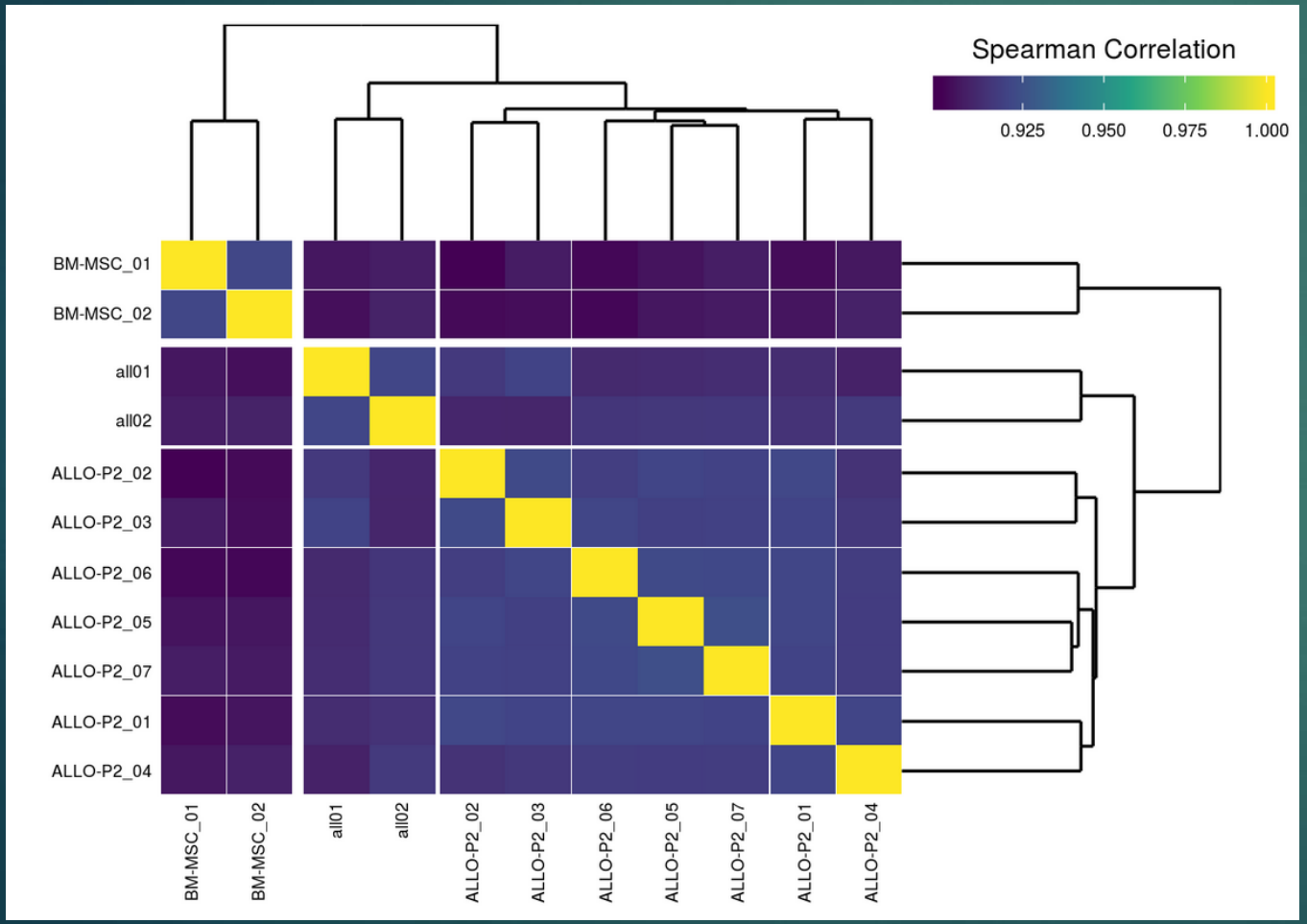
Log2(FC) = 0: no change

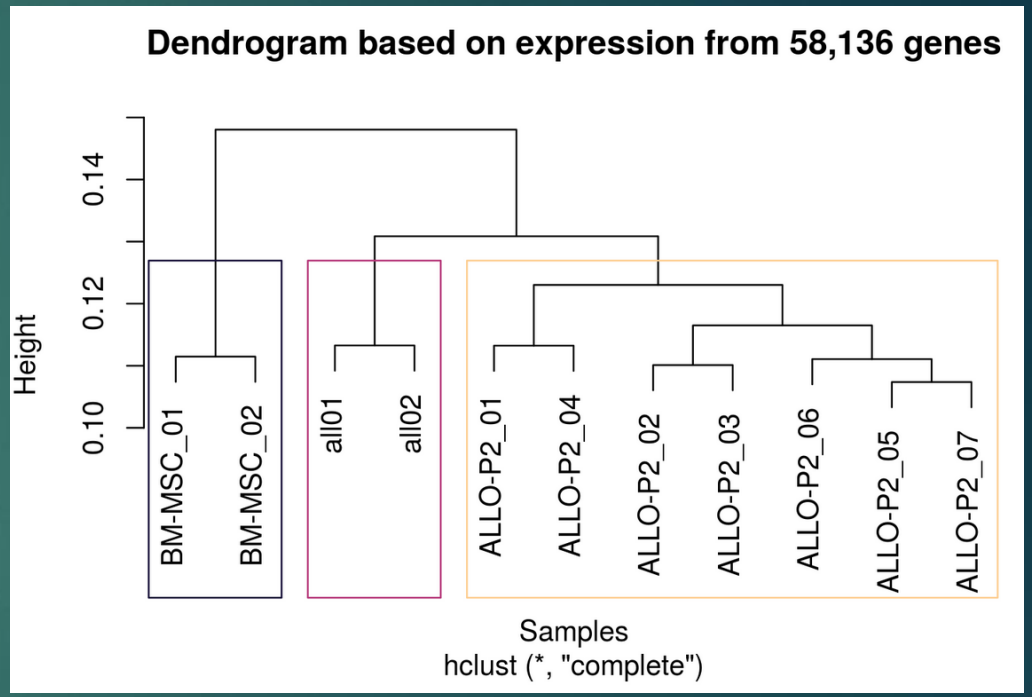
Log2(FC) < 0: gene under-expressed



Further analyses …

