VEuPathDB RNA-Seq alignment

EBI pipeline overview

EBI processing

Requirements:

Data

- For each genome an Ensembl core database
- A json dataset file

Programs

- An ensembl checkout with ehive
- The EBI RNA-Seq alignment pipeline
- Python package RSeQC (for infer_experiment.py)
- Python package htseq
- Trimmomatic
- bedtools

- bedSort
- bamutils

All data exchanged must follow the specifications.

The following processing is done by the EBI RNA-Seq pipeline.

Genome data preparation

- Extract the DNA sequence from the core into a fasta file
- Index the fasta file for hisat2 with hisat2-build (without splice sites file or exons file)
- Extract the gene models from the core in a gtf format (for htseq-count)
- Extract the gene models from the core in a bed format (for strand inference)

RNA-Seq data retrieval

For each run, download the fastq data files from ENA using its SRA accession.
All following processes are performed for each run.

Pre-alignment processes

Trimming

If the run has a flag “trim_reads” set to true, then fastq files are trimmed using trimmomatic using the following parameters:
ILLUMINACLIP:$adapters:2:30:10
LEADING:3
TRAILING:3
SLIDINGWINDOW:4:15
MINLEN:20

Where $adapters is the path to the adapters directory from trimmomatic. There is one directory for paired-end, and one for single-end reads (the paired/single information must be provided in the json dataset).

Strandness inference

To ensure that the aligner uses the correct strand information, an additional step to infer the strandness of the data is necessary.

Steps:

1. Create a subset of reads files with 1 million reads
2. Align those files (without strandness) with hisat2
3. Run infer_experiment.py on the alignment file

The inference compares how the reads are aligned compared to the known gene models:

- If most reads expected to be forward are forward (and vice versa), then the data is deemed as stranded in the forward direction.
- If most reads expected to be forward are reversed (and vice versa), then the data is deemed as stranded in the reverse direction.
- If the reads are equally in both directions, then the data is deemed unstranded.

A cut-off at 85% of aligned reads is applied to discriminate between stranded data: if the ratio is below this value, then the run is deemed unstranded.

Following hisat2 notation:

- If the data is stranded forward and single-ended, its strandness is stored as “F”
- If the data is stranded reversed and single-ended, its strandness is stored as “R”
- If the data is stranded forward and paired-ended, its strandness is stored as “FR”
- If the data is stranded reversed and paired-ended, its strandness is stored as “RF”

If the strandness or the pair/single-end values differ from those provided in the dataset json file, then the difference is noted in the log file with a WARNING. The infer_experiment output is also stored in this file.
Note that If the values differ, the pipeline continues running using the values inferred.

Alignment

Using hisat2 with the following parameters:

- --max-intronlen $max_intron_length (see for the value)
- --rna-strandness $strandness (if stranded, the value is either “F”, “R“, “FR”, or “RF”)

The --max-intronlen parameter is computed from the gene set data in the core database, as the maximum of:

- int(1.5 * $max_intron_length)
    - where $max_intron is the longest intron in the gene set
- 500,000

Hisat2 generates an unsorted sam file.

Bam generation

From the file generated by Hisat2, there are two additional conversion steps:

1. Main bam: sorting by position + conversion to bam file
2. Name sorted bam: sorting by read name + filter out unaligned reads + conversion to bam file

The first file is temporary and is deleted as the end of the pipeline.

The second file is to be conserved so as to be reused for htseq-count for the next time there is a gene set update.

The process also generates additional temporary bam files, extracted from the main bam:

- One for unique reads
- One for non-unique reads

If the data is stranded, then each unique/non-unique bam file is also split into:

- Forward stranded reads
- Reverse stranded reads

So if the data is stranded, 4 files are generated. If it is unstranded, 2 files are generated.

Post-alignment processing

Bam stats

Samtools stats are run on the main bam file, as well as all final split bam files.

The following values are computed:

- coverage (computed with bedtools genomecov)
- mapped (from samtools stats “reads_mapped” / “raw total sequences”)
- number_reads_mapped (from samtools stats “reads mapped”)
- average_reads_length (from samtools stats “average length”)
- number_pairs_mapped (if paired, from samtools stats “reads properly paired”)

Note that for the split bam files, the “mapped” number ratio is over the main bam “raw total sequences”.

Create BedGraph

For each split bam file, the pipeline creates a coverage file in BedGraph format, using:

- bamutils tobedgraph (with stranded parameters --plus or --minus if stranded)
- bedSort

Extract junctions

From the main bam file, the pipeline extracts all the splice junctions into a tabulated text file.

The pipeline extracts the junctions as follow:

- For each read aligned with “N”s in its cigar string
- Get the strand direction from the XS tag
- Get the uniqueness from the NH tag
- Create a splice junction for each group of Ns found in the cigar string, with the coordinates, strand and uniqueness

HTSeq-count

From the bam file sorted by name, the pipeline runs HTSeq-count:

- Once using unique reads
- Once using all reads (note that this file is named “nonunique” because the parameter used is “--nonunique all” instead of “--nonunique none”)
- If the reads are stranded in the forward direction, use --stranded=yes
- If the reads are stranded in the reverse direction, use --stranded=reverse
- If the reads are not stranded, use --stranded=no
- With parameter --order=name
- With parameter --type=exon --idattr=gene_id
- With parameter --mode union
- With the gtf file

This is done for each strand, so there will be 4 HTSeq-count files for stranded data, and 2 for unstranded data.

Finalisation

Once all files are generated for a run, the pipeline writes two metadata files:

- A commands file with the list of commands run (hisat2, etc.)
- A metadata file in json that contains in particular the strandness and paired/single end metadata. This is important for the ulterior htseq-count runs.

At the end of the pipeline, each run contains all its data files in a directory using its run name defined in the dataset file (note that the name can be any SRA accession like SRX…, this works as long as there is only one read per accession).

Each run directory contains the following files:

- mappingStats.txt

- metadata.json
- log.txt
- junctions.tab
- commands.json
- 2 or 4 htseq-count files
- 2 or 4 bed files
- EBI conserves the final name sorted bam file, but this file is not transferred to UPenn

Each run directory is in a dataset directory.
Each dataset directory is stored in a genome directory using the organism_abbrev for this genome.
Each genome directory is stored in a component directory (ToxoDB, VectorBase, etc.).

Gene set update

HTSeq-count rerun

When a gene set is updated, only the HTSeq-count needs to be rerun. In that case the pipeline will only retrieve the metadata from the metadata file (to get the strandness and paired/single status), and run htseq-count on the conserved bam file sorted by name.

When only HTSeq-count is rerun, only the htseq-count files need to be transferred; this can be done in the form of a single tar file.