PureCLIP Documentation

Release v1.0.0

Sabrina Krakau

Aug 10, 2021

Tutorials

1 Getting Started **3**
- 1.1 Requirements . . . 3
- 1.2 Preprocessing iCLIP/eCLIP-seq reads . . . 3

2 PureCLIP Tutorial **7**
- 2.1 Basic mode . . . 7
- 2.2 Incorporating input control data . . . 8
- 2.3 Incorporating CL-motif scores . . . 8
- 2.4 PureCLIPs output . . . 9
- 2.5 General user options . . . 10
- 2.6 Trouble Shooting - FAQs . . . 11

3 Simulation Framework **13**

4 Citation **15**

5 License **17**

6 Contact **19**

PureCLIP is a tool to detect protein-RNA interaction footprints from single-nucleotide CLIP-seq data, such as iCLIP and eCLIP.

CHAPTER 1

Getting Started

Before you start, please check the

1.1 Requirements

1.1.1 For PureCLIP in basic mode

- C++14 compliant compiler
- GSL
- cmake 3.0 or newer

1.1.2 For computing CL-motif scores

If you want to run PureCLIP incorporating CL-motifs, you need to install the MEME Suite, which contains FIMO and DREME, in order to enable the computation of position-wise CL-motif scores:

- MEME Suite (tested for v4.11.3)

For the analysis of your own data, please have a look how to:

1.2 Preprocessing iCLIP/eCLIP-seq reads

This tutorial will walk you through an example how to preprocess your iCLIP/eCLIP data before analysing it. This only serves as a guide and you can of course as well use your own CLIP-seq preprocessing workflow.

1.2.1 Requirements for this tutorial

The tutorial requires the following tools (and was tested for the specified versions):

- cutadapt (v1.12)
- samtools (v0.1.19-44428cd)
- STAR (v2.5.1b)
- UMI (v0.4.3)
- fastqc (v0.11.5)

1.2.2 Data retrieval

Let's start with downloading the data. We use the raw PUM2 eCLIP sequencing data from ENCODE (Van Nostrand et. al, 2016):

```
mkdir rep1 rep2
wget -O rep1/reads.R1.fastq.gz https://www.encodeproject.org/files/ENCFF956TOZ/
↪@@download/ENCFF956TOZ.fastq.gz
wget -O rep1/reads.R2.fastq.gz https://www.encodeproject.org/files/ENCFF133DNM/
↪@@download/ENCFF133DNM.fastq.gz
wget -O rep2/reads.R1.fastq.gz https://www.encodeproject.org/files/ENCFF041KJT/
↪@@download/ENCFF041KJT.fastq.gz
wget -O rep2/reads.R2.fastq.gz https://www.encodeproject.org/files/ENCFF462SCV/
↪@@download/ENCFF462SCV.fastq.gz
```

We download the human reference genome (primary assembly containing chromosomes and scaffolds) and the genome annotation (comprehensive set) from GENCODE:

```
wget -O ref.GRCh38.fa.gz ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_human/release_26/
↪GRCh38.primary_assembly.genome.fa.gz
gunzip ref.GRCh38.fa.gz

wget -O annotation.v26.gtf.gz ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_human/
↪release_26/gencode.v26.primary_assembly.annotation.gtf.gz
gunzip annotation.v26.gtf.gz
```

Please note, that the following steps need to be done for both replicates, while only being described for rep1.

1.2.3 Adaptor trimming

Possible adapter contaminations at 3' ends can be removed using the tool cutadapt (Martin, 2011) (as done by the YEO lab in the ENCODE processing pipeline), while discarding reads shorter than 18bp:

```
cutadapt -f fastq --match-read-wildcards --times 1 -e 0.1 -O 1 --quality-cutoff 6 -m␣
↪18 -a NNNNNAGATCGGAAGAGCACACGTCTGAACTCCAGTCAC -g CTTCCGATCTACAAGTT  -g␣
↪CTTCCGATCTTGGTCCT -A AACTTGTAGATCGGA -A AGGACCAAGATCGGA -A ACTTGTAGATCGGAA -A␣
↪GGACCAAGATCGGAA  -A CTTGTAGATCGGAAG  -A GACCAAGATCGGAAG -A TTGTAGATCGGAAGA -A␣
↪ACCAAGATCGGAAGA -A TGTAGATCGGAAGAG -A CCAAGATCGGAAGAG -A GTAGATCGGAAGAGC -A␣
↪CAAGATCGGAAGAGC -A TAGATCGGAAGAGCG  -A AAGATCGGAAGAGCG -A AGATCGGAAGAGCGT  -A␣
↪GATCGGAAGAGCGTC -A ATCGGAAGAGCGTCG -A TCGGAAGAGCGTCGT -A CGGAAGAGCGTCGTG -A␣
↪GGAAGAGCGTCGTGT -o rep1/reads.R1.trimmed.fastq.gz -p rep1/reads.R2.trimmed.fastq.gz␣
↪rep1/reads.R1.fastq.gz rep1/reads.R2.fastq.gz
```

