hichipper Documentation

Release 0.6.0

Caleb Lareau

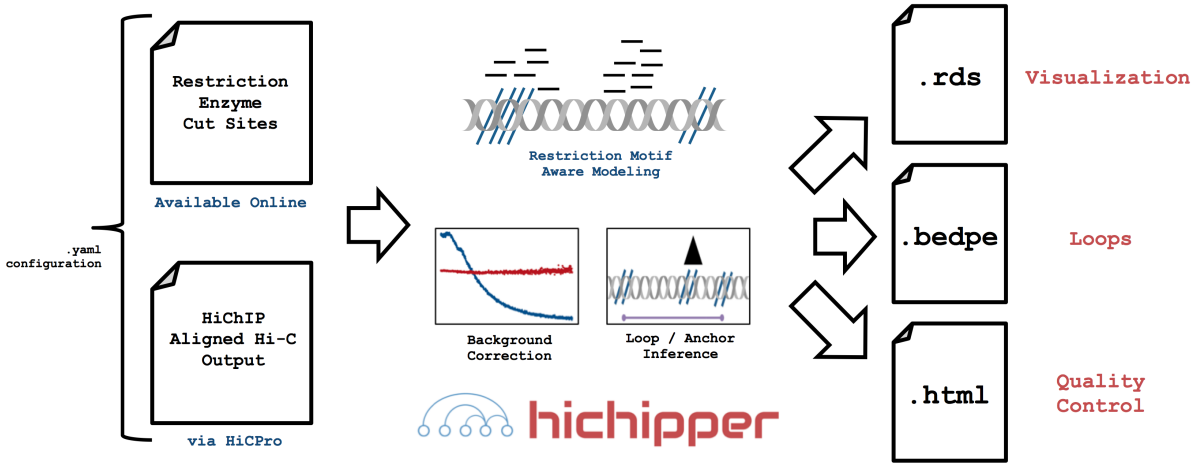
Sep 02, 2017

Contents

CHAPTER 1

About

hichipper is an open-source command-line toolkit that performs restriction fragment bias-aware preprocessing of HiChIP data. This package takes output from a HiC-Pro run and a sample manifest file (`.yaml`) that coordinates optional high-quality peaks (identified through ChIP-Seq) and restriction fragment locations as input and produces output that can be used to 1) determine library quality, 2) identify and characterize DNA loops and 3) interactively visualize loops. Loops are assigned strength and confidence metrics that can be used to evaluate samples individually or for differential analysis in downstream tools.



CHAPTER 2

Installation

Install stable version through PyPi

There are a few dependencies needed to get **hichipper** to run. All are very common bioinformatics tools / languages and should be readily available in most systems. However, **note that the current implementation of hichipper is not supported on Windows platforms**.

Depending on your python environment, we generally recommend using a virtual environment to keep python dependencies tidy. An example of installing **hichipper** inside a new python virtual environment called `venv` using the following sequence of commands–

```
virtualenv -p /usr/bin/python2.7 venv
source venv/bin/active
pip install hichipper
hichipper --version
```

Install via GitHub

Though **not recommended**, a bleeding-edge (development) version can be installed directly from Git. Again using a virtual environment–

```
virtualenv -p /usr/bin/python2.7 venv
source venv/bin/active
pip3 install git+ssh://git@github.com/aryeelab/search/tree/master/hichipper
```

While installing **hichipper** is obviously a great first step, make sure that all of the dependencies are met. Check out the next page for more detail.

CHAPTER 3

Dependencies

Required software for hichipper

The following dependencies need to be installed before running **hichipper**: bedtools, OpenSSL, libcurl, libxml2, and samtools. Depending on if you want some bonus functionality, you may need to download additional requirements.

Except for `bedtools` and `samtools`, these other dependencies came out of the box with the unix/linux systems that we've used **hichipper** on.