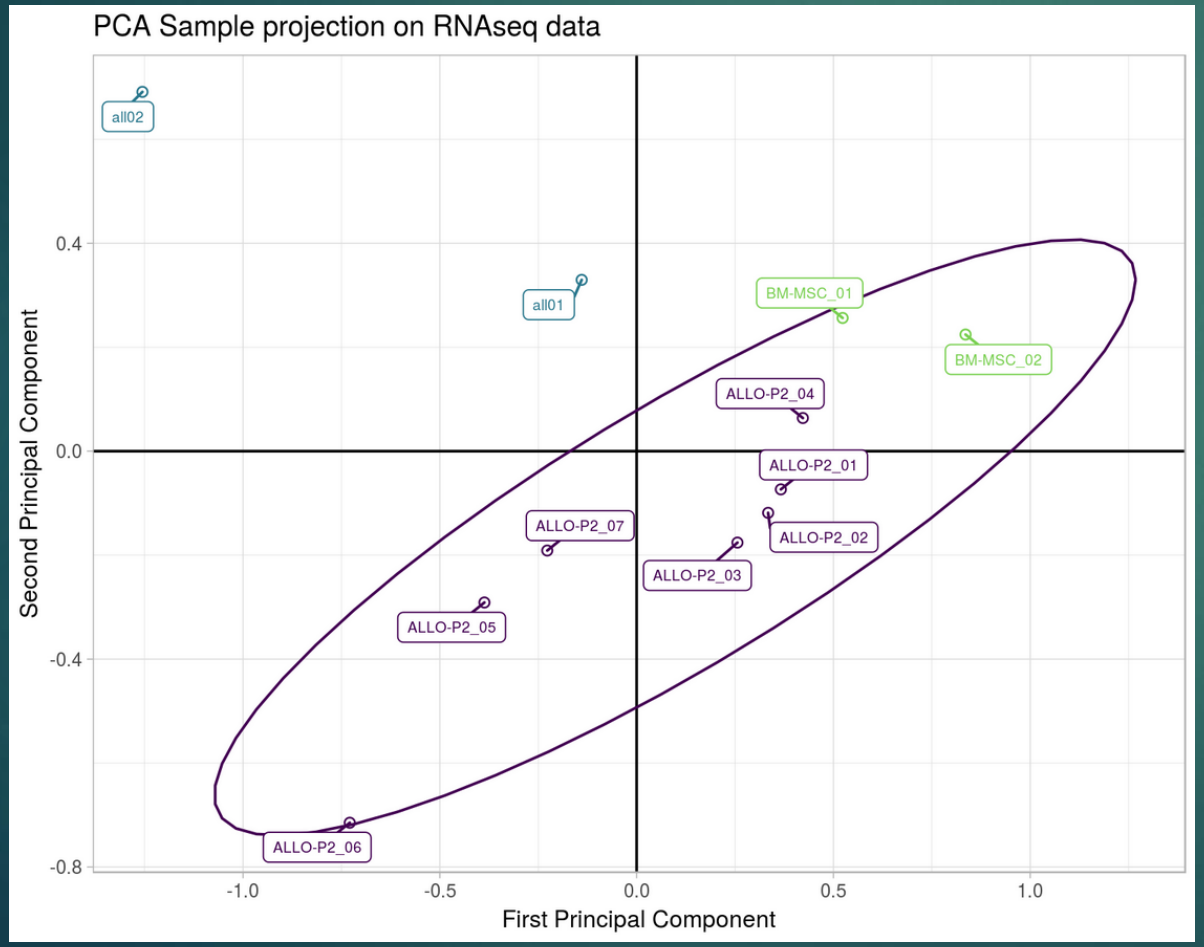
- Correlation between samples

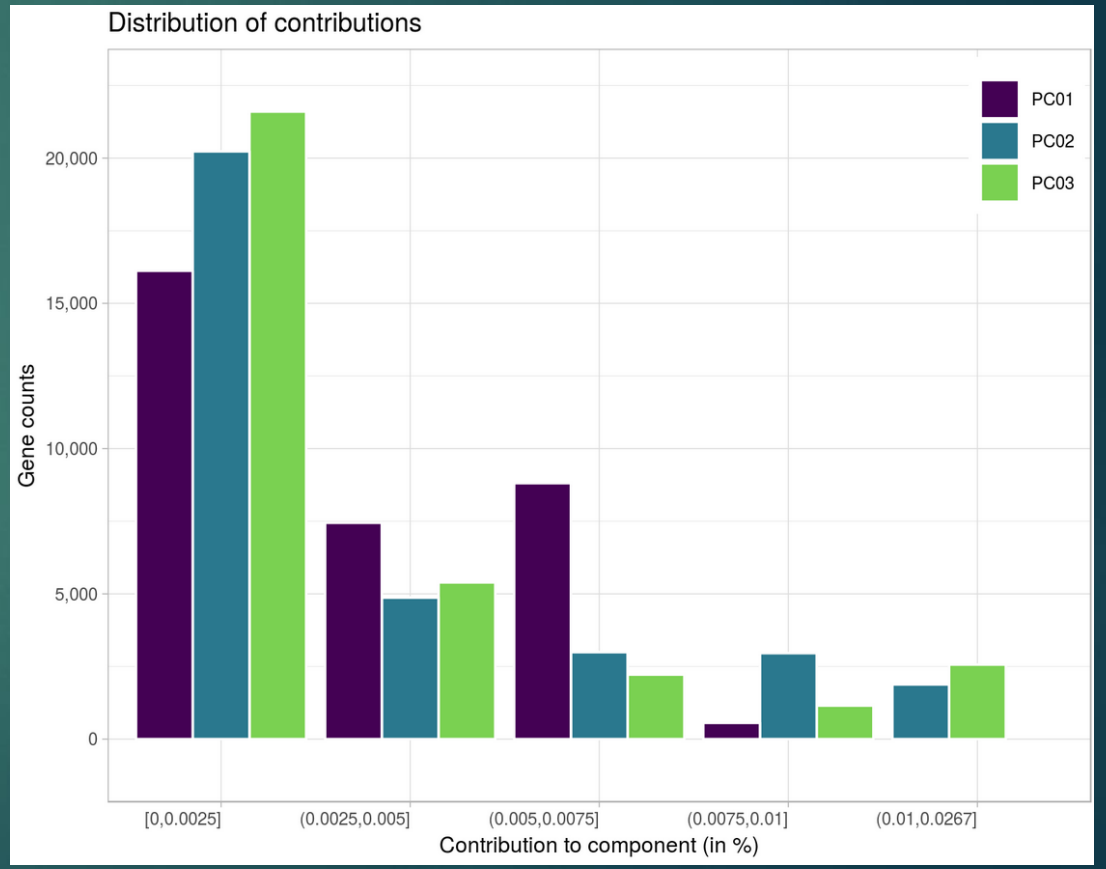




Further analyses …

- PCA
  - Reduce dimension and discriminate samples in a orthogonal plane





