RAPToR - Showcase

RAPToR 1.2.0

Romain Bulteau

March 2023

Contents

This vignette showcases different uses for RAPToR.

1 Improving Differential Expression analysis with age estimates

Including estimated age as a covariate in Differential Expression (DE) analysis can substantially reduce previously unexplained variation between samples. In this example, we show that even when chronological age of different time points is known, using estimated age results in better model fits and consequently better power to detect DE genes.

1.1 Data and strategy

Lehrbach et al. (2012) profiled *C. elegans* wild-type (`wt`) and *pash-1(mj100)* mutants (`mut`) at 4 time points (0, 6, 12, and 24 hours past the L4 stage), each in triplicate (Accession: E-MTAB-1333, `dslehrbach2012`).

Given the time-series experimental design, searching for Differentially Expresses (DE) genes between strains requires including development in the model. While true in any time series context, it is particularly important here as late-larval development of *C. elegans* is known for drastic changes in gene expression within short time frames (Snoek et al. (2014)).

Using identical DE models with either chronological age or RAPToR age estimates as covariates, we can determine which of the two is the best predictor and thus gives more power to detect differential expression.

Code to generate the `dslehrbach2012` object can be found at the end of this section.

```
library(RAPToR)
library(wormRef)

library(stats)
library(parallel)
library(limma)
```

1.2 Estimating sample age

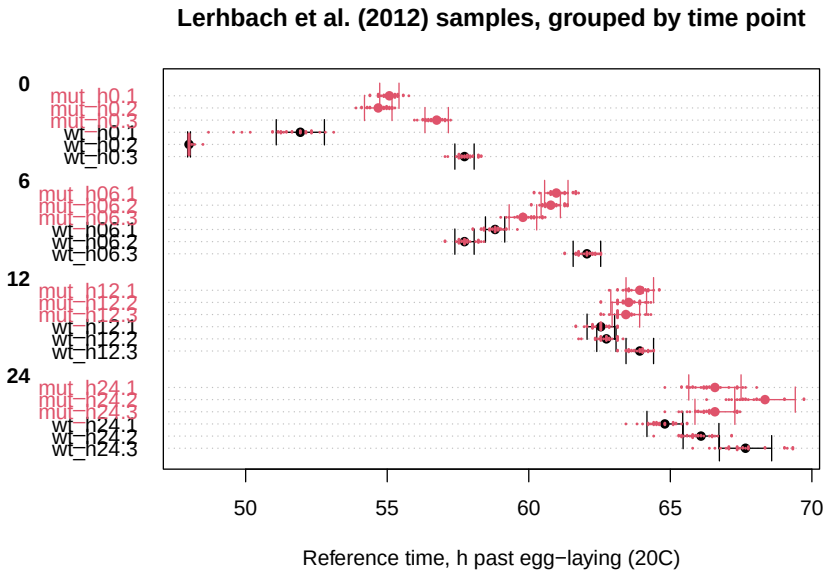
We start by applying a quantile-normalization and $log(X+1)$ transformation to the data.

```
dslehrbach2012$g <- limma::normalizeBetweenArrays(dslehrbach2012$g,
                                                  method = "quantile")
dslehrbach2012$g <- log1p(dslehrbach2012$g)
```

Next, we select a reference and stage the samples. Since sample (chronological) age ranges from the fourth larval molt (L4) to 24h past L4, we select `Cel_YA_2` from `wormRef` that covers this range.

```
# load reference
r_ya <- prepare_refdata("Cel_YA_2", "wormRef", 400)

# estimate sample age
ae_lerhbach <- ae(dslehrbach2012$g, r_ya)
#>                nb.genes
```

```
#> refdata            15710
#> samp               17441
#> intersect.genes    14952
#> Bootstrap set size is 4984
#> Performing age estimation...
#> Bootstrapping...
#>  Building gene subsets...
#>  Computing correlations...
#>  Performing age estimation...
#> Computing summary statistics...
```

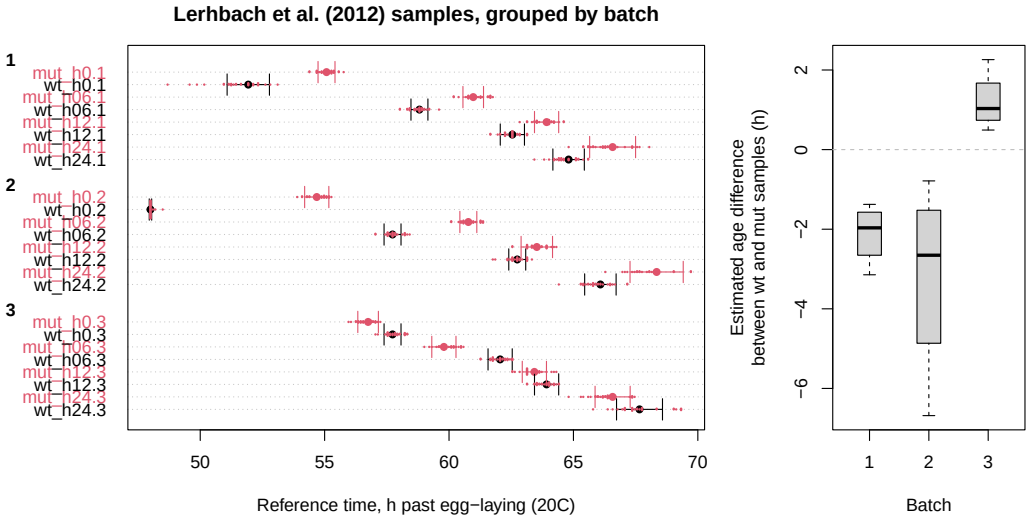


We can see quite a lot of variation in the sample timings, especially at the 0- and 6-hour time points.

Another curious effect can be noted, easier to see when grouping the samples by batch. In the first two replicates, mutants are systematically (and statistically significantly) older than WT, while the opposite effect is true for the third replicate.

```
# compute age difference between wt & mut at each time point
isWT <- dslehrbach2012$p$strain=="wt"
ae_dif <- ae_lerhbach$age.estimates[isWT,1] - ae_lerhbach$age.estimates[!isWT,1]

# test for effect significance with a linear model
batch <- dslehrbach2012$p$rep[isWT]
summary(lm(ae_dif~batch))
#>
#> Call:
#> lm(formula = ae_dif ~ batch)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.4889 -0.3440  0.0491  0.7862  2.4079
#>
#> Coefficients:
```

```
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -2.1130     0.7867  -2.686   0.0250 *
#> batch2       -1.0811     1.1126  -0.972   0.3566
#> batch3        3.3169     1.1126   2.981   0.0154 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.573 on 9 degrees of freedom
#> Multiple R-squared:  0.6535, Adjusted R-squared:  0.5765
#> F-statistic: 8.486 on 2 and 9 DF,  p-value: 0.008487
```

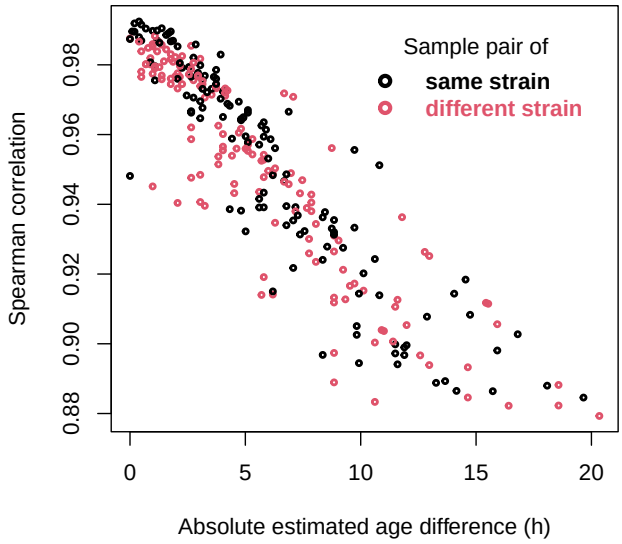


1.3 Impact of development on gene expression

Gene expression depends so strongly on development that correlation between samples is only predicted by their age difference, rather than by their genetic background.

The graph below shows correlation between all possible pairs of samples, with no clear effect of strain difference.

```
cc <- cor(dslehrbach2012$g, dslehrbach2012$g, method = "spearman")
```



1.4 Differential Expression analysis

We will use `limma` to perform a DE analysis with the following model:

$$X \sim strain \times \texttt{ns}(\texttt{age}, \texttt{df} = 2)$$

where age will either be chronological or estimated age, and the spline $\texttt{ns}()$ will be used to handle non-linear expression dynamics along time.

To find differential expression of genes along development or between strains translates to comparing the following nested models: (2) vs. (1) for development, (3) vs. (1) for strain.

$$Y = \beta_0 + \beta_1 I_{strain} + (\alpha_1 age_{sp1} + \alpha_2 age_{sp2}) + (\gamma_1 I_{strain} age_{sp1} + \gamma_2 I_{strain} age_{sp2})$$
$$Y = \beta_0 + \beta_1 I_{strain}$$
$$Y = \beta_0 + (\alpha_1 age_{sp1} + \alpha_2 age_{sp2})$$

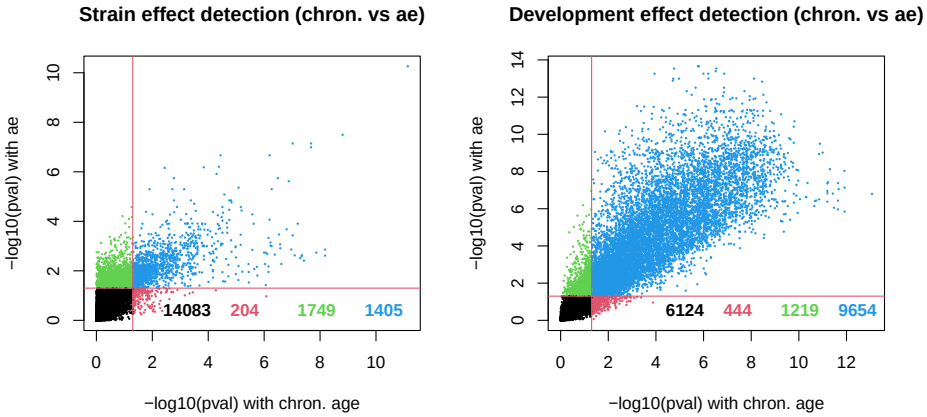
Genes will be considered DE with Benjamini-Holm adjusted p-values < 0.05 for the corresponding coefficients.