For eCLIP data, it was suggested (Van Nostrand et. al, 2016) to apply two rounds of adapter trimming to correct for possible double ligations events.

```
cutadapt -f fastq --match-read-wildcards --times 1 -e 0.1 -O 5 --quality-cutoff 6 -m␣
↪18 -A AACTTGTAGATCGGA -A AGGACCAAGATCGGA -A ACTTGTAGATCGGAA -A GGACCAAGATCGGAA -A␣
↪CTTGTAGATCGGAAG -A GACCAAGATCGGAAG -A TTGTAGATCGGAAGA -A ACCAAGATCGGAAGA -A␣
↪TGTAGATCGGAAGAG -A CCAAGATCGGAAGAG -A GTAGATCGGAAGAGC -A CAAGATCGGAAGAGC -A␣
↪TAGATCGGAAGAGCG -A AAGATCGGAAGAGCG -A AGATCGGAAGAGCGT -A GATCGGAAGAGCGTC -A␣
↪ATCGGAAGAGCGTCG -A TCGGAAGAGCGTCGT -A CGGAAGAGCGTCGTG -A GGAAGAGCGTCGTGT -o rep1/
↪reads.R1.trimmed2.fastq.gz -p rep1/reads.R2.trimmed2.fastq.gz rep1/reads.R1.trimmed.
↪fastq.gz rep1/reads.R2.trimmed.fastq.gz
```

1.2.4 Preparing read IDs for UMI

We remove PCR duplicates based on the mapping position and random barcodes using UMI, which requires the read ID in the format `_@HISEQ:87:00000000_BARCODE read1_`. Therefore we append the barcode to the read ID prior the mapping, using the following command:

```
zcat rep1/reads.R1.trimmed2.fastq.gz > rep1/reads.R1.trimmed2.fastq
zcat rep1/reads.R2.trimmed2.fastq.gz > rep1/reads.R2.trimmed2.fastq
awk -v l=10 'BEGIN{OFS=FS=" "} substr($1, 1, 1) == "@" {print "@" substr($1, (l+3),␣
↪500) "_" substr($1, 2, l) " " $2 }; substr($1, 1, 1) != "@" {print}; ' rep1/reads.
↪R1.trimmed2.fastq  | gzip > rep1/reads.R1.trimmed2.bc.fastq.gz
awk -v l=10 'BEGIN{OFS=FS=" "} substr($1, 1, 1) == "@" {print "@" substr($1, (l+3),␣
↪500) "_" substr($1, 2, l) " " $2 }; substr($1, 1, 1) != "@" {print}; ' rep1/reads.
↪R2.trimmed2.fastq  | gzip > rep1/reads.R2.trimmed2.bc.fastq.gz
```

where `l=10` denotes the used barcode length.

1.2.5 Read mapping with STAR

CLIP-seq reads can be mapped with the RNA-seq read aligner STAR (Dobin et. al, 2013). It allows to include genome annotations in order to enable the alignment against spliced transcripts. First, we need to prepare the genome index, using the annotation file:

```
mkdir genome_index
STAR --runThreadN 10 --runMode genomeGenerate --genomeDir genome_index/ --
↪genomeFastaFiles ref.GRCh38.fa --sjdbGTFfile annotation.v26.gtf --sjdbOverhang 49
```

