ribopipe Documentation

Release 0.1.6-beta

Jordan A. Berg

May 15, 2019

Contents

RiboPipe is toolkit designed for automating the assembly, quality control, and analysis of ribosome profiling and other single-end RNAseq datasets. It is intended to be easy to use for bench and computational biologists and is quick to use out of the box.

RiboPipe is developed by Jordan Berg in the Rutter Lab @ the University of Utah, along with other collaborators.

CHAPTER 1

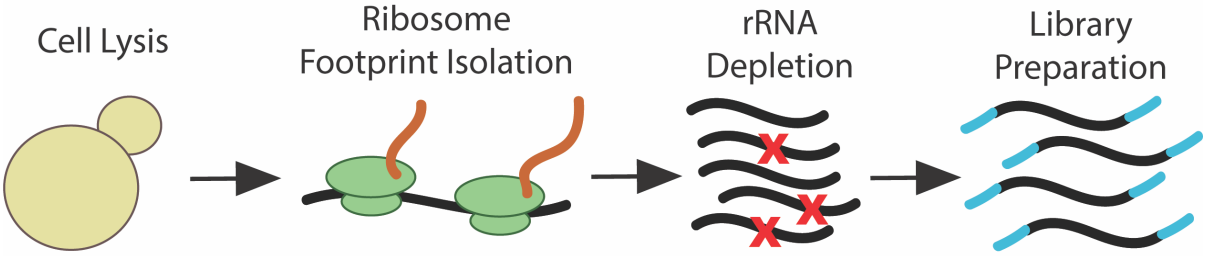
Table of contents

1.1 Overview

1.1.1 Background

Ribosome profiling utilizes Next Generation Sequencing to provide a detailed picture of the protein translation landscape within cells. Cells are lysed, translating ribosomes are isolated, and the ribosome protected mRNA fragments are integrated into a sequencing library. The library is then sequenced and raw data (often in the form of .fastq or .txt files) is generated. This pipeline is flexibly designed to be able to process and perform preliminary analyses on SE (single-end) short (<= 100 bp) read raw sequence data.

See this paper for a recent discussion and detailed protocol of the technique.



1.2 Installation

1.2.1 MacOS Installation

See local_install.sh in the resources folder for interactive script.

1) RiboPipe requires use of command line. Execute the following lines of code in Terminal (on Mac, open Spotlight and type 'Terminal'):

2) Install python3, wget, git, and git-lfs if not already done. You can check in command line by typing in the name of the package and checking if your system recognizes the package name.

```
$ sh -c "$(curl -fsSL https://raw.githubusercontent.com/Linuxbrew/install/master/
↪install.sh)"
$ echo "export PATH='$(brew --prefix)/bin:$(brew --prefix)/sbin'":'"$PATH"' >> ~/.
↪bash_profile
$ brew install python3 wget git git-lfs
$ git lfs install
```

3) You may need to manually point your system to python3. You can check this by typing `python` in the command line and seeing if it is running python3. At this time, RiboPipe only works when Python3 is set as the default. If it is not, do the following:

```
$ echo "alias python="python3" >> ~/.bash_profile
```

4) Download Conda, a package manager, for your operating system. Double click the *.pkg* file if on MacOS, the *.exe* file on Windows, or follow these instructions on Linux.

5) Download RiboPipe dependencies via conda:

```
$ conda install -y -c bioconda setuptools fastqc fastx_toolkit htseq picard samtools␣
↪hisat2 star bedtools deeptools scipy plastid pandas numpy matplotlib seaborn␣
↪pysam=0.14
```

6) Install RiboPipe dependencies via pip (should have come pre-installed with python3.4 or greater):

```
$ pip install multiqc
```

7) Download RiboPipe:

```
$ git clone https://github.com/RiboPipe/ribopipe.git
$ cd ribopipe
$ cd python setup.py install
```