Note: for the purpose of this vignette, we don't consider whether genes are up- or down-regulated, but only if a significant effect is detected.

We now run the same DE analysis using either chronological age or RAPToR age estimates as predictor, and compare results to determine if there is an improvement. Code for `DGE()` can be found at the end of this section.

```
# format gdata as log2 for limma input
X <- log2(exp(dslehrbach2012$g))

# with chron. age
dge.ca <- DGE(X = X, strain = dslehrbach2012$p$strain,
              age = dslehrbach2012$p$tpastL4,
              name = "dge.ca", return.model = T)
```

```
# with RAPToR ae
dge.ae <- DGE(X = X, strain = dslehrbach2012$p$strain,
              age = ae_lerhbach$age.estimates[,1],
              name = "dge.ae", return.model = T)
```

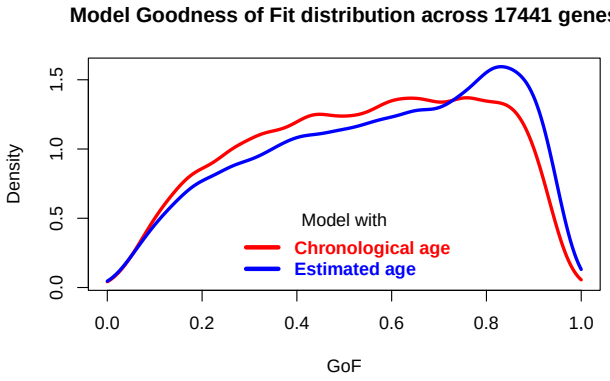


Estimated age shows a clear increase in performance over chronological age when comparing adjusted p-values of each gene for detection of an effect. Red bars in the plots above correspond to the 0.05 threshold for significance and color-coded gene counts for each quadrant are indicated in the bottom-right.

```
# compute nb. of DE genes for mutation
mut_dif.ca <- sum(dge.ca$tmut$adj.P.Val < 0.05)
mut_dif.ae <- sum(dge.ae$tmut$adj.P.Val < 0.05)

# compute nb. of DE genes for development
age_dif.ca <- sum(dge.ca$tage$adj.P.Val < 0.05)
age_dif.ae <- sum(dge.ae$tage$adj.P.Val < 0.05)

# percentage increase of mutation DE genes
100 * (mut_dif.ae - mut_dif.ca) / (mut_dif.ca) # mut pct increase
#> [1] 96.02237

# percentage increase of development DE genes
100 * (age_dif.ae - age_dif.ca) / (age_dif.ca) # age pct increase
#> [1] 7.674787
```

Indeed, we detect nearly twice as many DE genes for strain using RAPToR age estimates compared to chronological age, and around 8% more genes with development. Furthermore, the large proportion of genes changing with development $(57 - 62\%)$ compared to strain $(7 - 16\%)$ also shows how crucial it is to properly model developmental dynamics in DE analyses.

The increase in detected DE genes is partially explained by better model fits, shown below by the goodness-of-fit (GoF) distribution across genes. The goodness-of-fit (GoF) computed is an $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$ per gene.



1.5 Further support for the increase in performance

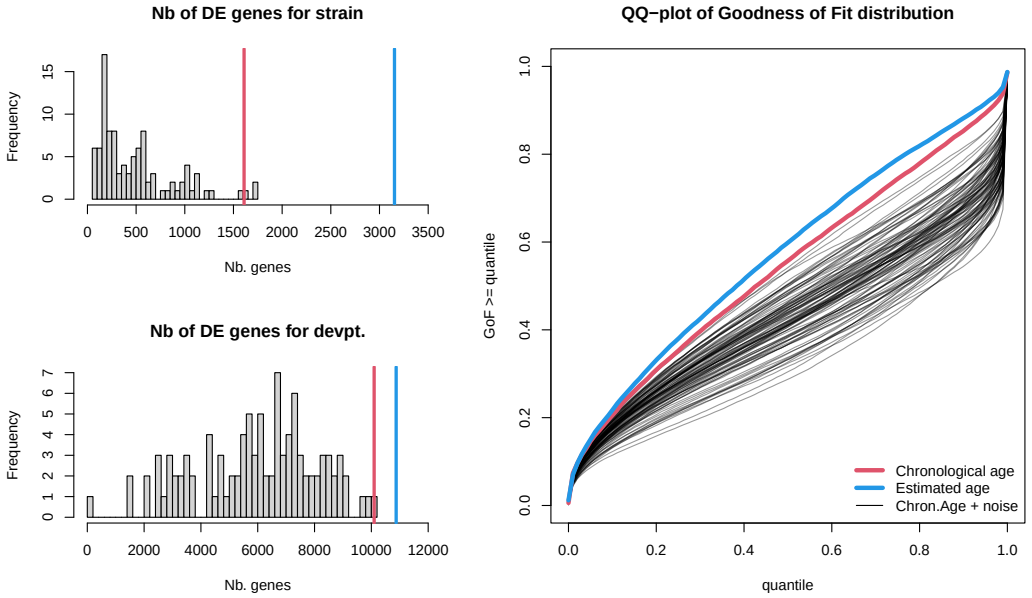
If the asynchronicity that RAPToR detects between samples is erroneous, then adding random noise around the chronological age values should yield similar results to using our age estimates in the model. To test this, we simulate age sets with added noise of similar distribution to the age differences observed between chronological and estimated age.

```
set.seed(10) # for reproducibility

age_diffs <- (dslehrbach2012$p$tpastL4 + 50) - ae_lerhbach$age.estimates[,1]
# Note : we add 50 to shift tpastL4 to the age values and avoid negative
#        tpastl4 values. This has no impact on the DE analysis.


# estimate density function of age_diffs
d_ad <- density(age_diffs)
# generate 100 age sets of with random age_diffs-like noise
n <- 100
rd_ages <- lapply(seq_len(n), function(i){
  (dslehrbach2012$p$tpastL4 + 50) +
    sample(x = d_ad$x, size = nrow(dslehrbach2012$p),
           prob = d_ad$y, replace = T)
})
```

As done above, we run identical DE models and compare results with the model goodness-of-fit (GoF) per gene and look at the number of DE genes found for strain and development (BH-adjusted p-value < 0.05).

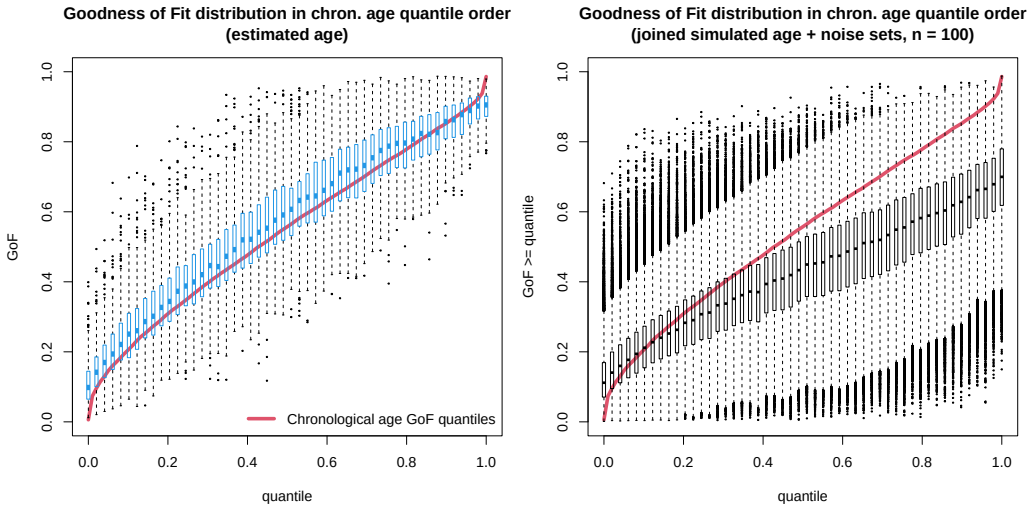
```
# setup cluster for parallelization
cl <- parallel::makeCluster(6, "FORK")

# do DGE on all age sets
rd_dges <- parLapply(cl, seq_len(n), function(i){
  cat("\r", i,"/",n)
  DGE(X, dslehrbach2012$p$strain, rd_ages[[i]], name = paste0("rd.",i))
})

stopCluster(cl)
gc()
```

```
# get quantiles of GoF for plotting
qts <- seq(0,1, length.out = 100)
quants <- lapply(seq_len(n), function(i){
  quantile(rd_dges[[i]]$gof, probs = qts)
})
quants <- do.call(rbind, quants)
```



We can see that using estimated age systematically leads to better model fits and increased detection of DE genes both for strain and development.

Gene GoF using estimated age and binned by chronological age GoF quantiles (below, left) also clearly shows that for the same genes, we tend to have better fits with estimated age, while that is not the case when adding noise.



1.6 Functions and code

1.6.1 Code to generate objects

```
data_folder <- "../inst/extdata/"

requireNamespace("wormRef", quietly = T)
requireNamespace("utils", quietly = T)

requireNamespace("biomaRt", quietly = T) # bioconductor
requireNamespace("limma", quietly = T)   # bioconductor
requireNamespace("affy", quietly = T)    # bioconductor
requireNamespace("gcrma", quietly = T)   # bioconductor
```

Note : set the `data_folder` variable to an existing path on your system where you want to store the objects.

