**Supplementary Protocol 2: eCLIP-seq Processing Pipeline**

**Programs Used & Version Information**

(For all custom scripts: <https://github.com/gpratt/gatk/releases/tag/2.3.2>)

**Yeo Lab Custom Script Versions:**

Barcode\_collapse\_pe.py: <https://github.com/YeoLab/gscripts/releases/tag/1.1>

Make\_bigwig\_files.py: <https://github.com/YeoLab/gscripts/releases/tag/1.1>

Clipper: <https://github.com/YeoLab/clipper/releases/tag/1.1>

Clip\_analysis: <https://github.com/YeoLab/clipper/releases/tag/1.1>

negBedGraph.py: <https://github.com/YeoLab/gscripts/releases/tag/1.1>

demux\_paired\_end.py: <https://github.com/YeoLab/gscripts/releases/tag/1.1>

fastq-sort: http://homes.cs.washington.edu/~dcjones/fastq-tools/fastq-tools-0.8.tar.gz

Peak\_input\_normalization\_wrapper.pl: <https://github.com/YeoLab/gscripts/tree/1.1/perl_scripts>

overlap\_peakfi\_with\_bam\_PE.pl: <https://github.com/YeoLab/gscripts/tree/1.1/perl_scripts>

compress\_l2foldenrpeakfi.pl: <https://github.com/YeoLab/gscripts/tree/1.1/perl_scripts>

**umi\_tools: https://github.com/CGATOxford/UMI-tools**

**Other programs used:**

FastQC: v. 0.10.1

Cutadapt: v. 1.9.dev1

STAR: v. STAR\_2.4.0i

Samtools: v. 0.1.19-96b5f2294a

bedToBigBed: v. 2.6

Bedtools: v. 2.25.0

R: v. 3.0.2

**Python and Python Package Versions:**

Python 2.7.11 :: Anaconda 2.1.0 (64-bit)

Pysam 0.8.3

Bx 0.5.0

HTSeq 0.6.1p1

Numpy 1.10.2

Pandas 0.17.0

Pybedtools 0.7.0

Sklearn 0.15.2

Scipy 0.16.1

Matplotlib 1.4.3

Gffutils 0.8.2

Seaborn 0.6.0

Statsmodels 0.5.0

**Perl Packages used:**

Statistics-Distributions-1.02

**Script Details**

Our entire processing pipeline is performed by two commands: (1) Demultiplexing of fastq files based on inline barcodes, and (2) A scala command that procedurally performs all subsequent processing steps in order. See the next section for detailed description of processing steps performed by the scala pipeline.

For single end demuxing consider using UMI tools https://github.com/CGATOxford/UMI-tools

**Steps used to generate the fastq files available on ENCODE DCC (input is HiSeq files from sequencing center):**

**Demultiplexing**:

**Script:**

demux\_paired\_end.py --fastq\_1 <fastq\_read\_1> --fastq\_2 <fastq\_read\_2> -b <barcode\_file.txt> --out\_file\_1 <fastq\_read\_1\_out> --out\_file\_2 <fastq\_read\_2\_out> --length <randomer\_length> -m <metrics\_file>

umi\_tools extract –stdin=<fastq\_file> --bc-pattern=NNNNNNNNNN --log=processed.log --stdout <fastq\_file\_out>

**Pipeline:**

**Script:**

java -Xms512m -Xmx512m -jar /path/to/gatk/dist/Queue.jar -S /path/to/qscripts/analyze\_clip\_seq\_encode.scala --input manifest.txt --barcoded --adapter AATGATACGGCGACCACCGAGATCTCTCTTTCCCTACACGACGCTCTTCCGATCT --adapter CAAGCAGAAGACGGCATACGAGATCGGTCTCGGCATTCCTGCTGAACCGCTCTTCCGATCT --adapter AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter ATTGCTTAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter ACAAGCCAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter AACTTGTAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter AGGACCAAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter ANNNNGGTCATAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter ANNNNACAGGAAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter ANNNNAAGCTGAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter ANNNNGTATCCAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --g\_adapter CTACACGACGCTCTTCCGATCT -qsub -jobQueue home-yeo -jobNative "-W group\_list=yeo-group" -runDir /path/to/output/directory -log result.log -keepIntermediates --job\_limit 400 -run

**Input manifest.txt documentation:**

This is a **tab separated file** that is 7 columns long.

**Column 1**: read 1 and read 2 input fastq files separated by a semi-colon.

**Column 2**: Species, either hg19 or mm9

**Column 3**: Biological Replicate ID. If two columns have the same ID they will be merged post mapping and duplicate removal.

