MicroExonator

Jan 14, 2021

Contents

MicroExonator is a fully-integrated computational pipeline that allows for systematic de novo discovery and quantification of microexons using raw RNA-seq data for any organism with a gene annotation. Compared to other available methods MicroExonator is more sensitive for discovering smaller microexons and it provides higher specificity for all lengths. Moreover, MicroExonator provides integrated downstream comparative analysis between cell types or tissues using Whippet. (Sterne-Weiler et al. 2018).

MicroExonator pipeline is divided in several modules:

- Discover
- Quantification
- Differential Inclusion
- Single cell analysis

Support

For questions, ideas, feature requests and potential bug reports submit an issue on our GitHub page or write us at gp7@sanger.ac.uk.

CHAPTER 1

Installation

To install MicroExonator follow these instructions:

1.1 Clone repository

Clone the github repository

```
git clone https://github.com/hemberg-lab/MicroExonator
```

Install Miniconda 3

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
chmod +x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh
```

Start using conda by opening a new terminal or just running:

```
bash
```

1.2 Set up a master virtual environment

Create a conda virtual enviroment with the necesary dependencies

```
conda create -n snakemake_env -c bioconda -c conda-forge snakemake
```

CHAPTER 2

Setup

Before runnung MicroExonator there are several files that needs to be created inside `MicroExonator/` root folder:

2.1 RNA-seq samples

Input RNA-seq data either a `local_samples.tsv`, `NCBI_accession_list.txt` or `sample_url.tsv` needs to be defined. If you want to run MicroExonator over RNA-seq samples that are locally stored, they need to be defined inside `local_samples.tsv`. MicroExonator can also download and run samples from NCBI if the corresponding SRA accession names are defined inside of `NCBI_accession_list.txt`, in addition any `fastq.gz` that can be directly download from a URL can be included into the aalysis by defining them inside a `sample_url.tsv`. You can find examples of these files inside the `Examples/` folder. Is posible to combine different types of input sources, but at least one of these files needs to be defined inside `MicroExonator/` root folder.

2.2 Cluster configuration

If you are working on a high performace cluster, then it is very likely that you need to submit jobs to queueing systems such as lsf, qsub, SLURM, etc. To make MicroExonator work with these queueing systems, you need to create a *cluster.json* file. We currently provide in the Examples folder a `cluster.json` file to run MicroExonator with lsf. To adapt MicroExonator to other quequing systems please see the SnakeMake documentation.

2.3 Config file

Each MicroExonator's module has certain compulsory and optional parameters that needs to be defined inside a `config.yaml` file. The necesary content of `config.yaml` is described on each moudle section and examples can be found at the `Examples/` folder.

CHAPTER 3

Discovery and Quantification

The main core modules from MicroExonator correspond to the Discovery and Quantification modules. On a regular MicroExonator run, both modules are run however is also posible to run these modules independely. By default, all the downloaded files are temporarely stored in `MicroExonator/FASTQ` folder, which means that several GBs of disk space may be required.

3.1 Configuration

Before running Discovery and Quantification modules is necesary to set all the required parameters inside the `config.yaml` file.

3.1.1 Minimal configuration

In this seccetion we describe the minimal set of parameters that needs to be defined before running MicroExonator. Since these modules correspond to the core of the pipeline, these parameters are compusory, even if they are not used by the rest of the modules. An example of a `config.yaml` file that contain all these parameters would look like this:

```
Genome_fasta : /path/to/Genome.fa
Gene_anontation_bed12 : /path/to/ensembl.bed12
GT_AG_U2_5 : /path/to/GT_AG_U2_5.good.matrix
GT_AG_U2_3 : /path/to/GT_AG_U2_3.good.matrix
conservation_bigwig : /path/to/conservation.bw
working_directory : /path/to/MicroExonator/
ME_len : 30
Optimize_hard_drive : T
min_number_files_detected : 3
```

Here:

- `Genome_fasta` is a multifasta file containg the cromosomes.
- `Gene_anontation_bed12` is a BED file containing the transcript annotation. A collection of these files can be found at UCSC Table Browser.

- `GT_AG_U2_5` and `GT_AG_U2_5` are splice site PWMs that can come from SpliceRack. Examples of these inmput files can be found at `/PWM` Folder. In th case there are not any splice site PWMs available for your specie of interest, you cam assing them *NA* as a value and MicroExonator will generate the PWM internally based on annotated splice sites.
- `conservation_bigwig` is a bigwig file containing genome-wide conservation scores generated by Pylop or PhastCons, which can be downloaded from UCSC genome browser for some species. If you do not have a `conservation_bigwig` file, you need to create an bigwig which has the value 0 for every position in your genome assembly.
- `working_directory` is the path to the MicroExonator folder that you are working with
- `ME_len` is microexon maximum lenght. The default value is 30.
- `Optimize_hard_drive` can be set as `T` or `F` (true of false). When it is set as `F`, fastq files will be downloaded or copied only once at `FASTQ/` directory. This can be inconvenient when you are analysing large amout of data (for instance when your input data is larger than 1TB), because the copy will not be deleted until MicroExonator finish completely. When `Optimize_hard_drive` is set as `T` instead, two independet local copies will be generated for the discovery and quantification moduled. rapAs these are temporary copies, every copied fastq file will be deleted as soon as is mapped to the splice junction tags, which means the disk space usage will be set to the minimum while MicroExonator is runing.
- `min_number_files_detected` correspond to the minumun number of in which a microexon needs to be found in order to concider it as high confidence. Setting this number to at least 2 is recommended single-end FASTQ files and 3 if paired end files are also present.

Note: If the corresponding files for `GT_AG_U2_5`, `GT_AG_U2_5` or `conservation_bigwig` are not available for the specie you are currently working on, you can set any of these parameters as `NA`.

3.1.2 Optional configuration

The following parameters can be specified to further optimize the Discovery and Quantification. The can also be omited, in case there are not relevant or available.

- `ME_DB` is a path to a known Microexon annotation file, such as the one that can be obtained from Vast DB. The input format must be in bed12 (additional formats will be supported in future versions)
- `min_reads_PSI` is the minimun number of reads that needs supoort the existence of a novel microexon to consider it as high confidence. The default value is 3, but 5 or more is recommended if enough RNA-seq samples are provided.
- `paired_samples` is a path to a tab-delimited file with two columns that indicate the correspndace between paired end samples names. This will enable MicroExonator to report a single quantification output per paired-end sample.

Note: In order to access `Vast DB` annotation files as bed12, you can try by connecting Vast DB track hub to UCSC Genome browser (instructions). Then by using the UCSC Table Browser you will be able to access their alternative splicing event annotation as bed12.

3.2 Run

If you are working remotely, we highly recommend creating a screen before running MicroExonator:

```
screen -S session_name  #choose a meaning full name to you
```

To activate snakemake enviroment

```
conda activate snakemake_env
```

In case you already have an older version of conda use instead:

```
source activate snakemake_env
```

Then run

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs}
```

Warning: In order to run the command above you need to replace `{cluster system params}` and `{number of parallel jobs}` with the appropiate values.

You should use `--cluster` only if you are working in a computer cluster that uses a queuing systems. We provide an example of `cluster.json` to work with lsf, in this case `cluster system params` should be replaced with the specific parameters of lsf. The `number of parallel` jobs can be a positive integer, the appropriate value depends on the capacity of your machine but for most users a value between 5 and 50 is appropriate. As an example the following command would be able to run MicroExonator using lsf and allowing for 500 parallel jobs.

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster "bsub -n
↪{cluster.nCPUs} -R {cluster.resources} -c {cluster.tCPU} -G {cluster.Group} -q
↪{cluster.queue} -o {cluster.output} -e {cluster.error} -M {cluster.memory}" --use-
↪conda -k  -j 500 -np
```

Before running it is recommended to check if SnakeMake can corretly generate all the steps given your input. To do this you can carry out a dry-run using the `-np` parameter:

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs} -np
```

The dry-run will display all the steps and commands that will be excecuted. If the dry-run cannot be initiated, make sure that you are running MicroExonator from inside the folder you cloned from this repository. Also make sure you have the right configuration inside `config.yaml`.

