Ribolog

Hossein Asgharian

Oct. 2019

Ribolog is an R package comprising tools to perform a variety of analyses based on ribosome profiling data.

Module 1: CELP (Consistent Excess of Loess Preds) identifies positions of translational pause (stalling) and corrects RPF counts to eliminate the impact of stalling bias. The output of CELP can be also used to model the factors that influence translational dynamics.

Module 2: PREP normalizes and combines RNA and RPF datasets and shapes them into a format ready for quality control (QC) and translational efficiency ratio (TER) anlaysis.

Module 3: QC includes three powerful tools to quantify and visualize reproducibility among replicates and inform hypothesis generation with respect to biological effects on translation. One is princiapl component analysis (PCA) which can be done on RPF counts, RNA counts or translational efficiencies (RPF/RNA). Two novel tests involve proportion of null features (not-differentially translated transcripts) and correlation of equivalent TER tests in rep-by-rep comparisons.

Module 4: TER tests the size and significance of differential translation rates among biological samples. Although better results are always obtained with sufficient replicates, Ribolog is still able to peform the TER test with only one replicate per sample. The TER test is not restricted to pairwise comparisons; complex experimental designs including any number of samples described by several indenpedent variables can be incorporated into a single model (both univariate and multivariate models are supported).

Fig. 1 demonstrates the typical Ribolog workflow. Details of each step and available options are described in the vignettes of individual modules and in function documentations.

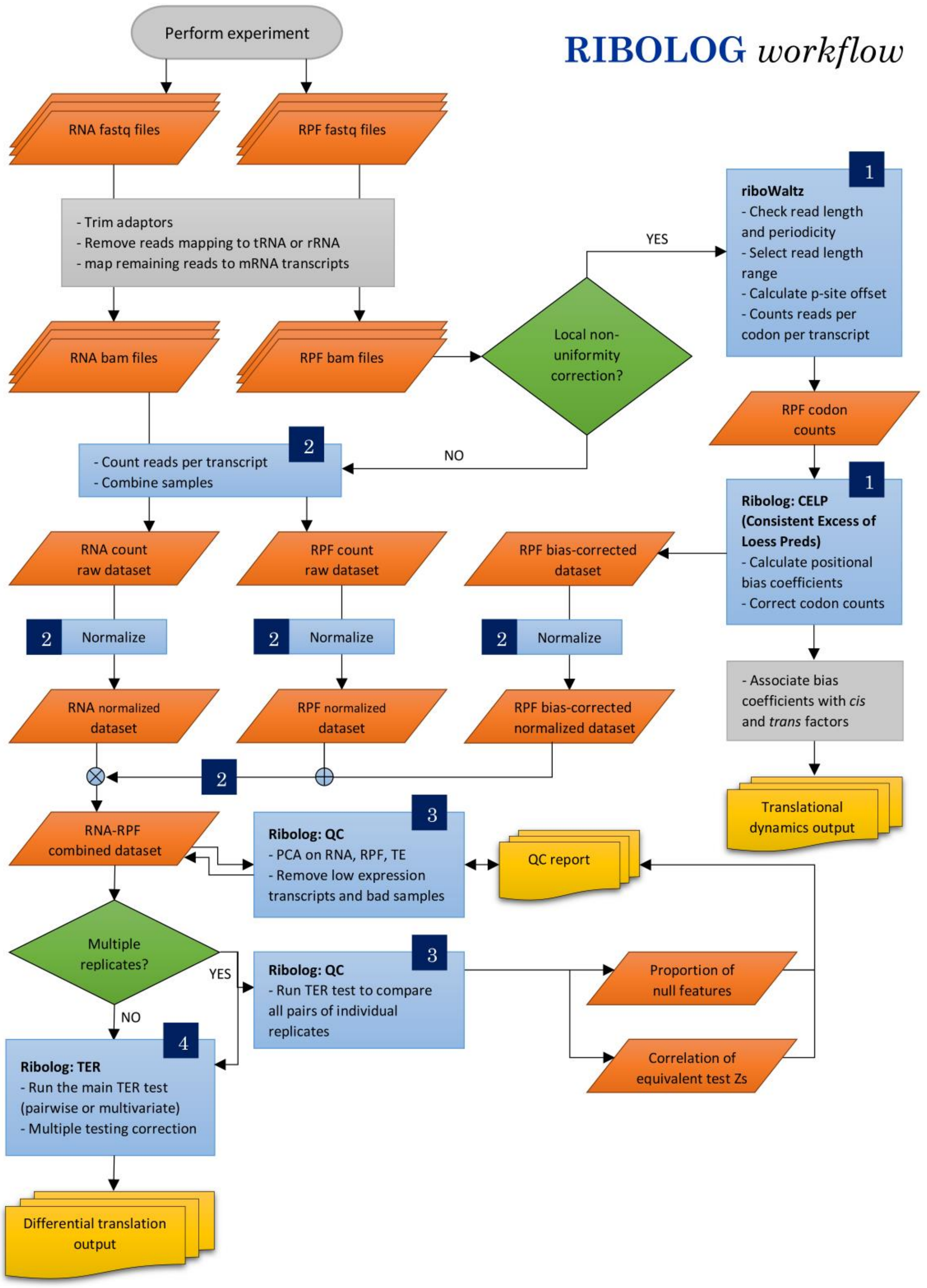

RIBOLOG workflow
Perform experiment
RNA fastq files
RPF fastq files
- Trim adaptors
- Remove reads mapping to tRNA or rRNA
- map remaining reads to mRNA transcripts
RNA bam files
RPF bam files
Local non-uniformity correction?
YES
NO
1
riboWaltz
- Check read length and periodicity
- Select read length range
- Calculate p-site offset
- Counts reads per codon per transcript
RPF codon counts
Ribolog: CELP (Consistent Excess of Loess Preds)
- Calculate positional bias coefficients
- Correct codon counts
- Associate bias coefficients with cis and trans factors
Translational dynamics output
2
- Count reads per transcript
- Combine samples
RNA count raw dataset
RPF count raw dataset
RPF bias-corrected dataset
Normalize
RNA normalized dataset
RPF normalized dataset
RPF bias-corrected normalized dataset
RNA-RPF combined dataset
3
Ribolog: QC
- PCA on RNA, RPF, TE
- Remove low expression transcripts and bad samples
QC report
Multiple replicates?
Ribolog: QC
- Run TER test to compare all pairs of individual replicates
Proportion of null features
Correlation of equivalent test Zs
4
Ribolog: TER
- Run the main TER test (pairwise or multivariate)
- Multiple testing correction
Differential translation output

Figure 1. The Ribolog workflow. First, bam files are read in. RNA and RPF reads per transcript are counted. RPF counts can be optionally corrected to remove stalling bias using the CELP method (module 1). RNA and RPF counts are then normalized for library size variation separately before or after merging (module 2). QC tests are run on the combined RNA-RPF data set to decide on the cutoffs to filter out low-count transcripts or bad samples (module 3). Some trial and error is usually needed until a satisfactorily reliable data set is obtained. This cleaned-up data set is input to module 4 to compare tranlational

efficiency ratio among samples and test the effect of biological factors (independent variables) on translation. Numbers 1-4 in dark blue rectangles refer to the module containing each operation. Grey boxes show steps that are performed outside Ribolog using other appropriate tools.

Ribolog is still a work in progress. Four additional modules are being prepared and will be released in near future.

Feel free to contact us about issues with current functions and modules, or with special requests or ideas about additional analyses you would like to be able to do with ribosome profiling data.

MODULE 1: CELP

Detecting and correcting translational stalling biases

The **CELP (Consistent Excess of Loess Preds)** method incorporated in **Ribolog** module 1 detects and corrects positional biases in RPF read count due to translational pause (stalling). The analysis of stalling serves two purposes:

- Understanding the dynamics of translational control
- Correcting RPF counts to avoid misconstruing stalling reads as signs of increased translation

To achieve this, we need the following input:

- Reference sequence fasta file
- Bam files from mapping RPF reads to the reference sequence (one file per sample)
- An annotation file listing names and segment lengths of transcripts in the reference

Because ribosome profiling mostly focuses on protein coding sequences (CDS), we recommend mapping to a reference transcriptome. If a gene-level analysis is desired (not an isoform-level analysis), we suggest choosing one transcript with the longest CDS per gene. Instructions and the python script to produce these files are provided in *section 1.0* below. Codon-level read counts are obtained by finding the most likely three nucleotides that occupied the ribosomal p-site at the time of the experiment using functions from package **riboWaltz** (https://github.com/LabTranslationalArchitectomics/riboWaltz). Functions borrowed exactly or modified slightly from the **riboWaltz** source code are characterized by the suffix `_rW`.

1.0. Before Ribolog: Obtain reference transcriptome fasta, generate annotation and ID mapper files

You need a reference file to map reads and generate an annotation file (for annotation file content see section 1.1). For ribosome profiling data, we recommend mapping to cDNA (instead of unprocessed transcriptome or genome) to make future steps of obtaining codon-level information faster, easier and less error-prone. From each gene, we choose one cDNA sequence (the one with the longest CDS). The reason for this choice is that sequencing coverage in most ribosome profiling data sets is not high enough to allow reliable isoform-specific analysis, and molecular biologists are usually interested in interpreting their data at the gene level. **Ribolog** functions use transcript IDs; however, it is useful to generate an ID mapper file to connect transcript IDs, gene IDs and gene names for downstream exploration and analysis of **Ribolog** output. cDNA fasta (one transcript per gene with the longest CDS), annotation and ID mapper files based on Ensembl genomes downloaded on June 5-8 2019 are included with the **Ribolog** package for the following 9 species:

- Human *Homo sapiens*
- Mouse *Mus musculus*
- Rat *Rattus norvegicus*
- Zebra fish *Danio rerio*
- Fruit fly *Drosophila melanogaster*
- Round worm *Caenorhabditis elegans*
- Maize *Zea mays*
- Thale cress *Arabidopsis thaliana*
- Yeast *Saccharomyces cerevisiae*

The files were generated using the folowing instructions:

1. Go to the Ensembl or Plant Ensembl website > Biomart
2. Choose database: "Ensembl genes 96"
3. Choose dataset: your species of interest e.g. "Mouse genes (GRCm38.p6)"
4. Choose Filters. Recommendation: GENE: Limit to genes (external references)… with CCDS ID(s) Only (if available for the target species) Gene type: protein_coding
   Transcript type: protein_coding
5. Choose Attributes. Recommendation: Sequences; SEQUENCES: cDNA sequences (do not specify any flanks); HEADER INFORMATION: (Important: check the boxes exactly in the order specified below) Gene stable ID Transcript stable ID Gene name CDS start (within cDNA) CDS end (within cDNA) Transcript length (including UTRs and CDS)
6. Press Results > Check Unique results only > Download to save the fasta file.
7. (Optional) Download the gtf file from ftp://ftp.ensembl.org/pub/release-96/gtf/
8. Run the script `Biomart_cDNA_fasta_to_rW_annotation_and_reheadered_longest_CDS_cDNA_fasta.py` with the following arguments: *fasta_in fasta_out annotation_out ID_mapper_out no_x_cds*

The *no_x_cds* argument must be given only if the 3' coordinate of CDS in the fasta header line corresponds to the last aminoacid (exludes the stop codon). This was true for the *Drosophila melanogaster* fasta file downloaded from Ensembl, for instance.

Examples:

```
$ python Biomart_cDNA_fasta_to_rW_annotation_and_reheadered_longest_CDS_cDNA_fasta.py Mouse_GR
Cm38.p6_cDNA.v1.txt Mouse_GRCm38_cDNA_longest_CDS.txt Mouse_GRCm38_annotation.txt Mouse_GRCm38
_ID_mapper.txt

$ python Biomart_cDNA_fasta_to_rW_annotation_and_reheadered_longest_CDS_cDNA_fasta.py Fly_BDGP
6.22.96_cDNA.v1.txt Fly_BDGP6.22.96_cDNA_longest_CDS.txt Fly_BDGP6.22.96_annotation.txt Fly_BD
GP6.22.96_ID_mapper.txt no_x_cds
```

RNA and RPF reads can be mapped to the transcriptome fasta using popular aligners e.g. bowtie2 or bwa. The fasta and annotation files will be used by several functions downstream. The ID mapper is provided to allow exploration of outcome by gene name at the end of analysis.

1.1. Read input files, calculate p-site offset and visualize periodicity

1.1.1. Read in the annotation file

Read the annotation from a .txt file into an R data table. The annotation file must have five columns named *transcript*, *l_tr*, *l_utr5*, *l_cds* and *l_utr3*. It lists the names, total lengths and lengths of segment (5' UTR, CDS and

3’ UTR) of transcripts in the reference file to which RPF reads were mapped. The output annotation data table will be used by several functions later on.

```
annotation_human_cDNA <- Ribolog::read_annotation("../data-raw/Human.GRC38.96_annotation.txt")
head(annotation_human_cDNA)

##         transcript l_tr l_utr5 l_cds l_utr3
## 1: ENST00000003084 6132    132  4443   1557
## 2: ENST00000001146 4732    204  1539   2989
## 3: ENST00000002125 2184     48  1326    810
## 4: ENST00000000233 1032     88   543    401
## 5: ENST00000002829 3607    289  2358    960
## 6: ENST00000001008 3715    170  1380   2165
```

1.1.2. Create a reads_list object from bam files

Read .bam files into a list of data frames. Each data frame contains reads information from one of the samples. The annotation data table produced by `read_annotation` is required to add CDS coordinates. Our sample dataset (LMCN) consists of 8 samples: four cell lines with two replicates each. Several functions borrowed from **riboWaltz** including `bamtolist_rW` print out progress messages that are suppressed here for brevity.

NOTE: All file and folder names should start with a letter (not a number, for instance) to avoid unexpected problems later.

```
reads_list_LMCN <- Ribolog::bamtolist_rW(bamfolder = "../data-raw/Bam/RPF",
    annotation = annotation_human_cDNA)

names(reads_list_LMCN)

## [1] "CN34_r1_rpf" "CN34_r2_rpf" "LM1a_r1_rpf" "LM1a_r2_rpf" "LM2_r1_rpf"
## [6] "LM2_r2_rpf"  "MDA_r1_rpf"  "MDA_r2_rpf"

head(reads_list_LMCN$CN34_r1_rpf)

##         transcript end5 end3 length cds_start cds_stop
## 1: ENST00000001146  150  169     20       205     1743
## 2: ENST00000001146  423  450     28       205     1743
## 3: ENST00000001146  457  478     22       205     1743
## 4: ENST00000001146  484  511     28       205     1743
## 5: ENST00000001146  493  513     21       205     1743
## 6: ENST00000001146  499  527     29       205     1743
```

1.1.3. Calculate p-site offset and create a reads_psite_list object

Run riboWlatz to estimate the most likely ribosomal p-site offsets for each read length group. Combine this information with the *reads_list* object produced by `bamtolist_rW` to create a *reads_psite_list* object. Each data frame in this list corresponds to one sample and contains the distance of the reads p-sites from the start and stop codons. It also shows whether each p-site falls in the 5’ UTR, CDs or 3’ UTR region.

```
psite_offset_LMCN <- Ribolog::psite_rW(reads_list_LMCN)
reads_psite_list_LMCN <- Ribolog::psite_info_rW(reads_list_LMCN,
    psite_offset_LMCN)

head(reads_psite_list_LMCN$CN34_r1_rpf)

##         transcript end5 psite end3 length cds_start cds_stop
## 1: ENST00000001146  150   160  169     20       205     1743
## 2: ENST00000001146  423   435  450     28       205     1743
## 3: ENST00000001146  457   466  478     22       205     1743
## 4: ENST00000001146  484   496  511     28       205     1743
## 5: ENST00000001146  493   502  513     21       205      1743
## 6: ENST00000001146  499   513  527     29       205      1743
##    psite_from_start psite_from_stop psite_region
## 1:              -45           -1583         5utr
## 2:              230           -1308          cds
## 3:              261           -1277          cds
## 4:              291           -1247          cds
## 5:              297           -1241          cds
## 6:              308           -1230          cds
```

