HiCHap

HiCHap provide a Python CPU-based implementation for Hi-C pipeline. Tranditional Hi-C pipeline and Haplotype Hi-C pipeline are both available.

More details in Github : (https://github.com/Prayforhanluo/HiCHap_master)

Requirement

1. Python 2.7+
2. Multiprocess
3. Numpy
4. Scipy
5. statsmodels
6. Scikit-Learn
7. xml
8. mirnylib
9. pysam
10. ghmm
11. Bio
12. bowtie2 (we used 2.2.9)
13. samtools (we used 1.5)

Using **conda** , **pip** or their source code to install them if neccessary.

Install HiCHap

Now just download the HiCHap source code and gunzip the files from PyPI or Github, and run the setup.py script::

```
$ python setup.py install
```

OR

```
$ pip install HiCHap
```

By the way, When I developed HiCHap, I have found a strange bug, that is when it has been installed. I use the hichap to get help information. But it bugs with **"pkg_resources.ResolutionError::"**. I just figure out how to fix it but i can't figure out why it happens. The fixing code is in the setup.py at the last few lines. If anyone can tell me how to avoid this bug. Please connect with me (hluo_lc@outlook.com) and thanks so much :)

HiCHap will be installed successfully if no exception occurs in the above process.

Input data formats

Hi-C data

Format of Hi-C data is fastq.gz, But canonical name will be nice. for example::

```
$ ls -lh

-rw-r--r-- 1 hluo CPeng 42G Sep 14 00:30 GM12878_R1_1.fastq.gz
-rw-r--r-- 1 hluo CPeng 42G Sep 14 00:30 GM12878_R1_2.fastq.gz
```

Both compresssed and uncompressed fastq data are available for Hi-C data

genome data

Format of genome data is .fa, make sure that all the versions of genome that contribute to other data (such as SNPs, mapping genome) are matched.

```
$ ls -lh
```

```
-rwxr-xr-- 1 hluo CPeng 2780868912 Sep  4  2017 mm10.fa
-rwxr-xr-- 1 hluo CPeng 3187102198 Sep  4  2017 hg19.fa
```

SNPs data

Format of SNPs data is self-defined and essential for haplotype Hi-C pipeline. We need a TXT-like type file as input.(.txt) The file has five columns :

1. chromosome ID
2. genome position
3. SNP base
4. Maternal base
5. Paternal base

For example::

```
$ ls -lh

-rw-r--r-- 1 hluo CPeng 56524229 Mar  9 11:07 GM12878_F1_maternal_paternal_SNP.txt

$ head -5 GM12878_F1_maternal_paternal_SNP.txt

1   10492   C   T   C
1   10927   A   A   G
1   10938   G   G   A
1   13302   C   C   T
1   13813   T   G   T
```

How to use HiCHap ?

First have a look at the help message! The executable code in the command line of HiCHap is **hichap**.

HiCHap has the general parameters : -w (--workspace), -log (--logfile) and -N (--NonAllelic).

There is a closer logical connection between hichap modules. The input of next sub-command may be the output of the previous module. For the simplicity, **hichap** will output the results to workspace by default and search the input in the workspace.For the oprability, users also can set the for input and output path by themselves.

If -N (--NonAllelic) is set on the sub-command. the sub-module will run on non-haplotype Hi-C pipeline.

Try to get help:

```
$ hichap --help

usage: hichap [-h] [-v] {rebuildG,rebuildF,GlobalMapping,Rescue,ReMapping,bamProcess,filtering,binning,correcting}
...
...
```

There are several major functions available in HiCHap serving as sub-commands.

Subcommand	Description
rebuildG	Build Genome index, genome size and enzyme fragment locations
rebuildF	Chunk fastq files.
GlobalMapping	Mapping the raw chunked reads to genome
Rescue	Rescue the unmapped reads by cutting the Ligation site.
ReMapping	Re-mapping the resuced reads to genome
bamProcess	Integrate all the mapping informations
filtering	Hi-C filtering and Allelic assignment (if neccessary).

binning	Building the Interaction Matrix.
correcting	Correcting the Interaction Matrix

Please use 'hichap SUB-COMMAND -h' to see the detail description for each option of each module.

rebuildG

For non-haplotype Hi-C, build the bowtie2 index and enzyme fragments location for genome.

