Processing HiChIP Data at OMRF

Caleb Lareau

Last updated 2016-10-10

About

Welcome to the wonderful world of HiChIP preprocessing! This vignette is designed to walk through the steps needed to process HiChIP datasets on the OMRF `ash` server. Before running this guide, make sure you have the latest available version of the vignette available here. If you have any questions/errors, feel free to contact Caleb Lareau.

The current state of HiChIP preprocessing is an *ad hoc* melting of several tools original designed for ChIAPET and HiC, including HiC Pro, ChIAPET2, and some in house scripts for QC metrics. One hopes that these will be better packaged together one day, but the current state necessitates a spaghetti of shell scripting and/or sequential commands. Thus, check back in case the process gets streamlined.

Alignment with Hi-C Pro/Bowtie2

Hi-C Pro is super useful but admittedly a pain to get working correctly. Here's a sample execution. First, load the module.

```
module load hic-pro
```

Next, navigate to a directory where a configuration file is present as well as the sample `.fastq` files are contained in a folder hierarchy. Here is an example configuration file for the MboI enzyme digestion aligned to hg19 with file extensions `*_1.fastq, *_2.fastq`. Your working directory should look something like this–

```
fastq/
|-- example
|   |-- exp_1.fastq.gz
|   |-- exp_2.fastq.gz
```

Hi-C Pro can run multiple samples at once with multiple `.fastq` files each as long as 1) each sample has it's own directory and 2) the file naming convention is consistent (*i.e.* `_1.fastq, _2.fastq`). Files can be `gzipped` or not; it doesn't matter. Okay, so now we run the first Hi-C Pro command to set up our analyses.

```
HiC-Pro -i fastq -c config-hicpro-mboi-ext12.txt -o output -p
```

The `-i` flag specifies the directory that contains the per-sample directories; the `-c` flag specifies the configuration file; `-o` flag creates an output directory for everything Hi-C Pro related; and the `-p` flag ensures allows the execution to be performed on the server rather than the session. You should see some message about qsub'ing some shell scripts if your configuration file took. Otherwise, you may need to edit the file specifications.

```
cd output
```

You should see two shell scripts, `HiCPro_step1_hic.sh` and `HiCPro_step2_hic.sh`.

```
1) EDIT the -M command with your username to get jobs
2) DELETE -l h_rt= command line
```

```
qsub HiCPro_step1_hic.sh
```

Creating Hi-C Output

While I would argue that really useful output of HiChIP are loop calls, one may be interested in the Hi-C heatmap. To get this, we need to run Step 2 of HiC-Pro, which should be super simple if you've made it this far. Simply 1) make the same modifications to `HiCPro_step2_hic.sh` as above and 2) execute `qsub HiCPro_step2_hic.sh` to get ordinal Hi-C output.

Calling Loops

```
module load samtools
samtools view -o exp_1.sam exp_1.hg19.bwt2merged.bam
samtools view -o exp_2.sam exp_2.hg19.bwt2merged.bam
```

```
module load chia-pet2
buildBedpe exp_1.sam exp_2.sam exp.bedpe 30 1 0
removeDup exp.bedpe exp_c.bedpe 1
buildTagAlign exp_c.bedpe exp.tag.bed
```

```
module load macs2
macs2 callpeak -t exp.tag.bed -g hs -f BED -n exp -q 0.10
```

```
module load bedtools
awk '{print $1"\t"$2"\t"$3"\t"$4}' exp_peaks.narrowPeak | bedtools sort |
  bedtools slop -g /usr/local/analysis/hic-pro/2.7.8/annotation/chrom_hg19.sizes -b 1000 |
  mergeBed -d 1000 > slopPeak.bed
```

Download the filtBedPe.py file and execute

```
python filtBedPe.py -input exp.tmp.bedpe -prefix exp -mindist 5000
```

QC Report