1.1.4. Plot read length distribution, ribosome occupancy and periodicity and choose appropriate read length range

The *reads_psite_list* object produced by `psite_info_rW` is the key input used by the **CELP** method. Before proceeding to **CELP** correction, it is recommended to print out some QC plots, browse patterns and decide what read lengths will be included for further analysis. Chosen read lengths must be relatively abundant across samples and show proper periodicity in the CDS region. The code for several useful plot types and two example plots are included here.

```
Ribolog::print_read_ldist(reads_list_LMCN, "../LMCN_RPF_Read_length_distributions.pdf")

Ribolog::print_read_end_heatmap(reads_list_LMCN, annotation_human_cDNA,
    "../LMCN_RPF_read_end_heatmaps.pdf")

Ribolog::print_rop(reads_psite_list_LMCN, annotation_human_cDNA,
    "../RPF_ribosome_occupancy_profiles_from_annotation.pdf")

Ribolog::print_period_region(reads_psite_list_LMCN, "../Periodicity_by_region.pdf")

Ribolog::print_period_region_length(reads_psite_list_LMCN, "../Periodicity_by_length_region.pd
f")
```

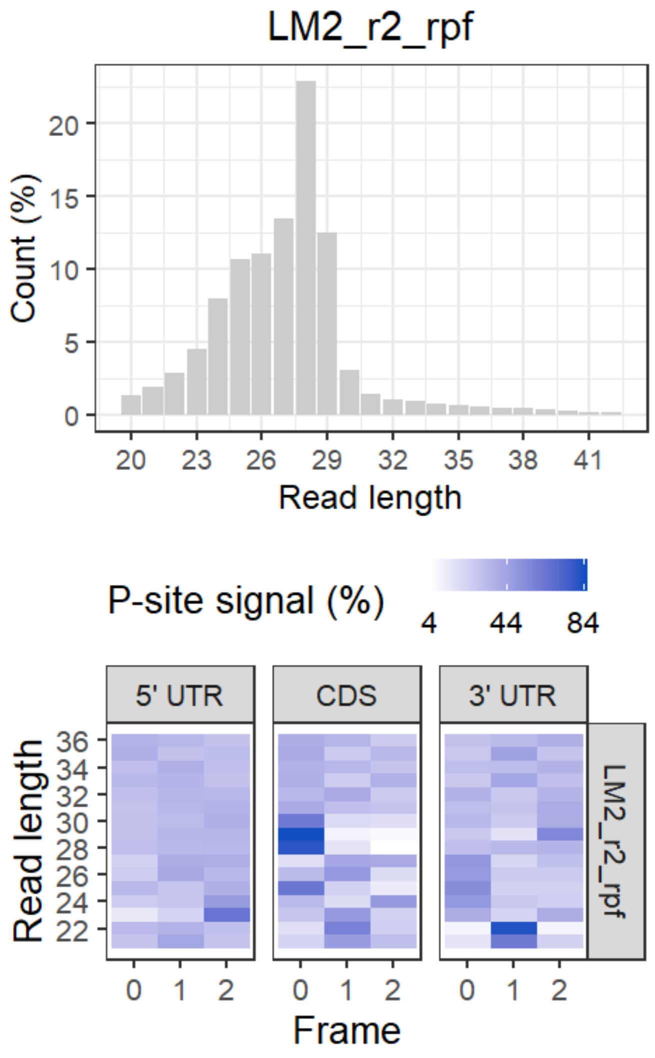


Figure 2. Read length distribution and periodicity plot for sample LM2_r2 of LMCN data set.

Suppose that after inspecting the plots across all samples we decide to move forward with reads in the length range of 24-32.

1.2. Generate codon read counts

The `psite_to_codon_count` function aggregates read p-site data to obtain codon read counts. It generates a tr_codon_read_count object which is a list of lists with the following structure: tr_codon_read_count$<sample.name>$<transcript.ID> data.frame: [1] codon_number [2] codon_type [3] aa_type [4] observed_count.

```r
tr_codon_read_count_LMCN <- Ribolog::psite_to_codon_count(reads_psite_list_LMCN,
    c(24:32), annotation_human_cDNA, "../data-raw/Human.GRC38.96_cDNA_longest_CDS.txt")
head(tr_codon_read_count_LMCN$CN34_r1_rpf$ENST00000000233, n = 30)
```

```
##    codon_number codon_type aa_type observed_count
## 1             1        ATG       M              0
## 2             2        GGC       G              0
## 3             3        CTC       L              0
## 4             4        ACC       T              0
## 5             5        GTG       V              0
## 6             6        TCC       S              0
## 7             7        GCG       A              0
## 8             8        CTC       L              0
## 9             9        TTT       F              1
## 10           10        TCG       S              1
## 11           11        CGG       R              1
## 12           12        ATC       I              1
## 13           13        TTC       F              0
## 14           14        GGG       G              3
## 15           15        AAG       K              1
## 16           16        AAG       K              0
## 17           17        CAG       Q              1
## 18           18        ATG       M              0
## 19           19        CGG       R              0
## 20           20        ATT       I              0
## 21           21        CTC       L              0
## 22           22        ATG       M              1
## 23           23        GTT       V              4
## 24           24        GGC       G              1
## 25           25        TTG       L              0
## 26           26        GAT       D              0
## 27           27        GCG       A              1
## 28           28        GCT       A              0
## 29           29        GGC       G              9
## 30           30        AAG       K              0
```

1.3. Run CELP

This is the main step in the **CELP** procedure. The `CELP_bias` function computes codon-level bias coefficients and bias-corrected read counts.

The procedure starts with running a loess curve on codon read counts along the transcript to borrow information from neighboring codons mitigating the uncertainty of p-site offset assignment and experimental stochasticity. Loess span parameter is calculated from the user-defined *codon_radius* (default=5) and CDS length. Then, bias coefficient is calculated for each codon by integrating information on the excess of loess-predicted read counts at that codon comapred to the transcript's background across samples. Finally, loess predicted count is divided by the bias coefficient to calculate the bias-corrected count. This function can be used in several modes (see function documentation for explanation of arguments). For example, the "direct" fitting method for loess takes longer but does not run into kd-tree-related memory issues. "Gini-moderated" correction ensures that the power of correction is proportional to the original level of heterogenity in read distribution along the transcript.

Codons with large bias coefficients are those with a consistent excess of reads across samples compared to the transcript background (reproducible peaks). They indicate translational stalling.

The `CELP_bias` function returns a list composed of two lists: [1] bias coefficients, and [2] bias-corrected read counts. The bias coefficient list has the following structure: list$<transcript.ID> data.frame: [1] codon_number [2] codon_type [3] aa_type [4] bias_coefficient. The bias-corrected read count list has the following structure: list$<sample.name>$<transcript.ID> data.frame: [1] codon_number [2] codon_type [3] aa_type [4] observed_count [5] bias_coefficient [6] corrected_count.

You can run CELP_bias with the default arguments:

```
tr_codon_bias_coeff_loess_corrected_count_LMCN <- Ribolog::CELP_bias(tr_codon_read_count_LMCN)

print((tr_codon_bias_coeff_loess_corrected_count_LMCN$tr_codon_read_count_loess_corrected$CN34
_r1_rpf$ENST00000000233)[c(30:49),
    ])
```

```
##    codon_number codon_type aa_type observed_count bias_coefficient
## 30           30        AAG       K              0        1.0049040
## 31           31        ACC       T              0        0.5313807
## 32           32        ACA       T              1        0.7907074
## 33           33        ATC       I              0        1.0000000
## 34           34        CTG       L              0        1.0000000
## 35           35        TAC       Y              6        1.0000000
## 36           36        AAA       K              1       10.7156229
## 37           37        CTG       L              1       24.5521153
## 38           38        AAG       K             61       35.7132009
## 39           39        TTG       L            103       32.7251422
## 40           40        GGG       G             14       24.5573561
## 41           41        GAG       E              1        7.4377293
## 42           42        ATT       I              1        1.9181915
## 43           43        GTC       V              0        0.5766887
## 44           44        ACC       T              0        0.9115349
## 45           45        ACC       T              1        0.5489206
## 46           46        ATC       I              0        0.4993800
## 47           47        CCA       P              0        0.6708692
## 48           48        ACC       T              1        0.7333517
## 49           49        ATA       I              0        1.0000000
##    corrected_count
## 30       2.1898191
## 31       2.1983444
## 32       0.0000000
## 33       0.0000000
## 34       0.0000000
## 35       0.0000000
## 36       1.1926788
## 37       1.2553613
## 38       1.3790503
## 39       1.4863645
## 40       1.6501106
## 41       2.1942565
## 42       2.0576454
## 43       0.0000000
## 44       0.0000000
## 45       0.0000000
## 46       0.7249453
## 47       0.5201916
## 48       0.4044470
## 49       0.0000000
```

A similar function `CELP_detect_bias` performs the first part of caculations and outputs the list containing bias coefficients but not bias-corrected RPF counts in individual samples. The purpose of this function is to provide a more concise output for studies of translational dynamics where codon decoding time or ribosome dwell time (proportional to bias coefficient) is the primary outcome of interest, not transcript-level differential translation rates.

1.4. Visualize translational bias

The function `visualize_CELP` plots codon-level observed (upward black bars) and corrected (downward purple bars) read counts and the bias coefficient (red line) along the transcript. This allows visual inspection of the prominent bias positions and a comparison of read count heterogeneity along the transcript before and after CELP bias correction. If *outfile* is not specified, plots are printed to the standard output (the Files/Plots/Packages/Help panel in Rstudio).

```
Ribolog::visualize_CELP(tr_codon_bias_coeff_loess_corrected_count_LMCN$tr_codon_read_count_loe
ss_corrected,
    transcript = "ENST00000000233", panel_rows = 2, panel_cols = 4)
```

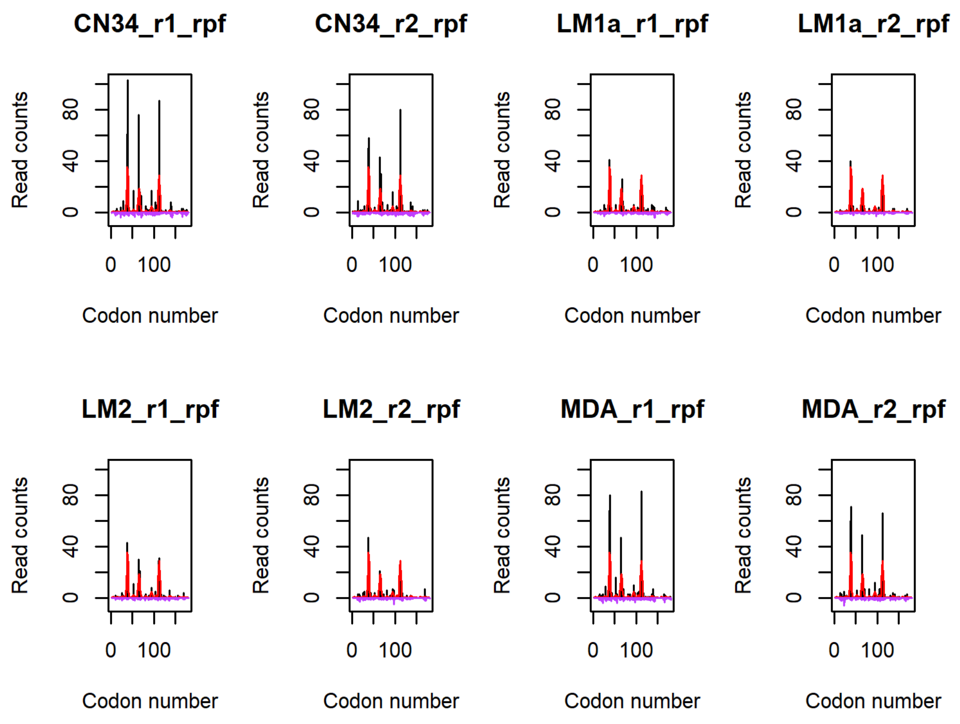


Figure 3. CELP bias coefficients (red line), observed read counts (black bars) and CELP-corrected read counts (purple bars) for transcript ENST00000000233 in all the samples in the LMCN dataset.

It is possible to choose one or a few particular samples to plot:

```
Ribolog::visualize_CELP(tr_codon_bias_coeff_loess_corrected_count_LMCN$tr_codon_read_count_loe
ss_corrected["CN34_r1_rpf"],
    transcript = "ENST00000000233", panel_rows = 1, panel_cols = 1)
```

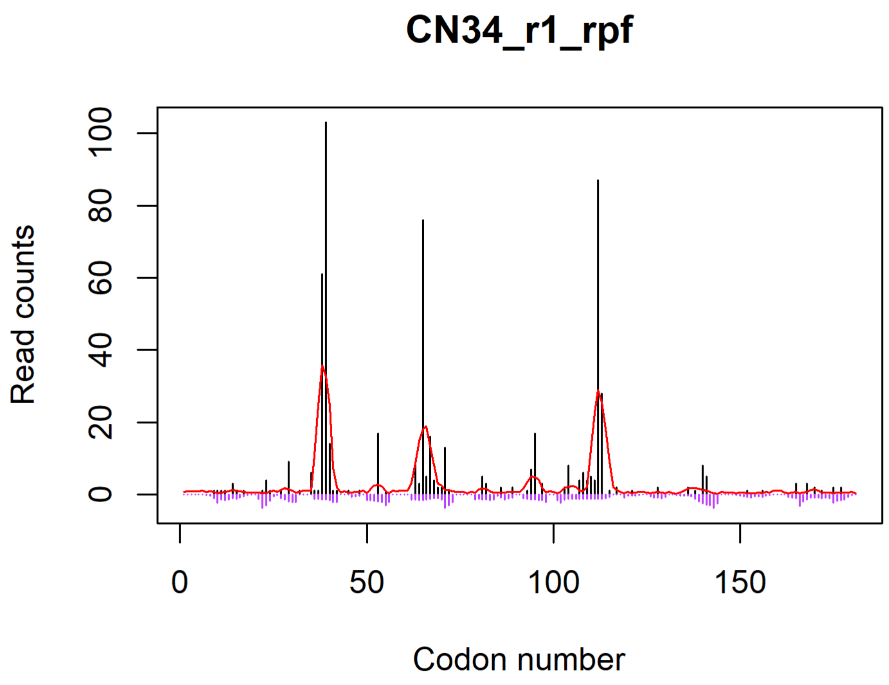


Figure 4. CELP bias coefficients (red line), observed read counts (black bars) and CELP-corrected read counts (purple bars) for transcript ENST00000000233 in sample CN34_r1.

Corrected read counts are shown in the downward direction for better visibility but obviously they do not represent negative values. Some codons have zero observed but non-zero corrected counts. This happens as a result of running the `loess` function on codon counts; zero-count codons can have non-zero loess-predicted values if there is a non-zero count codon nearby.

The range of plotted codons can be controlled to zoom in or avoid overcrowding in the case of long transcripts.

```
Ribolog::visualize_CELP(tr_codon_bias_coeff_loess_corrected_count_LMCN$tr_codon_read_count_loe
ss_corrected["CN34_r1_rpf"],
    transcript = "ENST00000367255", from_codon = 3500, to_codon = 3800)
```

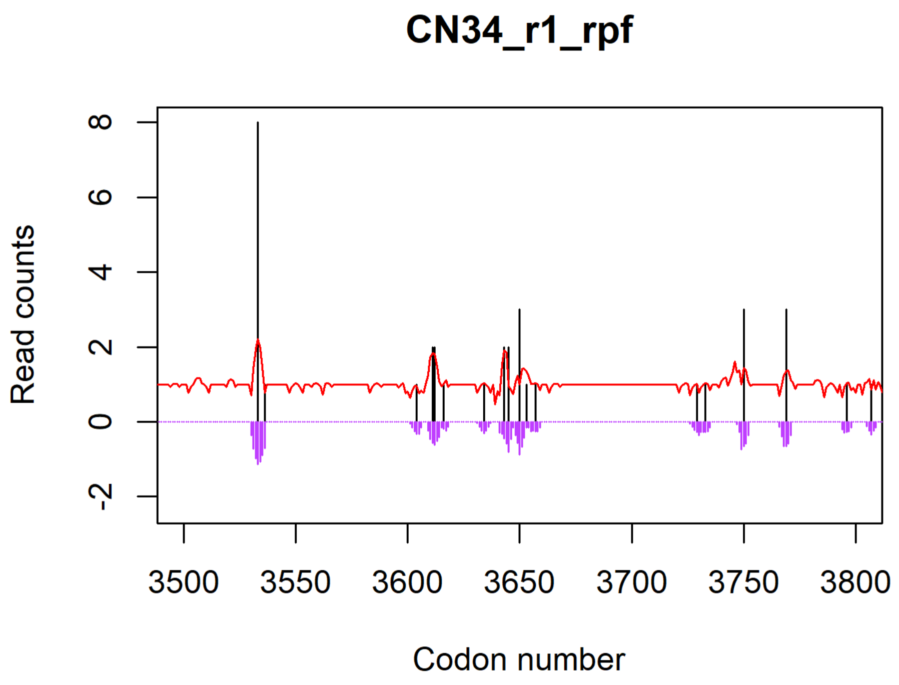


Figure 5. CELP bias coefficients (red line), observed read counts (black bars) and CELP-corrected read counts (purple bars) for transcript ENST00000367255, codons 3500-3800 in sample CN34_r1.

