Visualization, Quality Control and Statistical Analysis for Ribosome Profiling data : the **RiboVIEW** Package

Carine Legrand[*,†] Francesca Tuorto[‡]

September 2019

[*]Division of Epigenetics, DKFZ-ZMBH Alliance, German Cancer Research Center, Im Neuenheimer Feld 580, 69120 Heidelberg, Germany. c.legrand@dkfz.de

[†]Independent researcher, Kreuzstr. 5, 68259 Mannheim. carine.legrand1@gmail.com

[‡]Division of Epigenetics, DKFZ-ZMBH Alliance, German Cancer Research Center, Im Neuenheimer Feld 580, 69120 Heidelberg, Germany.

Contents

1 Introduction

The **RiboVIEW** package provides quality control, an unbiased estimate of enrichment and visualization tools for ribosome profiling data. This vignette describes the workflow from alignment files (BAM format) to **RiboVIEW** output files, including RiboQC and RiboMINE html files. This is done on the base of synthetic BAM files, which are small enough to be included in a software package, and which are fully defined :

- 2 conditions, 3 replicates per condition,
- 20000 reads, 100 mRNA
- A site mostly in-frame in simulated footprints (odds 50:11:1 in 0/+1/+2 reading frame)
- Codon Proline-CCC is paused in condition 2 as compared to condition 1.

Additionally, the following information is appended :

- Example Workflow from SRA to BAM files,
- File used to generate synthetic BAM files used in this vignette (in Python),
- Troubleshooting.

2 Installing RiboVIEW

2.1 Installing R package RiboVIEW

RiboVIEW can be installed in R ($\geq$ 3.4.4) using

```
> install.packages("RiboVIEW")
```

which will fetch the package directly from CRAN repository. After successful installation, package **RiboVIEW** is loaded by

```
> library(RiboVIEW)
```

Alternatively, it's possible to download the source package RiboVIEW_1.0.tar.gz from CRAN and install it using :

```
> install.packages("PATH/TO/FILE", repos = NULL, type="source")
```

Yet further alternatives exist, please consult CRAN help pages for details : https://cran.r-project.org/manuals.html.

If installing from source, the user should also make sure that the following R packages dependencies are present :

- dendextend ($\geq$ 1.8.0)

- ggplot2 ($\geq$ 3.0.0)
- gplots ($\geq$ 3.0.1)
- gridExtra ($\geq$ 2.3)
- latex2exp ($\geq$ 0.4.0)
- MASS ($\geq$ 7.3-50)
- png ($\geq$ 0.1-7)
- RColorBrewer ($\geq$ 1.1-2)
- PythonInR ($\geq$ 0.1-7)
- R.devices ($\geq$ 2.16.1)
- tseriesChaos ($\geq$ 0.1-13)
- tsne ($\geq$ 0.1-3)
- VennDiagram ($\geq$ 1.6.20)

2.2 Python dependencies

RiboVIEW package relies on Python for calculation of all base metrics, including enrichment (version 2.7.6 or superior, up to 2.7.9). This package was developed and tested in Python 2, though large parts of the code would probably also work well in Python 3.

Further, some Python modules are necessary as well. Some should already be present in the core Python distribution, others should be installed, for example using `pip` (see https://pip.pypa.io/en/stable/installing/ for pip installation and instructions).

The list of these dependencies is :

- Biopython ($\geq$ 1.72, available from https://biopython.org/wiki/Download)
- numpy ($\geq$ 1.8.2, available from https://www.numpy.org, see detailed installation instructions at https://www.scipy.org/install.html)
- pysam ($\geq$ 0.15.0, available from https://pypi.org/project/pysam/)

Now that everything is installed, it is possible to start using **RiboVIEW**. The next section describe how to setup **RiboVIEW** for a set of aligned reads (BAM files), and the corresponding experiment.

3 Setting up RiboVIEW

3.1 Input and output directory

Setup starts with defining the folder where input BAM files are located. These synthetic BAM files are provided with **RiboVIEW** package, they are located in the package subfolder `inst/extdata/`. The following command :

```
> idir <- paste(system.file(package="RiboVIEW"), "/extdata/",sep="")
> idir
```

will locate this folder on the user's computer. Second, the address of the output directory, which is where all plots, tables and RData files will be placed, should be specified, using for example :

```
> pathout <- paste(tempdir(),"/output-files-vignette-RiboVIEW/",sep="")
```

This command defines an address in a temporary directory, using command tempdir(). It is advised to define one's own address for the output folder :

```
> pathout <- "/Path/to/my/RiboVIEW/output/directory/"
```

When doing this, make sure to include a final slash "/" character. Run the following commands to create the output folder, and to navigate to it :

```
> system(paste("mkdir -p ",pathout,sep=""))
> setwd(pathout)
```

3.2 Reading BAM, FASTA and CDS annotation files

The addresses of synthetic BAMs provided in **RiboVIEW** package are provided by the following commands. These addresses are gathered in `list.bam`.

```
> readsBAM.1.1  <- paste(idir,"Cond1-Rep1.bam",sep="")
> readsBAM.1.2  <- paste(idir,"Cond1-Rep2.bam",sep="")
> readsBAM.1.3  <- paste(idir,"Cond1-Rep3.bam",sep="")
> readsBAM.2.1  <- paste(idir,"Cond2-Rep1.bam",sep="")
> readsBAM.2.2  <- paste(idir,"Cond2-Rep2.bam",sep="")
> readsBAM.2.3  <- paste(idir,"Cond2-Rep3.bam",sep="")
> list.bam <- list(readsBAM.1.1, readsBAM.1.2, readsBAM.1.3,
+                  readsBAM.2.1, readsBAM.2.2, readsBAM.2.3)
```

Similarly, addresses for reference sequences file (FASTA format), and CDS (CoDing Sequence) annotation file (tab-separated format), are read as follows :

```
> # Reference sequences for mRNAs
> refFASTA=paste(idir,"synth.fasta",sep="")
> # Reference annotation for mRNAs' CDS
> refCDS=paste(idir,"synth.tsv",sep="")
```

It's the responsibility of the user to generate CDS annotation in this tabular format. We provide the utility `gtf2table.py` as a help to generate this annotation from a GTF (General Transfer Format) file, as provided for example from Ensembl page "Downloads" / "Download data by FTP" : https://www.ensembl.org/info/data/ftp/. The tabular annotation should look like the following :

ID	localStart	localEnd	strandDir	exonLength
mRNA0	78	486	+	545
mRNA1	131	455	+	491
mRNA2	64	1384	+	1408
mRNA3	146	761	+	836
(etc.)				

