DOTSeq: Differential ORF Translation

true

2025

Introduction

DOTSeq is a R package for identifying differentially translated open reading frames (ORFs) from paired Ribo- and RNAseq data. Unlike most existing tools that operate at gene level, DOTSeq performs analysis at the ORF level, enabling the detection of:

- Differential translation between conditions, and
- Occupancy shifts of ribosomes on ORFs within a single gene

DOTSeq integrates both Ribo- and RNAseq read counts into a single generalized linear model (GLM) using a modified design matrix inspired by Riborex and the formula/testDTU approach from SatuRn. At present, DotSeq accepts count data generated with `featureCounts`. Development is underway to extend support for additional quantification tools, including `mmquant` and `HTseq`.

Load Library

First, we need to load the DOTSeq library.

```
library(DOTSeq)
```

Example Dataset

To demonstrate the use of `DotSeq`, we use the dataset generated by (Ly et al. 2024). They investigated the stringency of start codon selection in 3 different mammalian cell cycle stages. They prepared paired translation initiation site profiling, elongating ribosome profiling, and RNA sequencing data for synchronized interphase, mitotic arrested, and cycling mitotic HeLa cells. Two biological replicates were performed for each cell cycle stage. The raw sequencing reads is deposited on the NCBI Gene Expression Omnibus (GEO) under the accession number GSE230189.

```
dir <- system.file("extdata", package = "DOTSeq")
list.files(dir)
```

```
## [1] "dotseq.bed.gz"              "dotseq.gtf.gz"              "featureCounts.dotseq.out.gz" "metada
```

Input Data

The input for DotSeq is the read counts table summarized from featureCounts. This table should include both the Ribo- and RNAseq data combined. It should also be organised as a dataframe with rows correspond to genes and columns correspond to samples as shown below.

```
cnt <- read.table(file.path(dir, "featureCounts.dotseq.out.gz"), header=TRUE, comment.char ='#')
names(cnt) <- gsub(".*(SRR[0-9]+).*", "\\1", names(cnt))
```

We can check the first five lines of the table:

```
head(cnt)
```

```
##               Geneid  Chr     Start       End Strand Length SRR24230465 SRR24230467 SRR24230469 SRR24
## 1 ENSG00000000003.16 chrX 100627148 100627156      -      9           0           0           0
## 2 ENSG00000000003.16 chrX 100627247 100627331      -     85           0           0           0
## 3 ENSG00000000003.16 chrX 100627415 100627441      -     27           0           0           0
## 4 ENSG00000000003.16 chrX 100627525 100627610      -     86           0           0           0
## 5 ENSG00000000003.16 chrX 100627635 100627655      -     21           0           0           0
## 6 ENSG00000000003.16 chrX 100627674 100627700      -     27           0           0           0
##   SRR24230470 SRR24230481 SRR24230483 SRR24230485 SRR24230462 SRR24230466 SRR24230472 SRR24230474 SRR
## 1           0           0           0           0           0           1           0           0
## 2           0           0           0           0           0           0           0           0
## 3           0           0           0           0           0           0           0           0
## 4           0           0           0           0           0           0           0           0
## 5           0           0           0           0           0           0           0           0
## 6           0           0           0           0           0           0           0           0
##   SRR24230476 SRR24230478 SRR24230484
## 1           0           0           0
## 2           0           0           0
## 3           0           0           0
## 4           0           0           0
## 5           0           0           0
## 6           0           0           0
```

DOTSeq also requires annotation files describing the genomic locations of ORFs (open reading frames). Here, we specify the file path for the annotation files:

```
flat <- file.path(dir, "dotseq.gtf.gz")
bed <- file.path(dir, "dotseq.bed.gz")
```

Prepare Condition Table

Then we need to define the conditions/ treatments of samples in the Ribo- and RNAseq data.

```
meta <- read.table(file.path(dir, "metadata.txt"))
names(meta) <-  c("run","strategy","replicate","treatment","condition")
View(meta)
```

