Bulk RNA-seq: technical details and transcript usage

Lieven Clement Koen Van den Berge

```
library(tidyverse)
library(SummarizedExperiment)
library(edgeR)
library(BiocFileCache)
library(tximeta)
library(org.Hs.eg.db)
library(DESeq2)
library(stageR)
library(patchwork)
library(macrophage)
```

In the introduction to bulk RNA-seq we introduced count models and a complete edgeR workflow. In this chapter we look under the hood:

1. How generalised linear models (GLMs) are fitted and how inference works.
2. How edgeR estimates the variability with negative binomial (NB) and quasi-likelihood (QL) models, and how this compares to limma-voom and DESeq2.
3. The calibration of the tests under the null hypothesis.
4. Normalisation and independent filtering.
5. Analyses at the transcript level: differential transcript expression and differential transcript usage.

Generalised linear models

Simulated Poisson data

We simulate Poisson counts for 100 observations with two covariates, $\beta_0 = 2$, $\beta_1 = 0.8$ and $\beta_2 = 1.2$.

```r
set.seed(300)
X <- cbind(1, rnorm(100), rnorm(100))
betasTrue <- c(2, 0.8, 1.2)
etaTrue <- X %*% betasTrue
y <- rpois(100, exp(etaTrue))
```

Exponential family

GLMs assume that the response follows a distribution of the exponential family:

$$f(y_i|\theta_i, \phi) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)\right\}$$

with

- θ_i: the canonical parameter,
- ϕ: the dispersion parameter,
- $a(.)$, $b(.)$, $c(.)$: functions that depend on the distribution.

The first two moments follow from $b(.)$:

- $E[y_i] = \mu_i = b'(\theta_i)$,
- $\text{Var}[y_i] = a(\phi)\, b''(\theta_i)$.

The variance is thus a function of the mean.

Poisson.

$$f(y_i|\mu_i) = \frac{\mu_i^{y_i} e^{-\mu_i}}{y_i!} = \exp\left\{y_i \log(\mu_i) - \mu_i - \log(y_i!)\right\}$$

so $\theta_i = \log \mu_i$, $b(\theta_i) = \exp(\theta_i)$, $a(\phi) = 1$ and $c(y_i, \phi) = -\log(y_i!)$. Hence $E[y_i] = b'(\theta_i) = \exp(\theta_i) = \mu_i$ and $\text{Var}[y_i] = b''(\theta_i) = \mu_i$.

Components of a GLM

$$\begin{cases} y_i|\mathbf{x}_i & \sim & f(y_i|\theta_i, \phi) \\ E\left[y_i|\mathbf{x}_i\right] & = & \mu_i \\ g(\mu_i) & = & \eta_i \\ \eta_i & = & \mathbf{x}_i^T\beta \end{cases}$$

with $g(.)$ the link function, e.g.

- identity link $g(\mu) = \mu$ for the normal distribution,

- log link $g(\mu) = \log \mu$ for the Poisson distribution,
- logit link $g(\mu) = \log \frac{\mu}{1-\mu}$ for the Bernoulli distribution.

These are the **canonical** links: with a canonical link $\theta_i = \eta_i$.

Likelihood

- The likelihood is the probability (density) to observe the sample, as a function of the model parameters:
$$L(\beta|\mathbf{y}) = \prod_{i=1}^{n} f(y_i|\theta_i, \phi)$$
- We work with the log-likelihood
$$l(\beta|\mathbf{y}) = \sum_{i=1}^{n} \log f(y_i|\theta_i, \phi).$$
- For the Poisson GLM with log link:
$$l(\beta|\mathbf{y}) = \sum_{i=1}^{n} \{y_i \eta_i - e^{\eta_i} - \log y_i!\}, \quad \eta_i = \mathbf{x}_i^T \beta.$$

```
c(dpois = sum(dpois(y, exp(etaTrue), log = TRUE)),
  formula = sum(y * etaTrue - exp(etaTrue) - lfactorial(y)))
```

```
   dpois  formula
-260.485 -260.485
```

Maximum likelihood estimation

We choose the parameters that maximise the log-likelihood, i.e. we solve the **score equations**

$$S(\beta) = \frac{\partial l(\beta|\mathbf{y})}{\partial \beta} = \mathbf{0}.$$

With the chain rule we obtain

$$S(\beta) = \sum_{i=1}^{n} \frac{y_i - \mu_i}{\mathrm{Var}[y_i]} \frac{\partial \mu_i}{\partial \eta_i} \mathbf{x}_i = \mathbf{X}^T \mathbf{A} (\mathbf{y} - \mu),$$

with $\mathbf{A}$ a diagonal matrix with elements $a_{ii} = \left(\mathrm{Var}[y_i] \frac{\partial \eta_i}{\partial \mu_i}\right)^{-1}$.

- For the Poisson GLM with log link: $\text{Var}[y_i] = \mu_i$ and $\partial\mu_i/\partial\eta_i = \mu_i$, so $\mathbf{A} = \mathbf{I}$ and $S(\beta) = \mathbf{X}^T(\mathbf{y} - \mu)$.
- The score equations are non-linear in β because $\mu = \exp(\mathbf{X}\beta)$.
- Note that only the **mean and the variance** of y_i enter the score equations. This is the basis of quasi-likelihood.

Newton-Raphson and Fisher scoring

We find the root of the score function iteratively with the Newton-Raphson algorithm: walk along the tangent of the score function to the point where it crosses zero.

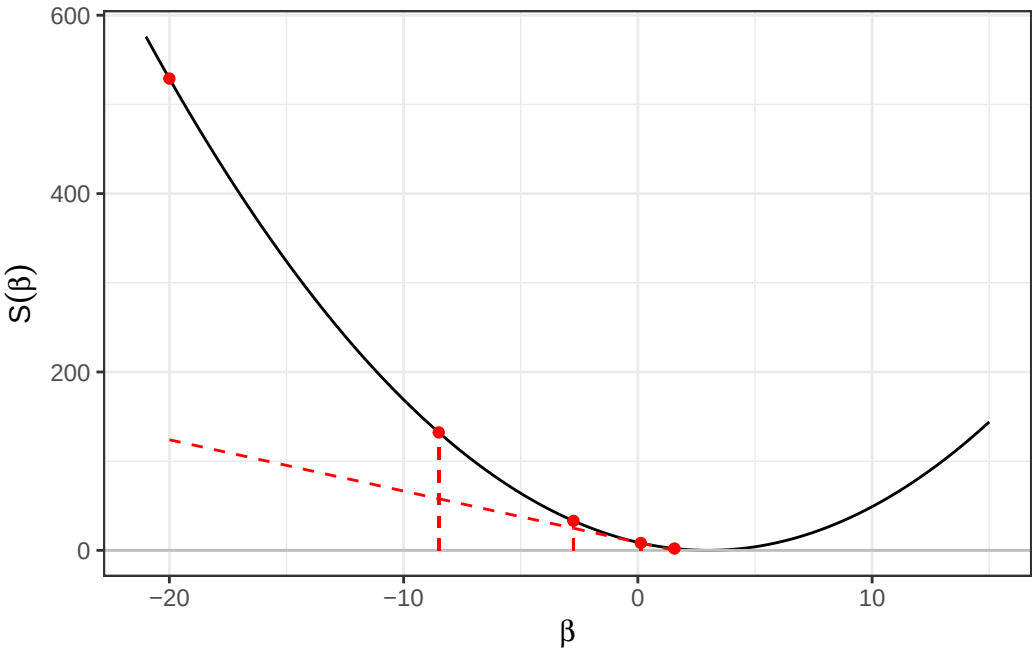


Figure 1: Newton-Raphson: starting from an initial guess, the algorithm walks along the tangent of the score function to the point where the tangent crosses zero, and repeats this until the root is found.

The update step is

$$\beta^{k+1} = \beta^k - \left(\frac{\partial S(\beta)}{\partial \beta}\bigg|_{\beta^k} \right)^{-1} S(\beta^k) = \beta^k + \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1} S(\beta^k),$$

with $\mathbf{W}$ a diagonal matrix with weights $w_{ii} = \text{Var}[y_i]^{-1} \left(\frac{\partial \mu_i}{\partial \eta_i}\right)^2$.

- $\mathbf{X}^T\mathbf{W}\mathbf{X}$ is the Fisher information matrix.
- With a canonical link the observed and expected Fisher information are equal, so Newton-Raphson and Fisher scoring coincide.

Iteratively reweighted least squares

Rewriting the update step gives

$$\beta^{k+1} = \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z}, \quad \mathbf{z} = \mathbf{X}\beta^k + \frac{\partial\eta}{\partial\mu}\left(\mathbf{y} - \mu\right).$$

Each iteration is a **weighted least squares** fit of the working response $\mathbf{z}$ on $\mathbf{X}$, as in a linear model. The weights account for the mean-variance relation: observations with a smaller variance get more weight. This is iteratively reweighted least squares (IRLS).

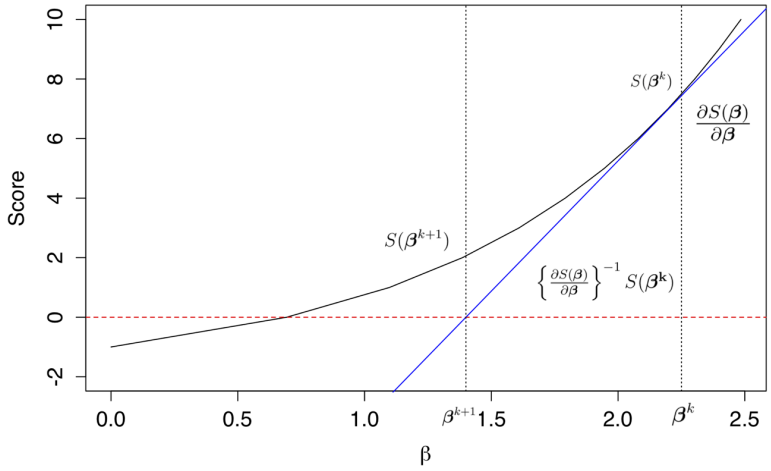


Figure 2: The IRLS algorithm.

For the Poisson GLM with log link: $z_i = \eta_i + e^{-\eta_i}(y_i - e^{\eta_i})$ and $w_{ii} = e^{\eta_i}$.

IRLS in R

We start from the fit of a linear model on the log counts and iterate until the deviance no longer changes (the deviance is defined below).

```r
poisDeviance <- function(y, eta) {
  2 * sum(ifelse(y == 0, 0, y * log(y)) - y * eta - (y - exp(eta)))
}

betas <- lm(log(y + 0.5) ~ -1 + X)$coef
eta <- X %*% betas
dev <- poisDeviance(y, eta)
devOld <- Inf
iteration <- 0
while ((devOld - dev) / dev > 1e-8) {
  iteration <- iteration + 1
  z <- eta + exp(-eta) * (y - exp(eta))  # working response
  w <- c(exp(eta))                        # weights
  wlsFit <- lm(z ~ -1 + X, weights = w)   # weighted least squares
  betas <- coef(wlsFit)
  eta <- X %*% betas
  devOld <- dev
  dev <- poisDeviance(y, eta)
  cat("iteration", iteration, "deviance", dev, "\n")
}
```

```
iteration 1 deviance 114.3748
iteration 2 deviance 114.3374
iteration 3 deviance 114.3374
```

The variance-covariance matrix of the estimators is the inverse of the Fisher information: $\Sigma_{\hat{\beta}} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}$.

```r
w <- c(exp(eta))
seBetas <- sqrt(diag(solve(t(X) %*% diag(w) %*% X)))
glmFitPois <- glm(y ~ -1 + X, family = poisson)
cbind(
  irls = c(betas, seBetas, deviance = dev),
  glm = c(coef(glmFitPois), summary(glmFitPois)$coef[, 2], deviance(glmFitPois))
)
```

```
                irls          glm
X1         1.96805691   1.96805691
X2         0.76136641   0.76136641
X3         1.23330031   1.23330031
           0.03814384   0.03814382
```

```
           0.01891208   0.01891208
           0.02556666   0.02556665
deviance 114.33739500 114.33739500
```

Our IRLS implementation gives the same estimates, standard errors and deviance as `glm`.

Inference

Wald test

The maximum likelihood estimator is asymptotically normal:

$$\hat{\beta} \sim N\left[\beta, \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\right].$$

For a contrast $\mathbf{L}^T\beta$:

$$W = \frac{\mathbf{L}^T\hat{\beta}}{\mathrm{SE}_{\mathbf{L}^T\hat{\beta}}} \approx N(0,1) \text{ under } H_0 : \mathbf{L}^T\beta = 0.$$

For c contrasts simultaneously, $\mathbf{L}$ is a $p \times c$ matrix and

$$W = \hat{\beta}^T\mathbf{L}\left(\mathbf{L}^T\Sigma_{\hat{\beta}}\mathbf{L}\right)^{-1}\mathbf{L}^T\hat{\beta} \approx \chi^2_c \text{ under } H_0 : \mathbf{L}^T\beta = \mathbf{0}.$$

Deviance and likelihood ratio test

- The **deviance** compares the fit of the model with the saturated model, which fits every observation perfectly ($\hat{\mu}_i = y_i$):

  $$D = 2\left[l(\mathbf{y}|\mathbf{y}) - l(\hat{\mu}|\mathbf{y})\right].$$

  It plays the role of the residual sum of squares in a linear model.
- The **likelihood ratio test** compares a full model with a nested reduced model, e.g. without the treatment effect:

  $$\mathrm{LRT} = 2\left[l(\hat{\beta}_{\text{full}}) - l(\hat{\beta}_0)\right] = D_0 - D_{\text{full}} \approx \chi^2_c \text{ under } H_0,$$

  with c the number of parameters that are dropped.