To generate `dslehrbach2012`:

```
# get probe ids from the arrayexpress
probe_ids <- read.table(
  "https://www.ebi.ac.uk/arrayexpress/files/A-AFFY-60/A-AFFY-60.adf.txt",
  h=T, comment.char = "", sep = "\t", skip = 17, as.is = T)


# pheno data
p_url <- paste0("https://www.ebi.ac.uk/arrayexpress/files/",
                "E-MTAB-1333/E-MTAB-1333.sdrf.txt")
p <- read.table(p_url, h = T, sep = "\t", comment.char = "", as.is = T)

# geno data
g_url <- unique(p$Comment..ArrayExpress.FTP.file.)
g_dir <- paste0(data_folder, "dslehrbach2012_raw/")
g_file <- paste0(g_dir, "raw.zip")

if(!file.exists(g_file)){
  dir.create(g_dir)
  utils::download.file(g_url, destfile = g_file)
}
utils::unzip(g_file, exdir = g_dir)

g <- affy::ReadAffy(filenames = p$Array.Data.File,
                    celfile.path = g_dir, phenoData = p)
g <- affy::expresso(g, bg.correct = F, normalize = F,
                    pmcorrect.method = "pmonly", summary.method = "median")
g <- 2^exprs(g) # expresso log2s the data
colnames(g) <- p$Source.Name

# format ids
g <- RAPToR::format_ids(g, probe_ids, from = 1, to = 7)
g <- RAPToR::format_ids(g, wormRef::Cel_genes, from = 3, to = 1)
```

```
# filter relevant fields
p <- p[, c(1,22,24)]
colnames(p) <- c("title", "tpastL4", "strain")
p$strain <- factor(p$strain,
                   levels = c('wild type', 'pash-1(mj100)'),
                   labels = c('wt', 'strain'))
p$rep <- factor(gsub("\\w+_h\\d+\\.(\\d)","\\1", p$title))

dslehrbach2012 <- list(g = g, p = p)
save(dslehrbach2012,
     file = paste0(data_folder, "dslerhbach2012.RData"), compress = "xz")

# cleanup
file.remove(g_file)
unlink(g_dir, recursive = T)

rm(g, g_url, g_dir, g_file, p, p_url, probe_ids)
```

1.6.2 DE functions

Functions to compute predictions (`pred_lmFit()`) and Goodness-of-fit (`GoF()`) from `limma` models.

```
pred_lmFit <- function(Fit){
# Predictions from a limma model
  tcrossprod(Fit$coefficients, Fit$design)
}

GoF <- function(Fit, X){
# Compute Goodness of Fit
  pred <- pred_lmFit(Fit)
  res <- (X - pred)
  ss <- apply(X, 1, function(ro) sum((ro - mean(ro))^2))
  Rsq <- sapply(seq_len(nrow(X)), function(i){
    1 - sum(res[i,]^2)/ss[i]
  })
  return(Rsq)
}
```

Function to run DE analysis with `limma`, as described in *Differential Expression analysis* (`DGE()`).

```
DGE <- function(X, strain, age, df = 2, name = NULL, return.model = FALSE){
  require(splines)
  if(! length(strain) == ncol(X) | ! length(age) == ncol(X))
    stop("strain and age must be of length ncol(X).")

  # make pdat df
  pdat <- data.frame(strain = factor(strain),
                     age = as.numeric(age),
                     row.names = colnames(X))

  # build design matrix
```

```r
d <- model.matrix(~ 1 + ns(age, df = df) * strain, data = pdat)

# fix colnames
colnames(d) <- c("b0", paste0(rep("a", df), 1L:df),
                 "strainmut", paste0(rep("strainmut.a", df), 1L:df))

# build contrast matrices for mut and age tests
if(df == 2){
  cm.mut <- makeContrasts(mut = strainmut,
                          mut.i1 = strainmut.a1,
                          mut.i2 = strainmut.a2,
                          levels = d)
  cm.age <- makeContrasts(a1, a2,
                          a.i1 = strainmut.a1,
                          a.i2 = strainmut.a2,
                          levels = d)
}

if(df == 3){
  cm.mut <- makeContrasts(mut = strainmut,
                          mut.i1 = strainmut.a1,
                          mut.i2 = strainmut.a2,
                          mut.i3 = strainmut.a3,
                          levels = d)
  cm.age <- makeContrasts(a1, a2, a3,
                          a.i1 = strainmut.a1,
                          a.i2 = strainmut.a2,
                          a.i3 = strainmut.a3,
                          levels = d)
}

# fit model
m.0 <- lmFit(object = X, design = d)
# get GoF
gof <- GoF(Fit = m.0, X = X)

# find DE genes for mut
m.m <- contrasts.fit(m.0, contrasts = cm.mut)
m.m <- eBayes(m.m)

# find DE genes for age
m.a <- contrasts.fit(m.0, contrasts = cm.age)
m.a <- eBayes(m.a)

Tmut <- topTable(m.m, adjust.method = "BH",
                 number = Inf,
                 sort.by = "none")[, c("F", "P.Value", "adj.P.Val")]
Tage <- topTable(m.a, adjust.method = "BH",
                 number = Inf,
                 sort.by = "none")[, c("F", "P.Value", "adj.P.Val")]
```

```
  res <- list(gof  = gof,
              tmut = Tmut,
              tage = Tage,
              name = name)
  if(return.model)
    res$model = m.0

  rm(m.0, m.m, m.a, Tmut, Tage, d, X, pdat, gof)
  gc(verbose = F)
  return(res)
}
```

2 Staging samples cross-species

Your organism of interest may not be well-studied or have an abundance of reference time-series data available. However, RAPToR still works when using a close organism as a reference, thanks to the conserved nature of developmental processes across species (particularly in early development).

Indeed, samples can be staged cross-species using ortholog genes.

2.1 Data and strategy

Li et al. (2014) defined a set of orthologs between *D. melanogaster* and *C. elegans* (hereafter, `glist`), which we will use to stage *C. elegans* single embryos on a *D. melanogaster* reference (of note, ensembl ortholog sets also work).

The *C. elegans* time-series of single-embryos was profiled and published by Levin et al. (2016) (Accession : GSE60755, `dslevin2016cel`). The *Drosophila melanogaster* embryonic development time-series is part of the modENCODE project and published by Graveley et al. (2011) (data downloaded from fruitfly.org, `dsgraveley2011`)

Code to generate `glist` (orthologs), `dslevin2016cel` (*C. elegans* data), and `dsgraveley2011` (*D. melanogaster* data) can be found at the end of this section.

```
library(RAPToR)
library(drosoRef)

library(limma)
library(stats)
```

2.2 Estimating the age of *C. elegans* embryos on a *D. melanogaster* reference

We start by applying a quantile-normalization and $log(X+1)$ transformation to the data.

```
dsgraveley2011$g <- limma::normalizeBetweenArrays(dsgraveley2011$g,
                                                  method = "quantile")
dsgraveley2011$g <- log1p(dsgraveley2011$g)

dslevin2016cel$g <- limma::normalizeBetweenArrays(dslevin2016cel$g,
```

```
                                                    method = "quantile")
dslevin2016cel$g <- log1p(dslevin2016cel$g)
```

4 outliers in the *C. elegans* data are then removed from the analysis (see *About the outliers*).

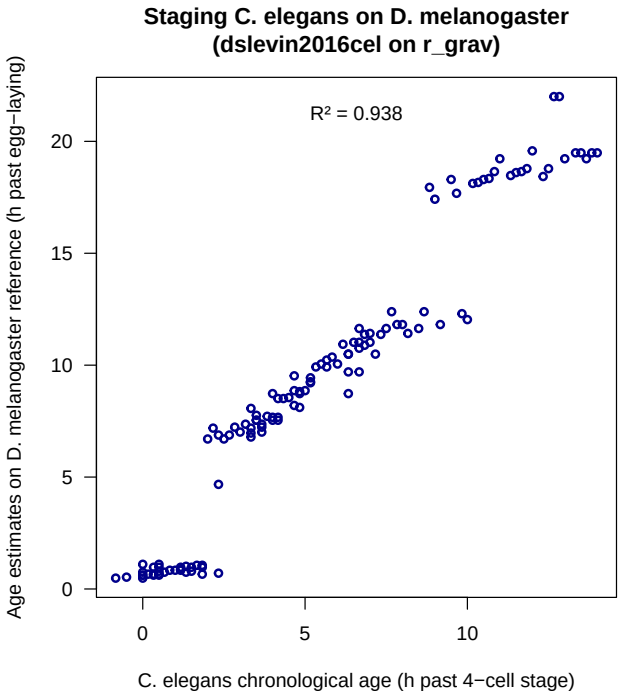
```
# outlier samples (see below)
rem <- c(sample_0001 = 53L, sample_0002 = 54L,
         sample_0003 = 58L, sample_0004 = 59L)
```

To stage samples we must build a reference from the *Drosophila* data, which happens to be the `Dme_embryo` reference of the `drosoRef` package. It can thus be directly loaded with `prepare_refdata()`.

```
# reference built from dsgraveley2011
r_grav <- prepare_refdata("Dme_embryo", "drosoRef", 500)
```

We then convert the *C. elegans* gene IDs to their *D. melanogaster* orthologs, and stage the *C. elegans* embryos with the Drosophila reference. In this case, many-to-one orthologs are averaged and one-to-many are assigned to the first ID match.

```
dslevin2016cel$g_dmel <- format_ids(dslevin2016cel$g, glist,
                                    from = "wb_id", to = "fb_id")
#> Kept 5714 out of 20168 - aggregated into 4214

ae_cel_on_dmel <- ae(dslevin2016cel$g_dmel, r_grav)
#>                 nb.genes
#> refdata            12408
#> samp                4214
#> intersect.genes     3194
#> Bootstrap set size is 1065
#> Performing age estimation...
#> Bootstrapping...
#>  Building gene subsets...
#>  Computing correlations...
#>  Performing age estimation...
#> Computing summary statistics...
```

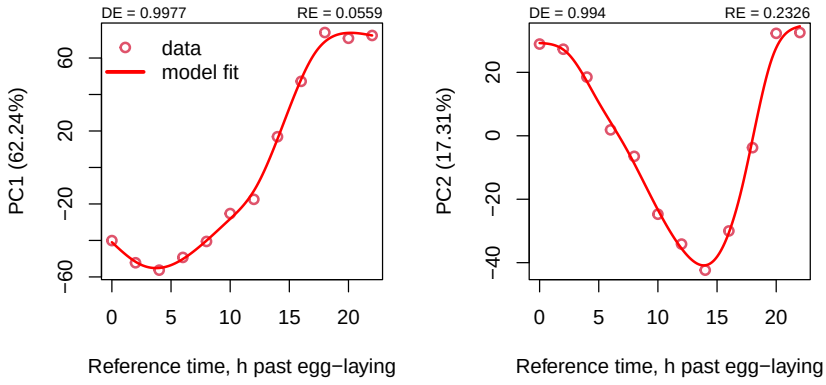


The gaps we see in the staging results are likely at timings where there are incompatible expression dynamics between the two species.

