TAQLoRe Documentation

Release 0.1

Tomasz Wrzesinski, Wilfried Haerty, Earlham Institute

Jun 18, 2020

Documentation index

TAQLoRE is a Snakemake-based pipeline to improve existing annotations and to quantify transcripts coming from long read amplicon-based cDNA sequencing technologies (Oxford Nanopore Technologies, PacBio). It was tested on Linux (CentOS 6) but it should work on Mac as well. Briefly, it uses LAST to align all reads to the transcriptome, then it discovers new exons by looking at insertions in alignments, it creates meta-gene with all known and novel exons, aligns all reads to it and generates a TMM-normalised read counts, together with expression heatmaps and PCA plots. It also identifies new splice sites by looking at perfectly aligned reads to the genome, and correcting all splice sites to the closest most abundant canonical ones. For more information, refer to *General concepts*.

CHAPTER 1

Installation

- Source code: GitHub
- Issue tracker: Issue tracker

CHAPTER 2

Citations of dependencies

Our pipeline is based on following software:

- Snakemake: Köster J, Rahmann S. "Snakemake - A scalable bioinformatics workflow engine". Bioinformatics. 2018 Oct 15;34(20):3600.
- LAST: Kiełbasa SM, Wan R, Sato K, Horton P, Frith MC. "Adaptive seeds tame genomic sequence comparison". Genome Res. 2011 Mar;21(3):487-93.
- GMAP: Wu TD, Watanabe CK. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. Bioinformatics. 2005 May 1;21(9):1859-75.
- Bedtools: Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010 Mar 15;26(6):841-2.
- Pybedtools: Dale RK, Pedersen BS, Quinlan AR. Pybedtools: a flexible Python library for manipulating genomic datasets and annotations. Bioinformatics. 2011 Dec 15;27(24):3423-4.

CHAPTER 3

Papers using the pipeline

The following papers/pre-prints that use our pipeline has been published:

- Clark M, Wrzesinski T, Garcia-Bea A, Kleinman J, Hyde T, Weinberger D, Haerty W, Tunbridge E. bioRxiv 260562.

CHAPTER 4

Authors

Developers:

- Wilfried Haerty (Earlham Institute)
- Tomasz Wrzesinski (Earlham Institute)

Contributors:

- Elizabeth Tunbridge (University of Oxford)
- Michael Clark (University of Melbourne)
- Nicola Hall (University of Oxford)
- Syed Hussain (University of Oxford)
- Hami Lee (University of Oxford)

CHAPTER 5

Things to add

- Splice-site-based pipeline (part4 and part5).
- Usage of splice-site-based approach (part4 and part5).
- Description of output files.
- Description of scripts.
- Description of example dataset.

5.1 General concepts

The pipeline comprise of five Snakemake files depending on the user demands. The whole pipeline is depicted on Figure 1.

The pipeline comprise of two different sub-workflows: **exon-based** (right-hand side of the figure) and **splice-site-based** (left-hand side of the figure). In the exon-based method, the pipeline annotates novel exons and novel transcript variants, using a known set of exons. It also quantifies reads aligned to obtained annotations after TMM normalisation (after filtering). In the splice-site-based methos, the pipeline identifies abundant canonical splice sites from read alignment to the genome. Particularly, it firstly annotates the perfectly aligned canonical splice sites (i.e. without removing any bases from reads at the splice breakpoint) and corrects every read to abundant canonical splice sites.

5.1.1 What TAQLoRe is

- It is a versatile tool to **IMPROVE** existing annotations (i.e. discovering novel exons, annotating exon skipping events)
- It is 'good enough' to quantify all transcripts, provided roughly equal number of reads in all samples
- It can annotate alternative splice sites although quantification step is less acurate than in the exon-level-based part of the pipeline

