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
Input normalization and IDR workflow: https://github.com/YeoLab/merge_peaks/releases/tag/0.0.6

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

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.

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>

Input file Documentation:
The input file is a tab separated file that describes the barcodes to demultiplex.

**Column 1:** Barcode to demultiplex  
**Column 2:** Human readable label to append to the demultiplexed file.

Example Manifest:

ACAAAGTT /full/path/to/files/file_R1.C01

Output:

Demultiplexed fastq files:

/full/path/to/files/file_R1.C01.fastq.gz  
/full/path/to/files/file_R2.C01.fastq.gz

In these files, the in-line barcode has been removed from R1 and the in-line randommer has been removed from R2 and appended to the ‘read name’ as follows:

Randommer ATCATGCAAT added to read name @HWI-D00611:179:C7MR1ANXX:6:1209:14781:97349 to create fastq entry:

@ATCATGCAAT:HWI-D00611:179:C7MR1ANXX:6:1209:14781:97349 1:N:0:CGCTCATTATAGAGGCCCACAAACTAAGACGGCATACGAGATCGGTCTCGGCATTCCTGCTGAACCGCTCTTCCGATCT

+ BFFFFFFFFBFFFFFF<BF<///<<<FF<BF<BFF</F<FFFF#

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 AATGATACGGCGACACCGAGATCTCTCTTTCCCTACACGACGCTCTTCCGATCT --adapter CCAACCTAAGAACGCGACGCTCTTCCTCCATCTACAGAGATTCTCGCTCACTCGCTGACACGCTCTCCGATCT --adapter AGATCGGACGACTCTTATGGTATTCTCAGAGTAGT --adapter ATGTGTAGTATCGGAAAGACTCTGCTGTAATGGGAAAAAAGTGT --adapter ACAAGCCAGATCGGAAGAGCGTCGTGTAGGAAAGGAGT --adapter
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-mer structures 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
AACTTGTAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter
AGGACCAAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter
ANNNNGTCTATGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter
ANNNNACGAGAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter
ANNNNAGCTGAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter
ANNNNGTATTCAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --adapter
-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

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

A01 AACTTGTAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
B06 AGGACCAAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
C01 AACTTGTAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
D08 ANNNAAGGTCTGAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
A03 ANNNNNGTCTATGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
G07 ANNNNACGAGAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
A04 ANNNNAGCTGAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
F05 ANNNNGTATTCAGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
R1L19:none AGATCGGAAGGCGTCGTGT

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
- TCTGTNNNNT G07
- CAGCTTNNNNT A04
- GGATACNNNNT F05
- To the 5' adapter
- CTTCCGATCT

**Human Readable Description of Steps**

Note: Until the merging step each script is run twice, one once for each barcode used

**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
```

```
fastqc /full/path/to/files/file_R2.C01.fastq.gz -o /full/path/to/files/ >
/full/path/to/files/file_R2.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 NNNNNAGATCGGAAGAGCACACGTCTGAACTCCAGTCAC -g
CTTCCGATCTACAAGTT -g CTTCCGATCTTGGTCTCT -A AACTTGTAGATCGGA -A
AGGACCAAGATCGGA -A ACTTGTAGATCGGA -A GGACCAAGATCGGA -A CTTGT
AGATCGGAAG -A GACCAAGATCGGAAG -A TTGATGATCGGAAGA -A ACACGATCGGAAGA -A
TGTAGATCGGAAGA -A CCAACGATCGGAAGA -A GATATCGGAAGA -A CAAATCGGAAGA -A
GGATACGGCGGACG -A AGATCGGAAGACG -A AGATCGGAAGACG -A GATATCGGAAGACG -A
GATCGGAAGACGCGTG -A ATCGGAAGACGCGTG -A TCGGAAGACGCGTG -A CGGAAGACGCGTG
-A GGAAGACGGCTCAGT -p
```