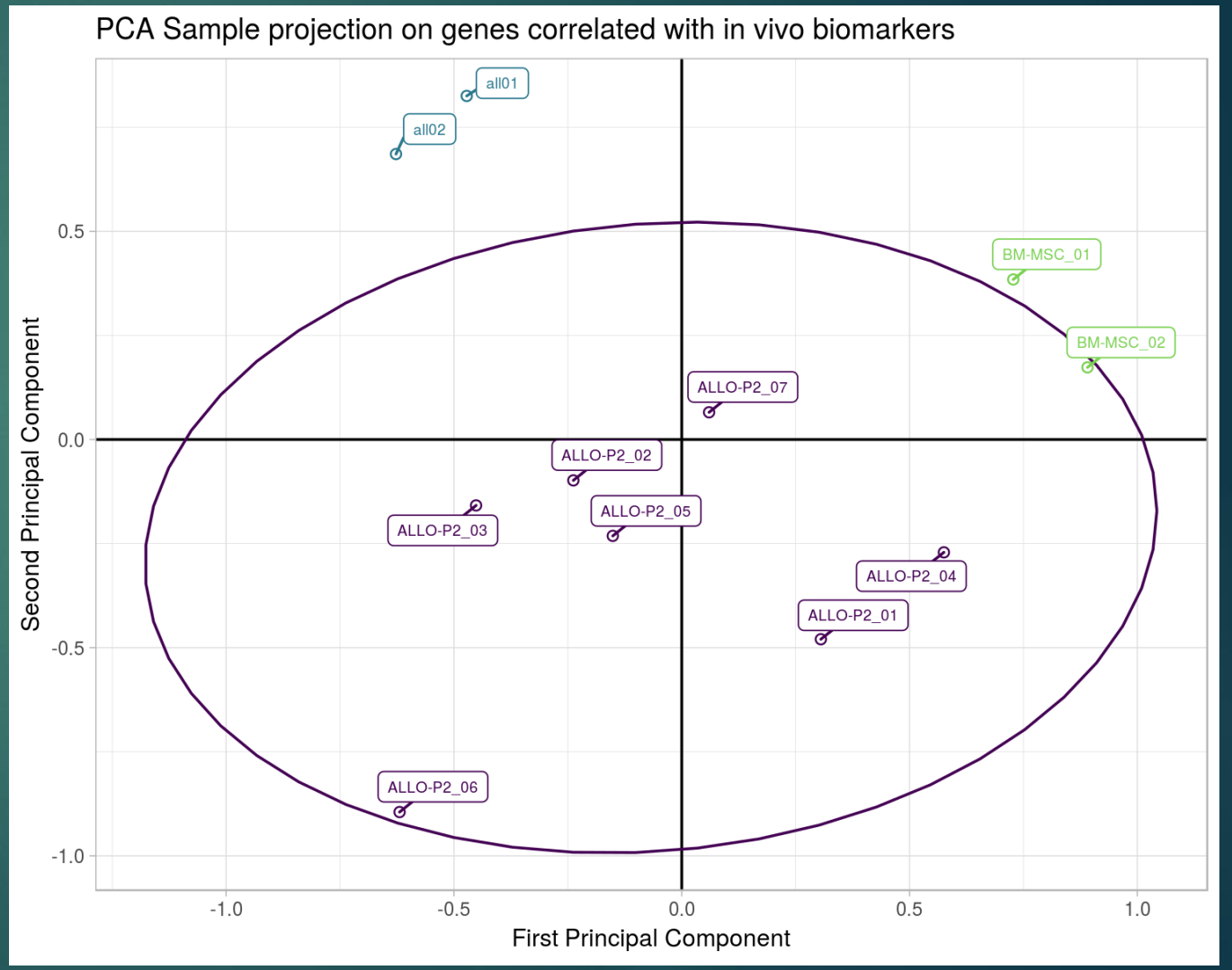
Further analyses …

- To identify in vivo biomarker

The spearman correlation is computed between in vivo measures and RNA-seq data.
The goal is to explain variation in the biomarkers that can be attributed to variation in gene expression.

regression analyses are use to quantify the strength of this relationship.

