rCube - RNA-Rates in R

Leonhard Wachutka[1,*], Carina Demel[2], Julien Gagneur[1]

[1] Department of Informatics, Technical University of Munich, Munich, Germany

[2] Max Planck Institute for biophysical Chemistry, Göttingen, Germany

[*] wachutka (at) in.tum.de

August 2, 2017

Abstract

rCube provides a framework for the estimation of RNA metabolism Rates in R (R^3). The rCube package complements the recently published transient transcriptome sequencing (TT-seq) protocol. It has been shown, that 4sU-labeling and subsequent purification of RNA allows to monitor local RNA synthesis. Therefore, the information from TT-seq/4sU-seq and total RNA-seq samples is used to model RNA synthesis, splicing, and degradation rates based on first-order kinetics. The rCube package works on count data and provides a series of functionalities to extract them from the desired features. It allows to extract junctions and constitutive exons from feature annotations, count reads from BAM-files, and normalize different samples against each other using a variety of different methods.

rCube version: 1.1.0

If you use rCube in published research, please cite:

B. Schwalb, M. Michel, B. Zacher, K. Frühauf, C. Demel, A. Tresch, J. Gageur, and P. Cramer:
TT-seq maps the human transient transcriptome.
Science (2016). doi:10.1126/science.aad9841 [1]

Contents

1 Background

As described in .... 4sU-seq allows to monitor changes in the RNA metabolism. If cells are exposed to 4sU, they rapidly take up this Uridine analog and incorporate it into newly-synthesized RNAs. This way, newly-synthesized RNAs are labeled and can be extracted from the total RNA in the sample. The longer the labeling time, e.i. the time from 4sU addition to harvesting the cells, the bigger is the proportion of labeled RNAs among all RNAs.

explain the why we have spike ins (diagram?)

explain time series vs only label total

ref to TTseq

ref to MSB

define synthesis, splicing, decay rate

define the read classifications: junction reads E-E, E-I, I-E junction

2 Getting started

This vignette provides a pipeline how to... starting from BAM files... You will learn how to estimate sample specific sequencing depths and cross-contamination rates from spike-in counts. These values can be used to normalize gene expression values obtained by RNA-Seq and thus estimate gene-specific synthesis and degradation rates. By extracting reads spanning junctions, splicing times can be estimated. For more robust estimation, multiple samples with different labeling times are taken into account. Before starting, the package must be loaded by:

```
library("rCube")
```

2.1 Experimental Design

Example data sets for a T cell activation are stored in the `inst/extdata` of the *rCube* package.

We first look at the experimental design file `experimentalDesign.txt`, that can be imported as a *data.frame*.

```
folder <- system.file("extdata/Jurkat", package='rCube')
folder
```

```
## [1] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/rCube/extdata/Jurkat"
```

```
#expDesign <- read.delim(file.path(folder, "experimentalDesign.txt"))
#expDesign
#setupExperiment(rows, designMatrix=expDesign, files=NULL)
```

Alternatively, the experimental design matrix can be constructed from the bam file names, if they follow the following convention {condition}_{L|T}_{labelingTime}_{replicate}.bam

```
bamfiles <- list.files(folder, pattern="*.bam$")
bamfiles
```

```
## [1] "ActivatedJurkat_L_5_1.bam" "ActivatedJurkat_L_5_2.bam"
## [3] "ActivatedJurkat_T_5_1.bam" "ActivatedJurkat_T_5_2.bam"
## [5] "RestingJurkat_L_5_1.bam"   "RestingJurkat_L_5_2.bam"
## [7] "RestingJurkat_T_5_1.bam"   "RestingJurkat_T_5_2.bam"
```

```
#setupExperiment(rows, designMatrix=NULL, files=bamfiles)
```

2.2 Read classification

2.3 Gene model

(exon, into and junctions) by gff de novo + gff

2.4 Counting

2.5 Spike-ins

The artifical spike-in annotations and labeling information can be loaded via:

```
data("spikeins")
data("spikeinLabeling")
spikeinLengths <- width(spikeins)
```

2.6 Spike-in design

2.7 Spike-in counting

2.8 Size factor based on spike-in

2.9 Estimate dispersion

2.10 Fit the rates

2.11 describe the different fitting functions

2.12 describe the class of the returned object (rCubeRates)

