riboSeqR

Thomas J. Hardcastle, Betty Y.W. Chung

June 30, 2024

Introduction

Ribosome profiling extracts those parts of a coding sequence currently bound by a ribosome (and thus, are likely to be undergoing translation). Ribosomes typically cover between 20-30 bases of the mRNA (dependant on conformational changes) and move along the mRNA three bases at a time. Sequenced reads of a given length are thus likely to lie predominantly in a single frame relative to the start codon of the coding sequence. This package presents a set of methods for parsing ribosomal profiling data from multiple samples and aligned to coding sequences, inferring frameshifts, and plotting the average and transcript-specific behaviour of these data. Methods are also provided for extracting the data in a suitable form for differential translation analysis. For a fuller description of these methods and further examples of their use, see Chung & Hardcastle *et al* (2015) [1].

Getting Data

`riboSeqR` currently reads alignment data from BAM files, or from flat text files, although BAM format is recommended.

Workflow Example

Begin by loading the riboSeqR library.

```
> library(riboSeqR)
```

Identify the data directory for the example data.

```
> datadir <- system.file("extdata", package = "riboSeqR")
```

The `fastaCDS` function can be used to guess at potential coding sequences from a (possibly compressed; see `base::file`) fasta file containing mRNA transcripts (note; do not use this on a genome!). These can also be loaded into a *GRanges* object from an annotation file.

```
> chlamyFasta <- paste(datadir, "/rsem_chlamy236_deNovo.transcripts.fa", sep = "")
> fastaCDS <- findCDS(fastaFile = chlamyFasta,
+                     startCodon = c("ATG"),
+                     stopCodon = c("TAG", "TAA", "TGA"))
```

The ribosomal and RNA (if available) alignment files are specified.

```
> ribofiles <- paste(datadir,
+                    "/chlamy236_plus_deNovo_plusOnly_Index", c(17,3,5,7), sep = "")
> rnafiles <- paste(datadir,
+                   "/chlamy236_plus_deNovo_plusOnly_Index", c(10,12,14,16), sep = "")
```

The aligned ribosomal (and RNA) data can be read in using the `readRibodata` function. The columns can be specified as a parameter of the `readRibodata` function if the data in the alignment files are differently arranged.

```
> riboDat <- readRibodata(ribofiles, replicates = c("WT", "WT", "M", "M"))
```

The alignments can be assigned to frames relative to the coding coordinates with the `frameCounting` function.

```
> fCs <- frameCounting(riboDat, fastaCDS)
```

The predominant reading frame, relative to coding start, can be estimated from the frame calling (or from a set of coordinates and alignment data) for each n-mer. The weighting decribes the proportion of n-mers fitting with the most likely frameshift. The reading frame can also be readily visualised using the `plotFS` function.

```
> fS <- readingFrame(rC = fCs); fS

[[1]]
          26    27   28   29  30
0        712 10579 2228 1175 227
1        865   696 2095  531 358
2        316  3318 7987 1919 947
frame.ML   1     0    2    2   2

[[2]]
          26   27   28  29  30
0        698 8485  597 160  73
1        715  537 1012 128 123
2        286 2663 3644 238 148
frame.ML   1    0    2   2   2

[[3]]
          26   27   28  29  30
0        542 5066  266 188  91
1        435  268  447  98  99
2        215 1467 1429 171 184
frame.ML   0    0    2   0   2

[[4]]
           26    27   28  29  30
0        1177 12129  622 201  74
1        1379   586  961 117 126
2         407  3690 3554 218 135
frame.ML    1     0    2   2   2

> plotFS(fS)
```

These can be filtered on the mean number of hits and unique hits within replicate groups to give plausible candidates for coding. Filtering can be limited to given lengths and frames, which may be inferred from the output of the `readingFrame` function.

```
> ffCs <- filterHits(fCs, lengths = c(27, 28), frames = list(1, 0),
+                    hitMean = 50, unqhitMean = 10, fS = fS)
```

We can plot the total alignment at the 5' and 3' ends of coding sequences using the `plotCDS` function. The frames are colour coded; frame-0 is red, frame-1 is green, frame-2 is blue.

```
> plotCDS(coordinates = ffCs@CDS, riboDat = riboDat, lengths = 27)
```

Note the frameshift for 28-mers.

```
> plotCDS(coordinates = ffCs@CDS, riboDat = riboDat, lengths = 28)
```