3.3 Describing the experiment

Typically, an experiment will have a `Control` (or `Wild-type`) condition and at least one `Intervention` (or `GeneKO`, etc.) condition. Here, since data is simulated, conditions are neutrally named `Condition1` and `Condition2`, and sample names are `C1.R1` for condition 1, replicate 1, `C1.R2` for condition 1, replicate 2, etc. Further, a numeric version of conditions 1 and 2 is coded by integers 1 and 2.

Condition names, condition numerals and sample names are passed on to **RiboVIEW** using following lines :

```
> XP.conditions   <- c("cond1","cond1","cond1","cond2", "cond2","cond2")
> XP.conditions.i <- c( 1,1,1,2,2,2)
> XP.names        <- c("C1.R1", "C1.R2", "C1.R3",
+                      "C2.R1", "C2.R2", "C2.R3")
```

The variable `XP.conditions.i` corresponds to `XP.conditions`, where condition 1 is coded "1" and condition 2 is coded "2".

4 Periodicity check and creation of enrichment files

Function "periodicity" creates a table of coverage in -20nt to +20nt (nt : nucleotide) around the A-site, for footprint lengths in [25;32]nt, using the following command line :

```
> periodicity(list.bam, refCDS, refFASTA, pathout, XP.names,
+   versionStrip = FALSE)
```

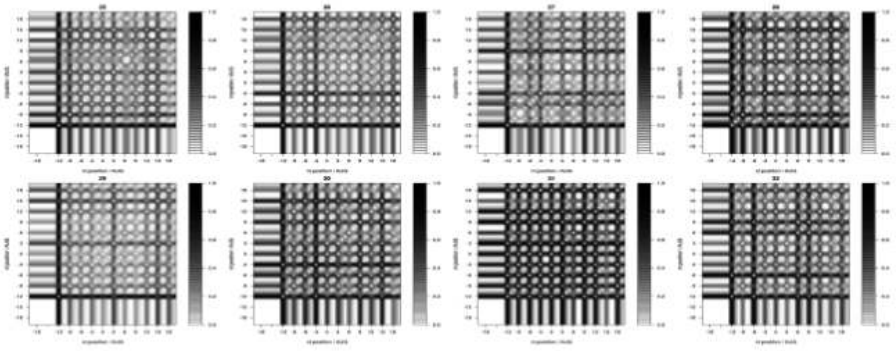
This function prints the number of transcripts found in FASTA file, and how many of them have a correct annotation. It also gives the address of the output file. For example, for condition 1, replicate 1, the address will be <pathout>/Sample-C1.R1_OL.txt. This file looks like :

position	25	26	27	28	29	30	31	32
-20	0	0	0	0	0	0	0	0
-19	0	0	0	0	0	0	0	0
-18	0	0	0	0	0	0	0	0
-17	0	0	0	0	0	0	0	0
-16	0	0	0	0	0	0	0	0
-15	0	0	0	0	0	0	0	0
-14	0	0	0	0	0	0	0	0
-13	0	0	0	0	0	0	0	0
-12	25	24	25	17	28	20	15	18
-11	4	2	6	7	2	6	4	4
-10	1	0	1	1	0	1	0	0
(etc.)								

It is now possible to select which footprint lengths are appropriate for further analysis, by using function `select.FPlen` and following instructions :

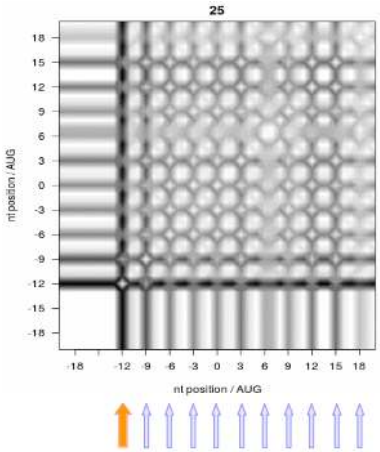
```
> attach(listminmax <- select.FPlen(list.bam, pathout, XP.names))

 Please select footprint lengths with - sufficient data,
                                      - sufficient periodicity, and
                                      - peak at -12 nt.
 (Expected peak at -12 not -15 since transition A-P at initiation
is almost immediate.)
Note acceptable footprint lengths, then press [enter] to continue.
```

The recurrence plots for all footprint lengths in one sample appear :



and the user is asked to pick appropriate footprint lengths for this sample.

There should be a peak at initiation of ribosome, in P-site, at -12nt and not before (black bar designated by an orange arrow in the figure below), and there should be periodic coverage every 3nt from -12nt to +18nt (blue arrows).



Here, footprint length of 25nt is appropriate. Looking successively at other footprint lengths, it appears that footprint lengths of 26nt, 27nt, ..., 32nt are also appropriate, in condition 1 and replicate 1.

This is repeated for replicates 2 and 3 of condition 1, and for all 3 replicates of condition 2. As a result, lengths 25nt to 32nt are suitable in all samples.

After that, function `selecFPlen` prompts for the minimum acceptable value :

```
Enter minimum acceptable footprint length
 (this will be applied for all samples),
 then press [enter] to continue :
```

Type 25 as the minimum and press enter. Now, one should see the following :

```
Similarly, enter MAXimum acceptable footprint length
 then press [enter] to continue :
```

Type 32 as the maximum and press enter.

As this first basic quality control is done, it is now possible to calculate codon enrichment, occupancy, etc. for each sample, by running the following command line :

enrichmentNoccupancy(list.bam, refCDS, refFASTA, mini, maxi, XP.names,pathout, versionStrip = FALSE)

```
Please enter the number of cores you would like to use for parallel
computations :
 (recommended value for your computer : 7),
 then press [enter] to continue.
```

The function `enrichmentNoccupancy` is tuned to parallelize calculations for faster generation of results. By default, it will take the suggested value (all cores minus 1, so that the user's computer stays available for other tasks). Nevertheless, the user can pick another value if relevant. Here we select 6, since there are only as many samples, and we type enter.

