Introduction to bulk RNA-seq

Koen Van den Berge Lieven Clement

```
library(tidyverse)
library(SummarizedExperiment)
library(edgeR)
library(BiocFileCache)
library(tximeta)
library(org.Hs.eg.db)
library(ExploreModelMatrix)
library(macrophage)
library(patchwork)
```

In the first module of the course we studied the proteome: the concentrations of proteins in biological samples. In this module we move to the transcriptome: the concentrations of messenger RNA (mRNA) molecules, which may be translated into proteins, but may also have functions on their own.

This chapter covers three parts:

1. How RNA-seq data are generated: from RNA molecules in a sample to a table of counts.
2. How we model count data: from the Poisson distribution for technical replicates to the negative binomial and quasi-likelihood models for biological replicates.
3. A complete differential expression workflow in R on the macrophage data set.

The statistical background of the methods (generalised linear models, quasi-likelihood, normalisation and filtering) and the analysis at the transcript level are covered in the technical details chapter.

The study of gene expression

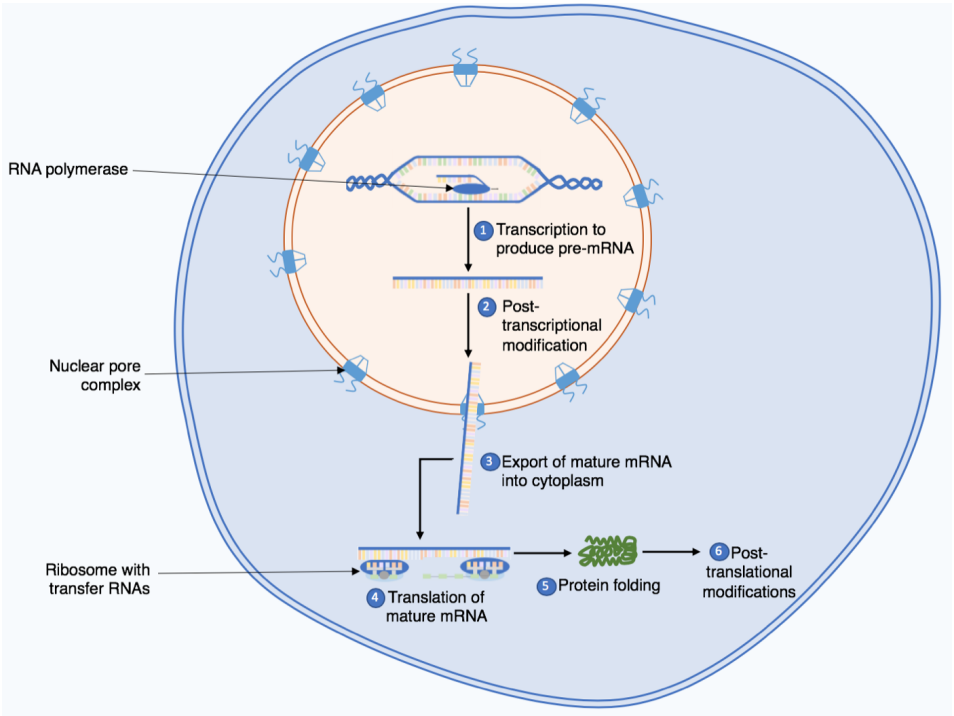


Figure 1: Central paradigm of biology: a gene, a specific region in the DNA, is first transcribed into RNA and then translated into a protein. For RNA genes the RNA molecule itself is the end product, which is referred to as non-coding RNA (ncRNA) (Source: Wikipedia).

- The proteome is only part of the picture: transcription is regulated and the mRNA concentrations of a gene reflect its activity in a sample.
- mRNA is much easier to measure at scale than proteins: sequencing gives a readout for (almost) all genes at once.

Alternative splicing

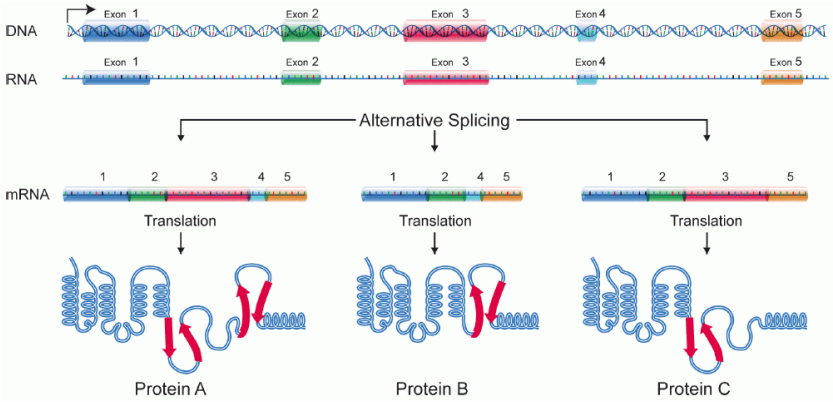


Figure 2: Alternative splicing allows a single gene in eukaryotes to code for multiple proteins. Particular exons of a gene may be included or excluded from the premature messenger RNA, so that exons are joined in different combinations, leading to different mRNA molecules (isoforms or transcripts). The proteins translated from alternatively spliced mRNAs differ in their amino acid sequence and, often, in their function (Source: Wikipedia).

- A gene can thus produce several **transcripts** (isoforms).
- We can quantify expression at the gene level, by combining all transcripts of a gene, or at the transcript level.
- Changes in the relative abundance of the transcripts of a gene (**differential transcript usage**) are discussed in the technical details chapter.

Sequencing technology

- mRNA is typically measured by sequencing.
- The technology evolves at an incredible speed: the output of so-called next generation sequencing machines has more than doubled every year, while the cost per gigabase keeps dropping.
- The large majority of RNA-seq data are generated by sequencing-by-synthesis on machines produced by Illumina.
- Long-read technologies from Pacific Biosciences and Oxford Nanopore sequence full-length molecules. They are increasingly used for transcript discovery and isoform quantification, but short reads remain the standard for gene expression studies because of their higher throughput and lower cost per read.

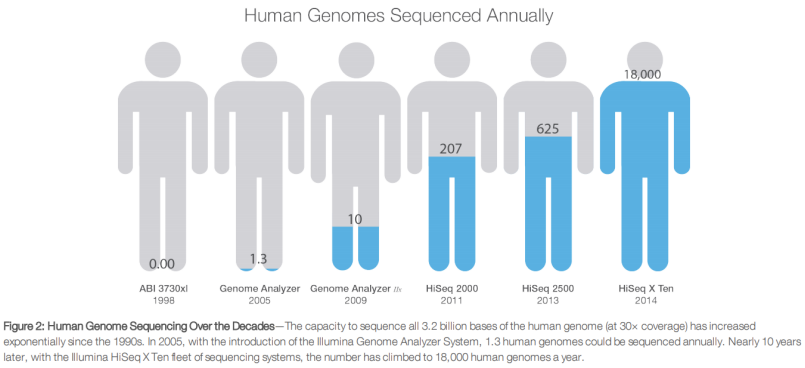


Figure 3: The data output revolution of sequencing machines (Source: Illumina).

From sample to data analysis

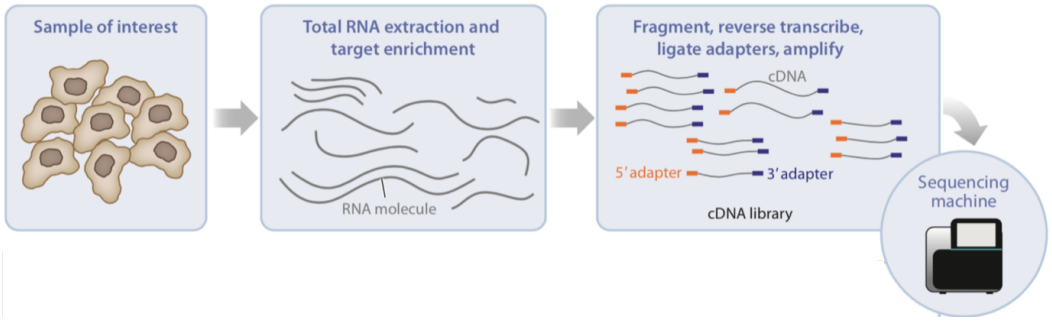


Figure 4: The RNA-seq workflow (adapted from Van den Berge et al., 2019).

Library preparation consists of the following steps (Van den Berge et al. 2019):

1. **Collect the samples of interest.** Several types of input samples are amenable to sequencing, such as fresh frozen tissue or FFPE preserved samples.
2. **Capture the mRNA molecules.** The cells are lysed and the RNA is extracted. mRNA is mostly captured by
    - poly-A selection, which selects polyadenylated RNA, or
    - ribosomal depletion, which removes ribosomal RNA, so that also non-polyadenylated RNA, e.g. long non-coding RNA, is retained.
3. **Fragmentation.** The captured molecules are fragmented chemically or mechanically, typically to a size of 300 to 500 bp.
4. **Reverse transcription.** The sequencers only sequence double-stranded DNA, so the RNA fragments are reverse transcribed into complementary DNA (cDNA).

5. **Adapter ligation.** Platform-specific adapter sequences are added to both ends of the fragments so that the sequencer can recognise them. Adapters can also carry a sample barcode, so that several samples can be pooled and sequenced together (multiplexing).
6. **PCR amplification** to increase the concentration of the library.
7. **Sequencing.** The library is loaded on the sequencer. See this video on sequencing-by-synthesis.

Illumina sequencing

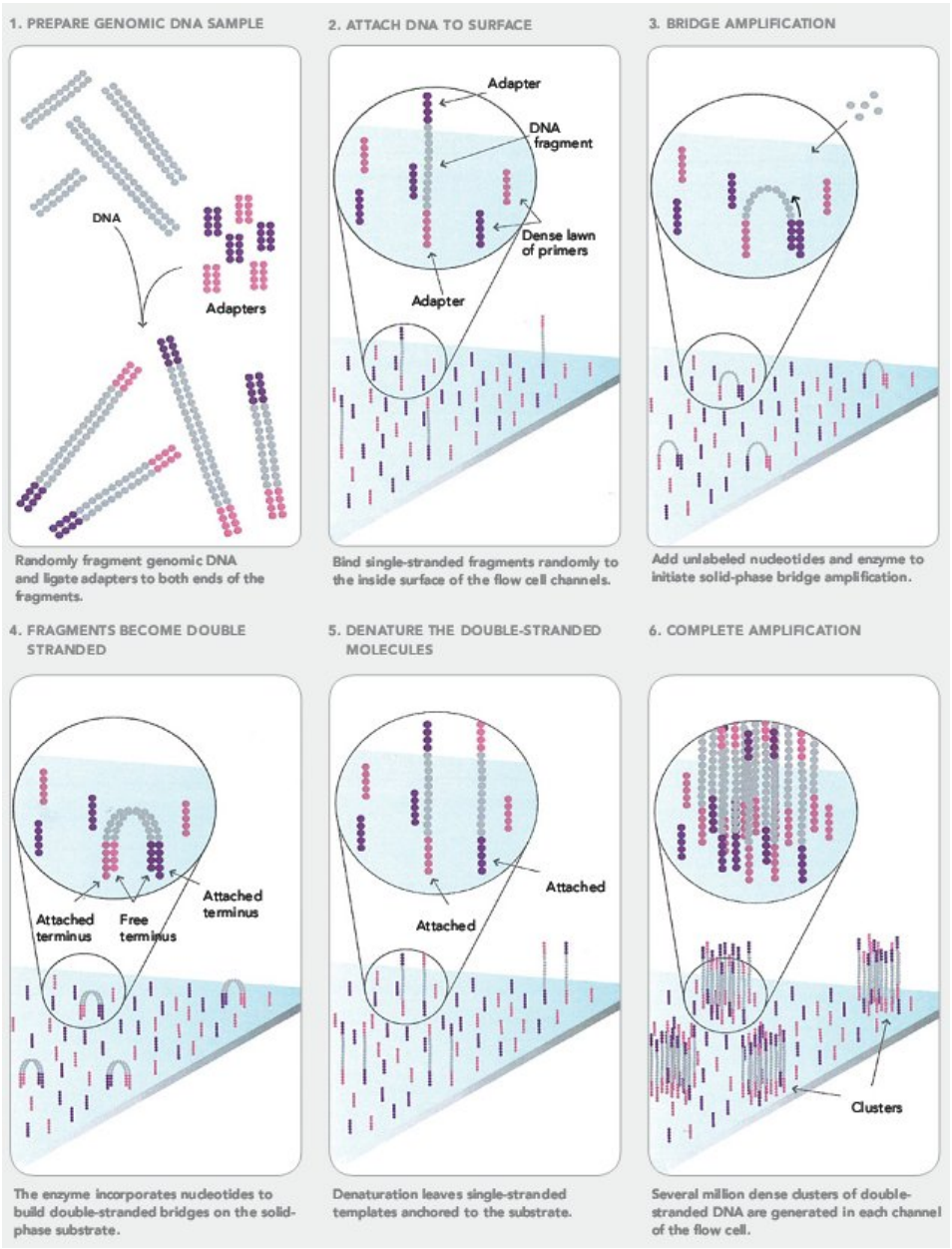


Figure 5: Illumina sequencing, steps 1 to 6 (Source: Illumina).

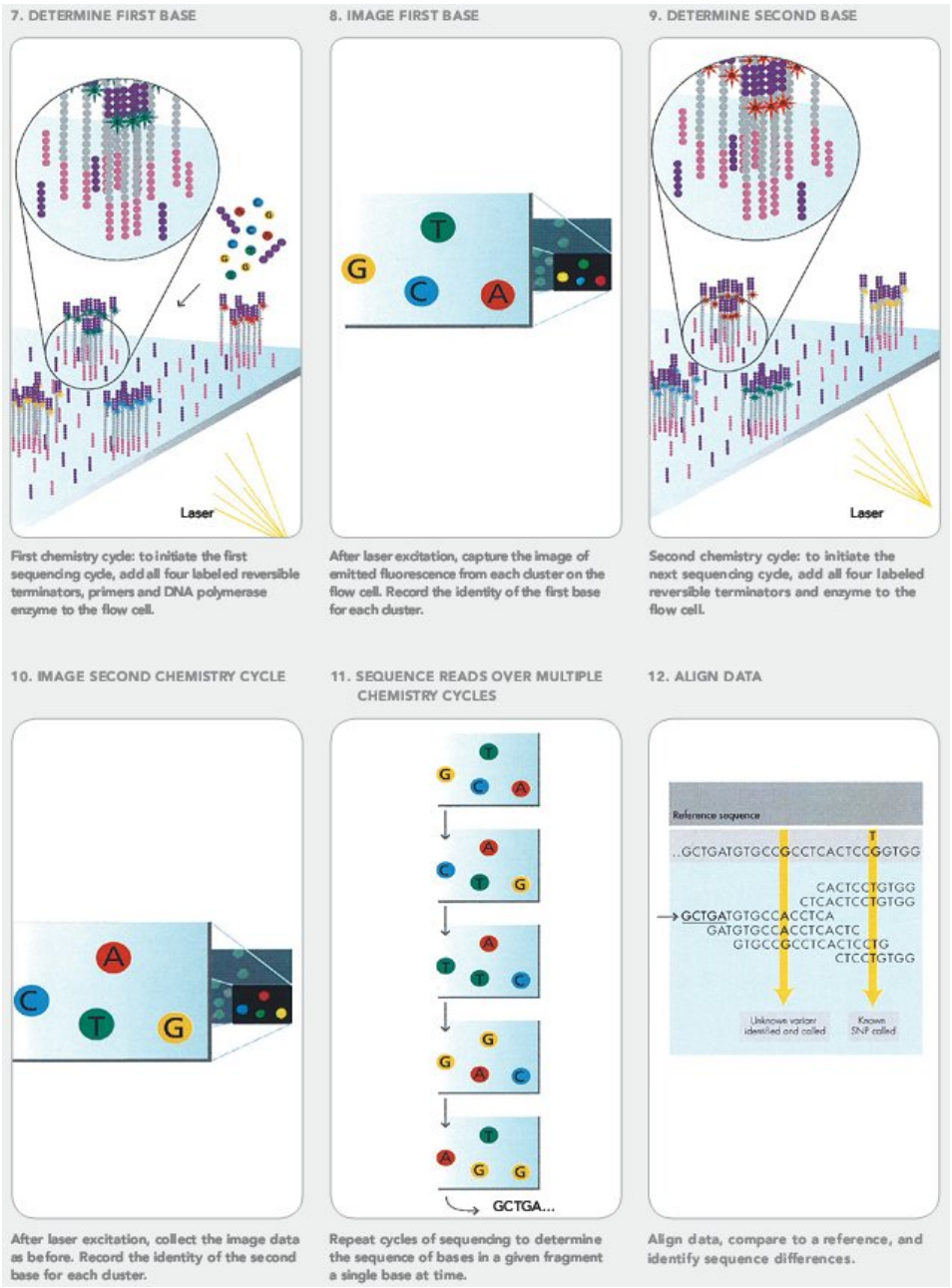


Figure 6: Illumina sequencing, steps 7 to 12 (Source: Illumina).