But just to be safe, on an Ubuntu system, all of the dependencies can be installed with:

```
apt-get install bedtools libssl-dev libcurl4-openssl-dev libxml2-dev
```

Additionally, `R` must be available in the environment as well as a reasonably recent version of pandoc and a few packages that can be downloaded running the following in an 'R' environment. :

```
install_pkgs <- function(pkg){
    new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
    if (length(new.pkg)) install.packages(new.pkg, dependencies = TRUE)
}
install_pkgs(c("DT", "data.table", "devtools", "foreach", "ggplot2", "knitr",
↪"networkD3", "readr", "reshape2"))

source("https://bioconductor.org/biocLite.R")
install_pkgs_bioc <- function(pkg){
    new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
    if (length(new.pkg)) biocLite(new.pkg, dependencies = TRUE)
}
install_pkgs_bioc(c("diffloop"))
```

or simply download this R script and run:

```
Rscript requirementsInstall.R
```

Convenient pandoc binaries for Linux, Mac and Windows are available for download from RStudio. If you prefer to install pandoc globally on your machine, installation instructions can be found here.

CHAPTER 4

Usage

Simple usage example

The example below uses the test dataset bundled with the **hichipper** package source code. Download the package, change to the test directory, and execute the basic working example with:

```
git clone https://github.com/aryeelab/hichipper.git
cd hichipper/tests
hichipper --out output1 yaml/one.yaml
```

Here's a more detailed description of what just happened. First, we had to create a sample description file that specifies how peaks are to be inferred (in this example, they are pre-specified from a ChIP-Seq experiment). Next, one must specify the location of a restriction fragment file. Finally, a path to the HiC-Pro output folder must be designated. These are encoded through the `peaks`, `resfrags`, and `hicpro_output` variables that will be parsed from the `.yaml` format.

1. Create a sample description file:

Description files can be created with the `.yaml` format.

Processing `.yaml` format

Example yaml format sample description file:

```
peaks:
 - chipseq/GM12878_SMC3_ChIPSeq_chr22.narrowPeak
resfrags:
 - resfrag/hg19_MboI_resfrag_chr22.bed.gz
hicpro_output:
 - hicpro
```

Note: This file is available as `example.yaml` in the `hichipper/tests` directory.

In this example, we call loops from two GM12878 samples using just chromosome 22 using pre-determined peaks from a ChIP-Seq file.

1. Run the pipeline:

```
hichipper --out output1 yaml/one.yaml
```

Additional details concerning user configuration options are shown below.

More typical example

While the example above references files that are part of the **hichipper** distribution, our experience using this tool in conjunction with HiC-Pro suggests that a file hierarchy like the following may be more typical.

```
ls -LR
```

