Rp-Bp

Release 3.0.0

Etienne Boileau

Sep 13, 2023

CONTENTS

CHAPTER

ONE

INTRODUCTION

Ribosome profiling (Ribo-seq) is an RNA-sequencing-based readout of RNA translation. Isolation and deep-sequencing of ribosome-protected RNA fragments (ribosome footprints) provides a genome-wide snapshot of the translatome at sub-codon resolution. Aligned by their P-site positions, the footprints from actively translating ribosomes should exhibit a 3-nt periodicity. To select reads and predict translation, most methods, including **Rp-Bp**, take advantage of this periodic signal.

Rp-Bp is an unsupervised Bayesian approach to predict translated open reading frames (ORFs) from ribosome profiles, using the automatic Bayesian Periodic fragment length and ribosome P-site offset Selection (BPPS), *i.e.* read lengths and ribosome P-site offsets are inferred from the data, without supervision. **Rp-Bp** is able to handle *de novo* translatome annotation by directly assessing the periodicity of the Ribo-seq signal.

Rp-Bp can be used for ORF discovery, or simply to estimate periodicity in a set of Ribo-seq replicates, *e.g.* to know which samples and read lengths are usable for downstream analyses. When used for ORF discovery, **Rp-Bp** automatically classifies ORFs into different biotypes or categories, relative to their host transcript.

1.1 Getting started

1.1.1 What is Rp-Bp?

Rp-Bp is an unsupervised Bayesian approach to predict translated open reading frames (ORFs) from ribosome profiles. **Rp-Bp** can be used for ORF discovery, or simply to estimate periodicity in a set of Ribo-seq samples.

To get started, you need

- Ribo-seq data (FASTQ)
- genome sequence and annotation for your organism (FASTA, GTF)
- ribosomal sequence for *in-silico* rRNA removal (FASTA)
- protocol-specific or general adapter sequences to be removed (FASTA)

1.1.2 Installation

Install with

```
# set up the conda channels if required
# conda config --add channels defaults
# conda config --add channels bioconda
# conda config --add channels conda-forge
# conda config --set channel_priority strict

# create a conda environment called rpbp and install rpbp
conda create -n rpbp rpbp
```

or use a container

```
# docker or...
docker pull quay.io/biocontainers/rpbp:<tag>
# ...singularity
singularity pull rpbp.sif docker://quay.io/biocontainers/rpbp:<tag>
```

There is no *latest* tag, you need to specify the version tag. See rpbp/tags for valid values for `<tag>`.

For detailed installation instructions, refer to Installation.

1.1.3 Rp-Bp quickstart

In a nutshell, you need to prepare genome indices and annotations for your organism by calling

```
prepare-rpbp-genome <config> [options]
```

To estimate periodicity on a set of Ribo-seq samples, or to run the ORF discovery pipeline, simply call

```
run-all-rpbp-instances <config> [options]
```

To get started, the package also includes a small example using a *C. elegans* dataset. Check the Tutorials.

For more information and guidelines on how to prepare the configuration file, refer to the User guide. For visualisation and quality control, see Visualization and quality control.

1.1.4 How to report issues

Bugs and issues should be reported in the bug tracker. Follow the instructions and guidelines given in the template.

1.1.5 How to contribute

Contributions are welcome! New code should follow Black and flake8. Install development dependencies inside a virtual environment, see *Contributing to Rp-Bp*. A typical development workflow would include *(i)* forking the repository, *(ii)* creating a new branch for your PR, *(iii)* adding features or bug fixes, *(iv)* making sure all tests are passing, *(v)* building the documentation if necessary, and *(vi)* opening a PR back to the main repository. If you're fixing a bug, add a test. Run it first to confirm it fails, then fix the bug, and run it again to confirm it's fixed. If adding a new feature, add a test, or first open an issue to discuss the idea.

Running the tests

We use pytest to test **Rp-Bp**. Currently, only regression tests are implemented. Dependencies can be installed with `pip install -e .[tests]`.

Building the docs

Dependencies for building the documentation can be installed with `pip install -e .[docs]`.

Semantic versioning

We try to follow semantic versioning.

1.1.6 How to cite

Brandon Malone, Ilian Atanassov, Florian Aeschimann, Xinping Li, Helge Großhans, Christoph Dieterich. Bayesian prediction of RNA translation from ribosome profiling, *Nucleic Acids Research*, Volume 45, Issue 6, 7 April 2017, Pages 2960-2972.

1.1.7 License

1.2 Installation

1.2.1 Containers

To use a container (Docker or Singularity) with **Rp-Bp** pre-installed, simply pull, and you're done!

```
# docker or...
docker pull quay.io/biocontainers/rpbp:<tag>
# ...singularity
singularity pull rpbp.sif docker://quay.io/biocontainers/rpbp:<tag>
```

There is no *latest* tag, you need to specify the version tag. See rpbp/tags for valid values for <tag>. Check the Tutorials on how to use the containers.

1.2.2 Conda installation

If required, set up the conda channels as described here, and install with

```
# preferably install in some conda environment...
conda install rpbp
```

or create an environment, called *rpbp*, containing the **Rp-Bp** package

```
conda create -n rpbp rpbp
```

Tip: Mamba can be used as a drop-in replacement, you can swap almost all commands between conda and mamba.

1.2.3 Contributing to Rp-Bp

To install the local VCS project in development mode

```
# create a conda environment...
conda create -n rpbp_dev
# ...activate it...
conda activate rpbp_dev
# ... and only install dependencies (rpbp_dev is now activated)
conda install --only-deps rpbp
# clone the git repository
git clone https://github.com/dieterich-lab/rp-bp.git && cd rp-bp
# install
pip --verbose install --no-deps -e . 2>&1 | tee install.log
```

PyPI installation

We do not recommend to install **Rp-Bp** directly from PyPI. However, if you already have the required dependencies installed on your system, to install

```
# create a virtual environment...
python3 -m venv rpbp_pypi
# ... activate it ...
source rpbp_pypi/bin/activate
# ... and install Rp-Bp (rpbp_pypi is now activated)
pip install rpbp
```

Required dependencies: Flexbar, Bowtie 2, STAR, Samtools.

Warning: Conda installation or containers include all dependencies. With a PyPI installation, you need to install required dependencies. Executables or binaries must be in your `$PATH`.

1.2.4 Uninstallation

Remove the conda environment

```
conda env remove --name rpbp
```

or remove the package installed in another environment

```
# remove the rpbp package from myenv environment...
conda remove -n myenv rpbp
```

To remove **Rp-Bp** if installed with pip

```
pip uninstall rpbp
```

If the package is installed in a dedicated python virtual environment, this environment can also be removed.

1.3 User guide

1.3.1 The Rp-Bp pipeline

You want to "quality control" your Ribo-seq samples for downstrean analyses, or run the Ribo-seq ORF discovery pipeline? First, you need to prepare genome indices and annotations for your organism. This has to be done once for any given reference genome and annotation.

The pipeline itself consists of two "modules": the *ORF profile construction*, where periodic read lengths and ribosome P-site offsets are inferred from the data; and the *translation prediction*, where translation events are predicted.

1.3.2 How to prepare the configuration file

A single YAML configuration file can be used for both index creation and running the pipeline. The following keys are read from the configuration file:

Index creation

- `gtf` *(required, input)* Path to the reference annotations (format GTF2).
- `fasta` *(required, input)* Path to the reference genome sequence (format FASTA).
- `ribosomal_fasta` *(required, input)* Path to the ribosomal sequence (format FASTA).
- `de_novo_gtf` *(optional, input)* An additional GTF containing annotations constructed from a *de novo* assembly. See More about prepare-rpbp-genome.
- `start_codons` *(optional, input)* A list of strings to use as start codons. Default: `ATG`.
- `stop_codons` *(optional, input)* A list of strings to use as stop codons. Default: `TAA`, `TGA`, `TAG`.
- `genome_name` *(required, output)* A descriptive name to use for the created files.
- `genome_base_path` *(required, output)* Output path (directory) for the transcript fasta and ORFs.
- `ribosomal_index` *(required, output)* Output path (directory/filename) for the Bowtie 2 index.
- `star_index` *(required, output)* Output path (directory) for the STAR index.
- `orf_note` *(optional, output)* An additional description used in the filenames. It should not contain spaces or special characters.

Pipeline

In addition to the above required keys:

- `riboseq_samples` *(required, input)* A dictionary *key: value*, where *key* is used to construct filenames, and *value* is the full path to the FASTQ.gz file for a given sample. The *key* should not contain spaces or special characters.
- `riboseq_biological_replicates` *(optional, input)* A dictionary *key: value*, where *key* is a condition, and *value* contains all samples which are replicates of the condition. Items of the *value* list must match the `riboseq_samples` *key*.
- `adapter_file` *(optional, input)* Path to adapter sequences to be removed (FASTA).
- `adapter_sequence` *(optional, input)* A single adapter sequence to be removed.

- `riboseq_data` *(required, output)* The base output location for all created files.
- `riboseq_sample_name_map` *(optional, output)* A dictionary *key: value*, where *key* is the same as `riboseq_samples` *key*, and *value* is a fancy name for *key* to use in downstream analyses.
- `riboseq_condition_name_map` *(optional, output)* A dictionary *key: value*, where *key* is the same as `riboseq_biological_replicates` *key*, and *value* is a fancy name for *key* to use in downstream analyses.
- `project_name` *(optional, output)* An additional description used in the filenames for downstream analyses. It should not contain spaces or special characters. See *Visualization and quality control*.
- `note` *(optional, output)* An additional description used in the filenames. It should not contain spaces or special characters.