Protocol choices

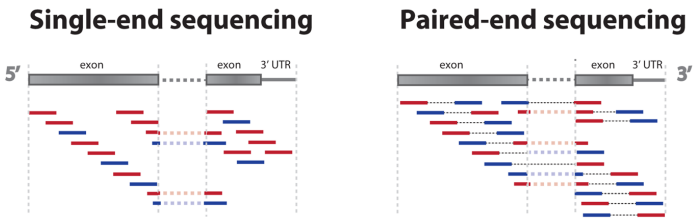


Figure 7: Single-end versus paired-end sequencing. In paired-end sequencing both ends of a fragment are sequenced (from Zhernakova et al., 2013, doi:10.1371/journal.pgen.1003594).

- **Single-end vs paired-end**: one or both ends of each fragment are sequenced. Paired-end reads carry more information to locate a fragment, which is especially useful at the transcript level.
- **Stranded vs unstranded**: stranded protocols preserve the information from which DNA strand the RNA was transcribed.
- **Read length and sequencing depth**: the number of reads per sample. For gene expression, more biological replicates are typically more valuable than a higher depth.

The sequencing output: FASTQ files

- The sequencer returns a FASTQ file per sample (two for paired-end data), with millions of sequences, called **reads**.
- Each read takes four lines:

    1. a sequence identifier, starting with `@`,
    2. the sequence,
    3. a separator line, starting with `+`,
    4. a quality score for every base call.

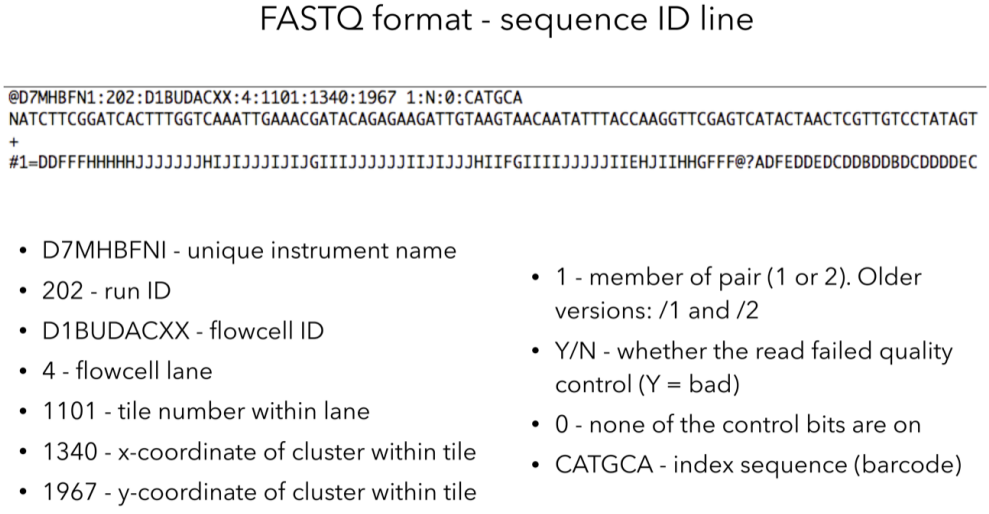


Figure 8: One read in a FASTQ file (slide courtesy of Charlotte Soneson).

The quality scores are Phred scores, $Q = -10 \log_{10} p$, with p the probability of an incorrect base call. They are stored as ASCII characters: with the common Phred+33 encoding the character with ASCII code $Q + 33$ is stored.

Phred score	Probability of an incorrect base call	Base call accuracy	Phred+33 character
10	1/10	90%	+
20	1/100	99%	5
30	1/1000	99.9%	?
40	1/10000	99.99%	I

Preprocessing of raw reads

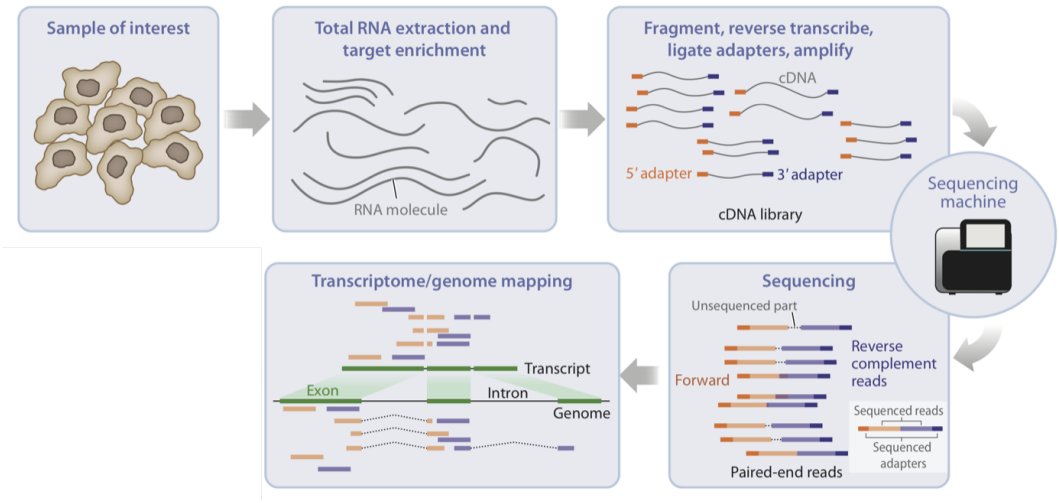


Figure 9: The RNA-seq workflow including preprocessing and mapping (adapted from Van den Berge et al., 2019).

Quality control

- Quality control (QC) checks the quality of each sample, e.g. to detect degraded RNA or adapter contamination.
- The most popular QC tool is FastQC. MultiQC aggregates the reports of many samples.
- Compare a report of good and bad Illumina data.

The most important plot shows the distribution of the Phred quality scores at every position in the reads.

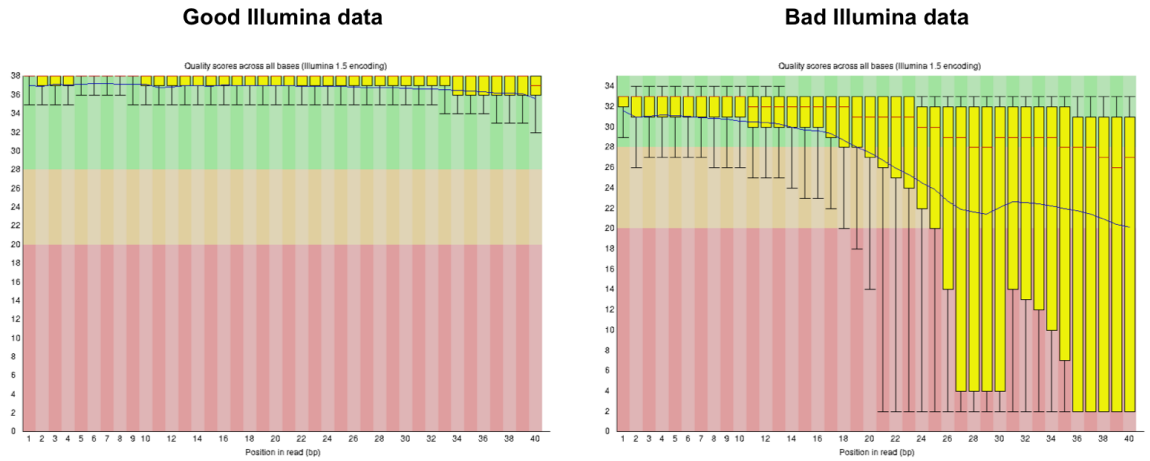


Figure 10: Per-base sequence quality of good and bad Illumina data: boxplots of the Phred scores at every position in the reads, with the mean in blue. Green: good quality, orange: reasonable, red: poor. Source: FastQC example reports.

- In good data the quality stays high over the whole read.
- In bad data the quality drops towards the end of the reads: these bases can be trimmed (see below).

FastQC flags every module with pass, warn or fail. Its thresholds assume a random sample of genomic DNA, so some flags are expected for RNA-seq data:

- **Sequence duplication levels**: highly expressed transcripts produce many identical fragments.
- **Per base sequence content**: the base composition of the first 10 to 15 bases is uneven because of the random hexamer priming during library preparation.
- **Overrepresented sequences**: often fragments of very highly expressed transcripts, e.g. mitochondrial or ribosomal RNA.

Read trimming and filtering

- Trimming of adapter sequences, barcodes, poly-A tails and, if needed, low quality bases at the end of reads. This is often already done by the sequencing provider.
- Filtering of low quality reads.
- In RNA-seq we do **not** remove duplicate reads: highly expressed transcripts naturally produce many identical fragments.

Mapping and quantification

- Mapping assigns each read to the genomic feature(s) (genes, transcripts, exons) that could have produced it.
- The number of reads assigned to a feature is a proxy for its expression in the sample.
- Mapping is hard: many reads cannot be assigned uniquely and are compatible with several genes or transcripts (**multi-mapping** reads).

Reference files

- A **reference genome** is a representative genome sequence of a species. It comes with an annotation file (GTF or GFF) that defines the coordinates of the genes, transcripts and exons.
- A **reference transcriptome** contains the sequences of all known transcripts.
- Both can be downloaded from e.g. Ensembl or GENCODE.

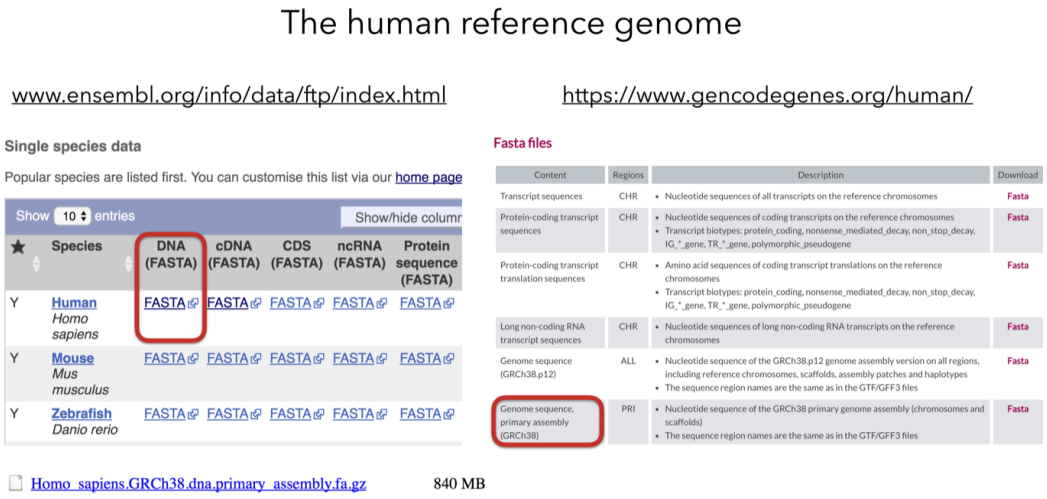


Figure 11: A reference genome (slide courtesy of Charlotte Soneson).

www.ensembl.org/info/data/ftp/index.html https://www.gencodegenes.org/human/

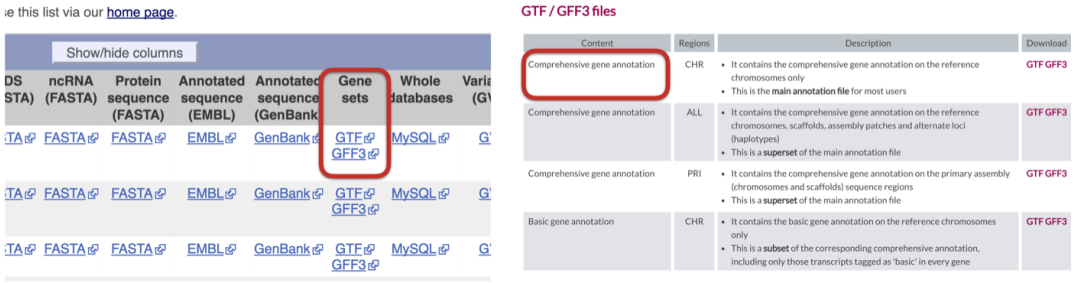


Figure 12: A reference transcriptome (slide courtesy of Charlotte Soneson).

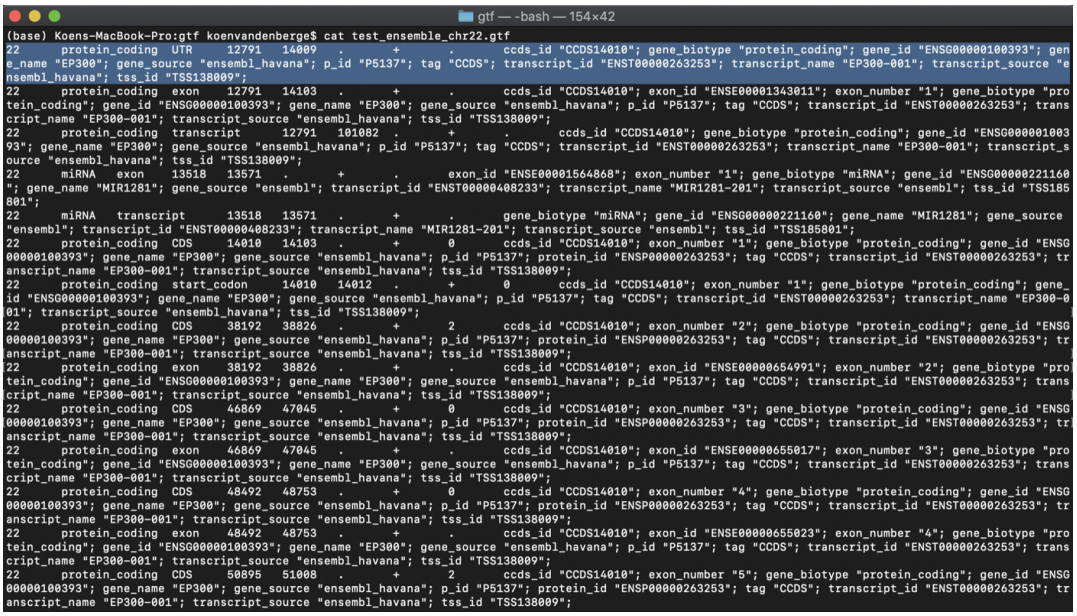


Figure 13: An example GTF file.

Alignment-based workflows

- Reads are aligned to the exact position in the reference genome, e.g. with STAR.
- Because of splicing, a read can span an exon-exon junction and has to be split over positions that can be thousands of basepairs apart (**spliced alignment**).
- The aligned reads are then counted per gene, e.g. with featureCounts. Many heuristic choices are involved: do we count reads that only partly overlap a gene, intronic reads, multi-mapping reads?