1.5. Generate transcript read counts

Observed or corrected codon read counts are summed up to produce transcript read counts. The analysis of translational efficiency is usually performed at transcript level.

```
rpf_observed_sum_LMCN <- Ribolog::codon2transcript(tr_codon_bias_coeff_loess_corrected_count_L
MCN$tr_codon_read_count_loess_corrected,
    count.type = "observed_count")
head(rpf_observed_sum_LMCN)

##        transcript CN34_r1_rpf CN34_r2_rpf LM1a_r1_rpf LM1a_r2_rpf
## 1 ENST00000000233         590         533         284         223
## 2 ENST00000000412         217         161          87          65
## 3 ENST00000000442          60          42          26          21
## 4 ENST00000001008         635         597         477         390
## 5 ENST00000001146          11           5           9           8
## 6 ENST00000002125          43          28           3          11
##   LM2_r1_rpf LM2_r2_rpf MDA_r1_rpf MDA_r2_rpf
## 1        321        298        502        477
## 2        106         81        173        137
## 3         34         26         40         36
## 4        476        493        540        520
## 5          7          8          5          3
## 6          8         17         23         21

rpf_corrected_sum_LMCN <- Ribolog::codon2transcript(tr_codon_bias_coeff_loess_corrected_count_
LMCN$tr_codon_read_count_loess_corrected,
    count.type = "corrected_count")
head(rpf_corrected_sum_LMCN)

##        transcript CN34_r1_rpf CN34_r2_rpf LM1a_r1_rpf LM1a_r2_rpf
## 1 ENST00000000233   176.30289  166.150805  103.883724   63.769528
## 2 ENST00000000412   133.18892  104.704272   50.074178   42.573252
## 3 ENST00000000442    53.57249   38.308924   24.045604   18.749959
## 4 ENST00000001008   318.88989  306.586381  206.258916  165.199522
## 5 ENST00000001146    12.12084    5.439828   10.089169    8.209649
## 6 ENST00000002125    39.27427   28.081875    2.601816    9.482828
##   LM2_r1_rpf LM2_r2_rpf MDA_r1_rpf MDA_r2_rpf
## 1  89.241289  96.336317 145.892079 126.287780
## 2  61.274603  55.723701 101.529527  80.797405
## 3  34.485613  27.158902  34.313709  28.176598
## 4 207.981666 221.579566 239.968214 217.801952
## 5   7.633892   8.176429   5.126584   3.107706
## 6   8.273504  16.590193  19.710282  18.740335
```

Module 2: PREP

Preparing the data set for quality control and analysis

The **PREP** module includes functions to import, merge and normalize RNA and RPF data. The resulting data set will be ready to be passed on to module 3 for quality control or module 4 for analysis of differential translational efficiency.

2.1. Create a merged RNA-RPF data set

The steps for creating a merged RNA-RPF data set differs based on whether or not bias correction using the CELP method (module 1) is being performed.

2.1.A Without CELP correction

Create the annotation data table:

```
annotation_human_cDNA <- Ribolog::read_annotation("../data-raw/Human.GRC38.96_annotation.txt")
head(annotation_human_cDNA)

##         transcript l_tr l_utr5 l_cds l_utr3
## 1: ENST00000003084 6132    132  4443   1557
## 2: ENST00000001146 4732    204  1539   2989
## 3: ENST00000002125 2184     48  1326    810
## 4: ENST00000000233 1032     88   543    401
## 5: ENST00000002829 3607    289  2358    960
## 6: ENST00000001008 3715    170  1380   2165
```

All the bam files (RNA and RPF) can be placed in the same folder and imported together, or imported separately and then merged:

```
rna_count_LMCN <- Ribolog::bam2count(bamfolder = "../data-raw/Bam/RNA",
    annotation = annotation_human_cDNA)
rpf_count_LMCN <- Ribolog::bam2count(bamfolder = "../data-raw/Bam/RPF",
    annotation = annotation_human_cDNA)

rna_rpf_count_LMCN <- merge(rna_count_LMCN, rpf_count_LMCN, by = "transcript")
head(rna_rpf_count_LMCN)

##        transcript CN34_r1_rna CN34_r2_rna LM1a_r1_rna LM1a_r2_rna
## 1 ENST00000000233         389         386         421         690
## 2 ENST00000000412        1833        1812        2393        3550
## 3 ENST00000000442         734         721         760        1181
## 4 ENST00000001008        1741        1602        2032        2965
## 5 ENST00000001146          77          73          38          94
## 6 ENST00000002125         138         132         143         230
##   LM2_r1_rna LM2_r2_rna MDA_r1_rna MDA_r2_rna CN34_r1_rpf CN34_r2_rpf
## 1        456        882        820        756         742         676
## 2       2493       5121       2772       2593         317         250
## 3        803       1489       1432       1402          98          80
## 4       2094       4214       2606       2478         936         867
## 5         62        148         71         61          17          11
## 6        176        344        282        233          58          37
##   LM1a_r1_rpf LM1a_r2_rpf LM2_r1_rpf LM2_r2_rpf MDA_r1_rpf MDA_r2_rpf
## 1         307         254        338        322        598        564
## 2         119         104        135        118        246        213
## 3          42          41         51         37         76         56
## 4         582         479        576        605        748        702
## 5          15           9          8          9         14          5
## 6           9          14         16         19         30         24
```

The bam2count function is designed to work on bam files generated by mapping to a transcriptome (not a genome). For each sample, it counts the number of reads mapping to different chromosomes ('seqname's in bam fields). Annotation must be provided to ensure a complete transcript list, including those with zero counts. Low count transcripts can be filtered out later using the min_count_filter function (covered in module 3). If reads were mapped to a genome, counting should be performed outside **Ribolog** to produce a similar data frame. It can be then passed on to *STEP 2*.

2.1.B With CELP correction

Create the RNA counts data frame using the bam2count function. Then, merge with the corrected RPF counts data frame produced by the codon2transcript function (covered in module 1).

```
rna_count_LMCN <- Ribolog::bam2count(bamfolder = "../data-raw/Bam/RNA",
    annotation = annotation_human_cDNA)

rna_CELP_rpf_count_LMCN <- merge(rna_count_LMCN, rpf_corrected_sum_LMCN,
    by = "transcript")
head(rna_CELP_rpf_count_LMCN)

##        transcript CN34_r1_rna CN34_r2_rna LM1a_r1_rna LM1a_r2_rna
## 1 ENST00000000233         389         386         421         690
## 2 ENST00000000412        1833        1812        2393        3550
## 3 ENST00000000442         734         721         760        1181
## 4 ENST00000001008        1741        1602        2032        2965
## 5 ENST00000001146          77          73          38          94
## 6 ENST00000002125         138         132         143         230
##   LM2_r1_rna LM2_r2_rna MDA_r1_rna MDA_r2_rna CN34_r1_rpf CN34_r2_rpf
## 1        456        882        820        756   176.30289  166.150805
## 2       2493       5121       2772       2593   133.18892  104.704272
## 3        803       1489       1432       1402    53.57249   38.308924
## 4       2094       4214       2606       2478   318.88989  306.586381
## 5         62        148         71         61    12.12084    5.439828
## 6        176        344        282        233    39.27427   28.081875
##   LM1a_r1_rpf LM1a_r2_rpf LM2_r1_rpf LM2_r2_rpf MDA_r1_rpf MDA_r2_rpf
## 1  103.883724   63.769528  89.241289  96.336317 145.892079 126.287780
## 2   50.074178   42.573252  61.274603  55.723701 101.529527  80.797405
## 3   24.045604   18.749959  34.485613  27.158902  34.313709  28.176598
## 4  206.258916  165.199522 207.981666 221.579566 239.968214 217.801952
## 5   10.089169    8.209649   7.633892   8.176429   5.126584   3.107706
## 6    2.601816    9.482828   8.273504  16.590193  19.710282  18.740335
```

Note: The corrected RPF data frame may contain slightly fewer transcripts than the output of `bam2count`. This is because transcripts with CDS length non-divisible by 3 are removed during CELP correction. Only transcripts with data in both data frames are retained after merging.

Note: The RPF counts from `bam2count` are expected to be higher than even the *uncorrected* RPF counts produced by the `codon2transcript` function (module 1). The reason is that reads are filtered based on length and mapping to CDS (eliminating those mapping to 5' UTR or 3' UTR) in the CELP procedure regardless of application of the bias correction itself.

2.2. Normalize counts for library size variation

By default, we ue the median-of-ratios method for library size normalization. Users can normalize their data in any other way they prefer outside **Ribolog** and pass on the result to modules 3 and 4. RNA data and RPF data must be normalized separately. The `normalize_median_of_ratios` function allows specification of data columns, and thus, indepedent normalization of RPF and RNA columns in a mixed RNA-RPF data set.

```
# Normalize RNA counts
rna_CELP_rpf_count_norm1_LMCN <- Ribolog::normalize_median_of_ratios(rna_CELP_rpf_count_LMCN,
    c(2:9))

## [1] "Normalization factors:"
## CN34_r1_rna CN34_r2_rna LM1a_r1_rna LM1a_r2_rna  LM2_r1_rna  LM2_r2_rna
##   0.7187785   0.6950616   0.8665587   1.2848121   0.8762268   1.7817562
##  MDA_r1_rna  MDA_r2_rna
##   1.2153191   1.1543571

# Normalize RPF counts
rna_CELP_rpf_count_norm2_LMCN <- Ribolog::normalize_median_of_ratios(rna_CELP_rpf_count_norm1_
LMCN,
    c(10:17))

## [1] "Normalization factors:"
## CN34_r1_rpf CN34_r2_rpf LM1a_r1_rpf LM1a_r2_rpf  LM2_r1_rpf  LM2_r2_rpf
##   1.6682204   1.5719904   0.7983419   0.7123556   0.8862191   0.8991526
##  MDA_r1_rpf  MDA_r2_rpf
##   1.2453150   1.0327013
```

To be concise, we rename the bias-corrected merged and normalized RNA-RPF data frame `rna_CELP_rpf_count_norm2_LMCN` to `rr_LMCN`.

```
rr_LMCN <- rna_CELP_rpf_count_norm2_LMCN
head(rr_LMCN)

##        transcript CN34_r1_rna CN34_r2_rna LM1a_r1_rna LM1a_r2_rna
## 1 ENST00000000233    541.1959    555.3465   485.82975   537.04350
## 2 ENST00000000412   2550.1598   2606.9633  2761.49782  2763.04991
## 3 ENST00000000442   1021.1769   1037.3182   877.03232   919.20055
## 4 ENST00000001008   2422.1649   2304.8318  2344.90747  2307.73042
## 5 ENST00000001146    107.1262    105.0267    43.85162    73.16245
## 6 ENST00000002125    191.9924    189.9112   165.02055   179.01450
##   LM2_r1_rna LM2_r2_rna MDA_r1_rna MDA_r2_rna CN34_r1_rpf CN34_r2_rpf
## 1  520.41321  495.01723  674.71993  654.90999  105.683213  105.694545
## 2 2845.15383 2874.13065 2280.88249 2246.27195   79.838919   66.606180
## 3  916.42941  835.69235 1178.29139 1214.52884   32.113554   24.369694
## 4 2389.79226 2365.08232 2144.29285 2146.64940  191.155731  195.030702
## 5   70.75794   83.06412   58.42087   52.84327    7.265732    3.460472
## 6  200.86124  193.06794  232.03783  201.84395   23.542616   17.863898
##   LM1a_r1_rpf LM1a_r2_rpf LM2_r1_rpf LM2_r2_rpf MDA_r1_rpf MDA_r2_rpf
## 1  130.124360    89.51924 100.698905 107.141235 117.152749 122.288775
## 2   62.722726    59.76405  69.141599  61.973577  81.529191  78.238890
## 3   30.119433    26.32107  38.913192  30.204999  27.554240  27.284363
## 4  258.359140   231.90599 234.684263 246.431556 192.696795 210.905078
## 5   12.637656    11.52465   8.614001   9.093483   4.116696   3.009298
## 6    3.259024    13.31193   9.335733  18.450922  15.827547  18.146907
```

The RNA-RPF merged and normalized data set is now ready for quality control.

Module 3: QC

Quality control of ribosome profiling data

The **QC** module includes functions to check the quality of ribosome profiling data focusing mainly on the reproducibility of translational efficiency (TE) among replicates. Tools for general QC of sequencing data (e.g. FASTQC) or 3-base periodicity of ribo-seq libraries (borrowed from the **riboWaltz** package and covered in module 1) are not repeated here.

Three ribosome-profiling-specific QC tools are provided in this module:

QC Test	Multiple replicates per sample	Before or After TER significance test
PCA of RNA, RPF and TE	Preferred	Before
Proportion of null features	Preferred	After
Correlation of equivalent TER tests	Required	After

3.1. Principal component analysis (PCA) of RNA, RPF and translational efficiency

The aim of performing PCA on a ribosome profiling data set is to check whether replicates of the same biological state or sample cluster together. In unreplicated datasets, one can check whether conditions that are expected to be more similar occupy nearby spots. This can be done on (normalized) RNA counts, RPF counts or their ratio known as translational efficiency $TE = \frac{RPF}{RNA}$ which is the main output of interest in ribosome profiling experiments.

3.1.1. Filter out low count transcripts

A minimum read count cutoff is usually applied to sequencing data to remove extremely low count features which would yield unreliable results. To demonstrate, we filter our dataset to keep only transcripts with RNA>=5 in all samples and *average* RPF>=2 across samples. This can be done using the `min_count_filter` function. You can choose methods between "all" or "average" and apply different cutoffs until you find a value setting that produces acceptable output in terms of replicate consistency.

Note: Replicate consistency is a necessary QC condition, not a sufficient one, because it reflects the collective behavior of all transcripts. As most studies aim to draw conclusions about individual transcripts or genes, extremely low count features should be avoided even if they do not diminish reproducibility among replicates.

```
# Filter for RNA>=5 in all samples
rr_LMCN.v1 <- Ribolog::min_count_filter(rr_LMCN, mincount = 5,
    columns = c(2:9), method = "all")
# Filter for average RPF>=2 across samples
rr_LMCN.v2 <- Ribolog::min_count_filter(rr_LMCN.v1, mincount = 2,
    columns = c(10:17), method = "average")
dim(rr_LMCN)

## [1] 19037    17

dim(rr_LMCN.v1)

## [1] 12473    17

dim(rr_LMCN.v2)

## [1] 11182    17
```

3.1.2. Standardize the data

PCA is a statistical procedure that decomposes the total variance in data to several orthogonal (not linearly correlated) components. The variance in a *transcript x sample* matrix of read counts originates not only from differences among samples, but also from differences of (mean) counts across transcripts (there is also an interaction term but we will not deal with that just now). The output of PCA is sensitive to the scale of input numbers. If PCA is performed on the raw RNA counts, the pattern will be driven disproportionately by highly expressed genes. The same is true about RPF counts or TE values and highly translated transcripts. It is therefore customary in some applications to center or standardize the data before performing the PCA. Centering the data means subtracting the row mean or column mean from each element. Standardization means dividing the centered data by the corresponding row or column standard deviation. The choice of row- or column- standardization depends on the data type and structure, and the aim of the study. We perform PCA to visualize similarities and distances among samples. Row centering will bring the mean count for each transcript to zero which means that the differences in the average read counts of transctipts will not be incorporated into the total variance. Row standardization accomplishes the same goal but also guranatees that each transcript adds exactly one unit of variance to the variance of the *transcript x sample* matrix. Thus, all transctipts contribute equally to the PCA pattern.