2.3 Adjusting the reference

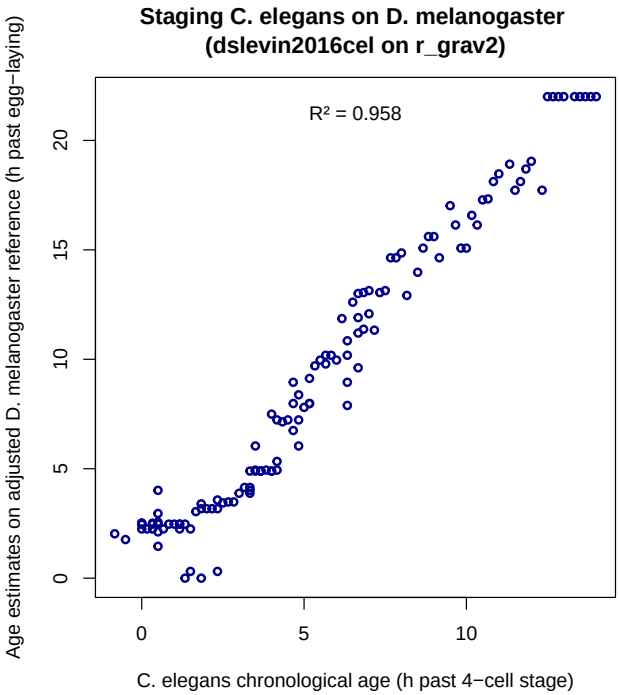
By re-building the *Drosophila* reference on the first 2 components which are broad or monotonic, we can keep only these expression dynamics of development in the reference. This can improve staging because it filters out the incompatible dynamics.

```
# build same model, but restrict interpolation to 2 components instead of 8
m_grav2 <- ge_im(X = dsgraveley2011$g, p = dsgraveley2011$p,
                 formula = "X ~ s(age, bs = 'cr')", nc = 2)

# make adjusted reference object
r_grav2 <- make_ref(m_grav2, n.inter = 500,
                    t.unit = "h past egg-laying",
                    metadata = list("organism"="D. melanogaster"))
```



We then stage the *C. elegans* embryos on this adjusted reference.

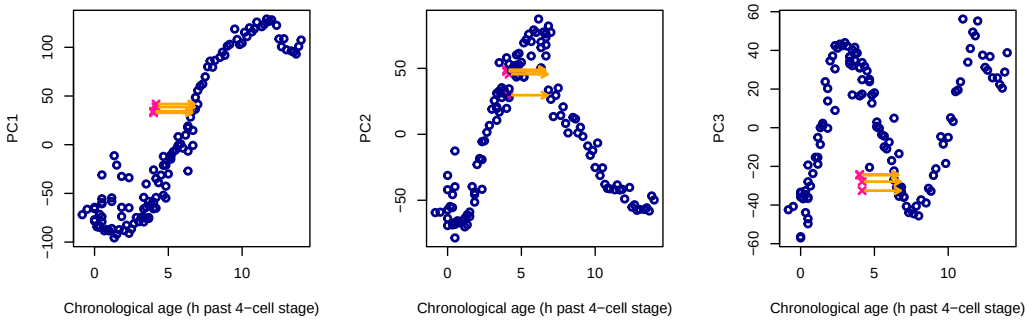
```
ae_cel_on_dmel2 <- ae(dslevin2016cel$g_dmel, r_grav2)
#>                nb.genes
#> refdata           12300
#> samp               4214
#> intersect.genes    3143
#> Bootstrap set size is 1048
#> Performing age estimation...
#> Bootstrapping...
#>  Building gene subsets...
#>  Computing correlations...
#>  Performing age estimation...
#> Computing summary statistics...
```



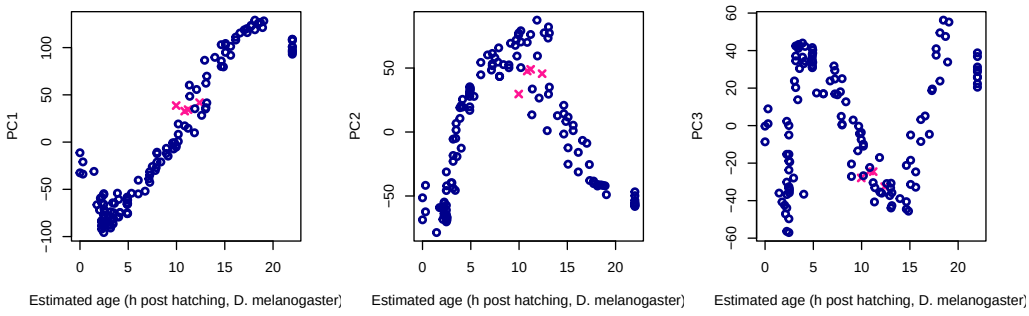
2.4 About the outliers

We removed 4 outlier samples from the analyses above because they have an erroneous noted age. This is evidenced by their position on principal components, where they appear to be around 2 hours "older" than their specified age.

```
pca_cel <- stats::prcomp(t(dslevin2016cel$g), rank = 10,
                         center = TRUE, scale = FALSE)
```



This is even further confirmed by their estimated age on the *Drosophila* reference, which places them as expected.



2.5 Code to generate objects

Required packages and variables:

```
data_folder <- "../inst/extdata/"

requireNamespace("wormRef", quietly = T)
requireNamespace("utils", quietly = T)
requireNamespace("GEOquery", quietly = T) # bioconductor
requireNamespace("Biobase", quietly = T)  # bioconductor
requireNamespace("biomaRt", quietly = T)  # bioconductor
```

Note : set the `data_folder` variable to an existing path on your system where you want to store the objects.

```
raw2tpm <- function(rawcounts, genelengths){
  if(nrow(rawcounts) != length(genelengths))
    stop("genelengths must match nrow(rawcounts).")
  x <- rawcounts/genelengths
  return(t( t(x) * 1e6 / colSums(x) ))
}

fpkm2tpm <- function(fpkm){
  return(exp(log(fpkm) - log(colSums(fpkm)) + log(1e6)))
}
```

To build `glist`, ortholog genes between *C. elegans* and *D. melanogaster* from Li et al. (2014) Supplementary Table 1:

```
tmp_file <- paste0(data_folder, "dmel_cel_orth.zip")
tmp_fold <- paste0(data_folder, "dmel_cel_orth/")
f_url <- paste0("https://genome.cshlp.org/content/suppl/2014/05/15/",
                "gr.170100.113.DC1/Supplemental_Files.zip")

utils::download.file(url = f_url, destfile = tmp_file)
utils::unzip(tmp_file, exdir = tmp_fold)

glist <- read.table(
  paste0(tmp_fold,
         "Supplementary\ files/TableS1\ fly-worm\ ortholog\ pairs.txt"),
  skip = 1, h=T, sep = "\t", as.is = T, quote = "\""
  )
colnames(glist) <- c("fb_id", "dmel_name", "cel_id", "cel_name")
glist$wb_id <- wormRef::Cel_genes[
  match(glist$cel_id, wormRef::Cel_genes$sequence_name), "wb_id"]

save(glist, file = paste0(data_folder, "sc2_glist.RData"), compress = "xz")

# cleanup
file.remove(tmp_file)
unlink(tmp_fold, recursive = T)
rm(tmp_file, tmp_fold, f_url)
```

To build `dsgraveley2011`, first get drosophila genes from ensembl:

```
mart <- biomaRt::useMart("ensembl", dataset = "dmelanogaster_gene_ensembl")
droso_genes <- biomaRt::getBM(attributes = c("ensembl_gene_id",
                                             "ensembl_transcript_id",
                                             "external_gene_name",
                                             "transcript_end",
                                             "transcript_start"),
                              mart = mart)
droso_genes$transcript_length <-
  droso_genes$transcript_end - droso_genes$transcript_start
droso_genes <- droso_genes[,c(1:3,6)]
colnames(droso_genes)[1:3] <- c("fb_id", "transcript_id", "gene_name")

rm(mart)
```

Then, download *D. melanogaster* data from Graveley et al. (2011) :

```
g_url_dsgraveley2011 <- paste0(
  "ftp://ftp.fruitfly.org/pub/download/modencode_expression_scores/",
  "Celniker_Drosophila_Annotation_20120616_1428_allsamps_",
  "MEAN_gene_expression.csv.gz")
g_file_dsgraveley2011 <- paste0(data_folder, "dsgraveley2011.csv.gz")
utils::download.file(g_url_dsgraveley2011, destfile = g_file_dsgraveley2011)


X_dsgraveley2011 <- read.table(gzfile(g_file_dsgraveley2011),
                               sep = ',', row.names = 1, h = T)
```

```
# convert gene ids to FBgn
X_dsgraveley2011 <- RAPToR::format_ids(X_dsgraveley2011, droso_genes,
                                       from = "gene_name", to = "fb_id")

# select embryo time series samples
X_dsgraveley2011 <- X_dsgraveley2011[,1:12]

P_dsgraveley2011 <- data.frame(
  sname = colnames(X_dsgraveley2011),
  age = as.numeric(gsub("em(\\d+)\\.\\d+hr", "\\1",
                        colnames(X_dsgraveley2011))),
  stringsAsFactors = FALSE)

dsgraveley2011 <- list(g = X_dsgraveley2011, p = P_dsgraveley2011)

save(dsgraveley2011,
     file = paste0(data_folder, "dsgraveley2011.RData"), compress = "xz")

# cleanup
file.remove(g_file_dsgraveley2011)
rm(g_url_dsgraveley2011, g_file_dsgraveley2011,
   X_dsgraveley2011, P_dsgraveley2011)
```