A configuration file with the above required keys is sufficient to run the pipeline with default parameters. To change the default parameters, see *Default parameters and options*. A "template" configuration file is available to download with the Tutorials. See also *More about biological replicates*, for an example of how to use biological replicates.

1.3.3 How to prepare genome indices and annotations

This has to be done once for any given reference genome and annotation.

Attention: Under the hood, **Rp-Bp** uses **STAR** to align reads to the genome. If the **STAR** version changes, the **STAR** index may have to be re-generated.

General usage

```
prepare-rpbp-genome <config> [options]
```

Rp-Bp can be run with the SLURM scheduler. For all options, consult the API. See also *How to prepare the configuration file*.

Required input

Reference annotations

The reference annotations (format GTF2), with *transcript*, *exon*, and *CDS* features (*start_codon* and *stop_codon* features are not required). The attribute field must include *transcript_id* and *gene_id*, and optionally *transcript_biotype*, *gene_name*, and *gene_biotype*. The annotations must match the version of the reference genome sequence.

Caution: The GTF2 format does not include the stop codon in the terminal exon, *i.e.* the stop codon is not included in the CDS. This is important, as **Rp-Bp** treats annotations according to the GTF2 (GTF) specifications.

Reference genome sequence

The input FASTA file should contain all top level sequence regions *excluding haplotypes* (this may significantly increase runtime, and bias the alignments, since these sequences can substiantially overlap the main reference). This generally matches the *primary assembly* file from Ensembl for the chosen assembly release. The identifiers must match those in the GTF annotation file.

Ribosomal sequence

A separate FASTA file for the ribosomal DNA (rDNA) sequence/cluster, which is generally not included in the genome assembly. This file can also include other sequences to filter out, depending on the goal of the analysis (*.e.g* snRNAs). We typically include the following

- The large and small ribosomal subunit sequences, *e.g.* from NCBI.
- The genomic tRNA sequences *e.g.* from GtRNAdb.
- Mt_rRNA, Mt_tRNA and rRNA genes from BioMart. In particular, we select those options for the "Gene type" filter. For "Attributes", we select "Sequences", and then specifically "Exon sequences". Additionally, including the "Gene type" in the header can be helpful for identifying where reads mapped, for quality control purposes.

Output files

Output files are written in the BED, FASTA, or TAB-delimited formats.

- <*ribosomal_index*> The Bowtie 2 index files.
- <*star_index*> The STAR index files.

The base path for the following file is: <*genome_base_path*>

- <*genome_name*>*.annotated.bed.gz* A BED12+ file containing all annotated transcripts, including "biotype", "gene_id", "gene_name", and "gene_biotype" information.

The base path for the following files is: <*genome_base_path*>*/transcript-index*

- <*genome_name*>*.transcripts.annotated.fa* A FASTA file with the annotated transcript sequences.
- <*genome_name*>*.orfs-genomic[.orf_note].bed.gz*. A BED12+ with the ORFs extracted from all transcripts. The ORFs are numbered, and their length is also reported. The ORF ids are of the form: *transcript_seqname:start-end:strand*. The start codon is included, but the stop codon is not.
- <*genome_name*>*.orfs-exons[.orf_note].bed.gz*. A BED6+ file with the ORF exons. The extra columns are *exon_index*, giving the order of the exon in the transcript, and *transcript_start*, giving the start position of that index in transcript coordinates.
- <*genome_name*>*.orfs-labels[.orf_note].tab.gz*. A TAB-delimited file with ORF categories and all compatible transcripts. See More about prepare-rpbp-genome to learn about ORF categories or labels.

Note: If a `de_novo_gtf` file is provided, intermediate output files are split into *annotated* and *de-novo*. The files used by the pipeline, as described above, are the "concatenation" of the respective *annotated* and *de-novo* files. In addition, a GTF file is created by concatenating `gtf` and `de_novo_gtf`. This new GTF file is written to `genome_base_path`. Be careful not to overwrite any existing GTF file there!

1.3.4 How to run the pipeline

See *ORF profile construction* and *Translation prediction* for a short description of required input and output files. See also *More about biological replicates*.

Rp-Bp output files are written in the BED, FASTA, sparse matrix market (MTX), or CSV format. Output from Flexbar, Bowtie2, and STAR are written in FASTQ or BAM formats.

Important: All Ribo-seq samples (including biological replicates) in the configuration file must be from the same organism and use the same `genome_base_path`, `star_index`, `ribosomal_index`, *etc.* Samples from different organisms or using different annotations must be "split" into different configuration files, and run separately.

General usage

```
# Only create the ORF profiles (estimate periodicity).
run-all-rpbp-instances <config> --profiles-only [options]

# Run the ORF discovery pipeline for all samples in the configuration file (only samples,
↪ i.e. do not merge the replicates).
run-all-rpbp-instances <config> [options]

# Run the ORF discovery pipeline for all samples in the configuration file, merge the␣
↪replicates, and make predictions for merged replicates.
run-all-rpbp-instances <config> --merge-replicates --run-replicates [options]
```

Rp-Bp can be run with the SLURM scheduler. For all options, consult the API. See also *How to prepare the configuration file*.

Tip: To be able to perform read filtering quality control, use the `-k/--keep-intermediate-files` option. Intermediate files *e.g.* Flexbar, or Bowtie2 output can be deleted afterwards, see *Visualization and quality control*.

More about biological replicates

The **Rp-Bp** pipeline handles replicates by adding the ORF profiles. The Bayes factors and predictions are then calculated based on the combined profiles. The `--merge-replicates` flag indicates that the replicates should be merged. By default, if the `--merge-replicates` flag is given, then predictions will not be made for the individual samples, unless the `--run-replicates` flag is also given, in which case predictions will be made for both the merged replicates as well as the individual samples. This is how you would prepare a configuration file for four samples of two different "conditons":

```
# example of 4 samples: 2 controls and 2 conditions
# <sample_name> below is replaced by ctrl1, ctrl2, cond1, and cond2
# <condition_name> below is replaced by ctrl and cond

riboseq_samples:
 ctrl1: /path/to/sample1.fastq.gz
 ctrl2: /path/to/sample2.fastq.gz
 cond1: /path/to/sample3.fastq.gz
```

(continues on next page)

(continued from previous page)

```
 cond2: /path/to/sample4.fastq.gz

riboseq_biological_replicates:
 ctrl:
  - ctrl1
  - ctrl2
 cond:
  - cond1
  - cond2

# fancy names to use for downstream analyses
riboseq_sample_name_map:
 ctrl1: Ctrl-1
 ctrl2: Ctrl-2
 cond1: Cond-1
 cond2: Cond-2

riboseq_condition_name_map:
 ctrl: Ctrl
 cond: Cond
```

1.3.5 ORF profile construction

To run the periodicity estimation only, pass the `--profiles-only` option.

Note: This part of the pipeline uses Flexbar, Bowtie2, and STAR to process and align Ribo-seq reads, however you can estimate periodicity (and predict translation events) using your own existing alignment files (BAM format), see How to use existing alignment files

---

Required input

All the input files are those specified by the configuration file.

Output files

The base path for the following files is: *<riboseq_data>/without-adapters*

- *<sample_name>[.note].fastq.gz* Clean reads (adapters and low-quality reads removed).

The base path for the following files is: *<riboseq_data>/with-rrna*

- *<sample_name>[.note].fastq.gz* Reads aligning to the ribosomal index. They may be kept for quality control, but are not used.

The base path for the following files is: *<riboseq_data>/without-rrna*

- *<sample_name>[.note].fastq.gz* Reads not aligning to the ribosomal index, *i.e.* after *in-silico* rRNA removal. These reads are used for the genome alignment step.

The base path for the following files is: *<riboseq_data>/without-rrna-mapping*

- *<sample_name>[.note].bam* A sorted BAM file with genome alignments (the *Aligned.sortedByCoord.out.bam* STAR output).
- *<sample_name>[.note]-unique.bam* A sorted BAM file with unique alignments (multimapping reads removed).

Note: If `keep_riboseq_multimappers` is `True` in the configuration file, then there will be no *-unique* files. In general, we do not recommend to keep multimappers.

The base path for the following files is: *<riboseq_data>/metagene-profiles*

- *<sample_name>[.note][-unique].metagene-profile.csv.gz* A CSV file with the metagene profiles constructed from aligned reads (given by the "position" or offset and "count" columns) for all read lengths ("length" column) found in a given sample. It include profiles for the annotated translation initiation site and translation termination site ("type" column).
- *<sample_name>[.note][-unique].metagene-periodicity-bayes-factors.csv.gz* A CSV file with the model outputs and Bayes factor estimates for all P-site offsets and read lengths.
- *<sample_name>[.note][-unique].periodic-offsets.csv.gz* A CSV file with the best P-site offset for each read length. All read lengths are included, even if the estimates do not meet the prediction criteria (filtering occurs on the fly).

The base path for the following files is: *<riboseq_data>/orf-profiles*

- *<sample_name>[.note][-unique].length-<lengths>.offset-<offsets>.profiles.mtx.gz* A MTX file with the profiles for all ORFs ("orf_num", "orf_position", *i.e.* position within the ORF, and "read_count"). The matrix market format uses base-1 indexing!
- *<condition_name>[.note][-unique].profiles.mtx.gz* Same as above for condition, if using `--merge-replicates`.