First, we need to create a TE data set by dividing RPF columns by their RNA counterparts from the same sample. This is done using the `create-te` function:

```
te_LMCN.v2 <- Ribolog::create_te(rr_LMCN.v2, idcolumns = 1, rnacolumns = c(2:9),
    rpfcolumns = c(10:17))
head(te_LMCN.v2)

##        transcript CN34_r1_te CN34_r2_te LM1a_r1_te LM1a_r2_te  LM2_r1_te
## 1 ENST00000000233 0.19527717 0.19032181 0.26783943 0.16668899 0.19349798
## 2 ENST00000000412 0.03130742 0.02554934 0.02271330 0.02162974 0.02430153
## 3 ENST00000000442 0.03144759 0.02349298 0.03434244 0.02863474 0.04246175
## 4 ENST00000001008 0.07891937 0.08461819 0.11017882 0.10049094 0.09820279
## 5 ENST00000001146 0.06782405 0.03294851 0.28819133 0.15752139 0.12173900
## 6 ENST00000002125 0.12262265 0.09406446 0.01974920 0.07436231 0.04647852
##    LM2_r2_te  MDA_r1_te  MDA_r2_te
## 1 0.21643941 0.17363167 0.18672608
## 2 0.02156255 0.03574458 0.03483055
## 3 0.03614368 0.02338491 0.02246498
## 4 0.10419576 0.08986496 0.09824850
## 5 0.10947547 0.07046619 0.05694762
## 6 0.09556699 0.06821106 0.08990563
```

Note: The order of samples must be the same in RNA and RPF columns.

Next step is centering or standardizing the data. We demonstrate this for the TE parameter, but the same procedure can be performed on RNA and RPF counts. Needless to say, RNA and RPF counts must be centered or standardized separately using the `columns` argument.

```
te_LMCN.v2.cent <- row_center(te_LMCN.v2, columns = c(2:9))
te_LMCN.v2.stnd <- row_standardize(te_LMCN.v2, columns = c(2:9))
```

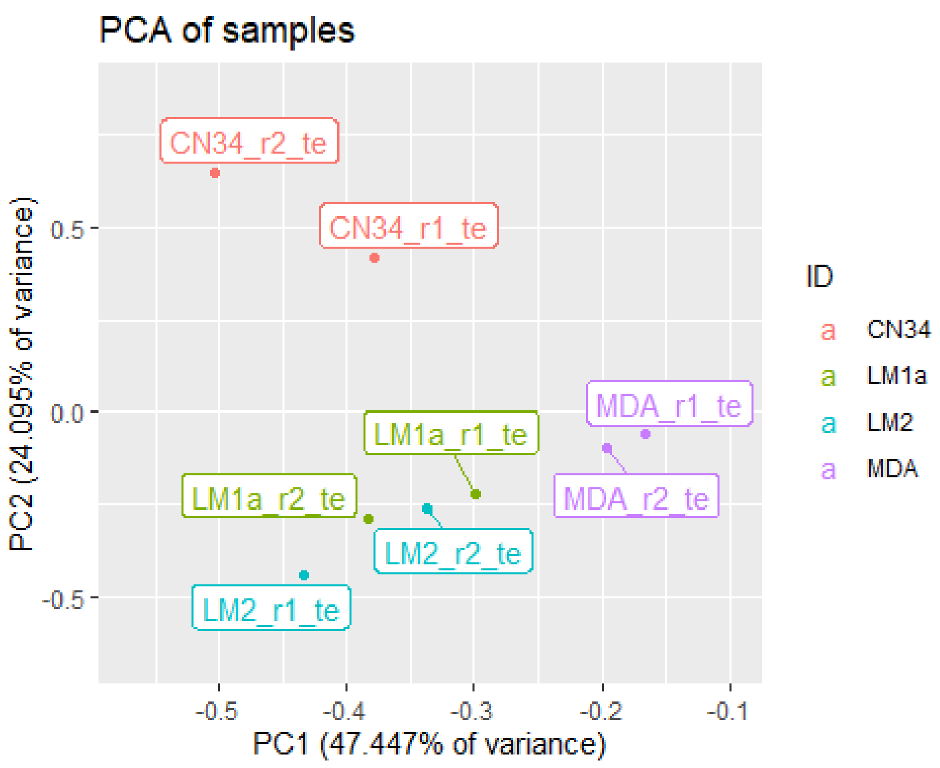
How should one decide whether to run PCA on the raw, centered or standandardized data set? Here is a general guideline: If you would like all genes/transcripts to weigh in equally, use a standardized or centered data set. If you would like to give more weight to highly expressed or translated genes/transcripts, do not center or standandardize.

NOTE: Centering and standardization are performed merely for visulization, not translational efficiency ratio (TER) analysis. The input to the `logit_seq` function which performs that task (module 4) is a data set that has been normalized and filtered to remove low-count transcripts, but not centered or standardized.

3.1.3. Produce the PCA plots

Only the numerical part of the data set is fed into the `pca_qc` function which means that the ID column(s) must be manually excluded. The argument `n` specifies the number of PCs to be plotted. Below, we compare the PCA pattern from the original, low-count removed, row-centered and row-standardized datasets:

```
# The orignial data set containing low count transcripts
# (infinite values generated by division by zero must be
# removed before PCA)
te_LMCN <- Ribolog::create_te(rr_LMCN, idcolumns = 1, rnacolumns = c(2:9),
    rpfcolumns = c(10:17))
te_LMCN.fin <- te_LMCN[is.finite(rowSums(te_LMCN[, -1])), ]
sample_attributes_LMCN <- read.xlsx("../data-raw/sample_attributes_LMCN.xlsx",
    sheetIndex = 1, header = TRUE)
Ribolog::pca_qc(te_LMCN.fin[, -1], n = 2, ID = sample_attributes_LMCN$cell_line[c(1:8)])

## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     0.3499 0.2493 0.15415 0.11642 0.11083 0.09582
## Proportion of Variance 0.4745 0.2409 0.09209 0.05253 0.04761 0.03559
## Cumulative Proportion  0.4745 0.7154 0.80752 0.86005 0.90766 0.94325
##                           PC7     PC8
## Standard deviation     0.0931 0.07729
## Proportion of Variance 0.0336 0.02315
## Cumulative Proportion  0.9768 1.00000
```

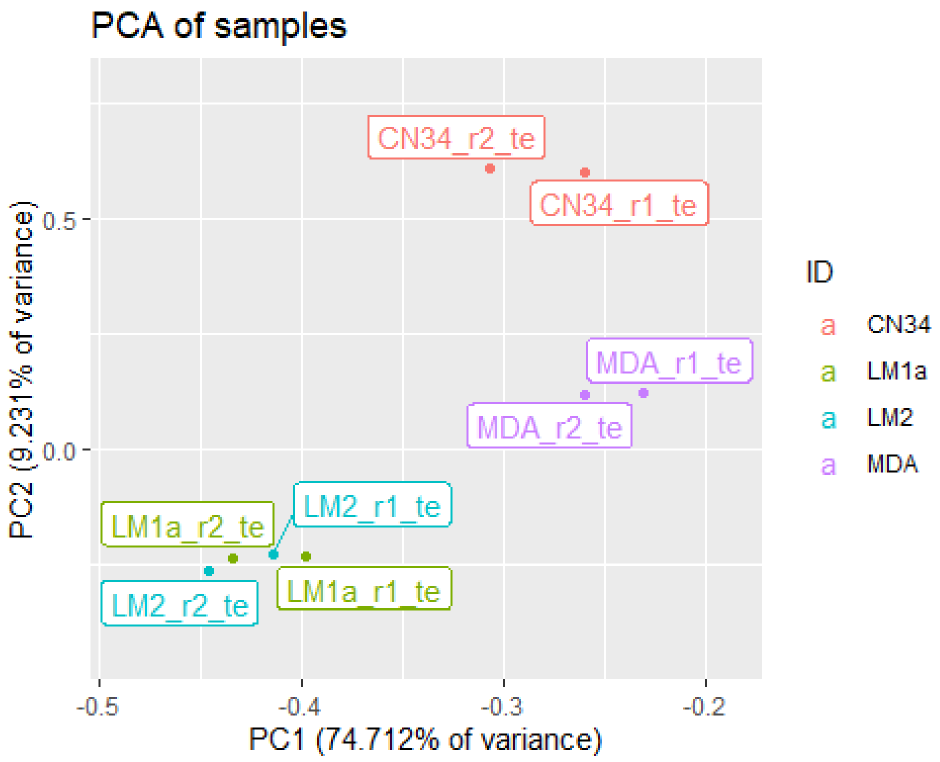


The optional `ID` argument of the `pca_qc` function is a vector used to color-code the samples on the PCA plot. Each element of this vector provides the ID value of the corresponding sample in the input data (argument `x`, *te_LMCN.fin* here) in the same order. Samples with the same `ID` value will be colored the same. It is often convenient to obtain this vector from an appropriate variable in the design matrix which describes the attributes of samples in the dataset, *sample_attributes_LMCN$cell_line* here. Only the first 8 elements are included because the next 8 elements are their exact duplicates (first 8 elements describe the RNA samples and second 8 elements describe corresponding RPF samples).

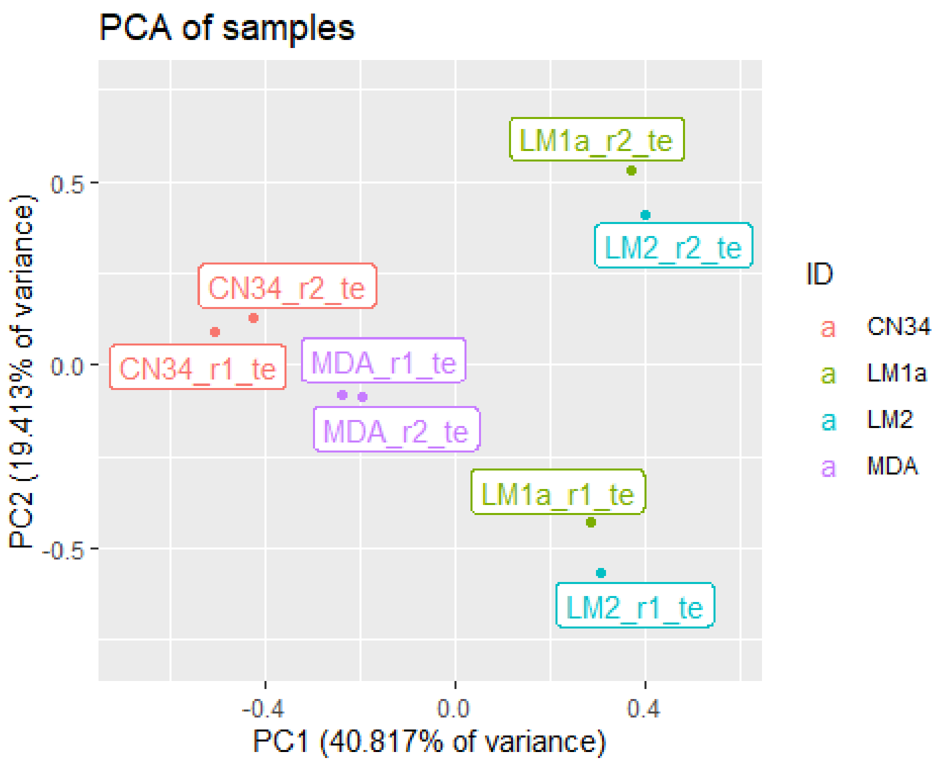
```
print(sample_attributes_LMCN)
```

```
##    sample_name read_type lung_metastasis cell_line replicate_no
## 1  CN34_r1_rna       RNA               N      CN34            1
## 2  CN34_r2_rna       RNA               N      CN34            2
## 3  LM1a_r1_rna       RNA               Y      LM1a            1
## 4  LM1a_r2_rna       RNA               Y      LM1a            2
## 5   LM2_r1_rna       RNA               Y       LM2            1
## 6   LM2_r2_rna       RNA               Y       LM2            2
## 7   MDA_r1_rna       RNA               N       MDA            1
## 8   MDA_r2_rna       RNA               N       MDA            2
## 9  CN34_r1_rpf       RPF               N      CN34            1
## 10 CN34_r2_rpf       RPF               N      CN34            2
## 11 LM1a_r1_rpf       RPF               Y      LM1a            1
## 12 LM1a_r2_rpf       RPF               Y      LM1a            2
## 13  LM2_r1_rpf       RPF               Y       LM2            1
## 14  LM2_r2_rpf       RPF               Y       LM2            2
## 15  MDA_r1_rpf       RPF               N       MDA            1
## 16  MDA_r2_rpf       RPF               N       MDA            2
##    replicate_name cell_line_origin
## 1         CN34_r1             CN34
## 2         CN34_r2             CN34
## 3         LM1a_r1             CN34
## 4         LM1a_r2             CN34
## 5          LM2_r1              MDA
## 6          LM2_r2              MDA
## 7          MDA_r1              MDA
## 8          MDA_r2              MDA
## 9         CN34_r1             CN34
## 10        CN34_r2             CN34
## 11        LM1a_r1             CN34
## 12        LM1a_r2             CN34
## 13         LM2_r1              MDA
## 14         LM2_r2              MDA
## 15         MDA_r1              MDA
## 16         MDA_r2              MDA
```

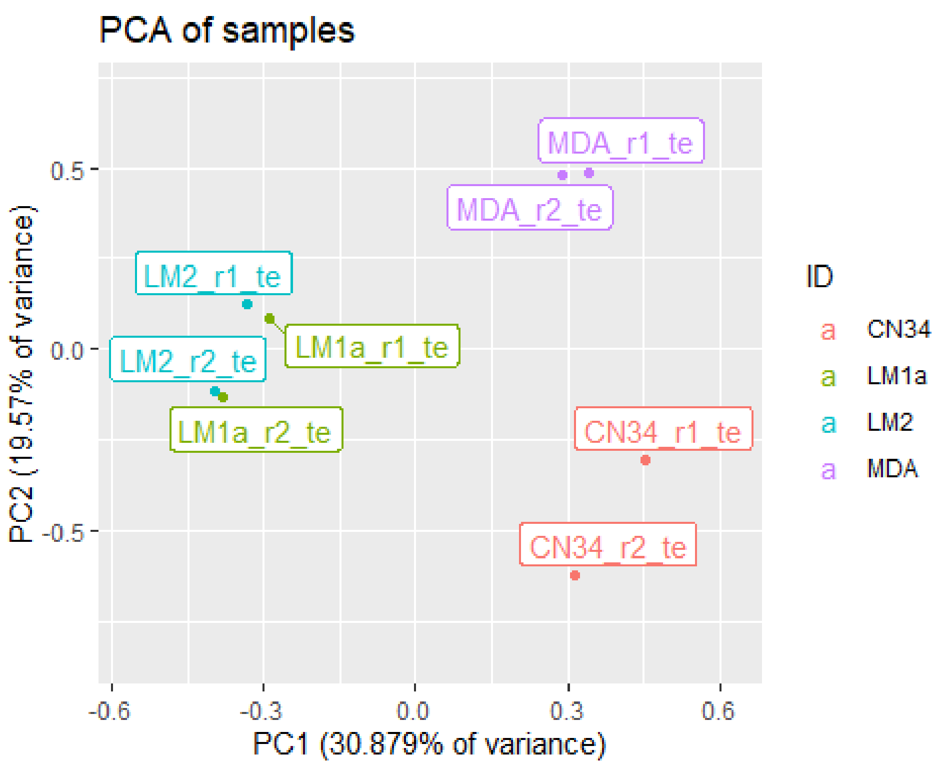
```
# Low count transcripts filtered out.
Ribolog::pca_qc(te_LMCN.v2[, -1], n = 2, ID = sample_attributes_LMCN$cell_line[c(1:8)])

## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     0.2365 0.08315 0.06455 0.05106 0.04157 0.03783
## Proportion of Variance 0.7471 0.09231 0.05563 0.03481 0.02307 0.01911
## Cumulative Proportion  0.7471 0.83944 0.89507 0.92989 0.95295 0.97206
##                            PC7     PC8
## Standard deviation     0.03574 0.02855
## Proportion of Variance 0.01706 0.01088
## Cumulative Proportion  0.98912 1.00000
```



```
# Low count transcripts filtered out, data row-centered.
Ribolog::pca_qc(te_LMCN.v2.cent[, -1], n = 2, ID = sample_attributes_LMCN$cell_line[c(1:8)])

## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     0.09413 0.06492 0.05786 0.04157 0.03798 0.03575
## Proportion of Variance 0.40817 0.19413 0.15421 0.07960 0.06644 0.05889
## Cumulative Proportion  0.40817 0.60230 0.75651 0.83611 0.90255 0.96144
##                            PC7       PC8
## Standard deviation     0.02893 1.116e-16
## Proportion of Variance 0.03856 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00
```



```
# Low count transcripts filtered out, data row-standardized.
Ribolog::pca_qc(te_LMCN.v2.stnd[, -1], n = 2, ID = sample_attributes_LMCN$cell_line[c(1:8)])

## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.4699 1.1702 1.0638 0.8474 0.76593 0.75775 0.67547
## Proportion of Variance 0.3088 0.1957 0.1618 0.1026 0.08385 0.08206 0.06521
## Cumulative Proportion  0.3088 0.5045 0.6662 0.7689 0.85273 0.93479 1.00000
##                            PC8
## Standard deviation     4.13e-15
## Proportion of Variance 0.00e+00
## Cumulative Proportion  1.00e+00
```



