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 publically 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:

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

And make the `enrichment_stats.sh` script executable:

```bash
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 narrow-peak 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:

```bash
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:**

```bash
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:**

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

**Example:**

```bash
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:**

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

**Example:**

```bash
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.
HiChiP Documentation, Release 0.1

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

Chapter 1. Overview
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Alignment function</th>
</tr>
</thead>
<tbody>
<tr>
<td>mem</td>
<td>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</td>
</tr>
<tr>
<td>-S</td>
<td>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</td>
</tr>
<tr>
<td>-S</td>
<td>skip mate rescue</td>
</tr>
<tr>
<td>-P</td>
<td>skip pairing; mate rescue performed unless -S also in use</td>
</tr>
<tr>
<td>-T0</td>
<td>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</td>
</tr>
<tr>
<td>-t</td>
<td>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’t know the number of cores, used the command lscpu and multiply Thread(s) per core x Core(s) per socket x Socket(s))</td>
</tr>
<tr>
<td>*.fasta or *.fa</td>
<td>Path to a reference file, ending with .fa or .fasta, e.g. hg38.fasta</td>
</tr>
<tr>
<td>*.fastq or *.fastq.gz</td>
<td>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 &lt; with the cat command (see example below)</td>
</tr>
<tr>
<td>-o</td>
<td>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 ‘</td>
</tr>
</tbody>
</table>

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 -S flag)

#### Command:

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

#### Example (one pair of fastq files):

```
bwa mem -SSP -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):

```
```

#### 1.3. From fastq to final valid pairs bam file
HiChiP Documentation, Release 0.1

**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 assign 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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>min-mapq</td>
<td>40</td>
<td>Mapq threshold for defining an alignment as a multi-mapping alignment. Alignment with mapq &lt;40 will be marked as type M (multi)</td>
</tr>
<tr>
<td>walks-policy</td>
<td>5unique</td>
<td>Walks is the term used to describe multiple ligation 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)</td>
</tr>
<tr>
<td>max-inter-align-gap</td>
<td>30</td>
<td>In cases where there is a gap between alignments, if the gap is 30 or smaller, ignore the gap, if the gap is &gt;30bp, mark as ‘null’ alignment</td>
</tr>
<tr>
<td>nproc-in</td>
<td>integer, e.g. 16</td>
<td>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</td>
</tr>
<tr>
<td>nproc-out</td>
<td>integer, e.g. 16</td>
<td>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</td>
</tr>
<tr>
<td>chroms-path</td>
<td>path to .genome file, e.g. hg38.genome</td>
<td></td>
</tr>
<tr>
<td>*.sam</td>
<td>path to sam file used as an input. If you are piping the input (stdin) skip this option</td>
<td></td>
</tr>
<tr>
<td>*pairsam</td>
<td>name of pairsam file for writing output results. You can choose to skip and pipe the output directly to the next cmd (pairtools sort)</td>
<td></td>
</tr>
</tbody>
</table>