In this example, we will only look at the cells treated with cyclohexamide (chx).

```
cond <- meta[meta$treatment=="chx",]
cond$treatment <- NULL
```

Testing for Differential ORF Translation

After the read count table and condition dataframe is ready, we can use fitDot(). This function integrates the dispersion estimation and normalization approach from `DESeq2` and the design matrix inspired by `Riborex` to prepare a `SatuRn` model fitting formula.

```
m <- fitDOT(countTable = cnt, conditionTable = cond, flattenedFile = flat, bed = bed, stringent = TRUE,
```

```
##  - Design formula: ~0 + condition + replicate + modality + effect1 + effect2
##  - Parse the flattened GFF file
```

```
## Warning: 0 aggregate geneIDs were found truncated in featureCounts output
```

```
##  - Create a DEXSeq object
```

```
## Warning in DESeqDataSet(rse, design, ignoreRank = TRUE): some variables in design formula are charac
```

```
##  - Keep ORFs where all replicates in at least one condition have counts >= 1
##  - Filter out 6267 single ORF genes
##  - Number of ORFs passing filter: 35863
##  - Fit quasi-binomial generalised linear models
```

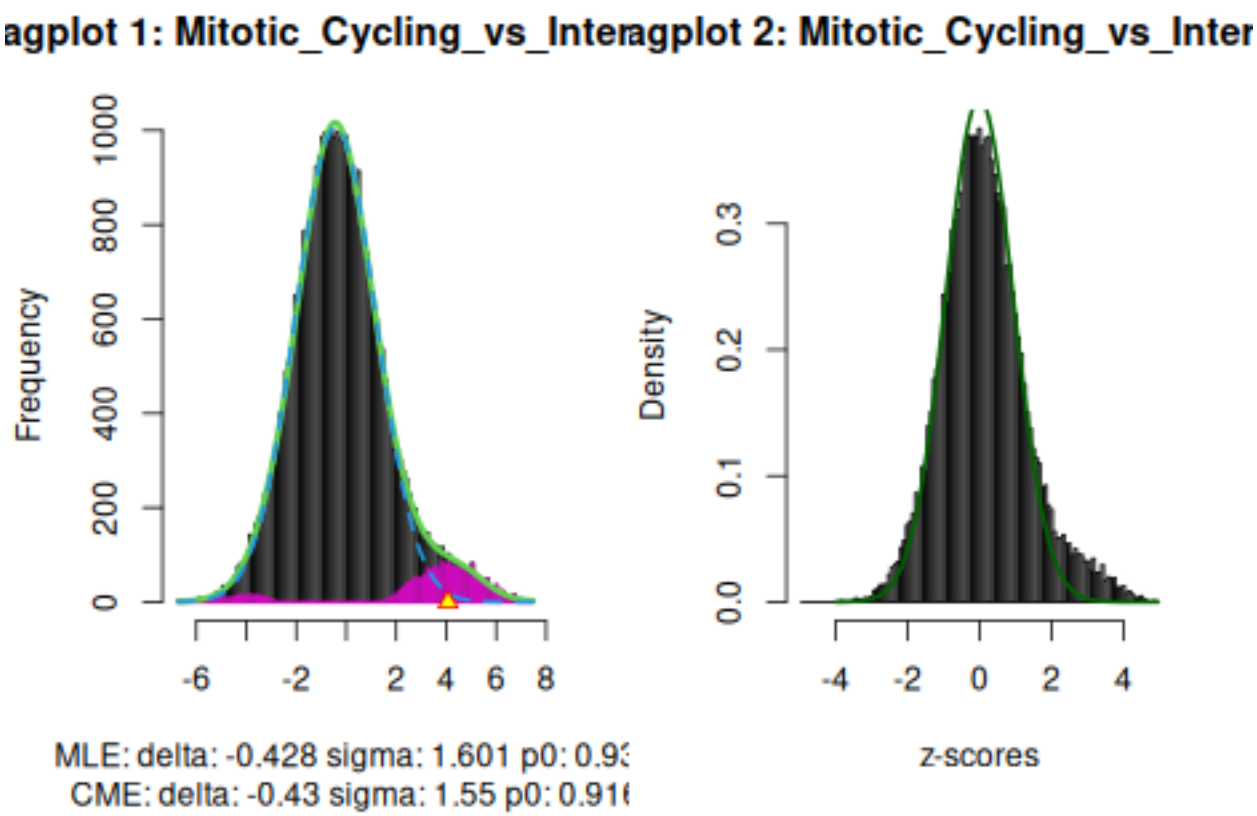
*Note: With `stringent=TRUE`, fitDot removes single-ORF genes and users should set this as 'TRUE' if wanting to calculate the occupancy shift.