To build `dslevin2016cel`, *C. elegans* single-embryo data from Levin et al. (2016)

```
geo_dslevin2016cel <- "GSE60755"

g_url_dslevin2016cel <- GEOquery::getGEOSuppFiles(geo_dslevin2016cel,
                                                  makeDirectory = FALSE,
                                                  fetch_files = FALSE)
g_file_dslevin2016cel <- paste0(data_folder, "dslevin2016cel.txt.gz")
utils::download.file(url = as.character(g_url_dslevin2016cel$url[1]),
                     destfile = g_file_dslevin2016cel)

X_dslevin2016cel <- read.table(gzfile(g_file_dslevin2016cel),
                               h = T, sep = '\t', as.is = T, row.names = 1,
                               comment.char = "")

# filter poor quality samples
cm_dslevin2016cel <- RAPToR::cor.gene_expr(X_dslevin2016cel, X_dslevin2016cel)
f_dslevin2016cel <- which(0.67 > apply(cm_dslevin2016cel, 1,
                                       quantile, probs = .99))
X_dslevin2016cel <- X_dslevin2016cel[, -f_dslevin2016cel]


# convert to tpm & FBgn
X_dslevin2016cel <- X_dslevin2016cel[
  rownames(X_dslevin2016cel)%in%wormRef::Cel_genes$sequence_name,]
X_dslevin2016cel <- raw2tpm(
  rawcounts = X_dslevin2016cel,
  genelengths = wormRef::Cel_genes$transcript_length[
    match(rownames(X_dslevin2016cel), wormRef::Cel_genes$sequence_name)]
```

```
  )
X_dslevin2016cel <- RAPToR::format_ids(X_dslevin2016cel, wormRef::Cel_genes,
                                       from = "sequence_name", to = "wb_id")


# pheno data
P_dslevin2016cel <- Biobase::pData(GEOquery::getGEO(geo_dslevin2016cel,
                                                    getGPL = F)[[1]])

# filter relevant fields/samples
P_dslevin2016cel <- P_dslevin2016cel[
  , c("title", "geo_accession", "time point (minutes after 4-cell):ch1")]
colnames(P_dslevin2016cel)[3] <- "time"
P_dslevin2016cel$title <- as.character(P_dslevin2016cel$title)

P_dslevin2016cel <- P_dslevin2016cel[
  P_dslevin2016cel$title %in% colnames(X_dslevin2016cel),]

# formatting
P_dslevin2016cel$age <- as.numeric(P_dslevin2016cel$time) / 60
P_dslevin2016cel <- P_dslevin2016cel[order(P_dslevin2016cel$age),]
X_dslevin2016cel <- X_dslevin2016cel[, P_dslevin2016cel$title]

P_dslevin2016cel$title <- gsub('Metazome_CE_timecourse_', '',
                               P_dslevin2016cel$title)
colnames(X_dslevin2016cel) <- P_dslevin2016cel$title

# remove extra outlier
X_dslevin2016cel <- X_dslevin2016cel[,-127]
P_dslevin2016cel <- P_dslevin2016cel[-127,]

dslevin2016cel <- list(g = X_dslevin2016cel, p = P_dslevin2016cel)


save(dslevin2016cel,
     file = paste0(data_folder, "dslevin2016cel.RData"), compress = "xz")

# cleanup
file.remove(g_file_dslevin2016cel)
rm(geo_dslevin2016cel, g_url_dslevin2016cel, g_file_dslevin2016cel,
   f_dslevin2016cel, cm_dslevin2016cel, X_dslevin2016cel, P_dslevin2016cel)
```

3 Capturing tissue-specific development

In some species, tissues can develop at different rates, and these rates can vary between individuals. In *C. elegans*, Perez et al. (2017) have shown such developmental heterochrony between soma and germline tissues.

By restricting the genes RAPToR uses for staging to those associated with (or expressed in) specific tissues, it is possible to estimate development specific to these tissues.

3.1 Data and strategy

Rockman, Skrovanek, and Kruglyak (2010) published microarray profiling of 208 recombinant inbred lines of *C. elegans* N2 and Hawaii (CB4856) strains (Accession: GSE23857, `dsrockman2010`). These 208 samples were described as "developmentally synchronized" in the original article. However, Francesconi and Lehner (2014) later showed there is significant developmental spread between samples, spanning around 20 hours of 20°C late-larval development, essentially making this dataset a very high-resolution time course.

We will start by staging samples with all available genes to get their "***global age***".

Then, we stage samples a second time, but restricting the input to a `germline` set of 2554 genes by combining the `germline_intrinsic`, `germline_oogenesis_enriched`, and `germline_sperm_enriched` categories defined in Perez et al. (2017) (based on previous work by Reinke et al. (2004)). This will give us a ***germline age*** for the samples.

Staging the samples a third time, with a `soma` set of 2718 genes from the `osc` (molting) gene category defined in Hendriks et al. (2014), will give us the ***soma age***.

Finally, we evaluate the results of staging using principal (PCA) or independent (ICA) components, that summarize the expression dynamics of the data. For example, we can expect that components corresponding to particular tissues will have noticeably cleaner dynamics along their respective age estimates (*e.g.* oscillatory molting dynamics would be less noisy with soma age than germline age).

Code to generate `dsrockman2010` (RIL expression data) and `gsubset` (germline/soma gene sets) can be found at the end of this section

```
library(RAPToR)
library(wormRef)

library(ica)
library(stats)
library(limma)

# for plotting
library(vioplot)
```

3.2 Estimating sample global age

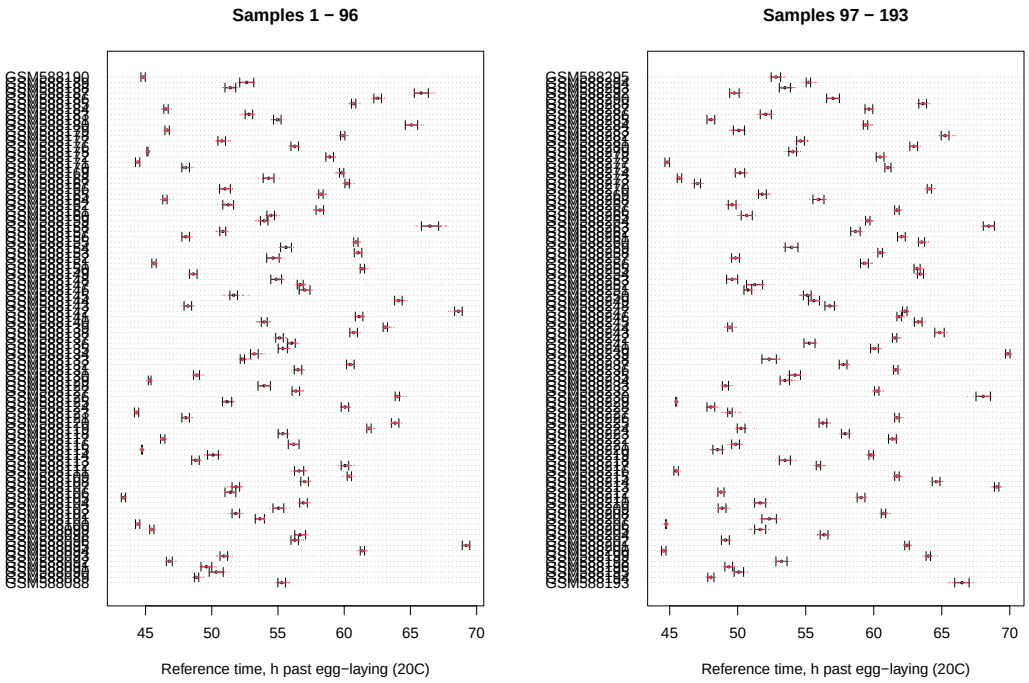
We start by applying a quantile-normalization and $log(X+1)$ transformation to the data.

```
dsrockman2010$g <- limma::normalizeBetweenArrays(dsrockman2010$g,
                                                 method = "quantile")
dsrockman2010$g <- log1p(dsrockman2010$g)
```

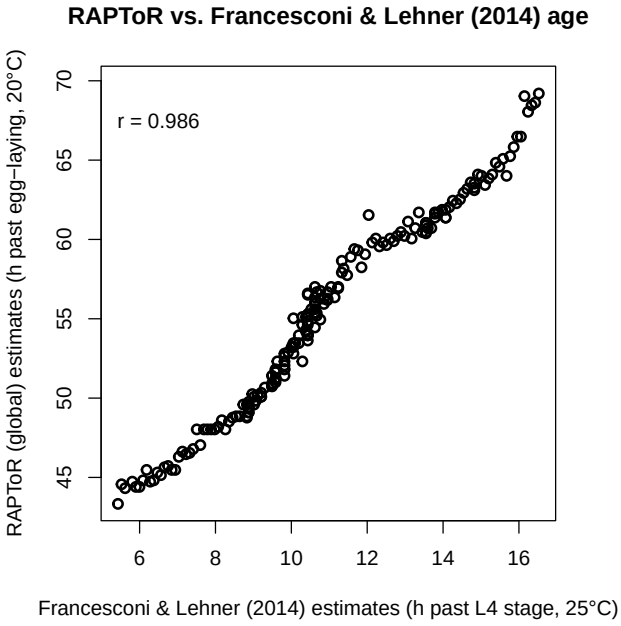
Then, we select an appropriate *C. elegans* reference (larval to young-adult), and stage the samples.

```
r_ya <- prepare_refdata("Cel_YA_1", "wormRef", n.inter = 400)

ae_dsrockman2010 <- ae(dsrockman2010$g, r_ya)
#>                 nb.genes
#> refdata            16327
#> samp               14440
#> intersect.genes    13288
```

```
#> Bootstrap set size is 4429
#> Performing age estimation...
#> Bootstrapping...
#>  Building gene subsets...
#>  Computing correlations...
#>  Performing age estimation...
#> Computing summary statistics...
```



Our estimates confirm the wide developmental spread and match the previously inferred ages by Francesconi and Lehner (2014).



3.3 Caracterization of expression dynamics

Expression dynamics in the data can be observed through ICA components, and enrichment in component gene loadings (or contributions) allows us to associate them to specific processes.