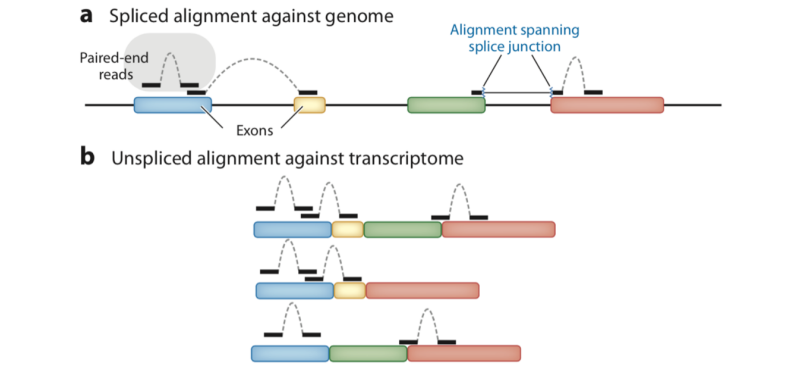


Figure 4

An illustration of spliced alignment of RNA sequencing (RNA-seq) fragments to a genome (*a*) and direct alignment to a transcriptome (*b*). Reads are designated by thick solid lines, while dashed arcs represent the pairing relationship between paired-end reads. This illustration depicts alignment to a single four-exon gene consisting of three distinct transcripts. In the spliced alignment (*a*), the left read of the rightmost pair is a junction-spanning alignment to the red–green exon boundary. In the direct alignment to the transcriptome (*b*), one observes how the same alignment (e.g., the alignment to the blue exon) is repeated for each transcript.

Figure 14: Unspliced and spliced alignment (from Van den Berge et al., 2019).

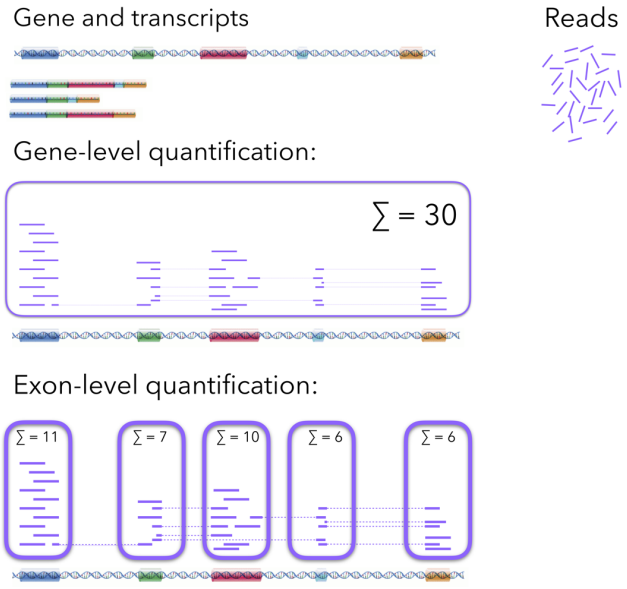


Figure 15: Gene- and exon-level read counting (adapted from Charlotte Soneson).

Alignment-free workflows

- Lightweight methods such as kallisto (Bray et al. 2016) and salmon (Patro et al. 2017) do not search the exact position of each read in the genome.
- They map reads to a reference transcriptome using k-mers, short subsequences of length k.
- For each read they determine the set of transcripts it is compatible with: its **equivalence class**.
- This is much faster and requires much less memory than spliced alignment to the genome.

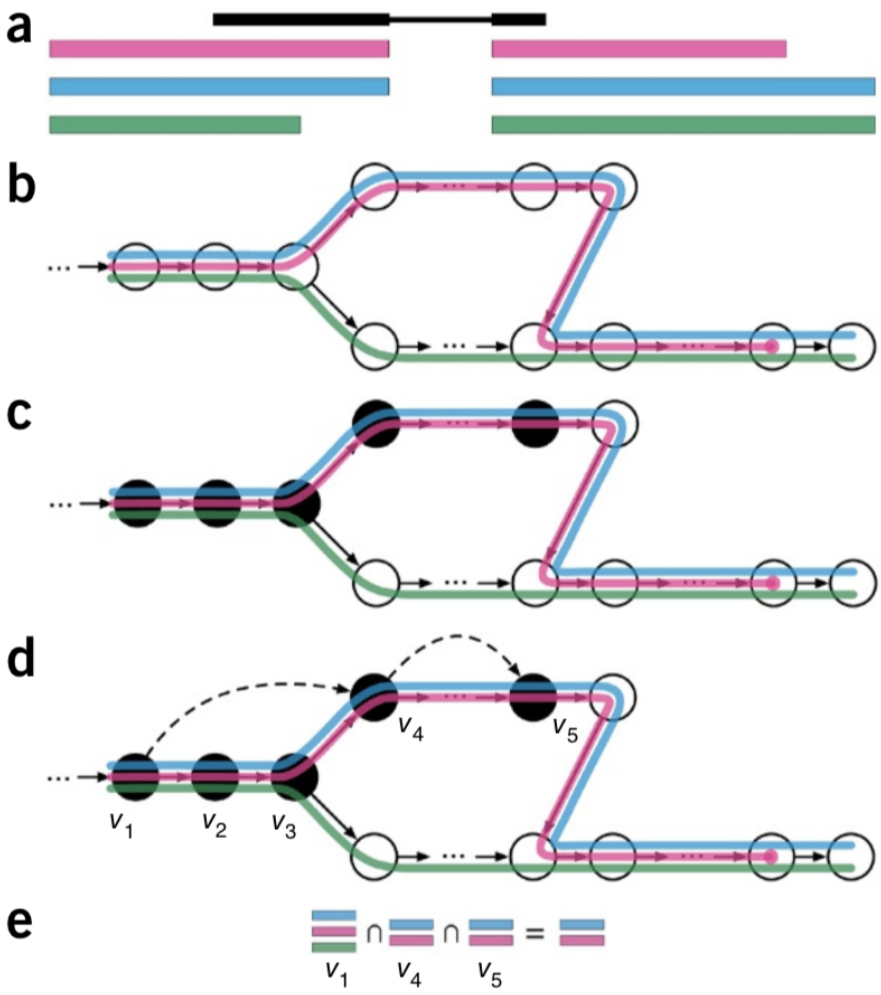


Figure 16: Overview of kallisto (from Bray et al., 2016).

Abundance estimation

- Counting reads per gene ignores the multi-mapping problem at the transcript level.
- Lightweight methods therefore **estimate** the transcript abundances with a statistical model.
- The number of reads in each equivalence class are sufficient statistics. The abundances are estimated by the EM algorithm, which assigns each read probabilistically to its compatible transcripts.
- The resulting counts are estimated counts, which are not necessarily integers.

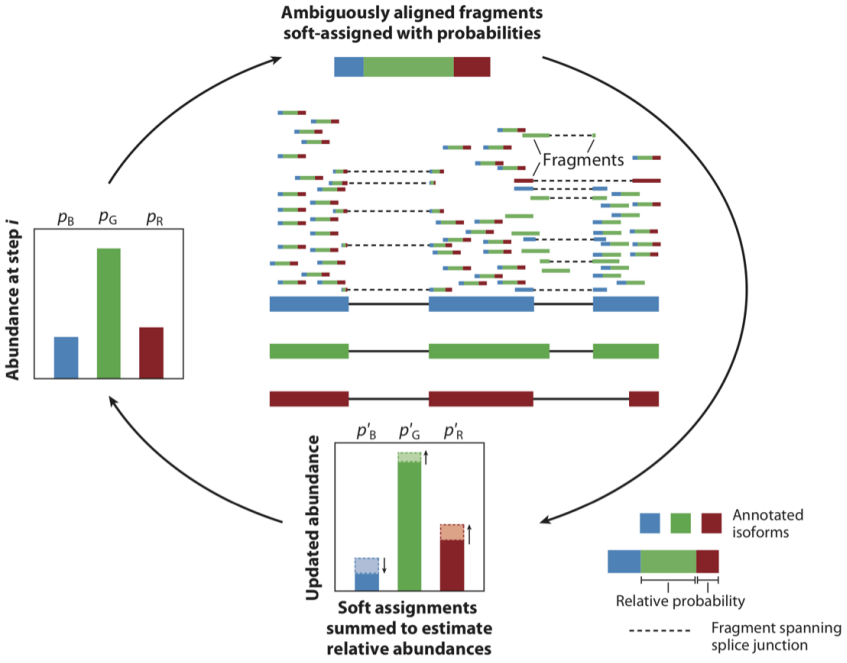


Figure 5

An illustration of the alignment of various reads to a gene with three isoforms: blue (B), green (G), and red (R). In this example, we wish to estimate the abundances of these isoforms, but most reads have ambiguous origins and need to be probabilistically assigned to the transcripts (relative probabilities for each read are shown by the magnitudes of the three colors). Some reads are consistent only with the B and G transcripts, and a few reads uniquely align to a single transcript (single color). In the expectation-maximization (or related) algorithm, given the current abundance estimates, fragments are probabilistically assigned to transcripts, and then estimated abundances are updated by summarizing the (proportional) allocations over all fragments; transcript abundance estimates are determined by iterating the procedure until convergence.

Figure 17: Abundance estimation with the EM algorithm (from Van den Berge et al., 2019).

Read-to-transcript ambiguity makes the estimated transcript counts more variable than gene counts. We come back to this when we analyse data at the transcript level in the technical details chapter.

Running salmon

With salmon, mapping and quantification takes two commands: building an index of the reference transcriptome, which only has to be done once, and quantifying each sample.

```
salmon index -t gencode.v29.transcripts.fa.gz -i gencode.v29_salmon_index --
gencode
```

```
salmon quant -i gencode.v29_salmon_index -l A \
  -1 SAMEA103884898_1.fastq.gz -2 SAMEA103884898_2.fastq.gz \
  --validateMappings --gcBias --numGibbsSamples 20 -o quants/SAMEA103884898
```

- `-l A`: salmon automatically detects the library type, e.g. whether the protocol is stranded.
- `--gcBias`: corrects for fragment GC-content bias.
- `--numGibbsSamples 20`: draws 20 Gibbs samples of the counts to quantify their uncertainty (see the technical details chapter).

We use the salmon quantifications of the macrophage data set, which we introduce below. They are distributed with the Bioconductor package macrophage.

```
macDir <- system.file("extdata", package = "macrophage")
list.files(file.path(macDir, "quants", "SAMEA103884898"))
```

```
[1] "aux_info"               "cmd_info.json"          "lib_format_counts.json"
[4] "libParams"              "logs"                   "quant.sf.gz"
```

Every sample has its own folder with a (here compressed) `quant.sf` file:

```
read.delim(file.path(macDir, "quants", "SAMEA103884898", "quant.sf.gz")) |>
  head()
```

```
                Name Length EffectiveLength      TPM NumReads
1 ENST00000456328.2   1657        1465.000 0.000000    0.000
2 ENST00000450305.2    632         440.000 0.000000    0.000
3 ENST00000488147.1   1351        1190.141 5.340816  144.212
4 ENST00000619216.1     68          15.000 0.000000    0.000
5 ENST00000473358.1    712         520.000 0.000000    0.000
6 ENST00000469289.1    535         344.000 0.000000    0.000
```

- `Length`: the length of the transcript.
- `EffectiveLength`: the number of positions where a fragment of typical length can start, corrected for sequence and GC bias.
- `TPM`: transcripts per million (see below).
- `NumReads`: the estimated number of reads (counts) for the transcript.

Abundance metrics

Let Y_{gi} be the count of feature g in sample i and $N_i = \sum_g Y_{gi}$ the sequencing depth (library size) of sample i.

- A sample that is sequenced deeper gives higher counts for all genes.
- A longer transcript produces more fragments than a shorter one at the same concentration.

Several metrics correct for these effects:

- **Counts per million**: $\text{CPM}_{gi} = \frac{Y_{gi}}{N_i} 10^6$.
- **Transcripts per million**: $\text{TPM}_{gi} = \frac{Y_{gi}/l_{gi}^{\text{eff}}}{\sum_{g'} Y_{g'i}/l_{g'i}^{\text{eff}}} 10^6$, with l_{gi}^{eff} the effective length. TPM estimates the relative concentration of a transcript in a sample.

For differential expression we model the **counts** and not CPM or TPM: the counts carry the information on the precision of the measurement. A count of 5 out of 1 million reads is much less precise than a count of 500 out of 100 million reads, while both have a CPM of 5. Differences in sequencing depth are accounted for in the model with offsets.

Other protocols: 3' tag sequencing and UMIs

- **3' tag sequencing** only sequences a short fragment at the 3' end of each molecule.
  - Each molecule gives one fragment, so there is no length bias.
  - There is no information on the isoforms.
- **Unique molecular identifiers (UMIs)** are random barcodes attached to each molecule before PCR amplification. Reads with the same UMI come from the same molecule, so PCR duplicates can be collapsed and molecules are counted instead of fragments.

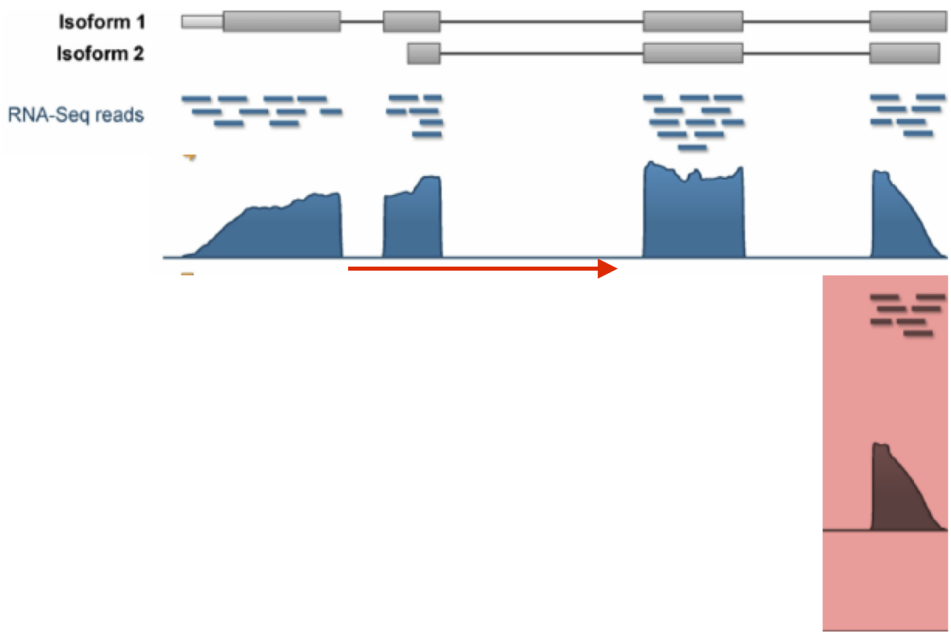


Figure 18: Full-length versus UMI-based protocols.

Both are central in single-cell RNA-seq, which we cover in the next module.

Modelling count data

Technical variability: the Poisson distribution

- A count of a gene can be seen as the number of fragments of that gene in a random sample of fragments from the library.
- If we sequence the same library repeatedly (**technical replicates**), the count of a gene follows approximately a Poisson distribution (Marioni et al. 2008):

$$Y_{gr} \sim \text{Poisson}(\mu_g), \quad E[Y_{gr}] = \text{Var}[Y_{gr}] = \mu_g.$$

- Count data thus have a **mean-variance relation**: the variance increases with the mean.

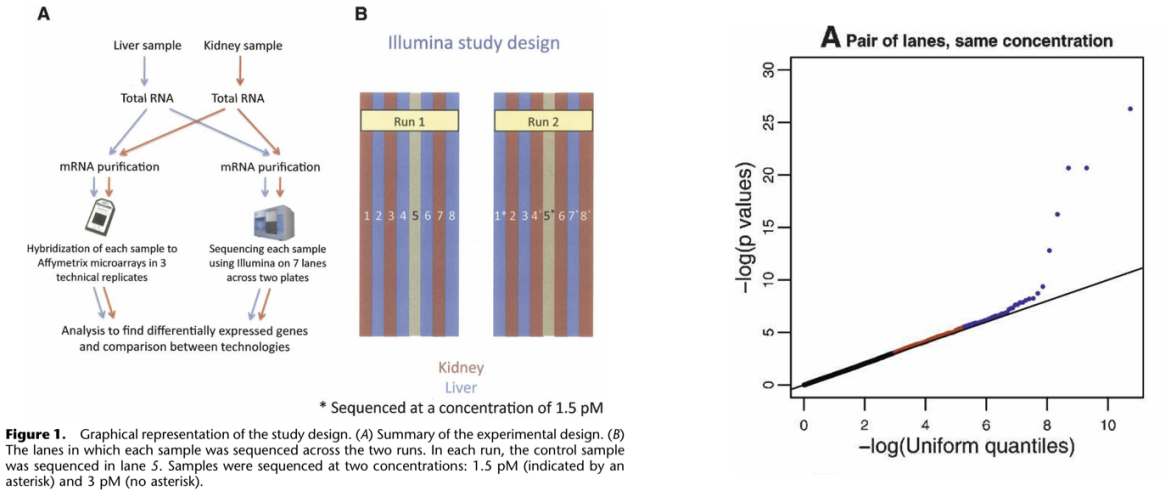


Figure 19: Left: design of the study, in which liver and kidney samples were sequenced in several lanes. Right: QQ-plot of the p-values of a test for a lane effect, comparing the counts of the same library in two lanes at the same concentration. Under the null hypothesis the reads in both lanes are random samples from the library (Poisson sampling), so the p-values are uniform and the points lie on the diagonal. From Marioni et al. (2008).