**Column 4**: 3’ adapters to be removed from the second read in the pair.

**Column 5**: minimum length of overlap between adapter and barcode for cutadapt. (Used with variable length barcode/random-mer structures).

**Column 6**: 5’ adapters to be removed from the first read in the pair.

**Column 7**: length of random-mers to be trimmed from the 3’ end of read 1

**Example Manifest:**

/full/path/to/files/file\_R1.C01.fastq.gz;/full/path/to/files/file\_R2.C01.fastq.gz hg19 Merged\_ID AACTTGTAGATCGGA;AGGACCAAGATCGGA;ACTTGTAGATCGGAA;GGACCAAGATCGGAA;CTTGTAGATCGGAAG;GACCAAGATCGGAAG;TTGTAGATCGGAAGA;ACCAAGATCGGAAGA;TGTAGATCGGAAGAG;CCAAGATCGGAAGAG;GTAGATCGGAAGAGC;CAAGATCGGAAGAGC;TAGATCGGAAGAGCG;AAGATCGGAAGAGCG;AGATCGGAAGAGCGT;GATCGGAAGAGCGTC;ATCGGAAGAGCGTCG;TCGGAAGAGCGTCGT;CGGAAGAGCGTCGTG;GGAAGAGCGTCGTGT 5 CTTCCGATCTACAAGTT;CTTCCGATCTTGGTCCT 5

**Inline barcode description:**

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

RiL19/none 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)

Column 6 is made by appending one of the barcodes below (these are the same barcode sequences used to demultiplex):

AAGCAAT A01

GGCTTGT B06

ACAAGTT C01

TGGTCCT D08

ATGACCNNNNT A03

TCCTGTNNNNT G07

CAGCTTNNNNT A04

GGATACNNNNT F05

To the 5’ adapter

CTTCCGATCT

**Human Readable Description of Steps**

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

fastqc /full/path/to/files/file\_R1.C01.fastq.gz -o /full/path/to/files/ > /full/path/to/files/file\_R1.C01.fastq.gz.dummy\_fastqc

**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' 'AGATCGGAAGAGCAC' '-a' 'GATCGGAAGAGCACA' '-a' 'ATCGGAAGAGCACAC' '-a' 'TCGGAAGAGCACACG' '-a' 'CGGAAGAGCACACGT' '-a' 'GGAAGAGCACACGTC' '-a' 'GAAGAGCACACGTCT' '-a' 'AAGAGCACACGTCTG' '-a' 'AGAGCACACGTCTGA' '-a' 'GAGCACACGTCTGAA' '-a' 'AGCACACGTCTGAAC' '-a' 'GCACACGTCTGAACT' '-a' 'CACACGTCTGAACTC' '-a' 'ACACGTCTGAACTCC' '-a' 'CACGTCTGAACTCCA' '-a' 'ACGTCTGAACTCCAG' '-a' 'CGTCTGAACTCCAGT' '-a' 'GTCTGAACTCCAGTC' '-a' 'TCTGAACTCCAGTCA' '-a' 'CTGAACTCCAGTCAC' '-o' '/full/path/to/files/file\_R1.C01.adapterTrim.fastq.gz' '/full/path/to/files/file\_R1.C01.fastq.gz' > /full/path/to/files/file\_R1.C01adapterTrim.metrics

**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' 'AGATCGGAAGAGCAC' '-a' 'GATCGGAAGAGCACA' '-a' 'ATCGGAAGAGCACAC' '-a' 'TCGGAAGAGCACACG' '-a' 'CGGAAGAGCACACGT' '-a' 'GGAAGAGCACACGTC' '-a' 'GAAGAGCACACGTCT' '-a' 'AAGAGCACACGTCTG' '-a' 'AGAGCACACGTCTGA' '-a' 'GAGCACACGTCTGAA' '-a' 'AGCACACGTCTGAAC' '-a' 'GCACACGTCTGAACT' '-a' 'CACACGTCTGAACTC' '-a' 'ACACGTCTGAACTCC' '-a' 'CACGTCTGAACTCCA' '-a' 'ACGTCTGAACTCCAG' '-a' 'CGTCTGAACTCCAGT' '-a' 'GTCTGAACTCCAGTC' '-a' 'TCTGAACTCCAGTCA' '-a' 'CTGAACTCCAGTCAC' '-o' '/full/path/to/files/file\_R1.C01.adapterTrim.round2.fastq.gz' '/full/path/to/files/file\_R1.C01.adapterTrim.fastq.gz' > /full/path/to/files/file\_R1.C01.adapterTrim.round2.metrics

