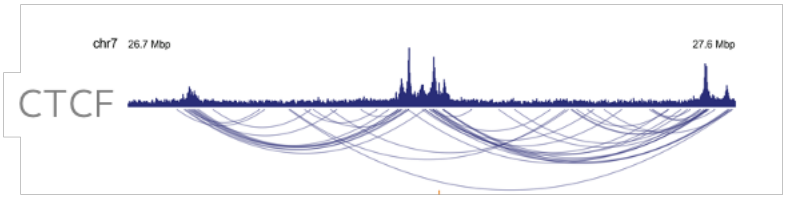
HiChiP Documentation

Release 0.1

Dovetail

Aug 01, 2023

CONTENTS:


chr7 26.7 Mbp
27.6 Mbp
CTCF

CHAPTER

ONE

OVERVIEW

- The Dovetail™ HiChIP MNase Kit combines the benefits of ChIP-seq with Hi-C, a proximity ligation method that captures long-range interactions using standard Illumina paired-end sequencing, enabling researchers to query protein-directed chromatin conformation mediated by specific proteins of interest.
- Key benefits of HiChIP:
  - Capture ChIP-seq and Hi-C data together in a single library
  - Map chromatin interactions at nucleosome level resolution
- The unique combination of the Dovetail™ Micro-C Proximity Ligation Assay with the Dovetail HiChIP approach enables the use of micrococcal nuclease (MNase) to fragment chromatin uniformly and without sequence bias prior to proximity ligation, eliminating the need for finicky sonication procedures and offering the maximal resolution (down to mono-nucleosome size) of chromatin interactions.
- Enrichment of protein-directed chromatin features enables high-resolution contact map generation with less read depth. Compared to a high resolution restriction enzyme-based Hi-C, Dovetail HiChIP data enables visualization of higher-order chromatin features, such as loops and chromatin interactions, at a fraction of the read depth leading to significant sequencing costs savings.
- This guide will take you step by step on how to QC your HiChIP library, how to interparate the QC results and how to call and plot significant interactions. If you don't yet have a sequenced HiChIP library and you want to get familiar with the data, you can download HiChIP sequences libraries from our publicaly available *data sets*.
- The QC process starts with aligning the reads to a reference genome then retaining high quality mapped reads. From there the mapped data will be used to generating a pairs file with pairtools, which categorizes pairs by read type and insert distance, this step both flags and removes PCR duplicates. Once pairs are categorized, counts of each class are summed and reported.
- If this is your first time following this tutorial, please check the *Before you begin page* first.

1.1 Before you begin

1.1.1 Have a copy of the HiChIP scripts on your machine:

Clone this repository:

```
git clone https://github.com/dovetail-genomics/HiChiP.git
```

And make the `enrichment_stats.sh` script executable:

```
chmod +x ./HiChiP/enrichment_stats.sh
```

1.1.2 Dependencies

Make sure that the following dependencies are installed:

- pysam
- tabulate
- bedtools
- deeptools
- matplotlib
- pandas
- numpy
- bwa
- pairtools
- samtools

If you are facing any issues with the installation of any of the dependencies, please contact the supporter of the relevant package.

python3 and pip3 are required, if you don't already have them installed, you will need sudo privileges.

- Update and install python3 and pip3:

```
sudo apt-get update
sudo apt-get install python3 python3-pip
```

- To set python3 and pip3 as primary alternative:

```
sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 1
sudo update-alternatives --install /usr/bin/pip pip /usr/bin/pip3 1
```

If you are working on a new machine and don't have the dependencies, you can use the `installDep.sh` script in this repository for updating your instance and installing the dependencies and python3. This process will take approximately 10' and requires sudo privileges. The script was tested on Ubuntu 18.04 with the latest version as of 04/11/2020

If you choose to run the provided installation script you will first need to set the permission to the file:

```
chmod +x ./HiChiP/installDep.sh
```

And then run the installation script:

```
./HiChiP/installDep.sh
```

Remember!

Once the installation is completed, sign off and then sign back to your instance to refresh the database of applications.

1.1.3 Input files

For this tutorial you will need:

- **fastq files** R1 and R2, either fastq or fastq.gz are acceptable
- **reference in a fasta file format**, e.g. hg38
- **peak calls from ChiP-seq experiment** (e.g. your own experiment or ENCODE gold standard in bed or narrowPeak format, as explained *here*), more details and links to ENCODE files can be found *here*.

If you don't already have your own input files or want to run a test on a small data set, you can download sample fastq files from the *HiChIP Data Sets section*. The 2M data set is suitable for a quick testing of the instructions in this tutorial.

The following files are suitable for testing, you can download them as follows:

```
wget https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/HiChiP_CTCF_2M_R1.fastq.gz
wget https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/HiChiP_CTCF_2M_R2.fastq.gz
wget https://www.encodeproject.org/files/ENCFF017XLW/@@download/ENCFF017XLW.bed.gz
```

For zipped bed files, unzip them after download is completed (no need to unzip fastq.gz files)

Example:

```
gunzip ENCFF017XLW.bed.gz
```

1.2 Pre-Alignment

For downstream steps you will need a genome file, genome file is a tab delimited file with chromosome names and their respective sizes. If you don't already have a genome file follow these steps:

1. Generate an index file for your reference, a reference file with only the main chromosomes should be used (e.g. without alternative or unplaced chromosomes).

Command:

```
samtools faidx <ref.fasta>
```

Example:

```
samtools faidx hg38.fasta
```

Faidx will index the ref file and create <ref.fasta>.fai on the reference directory.

2. Use the index file to generate the genome file by printing the first two columns into a new file.

Command:

```
cut -f1,2 <ref.fasta.fai> > <ref.genome>
```

Example:

```
cut -f1,2 hg38.fasta.fai > hg38.genome
```

In line with the 4DN project guidelines and from our own experience optimal alignment results are obtained with Burrows-Wheeler Aligner (bwa). Prior to alignment, generate a bwa index file for the chosen reference.

```
bwa index <ref.fasta>
```

Example:

```
bwa index hg38.fasta
```

No need to specify an output path, the bwa index files are automatically generated at the reference directory. Please note that this step is time consuming, however you need to run it only once for a reference.

To avoid memory issues, some of the steps require writing temporary files into a temp folder, please generate a temp folder and remember its full path. Temp files may take up to x3 of the space that the fastq.gz files are taking, that is, if the total volume of the fastq files is 5Gb, make sure that the temp folder can store at least 15Gb.

Command:

```
mkdir <full_path/to/tmpdir>
```

Example:

```
mkdir /home/ubuntu/ebs/temp
```

In this example the folder *temp* will be generated on a mounted volume called *ebs* on a user account *ubuntu*.

1.3 From fastq to final valid pairs bam file

fastq to final valid pairs bam file - for the impatient!

If you just want to give it a shot and run all the alignment and filtering steps without going over all the details, we made a shorter version for you, with all the steps piped, outputting a final bam file with its index file and a dup stats file, otherwise move to the next section *fastq to final valid pairs bam file - step by step*

Command:

```
bwa mem -5SP -T0 -t<cores> <ref.fa> <HiChiP.R1.fastq.gz> <HiChiP.R2.fastq.gz>| \
pairtools parse --min-mapq 40 --walks-policy 5unique \
--max-inter-align-gap 30 --nproc-in <cores> --nproc-out <cores> --chroms-path <ref.
↪genome> | \
pairtools sort --tmpdir=<full_path/to/tmpdir> --nproc <cores>|pairtools dedup --nproc-in
↪<cores> \
--nproc-out <cores> --mark-dups --output-stats <stats.txt>|pairtools split --nproc-in
↪<cores> \
--nproc-out <cores> --output-pairs <mapped.pairs> --output-sam -|samtools view -bS -@
↪<cores> | \
samtools sort -@<cores> -o <mapped.PT.bam>;samtools index <mapped.PT.bam>
```

Example:

```
bwa mem -5SP -T0 -t16 hg38.fasta HiChiP_CTCF_2M_R1.fastq.gz HiChiP_CTCF_2M_R2.fastq.gz|␣
↪pairtools parse --min-mapq 40 --walks-policy 5unique --max-inter-align-gap 30 --nproc-
↪in 8 --nproc-out 8 --chroms-path hg38.genome | pairtools sort --tmpdir=/home/ubuntu/
↪ebs/temp/ --nproc 16|pairtools dedup --nproc-in 8 --nproc-out 8 --mark-dups --output-
↪stats stats.txt|pairtools split --nproc-in 8 --nproc-out 8 --output-pairs mapped.pairs␣
↪--output-sam -|samtools view -bS -@16 | samtools sort -@16 -o mapped.PT.bam;samtools␣
↪index mapped.PT.bam
```


The full command above, with 2M read pairs on an Ubuntu 18.04 machine with 16 CPUs and 64GiB was completed in less than 5 minutes. On the same machine type.

1.3.1 fastq to final valid pairs bam file - step by step

Alignment

Now that you have a genome file, index file and a reference fasta file you are all set to align your HiChIP library to the reference. Please note the specific settings that are needed to map mates independently and for optimal results with our proximity library reads.

Parameter	Alignment function
mem	set the bwa to use the BWA-MEM algorithm, a fast and accurate alignment algorithm optimized for sequences in the range of 70bp to 1Mbp
-5	for split alignment, take the alignment with the smallest coordinate (5' end) as primary, the mapq assignment of the primary alignment is calculated independent of the 3' alignment
-S	skip mate rescue
-P	skip pairing; mate rescue performed unless -S also in use
-T0	The T flag set the minimum mapping quality of alignments to output, at this stage we want all the alignments to be recorded and thus T is set up to 0, (this will allow us to gather full stats of the library, at later stage we will filter the alignments by mapping quality
-t	number of threads, default is 1. Set the numbers of threads to not more than the number of cores that you have on your machine (If you don'd know the number of cores, used the command lscpu and multiply Thread(s) per core x Core(s) per socket x Socket(s))
*.fasta or *.fa	Path to a reference file, ending with .fa or .fasta, e,g, hg38.fasta
*.fastq or *.fastq.gz	Path to two fastq files; path to read 1 fastq file, followed by fastq file of read 2 (usually labeled as R1 and R2, respectively). Files can be in their compressed format (.fastq.gz) or uncompressed (.fastq). In case your library sequence is divided to multiple fastq files, you can use a process substitution < with the cat command (see example below)
-o	sam file name to use for output results [stdout]. You can choose to skip the -o flag if you are piping the output to the next command using '|'

Bwa mem will output a sam file that you can either pipe or save to a path using -o option, as in the example below (please note that version 0.7.17 or higher should be used, older versions do not support the *-5* flag)

Command:

```
bwa mem -5SP -T0 -t<threads> <ref.fasta> <HiChiP_R1.fastq> <HiChiP_R2.fastq> -o <aligned.
↪sam>
```

Example (one pair of fastq files):

```
bwa mem -5SP -T0 -t16 hg38.fasta HiChiP_CTCF_2M_R1.fastq.gz HiChiP_CTCF_2M_R2.fastq.gz -
↪o aligned.sam
```

Example (multiple pairs of fastq files):

```
bwa mem -5SP -T0 -t16 hg38.fasta <(zcat file1.R1.fastq.gz file2.R1.fastq.gz file3.R1.
↪fastq.gz) <(zcat file1.R2.fastq.gz file2.R2.fastq.gz file3.R2.fastq.gz) -o aligned.sam
```

Recording valid ligation events

We use the `parse` module of the `pairtools` pipeline to find ligation junctions in HiChIP (and other proximity ligation) libraries. When a ligation event is identified in the alignment file the pairtools pipeline will record the outer-most (5') aligned base pair and the strand of each one of the paired reads into `.pairsam` file (pairsam format captures SAM entries together with the Hi-C pair information). In addition, it will also asign a pair type for each event. e.g. if both reads aligned uniquely to only one region in the genome, the type UU (Unique-Unique) will be assigned to the pair. The following steps are necessary to identify the high quality valid pairs over low quality events (e.g. due to low mapping quality):

`pairtools parse` options:

Parameter	Value	Function
min-mapq	40	Mapq threshold for defining an alignment as a multi-mapping alignment. Alignment with mapq <40 will be marked as type M (multi)
walks-policy	5unique	Walks is the term used to describe multiple ligations events, resulting three alignments (instead of two) for a read pair. However, there are cases in which three alignment in read pairs are the result of one ligation event, pairtool parse can rescue this event. walks-policy is the policy for reporting un-rescuable walk. 5unique is used to report the 5'-most unique alignment on each side, if present (one or both sides may map to different locations on the genome, producing more than two alignments per DNA molecule)
max-inter-align-gap	30	In cases where there is a gap between alignments, if the gap is 30 or smaller, ignore the gap, if the gap is >30bp, mark as "null" alignment
nproc-in	integer, e.g. 16	pairtools has an automatic-guess function to identify the format of the input file, whether it is compressed or not. When needed, the input is decompressed by bgzip/lz4c. The option nproc-in set the number of processes used by the auto-guessed input decompressing command, if not specified, default is 3
nproc-out	integer, e.g. 16	pairtools automatic-guess the desired format of the output file (compressed or not compressed, based on file name extension). When needed, the output is compressed by bgzip/lz4c. The option nproc-out set the number of processes used by the auto-guessed output compressing command, if not specified, default is 8
chroms-path		path to .genome file, e.g. hg38.genome
*.sam		path to sam file used as an input. If you are piping the input (stdin) skip this option
*pairsam		name of pairsam file for writing output results. You can choose to skip and pipe the output directly to the next commad (pairtools sort)

`pairtools parse` command example for finding ligation events:

Command:

```
pairtools parse --min-mapq 40 --walks-policy 5unique --max-inter-align-gap 30 --nproc-in
↪<cores>\
--nproc-out <cores> --chroms-path <ref.genome> <aligned.sam> > <parsed.pairsam>
```

Example:

```
pairtools parse --min-mapq 40 --walks-policy 5unique --max-inter-align-gap 30 --nproc-in␣
↪8 --nproc-out 8 --chroms-path hg38.genome aligned.sam >  parsed.pairsam
```

At the parsing step, pairs will be flipped such that regardless of read1 and read2, pairs are always recorded with first side of the pair having the lower genomic coordinates.