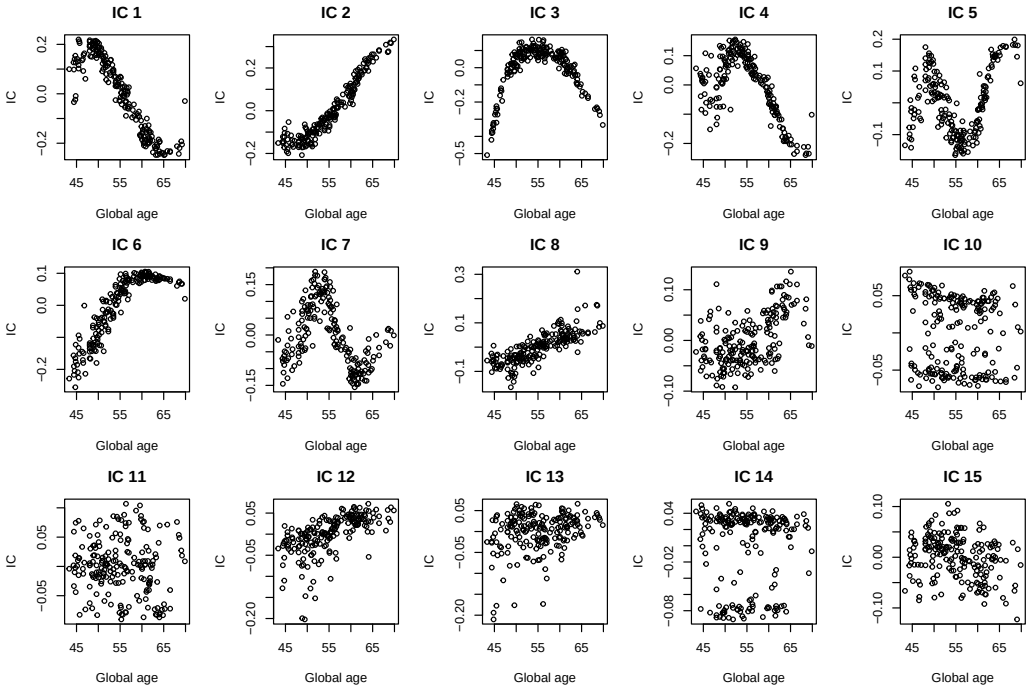
We find out how many independent components (IC) to extract from how many principal components are needed to explain 90% of the variance in the data.

```
pca_rock <- summary(stats::prcomp(t(dsrockman2010$g),
                                  rank = 30, center = TRUE, scale = FALSE))
nc <- sum(pca_rock$importance["Cumulative Proportion",] < .90) + 1
nc
#> [1] 48
```

Then, we perform an ICA extracting the 48 components.

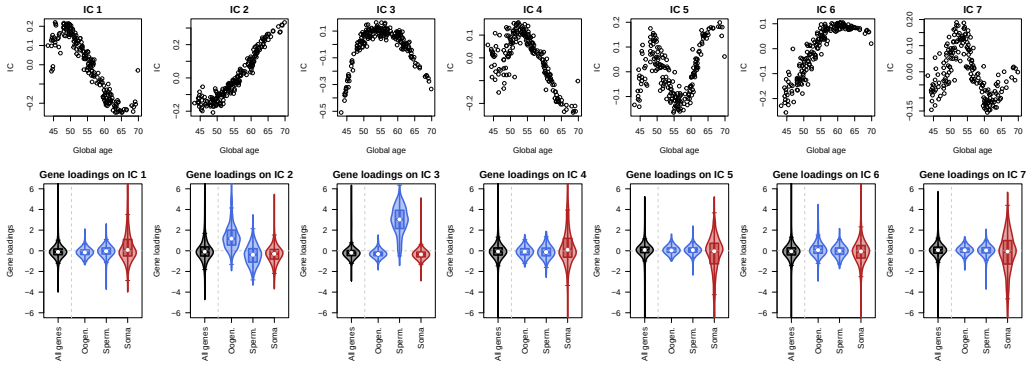
```
ica_rock <- ica::icafast(t(scale(t(dsrockman2010$g), center = T, scale = F)),
                         nc = nc)
```

We can plot the first few independent components along global estimated age of the samples :



It seems that past IC7, components don't have a definite link to developmental processes, which is expected given the (intended) genetic background heterogeneity in the samples. We can have a closer look at components clearly capturing development, and the enrichment of their loadings for the gene sets of interest.

```
dev_comps <- 1:7
```



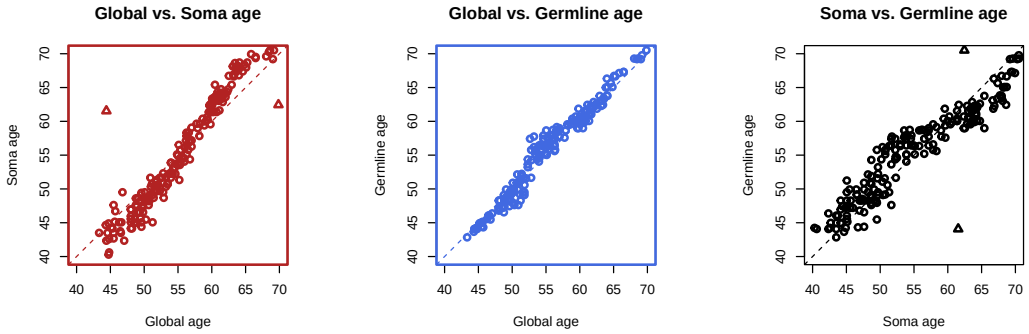
From the gene loadings above, we can establish that

- component IC2 is linked to oogenesis
- component IC3 is clearly associated with spermatogenesis
- components IC1, IC4, IC5, and IC7 have good contribution from the soma geneset, and show oscillatory dynamics.

3.4 Estimating tissue-specific age

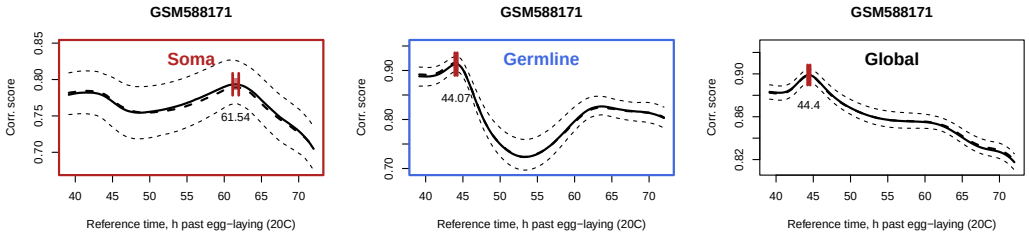
Now, we stage the samples using only germline or soma gene subsets.

```
ae_soma <- ae(
  dsrockman2010$g[rownames(dsrockman2010$g) %in% gsubset$soma,],
  r_ya)
#>                 nb.genes
#> refdata            16327
#> samp                2005
#> intersect.genes     1815
#> Bootstrap set size is 605
#> Performing age estimation...
#> Bootstrapping...
#>  Building gene subsets...
#>  Computing correlations...
#>  Performing age estimation...
#> Computing summary statistics...

ae_germline <- ae(
  dsrockman2010$g[rownames(dsrockman2010$g) %in% gsubset$germline,],
  r_ya)
#>                 nb.genes
#> refdata            16327
#> samp                1856
#> intersect.genes     1666
#> Bootstrap set size is 555
#> Performing age estimation...
#> Bootstrapping...
#>  Building gene subsets...
#>  Computing correlations...
#>  Performing age estimation...
#> Computing summary statistics...
```



We notice the soma estimates of two samples (marked with △) are quite off from their global or germline age. This caused by the combination of both the fact we used a small gene set for inferring age, and that very similar expression profiles can occur at different times in oscillatory profiles (which is the case for molting genes).

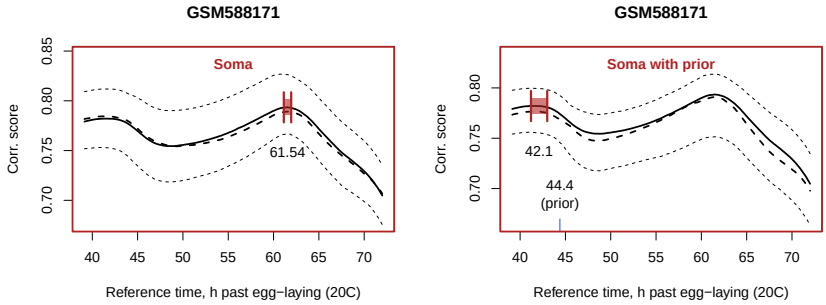
If we look at the global, soma, or germline correlation profiles of one of these samples, we can see 2 peaks of similar correlation for the soma, which is not the case for germline or global correlation.



This is the perfect situation to use a prior for staging. We can input the global age as a prior so that the correct peak of the soma correlation profile is favored in the erroneous samples.

```
ae_soma_prior <- ae(
  dsrockman2010$g[rownames(dsrockman2010$g)%in%gsubset$soma,],
  r_ya,
  prior = ae_dsrockman2010$age.estimates[,1], # gaussian prior values (mean)
  prior.params = 10                           # gaussian prior sd
  )
#>                 nb.genes
#> refdata            16327
#> samp                2005
#> intersect.genes     1815
#> Bootstrap set size is 605
#> Performing age estimation...
#> Bootstrapping...
#>  Building gene subsets...
#>  Computing correlations...
#>  Performing age estimation...
#> Computing summary statistics...
```

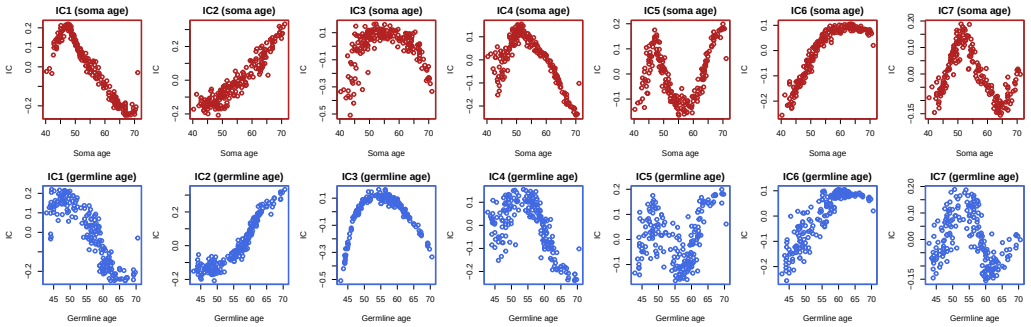
Indeed, the estimate is now shifted to the first (correct) peak. Note that the correlation profile itself (central black line) is unchanged (dotted lines can be different as they are derived from bootstrap estimates with random gene sets).



