htseq-clip

Apr 25, 2023

Contents:

htseq-clip is a toolset for the analysis of eCLIP/iCLIP datasets. This python package can be used to generate files necessary for data analysis using the companion R/Bioconductor package DEWSeq

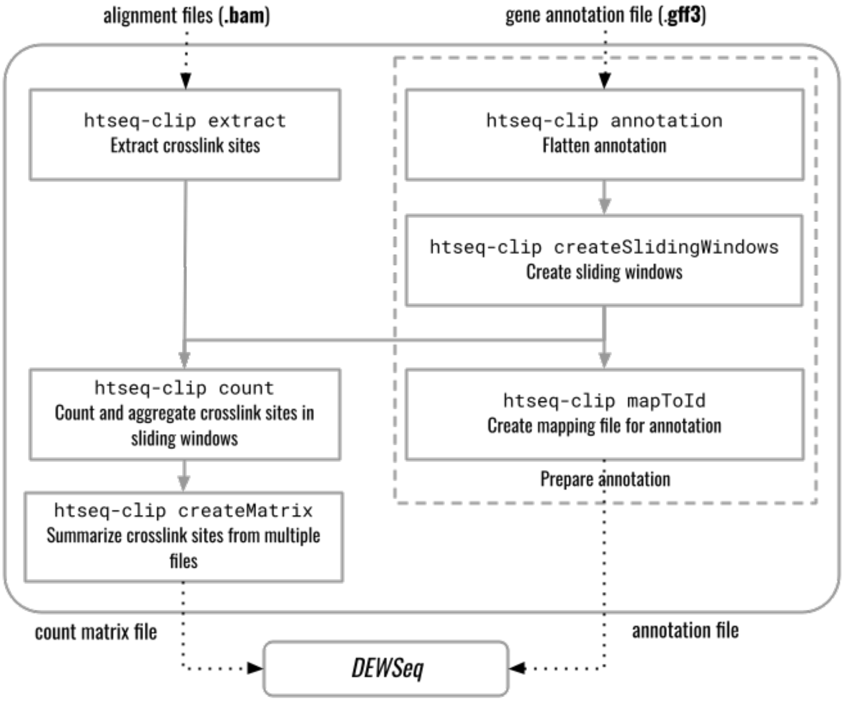


Fig. 1: htseq-clip data flow diagram

CHAPTER 1

overview

htseq-clip

htseq-clip is a toolset designed for the processing and analysis of eCLIP/iCLIP dataset. This package is designed primarily to do the following operations:

1.1 Prepare annotation

A suite of functions to process and flatten genome annotation file.

annotation

annotation function takes as input a GFF formatted genome annotation file and converts the annotations from GFF format to bed format. For an example, this function converts the following GFF annotation

chr1	HAVANA	exon	1373730	1373902		-	.	ID=exon:ENST00000338338.9:4;Parent=ENST00000338338.9;gene_id=ENSG0000017 202;exon_number=4;exon_id=ENSE00001611509.1;level=2;protein_id=ENSP0000034

and converts this entry into the following BED6 format

chromosome	start	end	name	score	strand
chr1	1373729	1373902	ENSG00000175756.13@AURKAIP1@protein_coding@exon@2/2@ENSG00000175756.13:exon000	0	-

Various attributes in the name column in this BED entry is seperated by `@` and the order is given below

atrribute	attribute description
ENSG00000175756.13	gene id
AURKAIP1	gene name
protein_coding	gene type
exon	gene feature (exon, intron, CDS,. . .)
2/4	2nd exon out of a total of 2 exons of this gene
ENSG00000175756.13:exon0002	unique id, merging gene id feature and feature number

`score` column in the BED file is re-purposed to indicate a `flag` which can be used as a measure of trust worthiness/ as a filter option for further analysis.

Flag can have the following different values:

Flag	description	trust worthiness
3	only one variant of start/end positions	high
2	same start position but different end positions	medium
1	different start positions but same end position	medium
0	different start and end positions	low

An exon from a gene can belong to multiple isoforms and therefore can have different start/end positions. `htseq-clip` combines all the position informations for each exon to one and takes the lowest/highest value as start/end position. As it is shown in the cartoon below, the first exon belongs to 3 different isoforms, so the Flag is *0‘* (**trust worthiness: low**) as the start and end positions varies. The second exon belongs to two different isoforms, but there is only one unique start and one unique end postion, hence the Flag is `3` (**trust worthiness: high**)