Sorting the pairsam file

The parsed pairs are then sorted using *pairtools sort*

`pairtools sort` options:

Parameter	Function
–tmpdir	Provide a full path to a temp directory. A good rule of thumb is to have a space available for this directory at a volume of x3 of the overall volume of the fastq.gz files. Using a temp directory will help avoid memory issues
–nproc	Number of processes to split the sorting work

Command:

```
pairtools sort --nproc <cores> --tmpdir=<path/to/tmpdir> <parsed.pairsam> > <sorted.
↪pairsam>
```

Example:

```
pairtools sort --nproc 16 --tmpdir=/home/ubuntu/ebs/temp/  parsed.pairsam > sorted.
↪pairsam
```

Important!

Please note that an absolute path for the temp directory is required for `pairtools sort`, e.g. path of the structure ~/ebs/temp/ or ./temp/ will not work, instead, something of this sort is needed /home/user/ebs/temp/

Removig PCR duplicates

`pairtools dedup` detects molecules that could be formed via PCR duplication and tags them as "DD" pair type. These pairs should be excluded from downstream analysis. Use the pairtools dedup command with the *–output-stats* option to save the dup stats into a text file.

`` `pairtools dedup` `` options:

Parameter	Function
–mark-dups	If specified, duplicate pairs are marked as DD in "pair_type" and as a duplicate in the sam entries
–output-stats	Output file for duplicate statistics. Please note that if a file with the same name already exists, it will be opened in the append mode

Command:

```
pairtools dedup --nproc-in <cores> --nproc-out <cores> --mark-dups --output-stats <stats.
↪txt> \
--output <dedup.pairsam> <sorted.pairsam>
```

Example:

```
pairtools dedup --nproc-in 8 --nproc-out 8 --mark-dups --output-stats stats.txt --output␣
↪dedup.pairsam sorted.pairsam
```

Generate .pairs and bam files

The `pairtools split` command is used to split the final `.pairsam` into two files: `.sam` (or `.bam`) and `.pairs` (`.pairsam` has two extra columns containing the alignments from which the HiChIP pair was extracted, these two columns are not included in `.pairs` files)

`pairtools split` options:

Parameter	Function
–output-pairs	Output pairs file. If the path ends with .gz or .lz4 the output is pbgzip-/lz4c-compressed. If you wish to pipe the command and output the pairs fils to stdout use - instead of file name
–output-sam	Output sam file. If the file name extension is .bam, the output will be written in bam format. If you wish to pipe the command, use - instead of a file name. please note that in this case the sam format will be used (and can be later converted to bam file e.g. with the command samtools view -bS -@16 -o temp.bam

Command:

```
pairtools split --nproc-in <cores> --nproc-out <cores> --output-pairs <mapped.pairs> \
--output-sam <unsorted.bam> <dedup.pairsam>
```

Example:

```
pairtools split --nproc-in 8 --nproc-out 8 --output-pairs mapped.pairs --output-sam␣
↪unsorted.bam dedup.pairsam
```

The `.pairs` file can be used for generating *contact matrix*

Generating the final bam file

For downstream steps, the bam file should be sorted, using the command *samtools sort*

`samtools sort` options:

Parameter	Function
-@	number of threads to use
-o	ile name. Write final output to FILE rather than standard output
-T	path to temp file. Using a temp file will help avoiding memory issues

Command:

```
samtools sort -@<threads> -T <path/to/tmpdir/tempfile.bam>-o <mapped.PT.bam> <unsorted.
↪bam>
```

Example:

```
samtools sort -@16 -T /home/ubuntu/ebs/temp/temp.bam -o mapped.PT.bam unsorted.bam
```

For future steps an index (.bai) of the bam file is also needed. Index the bam file:

Command:

```
samtools index <mapped.PT.bam>
```

Example:

```
samtools index mapped.PT.bam
```

The `mapped.PT.bam` is the final bam file that will be used downstream steps.

The above steps resulted in multiple intermediate files, to simplify the process and avoid intermediate files, you can pipe the steps as in the example above (*fastq to final valid pairs bam file - for the impatient*)

1.4 Library QC

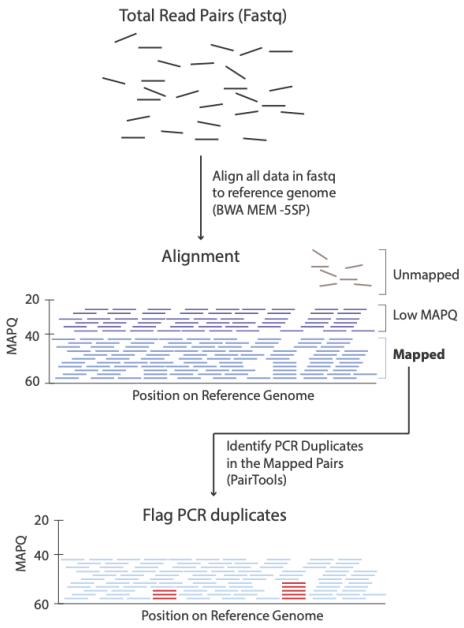
1.4.1 Proximity-ligation assessment

At step *Removing PCR duplicates* you used the flag *–output-stats*, generating a stats file in addition to the pairsam output (e.g. –output-stats stats.txt). The stats file is an extensive output of pairs statistics as calculated by pairtools, including total reads, total mapped, total dups, total pairs for each pair of chromosomes etc'. Although you can use directly the pairtools stats file as is to get informed on the quality of the HiChIP library, we find it easier to focus on a few key metrics. We include in this repository the script `get_qc.py` that summarize the paired-tools stats file and present them in percentage values in addition to absolute values.

The images below explains how the values on the QC report are calculated:

I. Aligning and filtering to remove low mapping quality and PCR duplicate read pairs

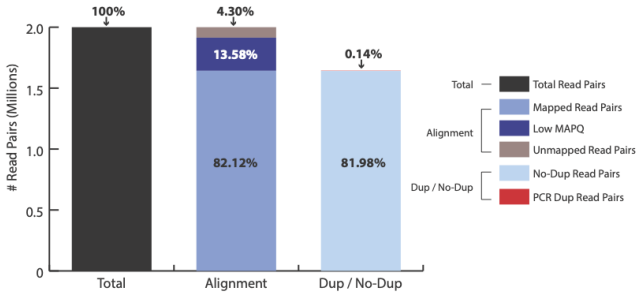
Process



Results

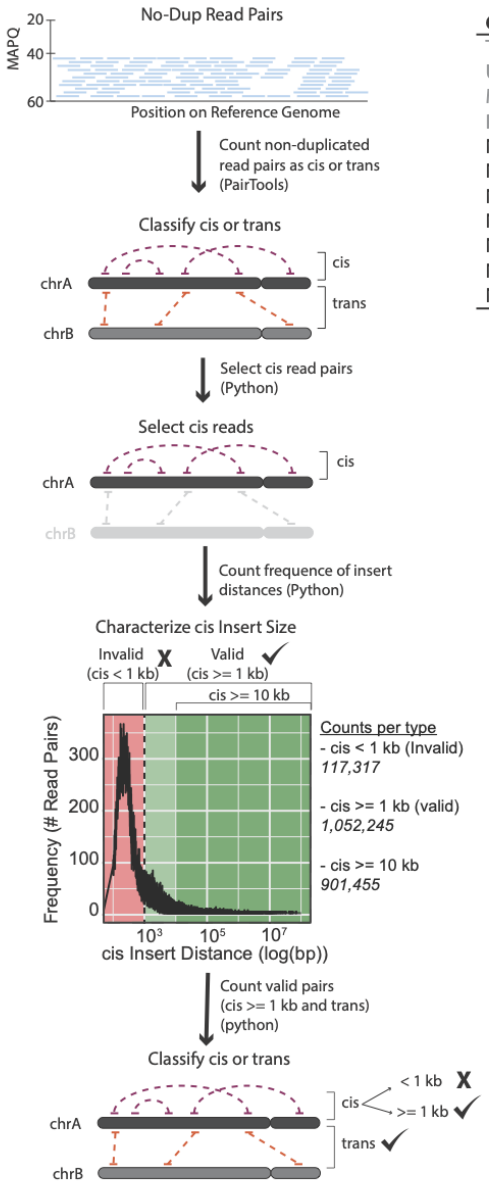
Category	Count	Percent
Total Read Pairs	2,000,000	100.00%
Unmapped Read Pairs	86,000	4.30%
Mapped Read Pairs	1,642,400	82.12%
PCR Dup Read Pairs	2,800	0.14%
No-Dup Read Pairs	1,640,800	81.98%
No-Dup Cis Read Pairs	1,169,563	71.28%
No-Dup Trans Read Pairs	471,317	28.72%
No-Dup Valid Read Pairs (cis >= 1 kb + trans)	1,523,483	92.85%
No-Dup Cis Read Pairs < 1kb	117,317	7.15%
No-Dup Cis Read Pairs >= 1kb	1,052,245	64.13%
No-Dup Cis Read Pairs >=10kb	901,455	54.94%

Proportion of Total Read Pairs



II. Classifying read pairs (cis or trans), characterizing insert size, and identifying valid pairs

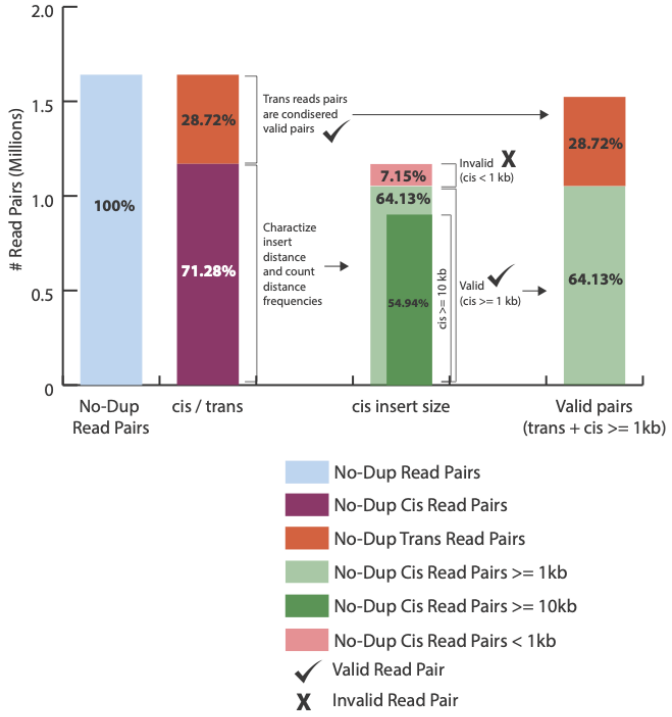
Process



Results

Category	Count	Percent
Total Read Pairs	2,000,000	100.00%
Unmapped Read Pairs	86,000	4.30%
Mapped Read Pairs	1,642,400	81.98%
PCR Dup Read Pairs	2,800	0.14%
No-Dup Read Pairs	1,640,800	82.04%
No-Dup Cis Read Pairs	1,169,563	71.28%
No-Dup Trans Read Pairs	471,317	28.72%
No-Dup Valid Read Pairs (cis >= 1 kb + trans)	1,523,483	92.85%
No-Dup Cis Read Pairs < 1kb	117,317	7.15%
No-Dup Cis Read Pairs >= 1kb	1,052,245	64.13%
No-Dup Cis Read Pairs >=10kb	901,455	54.94%

Proportion of No-Dup Read Pairs



Command:

```
python3 ./HiChiP/get_qc.py -p <stats.txt>
```

Example:

```
python3 ./HiChiP/get_qc.py -p stats.txt
```

After the script completes, it will print:

```
Total Read Pairs                              2,000,000  100%
Unmapped Read Pairs                           75,832     3.79%
Mapped Read Pairs                             1,722,285  86.11%
PCR Dup Read Pairs                            4,507      0.23%
No-Dup Read Pairs                             1,717,778  85.89%
No-Dup Cis Read Pairs                         1,385,238  80.64%
No-Dup Trans Read Pairs                       332,540    19.36%
No-Dup Valid Read Pairs (cis >= 1kb + trans)  875,804    50.98%
No-Dup Cis Read Pairs < 1kb                   841,974    49.02%
No-Dup Cis Read Pairs >= 1kb                  543,264    31.63%
No-Dup Cis Read Pairs >= 10kb                 193,061    11.24%
```

We consider a library prepared from a **mammalian** sample to be acceptable if: - Mapped nondupe pairs cis > 1,000 bp is greater than 20% of the total mapped No-Dup pairs.

1.4.2 ChiP enrichment

Calculating enrichment stats

Another key step in evaluating the quality of the HiChiP library is assessing the enrichment of HiChIP reads at protein binding sites, when protein binding sites correspond to a list of ChiP-Seq peaks.

Our QC pipeline supports as an input both peaks in a simple `bed` file format (containing three columns: chr, star, end) or ENCODE narrow peak format. For your convenience we include here *links* to some key examples of peak files from ENCODE ChiP-Seq experiments. All are of proteins for which Dovetail™ HiChIP MNase Kit has validated antibodies.

You can obtain gold-standards Chip-Seq peaks from databases, such as ENCODE, or generate your own list of peaks based on ChiP-Seq experiments, e.g. using MACS2.

To calculate stats of reads enrichment around ChIP peaks, we provide the `enrichment_stats.sh` script:

Reminder!

Did you remember to make the `enrichment_stats.sh` script executable?

If not, run the following command:

```
chmod +x ./HiChiP/enrichment_stats.sh
```

If you already ran this command, no need to run it again the execution permission is saved

---

Parameter	Function
-g	Input *genome file*
-b	Input *final bam file*
-p	Input (either in asimple bed format or narrow peak format)
-t	no. of threads
-x	Prefix for output file, enrichment stats will be saved to <prefix>_hichip_qc_metrics.txt

Command:

```
./HiChiP/enrichment_stats.sh -g <ref.genome> -b <mapped.PT.bam> -p <peaks.bed> -t <cores>
↪ -x <prefix>
```

Example:

```
./HiChiP/enrichment_stats.sh -g hg38.genome -b mapped.PT.bam -p ENCFF017XLW.bed -t 16 -x␣
↪CTCF
```

Tip!