The command line eg ::

```
$ hichap rebuildG -w ./GM12878 -log ./GM12878.log -N -g ./hg19.fa -e MboI -t 4
```

For haplotype Hi-C, build the parent genome and their index, enzyme fragments. The command line eg ::

```
$ hichap rebuildG -w ./GM12878 -log ./GM12878.log -g ./hg19.fa -S ./GM12878_F1_maternal_paternal_SNP.txt -e MboI -t 4
```

After rebuildG, a genome Folder (./GM12878/genome) contains the genome Index and fragments location txt will be created under the workspace(./GM12878)

rebuildF

Chunking the fastq by a given step. The command line eg ::

```
$ hichap rebuildF -w ./GM12878 -log ./GM12878.log -1 GM12878_R1_1.fastq.gz -2 GM12878_R1_2.fastq.gz -c 4000000 -t 2
```

After rebuildF, a folder(./Genome/fastqchunks) contains the chunked files will be created under the workspace(./Genome) eg:

```
$ ls -lh ./Genome/fastqchunks

-rw-r--r-- 1 hluo CPeng 410M Nov 13 10:44 GM12878_R1_chunk0_1.fastq.gz
-rw-r--r-- 1 hluo CPeng 410M Nov 13 10:43 GM12878_R1_chunk0_2.fastq.gz
...
...
-rw-r--r-- 1 hluo CPeng 407M Nov 13 10:49 GM12878_R1_chunk9_1.fastq.gz
-rw-r--r-- 1 hluo CPeng 406M Nov 13 10:48 GM12878_R22_chunk9_2.fastq.gz
```

GlobalMapping

After genome rebuilding and fastq chunking, You need to start mapping tasks. Each chunk means a single mapping task. We will try to reduce the cost of time by parallel mode. And for the different operating environment, two sets of Mapping API is designed for users.

1. PBS-Mode

If you use the clusters (based on PBS for job management), you can choose the PBS API for less time spending. You can submit N tasks to the computation nodes and M threads used for each task.

If non-haplotype Hi-C pipeline, the index parameter only have one chooice. For example:

```
$ nohup hichap GlobalMapping -w ./GM12878 -log GM12878.log -b ~/tools/bowtie2/bowtie2 -i ./GM12878/genome/hg19/hg19 -m PBS -pt 10 
```

if Haplotype Hi-C pipeline, the index parameter have two index. Maternal first, Paternal follows. For example:

```
$ nohup hichap GlobalMapping -w ./GM12878 -log GM12878.log -b ~/tools/bowtie2/bowtie2 -i ./GM12878/genome/Maternal/Maternal ./GM12
```

Make sure that this command-line is running on the login node or the node where jobs are submitted.

The key parameter of this command -m(--mode) must be PBS. -pt (--PBSthreads) 10 4 means running 10 chunks mapping tasks parallely, and each task use 4 threads. That is, 40 cores of clusters will be occupied. Using the **qstat** to check the tasks.

```
$ qstat

Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
2266086.admin              GM12878_R1      hluo                   0 R batch
2266087.admin              GM12878_R1      hluo                   0 R batch
2266088.admin              GM12878_R1      hluo                   0 R batch
2266089.admin              GM12878_R1      hluo                   0 R batch
2266090.admin              GM12878_R1      hluo                   0 R batch
2266091.admin              GM12878_R1      hluo                   0 R batch
2266092.admin              GM12878_R1      hluo                   0 R batch
2266093.admin              GM12878_R1      hluo                   0 R batch
2266094.admin              GM12878_R1      hluo                   0 R batch
2266095.admin              GM12878_R1      hluo                   0 R batch
```

2. WS-Mode

If you are not using the cluster system.Us WS API to start mapping tasks.

