Supplementary Protocol 2: eCLIP-seq Processing Pipeline

Requires:

- Sequencing data (in fastq format).
- Genome STAR index directory (fasta files may be downloaded from UCSC; hg19)
- Repeat element STAR index directory (fasta files may be downloaded from RepBase)
- FASTA file containing barcodes for demultiplexing reads (described below)
- chrom.sizes file (tabbed file containing chromosome name and length, can be downloaded from UCSC; hg19)

Installation (Programs Used & Version Information):

This pipeline can be run with the following software installed on your environment. We recommend installation through the Anaconda package manager. Complete documentation can be found here: https://github.com/yeolab/eclip

Yeo Lab Custom Script Versions:
makebigwigfiles.py: https://github.com/YeoLab/makebigwigfiles/
clipper: https://github.com/YeoLab/clipper
clip_analysis: https://github.com/YeoLab/clipper/releases/tag/1.1
demux.py: https://github.com/YeoLab/eclipdemux
Input normalization and IDR workflow (described below):
https://github.com/YeoLab/merge_peaks/releases/tag/0.0.6

Other programs used:
FastQC: v. 0.10.1
Cutadapt: v. 1.14
STAR: v. STAR_2.5.2b
Samtools: v. 1.6
bedToBigBed: v. 2.7
Bedtools: v. 2.27.1
R: v. 3.3.2
fastq-sort: v. 0.8.0
umi_tools: v. 1.0.0

Python 2.7.17 :: Anaconda 2.1.0 (64-bit)
Pysam 0.12.0.1
Bx 0.5.0
HTSeq 0.11.2
Numpy 1.10.2
Pandas 0.23.1
Pybedtools 0.8.1

Sklearn 0.17.1
Scipy 0.17.1
Matplotlib 2.2.2
Gffutils 0.8.7.1
Statsmodels 0.6.1

perl=5.10.1 (changes to sorting in 5.22 may work but cause slightly different peak output)
Statistics::Basic 1.6611
Statistics::Distributions 1.02
Statistics::R 0.34

Outline of workflow:

- (paired-end only) Demultiplexes paired-end reads using inline barcodes and extracts unique molecular identifiers (UMI) with `eclipdemux`. (single-end) extract unique molecular identifiers with `umi_tools`
- Trims adapters & adapter-dimers with `cutadapt`
- Maps to repeat elements with `STAR` and filter
- Maps filtered reads to genome with `STAR`
- Removes PCR-duplicates with umi_tools (single-end) or with a custom python script (`barcodecollapsepe.py`)
- (paired-end only) Merges multiple inline barcodes and filters R1 (uses only R2 for peak calling)
- Calls enriched peak regions (peak clusters) with `CLIPPER`
- Uses size-matched input sample to normalize and calculate fold-change enrichment within enriched peak regions with custom perl scripts (`overlap_peakfi_with_bam_PE.pl, peakscompress.pl`)

Usage:

Script Details

Our processing pipeline is performed using Common Workflow Language (CWL) definition files that take sequencing reads and returns per-replicate peaks that can be used as inputs into an IDR-based workflow to merge replicates. Complete documentation can be found here: https://github.com/yeolab/eclip

Human Readable Description of Steps

Note: For paired-end data, until the merging step each script is run twice, once for each barcode used

Identify unique molecular identifiers (UMIs) (SE): Use umi_tools to extract unique molecular barcodes.

```
umi_tools extract \
--random-seed 1 \
--bc-pattern NNNNNNNNNN \
--log EXAMPLE_SE.rep1_clip.---.--.metrics \
--stdin file_R1.fastq.gz \
--stdout EXAMPLE_SE.rep1.umi.r1.fq
```

Demultiplexing inline barcodes and identify UMIs (PE): Perform demultiplexing using a supplied barcodes FASTA file. Depending on protocol, `--length` may be longer than 5.

```
eclipdemux \
--metrics EXAMPLE_PE.rep1_clip.---.--.metrics \
--expectedbarcodeida C01 \
--expectedbarcodeidb D8f \
--fastq_1 file_R1.fastq.gz \
--fastq_2 file_R2.fastq.gz \
--newname rep2_clip \
--dataset EXAMPLE_PE \
--barcodesfile yeolabbarcodes_20170101.fasta \
--length 5
```

- Use the FASTA ID for `--expectedbarcodeida` and `--expectedbarcodeidb` corresponding to each expected barcode of your IP sample. Use "`NIL`" for barcode-less size-matched input
- `--length` describes the length of the UMI to save.
- `--metrics`, `--newname`, `--dataset` are all labels that can be used to customize output file names
- `--barcodesfile` contents are described as follows:

Inline barcode description:
Each inline barcode is ligated to the 5' end of Read1 and its id and sequence are listed below:

```
A01     ATTGCTTAGATCGGAAGAGCGTCGTGT
B06     ACAAGCCAGATCGGAAGAGCGTCGTGT
C01     AACTTGTAGATCGGAAGAGCGTCGTGT
D8f     AGGACCAAGATCGGAAGAGCGTCGTGT
A03     ANNNNGGTCATAGATCGGAAGAGCGTCGTGT
G07     ANNNNACAGGAAGATCGGAAGAGCGTCGTGT
A04     ANNNNAAGCTGAGATCGGAAGAGCGTCGTGT
F05     ANNNNGTATCCAGATCGGAAGAGCGTCGTGT
RiL19/NIL   AGATCGGAAGAGCGTCGTGT
```

(see eCLIP protocol document for full description of these oligos)

We have observed occasional double ligation events on the 5' end of Read1, and we have found that to fix this requires we run cutadapt twice. Additionally, because two adapters are used for each library (to ensure proper balancing on the Illumina sequencer), two separate barcodes may be ligated to the same Read1 5' end (often with 5' truncations). To fix this we split the barcodes up into 15bp chunks so that cutadapt is able to deconvolute barcode adapters properly (as by default it will not find adapters missing the first N bases of the adapter sequence)

The barcodes file is made by appending one of the barcodes below (these are the same barcode sequences used to demultiplex). See example file (yeolabbarcodes_20170101.fasta):

```
AAGCAAT A01
```

```
GGCTTGT B06
ACAAGTT C01
TGGTCCT D8f
ATGACCNNNNT  A03
TCCTGTNNNNT  G07
CAGCTTNNNNT  A04
GGATACNNNNT  F05
```

To the 5' adapter
```
CTTCCGATCT
```

Cutadapt round 1: Takes output from demultiplexed files. Run to trim off both 5' and 3' adapters on both reads

```
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 EXAMPLE_PE.rep2_clip.C01.r1.fqTr.fq \
-p EXAMPLE_PE.rep2_clip.C01.r2.fqTr.fq \
EXAMPLE_PE.rep2_clip.C01.r1.fq.gz \
EXAMPLE_PE.rep2_clip.C01.r2.fq.gz
```

Fastqc round 1: Run and examined by eye to make sure libraries look alright

```
fastqc -t 2 --extract -k 7 EXAMPLE_PE.rep2_clip.C01.r1.fqTr.fq -o .
fastqc -t 2 --extract -k 7 EXAMPLE_PE.rep2_clip.C01.r2.fqTr.fq -o .
```

Cutadapt round 2: Takes output from cutadapt round 1. Run to trim off the 3' adapters on read 2, to control for double ligation 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 EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.fq \
-p EXAMPLE_PE.rep2_clip.C01.r2.fqTrTr.fq \
EXAMPLE_PE.rep2_clip.C01.r1.fqTr.fq \
EXAMPLE_PE.rep2_clip.C01.r2.fqTr.fq
```

Fastqc round 2: Takes output from STAR rmRep. Runs a second round of fastqc to verify that after read grooming the data still is usable.

```
fastqc -t 2 --extract -k 7 EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.fq -o .
fastqc -t 2 --extract -k 7 EXAMPLE_PE.rep2_clip.C01.r2.fqTrTr.fq -o .
```

Fastq-sort: Takes cutadapt round 2 output and sorts to reduce randomness

```
fastq-sort --id EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.fq >
EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.sorted.fq
fastq-sort --id EXAMPLE_PE.rep2_clip.C01.r2.fqTrTr.fq >
EXAMPLE_PE.rep2_clip.C01.r2.fqTrTr.sorted.fq
```

STAR rmRep: Takes sorted output from cutadapt round 2. Maps to human specific version of RepBase used to remove repetitive elements, helps control for spurious artifacts from rRNA (& other) repetitive reads.