If your peak file is zipped make sure to unzip it before running the `enrichment_stats.sh` script, e.g.:

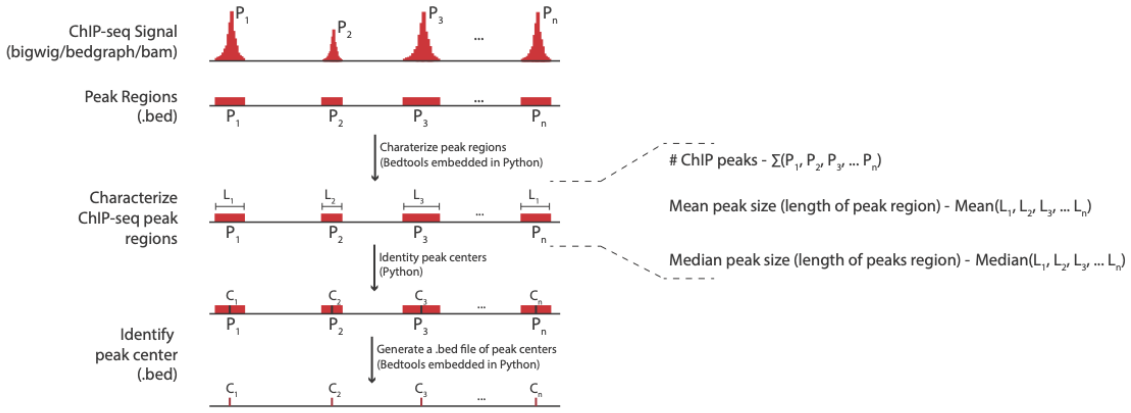
```
gunzip peak.bed.gz
```

In this example an output file *CTCF_hichip_qc_metrics.txt* will be created with the below information:

```
Total ChIP peaks                                                    41,017
Mean ChIP peak size                                                 309 bp
Median ChIP peak size                                               356 bp
Total reads in 500 bp around center of peaks                        321,368  7.91%
Total reads in 1000 bp around center of peaks                       458,843  11.3%
Total reads in 2000 bp around summits                               673,628  16.59%
Observed/Expected ratio for reads in 500 bp around center of peaks   11.92
Observed/Expected ratio for reads in 1000 bp around center of peaks  8.51
Observed/Expected ratio for reads in 2000 bp around center of peaks  6.25
```

The following image illustrates how enrichment around ChiP-Seq peaks is calculated:

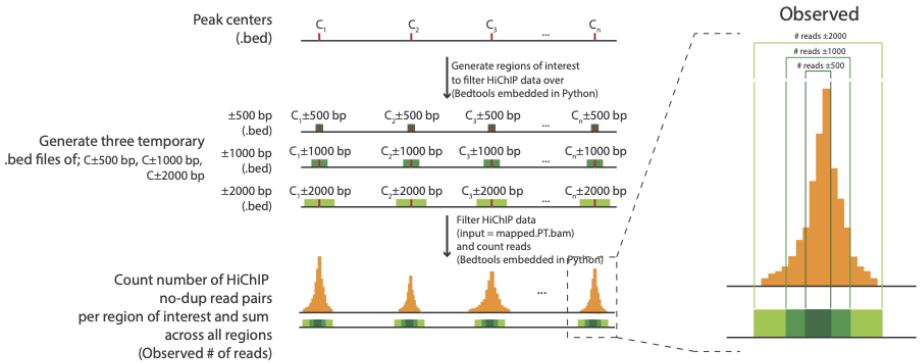
Step 1 - Count and characterize ChIP-seq peak regions (User provided or collected from ENCODE)



Category	Value	Percent
Total ChIP peaks	41,017	NA
Mean ChIP peak size	309 bp	NA
Median ChIP peak size	309 bp	NA
Total reads in 500 bp around center of peaks	393,163	9.46%
Total reads in 1000 bp around center of peaks	519,272	12.49%
Total reads in 2000 bp around center of peaks	692,305	16.66%
Observed/Expected ratio of reads in 500 bp around center of peaks	14.25	NA
Observed/Expected ratio of reads in 1000 bp around center of peaks	9.41	NA
Observed/Expected ratio of reads in 2000 bp around center of peaks	6.27	NA

Step 2 - Assess observed HiChIP coverage enrichment around peak centers and compare to expected value

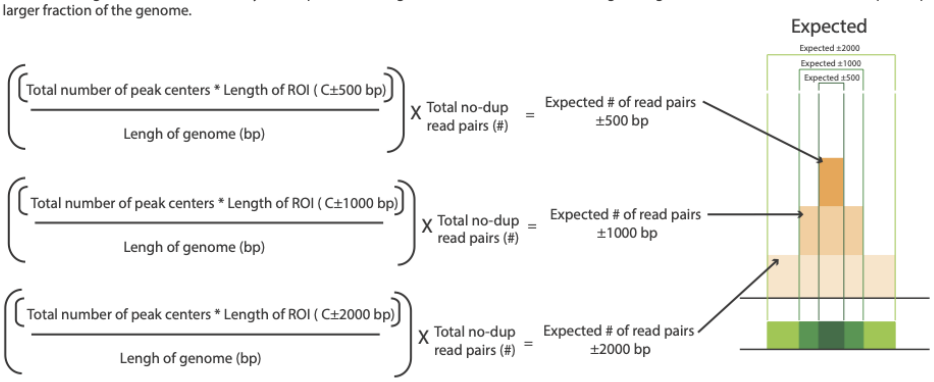
Step 2a - Count number of **observed** HiChIP read pairs in regions surrounding ChIP-seq peak centers



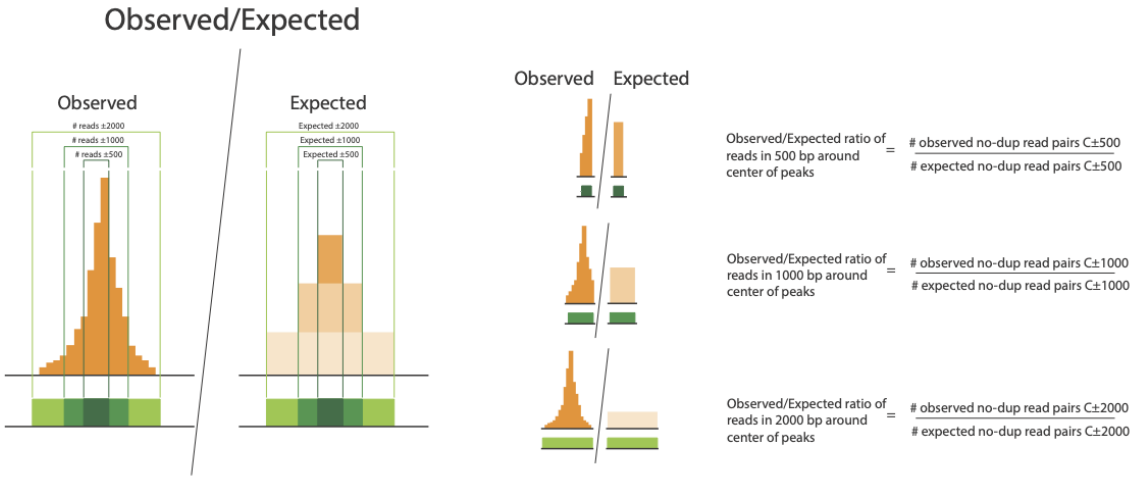
Category	Value	Percent
Total ChIP peaks	41,017	NA
Mean ChIP peak size	309 bp	NA
Median ChIP peak size	309 bp	NA
Total reads in 500 bp around center of peaks	393,163	9.46%
Total reads in 1000 bp around center of peaks	519,272	12.49%
Total reads in 2000 bp around center of peaks	692,305	16.66%
Observed/Expected ratio of reads in 500 bp around center of peaks	14.25	NA
Observed/Expected ratio of reads in 1000 bp around center of peaks	9.41	NA
Observed/Expected ratio of reads in 2000 bp around center of peaks	6.27	NA

Step 2b - Calculate the number of **expected** HiChIP read pairs in regions surrounding ChIP-seq peak centers

Expected value is calculated based onthe assumption that the total number of no-dup read pairs is evenly distrabuted across the ROI proportion of the genome. This is acheived by 1) determining what fraction of the genome length do the ROIs account for, then 2) evenly distrabuting the total no-dup read pairs across this fraction of the genome. Resulting in a constant coverage value at each bed entry. The expected coverage value decreases with increasing ROI legnth as the same number of no-dup read pairs are being dristrubted across a larger fraction of the genome.



Step 2c - Calculate the observed to expected ratio



Category	Value	Percent
Total ChIP peaks	41,017	NA
Mean ChIP peak size	309 bp	NA
Median ChIP peak size	309 bp	NA
Total reads in 500 bp around center of peaks	393,163	9.46%
Total reads in 1000 bp around center of peaks	519,272	12.49%
Total reads in 2000 bp around center of peaks	692,305	16.66%
Observed/Expected ratio of reads in 500 bp around center of peaks	14.25	NA
Observed/Expected ratio of reads in 1000 bp around center of peaks	9.41	NA
Observed/Expected ratio of reads in 2000 bp around center of peaks	6.27	NA

Plotting global enrichment around ChiP peaks

The `plot_chip_enrichment.py` and `plot_chip_enrichment_bed.py` scripts provide global evaluation of enrichment around known ChiP peaks. The script identifies the regions of ChiP peaks, sets a window of 1kb upstream and downstream of the peak's center, and based on the *bam file* of the valid pairs, calculates the aggregated read coverage within this window and plots the global fold coverage change based on the observed coverage divided by the mean coverage, as *illustrated*.

`plot_chip_enrichment.py` is intendent to be used when a `narrowPeak` file is available and `plot_chip_enrichment_bed.py` accept a simple `bed` file with peaks intervals as an input. Other than that, the two scripts accept the same parameters:

Parameter	Function
-bam	Input *final bam file*
-peaks	Input peaks in `narrowPeak` format (`plot_chip_enrichment.py`) or in simple chr,start,end `bed` format (plot_chip_enrichment_bed.py)
-output	ouptput file name to save the enrichment plot .png image

Command:

```
python3 plot_chip_enrichment.py -bam <mapped.PT.bam> -peaks <peaks.bed> -output
↪<enrichment.png>
```

or

```
python3 plot_chip_enrichment_bed.py -bam <mapped.PT.bam> -peaks <peaks.bed> -output
↪<enrichment.png>
```

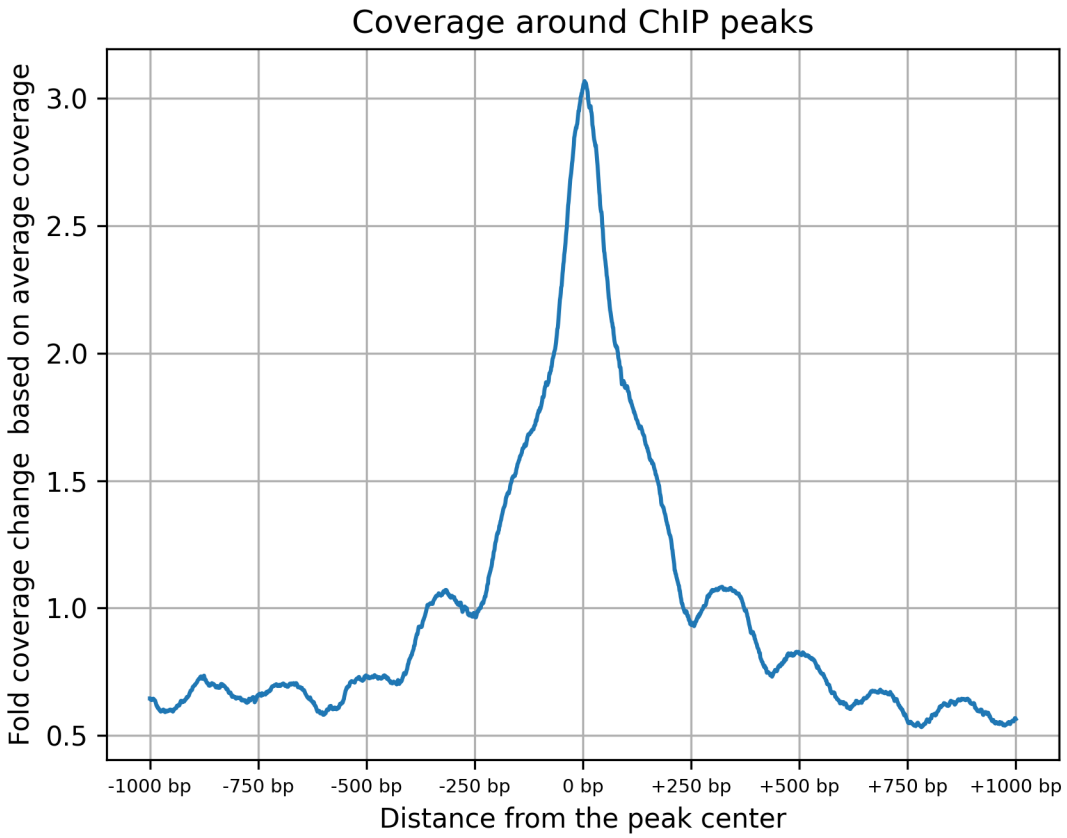
Example:

```
python3 ./HiChiP/plot_chip_enrichment.py -bam mapped.PT.bam -peaks ENCFF017XLW.bed -
↪output enrichment.png
```

or

```
python3 ./HiChiP/plot_chip_enrichment_bed.py -bam mapped.PT.bam -peaks peaks.bed -output␣
↪enrichment.png
```

Output plot:



Important!