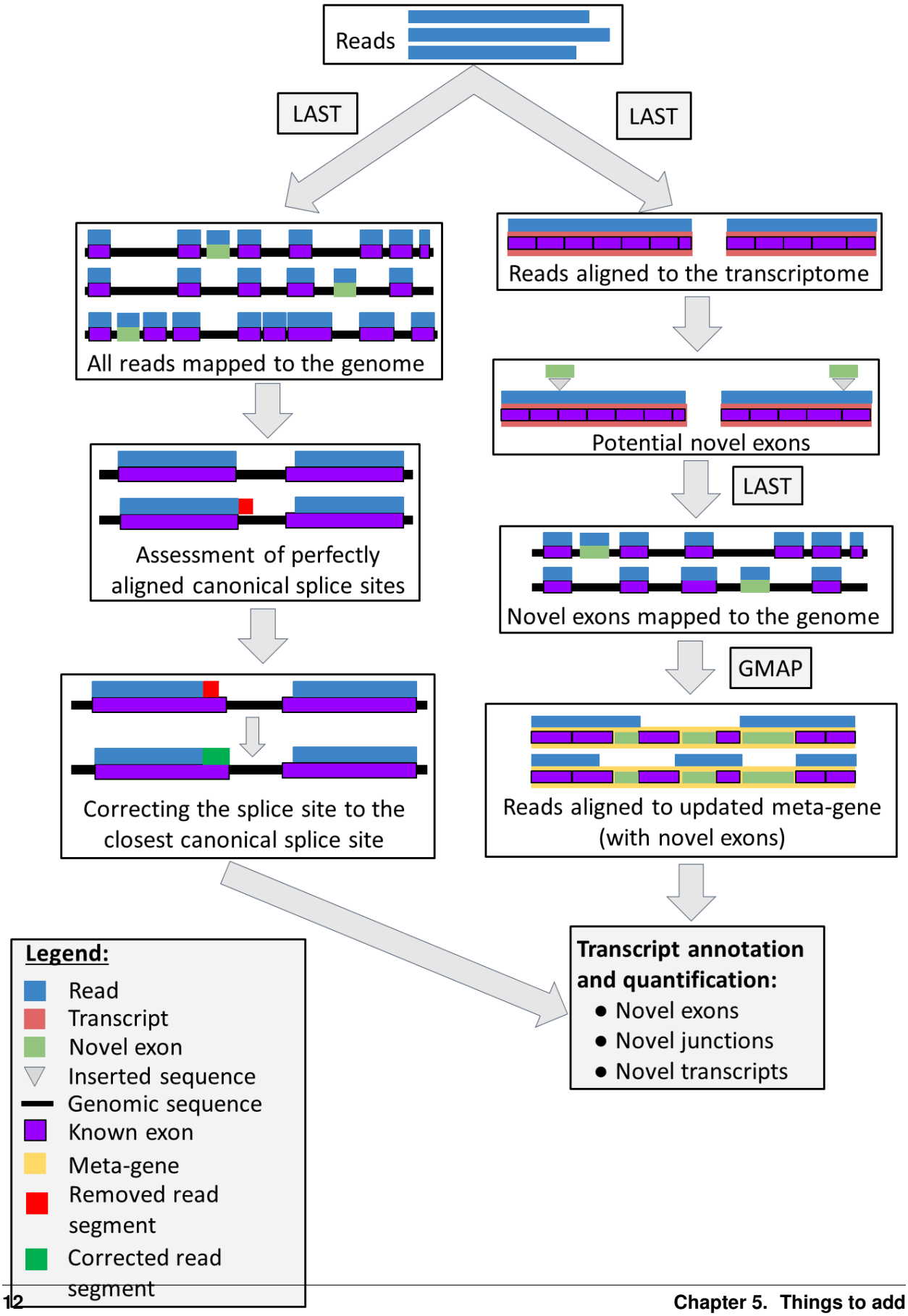

Fig. 1: Pipeline concept

5.1.2 What TAQLoRe is not

- It is not a tool to generate *de-novo* assembly of transcripts
- It is not a tool which works for the whole transcriptome (e.g. ONT direct RNA-Seq)
- It is not a tool to work without existing annotation
- It is not a tool to annotate non-canonical splice sites
- It is not a tool to annotate alternative UTRs and/or transcription start sites

5.1.3 Exon-based pipeline

Exon-based pipeline (right-hand side of the figure with *Pipeline concept*) can be used to obtain a high confidence annotation by annotating novel exons and using already known alternative splice sites to annotate and quantify all transcripts. It is split into three Snakefiles (part1-part3).

Part 1

The first part of the pipeline comprises of following steps:

1. Creating a transcriptome index.
2. Obtaining substitution rates from FASTA file with last-train.
3. Mapping reads to the reference transcriptome with LAST.
4. Obtaining novel exons from insertions in the reads (between exons).
5. Mapping novel exons to the genome, to obtain their exact genomic positions.
6. Filtering out alignments outside the gene of interest.
7. Generating a per-sample and merged BedGraph files with read counts aligned to novel exons.
8. Creating a file with meta-gene annotation.

After the Part 1 of the pipeline the user is expected to annotate UTRs in the file as UTRs should be ommited in the annotation. For more details see *Things to do after running part 1 of the pipeline*.

Part 2

The second part of the pipeline starts with meta-gene annotation file with annotated UTRs in the last column. Next steps are:

1. Creating a BED file with all the exons.
2. Generating FASTA file of the whole meta-gene.
3. Building GMAP index.
4. Aligning all the reads to the meta-gene.
5. Counting splicing patterns of all reads.
6. Removing exons covered by low number of reads.
7. Removing reads containing removed exons.
8. Downweighting reads that can be mapped to more than one transcript.
9. Creating a histogram of reads per sample.