**STAR rmRep**: Takes 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' '/projects/ps-yeolab/genomes/RepBase18.05.fasta/species\_specic/homo\_sapiens\_repbase' '--genomeLoad' 'LoadAndRemove' '--readFilesIn' '/full/path/to/files/file\_R1.C01.adapterTrim.round2.fastq.gz' '--outSAMunmapped' 'Within' '--outFilterMultimapNmax' '30' '--outFilterMultimapScoreRange' '1' '--outFileNamePrefix' '/full/path/to/files/file\_R1.C01.adapterTrim.round2.rep.bam' '--outSAMattributes' 'All' '--readFilesCommand zcat' '--outStd' 'BAM\_Unsorted' '--outSAMtype' 'BAM' 'Unsorted' '--outFilterType' 'BySJout' '--outReadsUnmapped' 'Fastx' '--outFilterScoreMin' '10' '--outSAMattrRGline' 'ID:foo' '--alignEndsType EndToEnd' > /full/path/to/files/file\_R1.C01.adapterTrim.round2.rep.bam

**Samtools view and count\_aligned\_from\_sam**: Takes output from STAR rmRep. Counts the number of reads mapping to each repetitive element.

samtools view /full/path/to/files/file\_R1.C01.fastq.gz.adapterTrim.round2.rep.bam | count\_aligned\_from\_sam.py > /full/path/to/files/file\_R1.C01.fastq.gz.adapterTrim.round2.rmRep.metrics

**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 /full/path/to/files/file\_R1.C01.fastq.gz.adapterTrim.round2.rep.bamUnmapped.out.mate1 -o /full/path/to/files/ > /full/path/to/files/file\_R1.C01.fastq.gz.adapterTrim.round2.rep.bamUnmapped.out.mate1.dummy\_fastqc

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

STAR '--runMode' 'alignReads' '--runThreadN' '8' '--genomeDir' '/projects/ps-yeolab/genomes/hg19/star\_sjdb' '--genomeLoad' 'LoadAndRemove' '--readFilesIn' '/full/path/to/files/file\_R1.C01.adapterTrim.round2.rep.bamUnmapped.out.mate1' '--outSAMunmapped' 'Within' '--outFilterMultimapNmax' '1' '--outFilterMultimapScoreRange' '1' '--outFileNamePrefix' '/projects/ps-yeolab3/encode/analysis/encode\_single\_end\_small/<read 1>.adapterTrim.round2.rmRep.bam' '--outSAMattributes' 'All' '--outStd' 'BAM\_Unsorted' '--outSAMtype' 'BAM' 'Unsorted' '--outFilterType' 'BySJout' '--outReadsUnmapped' 'Fastx' '--outFilterScoreMin' '10' '--outSAMattrRGline' 'ID:foo' '--alignEndsType EndToEnd' > /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.bam

**Sort bam file**: Takes output uniquely mapped human genome data and sorts and indexes the results

samtools sort /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.bam -o /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.bam

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

samtools index /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.bam

**umitools dedup**: takes output from STAR genome mapping, performs PCR duplicate removal

umi\_tools dedup -I /full/path/to/files/file\_R1.C01adapterTrim.round2.rmRep.sorted.bam --output-stats=deduplicated -S /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.rmDup.bam

**Sort bam file**: Takes output uniquely mapped human genome data and sorts and indexes the results

samtools sort /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.rmDup.bam -o /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.rmDup.sorted.bam

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

samtools index /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.rmDup.sorted.bam /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.rmDup.sorted.bam.bai

**make\_bigwig\_files.py**: Takes input from samtools view. Makes bw files to be uploaded to the genome browser or for other visualization.

make\_bigwig\_files.py --bam /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.rmDup.sorted.bam --genome /path/to/hg19.chrom.sizes --bw\_pos /full/path/to/files/CombinedID.merged.r2.norm.pos.bw --bw\_neg /full/path/to/files/CombinedID.merged.r2.norm.neg.bw

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

clipper '-b' /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.rmDup.sorted.bam' '-s' 'hg19' '-o' /full/path/to/files/file\_R1.C01.adapterTrim.round2.rmRep.sorted.rmDup.sorted.peaks.bam' '--save-pickle'

**fix\_scores.py**: Takes input from clipper: Fixes p-values to be bed compatible

python ~/gscripts/gscripts/clipseq/fix\_scores.py --bed /full/path/to/files/CombinedID.merged.r2.peaks.bed --out\_file /full/path/to/files/CombinedID.merged.r2.peaks.fixed.bed