yields (a slightly modified version of)

```
./hicpro/bowtie_results:
bwt2  bwt2_global  bwt2_local

./hicpro/bowtie_results/bwt2:
SRR3467175  SRR3467176  SRR3467177  SRR3467178

./hicpro/bowtie_results/bwt2/SRR3467175:
SRR3467175_1_hg19.bwt2merged.bam  SRR3467175_2_hg19.bwt2merged.bam  SRR3467175_hg19.
↪bwt2pairs.bam
SRR3467175_1_hg19.mapstat         SRR3467175_2_hg19.mapstat         SRR3467175_hg19.
↪bwt2pairs.pairstat*

./hicpro/bowtie_results/bwt2/SRR3467176:
SRR3467176_1_hg19.bwt2merged.bam  SRR3467176_2_hg19.bwt2merged.bam  SRR3467176_hg19.
↪bwt2pairs.bam
SRR3467176_1_hg19.mapstat         SRR3467176_2_hg19.mapstat         SRR3467176_hg19.
↪bwt2pairs.pairstat*

./hicpro/bowtie_results/bwt2/SRR3467177:
SRR3467177_1_hg19.bwt2merged.bam  SRR3467177_2_hg19.bwt2merged.bam  SRR3467177_hg19.
↪bwt2pairs.bam
SRR3467177_1_hg19.mapstat         SRR3467177_2_hg19.mapstat         SRR3467177_hg19.
↪bwt2pairs.pairstat*

./hicpro/bowtie_results/bwt2/SRR3467178:
SRR3467178_1_hg19.bwt2merged.bam  SRR3467178_2_hg19.bwt2merged.bam  SRR3467178_hg19.
↪bwt2pairs.bam
SRR3467178_1_hg19.mapstat         SRR3467178_2_hg19.mapstat         SRR3467178_hg19.
↪bwt2pairs.pairstat*

./hicpro/hic_results:
data

./hicpro/hic_results/data:
SRR3467175  SRR3467176  SRR3467177  SRR3467178

./hicpro/hic_results/data/SRR3467175:
SRR3467175_hg19.bwt2pairs.DEPairs*    SRR3467175_hg19.bwt2pairs.RSstat*   SRR3467175_
↪hg19.bwt2pairs.SinglePairs*
SRR3467175_hg19.bwt2pairs.DumpPairs*  SRR3467175_hg19.bwt2pairs.SCPairs*  SRR3467175_
↪hg19.bwt2pairs.validPairs*
```

```
SRR3467175_allValidPairs*

./hicpro/hic_results/data/SRR3467176:
SRR3467176_hg19.bwt2pairs.DEPairs*    SRR3467176_hg19.bwt2pairs.RSstat*   SRR3467176_
↪hg19.bwt2pairs.SinglePairs*
SRR3467176_hg19.bwt2pairs.DumpPairs*  SRR3467176_hg19.bwt2pairs.SCPairs*  SRR3467176_
↪hg19.bwt2pairs.validPairs*
SRR3467176_allValidPairs*

./hicpro/hic_results/data/SRR3467177:
SRR3467177_hg19.bwt2pairs.DEPairs*    SRR3467177_hg19.bwt2pairs.RSstat*   SRR3467177_
↪hg19.bwt2pairs.SinglePairs*
SRR3467177_hg19.bwt2pairs.DumpPairs*  SRR3467177_hg19.bwt2pairs.SCPairs*  SRR3467177_
↪hg19.bwt2pairs.validPairs*
SRR3467177_allValidPairs*

./hicpro/hic_results/data/SRR3467178:
SRR3467178_hg19.bwt2pairs.DEPairs*    SRR3467178_hg19.bwt2pairs.RSstat*   SRR3467178_
↪hg19.bwt2pairs.SinglePairs*
SRR3467178_hg19.bwt2pairs.DumpPairs*  SRR3467178_hg19.bwt2pairs.SCPairs*  SRR3467178_
↪hg19.bwt2pairs.validPairs*
SRR3467178_allValidPairs*

...
```

where files denoted in with an asterisk* are assumed to exist. Typically, an analysis folder may look like so:

```
fastq/
|-- SRR3467175
|   |-- SRR3467175_1.fastq.gz
|   |-- SRR3467175_2.fastq.gz
|-- SRR3467176
|   |-- SRR3467176_1.fastq.gz
|   |-- SRR3467176_2.fastq.gz
|-- SRR3467177
|   |-- SRR3467177_1.fastq.gz
|   |-- SRR3467177_2.fastq.gz
|-- SRR3467178
|   |-- SRR3467178_1.fastq.gz
|   |-- SRR3467178_2.fastq.gz
hicpro/
|-- HiCPro_step1_hic.sh
|-- bowtie_results/
|   |-- bwt2/
|   |   |-- SRR3467175
|   |   |   |-- SRR3467175_hg19.bwt2pairs.pairstat
|   |   |-- SRR3467176
|   |   |   |-- SRR3467176_hg19.bwt2pairs.pairstat
|   |   |-- SRR3467177
|   |   |   |-- SRR3467177_hg19.bwt2pairs.pairstat
|   |   |-- SRR3467178
|   |   |   |-- SRR3467178_hg19.bwt2pairs.pairstat
|-- hic_results/
|   |-- data/
|   |   |-- SRR3467175
|   |   |   |-- SRR3467175*RSstat
|   |   |   |-- SRR3467175*Pairs # 5 Files
|   |   |   |-- SRR3467175_allValidPairs
```

```
|   |   |-- SRR3467176
|   |   |   |-- SRR3467176*RSstat
|   |   |   |-- SRR3467176*Pairs # 5 Files
|   |   |   |-- SRR3467175_allValidPairs
|   |   |-- SRR3467177
|   |   |   |-- SRR3467177*RSstat
|   |   |   |-- SRR3467177*Pairs # 5 Files
|   |   |   |-- SRR3467175_allValidPairs
|   |   |-- SRR3467178
|   |   |   |-- SRR3467178*RSstat
|   |   |   |-- SRR3467178*Pairs # 5 Files
|   |   |   |-- SRR3467175_allValidPairs
GM12878_SMC3_ChIPSeq.narrowPeak
hg19_MboI_resfrag.bed.gz
yaml/
|-- one.yaml
config-hicpro-mboi-ext12.txt
```

where the results in the `hicpro` directory could have been obtained by running:

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

and subsequently executing the resulting `HiCPro_step1_hic.sh` and `HiCPro_step2_hic.sh`.