Note: If you are working remotelly, the connection is likely to die before MicroExonator finish. However, as long as you are working within an screen, you can re attach the screen and see MicroExonator progress. To list your active screens you can do:

```
screen -ls
```

To reattach and detach screens just use:

```
screen -r session_name  # only detached screen can be reattached
screen -d session_name
```

Warning: If you have any errors while you are running MicroExonator is useful to read the logs that are reported by the queuing system. Some errors may occur because when not enough memory has been allocated for a given step. Resources for each step can be this can be configured inside `cluster.json` (check example file at `MicroExonator/Examples/Cluster_config/lsf/`)

3.2.1 Running large datasets

Since MicroExonator wad developed as an snakemake workfolow, its possible to scale the analysis to big datasets. however there are a copule of recommendations you should keep in mind when you are running a large quantituity of samples.

Limit your downloading jobs

If you are fetching multiple fastq files from NCBI, is posible that some of the download processes will fail. To avoid overloading the connection with NCBI you can limmit the amout of dowloading jobs by assinging a maximun value to a resource called `get_data`. For example if you want to limmit the downloading process to only run one job at the time, the running command would be:

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs} --resources get_data=50
```

Optimize your hardrive space

If you want to process a large dataset, is likely that you will not have enough disk space to temporarely store all the FASTQ files at the same time. In this case we recommend to run the Discovery and Quantification module independely and set `Optimize_hard_drive` as `T` on the `config.yaml` file. In order to do this you use `discovery` as a target and snakemake will only excecute the Discovery module:

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs} discovery
```

By doing this, as `Optimize_hard_drive` is set as `T`, downloaded FASTQ files will be deleted as soon as they processed on this module. Once the pipeline finish the discovery module successfuly, you can resume the analysis by running MicroExonator with `quant` as a target:

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs} quant
```

Note that in this case, previously deleted FASTQ files will be downloaded again (or copied if you are using samples that are stored locally). The advantage of doing this is to reduce the amout of space required to to run the analysis as FASTQ files deleted as soon as they are processed these two modules.

Mind the number of files being generated

HPC systems often have a disk qouta limit, but they may also have quota for the maximun number of files that can be generated by each user. In the case you want delete uncesserary files after a MicroExonator run is completed, you can delete the `.snakemake/` and `logs/` folder.

```
rm -rf .snakemake logs
```

Note: If the pipeline gets interrupted or you encounter any error, you can re-intiate MicroExonator by submiting a running command again. This will not run again processes that finished successfuly, but it will only submit the jobs that are required to generate the files that are missing to complete the run. If at any point you want to start from scratch using the same path, you can delete `/download` folder to ensure every sample is processed again.

3.3 Output

The main results of MicroExonator discovery and quantification modules can be found at the Results folder. All the detected microexons that passed though the quantitative filters can be found at `out.high_quality.txt`. This is a tabular separated file with 14 columns that contain the folowing information:

Table 1: **out.high_quality.txt**

Column	Description
ME	Microexon Coordinates
Transcript	Transcript where the microexon was detected
Total_coverage	Total coverage across all microexon splice junctions
Total_SJs	Splice junctions where the micrexon was detected in
ME_coverages	Coma-separated coverage values for each microexon splice junction
ME_length	Microexon length
ME_seq	Microexon sequence
ME_matches	Microexon number of matches inside the intron
U2_score	U2 splicing score
Mean_conservation	Mean conservation values (if phylop score was provided)
P_MEs	Microexon confidence score
Total_ME	ME coordinates, U2 score and conservation for all microexon matches
ME_P_value	Value used for the final microexon filters
ME_type	Microexon type (IN, RESCUED or OUT)

MicroExonator also reports microexons that do not meet the confidence filtering criteria. Detected microexons that are equal or shorter than 3 nt are reported at `out_shorter_than_3_ME.txt`. Microexons that are longer than 3 nt, but did not have sufficiently low ME_P_value are reported at `out_low_scored_ME.txt`. Finally, microexons that had ME_P_values below the threshold, but they can also correspond to alternative splicing acceptors or donors (as the microexon sequence matches either at the begining or the end of an intron) are reported at `out.ambiguous.txt`.

On the other hand, microexon quantification is provided as `out_filtered_ME.PSI.txt` file. This file contains the following information:

Table 2: **out_filtered_ME.PSI.txt**

Column	Description
File	Sample name
ME_coords	Microexon Coordinates
SJ_coords	Splice junctions where the micrexon was detected in
ME_coverages	Coma-separtated values corresponding to microexon converage on each splice junction
SJ_coverages	Coma-separtated values corresponding to converage on each splice junction (exon skiping)
ME_coverages	Computed PSI values for a given microexon on a given sample
PSI	Lower bound of the PSI confidence interval
CI_Lo	Lower bound of the PSI confidence interval
CI_Hi	Upper bound of the PSI confidence interval
Alt5	Alternative donor coordinate
Alt3	Alternative aceptor coordinate
Alt5_coverages	Alternative donor coverage
Alt3_coverages	Alternative donor coverage

CHAPTER 4

Differential inclusion analysis

On this secction we descrive the a downstream module that was developed to perform alternative splicing analysis between sample groups. To quantify and assess differential inclusion of novel and annotated microexons, on this moudle we have integrated Whippet, which enables a fast and accurate assesment of alterntive splicing events across user-defined sample groups.

4.1 Install

To run this downstream module for the first time you need to create a environment that has *snakemake* and the version of *julia* that is compatible with *Whipet v0.11*. To creat this enviroment execute the following command inside `MicroExonator/` folder:

```
conda env create -f Whippet/julia_0.6.1.yaml
```

Then, activate the newly created enviroment:

```
source activate julia_0.6.1
```

Enter julia's interactive mode:

```
julia
```

Install Whippet by excecuting the following command on the interactive session:

```
Pkg.add("Whippet")
```

Note: To exit julia interactive session press `control + d`.

4.2 Configure

Here there is an list of the additonal keys that need to be incorporated as a part of config.yaml:

```
whippet_bin_folder : /path/to/miniconda/envs/julia_0.6.1/share/julia/site/v0.6/
↪Whippet/bin
Gene_anontation_GTF : /path/to/gene.annotation.gtf
whippet_delta : /path/to/whippet_delta.yaml
```

- `whippet_bin_folder` correspodn t the path of whippet binary folder (`Whippet/bin`) that is located inside `julia_0.6.1` virtual enviroment folder. The specific routh to `Whippet/bin` may variate, so it is important that you manually identify the correct path.
- `Gene_anontation_GTF` corresponds the path of a gene annotation file as Gene Transfer Format (GTF). Working with the same annotation data base than the one used on the previous steps is recommended.
- `whippet_delta` indicate the path of a YAML file you need to create to provide information about the desired comparisons between groups of samples.

4.2.1 whippet_delta YAML file

This file can contain the information to schedule any number of comparison between sample groups of any size. Every comparison should have the following structure inside the YAML file:

```
comparison_ID:
  A : sample1,sample2,sample3
  B : sample4,sample5,sample6
```

Where `sample1 ... sample6` correspond to base names given to each RNA-seq samples at the corresponding input files (See *Setup*) and *comparison_ID* to any given name for the sheduled comparison. As an example see the `YAML file` we used in our publication.

Warning: Inside this YAML file sample groups must be named `A` and `B`.

4.2.2 Optional parameters

If you just want to skip Discovery and Quantification modules and just asses alternative splicing events annotated at the provided GTF file, then include the following like at the configuratio file:

```
downstream_only : T
```

4.3 Run

In order to run this module you need to run the standar MicroExonator command, but providing `differential_inclusion` as a target. If you have not run previous `discovery` and `quantification` modules, MicroExonator will include them into the job plan (unless `downstream_only` is set as `T`)

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs} differential_inclusion
```

4.4 Output

Quantification files generated per each sample can be found at `Whipet/Quant`. Differentially included microexon analyses that can be obtained with Whippet, are reported at `Whippet/Delta` folder. MicroExonator performs these analyses using both PSI values calculated internally by the pipeline and PSI values directly calculated with Whippet. These results are reported under the same format than the `diff.gz` descrived at the Whippet's GitHub page. However, to provide easier interpretation, we filter the Whippet splicing nodes that correspond to microexon inclusion events, these are reported as `.microexons` files, where `.diff.ME.microexons` files correspond to the output when MicroExonator PSI values are taken as input and `.diff.microexons` when Whippet PSI values are taken as input.