8) Or, to download specific version:

```
$ tag='v0.1.4-beta'
$ wget https://github.com/RiboPipe/ribopipe/archive/$tag.zip
$ unzip ribopipe-${tag:1}.zip
$ mv ribopipe-${tag:1} ribopipe
$ cd ribopipe
$ cd python setup.py install
```

9) At the end of the installation instructions, an installation location will be given. Add this to your $PATH:

```
...
Installing ribopipe script to /Users/$USERNAME/anaconda3/bin

Installed /Users/$USERNAME/anaconda3/lib/python3.6/site-packages/RiboPipe-0.1.5b0-py3.
↪6.egg
Processing dependencies for RiboPipe==0.1.5b0
Finished processing dependencies for RiboPipe==0.1.5b0

$ echo "export PATH='/Users/$USERNAME/anaconda3/bin:$PATH' >> ~/.bash_profile
```

10) Test installation:

```
$ ribopipe --help
```

1.2.2 HPC Installation

See hpc_install.sh in the resources folder for interactive script. While the resource manager can install these for you, we will show you how to manually install all dependencies. These instructions may vary slightly from HPC to HPC.

1) Remove all pre-loaded software:

```
$ module purge
```

2) Install brew and related dependencies: Install python3, wget, git, and git-lfs if not already done. You can check in command line by typing in the name of the package and checking if your system recognizes the package name.

```
$ sh -c "$(curl -fsSL https://raw.githubusercontent.com/Linuxbrew/install/master/
↪install.sh)"
$ echo "export PATH='$(brew --prefix)/bin:$(brew --prefix)/sbin'":'"$PATH"' >> ~/.
↪bash_profile
$ brew install python3 wget git git-lfs
$ git lfs install
```

3) You may need to manually point your system to python3. You can check this by typing `python` in the command line and seeing if it is running python3. At this time, RiboPipe only works when Python3 is set as the default. If it is not, do the following:

```
$ echo "alias python="python3" >> ~/.bash_profile
```

4) Install anaconda (instructions retrieves most recent version as of time of writing):

```
$ wget https://repo.anaconda.com/archive/Anaconda3-5.3.0-Linux-x86_64.sh
$ chmod 700 Anaconda3-5.3.0-Linux-x86_64.sh
$ ./Anaconda3-5.3.0-Linux-x86_64.sh -b -p $HOME/.local/bin -s
$ export PATH="/uufs/chpc.utah.edu/common/home/$USER/.local/bin:$PATH"
$ conda install -y -c bioconda setuptools fastqc fastx_toolkit htseq picard samtools␣
↪star bedtools deeptools scipy plastid pandas numpy matplotlib seaborn pysam=0.14
```

5) Install pip related dependencies (should have come pre-installed with python3.4 or greater):

```
$ pip install multiqc
```

6) To download current repository:

```
$ git clone https://github.com/RiboPipe/ribopipe.git
$ cd ribopipe
$ python setup.py install --prefix ~/.local
```

7) Or, to download specific version

```
$ tag='v0.1.4-beta'
$ wget https://github.com/RiboPipe/ribopipe/archive/$tag.zip
$ unzip ribopipe-${tag:1}.zip
$ mv ribopipe-${tag:1} ribopipe
$ cd ribopipe
$ cd python setup.py install --prefix ~/.local
```

8) At the end of the installation instructions, an installation location will be given. Add this to your $PATH:

```
...
Installing ribopipe script to /uufs/chpc.utah.edu/common/home/$USER/.local/bin

Installed /uufs/chpc.utah.edu/common/home/$USER/.local/lib/python3.5/site-packages/
↪RiboPipe-0.1.5b0-py3.5.egg
Processing dependencies for RiboPipe==0.1.5b0
Finished processing dependencies for RiboPipe==0.1.5b0

$ echo "export PATH='/uufs/chpc.utah.edu/common/home/$USER/.local/bin:$PATH' >> ~/.
↪bash_profile
```

9) Test installation:

```
$ ribopipe --help
```

1.2.3 Test Installation

To test installation, run the following command:

```
$ ribopipe riboseq -i /path/to/ribopipe/test/ -o /path/you/create/ -r yeast -e␣
↪ingolia_2015 \
-p STAR -a CTGTAGGCACCATCAAT --platform ILLUMINA --count_cutoff 32 \
-s a_wild-type_DED1_replicate_1_15_deg c_ded1-cs_replicate_1_15_deg
```

If no errors are produced by the output, the installation was successful.