At the same time, all the other estimates are essentially unchanged.

```
# 80 & 141 are the offset samples
mean((ae_soma$age.estimates[,1] - ae_soma_prior$age.estimates[,1])[-c(80,141)])
#> [1] -0.014664
```

Now, we can observe the expression dynamics in the data along tissue-specific estimates.



As expected, components IC1, IC4, IC5, and IC7 that we previously associated with soma are much cleaner along the soma age, but very noisy along germline age.

At the same time, germline-associated components IC2 and IC3 appear quite noisy along soma age, but are very clean along germline age estimates.

This effect is the result of soma-germline heterochrony between the samples.

Note: In this vignette, we use `Cel_YA_1` to stage the samples, which is different from the RAPToR article (Bulteau and Francesconi (2022)) which uses `Cel_larv_YA`. The article shows a combination of soma-germline heterochrony *between* the reference and the samples, as well as among the samples. We shortened the analysis for the vignette to only show heterochrony among RILs (*e.g.* the ICA does not combine reference and samples, so components are different; however, enrichment of soma or germline genes to specific components/dynamics is still clear).

3.5 Code to generate objects

```
data_folder <- "../inst/extdata/"

requireNamespace("wormRef", quietly = T)
requireNamespace("utils", quietly = T)
requireNamespace("readxl", quietly = T)

requireNamespace("GEOquery", quietly = T) # bioconductor
requireNamespace("Biobase", quietly = T)  # bioconductor
requireNamespace("limma", quietly = T)    # bioconductor
```

Note : set the `data_folder` variable to an existing path on your system where you want to store the objects.

To generate `dsrockman2010`:

```
geo_dsrockman2010 <- "GSE23857"
geo_dsrockman2010 <- GEOquery::getGEO(geo_dsrockman2010, GSEMatrix = F)

# get pdata
P_dsrockman2010 <- do.call(
  rbind,
  lapply(GEOquery::GSMList(geo_dsrockman2010), function(go){
    unlist(GEOquery::Meta(go)[
      c("geo_accession", "characteristics_ch1", "characteristics_ch2",
        "label_ch1", "label_ch2")]
```

```
    )
  })
  )

P_dsrockman2010 <- as.data.frame(P_dsrockman2010, stringsAsFactors = F)

# get RG data from microarray
RG_dsrockman2010 <- list(
  R = do.call(cbind, lapply(GEOquery::GSMList(geo_dsrockman2010), function(go){
    GEOquery::Table(go)[, "R_MEAN_SIGNAL"]
  })),
  G = do.call(cbind, lapply(GEOquery::GSMList(geo_dsrockman2010), function(go){
    GEOquery::Table(go)[, "G_MEAN_SIGNAL"]
  }))
)

# normalize within channels
RG_dsrockman2010 <- limma::normalizeWithinArrays(RG_dsrockman2010,
                                                 method = "loess")
RG_dsrockman2010 <- limma::RG.MA(RG_dsrockman2010) # convert back from MA to RG

# get only the RIALs (not the mixed stage controls)
X_dsrockman2010 <- RG_dsrockman2010$R
X_dsrockman2010[, P_dsrockman2010$label_ch1 == "Cy5"] <-
  RG_dsrockman2010$G[, P_dsrockman2010$label_ch1 == "Cy5"]

# format probe/gene ids
gpl <- GEOquery::getGEO(GEOquery::Meta(
  GEOquery::GSMList(geo_dsrockman2010)[[1]]
  )$platform_id)
gpl <- GEOquery::Table(gpl)
gpl <- gpl[as.character(gpl$ID) %in% as.character(GEOquery::Table(
  GEOquery::GSMList(geo_dsrockman2010)[[1]]
  )$ID_REF), ]

sel <- gpl$ORF%in%wormRef::Cel_genes$sequence_name
gpl <- gpl[sel,]
X_dsrockman2010 <- X_dsrockman2010[sel,]


# filter bad quality samples
cm_dsrockman2010 <- cor(log1p(X_dsrockman2010), method = 'spearman')
f_dsrockman2010 <- which(0.95 > apply(cm_dsrockman2010, 1,
                                      quantile, probs = .95))

X_dsrockman2010 <- X_dsrockman2010[,-f_dsrockman2010]
P_dsrockman2010 <- P_dsrockman2010[-f_dsrockman2010,]

# format ids
rownames(X_dsrockman2010) <- gpl$ORF
X_dsrockman2010 <- RAPToR::format_ids(X_dsrockman2010, wormRef::Cel_genes,
```

```
                                    from = "sequence_name", to = "wb_id")


dsrockman2010 <- list(g = X_dsrockman2010, p = P_dsrockman2010)
save(dsrockman2010,
     file = paste0(data_folder, "dsrockman2010.RData"), compress = "xz")

rm(geo_dsrockman2010, gpl, sel,
   RG_dsrockman2010, X_dsrockman2010, P_dsrockman2010,
   cm_dsrockman2010, f_dsrockman2010)
```

To download the soma and germline gene sets (`gsubset`) :

```
library(readxl)
# germline sets from Perez et al. (2017)
germline_url <- paste0("https://static-content.springer.com/esm/",
                       "art%3A10.1038%2Fnature25012/MediaObjects/",
                       "41586_2017_BFnature25012_MOESM3_ESM.xlsx")
germline_file <- paste0(data_folder, "germline_gset.xlsx")
utils::download.file(url = germline_url, destfile = germline_file)

germline_set <- read_xlsx(germline_file, sheet = 3, na = "NA")[,c(1, 44:46)]
germline_set[is.na(germline_set)] <- FALSE
germline_set <- cbind(wb_id = germline_set[,1],
                      germline = apply(germline_set[, 2:4], 1,
                                       function(r) any(r)),
                      germline_set[, 2:4])

germline <- germline_set[germline_set$germline,1]
germline_intrinsic <- germline_set[germline_set$germline_intrinsic,1]
germline_oogenesis <- germline_set[germline_set$germline_oogenesis_enriched,1]
germline_sperm <- germline_set[germline_set$germline_sperm_enriched,1]

# soma set from Hendriks et al. (2014)
soma_url <- paste0("https://ars.els-cdn.com/content/image/",
                   "1-s2.0-S1097276513009039-mmc2.xlsx")
soma_file <- paste0(data_folder, "soma_gset.xlsx")
utils::download.file(url = soma_url, destfile = soma_file)

soma_set <- read_xlsx(soma_file, skip = 3, na = "NA")[,c(1, 4)]
soma_set$class <- factor(soma_set$class)

soma_set$soma <- soma_set$class == "osc"
soma_set <- soma_set[soma_set$soma, 1]

# save gene sets
gsubset <- list(germline = germline, soma = soma_set$`Gene WB ID`,
                germline_intrinsic = germline_intrinsic,
                germline_oogenesis = germline_oogenesis,
                germline_sperm = germline_sperm)

save(gsubset, file = paste0(data_folder, "sc3_gsubset.RData"), compress = "xz")
```

```
# cleanup
file.remove(germline_file)
file.remove(soma_file)
rm(germline_url, germline_file, germline_set,
   soma_url, soma_file, soma_set)
```