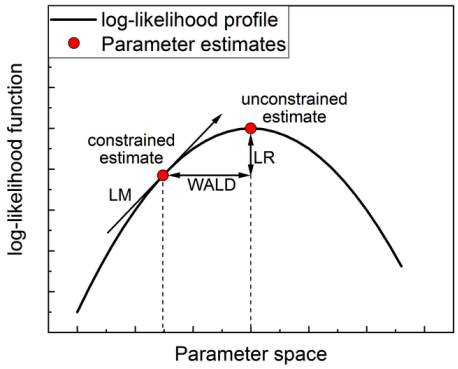


Figure 3: The Wald test looks at the distance between the estimate and the null value on the parameter scale, the likelihood ratio test at the difference in log-likelihood.

These results are asymptotic. With the typical 3 to 5 biological replicates per group in bulk RNA-seq, the large sample theory does not hold. Moreover, the dispersion of the NB distribution has to be estimated, which adds uncertainty that the χ^2 tests ignore. Quasi-likelihood F-tests address both issues.

Differential expression with edgeR

The negative binomial GLM

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

- For a known dispersion ϕ_g, the NB distribution is a member of the exponential family and the model is fitted with IRLS.
- The original edgeR pipeline (McCarthy et al. 2012) estimates the dispersions with an adjusted profile likelihood (Cox-Reid), which corrects for the estimation of the mean parameters, similar to REML in mixed models.
- Gene-wise dispersions are very unstable with few replicates. They are shrunken towards a trend in the average expression with empirical Bayes, and the genes are then tested with likelihood ratio tests.
- The uncertainty in the estimated dispersions is ignored by these tests.

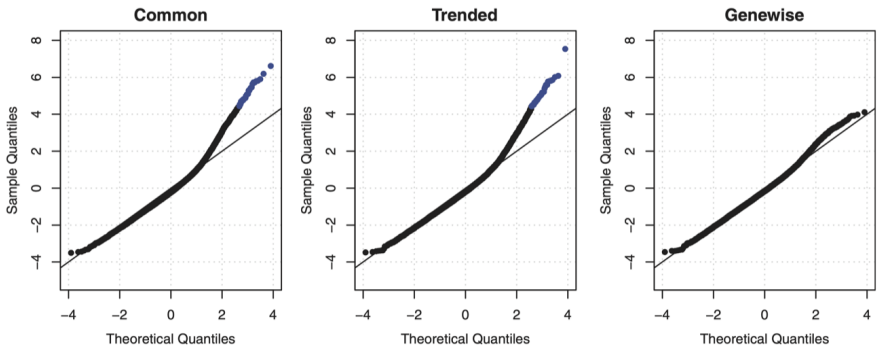


Figure 4: QQ-plots of goodness of fit statistics with a common, trended or empirical Bayes (tagwise) dispersion (from McCarthy et al., 2012).

Quasi-likelihood

In quasi-likelihood we no longer specify the full distribution, but only the mean and the variance:

$$\begin{cases} E[Y_i|\mathbf{x}_i] &= \mu_i \\ g(\mu_i) &= \mathbf{x}_i^T\beta \\ \text{Var}[Y_i] &= \sigma^2 V(\mu_i) \end{cases}$$

- $V(\mu_i)$ is the variance function of a known distribution, e.g. $V(\mu) = \mu$ (quasi-Poisson) or $V(\mu) = \mu + \phi\mu^2$ (quasi-NB).
- σ^2 is the **quasi-likelihood dispersion**, which allows for more ($\sigma^2 > 1$) or less ($\sigma^2 < 1$) variability than the original distribution.

Estimation

The estimating equations become

$$U(\beta) = \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{A}\,(\mathbf{y} - \mu) = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{X}^T\mathbf{A}\,(\mathbf{y} - \mu) = \mathbf{0}.$$

- The estimators of the mean parameters are identical to those of the original GLM: we can use the same IRLS algorithm.
- The variance-covariance matrix is inflated: $\Sigma_{\hat{\beta}} = \sigma^2(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}$.

- σ^2 is estimated from the residuals, e.g. based on the deviance:

$$\hat{\sigma}^2 = \frac{D}{n-p},$$

  the GLM counterpart of the residual variance $\hat{\sigma}^2 = \text{SSE}/(n-p)$ in a linear model.

Inference

- t-test: $t = \frac{\mathbf{L}^T\hat{\beta}}{\text{SE}^{QL}} \approx t_{n-p}$ under H_0, with $\text{SE}^{QL} = \hat{\sigma}\,\text{SE}^{ML}$.
- F-test: $F = \frac{\text{LRT}/c}{\hat{\sigma}^2} \approx F_{c,n-p}$ under H_0.

The t and F distributions account for the uncertainty in $\hat{\sigma}^2$, as in linear models.

Example: quasi-Poisson

```
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
glmPois <- glm(counts ~ outcome + treatment, family = poisson())
glmQPois <- glm(counts ~ outcome + treatment, family = quasipoisson())
```

The quasi-Poisson fit gives the same estimates, but its standard errors are inflated with $\hat{\sigma}$. `glm` estimates σ^2 with the Pearson residuals, $\hat{\sigma}^2 = \sum_i \frac{(y_i-\hat{\mu}_i)^2}{V(\hat{\mu}_i)}/(n-p)$.

```
sigma2 <- sum(residuals(glmPois, type = "pearson")^2) / df.residual(glmPois)
sigma2
```

```
[1] 1.2933
```

```
cbind(
  estimate = coef(glmPois),
  sePois = summary(glmPois)$coef[, 2],
  seQPois = summary(glmQPois)$coef[, 2],
  sePoisTimesSigma = summary(glmPois)$coef[, 2] * sqrt(sigma2)
)
```

```
                estimate    sePois   seQPois sePoisTimesSigma
(Intercept)  3.044522e+00 0.1708987 0.1943517        0.1943517
outcome2    -4.542553e-01 0.2021708 0.2299154        0.2299154
outcome3    -2.929871e-01 0.1927423 0.2191931        0.2191931
treatment2   1.397701e-16 0.2000000 0.2274467        0.2274467
treatment3  -2.415657e-16 0.2000000 0.2274467        0.2274467
```

The F-test for the outcome effect equals the LRT divided by its degrees of freedom and $\hat{\sigma}^2$:

```
glmQPois0 <- glm(counts ~ treatment, family = quasipoisson())
anova(glmQPois0, glmQPois, test = "F")
```

```
Analysis of Deviance Table

Model 1: counts ~ treatment
Model 2: counts ~ outcome + treatment
  Resid. Df Resid. Dev Df Deviance      F Pr(>F)
1         6    10.5814
2         4     5.1291  2   5.4523 2.1079  0.237
```

```
(deviance(glmQPois0) - deviance(glmQPois)) / 2 / sigma2
```

```
[1] 2.107903
```

The edgeR quasi-likelihood pipeline

edgeR (Lund et al. 2012; Chen et al. 2025) uses a quasi-NB model with a gene-specific QL dispersion:

$$\mathrm{Var}[Y_{gi}] = \sigma_g^2 \left(\mu_{gi} + \phi \mu_{gi}^2\right).$$

We illustrate it on the macrophage data of the introduction: 6 donors, with IFNγ priming and *Salmonella* infection in a 2×2 factorial design (Alasoo et al. 2018).

```
macDir <- system.file("extdata", package = "macrophage")
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"))
  )
st <- tximeta(coldata, dropInfReps = TRUE)
```

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

dge <- DGEListFromTximeta(sg)
dge$genes <- dge$genes[, c("SYMBOL", "Max2MinLength")]
design <- model.matrix(~ donor + ifng * salmonella, data = colData(sg))
keep <- filterByExpr(dge, design)
dge <- dge[keep, , keep.lib.sizes = FALSE]
dge <- normLibSizes(dge)
```

As in the introduction, `DGEListFromTximeta` stores the centred log average transcript lengths in `dge$offset.prior`. `getOffset` returns the offsets that edgeR uses in all models: the log effective library sizes plus these length offsets.

```
getOffset(dge)[1:3, 1:4]
```

```
                  SAMEA103885102 SAMEA103885347 SAMEA103885043 SAMEA103885392
ENSG00000000003.14       17.83113       17.81073       17.51715       17.36446
ENSG00000000419.12       17.83579       17.80954       17.53832       17.37225
ENSG00000000457.13       17.98175       17.84456       17.58191       17.30384
```

Step 1: a global NB dispersion

`glmQLFit` no longer needs gene-wise NB dispersions. It estimates a single NB dispersion ϕ from the most highly expressed genes, for which the Poisson noise is negligible.

```
fit <- glmQLFit(dge, design)
fit$dispersion
```

```
[1] 0.08474179
```

```
fit$top.proportion
```

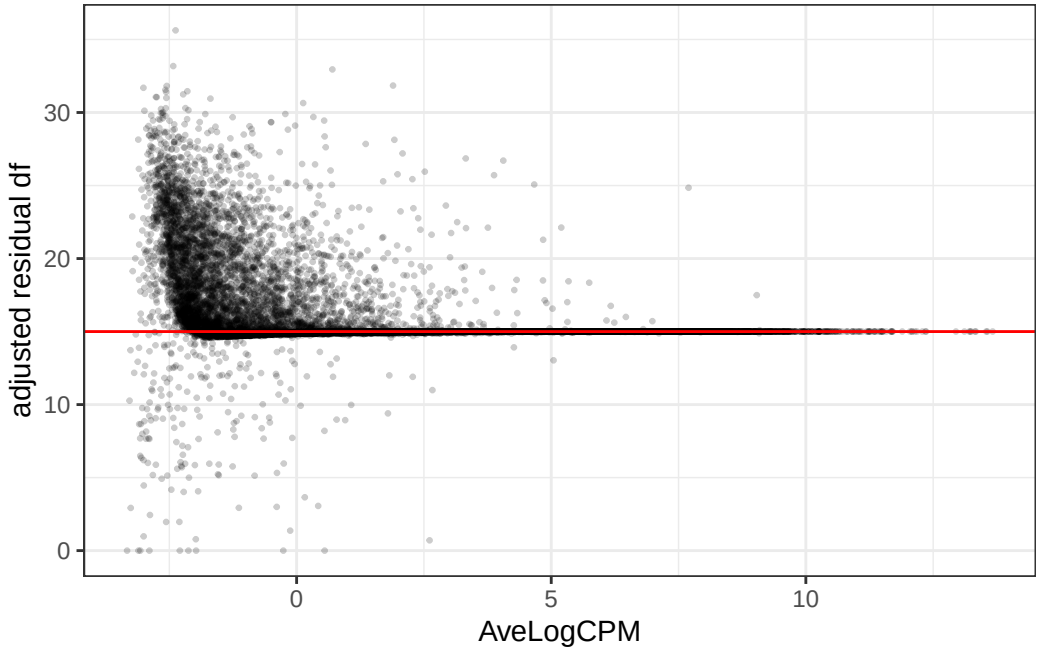
```
[1] 0.08323239
```

$\phi = 0.085$ corresponds to a biological coefficient of variation of 29%. All gene-specific variability is captured by the QL dispersions σ_g^2.

Step 2: QL dispersions with bias-adjusted degrees of freedom

The QL dispersion is estimated as $\hat{\sigma}_g^2 = D_g/\mathrm{df}_g$. For low counts the deviance does not follow a scaled χ^2_{n-p} distribution: its expected value is smaller than $n-p$, so $\hat{\sigma}_g^2$ would be biased downwards and low-count genes would look too stable. edgeR 4 therefore adjusts the deviance of each observation and uses **effective residual degrees of freedom**, which are smaller for low counts.

```
tibble(AveLogCPM = fit$AveLogCPM, df = fit$df.residual.adj) |>
  ggplot(aes(AveLogCPM, df)) +
  geom_point(alpha = 0.2, size = 0.5) +
  geom_hline(yintercept = ncol(dge) - ncol(design), col = "red") +
  ylab("adjusted residual df") +
  theme_bw()
```



The red line is the nominal $n - p = 15$ residual degrees of freedom.

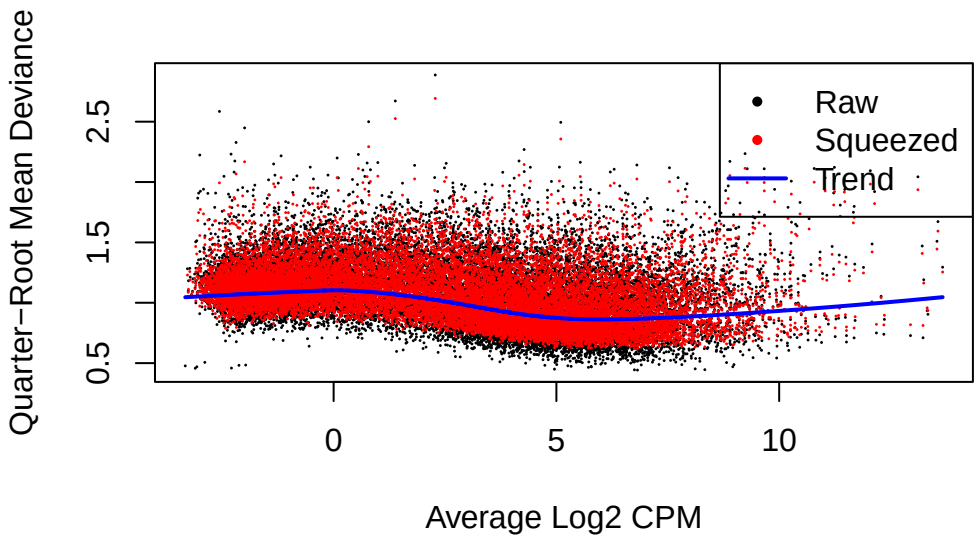
Step 3: empirical Bayes

The QL dispersions are shrunken towards a trend in the average expression with `squeezeVar`, exactly as the residual variances in limma and msqrob2:

$$\tilde{\sigma}_g^2 = \frac{d_0 s_0^2 + d_g \hat{\sigma}_g^2}{d_0 + d_g},$$

with s_0^2 the trend, d_0 the prior degrees of freedom and d_g the (adjusted) residual degrees of freedom of gene g.