Next, we map the reads (R1 and R2) against the indexed genome:

```
mkdir -p rep1/STAR
STAR --outSAMtype BAM SortedByCoordinate --runThreadN 10 --genomeDir genome_index/ --
↪readFilesIn rep1/reads.R1.trimmed2.bc.fastq.gz rep1/reads.R2.trimmed2.bc.fastq.gz --
↪readFilesCommand  zcat --outFilterType BySJout --outFilterMultimapNmax 1 --
↪alignSJoverhangMin 8 --alignSJDBoverhangMin 1 --outFilterMismatchNmax 999 --
↪outFilterMismatchNoverLmax 0.04 --scoreDelOpen -1 --alignIntronMin 20 --
↪alignIntronMax 1000000 --alignMatesGapMax 1000000 --outFileNamePrefix rep1/STAR/ --
↪alignEndsType EndToEnd
```

The parameter `--outFilterMultimapNmax 1` ensures only uniquely mapping reads will be reported. Since we used the primary assembly containing scaffolds as reference, this enables us to filter out reads that map both against a main chromosome and against a scaffold (e.g. ribosomal RNA). Furthermore, it is important to use the `--alignEndsType EndToEnd` setting, to ensure the mapping of the whole read. The aligned reads will be written then to STAR/Aligned.sortedByCoord.out.bam .

1.2.6 Filtering

We filter the aligned reads to obtain only reads mapping against the main chromosomes:

```
samtools index rep1/STAR/Aligned.sortedByCoord.out.bam
samtools view -hb -f 2 rep1/STAR/Aligned.sortedByCoord.out.bam -o rep1/aligned.f.bam␣
↪chr1:1 chr2:1 chr3:1 chr4:1 chr5:1 chr6:1 chr7:1 chr8:1 chr9:1 chr10:1 chr11:1␣
↪chr12:1 chr13:1 chr14:1 chr15:1 chr16:1 chr17:1 chr18:1 chr19:1 chr20:1 chr21:1␣
↪chr22:1 chrX:1 chrY:1
samtools index rep1/aligned.f.bam
```

1.2.7 PCR duplicate removal using UMI

For truncation based CLIP-seq data it is crucial to remove PCR duplicates to allow for an accurate crosslink site detection. We use the tool UMI (Smith et. al, 2017), which is able to handle errors within barcode sequences.

```
umi_tools dedup -I rep1/aligned.f.bam --paired -S rep1/aligned.f.duplRm.bam
```

1.2.8 Pooling and R2 retrieval

Finally, we merge the preprocessed alignments of the individual replicates:

```
samtools merge -f aligned.f.duplRm.pooled.bam rep1/aligned.f.duplRm.bam rep2/aligned.
↪f.duplRm.bam
```

and filter for R2, to keep only reads containing information about potential truncation events (for iCLIP data this would be R1):

```
samtools view -hb -f 130 aligned.f.duplRm.pooled.bam -o aligned.f.duplRm.pooled.R2.bam
samtools index aligned.f.duplRm.pooled.R2.bam
```

1.2.9 Quality control

It's always a good idea to assess the quality of the data prior to the actual analysis. For this we use fastqc:

```
mkdir fastqc
fastqc -o fastqc/ aligned.f.duplRm.pooled.R2.bam
```

CHAPTER 2

PureCLIP Tutorial

In the following tutorial we will describe for each PureCLIP mode how to run a minimal example.

Hint: Since PureCLIP includes information from the transcriptomic neighbourhood, it is important to use a suitable reference sequence when mapping the reads. Thus, when analysing CLIP data from proteins known to bind near exon-exon junctions on mRNAs, reads should be mapped directly against transcripts (e.g. as done by Haberman et al., 2017).

2.1 Basic mode

Generate sample files for minimal example: Show/Hide

As a first example you can download preprocessed data from ENCODE, and filter the paired-end data to keep only R2:

```
wget -O aligned.prepro.bam https://www.encodeproject.org/files/ENCFF280ONP/@@download/
↪ENCFF280ONP.bam
samtools view -hb -f 130 aligned.prepro.bam -o aligned.prepro.R2.bam
samtools index aligned.prepro.R2.bam
```

Additionally, we need the corresponding reference genome:

```
wget -O ref.hg19.fa.gz https://www.encodeproject.org/files/female.hg19/@@download/
↪female.hg19.fasta.gz
gunzip ref.hg19.fa.gz
```