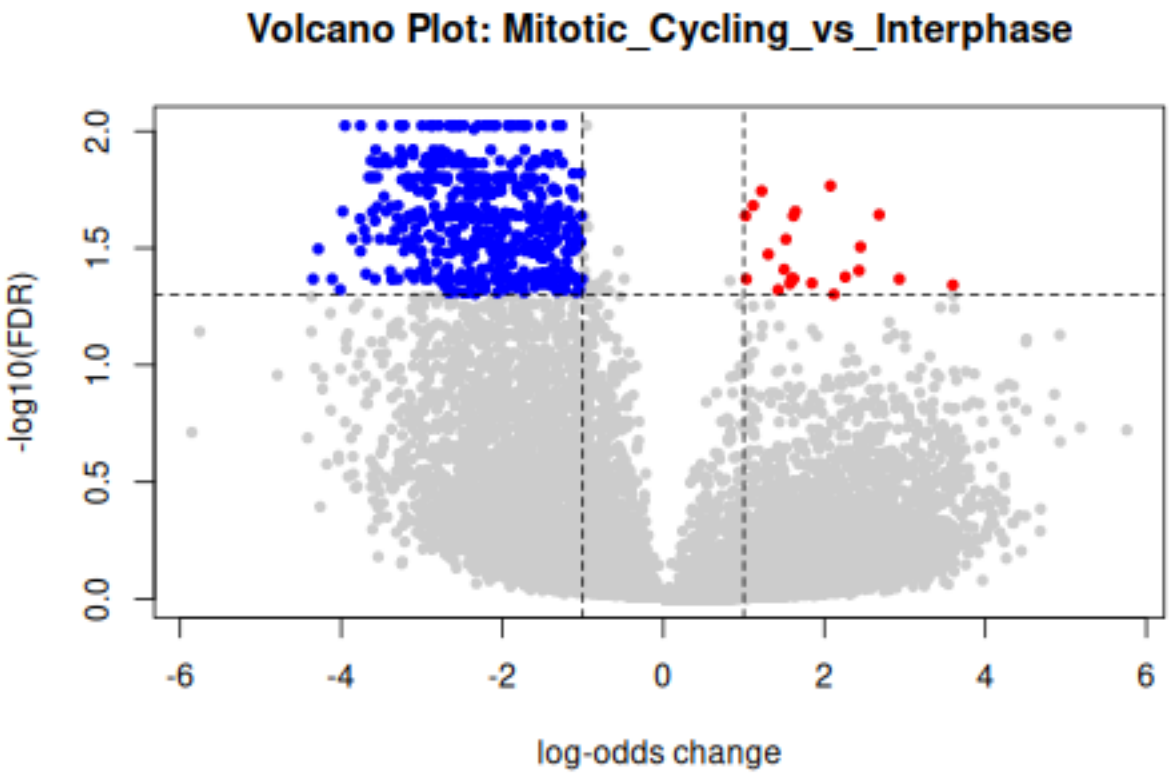
The default formula for `DotSeq` is `~0 + condition + replicate + modality + effect1 + effect2`