We can plot the alignment over an individual transcript sequence using the `plotTranscript` function. Observe that one CDS (on the right) contains the 27s in the same phase as the CDS (they are both red) while the putative CDSes to the left are not in phase with the aligned reads, suggesting either a sequence error in the transcript or a misalignment. The coverage of RNA sequenced reads is shown as a black curve (axis on the right).

```
> plotTranscript("CUFF.37930.1", coordinates = ffCs@CDS,
+                riboData = riboDat, length = 27, cap = 200)

NULL
```

We can extract the counts from a *riboCoding* object using the `sliceCounts` function

```
> riboCounts <- sliceCounts(ffCs, lengths = c(27, 28), frames = list(0, 2))
```

Counts for RNA-sequencing can be extracted using from the riboData object and the coding coordinates using the `rnaCounts` function. This is a relatively crude counting function, and alternatives have been widely described in the literature on mRNA-Seq.

```
> rnaCounts <- rnaCounts(riboDat, ffCs@CDS)
```

These data may be used in an analysis of differential translation through comparison with the RNA-seq data. See the description of a beta-binomial analysis in the *baySeq* vignettes for further details.

```
> library(baySeq)
> pD <- new("countData", replicates = ffCs@replicates,
+           data = list(riboCounts, rnaCounts),
+           groups = list(NDT = c(1,1,1,1), DT = c("WT", "WT", "M", "M")),
+           annotation = as.data.frame(ffCs@CDS),
+           densityFunction = bbDensity)
> libsizes(pD) <- getLibsizes(pD)
> pD <- getPriors(pD, cl = NULL)
> pD <- getLikelihoods(pD, cl = NULL)

.

> topCounts(pD, "DT", normaliseData = TRUE)

            seqnames start  end width strand frame startCodon stopCodon context
```

```
4 Cre17.g723750.t1.3   271  471   201      *     0        ATG       TGA GGAATGG
3        CUFF.9661.1   424  582   159      *     0        ATG       TGA ACAATGG
1       CUFF.43721.1     6  161   156      *     2        ATG       TAA GCAATGC
2        CUFF.9523.1    78 1040   963      *     2        ATG       TAA AAAATGG
  minus3 plus1    WT.1    WT.2     M.1     M.2      likes   DT    FDR.DT
4      G     G   11:11     1:1     2:2     1:1 0.05882355 M=WT 0.9411765
3      A     G     1:1     1:1     1:1     0:0 0.05882354 M=WT 0.9411765
1      G     C     5:5     4:4     3:3     0:0 0.05882353 M=WT 0.9411765
2      A     G 841:841 592:592 298:298 721:721 0.05882330 M=WT 0.9411765
    FWER.DT
4 0.9411765
3 0.9965398
1 0.9997965
2 0.9999880
```

```
[[1]]
          26    27   28   29  30
0        712 10579 2228 1175 227
1        865   696 2095  531 358
2        316  3318 7987 1919 947
frame.ML   1     0    2    2   2

[[2]]
          26   27   28  29  30
0        698 8485  597 160  73
1        715  537 1012 128 123
2        286 2663 3644 238 148
frame.ML   1    0    2   2   2

[[3]]
          26   27   28  29  30
0        542 5066  266 188  91
1        435  268  447  98  99
2        215 1467 1429 171 184
frame.ML   0    0    2   0   2

[[4]]
           26    27   28  29  30
0        1177 12129  622 201  74
1        1379   586  961 117 126
2         407  3690 3554 218 135
frame.ML    1     0    2   2   2
```