1.3.6 Translation prediction

Without the `--profiles-only` option, the pipeline will predict which ORFs show evidence of translation, using only the periodic footprint lengths. The `--merge-replicates` options is used to predict translation events in merged profiles, see *More about biological replicates*.

Tip: If you first created profiles and estimated periodicity using the `--profiles-only` option, you can decide to continue with the translation prediction step at a later stage. You only have to `run-all-rpbp-instances <config> [--merge-replicates] [--run-replicates]` using the same configuration file. Steps for which output files already exists will be skipped, unless the `--overwrite` option is set.

Required input

All the input files are those specified by the configuration file. In addition, metagene and ORF profile output files are required (see output files from *ORF profile construction*). If the pipeline is run sequentially, you do not normally have to worry about the intermediate output.

Output files

The base path for the following files is: *<riboseq_data>/orf-predictions*

- *<sample_name>[.note][-unique].length-<lengths>.offset-<offsets>.bayes-factors.bed.gz* A BED12+ file with model outputs for all ORFs. Additional columns include the ORF number, ORF length, model outputs, Bayes factor mean and variance, and P-site coverage across 3 frames.
- *<sample_name>[.note][-unique].length-<lengths>.offset-<offsets>[.filtered].predicted-orfs.bed.gz* Same format as above, with the predicted translation events. **This file contains the translated Ribo-seq ORFs**.
- *<sample_name>[.note][-unique].length-<lengths>.offset-<offsets>[.filtered].predicted-orfs.dna.fa* A FASTA file with the predicted translation events. The FASTA header matches the "id" column in the corresponding BED file. **This file contains the DNA sequence for each translated Ribo-seq ORF**.
- *<sample_name>[.note][-unique].length-<lengths>.offset-<offsets>[.filtered].predicted-orfs.protein.fa* A FASTA file with the predicted translation events. The FASTA header matches the "id" column in the corresponding BED file. **This file contains the protein sequence for each translated Ribo-seq ORF**.
- *<condition_name>[.note][-unique].bayes-factors.bed.gz* Same as above for condition, if using `--merge-replicates`.
- *<condition_name>[.note][-unique][.filtered].predicted-orfs.bed.gz* Same as above for condition, if using `--merge-replicates`.
- *<condition_name>[.note][-unique][.filtered].predicted-orfs.dna.fa* Same as above for condition, if using `--merge-replicates`.
- *<condition_name>[.note][-unique][.filtered].predicted-orfs.protein.fa* Same as above for condition, if using `--merge-replicates`.

Attention: Translation events are predicted using Bayesian model selection. Our model does not distinguishes between overlapping ORFs. To select the best overlapping ORF among a group of overlapping ORFs, we first select the longest ORF, then the highest Bayes factor. This is referred to as the *filtered* predictions.

In previous versions, both *filtered* and *unfiltered* (including all overlapping ORFs) predictions were written to file. In general, we recommend to use *filtered* predictions. Unless the `--write-unfiltered` option is used, **Rp-Bp** now only outputs the *filtered* predictions. If using `--write-unfiltered`, *unfiltered* predictions are also written to file, without the *[.filtered]* flag. Hence to avoid confusion with older results, the *filtered* predictions have kept the *[.filtered]* flag.

Note: If *smoothing parameters* (see *Default parameters and options*) are given in the configuration file, the following string *.frac-<smoothing_fraction>.rw-<smoothing_reweighting_iterations>* is also added to the file names. Default values (unless they are explicitly given in the configuration file) are not written.

1.3.7 Default parameters and options

The parameters and options decribed below are all optional. All parameters and options have default values that do not normally need to be modified.

Important: **Rp-Bp** parameters can be changed via the configuration file, and options for external programs (Flexbar, STAR) are handled via command line arguments. You do not need to include **Rp-Bp** parameters in the configuration file, unless you wish to change their values.

Flexbar and STAR options

Default options for external programs (Flexbar, STAR) are overridden via command line using `--flexbar-options` or `--star-options`. Currently, no options can be passed to Bowtie2.

Flexbar

- `max-uncalled` Default: 1.
- `pre-trim-left` Default: 0.
- `qtrim-format` Default: sanger.
- `qtrim` Default: TAIL.
- `qtrim-threshold` Default: 10.
- `zip-output` Default: GZ.

STAR

- `readFilesCommand` Default: zcat (gzcat for macOS).
- `limitBAMsortRAM` Default: 0 (set to `--mem` at run-time).
- `alignIntronMin` Default: 20.
- `alignIntronMax` Default: 100000.
- `outFilterMismatchNmax` Default: 1.
- `outFilterMismatchNoverLmax` Default: 0.04.
- `outFilterType` Default: BySJout.
- `outFilterIntronMotifs` Default: RemoveNoncanonicalUnannotated.
- `outSAMattributes` Default: AS NH HI nM MD.
- `outSAMtype` Default: BAM SortedByCoordinate.
- `sjdbOverhang` Default: 33.
- `seedSearchStartLmaxOverLread` Default: 0.5.
- `winAnchorMultimapNmax` Default: 100.

Rp-Bp parameters

- `keep_riboseq_multimappers` If `True` in the configuration file, then multimapping riboseq reads *will not* be removed. They will be treated as "normal" reads in every place they map, *i.e.* the weight of the read will not be distributed fractionally, probabilistically, *etc.* We do not in general recommend to use this option.
- `models_base` The path to the compiled models, if installed in a different location. The models are included with the source distribution and compiled as part of the installation. *Do not change this, unless you know what you are doing!*

Shared MCMC parameters

- `seed` The random seed for the MCMC sampling, used for periodicity estimation and translation prediction. Default: 8675309.
- `chains` The number of chains to use in the MCMC sampling, used for periodicity estimation and translation prediction. Default: 2

Metagene and periodicity estimation parameters

- `metagene_start_upstream` The number of bases upstream of the translation initiation site to begin constructing the metagene profile. Default: 300.
- `metagene_start_downstream` The number of bases downstream of the translation initiation site to end the metagene profile. Default: 300.
- `metagene_end_upstream` The number of bases upstream of the translation termination site to begin constructing the metagene profile. Default: 300.
- `metagene_end_downstream` The number of bases downstream of the translation termination site to end the metagene profile. Default: 300.
- `periodic_offset_start` The position, relative to the translation initiation site, to begin calculating periodicity Bayes factors. Default: -20 (inclusive).
- `periodic_offset_end` The position, relative to the translation initiation site, to stop calculating periodicity Bayes factors. Default: 0 (inclusive).
- `metagene_profile_length` The length of the profile to use in the models. `metagene_profile_length` + `periodic_offset_end` must be consistent with the length of the extracted metagene profile. Default: 21.
- `metagene_iterations` The number of iterations to use for each chain in the MCMC sampling. Default: 500 (includes warmup).
- `min_metagene_profile_count` Read lengths with fewer than this number of reads will not be used. Default: 1000.
- `min_metagene_bf_mean` If `max_metagene_bf_var` and `min_metagene_bf_likelihood` are None (null in YAML), this is taken as a hard threshold on the estimated Bayes factor mean. Default: 5.
- `max_metagene_bf_var` A hard threshold on the estimated Bayes factor variance. Default: None.
- `min_metagene_bf_likelihood` A threshold on the likelihood of periodicity. Default: 0.5.

Note: A profile is periodic if [P(bf > `min_metagene_bf_mean`)] > `min_metagene_bf_likelihood`. By default, we do not filter on the variance. If given, then both filters are applied and the result is the intersection.

Fixed lengths and offsets

- `use_fixed_lengths` If `True` in the configuration file, fixed values given by `lengths` and `offsets` are used (no periodicity estimation).
- `lengths` A list of read lengths to use for creating the profiles if the `use_fixed_lengths` option is `True`. Presumably, these are lengths that have periodic metagene profiles.
- `offsets` The P-site offset to use for each read length specifed by `lengths` if the `use_fixed_lengths` option is `True`. The number of offsets must match the number of lengths, and they are assumed to match. For example `lengths`: [26, 29] with `offsets`: [9, 12] means only reads of lengths 26 bp and 29 bp are used to create the profiles. The 26 bp reads will be shifted by 9 bp in the 5' direction, while reads of length 29 bp will be shifted by 12 bp.

Smoothing parameters

- `smoothing_fraction` The fraction of the data used when estimating each y-value for LOWESS. Default: 0.2.
- `smoothing_reweighting_iterations` The number of residual-based reweightings to perform for LOWESS. See the statsmodels documentation. Default: 0.

Translation prediction parameters

- `orf_min_length_pre` ORFs with length < `orf_min_length_pre` (nucleotides) are not processed. Default: 0 (ignore option).
- `orf_max_length_pre` ORFs with length > `orf_max_length_pre` (nucleotides) are not processed. Default: 0 (ignore option).
- `orf_min_length` Only ORFs with length > `orf_min_length` (nucleotides) are kept in the final set. Default: 8.
- `orf_min_profile_count_pre` ORF with profile sum < `orf_min_profile_count_pre` are not processed. Default: 5.
- `orf_min_profile_count` Only ORFs with profile sum > `orf_min_profile_count` are kept in the final set. Default: None.
- `translation_iterations` The number of iterations to use for each chain in the MCMC sampling. Default: 500 (includes warmup).
- `min_bf_mean` If `max_bf_var` and `min_bf_likelihood` are None (null in YAML), this is taken as a hard threshold on the estimated Bayes factor mean. Default: 5.
- `max_bf_var` A hard threshold on the estimated Bayes factor variance. Default: None.
- `min_bf_likelihood` A threshold on the likelihood to select an ORF as translated. Default: 0.5.
- `chisq_alpha` For the chi-square test, this value is first Bonferroni corrected based on the number of ORFs which pass the smoothing filters. It is then used as the significance threshold to select translated ORFs. Default: 0.01.