10. Filtering out transcripts with low abundance.
11. TMM normalisation of read counts.
12. Creating expression heatmaps.
13. Creating PCA plots, transcripts explaining first three PCs, correlations between first three PCs and read numbers.
14. Creating annotation table with coding status.
15. Creating BED12 files with all transcripts.
16. FASTA files with coding and non-coding transcripts' nucleotide sequences, together with sequences of transcripts containing novel exons.
17. Annotation table and FASTA files with nucleotide sequences for transcripts containing exon skipping events.

Part 3

Third part of the pipeline can be ran if there is a huge bias for read numbers (in the magnitude of 1000s) between different samples (which can be observed in the correlation plot generated at step no. 13 of Part 2). It starts with the splicing patterns of each read per sample. The steps of this part of the pipeline are:

1. Downsampling the splicing patterns per read in all samples.
2. Removing exons covered by low number of reads.
3. Removing reads containing removed exons.
4. Downweighting reads that can be mapped to more than one transcript.
5. Creating a histogram of reads per sample.
6. Filtering out transcripts with low abundance.
7. TMM normalisation of read counts.
8. Creating expression heatmaps.
9. Creating PCA plots, transcripts explaining first three PCs, correlations between first three PCs and read numbers.
10. Creating annotation table with coding status.
11. Creating BED12 files with all transcripts.
12. FASTA files with coding and non-coding transcripts' nucleotide sequences, together with sequences of transcripts containing novel exons.
13. Annotation table and FASTA files with nucleotide sequences for transcripts containing exon skipping events.

5.1.4 Splice-site-based pipeline

Splice-site-based pipeline (left side of the *Pipeline concept*) can be used to annotate abundant canonical splice sites. Because of relatively high error rate in long read sequencing technologies (especially Oxford Nanopore), quantification of obtained transcripts may be less accurate, nevertheless transcript order (from the most to the least abundant) should be preserved. The pipeline is split to two files (part4 and part5).

Part 4

This part is the complete pipeline to annotate and quantify alternative splice sites in all samples. Steps of this part of the pipeline comprise of:

1. Creating a genome index (with LAST).
2. Obtaining substitution rates from FASTA file with last-train.
3. Mapping reads to the reference genome with LAST.
4. Parsing alignments to separate aligned from unaligned parts of reads and to generate chromosomal positions of each exon.
5. Assessment of perfectly aligned splice sites.
6. Correcting reads to canonical splice sites (with different thresholds).
7. Creating a file with transcript names and sequences.
8. Downweighting multi-mapped reads.
9. Creating a histogram of reads per sample.
10. Filtering out transcripts with low abundance.
11. TMM normalisation of read counts (removing transcripts with geometric mean equal to zero, as well as adding 10^-6 to all downweighted counts).
12. Creating expression heatmaps.
13. Creating PCA plots, transcripts explaining first three PCs, correlations between first three PCs and read numbers (with native and log2-transformed expression values).
14. Creating annotation table with coding status.
15. Creating BED12 files with all transcripts.
16. Comparison between two approaches.

Part 5

The last part of the pipeline can be ran if there is a huge bias for read numbers (in the magnitude of 1000s) between different samples (which can be observed in the correlation plot generated at step no. 3 of Part 2). It starts with corrected reads. The steps of this part of the pipeline are:

1. Downsampling corrected reads.
2. Downweighting multi-mapped reads.
3. Creating a histogram of reads per sample.
4. Filtering out transcripts with low abundance.
5. TMM normalisation of read counts (removing transcripts with geometric mean equal to zero, as well as adding 10^-6 to all downweighted counts).
6. Creating expression heatmaps.
7. Creating PCA plots, transcripts explaining first three PCs, correlations between first three PCs and read numbers (with native and log2-transformed expression values).
8. Creating annotation table with coding status.
9. Creating BED12 files with all transcripts.
10. BedGraph file with annotation counts.

5.2 Installation

5.2.1 Software depenedencies

Our pipeline uses a number of software packages. To facilitate their installation you can use conda package manager.

Automated installation with conda

After downloading the proper conda version from *https://docs.conda.io/en/latest/miniconda.html*, you can install it by typing in the directory with the installation script:

```
$ bash ./Miniconda3-latest-Linux-x86_64.sh
```

Then, you need to install git and Snakemake:

```
$ conda install git snakemake-minimal
```

With git, you can download the GitHub repository with our pipeline (to the directory of your choice):

```
$ mkdir taqlore_dir
$ cd taqlore_dir
$ git pull https://github.com/twrzes/TAQLoRe.git
```

Then, you can run the Snakemake file which should install all the dependencies. Alternatively, especially if you use HPC to run our pipeline (which is preferred method of running it) and you do not have access to Internet on all nodes, you can pre-install the environment by using *conda env create*:

```
$ conda env create --file taqlore_dir/TAQLoRe/envs/taqlore.yaml
```

Manual installation

If you prefer not to use conda, you can install all the dependencies manually. Software versions in brackets denote ones that we used to develop our pipeline so they should work without any problems. Software dependencies include:

- Python (3.5.1)
- Perl (5.22.1)
- R (3.5.1)
- Bedtools (2.26.0)
- LAST (979)
- GMAP (20190315)
- Git (2.21)
- Snakemake (5.4.5)

Also, the workflow requires the following Python modules:

- pandas (0.24.2)
- numpy (1.16.3)
- tqdm (4.32.1)
- pysam (0.15.2)
- pybedtools (0.8.0)
- scipy (1.2.1)

- natsort (6.0.0)

It also requires following R libraries:

- heatmap3
- ggfortify
- ggplot2
- FactoMineR
- factoextra
- sva

After installing all the dependencies simply clone the GitHub repository (after creating a directory when you want to store the instance of the pipeline):

```
$ mkdir taqlore_dir
$ cd taqlore_dir
$ git pull https://github.com/twrzes/TAQLoRe.git
```

5.3 Running the pipeline - exon-based approach

As you may have read in *General concepts*, the pipeline as a whole is comprised of five parts, three of which are used for exon-based analyses, and remaining two for splice-site-based method.

5.3.1 Snakemake config file

The first step of the pipeline is editing the Snakemake JSON config file `config_TAQLoRe.json`. All the parameters and paths to input files are stored there. There are five main keys ('sections') in the configuration file.

workdir

In this section there is only one key - `` `workdir ``, denoting the path to the working directory.

samples

This section of config file contains two keys:

- `fasta_file_dir`: Directory where FASTA files are stored.

> **Warning:** All the FASTA files must have consistent naming in the format of {run_prefix}.{barcode}.fa, where {run_prefix} is a prefix of the run (i.e. date of sequencing), and {barcode} is a barcode number. This must also be consistent with barcode-to-sample mapping file, as described in *How to create a barcode-to-sample mapping file*.

- `downsampled_reads_num`: Number of reads to downsample to in third part of the pipeline - *Read counts bias - downsampling*.

input_files

This section contains paths to all necessary input files:

- `transcriptome_fasta`: Path to the FASTA file with all transcript sequences.

> **Warning:** All transcript names (i.e. headers of each FASTA sequence) **MUST NOT** contain any spaces, as they cannot be parsed correctly.
>
> Moreover, FASTA file should be single-line FASTA file (i.e. each sequence must contain two lines - first being FASTA header, and the second one being a sequence).

- `last_transcriptome_index`: Prefix of LAST index for a transcriptome.
- `last_transcriptome_index_dummy_file`: Path to the dummy file to be created after generating LAST index for a transcriptome (required to maintain the order of steps in the pipeline).
- `genome_fasta`: Path to the file with genome FASTA sequence.

> **Warning:** All chromosome names must be consistent between this file and both GTF file and sections of Snakemake config file. The file **MUST NOT** contain any spaces and be single-line FASTA file.

- `chrom_sizes`: Path to the file containing chromosome sizes for a genome.
- `last_genome_index`: Prefix of the LAST index for a genome.
- `last_genome_index_dummy_file`: Path to the dummy file being created after generating LAST index for a genome (required to maintain the order of steps in the pipeline).
- `gtf_file`: Path to the GTF file with gene annotations.

> **Warning:** All chromosome names must be consistent between this file and both genome FASTA file and sections of Snakemake config file.

- `all_exons_positions_ENSEMBL`: Path to the file containing gene annotations from ENSEMBL BioMart. You can find how to create this file in this section: *How to create an annotation file with ENSEMBL bioMart*.
- `barcode_to_sample_file`: Path to the file with barcode-to-sample mappings. The structure of the file can be found in this section: *How to create a barcode-to-sample mapping file*.
- `barcode_to_sample_file_downsampled`: Path to the file with barcode-to-sample mappings for downsampling. The structure of the file can be found in this section: *How to create a barcode-to-sample mapping file*. More about downsampling can be found in *Read counts bias - downsampling*.

How to create an annotation file with ENSEMBL bioMart

To create a file needed in the pipeline, you need to use ENSEMBL BioMart. The reason behind it is that some information required to run the pipeline may not be included in the GTF file (e.g. UTRs).

The file can be created with following steps:

1. Go to http://www.ensembl.org/biomart/martview
2. From **ENSEMBL Genes** database choose the dataset of interest (e.g. **Human Genes**).
3. From **Filters** section on the left-hand side of the website choose the gene of interest (e.g. select **Input external references ID list [Max 500 advised]**, and put ENSEMBL gene ID into the field).