- Marioni et al. (2008) sequenced the same library in several lanes and tested every gene for a lane effect: do the counts in the lanes differ more than expected from random sampling of reads, given the sequencing depth of each lane?
- "No lane effect" thus means pure sampling variability, i.e. Poisson counts with a mean proportional to the sequencing depth. A gene-specific lane effect and extra-Poisson variability between technical replicates cannot be distinguished.
- The p-values are uniform for almost all genes: technical replicates only vary by sampling, consistent with the Poisson model.

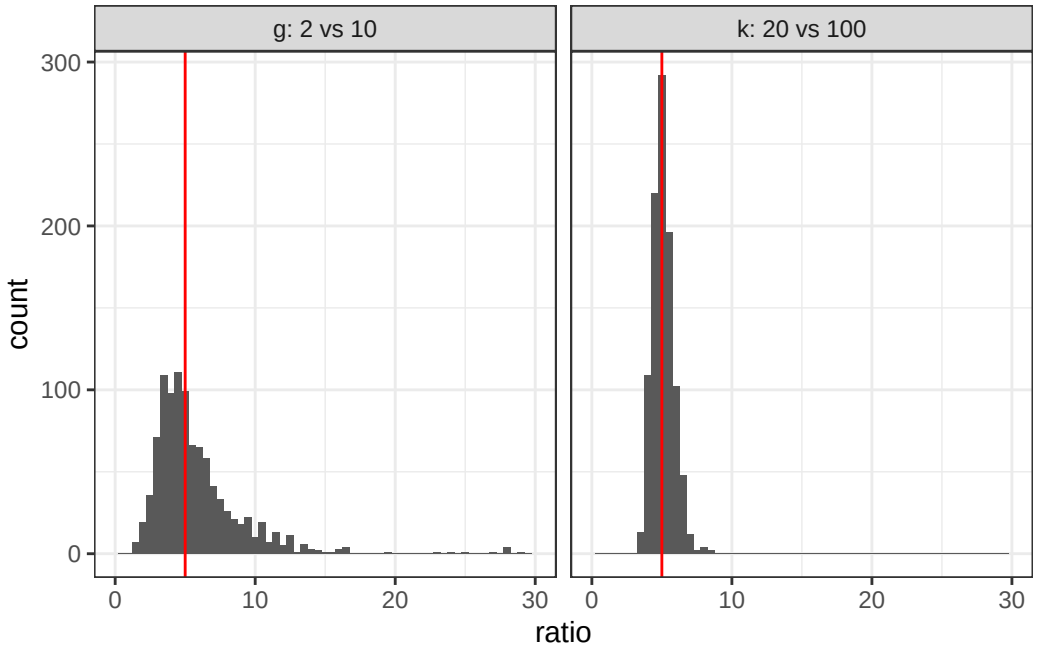
Relative uncertainty

Suppose that two genes have the same fold change of 5 between a tumour and a healthy tissue sample, each measured in three technical replicates:

- gene g: mean 2 in healthy and 10 in tumour tissue,
- gene k: mean 20 in healthy and 100 in tumour tissue.

For which gene is the fold change estimated more precisely?

```r
set.seed(11)
nSim <- 1000
foldChanges <- tibble(
  gene = rep(c("g: 2 vs 10", "k: 20 vs 100"), each = nSim),
  ratio = c(
    replicate(nSim, mean(rpois(3, 10)) / mean(rpois(3, 2))),
    replicate(nSim, mean(rpois(3, 100)) / mean(rpois(3, 20))))
)
foldChanges |>
  ggplot(aes(ratio)) +
  geom_histogram(binwidth = 0.5) +
  geom_vline(xintercept = 5, col = "red") +
  facet_wrap(~ gene) +
  xlim(0, 30) +
  theme_bw()
```



- The variance of the counts is larger for gene k, but the **relative** uncertainty is much smaller.
- The coefficient of variation of a Poisson count is $\text{CV} = \sqrt{\mu}/\mu = 1/\sqrt{\mu}$: 0.32 for a mean of 10 and 0.1 for a mean of 100.
- Fold changes of lowly expressed genes are thus much more noisy.

Why not a linear model on the counts?

The linear model of the first module assumes

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon, \quad \epsilon \sim N(\mathbf{0}, \sigma^2\mathbf{I}).$$

For counts:

- the variance is not constant but depends on the mean,
- counts are non-negative, while $\mathbf{X}\beta$ can be negative,
- low counts have a skewed distribution.

In proteomics we log-transformed the intensities. For counts, a log transformation leaves the data heteroscedastic and requires an arbitrary pseudo-count for zeros.

Generalised linear models

Generalised linear models (GLMs) extend the linear model in two ways:

1. The response follows a distribution of the **exponential family**, which includes the normal, binomial, Poisson and negative binomial distribution.
2. A **link function** $g(.)$ links the mean to the linear predictor: $g(E[Y_i]) = \mathbf{x}_i^T\beta$.

A Poisson GLM is defined as

$$\begin{cases} Y_i & \sim & \text{Poisson}(\mu_i) \\ \log \mu_i & = & \eta_i \\ \eta_i & = & \mathbf{x}_i^T\beta \end{cases}$$

- In the GLM we model $\log E[Y_i]$, the log of the mean. In a linear model on log-transformed counts we would model $E[\log Y_i]$, a geometric mean.
- The inverse link $\mu_i = \exp(\eta_i)$ ensures that the mean is positive.
- The parameters have a log fold change interpretation, as in proteomics.

How these models are fitted is discussed in the technical details chapter.

The parathyroid data

We illustrate the properties of count data on a subset of the parathyroid data set (Haglund et al. 2012). Tumour cultures of 4 patients were treated with diarylpropionitrile (DPN), 4-hydroxytamoxifen (OHT) or a control, and sequenced after 24 and 48 hours.

> Tissue for cell culturing was obtained from four chief cell parathyroid adenomas collected directly at surgery from female postmenopausal patients. Isolation of cells and culturing were performed essentially as previously described (22). Cells were plated and treated with 100 nM DPN (Tocris Bioscience, Minneapolis, MN) or 100 nM OHT (Sigma-Aldrich, St. Louis, MO) for 24 or 48 h, respectively. Untreated cells cultured in parallel were used as controls. Cells were harvested in RNeasy (QIAGEN AB, Hilden, Germany), and quality control was performed using Bioanalyzer (Agilent Technologies, Santa Clara, CA) and Nanodrop (Nanodrop Technology, Wilmington, DE) for all specimens. RNA samples were isolated from four different adenomas, and one sample (case 4, control 24 h) was omitted before transcriptome sequencing based on low RIN value. The entire sample set used for sequencing consisted of the treatment groups DPN 24 h (n = 4), DPN 48 h (n = 4), OHT 24 h (n = 4), OHT 48 h (n = 4), control 24 h (n = 3), and control 48 h (n = 4).

Figure 20: Design of the parathyroid experiment (Methods section of Haglund et al., 2012).

The data were distributed in the Bioconductor package `parathyroidSE`, which is no longer part of Bioconductor. We load the data directly from the archived package.

```
bfc <- BiocFileCache()
paraTar <- bfcrpath(bfc, "https://bioconductor.org/packages/3.21/data/experiment/src/contrib/
untar(paraTar, files = "parathyroidSE/data/parathyroidGenesSE.RData", exdir = tempdir())
load(file.path(tempdir(), "parathyroidSE/data/parathyroidGenesSE.RData"))
se <- parathyroidGenesSE
colData(se)
```

```
DataFrame with 27 rows and 8 columns
            run experiment  patient treatment     time submission     study
    <character>   <factor> <factor>  <factor> <factor>   <factor>  <factor>
```

```
1     SRR479052  SRX140503        1   Control      24h  SRA051611 SRP012167
2     SRR479053  SRX140504        1   Control      48h  SRA051611 SRP012167
3     SRR479054  SRX140505        1   DPN          24h  SRA051611 SRP012167
4     SRR479055  SRX140506        1   DPN          48h  SRA051611 SRP012167
5     SRR479056  SRX140507        1   OHT          24h  SRA051611 SRP012167
...         ...        ...      ...       ...      ...        ...       ...
23    SRR479074  SRX140523        4       DPN      48h  SRA051611 SRP012167
24    SRR479075  SRX140523        4       DPN      48h  SRA051611 SRP012167
25    SRR479076  SRX140524        4       OHT      24h  SRA051611 SRP012167
26    SRR479077  SRX140525        4       OHT      48h  SRA051611 SRP012167
27    SRR479078  SRX140525        4       OHT      48h  SRA051611 SRP012167
       sample
     <factor>
1   SRS308865
2   SRS308866
3   SRS308867
4   SRS308868
5   SRS308869
...       ...
23  SRS308885
24  SRS308885
25  SRS308886
26  SRS308887
27  SRS308887
```

Some samples (`experiment`) were sequenced in two runs: these are technical replicates.

```
techRep <- colData(se) |>
  as.data.frame() |>
  filter(duplicated(experiment)) |>
  pull(experiment)
techRep
```

```
[1] SRX140511 SRX140513 SRX140523 SRX140525
23 Levels: SRX140503 SRX140504 SRX140505 SRX140506 SRX140507 ... SRX140525
```

Technical replicates and normalisation

We compare the two runs of the first sample with technical replicates in an MA plot: the log2 fold change M against the average log2 count A.

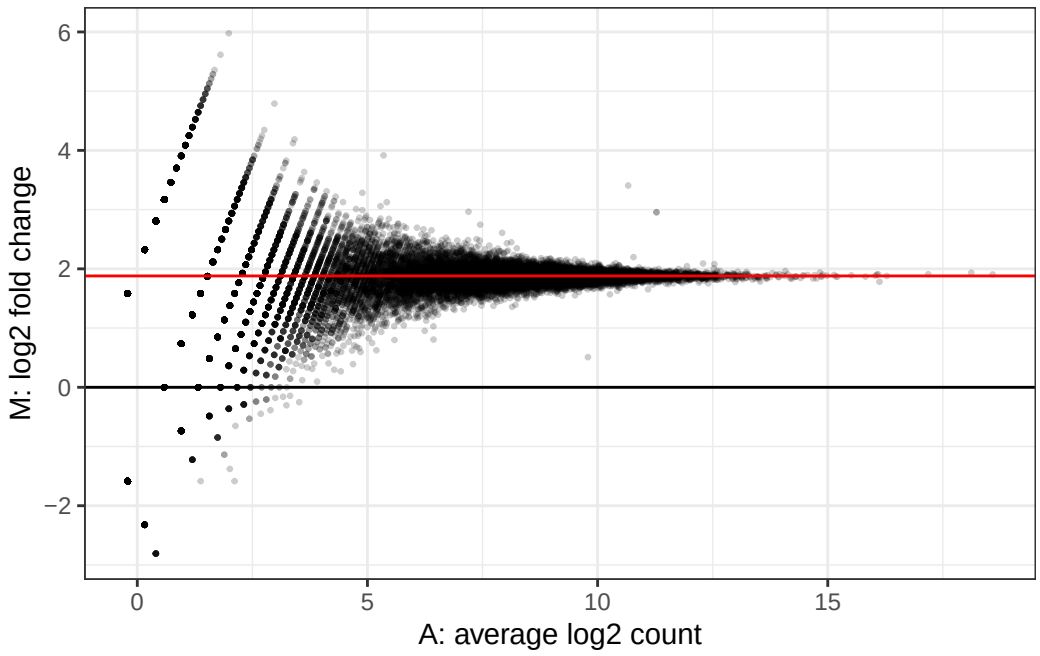
```
techCounts <- assay(se)[, colData(se)$experiment == techRep[1]]
colSums(techCounts)
```

```
[1]  5251911 19332369
```

```
maData <- tibble(
  A = rowMeans(log2(techCounts + 0.5)),
  M = log2(techCounts[, 2] + 0.5) - log2(techCounts[, 1] + 0.5)
) |>
  filter(rowSums(techCounts) > 0)

logFCdepth <- diff(log2(colSums(techCounts)))

maData |>
  ggplot(aes(A, M)) +
  geom_point(alpha = 0.2, size = 0.5) +
  geom_hline(yintercept = 0) +
  geom_hline(yintercept = logFCdepth, col = "red") +
  xlab("A: average log2 count") +
  ylab("M: log2 fold change") +
  theme_bw()
```



- The second run has about 3.7 times more reads than the first.
- The log fold changes of the genes fluctuate around the log fold change of the library sizes (red line), not around zero.
- The spread of M is much larger for low counts: the Poisson relative uncertainty.

We correct for differences in sequencing depth with an **offset** in the GLM:

$$\log \mu_{gi} = \mathbf{x}_i^T \beta_g + \log N_i \quad \Leftrightarrow \quad \log \frac{\mu_{gi}}{N_i} = \mathbf{x}_i^T \beta_g.$$

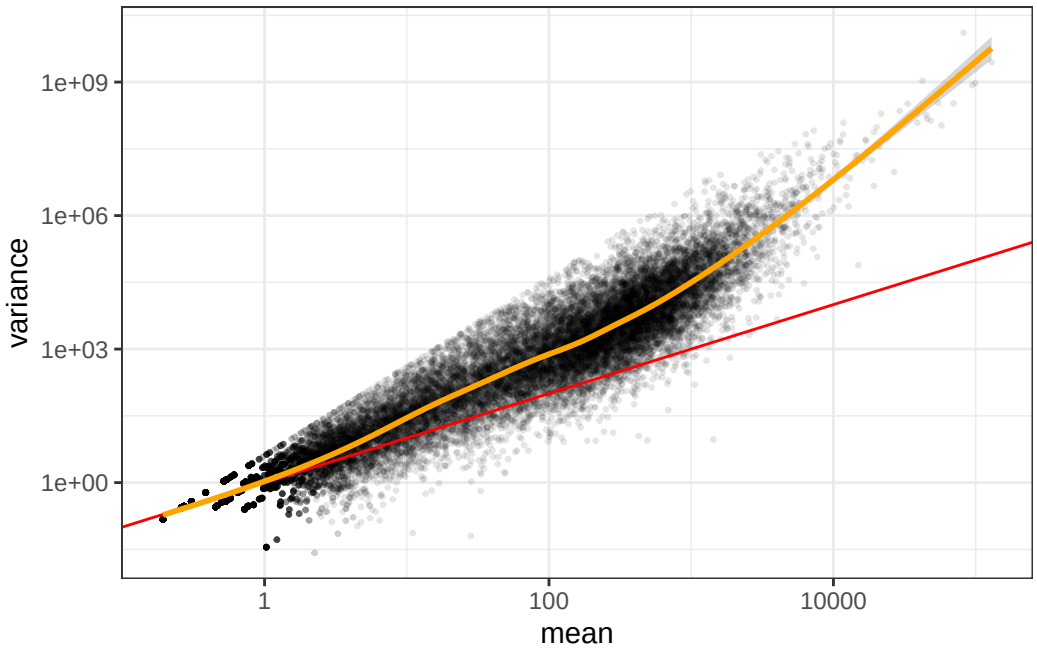
- An offset is a covariate with a known coefficient of 1: the parameters now model the mean count per read sequenced.
- A sample that is sequenced deeper contributes more information, because its counts are larger and thus relatively more precise.
- Better offsets than the raw library size exist: e.g. the TMM method of edgeR corrects for differences in RNA composition between samples (Robinson and Oshlack 2010). The idea is the same as the median of ratios normalisation in proteomics: most genes are assumed not to change. Details are in the technical details chapter.
- Technical replicates are Poisson and the sum of Poisson counts is again Poisson. We can therefore **sum** the counts of technical replicates.