```
/full/path/to/files/file_R1.C01.fastq.gz.adapterTrim.fastq.gz -p
/full/path/to/files/file_R2.C01.fastq.gz.adapterTrim.fastq.gz
/full/path/to/files/file_R1.C01.fastq.gz
/full/path/to/files/file_R2.C01.fastq.gz
```

**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 GGACCAAGATCGGA -A
ACTTGTAGATCGGA -A GGACCAAGATCGGA -A TGTAGATCGGA -A GACCAAGATCGGA -A
TGTAGATCGGA -A CCAACGATCGGA -A GATATCGGA -A CAAATCGGA -A
GGATACGGCGGACG -A AGATCGGAAGACG -A AGATCGGAAGACG -A GATATCGGAAGACG -A
GATCGGAAGACGCGTG -A ATCGGAAGACGCGTG -A TCGGAAGACGCGTG -A CGGAAGACGCGTG
-A GGAAGACGGCTCAGT
```

```
/full/path/to/files/file_R1.C01.fastq.gz.adapterTrim.metrics
```

```
/full/path/to/files/file_R2.C01.fastq.gz.adapterTrim.metrics
```
eCLIP-seq Processing Pipeline v2.0 20180724
For ENCODE release
Yeo Lab, UCSD - Contact geneyeo@ucsd.edu, elvannostrand@ucsd.edu

```
-o /full/path/to/files/file_R1.C01.fastq.gz.adapterTrim.round2.fastq.gz
-p /full/path/to/files/file_R2.C01.fastq.gz.adapterTrim.round2.fastq.gz
/full/path/to/files/file_R1.C01.fastq.gz.adapterTrim.right2.fastq.gz
/full/path/to/files/file_R2.C01.fastq.gz.adapterTrim.right2.fastq.gz
/full/path/to/files/file_R1.C01.fastq.gz.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 16 --genomeDir /path/to/RepBase_human_database_file --genomeLoad LoadAndRemove --readFilesIn /full/path/to/files/file_R1.C01.fastq.gz.adapterTrim.round2.fastq.gz /full/path/to/files/file_R2.C01.fastq.gz.adapterTrim.round2.fastq.gz --outSAMunmapped Within --outFilterMultimapNmax 30 --outFilterMultimapScoreRange 1 --outFileNamePrefix /full/path/to/files/file_R1.C01.fastq.gz.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 >

Samtools view and count_aligned_from_sam: Takes output from STAR rmRep. Counts the number of reads mapping to each repetitive element.

samtools view

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

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
/full/path/to/files/file_R1.C01.fastq.gz.adapterTrim.round2.rep.bamUnmapped.out.sorted.mate1
&& fastq-sort --id

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

STAR --runMode alignReads --runThreadN 16 --genomeDir /path/to/STAR_database_file --genomeLoad LoadAndRemove --readFilesIn /full/path/to/files/file_R1.C01.fastq.gz.adapterTrim.round2.rep.bamUnmapped.out.mate1
```

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

```bash
barcode_collapse_pe.py --bam
```

sortSam: Takes output from barcode collapse PE. Sorts resulting bam file for use downstream.

```bash
java -Xmx2048m -XX:+UseParallelOldGC -XX:ParallelGCThreads=4 -XX:GCTimeLimit=50 -XX:GCTier=2 -XX:+UseConcMarkSweepGC
Djava.io.tmpdir=/full/path/to/files/.queue/tmp -cp /path/to/gatk/dist/Queue.jar net.sf.picard.sam.SortSam
TMP_DIR=/full/path/to/files/.queue/tmp
SO=coordinate CREATE_INDEX=true
```

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

```bash
samtools index
/full/path/to/files/CombinedID.merged.bam
```

samtools merge: Takes inputs from multiple final bam files. Merges the two technical replicates for further downstream analysis.

```bash
samtools merge /full/path/to/files/CombinedID.merged.bam
```

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

```bash
samtools index /full/path/to/files/CombinedID.merged.bam
/full/path/to/files/CombinedID.merged.bam
```