```
plotQLDisp(fit)
```




```
fit$df.prior
```

```
[1] 3.903063
```

Step 4: moderated F-tests

$$F_g = \frac{\mathrm{LRT}_g / c}{\tilde{\sigma}_g^2} \approx F_{c, d_0 + d_g} \text{ under } H_0.$$

For a single parameter ($c = 1$), e.g. the IFNγ effect in uninfected cells, the F-test is equivalent to a moderated t-test:

```
qlf <- glmQLFTest(fit, coef = "ifngyes")
topTags(qlf, n = 5)
```

```
Coefficient:  ifngyes
                    SYMBOL Max2MinLength    logFC   logCPM        F
ENSG00000174944.8   P2RY14      1.473634 9.813128 5.197276 453.4029
ENSG00000111181.12 SLC6A12      2.430672 4.717016 4.299140 555.5078
ENSG00000125347.13    IRF1      1.110418 5.573370 9.630721 504.7893
ENSG00000234518.2     <NA>      1.120846 8.120867 3.013158 286.0034
ENSG00000137496.17  IL18BP      1.367702 4.068700 7.470237 469.9069
                         PValue          FDR
ENSG00000174944.8  5.449938e-18 1.047696e-13
ENSG00000111181.12 1.795615e-15 1.725945e-11
ENSG00000125347.13 4.256966e-15 2.727864e-11
ENSG00000234518.2  7.847271e-15 3.144090e-11
ENSG00000137496.17 8.177514e-15 3.144090e-11
```

```
summary(decideTests(qlf))
```

```
       ifngyes
Down      2648
NotSig   13475
Up        3101
```

With several parameters we test whether any of them differs from zero, e.g. whether a gene responds to any of the treatments ($c = 3$):

```
qlfAny <- glmQLFTest(fit, coef = c("ifngyes", "salmonellayes", "ifngyes:salmonellayes"))
topTags(qlfAny, n = 5)
```

```
Coefficient:  ifngyes salmonellayes ifngyes:salmonellayes
                     SYMBOL Max2MinLength logFC.ifngyes logFC.salmonellayes
ENSG00000255521.1   CD44-DT      1.818456     -1.320271           6.8065895
ENSG00000171989.5   LDHAL6B      1.037796      2.259354           0.7179383
ENSG00000182393.2     IFNL1      1.127983      1.191166           7.7333750
ENSG00000279965.1      <NA>      1.040066      2.733704           2.0793137
ENSG00000185433.8 LINC00158      2.222640      3.542199           7.5644149
                  logFC.ifngyes.salmonellayes    logCPM        F       PValue
ENSG00000255521.1                    4.601783 2.9315851 627.3802 1.933860e-25
ENSG00000171989.5                    5.626021 0.5407264 329.7240 7.550937e-25
ENSG00000182393.2                    5.840472 7.6971616 480.5455 1.181854e-24
ENSG00000279965.1                    6.388032 2.5174241 340.7399 3.563758e-23
ENSG00000185433.8                   -1.280528 2.3879599 374.8519 6.616562e-22
                           FDR
ENSG00000255521.1 3.717653e-21
ENSG00000171989.5 7.257961e-21
ENSG00000182393.2 7.573323e-21
ENSG00000279965.1 1.712742e-19
ENSG00000185433.8 2.543936e-18
```

```
summary(decideTests(qlfAny))
```

```
       ifngyes:salmonellayes-salmonellayes-ifngyes
NotSig                                        3971
Sig                                          15253
```

Effect of the length correction

We refit the model without the length offsets, i.e. on the summed counts with only the effective library sizes as offsets.

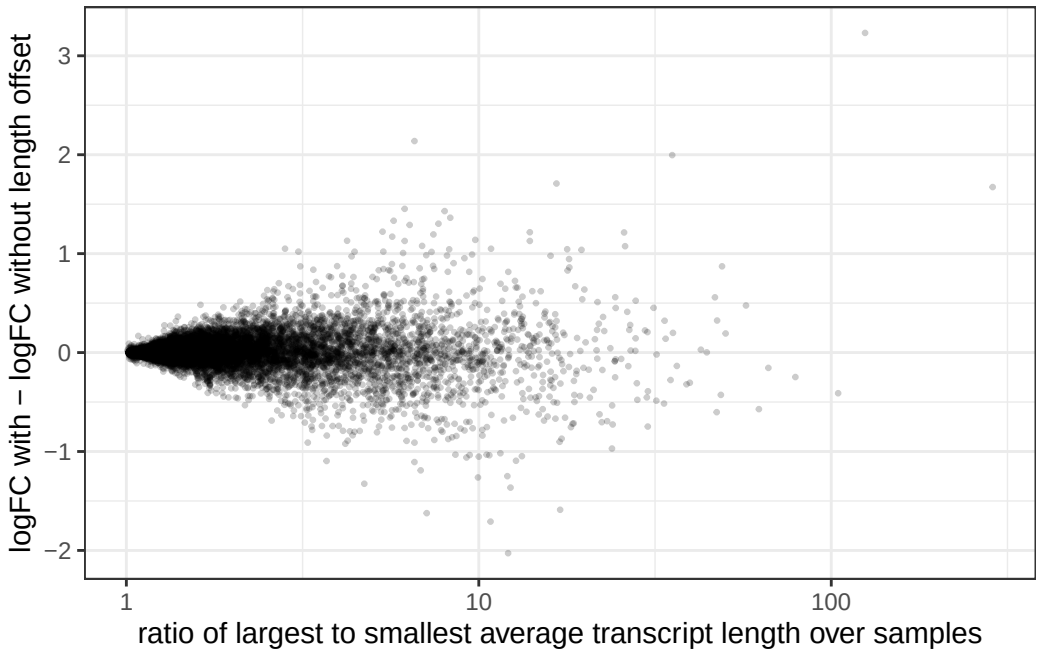
```
dgeNoLength <- dge
dgeNoLength$offset.prior <- NULL
fitNoLength <- glmQLFit(dgeNoLength, design)
qlfNoLength <- glmQLFTest(fitNoLength, coef = "ifngyes")
summary(decideTests(qlfNoLength))
```

```
       ifngyes
Down      2941
NotSig   13117
Up        3166
```

```
c(withLength = median(fit$s2.post), withoutLength = median(fitNoLength$s2.post))
```

```
   withLength withoutLength
    1.1729012     0.9188172
```

```
tibble(
  Max2MinLength = dge$genes$Max2MinLength,
  difference = qlf$table$logFC - qlfNoLength$table$logFC
) |>
  ggplot(aes(Max2MinLength, difference)) +
  geom_point(alpha = 0.2, size = 0.5) +
  scale_x_log10() +
  xlab("ratio of largest to smallest average transcript length over samples") +
  ylab("logFC with - logFC without length offset") +
  theme_bw()
```

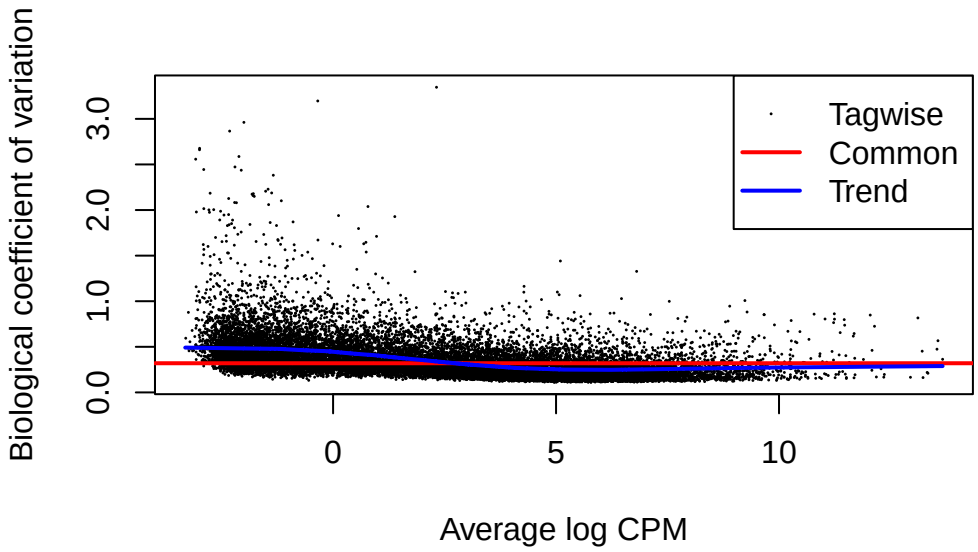


- The log fold changes only change noticeably for genes whose average transcript length differs between samples, i.e. genes with a change in transcript usage.
- For these genes, part of the apparent change in the summed counts is due to the change in transcript length, as described by Soneson et al. (2015).
- With the length offsets, the median QL dispersion is larger and fewer genes are significant. The variation of the average transcript length between samples is no longer interpreted as a change in expression.

Comparison with the legacy pipeline

The original QL pipeline (`legacy = TRUE`) first estimates trended NB dispersions with `estimateDisp`, and uses the nominal residual degrees of freedom.

```
dgeLegacy <- estimateDisp(dge, design)
plotBCV(dgeLegacy)
```

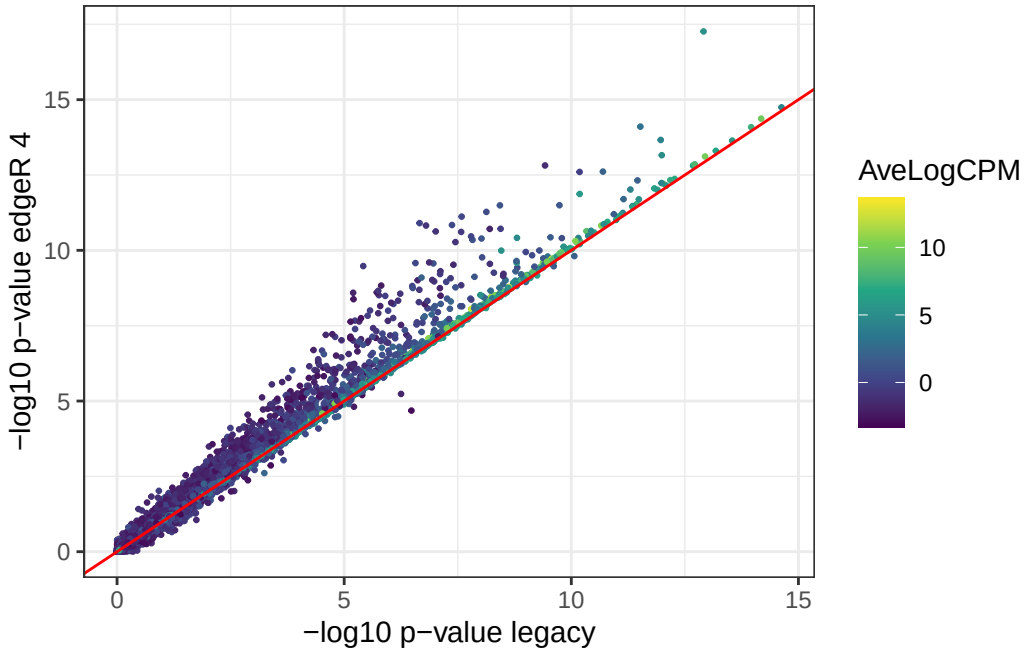



```
fitLegacy <- glmQLFit(dgeLegacy, design, legacy = TRUE)
qlfLegacy <- glmQLFTest(fitLegacy, coef = "ifngyes")
summary(decideTests(qlfLegacy))
```

```
       ifngyes
Down      2455
NotSig   13780
Up        2989
```

```
tibble(
  AveLogCPM = qlf$AveLogCPM,
  new = -log10(qlf$table$PValue),
  legacy = -log10(qlfLegacy$table$PValue)
```

```
) |>
  ggplot(aes(legacy, new, color = AveLogCPM)) +
  geom_point(size = 0.5) +
  geom_abline(intercept = 0, slope = 1, col = "red") +
  scale_color_viridis_c() +
  xlab("-log10 p-value legacy") +
  ylab("-log10 p-value edgeR 4") +
  theme_bw()
```



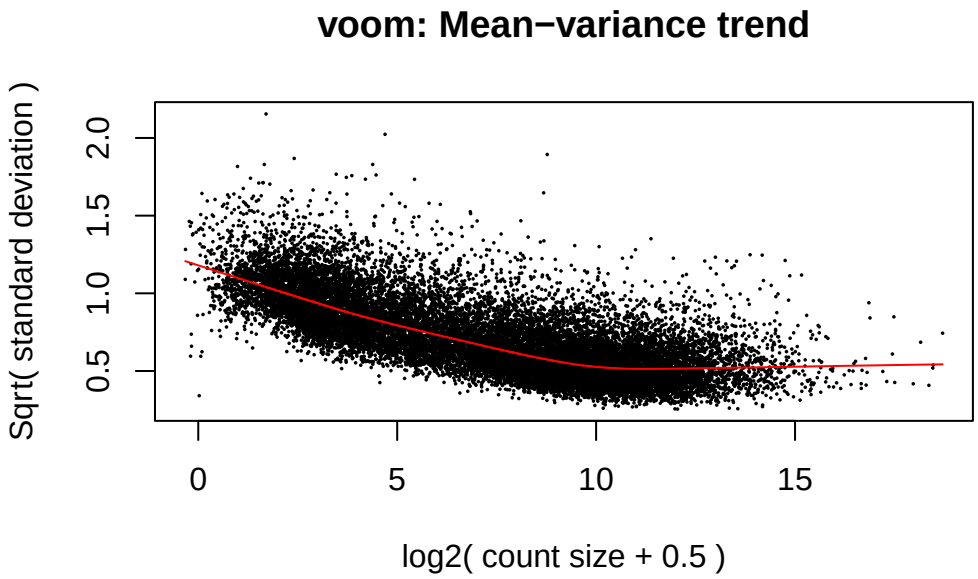
Both pipelines agree for most genes. The differences are mainly found for lowly expressed genes, for which the bias adjustment of the deviance and the degrees of freedom matters most: the legacy pipeline is conservative for these genes, and edgeR 4 gives them smaller p-values.

limma-voom

An alternative is to model log-transformed counts with linear models (Law et al. 2014).

- The log2 CPM values are still heteroscedastic: low counts are more variable.
- voom estimates the mean-variance trend of the log2 CPM values and converts it into a **precision weight** for every observation.
- The weighted log2 CPM values are then analysed with the limma pipeline: linear models, empirical Bayes moderation with `squeezeVar` and moderated t-tests.