**pairtools parse command example for finding ligation events:**

**Command:**

```
pairtools parse --min-mapq 40 --walks-policy 5unique --max-inter-align-gap 30 --nproc-in \n          --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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>--tmpdir</td>
<td>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</td>
</tr>
<tr>
<td>--nproc</td>
<td>Number of processes to split the sorting work</td>
</tr>
</tbody>
</table>

**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/`

**Removing 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mark-dups</td>
<td>If specified, duplicate pairs are marked as DD in “pair_type” and as a duplicate in the sam entries</td>
</tr>
<tr>
<td>--output-stats</td>
<td>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</td>
</tr>
</tbody>
</table>

**Command:**

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

**Example:**

1.3. From fastq to final valid pairs bam file
HiChiP Documentation, Release 0.1

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>--output-pairs</td>
<td>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 files to stdout use - instead of file name</td>
</tr>
<tr>
<td>--output-sam</td>
<td>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</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-@</td>
<td>number of threads to use</td>
</tr>
<tr>
<td>-o</td>
<td>file name. Write final output to FILE rather than standard output</td>
</tr>
<tr>
<td>-T</td>
<td>path to temp file. Using a temp file will help avoiding memory issues</td>
</tr>
</tbody>
</table>

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

**Process**

- Total Read Pairs (Fastq)
- Align all data in fastq to reference genome (BWA MEM -SP)
- Alignment
- Identify PCR Duplicates in the Mapped Pairs (PairTool)
- Flag PCR duplicates

**Results**

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Read Pairs</td>
<td>2,000,000</td>
<td>100.00%</td>
</tr>
<tr>
<td>Unmapped Read Pairs</td>
<td>86,000</td>
<td>4.30%</td>
</tr>
<tr>
<td>Mapped Read Pairs</td>
<td>1,642,400</td>
<td>82.12%</td>
</tr>
<tr>
<td>PCR Dup Read Pairs</td>
<td>2,800</td>
<td>0.14%</td>
</tr>
<tr>
<td>No-Dup Read Pairs</td>
<td>1,640,800</td>
<td>81.98%</td>
</tr>
<tr>
<td>No-Dup Cis Read Pairs</td>
<td>1,069,563</td>
<td>51.29%</td>
</tr>
<tr>
<td>No-Dup Trans Read Pairs</td>
<td>471,317</td>
<td>22.72%</td>
</tr>
<tr>
<td>No-Dup Valid Read Pairs (cis &gt;= 1 kb + trans)</td>
<td>1,523,483</td>
<td>92.85%</td>
</tr>
<tr>
<td>No-Dup Cis Read Pairs &lt; 1kb</td>
<td>177,317</td>
<td>7.13%</td>
</tr>
<tr>
<td>No-Dup Cis Read Pairs &gt;= 1kb</td>
<td>1,052,245</td>
<td>64.13%</td>
</tr>
<tr>
<td>No-Dup Cis Read Pairs &gt;=10kb</td>
<td>907,455</td>
<td>54.94%</td>
</tr>
</tbody>
</table>

Proportion of Total Read Pairs

---

**Alignment Efficiency**

- Total: 100%
- Alignment: 95.50%
- Dup / No-Dup: 81.99%

---

**Position on Reference Genome**

- Unmapped
- Low MAPQ
- Mapped

---

**Flagging PCR duplicates**

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

**Process**

1. Count non-duplicated read pairs as cis or trans (PairedTools)
2. Select cis read pairs (Python)
3. Characterize cis Insert Size
   - Invalid: (cis < 1 kb)
   - Valid: (cis >= 1 kb)
4. Count valid pairs (cis >= 1 kb and trans) (Python)

**Results**

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

Proportion of No-Dup Read Pairs

**Command:**

1.4. Library QC
python3 ./HiChiP/get_qc.py -p <stats.txt>

Example:

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

After the script completes, it will print:

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Read Pairs</td>
<td>2,000,000</td>
<td>100%</td>
</tr>
<tr>
<td>Unmapped Read Pairs</td>
<td>75,832</td>
<td>3.79%</td>
</tr>
<tr>
<td>Mapped Read Pairs</td>
<td>1,722,285</td>
<td>86.11%</td>
</tr>
<tr>
<td>PCR Dup Read Pairs</td>
<td>4,507</td>
<td>0.23%</td>
</tr>
<tr>
<td>No-Dup Read Pairs</td>
<td>1,717,778</td>
<td>85.89%</td>
</tr>
<tr>
<td>No-Dup Cis Read Pairs</td>
<td>1,385,238</td>
<td>80.64%</td>
</tr>
<tr>
<td>No-Dup Trans Read Pairs</td>
<td>332,540</td>
<td>19.36%</td>
</tr>
<tr>
<td>No-Dup Valid Read Pairs (cis &gt;= 1kb + trans)</td>
<td>875,804</td>
<td>50.98%</td>
</tr>
<tr>
<td>No-Dup Cis Read Pairs &lt; 1kb</td>
<td>841,974</td>
<td>49.02%</td>
</tr>
<tr>
<td>No-Dup Cis Read Pairs &gt;= 1kb</td>
<td>543,264</td>
<td>31.63%</td>
</tr>
<tr>
<td>No-Dup Cis Read Pairs &gt;= 10kb</td>
<td>193,061</td>
<td>11.24%</td>
</tr>
</tbody>
</table>

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
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-g</td>
<td>Input <em>genome file</em></td>
</tr>
<tr>
<td>-b</td>
<td>Input <em>final bam file</em></td>
</tr>
<tr>
<td>-p</td>
<td>Input (either in asimple bed format or narrow peak format)</td>
</tr>
<tr>
<td>-t</td>
<td>no. of threads</td>
</tr>
<tr>
<td>-x</td>
<td>Prefix for output file, enrichment stats will be saved to <code>&lt;prefix&gt;_hichip_qc_metrics.txt</code></td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total ChIP peaks</td>
<td>41,017</td>
</tr>
<tr>
<td>Mean ChIP peak size</td>
<td>309 bp</td>
</tr>
<tr>
<td>Median ChIP peak size</td>
<td>356 bp</td>
</tr>
<tr>
<td>Total reads in 500 bp around center of peaks</td>
<td>321,368 7.91%</td>
</tr>
<tr>
<td>Total reads in 1000 bp around center of peaks</td>
<td>458,843 11.3%</td>
</tr>
<tr>
<td>Total reads in 2000 bp around summits</td>
<td>673,628 16.59%</td>
</tr>
<tr>
<td>Observed/Expected ratio for reads in 500 bp around center of peaks</td>
<td>11.92</td>
</tr>
<tr>
<td>Observed/Expected ratio for reads in 1000 bp around center of peaks</td>
<td>8.51</td>
</tr>
<tr>
<td>Observed/Expected ratio for reads in 2000 bp around center of peaks</td>
<td>6.25</td>
</tr>
</tbody>
</table>

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)

1. **Identify peak centers**
   - Generate a `bed` file of peak centers (Bedtools embedded in `Python`)

2. **Characterize ChIP-seq peak regions**
   - Characterize peak regions (Bedtools embedded in `Python`)

3. **Peak Regions**
   - `bed`

4. **ChiP-seq Signal**
   - `bigwig`, `bedgraph`, `bam`

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total ChiP peaks</td>
<td>41,017</td>
<td>NA</td>
</tr>
<tr>
<td>Mean ChiP peak size</td>
<td>309 bp</td>
<td>NA</td>
</tr>
<tr>
<td>Median ChiP peak size</td>
<td>309 bp</td>
<td>NA</td>
</tr>
<tr>
<td>Total reads in 500 bp around center of peaks</td>
<td>393,163</td>
<td>9.46%</td>
</tr>
<tr>
<td>Total reads in 1000 bp around center of peaks</td>
<td>519,272</td>
<td>12.49%</td>
</tr>
<tr>
<td>Total reads in 2000 bp around center of peaks</td>
<td>692,305</td>
<td>16.66%</td>
</tr>
<tr>
<td>Observed/Expected ratio of reads in 500 bp around center of peaks</td>
<td>14.25</td>
<td>NA</td>
</tr>
<tr>
<td>Observed/Expected ratio of reads in 1000 bp around center of peaks</td>
<td>5.41</td>
<td>NA</td>
</tr>
<tr>
<td>Observed/Expected ratio of reads in 2000 bp around center of peaks</td>
<td>6.27</td>
<td>NA</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total ChIP peaks</td>
<td>41,017</td>
<td>NA</td>
</tr>
<tr>
<td>Mean ChIP peak size</td>
<td>309 bp</td>
<td>NA</td>
</tr>
<tr>
<td>Median ChIP peak size</td>
<td>309 bp</td>
<td>NA</td>
</tr>
<tr>
<td>Total reads in 500 bp around center of peaks</td>
<td>393,163</td>
<td>9.46%</td>
</tr>
<tr>
<td>Total reads in 1000 bp around center of peaks</td>
<td>519,272</td>
<td>12.49%</td>
</tr>
<tr>
<td>Total reads in 2000 bp around center of peaks</td>
<td>692,305</td>
<td>16.66%</td>
</tr>
<tr>
<td>Observed/Expected ratio of reads in 500 bp around center of peaks</td>
<td>14.25</td>
<td>NA</td>
</tr>
<tr>
<td>Observed/Expected ratio of reads in 1000 bp around center of peaks</td>
<td>9.41</td>
<td>NA</td>
</tr>
<tr>
<td>Observed/Expected ratio of reads in 2000 bp around center of peaks</td>
<td>6.27</td>
<td>NA</td>
</tr>
</tbody>
</table>
Step 2b - Calculate the number of expected HiChiP read pairs in regions surrounding ChIP-seq peak centers

Expected value is calculated based on the assumption that the total number of no-dup read pairs is evenly distributed across the ROI proportion of the genome. This is achieved by 1) determining what fraction of the genome length do the ROIs account for, then 2) evenly distributing 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 length as the same number of no-dup read pairs are being distributed across a larger fraction of the genome.

\[
\frac{\text{Total number of peak centers} \times \text{Length of ROI (C ≤ 500 bp)}}{\text{Length of genome (bp)}} \times \text{Total no-dup read pairs (b)} = \frac{\text{Expected # of read pairs ± 500 bp}}
\]

\[
\frac{\text{Total number of peak centers} \times \text{Length of ROI (C ≤ 1000 bp)}}{\text{Length of genome (bp)}} \times \text{Total no-dup read pairs (b)} = \frac{\text{Expected # of read pairs ± 1000 bp}}
\]

\[
\frac{\text{Total number of peak centers} \times \text{Length of ROI (C ≤ 2000 bp)}}{\text{Length of genome (bp)}} \times \text{Total no-dup read pairs (b)} = \frac{\text{Expected # of read pairs ± 2000 bp}}
\]

Step 2c - Calculate the observed to expected ratio

\[
\frac{\text{Observed# of reads} \times \text{Expected # of reads}}{\text{Expected # of reads}} = \frac{\text{Observed/Expected ratio of reads in 500 bp around center of peaks}}{\text{Expected # of reads C ≤ 500}}
\]

\[
\frac{\text{Observed# of reads} \times \text{Expected # of reads}}{\text{Expected # of reads}} = \frac{\text{Observed/Expected ratio of reads in 1000 bp around center of peaks}}{\text{Expected # of reads C ≤ 1000}}
\]

\[
\frac{\text{Observed# of reads} \times \text{Expected # of reads}}{\text{Expected # of reads}} = \frac{\text{Observed/Expected ratio of reads in 2000 bp around center of peaks}}{\text{Expected # of reads C ≤ 2000}}
\]

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-bam</td>
<td>Input <code>final bam file</code></td>
</tr>
<tr>
<td>-peaks</td>
<td>Input peaks in <code>narrowPeak</code> format (<code>plot_chip_enrichment.py</code>) or in simple <code>chr,start,end</code> bed format (<code>plot_chip_enrichment_bed.py</code>)</td>
</tr>
<tr>
<td>-output</td>
<td>Output file name to save the enrichment plot <code>.png</code> image</td>
</tr>
</tbody>
</table>

**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 -o <enrichment.png>
```

or

```
python3 ./HiChiP/plot_chip_enrichment_bed.py -bam mapped.PT.bam -peaks peaks.bed -o <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:
  ```bash
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 length of the peak)

- All intervals in the bed files are used for the meta-analysis when plot_chip_enrichment_bed.py is used
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.
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.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Shallow Seq (20M)</th>
<th>Deep Seq (100-200M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-Dup Read Pairs</td>
<td>&gt;75%</td>
<td>&gt;50%</td>
</tr>
<tr>
<td>No-dup cis read pairs 1kb</td>
<td>&gt;20%</td>
<td>&gt;20%</td>
</tr>
<tr>
<td>Total reads in 1000 bp around center of peaks</td>
<td>&gt;2%</td>
<td>&gt;2%</td>
</tr>
</tbody>
</table>