1.2.4 Additional Help

Manually installing a package:

Sometimes a package may need to be manually installed. In these cases, a pattern as follows may be used (example given is loading on HPC).

```
$ wget https://github.com/simon-anders/htseq/archive/release_0.11.0.zip
$ unzip htseq-release_0.11.0.zip
$ rm htseq-release_0.11.0.zip
$ cd htseq-release_0.11.0
$ python setup.py install --prefix ~/.local
$ cd ../
$ echo "export PATH='/uufs/chpc.utah.edu/common/home/$USER/.local/bin:$PATH' >> ~/.
↪bash_profile"
```

Or...

```
$ git clone https://github.com/simon-anders/htseq.git
$ cd htseq
$ python setup.py install --prefix ~/.local
$ cd ../
$ echo "export PATH='/uufs/chpc.utah.edu/common/home/$USER/.local/bin:$PATH' >> ~/.
↪bash_profile"
```

Getting publicly available raw data from GEO:

Raw data from previous studies that have been made publicly available can be accessed through the GEO database

Please see this example script for examples of how to retrieve this data. This thread is also helpful.

1.3 Quickstart

To do a full install on a local machine or HPC, see the Installation page.

1.3.1 Singularity

For a faster way to get started, consider using a Singularity image for RiboPipe, hosted here.

1) Singularity offers a fast, reproducible way of running RiboPipe where all dependencies and OS are bundled into a single disk image. These singularity "containers" are widely used in cloud computing. Download if you do not already have it (must be on a Linux OS). Often, cloud computing environments come with this software pre-installed.

2) Download a RiboPipe singularity container:

```
$ singularity pull library://sylabsed/linux/ribopipe
```

3) Run Ribopipe:

```
$ raw_data=/path/to/raw/data
$ output_data=/path/to/output/data
$ singularity exec ribopipe.sif riboseq -i $raw_data -o $output_data ...
```

1.3.2 Local

1) Move raw data to directory of choice

2) Create empty output directory

3) Run ribopipe:

```
$ raw_data=/path/to/raw/data
$ output_data=/path/to/output/data
$ ribopipe riboseq -i $raw_data -o $output_data ...
```

4) Collect raw_counts.csv output in $output_data/assembly/counts and edit sample_info.csv

5) Run diffex:

```
$ ribopipe_path=/path/to/ribopipe
$ ribopipe diffex -i $output_data/assembly/counts/raw_counts.csv -d $ribopipe_path/
↪resources/sample_info.csv -o output_name --type riboseq
```

1.3.3 HPC

1) Modify hpc_run_template.sh in the resources folder for an example script for submitting the pipeline job to the HPC and make sure dependencies listed in this script are on the HPC system, else they need to be locally installed 2) Run the script by executing the following:

```
$ sbatch hpc_run_template.sh
```

If you want the slurm output file to be sent to the SLURM directory to avoid storage space issues on your interactive node, then in the *#SBATCH -o slurmjob-%j* line, replace it with the path to your SLURM directory:

```
$ #SBATCH -o /scratch/general/lustre/INPUT_USER_ID_HERE/slurmjob-%j
```

1.4 General Usage

The purpose of RiboPipe is to automate the alignment, quality control, and initial analysis of ribosome profiling and other short, single-end read data. It is intended that input data is a directory of .fastq formatted files. However, when using the intermediate submodules, such as `align` or `quality`, input will vary and is explicated in the `--help` menu for each submodule.