Thus, the `yaml/one.yaml` file needed for **hichipper** when executed from the current working directory would look like this:

```
peaks:
  - GM12878_SMC3_ChIPSeq.narrowPeak
resfrags:
  - hg19_MboI_resfrag.bed.gz
hicpro_output:
  - hicpro
```

And could be executed running this command:

```
hichipper --out GM12878 config.yaml
```

would yield the default output from **hichipper**.

CHAPTER 5

Version control

Noteworthy changes in hichipper

Split .fastq files as input to HiC-Pro

As of version `0.5.3` of **hichipper**, users should be able to input split `.fastq` files into HiC-Pro and have **hichipper** function properly. No extra user flags are needed for this functionality. Thanks to our early users for helping us figure this out.

A note on duplicates

In certain versions (`0.4.4` to `0.5.3`) of **hichipper**, duplicates were not being filtered out by default.

CHAPTER 6

Configuration

hichipper parameters

Running

```
hichipper --help
```

shows the parameters that can be used in this software package as reproduced below.

```
Usage: hichipper [OPTIONS] MANIFEST

  A preprocessing and QC pipeline for HiChIP data.

Options:
  --out TEXT                    Output directory name; must not be an already␣
↪existing directory [Required]
  --min-dist TEXT               Minimum distance; default = 5000
  --max-dist TEXT               Peak padding width (applied on both left and
                                right); default = 2000000
  --macs2-string TEXT           String of arguments to pass to MACS2; only is
                                called when peaks are set to be called;
                                default = "-q 0.01 --extsize 147 --nomodel"
  --macs2-genome TEXT           Argument to pass to the -g variable in MACS2
                                (mm for mouse genome; hs for human genome);
                                default = "hs"
  --peak-pad TEXT               Peak padding width (applied on both left and
                                right); default = 500
  --merge-gap TEXT              Merge nearby peaks (after all padding is
                                complete); default = 500
  --keep-temp-files             Keep temporary files?
  --skip-background-correction  Skip restriction fragment aware background
                                correction?
  --skip-resfrag-pad            Skip restriction fragment aware padding
  --skip-qc                     Skip QC report generation?
  --skip-diffloop               Skip analyses in diffloop (e.g. Mango loop
```

```
                                calling; .rds generation)
  --make-ucsc                   Make additional output files that can support
                                viewing in UCSC genome browser; requires tabix
                                and htslib tools.
  --keep-samples TEXT           Comma separated list of sample names to keep;
                                ALL (special string) by default
  --ignore-samples TEXT         Comma separated list of sample names to
                                ignore; NONE (special string) by default
  --read-length TEXT            Length of reads from sequencing runs; default = 75
  --version                     Show the version and exit.
  --help                        Show this message and exit.
```

Running

```
hichipper --version
```

will show the version of this package currently installed.

```
hichipper, version 0.6.0
```

Check the badge up top to see if a newer version is available or try directly through `pip`:

```
pip install hichipper --upgrade
```

Unless these flags are supplied, the pipeline will attempt to run. Minimally sufficient parameters include the `--out` flag and a `.yaml` file as shown in the example executions. Below are some explanations of the additional parameters than can be configured when executing the pipeline.

Restriction-fragment aware padding

Parameter explanations

Most of the parameter options are fairly straight forward. Running `hichipper --version` or `hichipper --help` doesn't run the tool but supplies the information noted above. Otherwise, the default run mode requires a `.yaml` file supplied in addition to the `--out` parameter, which specifies the output directory of the run. Users can decide to customize final output by using boolean flags or supply variable text input. The following cartoon shows a graphical overview of important parameters to consider when running **hichipper**.