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 and 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:
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Xmx</td>
<td>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 <code>free -h</code> and check the value under <code>total</code></td>
</tr>
<tr>
<td>Djava.awt.headless=true</td>
<td>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)</td>
</tr>
<tr>
<td>pre</td>
<td>The pre command allows users to create .hic files from their own data</td>
</tr>
<tr>
<td>--threads</td>
<td>Specifies the numbers of threads to be used (integer number)</td>
</tr>
<tr>
<td>*.pairs or *.pairs.gz</td>
<td>input file for generating the contact matrix</td>
</tr>
<tr>
<td>*.genome</td>
<td>genome file, listing the chromosomes and their sizes</td>
</tr>
<tr>
<td>*.hic</td>
<td>hic output file, containing the contact matrix</td>
</tr>
</tbody>
</table>

**Tip no.1**

Please note that if you have an older version 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 differences 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
Generating 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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;genome_fils&gt;:&lt;bin size&gt;</code></td>
<td>Specifies the reference <code>genome</code> file, followed with <code>:</code> and the desired bin size in bp</td>
</tr>
<tr>
<td><code>-p</code></td>
<td>Number of processes to split the work between (integer), default: 8</td>
</tr>
<tr>
<td><code>*.pairs.gz</code></td>
<td>Path to bgzip compressed and indexed <code>.pairs</code> file</td>
</tr>
<tr>
<td><code>*.cool</code></td>
<td>Name of output file</td>
</tr>
</tbody>
</table>

**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
```

Generating 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 regions 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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-balance</code></td>
<td>Apply balancing to each zoom level. Off by default</td>
</tr>
<tr>
<td><code>-p</code></td>
<td>Number of processes to use for batch processing chunks of pixels, default: 1</td>
</tr>
<tr>
<td><code>*.cool</code></td>
<td>Name of contact matrix input file</td>
</tr>
</tbody>
</table>

**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:
  - H3K27ac or H3K4me3 – means identifying regions that interact with the enhancer or active promoter 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!

<table>
<thead>
<tr>
<th>Tool</th>
<th>Repo</th>
<th>Input format</th>
<th>Resolution Option</th>
<th>Considers type of interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>FitHiChIP</td>
<td><a href="https://ay-lab.github.io/FitHiChIP/">https://ay-lab.github.io/FitHiChIP/</a></td>
<td></td>
<td>HiC-Pro Valid-Pairs</td>
<td>Yes</td>
</tr>
<tr>
<td>Cloops</td>
<td><a href="https://github.com/YaqiangCao/cLoops">https://github.com/YaqiangCao/cLoops</a></td>
<td>Bedpe</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>HiChIP-PER</td>
<td><a href="https://github.com/aryeelab/hichipper">https://github.com/aryeelab/hichipper</a></td>
<td></td>
<td>HiC-Pro Valid-Pairs</td>
<td>Yes</td>
</tr>
<tr>
<td>HiCCUPS</td>
<td><a href="https://github.com/aidenlab/juicer/wiki/HiCCUPS">https://github.com/aidenlab/juicer/wiki/HiCCUPS</a></td>
<td>.hic</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

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 HiChIP

## Important parameters

- `ValidPairs`: The file containing the valid pairs from HiPro pipeline. Can be either a text file, or a gzipped text file.

- `OutputBase`: The output base directory under which all results will be stored.

- `InteractionType`: Interaction type. Options include:
  - 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 (H-C data)
  - 5: run everything

- `Resolution`: Resolution. 2.5 - 5 Kb is a good start.

- `LowerDistance`: Lower distance. This depends on the anchor in question - CTCF 20 Kb is a good start, H3K27ac or PolII a lowering 10 Kb is a good idea.

- `Stringency`: Stringency. Options include:
  - 1: HiCHIP(S) - More stringent (peak to peak)
  - 2: HiCHIP(PL) - Less stringent (peak to all, peak to non-peak)

- `BiasFactor`: Bias factor. Options include:
  - 1: Coverage (recommended)
  - 2: ICE

- `Sample`: Sample name.

- `Prefix`: Prefix file name for results files.

- `Overlap`: Overlap parameter. If 1, overwrites any existing output file. Otherwise (0), does not overwrite any output file.

# Output base directory under which all results will be stored

## 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 (H-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 lowering 10 Kb is a good idea.

## Stringency:

- 1: HiCHIP(S) - More stringent (peak to peak)
- 2: HiCHIP(PL) - Less stringent (peak to all, peak to non-peak)

## Bias factor:

- 1: Coverage (recommended)
- 2: ICE
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 recommend 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:**
  
  ```bash
  grep -v '#' <*.pairs> | awk -F"\t" '{print $1"\t"$2"\t"$3"\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"$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.
BIN SIZE=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)
## 1.6. HiChIP Loop Calling

- Run FitHiChIP through docker
  
  **Command:**
  
  ```bash
  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-Dimensional 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

Answering biological questions with proximity-ligation data might seem challenging at first glance. However, once the data are converted into simplified tables, very standard data analysis theory can be applied in basic computing language. This document aims to provide an example and clearly outline the steps used to compare HiChIP data sets from different experimental conditions. Here we use open-source proximity-ligation tools combined with standard analytical approaches in the statistical package R to identify differential interactions between a wildtype (Sample A) and treated sample (Sample B). This will be achieved by asking a series of basic question of the data, providing the code the address that question, then summarizing the results. We will be following the workflow as outlined in Figure 1 and asking the following questions:

1) How to I process the data?
2) Do the data look good?
3) Can differential HiChIP interactions be detected?
4) Do the contact matrices support the loop analysis?
5) What functions are enriched in those differential loops?
6) Which genes within those functions should highlighted?
7) How does looping at both conditions behave at those genes?

It should be noted that apart from the QC steps, these tools are open-sourced and not owned or managed by Dovetail Genomics. As such any errors or trouble shooting that should be required for the installation or execution of any of these tools should be through the github pages of those tools, not by Dovetail Genomics.

The main goal of this tutorial is to get you comfortable with the data structure and format, and to teach you how to ask questions of chromatin interaction data. This is an example of how one could go from fastq to learning what functions and genes are specifically associated with looping across at wildtype-treatment experimental design. Ideally once you have completed this workflow you should be able to ask your own question of loop data!

Figure 1:
1.7.2 Data Access

For access to all input files, including sample A and B fastqs, reference genomes, and example output used in this tutorial, please see the datasets section.
1.7.3 Experimental Design

Cultured mouse cells were subjected to an experimental condition. Ten million wildtype cells (sample A) and treated cells (sample B) were grown, isolated, and flash frozen. The cells were divided into two - 5 million technical replicates and Dovetail genomics MNase-HiChIP was performed against H3K27ac antibody resulting in four libraries (two per condition). Libraries were sequenced to ~100 million total read pairs (2x150bp) and the resulting fastqs were merged prior to data processing with the cat command in linux.

1.7.4 1. Data Processing

**Question:** How do I process these data?

**Process:** Merged fastqs were aligned with bwa mem -5SP and deduplicated and converted into both alignment files (.bam) and pairs coordinate files (.pairs) following the best practices (https://hichip.readthedocs.io/en/latest/fastq_to_bam.html). Valid pairs were converted into contact matrices (.cool) with cooler cloud pairix function. Finally, to prepare for subsequent analyses, coverage bedgraph files were generated from the bam file using deepTools bamCoverage, normalized to RPKM. Since these HiChIP experiments do not have the recommended paired ChIP-seq experiments, the primary alignment was isolated and converted into bed format, then peaks are called with macs2. Computational Steps for this section are outlined in steps 1.1 – 1.6. This step can take about 12-24 hours of compute hours.

**Results:** Through the data processing we know have alignments, pair files, contact matrices, alignment files, and narrowPeak files that required for downstream analyses. This workflow generated all the file types denoted by grey boxes in Figure 1. We have also generated 1D peak files as denoted in the purple box in Figure 1.

**Tools needed:**
- bwa
- pairtools
- samtools
- deepTools
- pairix
- cooler
- bedtools
- macs2

**Commands:**

- **1.1 Align, parse pairs to get bam and pairs file**