Calculations start, and an evaluation of duration is shown, as well as a list of outputs generated. These outputs are :

File name "Sample-C1.R1_..."	Description
BCO	Codon occupancy for A-site, P-site, E-site and 3 codons upstream and downstream
Codon-enrichment-unbiased_mean	Unbiased codon enrichment for A-site, P-site, E-site and 90 codons upstream and downstream, average over mRNAs
Codon-enrichment-unbiased_sd	Unbiased codon enrichment for A-site, P-site, E-site and 90 codons upstream and downstream, standard deviation over mRNAs
Hussmann_mean	Codon enrichment following Hussmann et al. 2015 (doi : 10.1371/journal.pgen.1005732), average
Hussmann_sd	Codon enrichment following Hussmann et al. 2015 (doi : 10.1371/journal.pgen.1005732), standard deviation
Sample-C1.R1.limCod	Codon count at footprint start and footprint end
Sample-C1.R1.limNt	Nucleotide count at footprint start and footprint end
Sample-C1.R1.metagene	Coverage of codons in A-site at standardized positions along CDS, to construct metagene plot
Sample-C1.R1.nbreads	Count of footprints per mRNA
Sample-C1.R1.paused	Repertoire of paused sites (in A site, codons 3 times more paused than codons in the vicinity)
Sample-C1.R1.RPKM	Count of reads per kilobase exon per million mapped
Sample-C1.R1.SCO	Single-Codon Occupancy = single-codon coverage in A-site / average coverage in an mRNA

These files constitute the basis for most quality controls and visualization plots and metrics. When calculations stop and a new prompt appears, all these files will have been created in the output folder defined earlier (paragraph 3.1). Now, everything is ready for creating visualization and quality control values and plots.

5 Visualization

Visualization relies on mean, standard deviation and standard error by condition, as well as ratios between conditions. This is calculated by function `generate.m.s` :

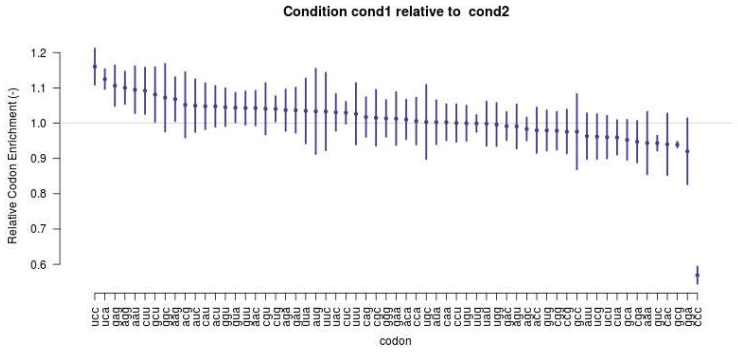
```
> generate.m.s(XP.conditions, XP.names, pathout, B=1000)
```

Next, enrichment plots, tracks and Venn diagrams are created by the following :

```
> visu.m.s.enrichmnt.res <- visu.m.s.enrichmnt(XP.conditions,
+   XP.names, pathout)
> visu.m.s.enrichmnt.res
> visu.tracks.res <- visu.tracks(XP.conditions, XP.names, pathout,
```

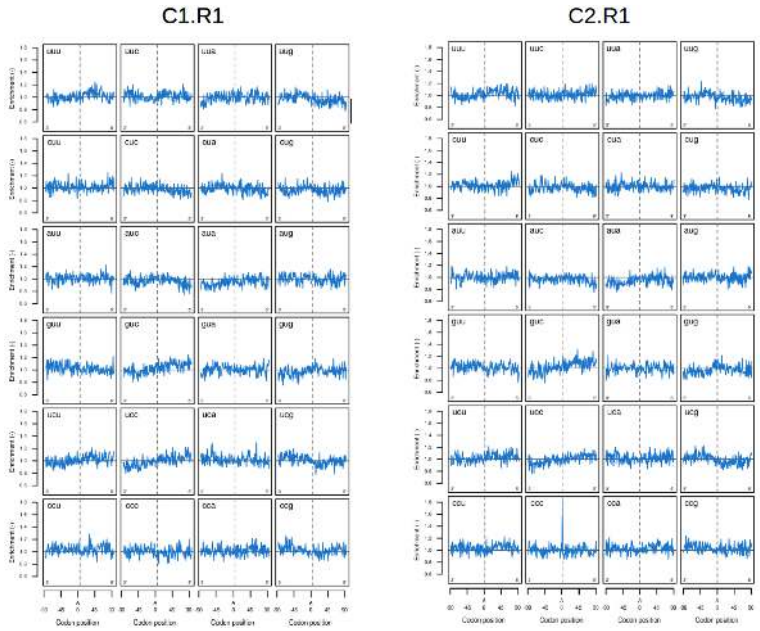
```
+   refCDS, mRNA="random", codon.labels=FALSE,
+   codon.col="darkslateblue")
> visu.tracks.res
> Venn.all.res <- Venn.all(XP.names, pathout)
> Venn.all.res
> enricht.aroundA.res <- enricht.aroundA(XP.conditions,
+   XP.names, pathout)
> enricht.aroundA.res
```

These functions do not return plots directly, instead they store them in the output folder, and register their addresses in variables *visu.m.s.enrichmnt.res*, *visu.tracks.res*, *Venn.all.res* and *enricht.aroundA.res*. These addresses are later used in HTML reports. For example, enrichment for 2 comparisons, stored as *visu.m.s.enrichmnt.res*$*plot.enrich*, is given in the next figure :



In agreement with simulated conditions, Proline-CCC codon is less enriched in condition 1 as compared to condition2, as is visible on this figure.