**bedToBigBed:** Converts bed file to bigBed file for uploading to the genomeBrowser .

bedToBigBed /full/path/to/files/CombinedID.merged.r2.peaks.fixed.bed /path/to/hg19.chrom.sizes /full/path/to/files/CombinedID.merged.r2.peaks.fixed.bb -type=bed6+4

**Peak normalization vs SMInput**

Peak normalization vs paired SMInput datasets is run as a second processing pipeline (*Peak\_input\_normalization\_wrapper.pl*). 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:

uID \t RBP \t Cell line \t CLIP\_rep1 \t CLIP\_rep2 \t INPUT

001 RBP1 HepG2 /full/path/to/files/CombinedID\_rep1.merged.r2.bam /full/path/to/files/CombinedID\_rep2.merged.r2.bam /full/path/to/files/CombinedID\_INPUT.merged.r2.bam

002 RBP2 K562 /full/path/to/files/CombinedID2\_rep1.merged.r2.bam /full/path/to/files/CombinedID2\_rep2.merged.r2.bam /full/path/to/files/CombinedID2\_INPUT.merged.r2.bam

uID = a unique identifier for each experiment

RBP, Cell line = dataset descriptors (for labeling purposes, not used in pipeline itself)

CLIP\_rep1 = full path to CLIP (replicate 1) bam file (output from **\*\*samtools view** above)

CLIP\_rep2 = full path to CLIP (replicate 2) bam file (output from **\*\*samtools view** above) (this field is OPTIONAL – it is used for for ENCODE-style experiments with 2 eCLIP paired with 1 SMInput. For experiments where each CLIP has a paired SMInput, simply remove this column and use a 5 column manifest file).

INPUT = full path to paired SMInput bam file (output from **\*\*samtools view** above)

* Note that this pipeline expects the .peak.bed files to be in the same folder as .bam files (as is standard output from the processing pipeline)

Final output for this pipeline is an input normalized peak file (in bed format):

/full/path/to/desired\_output\_directory/uID\_Rep.basedon\_uID\_Rep.peaks.l2inputnormnew.bed.compressed.bed

Formatted as follows:

Chr \t start \t stop \t log10(p-value eCLIP vs SMInput) \t log2(fold-enrichment in eCLIP vs SMInput) \t strand

Command line example:

>perl Peak\_input\_normalization\_wrapper.pl /full/path/to/manfest\_file.txt /full/path/to/desired\_output\_directory/

The wrapper performs the following steps:

1. Create output directory (if not existing)
2. Create soft-links in output directory to .bam and .peak.bed files listed in manifest
3. Count usable read numbers for each .bam file (samtools view -c -F 4) and write to a tab-delimited text file (/full/path/to/manfest\_file.txt.mapped\_read\_num)
4. Runs overlap\_peakfi\_with\_bam\_PE.pl:

Takes in two bam files (eCLIP and SMInput) and a bed file of peak regions. For each peak, counts the # of overlapping reads in eCLIP and SMInput bam files, and performs Yates’ Chi-square (in Perl) or Fisher’s Exact Test (using the R statistics package) to determine enrichment significance in eCLIP relative to SMInput.

perl overlap\_peakfi\_with\_bam\_PE.pl /full/path/to/desired\_output\_directory/CombinedID\_rep1.merged.r2.bam /full/path/to/desired\_output\_directory/CombinedID\_INPUT.merged.r2.bam /full/path/to/desired\_output\_directory/CombinedID\_rep1.merged.r2.peaks.bed

/full/path/to/manfest\_file.txt.mapped\_read\_num /full/path/to/desired\_output\_directory/uID\_Rep.basedon\_uID\_Rep.peaks.l2inputnormnew.bed

Output file has bed format:

Chr \t start \t stop \t log10(p-value eCLIP vs SMInput) \t log2(fold-enrichment in eCLIP vs SMInput) \t strand

1. Runs compress\_l2foldenrpeakfi\_for\_replicate\_overlapping\_bedformat.pl:

As CLIPper is run at the transcript level, occasionally multiple clusters can overlap the same genomic positions. This script steps through all peaks, and resolves overlapping peaks by only keeping the peak with the greater enrichment in eCLIP over SMInput.

perl compress\_l2foldenrpeakfi.pl /full/path/to/desired\_output\_directory/uID\_Rep.basedon\_uID\_Rep.peaks.l2inputnormnew.bed

Writes output to bed format file (same columns as above): /full/path/to/desired\_output\_directory/uID\_Rep.basedon\_uID\_Rep.peaks.l2inputnormnew.bed.compressed.bed

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.