If non-haplotype Hi-C pipeline, the index parameter only have one index. For example:

```
$ hichap GlobalMapping -w ./GM12878 -log GM12878.log -b ~/tools/bowtie2/bowtie2 -i ./GM12878/genome/hg19/hg19 -m WS -wt 16
```

If haplotype Hi-C pipeline, the index parameter have two index. Maternal first, Paternal followed. For example:

```
$ hichap GlobalMapping -w ./GM12878 -log GM12878.log -b ~/tools/bowtie2/bowtie2 -i ./GM12878/genome/Maternal/Maternal ./GM12878/ge
```

The key parameter of this command -m(--mode) must be WS. -wt (--PBSthreads) 16 means that the total threads we will share for 4 mapping tasks. That is, 4 chunk mapping tasks are running paralelly and each task occupies 4 threads.

Rescue

Reads rescue. For unmapped reads in GlobalMapping, hichap will search the ligation-site and using the resuce mode to make full use of sequence information on reads.

if non-haplotype Hi-C pipeline:

```
$ hichap Rescue -w ./GM12878 -log GM12878.log -e MboI -t 8 -N
```

Haplotype Hi-C pipeline:

```
$ hichap Rescue -w ./GM12878 -log GM12878.log -e MboI -t 8
```

ReMapping

Except for the inputs, the rest is same as **GlobalMapping** . take a look at **GlobalMapping** for more informations

bamProcess

Integrate all the mapping informations.

If non-haplotype Hi-C pipeline, the fragment parameter only have one fragment location file and the SNP parameter should be defualt (None). For example:

```
$ hichap bamProcess -w ./GM12878 -log ./GM12878.log -N -f ./GM12878/genome/GATC_hg19_fragments.txt -t 16 --rfo
```

if haplotype Hi-C pipeline, the fragment parameter have two fragment location files.Maternal first and Paternal followed. The SNP parameter should be set. For example:

```
$ hichap bamProcess -w ./GM12878 -log ./GM12878.log -f ./GM12878/genome/GATC_Maternal_fragments.txt ./GM12878/genome/GATC_Paternal
```

The parameter --rfo means unique filtering softly. If your sequence data has a high sequencing depth, you can remove this parameter but indeed hurt the data utilization.

filtering

The **filtering** sub-command of **hichap** is designed to preform some basic filtering on the aligned Hi-C read pairs:

Hi-C filtering

1. Remove redundant PCR duplicates
2. Remove the read pair that maps to the same restriction fragment

Allelic assignment

1. assignment maternal interaction pairs
2. assignment paternal interaction pairs
3. assignment regroup interaction pairs

Here's the command you should type in the terminal:

IF non-haplotype Hi-C pipeline

```
$ hichap filtering -w ./GM12878 -log ./GM12878.log  -N -t 16
```

If haplotype Hi-C pipeline

```
$ hichap filtering -w ./GM12878 -log ./GM12878.log -t 16
```

After this sub-command, some bed files will created under the workspace. "Filtered_Bed" Folder for non-haplotype Hi-C pipeline, "Allelic_Bed" Folder for haplotype Hi-C pipeline.The main file is "**_Valid_sorted.bed". It has 23 columns. That is the Hi-C valid interaction pairs. you can do some custom processing with this file.The description of each column is :

-------	**Hi-C interaction pairs**
column	description
1	Pair Name
2	R1 mate Reference
3	R1 mate Strand
4	R1 mate Position
5	R1 mate Length
6	R1 mate AS score
7	R1 mate Fragment Middle point
8	R1 mate SNP Matching num (Non-haplotype results in 0)
9	R1 mate Reference
10	R2 mate Strand
11	R2 mate Position
12	R2 mate Length
13	R2 mate AS score
14	R2 mate Fragment Middle point

15	R2 mate SNP Matching num (Non-haplotype results in 0)
--------	candidate mate if it is possible
16	Candidate mate Reference
17	Candidate mate Strand
18	Candidate mate Position
19	Candidate mate Length
20	Candidate mate AS score
21	Candidate mate Fragment Middle point
22	Candidate mate SNP Matching num (Non-haplotype results in 0)
23	Candidate Index for which mate.