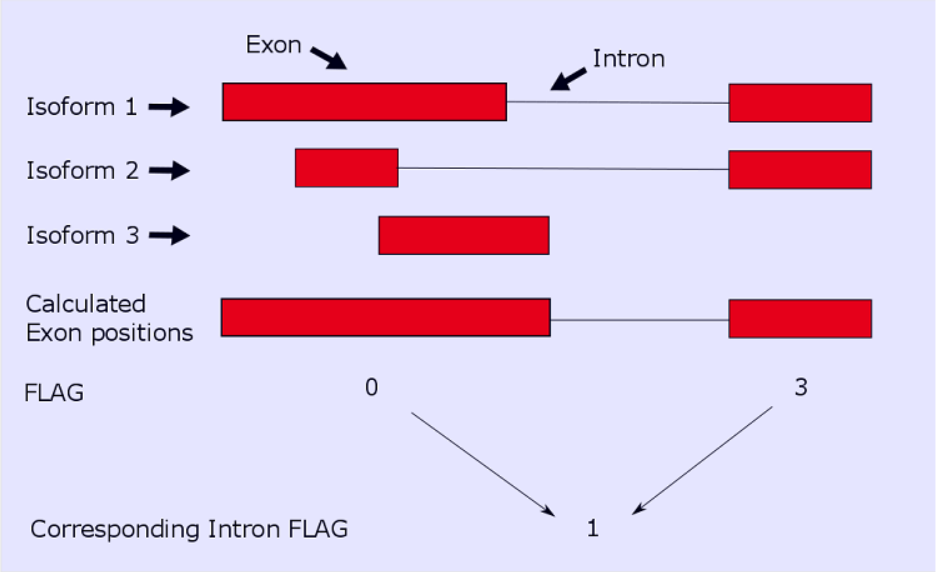
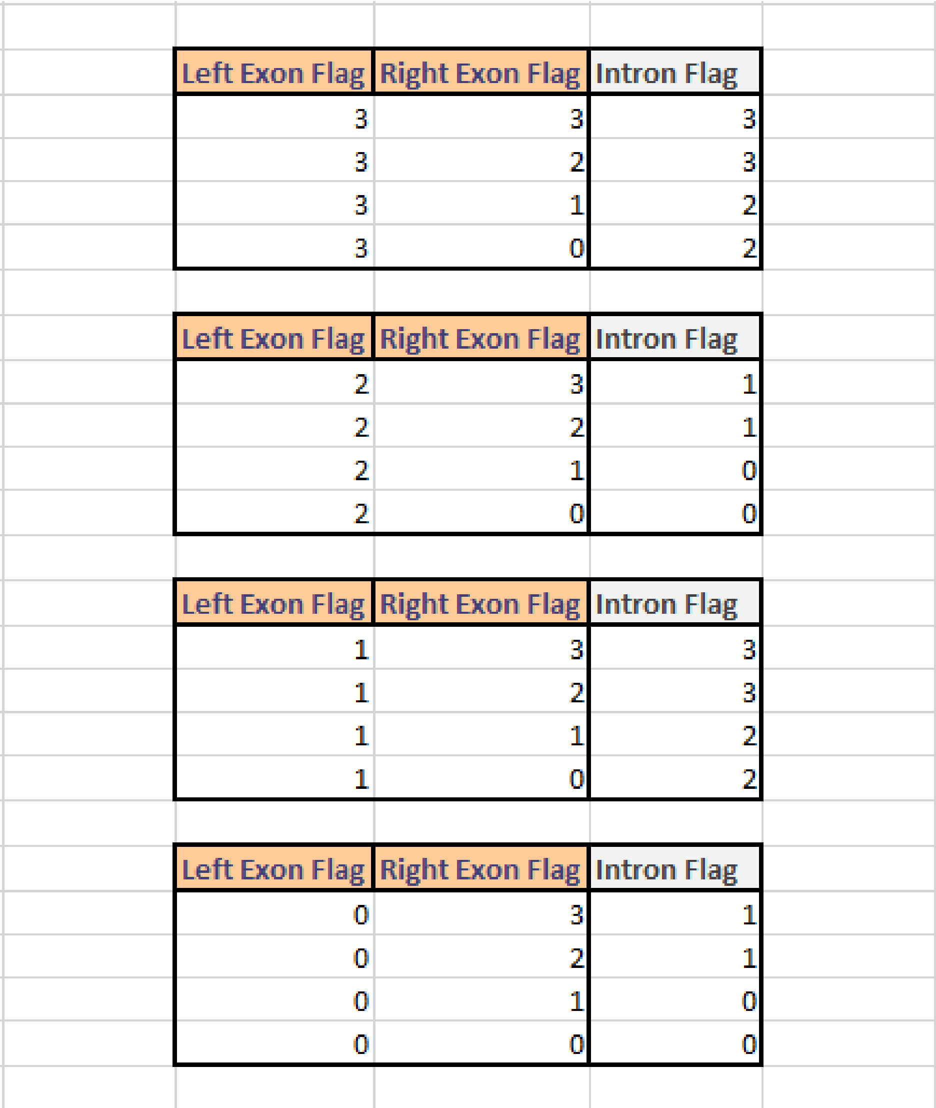