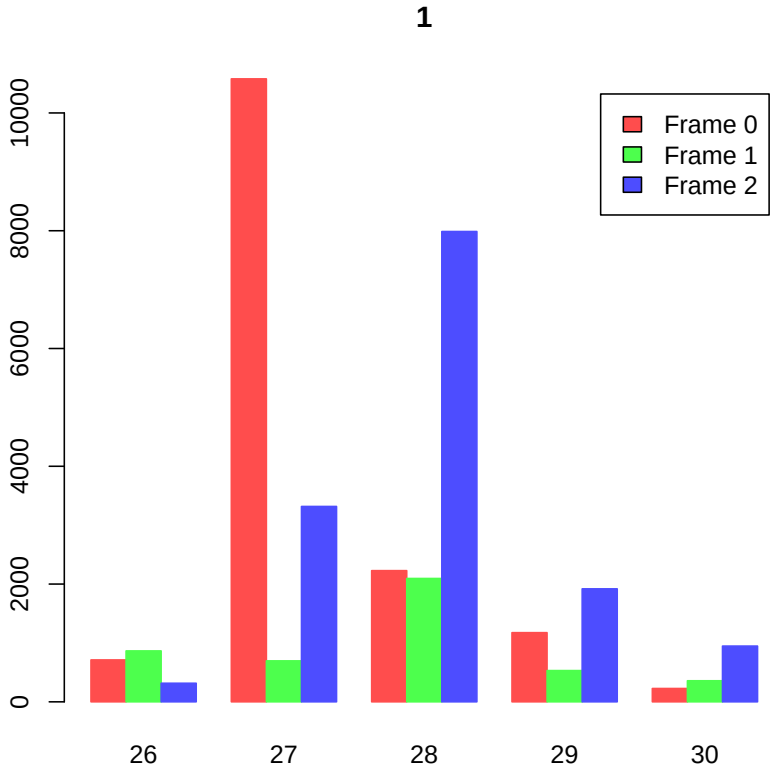

Figure 1: Number of n-mers in each frame relative to coding start
27-mers are predominantly in frame-1, while 28-mers are chiefly in frame-0.