  ```
  ```

- **1.2 Make bedgraphs from bam files**

  ```
  ```
HiChIP Documentation, Release 0.1

bamCoverage --normalizeUsing RPKM -of bedgraph --ignoreDuplicates -bs 50 -p 16 -b sampleA.bam -o sampleA.RPKM.bedgraph
bamCoverage --normalizeUsing RPKM -of bedgraph --ignoreDuplicates -bs 50 -p 16 -b sampleB.bam -o sampleB.RPKM.bedgraph

- 1.3 Convert to hicpro format, by re-ordering the columns, for FitHiChIP loop calling

zcat sampleA.pairs.gz | grep -v '#' | awk -F"\t" '{print $1"\t"$2"\t"$3"\t"$6"\t"$4"\t"$5 "\t"$7}'} | gzip -c >sampleA.hicpro.pairs.gz
zcat sampleA.pairs.gz | grep -v '#' | awk -F"\t" '{print $1"\t"$2"\t"$3"\t"$6"\t"$4"\t"$5 "\t"$7}'} | gzip -c >sampleA.hicpro.pairs.gz

- 1.4.1 index pairs

pairix sampleA.pairs.gz
pairix sampleB.pairs.gz

- 1.4.2 make 5kb matrices

cooler cload pairix -p 24 mm10.geome:5000 sampleA.pairs.gz sampleA.5kb.cool
cooler cload pairix -p 24 mm10.geome:5000 sampleB.pairs.gz sampleB.5kb.cool

- 1.5 Get primary alignment from the bam file and convert to bed format

samtools view -@24 -h -F 0x900 sampleA.bam | bedtools bamtobed -i stdin > sampleA.primary.aln.bed
samtools view -@24 -h -F 0x900 sampleB.bam | bedtools bamtobed -i stdin > sampleB.primary.aln.bed

- 1.6 Call peaks

macs2 callpeak -t sampleA.primary.aln.bed -n sampleA --nomodel
macs2 callpeak -t sampleB.primary.aln.bed -n sampleB --nomodel

1.7.5 2. Library Quality Control

**Question:** Are the data of good quality?

**Process:** Here we use the HiChIP QC scripts (available here: https://hichip.readthedocs.io/en/latest/library_qc.html) to assess the proximity-igation and chromatin immunoprecipitation quality. The QC process utilizes the pairtools stats file and a file of ChIP peaks for H3K27ac. In this example, ChIP-seq was not performed prior to the HiChIP. Therefore, the ENCODE file (encodeproject.org/files/ENCFF135ATY/) was used to assess enrichment observed vs. expected ratio. The command line-based steps used are denoted in steps 2.1 to 2.4. This step takes about 4 hours with deeply sequenced data sets (~200 million reads pairs).

**Results:** Both samples display similar high alignment (>80%) and acceptable duplication rates (< 30%) consistent with expectations of sufficiently complex HiChIP library, resulting in greater than 54% of the data as valid non duplicated pairs. Furthermore, the libraries show successful proximity-igation and sufficient long-range interactions with more than 22% of the non-duplicated cis insert size were > 1kb (Figure 2A; Table 1). In the assessment of the IP-enrichment, we see that both libraries show a high concentration of reads at the peak centers with an observed:expected ratio of > 3 (Table 1), combined with the meta-data coverage plots at all H3K27ac in the ENCODE file (Figure 2B) indicate a successful IP. The IP scores may not perfectly reflect the experiment as the peak files were generated from a different experiment and do not reflect the conditions of the experimentally derived HiChIP libraries, but this is a suitable proxy to assess library quality. Both libraries pass the quality control thresholds, and we can proceed to further analyses.
Tools needed:

- Scripts from the HiChIP Library QC Page.

Commands:

- 2.1 Get alignment and proximity-ligation QC stats from the pairtools stats file

```bash
python HiChIP/getqc.py -p sampleA.txt
python HiChIP/getqc.py -p sampleB.txt
```

- 2.2 Download peaks file

```bash
wget https://www.encodeproject.org/files/ENCFF135ATY/@@download/ENCFF135ATY.bed.gz
gunzip ENCFF135ATY.bed.gz
```

- 2.3 Get enrichment stats against encode file from the bam file (this takes a little while)

```bash
HiChIP/./enrichment_stats.sh -g mm10.genome -b sampleA.bam -p ENCFF135ATY.bed -t 16 -x sampleA.IP.txt
HiChIP/./enrichment_stats.sh -g mm10.genome -b sampleB.bam -p ENCFF135ATY.bed -t 16 -x sampleB.IP.txt
```

- 2.4 Plot enrichment over the ENCODE Peaks file from the bam

```bash
python HiChIP/plot_chip_enrichment.py -bam Coverage_files/sampleA.bam -peaks ENCFF135ATY.bed -output sampleA.IP.png
python HiChIP/plot_chip_enrichment.py -bam Coverage_files/sampleB.bam -peaks ENCFF135ATY.bed -output sampleB.IP.png
```

Table 1: HiChIP QC summary table.

<table>
<thead>
<tr>
<th>Metric</th>
<th>sampleA</th>
<th>sampleB</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Read Pairs</td>
<td>196,210,956</td>
<td>198,527,721</td>
<td>100%</td>
</tr>
<tr>
<td>Unmapped Read Pairs</td>
<td>8,428,185</td>
<td>10,514,980</td>
<td>4.30%</td>
</tr>
<tr>
<td>Mapped Read Pairs</td>
<td>165,423,653</td>
<td>161,627,800</td>
<td>84.31%</td>
</tr>
<tr>
<td>PCR Dup Read Pairs</td>
<td>58,058,297</td>
<td>48,630,488</td>
<td>29.59%</td>
</tr>
<tr>
<td>No-Dup Read Pairs</td>
<td>107,365,356</td>
<td>112,997,312</td>
<td>54.72%</td>
</tr>
<tr>
<td>No-Dup CIS Read Pairs</td>
<td>85,112,000</td>
<td>85,317,149</td>
<td>79.27%</td>
</tr>
<tr>
<td>No-Dup Trans Read Pairs</td>
<td>22,253,356</td>
<td>27,680,163</td>
<td>20.73%</td>
</tr>
<tr>
<td>No-Dup Valid Read Pairs (cis &gt;= 1kb + trans)</td>
<td>67,149,968</td>
<td>77,417,472</td>
<td>62.54%</td>
</tr>
<tr>
<td>No-Dup CIS Read Pairs &lt; 1kb</td>
<td>40,215,388</td>
<td>35,579,840</td>
<td>37.46%</td>
</tr>
<tr>
<td>No-Dup CIS Read Pairs &gt;= 1kb</td>
<td>44,896,612</td>
<td>49,737,309</td>
<td>41.82%</td>
</tr>
<tr>
<td>No-Dup CIS Read Pairs &gt;= 10kb</td>
<td>19,339,665</td>
<td>25,978,289</td>
<td>18.01%</td>
</tr>
<tr>
<td>% of total read pairs</td>
<td></td>
<td></td>
<td>22.99%</td>
</tr>
<tr>
<td>% of No-Dup read pairs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: HiChIP QC Results. A Alignment, PCR, and proximity ligation stats. Alignment stats are presented as a percentage of total read pairs. Proximity-ligation stats are presented as a percentage of No-Dup read pairs. B Chromatin IP efficiency, meta-peak analyses showing coverage as log fold change over the mean coverage across all encode peaks, and stats reporting the percent of No-dup reads over peaks and the observed/Expected ratio as summarized in Table 1.

1.7. HiChIP Comparative Analyses
1.7.6 3. Loop calling and differential looping

**Question:** Can differential HiChIP interactions be detected?

**Process:** To address this question we need to determine which interactions are significantly enriched within each condition, also known as loop calling, then determine which loops are unique to a condition or shared across conditions. The inputs are the pairs files in HiC-Pro format (step 1.3) and the narrowPeak files generated in step (1.6). Then we run FitHiChIP, through Docker, and point to a configuration file. In this experiment loops will be identified at 5kb resolution, in a All-to-All manner, default loop ranges (20kb-2mb), coverage bias turned on, FitHiChIP(L) background modeling, and merging redundant loops. The configuration files are available in the directory you downloaded at the start of the workshop. Loop files will be extracted from the FitHiChIP output, NOTE- the output file has a “.bed” moniker but is actually in bedpe format. Using R, we’ll reformat the FitHiChIP output, then use bedtools pairToPair command with the “both” option flagged to identify loops between the two conditions that share the same anchor positions. Finally, we use the eulerr package in R to plot a Venn diagram. This process is outlined in the command steps 3.1 - 3.6. This process takes about 2 hours, most of which is on running FitHiChIP.

**Results:** Sample A (3,240 total loops) contains ~700 more loops than sample B (2,484 total loops). Of which 652 loops are shared between the two conditions, resulting in roughly 3,200 and 2,500 unique loops in sample A and sample B, respectively. These results are summarized in Figure 3.

**Tools needed:**
- FitHiChIP
- bedtools
- R
  - Packages: eulerr

**Commands:**

- 3.1 Call loops with FitHiChIP pointing the configuration file to the HiCPro format pairs generated in step #1.5
  
  ```
  ./FitHiChIP_Docker.sh -C sampleA.config.file.txt
  ./FitHiChIP_Docker.sh -C sampleB.config.file.txt
  ```

- 3.2 Copy loop files to a into working directory from FitHiChIP output (these files are buried pretty deep in the FitHiChIP output)
• 3.3 Clean files to and convert to bedpe format in R