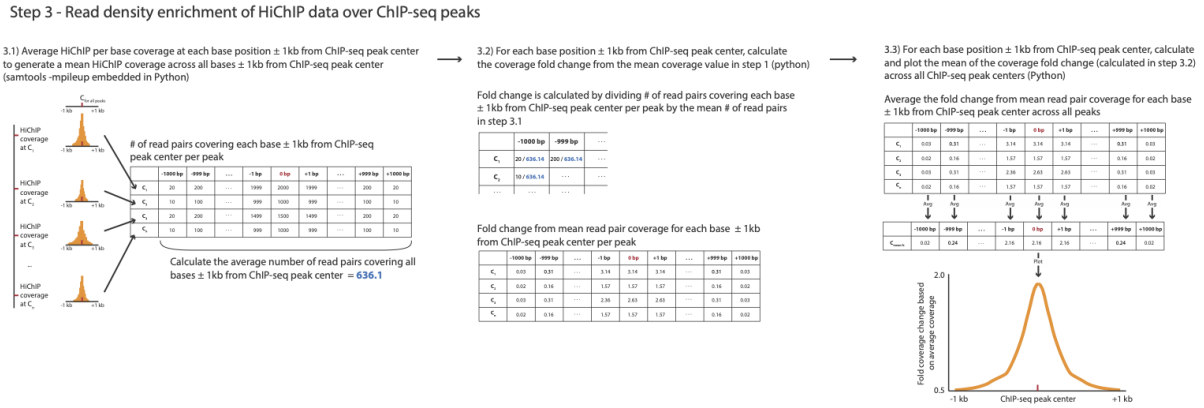
- `plot_chip_enrichment.py` will accept only `narrowPeak` format which has to include 10 columns, with the following specifications: - chromosome, start, end, in the three first columns - Peak Signal value at column #7 - Peak offset value at column #10 (when offset is the distance between the start position and the center of the peaks)
- If your peak file does not follow the above structure you can modify it into a simple bed file by extracting only the three first columns into a new file that can be used with the plot_chip_enrichment_bed.py script.
- `plot_chip_enrichment_bed.py` will accept only bed files with 3 columns. If your bed file includes more than three columns, extract the three first columns into a new file
- Example, how to extract only the first three columns:

```
cut -f1,2,3 input.bed > output.bed
```

There are two minor differences between the two scripts:

- `plot_chip_enrichment.py` calculates the center of the peak according to `start` + `offset` `plot_chip_enrichment_bed.py` chooses the center of the peak as the middle point between `start` and `end`. Both will calculate the aggregated enrichment -1kb and +1kb of the center of the peak (no matter the legnth of the peak)
- All intervals in the bed files are used for the meta-analysis when `plot_chip_enrichment_bed.py` is used `narrowPeak` format includes information on peak signal, this information is used to filter out peaks with extreme

values (either very low or very high signals) prior to meta-analysis



1.4.3 QC Assessment

Pass/No Pass Metrics

Now that you have successfully completed the QC scripts, it is time to determine if the HiChIP library is of high quality. The QC metrics calculated above can be distilled down to three key quantitative metrics and one qualitative step to help you assess the quality of the library before proceeding into deep sequencing.

Output from get_qc.py

```
Total Read Pairs                                2,000,000  100%
Unmapped Read Pairs                             75,832     3.79%
Mapped Read Pairs                               1,722,285  86.11%
PCR Dup Read Pairs                              4,507      0.23%
1. No-Dup Read Pairs                            1,717,778  85.89%
   No-Dup Cis Read Pairs                        1,385,238  80.64%
   No-Dup Trans Read Pairs                      332,540    19.36%
   No-Dup Valid Read Pairs (cis >= 1kb + trans) 875,804    50.98%
   No-Dup Cis Read Pairs < 1kb                  841,974    49.02%
2. No-Dup Cis Read Pairs >= 1kb                 543,264    31.63%
   No-Dup Cis Read Pairs >= 10kb                193,061    11.24%
```

Output from enrichment_stats.sh

```
Total ChIP peaks                                                41,017
Mean ChIP peak size                                             309 bp
Median ChIP peak size                                           356 bp
Total reads in 500 bp around center of peaks                    321,368  7.91%
3. Total reads in 1000 bp around center of peaks                458,843  11.3%
Total reads in 2000 bp around summits                           673,628  16.59%
Observed/Expected ratio for reads in 500 bp around center of peaks  11.92
Observed/Expected ratio for reads in 1000 bp around center of peaks 8.51
Observed/Expected ratio for reads in 2000 bp around center of peaks 6.25
```

1. No-Dup Read Pairs – This value is reflective of the alignment rate and PCR duplication rate. It should be noted that this value scales inversely with sequencing depth.
2. No-dup cis read pairs 1kb – This value demonstrates that the proximity-ligation step was successful, and the majority of the data are useful in downstream analyses (e.g. loop calling). This value can be dependent on the protein of interest, for example CTCF has a very long-range contact profile while POL2A has a much more localized contact domain. The cut-off used here is applicable across different protein targets.
3. Total reads in 1000 bp around center of peaks – This value demonstrates that chromatin enrichment was successful. This metric is very similar to Fraction of Reads in Peaks (FRiP) score that is used to assess ChIP-seq data as defined by ENCODE. Our defined cut-off value is slightly more stringent than the ENCODE standard of 1%. It should be noted that this value is dependent on the peak set used, the value can be underestimated if you used a peak set that is not reflective of your experiment (e.g publicly available).
4. Visual assessment of HiChIP coverage in IGV – This step provides you a visual peace of mind that your IP-enrichment was successful. Alignments (.bam) should be converted into bigwig format with deepTools bamCoverage https://deeptools.readthedocs.io/en/develop/content/tools/bamCoverage.html).

While the QC process can be boiled down to these key values, the remaining values of the QC process are used to diagnose and troubleshoot a library that falls into the "No Pass" category. Therefore, it is important to generate all the values in the QC process in case there is a need for troubleshooting.

Pass/No Pass Values

The table below summarizes the minimum passing values for the metrics defined above. The cut-off values were determined for both shallow sequenced (20 million read pairs 2 x 150 bp) and deep sequenced data (100-200 Million read pairs 2 x 150 bp), as the percentage of mapped no-dup pairs changes with the sequencing depth.

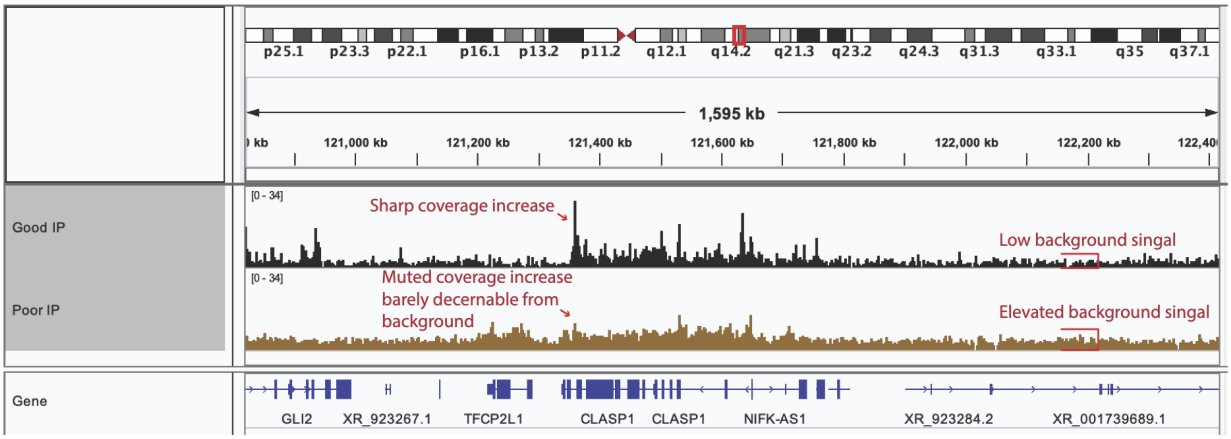
Metric	Shallow Seq (20M)	Deep Seq (100-200M)
No-Dup Read Pairs	>75%	>50%
No-dup cis read pairs 1kb	>20%	>20%
Total reads in 1000 bp around center of peaks	>2%	>2%

Visual Inspection Of The Alignments

Once you have compared your library QC values to the minimal quantitative requirements for a library to pass QC, you can move on visual assessment in IGV. Here we used the Integrated Genome Viewer (which can be downloaded an installed here). IGV is standard genome browser for visualizing NGS data in track format. Simply load your bigwigs into IGV then zoom in to a 1-2 Mbp window. In this step, we are looking to see if the data suggest that there has been enrichment.

- Good IP – exhibit distinct signals of sharply increased coverage from a low background indicating the location of the protein-DNA binding Site.
- Poor IP – exhibit no or weak coverage increases and are often accompanied by an elevated background signal.

Below is an annotated screenshot from IGV showing examples of both good and bad IP of shallow sequenced (20 M read pairs) libraries. The library exhibiting good IP characteristics (top track in black) shows clear, sharp coverage enrichment, and low background signal, where the library with poor IP, (bottom track in brown) has a high background signal and muted coverage enrichment.



Final Determination

If your libraries pass the minimum threshold for each of the 3 quantitative metrics, and the visual inspection:

- For shallow sequenced libraries - proceed to deep sequencing (~150 M read pairs per library)
- For deep sequencing – proceed with downstream analyses

If the libraries fail one or more of the 3 quantitative metrics or the visual inspection - please reach out to our support team at: support@dovetail-genomics.com

1.5 Generating Contact Matrix

There are two common formats for contact maps, the Cooler format and Hic format. Both are compressed and sparsed formats to avoid large storage volumes; For a given n number of bins in the genome, the size of the matrix would be n^2, in addition, typically more than one resolution (bin size) is being used.

In this section we will guide you on how to generate both matrices types, *HiC* and *cool* based on the *.pairs file* that you generated in the *previous section* and how to visualize them.

1.5.1 Generating `HiC` contact maps using Juicer tools

Additional Dependencies

- Juicer Tools - Download the JAR file for juicertools and place it in the same directory as this repository and name it as `juicertools.jar`. You can find the link to the most recent version of Juicer tools here e.g.:

```
wget https://s3.amazonaws.com/hicfiles.tc4ga.com/public/juicer/juicer_tools_1.22.01.jar
mv juicer_tools_1.22.01.jar ./HiChiP/juicertools.jar
```

- Java - If not already installed, you can install Java as follows:

```
sudo apt install default-jre
```

From `.pairs` to `.hic` contact matrix

- Juicer Tools is used to convert `.pairs` file into a HiC contact matrix.
- `HiC` is highly compressed binary representation of the contact matrix
- Provides rapid random access to any genomic region matrix
- Stores contact matrix at 9 different resolutions (2.5M, 1M, 500K, 250K, 100K, 50K, 25K, 10K, and 5K)
- Can be programmatically manipulated using straw python API

The *.pairs* file that you generated in the *From fastq to final valid pairs bam file* section can be used directly with `Juicer tools` to generate the *HiC* contact matrix:

Parameter	Function
-Xmx	The flag Xmx specifies the maximum memory allocation pool for a Java virtual machine, from our experience 48000m works well when processing human data sets. If you are not sure how much memory your system has, run the command `free -h` and check the value under *total*
Djava.awt.headless=true	Java is ran in a headless mode when the application does not interact with a user (if not specified, the default is Djava.awt.headless=false)
pre	The pre command allows users to create .hic files from their own data
–threads	Specifies the numbers of threads to be used (integer number)
*.pairs or *.pairs.gz	input file for generating the contact matrix
*.genome	genome file, listing the chromosomes and their sizes
*.hic	hic output file, containing the contact matrix

Tip no.1

Please note that if you have an older vesrion of `Juicer tools`, generating contact map directly from `.pairs` file may not be supported. We recommend updating to a newer version. As we tested, the `pre` utility of the version 1.22.01 support the .pairs to HiC function.

Command:

```
java -Xmx<memory>  -Djava.awt.headless=true -jar <path_to_juicer_tools.jar> pre --
↪threads <no_of_threads> <mapped.pairs> <contact-map.hic> <ref.genome>
```

Example:

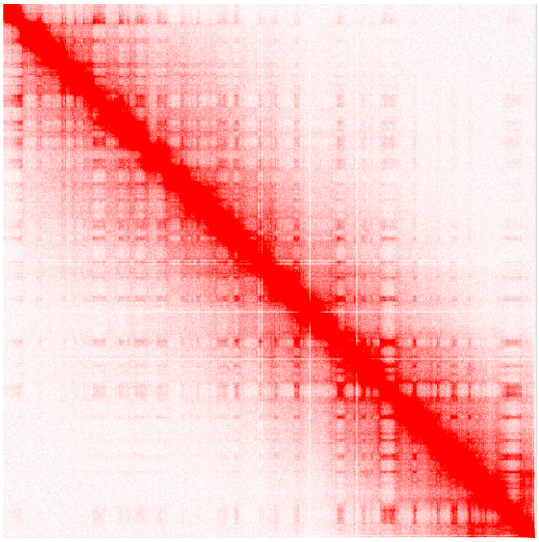
```
java -Xmx48000m  -Djava.awt.headless=true -jar ./HiChiP/juicertools.jar pre --threads 16␣
↪mapped.pairs contact_map.hic hg38.genome
```

Tip no.2

`Juicer tools` offers additional functions that were not discussed here, including matrix normalization and generating matrix for only specified regions in the genome. To learn more about advanced options, please refer to the Juicer Tools documentation.

Visualizing `.hic` contact matrix

The visualization tool `Juicebox` can be used to visualize the contact matrix. You can either download a local version of the tool to your computer as a Java application or use a web version of Juicebox. Load your `.hic` file to visualize the contact map and zoom in to areas of interest.



1.5.2 Generating `cooler` contact maps

Additional Dependencies

Installing Cooler and its dependencies

- libhdf5 - `sudo apt-get install libhdf5-dev`
- h5py - `pip3 install h5py`
- cooler - `pip3 install cooler`

For any issues with `cooler` installation or its dependencies, please refer to the cooler installation documentation

Installing Pairix

Pairix is a tool for indexing and querying on a block-compressed text file containing pairs of genomic coordinates. You can install it directly from its github repository as follows:

```
git clone https://github.com/4dn-dcic/pairix
cd pairix
make
```

Add the bin path, and utils path to PATH and exit the folder:

```
PATH=~/pairix/bin/:~/pairix/util:~/pairix/bin/pairix:$PATH
cd ..
```

Important!

make sure to modify the following example with the path to your *pairix* installation folder. If you are not sure what is the path you can check it with the command *pwd* when located in the *pairix* folder.