Furthermore, enrichment of each codon in and around the A-site is displayed as follows (truncated view) :



This corresponds to files `enrichment-all_C1.R1` and `enrichment-all_C2.R1` (.png or .eps depending on the format of these plots), stored in the output folder which was defined previously. Again, Proline-CCC enrichment is clearly visible in condition 2 as compared to condition 1. Some profiles display noise or seeming trends, for instance codon GUC, however this corresponds to randomness only. Both noise and perceived patterns smooth out when generating more reads, for instance 100000 instead of 20000 in the present case (this can be done with the Python script provided in appendix B, using NbReads=100000).

Visualization graphics are integrated later on in the "RiboMINE" html report.

6 Quality controls

6.1 Replicates

Replicates consistency is examined using the intersection of translated mRNAs in several replicates (Venn diagram per condition), footprint count correlation at gene level (permissive, since 0.90 is easily attained, even in relatively poor datasets), and at codon level (more stringent, only high coverage datasets can reach high correlation levels). Finally, a heatmap with hierarchical clustering reveals if replicates do effectively cluster more closely together than if they were chosen at random.

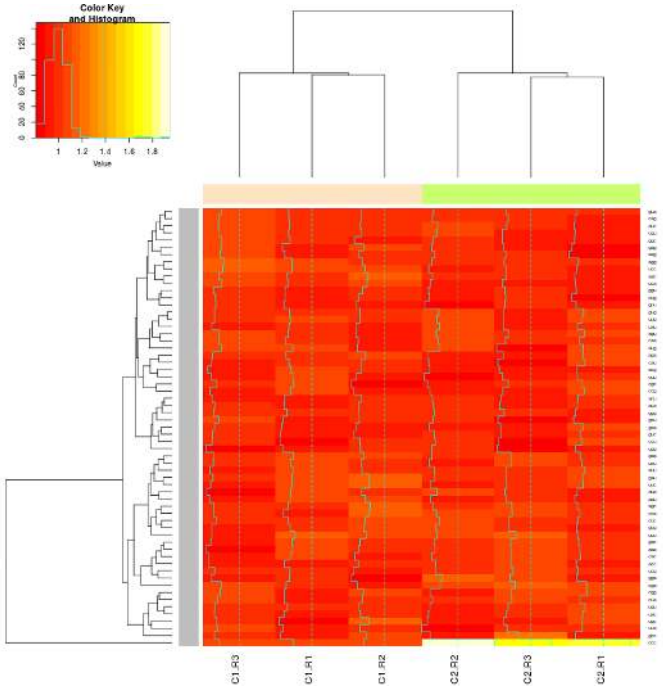
These quality controls are created using following commands :

```
> repl.correl.counts.Venn.res <- repl.correl.counts.Venn(XP.conditions,
+   XP.names, pathout)
> repl.correl.counts.Venn.res
> repl.correl.gene.res <- repl.correl.gene(XP.conditions, XP.names,
```

```
+   pathout)
> repl.correl.gene.res
> repl.correl.codon.res <- repl.correl.codon(list.bam, refCDS,
+   refFASTA, mini, maxi, XP.names, XP.conditions, pathout)
> repl.correl.codon.res
> repl.correl.heatmap.res <- repl.correl.heatmap(XP.conditions.i,
+   XP.names, pathout)
> repl.correl.heatmap.res
```

Again, these functions do not return plots directly, but store them in the output folder. Addresses of plots, and metrics for each quality control are stored in variables *repl.correl.counts.Venn.res*, *repl.correl.gene.res*, *repl.correl.codon.res* and *repl.correl.heatmap.e.res*. A brief text interpreting the quality value is also given. These addresses, values and text are later used in HTML reports.

Below is for example the heatmap plot :



The dendrogram clearly indicates separation between condition 1 and condition 2. Furthermore, Spearman correlation coefficient between conditions and actual clustering is 1.0, unsurprisingly. This correlation coefficient is given by variable *repl.correl.heatmap.e.res*$*value*.

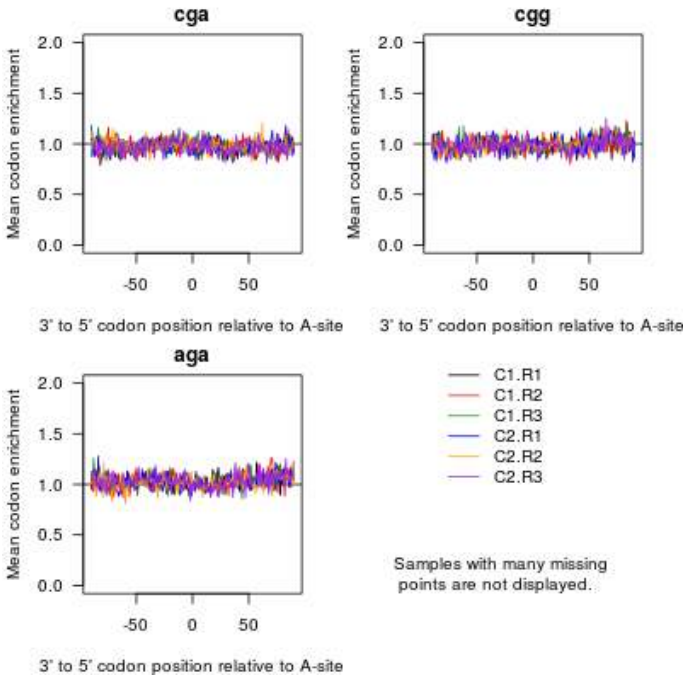
6.2 Drug artifacts

Presence or not of drug artifacts is examined using Arginine codons enrichment in a window of +/- 90nt around A-site, and using logo plots of nucleotides and codons at footprint boundaries.