```
v <- voom(dge, design, plot = TRUE)
```




```
fitVoom <- lmFit(v, design) |>
  eBayes()
summary(decideTests(fitVoom))[, "ifngyes"]
```

```
  Down NotSig     Up
  2664  13375   3185
```

- voom is fast and has access to the full limma toolbox: e.g. random effects via `duplicateCorrelation`, array weights and gene set tests.
- It works well when the library sizes are not too different and there are sufficient replicates.
- It becomes problematic for small counts. After the log transformation, counts of 0, 1 or 2 take only a few discrete values that depend on the prior count, and their distribution is far from normal. The mean-variance trend is also estimated less reliably at low counts, so these observations get less appropriate weights.
- edgeR 4 models the counts directly and corrects the deviance and degrees of freedom for small counts, which gives better error control when the data are dominated by small counts (Chen et al. 2025).

DESeq2

DESeq2 (Love et al. 2014) also uses an NB GLM, but

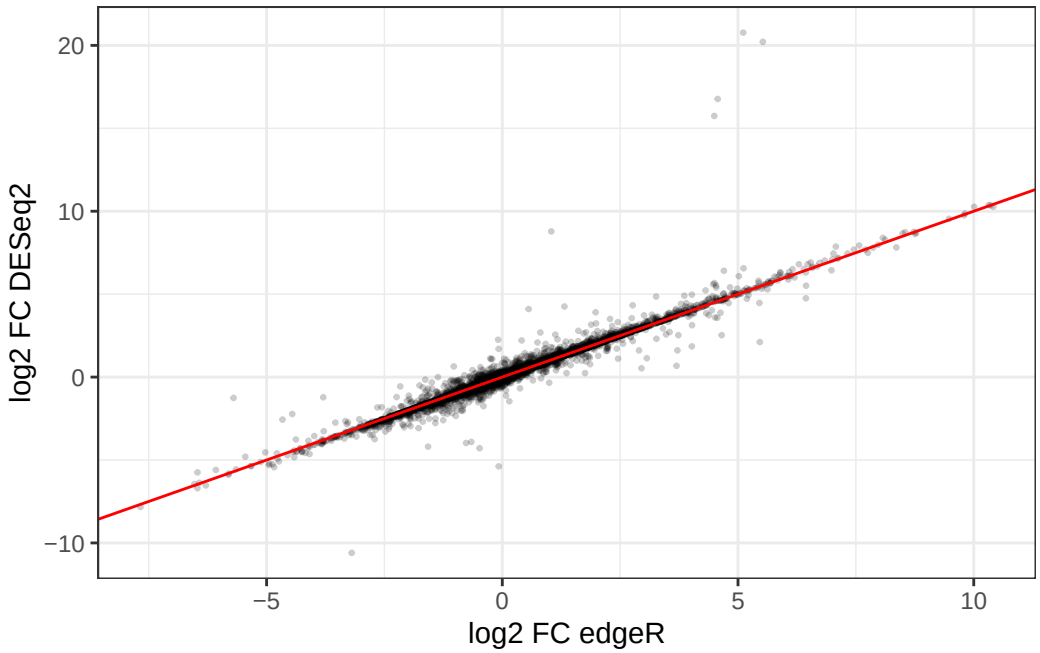
- normalises with the median of ratios method (as in proteomics),
- estimates gene-wise dispersions ϕ_g and shrinks them towards a trend with empirical Bayes,
- uses Wald tests (or LRTs) based on the normal distribution,
- optionally shrinks the log fold changes (e.g. with apeglm) for ranking and visualisation.

`DESeqDataSet` takes the tximeta object directly. It rounds the estimated counts and, like `DGEListFromTximeta` for edgeR, uses the average transcript lengths as additional offsets. Both analyses thus correct for transcript length.

```
dds <- DESeqDataSet(sg, design = ~ donor + ifng * salmonella)
dds <- dds[rownames(dge), ]
dds <- DESeq(dds, quiet = TRUE)
resDESeq <- results(dds, name = "ifng_yes_vs_no")
summary(resDESeq, alpha = 0.05)
```

```
out of 19224 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 3244, 17%
LFC < 0 (down)     : 2930, 15%
outliers [1]       : 0, 0%
low counts [2]     : 0, 0%
(mean count < 1)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

```
tibble(
  edgeR = qlf$table$logFC,
  DESeq2 = resDESeq$log2FoldChange
) |>
  ggplot(aes(edgeR, DESeq2)) +
  geom_point(alpha = 0.2, size = 0.5) +
  geom_abline(intercept = 0, slope = 1, col = "red") +
  xlab("log2 FC edgeR") +
  ylab("log2 FC DESeq2") +
  theme_bw()
```



The log fold changes agree well. The Wald tests of DESeq2 do not account for the uncertainty in the dispersion estimates and can be somewhat liberal with few replicates.

Calibration under the null

- With strong treatment effects, thousands of genes are significant.
- Can we trust the p-values, or are the tests too liberal?
- We can only check this on data for which we know that the null hypothesis holds for all genes.

→ Mock analysis!

- The naive samples of the 6 donors contain no treatment effect.
- If we split the donors at random in two groups of 3 and test for a difference between the groups, every discovery is a false positive.
- These **mock comparisons** keep everything that makes real data difficult: the biological variability between donors, the mean-variance relation, low counts and the correlation between genes.
- There are $\binom{6}{3}/2 = 10$ different splits of the 6 donors into two groups of 3.
- We analyse every split with edgeR 4 (QL F-test) and with the legacy NB likelihood ratio test (`glmFit` and `glmLRT`).

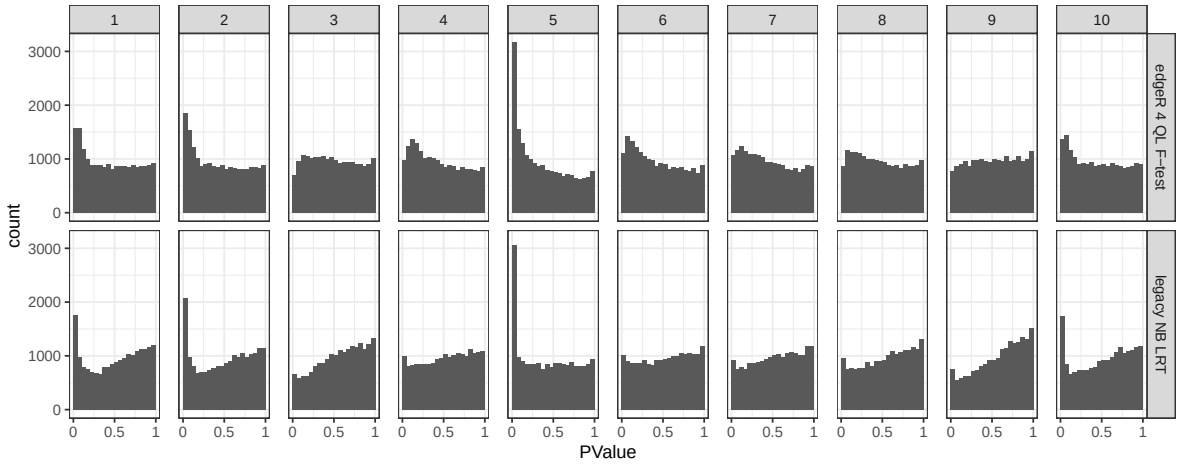
```
naiveSamples <- sg$condition == "naive"
dgeNaive <- DGEListFromTximeta(sg[, naiveSamples])
dgeNaive <- dgeNaive[rownames(dge), , keep.lib.sizes = FALSE]
dgeNaive <- normLibSizes(dgeNaive)
donorNaive <- droplevels(sg$donor[naiveSamples])

donorSplits <- combn(levels(donorNaive), 3)
donorSplits <- donorSplits[, donorSplits[1, ] == levels(donorNaive)[1]]
ncol(donorSplits)
```

```
[1] 10
```

```
mockPValues <- lapply(seq_len(ncol(donorSplits)), function(split) {
  mockGroup <- factor(donorNaive %in% donorSplits[, split])
  mockDesign <- model.matrix(~ mockGroup)
  fitQL <- glmQLFit(dgeNaive, mockDesign)
  fitLRT <- glmFit(estimateDisp(dgeNaive, mockDesign), mockDesign)
  tibble(
    split = split,
    `edgeR 4 QL F-test` = glmQLFTest(fitQL, coef = 2)$table$PValue,
    `legacy NB LRT` = glmLRT(fitLRT, coef = 2)$table$PValue
  )
}) |>
  bind_rows() |>
  pivot_longer(-split, names_to = "method", values_to = "PValue")
```

```
mockPValues |>
  ggplot(aes(PValue)) +
  geom_histogram(breaks = seq(0, 1, 0.05)) +
  facet_grid(method ~ split) +
  scale_x_continuous(breaks = c(0, 0.5, 1), labels = c("0", "0.5", "1")) +
  theme_bw() +
  theme(panel.spacing.x = unit(0.8, "lines"))
```




```
mockPValues |>
  group_by(method, split) |>
  summarise(
    p05 = mean(PValue < 0.05),
    p01 = mean(PValue < 0.01),
    falseDiscoveries = sum(p.adjust(PValue, "BH") < 0.05),
    .groups = "drop_last"
  ) |>
  summarise(
    pct05 = 100 * mean(p05),
    pct01 = 100 * mean(p01),
    meanFD = mean(falseDiscoveries),
    maxFD = max(falseDiscoveries)
  ) |>
  as.data.frame()
```

```
             method    pct05    pct01 meanFD maxFD
1 edgeR 4 QL F-test 7.002705 1.399293    0.0     0
2     legacy NB LRT 7.244590 2.593113  160.3  1484
```

`pct05` and `pct01` are the average percentages of p-values below 0.05 and 0.01, `meanFD` and `maxFD` the average and maximum number of false discoveries at a 5% FDR over the 10 splits.

- Under the null, about 5% of the p-values should be below 0.05 and 1% below 0.01, and at a 5% FDR we expect no or very few discoveries.
- The p-value histograms of the edgeR 4 QL F-test are approximately uniform. There is some variation between the splits: in some splits the two mock groups differ by chance more than in others, e.g. because of differences in the processing of the donors.

- The edgeR 4 QL F-test does not return a single false discovery in any of the splits.
- The p-value histograms of the legacy LRT are not uniform: they have a peak near 0 and increase towards 1. The LRT treats the estimated dispersions as known. The dispersions are shrunken towards a trend, so for some genes they are too small and the test is liberal, while for others they are too large and the test is conservative. With few replicates, the uncertainty of the dispersions matters most.
- The liberal tail leads to false discoveries in every split, up to more than a thousand. This is why edgeR uses QL F-tests, which account for the uncertainty in the gene-wise variability.

Normalisation

Normalisation corrects for technical differences between samples:

- **Sequencing depth**: a sample that is sequenced deeper has higher counts for all genes.
- **RNA composition**: if a few genes are very highly expressed in one sample, they take up a large fraction of the reads, and all other genes get lower counts.
- Other technical effects, e.g. GC content or transcript length, can also be accounted for.