For any issues with `pairix`, please refer to the pairix documentation

From `.pairs` to `cooler` contact matrix

- Cooler tools is used to convert **indexed** `.pairs` file into cool and mcool contact matrices
- `Cooler` generates a sparse, compressed, and binary persistent representation of proximity ligation contact matrix
- Store matrix as HDF5 file object
- Provides python API to manipulate contact matrix
- Each cooler matrix is computed at a specific resolution
- Multi-cool (mcool) files store a set of cooler files into a single HDF5 file object
- Multi-cool files are helpful for visualization

Indexing the `.pairs` file

We will use the `cload pairix` utility of `Cooler` to generate contact maps. This utility requires the `.pairs` file to be indexed. `Pairix` is used for indexing compressed `.pairs` files. The files should be compresses with bgzip (which should already be installed on your machine). If your `.pairs` file is not yet bgzip compressed, first compress it as follows:

Command:

```
bgzip <mapped.pairs>
```

Example:

```
bgzip mapped.pairs
```

Following this command `mapped.pairs` will be replaced with its compressed form `mapped.pairs.gz`

Note!

Compressing the `.pairs` file with `gzip` instead of `bgzip` will also result in a compressed file with the `.gz` suffix, but due to format differnces it will not be accepted as an input for `pairix`.

Next, index the file `.pairs.gz` file:

Command:

```
pairix <mapped.pairs.gz>
```

Example:

```
pairix mapped.pairs.gz
```

Genereting single resolution contact map files

As mentioned above, we will use the `cload pairix` utility of `Cooler` to generate contact maps:

`cooler cload pairix` usage:

Parameter	Function
<genome_fils>:<bin size>	Specifies the reference *.genome file*, followed with``:`` and the desired bin size in bp
-p	Number of processes to split the work between (integer), default: 8
*.pairs.gz	Path to `bgzip` compressed and indexed `.pairs` file
*.cool	Name of output file

Command:

```
cooler cload pairix -p <cores> <ref.genome>:<bin_size_in_bp> <mapped.pairs.gz> <matrix.
↪cool>
```

Example:

```
cooler cload pairix -p 16 hg38.genome:1000 mapped.pairs.gz matrix_1kb.cool
```

Genereting multi-resolutions files and visualizing the contact matrix

When you wish to visualize the contact matrix, it is highly recommended to generate a multi-resolution `.mcool` file to allow zooming in and out to inspect regions of interest. The cooler `zoomify` utility allows you to generate a multi-resolution cooler file by coarsening. The input to `cooler zoomify` is a single resolution `.cool` file, to allow zooming in into regoins of interest we suggest to generate a `.cool` file with a small bin size, e.g. 1kb. Multi-resolution files uses the suffix `.mcool`.

`cooler zoomify` usage:

Parameter	Function
–balance	Apply balancing to each zoom level. Off by default
-p	Number of processes to use for batch processing chunks of pixels, default: 1
*.cool	Name of contact matrix input file

*Command:**

```
cooler zoomify --balance -p <cores> <matrix.cool>
```

Example:

```
cooler zoomify --balance -p 16 matrix_1kb.cool
```

The example above will result in a new file named *matrix_1kb.mcool* (no need to specify output name)

Tip

`Cooler` offers additional functions that were not discussed here, including generating a cooler from a pre-binned matrix, matrix normalization and more. To learn more about advanced options, please refer to the cooler documentation

---

HiGlass is an interactive tool for visualizing `.mcool` files. To learn more about how to set up and use HiGlass follow the HiGlass tutorial

1.6 HiChIP Loop Calling

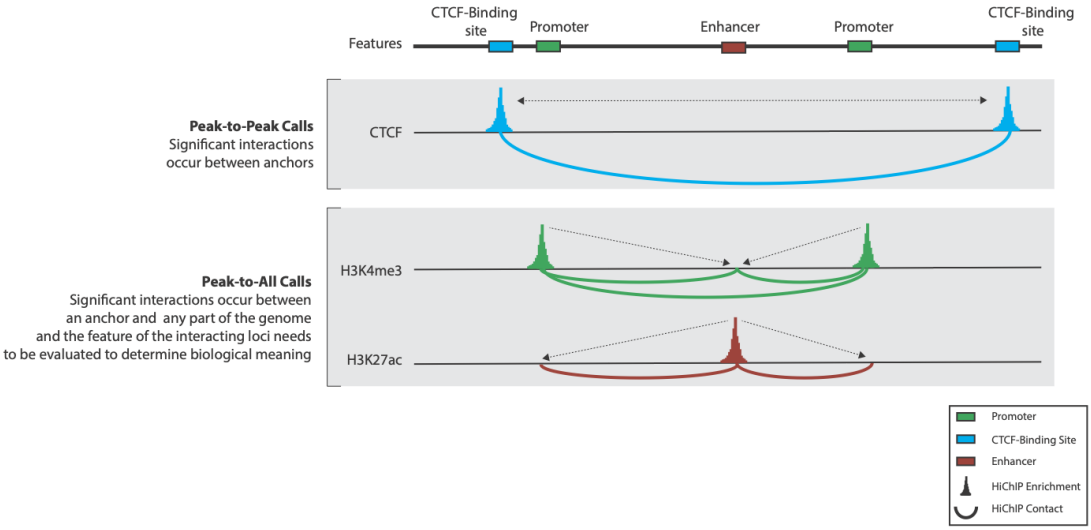
1.6.1 Introduction

This workflow is a simple guide to identify loops in HiChIP data. Before you get started please read this short introduction which will help you better understand what loops are in the context of HiChIP assays and why we're going to focus on FitHiChIP tool as the tool to use. I would like preface this work by saying there is no "one correct way" to analyze HiChIP data. This is just an example workflow that will enable you identify significant interactions in HiChIP data. The biological implications of those interactions should be interpreted through the lenses of the protein target or biological question you're asking!

What are chromatin Loops in the context of HiChIP?

- HiChIP loops are - significant interactions between a protein-anchor and the surrounding genome.
- The biological interpretations of the interactions is based on the protein of interest for example:
  - CTCF – Identifying actual chromatin loops. Meaning, CTCF-cohesin mediated looping.
  - H3K27ac or H3K4me3 – means identifying regions that interact with the enhancer or active promotor marker respectively. These interactions do not necessarily reflect the canonical loop formation but could reflect short range folding.
- As these interactions can reflect more than just canonical "loops", they will simply be referred to as significant interactions for the rest of the documentation.
- Types of significant HiChIP interactions
  - Peak-to-Peak - ChIP-seq anchors will only interact with other anchors like CTCF chromatin loops
  - Peak-to-All - ChIP-seq anchors will interact with any of the surrounding genome weither or those regions correspond with an anchor site, like H3K27ac, H3K4me3, or PolII
  - All-to-All - This is for a non-targeted Hi-C data contact matrix and will not be used here.

HiChIP interactions type:



1.6.2 Resolution

Resolution can play an odd roll in HiChIP significant interaction detection. Typically, only 5-10 reads-pairs per interaction are required to statistically identify an interaction with HiChIP as opposed to the 100-1000 read-pairs required for non-targeted Hi-C assays. However, many loop callers work on contact matrices which require binning at a specified resolution(s) to build. Typically, 1 kb, 5 kb, or 10 kb for HiChIP data. As such, the fewer reads you have, the larger the bin size must be to have enough read support to run statistics on.

Additionally, the biological nature of the protein target can impact the resolution of that you are interested in looking at. Proteins with a larger footprint (PolII) might require larger bin size, where smaller footprints (CTCF) might need a smaller bin. What becomes problematic, is when the bin size is so large that many anchors are captured in a single bin. This is important to keep in mind.

With all that being said, most HiChIP/ChIA-PET analyses are conducted between 2.5 - 5 kb. Our best recommendation is to try calling significant interactions at different resolutions. Generate lists of significant interactions at multiple resolutions and filter to keep only unique entries.

1.6.3 Tool landscape

There are many tools available to identify significant interactions. Below is a table that outlines just a subset of tools, where to get them, requirement to specify a resolution, ability to select the type of interaction, and input file structure of the HiChIP data. They all have their own pro's and con's, but there is no clearly established way to analyze HiChIP data, and it largely depends on your biological questions. So always keep that in mind and make sure the tool you're using makes sense with the biological question you are asking!

Tool	Repo	Input format	Resolution Option	Considers type of interaction
FitHiChIP	https://ay-lab.github.io/FitHiChIP/	HiC-Pro Valid-Pairs	Yes	Yes
Cloops	https://github.com/YaqiangCao/cLoops	Bedpe	No	No
HiChIP-PER	https://github.com/aryeelab/hichipper	HiC-Pro Valid-Pairs	Yes	No
HiC-CUPs	https://github.com/aidenlab/juicer/wiki/HiCCUPS	.hic	Yes	No

1.6.4 Why FitHiChIP?

We have chosen `FitHiChIP` for this workflow for a few reasons:

1. The install is very easy, and it can manage all the dependencies through Docker or Singularity (if you don't have sudo privileges)
2. It is very flexible in term of input, `.pairs`, or interaction tabel in `.bedpe` format.
3. Has the ability to select bias type
4. Can specify the type of interaction to assess
5. Output is easily integrate-able to other workflows

Below is an annotated configuration file with some of the key parameters to consider

```
#====================================
# Sample configuration file for running FitHiChIP
#====================================

#*****************************
# important parameters
#*****************************

# File containing the valid pairs from HiCPro pipeline
# Can be either a text file, or a gzipped text file
ValidPairs=./TestData/Sample_ValidPairs.txt.gz

# File containing the bin intervals (according to a specified bin size)
# which is an output of HiC-pro pipeline
# If not provided, this is computed from the parameter 1
Interval=

# File storing the contact matrix (output of HiC-pro pipeline)
# should be accompanied with the parameter 2
# if not specified, computed from the parameter 1
Matrix=

# Pre-computed locus pair file
# of the format:
# chr1  start1  end1    chr2    start2  end2    contactcounts
Bed=


# File containing reference ChIP-seq / HiChIP peaks (in .bed format)
# mandatory parameter
PeakFile=./TestData/Sample.Peaks.gz

# Output base directory under which all results will be stored
OutDir=./TestData/results/

#Interaction type - 1: peak to peak 2: peak to non peak 3: peak to all (default) 4: all to all 5: everything from 1 to 4.
IntType=3

# Size of the bins [default = 5000], in bases, for detecting the interactions.
BINSIZE=5000

# Lower distance threshold of interaction between two segments
# (default = 20000 or 20 Kb)
LowDistThr=20000

# Upper distance threshold of interaction between two segments
# (default = 2000000 or 2 Mb)
UppDistThr=2000000

# Applicable only for peak to all output interactions - values: 0 / 1
# if 1, uses only peak to peak loops for background modeling - corresponds to FitHiChIP(S)
# if 0, uses both peak to peak and peak to nonpeak loops for background modeling - corresponds to FitHiChIP(L)
UseP2PBackgrnd=1

# parameter signifying the type of bias vector - values: 1 / 2
# 1: coverage bias regression   2: ICE bias regression
BiasType=1

# following parameter, if 1, means that merge filtering (corresponding to either FitHiChIP(L+M) or FitHiChIP(S+M))
# depending on the background model, would be employed. Otherwise (if 0), no merge filtering is employed. Default: 1
MergeInt=1

# FDR (q-value) threshold for loop significance
QVALUE=0.01

# File containing chromomosome size values corresponding to the reference genome.
ChrSizeFile=./TestData/chrom_hg19.sizes

# prefix string of all the output files (Default = 'FitHiChIP')
PREFIX=FitHiChIP

# Binary variable 1/0: if 1, overwrites any existing output file. otherwise (0), does not overwrite any output file.
OverWrite=1
```

full path to HiC-Pro format valid pairs file

full path to ChIP-seq peaks

full path to output directory

Interaction type:
1 - peak to peak (e.g. CTCF)
2 - peak to non-peak (e.g. PolII)
3 - peak to all (H3K4me3, H3K27ac)
4 - All to all (Hi-C data)
5 - run everything

Resolution:
2.5 - 5 Kb is a good start

Lower distance:
This depends on the anchor in question -
CTCF 20 kb is a good start, H3K27ac or PolII a lowering10 kb is good idea

Stringency:
1 - FitHiChIP(S) - More stringent (peak to peak)
2 - FitHChIP(L) - Less stringent (peak to all, peak to non-peak)

Bias factor
1 - Coverage (reccomended)
2 - ICE

full path to .genome file

prefix file name for results files

1.6.5 Input files

This workflow assumes you have completed the *Step-by-step guide to Process HiChIP data*. The two key files required are:

- *Filtered Pairs file* - output from *From fastq to final valid pairs* workflow.
- Bed file of ChIP-seq anchors for your protein of interest, e.g. as you used in the *QC step*. We included in the *datasets section* links to some useful ChIP-seq bed files from the Encode project.

Testing!

If you are looking for a dataset to practice this walkthrough, I reccomend the GM12878 CTCF (deep sequencing) from our publicaly available *datasets*

1.6.6 Tools

- FitHiChIP
  - Full documentation is here
  - I recommend you read it! It is very well documented!