CHAPTER 5

Single cell analysis

On this secction we describe how to perform analysis of single-cell RNA-seq data to quantify microexons across populations of cells and alternative splicing events across them. Since single cell experiments usually provides a shallow sequencing depth for each cell, we have developed a pseudo-pooling strategy to assess differential inclusion of microexons and other types of alternative splicing events across defined groups of cells (normally corresponding to cell-types previously defined by gene expresion profiles). We named this single cell analysis module `snakepool` an on this secction we describe how to use it.

Note: Before using this module you must follow the same installation instructions decrived in *Differential inclusion analysis* secction.

5.1 Configuration

To run this secction the following parameters needs to be incorporated at `config.yaml`.

```
Single_Cell : T
cluster_metadata : /lustre/scratch117/cellgen/team218/gp7/Micro-exons/Runs/Paper/
↪MicroExonator/Whippet/Tasic_clustering.txt
cluster_name : broad_type
file_basename : Run_s
cdf_t : 0.8
min_p_mean : 0.9
min_delta : 0.1
min_rep : 25
run_metadata : /lustre/scratch117/cellgen/team218/gp7/Micro-exons/Runs/Paper/
↪MicroExonator/Whippet/Tasic_run.txt
```

- `Single_Cell` correspond to an optiona parameter that needs to be set as `T` in order to run `snakepool`.
- `cluster_metadata` must indicate the path of a tabular separated file that contain at least two colums to indicate the cluster and file base names. This file must have the first row as header.

- `cluster_name` indicate the name of the column, inside `cluster_metadata`, which has cluster name information.
- `file_baseame` indicate the name of the column, inside `cluster_metadata`, which has the sample names. These needs to match with sample names defined on the input files (See *Setup*)
- `cdf_t` parameter set a theshold to run a Cumulative distribution function over the resultant Probability values obtained for each node across comutational replicates, asuming these fit a beta distribution. Which in practical terms can be considered as a user-defined threshold (between 0.5 and 1) to calculate a p-value assosiated to node's probability of differential inclusion across computational replicates.
- `min_p_mean` corresponds to a threshold of mean probability values across computational samples to define a node as differentially included across the comparing cell-types.
- `min_p_delta` corresponds to a threshold of mean delta PSI values across computational samples to define a node as differentially included across the comparing cell-types.
- `min_rep` minimun set of computational replicates that can be considered to define a node as differentially included across the comparing cell-types. This parameter is relevant because when nodes have limmited read coverage across cells, only some few computational replictates could enable quantitative alternative splicing analyses, leading to unreliable assesment of its alternative inclusion. The number of compuational repeats is defined for each sample inside *run_metadata*. We recomend to set this value to at least the half of the compuational replicates that are scheduled to run by the user.
- `run_metadata` indicates the path of a tabulat separated file which contain information about user-defined comparisons across cell-types. Additional information about this file can be found bellow.

Warning: If `Single_Cell` is set to `T` all the other parameters listed above will become compulsory. Thus, you should only activate this module if you have the required parameters on `config.yaml`.

5.1.1 run_metadata

The file indicated by `run_metadata` must be a tabular separated file containing the following columns:

Table 1: **run_metadata.tsv**

Column	Description
Compare_ID	User-defined name for scheduled comparions
A.cluster_names	Comma-separated list of cell-types to be concider as part of *sample group A*
A.number_of_pools	Number of pseudo-bulk to be generated for *sample group A*
B.cluster_names	Comma-separated list of cell-types to be concider as part of *sample group B*
B.number_of_pools	Number of pseudo-bulk to be generated for *sample group B*
Repeat	Node type. For more information visit Whippet's GitHub page.

Warning: The order of the columns is not relevant, however the column names must correspond to the ones indicated above. Additional columns with other names will be ignored.