Biological replicates: overdispersion

We look at the four biological replicates of the control treatment at 48 hours.

```
bioReps <- colData(se)$treatment == "Control" & colData(se)$time == "48h"
bioCounts <- assay(se)[, bioReps]
bioCounts <- bioCounts[rowSums(bioCounts) > 0, ]
cpmBio <- sweep(bioCounts, 2, colSums(bioCounts), "/") * mean(colSums(bioCounts))

tibble(mean = rowMeans(cpmBio), variance = rowVars(cpmBio)) |>
  filter(variance > 0) |>
  ggplot(aes(mean, variance)) +
  geom_point(alpha = 0.1, size = 0.5) +
  geom_abline(intercept = 0, slope = 1, col = "red") +
  geom_smooth(col = "orange") +
  scale_x_log10() +
  scale_y_log10() +
  theme_bw()
```



- With thousands of genes we can assess the mean-variance relation empirically.
- The data are **overdispersed** with respect to the Poisson distribution (red line): biological replicates differ more than technical replicates.
- The variance increases approximately quadratically with the mean.

The negative binomial distribution

This motivates the negative binomial (NB) distribution:

$$\begin{cases} Y_{gi} & \sim & NB(\mu_{gi}, \phi_g) \\ \log \mu_{gi} & = & \mathbf{x}_i^T \beta_g + \log N_i \end{cases} \quad \text{with} \quad \text{Var}[Y_{gi}] = \mu_{gi} + \phi_g \mu_{gi}^2.$$

		Sequencing		Biology
total variability	=	technical variability	+	biological variability
$\text{Var}[Y_{gi}]$	=	μ_{gi}	+	$\phi_g \mu_{gi}^2$
total CV^2	=	$1/\mu_{gi}$	+	ϕ_g

- ϕ_g is the squared **biological coefficient of variation** (BCV).
- For low counts the technical (Poisson) variability dominates: a higher sequencing depth helps.

- For high counts the biological variability dominates: only more biological replicates help.
- With few replicates per gene, ϕ_g is estimated very imprecisely. Like in proteomics, we will borrow information across genes.

Gene-specific variability: quasi-likelihood and empirical Bayes

The current version of edgeR (Chen et al. 2025) uses a quasi-negative binomial model (Lund et al. 2012):

$$\begin{cases} E[Y_{gi}] &= \mu_{gi} \\ \log \mu_{gi} &= \mathbf{x}_i^T \beta_g + \log N_i^{\text{eff}} \\ \text{Var}[Y_{gi}] &= \sigma_g^2 \left(\mu_{gi} + \phi \mu_{gi}^2\right) \end{cases}$$

- ϕ is a single NB dispersion for all genes: it describes the global shape of the mean-variance relation.
- σ_g^2 is a gene-specific **quasi-likelihood dispersion**, which captures how much more (or less) variable a gene is than the global trend.
- σ_g^2 is estimated from the residual deviance divided by its degrees of freedom: the GLM counterpart of the residual variance s_g^2 in a linear model.
- With few replicates σ_g^2 is unstable. It is therefore shrunken towards a common trend with **empirical Bayes**, using the same `squeezeVar` function that msqrob2 uses.
- Inference uses moderated F-tests (moderated t-tests for a single contrast).

This is the same strategy as in the proteomics module:

	msqrob2 (proteomics)	edgeR quasi-likelihood (RNA-seq)
Data	log2 intensities	counts
Model per feature	robust linear (mixed) model	quasi-NB GLM with log link
Normalisation	median of ratios, centring	offsets: effective library sizes (TMM) and average transcript lengths
Feature-specific variability	residual variance s_g^2	QL dispersion σ_g^2
Empirical Bayes	`squeezeVar`: shrink s_g^2 to a common prior	`squeezeVar`: shrink σ_g^2 to a trend in the average expression
Test	moderated t	moderated F / t
Multiple testing	FDR (Benjamini-Hochberg)	FDR (Benjamini-Hochberg)

The details are covered in the technical details chapter.

DESeq2 (Love et al. 2014) is the other widely used tool.

- It uses the same NB GLM
- But estimates a dispersion ϕ_g per gene (similarly to edgeR prior to version 4), shrinks it with empirical Bayes towards a trend, and uses Wald tests (relying on asymptotic normality).
- Both tools typically give similar results.
- The quasi-likelihood F-tests of edgeR account for the uncertainty in the estimated dispersion and therefore control the type I error better with few replicates.

limma-voom (Law et al. 2014) takes a different route:

- it does not model the counts, but the log2 CPM values with linear models, as we did for the log2 intensities in proteomics.
- Log2 CPM values are still heteroscedastic: low counts are more variable.
- voom estimates the mean-variance trend of the log2 CPM values and turns it into a precision weight for every observation.
- The weighted log2 CPM values are analysed with the limma pipeline: linear models, empirical Bayes moderation with `squeezeVar` and moderated t-tests.
- voom is fast, works well with sufficient replicates and gives access to the full limma toolbox, e.g. random effects via `duplicateCorrelation` for designs with repeated measures.
- voom becomes problematic for small counts: on the log scale, counts of 0, 1 or 2 become a few discrete values that depend on the prior count, and their distribution is far from normal, which precision weights cannot repair.
- edgeR models the counts directly and handles small counts better.
- This becomes even more important for single-cell data, where most counts are very small.

A complete workflow: the macrophage data

Background

Macrophages are immune cells that detect and engulf pathogens. Alasoo et al. (2018) differentiated macrophages from induced pluripotent stem cells (iPSCs) of a large panel of donors and exposed them to four conditions:

- naive: no stimulation,
- IFNg: priming with interferon gamma (IFNγ), a cytokine that activates macrophages, for 18 hours,
- SL1344: infection with *Salmonella enterica* serovar Typhimurium (strain SL1344) for 5 hours,
- IFNg_SL1344: priming with IFNγ followed by infection with *Salmonella*.

The Bioconductor package macrophage contains the salmon quantifications of the 24 samples of 6 donors (ENA: PRJEB18997).

Research questions:

1. Which genes respond to IFNγ priming in uninfected macrophages?
2. Which genes respond to IFNγ priming in infected macrophages?
3. Which genes respond to *Salmonella* infection in naive macrophages?
4. Which genes respond to *Salmonella* infection in IFNγ-primed macrophages?
5. For which genes does IFNγ priming change the response to the infection?

The last question is the core of the study: IFNγ primes the macrophages, e.g. by opening enhancers, so that they respond differently to the pathogen (Alasoo et al. 2018).

- What is the experimental unit?
- What is the design?

```
coldata <- read.csv(file.path(macDir, "coldata.csv")) |>
  transmute(
    names,
    files = file.path(macDir, "quants", names, "quant.sf.gz"),
    donor = factor(line_id),
    condition = factor(condition_name, levels = c("naive", "IFNg", "SL1344", "IFNg_SL1344"))
    ifng = factor(ifelse(grepl("IFNg", condition_name), "yes", "no")),
    salmonella = factor(ifelse(grepl("SL1344", condition_name), "yes", "no"))
  )
coldata |>
  dplyr::select(-files)
```

```
            names  donor   condition ifng salmonella
1  SAMEA103885102 diku_1       naive   no         no
2  SAMEA103885347 diku_1        IFNg  yes         no
3  SAMEA103885043 diku_1      SL1344   no        yes
4  SAMEA103885392 diku_1 IFNg_SL1344  yes        yes
5  SAMEA103885182 eiwy_1       naive   no         no
6  SAMEA103885136 eiwy_1        IFNg  yes         no
7  SAMEA103885413 eiwy_1      SL1344   no        yes
8  SAMEA103884967 eiwy_1 IFNg_SL1344  yes        yes
9  SAMEA103885368 fikt_3       naive   no         no
10 SAMEA103885218 fikt_3        IFNg  yes         no
11 SAMEA103885319 fikt_3      SL1344   no        yes
12 SAMEA103885004 fikt_3 IFNg_SL1344  yes        yes
13 SAMEA103885284 ieki_2       naive   no         no
14 SAMEA103885059 ieki_2        IFNg  yes         no
15 SAMEA103884898 ieki_2      SL1344   no        yes
16 SAMEA103885157 ieki_2 IFNg_SL1344  yes        yes
```

```
17 SAMEA103885111 podx_1       naive   no         no
18 SAMEA103884919 podx_1        IFNg  yes         no
19 SAMEA103885276 podx_1      SL1344   no        yes
20 SAMEA103885021 podx_1 IFNg_SL1344  yes        yes
21 SAMEA103885262 qaqx_1       naive   no         no
22 SAMEA103885228 qaqx_1        IFNg  yes         no
23 SAMEA103885308 qaqx_1      SL1344   no        yes
24 SAMEA103884949 qaqx_1 IFNg_SL1344  yes        yes
```

```
table(coldata$ifng, coldata$salmonella)
```

```
     no yes
  no  6   6
  yes 6   6
```

- The cells of every donor are exposed to all four conditions: the donors are blocks, as in the mouse T-cell example of the proteomics module.
- The treatments have a 2×2 **factorial** structure: IFNγ priming (no/yes) and *Salmonella* infection (no/yes).

Data import

The tximeta package (Love et al. 2020) imports salmon quantifications into a `SummarizedExperiment`. It recognises the reference transcriptome that was used for the quantification and adds the transcript annotation. The quantifications also contain 20 Gibbs samples per sample; we only need them for the transcript-level analyses in the technical details chapter, so we do not import them here.

```
st <- tximeta(coldata, dropInfReps = TRUE)
```

```
st
```

```
class: RangedSummarizedExperiment
dim: 205870 24
metadata(6): tximetaInfo quantInfo ... txomeInfo txdbInfo
assays(3): counts abundance length
rownames(205870): ENST00000456328.2 ENST00000450305.2 ...
  ENST00000387460.2 ENST00000387461.2
rowData names(3): tx_id gene_id tx_name
```

```
colnames(24): SAMEA103885102 SAMEA103885347 ... SAMEA103885308
  SAMEA103884949
colData names(5): names donor condition ifng salmonella
```

We summarise the transcript-level counts to the gene level, and add gene symbols and Entrez IDs.

```
sg <- summarizeToGene(st)
sg <- addIds(sg, "SYMBOL", gene = TRUE)
sg <- addIds(sg, "ENTREZID", gene = TRUE)
sg
```

```
class: RangedSummarizedExperiment
dim: 58294 24
metadata(7): tximetaInfo quantInfo ... txdbInfo assignRanges
assays(3): counts abundance length
rownames(58294): ENSG00000000003.14 ENSG00000000005.5 ...
  ENSG00000285993.1 ENSG00000285994.1
rowData names(4): gene_id tx_ids SYMBOL ENTREZID
colnames(24): SAMEA103885102 SAMEA103885347 ... SAMEA103885308
  SAMEA103884949
colData names(5): names donor condition ifng salmonella
```

- A `SummarizedExperiment` stores the assays (counts, abundances (TPM) and average transcript lengths), the sample annotation in `colData` and the feature annotation in `rowData`.
- We keep the stable Ensembl gene IDs as row names and use the symbols for interpretation: symbols are not unique and change over time.

Transcript length

The gene-level counts are the sums of the estimated counts of the transcripts of a gene. `summarizeToGene` also computes, for every gene and sample, the **average transcript length**: the average of the effective lengths of the transcripts of the gene, weighted by their abundance.

- Within a gene, the length is the same in all samples as long as the gene uses its transcripts in the same proportions. It then cancels in comparisons between samples.
- If the usage of the transcripts changes, e.g. the treatment shifts the expression of a gene to a longer isoform, the gene produces more fragments without containing more molecules. Summed counts then suggest a change in expression that is due to the change in transcript length.

- Soneson et al. (2015) showed that this leads to false positives in differential gene expression analyses on summed counts, and that it is solved by using the average transcript length as an additional offset in the model.

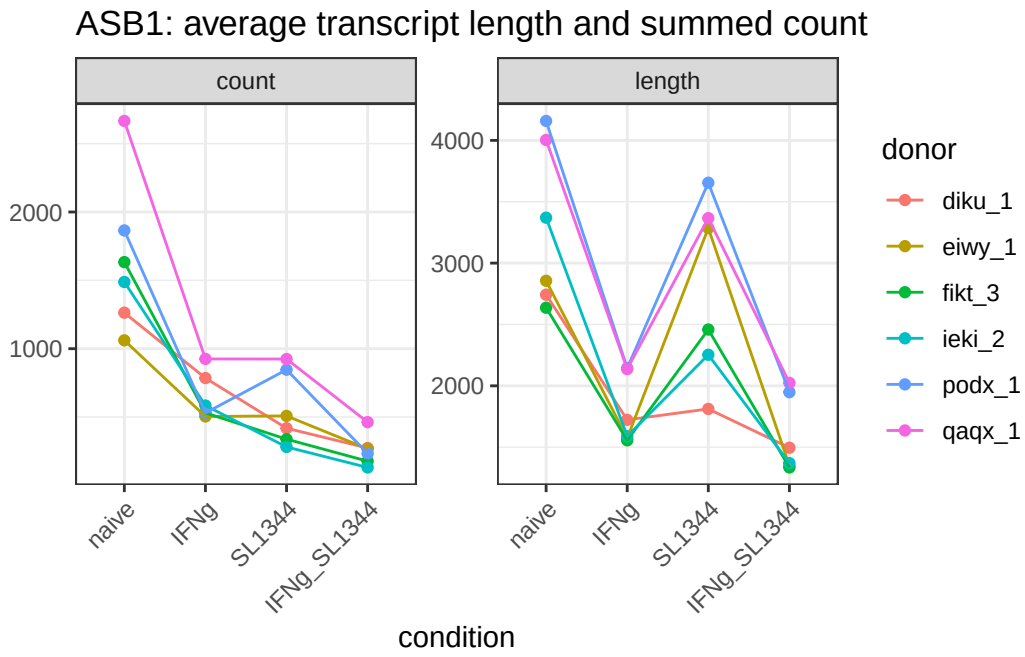
The average transcript lengths are stored in the `length` assay:

```
assay(sg, "length")[1:3, 1:4]
```

```
                  SAMEA103885102 SAMEA103885347 SAMEA103885043 SAMEA103885392
ENSG00000000003.14      1991.0755      2013.4932       1970.577      1997.0522
ENSG00000000005.5        731.1250       731.1250        731.125       731.1250
ENSG00000000419.12       814.8069       819.1806        819.849       819.8217
```