1.6.7 Workflow Overview

- *Convert filtered pairs file to Hi-C Pro valid pairs format.*
- *Modify FitHiChIP configuration file*
- *Run FitHiChIP through docker* - FitHiChIP is a single executable that:
  - Builds a table of interactions (bedpe-like version of a contact matrix)
  - Corrects for biases (coverage or ICE)
  - Filters data for the type of interactions (Peak-to-Peak, Peak-to-All, or All-to-All)
  - Builds a contact frequency to insert distance model from the filtered interactions.
  - Assigns P-values and Q-values (false discovery rate - FDR) to interactions.
  - Will merge near-by interaction that pass a Q-value threshold.
  - Report a bedpe-like file of total and merged interactions filtered by a Q value.
- *Inspect the output report*

1.6.8 Workflow

- Convert filtered pairs file to Hi-C Pro valid pairs format

Command:

```
grep -v '#' <*.pairs>| awk -F"\t" '{print $1"\t"$2"\t"$3"\t"$6"\t"$4"\t"$5"\t"$7}' |␣
↪gzip -c > <output.pairs.gz>
```

Example:

```
grep -v '#' mapped.pairs| awk -F"\t" '{print $1"\t"$2"\t"$3"\t"$6"\t"$4"\t"$5"\t"$7}' |␣
↪gzip -c > hicpro_mapped.pairs.gz
```

- Modify the `configuration file` to desired specifications:
  - We'll be using coverage bias because these data are MNase based, not RE-based
  - If using CTCT use Peak-to-Peak as outlined earlier, CTCF data is a peak to peak interaction, other protein like H3K27ac and H3K4me3 you're going to want to use Peak-to-All.

Adjusting the configuration file . Entries that need to be adjusted are highlighted:

```
#=====================================
# Sample configuration file for running FitHiChIP
#=====================================
#*****************************
# important parameters
#*****************************
# File containing the valid pairs from HiCPro pipeline
# Can be either a text file, or a gzipped text file
ValidPairs=/path_to_hicpro_pairs/prefix.hicpro.valid.pairs.gz
# File containing the bin intervals (according to a specified bin size)
# which is an output of HiC-pro pipeline
# If not provided, this is computed from the parameter 1
Interval=
# File storing the contact matrix (output of HiC-pro pipeline)
# should be accompanied with the parameter 2
# if not specified, computed from the parameter 1
Matrix=
# Pre-computed locus pair file
# of the format:
# chr1  start1  end1    chr2    start2  end2    contactcounts
Bed=
# File containing reference ChIP-seq / HiChIP peaks (in .bed format)
# mandatory parameter
PeakFile=/path_to_ChIP_peaks/peaks.bed
# Output base directory under which all results will be stored
OutDir=/path_to_output/fithichip_test_1kb
#Interaction type - 1: peak to peak 2: peak to non peak 3: peak to all (default) 4: all␣
↪to all 5: everything from 1 to 4.
IntType=1
# Size of the bins [default = 5000], in bases, for detecting the interactions.
BINSIZE=2500
# Lower distance threshold of interaction between two segments
# (default = 20000 or 20 Kb)
LowDistThr=20000
# Upper distance threshold of interaction between two segments
# (default = 2000000 or 2 Mb)
UppDistThr=2000000
# Applicable only for peak to all output interactions - values: 0 / 1
# if 1, uses only peak to peak loops for background modeling - corresponds to␣
↪FitHiChIP(S)
# if 0, uses both peak to peak and peak to nonpeak loops for background modeling -␣
↪corresponds to FitHiChIP(L)
```

(continues on next page)

(continued from previous page)

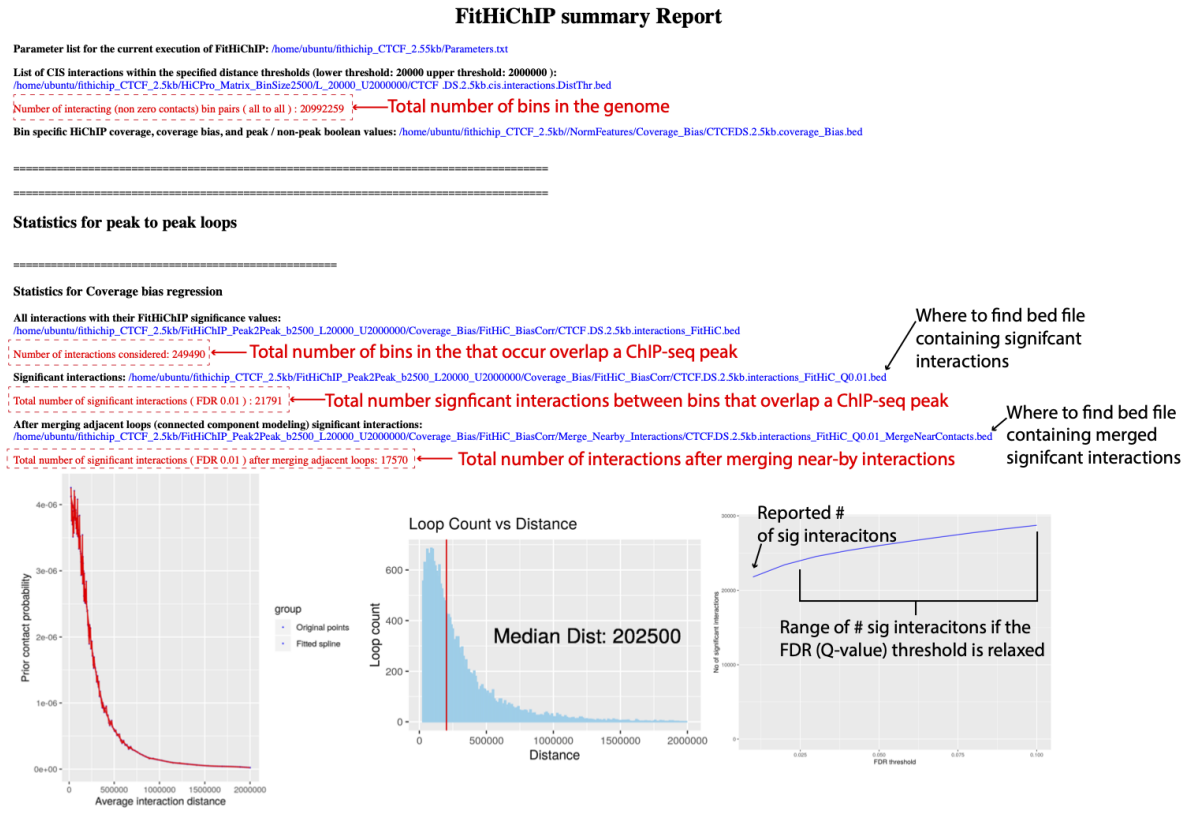
```
UseP2PBackgrnd=1
# parameter signifying the type of bias vector - values: 1 / 2
# 1: coverage bias regression   2: ICE bias regression
BiasType=1
# following parameter, if 1, means that merge filtering (corresponding to either␣
↪FitHiChIP(L+M) or FitHiChIP(S+M))
# depending on the background model, would be employed. Otherwise (if 0), no merge␣
↪filtering is employed. Default: 1
MergeInt=1
# FDR (q-value) threshold for loop significance
QVALUE=0.01
# File containing chromomosome size values corresponding to the reference genome.
ChrSizeFile=/path_to_genome_file/hg38.genome
# prefix string of all the output files (Default = 'FitHiChIP').
PREFIX=prefix.2.5kb
# Binary variable 1/0: if 1, overwrites any existing output file. otherwise (0), does␣
↪not overwrite any output file.
OverWrite=1
```

- Run FitHiChIP through docker

  Command:

  ```
  FitHiChIP_Docker.sh -C config.txt
  ```

- Inspect the report



1.6.9 Output

FitHiChIP merged interactions output

1.6.10 What if?

- I don't have a bed file of ChIP-seq anchors or I can't find a representative bed file for my antibody or sample type?
  - Follow our guide to *calling 1-Demensional peaks with HiChIP data using MACS2*
- I want to use a different tool to identify significant interactions.
  - That is great! This is just one way please refer to tool you'd like to use for documentation. This is just one example of how to find significant interactions in HiChIP data. The key things to consider are the input formats of the data the tool requests.
- I need to do differential analyses.
  - The output of this workflow is nice because the output is a bed file and if you have two samples one could just do a `bedtools intersect` to classify interactions as shared or unique to each sample.

1.6.11 What next?

- Visualization
  - Continue with *plotting HiChIP interactions in R*
  - Import to the Wash-U epigenome browser (more information in this link)

1.7 HiChIP Comparative Analyses

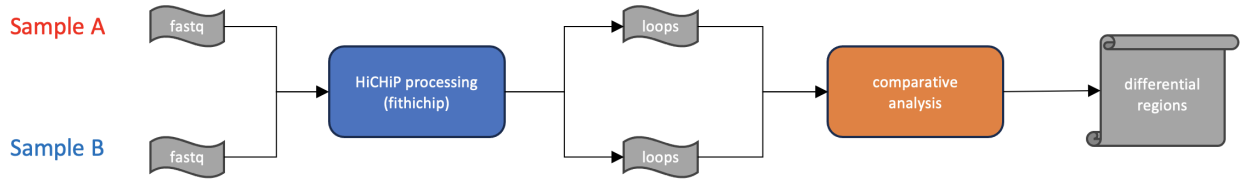
1.7.1 Introduction

Biological questions are seldom answered by analysing single samples in isolation. It is often the case that an experiment aims to make comparisons between two (or more) biological conditions, such as:

1) Untreated wild type vs treatment
2) Wild type vs knockout
3) Normal sample vs tumor

In all cases the goal is to produce a list of differentially interacting regions in one condition relative to the other. The main output for comparitive analses is analogous to what is expeected for differential gene expression, where the primary result is a table of regions, the fold change between conditions, and a statistical measure of signficance. For HiCHiP, the unit for comparison are the loop calls identified using the FitHiChIP software as described in *previous steps*.

Figure 1:



1.7.2 Differential Analysis

Question: How do I perform differential analyses for HiChIP?

Process: Results files from FitHiChIP are used to construct a differential design, and comparison is performed using the scripts bundled with fithichip software.

Results: Final results consist of a table of differentially interacting regions, fold change, and measure of statistical signficance.

Files and tools needed:

- FitHiChIP loop calls for each condition: *PREFIX*.interactions_FitHiC.bed
- FitHiChIP differential analysis software and scripts
- Associated ChIP-seq peak files [optional]

As the design of differential analysis experiments are unique to each biological question, there are multiple possibilites for how the analysis can be set up. A common scenario is to compare two conditions where each condition has two replicates, and is described in the *FitHiChIP documentation pages*.

Interpreting results:

FitHiChIP differential analysis produces a number of intermediate in addition to the final results table. The most important is the list of significant loops and is named "Loops_EdgeR_Default_SIG.bed". In general, the interpretation of differential loop analysis is the same as what is familiar for gene expression analysis, where intereactions can be prioritized based on the fold change and statistical significance. An example output file is given below.

chr1	start1	end1	chr2	start2	end2	group1_RawCC	group1_QVal	group2_RawCC	group2_QVal	logFC	logCPM	PValue	FDR	group1_SigRepl	group2_SigRepl
chr1	930000	940000	chr1	940000	950000	0	1	37	0.8722	8.2960	5.4070	0.0000	0.0022	0	0
chr1	940000	950000	chr1	950000	960000	0	1	40	0.9991	8.4081	5.5105	0.0000	0.0016	0	0
chr1	1020000	1030000	chr1	1030000	1040000	0	1	43	1	8.5122	5.6071	0.0000	0.0012	0	0
chr1	1030000	1040000	chr1	1040000	1050000	0	1	55	0.872	8.8664	5.9403	0.0000	0.0004	0	0
chr1	1060000	1070000	chr1	1070000	1080000	0	1	48	0.9865	8.6705	5.7550	0.0000	0.0007	0	0
chr1	1070000	1080000	chr1	1080000	1090000	0	1	25	0.8722	7.7326	4.8990	0.0003	0.0127	0	0
chr1	1140000	1150000	chr1	1150000	1160000	0	1	30	0.9681	7.9945	5.1326	0.0001	0.0052	0	0
chr1	1150000	1160000	chr1	1160000	1170000	0	1	40	0.9984	8.4081	5.5105	0.0000	0.0016	0	0
chr1	1250000	1260000	chr1	1260000	1270000	28	0.8747	0	1	-7.7365	4.9817	0.0002	0.0109	0	0

The most relevant fields from the output will be:

- logFC – the log fold change in coverage between the two conditions
- FDR – a p-value, after correction for multiple hypothesis testing, on the statistical signficance of the observed fold change

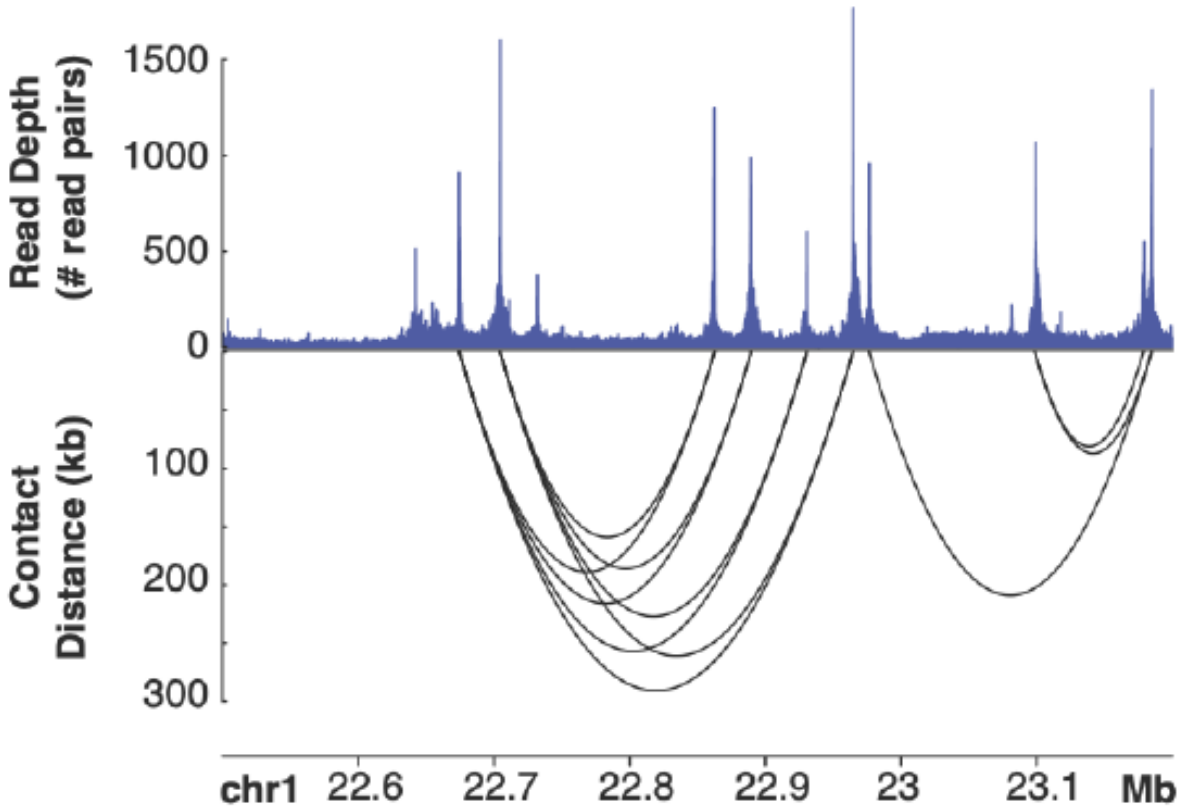
Considerations:

- Replication – It is generally advisable to have technical replicates for differential analyses, as this will produce more statistically robust results. FitHiChIP is still able to perform differential analysis with single-replicate samples, and in this case reverts to the square-root-dispersion method used by EdgeR.
- Paired ChIP-seq experiments – As mentioned above, it is best practices to have paired ChIP-seq experiments. If that is not do-able, FitHiChIP is bundled with a script that can call peaks de novo from the HiChIP data directly.