We illustrate it on the parathyroid data. We first sum the counts of the technical replicates.

```
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"))

paraCounts <- t(rowsum(t(assay(parathyroidGenesSE)), colData(parathyroidGenesSE)$experiment))
paraSamples <- colData(parathyroidGenesSE) |>
  as.data.frame() |>
  distinct(experiment, .keep_all = TRUE) |>
  dplyr::select(experiment, patient, treatment, time)
paraCounts <- paraCounts[, paraSamples$experiment]
dim(paraCounts)
```

```
[1] 63193    23
```

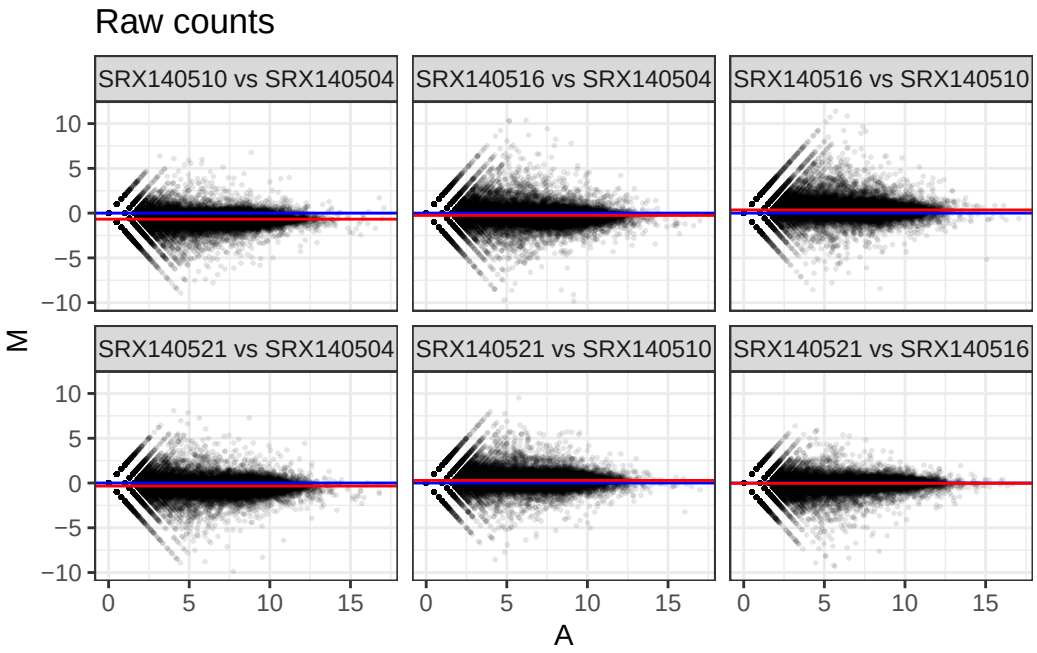
We compare the four control samples at 48 hours in MD plots: the difference M in log2 count against the average log2 count A, for every pair of samples.

```r
mdPairs <- function(counts, libSize) {
  pairs <- combn(ncol(counts), 2)
  map(seq_len(ncol(pairs)), function(k) {
    i <- pairs[1, k]
    j <- pairs[2, k]
    use <- counts[, i] > 0 & counts[, j] > 0
    tibble(
      pair = paste(colnames(counts)[j], "vs", colnames(counts)[i]),
      A = (log2(counts[use, i]) + log2(counts[use, j])) / 2,
      M = log2(counts[use, j] / libSize[j]) - log2(counts[use, i] / libSize[i])
    )
  }) |>
    bind_rows()
}

ctrl48 <- paraSamples$treatment == "Control" & paraSamples$time == "48h"
ctrlCounts <- paraCounts[, ctrl48]
colSums(ctrlCounts) / 1e6
```

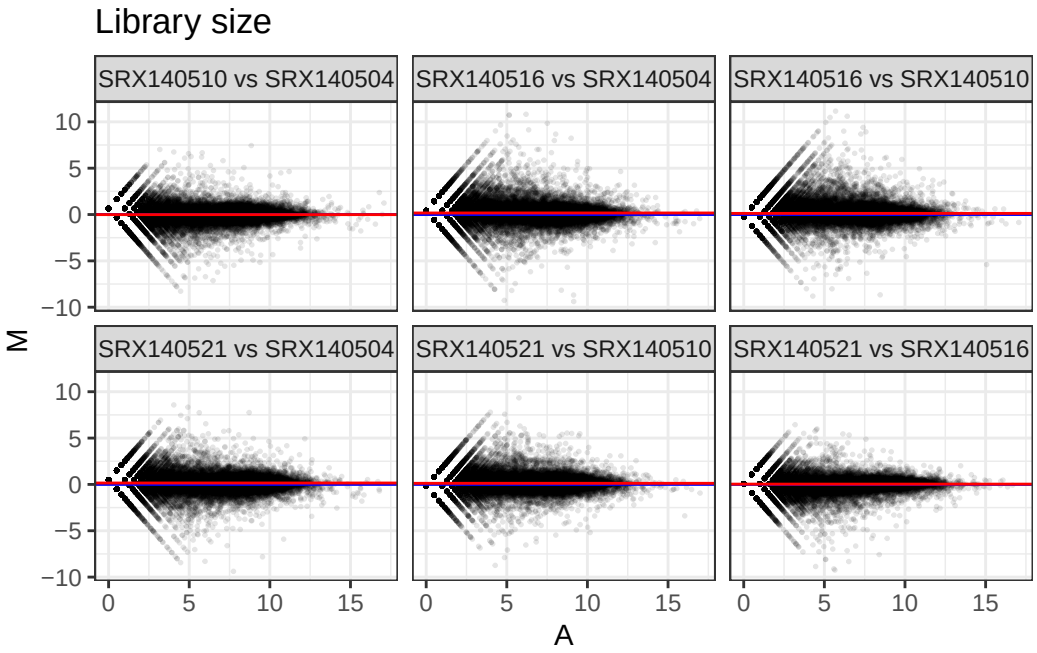
```
SRX140504 SRX140510 SRX140516 SRX140521
10.827109  6.844144  8.064268  7.701432
```

```r
mdPlot <- function(mdData) {
  mdData |>
    group_by(pair) |>
    mutate(median = median(M)) |>
    ggplot(aes(A, M)) +
    geom_point(alpha = 0.1, size = 0.3) +
    geom_hline(yintercept = 0, col = "blue") +
    geom_hline(aes(yintercept = median), col = "red") +
    facet_wrap(~ pair) +
    theme_bw()
}
mdPlot(mdPairs(ctrlCounts, rep(1, ncol(ctrlCounts)))) +
  ggtitle("Raw counts")
```



The log fold changes between replicates are not centred around zero (blue line). Dividing by the library size, i.e. using the log library size as offset, removes most of the bias:

```
mdPlot(mdPairs(ctrlCounts, colSums(ctrlCounts))) +
  ggtitle("Library size")
```



TMM (edgeR)

The trimmed mean of M-values (TMM) method (Robinson and Oshlack 2010) assumes that most genes are not differentially expressed.

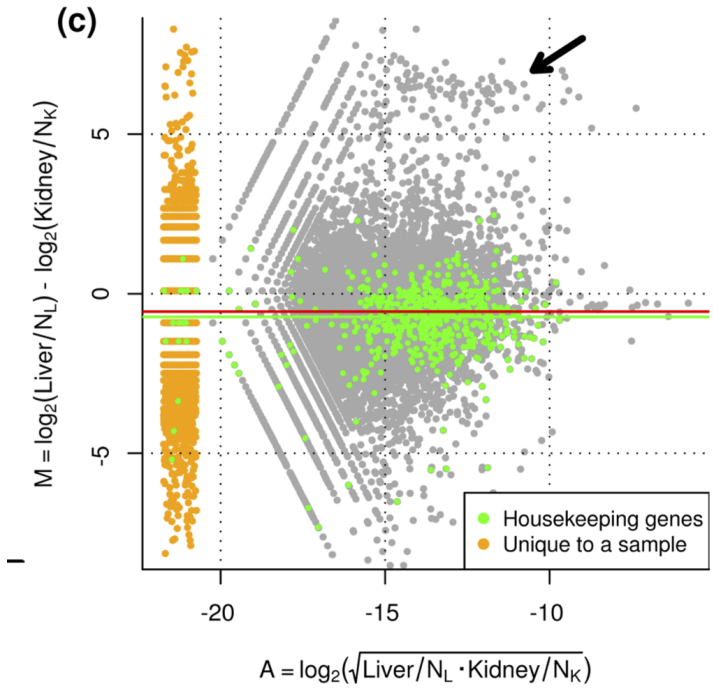


Figure 5: Composition bias: a few highly expressed genes in one sample take up a large fraction of the reads (from Robinson and Oshlack, 2010).

For every sample i, a normalisation factor F_i is computed with respect to a reference sample r:

$$\log_2 F_i = \frac{\sum_{g \in G^*} w_{gi} M_{gi}}{\sum_{g \in G^*} w_{gi}}, \quad M_{gi} = \log_2 \frac{Y_{gi}/N_i}{Y_{gr}/N_r}, \quad w_{gi} = \left(\frac{N_i - Y_{gi}}{N_i Y_{gi}} + \frac{N_r - Y_{gr}}{N_r Y_{gr}} \right)^{-1},$$

- G^*: the genes that remain after trimming the 30% most extreme M values and the 5% most extreme average expression values,
- w_{gi}: precision weights, the inverse of the approximate variance of M_{gi}, so that low counts get less weight.

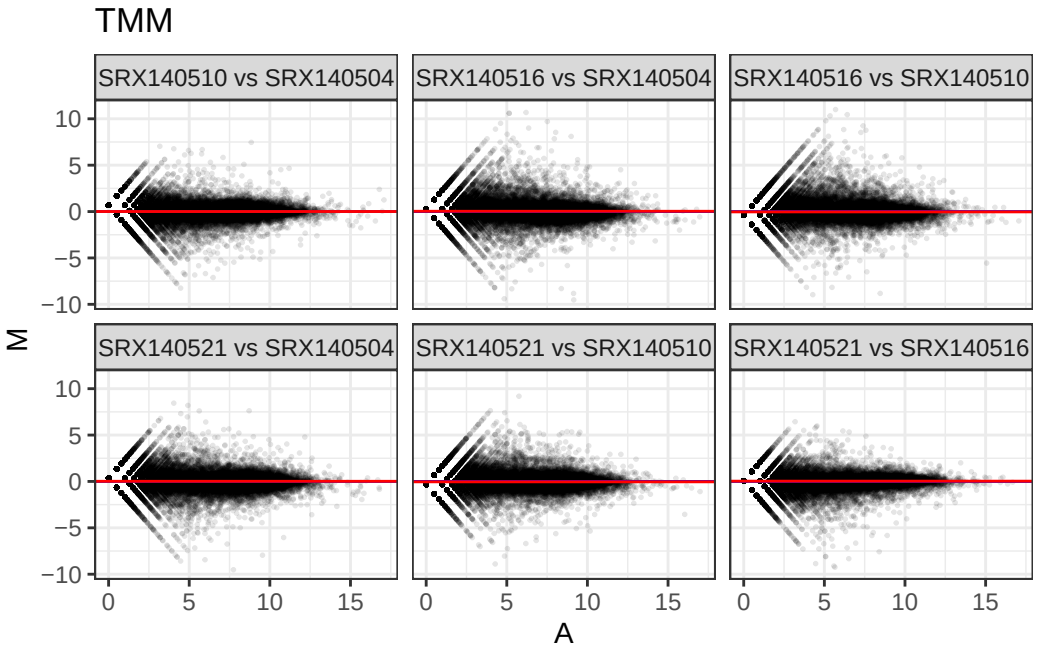
The effective library size $N_i^{\text{eff}} = N_i F_i$ is used as offset.

```
dgePara <- DGEList(ctrlCounts) |>
  normLibSizes()
dgePara$samples
```

```
       group lib.size norm.factors
```

```
SRX140504      1 10827109    0.9568210
SRX140510      1  6844144    0.9443689
SRX140516      1  8064268    1.0511507
SRX140521      1  7701432    1.0528406
```

```
effLibSize <- dgePara$samples$lib.size * dgePara$samples$norm.factors
mdPlot(mdPairs(ctrlCounts, effLibSize)) +
  ggtitle("TMM")
```



Median of ratios (DESeq2)

The median of ratios method (Love et al. 2014) that we used in proteomics computes a size factor for each sample with respect to a pseudo-reference sample, the geometric mean of the counts over all samples:

$$s_i = \text{median}_{g:Y_g^*>0} \frac{Y_{gi}}{Y_g^*}, \quad Y_g^* = \left(\prod_{i=1}^{n} Y_{gi} \right)^{1/n}.$$

```
ddsPara <- DESeqDataSetFromMatrix(ctrlCounts, colData = DataFrame(sample = colnames(ctrlCount
ddsPara <- estimateSizeFactors(ddsPara)
tibble(
  sample = colnames(ctrlCounts),
  TMM = effLibSize / exp(mean(log(effLibSize))),
  MoR = sizeFactors(ddsPara) / exp(mean(log(sizeFactors(ddsPara))))
)
```

```
# A tibble: 4 x 3
  sample      TMM   MoR
  <chr>     <dbl> <dbl>
1 SRX140504 1.26  1.27
2 SRX140510 0.785 0.790
3 SRX140516 1.03  1.02
4 SRX140521 0.984 0.979
```

After rescaling both to a geometric mean of one, both methods give very similar normalisation factors.

Independent filtering

- Lowly expressed genes have little power to be found DE, but they count in the multiple testing correction.
- Removing them before testing can increase the number of discoveries at the same FDR.
- But we then only test the genes that pass the filter. The FDR is only controlled if the filter does not change the distribution of the test statistic of the null genes (Bourgon et al. 2010).

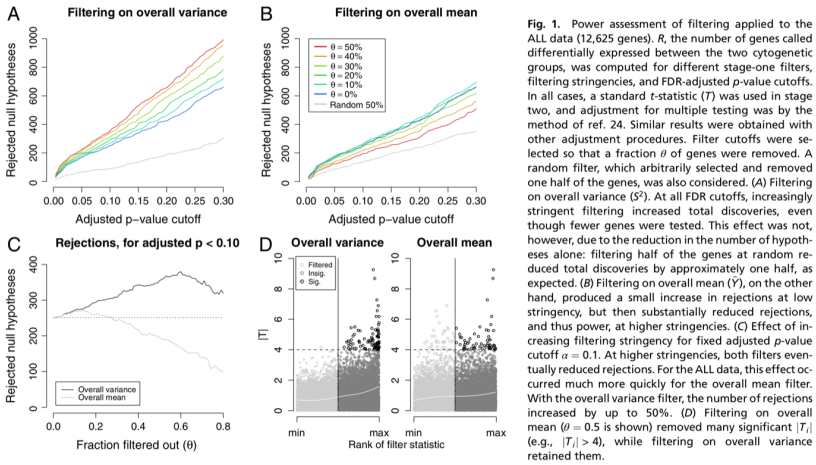


Fig. 1. Power assessment of filtering applied to the ALL data (12,625 genes). *R*, the number of genes called differentially expressed between the two cytogenetic groups, was computed for different stage-one filters, filtering stringencies, and FDR-adjusted *p*-value cutoffs. In all cases, a standard *t*-statistic (*T*) was used in stage two, and adjustment for multiple testing was by the method of ref. 24. Similar results were obtained with other adjustment procedures. Filter cutoffs were selected so that a fraction θ of genes were removed. A random filter, which arbitrarily selected and removed one half of the genes, was also considered. (*A*) Filtering on overall variance (S^2). At all FDR cutoffs, increasingly stringent filtering increased total discoveries, even though fewer genes were tested. This effect was not, however, due to the reduction in the number of hypotheses alone: filtering half of the genes at random reduced total discoveries by approximately one half, as expected. (*B*) Filtering on overall mean ($\bar{Y}$), on the other hand, produced a small increase in rejections at low stringency, but then substantially reduced rejections, and thus power, at higher stringencies. (*C*) Effect of increasing filtering stringency for fixed adjusted *p*-value cutoff $\alpha = 0.1$. At higher stringencies, both filters eventually reduced rejections. For the ALL data, this effect occurred much more quickly for the overall mean filter. With the overall variance filter, the number of rejections increased by up to 50%. (*D*) Filtering on overall mean ($\theta = 0.5$ is shown) removed many significant $|T_i|$ (e.g., $|T_i| > 4$), while filtering on overall variance retained them.