2.13 Input Data

The rCube package works on *rCubeExperiment* containers, that rely on the *SummerizedExperiment* class. The `rowRanges` of the *rCubeExperiment* is a *GRanges* object of features, for which RNA rates should be estimated. Experimental sample

information can be either provided by a design matrix or this information can be extracted from the BAM-file names (when they fulfil the required structure). The file name should be a string containing condition, labelingTime (as integer), L/T sample information, and replicate (integer/string), separated by a "_". Then, an empty *rCubeExperiment*, e.g. for the artificial spike-ins, can be constructed as follows:

```
data("designMatrix")

spikeinCounts <- setupExperimentSpikeins(rows=spikeins,
                                         designMatrix=designMatrix,
                                         length=spikeinLengths,
                                         labelingState=spikeinLabeling)
```

The `setupExperiment` can be used analogically for genes/exons/introns/junctions. Here, only the `rows` and either `designMatrix` or `files` has to be set. See also section 2.14.

The individual information from the *rCubeExperiment* can be assessed by:

```
# feature information
rowRanges(spikeinCounts)
```

```
## GRanges object with 6 ranges and 9 metadata columns:
##           seqnames    ranges strand |       source       type     score
##              <Rle> <IRanges>  <Rle> |     <factor>   <factor> <numeric>
##    Spike2    chrS2 [1,  982]      + | Fruehauf2013 transcript      <NA>
##   Spike12   chrS12 [1,  947]      + | Fruehauf2013 transcript      <NA>
##    Spike4    chrS4 [1, 1011]      + | Fruehauf2013 transcript      <NA>
##    Spike5    chrS5 [1, 1012]      + | Fruehauf2013 transcript      <NA>
##    Spike8    chrS8 [1, 1076]      + | Fruehauf2013 transcript      <NA>
##    Spike9    chrS9 [1, 1034]      + | Fruehauf2013 transcript      <NA>
##               phase     gene_id transcript_id    length labelingState
##           <integer> <character>   <character> <integer>      <factor>
##    Spike2      <NA>      Spike2        Spike2      <NA>          TRUE
##   Spike12      <NA>     Spike12       Spike12      <NA>         FALSE
##    Spike4      <NA>      Spike4        Spike4      <NA>          TRUE
##    Spike5      <NA>      Spike5        Spike5      <NA>         FALSE
##    Spike8      <NA>      Spike8        Spike8      <NA>          TRUE
##    Spike9      <NA>      Spike9        Spike9      <NA>         FALSE
##           labeledSpikein
##                <logical>
##    Spike2          FALSE
##   Spike12          FALSE
##    Spike4          FALSE
##    Spike5          FALSE
##    Spike8          FALSE
##    Spike9          FALSE
##   -------
##   seqinfo: 6 sequences from an unspecified genome; no seqlengths
```

```
# sample information
colData(spikeinCounts)
```

```
## DataFrame with 8 rows and 5 columns
##           sample condition       LT labelingTime replicate
##         <factor>  <factor> <factor>    <numeric>  <factor>
## A_L_5_1  A_L_5_1         A        L            5         1
## A_L_5_2  A_L_5_2         A        L            5         2
## B_L_5_1  B_L_5_1         B        L            5         1
```

```
## B_L_5_2  B_L_5_2         B        L            5         2
## A_T_5_1  A_T_5_1         A        T            5         1
## A_T_5_2  A_T_5_2         A        T            5         2
## B_T_5_1  B_T_5_1         B        T            5         1
## B_T_5_2  B_T_5_2         B        T            5         2
```

```
# read counts
assay(spikeinCounts)
```

```
##         A_L_5_1 A_L_5_2 B_L_5_1 B_L_5_2 A_T_5_1 A_T_5_2 B_T_5_1 B_T_5_2
## Spike2       NA      NA      NA      NA      NA      NA      NA      NA
## Spike12      NA      NA      NA      NA      NA      NA      NA      NA
## Spike4       NA      NA      NA      NA      NA      NA      NA      NA
## Spike5       NA      NA      NA      NA      NA      NA      NA      NA
## Spike8       NA      NA      NA      NA      NA      NA      NA      NA
## Spike9       NA      NA      NA      NA      NA      NA      NA      NA
```

2.14 Read counting

All RNA rate estimations of this package rely on read counts. These can be either provided as count matrices, or read counts can be obtained from BAM files using the rCube pipeline.

```
#TODO
```

Alternatively, count matrices can be assigned to the correctly formatted, empty *rCubeExperiment* object:

```
#TODO
```

3 Normalization of TT-seq/4sU-seq and RNA-Seq samples

The three possible normalization methods are described in detail below.