If haplotype Hi-C pipeline.the results files of haplotype interactions have the target like "M_M", "P_P", "M_P", "P_M", "Bi_Allelic".The "M_M" represent the maternal-maternal interactions. The "M_P" represent the maternal-paternal interactions. "Bi_Allelic" represent can't assign to parent. The files have 5 columns. The decription of each column is :

-----	**Haplotype Hi-C interactions**
column	description
1	chromosome ID for interaction loci 1
2	fragment ID for interaction loci 1
3	chromosome ID for interaction loci 2
4	fragment ID for interaction loci 2
5	assignment target (R1 means assigned by R1, R2 means assigned by R2, Both means both mate can be assigned)

binning

Building interaction matrix with the bed files created by **filtering**. The matrix will be save as NPZ format (numpy.savez) The single-line command:

If non-haplotype Hi-C pipeline

```
$ hichap binning -w ./GM12878 -log GM12878.log -b ./GM12878_R1_workspace/Allelic_Bed ./GM12878_R2_workspace/Allelic_Bed -gs ./GM12
```

If haplotype Hi-C pipeline

```
$ hichap binning -w ./Gm12878 -log ./GM12878.log -b ./GM12878_R1_workspace/Allelic_Bed ./GM12878_R2_workspace/Allelic_Bed -gs ./GM
```

-b (--bedPath) filtering output bed path (Allelic_Bed or Filtered_Bed) of all the replicates. The path should follow one by one.

-wR (--wholeRes) means the resolution of whole genome interaction matrix

-lR (--localRes) means the resolution of intra-chromosome interaction matrix

correcting

Correcting the interaction matrix. we will perform ICE correction for non-haplotype interaction matrix and HiCHap correction for haplotype interaction matrix.

For non-haplotype Hi-C interaction matrix.

Intra-chomosome interaction matrix:

```
$ hichap correcting -p ./GM12878/Merged_Reps_Local_Chromosome_Matrix.npz -M Single -SM N -c local -o ./CorMatrix
```

Whole-chromosme interaction matrix:

```
$ hichap correcting -p ./GM12878/Merged_Reps_Whole_Genome_Matrix.npz -M Single -SM N -c whole -o ./CorMatrix
```

For haplotype Hi-C interaction matrix:

correcting all the interaction matrix:

```
$ hichap correcting -p ./GM12878/RawMatrix -M Allelic -o ./CorMatrix
```

correcting one of the interaction matrix:

One Mate npz matrix correcting:

```
$ hichap correcting -p ./GM12878/RawMatrix/Merged_Reps_Local_Chromosome_Matrix.npz ./GM12878/RawMatrix/Merged_Reps_One_Mate_Matern
```

Both Mate npz matrix correcting:

```
$ hichap correcting -p ./GM12878/RawMatrix/Merged_Reps_Both_Mate_Maternal.npz -M Single -SM A -o ./CorMatrix
```

How to get matrix with python code.