Note: We recomend to consider `A.number_of_pools` and `B.number_of_pools` values that ensure that at least five cells are merged within each pseudo-bulk pool. We also suggest 10 as a minimun value of `Repeat`, higher values will enable better estimation of parameters to fit the resultant probabilities into beta distribution models.

5.1.2 Optional Configuration

The following parameter are optionals to be defined inside `config.yaml` file:

```
snakepool_seed : 123
Only_snakepool : T
Get_Bamfiles : T
```

- `snakepool_seed` define a specific seed for pseudo number geration. This number influence the arrangement cells into the corresponding pseudo-bulks. Mataining the same seed ensures reproducibility of the results and prevent snakemake of overwrite completed results.
- `Only_snakepool` is a bolean variable that if its defined as `T` it will force MicroExonator to skip Disovery and Quantification modules. This mode is useful for users who are only interested to find alterantive splicing events from splicing nodes that can be extracted from the annotation.
- `Get_Bamfiles` correspond to a bolean variable that if its defined as `T` enable the generation of BAM files that can be used for visualization purposes.

5.2 Run

After setting up all the files described above, this single cell analysis module can be run by adding *snakepool* as target for snakemake:

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs} snakepool
```

Note: It is allways a good idea to use `-np` to execute an snakemake `dry-run` before submiting a large set of jobs.

5.2.1 Unpooled quantification (optional)

In order to generate PSI quantification files at the single cell level (as opposed to pseudo-bulks), you can run MicroExonator with `quant_unpool_single_cell` as a target for snakemake. By doing this *.psi.gz* files will be generated at `Whippet/Quant/Single_Cell/Unpooled/` folder:

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs} quant_unpool_single_cell
```

This can enable users do run custom downstream analysis over alternative splicing quantification files generated for every cell by separated.

Warning: Only FASTQ files from cells annotated on `cluster_metadata` file will be processed.

5.3 Output

Direct results from *whippet-delta* for every comparison across each computational replicate can be found at *Whippet/Delta/Single_Cell/*. Integrated results for each comparion can be found at `Whippet/Delta/Single_Cell/Sig_nodes`, these resuls are structured as follow:

Table 2: **all_nodes.microexons.txt**

Column	Description
Gene	Gene ID
Node	Node number inside the gene
Coord	Node coordinate
Strand	Plus or minus strand
Type	Node type. For more information visit Whippet's GitHub page.
Psi_A.mean	Mean PSI values for group `cluster A` across computational replicates.
Psi_B.mean	Mean PSI values for group `cluster B` across computational replicates.
DeltaPsi.mean	Mean DeltaPsi values obtained across computationa replicates.
DeltaPsi.sd	Standar deviation of DeltaPsi values obtained across computationa replicates.
Probability.mean	Mean probability of differential inclusion obtained across computationa replicates.
Probability.var	Variance of probability across computationa replicates.
N.detected.reps	Number of replicates in which the differential inclusion could be assessed.
cdf.beta	p-value of being above the used defined probability threshold `cdf_t`
is.diff	Bolean variable defining wheather the node was differentially included accoding to the user-defined criteria (`min_rep`, `min_p_mean` and `min_delta`)
microexon_ID	Microexon ID based on its genomic coodinates.

5.4 Visualization

In order to visualize the results, you need to instruct `whippet-quant` to generate SAM files by incorporating the following parameter with a `True` value inside `config.yaml`:

```
Get_Bamfiles : T
```

Samfiles are further converted to BAM files and corresponding index files are genrated to enable their visualization. To generate these BAMs `cluster_bams` needs to be defined as an snakemake target:

```
snakemake -s MicroExonator.skm  --cluster-config cluster.json --cluster {cluster␣
↪system params} --use-conda -k  -j {number of parallel jobs} cluster_bams
```

A BAM file will be generated for every cell-type defined at `cluster_metadata` file. Given the coodinates of differentially included splicing nodes and the correspondig BAM files, sashimi plots can be generated by using tools such as ggsashimi or IGV.

CHAPTER 6

MIT License (MIT)

CHAPTER 7

Support

For questions, ideas, feature requests and potential bug reports please contact gp7@sanger.ac.uk.