Figure 6: Figure 1 of Bourgon et al. (2010).

Filtering is valid if the filter statistic is independent of the test statistic under the null hypothesis.

We illustrate this with simulated null data: two groups of 5 samples, no DE genes.

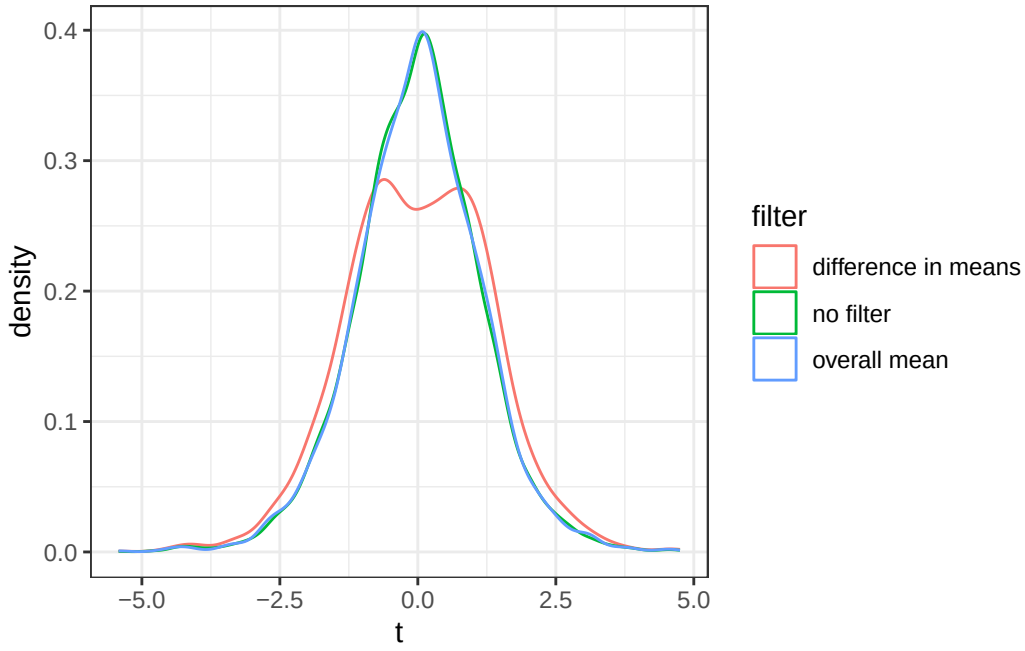
```
set.seed(24)
nGenes <- 5000
group <- factor(rep(c("A", "B"), each = 5))
mu <- 2^runif(nGenes, 0, 10)
simCounts <- matrix(rnbinom(nGenes * 10, mu = mu, size = 5), nrow = nGenes)

rowT <- function(x, group) {
  xA <- x[, group == "A"]
  xB <- x[, group == "B"]
  se <- sqrt(matrixStats::rowVars(xA) / ncol(xA) + matrixStats::rowVars(xB) / ncol(xB))
  (rowMeans(xB) - rowMeans(xA)) / se
}
tStat <- rowT(log2(simCounts + 1), group)

filters <- tibble(
  t = tStat,
  `overall mean` = rowMeans(simCounts),
  `difference in means` = abs(rowMeans(simCounts[, group == "B"]) - rowMeans(simCounts[, grou
) |>
  mutate(
    across(c(`overall mean`, `difference in means`), ~ .x > quantile(.x, 0.5))
```

```
  )

bind_rows(
  tibble(filter = "no filter", t = filters$t),
  tibble(filter = "overall mean", t = filters$t[filters$`overall mean`]),
  tibble(filter = "difference in means", t = filters$t[filters$`difference in means`])
) |>
  ggplot(aes(t, color = filter)) +
  geom_density() +
  theme_bw()
```



- Filtering on the overall mean (independent of the group labels) leaves the null distribution of the test statistic unchanged.
- Filtering on the difference in means selects genes with large test statistics: the null distribution changes and the FDR is no longer controlled.

`filterByExpr` uses the counts per million, irrespective of the group labels: it keeps genes with a CPM above a threshold, corresponding to about 10 reads in a sample with the median library size, in at least as many samples as the smallest group (or more precisely, the inverse of the largest leverage of the design).

Transcript-level analysis

DGE, DTE and DTU

Salmon quantifies the expression of every transcript. This allows three types of analyses (Soneson et al. 2015):

- **Differential gene expression (DGE)**: does the total expression of the gene change?
- **Differential transcript expression (DTE)**: does the expression of a transcript change?
- **Differential transcript usage (DTU)**: does the relative abundance of the transcripts of a gene change?

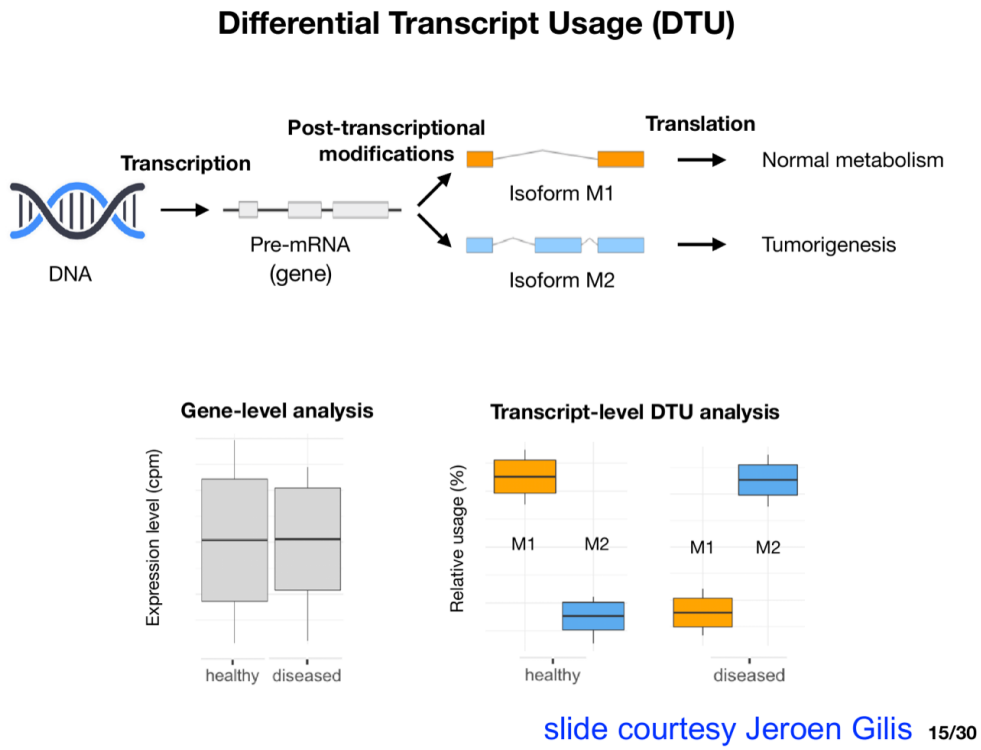


Figure 7: Differential transcript usage: the gene expression stays the same, while the usage of the isoforms switches (slide courtesy of Jeroen Gilis).

Transcript-level data

For the transcript-level analyses we use the naive and IFNγ-primed samples of the macrophage data: a randomised complete block design with the 6 donors as blocks. The salmon quan-

tifications of the macrophage package contain 20 Gibbs samples per sample, which we now need.

```
macData <- coldata |>
  filter(condition %in% c("naive", "IFNg")) |>
  mutate(condition = droplevels(condition))
macData |>
  dplyr::select(names, donor, condition)
```

```
            names  donor condition
1  SAMEA103885102 diku_1     naive
2  SAMEA103885347 diku_1      IFNg
3  SAMEA103885182 eiwy_1     naive
4  SAMEA103885136 eiwy_1      IFNg
5  SAMEA103885368 fikt_3     naive
6  SAMEA103885218 fikt_3      IFNg
7  SAMEA103885284 ieki_2     naive
8  SAMEA103885059 ieki_2      IFNg
9  SAMEA103885111 podx_1     naive
10 SAMEA103884919 podx_1      IFNg
11 SAMEA103885262 qaqx_1     naive
12 SAMEA103885228 qaqx_1      IFNg
```

```
macDesign <- model.matrix(~ donor + condition, data = macData)
```

We import the quantifications with tximeta, which also imports the Gibbs samples as additional assays (`infRep1`, `infRep2`, …).

```
macSt <- tximeta(macData)
assayNames(macSt) |> head()
```

```
[1] "counts"    "abundance" "length"    "infRep1"   "infRep2"   "infRep3"
```

Quantification uncertainty

- Many reads are compatible with several transcripts of the same gene. Their assignment is estimated with the EM algorithm.
- Transcript counts therefore have an additional source of variability: **read-to-transcript ambiguity**. Transcripts that share most of their sequence with other transcripts have a much more uncertain count.

- Salmon and kallisto can quantify this uncertainty with bootstrap or Gibbs samples (inferential replicates), e.g. with `salmon quant ... --numGibbsSamples 20`.
- `catchSalmon` estimates an overdispersion factor for every transcript from the inferential replicates (Baldoni et al. 2024). If a count is c times more variable than expected because of the ambiguity, dividing the count by c gives a **scaled count** with the usual mean-variance relation, which can be analysed with the standard edgeR pipeline.

Why does dividing work? Suppose that the ambiguity inflates the technical variance of the count Y_{ti} of transcript t by a factor c_t:

$$\mathrm{Var}(Y_{ti}) \approx c_t \, \mu_{ti}.$$

The scaled count $\tilde{Y}_{ti} = Y_{ti}/c_t$ then has

$$E[\tilde{Y}_{ti}] = \frac{\mu_{ti}}{c_t}, \quad \mathrm{Var}(\tilde{Y}_{ti}) = \frac{c_t \, \mu_{ti}}{c_t^2} = \frac{\mu_{ti}}{c_t} = E[\tilde{Y}_{ti}],$$

so it again has a Poisson-like technical variability, and biological variability can be modelled on top of it as usual.

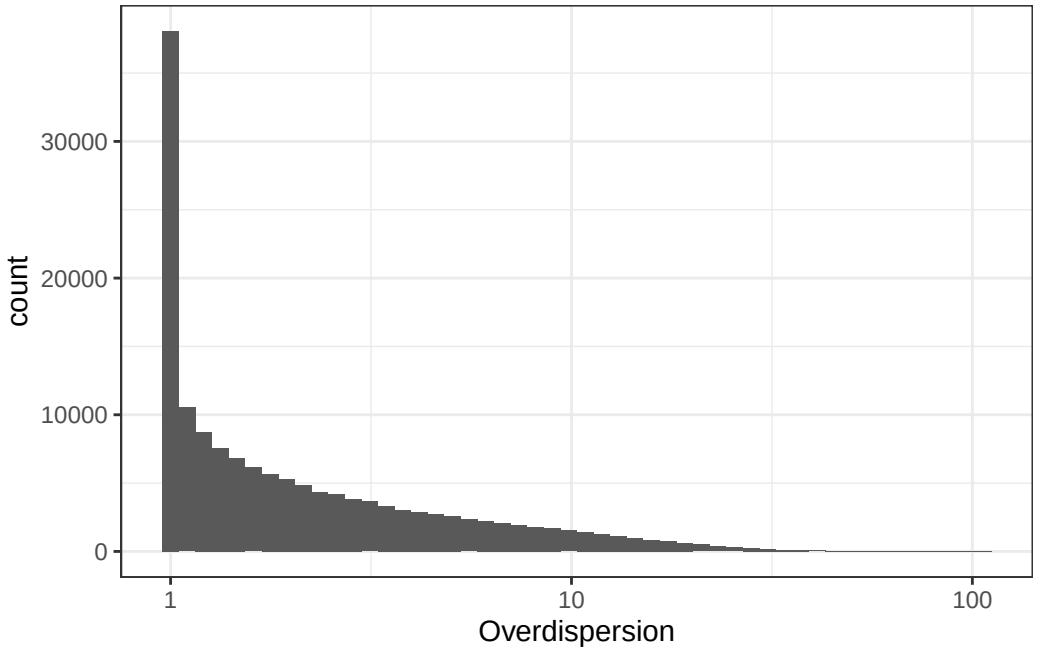
- The scaled count is an **effective count**: a transcript whose count is 4 times too variable carries the same information as a count that is 4 times smaller.
- The factor c_t is estimated from the inferential replicates: within every sample the variance of the bootstrap or Gibbs counts is compared with their mean, this ratio is pooled over the samples and moderated, and it is at least 1.
- The idea is similar to that of the juggling offsets below: both make the standard bulk RNA-seq tools applicable by changing their input. Scaling changes the counts, juggling offsets change the offset.