```r
# load libraries
library(eulerr)

# Import data
lA <- read.table("sampleA_5kb.interactions_FitHiC_Q0.1_MergeNearContacts.bed", header=TRUE)
lB <- read.table("sampleB_5kb.interactions_FitHiC_Q0.1_MergeNearContacts.bed", header=TRUE)

# Add column for unique loop ID and binpair ID
lA$LoopID <- paste("sampleA", seq_along(lA[,1]), sep="-")
lB$LoopID <- paste("sampleB", seq_along(lB[,1]), sep="-")
lA$Bin_Pair_ID <- paste(lA$chr1, lA$s1, lA$e1, lA$chr2, lA$s2, lA$e2, sep="-")
lB$Bin_Pair_ID <- paste(lB$chr1, lB$s1, lB$e1, lB$chr2, lB$s2, lB$e2, sep="-")

# Select desired columns
lA.f <- subset(lA, select=c("chr1", "s1", "e1", "chr2", "s2", "e2", "cc", "LoopID"))
lB.f <- subset(lB, select=c("chr1", "s1", "e1", "chr2", "s2", "e2", "cc", "LoopID"))

# Print files as bedpe
write.table(lA.f,"sampleA.clean.loops.bedpe",row.names=FALSE,sep="\t", col.names=FALSE, quote = FALSE)
write.table(lB.f,"sampleB.clean.loops.bedpe",row.names=FALSE,sep="\t", col.names=FALSE, quote = FALSE)

# Count number loops
nrow(lA)
nrow(lB)

# Draw Venndiagram
A = as.vector(lA$Bin_Pair_ID)
B = as.vector(lB$Bin_Pair_ID)
L=list(SampleA=A, SampleB=B)
L.plot <- plot(venn(L), fills = list(fill = c("blue", "red")), alpha=0.5)
pdf("Loop_summary.pdf", height=10, width=10)
L.plot
dev.off()

# Leave R
q()
```

• 3.4 Use bedtools pairToPair to find shared loops and print bedpes

```
pairToPair -a sampleA.clean.loops.bedpe -b sampleB.clean.loops.bedpe -type both >
--overlapping_notclean.loops.bedpe
```

• 3.5 Cut columns into new files (one for plotting and one for filtering)
cut -f 1-6 overlapping_notclean.loops.bedpe > shared.clean.loops

cut -f8,16 overlapping_notclean.loops.bedpe > shared.list.txt

• 3.6 Use R to make tables of shared and unique loops based on loop ID

```r
# load libraries
library(dplyr)

# import data (clean loop tables and list of overlapping IDs)
lA <- read.table("sampleA.clean.loops.bedpe", header=FALSE)
lB <- read.table("sampleB.clean.loops.bedpe", header=FALSE)
IDs <- read.table("shared.list.txt", header=FALSE)

# rename columns
lA <- rename(lA, chr1 = V1, s1 = V2, e1 = V3, chr2 = V4, s2 = V5, e2 = V6, A_count = V7, sampleA_ID = V8)
lB <- rename(lB, chr1 = V1, s1 = V2, e1 = V3, chr2 = V4, s2 = V5, e2 = V6, B_count = V7, sampleB_ID = V8)
IDs <- rename(IDs, sampleA_ID = V1, sampleB_ID = V2)

# use anti join to get unique lists
uA <- anti_join(lA, IDs, by="sampleA_ID")
uB <- anti_join(lB, IDs, by="sampleB_ID")

# select rows
uA.f <- subset(uA, select=c("chr1", "s1", "e1", "chr2", "s2", "e2"))
uB.f <- subset(uB, select=c("chr1", "s1", "e1", "chr2", "s2", "e2"))

# print unique list
write.table(uA.f,"unique.sampleA.loops.bedpe",row.names=FALSE,sep="\t", col.names=FALSE, quote = FALSE)
write.table(uB.f,"unique.sampleB.loops.bedpe",row.names=FALSE,sep="\t", col.names=FALSE, quote = FALSE)

# leave R
q()
```

Figure 3. Summary of loop calls and differential looping. Venn diagram of overlapping produced in step 3.3.7, and slightly cleaned up in illustrator. The table of loop counts from running wc -l on the generated bedpe files.
1.7.7 4. APA analysis to confirm loop analysis

**Question:** Do the contact matrices support the loop analysis?

**Process:** To check the results of the loop comparison we need to observe how to contact matrices behave at the shared and condition-specific unique loops. To do this, we need to average contact matrices at all loop sets (shared, unique A, and unique B) for both matrices. This is known as aggregate peak analysis (APA). Here we’ll used the bedpe’s generated in steps 3.5 and 3.6.6 as along with the contact matrices generated in step 1.6.2 in the tool coolpup.py to build the APA matrices, then plot with plotpup within the coolpup.py package. This approach not only generates matrices, but it also generates an APA score, which in the case of coolpup, is the Z-score of the mean contact density in the middle of the matrix to the mean of the entire matrix. Higher APA scores indicate stronger enrichment. This process is detailed in steps 4.1 - 4.2. This process takes about 15 mins.

**Results:** In general sample A shows a strong enrichment score at loops unique to sample A and shared loops, confirming that sample A has more loops than sample B. To understand if condition-specific loops are truly unique we ask how the contact matrix of the opposing sample behaves at unique loop sets. APA scores are strongest at their condition-specific loop calls, where in the other samples there is two-fold drop in contact enrichment. Moreover, there is a loss of a bright, punctate spot in the center of matrix. This suggests that while there is contact enrichment at loop sites in the opposing
sample, they are significantly weaker. These finding posit that loops unique to a condition are truly unique. For loops that are shared across the conditions we see that APA scores are equivalent and display a strong contact signal in the center to the aggregated matrix, confirming that shared loops are, in fact, shared across the two conditions (Figure 4).

**Tools needed:**

- coolpup.py

**Commands:**

- 4.1 Build aggregate contact signal over shared and unique loops with coolpup

```bash
coolpup.py sampleA.5kb.cool unique.sampleA.loops.bedpe --unbalanced --n_proc 24 --outname sA_matrix.vs.uAloops.txt
coolpup.py sampleA.5kb.cool shared.clean.loops --unbalanced --n_proc 24 --outname sA_matrix.vs.sharedloops.txt
coolpup.py sampleA.5kb.cool unique.sampleB.loops.bedpe --unbalanced --n_proc 24 --outname sA_matrix.vs.uBloops.txt
coolpup.py sampleB.5kb.cool unique.sampleA.loops.bedpe --unbalanced --n_proc 24 --outname sB_matrix.vs.uAloops.txt
coolpup.py sampleB.5kb.cool shared.clean.loops --unbalanced --n_proc 24 --outname sB_matrix.vs.sharedloops.txt
coolpup.py sampleB.5kb.cool unique.sampleB.loops.bedpe --unbalanced --n_proc 24 --outname sB_matrix.vs.uBloops.txt
```

- 4.2 Plot aggregate contacts with plotpup