1.8 Plotting HiChIP Data

1.8.1 Introduction

The purpose of this document is to provide a step-by-step walkthrough to plot significant interactions or "loops" generated through HiChIP data at regions of interests with minimal computational expertise, as seen in the figure below. This workflow assumes you have completed the previous steps *From fastq to final valid pairs bam file*, *Library QC* and *FitHiChIP Loop Calling*. This guide will use the output bam file generated during the data processing and the merged interactions file from FitHiChIP walkthrough. We will be using the bioconductor package Sushi in R to plot both the coverage and contact arcs, as in the image below.



1.8.2 Inputs

- *Pairtools bam file*
- *FitHiChIP merged interactions output* e.g. interactions_FitHiC_Q0.01_MergeNearContacts.bed

---

Testing!

If you are looking for a dataset to practice this walkthrough, I reccomend the GM12878 CTCF (deep sequencing) from our publicaly available *datasets*

1.8.3 Tools and Data Used

- deepTools
- R with the following packages:
  - Sushi

1.8.4 Basic Workflow

The basic workflow is as follows:

1. Converting the alignments to a contact matrix and a coverage bedgraph
2. Open R and load libraries
3. Import coverage bedgraph and merged contacts files into R and add a column for distance in merged contacts
4. Set genomic regions
5. Plot and Print

1.8.5 Walkthrough

1. Convert bam to bedgraph with deepTools bamCoverage -b mapped.bam -of bedgraph -p 36 -o prefix.coverage.bedgraph

2. Open R

Command:

```
R
```

3. Load libraries

Command:

```
library("Sushi")
```

4. Load data

Command:

```
cov <- read.table("prefix.coverage.bedgraph")
arc <- read.table("prefix_ interactions_FitHiC_Q0.01_MergeNearContacts.bed", header=TRUE)
```

5. Inspect arc file structure

Command:

```
head(arc)
```

```
> head(arc)
  chr1       s1       e1 chr2       s2       e2 cc P.Value_Bias Q.Value_Bias  bin1_low bin1_high  bin2_low bin2_high sumCC StrongConn
1 chr1 24202000 24203000 chr1 24391000 24392000 38 5.349360e-28 7.922064e-26  24201000  24203000  24390000  24393000   106  0.8333333
2 chr1 178031000 178032000 chr1 178190000 178191000 10 5.394300e-05 1.008103e-03 178031000 178033000 178189000 178191000    37  1.0000000
3 chr1 204565000 204566000 chr1 204764000 204765000 19 1.193927e-11 5.279233e-10 204565000 204567000 204763000 204765000    62  1.0000000
4 chr1 181190000 181191000 chr1 181423000 181424000 26 2.515409e-18 1.966821e-16 181189000 181191000 181423000 181426000    85  0.6666667
5 chr1  74772000  74773000 chr1  75203000  75204000 58 1.482219e-64 1.187291e-61  74771000  74773000  75203000  75205000   123  1.0000000
6 chr1  12124000  12125000 chr1  12209000  12210000 26 3.558372e-20 3.176450e-18  12124000  12126000  12208000  12210000    89  1.0000000
```

Here we see that the structure of the significant interactions is structured like a bedpe file with position 1 as - chr1, start1, end1 and position 2 – chr2, start2, end2, make up the first six column entries. This is the key structure sushi needs to plot bedpe as “arcs“ or “loops”.

The other key factor needed is the height of the arc that Sushi will plot. The rest of the columns point to stats regarding the interactions between position 1 and position 2 that could be used as a height scaler. A common way to plot HiChIP interactions that is visually pleasing is scale the height by the distance of the interaction, therefore we need to add a column of the distance between the start of position 1 and end of position 2

6. Add a column for distance in merged contacts file

Command:

```
arc$dist <- abs(arc$e2 - arc$s1)
```

7. Inspect arc file to see distance

Command:

```
head(arc)
```

```
> head(arc)
  chr1       s1       e1 chr2       s2       e2 cc P.Value_Bias Q.Value_Bias  bin1_low bin1_high  bin2_low bin2_high sumCC StrongConn   dist
1 chr1 24202000 24203000 chr1 24391000 24392000 38 5.349360e-28 7.922064e-26 24201000 24203000 24390000 24393000   106  0.8333333 190000
2 chr1 178031000 178032000 chr1 178190000 178191000 10 5.394300e-05 1.008103e-03 178031000 178033000 178189000 178191000    37  1.0000000 160000
3 chr1 204565000 204566000 chr1 204764000 204765000 19 1.193927e-11 5.279233e-10 204565000 204567000 204763000 204765000    62  1.0000000 200000
4 chr1 181190000 181191000 chr1 181423000 181424000 26 2.515409e-18 1.966821e-16 181189000 181191000 181423000 181426000    85  0.6666667 234000
5 chr1  74772000  74773000 chr1  75203000  75204000 58 1.482219e-64 1.187291e-61  74771000  74773000  75203000  75205000   123  1.0000000 432000
6 chr1  12124000  12125000 chr1  12209000  12210000 26 3.558372e-20 3.176450e-18  12124000  12126000  12208000  12210000    89  1.0000000  86000
```

8. Set region of interest for this example a 1.5 Mb region on chr8

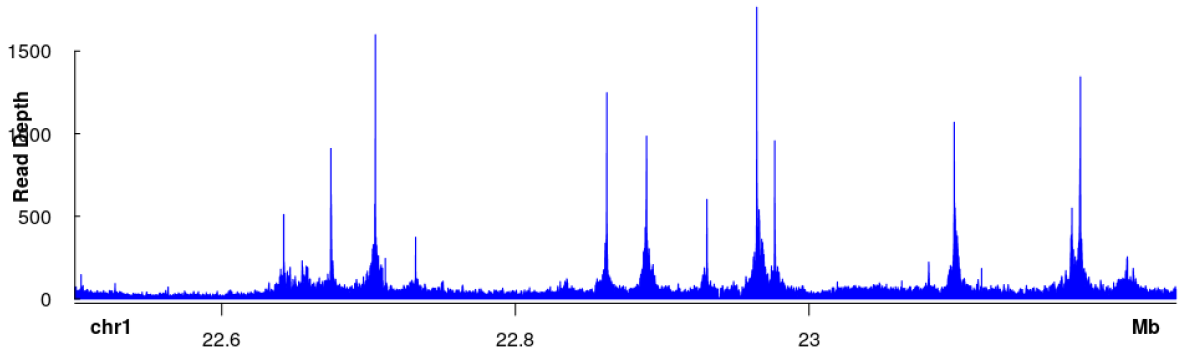
Command:

```
chrom = "chr8"
chromstart = 22500000
chromend = 23200000
```

9. Inspect coverage plot

Command:

```
plotBedgraph(cov,chrom,chromstart,chromend)
labelgenome(chrom,chromstart,chromend,n=4,scale="Mb")
mtext("Read Depth",side=2,line=1.75,cex=1,font=2)
axis(side=2,las=2,tcl=.2)
```



10. Plot arcs with arc heights based on distance

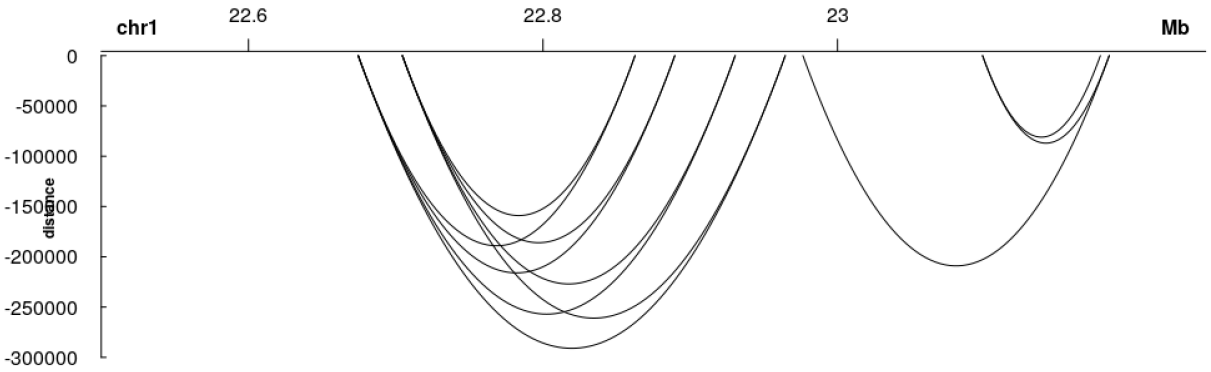
Command:

```
plotBedpe(arc,chrom,chromstart,chromend,heights = arc$dist,plottype="loops", flip=TRUE)
labelgenome(chrom, chromstart,chromend,side=3, n=3,scale="Mb")
```

(continues on next page)

(continued from previous page)

```
axis(side=2,las=2,tcl=.2)
mtext("distance",side=2,line=1.75,cex=.75,font=2)
```

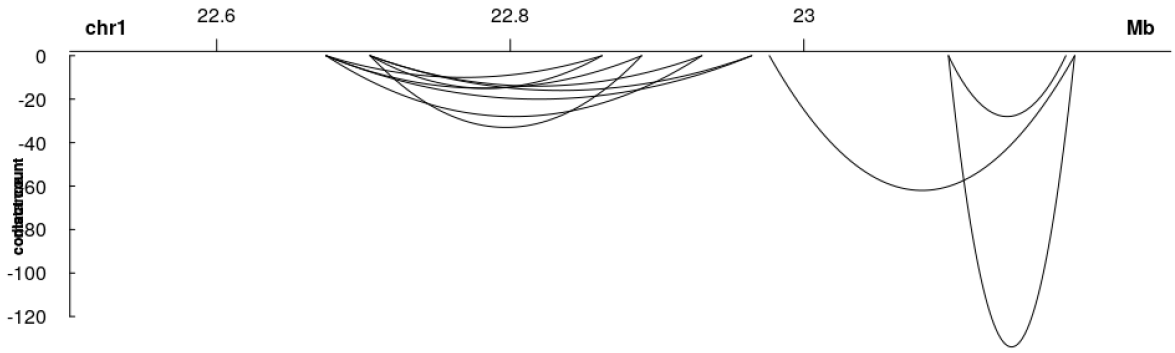


While aesthetically pleasing, the arc file has much more informative information than the distance which is already captured on the x-axis. One could scale the height to the P or Q-values. Or could even add a color scale based on those statistical qualifiers (see the Sushi documentation for other variations on this). To demonstrate an additional layer of information in the arc plot, we can scale the arc height to the number of contacts interacting between position 1 and position 2.

11. Plot arcs with arc heights based on contact frequency

Command:

```
plotBedpe(arc,chrom,chromstart,chromend,heights = arc$sumCC,plottype="loops", flip=TRUE)
labelgenome(chrom, chromstart,chromend,side=3, n=3,scale="Mb")
axis(side=2,las=2,tcl=.2)
mtext("contact freq",side=2,line=1.75,cex=.75,font=2)
```



Finally, we want to generate a PDF file for our records or to clean up in a PDF editor such as Adobe Illustrator.