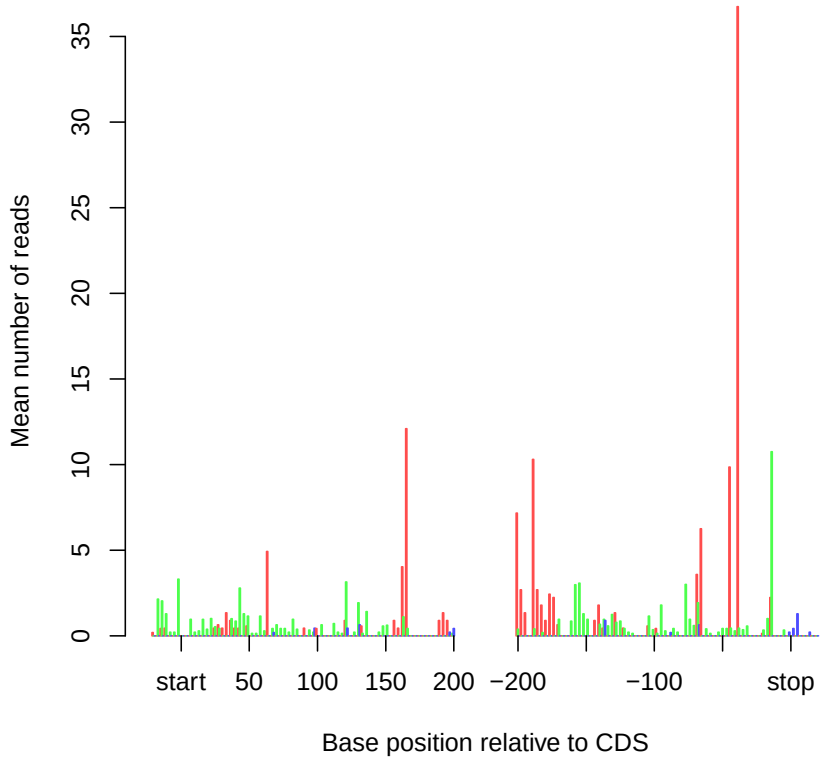

Figure 2: Average alignment of 27-mers to 5' and 3' ends of coding sequences

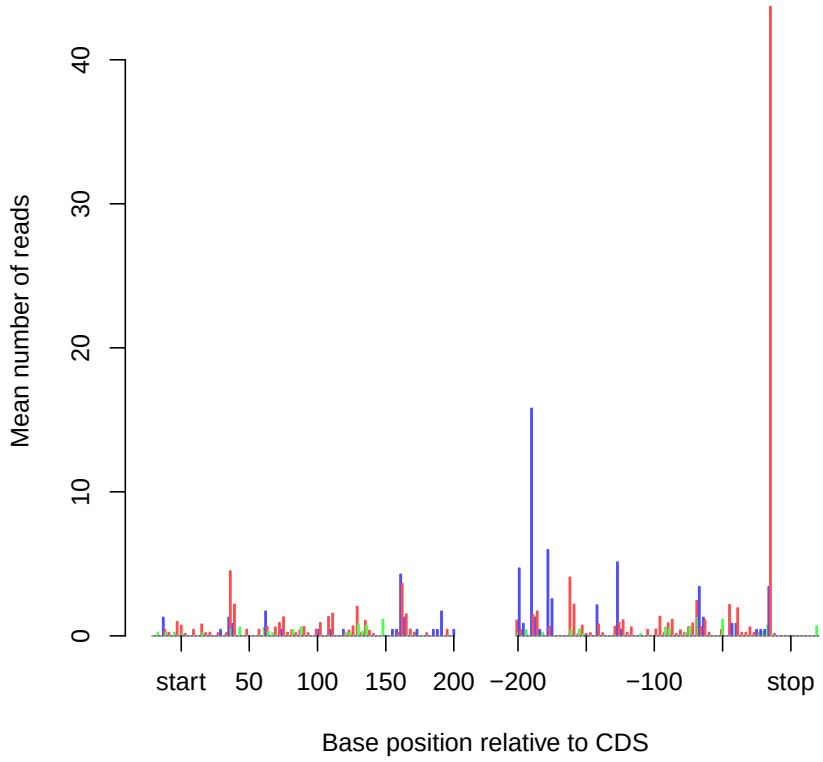

Figure 3: Average alignment of 28-mers to 5' and 3' ends of coding sequences

```
NULL
```

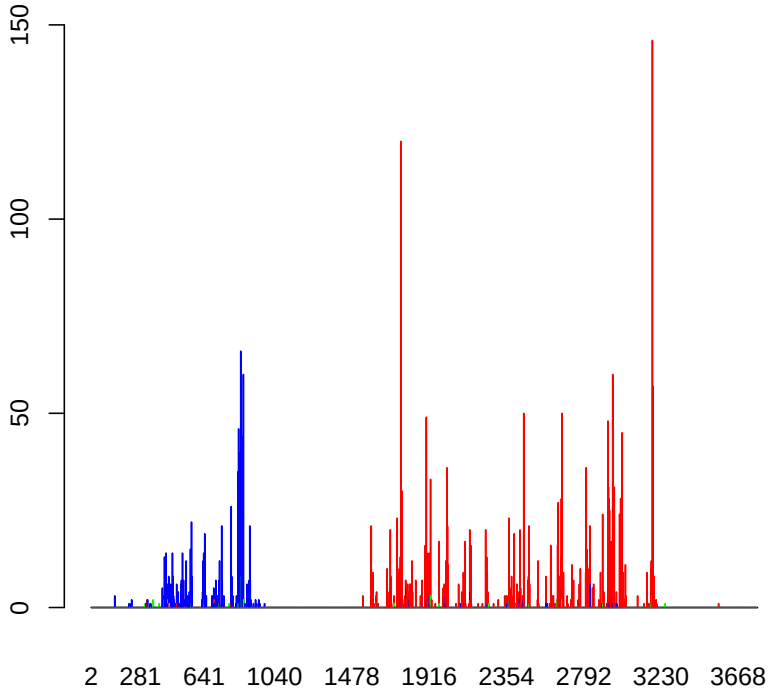

Figure 4: Alignment to individual transcript

Session Info

```
> sessionInfo()

R version 4.4.1 (2024-06-14)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04 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-r0.3.26.so;  LAPACK version 3.12.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=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
[8] base

other attached packages:
[1] baySeq_2.39.0        riboSeqR_1.39.0      abind_1.4-5
[4] GenomicRanges_1.57.1 GenomeInfoDb_1.41.1  IRanges_2.39.0
[7] S4Vectors_0.43.0     BiocGenerics_0.51.0

loaded via a namespace (and not attached):
 [1] limma_3.61.2            BiocStyle_2.33.1        jsonlite_1.8.8
 [4] compiler_4.4.1          BiocManager_1.30.23     crayon_1.5.3
 [7] Rcpp_1.0.12             seqLogo_1.71.0          Rsamtools_2.21.0
[10] bitops_1.0-7            Biostrings_2.73.1       parallel_4.4.1
[13] statmod_1.5.0           BiocParallel_1.39.0     yaml_2.3.8
[16] fastmap_1.2.0           lattice_0.22-6          R6_2.5.1
[19] XVector_0.45.0          knitr_1.47              maketools_1.3.0
[22] GenomeInfoDbData_1.2.12 rlang_1.1.4             xfun_0.45
[25] sys_3.4.2               cli_3.6.3               zlibbioc_1.51.1
[28] grid_4.4.1              locfit_1.5-9.10         digest_0.6.36
[31] edgeR_4.3.4             evaluate_0.24.0         codetools_0.2-20
[34] buildtools_1.0.0        rmarkdown_2.27          httr_1.4.7
[37] tools_4.4.1             htmltools_0.5.8.1       UCSC.utils_1.1.0
```

References

[1] BY Chung and TJ Hardcastle and JD Jones and N Irigoyen and AE Firth and DC Baulcombe and I Brierley *The use of duplex-specific nuclease in ribosome profiling and a user-friendly software package for Ribo-seq data analysis.* RNA (2015).