Note: A Ribo-seq ORF is translated if [P(bf > `min_bf_mean`)] > `min_bf_likelihood`. By default, we do not filter on the variance. If given, then both filters are applied and the result is the intersection.

Attention: Chi-square values are reported, but they are not used for prediction, unless the `chi_square_only` flag is present in the configuration file, in which case the translation models are not fit to the data, and the posterior distributions are not estimated. This is mostly kept for historical reasons, and may eventually be removed.

1.4 Visualization and quality control

Rp-Bp provides two *interactive dashboards* or *web applications*, one for read and periodicity quality control, the other to facilitate Ribo-seq ORFs discovery. The latter has an integrated IGV browser for the visual exploration of predicted Ribo-seq ORFs. To navigate the apps is easy, just follow the "hints". Most items are interactive.

Before you can visualize the results, you need to prepare the input for the dashboards.

1.4.1 How to prepare the input for the dashboards

Summarizing the profile construction

To prepare the input for the *profile construction dashboard*

```
summarize-rpbp-profile-construction <config> [options]
```

With the `-c/--create-fastqc-reports` flag, FastQC reports are be created. For all options, consult the API.

Required input

Output files from the *ORF profile construction* step.

Output files

The base path for these files is: *<riboseq_data>/analysis/profile_construction*.

- *<project_name>[.note].read-filtering-counts.csv.gz* A CSV file with read counts after each step of the pipeline, one row per sample, with the following columns: "note" (sample name), "raw_data_count" (reads in the original FASTQ), "without_adapters_count" (reads remaining after running Flexbar), "without_rrna_count" (reads remaining after rRNA removal), "genome_count" (genome alignments), "unique_count" (unique genome alignments), "length_count" (periodic reads).
- *<project_name>[.note].length-distribution.csv.gz* A CSV file with counts of aligned reads for each length, one row per sample.
- *<project_name>[.note][-unique].periodic-offsets.csv.gz* A CSV file with all P-site offsets and read lengths for each samples.
- *<project_name>[.note][-unique].frame-counts.csv.gz* A CSV file with P-site adjusted coverage across 3 frames summed across all ORFs for each sample.

Summarizing the Rp-Bp predictions

To prepare the input for the *predictions dashboard*

```
summarize-rpbp-predictions <config> [options]
```

For all options, consult the API.

Required input

Output files from the *translation prediction* step.

Output files

The base path for these files is: *<riboseq_data>/analysis/rpbp_predictions*.

- *<project_name>[.note][-unique][.filtered].predicted-orfs.bed.gz* A BED12+ file with the combined predicted translation events from all samples and replicates. Additional columns include sample or replicate, Bayes factor mean and variance, P-site coverage across 3 frames, ORF number, ORF length, label, host transcript biotype, and associated gene id, name and biotype, and compatible transcripts.
- *<project_name>[.note][-unique][.filtered].igv-orfs.bed.gz* A BED12 file with all unique ORFs.
- *<project_name>.summarize_options.json* A json summary file for the app.
- *<genome_name>.circos_graph_data.json* A json file with the ORF distribution across chromosomes.

Hint: Use *<project_name>[.note][-unique][.filtered].predicted-orfs.bed.gz* has a final output combining all predictions across your samples and/or replicates (including Bayes factor mean and variance for each sample, *etc.*). If you need a unique list of Ribo-seq ORFs (only coordinates *i.e* standard BED12, without duplicated entries for samples and/or replicates), use *<project_name>[.note][-unique][.filtered].igv-orfs.bed.gz*.

1.4.2 How to launch the web applications

To launch the *profile construction dashboard*

```
rpbp-profile-construction-dashboard -c CONFIG
```

The application has multiple views to facilitate quality control, *e.g.*

To launch the *predictions dashboard*

```
rpbp-predictions-dashboard -c CONFIG
```

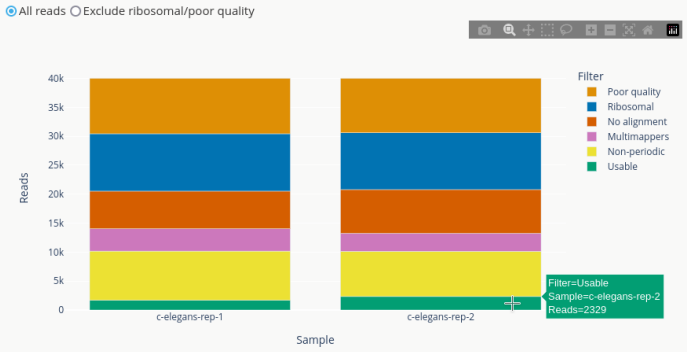
The application has multiple views to facilitate ORF discovery, including an integrated IGV browser for the visual exploration of predicted Ribo-seq ORFs, *e.g.*

Try it out, and see more!

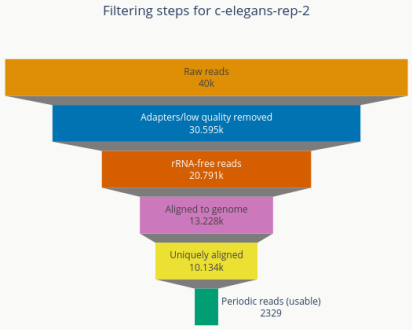
For all options, consult the API.

Note: Any of the above command will open a browser page with the web application running locally. You can also specify a `--host` and a `--port`, *e.g.* if launching the app from a remote server. In the latter case, you have to

1. Stacked bars and funnel charts showing the number of reads after each step of the filtering pipeline

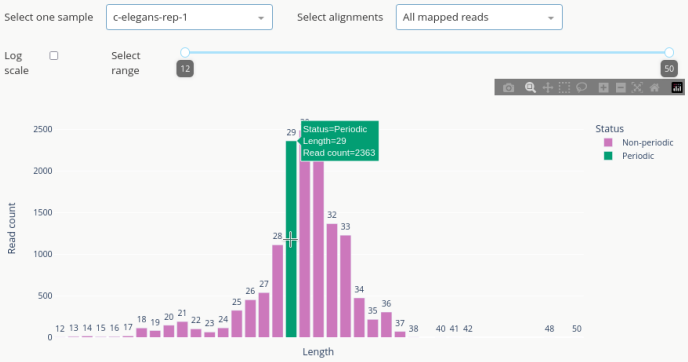


Hint: Hover over bars to create a funnel chart (right) for a given sample.

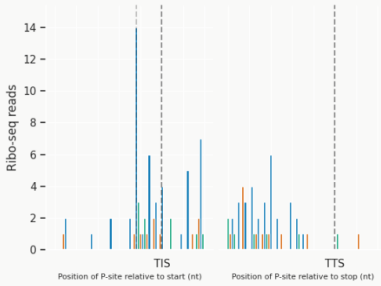


Hint: Hover over chart to see proportions.

2. Per sample read length distribution and metagene profiles



Hint: Hover over bars to see the metagene profile for this read length (right).



Read length periodicity

Selected sample: c-elegans-rep-1

Length: 29

P-site offset: -12

Status: Used for analysis

3. ORF predictions (Table)

Download CSV

Chrom	ORF_ID	ORF_length	Category	Transcript_biotype	Transcripts	Gene_ID	Gene_name	Gene_biotype
filter data...			filter data...					
I	B0041.4_I:4645739-4646957:-	345	CDS	protein_coding	B0041.4	WBGene00004415	rpl-4	protein_coding
		423	altCDS	protein_coding	B0511.10	WBGene00001228	eif-3.E	protein_coding
		432	CDS	protein_coding	B0511.10	WBGene00001228	eif-3.E	protein_coding
		270	CDS	protein_coding	B0511.5	WBGene00015231	cutl-14	protein_coding
		135	CDS	protein_coding	C04F12.4.2,C04F12.…	WBGene00004426	rpl-14	protein_coding
		198	CDS	protein_coding	C09D4.5.1,C09D4.5.…	WBGene00004431	rpl-19	protein_coding
I	C30F8.2.3_I:5080312-5082257:-	348	CDS	protein_coding	C30F8.2.3,C30F8.2.…	WBGene00016258	vha-16	protein_coding
I	C32E8.2b_I:3788041-3788859:+	88	CDS	protein_coding	C32E8.2b	WBGene00004425	rpl-13	protein_coding
I	C34B7.3_I:8341951-8346022:-	493	CDS	protein_coding	C34B7.3	WBGene00007913	cyp-36A1	protein_coding
I	C44E4.4_I:4634495-4635399:+	283	altCDS	protein_coding	C44E4.4	WBGene00016653	C44E4.4	protein_coding

Condition	BF mean	BF var	P-sites	in-frame
c-elegans-rep-1	251	19727	11	64%
c-elegans-rep-2	813	3	38	84%

open a browser page at the correct address. For example, you use `--host 123.123.123.123`, then open a page on *http://123.123.123.123:8050/*.