Quality controls for drugs artifacts are run as follows :

```
> chx.artefacts.res <- chx.artefacts(XP.conditions, XP.names,
+   pathout)
> chx.artefacts.res
> ntcodon.freq.nt.res <- ntcodon.freq.nt(XP.conditions, XP.names,
+   pathout)
> ntcodon.freq.nt.res
> ntcodon.freq.cod.res <- ntcodon.freq.cod(XP.conditions,
+   XP.names, pathout)
> ntcodon.freq.cod.res
```

The graphic corresponding to Arginine codons enrichment is as follows :



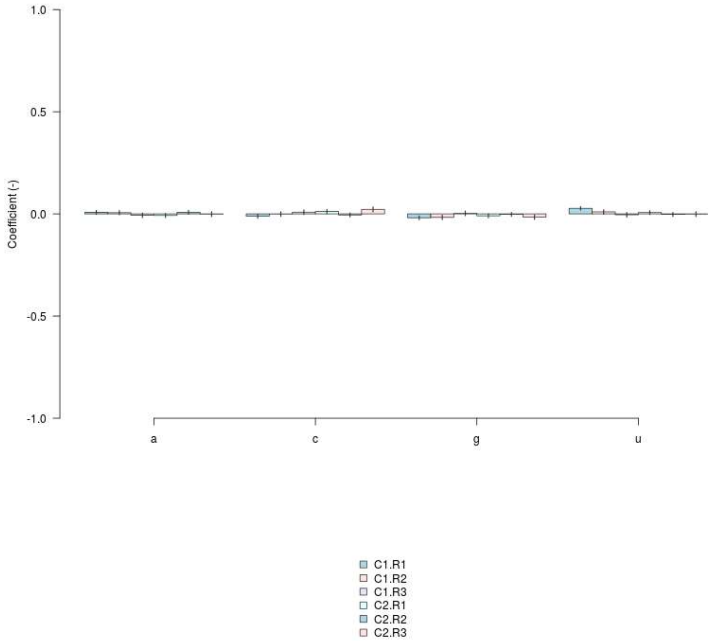
Notably, enrichment for all codons is given in the HTML report `Results-Mine.html`, since functional differences might also appear on some other codons.

6.3 Batch effects

Complementing replicates lookup, unsupervised search of batch effects is performed.

```
> batch.effects.lm.e.res <- batch.effects.lm.e(XP.conditions, XP.names, pathout)
> batch.effects.lm.e.res
> batch.effects.pca.res <- batch.effects.pca(XP.conditions, XP.names, pathout)
> batch.effects.pca.res
```

Potential batch effects on nucleotides are given below:



This plot displays the coefficient of a linear fit of codon enrichment relative to a, c, g or u-content in codons, for each replicate. If we denote k the coefficient, or slope, of the linear fit of condition 1, replicate 1 enrichment on c-content, then any additional c in codons is associated with a k*1 increase (or decrease in case k is negative) mean enrichment of these codons. Clearly, coefficients are very small in regard with the range of values [-1 ; 1]. This y-axis range can be larger if necessary, but it cannot be smaller, with the purpose of avoiding any over-interpretion of a slope which is tiny enough to be irrelevant, which is the case here.

Nevertheless, there could be a relevantly high coefficient, with error bars giving an indication if the coefficient could be significantly different from 0. More precisely, p-values are given in $batch.effects.lm.res$recommendation.mat$, as well as linear fit coefficients and associated standard errors. These values for replicate 1 in condition 1 are reproduced here :

```
> library(gridExtra)
> grid.table(round(batch.effects.lm.res$recommendation.mat, 4)[,1:3])
```

	C1.R1, Coeff	StdErr	p
a	0.0076	0.0097	0.4358
c	-0.0113	0.0101	0.2668
g	-0.0189	0.0092	0.0436
u	0.0272	0.0085	0.0022

The lowest p-value after correction for multiple testing is given in $batch.effects.lm.res$value$, as follows :

```
> batch.effects.lm.res$value

[1] 0.01329084
```

This adjusted p-value corresponds to the fit on u-content in condition 1, replicate 1 (raw p-value = 0.0022). It is lower than the cutoff 0.05 and therefore significant, however, as highlighted above, the corresponding coefficient is too small to be relevant (this coefficient is 0.0272 per additional u). This likely corresponds to a false positive, since the false positive rate, using 0.05 as a cutoff, is 5% of the 16 computed p-values, that is to say about 1 false positive p-value in this set of values.

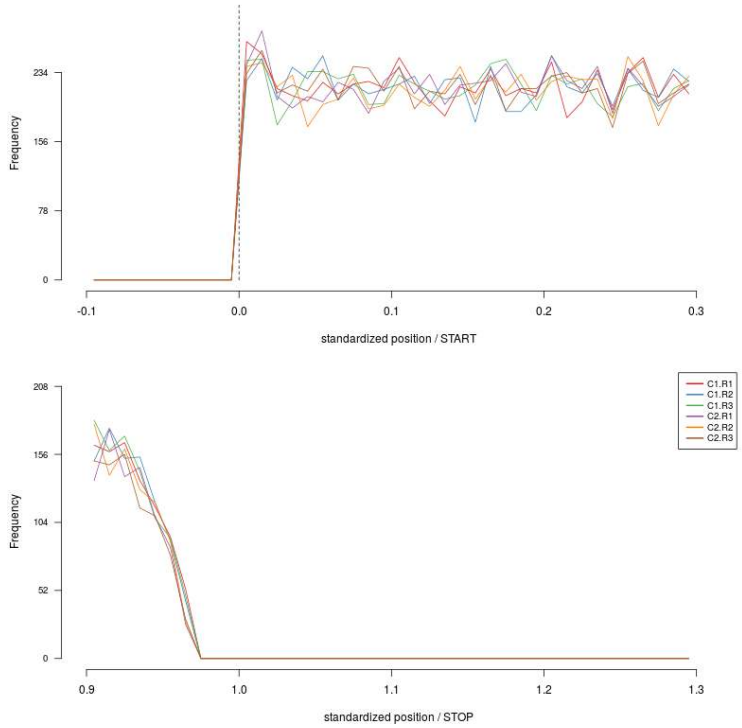
6.4 Metagene

Further, plotting footprint density along a normalized transcript (metagene) allows to visualize if monosome selection was adequate, if there were inflation of reads just after AUG (ribosome leakage in case of incomplete initiation inhibition), or if there was leakage (readthrough) at one of the STOP codons.