3.1 Normalization by fitting a GLM to artifical spike-in read counts

By normalization, we want to account for the sequencing depth of different samples and especially adjust the ratio between Labeled to Total RNA-Seq libraries. Additionally, the labeled RNA extraction is not perfect and some unlabeled RNA may contaminate the labeled RNA fraction. As the gene-expression also always varies a little bit for biological samples, read counts from real genes might lead to confusing estimations. Therefore, we apply our normalization approach only to artificial spike-ins from the External RNA Control Consortium (ERCC), for which the initial amount of each spike-in is known and the same across all samples. Some of the spike-ins were 4sU-labeled by in-vitro transcription, so the cross-contamination of unlabeled spike-ins in labeled samples can be monitored. The distribution of 4sU-labeled and unlabeld spike-ins among different samples can be monitored with `plotSpikeinCountsVsSample`.

```
data(spikeinCounts)
plotSpikeinCountsVsSample(spikeinCounts)
```

The goal of this package is to reliably estimate sequencing depths and cross-contamination rates per sample, given 4sU- and total RNA-Seq data.

The sample specific parameters like sequencing depth and cross-contamination rate are estimated from spike-in counts only. Therefore, we fit a generalized linear model (GLM) of the Negative Binomial family with a log link function. The response of the GLM are the observed spike-in counts, and the terms that specify the linear predictor of the response are comprised of:

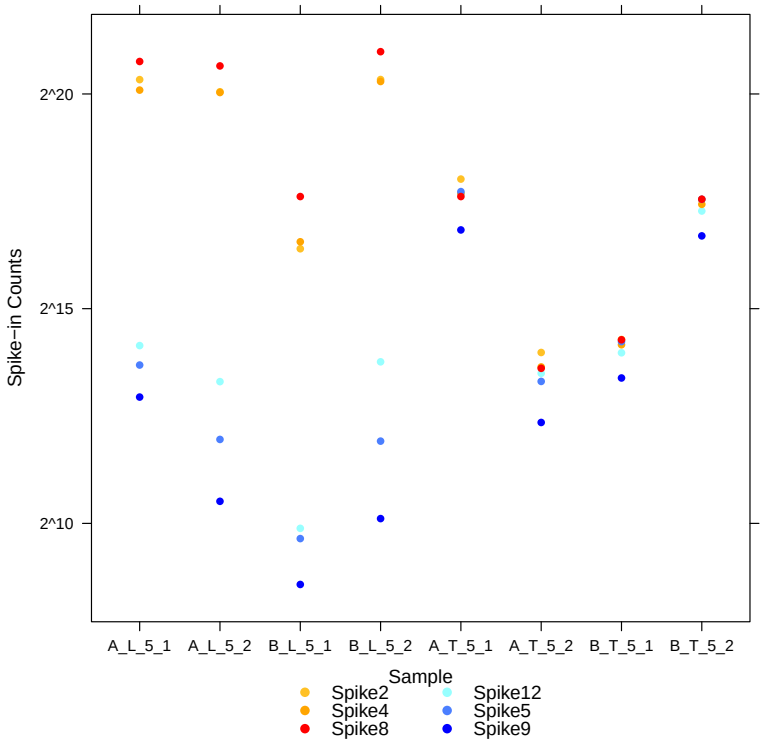

Figure 1: **Spikein-vs-Sample-plot.**

- a sample specific factor (that reflects the sample specific sequencing depth),
- a labeled sample specific factor (that reflects the control for cross contamination (only estimated for unlabeled spike-ins in labeled samples)), and
- a spike-in specific factor to allow for some spike-in specific variation e.g. due to sequence biases.