```bash
plotpup.py sA_matrix.vs.uAloops.txt sA_matrix.vs.sharedloops.txt sA_matrix.vs.uBloops.txt sB_matrix.vs.uAloops.txt sB_matrix.vs.sharedloops.txt sB_matrix.vs.uBloops.txt --row_names sampleA, sampleB --col_names unique_A, shared, unique_B --n_cols 3 --vmin 1 --vmax 100 --output loops.png
```

Figure 4. APA results. Aggregated matrices and APA scores shown for all loop sets in both samples. Top row the sample A matrix is aggregated at sample A unique loops (left column), shared loops (center column), and sample B unique loops (right column). The same loops sets were used to aggregate the sample B matrix in the bottom row. APA scores (Z-scores) are shown in the top left of each APA matrix.
1.7.8 5. Annotate and GO Enrichment Analysis

**Question:** What functions are enriched in those differential loops?

**Process:** Most annotation workflows use genomic ranges or bed file files to perform annotation. As loop files are in genomic interaction format or bedpe files they don’t directly plug into these analyses. To solve this - it is good think about the bedpe files as simply two bed entries of loop anchor positions side-by-side. With this understanding, we simply need to isolate the anchors in reach row into a separate bed files, merge, and sort to retain unique entries. This is quickly achieved with basic linux language, cut, cat, and sort. Now that we have our bed files of unique anchor position for each loop type, we can simply plug these data into an annotation package in R, such as ChIPseeker. Following the annotation, a Gene Ontology (GO) analyses can be performed to see which functions are enriched at loop anchors for each loop type. You can follow this workflow with steps 5.1 – 5.4. This process takes about 5 mins once all the packages are installed.

**Results:** Many of the functions that are enriched at loop anchors occur in both conditions. As our focus is the differences between sample A and sample B, we will not focus on the shared category. To this end, the functional differences can be observed through the presence/absence of functions as in the GnRH signaling pathway and the Neurotrophin signaling pathway are present in sample A anchors, but not in sample B. More subtle change can be seen through the gene ratio (# of genes / total genes in loop anchors) such as Salmonella infection or Proteoglycans in Cancer where sample A has more genes in the pathway associated with loop anchors. Differences in the significance of the enrichment also account for many of changes between sample A and sample B. There are numerous functions that fall into this last category, such as Hepatocellular carcinoma, Gastric acid secretion, Growth hormone synthesis, secretion and action, and Glimo (Figure 5).

**Tools needed:**
- R
  - Packages: ChIPseeker; TxDb.Mmuscules.UCSC.mm10.knownGene; EnsDb.Mmuscules.v79, clusterProfiler, AnnotationDbi, org.Mm.eg.db, dplyr
Commands:

- 5.1 Make bedfile of loop anchors

```bash
cut -f 1-3 unique.sampleA.loops.bedpe > sampleA.anchor1.bed
cut -f 4-6 unique.sampleA.loops.bedpe > sampleA.anchor2.bed
cut -f 1-3 unique.sampleB.loops.bedpe > sampleB.anchor1.bed
cut -f 4-6 unique.sampleB.loops.bedpe > sampleB.anchor2.bed
cut -f 1-3 shared.loops.bedpe > shared.anchor1.bed
cut -f 4-6 shared.loops.bedpe > shared.anchor2.bed
```

- 5.2 cat anchor1 and anchor2 per loop condition and sort to retain unique entries

```bash
cat sampleA.anchor* | sort -u > sampleA.All.anchors.bed
cat sampleB.anchor* | sort -u > sampleB.All.anchors.bed
cat shared.anchor* | sort -u > shared.All.anchors.bed
```

- 5.3 move merged and sorted anchor beds into new directory called anchors

```bash
mkdir anchors
mv *.All.* anchors/
```

- 5.4 Annotate in R and run GO enrichment analysis

```r
# Load libraries
library(ChIPseeker)
library(TxDb.Mmusculus.UCSC.mm10.knownGene)
library(EnsDb.Mmusculus.v79)
library(clusterProfiler)
library(AnnotationDbi)
library(org.Mm.eg.db)
library(dplyr)

# Import and organize anchor files
samplefiles <- list.files("anchors/", pattern= ".bed", full.names=T)
samplefiles <- as.list(samplefiles)
names(samplefiles) <- c("sampleA", "sampleB", "shared")

# Set reference database to annotate against
txdb <- TxDb.Mmusculus.UCSC.mm10.knownGene

# Annotate
peakAnnoList <- lapply(samplefiles, annotatePeak, TxDb=txdb, tssRegion=c(-1000, 1000), verbose=FALSE)

# GO analyses
genes = lapply(peakAnnoList, function(i) as.data.frame(i)$geneId)
compKEGG <- compareCluster(geneCluster = genes, fun="enrichKEGG", organism = "mouse", pvalueCutoff = 0.05, pAdjustMethod = "BH")

# Summarize GO in a dotplot
p1 <- dotplot(compKEGG, showCategory = 20, title = "KEGG Pathway Enrichment Analysis")
pdf("GO.pdf", height=15, width=8)
p1
dev.off()
```

(continues on next page)
Figure 5. GO enrichment analysis results
1.7.9 6. Identify regions of interest through Pathway Analysis

**Question:** Which genes within those functions should be highlighted?

**Process:** In this step, we will isolate the gene IDs that are present in both samples for the function we are interested in, and plot them on a KEGG pathway to see which genes have condition-specific loop anchors associated with them. To isolate the gene IDs, we’ll use the KEGG.GO.txt file generated 5.4.7 and select the function we’re interested in, in this case the Growth hormone synthesis, secretion and action (GHSSA). From there we perform a series of formatting steps to get the gene IDs into a vector format in R. In order to color the genes in the plot there needs to be a numeric value associated with each gene ID. As the data are currently in binary format (present/absent) we can apply a pseudo-log fold change, were absent equals 1 and present equal 20 to have a dramatic log fold change. Now we simply take the log of pseudo scores for gene IDs in sample A divided by the pseudo score for gene IDs in sample B. Next the gene IDs are merged with the pseudo-logFC score into a list format then plot the pathway of interest with pathview in R. This step takes ~5 mins.

**Results:** Through this approach we can easily identify which genes occur at loop anchors in sample A (red color), sample B (green color), or those with loop anchors in both samples (grey). This analysis indicates that the treatment leads to a general loss of looping associated with the GHSSA function with 15 genes showing in sample A loop anchors and only 5 genes showing enrichment in sample B loop anchors. One could interpret this image as red gene boxes indicate a loss of looping during the treatment, where green gene boxes highlight gained loops, and grey are unchanged by the treatment (Figure 6).

**Tools needed:**

- R
  - Packages: splitstackshape; data.table; dplyr; tidyr; pathview

**Commands:**

- 6.1 load libraries

```r
library(splitstackshape)
library(data.table)
library(dplyr)
library(tidyr)
library(pathview)
```

- 6.2 import data

```r
t <- read.table("KEGG.GO.txt", sep=’\t’, header=TRUE)
```

- 6.3 Select genes IDs in pathway of interest and format to get master list of IDs across sample A and B