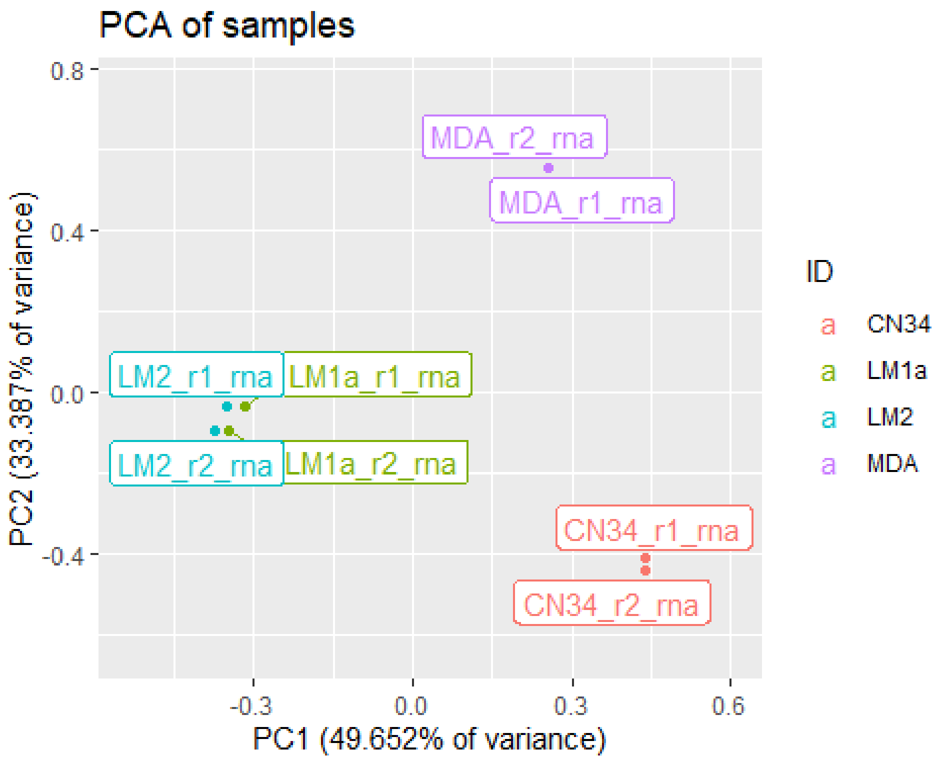
Here are some observations from the above plots:

- The two biological replicates of the non-metastatic lines (CN34 and MDA) cluster together. Reps of the metastatic lines (LM1a and LM2) do not behave so regularly.
- Filtering out low count transcripts and centering or standardization remarkably improve the distinction among cell lines and co-clustering of reps.
- In the filtered and standardized data set, PC1 which explains ~31% of the total variance clearly separates the two metastatic cell lines from the two non-metastatic cell lines.

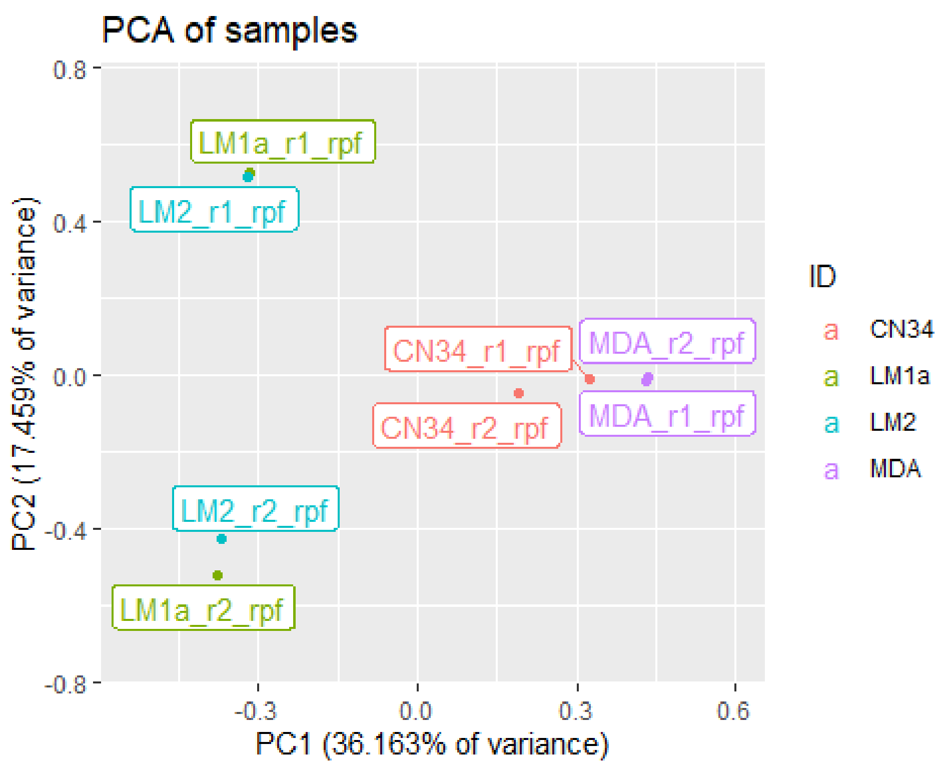
Even if samples were not sequenced in duplicates, we could still see that metastatic state was a more important determinant of translational landscape than the cell line's origin because the first PC separated metastatic samples from non-metastatic ones. This is the sort of biological insight gleaned from PCA analysis beyond replicate consistency.

To further investigate why TE of the LM1a and LM2 reps seem somewhat mismacthed, we repeat the PCA on RNA and RPF data:

```
# Standardize RNA counts
rr_LMCN.v3 <- row_standardize(rr_LMCN.v2, columns = c(2:9))
# Standardize RPF counts
rr_LMCN.v4 <- row_standardize(rr_LMCN.v3, columns = c(10:17))
# PCA on RNA counts
Ribolog::pca_qc(rr_LMCN.v4[, c(2:9)], n = 2, ID = sample_attributes_LMCN$cell_line[c(1:8)])

## Importance of components:
##                           PC1    PC2     PC3    PC4     PC5     PC6
## Standard deviation     1.8630 1.5276 0.60602 0.5147 0.48930 0.41338
## Proportion of Variance 0.4965 0.3339 0.05254 0.0379 0.03425 0.02445
## Cumulative Proportion  0.4965 0.8304 0.88293 0.9208 0.95509 0.97953
##                            PC7       PC8
## Standard deviation     0.37824 3.256e-15
## Proportion of Variance 0.02047 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00
```



```
# PCA on RPF counts
Ribolog::pca_qc(rr_LMCN.v4[, c(10:17)], n = 2, ID = sample_attributes_LMCN$cell_line[c(1:8)])

## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5     PC6     PC7
## Standard deviation     1.5910 1.1055 0.9596 0.8644 0.7879 0.77001 0.60368
## Proportion of Variance 0.3616 0.1746 0.1316 0.1067 0.0887 0.08471 0.05206
## Cumulative Proportion  0.3616 0.5362 0.6678 0.7745 0.8632 0.94794 1.00000
##                              PC8
## Standard deviation     3.566e-15
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```



PC1 separates the metastatic from non-metastatic lines in both RNA and RPF plots. Replicate consistency seems better for RNA compared to RPF which could be due to the larger RNA read counts in general. The mismatch between reps of the two metastatic cell lines appears to originate from their RPF counts.

The above analyses inform our next step (translational efficiency significance testing): There is a clear difference between metatstatic and non-metastatic lines, but the distinction within these groups is not as large or reliable. Therefore, the most biologically relevant analysis would be to compare TE between metastatic and non-metastatic groups.

3.2. Proportion of null features (non-differentially translated transcripts)

The `pi0est` function from the **qvalue** package estimates the proportion of null features (vs. alternative features) from the distribution of multiple p-values produced by a test. To demonstrate its use in quality control of ribosome profiling data, we compare the proportion of null features π_0 from testing TER of CN34 rep 1 vs CN34 rep 2 or a replicate from any other cell line LM1a rep 1.

The 8th column in the regression output produced by `logit_seq` function contains the p-values of interest (more details in module 4).

```
fit_CN34.1_CN34.2 <- Ribolog::logit_seq(rr_LMCN.v2[, c(2, 3,
    10, 11)], sample_attributes_LMCN[c(1, 2, 9, 10), ], read_type ~
    replicate_name, as.vector(rr_LMCN.v2$transcript))

pi0_CN34.1_CN34.2 <- qvalue::pi0est(fit_CN34.1_CN34.2[, 8])$pi0
print(pi0_CN34.1_CN34.2)

## [1] 0.9461568
```

```
fit_CN34.1_LM1a.1 <- Ribolog::logit_seq(rr_LMCN.v2[, c(2, 4,
    10, 12)], sample_attributes_LMCN[c(1, 3, 9, 11), ], read_type ~
    replicate_name, as.vector(rr_LMCN.v2$transcript))

pi0_CN34.1_LM1a.1 <- qvalue::pi0est(fit_CN34.1_LM1a.1[, 8])$pi0
print(pi0_CN34.1_LM1a.1)

## [1] 0.7779717

fit_CN34.1_LM2.1 <- Ribolog::logit_seq(rr_LMCN.v2[, c(2, 6, 10,
    14)], sample_attributes_LMCN[c(1, 5, 9, 13), ], read_type ~
    replicate_name, as.vector(rr_LMCN.v2$transcript))

pi0_CN34.1_LM2.1 <- qvalue::pi0est(fit_CN34.1_LM2.1[, 8])$pi0
print(pi0_CN34.1_LM2.1)

## [1] 0.7152996

fit_CN34.1_MDA.1 <- Ribolog::logit_seq(rr_LMCN.v2[, c(2, 8, 10,
    16)], sample_attributes_LMCN[c(1, 7, 9, 15), ], read_type ~
    replicate_name, as.vector(rr_LMCN.v2$transcript))

pi0_CN34.1_MDA.1 <- qvalue::pi0est(fit_CN34.1_MDA.1[, 8])$pi0
print(pi0_CN34.1_MDA.1)

## [1] 0.7955519
```

Only 5.4% of transcripts are estimated to be differentially translated when the two CN34 replicates are compared, whereas 22.2%, 28.5% and 20.4% are estimate to be differentially translated between first reps of CN34 vs LM1a, LM2 and MDA, respectively. This is consistent with the PCA output indicating that the two CN34 replicates are more similar to each other than they are to other samples. It also shows that CN34 is more similar to the other non-metastatic line than it is to either of the metastatic ones. Between the two metastatic lines, CN34 is closer to LM1a which originated from it. The proportion of differentially translated transcripts is well below 50% in all tests, indicating that the majority of transcripts are translated somewhat similarly between the compared cell lines.

The proportion of null feature (not differentially translated transcripts) between all pairs of sample replicates can be computed and plotted automatically using the procedure described below.

3.2.1. Convert the RNA-RPF data frame to a sample-by-sample list

The data frame containing RNA and RPF read counts is split to a list based on the values of the parameter `uniqueID`. `uniqueID` is one of the variables in the design matrix which specifies the name of the experimental replicate from which one RNA library and one RPF libray was made. In the case of our LMCN data set, this role is served by the variable `replicate_name`:

```
##    sample_name read_type lung_metastasis cell_line replicate_no
## 1  CN34_r1_rna       RNA               N      CN34            1
## 2  CN34_r2_rna       RNA               N      CN34            2
## 3  LM1a_r1_rna       RNA               Y      LM1a            1
## 4  LM1a_r2_rna       RNA               Y      LM1a            2
## 5   LM2_r1_rna       RNA               Y       LM2            1
## 6   LM2_r2_rna       RNA               Y       LM2            2
## 7   MDA_r1_rna       RNA               N       MDA            1
## 8   MDA_r2_rna       RNA               N       MDA            2
## 9  CN34_r1_rpf       RPF               N      CN34            1
## 10 CN34_r2_rpf       RPF               N      CN34            2
## 11 LM1a_r1_rpf       RPF               Y      LM1a            1
## 12 LM1a_r2_rpf       RPF               Y      LM1a            2
## 13  LM2_r1_rpf       RPF               Y       LM2            1
## 14  LM2_r2_rpf       RPF                Y       LM2            2
## 15  MDA_r1_rpf       RPF                N       MDA            1
## 16  MDA_r2_rpf       RPF                N       MDA            2
##    replicate_name cell_line_origin
## 1         CN34_r1             CN34
## 2         CN34_r2             CN34
## 3         LM1a_r1             CN34
## 4         LM1a_r2             CN34
## 5          LM2_r1              MDA
## 6          LM2_r2              MDA
## 7          MDA_r1              MDA
## 8          MDA_r2              MDA
## 9         CN34_r1             CN34
## 10        CN34_r2             CN34
## 11        LM1a_r1             CN34
## 12        LM1a_r2             CN34
## 13         LM2_r1              MDA
## 14         LM2_r2              MDA
## 15         MDA_r1              MDA
## 16         MDA_r2              MDA
```

```
rr_LMCN.v2.split <- Ribolog::partition_to_uniques(x = rr_LMCN.v2[,
    -1], design = sample_attributes_LMCN, uniqueID = "replicate_name")
names(rr_LMCN.v2.split)
```

```
## [1] "CN34_r1" "CN34_r2" "LM1a_r1" "LM1a_r2" "LM2_r1"  "LM2_r2"  "MDA_r1"
## [8] "MDA_r2"
```

```
print(rr_LMCN.v2.split$CN34_r1[, c(1:10)])
```

```
##             sample_name read_type lung_metastasis cell_line replicate_no
## CN34_r1_rna CN34_r1_rna       RNA               N      CN34            1
## CN34_r1_rpf CN34_r1_rpf       RPF               N      CN34            1
##             replicate_name cell_line_origin        1          2          3
## CN34_r1_rna        CN34_r1             CN34 541.1959 2550.15982 1021.17693
## CN34_r1_rpf        CN34_r1             CN34 105.6832   79.83892   32.11355
```

For the sake of brevity, only count data for the first 3 transcripts are printed out. Notice that the design attributes of each sample is merged with its counts data. This will make the future step of TER significance testing more straightforward.

Input to the `partition_to_uniques` functions must contain only the RNA/RPF data. In the example above, the first column is excluded because it listed transcript IDs. Order of the RNA/RPF columns in the input data matrix must correspond to the rows in the design matrix (compare the order of elements in the `sample_attributes_LMCN` column `sample_name` with the order of `rr_LMCN.v2` data columns):

```
names(rr_LMCN.v2[, -1])
```

```
##  [1] "CN34_r1_rna" "CN34_r2_rna" "LM1a_r1_rna" "LM1a_r2_rna" "LM2_r1_rna"
##  [6] "LM2_r2_rna"  "MDA_r1_rna"  "MDA_r2_rna"  "CN34_r1_rpf" "CN34_r2_rpf"
## [11] "LM1a_r1_rpf" "LM1a_r2_rpf" "LM2_r1_rpf"  "LM2_r2_rpf"  "MDA_r1_rpf"
## [16] "MDA_r2_rpf"
```

3.2.2. Perform translational efficiency ratio (TER) tests on all pairs of samples

With n=8 samples (elements of the split list), C(n,2)=28 pairwise TER tests are performed. At this stage, we need an additional important argument `groupID` which -like `uniqueID`- is another variable or column from the design matrix. All samples having the same `groupID` are considered replicates of the same biological material. In the LMCN dataset, the most sensible choice for groupID is “cell_line” which takes four values “CN34”, “LM1a”, “LM2” or “MDA”.