Additionally, the length of each spike-in is used as an offset, i.e. a known slope for the covariate.

```
data(geneCounts)
data(spikeinCounts)
geneCounts <- estimateSizeFactors(geneCounts, spikeinCounts, method="spikeinGLM")
colnames(colData(geneCounts))
```

```
## [1] "sample"              "condition"           "LT"
## [4] "labelingTime"        "replicate"           "sequencing.depth"
## [7] "cross.contamination"
```

```
geneCounts$sequencing.depth
```

```
## A_L_5_1 A_L_5_2 B_L_5_1 B_L_5_2 A_T_5_1 A_T_5_2 B_T_5_1 B_T_5_2
## 1.00000 0.90453 0.08914 1.10879 0.16871 0.00919 0.01453 0.14209
```

```
geneCounts$cross.contamination
```

```
## A_L_5_1 A_L_5_2 B_L_5_1 B_L_5_2 A_T_5_1 A_T_5_2 B_T_5_1 B_T_5_2
## 0.01200 0.00445 0.00721 0.00411 1.00000 1.00000 1.00000 1.00000
```

Note, the cross-contamination value for all total RNA-seq samples is 1, as 100% of the unlabeled RNAs are supposed to be in the sample. Additional fitting results are stored in the `metadata` of the resulting *rCubeExperiment* object.

```
metadata(geneCounts)
```

3.2 Normalization using mean counts of artifical spike-ins

3.3 Normalization using joint model

4 Estimating gene-specific synthesis and degradation rates

Using the sample-specific values for sequencing depth and cross-contamination as estimated in the previous section, we can now normalize all the samples. It is especially important to bring labeled (4sU-seq/TT-seq) and total RNA-seq samples to comparable scales. Labeled and total RNA-seq samples can be sequenced at the same depth, and the same amount of RNA is used for library preparation, but the resulting read counts do not reflect the true ratio of labeled vs all RNAs in the cells, where the amount of newly-synthesized, labeled RNA should be much less than the total RNA amount. Therefore it is necessary to upscale the read counts from total RNA-seq samples compared to the labeled RNA read counts.

4.1 Providing gene-wise dispersion estimates

Usually, read counts in different RNA-seq samples underly fluctuations due to biological or technical variances. To take these fluctuations into account, we estimate each gene's dispersion. For each gene, a single dispersion estimate for all 4sU-Seq samples and for all Total RNA-Seq samples is needed. Here, we can use the method provided in the DESeq2 package [2] The wrapper function `estimateSizeDispersions` applies the DESeq algorithm to all genes, while separating the count table according to the RNA-Seq protocol (labeled or total RNA). It is possible to choose between all provided DESeq dispersion estimates, namely the genewise maximum likelihood dispersion estimate ("dispGeneEst"), the smooth curve fitted through the gene-wise disperion estimates ("dispFit") and the genewise dispersion estimates shrunken towards the fitted curve ("dispMAP", default). The input is an *rCubeExperiment* object with read counts for the features of interest. The function returns an updated *rCubeExperiment* object with two additional columns in the `rowRanges`, namely `dispersion_L` and `dispersion_T`.

```
geneCounts <- estimateSizeDispersions(geneCounts, method='DESeqDispMAP')
rowRanges(geneCounts)
```

```
## GRanges object with 12 ranges and 2 metadata columns:
##           seqnames       ranges strand |      dispersion_L        dispersion_T
##              <Rle>    <IRanges>  <Rle> |         <numeric>           <numeric>
##    gene 1  chrTest [ 724, 2389]      * | 0.267593980890705 0.00285282168427403
##    gene 2  chrTest [1331, 2444]      * | 0.210169732934922 0.00246148010370263
##    gene 3  chrTest [1209, 2066]      * | 0.217187602571734 0.00209222050855509
##    gene 4  chrTest [1222, 3157]      * | 0.249077753495831 0.00207165673496985
##    gene 5  chrTest [ 828, 1607]      * | 0.203015823141568  0.0021568661882415
##       ...      ...          ...    ... .               ...                 ...
##    gene 8  chrTest [1055, 2066]      * | 0.217187602571734 0.00246148010370263
##    gene 9  chrTest [1218, 1881]      * | 0.885921521910211    0.10447013633359
##   gene 10  chrTest [1712, 2766]      * | 0.217187602571734 0.00246148010370263
##   gene 11  chrTest [ 646, 1502]      * | 0.202381362920392 0.00289479811680433
##   gene 12  chrTest [1381, 2451]      * |                10                  10
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

4.2 Feature-specific synthesis and degratation rate estimates

After estimating sequencing depth and cross-contamination rates per sample (see Section 3) and extracting feature-specific dispersion estimates (see Section 4.1), we can now estimate RNA synthesis and degradation rate for each feature and condition individually. Multiple replicates for the same condition can be used for a joint estimation. The user has to specify for which replicate or combination of replicates the results should be estimated. Therefore, the `replicate` parameter is a vector of all combinations that should be evaluated. For the joint estimation for multiple replicates, these have to be given as a string separeted by a ":". In the following example, we will obtain individual results for replicate 1 and 2 and also results for a joint estimation.