**samtools view:** Takes output from sortSam. Only outputs the second read in each pair for use with single stranded peak caller. This is the final bam file to perform analysis on.

```bash
samtools view -bhf 128 /full/path/to/files/CombinedID.merged.bam >
/full/path/to/files/CombinedID.merged.r2.bam
```
make_bigwig_files.py: Takes input from samtools view. Makes bw files to be uploaded to the genome browser or for other visualization.

```bash
```

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

```bash
clipper -b /full/path/to/files/CombinedID.merged.r2.bam -s hg19 -o /full/path/to/files/CombinedID.merged.r2.peaks.bed --bonferroni --superlocal -d threshold-method binomial --save-pickle
```

fix_scores.py: Takes input from clipper: Fixes p-values to be bed compatible

```bash
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.

```bash
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 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:**

- perl=5.10.1 (changes to sorting in 5.22 may cause slightly different peak output)
  - Statistics::Basic
  - Statistics::Distributions
  - Statistics::R
- IDR=2.0.2
- python=3.4.5
  - numpy=1.11
  - pandas=0.20
  - scipy=0.18
  - setuptools=27.2
  - matplotlib=2.0
- cwl=1.0
eCLIP-seq Processing Pipeline v2.0 20180724
For ENCODE release
Yeo Lab, UCSD - Contact geneyeo@ucsd.edu, elvannostrand@ucsd.edu

Installation:

- Run and source the source create_environment.sh bash script
- Install perlrew: https://perlbrew.pl/ (skip if you want to use your system perl)
- run and source the source run_perlbrew_perl5.10.1.sh bash script (skip if you want to use your system perl)
- Install perl modules:
  - cpan install Statistics::Basic
  - cpan install Statistics::Distributions
  - cpan install Statistics::R

Outline of workflow:

- Normalize CLIP BAM over INPUT for each replicate (overlap_peakfi_with_bam_PE.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_PE.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:
repl_clip_bam_file:
  class: File
  path: 204_01_RBFOX2.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:
repl_input_bam_file:
For ENCODE release
Yeo Lab, UCSD - Contact geneyeo@ucsd.edu, elvannostrand@ucsd.edu

class: File
path: RBFOX2-204-INPUT_S2_R1.unassigned.adapterTrim.round2.rmRep.rmDup.sorted.r2.bam

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

<table>
<thead>
<tr>
<th>rep1_peaks_bed_file:</th>
</tr>
</thead>
<tbody>
<tr>
<td>class: File</td>
</tr>
<tr>
<td>path: 204_01_RBFOX2.merged.r2.peaks.bed</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>rep2_clip_bam_file:</th>
</tr>
</thead>
<tbody>
<tr>
<td>class: File</td>
</tr>
<tr>
<td>path: 204_02_RBFOX2.merged.r2.bam</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>rep2_input_bam_file:</th>
</tr>
</thead>
<tbody>
<tr>
<td>class: File</td>
</tr>
<tr>
<td>path: RBFOX2-204-INPUT_S2_R1.unassigned.adapterTrim.round2.rmRep.rmDup.sorted.r2.bam</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>rep2_peaks_bed_file:</th>
</tr>
</thead>
<tbody>
<tr>
<td>class: File</td>
</tr>
<tr>
<td>path: 204_02_RBFOX2.merged.r2.peaks.bed</td>
</tr>
</tbody>
</table>

Final output files:

### FINAL OUTPUTS
| merged_peaks_custombed: 204.01v02.IDR.out.0102merged.bed |
| merged_peaks_bed: 204.01v02.IDR.out.0102merged.custombed |
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.
  - chrom
  - 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 log10 p value (set to 400 if Perl Statistics::Distributions reports ‘p = 0’)
  - log2 fold change
  - entropy
- `*.idr.out`: output from IDR
- `*.idr.out.bed`: output from IDR as a bed file
- `*.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 pvalue
  - rep2 -log10 pvalue
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.