For most genes they hardly vary between the samples, but for some genes they change with the treatment, e.g. *ASB1*:

```
asb1 <- rownames(sg)[which(rowData(sg)$SYMBOL == "ASB1")]
tibble(
  length = assay(sg, "length")[asb1, ],
  count = assay(sg, "counts")[asb1, ],
  donor = sg$donor,
  condition = sg$condition
) |>
  pivot_longer(c(length, count), names_to = "assay") |>
  ggplot(aes(condition, value, group = donor, color = donor)) +
  geom_point() +
  geom_line() +
  facet_wrap(~ assay, scales = "free_y") +
  ggtitle("ASB1: average transcript length and summed count") +
  ylab(NULL) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



IFNγ shifts the expression of *ASB1* to shorter transcripts: the average transcript length almost halves. Shorter transcripts produce fewer fragments per molecule, so the summed counts drop as well.

`DGEListFromTximeta` creates the `DGEList` for edgeR and stores the centred log average transcript lengths in `offset.prior`. edgeR adds them to the log effective library sizes, so that every model below automatically corrects for the average transcript length. This is the edgeR counterpart of DESeq2, which uses the average transcript lengths as offsets when it is given a tximeta object.

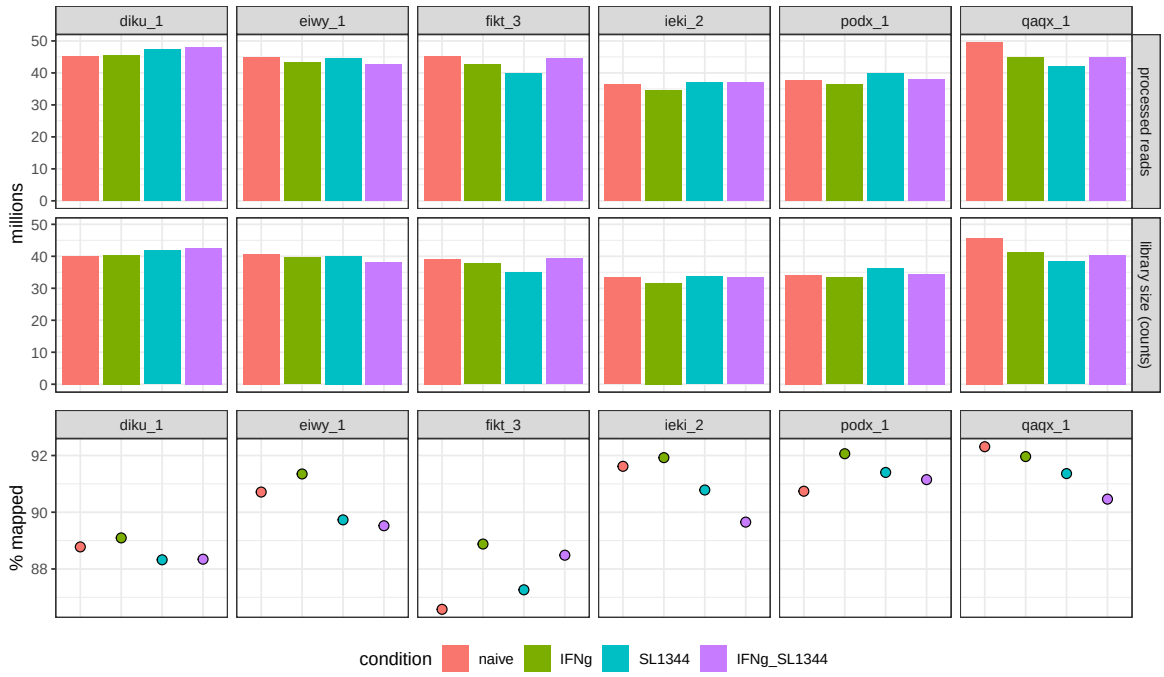
Filtering and normalisation

We store the counts in a `DGEList`, the data container of edgeR.

```
dge <- DGEListFromTximeta(sg)
dge$genes <- dge$genes[, c("SYMBOL", "ENTREZID")]
colnames(dge) <- paste(sg$donor, sg$condition, sep = "_")
design <- model.matrix(~ donor + ifng * salmonella, data = colData(sg))
```

A first quality check is the sequencing depth of the samples. tximeta also stores the run information of salmon, e.g. the number of reads (fragments) that were processed and the percentage that could be mapped to the transcriptome. The library size in the `DGEList` is the sum of the estimated counts over all genes.

```
noXaxis <- theme(axis.text.x = element_blank(), axis.ticks.x = element_blank(),
                 axis.title.x = element_blank())
qcMac <- tibble(
  donor = sg$donor,
  condition = sg$condition,
  processed = metadata(sg)$quantInfo$num_processed / 1e6,
  percentMapped = metadata(sg)$quantInfo$percent_mapped,
  libSize = dge$samples$lib.size / 1e6
)
libPlot <- qcMac |>
  pivot_longer(c(processed, libSize), names_to = "type", values_to = "millions") |>
  mutate(type = factor(type, levels = c("processed", "libSize"),
                       labels = c("processed reads", "library size (counts)"))) |>
  ggplot(aes(condition, millions, fill = condition)) +
  geom_col() +
  facet_grid(type ~ donor) +
  ylab("millions") +
  theme_bw() +
  noXaxis
mapPlot <- ggplot(qcMac, aes(condition, percentMapped, fill = condition)) +
  geom_point(shape = 21, size = 2.5) +
  guides(fill = "none") +
  facet_grid(~ donor) +
  ylab("% mapped") +
  theme_bw() +
  noXaxis
libPlot / mapPlot + plot_layout(heights = c(2, 1), guides = "collect") &
  theme(legend.position = "bottom")
```



- All samples were sequenced to a depth of 34 to 50 million reads, and 87% to 92% of the reads could be assigned to the transcriptome.
- The differences in depth are moderate (at most a factor 1.4 in library size) and are not related to the treatment: the offsets in the model correct for them.
- A sample with a much lower depth or mapping rate than the others should be inspected: it has less information, and a low mapping rate can point to contamination, degraded RNA or a wrong reference.

Genes with very low counts carry almost no information and are removed. `filterByExpr` keeps genes with a sufficiently large count per million in a minimum number of samples, based on the design.

```
keep <- filterByExpr(dge, design)
table(keep)
```

```
keep
FALSE  TRUE
39070 19224
```

```
dge <- dge[keep, , keep.lib.sizes = FALSE]
```

The effective library sizes are computed with TMM normalisation.

```
dge <- normLibSizes(dge)
head(dge$samples)
```

```
                  group lib.size norm.factors
diku_1_naive          1 40051744    1.3582924
diku_1_IFNg           1 40444735    1.3032501
diku_1_SL1344         1 41811892    0.9603890
diku_1_IFNg_SL1344    1 42515031    0.8000083
eiwy_1_naive          1 40723483    1.2476363
eiwy_1_IFNg           1 39689006    1.0866330
```

`dge$offset.prior` contains the centred log average transcript lengths. The offsets that edgeR uses in all models are the log effective library sizes plus these length offsets:

```
dge$offset.prior[1:3, 1:4]
```

```
                  SAMEA103885102 SAMEA103885347 SAMEA103885043 SAMEA103885392
ENSG00000000003.14     0.01922292     0.03041910     0.00887437     0.02222015
ENSG00000000419.12     0.02387938     0.02923286     0.03004843     0.03001511
ENSG00000000457.13     0.16984185     0.06425038     0.07363709    -0.03839328
```

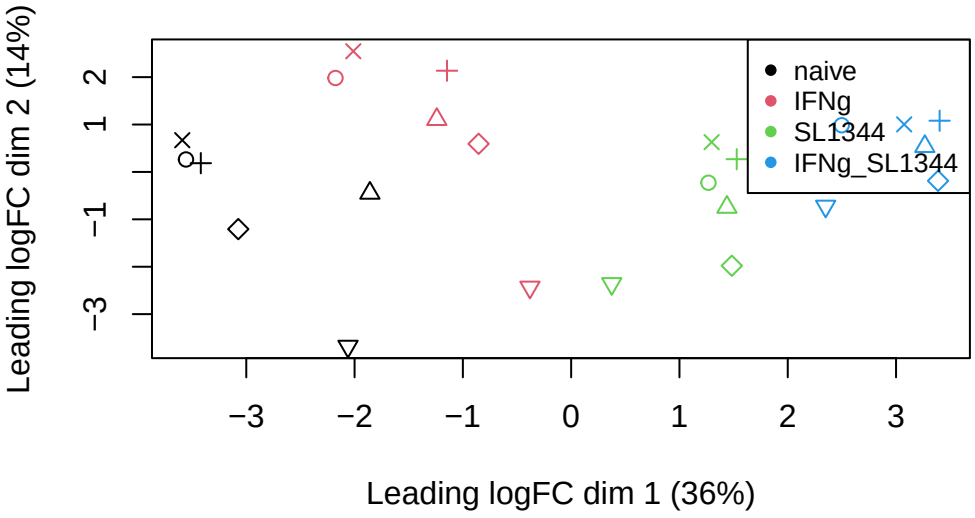
```
logEffLibSize <- log(dge$samples$lib.size * dge$samples$norm.factors)
all.equal(
  unname(getOffset(dge)),
  unname(sweep(dge$offset.prior, 2, logEffLibSize, "+"))
)
```

```
[1] TRUE
```

Exploratory data analysis

We visualise the samples with multidimensional scaling (MDS) on the log2 CPM values. The distance between two samples is the typical log2 fold change between them for the 500 genes that differ most.

```
plotMDS(dge, col = as.double(sg$condition), pch = as.double(sg$donor), labels = NULL)
legend("topright", legend = levels(sg$condition), col = 1:4, pch = 16, cex = 0.8)
```



- The first dimension separates the infected from the uninfected samples: *Salmonella* is the largest source of variability in the experiment.
- Within infection status, the IFNγ-primed samples are shifted to the right.
- The second dimension mainly reflects differences between the donors (symbols): there is a block effect.

Design

The model for the log mean of gene g in sample i contains a block effect for the donor and the main effects of IFNγ and *Salmonella* and their interaction:

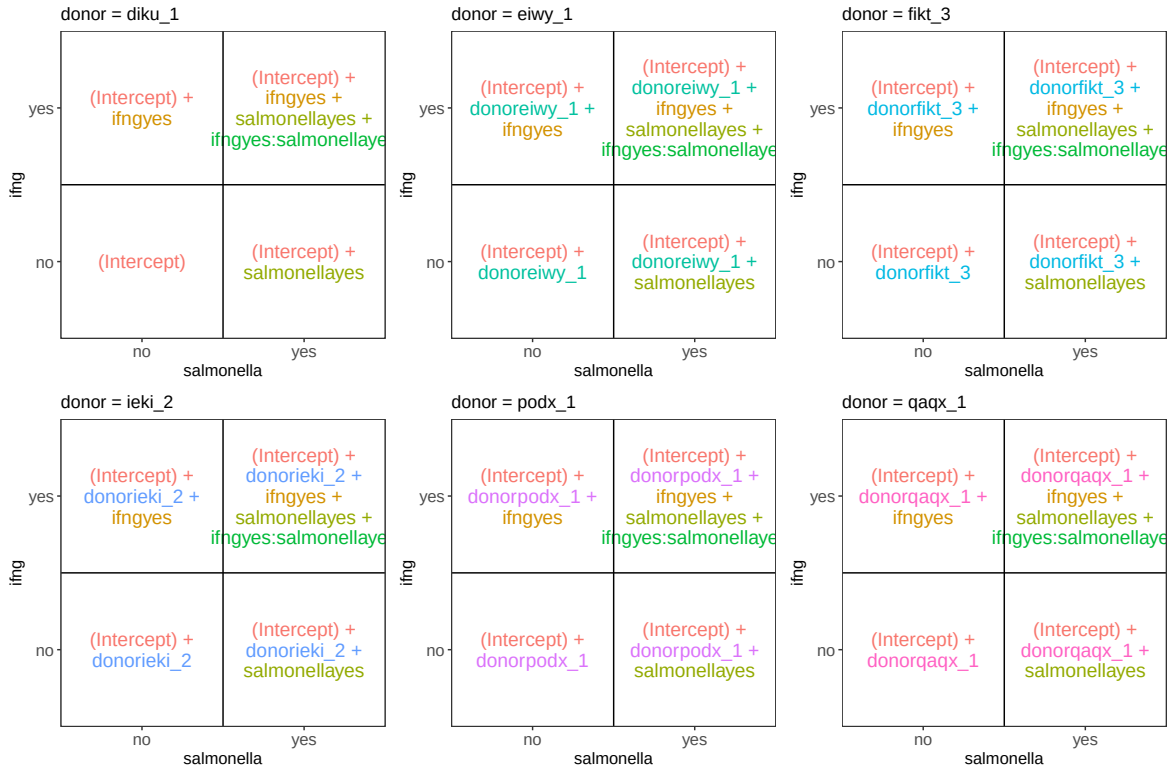
$$\log \mu_{gi} = \beta_{g0} + \beta_{g,\text{donor}(i)} + \beta_{g,\text{IFN}\gamma}\, x_{i,\text{IFN}\gamma} + \beta_{g,\text{Sal}}\, x_{i,\text{Sal}} + \beta_{g,\text{IFN}\gamma:\text{Sal}}\, x_{i,\text{IFN}\gamma}\, x_{i,\text{Sal}} + \log N_i^{\text{eff}},$$

with $x_{i,\text{IFN}\gamma}$ and $x_{i,\text{Sal}}$ indicator variables for IFNγ priming and *Salmonella* infection.

```
colnames(design)
```

```
[1] "(Intercept)"          "donoreiwy_1"          "donorfikt_3"
[4] "donorieki_2"          "donorpodx_1"          "donorqaqx_1"
[7] "ifngyes"              "salmonellayes"        "ifngyes:salmonellayes"
```

```
vd <- VisualizeDesign(colData(sg) |> as.data.frame(),
                      designFormula = ~ donor + ifng * salmonella)
wrap_plots(vd$plotlist, ncol = 3)
```



- Every donor has its own 2×2 panel with the four treatment combinations.
- The first donor (`diku_1`) is the reference: its naive samples are modelled by the intercept alone.
- For the other donors, the donor effect (e.g. `donoreiwy_1`) is added to all four cells of the panel: it shifts the whole panel by the same amount.
- The treatment coefficients `ifngyes`, `salmonellayes` and `ifngyes:salmonellayes` appear in the same cells of every panel: the treatment effects are assumed to be the same in every donor.
- Hence, each treatment effect is estimated within donors, and the donor-to-donor variability in baseline expression drops out.
- `ifngyes`: log fold change of IFNγ priming in uninfected cells.
- `salmonellayes`: log fold change of *Salmonella* infection in naive cells.

- `ifngyes:salmonellayes`: interaction, i.e. the difference between the IFNγ effect in infected and in uninfected cells.
- The IFNγ effect in infected cells is the sum `ifngyes` + `ifngyes:salmonellayes`.
- The *Salmonella* effect in IFNγ-primed cells is the sum `salmonellayes` + `ifngyes:salmonellayes`.
- The interaction can be read in two equivalent ways:

$$\begin{aligned}\beta_{g,\text{IFN}\gamma:\text{Sal}} &= (\text{Sal effect in primed cells}) - (\text{Sal effect in naive cells}) \\ &= (\text{IFN}\gamma \text{ effect in infected cells}) - (\text{IFN}\gamma \text{ effect in uninfected cells}).\end{aligned}$$

  "IFNγ priming changes the response to the infection" and "the infection changes the response to IFNγ" are the same hypothesis.

We encode the five research questions as contrasts, i.e. linear combinations of the model parameters:

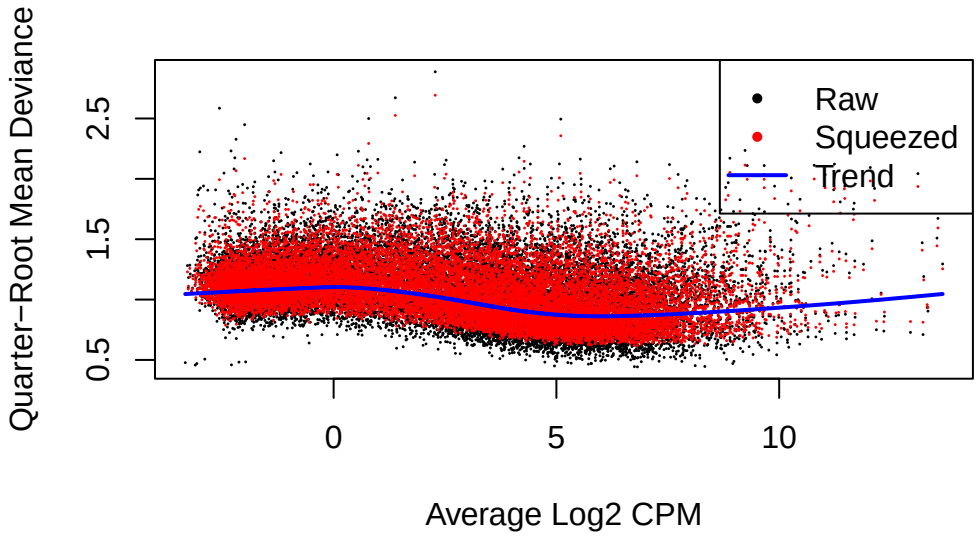
	research question	contrast
1	`IFNg`	`ifngyes`
2	`IFNg_infected`	`ifngyes + ifngyes:salmonellayes`
3	`Salmonella`	`salmonellayes`
4	`Salmonella_primed`	`salmonellayes + ifngyes:salmonellayes`
5	`interaction`	`ifngyes:salmonellayes`

```
L <- matrix(0, nrow = ncol(design), ncol = 5,
            dimnames = list(colnames(design),
                            c("IFNg", "IFNg_infected", "Salmonella",
                              "Salmonella_primed", "interaction")))
L["ifngyes", c("IFNg", "IFNg_infected")] <- 1
L["salmonellayes", c("Salmonella", "Salmonella_primed")] <- 1
L["ifngyes:salmonellayes", c("IFNg_infected", "Salmonella_primed", "interaction")] <- 1
L[-(1:6), ]
```

```
                     IFNg IFNg_infected Salmonella Salmonella_primed
ifngyes                 1             1          0                 0
salmonellayes           0             0          1                 1
ifngyes:salmonellayes   0             1          0                 1
                     interaction
ifngyes                        0
salmonellayes                  0
ifngyes:salmonellayes          1
```

Model fit

```
fit <- glmQLFit(dge, design)
plotQLDisp(fit)
```



- The black dots are the raw QL dispersions $\hat{\sigma}_g^2$ of the genes; the red dots are the squeezed QL dispersions; the blue line is the trend.
- As in msqrob2, the squeezed dispersions are a compromise between the gene-wise estimates and the trend.