4. From **Attributes** section on the left-hand side of the website select **Structures**, deselect all attributes (from **GENE** subsection), and select following attributes (**IN THIS PARTICULAR ORDER**) (subsections of attributes are written in square brackets):

- Gene stable ID [GENE]
- Gene start (bp) [GENE]
- Gene end (bp) [GENE]
- Transcript stable ID [GENE]
- Transcript start (bp) [GENE]
- Transcript end (bp) [GENE]
- Transcription start site (TSS) [GENE]
- Exon stable ID [EXON]
- Exon region start (bp) [EXON]
- Exon region end (bp) [EXON]
- Exon rank in transcript [EXON]
- cDNA coding start [EXON]
- cDNA coding end [EXON]
- Genomic coding start [EXON]
- Genomic coding end [EXON]

5. Click **Results** button on top-left side of the website.
6. From **Export all results to ** section, choose **File** and **TSV** format, then click **Go** and save the file in the desired location.

How to create a barcode-to-sample mapping file

The file with metadata is a tab-delimited file (without header) containing three columns:

Run prefix	Barcode	Sample name
2017_01_13	barcode01	Jan_5238_cingulate
2017_06_15	barcode12	Jun_5346_striatum

where:

- `Run prefix` - is the run prefix for each file. This can be e.g. a date of the sequencing or any string that denotes different sequencing batches.
- `Barcode` - is the barcode of each file.
- `Sample name` - is the underscore-separated string with a following structure: `{run_name}_{sample_id}.{sample_sub_id}`. In the example above `{run_name}` denotes two sequencing runs (one in January, second one in June), `{sample_id}` denotes different individuals, and `{sample_sub_id}` denotes different brain regions.

gene_info

This section of config file contains the information about analysed gene.

- `gene_name`: Gene name for the gene of interest.
- `chromosome_name`: Chromosome name for the gene of interest.

> **Warning:** All chromosome names must be consistent between this section and both genome FASTA file and GTF file.

- `gene_start`: Start position of the gene (0-based coordinates).
- `gene_end`: End position of the gene (0-based coordinates).
- `strand`: Strand of the gene ('+' or '-').

parameters

This section of config file contains all the parameters being used by scripts.

- `min_prop`: Minimum proportion of read covering a transcript.
- `min_prop_align`: Minimum proportion of transcript being covered by aligned read.
- `min_insert`: Minimum length of potential novel exon (insertion in the alignment).
- `min_exon_distance`: Minimum distance from annotated exon.
- `distance_between_exons`: Minimum distance between exons (both known and novel).
- `exon_coverage_threshold`: Minimum coverage of exon for exon to be included in transcripts.
- `min_exon_length`: Minimum length of the exon.
- `min_reads_threshold`: Minimum number of reads covering the exon for the exon to be included in transcripts.
- `min_num_individuals_threshold`: Minimum number of different individuals (`{sample_id}`) having at least `min_reads_threshold` reads per exon.
- `min_num_libraries_threshold`: Minimum number of different tissues (`{sample_sub_id}`) having at least `min_reads_threshold` reads per exon.
- `sum_threshold`: Minimum sum of reads for a transcript to be included in the annotation.
- `reads_in_sample_threshold`: Minimum number of reads per sample for a transcript to be included in the annotation.
- `sample_threshold`: Minimum number of samples having at least `reads_in_sample_threshold` reads for a transcript to be included in the annotation.

5.3.2 Running the pipeline

Local computer/server

To run each part of the pipeline on a computer/server, you can run it by typing:

```
$ cd /path/to/TAQLoRe
$ conda activate taqlore
$ snakemake -j number_of_cores -s TAQLore_part_1
$ snakemake -j number_of_cores -s TAQLore_part_2
```

etc.

where `{number_of_cores}` is the number of cores to use for a Snakemake run.

Note: LAST alignements are very memory-intensive - for 100k reads LAST uses ~48G of memory (human genome/transcriptome). Therefore, the best way to run the pipeline is to use HPC (see below).

Note: The way how snakemake is run in the example above requires a user to pre-install the whole environment located in *envs/taqlore.yaml*. To pre-install this environment please refer to *Installation*. If a user wants to create conda environment from scratch, they can use *snakemake -j {number_of_cores} -s TAQLore_part_1 –use-conda* command which will create a local copy of the whole environment in the working directory.

High Performance Computing

In order to run Snakemake pipeline on a computational cluster (preferred way), two additional files must be created

Cluster configuration file

For additional information, refer to Snakemake documentation.

This file contains all parameters for each jobs to be used by the scheduler, such as time, memory, number of CPUs, partition name, etc.