`catchSalmon` reads the salmon output directly. When the data are imported with tximeta, `DGEListFromTximeta` does the same: it estimates the overdispersion factors from the inferential replicates and, with `divide = TRUE`, returns the divided counts (Baldoni et al. 2024, 2025).

```
macRaw <- DGEListFromTximeta(macSt)
macDivided <- DGEListFromTximeta(macSt, divide = TRUE)
summary(macDivided$genes$Overdispersion)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   1.104   2.620   2.335 106.408
```

```
tibble(Overdispersion = macDivided$genes$Overdispersion) |>
  filter(Overdispersion > 1) |>
  ggplot(aes(Overdispersion)) +
  geom_histogram(bins = 50) +
  scale_x_log10() +
  theme_bw()
```



Some transcripts have counts that are many times more variable than their expression level suggests.

Long reads (PacBio, Oxford Nanopore) sequence full-length transcripts and thus remove most of the read-to-transcript ambiguity. Their lower throughput and higher error rate, however, imply that short reads remain the standard for quantification.

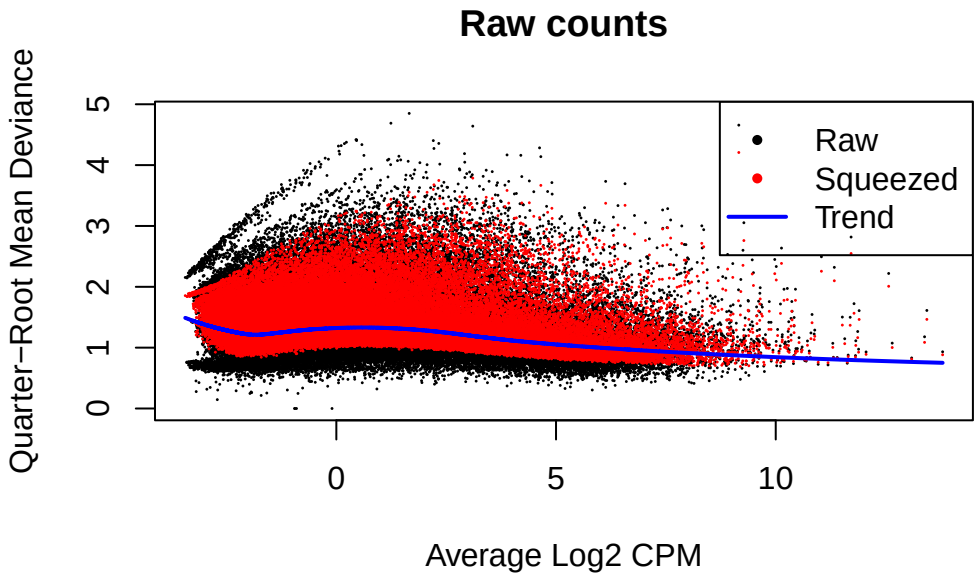
Differential transcript expression

We analyse the raw and the divided transcript counts with the same edgeR pipeline.

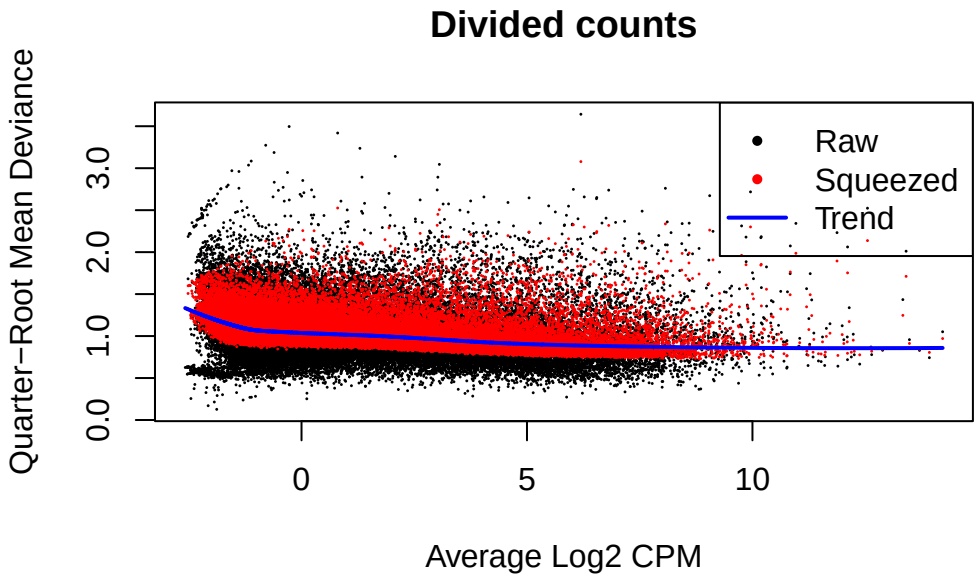
```
dteAnalysis <- function(y) {
  y$genes <- data.frame(tx = y$genes$tx_name, gene = y$genes$gene_id,
                        Overdispersion = y$genes$Overdispersion, row.names = rownames(y))
  keep <- filterByExpr(y, macDesign)
```

```
  y <- y[keep, , keep.lib.sizes = FALSE]
  y <- normLibSizes(y)
  fit <- glmQLFit(y, macDesign)
  list(y = y, fit = fit, test = glmQLFTest(fit, coef = "conditionIFNg"))
}
dteRaw <- dteAnalysis(macRaw)
dteScaled <- dteAnalysis(macDivided)
```

```
plotQLDisp(dteRaw$fit, main = "Raw counts")
```




```
plotQLDisp(dteScaled$fit, main = "Divided counts")
```




```
c(raw = median(dteRaw$fit$s2.post), divided = median(dteScaled$fit$s2.post))
```

```
     raw  divided
2.934842 1.152101
```

- With the raw counts, the QL dispersions are inflated: the ambiguity overdispersion is not captured by the NB model.
- After dividing by the overdispersion, the QL dispersions are centred around one.

```
summary(decideTests(dteScaled$test))
```

```
       conditionIFNg
Down            7460
NotSig         36607
Up              7833
```

Differential transcript usage

The usage of transcript t of gene g in sample i is its fraction of the gene's total count:

$$\pi_{gti} = \frac{E[Y_{gti}]}{E[Y_{g.i}]}, \quad Y_{g.i} = \sum_t Y_{gti}.$$

Only genes with at least two transcripts can show DTU.

```
yTx <- dteScaled$y
multiTxGenes <- yTx$genes |>
  dplyr::count(gene) |>
  filter(n > 1) |>
  pull(gene)
dtu <- yTx[yTx$genes$gene %in% multiTxGenes, ]
c(genes = length(multiTxGenes), transcripts = nrow(dtu))
```

```
      genes transcripts
      10731       44973
```

Juggling offsets

In DGE we use the (effective) library size as offset, so that the parameters model the expression relative to the sequencing depth. For DTU we **replace this offset by the total count of the gene** (Segers et al. 2023):

$$\begin{cases} E[Y_{gti}] &= \mu_{gti} \\ \log \mu_{gti} &= \mathbf{x}_i^T \beta_{gt} + \log Y_{g.i} \\ \mathrm{Var}[Y_{gti}] &= \sigma^2_{gt}(\mu_{gti} + \phi \mu^2_{gti}) \end{cases} \quad \Rightarrow \quad \log \frac{\mu_{gti}}{Y_{g.i}} = \mathbf{x}_i^T \beta_{gt}.$$

- The parameters now model the log usage of the transcript.
- A condition effect in β_{gt} is a change in usage: DTU.
- The gene total already accounts for the sequencing depth: no further normalisation is needed.
- Everything else in the edgeR pipeline stays the same: QL dispersions, empirical Bayes and moderated F-tests. The same trick works with DESeq2.
- The gene total is plugged in as known; its uncertainty is ignored.

```
geneTotals <- rowsum(dtu$counts, dtu$genes$gene)[dtu$genes$gene, ]
dtu$offset <- log(pmax(geneTotals, 1))

fitDtu <- glmQLFit(dtu, macDesign)
qlfDtu <- glmQLFTest(fitDtu, coef = "conditionIFNg")
topTags(qlfDtu, n = 5)
```

```
Coefficient:  conditionIFNg
                                  tx                gene Overdispersion
ENST00000572992.1 ENST00000572992.1  ENSG00000175643.9       1.110110
ENST00000482504.1 ENST00000482504.1 ENSG00000097033.14       1.434778
ENST00000370558.8 ENST00000370558.8 ENSG00000097033.14       1.632881
ENST00000439685.6 ENST00000439685.6 ENSG00000091986.15       1.473807
ENST00000470745.5 ENST00000470745.5 ENSG00000165637.13       1.384039
                       logFC    logCPM         F       PValue          FDR
ENST00000572992.1  5.3601851 3.2669616  430.7295 1.912127e-10 4.828838e-06
ENST00000482504.1  0.4264574 5.9518368 1080.1001 2.147439e-10 4.828838e-06
ENST00000370558.8 -0.2212980 6.8474862  556.2743 3.689324e-09 4.434938e-05
ENST00000439685.6  0.2426146 7.0326825  523.2968 4.790388e-09 4.434938e-05
ENST00000470745.5  4.6238825 0.3632601  183.1752 4.930667e-09 4.434938e-05
```

```
summary(decideTests(qlfDtu))
```

```
       conditionIFNg
Down            1209
NotSig         42601
Up              1163
```

The estimated global NB dispersion is close to zero:

```
fitDtu$dispersion
```

```
[1] 5.285295e-06
```

Given the gene total, the transcript counts are no longer overdispersed by the biological variation of the gene expression, which cancels out in the usage. The model then reduces to a quasi-Poisson model on the usages, close to the quasi-binomial model of the dedicated DTU tool satuRn.

diffSplice

edgeR and limma also have a dedicated function for differential usage. `diffSplice` uses the transcript-level fit with the usual offsets and tests whether the log fold change of a transcript differs from the log fold change of the gene as a whole. This is a test for a transcript $\times$ condition interaction within each gene. In recent versions of edgeR and limma, `diffSplice` is a generic function that works on both edgeR (`glmQLFit`) and limma (`voomLmFit`) fits; it replaces the older `diffSpliceDGE` (Baldoni et al. 2025).

```
fitTx <- dteScaled$fit
ds <- diffSplice(fitTx[fitTx$genes$gene %in% multiTxGenes, ],
                 coef = "conditionIFNg", geneid = "gene", exonid = "tx", verbose = FALSE)
topSplice(ds, test = "t", number = 5)
```

```
                                tx               gene Overdispersion     logFC
ENST00000557239.5 ENST00000557239.5 ENSG00000140105.17      10.363184 -
3.749029
ENST00000554772.5 ENST00000554772.5 ENSG00000140105.17       3.568620  2.876456
ENST00000544944.6 ENST00000544944.6 ENSG00000104859.14       3.029913  2.033541
ENST00000435136.6 ENST00000435136.6 ENSG00000087266.15      15.946682  6.513229
ENST00000490682.5 ENST00000490682.5 ENSG00000064012.21       1.256303  2.196197
                          t      P.Value          FDR
ENST00000557239.5 -12.98560 4.096922e-27 1.842509e-22
ENST00000554772.5  11.54364 4.791145e-23 1.077361e-18
ENST00000544944.6  14.49752 8.389585e-20 1.233792e-15
ENST00000435136.6  11.73338 1.097363e-19 1.233792e-15
ENST00000490682.5  12.62494 2.872280e-18 2.583501e-14
```

```
topSplice(ds, test = "F", number = 5)
```

```
                                gene NExons        F      P.Value          FDR
ENSG00000140105.17 ENSG00000140105.17     28 20.26873 1.105625e-39 1.186446e-
35
ENSG00000065882.15 ENSG00000065882.15      8 48.66688 3.025609e-21 1.623390e-
17
ENSG00000087266.15 ENSG00000087266.15     17 16.80134 3.019978e-20 1.080246e-
16
ENSG00000104859.14 ENSG00000104859.14      9 26.85855 8.261721e-16 2.216413e-
12
ENSG00000064012.21 ENSG00000064012.21      9 23.27945 1.838255e-15 3.945264e-
12
```

```
topSplice(ds, test = "simes", number = 5)
```

```
                                gene NExons      P.Value          FDR
ENSG00000140105.17 ENSG00000140105.17     28 1.147138e-25 1.230994e-21
ENSG00000104859.14 ENSG00000104859.14      9 7.550626e-19 4.051288e-15
ENSG00000087266.15 ENSG00000087266.15     17 1.865516e-18 6.672952e-15
ENSG00000064012.21 ENSG00000064012.21      9 2.585052e-17 6.935048e-14
ENSG00000065882.15 ENSG00000065882.15      8 3.808713e-17 8.174261e-14
```

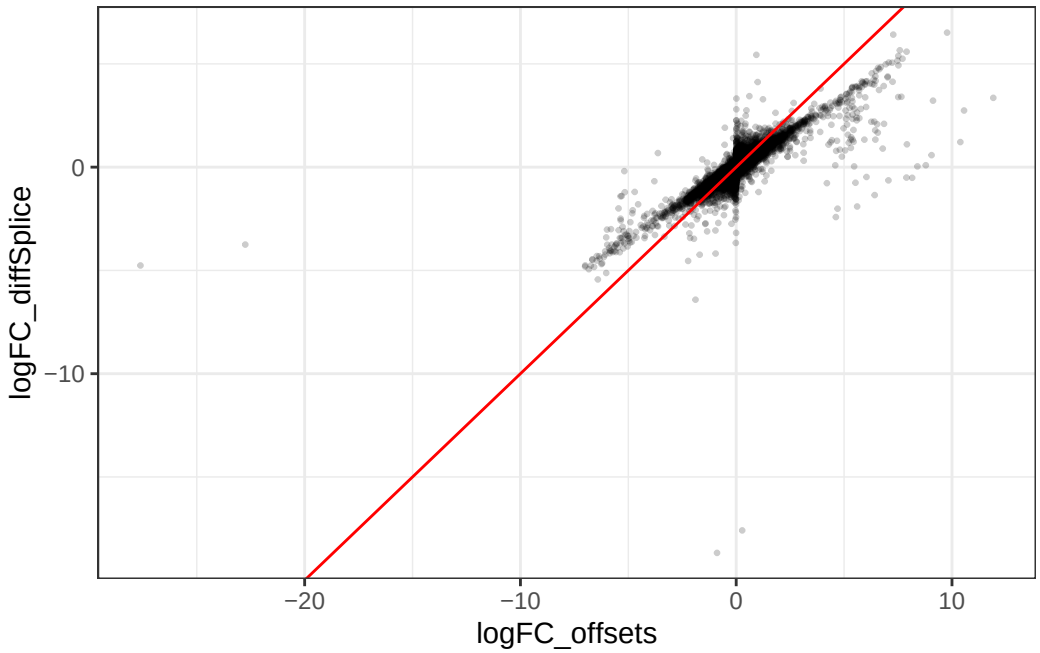
- `test = "t"`: transcript-level tests.
- `test = "F"`: gene-level F-test of any DTU within the gene. It is most powerful when several transcripts change in opposite directions.
- `test = "simes"`: gene-level test that combines the transcript-level p-values with the Simes method. It is most powerful when one transcript stands out.