```r
t1 <- t[which(t$ID == "mmu04935"),]
t2 <- select(t1, Cluster, geneID)
row.names(t2) <- t2$Cluster
t3 <- filter(t2, Cluster !="shared")
ts <- cSplit(t3, "geneID", "/")
tt <- transpose(ts)
header.true <- function(tt) {names(tt) <- as.character(unlist(tt[1,]))
  tt[-1,]
}
tt1 <- header.true(tt)
a <- select(tt1, sampleA)
a<-rename(a, ID=sampleA)
b <- select(tt1, sampleB)
```

(continues on next page)
b <- rename(b, ID = sampleB)
ml <- rbind(a, b)
m <- unique(ml)
m2 <- unique(ml)

6.4 build table of presence and absence in sample A and B
m$SampleA <- do.call(paste0, m) %in% do.call(paste0, a)
m2$SampleB <- do.call(paste0, m2) %in% do.call(paste0, b)
df <- merge (m, m2, by = "ID")
df <- df %>% drop_na()

6.5 change binary presence or absence into values to get a fold change of presence or absence
df$sA.presence <- ifelse(df$SampleA == "TRUE", 20, 1)
df$sB.presence <- ifelse(df$SampleB == "TRUE", 20, 1)
df$logfc <- log(df$sA.presence / df$sB.presence)

6.6 make List of gene IDs and fold changes
d <- select(df, ID, logfc)
mspecies <- "mmu04935"
genes <- c(d$ID)
logFC <- d$logfc
names(logFC) <- genes

6.7 plot with pathview
pathview(gene.data = logFC, species = "mmu", pathway = mypathway)

Figure 6. Differential gene enrichment at loop anchors in the Growth hormone synthesis, secretion and action (GHSSA) pathway. Red represents genes associated in sample A-specific loop anchors, green are present only in sample loop anchors, and grey are present in both loop anchors.
1.7.10 7. Plotting regions of interest

**Question:** How does looping at both conditions behave at those genes?

**Process:** All the previous work has lead to a point where we can now plot coverage signal showing H3K27ac enrichment and the significant interaction as arcs to understand how the treatment impacts a specific region. He we select the first gene in the pathway Ghrh, growth hormone releasing hormone which is a gained loop anchor. In order to inspect looping dynamics we plot the coverage, from the .bedgraph files generated in step 1.2 and the FitHiChIP loops generated in step 3.2 in in window 600kb window centered at the Ghrh promoter using the R package Sushi. This plotting excersie is captured in steps 7.1 – 7.2. This process takes about 15 mins.

**Results:** H3K27ac enrichment is almost identical between the two the two conditions. However, the looping dynamics are dramatically different. In this region, loops in sample B are more numerous, and stronger in contact strength (indicated by number of contacts). Specifically, Ghrh has a weak interaction with an H3K27ac enhancer ~130kb downstream at the SRC and 2819923M15Rik loci. This particular loop is absent in sample A. The resulting plot (Figure 7A) was cleaned up in the PDF editor Adobe Illustrator (Figure 7B).

**Tools needed:**
- Adobe Illustrator or similar
HiChiP Documentation, Release 0.1

- **R**
  - Packages: sushi

**Commands:**

- **7.1** download and unzip refseq genes list in gtf format from UCSC

```
wget https://hgdownload.soe.ucsc.edu/goldenPath/mm10/bigZips/genes/mm10.refGene.gtf.gz
gunzip mm10.refGene.gtf.gz
```

- **7.2** Plot results in R

```
# load libraries
library(Sushi)
library(dplyr)
library(splitstackshape)

# import coverage
sA.covs <- read.table("sampleA.RPKM.bedgraph", header=F)
sB.covs <- read.table("sampleB.RPKM.bedgraph", header=F)

# import loops
sA.arc <- read.table("sampleA_5kb.interactions_FitHiC_Q0.1_MergeNearContacts.bed", header=T)
sB.arc <- read.table("sampleB_5kb.interactions_FitHiC_Q0.1_MergeNearContacts.bed", header=T)

# import genes and format for sushi plotting
g <- read.table("mm10.refGene.gtf", header=F, sep='\t')
g2 <- cSplit(g, "V9", ":")
g3 <- select(g2, V1, V2, V3, V4, V5, V6, V7, V8, V9_1)
g4 <- cSplit(g3, "V9_1", " ")
g5 <- select(g4, V1, V2, V3, V4, V5, V6, V7, V8, V9_1_2)
g5 <- rename(g5, "chrom"=V1, "source"=V2, "type"=V3, "start"=V4, "end"=V5, "score"=V6, "strand"=V7, "gene"=V9_1_2)
trx <- g5[which(g5$type=='transcript'),]
genes <- select(trx, chrom, start, end, gene, score, strand, type)

# Set ROI for GRGH
chrom="chr2"
chromstart=157238902
chromend=157582101

# Plot ROI
pdfname = "GRHRH.pdf"
makepdf = TRUE

if(makepdf == TRUE)
{
  pdf(pdfname, height =5, width=10)
}

#set layout
layout(matrix(c(1,1,1,1,
```
2,2,2,2,
3,3,3,3,
4,4,4,4,
5,5,5,5)
, 5, 4, byrow=TRUE))

# set margins
par(mgp=c(0, 0.1, 0))

# plot sampleA coverage
par(mar=c(0.5,4,0.5,4))
plotBedgraph(sA.covs, chrom, chromstart, chromend, color=SushiColors(2)(2)[1], ymax=1.5)
axis(side=2,las=2,tcl=.2)
mtext("Read Depth",side=2,line=1.75,cex=.5,font=2)
legend("topright",inset=0,legend=c("Sample A","Sample B"),fill=SushiColors(2)(2),
→border=SushiColors(2)(2),text.font=2,cex=0.75)

# plot sampleA FitHiChIP 5kb loops
par(mar=c(0.5,4,0.5,4))
plotBedpe(sA.arc, chrom, chromstart, chromend, heights = sA.arc$sumCC, plottype="loops",␣
→flip=TRUE, color=SushiColors(2)(2)[1], ymax=4)
axis(side=2,las=2,tcl=.2)
mtext("# Contacts",side=2,line=1.75,cex=.5,font=2)

# plot sampleB coverage
par(mar=c(0.5,4,0.5,4))
plotBedgraph(sB.covs, chrom, chromstart, chromend, color=SushiColors(2)(2)[2], ymax=1.8)
axis(side=2,las=2,tcl=.2)
mtext("Read Depth",side=2,line=1.75,cex=.5,font=2)

# plot sampleB FitHiChIP 5kb loops
par(mar=c(0.5,4,0.5,4))
plotBedpe(sB.arc, chrom, chromstart, chromend, heights = sB.arc$sumCC, plottype="loops",␣
→flip=TRUE, color=SushiColors(2)(2)[2], ymax=1)
axis(side=2,las=2,tcl=.2)
mtext("# Contacts",side=2,line=1.75,cex=.5,font=2)

# plot gene track
par(mar=c(2,4,2,4))
plotGenes(genes , chrom,chromstart,chromend, types=genes$type, maxrows=20,bheight=0.08,␣
→plotgenetype="box",bentline=TRUE,col="black",␣
→labeloffset=.5,fontsize=0.9,arrowlength = 0.2,labeltext=TRUE)
lablegenoma( chrom, chromstart,chromend,n=3,yscale="Mb")

# turn off plotter
if(makepdf == TRUE)
{
  dev.off()
}

# leave R
q()
Figure 7. A) The output from plotting in Sushi. B) The image cleaned up in the pdf editor Adobe Illustrator.
1.7.11 Final thoughts and considerations

What if you don’t have paired ChIP-seq experiments? Best practices in HiChIP analyses is to have paired ChIP-seq experiments. This is because these approaches ask different molecular biology questions:

- ChIP = Where is protein bound to DNA?
- HiChIP = where is protein bound to DNA and what other loci are interacting with it?

This subtle difference means that HiChIP as enrichment signal falling outside of the target protein binding site. If there is no way to access ChIP data you can identify chromatin-IP peaks in the HiChIP libraries, but you must get over two issues:

- The long-range information in the sequence data breaks the assumption peak callers have about the structure of the paired-end read (they expect insert size of <1kb, where HiChIP has many PE reads with insert size >1kb)
- The off-target enrichment associated with interacting loci looks like increase background noise to Peak Callers

To overcome the paired-end insert size assumption - we recommend to isolate the primary alignment from the bam and convert into bed format then feed into the peak caller. These results should be used with caution as there is still a background noise problem. Therefore, it is a good idea to filter these peaks to take only the top 75th percentile of the Q-scores to only take the “good peaks”.

Commands:

Get primary alignment from the bam file and convert to bed format