```
STAR \
--runMode alignReads \
--runThreadN 8 \
--genomeDir homo_sapiens_repbase_v2 \
--genomeLoad NoSharedMemory \
--alignEndsType EndToEnd \
--outSAMunmapped Within \
--outFilterMultimapNmax 30 \
--outFilterMultimapScoreRange 1 \
--outFileNamePrefix EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.sorted.STAR \
--outSAMtype BAM Unsorted \
```

```
--outFilterType BySJout \
--outBAMcompression 10 \
--outReadsUnmapped Fastx \
--outFilterScoreMin 10 \
--outSAMattrRGline ID:foo \
--outSAMattributes All \
--outSAMmode Full \
--outStd Log \
--readFilesIn EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.sorted.fq
EXAMPLE_PE.rep2_clip.C01.r2.fqTrTr.sorted.fq
```

Re-name files: re-name repeat-mapped outputs

```
mv EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.sorted.STARAligned.out.bam
EXAMPLE_PE.rep2_clip.C01.r1.fq.repeat-mapped.bam

mv EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.sorted.STARUnmapped.out.mate1
EXAMPLE_PE.rep2_clip.C01.r1.fq.repeat-unmapped.fq
mv EXAMPLE_PE.rep2_clip.C01.r1.fqTrTr.sorted.STARUnmapped.out.mate2
EXAMPLE_PE.rep2_clip.C01.r2.fq.repeat-unmapped.fq
```

Fastq-sort: Takes unmapped output from STAR rmRep and sorts it to account for issues with STAR not outputting first and second mate pairs in order

```
fastq-sort --id EXAMPLE_PE.rep2_clip.C01.r1.fq.repeat-unmapped.fq >
EXAMPLE_PE.rep2_clip.C01.r1.fq.repeat-unmapped.sorted.fq
fastq-sort --id EXAMPLE_PE.rep2_clip.C01.r2.fq.repeat-unmapped.fq >
EXAMPLE_PE.rep2_clip.C01.r2.fq.repeat-unmapped.sorted.fq
```

STAR genome mapping: Takes output from STAR rmRep. Maps unique reads to the human genome

```
STAR \
--runMode alignReads \
--runThreadN 8 \
--genomeDir /stage/hg19_star_sjdb \
--genomeLoad NoSharedMemory \
--readFilesIn \
EXAMPLE_PE.rep2_clip.C01.r1.fq.repeat-unmapped.sorted.fq \
EXAMPLE_PE.rep2_clip.C01.r2.fq.repeat-unmapped.sorted.fq \
--outSAMunmapped Within \
--outFilterMultimapNmax 1 \
--outFilterMultimapScoreRange 1 \
--outFileNamePrefix EXAMPLE_PE.rep2_clip.C01.r1.fq.repeat-unmapped.sorted.STAR
\
--outSAMattributes All \
--outSAMtype BAM Unsorted \
--outFilterType BySJout \
--outReadsUnmapped Fastx \
--outFilterScoreMin 10 \
--outSAMattrRGline ID:foo \
--outStd Log \
--alignEndsType EndToEnd \
--outBAMcompression 10 \
--outSAMmode Full
```

Re-name BAM: rename genome-mapped outputs

```
mv EXAMPLE_PE.rep2_clip.C01.r1.fq.repeat-unmapped.sorted.STARAligned.out.bam
EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mapped.bam
```

Name sort BAM: sort output from STAR by name to ensure read pairs are adjacent.

```
samtools sort -n -o EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.bam
EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mapped.bam
```

Barcode_collapse_pe (PE): takes output from STAR genome mapping. Custom random-mer-aware script for PCR duplicate removal.

```
barcodecollapsepe.py \
-o EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.rmDup.bam \
-m EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.rmDup.metrics \
-b EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.bam
```

Position sort BAM: Takes output from barcode collapse PE (or from SE namesort bam). Sorts resulting bam file for use downstream.

```
samtools sort -o EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.bam EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDup.bam
```

Barcode_collapse_se (SE): takes output from STAR genome mapping. Use umi_tools dedup to identify the extracted random-mer from the previous step and perform PCR duplicate removal.

```
umi_tools dedup \
--random-seed 1 \
---I EXAMPLE_SE.rep1_clip.umi.r1.fq.genome-mappedSoSo.bam \
--method unique \
--output-stats EXAMPLE_SE.rep1_clip.umi.r1.fq.genome-mappedSoSo.txt \
-S EXAMPLE_SE.rep1_clip.umi.r1.fq.genome-mappedSoSo.rmDup.bam
```

Samtools index: Takes output from sortSam, makes bam index for use downstream.

```
samtools index EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.rmDupSo.bam
```

Samtools merge (PE only): Takes inputs from multiple final bam files. Merges the two technical replicates for further downstream analysis.

```
samtools  merge EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.bam EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.bam EXAMPLE_PE.rep2_clip.D8f.r1.fq.genome-
mappedSo.rmDupSo.bam
```

Samtools index: Takes output from sortSam, makes bam index for use downstream.