Contrast

Once the model is fitted using fitDOT(), users can select the specific contrast (i.e., condition comparison) they wish to test from the contrast matrix stored in the output object `m`. This function will also generate a volcano plot. For example, the following code tests the second contrast (Mitotic Cycling vs Interphase) in the matrix:

```
results <- testDOT(sumExp = m$sumExp, m$contrastMat[, 2], main = colnames(m$contrastMat)[[2]], diagplot
```






```
results <- merge(m$orfs, results, by = "row.names")
```

From this object, we can also obtain the dataframe of differentially translated ORFs that are significant with a false discovery rate of 5%:

```
sigRes <- results[results$empirical.fdr<0.05,]
```

Visualisation

Users can specify which ORF they want to compare on the heatmap. For example, this heatmap is showing the differential translation between uORF and mORF:

```
pairedDOTHeatmap(results, "uORF", "hsapiens_gene_ensembl", "hgnc_symbol", "uORF.dot.pdf")
```

and this heatmap is showing the differential translation between dORF and mORF:

```
pairedDOTHeatmap(results, "dORF", "hsapiens_gene_ensembl", "hgnc_symbol", "dORF.dot.pdf")
```

Session Info

```
sessionInfo()
```

```
## R version 4.5.0 (2025-04-11)
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 24.04.2 LTS
##
```