Francesconi and Lehner (2014) sample timings (`francesconi_time`):

```
# Copied from supp data of Francesconi & Lehner (2014)
francesconi_time <- data.frame(
  time = c(
  4.862660944, 4.957081545, 5.051502146, 5.145922747, 5.240343348, 5.334763948,
  5.429184549, 5.52360515, 5.618025751, 5.712446352, 5.806866953, 5.901287554,
  5.995708155, 6.090128755, 6.184549356, 6.278969957, 6.373390558, 6.467811159,
  6.56223176, 6.656652361, 6.751072961, 6.845493562, 6.939914163, 7.034334764,
  7.128755365, 7.223175966, 7.317596567, 7.412017167, 7.506437768, 7.600858369,
  7.69527897, 7.789699571, 7.884120172, 7.978540773, 8.072961373, 8.167381974,
  8.261802575, 8.356223176, 8.450643777, 8.545064378, 8.639484979, 8.733905579,
  8.82832618, 8.82832618, 8.82832618, 8.875536481, 8.875536481, 8.875536481,
  8.875536481, 8.875536481, 8.875536481, 8.875536481, 8.875536481, 8.875536481,
  8.969957082, 9.017167382, 9.017167382, 9.064377682, 9.064377682, 9.111587983,
  9.206008584, 9.206008584, 9.206008584, 9.300429185, 9.394849785, 9.489270386,
  9.489270386, 9.489270386, 9.489270386, 9.489270386, 9.583690987, 9.583690987,
  9.583690987, 9.583690987, 9.583690987, 9.583690987, 9.583690987, 9.583690987,
  9.630901288, 9.725321888, 9.819742489, 9.819742489, 9.819742489, 9.819742489,
  9.819742489, 9.819742489, 9.819742489, 9.91416309, 10.00858369, 10.05579399,
  10.05579399, 10.05579399, 10.05579399, 10.10300429, 10.19742489, 10.19742489,
  10.29184549, 10.29184549, 10.29184549, 10.38626609, 10.38626609, 10.38626609,
  10.43347639, 10.43347639, 10.43347639, 10.43347639, 10.43347639, 10.43347639,
  10.43347639, 10.43347639, 10.43347639, 10.43347639, 10.43347639, 10.43347639,
  10.527897, 10.6223176, 10.6223176, 10.6223176, 10.6223176, 10.6223176,
  10.6223176, 10.6223176, 10.6695279, 10.6695279, 10.6695279, 10.7639485,
  10.7639485, 10.7639485, 10.8583691, 10.8583691, 10.9527897, 10.9527897,
  10.9527897, 11.0472103, 11.1416309, 11.2360515, 11.2360515, 11.3304721,
  11.3304721, 11.3776824, 11.472103, 11.56652361, 11.66094421, 11.75536481,
  11.84978541, 11.94420601, 12.03862661, 12.13304721, 12.22746781, 12.32188841,
  12.41630901, 12.51072961, 12.60515021, 12.69957082, 12.79399142, 12.88841202,
  12.98283262, 13.07725322, 13.17167382, 13.26609442, 13.36051502, 13.45493562,
  13.54935622, 13.54935622, 13.54935622, 13.54935622, 13.54935622, 13.59656652,
  13.69098712, 13.78540773, 13.78540773, 13.78540773, 13.87982833, 13.97424893,
  14.06866953, 14.06866953, 14.06866953, 14.16309013, 14.25751073, 14.35193133,
  14.44635193, 14.54077253, 14.63519313, 14.72961373, 14.82403433, 14.82403433,
  14.82403433, 14.91845494, 14.96566524, 15.01287554, 15.10729614, 15.20171674,
  15.29613734, 15.39055794, 15.48497854, 15.57939914, 15.67381974, 15.76824034,
  15.86266094, 15.95708155, 16.05150215, 16.14592275, 16.24034335, 16.33476395,
  16.42918455, 16.52360515),
  geo_accession = c(
  "GSM588291", "GSM588174", "GSM588110", "GSM588097", "GSM588271", "GSM588203",
  "GSM588105", "GSM588200", "GSM588123", "GSM588122", "GSM588115", "GSM588100",
  "GSM588171", "GSM588190", "GSM588229", "GSM588206", "GSM588277", "GSM588129",
  "GSM588175", "GSM588151", "GSM588273", "GSM588216", "GSM588099", "GSM588117",
  "GSM588179", "GSM588164", "GSM588184", "GSM588092", "GSM588285", "GSM588272",
```

```
  "GSM588228", "GSM588121", "GSM588170", "GSM588194", "GSM588143", "GSM588149",
  "GSM588156", "GSM588220", "GSM588212", "GSM588089", "GSM588209", "GSM588253",
  "GSM588091", "GSM588113", "GSM588130", "GSM588202", "GSM588191", "GSM588244",
  "GSM588227", "GSM588197", "GSM588233", "GSM588292", "GSM588163", "GSM588196",
  "GSM588224", "GSM588283", "GSM588267", "GSM588257", "GSM588221", "GSM588274",
  "GSM588090", "GSM588114", "GSM588195", "GSM588265", "GSM588182", "GSM588093",
  "GSM588157", "GSM588251", "GSM588177", "GSM588188", "GSM588269", "GSM588145",
  "GSM588205", "GSM588162", "GSM588210", "GSM588166", "GSM588125", "GSM588252",
  "GSM588207", "GSM588173", "GSM588102", "GSM588286", "GSM588107", "GSM588238",
  "GSM588189", "GSM588106", "GSM588295", "GSM588192", "GSM588134", "GSM588183",
  "GSM588103", "GSM588198", "GSM588293", "GSM588218", "GSM588259", "GSM588234",
  "GSM588137", "GSM588152", "GSM588133", "GSM588250", "GSM588168", "GSM588235",
  "GSM588148", "GSM588279", "GSM588140", "GSM588241", "GSM588111", "GSM588231",
  "GSM588128", "GSM588131", "GSM588101", "GSM588088", "GSM588281", "GSM588159",
  "GSM588249", "GSM588290", "GSM588118", "GSM588154", "GSM588136", "GSM588268",
  "GSM588204", "GSM588160", "GSM588135", "GSM588098", "GSM588294", "GSM588225",
  "GSM588181", "GSM588248", "GSM588096", "GSM588217", "GSM588147", "GSM588176",
  "GSM588116", "GSM588146", "GSM588127", "GSM588104", "GSM588108", "GSM588262",
  "GSM588223", "GSM588161", "GSM588237", "GSM588172", "GSM588284", "GSM588256",
  "GSM588165", "GSM588211", "GSM588242", "GSM588169", "GSM588240", "GSM588264",
  "GSM588219", "GSM588287", "GSM588124", "GSM588178", "GSM588167", "GSM588258",
  "GSM588232", "GSM588141", "GSM588112", "GSM588208", "GSM588215", "GSM588132",
  "GSM588278", "GSM588275", "GSM588155", "GSM588153", "GSM588109", "GSM588185",
  "GSM588138", "GSM588094", "GSM588226", "GSM588236", "GSM588266", "GSM588119",
  "GSM588222", "GSM588246", "GSM588150", "GSM588261", "GSM588201", "GSM588247",
  "GSM588186", "GSM588280", "GSM588255", "GSM588288", "GSM588260", "GSM588139",
  "GSM588245", "GSM588270", "GSM588276", "GSM588199", "GSM588254", "GSM588120",
  "GSM588144", "GSM588243", "GSM588214", "GSM588180", "GSM588126", "GSM588282",
  "GSM588187", "GSM588158", "GSM588193", "GSM588213", "GSM588230", "GSM588263",
  "GSM588142", "GSM588095"),
  stringsAsFactors = F)
rownames(francesconi_time) <- francesconi_time$geo_accession
```

Back to top

References

Bulteau, Romain, and Mirko Francesconi. 2022. "Real Age Prediction from the Transcriptome with RAPToR." *Nature Methods*, 1–7.

Francesconi, Mirko, and Ben Lehner. 2014. "The Effects of Genetic Variation on Gene Expression Dynamics During Development." *Nature* 505 (7482): 208.

Graveley, Brenton R, Angela N Brooks, Joseph W Carlson, Michael O Duff, Jane M Landolin, Li Yang, Carlo G Artieri, et al. 2011. "The Developmental Transcriptome of Drosophila Melanogaster." *Nature* 471 (7339): 473.

Hendriks, Gert-Jan, Dimos Gaidatzis, Florian Aeschimann, and Helge Großhans. 2014. "Extensive Oscillatory Gene Expression During c. Elegans Larval Development." *Molecular Cell* 53 (3): 380–92.

Lehrbach, Nicolas J, Cecilia Castro, Kenneth J Murfitt, Cei Abreu-Goodger, Julian L Griffin, and Eric A Miska. 2012. "Post-Developmental microRNA Expression Is Required for Normal Physiology, and Regulates Aging in Parallel to Insulin/IGF-1 Signaling in c. Elegans." *Rna* 18 (12): 2220–35.

Levin, Michal, Leon Anavy, Alison G Cole, Eitan Winter, Natalia Mostov, Sally Khair, Naftalie Senderovich, et al. 2016. "The Mid-Developmental Transition and the Evolution of Animal Body Plans." *Nature* 531 (7596): 637.

Li, Jingyi Jessica, Haiyan Huang, Peter J Bickel, and Steven E Brenner. 2014. "Comparison of d. Melanogaster and c. Elegans Developmental Stages, Tissues, and Cells by modENCODE RNA-Seq Data." *Genome Research* 24 (7): 1086–1101.

Perez, Marcos Francisco, Mirko Francesconi, Cristina Hidalgo-Carcedo, and Ben Lehner. 2017. "Maternal Age Generates Phenotypic Variation in Caenorhabditis Elegans." *Nature* 552 (7683): 106.

Reinke, Valerie, Inigo San Gil, Samuel Ward, and Keith Kazmer. 2004. "Genome-Wide Germline-Enriched and Sex-Biased Expression Profiles in Caenorhabditis Elegans."

Rockman, Matthew V, Sonja S Skrovanek, and Leonid Kruglyak. 2010. "Selection at Linked Sites Shapes Heritable Phenotypic Variation in c. Elegans." *Science* 330 (6002): 372–76.

Snoek, L Basten, Mark G Sterken, Rita JM Volkers, Mirre Klatter, Kobus J Bosman, Roel PJ Bevers, Joost AG Riksen, Geert Smant, Andrew R Cossins, and Jan E Kammenga. 2014. "A Rapid and Massive Gene Expression Shift Marking Adolescent Transition in c. Elegans." *Scientific Reports* 4 (1): 1–5.

SessionInfo

```
sessionInfo()
```

```
#> R version 4.1.2 (2021-11-01)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 22.04.1 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
#> LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=C                  LC_COLLATE=en_US.UTF-8
#>  [5] LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] splines   parallel  stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] vioplot_0.4.0      zoo_1.8-11         sm_2.2-5.7.1       ica_1.0-3
#>  [5] drosoRef_0.2.0     limma_3.50.3       wormRef_0.5.0      RAPToR_1.2.0
#>  [9] ggpubr_0.6.0       ggplot2_3.4.0.9000 BiocStyle_2.22.0
#>
#> loaded via a namespace (and not attached):
#>  [1] Rcpp_1.0.10         lattice_0.20-45     tidyr_1.3.0
#>  [4] digest_0.6.31       utf8_1.2.3          R6_2.5.1
```

```
#>  [7] backports_1.4.1     evaluate_0.20       highr_0.10
#> [10] pillar_1.8.1        Rdpack_2.4          rlang_1.0.6
#> [13] rstudioapi_0.14     data.table_1.14.6   car_3.1-1
#> [16] jquerylib_0.1.4     magick_2.7.3        Matrix_1.5-3
#> [19] rmarkdown_2.20.1    labeling_0.4.2      stringr_1.5.0
#> [22] tinytex_0.44        munsell_0.5.0       broom_1.0.3
#> [25] compiler_4.1.2      xfun_0.37           pkgconfig_2.0.3
#> [28] mgcv_1.8-41         htmltools_0.5.4     tidyselect_1.2.0
#> [31] tibble_3.1.8        bookdown_0.32       codetools_0.2-18
#> [34] fansi_1.0.4         dplyr_1.1.0         withr_2.5.0
#> [37] rbibutils_2.2.13    grid_4.1.2          nlme_3.1-161
#> [40] jsonlite_1.8.4      gtable_0.3.1        lifecycle_1.0.3
#> [43] magrittr_2.0.3      scales_1.2.1        cli_3.6.0
#> [46] stringi_1.7.12      cachem_1.0.6        carData_3.0-5
#> [49] farver_2.1.1        ggsignif_0.6.4      pryr_0.1.6
#> [52] bslib_0.4.2         generics_0.1.3      vctrs_0.5.2
#> [55] tools_4.1.2         glue_1.6.2          beeswarm_0.4.0
#> [58] purrr_1.0.1         abind_1.4-5         fastmap_1.1.0
#> [61] yaml_2.3.7          colorspace_2.1-0    BiocManager_1.30.19
#> [64] rstatix_0.7.2       knitr_1.42          sass_0.4.5
```