1.5 More about `prepare-rpbp-genome`

As part of the index creation step, **Rp-Bp** extract all putative ORFs and assign to each one a label based on its position relative to the annotated transcript-exon structure. Labels can be useful to quickly identify certain ORF types that may be of particular interest, *e.g.* upstream ORFs, or ORFs from non-coding RNAs.

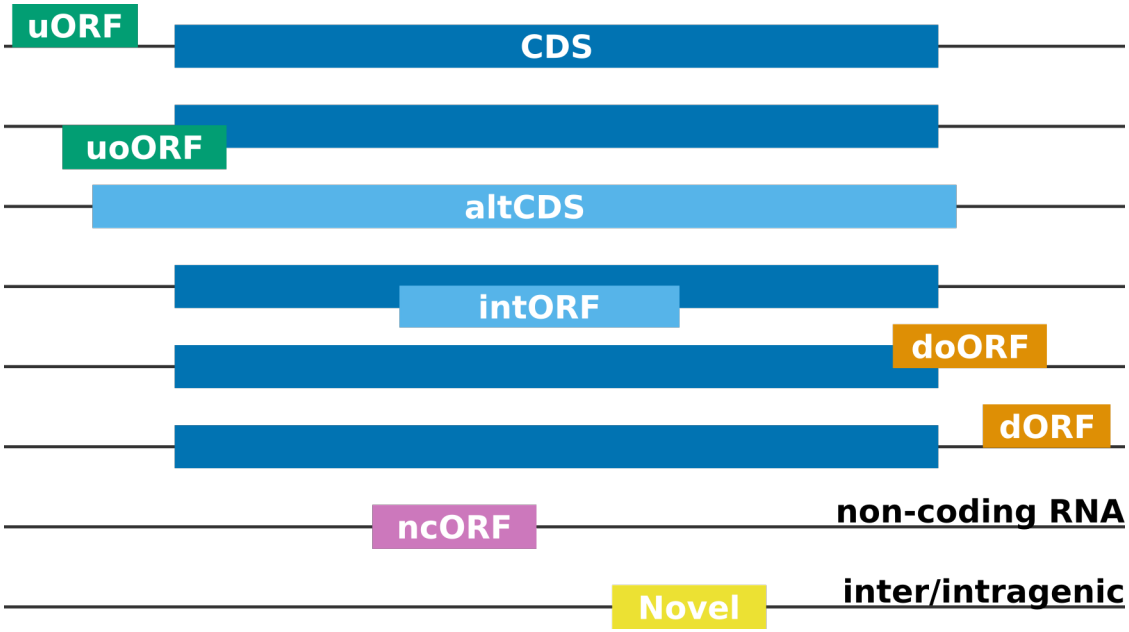
Note: The ORF "id" is of the form *transcript_seqname:start-end:strand*. *seqname* is the chromosome or contig. The start codon is included, but the stop codon is not. The host *transcript* "id" should not contain underscores!

Hint: In some cases, the ORF label may not be consistent with the host transcript, as reported by the ORF "id". To resolve such seemingly incoherent assignments, compatible transcripts are reported for each ORF in *<genome_name>.orfs-labels.annotated[.orf_note].tab.gz* and shown in the prediction dashboard (see Visualization and quality control).

1.5.1 Categories of Ribo-seq ORFs

- **CDS**: Canonical (annotated) coding sequence
- **altCDS**: Alternative CDS (*e.g.* N/C-terminus extension/truncation, alternatively spliced variants, *etc.*)
- **intORF**: Translation event within a CDS (in- or out-of-frame)
- **uORF/uoORF**: Translation event in the 5' untranslated region (UTR) of or partially overlapping an annotated protein-coding gene
- **dORF/doORF**: Translation event in the 3' untranslated region (UTR) of or partially overlapping an annotated protein-coding gene
- **ncORF**: Translation event in an RNA annotated as non-coding (lncRNA, pseudogene, *etc.*)
- **Novel**: Translation event inter- or intragenic (only when **Rp-Bp** is run with a *de novo* assembly, see below)

Labels such as *overlap* or *suspect* arise when **Rp-Bp** is not able to resolve the position of an ORF without ambiguity. In practice, for standard annotations, we do not see these categories.



1.5.2 More about *de novo* ORF discovery

For **Rp-Bp**, there is no difference between annotated and *de novo* assembled transcripts. In both cases, ORFs are extracted from the transcripts based on the given start and stop codons. However, it is often of scientific interest to identify *Novel* Ribo-seq ORFs. These are the most interesting, as they do not overlap the annotations at all, but **Rp-Bp** also identifies *Novel altCDS* and *Novel ncORF*.

Hence, when matching RNA-seq is available (same reference genome), we highly recommend to create a *de novo* assembly. The only requirement is that the assembler produces a valid GTF file (or a format that can be converted to GTF). In a *de novo* assembly, coding regions are typically not identified (that is what Ribo-seq is for!). However, if your assembly also includes CDS annotations, they must satisfy the start/stop codon GTF2 specifications (stop codon not included in the CDS.)

1.6 Tutorials

1.6.1 Run the ORF discovery pipeline

For this tutorial, we use the *c-elegans-chrI-example* with the *c-elegans-test.yaml* configuration file.

How to run the pipeline

To run the pipeline, change the paths in the configuration file to point to the location where you extracted the data, *e.g. /home/user/data/c-elegans-chrI-example*. You can do this using a text editor, or simply by modifying the file in place

```
sed -i 's|/path/to/your/c-elegans-example|'`pwd`'|g' c-elegans-test.yaml
```

Important: Default parameters were modified for the example and included in the configuration file. If you use this configuration file as a general template for your data, do not forget to remove everything below the line "REMOVE BELOW THIS LINE IF YOU USE THIS CONFIGURATION FILE AS TEMPLATE FOR YOUR DATA".

You can now create the genome indices and annotations. For this small example, it is important to downscale the STAR `--genomeSAindexNbases` parameter as follows

```
prepare-rpbp-genome c-elegans-test.yaml --star-options "--genomeSAindexNbases 10" --num-
↪cpus 4 --logging-level INFO --log-file rpbp-genome.log
```

The file *rpbp-genome.log* contains logging output for the reference preprocessing. You now have a new directory called *WBcel235.79.chrI* with genome indices and annotations.

Finally, run the ORF discovery pipeline

```
run-all-rpbp-instances c-elegans-test.yaml --merge-replicates --run-replicates --keep-
↪intermediate-files --num-cpus 4 --logging-level INFO --log-file rpbp-pipeline.log
```

The file *rpbp-pipeline.log* contains logging output for the different processing steps. You now have four new directories (*with-*, *without-*) including output from Flexbar, Bowtie2, and STAR, and directories with **Rp-Bp** output: *metagene-profiles*, *orf-profiles*, and *orf-predictions*. The *orf-predictions* include the output for each sample *c-elegans-rep-1* and *c-elegans-rep-2* as well as for the merged replicates *c-elegans-test*.

How to summarize the results and launch the apps

Prepare the summary output for the *profile construction dashboard*

```
summarize-rpbp-profile-construction c-elegans-test.yaml --num-cpus 4 --logging-level␣
↪INFO --log-file rpbp-profile-summary.log
```

and for the *predictions dashboard*

```
summarize-rpbp-predictions c-elegans-test.yaml --no-replicates --circos-bin-width 10000 -
↪-circos-show-chroms I --logging-level INFO --log-file rpbp-predictions-summary.log
```

Due to the size of the data, we reduce the bin width for the Circos plot. We also need to specify which sequences or chromosomes we want to include (by default, only numbered chromosomes and X/x, Y/y are shown). You now have a new directory *analysis* with *profile_construction* and *rpbp_predictions* output.

Launch any of the web applications with

```
rpbp-profile-construction-dashboard -c c-elegans-test.yaml
```

or

```
rpbp-predictions-dashboard -c c-elegans-test.yaml
```

To navigate the apps is easy, just follow the "hints". Most items are interactive. Press `CTRL+C` to quit.

Attention: For the apps only, the configuration file is passed using a (required) named argument `-c/--config CONFIG`.

Note: Any of the above command will open a browser page with the web application running locally. You can also specify a `--host` and a `--port`, *e.g.* if launching the app from a remote server. In the latter case, you have to open a browser page at the correct address. For example, if you use `--host 123.123.123.123`, then open a page on *http://123.123.123.123:8050/*.

1.6.2 How to use existing alignment files

For this tutorial, we use the *c-elegans-chrI-example* with the *c-elegans.alignments-only.yaml* configuration file.

The **Rp-Bp** pipeline is designed to handle all steps from raw FASTQ files up to the final list of translated Ribo-seq ORFs, but you can start the pipeline from any step.

Caution: Do not use the `--overwrite` flag!

For example, the trimming, filtering and alignment steps (performed with Flexbar, Bowtie2, and STAR) could be handled using a different preprocessing strategy and/or different software. In this case, the configuration file must be created as usual, but the `riboseq_samples` dictionary only needs to contain the *key* for the samples (the actual path can be left blank or filled with any arbitrary value). This *key* must match the name of any existing files, and these files must be placed at the appropriate location according to the naming convention and nomenclature of **Rp-Bp**.