The example JSON file to run using SLURM scheduler (Earlham Institute's infrastructure) looks like this:

```
{
  "__default__" :
    {
        "nodes" : 1,
        "time" : "7-00:00:00",
        "n" : 1,
        "ntasks-per-node": 1,
        "cpu" : 1,
        "partition" : "medium",
        "memory" : "64G",
        "job_name" : "{rule}.{wildcards}",
        "out" : "slurm.%N.%j.{rule}.{wildcards}.out",
        "err" : "slurm.%N.%j.{rule}.{wildcards}.err"
    },
    "last_index_transcriptome" :
    {
        "time" : "1-00:00:00",
        "cpu" : 8,
        "memory" : "64G"
    },
    "last_train_gap_mismatch_transcriptome" :
    {
        "time" : "1-00:00:00",
```

(continues on next page)

(continued from previous page)

```
        "cpu" : 8,
        "memory" : "64G"
    },
    "last_align_transcriptome" :
    {
        "time" : "7-00:00:00",
        "cpu" : 8,
        "memory" : "64G"
    },
    "last_index_genome" :
    {
        "time" : "1-00:00:00",
        "cpu" : 8,
        "memory" : "64G"
    },
    "novel_exons_alignment_to_genome_parsing_last_maf" :
    {
        "time" : "00:45:00",
        "cpu" : 8,
        "partition" : "short",
        "memory" : "32G"
    },
    "novel_exons_genomic_coordinates":
    {
        "time" : "00:45:00",
        "partition" : "short",
        "memory" : "16G"
    },
    "filtering_genomic_positions_gene_boundaries" :
    {
        "time" : "00:05:00",
        "partition" : "short",
        "memory" : "4G"
    },
    "novel_exons_per_library" :
    {
        "time" : "00:45:00",
        "partition" : "short",
        "memory" : "4G"
    },
    "novel_exons_summary" :
    {
        "time" : "00:45:00",
        "partition" : "short",
        "memory" : "4G"
    },
    "generate_BedGraph_sum" :
    {
        "time" : "00:45:00",
        "partition" : "short",
        "memory" : "4G"
    },
    "novel_exons_file_1nt_coordinates" :
    {
        "time" : "00:01:00",
        "partition" : "short",
        "memory" : "2G"
```

(continues on next page)

(continued from previous page)

```
    },
    "coordinates_genomic_meta_gene_exons" :
    {
        "time" : "00:05:00",
        "partition" : "short",
        "memory" : "16G"
    }
}
```

Note: `__default__` section specifies parameters of default job, while parameters under rule names (copied from Snakemake files, e.g. `last_index_transcriptome`) denote deviation(s) from default rule (e.g. different time, memory, number of CPUs, etc.).

Batch script to submit

In order to run each part of the pipeline using HPC, a batch script to submit must be created. An example of the batch script (SLURM scheduler, Earlham Institute's infrastructure) can be seen below:

```
#!/bin/bash
#SBATCH -p medium
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 1
#SBATCH --mem 4G
#SBATCH -t 7-00:00:00
#SBATCH -o slurm.%N.%j.out
#SBATCH -e slurm.%N.%j.err
#SBATCH --mail-type=ALL
#SBATCH --mail-user=some.user@some.insitute.ac.uk

module load conda

conda activate taqlore

srun snakemake -s TAQLoRe_part1 --latency-wait 60 -j {number_of_jobs} --cluster-
↪config /path/to/cluster/config.json --cluster "sbatch -p {cluster.partition} -N
↪{cluster.nodes} -n {cluster.n} --ntasks-per-node={cluster.ntasks-per-node} -c
↪{cluster.cpu} -t {cluster.time} --mem {cluster.memory} -J {cluster.job_name} -o␣
↪slurm.%N.%j.out -e slurm.%N.%j.err --mail-type=FAIL --mail-user=some.user@some.
↪insitute.ac.uk"
```

where:

- `--cluster-config` denotes path to the *Cluster configuration file*.
- `{number_of_jobs}` denotes number of jobs submitted to the cluster at once.

The script can be submitted with command (SLURM scheduler):

```
$ sbatch batch_script_to_submit.sh
```

where `batch_script_to_submit.sh` is the name of the file which contents are shown above (*Batch script to submit*).

5.3.3 Things to do after running part 1 of the pipeline

The first part of the pipeline ends with creating a meta-gene annotation file in `{workdir}/results/meta_gene_construction/meta_gene_genomic_exon_coordinates.txt`.

In order to run the second part of the pipeline, the user must annotate UTRs in the `meta_gene_genomic_exon_coordinates.txt` file and/or add/remove additional/unnecessary exons. The reason behind it is that there are different sources of UTRs (e.g. ENSEMBL, GENCODE, RefSeq, UCSC, etc.) and the differences between annotations can be significant.