These quality controls are generated using :

```
> metagene.res <- metagene.all(XP.conditions, XP.names, pathout)
> metagene.monosome.res <- metagene.res[[1]]
> metagene.monosome.res
> metagene.inflation.res <- metagene.res[[2]]
> metagene.inflation.res
> metagene.leakage.res <- metagene.res[[3]]
> metagene.leakage.res
```

For instance, leakage at AUG or STOP codon would be apparent on the plot below :



In this ideal case, in accordance with simulation setting, there is no leakage. This is further described by the associated *value* variable in *metagene.leakage.res* :

```
> metagene.leakage.res$value
```

```
$start
$start$p
[1] 1


$stop
$stop$median
[1] 0

$stop$sd
[1] 0
```

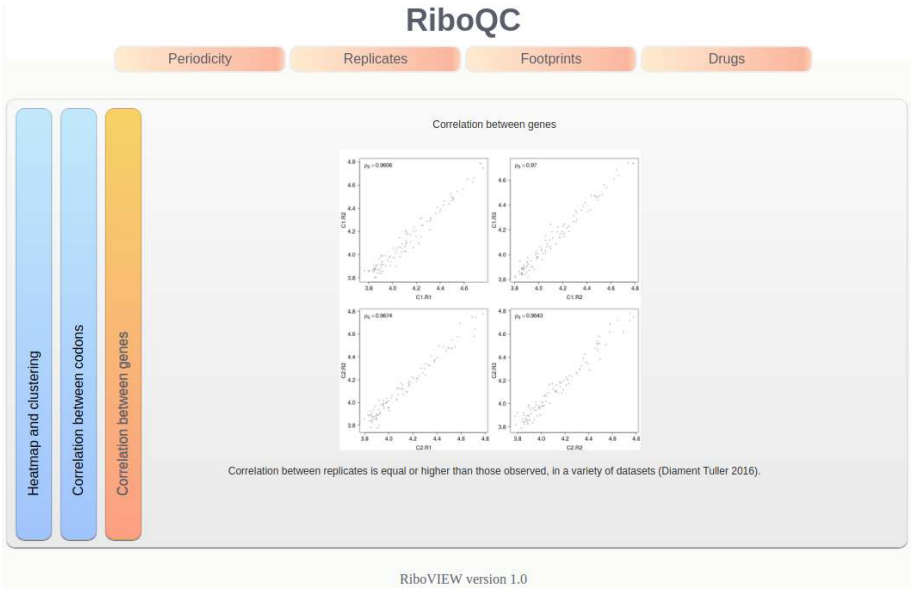
7 HTML outputs

7.1 Quality control

HTML report for quality control is generated using the following command :

```
> outputQc(pathout, XP.conditions)
```

This creates the file `Results-Qc.html`. Opened with an internet browser (the page worked fine under Chrome, Safari, Firefox, Brave and IE), the report looks like the following :



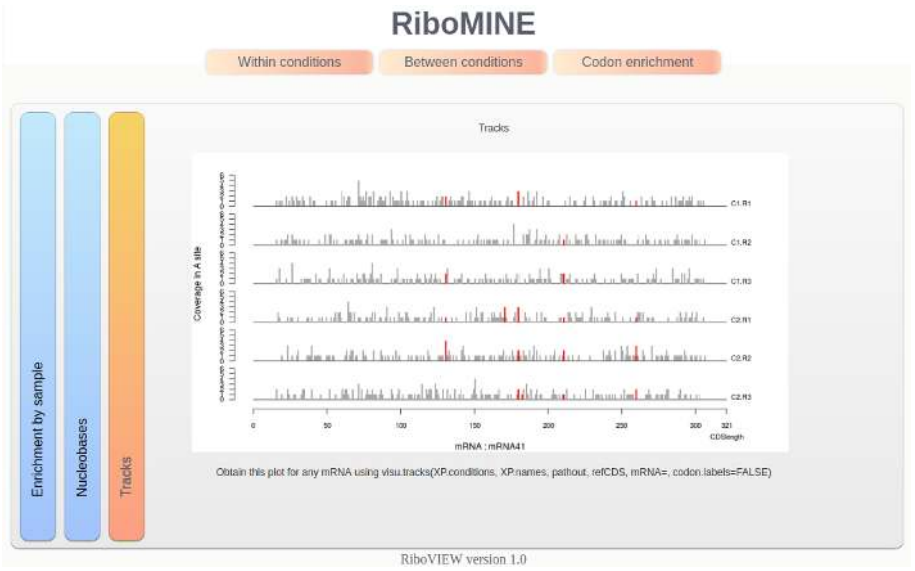
This instance of the report corresponds to category `Replicates`, tab `Correlation between genes`. The user can then navigate by selecting one of the four categories `Periodicity`, `Replicates`, `Footprints` and `Drugs`, and by clicking the desired tab inside of each category.

7.2 Visualization

HTML report for visualization is generated using the following command :

```
> outputMine(pathout, XP.conditions)
```

This creates the file `Results-Mine.html`. Similarly, the report opened in an internet browser looks as follows :



This instance of the report corresponds to category `Within conditions`, tab `Tracks`. Again, the user can navigate by selecting one of the categories `Within conditions`, `Between conditions` and `Codon enrichment` and clicking the tab of interest.