In our example, the files should be placed (or symlinked) to the following locations:

- **Trimmed and quality filtered reads**
  <riboseq_data>/without-adapters/<sample_name>[.note].fastq.gz
- **Reads not aligning to ribosomal sequences**
  <riboseq_data>/without-rrna/<sample_name>[.note].fastq.gz
- **Aligned reads**
  <sample_name>[.note].Aligned.sortedByCoord.out.bam or *<sample_name>[.note].bam* or *<sample_name>[.note]-unique.bam* (unless the `keep_riboseq_multimappers` configuration option is given)

See the *General usage* for more details about required input and output files. Only the "last files" must be in the expected location. For example, if trimming, filtering and alignment has been performed, only the alignment files must be present.

Example

This example shows how to run **Rp-Bp** starting with the alignment files created in the first tutorial *Run the ORF discovery pipeline*.

Important: You first need to go through the first tutorial, and "run the example dataset" to be able to reproduce this example, as it uses existing alignment files!

Navigate to the same *c-elegans-chrI-example* directory (where you extracted the data). Create a new directory called *alignments-only*, and symlink existing alignment files

```
mkdir alignments-only && cd alignments-only && mkdir without-rrna-mapping && cd without-
↪rrna-mapping
# you are now inside */c-elegans-chrI-example/alignments-only/without-rrna-mapping
# symlink files using absolute or relative path
# e.g. ln -s <original-example>/without-rrna-mapping/c-elegans-rep-1Aligned.
↪sortedByCoord.out.bam .
# here we use relative paths
ln -s ../../without-rrna-mapping/c-elegans-rep-1Aligned.sortedByCoord.out.bam .
ln -s ../../without-rrna-mapping/c-elegans-rep-2Aligned.sortedByCoord.out.bam .
# return to c-elegans-chrI-example
cd ../../
```

Change the paths in the configuration file, as explained in the first tutorial, or simply modify in place

```
sed -i 's|/path/to/your/c-elegans-example|'`pwd`'|g' c-elegans.alignments-only.yaml
```

It is now important to also adjust the configuration file by adding the *alignments-only* directory at the end of the path given by the `riboseq_data` key in *c-elegans.alignments-only.yaml*

```
# e.g. if your path is /home/user/data
riboseq_data: /home/user/data/c-elegans-chrI-example/alignments-only
```

Hint: Instead of the above commands, you can also create directories using your menu/shortcuts, *etc.*, copy files by hand, and/or use a text editor to modify the configuration file.

Run the pipeline

```
run-all-rpbp-instances c-elegans.alignments-only.yaml --merge-replicates --run-
↪replicates --num-cpus 4 --logging-level INFO --log-file alignments-only.txt
```

Logging reports a few `WARNING ... Some input files are missing. Skipping call...`, but the alignment files are eventually found, and the pipeline proceeds from there. You should end up with the same final output as that obtained in the first tutorial.

1.6.3 How to use the containers

First pull a Docker or Singularity container, see installation. For this tutorial, we use a general mechanism for persisting data, which allows to create and modify files on the host system from within the container. We use the *c-elegans-chrI-example* with the *c-elegans-test.yaml* configuration file, see also How to run the pipeline.

Note: You can also launch a container with an interactive shell *e.g.* `docker run -it quay.io/biocontainers/rpbp:3.0.1--py310h30d9df9_0 /bin/bash` or `singularity shell rpbp.sif`. With `singularity shell`, `$HOME` is mounted by default.

Attention: In the following, do not forget to modify the tag `3.0.1--py310h30d9df9_0` according to what you pulled! For Singularity, adjust the name of the Singularity image format file `rpbp.sif` and/or the path according to your needs.

How to run the pipeline

To run the pipeline, change the paths in the configuration file to point to the location where the directory is mounted in the container. You can do this using a text editor, or simply by modifying the file in place

```
sed -i 's|/path/to/your/c-elegans-example|/data|g' c-elegans-test.yaml
```

Important: Default parameters were modified for the example and included in the configuration file. If you use this configuration file as a general template for your data, do not forget to remove everything below the line "REMOVE BELOW THIS LINE IF YOU USE THIS CONFIGURATION FILE AS TEMPLATE FOR YOUR DATA".

You can now create the genome indices and annotations. For this small example, it is important to downscale the STAR `--genomeSAindexNbases` parameter as follows

```
docker run --volume `pwd`:/data quay.io/biocontainers/rpbp:3.0.1--py310h30d9df9_0␣
↪prepare-rpbp-genome /data/c-elegans-test.yaml --star-options "--genomeSAindexNbases 10
↪" --num-cpus 4 --logging-level INFO --log-file /data/rpbp-genome.log
```

```
singularity run --bind `pwd`:/data rpbp.sif prepare-rpbp-genome /data/c-elegans-test.
↪yaml --star-options "--genomeSAindexNbases 10" --num-cpus 4 --logging-level INFO --log-
↪file /data/rpbp-genome.log
```

The file *rpbp-genome.log* contains logging output for the reference preprocessing. You now have a new directory called *WBcel235.79.chrI* with genome indices and annotations.

Finally, run the ORF discovery pipeline

```
docker run --volume `pwd`:/data quay.io/biocontainers/rpbp:3.0.1--py310h30d9df9_0 run-
↪all-rpbp-instances /data/c-elegans-test.yaml --merge-replicates --run-replicates --
↪keep-intermediate-files --num-cpus 4 --logging-level INFO --log-file /data/rpbp-
↪pipeline.log
```

```
singularity run --bind `pwd`:/data rpbp.sif run-all-rpbp-instances /data/c-elegans-test.
↪yaml --merge-replicates --run-replicates --keep-intermediate-files --num-cpus 4 --
↪logging-level INFO --log-file /data/rpbp-pipeline.log
```

The file *rpbp-pipeline.log* contains logging output for the different processing steps. You now have four new directories (*with-*, *without-*) including output from Flexbar, Bowtie2, and STAR, and directories with **Rp-Bp** output: *metagene-profiles*, *orf-profiles*, and *orf-predictions*. The *orf-predictions* include the output for each sample *c-elegans-rep-1* and *c-elegans-rep-2* as well as for the merged replicates *c-elegans-test*.

How to summarize the results and launch the apps

Prepare the summary output for the *profile construction dashboard*

```
docker run --volume `pwd`:/data quay.io/biocontainers/rpbp:3.0.1--py310h30d9df9_0␣
↪summarize-rpbp-profile-construction /data/c-elegans-test.yaml --num-cpus 4 --logging-
↪level INFO --log-file /data/rpbp-profile-summary.log
```

```
singularity run --bind `pwd`:/data rpbp.sif summarize-rpbp-profile-construction /data/c-
↪elegans-test.yaml --num-cpus 4 --logging-level INFO --log-file /data/rpbp-profile-
↪summary.log
```

and for the *predictions dashboard*

```
docker run --volume `pwd`:/data quay.io/biocontainers/rpbp:3.0.1--py310h30d9df9_0␣
↪summarize-rpbp-predictions /data/c-elegans-test.yaml --no-replicates --circos-bin-
↪width 10000 --circos-show-chroms I --logging-level INFO --log-file /data/rpbp-
↪predictions-summary.log
```

```
singularity run --bind `pwd`:/data rpbp.sif summarize-rpbp-predictions /data/c-elegans-
↪test.yaml --no-replicates --circos-bin-width 10000 --circos-show-chroms I --logging-
↪level INFO --log-file /data/rpbp-predictions-summary.log
```

Due to the size of the data, we reduce the bin width for the Circos plot. We also need to specify which sequences or chromosomes we want to include (by default, only numbered chromosomes and X/x, Y/y are shown). You now have a new directory *analysis* with *profile_construction* and *rpbp_predictions* output.

Launch any of the web applications with

```
docker run -p 8050:8050 --volume `pwd`:/data quay.io/biocontainers/rpbp:3.0.1--
↪py310h30d9df9_0 rpbp-profile-construction-dashboard -c /data/c-elegans-test.yaml --
↪host="0.0.0.0"
```

```
singularity run --bind `pwd`:/data rpbp.sif rpbp-profile-construction-dashboard -c /data/
↪c-elegans-test.yaml --host="0.0.0.0"
```

or

```
docker run -p 8050:8050 --volume `pwd`:/data quay.io/biocontainers/rpbp:3.0.1--
↪py310h30d9df9_0 rpbp-predictions-dashboard -c /data/c-elegans-test.yaml --host="0.0.0.0
↪"
```

```
singularity run --bind `pwd`:/data rpbp.sif rpbp-predictions-dashboard -c /data/c-
↪elegans-test.yaml --host="0.0.0.0"
```

You then have to open a browser page at the correct address, *e.g.* you see `Running on http://127.0.0.1:8050`, click on this link, or open a browser page at this address. To navigate the apps is easy, just follow the "hints". Most items are interactive. Press `CTRL+C` to quit.

Attention: For the apps only, the configuration file is passed using a (required) named argument `-c/--config CONFIG`.

The tutorials includes a small test Ribo-seq dataset and reference **Rp-Bp** output that is also used for regression tests. It has been constructed to include some reads which uniquely map to the annotated transcripts, some reads which map to ribosomal sequences, some reads which do not uniquely map to the genome, and some reads which are filtered due to quality issues.

1.6.4 Download the dataset

Click here and unzip the archive. Navigate to the *c-elegans-chrI-example* directory.