UTRs must be added as the last column to the `meta_gene_genomic_exon_coordinates.txt` file, for each exon position in the meta-gene. The last column should have 'UTR' string denoting that a position is an UTR, and any other string (e.g. 'NA') denoting the non-UTR/unknown status of a genomic region in a gene. Example:

- `meta_gene_genomic_exon_coordinates.txt` after part1 of the pipeline:

```
1   277     1970786 1971062 ENST00000543114 ENSE00001774617 No
278 416     1971063 1971201 ENST00000543114 ENSE00001774617 Yes
417 656     2053298 2053537 ENST00000335762;ENST00000399655 ENSE00001539923;
↪ENSE00001539923 No;No
657 681     2053538 2053562 ENST00000335762;ENST00000399655;ENST00000480911␣
↪ENSE00001539923;ENSE00001539923;ENSE00001839973 No;No;No
682 730     2053563 2053611 ENST00000335762;ENST00000399655;ENST00000480911;
↪ENST00000399595;ENST00000399644;ENST00000399638;ENST00000399597;ENST00000399621;
↪ENST00000399637;ENST00000399591;ENST00000399641;ENST00000347598;ENST00000399606;
↪ENST00000399601;ENST00000344100;ENST00000399629;ENST00000327702;ENST00000399649;
↪ENST00000402845;ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454␣
↪ENSE00001539923;ENSE00001539923;ENSE00001839973;ENSE00001539466;ENSE00001539466;
↪ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;
↪ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;
↪ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;
↪Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes
...
23196       23495   2690900 2691199 ENST00000335762;ENST00000399655;
↪ENST00000399595;ENST00000399644;ENST00000399638;ENST00000399597;ENST00000399621;
↪ENST00000399637;ENST00000399591;ENST00000399641;ENST00000347598;ENST00000399606;
↪ENST00000399601;ENST00000344100;ENST0000039962 ;ENST00000327702;ENST00000399649;
↪ENST00000402845;ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454;
↪ENST00000616390 ENSE00002228600;ENSE00001539473;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001539391;
↪ENSE00001539391;ENSE00001539391;ENSE00001539391;ENSE00003738703 Yes;Yes;Yes;Yes;
↪Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes
23496       23563   2691200 2691267 ENST00000335762;ENST00000399655;
↪ENST00000399595;ENST00000399644;ENST00000399638;ENST00000399597;ENST00000399621;
↪ENST00000399637;ENST00000399591;ENST00000399641;ENST00000347598;ENST00000399606;
↪ENST00000399601;ENST00000344100;ENST00000399629;ENST00000327702;ENST00000399649;
↪ENST00000402845;ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454;
↪ENST00000616390 ENSE00002228600;ENSE00001539473;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001539391;
↪ENSE00001539391;ENSE00001539391;ENSE00001539391;ENSE00003738703 No;No;No;No;No;
↪No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No
23564       23589   2691268 2691293 ENST00000335762;ENST00000399655;
↪ENST00000399595;ENST00000399644;ENST00000399638;ENST00000399597;ENST00000399621;
↪ENST00000399637;ENST00000399591;ENST00000399641;ENST00000347598;ENST00000399606;
↪ENST00000399601;ENST00000344100;ENST00000399629;ENST00000327702;ENST00000399649;
↪ENST00000402845;ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454␣
↪ENSE00002228600;ENSE00001539473;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001539391;ENSE00001539391;
↪ENSE00001539391;ENSE00001539391 No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;
```

(continues on next page)

(continued from previous page)

```
23590       23706   2691294 2691410 ENST00000335762;ENST00000399655;
↪ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454 ENSE00002228600;
↪ENSE00001539473;ENSE00001539391;ENSE00001539391;ENSE00001539391;ENSE00001539391␣
↪No;No;No;No;No;No
23707       24455   2691411 2692159 ENST00000399655;ENST00000399603;
↪ENST00000399634;ENST00000399617;ENST00000406454 ENSE00001539473;ENSE00001539391;
↪ENSE00001539391;ENSE00001539391;ENSE00001539391 No;No;No;No;No
24456       30246   2692160 2697950 ENST00000399655 ENSE00001539473 No
```