A Example workflow from SRA to BAM files

This workflow is adapted to a situation where SRA files have been downloaded from Gene Expression Omnibus (GEO) database (https://www.ncbi.nlm.nih.gov/geo). Files from GEO usually bear names like `SRR<some numbers>.sra`. This workflow is accordingly tailored to such file names. The user is free to adapt this aspect. Furthermore, adapter trimming might need some modifications, depending on experimental and sequencing setting.

```
#!/bin/bash

path=/home/user/Riboview/data/GSExxxxx/SRA
cd $path

# Check consistency of SRA files
for sample in SRR1 \
              SRR2 \
              SRR3 \
              SRR4
do
  vdb-validate $sample'.sra'
done &> screenoutput_sra-validate-stats.log


#### SRA to FASTQ.gz and FASTQC1
mkdir Fastqc1_Downloaded
for sample in SRR1 \
              SRR2 \
              SRR3 \
              SRR4
do
  echo $sample
  # sra to fastq.gz
  fastq-dump -Z --gzip $sample.sra > $sample.fastq.gz
  # fastqc
  fastqc $sample.fastq.gz -o Fastqc1_Downloaded
  echo
done &> screenoutput_Fastqc1.log &


#### Remove adapters
mkdir Fastqc2_Cutadapt
for sample in `ls SRR[1234].fastq.gz | sed 's/.fastq.gz//g'`
do
  echo $sample
  # cutadapt v1.8.1
  cutadapt  --format=fastq             \
            -a CTGTAGGCACCATCAAT       \
            --error-rate=0.1           \
```

```
            --times=2                   \
            --overlap=1                 \
            --output=$sample.cutadapt.gz \
             $sample.fastq.gz
  # fastqc
  fastqc $sample.cutadapt.gz -o Fastqc2_Cutadapt
done  &> screenoutput_cutadaptNfastqc2.log &


## Quality-trim both ends and inside (sliding window), size-select, and
     (optionally, not done here) remove adapter
mkdir Fastqc3_QCed
for sample in SRR1 \
              SRR2 \
              SRR3 \
              SRR4
do
  echo $sample
  # Quality trimming using Trimmomatic
  java -jar /home/user/prog/Trimmomatic-0.38/trimmomatic-0.38.jar \
           SE \
           -phred33 \
           $sample.fastq.gz  \
           $sample.QCed.fq.gz   \
           LEADING:30           \
           TRAILING:30          \
           MINLEN:25            \
           CROP:36
  # fastqc
  fastqc $sample.QCed.fq.gz -o Fastqc3_QCed
done  &> screenoutput_QCedNfastqc3.log &


### Depletion of rRNA, tRNA, MT-trmRNAetc
# Build reference
bowtie-build \
  /home/user/Ref/organism_trRNA_MT-trmRNA_v1.fasta \
                      organism_trRNA_MT-trmRNA_v1
# Align with Bowtie
for sample in `ls SRR[1234].QCed.fq.gz | sed 's/.QCed.fq.gz//g'`
do
  echo $sample
  gzip -dc $sample.QCed.fq.gz | bowtie \
                                        --sam \
                                        --seedmms 2 \
                                        --seedlen 11 \
                                        --seed 494123 \
                                        --maqerr 70 \
                                        --tryhard \
                                        -k 1 \
```

```
                                        --un $sample.
                                            deplof_trRNA_MTtrmRNA.fq \
                                        --best \
                                        --maxbts 800 \
                                        organism_trRNA_MT-trmRNA_v1 \
                                        - \
                                    | samtools view -hb - > $sample.
                                        trRNA_MTtrmRNA.bam
done &> screenoutput_Align.depl.log &


### Align to mRNA
# Build reference
bowtie-build \
    /home/user/Ref/organism_mRNA_v1.fasta \
                        organism_mRNA_v1
# Align, uniquely, with Bowtie
for sample in `ls SRR[1234].fastq.gz | sed 's/.fastq.gz//g'`
do
  echo $sample
  bowtie \
          --sam \
          --seedmms 2 \
          --seedlen 11 \
          --seed 764351 \
          --maqerr 70 \
          -m 1 \
          organism_mRNA_v1 \
          $sample.deplof_trRNA_MTtrmRNA.fq \
     | samtools view -hb - > $sample.mRNA.bam
done &> Align.mRNA.screen &


#### Checksums
for fichier in *.mRNA.bam
do
  echo $fichier
  md5sum $fichier > $fichier.md5sum
done
```