```bash
samtools view -@24 -h -F 0x900 sampleA.bam | bedtools bamtobed -i stdin > sampleA.primary.aln.bed
samtools view -@24 -h -F 0x900 sampleB.bam | bedtools bamtobed -i stdin > sampleB.primary.aln.bed
```

Get primary alignment from the bam file and convert to bed format

```bash
macs2 callpeak -t sampleA.primary.aln.bed -n sampleA --nomodel
macs2 callpeak -t sampleB.primary.aln.bed -n sampleB --nomodel
```

Considerations:

- Replication – In this experiment biological replication is not used, but technical replication was used to minimize the number PCR duplicates and maximize the amount use-able data from the sequencer. If you use biological replicates, you should loop call independently on each replicate then you can apply the same differential looping analyses to identify a core-set of loops that are shared across all replicates within a condition then move on to differential looping between condition A and condition B
- Paired ChIP-seq experiments – As mentioned above, it is best practices to have paired ChIP-seq experiments. If that is not do-able, there are ways to work around it, you just need to be aware of the data structure and what the molecular biology is behind the sequence data you are using to call peaks.
- Workflow efficiency – The walkthrough guide that accompanies this workshop is meant to be detailed breakdown of the step-by-step ‘recipe’ for how to work through HiChIP data. There are certainly ways to improve upon this or wrap part of this into little R scripts that perform many steps in one executable. Please feel free to do so!
- What is your question – it is always good to keep your biological question in mind. This guide is meant to help address a set of questions that are commonly asked about loop analysis. There are, of course, many other questions one could ask of these data. Hopefully, this workshop empowers you to ask your own question now you’re comfortable with the data structure!
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.

![Plot of HiChIP data with coverage and contact distance](image)

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 recommend 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
R
```

3. Load libraries

Command:

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

4. Load data

Command:

```R
cov <- read.table("prefix.coverage.bedgraph")
arc <- read.table("prefix_interactions_FitHiC_Q0.01_MergeNearContacts.bed", header=TRUE)
```

5. Inspect arc file structure

Command:

```R
head(arc)
```

1.8. Plotting HiChIP Data
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:

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

7. Inspect arc file to see distance

Command:

```r
head(arc)
```

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

Command:

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

9. Inspect coverage plot

Command:

```r
plotBedgraph(cov,chrom,chromstart,chromend)
```

```r
labelgenome(chrom,chromstart,chromend,n=4,scale="Mb")
```

```r
mtext("Read Depth",side=2,line=1.75,cex=1,font=2)
```

```r
axis(side=2,las=2,tcl=.2)
```
10. Plot arcs with arc heights based on distance

Command:

```r
plotBedpe(arc, chrom, chromstart, chromend, heights = arc$dist, 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)
```

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:

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

(continues on next page)
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:**

```r
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=.75,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):

![Figures](image)

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 \textit{FitHiChIP} on both peak-to-peak and peak-to-all settings.
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 recommend the GM12878 CTCF (deep sequencing) from our publically 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 function 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 piped 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:

<table>
<thead>
<tr>
<th>Library</th>
<th>Link</th>
</tr>
</thead>
</table>
| 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

<table>
<thead>
<tr>
<th>Sample</th>
<th>Target</th>
<th>Accession</th>
<th>URL</th>
<th>Output type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM12878</td>
<td>CTCF</td>
<td>ENCFF017XLW</td>
<td><a href="https://www.encodeproject.org/files/ENCFF017XLW/@@download/ENCFF017XLW.bed.gz">https://www.encodeproject.org/files/ENCFF017XLW/@@download/ENCFF017XLW.bed.gz</a></td>
<td>conservative IDR thresholded peaks</td>
</tr>
<tr>
<td>IMR-90</td>
<td>H3K4ac</td>
<td>ENCFF823NUO</td>
<td><a href="https://www.encodeproject.org/files/ENCFF823NUO/@@download/ENCFF823NUO.bed.gz">https://www.encodeproject.org/files/ENCFF823NUO/@@download/ENCFF823NUO.bed.gz</a></td>
<td>replicated peaks</td>
</tr>
<tr>
<td>GM12878</td>
<td>H3K4me3</td>
<td>ENCFF188SZS</td>
<td><a href="https://www.encodeproject.org/files/ENCFF188SZS/@@download/ENCFF188SZS.bed.gz">https://www.encodeproject.org/files/ENCFF188SZS/@@download/ENCFF188SZS.bed.gz</a></td>
<td>replicated peaks</td>
</tr>
<tr>
<td>IMR-90</td>
<td>H3K14ac</td>
<td>ENCFF106EAN</td>
<td><a href="https://www.encodeproject.org/files/ENCFF106EAN/@@download/ENCFF106EAN.bed.gz">https://www.encodeproject.org/files/ENCFF106EAN/@@download/ENCFF106EAN.bed.gz</a></td>
<td>replicated peaks</td>
</tr>
<tr>
<td>GM12878</td>
<td>H3K27ac</td>
<td>ENCFF367KIF</td>
<td><a href="https://www.encodeproject.org/files/ENCFF367KIF/@@download/ENCFF367KIF.bed.gz">https://www.encodeproject.org/files/ENCFF367KIF/@@download/ENCFF367KIF.bed.gz</a></td>
<td>replicated peaks</td>
</tr>
<tr>
<td>GM12878</td>
<td>H3K27me3</td>
<td>ENCFF153VOQ</td>
<td><a href="https://www.encodeproject.org/files/ENCFF153VOQ/@@download/ENCFF153VOQ.bed.gz">https://www.encodeproject.org/files/ENCFF153VOQ/@@download/ENCFF153VOQ.bed.gz</a></td>
<td>replicated peaks</td>
</tr>
<tr>
<td>GM12878</td>
<td>SMC3</td>
<td>ENCFF534PUK</td>
<td><a href="https://www.encodeproject.org/files/ENCFF534PUK/@@download/ENCFF534PUK.bed.gz">https://www.encodeproject.org/files/ENCFF534PUK/@@download/ENCFF534PUK.bed.gz</a></td>
<td>bed</td>
</tr>
<tr>
<td>MCF-7</td>
<td>Klf4</td>
<td>ENCFF287QDZ</td>
<td><a href="https://www.encodeproject.org/files/ENCFF287QDZ/@@download/ENCFF287QDZ.bed.gz">https://www.encodeproject.org/files/ENCFF287QDZ/@@download/ENCFF287QDZ.bed.gz</a></td>
<td>conservative IDR thresholded peaks</td>
</tr>
<tr>
<td>GM23338</td>
<td>Nanog</td>
<td>ENCFF897LBK</td>
<td><a href="https://www.encodeproject.org/files/ENCFF897LBK/@@download/ENCFF897LBK.bed.gz">https://www.encodeproject.org/files/ENCFF897LBK/@@download/ENCFF897LBK.bed.gz</a></td>
<td>conservative IDR thresholded peaks</td>
</tr>
<tr>
<td>GM12878</td>
<td>POLR2A</td>
<td>ENCFF794VYB</td>
<td><a href="https://www.encodeproject.org/files/ENCFF794VYB/@@download/ENCFF794VYB.bed.gz">https://www.encodeproject.org/files/ENCFF794VYB/@@download/ENCFF794VYB.bed.gz</a></td>
<td>conservative IDR thresholded peaks</td>
</tr>
</tbody>
</table>

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:

```bash
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:

```bash
wget https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/Reference_Genome/mm10.fa
```
### Data Set

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fastqs (Sample A)</td>
<td>• <a href="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_R1.fastq.gz</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/fastq_inputs/sampleA_R2.fastq.gz">https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/fastq_inputs/sampleA_R2.fastq.gz</a></td>
</tr>
<tr>
<td>Fastqs (Sample B)</td>
<td>• <a href="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_R1.fastq.gz</a></td>
</tr>
<tr>
<td></td>
<td>• <a href="https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/fastq_inputs/sampleB_R2.fastq.gz">https://s3.amazonaws.com/dovetail.pub/HiChIP/compare_samples/fastq_inputs/sampleB_R2.fastq.gz</a></td>
</tr>
</tbody>
</table>

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