- `meta_gene_genomic_exon_coordinates.txt` after adding UTR annotations (before running part2 of the pipeline):

```
1   277     1970786 1971062 ENST00000543114 ENSE00001774617 No      UTR
278 416     1971063 1971201 ENST00000543114 ENSE00001774617 Yes     NA
417 656     2053298 2053537 ENST00000335762;ENST00000399655 ENSE00001539923;
↪ENSE00001539923 No;No   UTR
657 681     2053538 2053562 ENST00000335762;ENST00000399655;ENST00000480911␣
↪ENSE00001539923;ENSE00001539923;ENSE00001839973 No;No;No        UTR
682 730     2053563 2053611 ENST00000335762;ENST00000399655;ENST00000480911;
↪ENST00000399595;ENST00000399644;ENST00000399638;ENST00000399597;ENST00000399621;
↪ENST00000399637;ENST00000399591;ENST00000399641;ENST00000347598;ENST00000399606;
↪ENST00000399601;ENST00000344100;ENST00000399629;ENST00000327702;ENST00000399649;
↪ENST00000402845;ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454␣
↪ENSE00001539923;ENSE00001539923;ENSE00001839973;ENSE00001539466;ENSE00001539466;
↪ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;
↪ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;ENSE00001539466;
↪ENSE00001539466;ENSE00001539466;ENSE00001539466 Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;
↪Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes     NA
...
23196       23495   2690900 2691199 ENST00000335762;ENST00000399655;
↪ENST00000399595;ENST00000399644;ENST00000399638;ENST00000399597;ENST00000399621;
↪ENST00000399637;ENST00000399591;ENST00000399641;ENST00000347598;ENST00000399606;
↪ENST00000399601;ENST00000344100;ENST00000399629;ENST00000327702;ENST00000399649;
↪ENST00000402845;ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454;
↪ENST00000616390 ENSE00002228600;ENSE00001539473;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001539391;ENSE00001539391;
↪ENSE00001539391;ENSE00001539391;ENSE00001539391;ENSE00003738703 Yes;Yes;Yes;Yes;
↪Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes;Yes     ␣
↪NA
23496       23563   2691200 2691267 ENST00000335762;ENST00000399655;
↪ENST00000399595;ENST00000399644;ENST00000399638;ENST00000399597;ENST00000399621;
↪ENST00000399637;ENST00000399591;ENST00000399641;ENST00000347598;ENST00000399606;
↪ENST00000399601;ENST00000344100;ENST00000399629;ENST00000327702;ENST00000399649;
↪ENST00000402845;ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454;
↪ENST00000616390 ENSE00002228600;ENSE00001539473;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001539391;ENSE00001539391;
↪ENSE00001539391;ENSE00001539391;ENSE00001539391;ENSE00003738703 No;No;No;No;No;
↪No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No    UTR
23564       23589   2691268 2691293 ENST00000335762;ENST00000399655;
↪ENST00000399595;ENST00000399644;ENST00000399638;ENST00000399597;ENST00000399621;
↪ENST00000399637;ENST00000399591;ENST00000399641;ENST00000347598;ENST00000399606;
↪ENST00000399601;ENST00000344100;ENST00000399629;ENST00000327702;ENST00000399649;
↪ENST00000402845;ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454␣
↪ENSE00002228600;ENSE00001539473;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001724521;
↪ENSE00001724521;ENSE00001724521;ENSE00001724521;ENSE00001539391;ENSE00001539391;
↪ENSE00001539391;ENSE00001539391 No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;No;
↪No;No;No;No;No;No       UTR
```

(continues on next page)

(continued from previous page)

```
23590       23706   2691294 2691410 ENST00000335762;ENST00000399655;
↪ENST00000399603;ENST00000399634;ENST00000399617;ENST00000406454 ENSE00002228600;
↪ENSE00001539473;ENSE00001539391;ENSE00001539391;ENSE00001539391;ENSE00001539391␣
↪No;No;No;No;No;No        UTR
23707       24455   2691411 2692159 ENST00000399655;ENST00000399603;
↪ENST00000399634;ENST00000399617;ENST00000406454 ENSE00001539473;ENSE00001539391;
↪ENSE00001539391;ENSE00001539391;ENSE00001539391 No;No;No;No;No  UTR
24456       30246   2692160 2697950 ENST00000399655 ENSE00001539473 No      UTR
```

After adding UTR annotation to the `meta_gene_genomic_exon_coordinates.txt` file, the pipeline can be run as usual (file: `TAQLoRe_part2`).

5.3.4 Read counts bias - downsampling

Third part of the pipeline () can be run in order to remove read counts bias (i.e. different number of reads in analysed samples). This step is necessary if you have huge differences in read numbers (> 5000 reads difference between samples with highest and lowest number of reads).

Before running part3 of the pipeline

Before running part3 of the pipeline two additional steps are necessary:

- Before running the pipeline `config_TAQLoRe.json` needs to be edited (in the section `"samples"`, sub-section `downsampled_reads_num`), to reflect the number of reads to choose for all the files. The downsampling will be done on `/path/to/workdir/results/meta_gene_exon_counts_splicing_patterns/{run_prefix}.{barcode}_splicing_patterns_cds.tmp` files, where `{run_prefix}` and `{barcode}` first and second column from *How to create a barcode-to-sample mapping file* file, respectively. Thus, to see the number of reads in each file (sorted by the number of reads) the following command may be invoked to see the number of reads in each sample:

```
$ cd /path/to/workdir
$ for i in `ls results/meta_gene_exon_counts_splicing_patterns/*_splicing_patterns_cds.tmp`; do j=`ca
```

- A meta-data file needs to be edited, as removing some outliers with the lowest numbers of reads might be necessary to accurately compare the expression between samples. The file has the same structure as the one in *How to create a barcode-to-sample mapping file*. The path to this file should be put in `config_TAQLoRe.json`, section `input_files`, sub-section `barcode_to_sample_file_downsampled`.

After these steps the pipeline can be run as usual (file: `TAQLore_part3`).

5.4 Running the pipeline - splice-site-based approach

5.5 Output files - exon-based approach

5.6 Output files - exon-based approach