Content

- *c-elegans-test.yaml* The YAML configuration file.
- *c-elegans.alignments-only.yaml* Another YAML configuration to explain *How to use existing alignment files*.

input

- *c-elegans.test-chrI.rep-1.fastq.gz* Ribo-seq sample 1.
- *c-elegans.test-chrI.rep-2.fastq.gz* Ribo-seq sample 2.
- *WBcel235.chrI.fa* The reference sequence of Chromosome I for *C. elegans*.
- *WBcel235.79.chrI.gtf* The Ensembl, version 79 annotation, for Chromosome I for *C. elegans*.
- *X03680_1.fasta* The sequence of the ribosomal subunits for *C. elegans*. The reference accession is *X03680.1*.

- *riboseq-adapters.fa* An example adapter file. It includes typical TruSeq and ArtSeq adapters, as well as a few adapters from the literature. It also includes a custom adapter used to create the sample dataset.

reference

This directory contains reference genome annotations and predictions generated by **Rp-Bp**.

1.7 Frequently asked questions

1.7.1 I don't want/I can't install it on my computer. Can I still use Rp-Bp without installing the package?

Yes, you can use a Docker or a Singularity container. Simply pull, and you're done! See *Installation* for instructions. Example calls are also given in the user guide and tutorials.

1.7.2 How do I launch the app remotely?

By default, the application is opened in a browser page on *localhost:8050*. You don't have to actually worry about this. But you can also specify a `--host` and a `--port` when calling the app, enabling you to launch it from a remote server. In the latter case, however, you have to open a browser page at the correct address. For example, if you use `--host 123.123.123.123`, then open a page on *http://123.123.123.123:8050/*. To launch the app from a container, see *How to use the containers*.

1.7.3 Why is the ORF label not consistent with the host transcript?

In some cases, the ORF label may not be consistent with the host transcript, as reported by the ORF "id" (*transcript_seqname:start-end:strand*). To resolve such seemingly incoherent assignments, compatible transcripts are reported for each ORF in *<genome_name>.orfs-labels.annotated[.orf_note].tab.gz* and shown in the prediction dashboard, see *Visualization and quality control*.

1.7.4 I have my own alignments, can I use Rp-Bp?

The short answer is yes. The pipeline is designed to handle all steps from raw FASTQ files up to the final list of translated Ribo-seq ORFs, but you can start the pipeline from any step. Check the tutorial *How to use existing alignment files*.

1.7.5 I get errors when calling `summarize-rpbp-predictions`

The most common errors *e.g.* `AttributeError: 'float' object has no attribute 'left'` are due to the bin width for the Circos plot. Try reducing it using `--circos-bin-width VALUE` (default VALUE: 10000000). You may also have to use `--circos-show-chroms` if your organism has a different nomenclature. Use `summarize-rpbp-predictions --help` for details.

1.7.6 The IGV browser does not load or the predictions app fails to start

The most likely reason is that your reference genome sequence (given by the config key `fasta`) is not indexed, *i.e* the file ***.fasta.fai** is missing. You can create the missing index using samtools faidx.

1.7.7 I get `RuntimeWarning: invalid value encountered in divide res, _ = _lowess(y, x, x, np.ones_like(x),`

This happens for 3 nt ORFs.

1.8 API

1.8.1 prepare-rpbp-genome

Prepare genome indices and annotations to use with the Rp-Bp profile construction/periodicity estimation and ORF discovery pipeline.

```
usage: prepare-rpbp-genome [-h] [--overwrite]
                           [--star-executable STAR_EXECUTABLE]
                           [--star-options [STAR_OPTIONS ...]]
                           [--num-cpus NUM_CPUS] [--mem MEM] [--time TIME]
                           [--partitions PARTITIONS] [--no-output]
                           [--no-error] [--stdout-file STDOUT_FILE]
                           [--stderr-file STDERR_FILE] [--do-not-call]
                           [--use-slurm]
                           [--mail-type [{NONE,BEGIN,END,FAIL,REQUEUE,ALL,STAGE_OUT,TIME_
↪LIMIT,TIME_LIMIT_90,TIME_LIMIT_80,TIME_LIMIT_50,ARRAY_TASKS} ...]]
                           [--mail-user MAIL_USER] [--log-file LOG_FILE]
                           [--enable-ext-logging] [--log-stdout]
                           [--no-log-stderr]
                           [--logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                           [--file-logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}]
                           [--stdout-logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}]
```

(continues on next page)

(continued from previous page)

```
                          [--stderr-logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}]
                          config
```

Positional Arguments

config A YAML configuration file.

Named Arguments

--overwrite Overwrite existing output.

Default: False

STAR options

--star-executable The name of the STAR executable

Default: "STAR"

--star-options A space-delimited list of options to pass to STAR. Each option is quoted separately as in `"--starOption value"`, using soft quotes, where starOption is the long parameter name from STAR, and value is the value given to this parameter. If specified, STAR options will override default settings.

slurm options

--num-cpus The number of CPUs to use (not only for SLURM). For STAR, `--num-cpus` are threads, but in general, this is number of processes to spawn. This value should not be greater than the number of cores available. When used with SLURM, this is equivalent to: `--ntasks 1 --cpus-per-task <num-cpus>`.

Default: 1

--mem Real memory required (per node), mostly for STAR genome indexing (not only for SLURM). When used with SLURM, this is equivalent to: `--mem=<mem>`.

Default: "2G"

--time Set a limit on the total run time of the job allocation. This is equivalent to: `--time <time>`.

--partitions Request a partition for the resource allocation. This is equivalent to: `-p <partitions>`.

--no-output Redirect stdout to /dev/null. This is equivalent to: `--output=/dev/null`. By default, stdout is redirected to `--output=slurm-*.out`.

Default: False

--no-error Redirect stderr to /dev/null. This is equivalent to: `--output=/dev/null`. By default, stderr is redirected to `--output=slurm-*.err`.

Default: False

--stdout-file Log file (stdout) if not `--no-output`. This is equivalent to: `--output=stdout-file`.

--stderr-file Log file (stderr) if not `--no-error`. This is equivalent to: `--output=stderr-file`.

--do-not-call Do not execute the program (dry run).

Default: False

--use-slurm Submitted calls to SLURM.

Default: False

--mail-type Possible choices: NONE, BEGIN, END, FAIL, REQUEUE, ALL, STAGE_OUT, TIME_LIMIT, TIME_LIMIT_90, TIME_LIMIT_80, TIME_LIMIT_50, ARRAY_TASKS

Notify user by email when certain event types occur, if `--mail-user` is specified.

Default: ['FAIL', 'TIME_LIMIT']

--mail-user User to receive email notification of state changes as defined by `--mail-type`.

logging options

--log-file Log file (logging is redirected to this file, in addition to stdout and stderr, if specified).

Default: ""

--enable-ext-logging Enable logging for external programs that may be disabled by default, *e.g.* CmdStanPy.

Default: False

--log-stdout Log to stdout (in addition to a file and stderr, if specified).

Default: False

--no-log-stderr Do not send logging to stderr.

Default: False

--logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for all logs.

Default: "WARNING"

--file-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for the log file. This option overrides `--logging-level`.

Default: "NOTSET"

--stdout-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for stdout. This option overrides `--logging-level`.

Default: "NOTSET"

--stderr-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for stderr. This option overrides `--logging-level`.

Default: "NOTSET"

1.8.2 run-all-rpbp-instances

Call Rp-Bp for each sample in the configuration file.

```
usage: run-all-rpbp-instances [-h] [--profiles-only] [--merge-replicates]
                              [--run-replicates] [--write-unfiltered] [-k]
                              [--overwrite] [--tmp TMP]
                              [--flexbar-options [FLEXBAR_OPTIONS ...]]
                              [--star-executable STAR_EXECUTABLE]
                              [--star-options [STAR_OPTIONS ...]]
                              [--num-cpus NUM_CPUS] [--mem MEM] [--time TIME]
                              [--partitions PARTITIONS] [--no-output]
                              [--no-error] [--stdout-file STDOUT_FILE]
                              [--stderr-file STDERR_FILE] [--do-not-call]
                              [--use-slurm]
                              [--mail-type [{NONE,BEGIN,END,FAIL,REQUEUE,ALL,STAGE_OUT,
↪TIME_LIMIT,TIME_LIMIT_90,TIME_LIMIT_80,TIME_LIMIT_50,ARRAY_TASKS} ...]]
                              [--mail-user MAIL_USER] [--log-file LOG_FILE]
                              [--enable-ext-logging] [--log-stdout]
                              [--no-log-stderr]
                              [--logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,CRITICAL}
↪]
                              [--file-logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}]
                              [--stdout-logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}]
                              [--stderr-logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}]
                              config
```

Positional Arguments

config A YAML configuration file.

Named Arguments

--profiles-only Run the periodicity estimation only (ORF profile construction) for each sample in the configuration file.

Default: False

--merge-replicates Predict Ribo-seq ORFs in merged profiles. If `--merge-replicates`, then use `--run-replicates` to also predict Ribo-seq ORFs in all samples.

Default: False

--run-replicates With `--merge-replicates`, predict Ribo-seq ORFs in all samples and in merged profiles. This has no effect without `--merge-replicates`, *i.e.* predictions are made for all samples by default.

Default: False

--write-unfiltered In addition to the default filtered predictions, output all overlapping predicted Ribo-seq ORFs.

Default: False

-k, --keep-intermediate-files Keep all intermediate files. Intermediate files are necessary to perform read filtering quality control.

Default: False

--overwrite Overwrite existing output.

Default: False

--tmp A temporary directory (STAR).