Fig. 1: Cartoon showing flag generation process

The corresponding intron Flag is calculated as follows: if the left exon Flag is 0 and the right exon Flag is 3 the intron Flag is 1 : because for the start position(s) can exist different variants, but for the end position(s) there exist only one variant. The intron flag is calculated depending on the 2 exon flags where the intron is between. Given below is a table to lookup which variations of exon flags yield to the corresponding intron flag.



Left Exon Flag	Right Exon Flag	Intron Flag
3	3	3
3	2	3
3	1	2
3	0	2

Left Exon Flag	Right Exon Flag	Intron Flag
2	3	1
2	2	1
2	1	0
2	0	0

Left Exon Flag	Right Exon Flag	Intron Flag
1	3	3
1	2	3
1	1	2
1	0	2

Left Exon Flag	Right Exon Flag	Intron Flag
0	3	1
0	2	1
0	1	0
0	0	0

Fig. 2: Intron Flag lookup table

createSlidingWindows

createSlidingWindows function takes as input a flattened annotation BED file created by the annotation function and splits each individual BED entries into overlapping windows. `--windowSize` parameter controls the size of each window and `--windowStep` controls the overlap of each neighboring windows from the same feature

Continuing with the example entry above, the first 5 sliding windows generated from the *BED6 flattened entry* are given below:

chromosome	start	end	name	score	strand
chr1	1373729	1373779	ENSG00000175756.13@AURKAIP1@protein_coding@exon@2/2@ENSG00000175756.13:exon0002	0	-
chr1	1373749	1373799	ENSG00000175756.13@AURKAIP1@protein_coding@exon@2/2@ENSG00000175756.13:exon0002	0	-
chr1	1373769	1373819	ENSG00000175756.13@AURKAIP1@protein_coding@exon@2/2@ENSG00000175756.13:exon0002	0	-
chr1	1373789	1373839	ENSG00000175756.13@AURKAIP1@protein_coding@exon@2/2@ENSG00000175756.13:exon0002	0	-
chr1	1373809	1373859	ENSG00000175756.13@AURKAIP1@protein_coding@exon@2/2@ENSG00000175756.13:exon0002	0	-

Each sliding window listed here is 50bp long, as default value for `--windowSize` argument is `50` and the difference between start positions of each is 20bp, as the default value for `--windowStep` argument is `20`

Following the convention in *flattened annotation* the attributes in sliding windows name column are also seperated by `@` and the first 5 attributes in the name column here are exactly the same as that of *flattened annotation name column* An example is given below

atrribute	attribute description	Found in *flattend name attribute*
ENSG00000175756.13	gene id	Yes
AURKAIP1	gene name	Yes
protein_coding	gene type	Yes
exon	gene feature (exon, intron, CDS,. . .)	Yes
2/2	2nd exon out of a total of 2 exons of this gene	Yes
ENSG00000175756.13:exon0002W0001	unique id, merging gene id feature, feature number and window number (W : window)	No
1	1st window of this feature	No

Note: There will be zero overlap between neighboring windows from two separate gene features

1.2 Extract crosslink sites

Extract and process crosslink sites from alignment file.

extract

extract function takes as input an alignment file (.bam) and extracts and writes either start, insertion, deletion, middle or end site into a BED6 formatted file. The argument `--site` determines crosslink site choice.

Given below is an example paired end sequence and start, middle and end positions extracted from the second mate of this fragment

TTATTACAGC:K00180:131:H7J3YBBXX:3:2123:15057:19918	99	chr1	1373726	255	33M	=	1373729	41	TTTTAAAGGCTGAGTCCTCTGAGAATTTATTAC	[illegible]	NH:i:1	HI:i:1	AS:i:60	nM:i:5	NM:i:4	MD:Z:0C0A0G0C29	jM:B:c,-1	jI:B:i,-1	RG:Z:foo
TTATTACAGC:K00180:131:H7J3YBBXX:3:2123:15057:19918	147	chr1	1373729	255	38M	=	1373726	-41	TAAAGGCTGAGTCCTCTGAGAATTTATTACTACGGATC	[illegible]	NH:i:1	HI:i:1	AS:i:60	nM:i:5	NM:i:1	MD:Z:0G37	jM:B:c,-1	jI:B:i,-1	RG:Z:foo

start site

chromosome	start	end	name	score	strand
chr1	1373765	1373766	TTATTACAGC:K00180:131:H7J3YBBXX:3:2123:15057:19918	1	38	-

middle site

chromosome	start	end	name	score	strand
chr1	1373746	1373747	TTATTACAGC:K00180:131:H7J3YBBXX:3:2123:15057:19918	1	38	-

end site

chromosome	start	end	name	score	strand
chr1	1373727	1373728	TTATTACAGC:K00180:131:H7J3YBBXX:3:2123:15057:19918	1	38	-