As noted in orange, defined peaks are automatically padded by some integer width from the `--peak-pad` flag. By default, this pad extends 500 base pairs in either direction. Padding the peaks boosts the number of PETs that can be mapped to loops. For example, `PET II` would not be considered in loop since the left end of the read does not overlap with the called peak (black). However, it does overlap with the padded peak, so it is retained with padding. When two peaks are close to one another, they may be merged using the `--merge-gap` command. As suggested in the figure, the padded peaks `B` and `C` may be sufficiently close to be merged into a single anchor. Note that this can lead to some PETs becoming self-ligation (e.g. `I-III`). Note, the `--merge-gap` command is equivalent to running bedtools merge -d on the padded anchors. By default, the `merge-gap` is 500 base pairs. Specifying this to `0` can cause issues, particularly when the width of a `PET` spans multiple peaks.

The `dist` or distance between two peaks is noted in black as the center of two peaks. The `--min-dist` flag is the smallest and `--max-dist` is the largest integer number that ensures this distance falls between to be considered in a loop. These defaults are 5Kb and 2Mb as smaller reads are likely self-ligations whereas larger reads are unlikely to be biologically real loops.

From our inspection of the HiChIP data, we determined that reads for putative loops localize to the edges of restriciton fragments and that increasing the padding to the edges of fragments can prove beneficial for maximizing the read density associated with loops. By default, **hichipper** adds additional padding to the edges of restriction fragments

unless the `--skip-resfrag-pad` pad is thrown. To give an example, consider peak `C`. Under the default options, the anchor corresponding to this peak would span restriction fragments `7` and `9` (assuming that the `merge-gap` is small enough such that `B` and `C` are not merged). This is because the additional restriction fragment padding occurs after the original padding (orange), and **hichipper** padds to the edges of whatever fragment pads are overlapping the peak. However, if the user specified `--peak-pad 0`, the anchor corresponding to `C` would only span restriction motifs `8` and `9` since the peak only sits on that singular fragment. While we have found the default options to be sensible, the user can adjust these padding parameters to potentially increase precision of anchors calls possibly at the expense of PET density or vice-versa.

Finally, we note the `--macs2-genome` and `--macs2-string` which by default has parameters that we felt were suited appropriately for processing HiChIP data. However, users can modify these when performing peak-calling from HiChIP data directly.

User parameter recommendations

- If `R` is not in the system or if the `R` package dependencies could not be installed, the following flags should be added:

```
--skip-resfrag-pad --skip-diffloop --skip-qc --skip-background-correction
```

- In the current version of **hichipper**, the novel background correction implementation is quite memory intense. Thus, *users running **hichipper** on a laptop or other low RAM machine* should likely skip the adaptive background correction.

```
--skip-background-correction
```

CHAPTER 7

Peaks

Peaks with hichipper

To call peaks from HiChIP data directly, **hichipper** aggregates read density from either all samples or each sample individually. Additionally, users can specify whether all read density is used or if only self-ligation reads are used. To specify these options, put the appropriate string of the form `{COMBINED,EACH},{ALL,SELF}` in the `peaks` slot of the `.yaml`.

For example, to replicate the peak calling performed in Mumbach *et al.*, one would use the following `.yaml`:

```
peaks:
  - COMBINED,SELF
resfrags:
  - hg19_MboI_resfrag.bed.gz
hicpro_output:
  - hicpro
```

Alternatively, we can call peaks from the HiChIP data for each sample individually using all reads using this specification–

```
peaks:
  - EACH,ALL
resfrags:
  - hg19_MboI_resfrag.bed.gz
hicpro_output:
  - hicpro
```

The figure below shows all options for peak specification in **hichipper** including every option for inferring peaks which are noted in the table.

Alternatively, users can pre-specify a set of peaks to used. In this case, a "connectome" will be inferred between the peaks specified in the `.bed` file. Of note, pre-specified peaks will still be padded either by fixed amounts or to the edges of the restriction fragment pads (or both) unless the user specifies these flags differently (see below).

```
peaks:
  - predeterminedPeaks.bed
resfrags:
  - hg19_MboI_resfrag.bed.gz
hicpro_output:
  - hicpro
```

Note: the input of pre-determined peaks does not have to explicitly be a `.bed` file. Rather, any file name is acceptable so long as the first three columns indicate appropriate genomic loci as if it were a `.bed` file. For example, `.narrowPeak` files from `macs2` should be fine.