RiboPipe was created with the novice bioinformatician in mind, and the documentation written assuming no background in using Command Line or programming. As such, instructions have been created with as much detail as possible. Additionally, **'walkthrough videos <>'_** are available. In cases where RiboPipe is being used on a cloud computing cluster, or Linux machine with Singularity installed, a singularity container image can be used to avoid the installation process. Additionally, alignment references have been created for a variety of model organisms using current builds to help users avoid this step. Details on how to incorporate a newly created reference not already provided within the RiboPipe infrastructure can be found here.

RiboPipe can be run essentially from beginning to end as a pipeline, or as individual submodules. We will describe each option in detail below.

1.4.1 Important File Naming Conventions

In order for many of the RiboPipe functions to perform properly and for the output to be reliable after alignment (except for generation of a raw counts table), file naming conventions must be followed.

1) Download your raw sequence data and place in a folder – this folder should contain all the sequence data and nothing else.

2) Make sure files follow a pattern naming scheme. For example, if you had 3 genetic backgrounds of ribosome profiling data, the naming scheme would go as follows:

```
ExperimentName_BackgroundA_FP.fastq(.qz)
ExperimentName_BackgroundA_RNA.fastq(.qz)
ExperimentName_BackgroundB_FP.fastq(.qz)
ExperimentName_BackgroundB_RNA.fastq(.qz)
ExperimentName_BackgroundC_FP.fastq(.qz)
ExperimentName_BackgroundC_RNA.fastq(.qz)
```

3) If the sample names are replicates, their sample number needs to be indicated.

4) If you want the final count table to be in a particular order and the samples ordered that way are not alphabetically, append a letter in front of the sample name to force this ordering.

```
ExperimentName_a_WT_FP.fastq(.qz)
ExperimentName_a_WT_RNA.fastq(.qz)
ExperimentName_b_exType_FP.fastq(.qz)
ExperimentName_b_exType_RNA.fastq(.qz)
```

5) If you have replicates:

```
ExperimentName_a_WT_1_FP.fastq(.qz)
ExperimentName_a_WT_1_RNA.fastq(.qz)
ExperimentName_a_WT_2_FP.fastq(.qz)
ExperimentName_a_WT_2_RNA.fastq(.qz)
ExperimentName_b_exType_1_FP.fastq(.qz)
ExperimentName_b_exType_1_RNA.fastq(.qz)
ExperimentName_b_exType_2_FP.fastq(.qz)
ExperimentName_b_exType_2_RNA.fastq(.qz)
```

6) If you are just running RNAseq files through the pipeline, you only need the RNA samples in your input directory and specify the rnaseq module:

```
ExperimentName_a_WT_1_RNA.fastq(.qz)
ExperimentName_a_WT_2_RNA.fastq(.qz)
ExperimentName_b_exType_1_RNA.fastq(.qz)
ExperimentName_b_exType_2_RNA.fastq(.qz)

ribopipe rnaseq -i input_directory ...
```

1.4.2 `riboseq` module

Pipeline for handling raw ribosome profiling sequence data. Runs quality and adaptor trimming, alignment, quality control, and formatting on a directory of raw ribosome profiling sequence data.

The `riboseq` module can be run as follows:

```
$ ribopipe riboseq -i $DATA_PATH/raw_ingolia/ -o $DATA_PATH/out_ingolia/ -r yeast -e␣
↪ingolia_2015 \
                   -s a_WT_DED1_1_15deg b_WT_DED1_2_15deg c_ded1_cs_1_15deg d_ded1_cs_
↪2_15deg \
                   e_WT_DED1_1_37deg f_WT_DED1_2_37deg g_ded1_ts_1_37deg h_ded_ts_2_
↪37deg \
                   i_WT_TIF1_1_30deg j_WT_TIF1_2_30deg k_tif1_ts_1_30deg l_tif1_ts_2_
↪30deg \
                   m_WT_TIF1_1_37deg n_WT_TIF1_2_37deg o_tif1_ts_1_37deg p_tif1_ts_2_
↪37deg \
                   -p HISAT2 -a CTGTAGGCACCATCAAT --platform ILLUMINA --count_cutoff␣
↪32
```