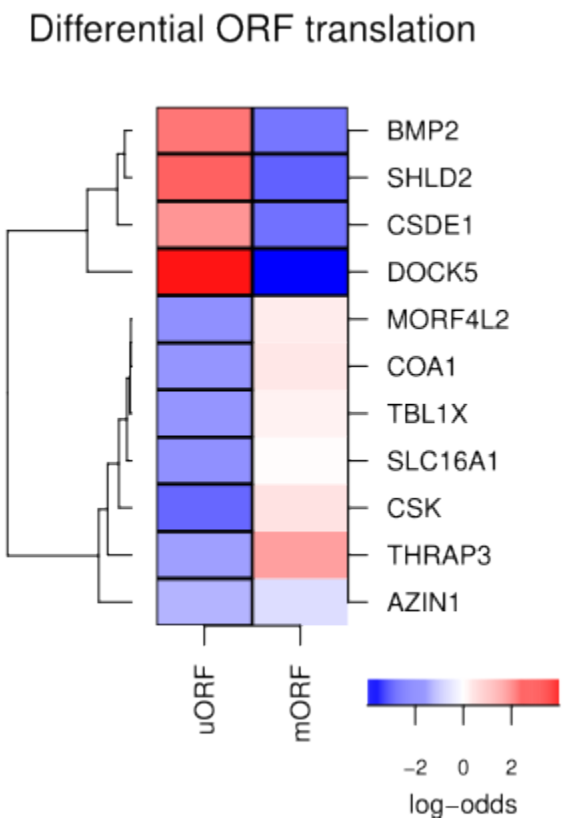


Figure 1: uORF_Heatmap

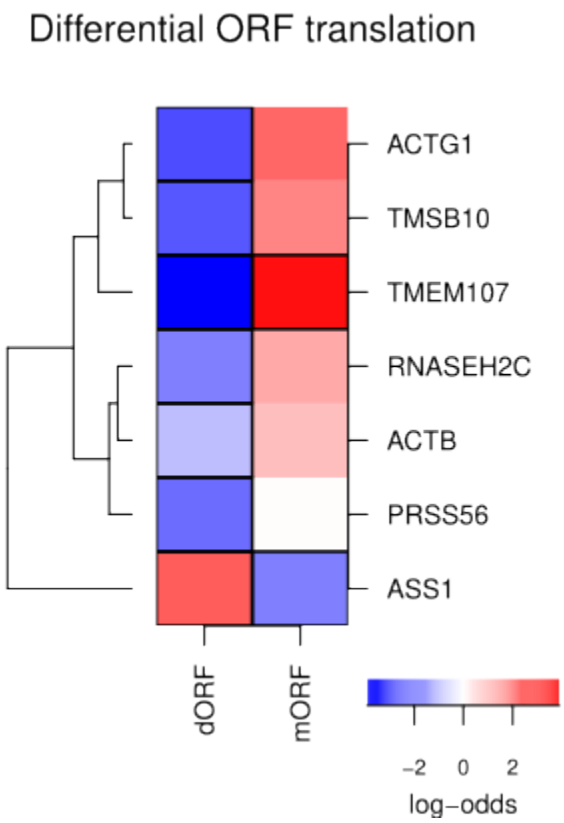


Figure 2: dORF_Heatmap

```
## 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-r0.3.26.so;  LAPACK version 3.12.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8        LC_COLLATE=en_U
##  [6] LC_MESSAGES=en_US.UTF-8    LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Etc/UTC
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] rmarkdown_2.29              knitr_1.50                  polyester_1.29.1           DOTSeq_0.12
##  [6] satuRn_1.17.0               DEXSeq_1.55.1               RColorBrewer_1.1-3         AnnotationDb
## [11] BiocParallel_1.43.4         SummarizedExperiment_1.39.1 Biobase_2.69.0             MatrixGener
## [16] rtracklayer_1.69.1          GenomicRanges_1.61.1        Seqinfo_0.99.2             IRanges_2.43
## [21] BiocGenerics_0.55.1         generics_0.1.4              biomaRt_2.65.1
##
## loaded via a namespace (and not attached):
##   [1] DBI_1.2.3                bitops_1.0-9             pbapply_1.7-4            httr2_1.2.1
##   [7] magrittr_2.0.3           compiler_4.5.0           RSQLite_2.4.2            png_0.1-8
##  [13] profvis_0.4.0            stringr_1.5.1            pkgconfig_2.0.3          crayon_1.5.3
##  [19] XVector_0.49.0           ellipsis_0.3.2           logspline_2.1.22         Rsamtools_2.25.2
##  [25] ps_1.9.1                 tinytex_0.57             purrr_1.1.0              bit_4.6.0
##  [31] jsonlite_2.0.0           progress_1.2.3           blob_1.2.4               later_1.4.2
##  [37] prettyunits_1.2.0        R6_2.6.1                 bslib_0.9.0              stringi_1.8.7
##  [43] pkgload_1.4.0            boot_1.3-31              jquerylib_0.1.4          Rcpp_1.1.0
##  [49] Matrix_1.7-3             splines_4.5.0            tidyselect_1.2.1         rstudioapi_0.17.1
##  [55] miniUI_0.1.2             codetools_0.2-20         hwriter_1.3.2.1          curl_6.4.0
##  [61] lattice_0.22-7           tibble_3.3.0             withr_3.0.2              shiny_1.11.1
##  [67] desc_1.4.3               survival_3.8-3           urlchecker_1.0.1         BiocFileCache_2.99.5
##  [73] pillar_1.11.0            filelock_1.0.3           RCurl_1.98-1.17          hms_1.1.3
##  [79] xtable_1.8-4             glue_1.8.0               tools_4.5.0              BiocIO_1.19.0
##  [85] GenomicAlignments_1.45.2 fs_1.6.6                 XML_3.99-0.18            grid_4.5.0
##  [91] cli_3.6.5                rappdirs_0.3.3           S4Arrays_1.9.1           dplyr_1.1.4
##  [97] digest_0.6.37            SparseArray_1.9.1        htmlwidgets_1.6.4        geneplotter_1.87.0
## [103] memoise_2.0.1            htmltools_0.5.8.1        lifecycle_1.0.4          httr_1.4.7
## [109] bit64_4.6.0-1
```

Ly, Jimmy, KeHui Xiang, Kuan-Chung Su, Gunter B. Sissoko, David P. Bartel, and Iain M. Cheeseman. 2024. "Nuclear release of eIF1 restricts start-codon selection during mitosis." *Nature* 635: 490–98. https://doi.org/10.1038/s41586-024-08088-3.