2.1.1 PureCLIP

To run PureCLIP in basic mode, it requires BAM and BAI files, the reference genome and a specified output file:

```
pureclip -i aligned.prepro.R2.bam -bai aligned.prepro.R2.bam.bai -g ref.hg19.fa -iv
↪'chr1;chr2;chr3;' -nt 10 -o PureCLIP.crosslink_sites.bed
```

With `-iv` the chromosomes (or transcripts) can be specified that will be used to learn the parameters of PureCLIPs HMM. This reduces the memory consumption and runtime. Usually, learning on a small subset of the chromosomes, e.g. Chr1-3, does not impair the results noticeable. However, in the case of very sparse data this can be adjusted. With `-nt` the number threads for parallelization can be specified.

2.2 Incorporating input control data

Generate sample files for minimal example: Show/Hide

Beside the target PUM2 eCLIP data, we download the corresponding preprocessed input data from ENCODE, and again filter the paired-end data to keep only R2:

```
wget -O input.aligned.prepro.bam https://www.encodeproject.org/files/ENCFF043ERY/
↪@@download/ENCFF043ERY.bam
samtools view -hb -f 130 input.aligned.prepro.bam -o input.aligned.prepro.R2.bam
samtools index input.aligned.prepro.R2.bam
```

2.2.1 PureCLIP

To run PureCLIP with input control data, additionally hand over the (preprocessed) BAM file from the input experiment with `-ibam` and the associated BAI file with `-ibai`:

```
pureclip -i aligned.prepro.R2.bam -bai aligned.prepro.R2.bam.bai -g ref.hg19.fa -iv
↪'chr1;chr2;chr3;' -nt 10 -o PureCLIP.crosslink_sites.cov_inputSignal.bed -ibam␣
↪input.aligned.prepro.R2.bam -ibai input.aligned.prepro.R2.bam.bai
```

2.3 Incorporating CL-motif scores

In order to address the crosslinking sequence bias, PureCLIP can incorporate information about motifs that are known to be preferentially crosslinked, also called CL-motifs (see also Haberman et al., 2017). For each CL-motif PureCLIP learns the influence on the crosslinking probability. This can be particular useful for proteins binding to sequence motifs distinct from such CL-motifs.

2.3.1 Compute CL-motif scores

To incorporate CL-motifs into the model of PureCLIP, first we need to compute position-wise CL-motif scores, indicating the positions CL-affinity. You can use the provided precompiled list of common_CL-motifs:

```
wget -O motifs.txt https://raw.githubusercontent.com/skrakau/PureCLIP_data/master/
↪common_CL-motifs/dreme.w10.k4.txt
wget -O motifs.xml https://raw.githubusercontent.com/skrakau/PureCLIP_data/master/
↪common_CL-motifs/dreme.w10.k4.xml
```

If you want to compute the CL-motifs specific to the used eCLIP experiment, please have a look here. Assume we have given a set of CL-motifs, we use FIMO (Grant et. al, 2011) to compute motif occurrences associated with a score within the reference sequence. The following provided script (distributed with PureCLIP) first retrieves reference regions covered by the target experiment, then runs FIMO to compute position-wise CL-motif match scores within such regions and chooses for each position the motif with the highest score:

```
export BEDTOOLS=/path/to/bedtools        # if not specified, PATH is searched
export FIMO=/path/to/fimo                # if not specified, PATH is searched
export WINEXTRACT=/path/to/winextract    # built together with PureCLIP
compute_CLmotif_scores.sh ref.hg19.fa aligned.prepro.R2.bam motifs.xml motifs.txt␣
↪fimo_clmotif_occurences.bed
```

The resulting CL-motif scores are written to `fimo_clmotif_occurences.bed`.