An explanation of the `riboseq` submodule arguments can be found in the following tables:

Table 1: `riboseq` required arguments

Argument	Description
`-i INPUT, --input INPUT`	Specify full PATH to input directory
`-o OUTPUT, --output OUTPUT`	Specify full PATH to output directory
`-r <yeast>, <human>, <mouse>, --reference <yeast>, <human>, <mouse>`	Specify model organism used for experiments. Pipeline will align a current, curated reference
`-e <string>, --experiment <string>`	Provide experiment name to prepend to output files
`-s <string> [<string> ...], --samples <string> [<string> ...]`	Space delimited list of samples in order. If replicates are included, indicate number in sample name to delineate.
`-a <string> [<string> ...], --adaptor <string> [<string> ...]`	Sequence of 3' linker (only supports one 3' linker currently) (default: `AACTGTAGGCACCATCAAT`). If no adaptor was used, specify `None`'

Table 2: `riboseq` optional arguments

Argument	Description
`-m <int>, --max_processors <int>`	Number of max processors pipeline can use for multiprocessing tasks (default: No limit)
`-f, --footprints_only`	Select this option if ONLY providing raw footprint sequence data
`--min_overlap <integer>`	Minimum number of bases that must match on a side to combine sequences for `rrna_prober`
`-p <HISAT2>, <STAR>, --program <HISAT2>, <STAR>`	Alignment software to be used to align reads to reference (default: `STAR`)
`--read_length_min <int>`	Minimum read length threshold to keep for reads (default: `11`)
`--read_length_max <int>`	Maximum read length threshold to keep for reads (default: `50`)
`--read_quality <int>`	PHRED read quality threshold (default: `28`)
`--platform <SANGER>, <ILLUMINA>`	Sequencing platform used (default: `ILLUMINA`)
`--full_genome`	Add this option to map reads to full genome. If not given, will not count any read mapping to the first 45 nt of transcripts for ribosome profiling
`--count_cutoff <int>`	Minimum counts threshold. Will remove any row in the final count tables if any sample does not meet this cutoff threshold

Additional information:

`-s`, `--samples`: For ribosome profiling, do not differentiate between footprint and RNA samples.

`--full_genome`: It is recommended to NOT include this option as ribosome profiling data for this region is often unreliable.

Follow up with the `diffex` submodule for differential expression analysis.

1.4.3 `rnaseq` module

Similar to the `riboseq` module, but tuned for small RNAseq (i.e. anything single-end under 100 bps). It is expected that in the future, RiboPipe will be able to perform automated paired-end and genome sequencing assembly and alignment.

The `rnaseq` module can be run as follows:

```
$ ribopipe rnaseq -i $DATA_PATH/raw_dang_berger/ -o $DATA_PATH/out_dang_berger/ -r␣
↪yeast -e dang_berger_2014 \
                  -s WT_NR_A WT_NR_B WT_CR_A WT_CR_B \
                  -p HISAT2 -a TGGAATTCTCGGGTGCCAAGG --platform ILLUMINA --replicates
```

An explanation of the `riboseq` submodule commands can be found in the following tables:

Table 3: `rnaseq` required arguments

Argument	Description
`-i INPUT, --input INPUT`	Specify full PATH to input directory
`-o OUTPUT, --output OUTPUT`	Specify full PATH to output directory
`-r <yeast>, <human>, <mouse>, --reference <yeast>, <human>, <mouse>`	Specify model organism used for experiments. Pipeline will align a current, curated reference
`-e <string>, --experiment <string>`	Provide experiment name to prepend to output files
`-s <string> [<string> ...], --samples <string> [<string> ...]`	Space delimited list of samples in order. If replicates are included, indicate number in sample name to delineate.
`-a <string> [<string> ...], --adaptor <string> [<string> ...]`	Sequence of 3' linker (only supports one 3' linker currently) (default: `AACTGTAGGCACCATCAAT`). If no adaptor was used, specify `None`'