```
samtools  index  EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.bam
```

Samtools view (PE only): Takes output from samtools merge. Only outputs the second read in each pair for use with a single stranded peak caller. This is the final bam file to perform analysis on.

```
samtools view -f 128 -b -o EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.bam EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.bam
```

Make normalized read density bigwig files: Takes input from samtools view. Makes bw files to be uploaded to the genome browser or for other visualization. Use `--direction f` for SE clip as reads are not reversed.

```
makebigwigfiles \
--bw_pos EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.norm.pos.bw \
--bw_neg EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.norm.neg.bw \
--bam EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.rmDupSo.merged.r2.bam \
--genome hg19.chrom.sizes \
--direction r
```

Clipper: Takes results from samtools view. Calls peaks on those files.

```
clipper \
--species hg19 \
--bam EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.rmDupSo.merged.r2.bam \
--save-pickle \
--outfile EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.peakClusters.bed
```

Input normalization: Compares the number of reads within the IP sample to the number of reads within the size-matched INPUT sample across Clipper-called peak clusters. This step is performed both within this pipeline as well as within the `merge_peaks` pipeline using the same perl scripts.

```
samtools view -cF 4 EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.bam > ip_mapped_readnum.txt
samtools view -cF 4 EXAMPLE_PE.rep2_input.NIL.r1.fq.genome-
mappedSo.rmDupSo.r2.bam > input_mapped_readnum.txt

overlap_peakfi_with_bam_PE.pl \
EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.rmDupSo.merged.r2.bam \
EXAMPLE_PE.rep2_input.NIL.r1.fq.genome-mappedSo.rmDupSo.r2.bam \
EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.peakClusters.bed \
ip_mapped_readnum.txt \
input_mapped_readnum.txt \
EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.peakClusters.normed.bed

perl compress_l2foldenrpeakfi_for_replicate_overlapping_bedformat.pl \
EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.peakClusters.normed.bed \
EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.peakClusters.normed.compressed.bed
```

Peak normalization vs SMInput and

reproducible peak / IDR analysis

Peak normalization vs paired SMInput datasets is run as a second processing pipeline (`merge_peaks`) available with additional documentation on github (https://github.com/YeoLab/merge_peaks). Input files for normalization pipeline include .bam and .peak.bed files (generated through the pipeline above), as well as a manifest file pairing eCLIP datasets with their paired SMInput datasets as follows.

Requires:

- Read 2 BAM file outputs from the core pipeline for each replicate IP and INPUT (ie. `EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.rmDupSo.merged.r2.bam`, `EXAMPLE_PE.rep2_input.NIL.r1.fq.genome-mappedSo.rmDupSo.r2.bam`). Or Read1 BAM file outputs if processing SE reads (ie. `EXAMPLE_SE.rep1_clip.umi.r1.fq.genome-mappedSoSo.rmDup.bam`)

- CLIPPER peak files from the core pipeline for each replicate (ie. `EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-mappedSo.rmDupSo.merged.r2.peakClusters.bed`)

Installation:

This pipeline can be run with the following software installed on your environment. We recommend installation through the Anaconda package manager:

- **perl=5.10.1 (changes to sorting in 5.22 may work but cause slightly different peak output)**
  - Statistics::Basic 1.6611
  - Statistics::Distributions 1.02
  - Statistics::R 0.34
- **IDR=2.0.2 (base environment):**
  - python=3.4.5
  - numpy=1.11.3
  - scipy=0.19.1
  - setuptools=32.3.1.post20170108
  - matplotlib=2.0.0b4
- cwl=1.0

Outline of workflow:

- Normalize CLIP BAM over INPUT for each replicate (`overlap_peakfi_with_bam.cwl`)

- Peak compression/merging on input-normalized peaks for each replicate (`compress_l2foldenrpeakfi_for_replicate_overlapping_bedformat_outputfull.cwl`)
- Entropy calculation on CLIP and INPUT read probabilities within each peak for each replicate (`make_informationcontent_from_peaks.cwl`)
- Reformat *.full files into *.bed files for each replicate (`full_to_bed.cwl`)
- Run IDR on peaks ranked by entropy (`idr.cwl`)
- Calculates summary statistics at different IDR cutoffs (`parse_idr_peaks.cwl`)
- Normalize CLIP BAM over INPUT using new IDR peak positions (`overlap_peakfi_with_bam.cwl`)
- Identifies reproducible peaks within IDR regions (`get_reproducing_peaks.cwl`)

Usage:

Below is a description of all fields required to be filled out in the manifest file. See `https://github.com/YeoLab/merge_peaks/blob/master/example/204_RBFOX2.yaml` for a full example for the ENCODE RBFOX2 HepG2 eCLIP experiment.

BAM file containing the merged-barcode (read 2 only) post PCR duplicate removal CLIP reads mapping to the genome for Replicate 1:

```
rep1_clip_bam_file:
  class: File
  path: EXAMPLE_PE.rep1_clip.A01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.bam
```

BAM file containing the merged-barcode (read 2 only) post PCR duplicate removal INPUT reads mapping to the genome for Replicate 1:

```
rep1_input_bam_file:
  class: File
  path: EXAMPLE_PE.rep1_input.A01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.bam
```

BED file containing the called peak clusters for Replicate 1 **Output from CLIPPER**. This pipeline performs input normalization:

```
rep1_peaks_bed_file:
  class: File
  path: EXAMPLE_PE.rep1_clip.A01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.peakClusters.bed
```

BAM file containing the merged-barcode (read 2 only) post PCR duplicate removal CLIP reads mapping to the genome for Replicate 2:

```
rep2_clip_bam_file:
  class: File
  path: EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.bam
```

BAM file containing the merged-barcode (read 2 only) post PCR duplicate removal INPUT reads mapping to the genome for Replicate 2:

```
rep2_input_bam_file:
  class: File
  path: EXAMPLE_PE.rep2_input.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.bam
```

BED file containing the called peak clusters for Replicate 2 **Output from CLIPPER**. This pipeline performs input normalization:

```
rep2_peaks_bed_file:
  class: File
  path: EXAMPLE_PE.rep2_clip.C01.r1.fq.genome-
mappedSo.rmDupSo.merged.r2.peakClusters.bed
```

Final output files:

```
### FINAL OUTPUTS
merged_peaks_custombed: reproducible_peaks.bed
merged_peaks_bed: reproducible_peaks.custombed
```

Yeo Lab, UCSD - Contact geneyeo@ucsd.edu , elvannostrand@ucsd.edu

To run the workflow:

- Ensure that the yaml file is accessible and that `wf_get_reproducible_eclip_peaks.cwl` is in your `$PATH`.
- Type: `./204_RBFOX2.yaml`

Outputs

- `*.merged_peaks_bed`: this is the BED6 file containing reproducible peaks as determined by entropy-ordered peaks between two replicates.
  - chromosome
  - start
  - end
  - geomean of the log2 fold changes
  - minimum of the -log10 p-value between two replicates
  - strand This is probably what will be useful.
- `*.full` files: these tabbed outputs have the following columns (in order):
  - chromosome
  - start
  - end
  - name (colon separated region)
  - reads in CLIP
  - reads in INPUT
  - p-value
  - chi value or (F)isher
  - (F)isher or (C)hi square test
  - enriched or depleted
  - negative log10p value (set to 400 if Perl Statistics::Distributions reports 'p = 0')
  - log2 fold change
  - entropy
- `*.idr.out`: direct output from IDR, used only as an intermediate for QC checks
  - IDR chromosome
  - IDR start
  - IDR end
  - IDR name
  - IDR score (min(int(log2(-125*IDR*), 1000))
  - IDR strand
  - localIDR (-log10(*Local IDR*))
  - globalIDR (-log10(*Global IDR*))
  - Rep1 merged peak start
  - Rep1 merged peak end

    - Rep1 entropy
    - Rep2 merged peak start
    - Rep2 merged peak end
    - Rep2 entropy
- `*.idr.out.normed.bed`: output from IDR as a bed file
    - chromosome
    - start
    - end
    - minimum of the $-\log_{10}$(p-value) of the region in rep1 and rep2
    - geometric mean of the l2fc
    - strand
- `*.custombed`: contains individual replicate information. The headers are:
    - IDR region (entire IDR identified reproducible region)
    - peak (reproducible peak region)
    - geomean of the l2fc
    - rep1 log2 fold change
    - rep2 log2 fold change
    - rep1 -log10 p value
    - rep2 -log10 p value

Changelog:

1.P.2 20160426 – Clarified section regarding inline demultiplexing to specify which steps occur to generate fastq files which are submitted to the ENCODE DCC.
2.0 20180724 – Replaced input normalization section with new CWL pipeline, which now includes both input normalization as well as identification of reproducible peaks using the IDR framework
2.2 20200409 - Fixed steps to include steps for SE processing and renamed example files to mirror actual outputs from the latest pipeline version.