B Python script for synthetic BAM files

This Python script generates synthetic mRNAs (`synth.fasta`), their coding sequence annotation (`synthetic.tsv`), and simulates footprint reads from this pool of mRNAs for 2 conditions and 3 replicates per condition. Condition 1 is a control condition, while condition 2 exhibits enriched Pro-CCC codon. This is simulated by a doubled probability of sampling a footprint with A-site located at a Pro-CCC codon. This script requires SAMtools (http://www.htslib.org), as well as Python packages pysam and numpy to run properly.

```
#!/usr/bin/python
#
# Purpose : generate fasta, tsv file and bam files for package-included
#   synthetic files.
#
# Setting : Codon Proline CCC enriched (slower) in condition 2 :
#             sampling probability *2 for this codon.
#
#
import pysam
import os
import random

## Parameters
NbReads = 20000
filedir  = "./synth-files-vignette/"
# Create input directory
os.system("mkdir -p "+filedir)
tsvFile = filedir+"synth.tsv"
faFile  = filedir+"synth.fasta"
seed    = 7234
# Set random seed
random.seed(seed)
Nmrna  = 100
# t and not u because samtools considers only t.
bases  = ['a','c','g','t']
# Codons to sample from
codons = ['aaa','aac','aag','aat','aca','acc','acg','act','aga','agc',
          'agg','agt','ata','atc','atg','att','caa','cac','cag','cat',
          'cca','ccc','ccg','cct','cga','cgc','cgg','cgt','cta','ctc',
          'ctg','ctt','gaa','gac','gag','gat','gca','gcc','gcg','gct',
          'gga','ggc','ggg','ggt','gta','gtc','gtg','gtt','tac','tat',
          'tca','tcc','tcg','tct','tgc','tgg','tgt','tta','ttc','ttg',
          'ttt']
stops  = ['taa','tag','tga']
NbCond = 2
NbRep  = 3

## Initialize synthetic mRNAs in virtual table name_i,length_i,
```

```
    sequence_i
#  Note : no UTRs and here lenNt = lenCDS (+3nt of stop codon)
mRNAname     = [None] * Nmrna
CDSlenC      = [None] * Nmrna
CDSlenNt     = [None] * Nmrna
mRNAnt5UTR   = [None] * Nmrna
mRNAnt3UTR   = [None] * Nmrna
mRNAlenNt    = [None] * Nmrna
mRNAseq      = [None] * Nmrna
Pos_CCC      = [None] * Nmrna
outFA        = open(faFile, 'w')
outTSV       = open(tsvFile, 'w')
outTSV.write("ID\tlocalStart\tlocalEnd\tstrandDir\texonLength\n")

## Generate synthetic mRNAs
for i in range(Nmrna) :

    mRNAname[i]    = "mRNA"+str(i)
    CDSlenC[i]     = random.randint(50,500)
    CDSlenNt[i]    = 3*CDSlenC[i]
    mRNAnt5UTR[i]  = random.randint(20,200)
    mRNAnt3UTR[i]  = random.randint(0,100)
    mRNAlenNt[i]   = mRNAnt5UTR[i] + CDSlenNt[i] + mRNAnt3UTR[i]

    # Elements for sequence construction
    mRNA_Pre5      = ''.join([random.choice(bases) for ii in range(
        mRNAnt5UTR[i])])
    mRNA_PreC0     = [random.choice(codons) for ii in range(CDSlenC[i
        ]-2)]
    mRNA_PreC      = ''.join(mRNA_PreC0)
    mRNA_Stop      = random.choice(stops)
    mRNA_Pre3      = ''.join([random.choice(bases) for ii in range(
        mRNAnt3UTR[i])])

    # Construct a sequence
    mRNAseq[i]     = mRNA_Pre5 + 'atg' + mRNA_PreC + mRNA_Stop +
        mRNA_Pre3
    # ii+1 since AUG is not yet in mRNA_PreC
    # (ii+1)<(CDSlenC[i]-15) to yield valid footprints later on
    Pos_CCC[i]     = [ii+1 for ii,x in enumerate(mRNA_PreC) if x=="ccc"
         if (ii+1)<(CDSlenC[i]-15)]

    # Write entry to fasta
    outFA.write(">"+mRNAname[i]+"\n")
    for j in range(mRNAlenNt[i]/100) :
        outFA.write(mRNAseq[i][(j*100):(j*100+100)]+"\n")

    j=j+1
    outFA.write(mRNAseq[i][(j*100):]+"\n")
```

```
    # Write entry to TSV
    outTSV.write(mRNAname[i]+"\t"+str(mRNAnt5UTR[i])+"\t"+str(
        mRNAnt5UTR[i]+CDSlenNt[i])+"\t+\t"+str(mRNAlenNt[i])+"\n")

outFA.close()
outTSV.close()


## Generate SAM/BAM file for each condition and replicate
for Cond in range(1,1+NbCond) :
    for Rep in range(1,1+NbRep) :

        samFile = filedir+"Cond"+str(Cond)+"-Rep"+str(Rep)+".sam"
        bamFile = filedir+"Cond"+str(Cond)+"-Rep"+str(Rep)+".bam"

        ## Open SAM output file for writing
        with open(samFile, 'w') as fout :

            ## Create sam file header
            fout.write("@HD\tVN:1.0\tSO:unsorted\n")
            for i in range(Nmrna) :
                fout.write("@SQ SN:"+mRNAname[i]+"      LN:"+str(
                    mRNAlenNt[i])+"\n")

            fout.write("@PG     ID:Synthetic\n")

            ## Create sam file entries
            for i in range(NbReads) :
                imRNA = random.randint(0,Nmrna-1)

                Qname = "C"+str(Cond)+"-R"+str(Rep)+"."+str(i)
                Flag  = "0"
                Rname = mRNAname[imRNA]

                # Sample positions that would yield valid footprints :
                #   pos cannot be in the last 35nt of CDS,
                #    - with periodicity ("1+3*random.randint(...)"),
                #    - locate in P-site if AUG, else in A (AP_Offset),
                #    - mostly in-frame ("random.choice([0]*50+...").

                # More weight (pause) on Pro-CCC, in Cond2
                k = 2
                indices = (range(CDSlenC[imRNA] - 15) + (k-1)*Pos_CCC[
                    imRNA])
                Pos_codonInCDS_inequal_pre = random.choice(indices)
                Pos_codonInCDS_inequal  = 1 + 3 *
                    Pos_codonInCDS_inequal_pre  #0,(CDSlenC[imRNA]-15))

                # Sample equally along CDS
                Pos_codonInCDS_equally = 1 + 3 * random.randint(0,(
```

```
        CDSlenC[imRNA]-15))

    # Attribute footprint reads with equal or unequal
        sampling depending on experimental condition
    if Cond==2 :
        Pos_codonInCDS = Pos_codonInCDS_inequal
    else :
        Pos_codonInCDS = Pos_codonInCDS_equally

    AP_Offset = -15
    if Pos_codonInCDS == 1 :
        AP_Offset = -12

    # Elements for SAM/BAM file entry
    Pos = mRNAnt5UTR[imRNA] + Pos_codonInCDS + AP_Offset +
        random.choice([0]*50+[1]*10+[2]*1)
    Mapq  = "255"
    Rnext = "*"
    Pnext = "0"
    Tlen  = random.randint(25,32) #      Tlen  = "30"
    Cigar = str(Tlen)+"M"
    Seq   = mRNAseq[imRNA][(Pos-1):(Pos-1+Tlen)]
    Qual  = "".join(Tlen*["Z"])

    # Write to SAM
    fout.write(Qname+"\t"+Flag+"\t"+Rname+"\t"+str(Pos)+"\t
        "+Mapq+"\t"+
               Cigar+"\t"+Rnext+"\t"+Pnext+"\t"+str(Tlen)+"
                   \t"+Seq+"\t"+Qual+"\n")


## Convert sam to bam
os.system("samtools view -hb "+samFile+" > "+bamFile)
```

C Troubleshooting

C.1 Output files are not located in the working directory.

When defining the working directory, the user should make sure to include a final "/".

```
> wdir <- "/Path/to/my/working/directory/"
```

C.2 RiboVIEW cannot find a file.

Were all commands run, following the same order as in the template ? Some files and plots produced at one step are indeed necessary for other commands later on, so that it's preferable to run these commands in full and in the recommended order.

Or, possibly, was the output folder moved to another place ? Most commands rely on the output folder specified in variable *pathout* and on $XP.names$, $XP.names.i$ and $XP.conditions$. Make sure these didn't change inadvertently. If one of these variables was changed on purpose, it's probably safest to make a backup of the old output directory, and to run RiboVIEW freshly from start with these newly defined variables.

C.3 R returns a warning like : "The following objects are masked from (...)"

This might happen when running "attach(listminmax <- select.FPlen(list.bam, pathout, XP.names))" twice. If the values in 'mini' and 'maxi' are the ones the user selected, it's fine to ignore this message. Otherwise, it is safer to run this function in 2 steps :

```
> listminmax <- select.FPlen(list.bam, pathout, XP.names)
> mini <- listminmax[[1]]
> maxi <- listminmax[[2]]
```