Loading the Matrix Data. Open a python interpreter and follow the code below:

```
>>> import numpy as np
>>> Raw_Lib = np.load('Merged_Reps_One_Mate_Paternal.npz')
>>> Raw_Lib.keys() # chromosomes
    ['20','21','22','1','3','2','5','4','7','6','9','8','X','11','10','13','12','15','14','17','16','19','18']
>>> Raw_Lib['1'] #Raw maternal matrix for chromosome 1
    array([[  3,   1,   1, ...,   0,   0,   0],
           [  1,   6,   8, ...,   0,   2,   0],
           [  3,   3,   2, ...,   0,   0,   0],
           ...,
           [  0,   0,   0, ..., 173,  91,   6],
           [  0,   0,   0, ...,  22, 242,  13],
           [  0,   0,   0, ...,   0,  18,   6]])

>>> Cor_Lib = np.load('')
>>> Cor_Lib.keys()
    ['Matrix', 'Gap']
>>> Cor_Matrix = Cor_Lib['Matrix'][()]
>>> Cor_Matrix.keys()
    ['20','21','22','1','3','2','5','4','7','6','9','8','X','11','10','13','12','15','14','17','16','19','18']
>>> Cor_Matrix['1'] # Correct maternal matrix for chromosome 1
    array([[2.31312501e+04, 1.52458765e+02, 7.43905676e+01, ...,
            0.00000000e+00, 0.00000000e+00, 4.21820195e+01],
           [1.52458765e+02, 9.11929045e+03, 3.59742291e+03, ...,
            0.00000000e+00, 1.88965518e+01, 1.45028226e+01],
           [7.43905676e+01, 3.59742291e+03, 7.19421465e+03, ...,
            2.67476970e+00, 7.61968813e+00, 1.30231897e+01],
           ...,
           [0.00000000e+00, 0.00000000e+00, 2.67476970e+00, ...,
            6.11625344e+03, 1.62810585e+03, 1.28907394e+03],
           [0.00000000e+00, 1.88965518e+01, 7.61968813e+00, ...,
            1.62810585e+03, 7.14459658e+03, 3.18665620e+03],
           [4.21820195e+01, 1.45028226e+01, 1.30231897e+01, ...,
            1.28907394e+03, 3.18665620e+03, 6.86309532e+03]])
```

Chromosome Structure Analysis

Chomosome structure analysis is integrated in the module **StructureFind**. The source code can be found in the lib/StructureFind.py.

Use the API like :

```
>>> from HiCHap.StructureFind import StructureFind
>>> #============= Compartment==============
>>> PC_paternal = StructureFind(RawNPZ = 'Merged_Reps_One_Mate_Paternal.npz',
                                CorNPZ = 'Correct_Merged_Reps_One_Mate_Paternal.npz',
                                Res = 200000)
>>> PC_paternal.run_Compartment(OutPath = './Paternal_Compartment')

>>> #============= TADs calling=============
>>> TADs_paternal = StructureFind(RawNPZ = 'Merged_Reps_One_Mate_Paternal.npz',
                                  CorNPZ = 'Correct_Merged_Reps_One_Mate_Paternal.npz',
                                  Res = 40000)
>>> TADs_paternal.run_TADs(OutPath = './Paternal_TADs',
                           state_num = 3,
                           window = 600000)

>>> #============= Loops calling=============
>>> #Non-haplotype
>>> Test_Raw_loops = StructureFind(RawNPZ = 'Merged_Reps_Local_Chromosome_Matrix.npz',
                                   CorNPZ = 'Correct_Merged_Reps_Local_Chromosome_Matrix.npz',
                                   Res = 10000)
>>> Test_Raw_loops.run_Loops(OutPath = 'Raw', Allelic = False)

>>># Haplotype
>>> Loops_paternal = StructureFind(RawNPZ = 'Merged_Reps_One_Mate_Paternal.npz',
                                   CorNPZ = 'Correct_Merged_Reps_One_Mate_Paternal.npz',
                                   Res = 40000)

>>> Loops_paternal.run_Loops(OutPath = './Paternal_Loops', Allelic = True)
```

That is ! Notice that the API **run_Compartment** and **run_TADs** are both available for non-haplotype and haplotype. **run_Loops** need the parameter Allelic (bool type) for non-haplotype (False) and haplotype (True)

Allelic Specificity of loops

The calculation method of Allelic Speicficity is integrated in the **StructureFind**. It need a candidate loop file as input. The format of it is 3 columns and n rows text file.The columns represent the loop location on the genome:

1) chromomsome ID. 2) start. 3) end.

Use the API like:

```
>>> from HiCHap.StructureFind import AllelicSpecificty
>>> AS_test = AllelicSpecificty(Maternal_NPZ = './Correct_Merged_Reps_One_Mate_Maternal.npz',
                                Paternal_NPZ = './Correct_Merged_Reps_One_Mate_Paternal.npz',
                                Loop_file = './Candidate_Loops.txt',
                                Res = 40000)
>>> AS_test.AllelicSpecificityCalling()
```