```
rates <- estimateRateByFirstOrderKinetics(geneCounts,
                                          replicate=c(1, 2, "1:2"),
                                          method='single',
                                          BPPARAM=BiocParallel::MulticoreParam(1))
```

5 Estimating splicing times from junction read counts

6 Session Information

This vignette was generated using the following package versions:

```
sessionInfo()
```

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: OS X El Capitan 10.11.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] rCube_1.1.0                SummarizedExperiment_1.7.5
##  [3] DelayedArray_0.3.19        matrixStats_0.52.2
##  [5] Biobase_2.37.2             GenomicRanges_1.29.12
##  [7] GenomeInfoDb_1.13.4        IRanges_2.11.12
##  [9] S4Vectors_0.15.5           BiocGenerics_0.23.0
## [11] knitr_1.16
##
## loaded via a namespace (and not attached):
##  [1] bit64_0.9-7              splines_3.4.1            Formula_1.2-2
##  [4] highr_0.6                latticeExtra_0.6-28      blob_1.1.0
##  [7] GenomeInfoDbData_0.99.1  Rsamtools_1.29.0         yaml_2.1.14
## [10] RSQLite_2.0              backports_1.1.0          lattice_0.20-35
## [13] digest_0.6.12            RColorBrewer_1.1-2       XVector_0.17.0
## [16] checkmate_1.8.3          colorspace_1.3-2         htmltools_0.3.6
## [19] Matrix_1.2-10            plyr_1.8.4               DESeq2_1.17.12
## [22] XML_3.98-1.9             genefilter_1.59.0        zlibbioc_1.23.0
## [25] xtable_1.8-2             scales_0.4.1             BiocParallel_1.11.4
## [28] htmlTable_1.9            tibble_1.3.3             annotate_1.55.0
## [31] ggplot2_2.2.1            nnet_7.3-12              lazyeval_0.2.0
## [34] survival_2.41-3          magrittr_1.5             memoise_1.1.0
## [37] evaluate_0.10.1          MASS_7.3-47              foreign_0.8-69
## [40] tools_3.4.1              data.table_1.10.4        BiocStyle_2.5.8
## [43] stringr_1.2.0            munsell_0.4.3            locfit_1.5-9.1
```

```
## [46] cluster_2.0.6            AnnotationDbi_1.39.2     Biostrings_2.45.3
## [49] compiler_3.4.1           rlang_0.1.1              grid_3.4.1
## [52] RCurl_1.95-4.8           htmlwidgets_0.9          bitops_1.0-6
## [55] base64enc_0.1-3          rmarkdown_1.6            gtable_0.2.0
## [58] DBI_0.7                  reshape2_1.4.2           GenomicAlignments_1.13.4
## [61] gridExtra_2.2.1          bit_1.1-12               Hmisc_4.0-3
## [64] rprojroot_1.2            stringi_1.1.5            Rcpp_0.12.12
## [67] geneplotter_1.55.0       rpart_4.1-11             acepack_1.4.1
```

7 References

[1] Björn Schwalb, Margaux Michel, Benedikt Zacher, Carina Demel, Achim Tresch, and Julien Gagneur. TT-seq maps the human transient transcriptome. *Science*, 352(6290):1225–1228, 2016.

[2] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2. *Genome Biology*, 15(12):550, 2014.