Flexbar options

--flexbar-options A space-delimited list of options to pass to Flexbar. Each option is quoted separately as in `"--flexbarOption value"`, using soft quotes, where flexbarOption is the long parameter name from Flexbar, and value is the value given to this parameter. If specified, Flexbar options will override default settings.

STAR options

--star-executable The name of the STAR executable

Default: "STAR"

--star-options A space-delimited list of options to pass to STAR. Each option is quoted separately as in `"--starOption value"`, using soft quotes, where starOption is the long parameter name from STAR, and value is the value given to this parameter. If specified, STAR options will override default settings.

slurm options

--num-cpus The number of CPUs to use (not only for SLURM). For STAR, `--num-cpus` are threads, but in general, this is number of processes to spawn. This value should not be greater than the number of cores available. When used with SLURM, this is equivalent to: `--ntasks 1 --cpus-per-task <num-cpus>`.

Default: 1

--mem Real memory required (per node), mostly for STAR genome indexing (not only for SLURM). When used with SLURM, this is equivalent to: `--mem=<mem>`.

Default: "2G"

--time Set a limit on the total run time of the job allocation. This is equivalent to: `--time <time>`.

--partitions Request a partition for the resource allocation. This is equivalent to: `-p <partitions>`.

--no-output Redirect stdout to /dev/null. This is equivalent to: `--output=/dev/null`. By default, stdout is redirected to `--output=slurm-*.out`.

Default: False

--no-error Redirect stderr to /dev/null. This is equivalent to: `--output=/dev/null`. By default, stderr is redirected to `--output=slurm-*.err.`

Default: False

--stdout-file Log file (stdout) if not `--no-output`. This is equivalent to: `--output=stdout-file`.

--stderr-file Log file (stderr) if not `--no-error`. This is equivalent to: `--output=stderr-file`.

--do-not-call Do not execute the program (dry run).

Default: False

--use-slurm Submitted calls to SLURM.

Default: False

--mail-type Possible choices: NONE, BEGIN, END, FAIL, REQUEUE, ALL, STAGE_OUT, TIME_LIMIT, TIME_LIMIT_90, TIME_LIMIT_80, TIME_LIMIT_50, ARRAY_TASKS

Notify user by email when certain event types occur, if `--mail-user` is specified.

Default: ['FAIL', 'TIME_LIMIT']

--mail-user User to receive email notification of state changes as defined by `--mail-type`.

logging options

--log-file Log file (logging is redirected to this file, in addition to stdout and stderr, if specified).

Default: ""

--enable-ext-logging Enable logging for external programs that may be disabled by default, *e.g.* CmdStanPy.

Default: False

--log-stdout Log to stdout (in addition to a file and stderr, if specified).

Default: False

--no-log-stderr Do not send logging to stderr.

Default: False

--logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for all logs.

Default: "WARNING"

--file-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for the log file. This option overrides `--logging-level`.

Default: "NOTSET"

--stdout-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for stdout. This option overrides `--logging-level`.

Default: "NOTSET"

--stderr-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for stderr. This option overrides `--logging-level`.

Default: "NOTSET"

1.8.3 summarize-rpbp-profile-construction

Summarize the ORF profile creation step and prepare data for the web application.

```
usage: summarize-rpbp-profile-construction [-h] [-c] [--overwrite] [--tmp TMP]
                                           [-p NUM_CPUS] [--log-file LOG_FILE]
                                           [--enable-ext-logging]
                                           [--log-stdout] [--no-log-stderr]
                                           [--logging-level {NOTSET,DEBUG,INFO,WARNING,
↪ERROR,CRITICAL}]
                                           [--file-logging-level {NOTSET,DEBUG,INFO,
↪WARNING,ERROR,CRITICAL}]
                                           [--stdout-logging-level {NOTSET,DEBUG,INFO,
↪WARNING,ERROR,CRITICAL}]
                                           [--stderr-logging-level {NOTSET,DEBUG,INFO,
↪WARNING,ERROR,CRITICAL}]
                                           config
```

Positional Arguments

config A YAML configuration file. The same used to run the pipeline.

Named Arguments

-c, --create-fastqc-reports Create FastQC reports.

Default: False

--overwrite Overwrite existing output

Default: False

--tmp A temporary directory (FastQC).

-p, --num-cpus The number of CPUs to use.

Default: 1

logging options

--log-file Log file (logging is redirected to this file, in addition to stdout and stderr, if specified).

Default: “”

--enable-ext-logging Enable logging for external programs that may be disabled by default, *e.g.* CmdStanPy.

Default: False

--log-stdout Log to stdout (in addition to a file and stderr, if specified).

Default: False

--no-log-stderr Do not send logging to stderr.

Default: False

--logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for all logs.

Default: "WARNING"

--file-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for the log file. This option overrides `--logging-level`.

Default: "NOTSET"

--stdout-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for stdout. This option overrides `--logging-level`.

Default: "NOTSET"

--stderr-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for stderr. This option overrides `--logging-level`.

Default: "NOTSET"

1.8.4 summarize-rpbp-predictions

Summarizes the ORF prediction step and prepare data for the web application.

```
usage: summarize-rpbp-predictions [-h] [--min-samples MIN_SAMPLES] [-k]
                                  [--no-replicates] [--use-unfiltered]
                                  [--use-name-maps]
                                  [--match-standardized-orfs]
                                  [--circos-bin-width CIRCOS_BIN_WIDTH]
                                  [--circos-show-chroms CIRCOS_SHOW_CHROMS [CIRCOS_SHOW_
↪CHROMS ...]]
                                  [--show-orf-periodicity]
                                  [--image-type IMAGE_TYPE] [--overwrite]
                                  [--log-file LOG_FILE] [--enable-ext-logging]
                                  [--log-stdout] [--no-log-stderr]
                                  [--logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}]
                                  [--file-logging-level {NOTSET,DEBUG,INFO,WARNING,ERROR,
↪CRITICAL}]
                                  [--stdout-logging-level {NOTSET,DEBUG,INFO,WARNING,
↪ERROR,CRITICAL}]
                                  [--stderr-logging-level {NOTSET,DEBUG,INFO,WARNING,
↪ERROR,CRITICAL}]
                                  config
```

Positional Arguments

config A YAML configuration file. The same used to run the pipeline.

Named Arguments

--min-samples An ORF is filtered out if not predicted in at least `--min-samples` number of samples. By default all ORFs are kept. This is ignored if merged replicates are included in the output.

Default: 1

-k, --keep-other Include ORFs labeled as *other*, if present. They are discarded by default.

Default: False

--no-replicates If Rp-Bp was run with `--merge-replicates`, predictions from merged replicates are included by default, unless this flag is present.

Default: False

--use-unfiltered Use the *unfiltered* ORF predictions. Unless Rp-Bp was run with `--write-unfiltered`, these will not be available. By default, the *filtered* predictions are used.

Default: False

--use-name-maps Use `riboseq_sample_name_map` and `riboseq_condition_name_map` from the config. Do not use this flag when preparing results for the dashboard, mapping is done in the app.

Default: False

--match-standardized-orfs Add matching ORF names from https://doi.org/10.1038/s41587-022-01369-0 (Human data)

Default: False

--circos-bin-width Bin width for counting ORF predictions along chromosomes. Same bin width for all chromosomes. Last bin adjusted ad hoc to chromosome size.

Default: 10000000

--circos-show-chroms A list of chromosomes for which predictions are shown. By default, only numbered chromosomes (and X/x, Y/y) are shown, to avoid cluttering the figure. Use this option for organisms with a different nomenclature, or to show additional chromosomes or contigs.

Default: ['\d', 'X', 'x', 'Y', 'y']

--show-orf-periodicity Create bar charts showing the periodicity of each ORF type (not for the app).

Default: False

--image-type Format for `--show-orf-periodicity`.

Default: "eps"

--overwrite Overwrite existing output.

Default: False

logging options

--log-file Log file (logging is redirected to this file, in addition to stdout and stderr, if specified).

Default: “”

--enable-ext-logging Enable logging for external programs that may be disabled by default, *e.g.* CmdStanPy.

Default: False

--log-stdout Log to stdout (in addition to a file and stderr, if specified).

Default: False

--no-log-stderr Do not send logging to stderr.

Default: False

--logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for all logs.

Default: “WARNING”

--file-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for the log file. This option overrides `--logging-level`.

Default: “NOTSET”

--stdout-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for stdout. This option overrides `--logging-level`.

Default: “NOTSET”

--stderr-logging-level Possible choices: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL

Logging level for stderr. This option overrides `--logging-level`.

Default: “NOTSET”

1.8.5 rpbp-profile-construction-dashboard

Launch a Dash app for quality control and visualization of ribosome profiling data processed with Rp-Bp.

```
usage: rpbp-profile-construction-dashboard [-h] [-c CONFIG] [-d] [--host] [--port]
```

Named Arguments

-c, --config A YAML configuration file (required). The same used to run the pipeline.

-d, --debug Enable debug mode.

Default: False.

--host Host.

Default: localhost

--port Port Number.

Default: 8050

1.8.6 rpbp-predictions-dashboard

Launch a Dash app to visualize Ribo-seq ORF predicted with Rp-Bp.

```
usage: rpbp-predictions-dashboard [-h] [-c CONFIG] [-d] [--host] [--port]
```

Named Arguments

-c, --config	A YAML configuration file (required). The same used to run the pipeline.
-d, --debug	Enable debug mode.
	Default: False.
--host	Host.
	Default: localhost
--port	Port Number.
	Default: 8050