```r
rr_LMCN.v2.pairwise <- Ribolog::TER_all_pairs(x = rr_LMCN.v2.split,
    design = sample_attributes_LMCN, outcome = "read_type", uniqueID = "replicate_name",
    groupID = "cell_line")
```

Let us look more closely into the content of two elemets of this 28-element list:

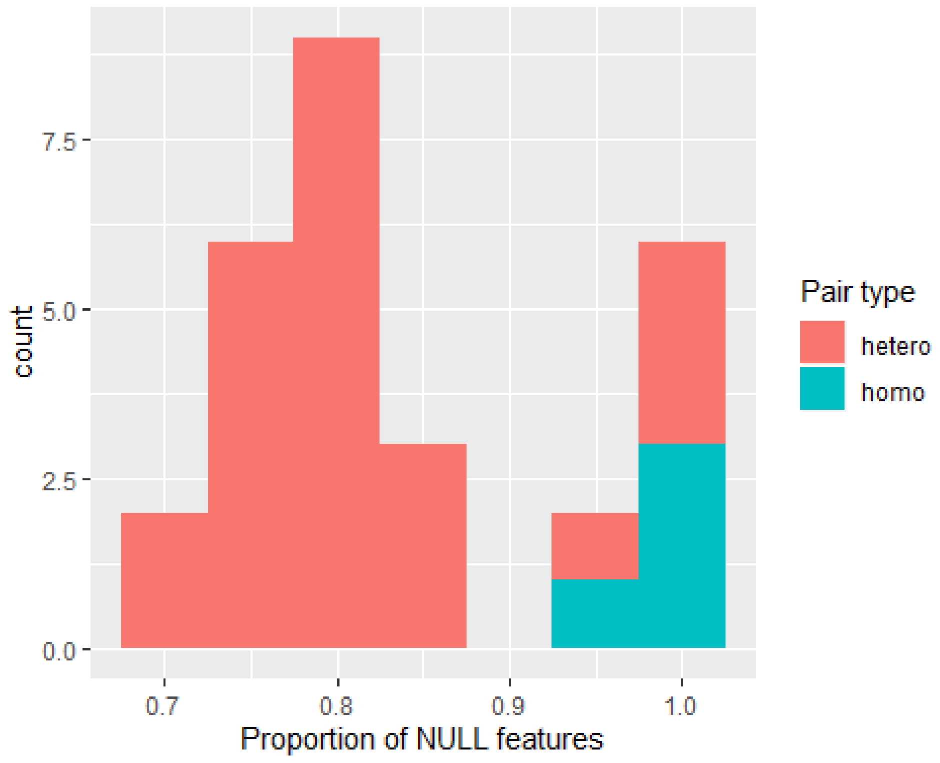
```r
str(rr_LMCN.v2.pairwise$CN34_r1_vs_CN34_r2)

## List of 4
##  $ uniqueIDs: chr [1:2] "CN34_r1" "CN34_r2"
##  $ groupIDs : chr [1:2] "CN34" "CN34"
##  $ pair_type: chr "homo"
##  $ fit      : num [1:11182, 1:8] -1.63 -3.46 -3.46 -2.54 -2.69 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:8] "Estimate.(Intercept)" "Std. Error.(Intercept)" "z value.(Intercept)"
"Pr(>|z|).(Intercept)" ...

str(rr_LMCN.v2.pairwise$CN34_r1_vs_LM1a_r1)

## List of 4
##  $ uniqueIDs: chr [1:2] "CN34_r1" "LM1a_r1"
##  $ groupIDs : chr [1:2] "CN34" "LM1a"
##  $ pair_type: chr "hetero"
##  $ fit      : num [1:11182, 1:8] -1.63 -3.46 -3.46 -2.54 -2.69 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:8] "Estimate.(Intercept)" "Std. Error.(Intercept)" "z value.(Intercept)"
"Pr(>|z|).(Intercept)" ...
```

CN34_r1 and CN34_r2 are replicates of the same biological material, cell line CN34. They have the same groupID "CN34"; therefore, they constitute a homogeneous or *"homo"* pair. On the other hand, CN34_r1 and LM1a_r1 have different groupIDs "CN34" and "LM1a", and are a heterogeneous or *"hetero"* pair. The fourth element of the list (`fit`) is the standard output of the TER test produced by the `logit_seq` function (see module 4 for more details).

3.2.3. Estimate and plot the proportion of null features

We estimate the proportion of null features designated π_0 from each one of the C(n,2) test p-value vectors using the Storey method implemented in the **qvalue** package. Then, we plot of a histogram of π_0s color-coded for homo and hetero pairs.

```r
pi0df_LMCN <- Ribolog::pairs2pi0s(rr_LMCN.v2.pairwise)
```



```
print(pi0df_LMCN)
```

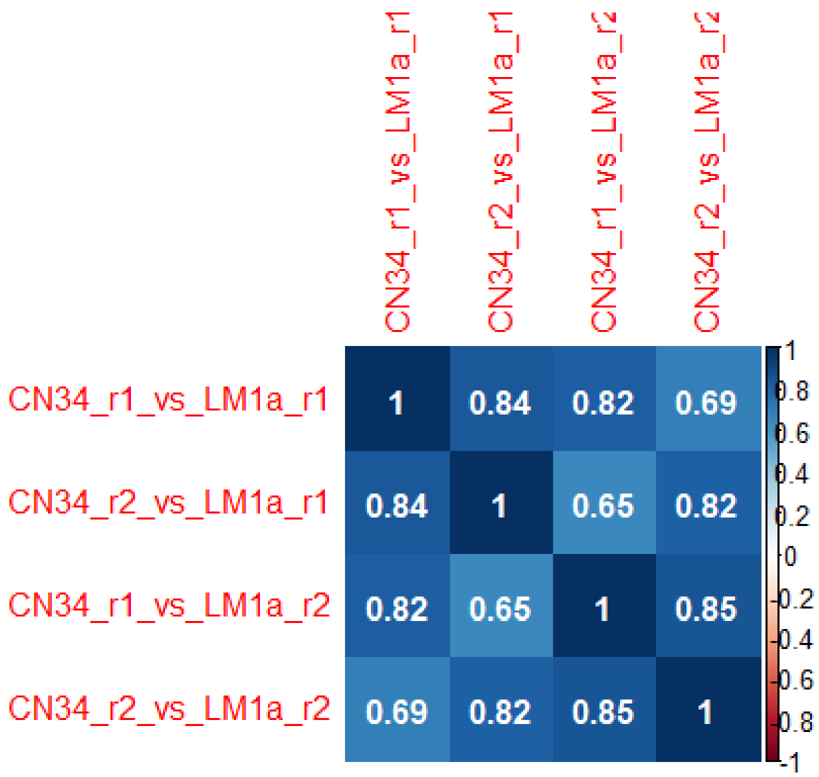
```
##                      uniqueID1 uniqueID2 groupID1 groupID2 pair_type
## CN34_r1_vs_CN34_r2   CN34_r1   CN34_r2      CN34     CN34      homo
## CN34_r1_vs_LM1a_r1   CN34_r1   LM1a_r1      CN34     LM1a    hetero
## CN34_r2_vs_LM1a_r1   CN34_r2   LM1a_r1      CN34     LM1a    hetero
## CN34_r1_vs_LM1a_r2   CN34_r1   LM1a_r2      CN34     LM1a    hetero
## CN34_r2_vs_LM1a_r2   CN34_r2   LM1a_r2      CN34     LM1a    hetero
## LM1a_r1_vs_LM1a_r2   LM1a_r1   LM1a_r2      LM1a     LM1a      homo
## CN34_r1_vs_LM2_r1    CN34_r1    LM2_r1      CN34      LM2    hetero
## CN34_r2_vs_LM2_r1    CN34_r2    LM2_r1      CN34      LM2    hetero
## LM1a_r1_vs_LM2_r1    LM1a_r1    LM2_r1      LM1a      LM2    hetero
## LM1a_r2_vs_LM2_r1    LM1a_r2    LM2_r1      LM1a      LM2    hetero
## CN34_r1_vs_LM2_r2    CN34_r1    LM2_r2      CN34      LM2    hetero
## CN34_r2_vs_LM2_r2    CN34_r2    LM2_r2      CN34      LM2    hetero
## LM1a_r1_vs_LM2_r2    LM1a_r1    LM2_r2      LM1a      LM2    hetero
## LM1a_r2_vs_LM2_r2    LM1a_r2    LM2_r2      LM1a      LM2    hetero
## LM2_r1_vs_LM2_r2      LM2_r1    LM2_r2       LM2      LM2      homo
## CN34_r1_vs_MDA_r1    CN34_r1    MDA_r1      CN34      MDA    hetero
## CN34_r2_vs_MDA_r1    CN34_r2    MDA_r1      CN34      MDA    hetero
## LM1a_r1_vs_MDA_r1    LM1a_r1    MDA_r1      LM1a      MDA    hetero
## LM1a_r2_vs_MDA_r1    LM1a_r2    MDA_r1      LM1a      MDA    hetero
## LM2_r1_vs_MDA_r1      LM2_r1    MDA_r1       LM2      MDA    hetero
## LM2_r2_vs_MDA_r1      LM2_r2    MDA_r1       LM2      MDA    hetero
## CN34_r1_vs_MDA_r2    CN34_r1    MDA_r2      CN34      MDA    hetero
## CN34_r2_vs_MDA_r2    CN34_r2    MDA_r2      CN34      MDA    hetero
## LM1a_r1_vs_MDA_r2    LM1a_r1    MDA_r2      LM1a      MDA    hetero
## LM1a_r2_vs_MDA_r2    LM1a_r2    MDA_r2      LM1a      MDA    hetero
## LM2_r1_vs_MDA_r2      LM2_r1    MDA_r2       LM2      MDA    hetero
## LM2_r2_vs_MDA_r2      LM2_r2    MDA_r2       LM2      MDA    hetero
## MDA_r1_vs_MDA_r2      MDA_r1    MDA_r2       MDA      MDA      homo
##                            pi0
## CN34_r1_vs_CN34_r2 0.9461568
## CN34_r1_vs_LM1a_r1 0.7779717
## CN34_r2_vs_LM1a_r1 0.7757293
## CN34_r1_vs_LM1a_r2 0.7640147
## CN34_r2_vs_LM1a_r2 0.7922021
## LM1a_r1_vs_LM1a_r2 1.0000000
## CN34_r1_vs_LM2_r1  0.7152996
## CN34_r2_vs_LM2_r1  0.7529342
## LM1a_r1_vs_LM2_r1  1.0000000
## LM1a_r2_vs_LM2_r1  1.0000000
## CN34_r1_vs_LM2_r2  0.6953754
## CN34_r2_vs_LM2_r2  0.7325974
## LM1a_r1_vs_LM2_r2  0.9743489
## LM1a_r2_vs_LM2_r2  1.0000000
## LM2_r1_vs_LM2_r2   0.9822130
## CN34_r1_vs_MDA_r1  0.7955519
## CN34_r2_vs_MDA_r1  0.7515743
## LM1a_r1_vs_MDA_r1  0.8074648
## LM1a_r2_vs_MDA_r1  0.8154790
## LM2_r1_vs_MDA_r1   0.8108805
## LM2_r2_vs_MDA_r1   0.7647829
## CN34_r1_vs_MDA_r2  0.8013137
## CN34_r2_vs_MDA_r2  0.7782151
## LM1a_r1_vs_MDA_r2  0.8623039
## LM1a_r2_vs_MDA_r2  0.8340768
## LM2_r1_vs_MDA_r2   0.8573638
## LM2_r2_vs_MDA_r2   0.7506477
## MDA_r1_vs_MDA_r2   0.9895269
```

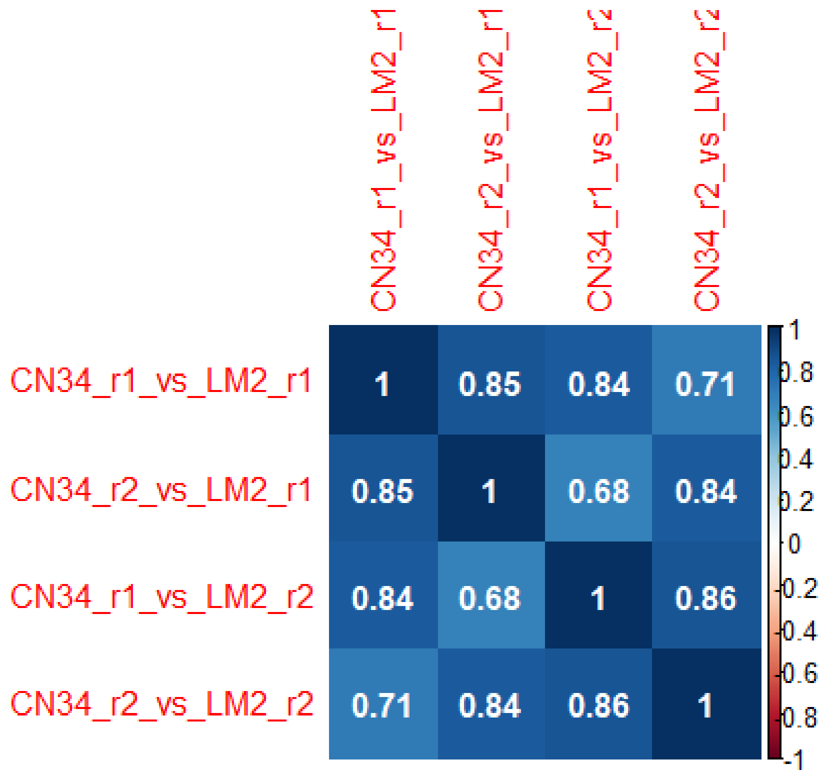
Expectedly, most of the hetero pairs (brick red) show lower π_0 compared to the homo pairs (green). There are four hetero pairs that cluster with homo pairs. Inspection of the data frame shows that these four involve comparisons of the LM1a and LM2 cell lines. This is consistent with the near identicality of these samples demonstrated by their PCA patterns in the previous section.

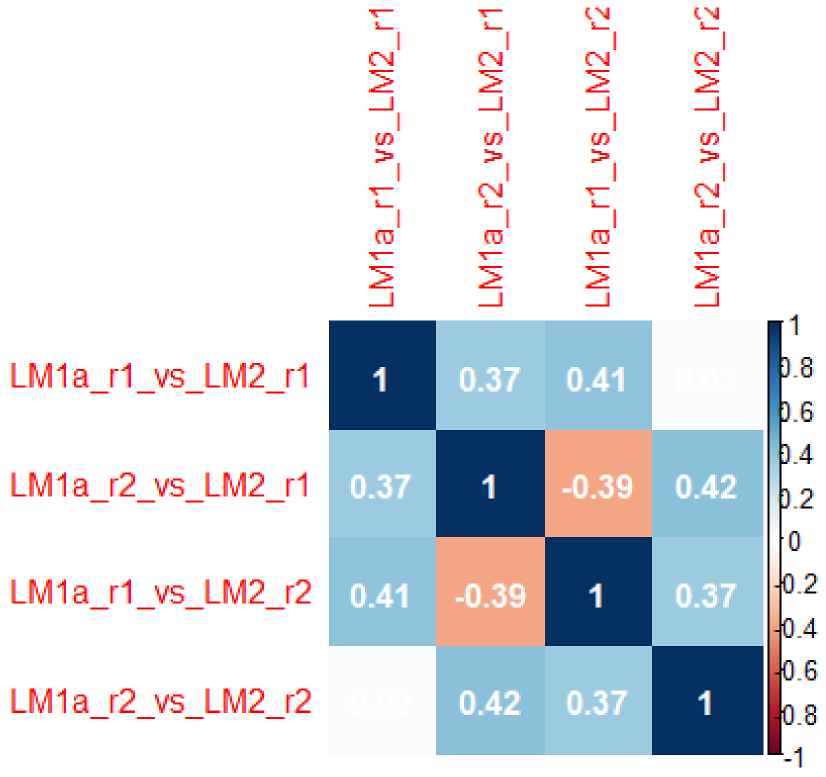
3.3. Correlogram of equivalent test statistics

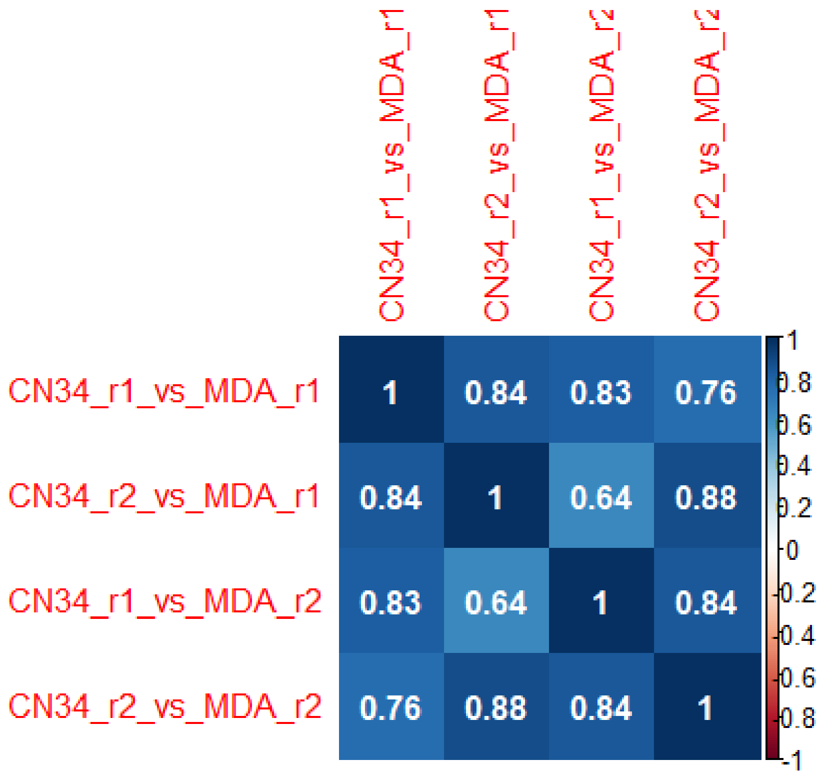
The Ribolog TER test can be performed on single replicates per biological sample. In a replicated experiment such as (sample A: reps A1 and A2 + sample B: reps B1 and B2), correlation coefficients of regression z scores from equivalent tests i.e. A1 vs B1, A2 vs B1, A1 vs B2 and A2 vs B2 can be used to evaluate replicate homogeneity and help determine the minimum advisable number of replicates to achieve reproducibility of conclusions.

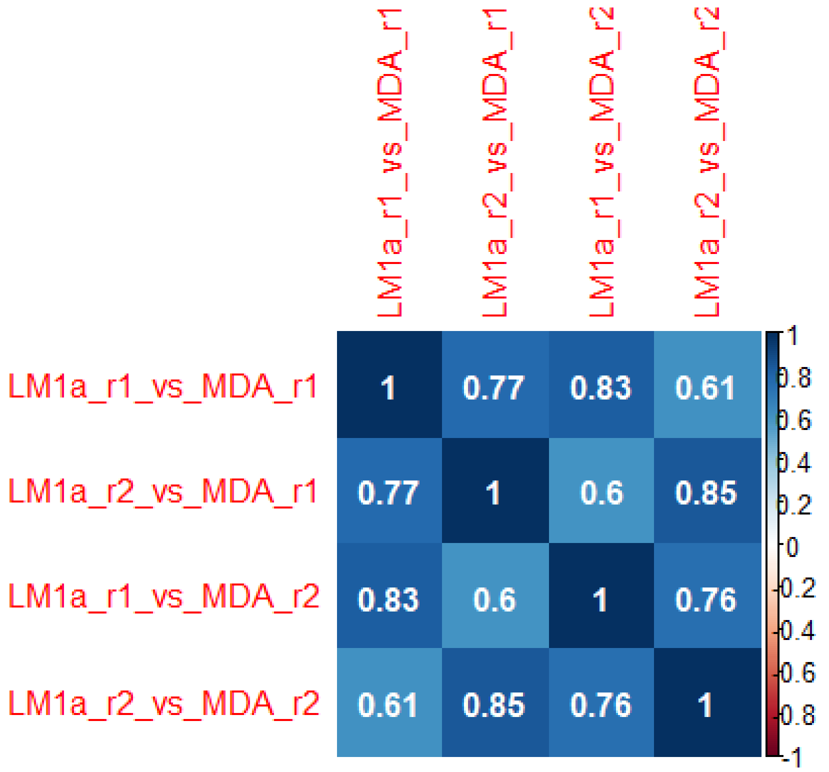
```
rr_LMCN.v2.correlograms <- pairs2correlograms(rr_LMCN.v2.pairwise)
```


CN34_r1_vs_LM1a_r1
CN34_r2_vs_LM1a_r1
CN34_r1_vs_LM1a_r2
CN34_r2_vs_LM1a_r2
CN34_r1_vs_LM1a_r1 1 0.84 0.82 0.69
CN34_r2_vs_LM1a_r1 0.84 1 0.65 0.82
CN34_r1_vs_LM1a_r2 0.82 0.65 1 0.85
CN34_r2_vs_LM1a_r2 0.69 0.82 0.85 1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

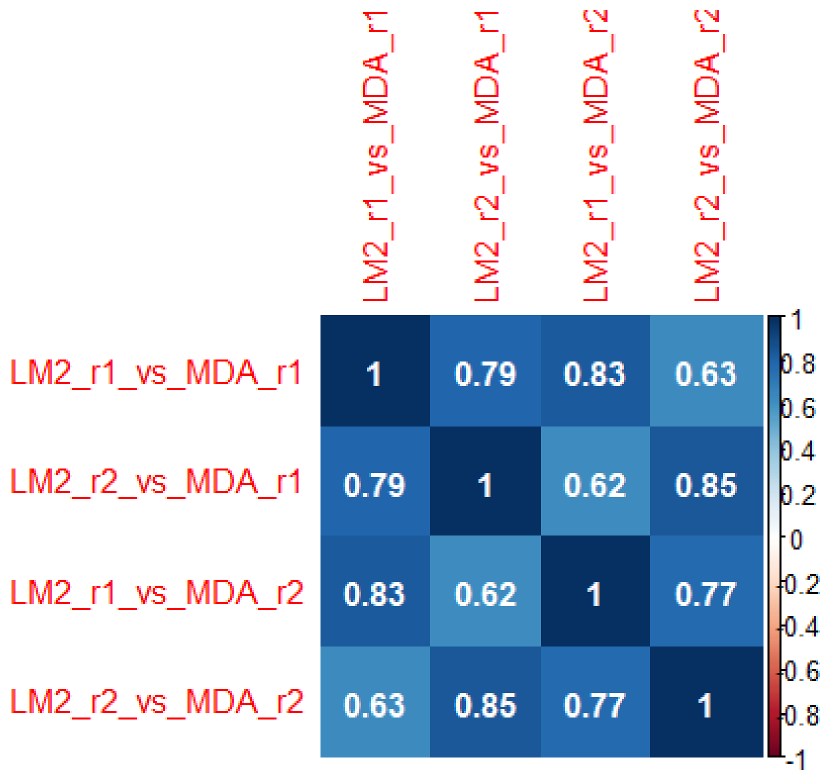

CN34_r1_vs_LM2_r1
CN34_r2_vs_LM2_r1
CN34_r1_vs_LM2_r2
CN34_r2_vs_LM2_r2
CN34_r1_vs_LM2_r1 1 0.85 0.84 0.71
CN34_r2_vs_LM2_r1 0.85 1 0.68 0.84
CN34_r1_vs_LM2_r2 0.84 0.68 1 0.86
CN34_r2_vs_LM2_r2 0.71 0.84 0.86 1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


LM1a_r1_vs_LM2_r1
LM1a_r2_vs_LM2_r1
LM1a_r1_vs_LM2_r2
LM1a_r2_vs_LM2_r2
1 0.37 0.41 0.02
0.37 1 -0.39 0.42
0.41 -0.39 1 0.37
0.02 0.42 0.37 1
1 0.8 0.6 0.4 0.2 0 -0.2 -0.4 -0.6 -0.8 -1


CN34_r1_vs_MDA_r1
CN34_r2_vs_MDA_r1
CN34_r1_vs_MDA_r2
CN34_r2_vs_MDA_r2
1 0.84 0.83 0.76
0.84 1 0.64 0.88
0.83 0.64 1 0.84
0.76 0.88 0.84 1
1 0.8 0.6 0.4 0.2 0 -0.2 -0.4 -0.6 -0.8 -1



	LM1a_r1_vs_MDA_r1	LM1a_r2_vs_MDA_r1	LM1a_r1_vs_MDA_r2	LM1a_r2_vs_MDA_r2
LM1a_r1_vs_MDA_r1	1	0.77	0.83	0.61
LM1a_r2_vs_MDA_r1	0.77	1	0.6	0.85
LM1a_r1_vs_MDA_r2	0.83	0.6	1	0.76
LM1a_r2_vs_MDA_r2	0.61	0.85	0.76	1



	LM2_r1_vs_MDA_r1	LM2_r2_vs_MDA_r1	LM2_r1_vs_MDA_r2	LM2_r2_vs_MDA_r2
LM2_r1_vs_MDA_r1	1	0.79	0.83	0.63
LM2_r2_vs_MDA_r1	0.79	1	0.62	0.85
LM2_r1_vs_MDA_r2	0.83	0.62	1	0.77
LM2_r2_vs_MDA_r2	0.63	0.85	0.77	1

The z scores of equivalent tests are 60-88% correlated in all pairwise comparisons except for LM1a vs. LM2. The highest correlation is seen between the equivalent CN34 vs MDA tests. This is consistent with the observations from PCA and π_0 plots. It highlights the fact that CN34 replicates and MDA replicates are sufficiently similar but that is not the case with LM1a and LM2. This is either an indication of issues in the experimental sample preparation steps of these cell lines or due to higher biological stochasticity of translational patterns in metastatic cell lines. The more highly variable biological samples ought to be represented by more replicates to achieve reproducibility.

If the QC measures cause concern, you may go back to previous steps to remove bad samples or try alternative transcript filtering strategies; then reproduce the QC measures until a satisfactorily reliable data set is obtained. The cleaned up and finalized data set will be used for the main TER analysis laid out in module 4.

Module 4: TER

Basic differential translational efficiency test

This module is the heart of the **Ribolog** package as it contains the differential translational efficiency significance test by the `logit_seq` function.

4.1. Read in the design matrix

We have prepared our dataset for the test in previous modules: corrected stalling biases using the CELP method in module 1, combined RNA and RPF counts and normalized for library size variation in module 2 and removed low count transcripts and confirmed replicate consistency in module 3. The prepared data set looks like this:

```
##        transcript CN34_r1_rna CN34_r2_rna LM1a_r1_rna LM1a_r2_rna
## 1 ENST00000000233    541.1959    555.3465   485.82975   537.04350
## 2 ENST00000000412   2550.1598   2606.9633  2761.49782  2763.04991
## 3 ENST00000000442   1021.1769   1037.3182   877.03232   919.20055
## 4 ENST00000001008   2422.1649   2304.8318  2344.90747  2307.73042
## 5 ENST00000001146    107.1262    105.0267    43.85162    73.16245
## 6 ENST00000002125    191.9924    189.9112   165.02055   179.01450
##   LM2_r1_rna LM2_r2_rna MDA_r1_rna MDA_r2_rna CN34_r1_rpf CN34_r2_rpf
## 1  520.41321  495.01723  674.71993  654.90999  105.683213  105.694545
## 2 2845.15383 2874.13065 2280.88249 2246.27195   79.838919   66.606180
## 3  916.42941  835.69235 1178.29139 1214.52884   32.113554   24.369694
## 4 2389.79226 2365.08232 2144.29285 2146.64940  191.155731  195.030702
## 5   70.75794   83.06412   58.42087   52.84327    7.265732    3.460472
## 6  200.86124  193.06794  232.03783  201.84395   23.542616   17.863898
##   LM1a_r1_rpf LM1a_r2_rpf LM2_r1_rpf LM2_r2_rpf MDA_r1_rpf MDA_r2_rpf
## 1  130.124360    89.51924 100.698905 107.141235 117.152749 122.288775
## 2   62.722726    59.76405  69.141599  61.973577  81.529191  78.238890
## 3   30.119433    26.32107  38.913192  30.204999  27.554240  27.284363
## 4  258.359140   231.90599 234.684263 246.431556 192.696795 210.905078
## 5   12.637656    11.52465   8.614001   9.093483   4.116696   3.009298
## 6    3.259024    13.31193   9.335733  18.450922  15.827547  18.146907
```

We also need the design matrix (meta data) which describes the attributes of each sample:

```
sample_attributes_LMCN <- read.xlsx("../data-raw/sample_attributes_LMCN.xlsx",
    sheetIndex = 1, header = TRUE)
print(sample_attributes_LMCN)
```

```
##    sample_name read_type lung_metastasis cell_line replicate_no
## 1  CN34_r1_rna       RNA               N      CN34            1
## 2  CN34_r2_rna       RNA               N      CN34            2
## 3  LM1a_r1_rna       RNA               Y      LM1a            1
## 4  LM1a_r2_rna       RNA               Y      LM1a            2
## 5   LM2_r1_rna       RNA               Y       LM2            1
## 6   LM2_r2_rna       RNA               Y       LM2            2
## 7   MDA_r1_rna       RNA               N       MDA            1
## 8   MDA_r2_rna       RNA               N       MDA            2
## 9  CN34_r1_rpf       RPF               N      CN34            1
## 10 CN34_r2_rpf       RPF               N      CN34            2
## 11 LM1a_r1_rpf       RPF               Y      LM1a            1
## 12 LM1a_r2_rpf       RPF               Y      LM1a            2
## 13  LM2_r1_rpf       RPF               Y       LM2            1
## 14  LM2_r2_rpf       RPF               Y       LM2            2
## 15  MDA_r1_rpf       RPF               N       MDA            1
## 16  MDA_r2_rpf       RPF               N       MDA            2
##    replicate_name cell_line_origin
## 1         CN34_r1             CN34
## 2         CN34_r2             CN34
## 3         LM1a_r1             CN34
## 4         LM1a_r2             CN34
## 5          LM2_r1              MDA
## 6          LM2_r2              MDA
## 7          MDA_r1              MDA
## 8          MDA_r2              MDA
## 9         CN34_r1             CN34
## 10        CN34_r2             CN34
## 11        LM1a_r1             CN34
## 12        LM1a_r2             CN34
## 13         LM2_r1              MDA
## 14         LM2_r2              MDA
## 15         MDA_r1              MDA
## 16         MDA_r2              MDA
```

NOTE: The order of samples in the design matrix MUST be exactly the same as that in the read count data set.

4.2. Run translational efficiency ratio test

Now we are ready to perform the translational efficiency test using the `logit_seq` function. TE is the RPF/RNA ratio. If we are interested in comparing TE between the metastatic and non-metastatic samples, we set the model to *read_type ~ lung_metastasis*. The count data set (argument x) must contain only numeric variables, therefore column 1 (transcript) is excluded from x and provided separately as the `feature list` in the end.

NOTE: The input data set should not contain any transcripts where RNA counts are zero in all samples. Translational efficiency $TE = RPF/RNA$ cannot be calculated in such cases and the function will return an error.