Note: In a paired end alignment file, argument `--mate` is used to choose the read/mate from which crosslink sites are extracted. The sequencing protocol used to generate the file determines whether the crosslink site is located on the first mate or the second mate. Please consult your sequencing protocol to decide which mate to use.

1.3 Count crosslink sites

Calculate the number of extracted crosslink sites per given gene annotation feature.

count

count function takes as input either a flattened annotation file generated by annotation function or a sliding windows file generated by createSlidingWindows function and a crosslink sites file generated by extract function and for each entry/window in the annotation/sliding windows file count the number of crosslink sites in the region.

Given below is an example output entries from count function for sliding windows in *createSlidingWindows example*.

unique_id	window_number	window_length	crosslink_count_total	crosslink_count_position_nr	crosslink_count_position_max	crosslink_density
ENSG00000175756.13:exon0002W00001	1	50	4	3	2	0.06
ENSG00000175756.13:exon0002W00002	2	50	17	12	3	0.24
ENSG00000175756.13:exon0002W00003	3	50	159	25	76	0.5
ENSG00000175756.13:exon0002W00004	4	50	207	26	76	0.52
ENSG00000175756.13:exon0002W00005	5	50	183	21	76	0.42

Here is a brief explanation of the columns in the table above

column heading	description
unique_id	unique id of the entry, as described in *sliding window attribute table*
window_number	window number, as described in *sliding window attribute table*
window_length	total length of this window (in bp)
crosslink_count_total	total number of crosslink sites
crosslink_count_position_nr	number of positions with crosslink sites in this window
crosslink_count_position_max	maximum number of crosslink sites found at a single position
crosslink_density	calculated as: $\frac{crosslink\_count\_position\_nr}{window\_length}$

Note: Please refer to *createMatrix function* for merging count tables from multiple samples.

1.4 Further analysis

Further analysis and processing of crosslink windows is done using R/Bioconductor package DEWSeq. Please refer to the user manual of this package for requirements, installation and help.

CHAPTER 2

requirements and installation

2.1 requirements

- Python >= 3.5
- HTSeq package

Note: HTSeq uses pysam package for processing alignment files. Please consult HTSeq manual and pysam manual for requirements of both packages.

2.2 installation

2.2.1 quick installation

If a user has a local python environment with all the dependencies for HTSeq and pysam installed, then htseq-clip can be installed as:

```
$ pip install htseq-clip
```

2.2.2 conda environment

We strongly encourage the use of conda package management system for multiple Python versions/various incompatible package installations. Please install conda on your computer following the guidelines. Once conda installation is successful, create a new `htseq-clip` enviroment as:

```
(base) $ conda create -n htseq-clip
```

and activate the environment:

```
(base) $ conda activate htseq-clip
```

now install the dependencies:

```
(htseq-clip) $ conda install -c bioconda pysam
....
(htseq-clip) $ conda install -c bioconda htseq
```

now htseq-clip can be installed in this environment as:

```
(htseq-clip) $ pip install htseq-clip
```

CHAPTER 3

documentation

After successful installation of the package use

```
$ htseq-clip -h
```

for a brief description of the functions available in htseq-clip. The available functions can be categorized into 4 different classes given below.

3.1 Prepare annotation

3.1.1 annotation

Flattens a given annotation file in GFF format to BED6 format

Arguments

- `-g/--gff` GFF formatted annotation file, supports .gz files
- `-u/--geneid` Gene id attribute in GFF file (default: gene_id)
- `-n/--genename` Gene name attribute in GFF file (default: gene_name)
- `-t/--genetype` Gene type attribute in GFF file (default: gene_type)
- `--splitExons` This flag splits exons into components such as 5' UTR, CDS and 3' UTR
- `--unsorted` Use this flag if the GFF file is unsorted
- `-o/--output` Output file name. If the file name is given with .gz suffix, it is gzipped. If no file name is given, output is print to console

Note: The default values for `--geneid`, `--genename` and `--genetype` arguments follow gencode GFF format