```
fit$df.prior
```

```
[1] 3.903063
```

The prior degrees of freedom express how much information is borrowed from the trend, in units of residual degrees of freedom.

Inference

We test every contrast with a moderated F-test.

```
tests <- lapply(colnames(L), function(contrast) glmQLFTest(fit, contrast = L[, contrast]))
names(tests) <- colnames(L)
deCounts <- sapply(tests, function(test) summary(decideTests(test)))
rownames(deCounts) <- c("Down", "NotSig", "Up")
deCounts
```

```
        IFNg IFNg_infected Salmonella Salmonella_primed interaction
Down    2648          4105       5259              6031        2025
NotSig 13475         10742       9111              7839       15732
Up      3101          4377       4854              5354        1467
```

The table gives the number of down-regulated, not significant and up-regulated genes at a 5% FDR.

- *Salmonella* infection changes the expression of more than half of the genes, and even more genes in IFNγ-primed cells.
- IFNγ priming has a strong effect in uninfected cells and an even stronger effect in infected cells.
- For 3492 genes there is a significant interaction: IFNγ priming changes their response to the infection.

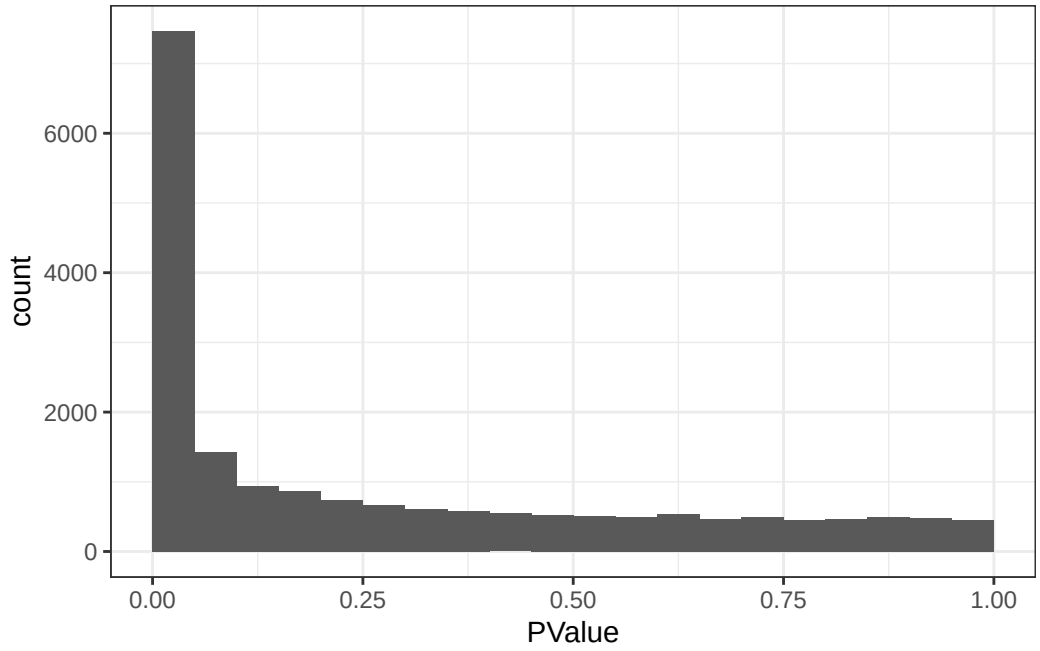
We focus on the effect of IFNγ priming in uninfected cells.

```
topTags(tests$IFNg)
```

```
Coefficient:  1*ifngyes
                     SYMBOL ENTREZID     logFC   logCPM        F       PValue
ENSG00000174944.8   P2RY14     9934  9.813128 5.197276 453.4029 5.449938e-18
ENSG00000111181.12 SLC6A12     6539  4.717016 4.299140 555.5078 1.795615e-15
ENSG00000125347.13    IRF1     3659  5.573370 9.630721 504.7893 4.256966e-15
ENSG00000234518.2     <NA>     <NA>  8.120867 3.013158 286.0034 7.847271e-15
ENSG00000137496.17  IL18BP    10068  4.068700 7.470237 469.9069 8.177514e-15
ENSG00000172399.5    MYOZ2    51778 -7.675050 4.274464 312.5865 2.175748e-14
ENSG00000204257.14 HLA-DMA     3108  4.082390 5.654779 419.7992 2.281316e-14
ENSG00000145365.10    TIFA    92610  5.202917 6.596543 384.8440 5.003047e-14
ENSG00000121380.12 BCL2L14    79370  5.912837 4.872830 296.6779 6.961950e-14
ENSG00000162645.12    GBP2     2634  6.678902 9.900451 367.0247 7.672729e-14
                          FDR
```

```
ENSG00000174944.8  1.047696e-13
ENSG00000111181.12 1.725945e-11
ENSG00000125347.13 2.727864e-11
ENSG00000234518.2  3.144090e-11
ENSG00000137496.17 3.144090e-11
ENSG00000172399.5  6.265147e-11
ENSG00000204257.14 6.265147e-11
ENSG00000145365.10 1.202232e-10
ENSG00000121380.12 1.475005e-10
ENSG00000162645.12 1.475005e-10
```

```r
ttAll <- topTags(tests$IFNg, n = Inf)$table
ttAll |>
  ggplot(aes(PValue)) +
  geom_histogram(breaks = seq(0, 1, 0.05)) +
  theme_bw()
```



The p-value histogram has a peak near zero and is uniform elsewhere: many genes respond to IFNγ.

Effect of the length offsets

We refit the model without the length offsets, i.e. with only the effective library sizes as offsets, and compare the IFNγ effect on *ASB1*.

```
dgeNoLength <- dge
dgeNoLength$offset.prior <- NULL
fitNoLength <- glmQLFit(dgeNoLength, design)
ttNoLength <- topTags(glmQLFTest(fitNoLength, contrast = L[, "IFNg"]), n = Inf)$table

data.frame(
  rbind(ttAll[asb1, c("logFC", "PValue", "FDR")],
        ttNoLength[asb1, c("logFC", "PValue", "FDR")]),
  row.names = c("with length offset", "without length offset")
)
```

```
                           logFC       PValue          FDR
with length offset    -0.1768975 2.719092e-01 0.4448665932
without length offset -1.0485918 5.649011e-06 0.0000825202
```

```
lengthNaive <- assay(sg, "length")[asb1, sg$condition == "naive"]
lengthIFNg <- assay(sg, "length")[asb1, sg$condition == "IFNg"]
mean(log2(lengthIFNg / lengthNaive))
```

```
[1] -0.8728026
```

- Without the length offset, *ASB1* is significantly down-regulated by IFNγ.
- The average log2 ratio of the transcript lengths explains most of this fold change: the log fold change with the length offset is approximately the log fold change without it minus the log2 length ratio.
- With the length offset, the number of *ASB1* molecules hardly changes: *ASB1* shows a change in transcript usage (DTU, see the technical details chapter), not in expression.

Over all genes:

```
table(
  withLength = ttAll[rownames(dge), "FDR"] < 0.05,
  withoutLength = ttNoLength[rownames(dge), "FDR"] < 0.05
)
```

```
          withoutLength
withLength FALSE  TRUE
     FALSE 12861   614
     TRUE    256  5493
```

Testing for a minimal fold change

With strong treatment effects, thousands of genes are significant, and many of them have small fold changes that are biologically less relevant. Selecting genes by an FDR cut-off and a fold change cut-off, as on a volcano plot, does not control the FDR for the combined criterion. TREAT (McCarthy and Smyth 2009) instead tests the null hypothesis $H_0 : |\beta_g| \leq \tau$ against $H_1 : |\beta_g| > \tau$ for a minimal log fold change τ, here a fold change of 2.

```
treatIFNg <- glmTreat(fit, contrast = L[, "IFNg"], lfc = 1)
summary(decideTests(treatIFNg))
```

```
       1*ifngyes
Down         517
NotSig     17888
Up           819
```

```
topTags(treatIFNg)
```

```
Coefficient:  1*ifngyes
                    SYMBOL ENTREZID     logFC unshrunk.logFC   logCPM
ENSG00000174944.8   P2RY14     9934  9.813128       9.847633 5.197276
ENSG00000111181.12 SLC6A12     6539  4.717016       4.720178 4.299140
ENSG00000125347.13    IRF1     3659  5.573370       5.573587 9.630721
ENSG00000234518.2     <NA>     <NA>  8.120867       8.255683 3.013158
ENSG00000172399.5    MYOZ2    51778 -7.675050      -7.699157 4.274464
ENSG00000137496.17  IL18BP    10068  4.068700       4.068920 7.470237
ENSG00000162645.12    GBP2     2634  6.678902       6.679412 9.900451
ENSG00000145365.10    TIFA    92610  5.202917       5.203468 6.596543
ENSG00000204257.14 HLA-DMA     3108  4.082390       4.083187 5.654779
ENSG00000121380.12 BCL2L14    79370  5.912837       5.998430 4.872830
                          PValue          FDR
ENSG00000174944.8  1.044268e-17 2.007500e-13
ENSG00000111181.12 1.087417e-14 8.687812e-11
ENSG00000125347.13 1.355776e-14 8.687812e-11
ENSG00000234518.2  2.821628e-14 1.356074e-10
ENSG00000172399.5  5.339002e-14 2.052740e-10
ENSG00000137496.17 8.256666e-14 2.645436e-10
ENSG00000162645.12 1.621779e-13 4.453868e-10
ENSG00000145365.10 1.924719e-13 4.625101e-10
ENSG00000204257.14 2.287515e-13 4.886132e-10
ENSG00000121380.12 4.576780e-13 8.798402e-10
```

Visualisation

We visualise the IFNγ effect in uninfected cells and colour every gene according to the test(s) in which it is significant at a 5% FDR: the ordinary QL test and TREAT. All genes that are significant in TREAT are also significant in the QL test.

```
deStatus <- tibble(
  logFC = tests$IFNg$table$logFC,
  logCPM = tests$IFNg$table$logCPM,
  PValue = tests$IFNg$table$PValue,
  PValueTreat = treatIFNg$table$PValue,
  FDR = p.adjust(tests$IFNg$table$PValue, "BH"),
  FDRtreat = p.adjust(treatIFNg$table$PValue, "BH")
) |>
  mutate(
    status = case_when(
      FDRtreat < 0.05 ~ "QL test and TREAT",
      FDR < 0.05 ~ "QL test only",
      TRUE ~ "not significant"
    ) |>
      factor(levels = c("not significant", "QL test only", "QL test and TREAT"))
  ) |>
  arrange(status)
table(deStatus$status)
```

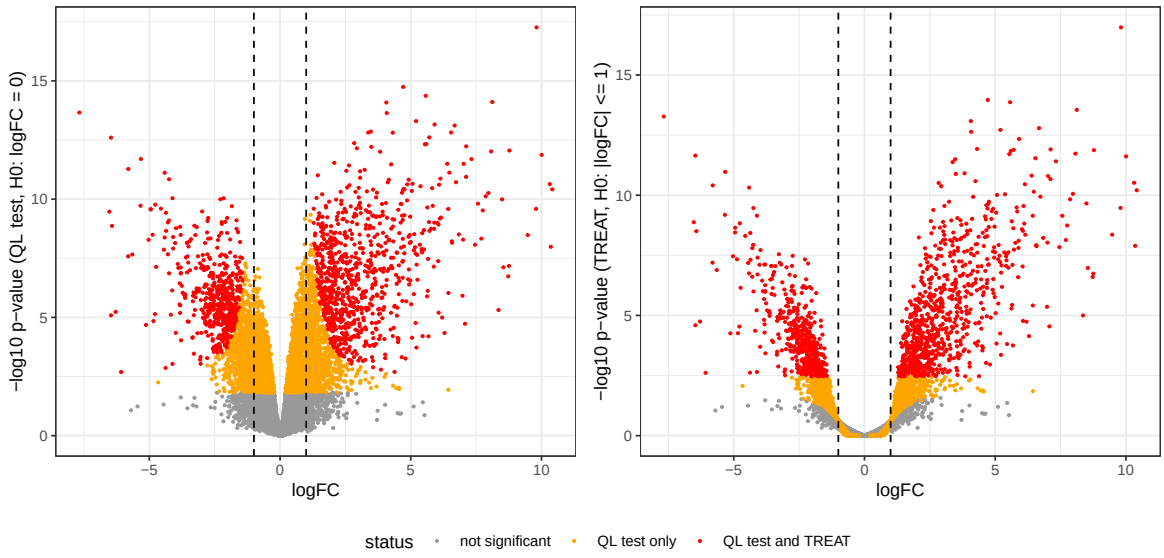
```
  not significant      QL test only QL test and TREAT
            13475              4413              1336
```

```
statusColors <- c("grey60", "orange", "red")
```

The left volcano plot shows the p-values of the QL test ($H_0 : \beta_g = 0$), the right one the p-values of TREAT ($H_0 : |\beta_g| \leq 1$).

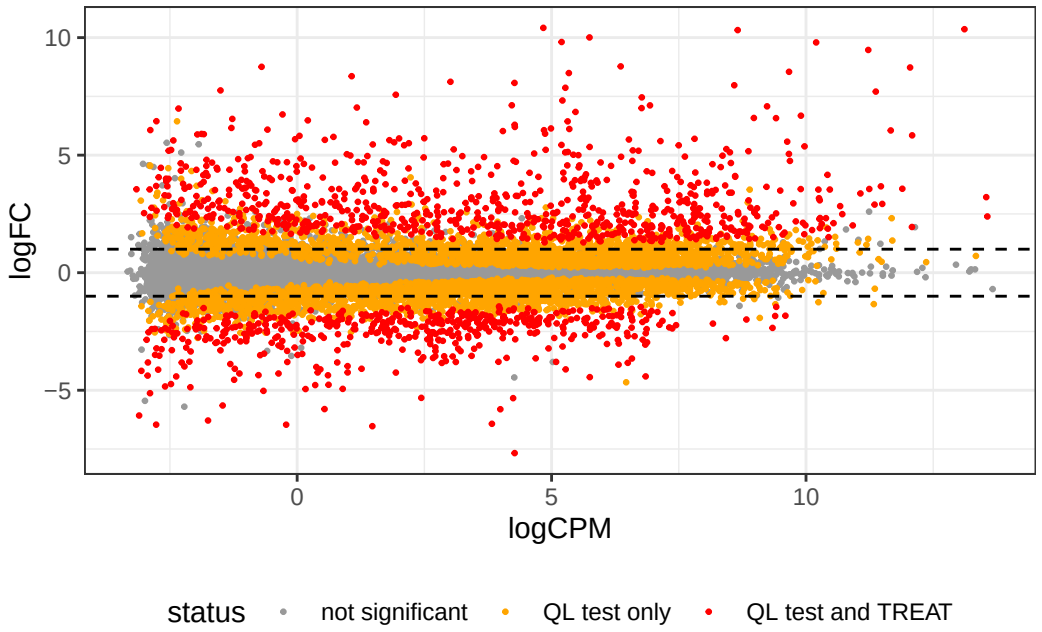
```
volcanoQL <- deStatus |>
  ggplot(aes(logFC, -log10(PValue), color = status)) +
  geom_point(size = 0.5) +
  geom_vline(xintercept = c(-1, 1), linetype = "dashed") +
  scale_color_manual(values = statusColors) +
  ylab("-log10 p-value (QL test, H0: logFC = 0)") +
  theme_bw()
```