Multiple ChIP-Seq peaks as input

As raised in this issue, if you have multiple samples and multiple ChIP-Seq or related high-quality peak definitions to be used as an input, the way to do this is to create two or more `.yaml` files, each one specifying its own bed file of peaks. Then, execute `hichipper` such that you restrict the analysis to the sample you want per bed file using the `--keep-samples` or `--ignore-samples` flags. Thanks to user **sb5169** for bringing this up.

HiChIP-Specific Bias Correction

A key difference of HiChIP data compared to ChIA-PET, ChIP-Seq, and related immunoprecipitation assays is the a notable bias where a greater read density accumulates near the motif used in the restriction enzyme digestion. The image below shows the ratio of the treatment to the background (the statistic used in macs2 to call peaks) as a function of distance to the nearest restriction fragment locus. Note the plot below–

A more detailed description of this bias and our analysis is contained in this writeup.

CHAPTER 8

Output

Output

Per-run output files

Each time the user runs **hichipper**, a `*.hichipper.log` file containing information pertaining to the flow of the software execution is placed in the `out` directory (specified by the `--out` flag). Unless otherwise specified, a file ending in `hichipper-qcReport.html` provides an interactive quality control report for all samples.

Per-sample output files

Per sample, six (yes, 6, but don't worry– there's lots of redundancy) output files are created. They are:

1. `*.stat` Key summary statistics that show the number of PETs meeting certain criteria
2. `*.inter.loop_counts.bedpe` Interchromosomal looping between anchor loci.
3. `*.intra.loop_counts.bedpe` Intrachromosomal looping between **all** anchor loci
4. `*.filt.intra.loop_counts.bedpe` Intrachromosomal looping between anchor loci where loops meet min/max distance requirements.
5. `*interactions.all.mango` The same set of loops as 4 but with per-loop FDR measures from the loop proximity bias correction algorithm originally implemented in Mango and presented in the same format.
6. `*.rds` The same set of loops as 4 but in an R binary compressed format of a `loops()` S4 object from diffloop. Can immediately be imported for interactive visualization in DNAlandscapeR.

So, outputs 4, 5, and 6 are identical except in presentation. These data are a subset of those presented in 3. Interchromosomal interactions from 2 are often discarded by other preprocessing pipelines, but they may hold value. If the `qcReport` is generated, then the `.stat` file won't tell you anything new. However, if `R` is not installed on your machine, this will be a useful file for assessing the quality of your library.

CHAPTER 9

Differential

Finding differential loops

Have you generated a bunch of HiChIP samples and want to see what's different between them? Check out the diffloop vignette for an example analysis comparing loops from ChIA-PET (a similar 3C method to HiChIP) between K562 and MCF-7. Installation instructions for this package are shown in the dependencies section.

CHAPTER 10

Visualization

Visualizing loops

Interactive visualization of loops

One you've (hopefully) assessed that your samples look good, now go visualize them! One option is to link the `.bedpe` file to the WashU Genome Browser. Another option is to upload the `.rds` to our genome topology browser, DNAlandscapeR. Navigate to the **Guide** tab to get a sense of how the browser works and ultimately add your sample(s) to a local user session using the **Import** tab. Note: the browser currently supports hg19/hg37 and mm9 genome builds.

Visualization in UCSC

Users can specify the `--make-ucsc` flag to produce output that can be imported into UCSC. See this discussion for an overview of the format. In order to produce this output, **hichipper** needs access to the htslib suite of tools in the computational environment. You can see if you have these dependences available (namely, `tabix` and `bgzip`) by making sure the following works:

```
tabix --version
```

Specifying this flag will create the additional files `*.txt.gz` and `.txt.gz.tbi`, which can be used to make a UCSC track. (Shout out to Gary for helping us with this!)

CHAPTER 11

Author

The primary developer is Caleb Lareau in the Aryee Lab.

CHAPTER 12

Citation

If you use **hichipper** in your research, please cite our tool at the following URL:

`http://aryeelab.org/hichipper`

CHAPTER 13

Bugs / Errors

Please let us know if you find any errors/inconsistencies in the documentation or code by filing a new Github Issue.