Usage

```
$ htseq-clip annotation -h
```

3.1.2 createSlidingWindows

Create sliding windows from the flattened annotation file

Arguments

- `-i/--input` Flattened annoation file, see *annotation*
- `-w/--windowSize` Window size in number of base pairs for the sliding window (default: 50)
- `-s/--windowStep` Window step size for sliding window (default: 20)
- `-o/--output` Output file name. If the file name is given with .gz suffix, it is gzipped. If no file name is given, output is print to console

Usage

```
$ htseq-clip createSlidingWindows -h
```

3.1.3 mapToId

Extract "name" column from the annotation file and map the entries to unique id and print out in tab separated format

Arguments

- `-a/--annotation` Flattened annotation file from *annotation* or sliding window file from *createSlidingWindows*
- `-o/--output` Output file name. If the file name is given with .gz suffix, it is gzipped. If no file name is given, output is print to console

Usage

```
$ htseq-clip mapToId -h
```

3.2 Extract crosslink sites

3.2.1 extract

Extract crosslink sites, insertions or deletions

Arguments

- `-i/--input` Input .bam file. Input bam file must be co-ordinate sorted and indexed
- `-e/--mate` for paired end sequencing, select the read/mate to extract the crosslink sites from, accepted choices: `1, 2`
  - `1` use the first mate in pair
  - `2` use the second mate in pair
- `-s/--site` Crosslink site choices, accepted choices: `s, i, d, m, e` (default: e)

  - s startsite,
  - i insertion site
  - d deletion site
  - m middle site
  - e end site

- -g/--offset Number of nucleotides to offset for crosslink sites (default: 0)
- --ignore Use this flag to ignore crosslink sites outside of genome annotations
- -q/--minAlignmentQuality Minimum alignment quality (default: 10)
- -m/--minReadLength Minimum read length (default: 0)
- -x/--maxReadLength Maximum read length (default: 500)
- -l/--maxReadInterval Maximum read interval length (default: 10000)
- --primary Use this flag consider only primary alignments of multimapped reads
- -c/--cores Number of cores to use for alignment parsing (default: 5)
- -t/--tmp Path to create and store temp files (default behavior: use parent folder from "–output" parameter)
- -o/--output Output file name. If the file name is given with .gz suffix, it is gzipped. If no file name is given, output is print to console

Usage

```
$ htseq-clip extract -h
```

Note: To extract 1``st offset position of second mate (``2) start site (s) in eCLIP, use: --mate 2 --site s --offset -1

3.3 Count crosslink sites

3.3.1 count

Counts the number of crosslink/deletion/insertion sites

Arguments

- -i/--input Extracted crosslink sites, see *extract*
- -a/--ann Flattened annotation file, see *annotation* OR sliding windows file, see *createSlidingWindows*
- --unstranded crosslink site counting is strand specific by default. Use this flag for non strand specific crosslink site counting
- -o/--output Output file name. If the file name is given with .gz suffix, it is gzipped. If no file name is given, output is print to console

Usage

```
$ htseq-clip count -h
```

3.4 Helper functions

3.4.1 createMatrix

Create R friendly output matrix file from count function output files

Arguments

- `-i/--inputFolder` Folder name with output files from count function, see *count*
- `-b/--prefix` Use files only with this given file name prefix (default: None)
- `-e/--postfix` Use files only with this given file name postfix (default: None)
- `-o/--output` Output file name. If the file name is given with .gz suffix, it is gzipped. If no file name is given, output is print to console

Warning: either `--prefix` or `--postfix` argument must be given

Usage

```
$ htseq-clip createMatrix -h
```

3.4.2 createMaxCountMatrix

Create R friendly output matrix file from `crosslink_count_position_max` column in *count* function output files. This file can be used to filter down the output file from `createMatrix` function during downstream statistical analysis.

Arguments

- `-i/--inputFolder` Folder name with output files from count function, see *count*
- `-b/--prefix` Use files only with this given file name prefix (default: None)
- `-e/--postfix` Use files only with this given file name postfix (default: None)
- `-o/--output` Output file name. If the file name is given with .gz suffix, it is gzipped. If no file name is given, output is print to console

Warning: either `--prefix` or `--postfix` argument must be given

Usage

```
$ htseq-clip createMatrix -h
```

CHAPTER 4

references

Simon Anders, Paul Theodor Pyl, Wolfgang Huber *HTSeq — A Python framework to work with high-throughput sequencing data* Bioinformatics (2014), in print, online at doi:10.1093/bioinformatics/btu638

Matthias Hentze and Wolfgang Huber Labs, EMBL Heidelberg.

We would also like to acknowledge contributions from Marko Fritz and Nadia Ashraf in developing this package.