12. Align and print both plots to a PDF file

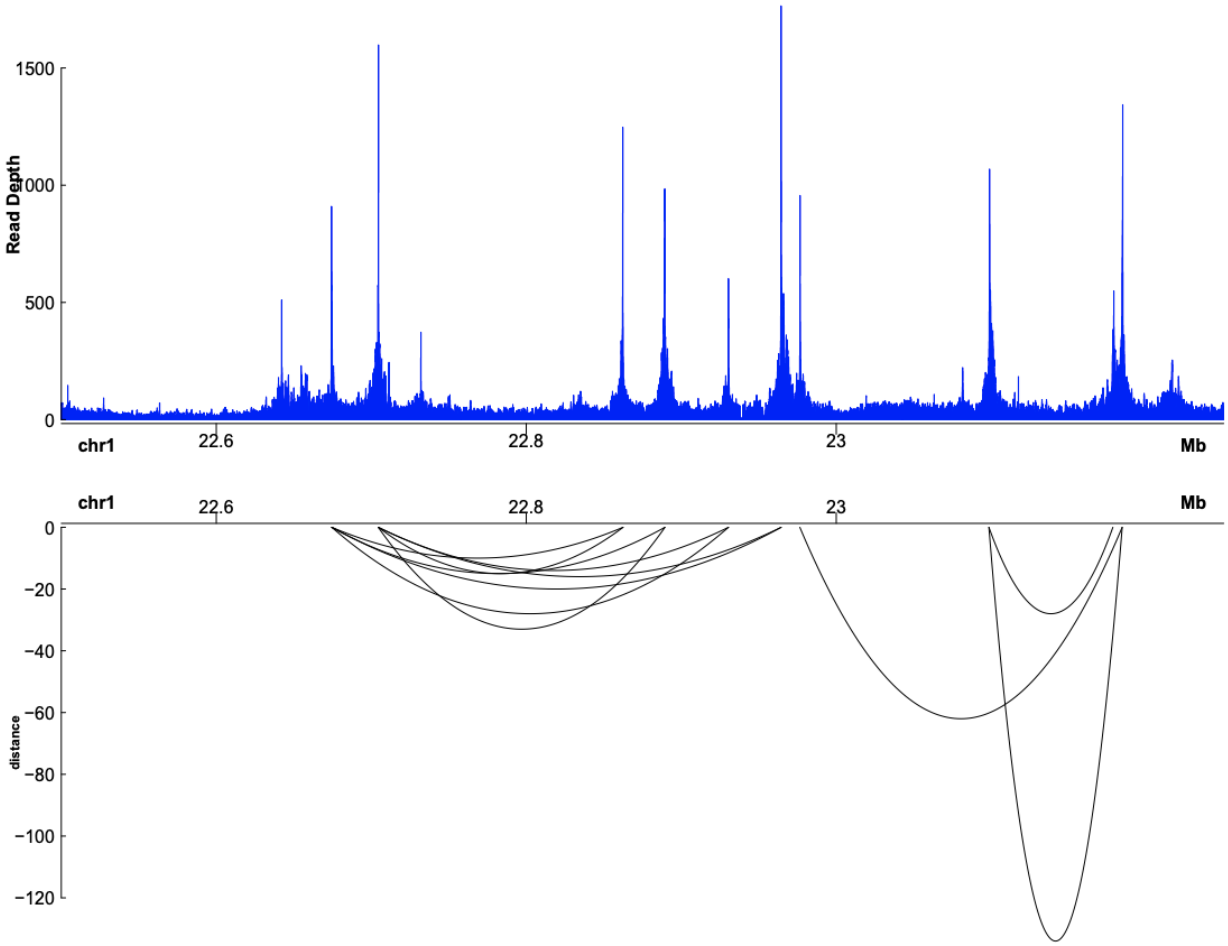
Tip!

where "{}" I'd recommend pasting line-by-line rather than bulk copy and paste

Command:

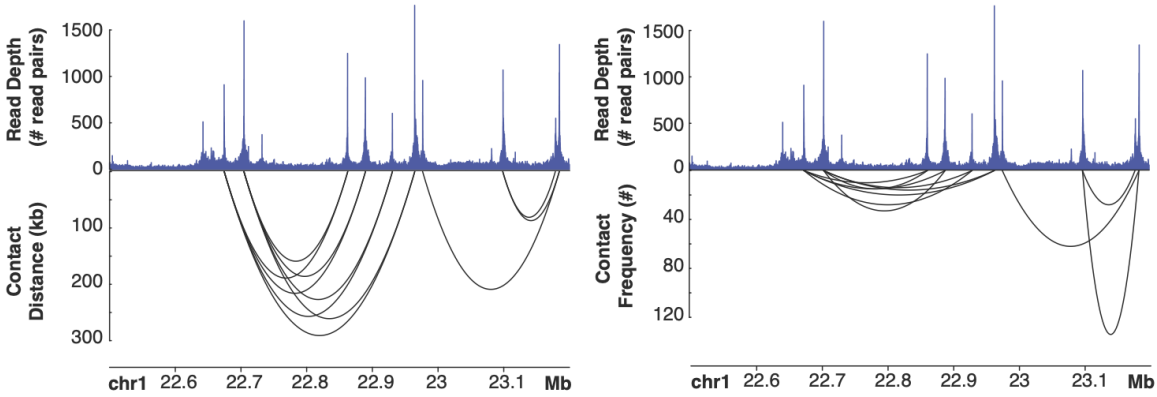
```
pdfname <- "hichip.cov.arcs.pdf"
makepdf = TRUE
if(makepdf==TRUE)
      {
      pdf(pdfname , height=10, width=12)
      }

##set layout
layout(matrix(c(1,
      2
      ), 2,1, byrow=TRUE))
par(mgp=c(3,.3,0))

##plot coverage
par(mar=c(3,4,2,2))
plotBedgraph(cov,chrom,chromstart,chromend)
labelgenome(chrom,chromstart,chromend,n=4,scale="Mb")
mtext("Read Depth",side=2,line=1.75,cex=1,font=2)
axis(side=2,las=2,tcl=.2)

##plot arcs with height based on contact frequency
par(mar=c(3,4,2,2))
plotBedpe(arc,chrom,chromstart,chromend,heights = arc$sumCC,plottype="loops", flip=TRUE)
labelgenome(chrom, chromstart,chromend,side=3, n=3,scale="Mb")
axis(side=2,las=2,tcl=.2)
mtext("distance",side=2,line=1.75,cex=.75,font=2)

if (makepdf==TRUE)
{
dev.off()
}
```

The resulting figure should look like the one below:



There're figures, then there are Figures

The outlined workflow provides a rudimentary plot that illustrates the coverage and proximity-ligation links contained in HiChIP data. There is a lot more you can do to beautify the plots or to place the data in context of additional findings. In other words, there is more that should be done to generate a publishable figure. The Bioconductor package 'Sushi' has a plethora of ways to customize plots. Further documentation on this can be found here. Alternately the one could clean up the figure in a PDF editor, such as Adobe Illustrator. A few extra minutes in Illustrator provides the final figure below where contact arcs are plotted both by height in reference to the coverage (left) and by contact frequency (right):

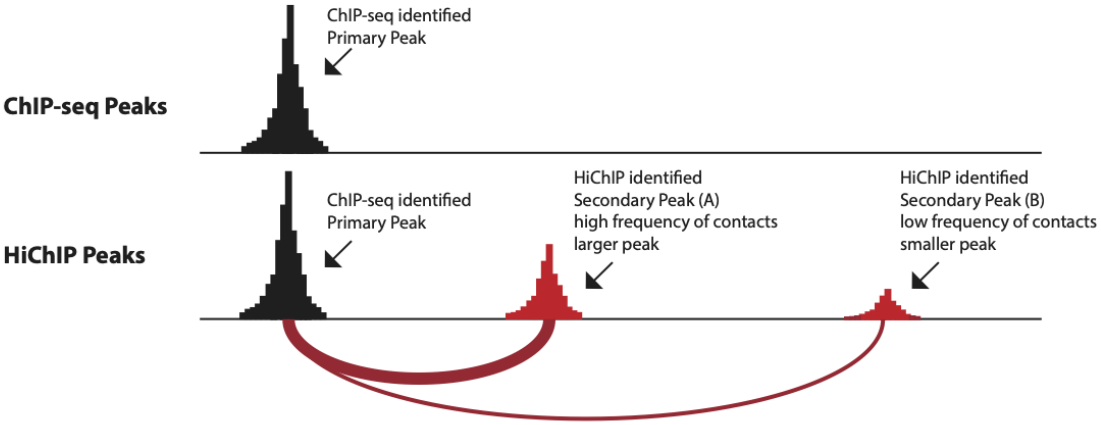


1.9 Calling 1D peaks with MACS2 on HiChIP data

1.9.1 Introduction

Understanding where protein's bind the DNA is a hallmark of ChIP-seq experiments. Typically, MACS2 is used on ChIP-seq data to identify peak signal from the background noise and confirm where these binding sites are located. When it comes to HiChIP these binding sites (or anchors) are important to understand which molecule of the proximity-ligation step occurs at an anchor site or non-anchor site. This is information is used to QC the HiChIP library and is a requirement for identifying significant chromatin interactions. You may not have done any ChIP-seq work on a particular sample or maybe there is no publicly available data that reflects your specific sample type or experimental conditions. While it is most ideal to use ChIP-seq derived peak signals, it is possible to use the HiChIP data to call 1-dimensional peaks like you normally would in the ChIP-seq experiment.

There are a few things to keep in mind when using HiChIP data to call 1-D peaks: 1. You may be identifying secondary peaks along with the primary peaks (see figure below) and without a ChIP-seq dataset, it would be hard to discern one peak type from the other. 2. Using `.bam` files that were processed and filtered through pairtools do not integrate nicely with MACS2. There is a simple solution for that which we will cover here. 3. If you do call peaks with the HiChIP data, you should *run FitHiChIP* on both peak-to-peak and peak-to-all settings.


ChIP-seq Peaks
ChIP-seq identified
Primary Peak
HiChIP Peaks
ChIP-seq identified
Primary Peak
HiChIP identified
Secondary Peak (A)
high frequency of contacts
larger peak
HiChIP identified
Secondary Peak (B)
low frequency of contacts
smaller peak

1.9.2 Input files

- *.bam* file generated at the *from fastq to final valid pairs bam file* step.

Testing!

If you are looking for a dataset to practice this walkthrough, I reccomend the GM12878 CTCF (deep sequencing) from our publicaly available *datasets*

1.9.3 Additional tools needed

- MACS2

1.9.4 Workflow Overview

1. Select the primary alignment in the bam file and convert to bed format.
2. Run MACS2.

1.9.5 Workflow

1. Select the primary alignment in the bam file and convert to bed format.

Command:

```
samtools -view -h -F 0x900 mapped.bam | bedtools bamtobed -i stdin > prefix.primary.aln.
↪bed
```

Here we're using samtools `-view` funtcion to retain the header (-h) and filter and keep (-F) the primary alignment (flag ID – 0X900) of the input bam file. Then the filtered alignments are being pipped into bedtools to convert the alignment (bam format) to bed format using the input flag for a UNIX piped input (stdin). Resulting in a final bed file.

2. Run MACS2.

Command:

```
Macs2 callpeak -t prefix.primary.aln.bed -n prefix.macs2
```

Here we are using macs2 callpeak function the treatment file (-t) which is the primary alignment bed file, with a particular prefix assigned to the outputs (-n).

1.10 HiChIP Data Sets

To download one of the data sets, simply use the wget command:

```
wget https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/HiChiP_CTCF_2M_R1.fastq.gz
wget https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/HiChiP_CTCF_2M_R2.fastq.gz
```

For testing purposes, we recommend using the 2M reads data sets, for any other purpose we recommend using the 800M reads data set.

1.10.1 Sequenced (human) libraries:

Library	Link
GM12878 CTCF 2M	• https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/HiChiP_CTCF_2M_R1.fastq.gz • https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/HiChiP_CTCF_2M_R2.fastq.gz
GM12878 CTCF (deep sequencing)	• https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/CTCF-DS_R1.fastq.gz • https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/CTCF-DS_R2.fastq.gz
GM12878 H3K27Ac (deep sequencing)	• https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/H3K27Ac_R1.fastq.gz • https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/H3K27Ac_R2.fastq.gz
GM12878 H3K4me3 (deep sequencing)	• https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/H3K4me3_R1.fastq.gz • https://s3.amazonaws.com/dovetail.pub/HiChIP/fastqs/H3K4me3_R2.fastq.gz

1.10.2 Human, hg38, Peak files from ENCODE project

Sample	Target	Accession	URL	Output type
GM12878	CTCF	ENCFF017XLW	https://www.encodeproject.org/files/ENCFF017XLW/@@download/ENCFF017XLW.bed.gz	conservative IDR thresholded peaks
IMR-90	H3K4ac	ENCFF823NUO	https://www.encodeproject.org/files/ENCFF823NUO/@@download/ENCFF823NUO.bed.gz	replicated peaks
GM12878	H3K4me3	ENCFF188SZS	https://www.encodeproject.org/files/ENCFF188SZS/@@download/ENCFF188SZS.bed.gz	replicated peaks
IMR-90	H3K14ac	ENCFF106EAN	https://www.encodeproject.org/files/ENCFF106EAN/@@download/ENCFF106EAN.bed.gz	replicated peaks
GM12878	H3K27ac	ENCFF367KIF	https://www.encodeproject.org/files/ENCFF367KIF/@@download/ENCFF367KIF.bed.gz	replicated peaks
GM12878	H3K27me3	ENCFF153VOQ	https://www.encodeproject.org/files/ENCFF153VOQ/@@download/ENCFF153VOQ.bed.gz	replicated peaks
GM12878	H3K36me3	ENCFF268HMO	https://www.encodeproject.org/files/ENCFF268HMO/@@download/ENCFF268HMO.bed.gz	replicated peaks
GM12878	SMC3	ENCFF534PUK	https://www.encodeproject.org/files/ENCFF534PUK/@@download/ENCFF534PUK.bed.gz	bed
MCF-7	Klf4	ENCFF287QDZ	https://www.encodeproject.org/files/ENCFF287QDZ/@@download/ENCFF287QDZ.bed.gz	conservative IDR thresholded peaks
GM23338	Nanog	ENCFF897LBK	https://www.encodeproject.org/files/ENCFF897LBK/@@download/ENCFF897LBK.bed.gz	conservative IDR thresholded peaks
GM12878	POLR2A	ENCFF794VYB	https://www.encodeproject.org/files/ENCFF794VYB/@@download/ENCFF794VYB.bed.gz	conservative IDR thresholded peaks

1.10.3 Data used for HiChIP Comparative Analysis (Mouse, mm10)

To get a list of all the files generated from the HiChIP Comparative Analysis tutorial, including the required reference genomes, you can use the command:

```
aws s3 ls s3://dovetail.pub/HiChIP/compare_samples/
```

Use wget to download any given file, replacing "s3://" with "https://s3.amazonaws.com/", followed by the remaining path to the file. For example:

```
wget https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/Reference_Genome/mm10.
↪fa
```

Data Set	Link
Fastqs (Sample A)	• https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/fastq_inputs/sampleA_R1.fastq.gz • https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/fastq_inputs/sampleA_R2.fastq.gz
Fastqs (Sample B)	• https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/fastq_inputs/sampleB_R1.fastq.gz • https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/fastq_inputs/sampleB_R2.fastq.gz

Note: The full dataset, including input files and generated output is ~183Gb (roughly 5h with a network speed of 10Mb/s).

1.11 Support

For help or questions related please open a new issue on the github repository or send an email to: support@dovetail-genomics.com

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search