Table 4: `rnaseq` optional arguments

Argument	Description
`-m <int>, --max_processors <int>`	Number of max processors pipeline can use for multiprocessing tasks (default: No limit)
`--replicates`	Select this option if samples are replicates (do not use if 3+ replicates).
`-p <HISAT2>, <STAR>, --program <HISAT2>, <STAR>`	Alignment software to be used to align reads to reference (default: `STAR`)
`--read_length_min <int>`	Minimum read length threshold to keep for reads (default: `11`)
`--read_length_max <int>`	Maximum read length threshold to keep for reads (default: `50`)
`--read_quality <int>`	PHRED read quality threshold (default: `28`)
`--platform <SANGER>, <ILLUMINA>`	Sequencing platform used (default: `ILLUMINA`)
`--full_genome`	Add this option to map reads to full genome. If not given, will only map reads to transcripts.
`--count_cutoff <int>`	Minimum counts threshold. Will remove any row in the final count tables if any sample does not meet this cutoff threshold

Additional information:

`--replicates`: Make sure when passing the argument to list samples these are sorted so replicates are next to each other in this list

Follow up with the `diffex` submodule for differential expression analysis.

1.4.4 `trim` module

Perform only trimming and quality control steps, outputting general read information. Will output trimmed reads ready for alignment.

The `trim` module can be run as follows:

```
$ ribopipe trim -i $DATA_PATH/input_files/ -o $DATA_PATH/output_files/ \
                  -a TGGAATTCTCGGGTGCCAAGG --platform ILLUMINA
```

An explanation of the `trim` submodule commands can be found in the following tables:

Table 5: `trim` required arguments

Argument	Description
`-i INPUT, --input INPUT`	Specify full PATH to input directory
`-o OUTPUT, --output OUTPUT`	Specify full PATH to output directory
`-a <string> [<string> ...], --adaptor <string> [<string> ...]`	Sequence of 3' linker (only supports one 3' linker currently) (default: `AACTGTAGGCACCATCAAT`). If no adaptor was used, specify `None`'

Table 6: `trim` optional arguments

Argument	Description
`-m <int>, --max_processors <int>`	Number of max processors pipeline can use for multiprocessing tasks (default: No limit)
`--read_length_min <int>`	Minimum read length threshold to keep for reads (default: `11`)
`--read_length_max <int>`	Maximum read length threshold to keep for reads (default: `50`)
`--read_quality <int>`	PHRED read quality threshold (default: `28`)
`--platform <SANGER>, <ILLUMINA>`	Sequencing platform used (default: `ILLUMINA`)

1.4.5 `align` module

Perform only alignment of input files – assumes files have already been trimmed if required. Will output alignment files and genome browser compatible files.

The `align` module can be run as follows:

```
$ ribopipe align -i $DATA_PATH/input_files/ -o $DATA_PATH/output_files/ -r yeast -e␣
↪align_only_test \
                  -s sample1_FP sample1_RNA sample2_FP sample2_RNA
                  -a TGGAATTCTCGGGTGCCAAGG --platform ILLUMINA
```

An explanation of the `align` submodule commands can be found in the following tables:

Table 7: `align` required arguments

Argument	Description
`-i INPUT, --input INPUT`	Specify full PATH to input directory
`-o OUTPUT, --output OUTPUT`	Specify full PATH to output directory
`-r <yeast>, <human>, <mouse>, --reference <yeast>, <human>, <mouse>`	Specify model organism used for experiments. Pipeline will align a current, curated reference
`-e <string>, --experiment <string>`	Provide experiment name to prepend to output files
`-s <string> [<string> ...], --samples <string> [<string> ...]`	Space delimited list of samples in order. If replicates are included, indicate number in sample name to delineate.
`t <riboseq>, <rnaseq>, --type <riboseq>, <rnaseq>`	Sequencing type – `riboseq` for ribosome profiling or `rnaseq` for single-end short read sequence data.