PCA on genes significantly correlated with in vivo biomarkers



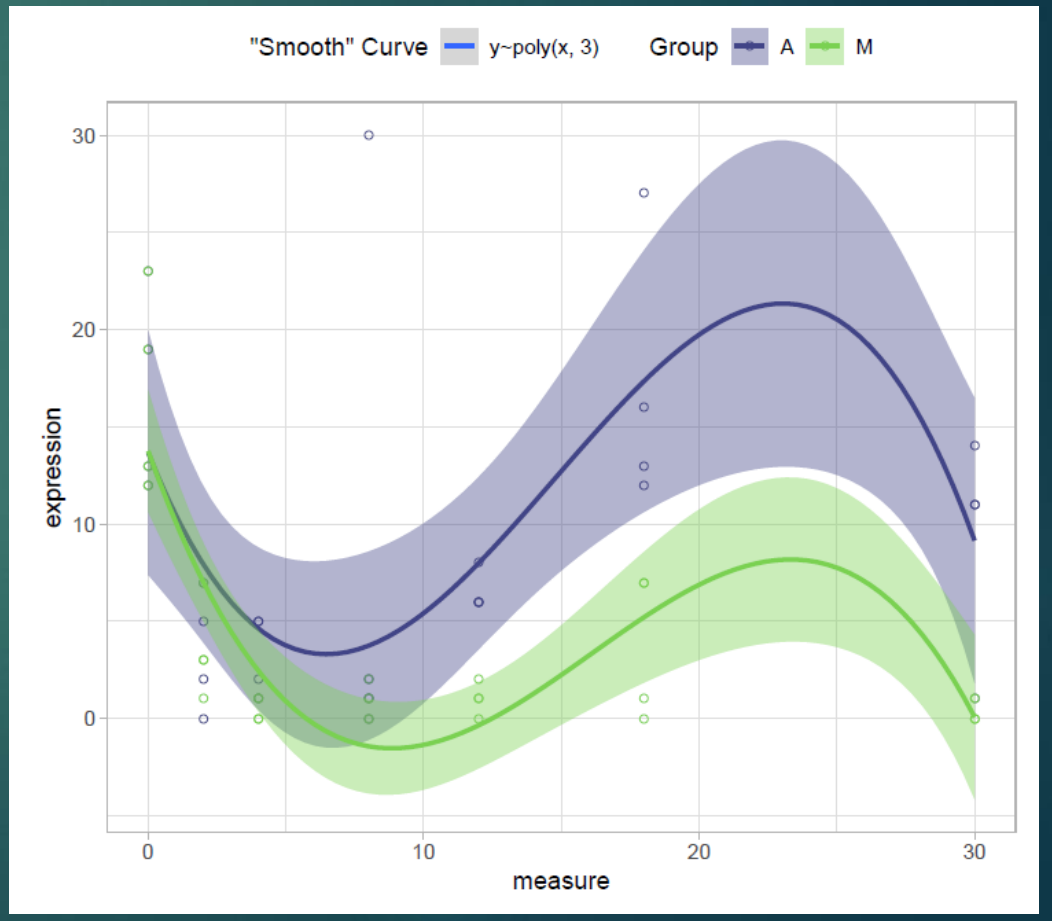
Further analyses …

- Kinetics

```
lm(expression ~ group + poly(measure,3) + poly(measure,3):group)
```

$$expression \ \ function\ of$$
$$group + measure + measure^2 + \ measure^3 \ and$$
$$interaction\ (measure + measure^2 + \ measure^3)\ knowing\ group$$

term	estimate	std.error	statistic	p.value
(Intercept)	9.2153	1.0688	8.6220	3.05e−11
poly(measure, 3)1	7.4623	7.7103	0.9678	0.338
poly(measure, 3)2	0.9372	7.8472	0.1194	0.905
poly(measure, 3)3	−32.6367	7.9723	−4.0937	1.66e−04
groupM	−5.3229	1.5257	−3.4888	1.06e−03
poly(measure, 3)1:groupM	−26.6491	11.3645	−2.3449	0.023
poly(measure, 3)2:groupM	16.2497	11.3333	1.4338	0.158
poly(measure, 3)3:groupM	6.2513	11.3042	0.5530	0.583



Further analyses …

- Classification (and Correlation between genes)
  - Supervised
    - Hierarchical classification (ascending / descending)
  - Unsupervised
    - K-means => need to give *a priori* the number of clusters
    - t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Enrichment analysis
  - Database:
    - The Gene Ontology Consortium (GO): hierarchical relationship of genes
    - Kyoto Encyclopedia of Genes and Genomes (KEGG): gene pathway
  - R packages, e.g.:
    - { RDAVIDWebService }, { clusterProfiler }

References

1. WANG, Zhong, GERSTEIN, Mark, et SNYDER, Michael. RNA-Seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 2009, vol. 10, no 1, p. 57.
2. LOVE, Michael I., SONESON, Charlotte, et ROBINSON, Mark D. Importing transcript abundance datasets with tximport. *dim (txi. inf. rep $ infReps $ sample1)*, 2017, vol. 1, no 178136, p. 5.
3. LOVE, Michael, ANDERS, Simon, et HUBER, Wolfgang. Differential analysis of count data–the DESeq2 package. *Genome Biol*, 2014, vol. 15, no 550, p. 10.1186.
4. DESeq2 vignette: http://www.bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html
5. TPM / RPKM / FPKM: https://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/

Thank you,
Any questions ?