```
volcanoTreat <- deStatus |>
  ggplot(aes(logFC, -log10(PValueTreat), color = status)) +
  geom_point(size = 0.5) +
  geom_vline(xintercept = c(-1, 1), linetype = "dashed") +
  scale_color_manual(values = statusColors) +
  ylab("-log10 p-value (TREAT, H0: |logFC| <= 1)") +
  theme_bw()
volcanoQL + volcanoTreat +
  plot_layout(guides = "collect") &
  theme(legend.position = "bottom")
```



- In the QL volcano plot the p-values decrease steadily with the absolute log fold change: the typical V shape.
- In the TREAT volcano plot genes with an estimated $|\text{logFC}| \leq 1$ get p-values close to 1: TREAT only finds evidence for fold changes that are clearly larger than 2.
- For genes with a large fold change, both p-values are similar.

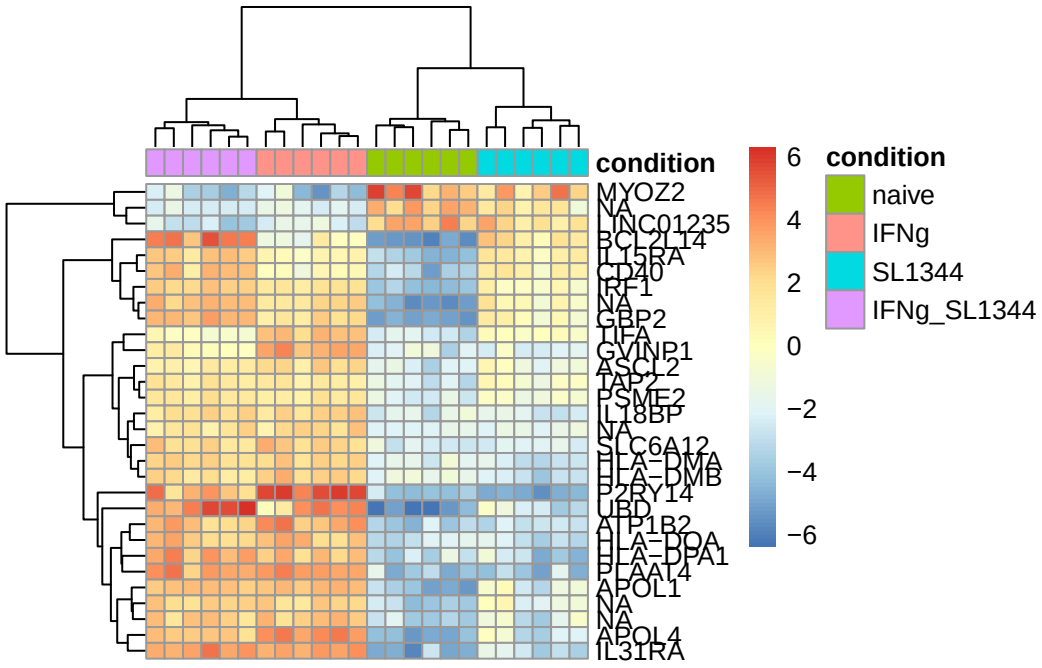
```
deStatus |>
  ggplot(aes(logCPM, logFC, color = status)) +
  geom_point(size = 0.5) +
  geom_hline(yintercept = c(-1, 1), linetype = "dashed") +
  scale_color_manual(values = statusColors) +
  theme_bw() +
  theme(legend.position = "bottom")
```



- The QL test detects every change that differs significantly from zero, including many genes with a small fold change.
- TREAT only selects genes whose fold change is significantly larger than 2 in absolute value.
- Genes with an estimated log fold change just outside the dashed lines are not selected by TREAT: their fold change is not *significantly* larger than 2. Selecting genes with FDR $<$ 0.05 and $|\text{logFC}| > 1$ would select them, without controlling the FDR for this criterion.

The heatmap shows the 30 most significant genes of the TREAT analysis in all samples.

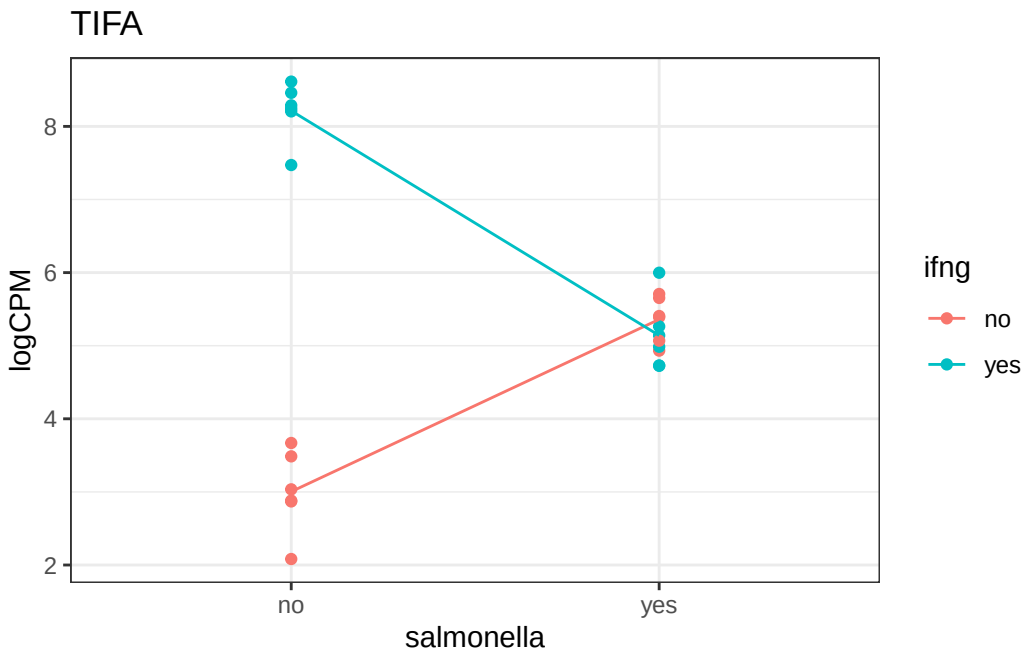
```
topGenes <- rownames(topTags(treatIFNg, n = 30))
logCPM <- cpm(dge, log = TRUE)[topGenes, ]
rownames(logCPM) <- dge$genes[topGenes, "SYMBOL"]
pheatmap::pheatmap(
  logCPM - rowMeans(logCPM),
  scale = "none",
  annotation_col = data.frame(condition = sg$condition, row.names = colnames(dge)),
  show_colnames = FALSE
)
```



- The samples cluster according to IFNγ priming, with and without infection.
- The block effect of the donors is much smaller than the treatment effects for these genes.

We plot the expression of the gene with the strongest interaction, i.e. the gene with the smallest p-value in the ordinary QL test of the interaction contrast.

```
topInteraction <- rownames(topTags(tests$interaction, n = 1))
tibble(
  logCPM = cpm(dge, log = TRUE)[topInteraction, ],
  donor = sg$donor,
  ifng = sg$ifng,
  salmonella = sg$salmonella
) |>
  ggplot(aes(salmonella, logCPM, color = ifng, group = ifng)) +
  geom_point() +
  stat_summary(fun = mean, geom = "line") +
  ggtitle(dge$genes[topInteraction, "SYMBOL"]) +
  theme_bw()
```



- IFNγ priming and *Salmonella* infection both induce *TIFA* as compared to uninfected, non-primed macrophages.
  - In infected cells, however, the expression is about the same with and without priming: the IFNγ effect in infected cells, `ifngyes` + `ifngyes:salmonellayes`, is much smaller than in uninfected cells.
  - Equivalently, the effect of the infection is much smaller in primed than in non-primed macrophages: it even flips sign.
  - The interaction measures exactly this difference between the effects:

$$\begin{aligned}\beta_{g,\text{IFN}\gamma\text{:Sal}} &= \log_2 \text{FC}^{\text{primed}}_{\text{infected vs uninfected}} - \log_2 \text{FC}^{\text{non-primed}}_{\text{infected vs uninfected}} \\ &= \log_2 \text{FC}^{\text{infected}}_{\text{primed vs non-primed}} - \log_2 \text{FC}^{\text{uninfected}}_{\text{primed vs non-primed}}.\end{aligned}$$

    The effect of the infection depends on the priming status or, equivalently, the effect of the priming depends on the infection status.

Gene set analysis

Which biological processes are affected by IFNγ?

The simplest approach is **over-representation analysis** (ORA):

1. Classify each gene as DE or not (e.g. FDR < 0.05).
2. For each gene set, e.g. a Gene Ontology (GO) term, build a 2x2 table:

	in gene set	not in gene set
DE	a	b
not DE	c	d

3. Test whether the gene set contains more DE genes than expected by chance with a one-sided Fisher exact test (hypergeometric test).
4. Correct for multiple testing over all gene sets.

`goana` performs ORA for all GO terms. With thousands of DE genes, ORA mainly picks up very general GO terms. We therefore use the genes that are significant in the TREAT analysis, i.e. with a fold change that is significantly larger than 2, separately for up- and down-regulated genes, and only consider GO terms with 10 to 500 genes. The universe consists of the genes that passed the filtering: genes that were not measured cannot be DE.

```
ttTreat <- topTags(treatIFNg, n = Inf)$table |>
  filter(!is.na(ENTREZID))
deGenes <- list(
  Up = ttTreat |> filter(FDR < 0.05, logFC > 0) |> pull(ENTREZID),
  Down = ttTreat |> filter(FDR < 0.05, logFC < 0) |> pull(ENTREZID)
)
lengths(deGenes)
```

```
  Up Down
 699  450
```

```
go <- goana(deGenes, universe = ttTreat$ENTREZID, species = "Hs")
go |>
  filter(Ont == "BP", N >= 10, N <= 500) |>
  topGO(sort = "Up", number = 10)
```

```
GO:0002250
GO:0002460 adaptive immune response based on somatic recombination of immune receptors built
GO:0002252
GO:0002253
GO:0042110
GO:0002831                                                                                  re
GO:0002694
GO:0050865
GO:0050863
GO:0034341
```

```
           Ont   N  Up Down         P.Up    P.Down
GO:0002250  BP 377 101   11 4.696410e-53 0.4595854
GO:0002460  BP 258  68    4 3.159146e-35 0.9262609
GO:0002252  BP 449  88   14 1.328718e-34 0.3490455
GO:0002253  BP 480  84   13 2.481724e-29 0.5575660
GO:0042110  BP 426  78   14 1.292884e-28 0.2810402
GO:0002831  BP 489  83   13 5.017719e-28 0.5847490
GO:0002694  BP 447  79   10 6.667978e-28 0.7862718
GO:0050865  BP 493  82   12 4.537449e-27 0.7035397
GO:0050863  BP 297  63    9 6.588703e-27 0.4274585
GO:0034341  BP  99  38    1 8.737940e-27 0.9366909
```

The up-regulated genes are enriched for the response to type II interferon (IFNγ is the only type II interferon), the adaptive immune response and T cell activation, e.g. the HLA class II genes that present antigens to T cells.

Limitations of ORA:

- The result depends on the arbitrary FDR cut-off and ignores the effect sizes.
- Genes in a set are correlated, while the test assumes independence: the p-values are too small.
- Long and highly expressed genes are more easily detected as DE.

Methods that use the statistics of all genes avoid the cut-off, e.g. `camera` (competitive test accounting for correlation between genes) and `fry` (self-contained test), which are available in limma and edgeR. You will use them in the tutorial.

Summary

- RNA-seq measures expression by sequencing fragments of cDNA. Reads are mapped to a reference and summarised in a count table.
- Counts of technical replicates are Poisson; biological replicates are overdispersed, with a quadratic mean-variance relation.
- GLMs model the counts directly. Differences in sequencing depth, RNA composition and average transcript length are accounted for with offsets.
- edgeR combines a global NB mean-variance shape with gene-specific quasi-likelihood dispersions, shrunken with empirical Bayes: the same strategy as msqrob2.
- Factorial designs are analysed with contrasts of the model parameters; TREAT tests for a minimal fold change.
- Gene set analysis helps to interpret the list of DE genes.

References

Alasoo, Kaur, Julia Rodrigues, Subhankar Mukhopadhyay, et al. 2018. "Shared Genetic Effects on Chromatin and Gene Expression Indicate a Role for Enhancer Priming in Immune Response." *Nature Genetics* 50 (3): 424–31.

Bray, Nicolas L, Harold Pimentel, Páll Melsted, and Lior Pachter. 2016. "Near-Optimal Probabilistic RNA-seq Quantification." *Nature Biotechnology* 34 (5): 525–27.

Chen, Yunshun, Lizhong Chen, Aaron T L Lun, Pedro L Baldoni, and Gordon K Smyth. 2025. "edgeR V4: Powerful Differential Analysis of Sequencing Data with Expanded Functionality and Improved Support for Small Counts and Larger Datasets." *Nucleic Acids Research* 53 (2): gkaf018.

Haglund, Felix, Ran Ma, Mikael Huss, et al. 2012. "Evidence of a Functional Estrogen Receptor in Parathyroid Adenomas." *The Journal of Clinical Endocrinology & Metabolism* 97 (12): 4631–39.

Law, Charity W, Yunshun Chen, Wei Shi, and Gordon K Smyth. 2014. "Voom: Precision Weights Unlock Linear Model Analysis Tools for RNA-seq Read Counts." *Genome Biology* 15 (2): R29.

Love, Michael I, Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-seq Data with DESeq2." *Genome Biology* 15 (12): 550.

Love, Michael I, Charlotte Soneson, Peter F Hickey, et al. 2020. "Tximeta: Reference Sequence Checksums for Provenance Identification in RNA-seq." *PLoS Computational Biology* 16 (2): e1007664.

Lund, Steven P, Dan Nettleton, Davis J McCarthy, and Gordon K Smyth. 2012. "Detecting Differential Expression in RNA-sequence Data Using Quasi-Likelihood with Shrunken Dispersion Estimates." *Statistical Applications in Genetics and Molecular Biology* 11 (5).

Marioni, John C, Christopher E Mason, Shrikant M Mane, Matthew Stephens, and Yoav Gilad. 2008. "RNA-seq: An Assessment of Technical Reproducibility and Comparison with Gene Expression Arrays." *Genome Research* 18 (9): 1509–17.

McCarthy, Davis J, and Gordon K Smyth. 2009. "Testing Significance Relative to a Fold-Change Threshold Is a TREAT." *Bioinformatics* 25 (6): 765–71. https://doi.org/10.1093/bioinformatics/btp053.

Patro, Rob, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. 2017. "Salmon

Provides Fast and Bias-Aware Quantification of Transcript Expression." *Nature Methods* 14 (4): 417–19.

Robinson, Mark D, and Alicia Oshlack. 2010. "A Scaling Normalization Method for Differential Expression Analysis of RNA-seq Data." *Genome Biology* 11 (3): R25.

Soneson, Charlotte, Michael I Love, and Mark D Robinson. 2015. "Differential Analyses for RNA-seq: Transcript-Level Estimates Improve Gene-Level Inferences." *F1000Research* 4: 1521.

Van den Berge, Koen, Katharina M Hembach, Charlotte Soneson, et al. 2019. "RNA Sequencing Data: Hitchhiker's Guide to Expression Analysis." *Annual Review of Biomedical Data Science* 2: 139–73.