2.3.2 PureCLIP

The computed scores are then handed over to PureCLIP together with the parameter `-nim 4`, indicating that scores with associated motif IDs 1-4 will be used (default: only scores with motif ID 1 are used).

```
pureclip -i aligned.prepro.R2.bam -bai aligned.prepro.R2.bam.bai -g ref.hg19.
↪fa -o PureCLIP.crosslink_sites.cov_CLmotifs.bed -nt 10 -iv 'chr1;chr2;chr3;
↪' -nim 4 -fis fimo_clmotif_occurences.bed
```

2.4 PureCLIPs output

2.4.1 Crosslink sites

The main output of PureCLIP is a BED6 file containing individual crosslink sites associated with a score:

1. **chr:** Name of the chromosome or scaffold.
2. **start:** Position of crosslink site.
3. **end:** Position behind crosslink site (start+1).
4. **state:** '3' (0-based, corresponds to state '4' in the PureCLIP publication)
5. **score:** log posterior probability ratio of the first and second likely state.
6. **strand:** + or -

2.4.2 Binding regions

Optionally, if an output file for binding regions is specified with `-or`, individual crosslink sites with a distance <= d (specified with `-dm`, default 8 bp) are merged and given out in a separate BED6 file:

1. **chr:** Name of the chromosome or scaffold.
2. **start:** Start position, position of first crosslink site.
3. **end:** End position, position behind last crosslink site.
4. **indiv. scores:** 'score1;score2;score3;'
5. **score:** Sum of log posterior probability ratio scores.
6. **strand:** + or -

2.5 General user options

For a full list of user options, please type:

```
pureclip --help
```

Hint: By default, the parameters are set to values optimized for proteins binding to short defined binding regions, e.g. proteins binding to short specific motifs such as PUM2 and RBFOX2. With the `-bc` option this behaviour can be changed. Use `-bc 1` for proteins causing larger crosslink clusters with relatively lower read start counts (e.g. proteins binding to low complexity motifs).

Hint: In case of data suffering from a large fraction of non-coinciding read start counts, for example caused by constrained ends, decrease the initial binomial probability parameter of the 'crosslink' state (e.g. `-b2p 0.03`). Additionally use the option `-antp` to enable the computation of a suitable n threshold, which is applied when learning parameters linked to the 'crosslink' state. With this we avoid the parameter learning to be impaired by lower covered regions where no distinction between 'crosslink' and 'non-crosslink' states is possible.

2.5.1 Remarks

Floating point precision:

- By default PureCLIP stores emission, state posterior and forward-backward probabilities as double floating-point numbers. However, for some data, in particular when including background control data that additionally contains artefacts, emission probabilities of outliers can become very small. In such cases, in order to allow the required computations, a higher floating point precision is required, i.e. long double precision, which can be applied with the parameter `-ld`. Note that this comes with a higher memory consumption.

Training set to learn model parameters:

- In order to reduce the memory consumption of PureCLIP, we learned the model parameters used in the PureCLIP paper only for a subset of chromosomes, i.e. `-iv 'chr1;chr2;chr3;'`. When using PureCLIP in basic mode, i.e. without incorporating any covariates, for the evaluated data this does not significantly change the results. However, it should be noted that when incorporating input signal, PureCLIPs precision usually improves when learning on a larger set.

Gamma shape parameters when incorporating background control data:

- By default, PureCLIP enforces the shape parameter of the 'non-enriched' gamma distribution to be less or equal than the shape parameter of the 'enriched' distribution. This constraint can be turned off using `-fk` (it could be observed to improve results for some datasets).

Mapping artefacts:

- In general mapping artefacts should be handled during preprocessing. However, the parameter `-mkn` can be used to define the max. k/N ratio (#read start sites/fragment coverage in region) when learning truncation probabilities for the 'non-crosslink' and 'crosslink' states. A large number of extreme high ratios, e.g. 1, might originate from mapping artifacts and can disturb the parameter learning so that PureCLIP becomes insensitive for sites with lower ratios.

2.6 Trouble Shooting - FAQs

Note: We currently work on PureCLIP to improve its usability and robustness, and are therefore happy about any feedback. If you run into any problems, please do not hesitate to contact us, we would be happy to help.

CHAPTER 3

Simulation Framework

If you want to simulate your own iCLIP/eCLIP data, please have a look at the sim_iCLIP git repository and the corresponding wiki.

CHAPTER 4

Citation

Krakau S, Richard H, Marsico A: PureCLIP: Capturing target-specific protein-RNA interaction footprints from single-nucleotide CLIP-seq data. Genome Biology 2017; 18:240; https://doi.org/10.1186/s13059-017-1364-2

CHAPTER 5

License

PureCLIP is licensed under the GPLv3.

CHAPTER 6

Contact

If you have any questions or comments, please contact krakau@molgen.mpg.de.