```
fit1_LMCN <- Ribolog::logit_seq(rr_LMCN.v2[, -1], sample_attributes_LMCN,
    read_type ~ lung_metastasis, as.vector(rr_LMCN.v2$transcript))
head(fit1_LMCN)

##                   Estimate.(Intercept) Std. Error.(Intercept)
## ENST00000000233            -1.683004             0.05128672
## ENST00000000412            -3.453977             0.05804273
## ENST00000000442            -3.688529             0.09594590
## ENST00000001008            -2.435206             0.03710865
## ENST00000001146            -2.896816             0.24309348
## ENST00000002125            -2.381597             0.12038173
##                 z value.(Intercept) Pr(>|z|).(Intercept)
## ENST00000000233           -32.81558        3.529545e-236
## ENST00000000412           -59.50749         0.000000e+00
## ENST00000000442           -38.44384         0.000000e+00
## ENST00000001008           -65.62368         0.000000e+00
## ENST00000001146           -11.91647         9.713957e-33
## ENST00000002125           -19.78370         4.113321e-87
##                 Estimate.lung_metastasisY Std. Error.lung_metastasisY
## ENST00000000233                 0.1210466                  0.07389310
## ENST00000000412                -0.3378317                  0.08602955
## ENST00000000442                 0.3470633                  0.13210388
## ENST00000001008                 0.1646610                  0.05012795
## ENST00000001146                 1.0298665                  0.29439779
## ENST00000002125                -0.4300146                  0.19593592
##                 z value.lung_metastasisY Pr(>|z|).lung_metastasisY
## ENST00000000233                 1.638131              1.013943e-01
## ENST00000000412                -3.926926              8.603853e-05
## ENST00000000442                 2.627200              8.609066e-03
## ENST00000001008                 3.284814              1.020497e-03
## ENST00000001146                 3.498214              4.683852e-04
## ENST00000002125                -2.194670              2.818731e-02
```

Regression coefficient is the natural log of translational efficiency ratio (TER). For example, Estimate.lung_metastasisY=0.1210466 (p=0.1013943) for transcript ENST00000000233. This means that:

$$\frac{TE_{Lung\ metastasis='Y'}}{TE_{Lung\ metastasis='N'}} = exp(0.1210466) = 1.1286775$$

Translational efficiency of transcript ENST00000000233 is estimated to be 12.87% higher in metastatic samples compared to non-metastatic ones. However, this difference is not statistically significant (p=0.1014).

Regression reports usually include only a regression coefficient (Estimate) and a p-value. We keep SE and z in the output data frame to enable certain tasks e.g. generation of correlograms from z scores (module 3, *TEST 3*).

Finally, p-values are corrected for multiple testing. Run ?adj_TER_p to see all the available methods. Two examples are shown below. Each column of p-values is corrected for multiple testing separately.

```
fit1_LMCN_FDR <- Ribolog::adj_TER_p(fit1_LMCN, pcols = c(4, 8),
    adj.method = "fdr")
names(fit1_LMCN_FDR)

##  [1] "Estimate..Intercept."          "Std..Error..Intercept."
##  [3] "z.value..Intercept."           "Pr...z....Intercept."
##  [5] "Estimate.lung_metastasisY"     "Std..Error.lung_metastasisY"
##  [7] "z.value.lung_metastasisY"      "Pr...z...lung_metastasisY"
##  [9] "fdr.Pr...z....Intercept."      "fdr.Pr...z...lung_metastasisY"

fit1_LMCN_qval <- Ribolog::adj_TER_p(fit1_LMCN, pcols = c(4,
    8), adj.method = "qvalue")
names(fit1_LMCN_qval)

##  [1] "Estimate..Intercept."             "Std..Error..Intercept."
##  [3] "z.value..Intercept."              "Pr...z....Intercept."
##  [5] "Estimate.lung_metastasisY"        "Std..Error.lung_metastasisY"
##  [7] "z.value.lung_metastasisY"         "Pr...z...lung_metastasisY"
##  [9] "qvalue.Pr...z....Intercept."      "qvalue.Pr...z...lung_metastasisY"
```

The logistic regression model can have more than one predictor. Suppose that we want to know the relative effects of genetic background (cell line origin) and metastatic state as well as their interaction:

```
fit2_LMCN <- Ribolog::logit_seq(rr_LMCN.v2[, -1], sample_attributes_LMCN,
    read_type ~ lung_metastasis * cell_line_origin, as.vector(rr_LMCN.v2$transcript))
fit2_LMCN_qval <- Ribolog::adj_TER_p(fit2_LMCN, c(4, 8, 12, 16),
    adj.method = "qvalue")
head(fit2_LMCN_qval)

##                   Estimate..Intercept. Std..Error..Intercept.
## ENST00000000233            -1.646270             0.07511844
## ENST00000000412            -3.561484             0.08379978
## ENST00000000442            -3.595786             0.13486111
## ENST00000001008            -2.504725             0.05292421
## ENST00000001146            -2.984617             0.31289988
## ENST00000002125            -2.221730             0.16361311
##                   z.value..Intercept. Pr...z....Intercept.
## ENST00000000233           -21.915664        1.841877e-106
## ENST00000000412           -42.499913         0.000000e+00
## ENST00000000442           -26.662885        1.268807e-156
## ENST00000001008           -47.326640         0.000000e+00
## ENST00000001146            -9.538569         1.448164e-21
## ENST00000002125           -13.579169         5.322709e-42
##                   Estimate.lung_metastasisY Std..Error.lung_metastasisY
## ENST00000000233                0.1079058                  0.10570350
## ENST00000000412               -0.2474701                  0.12396596
## ENST00000000442                0.1355264                  0.19094635
## ENST00000001008                0.2544818                  0.07110286
## ENST00000001146                1.4071169                  0.38449842
## ENST00000002125               -0.8113620                  0.30003746
##                   z.value.lung_metastasisY Pr...z...lung_metastasisY
## ENST00000000233               1.0208345             0.3073328710
## ENST00000000412              -1.9962743             0.0459040762
## ENST00000000442               0.7097619             0.4778517889
## ENST00000001008               3.5790655             0.0003448250
## ENST00000001146               3.6596169             0.0002525926
## ENST00000002125              -2.7042023             0.0068468582
##                   Estimate.cell_line_originMDA
## ENST00000000233                  -0.06807629
## ENST00000000412                   0.21735796
## ENST00000000442                  -0.18004719
## ENST00000001008                   0.14089239
## ENST00000001146                   0.23645949
## ENST00000002125                  -0.32543312
##                   Std..Error.cell_line_originMDA z.value.cell_line_originMDA
## ENST00000000233                       0.10281627                  -0.6621159
## ENST00000000412                       0.11619968                   1.8705556
## ENST00000000442                       0.19192458                  -0.9381143
## ENST00000001008                       0.07424089                   1.8977734
## ENST00000001146                       0.49720721                   0.4755754
## ENST00000002125                       0.24188407                  -1.3454095
##                   Pr...z...cell_line_originMDA
## ENST00000000233                   0.50789696
## ENST00000000412                   0.06140670
## ENST00000000442                   0.34818568
## ENST00000001008                   0.05772594
## ENST00000001146                   0.63437694
## ENST00000002125                   0.17849302
##                   Estimate.lung_metastasisY.cell_line_originMDA
## ENST00000000233                                      0.02014226
## ENST00000000412                                     -0.18392719
## ENST00000000442                                   0.40754227
## ENST00000001008                                  -0.18146666
## ENST00000001146                                  -0.82076827
## ENST00000002125                                   0.70690988
##                 Std..Error.lung_metastasisY.cell_line_originMDA
## ENST00000000233                                      0.1479791
## ENST00000000412                                      0.1721896
## ENST00000000442                                      0.2648514
## ENST00000001008                                      0.1002755
## ENST00000001146                                      0.6000997
## ENST00000002125                                      0.4003600
##                 z.value.lung_metastasisY.cell_line_originMDA
## ENST00000000233                                   0.1361156
## ENST00000000412                                  -1.0681668
## ENST00000000442                                   1.5387581
## ENST00000001008                                  -1.8096818
## ENST00000001146                                  -1.3677199
## ENST00000002125                                   1.7656855
##                 Pr...z...lung_metastasisY.cell_line_originMDA
## ENST00000000233                                    0.89172991
## ENST00000000412                                    0.28544528
## ENST00000000442                                    0.12386337
## ENST00000001008                                    0.07034514
## ENST00000001146                                    0.17139978
## ENST00000002125                                    0.07744863
##                 qvalue.Pr...z....Intercept.
## ENST00000000233              3.809054e-108
## ENST00000000412               0.000000e+00
## ENST00000000442              3.400758e-158
## ENST00000001008               0.000000e+00
## ENST00000001146               1.821333e-23
## ENST00000002125               7.652281e-44
##                 qvalue.Pr...z...lung_metastasisY
## ENST00000000233                     0.2464058971
## ENST00000000412                     0.0637926056
## ENST00000000442                     0.3259909337
## ENST00000001008                     0.0012449317
## ENST00000001146                     0.0009597108
## ENST00000002125                     0.0147193720
##                 qvalue.Pr...z...cell_line_originMDA
## ENST00000000233                           0.4075439
## ENST00000000412                           0.1099547
## ENST00000000442                           0.3327599
## ENST00000001008                           0.1053872
## ENST00000001146                           0.4558585
## ENST00000002125                           0.2230103
##                 qvalue.Pr...z...lung_metastasisY.cell_line_originMDA
## ENST00000000233                                            0.6904336
## ENST00000000412                                            0.4317245
## ENST00000000442                                            0.2812951
## ENST00000001008                                            0.2036766
## ENST00000001146                                            0.3330510
## ENST00000002125                                            0.2159364
```

Above is the output of the following regression equation solved separately for each transcript:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{12} X_1 X_2 + \epsilon$$

Where:

$Y: \log(TE)$

$\beta_0: intercept$

$X_1: cell\ line\ origin\ (CN34\ and\ LM1a: X_1 = 0,\ MDA\ and\ LM2: X_1 = 1)$

$X_2: lung\ metastasis\ (CN34\ and\ MDA: X_2 = 0,\ LM1a\ and\ LM2: X_2 = 1)$

$X_1X_2: interaction\ of\ X_1\ and\ X_2$

$\epsilon: error\ term$

In other words, the design matrix can be summerized thus:

Cell line	X_1 (cell line origin)	X_2 (lung metastasis)
CN34	0	0
LM1a	0	1
LM2	1	1
MDA	1	0

Translational efficiency ratio (TER) between any two cell lines can be easily calculated by replacing the corresponding values of X_1 and X_2 into the parameterized (solved) regression equation. For example, we can calculate the TER for transcript ENST00000000233 between cell lines LM2 and CN34:

$$log\left(\frac{TE_{LM2}}{TE_{CN34}}\right) = log(TE)_{LM2} - log(TE)_{CN34} = (\beta_0 - \beta_0) + \beta_1(1-0) + \beta_2(1-0) + \beta_{12}(1-0)$$

$$= \beta_1 + \beta_2 + \beta_{12} = 0.1079058 - 0.0680763 + 0.0201423 = 0.0599718$$

$$\frac{TE_{LM2}}{TE_{CN34}} = exp(0.0599718) = 1.0618066$$

However, note that in the case of transcript ENST00000000233, none of the regression coefficients $\beta_1, \beta_2, \beta_{12}$ is significantly different from zero after multiple testing correction (check out the q-values in the output).

An important advantage of **Ribolog** is that it can run the TER test using only a single replicate per sample, or a single sample per biological condition. Below, we compare CN34 and LM1a lines using only one replicate from each:

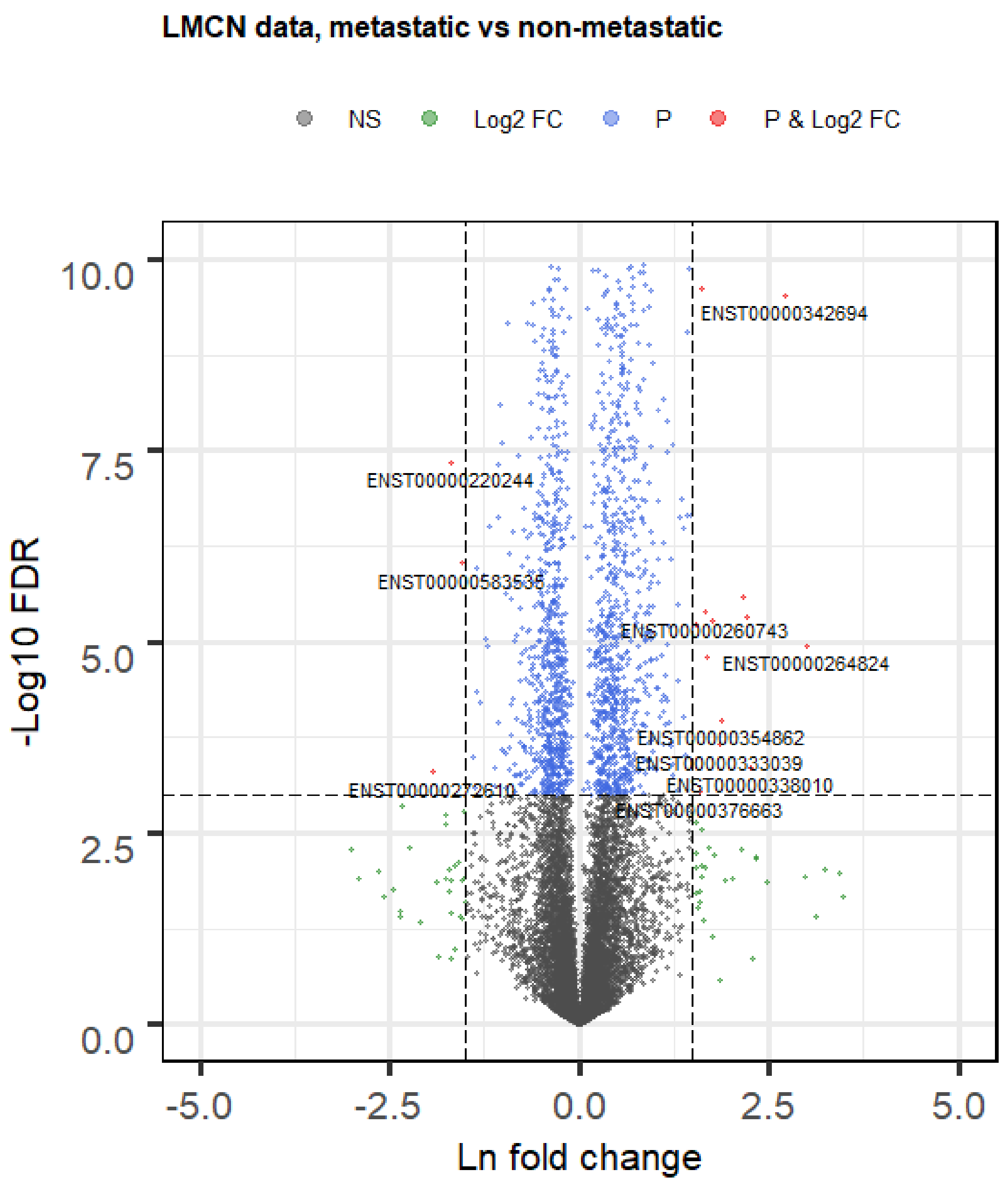
```
fit3_LMCN <- Ribolog::logit_seq(rr_LMCN.v2[, c(2, 4, 10, 12)],
    sample_attributes_LMCN[c(1, 3, 9, 11), ], read_type ~ cell_line,
    as.vector(rr_LMCN.v2$transcript))
fit3_LMCN_qval <- Ribolog::adj_TER_p(fit3_LMCN, c(4, 8), adj.method = "qvalue")
names(fit3_LMCN_qval)
```

```
##  [1] "Estimate..Intercept."          "Std..Error..Intercept."
##  [3] "z.value..Intercept."           "Pr...z....Intercept."
##  [5] "Estimate.cell_lineLM1a"        "Std..Error.cell_lineLM1a"
##  [7] "z.value.cell_lineLM1a"         "Pr...z...cell_lineLM1a"
##  [9] "qvalue.Pr...z....Intercept."   "qvalue.Pr...z...cell_lineLM1a"
```

Notice that the only difference between CN34 and LM1a cell lines is metastatic state. Using the data from these two cell lines alone, the models *read_type ~ cell_line* and *read_type ~ lung_metastasis* produces the same quantitative output.

We can visualize the results in volcano plots. For example, using the `EnhancedVolcano` function from **EnhancedVolcano** package:

```
vplot1 <- EnhancedVolcano(fit1_LMCN_FDR, lab = rownames(fit1_LMCN_FDR),
    x = "Estimate.lung_metastasisY", xlab = "Ln fold change",
    y = "fdr.Pr...z...lung_metastasisY", ylab = "-Log10 FDR",
    title = "LMCN data, metastatic vs non-metastatic", titleLabSize = 12,
    border = "full", pCutoff = 0.001, FCcutoff = 1.5, xlim = c(-5,
        5), ylim = c(0, 10))
vplot1
```


LMCN data, metastatic vs non-metastatic
NS
Log2 FC
P
P & Log2 FC
ENST00000342694
ENST00000220244
ENST00000583535
ENST00000260743
ENST00000264824
ENST00000354862
ENST00000333039
ENST00000338010
ENST00000272610
ENST00000376663
-Log10 FDR
Ln fold change