In the simulation study of Baldoni et al. (2025), the edgeR and limma pipelines on divided counts controlled the FDR in all scenarios, also for small sample sizes, and edgeR with Simes p-values was the most powerful.

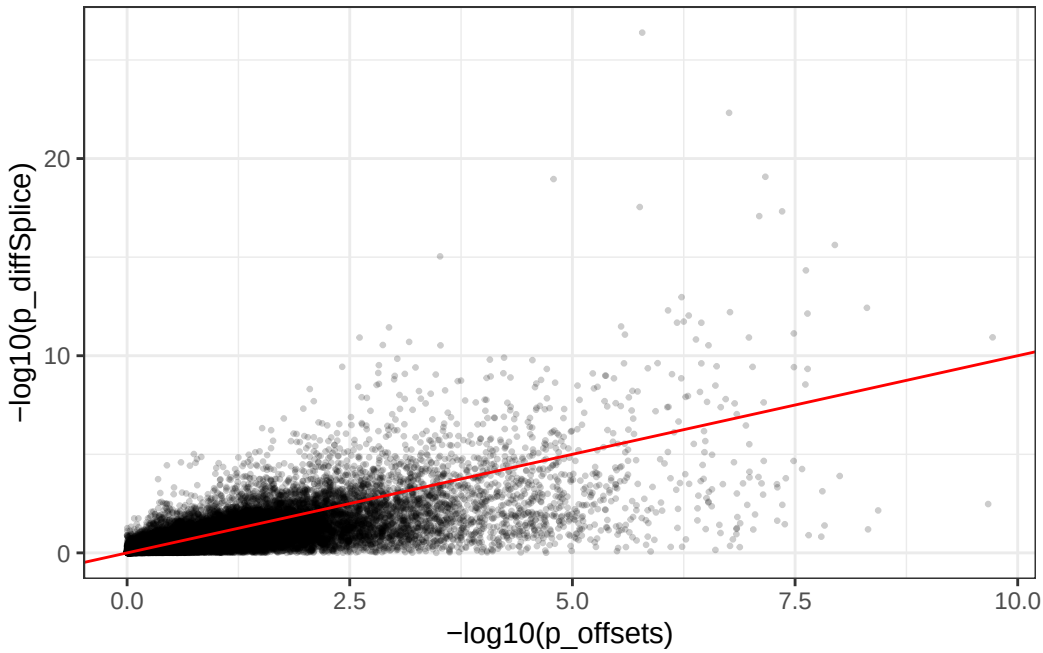
We compare both approaches on the transcripts that are analysed by both.

```
dtuComparison <- inner_join(
  tibble(tx = dtu$genes$tx, logFC_offsets = qlfDtu$table$logFC, p_offsets = qlfDtu$table$PVal
  tibble(tx = ds$genes$tx, logFC_diffSplice = as.vector(ds$coefficients), p_diffSplice = as.v
  by = "tx"
) |>
  mutate(
    offsets = p.adjust(p_offsets, "BH") < 0.05,
    diffSplice = p.adjust(p_diffSplice, "BH") < 0.05
  )

dtuComparison |>
  ggplot(aes(logFC_offsets, logFC_diffSplice)) +
  geom_point(alpha = 0.2, size = 0.5) +
  geom_abline(intercept = 0, slope = 1, col = "red") +
  theme_bw()
```




```
dtuComparison |>
  ggplot(aes(-log10(p_offsets), -log10(p_diffSplice))) +
  geom_point(alpha = 0.2, size = 0.5) +
  geom_abline(intercept = 0, slope = 1, col = "red") +
  theme_bw()
```




```
c(
  logFC = cor(dtuComparison$logFC_offsets, dtuComparison$logFC_diffSplice, method = "spearman
  pValue = cor(dtuComparison$p_offsets, dtuComparison$p_diffSplice, method = "spearman")
)
```

```
    logFC    pValue
0.9040687 0.7859980
```

```
table(offsets = dtuComparison$offsets, diffSplice = dtuComparison$diffSplice)
```

```
       diffSplice
offsets FALSE  TRUE
  FALSE 42096   505
  TRUE   1503   869
```

Both approaches agree well on the direction and size of the usage changes, but only moderately on the ranking. The offset model estimates a dispersion per transcript, while diffSplice shares one dispersion across all transcripts of a gene, so transcripts with a variability that differs from the rest of their gene are ranked differently.

Dedicated DTU methods also exist, e.g. DEXSeq, DRIMSeq and satuRn.

Stage-wise testing

DTU analyses test many transcripts per gene. Biologists are mostly interested in

1. which genes show DTU, and
2. which transcripts within these genes are responsible.

stageR (Van den Berge et al. 2017) controls the FDR at the gene level with a two-stage procedure:

1. **Screening stage**: test every gene for DTU (any transcript) and control the FDR over genes.
2. **Confirmation stage**: for the significant genes only, test the individual transcripts with a family-wise error rate correction within the gene.

We use the gene-level Simes test of `diffSplice` for screening and the transcript-level tests of the offset model for confirmation.

```
pScreen <- as.vector(ds$gene.simes.p.value)
names(pScreen) <- ds$gene.genes$gene
pConfirmation <- matrix(qlfDtu$table$PValue, ncol = 1,
                        dimnames = list(dtu$genes$tx, "transcript"))
tx2gene <- data.frame(transcript = dtu$genes$tx, gene = dtu$genes$gene)

stageRObj <- stageRTx(pScreen = pScreen, pConfirmation = pConfirmation, tx2gene = tx2gene)
stageRObj <- stageWiseAdjustment(stageRObj, method = "dtu", alpha = 0.05)
padj <- getAdjustedPValues(stageRObj, order = TRUE, onlySignificantGenes = TRUE)
head(padj)
```

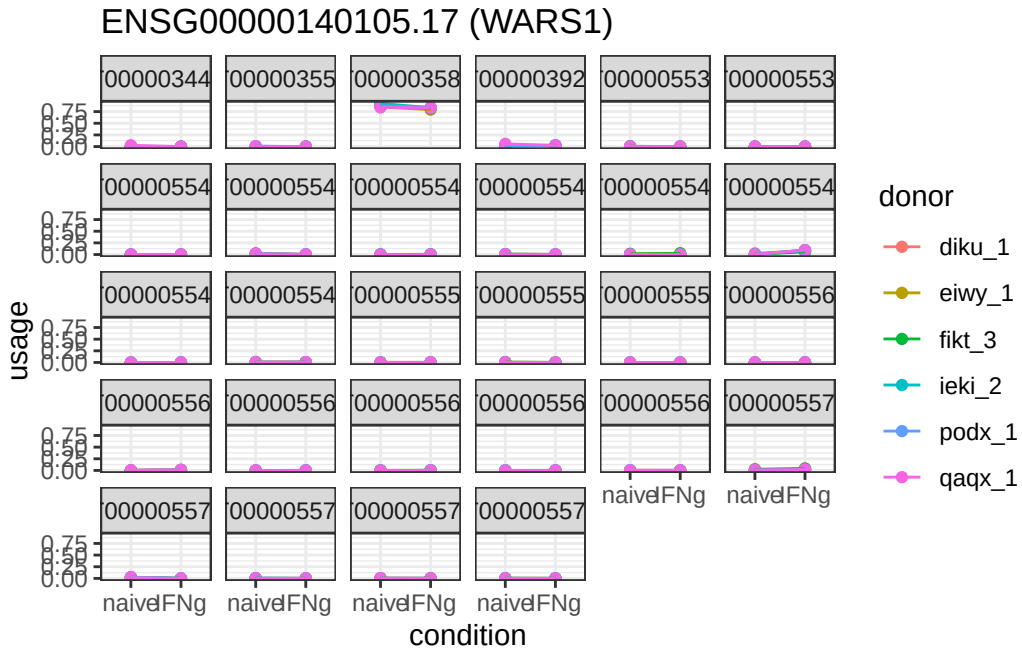
```
              geneID              txID         gene   transcript
1 ENSG00000140105.17 ENST00000344102.9 1.230994e-21 2.780907e-05
2 ENSG00000140105.17 ENST00000554772.5 1.230994e-21 6.063529e-05
3 ENSG00000140105.17 ENST00000554331.5 1.230994e-21 2.328664e-04
4 ENSG00000140105.17 ENST00000557239.5 1.230994e-21 5.510484e-04
5 ENSG00000140105.17 ENST00000556504.5 1.230994e-21 1.063998e-02
6 ENSG00000140105.17 ENST00000557135.5 1.230994e-21 5.801945e-02
```

```
c(genes = length(unique(padj$geneID)), transcripts = sum(padj$transcript < 0.05))
```

```
      genes transcripts
        801        1006
```

We plot the usages of the transcripts of the top gene, *WARS1*, a gene that is strongly induced by interferon gamma:

```
topGene <- padj$geneID[1]
usage <- dtu$counts[dtu$genes$gene == topGene, ] /
  geneTotals[dtu$genes$gene == topGene, ]
usage |>
  as_tibble(rownames = "tx") |>
  pivot_longer(-tx, names_to = "names", values_to = "usage") |>
  left_join(macData, by = "names") |>
  ggplot(aes(condition, usage, group = donor, color = donor)) +
  geom_point() +
  geom_line() +
  facet_wrap(~ tx) +
  ggtitle(paste0(topGene, " (", mapIds(org.Hs.eg.db, sub("\\..*", "", topGene), "SYMBOL", "E
  theme_bw()
```



Summary

- GLMs are fitted with IRLS: iterated weighted least squares fits of a working response. Inference with Wald and likelihood ratio tests is asymptotic.

- Quasi-likelihood only specifies the mean and the variance. The QL dispersion is estimated from the deviance and leads to t- and F-tests.
- edgeR 4 combines a global NB dispersion with gene-specific QL dispersions with bias-adjusted degrees of freedom, moderated with empirical Bayes: the same strategy as limma and msqrob2.
- Mock comparisons on real null data show that the edgeR 4 QL F-tests are well calibrated, while the legacy LRT is liberal. Ignoring that samples of the same donor are not independent (pseudo-replication) leads to massive numbers of false positives.
- Normalisation methods (TMM, median of ratios) assume that most genes are not DE and provide offsets.
- Independent filtering is valid when the filter does not use the group labels.
- At the transcript level, read-to-transcript ambiguity adds variability. DTU can be analysed with the standard tools by replacing the library size offset with the gene total, and stage-wise testing controls the FDR at the gene level.

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.

Baldoni, Pedro L, Lizhong Chen, Mengbo Li, Yunshun Chen, and Gordon K Smyth. 2025. "Dividing Out Quantification Uncertainty Enables Assessment of Differential Transcript Usage with Limma and edgeR." *Nucleic Acids Research* 53 (22): gkaf1305. https://doi.org/10.1093/nar/gkaf1305.

Baldoni, Pedro L, Yunshun Chen, Soroor Hediyeh-zadeh, et al. 2024. "Dividing Out Quantification Uncertainty Allows Efficient Assessment of Differential Transcript Expression with edgeR." *Nucleic Acids Research* 52 (3): e13.

Bourgon, Richard, Robert Gentleman, and Wolfgang Huber. 2010. "Independent Filtering Increases Detection Power for High-Throughput Experiments." *Proceedings of the National Academy of Sciences* 107 (21): 9546–51.

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.

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.

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).

McCarthy, Davis J, Yunshun Chen, and Gordon K Smyth. 2012. "Differential Expression Analysis of Multifactor RNA-Seq Experiments with Respect to Biological Variation." *Nucleic Acids Research* 40 (10): 4288–97.

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

Segers, Alexandre, Jeroen Gilis, Mattias Van Heetvelde, Davide Risso, Elfride De Baere, and Lieven Clement. 2023. "Juggling Offsets Unlocks RNA-seq Tools for Fast and Scalable Differential Usage, Aberrant Splicing and Expression Retrieval." *bioRxiv*, 2023.06.29.547014.

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, Charlotte Soneson, Mark D Robinson, and Lieven Clement. 2017. "stageR: A General Stage-Wise Method for Controlling the Gene-Level False Discovery Rate in Differential Expression and Differential Transcript Usage." *Genome Biology* 18: 151.