Table 8: `align` optional arguments

Argument	Description
`-m <int>, --max_processors <int>`	Number of max processors pipeline can use for multiprocessing tasks (default: No limit)
`-p <HISAT2>, <STAR>, --program <HISAT2>, <STAR>`	Alignment software to be used to align reads to reference (default: `STAR`)
`--full_genome`	Add this argument if you wish to align to the full genome. See notes in `riboseq` and `rnaseq` sections above for more information.
`--count_cutoff <int>`	Minimum counts threshold. Will remove any row in the final count tables if any sample does not meet this cutoff threshold

1.4.6 `quality` module

Takes a table of raw counts generates paired scatter plots.

The `quality` module can be run as follows:

```
$ ribopipe quality -i $DATA_PATH/raw_counts.csv -o $DATA_PATH/output_location/ \
                   -t riboseq
```

An explanation of the `quality` submodule commands can be found in the following tables:

Table 9: `quality` required arguments

Argument	Description
`-i INPUT, --input INPUT`	Input table (must be .csv file) of raw counts
`-o OUTPUT, --output OUTPUT`	Specify full PATH to output directory
`-t <riboseq>, <rnaseq>, --type <riboseq>, <rnaseq>`	Sequencing type – `riboseq` for ribosome profiling or `rnaseq` for single-end short read sequence data.

Additional information:

Table must have samples set as columns and must be ordered as sample1_FP, sample1_RNA, sample2_FP, sample2_RNA, etc. or as sample1_rep1_RNA, sample1_rep2_RNA, sample2_rep1_RNA, sample2_rep2_RNA.

1.4.7 `rrna_prober` module

As the bulk of total cellular RNA consists mostly of ribosomal RNA that is often not of interest and crowds out measurement of other RNAs, an rRNA depletion step is commonly performed before creating a cDNA sequencing library. Several commercial kits are available that can effectively deplete samples of full length rRNAs. However, inherent in ribosome profiling protocols, samples are treated with an RNase, which degrades full length rRNAs, making them more difficult to deplete before creating the cDNA library. One way to better deplete these rRNA fragments in ribosome profiling samples is to create biotinylated RNA probes for overrepresented rRNA fragments to extract these fragments (see this paper for a recent discussion). `rrna_prober` automates the identification of overrepresented species and outputs a table with these sequences ordered by abundance for ease of probe generation. This table is output in stdout in this module, or as a `txt` file in the `highlights/` output folder when running the full `riboseq` submodule.

The `rrna_prober` module can be run as follows:

```
$ ribopipe rrna_prober -i $DATA_PATH/sample1.zip $DATA_PATH/sample2.zip ... \
                       -o $DATA_PATH/output_location/
```

An explanation of the `rrna_prober` submodule commands can be found in the following tables:

Table 10: `rrna_prober` required arguments

Argument	Description
`-i <string> [<string> ...], --input <string> [<string> ...]`	Space delimited list of zipped files (include full paths to these files)
`-o <string>, --output <string>`	Output file name to write output to

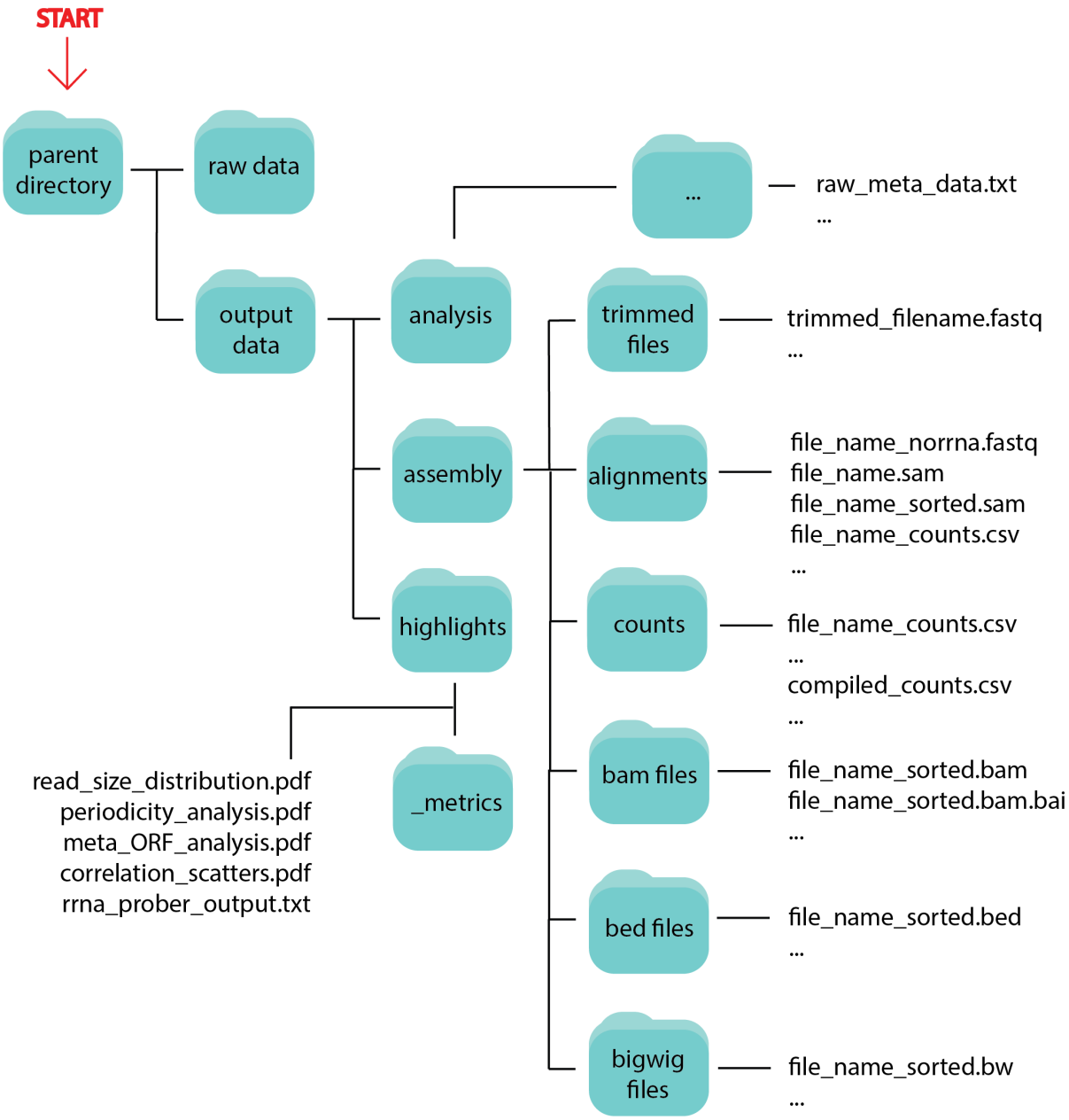
1.4.8 `gene_dictionary` module

1.4.9 `diffex` module

1.4.10 `truncate` module

1.4.11 Data Output

Running `riboseq` or `rnaseq` will output all intermediate and final data files in a tree structure as seen below:



1.5 Advanced Usage

1.5.1 Creating and Implementing Reference

CHAPTER 2

Tutorial

Video walkthrough coming soon! See the Quickstart guide for more information.

CHAPTER 3

Important notes

See the Important File Naming Conventions heading of the General Usage page for more information.

CHAPTER 4

Performance

Coming soon

CHAPTER 5

License

RiboPipe is freely available under a GNU General Public License (v3.0).

CHAPTER 6

Questions?

If you have questions, requests, or bugs to report, please use the Github issues forum.