Manual of the HiCorrector 1.2

A fast, scalable and memory-efficient package for normalizing large-scale Hi-C data

Wenyuan Li, Ke Gong, Qingjiao Li, Frank Alber and Xianghong Jasmine Zhou

Molecular and Computational Biology, University of Southern California, Los Angeles, CA 90089, USA.

Contact: alber@usc.edu or xjzhou@usc.edu

June 2, 2015

Table of Contents

Introduction

Genome-wide proximity ligation assays, e.g. Hi-C and its variant TCC, have recently become important tools to study spatial genome organization. Removing biases from chromatin contact matrices generated by such techniques is a critical preprocessing step of subsequent analyses. The continuing decline of sequencing costs has led to an ever-improving resolution of the Hi-C data, resulting in very large matrices of chromatin contacts. Such large-size matrices, however, pose a great challenge on the memory usage and speed of its normalization. Therefore, there is an urgent need for fast and memory-efficient methods for the normalization of Hi-C data.

HiCorrector is an easy-to-use, open source package of several Hi-C data normalization algorithms. Its salient features are

- **Scalability** – the software is capable of normalizing Hi-C data of any reasonable times;
- **Memory efficiency** – the sequential version can run on any single computer with very limited memory, no matter how little;
- **Fast speed** – the parallel version can run very fast on multiple computing nodes with limited local memory.

The sequential version is implemented in ANSI C and can be easily compiled on any system; the parallel version is implemented in ANSI C with the MPI library (a standardized and portable parallel environment designed for solving large-scale scientific problems). The package is freely available at http://zhoulab.usc.edu/HiCorrector/.

Getting Started

Compilation and Installation

The basic C compiler (like "cc") can be used with the library option ("-lmpi" in unix) to compile the MPI C code. However, the most generic way to compile MPI/C program is using "mpicc" script provided by many MPI implementations, such as OPENMPI. These commands appear similarly to the basic "cc" command, however they transparently set the include paths and link to the appropriate libraries.

Unzip the package file (HiCorrector.tar.gz), then go to the directory "**`src/`**". In the unix environment, please open the file "**`src/Makefile`**" to check the MPI library directory which is assigned to the variable "LIBSMPI" and the MPI compiler's location and name which is assigned to the variable "CCMPI". Then you may run the command ">> make" to compile the source codes. Below is the part of an example Makefile:

```
# compiler and options for ANSI C
CC=cc
CFLAGS= -Wall -O3
LIBS= -lm

# mpi compiler and options for ANSI C and mpi library
CCMPI=/usr/usc/openmpi/default/bin/mpicc
CFLAGSMPI=
LIBSMPI= -L/usr/usc/openmpi/1.6.4/lib
INCLUDE=
```

In the USC HPCC system, I used the commonly used OPENMPI library (http://www.open-mpi.org/). See more details in http://hpcc.usc.edu/support/documentation/examples-of-mpi-programs/ .

After building the source codes, you will see five executable command files in the directory "**bin/**" and "**src/**":

- "**ic**": the command for the original iterative correction algorithm running in a single computer.
- "**ic_mes**": the command for the memory-efficient sequential iterative correction algorithm running in a single computer.
- "**ic_mep**": the command for the memory-efficient parallel iterative correction algorithm. It runs in the MPI environment.
- "**split_data**": the command for splitting the file of the whole data to multiple data-block files. It's a sequential program working on the single computer.
- "**split_data_parallel**": the command for splitting the file of the whole data to multiple data-block files. It's a parallel program working on the MPI environment.
- "**export_norm_data**": the command for generating the normalized matrix by using the original data and the bias factors obtained from iterative correction algorithm. It's a sequential program working on the single computer.

A Quick Example

After unzipping the package file, there is a directory "**example/**". In this directory, you may find an example contact matrix file, "**contact.matrix**", which is a simulated data of size 1000 X 1000.

Three bash scripts files are in this directory for guiding you to run the programs:

"**run_example_ic.sh**": this script will create an output directory to save the output results of running the command "**ic**" on the example matrix file.

"**run_example_ic_mes.sh**": this script will create an output directory to save the output results of running the command "**ic_mes**" on the example matrix file.

"**run_example_ic_mep.sh**": this script will create an output directory to save the output results of running the command "**ic_mep**" on the example matrix file. Before running this script, please make sure you installed the MPI library and system on your computer cluster system.

"**run_export_norm_data.sh**": this script will create an output directory to save the output results of running the command "**export_norm_data**" on the example matrix file and bias vector file.

Overview of the Package

1. Prepare input contact matrix file

Given a contact matrix with *N* rows and *N* columns, the input contact matrix file format is a tab-delimited text file: each line is a row of the matrix, *N* numbers in each line are separated by a delimiter "TAB". So there are *N* lines in this file.
In the future updates, we will consider to use the binary file format, which is faster than text file to load into the memory. But the text file is easier for human recognition.

Optional step: split the massive input matrix file into multiple small files

For very large matrix file, we could split a single large file to multiple small files to alleviate the system's I/O cache pressure. So it is recommended to perform the split file step if the file I/O performance is not well.

2. Run one of three IC algorithms

After preparing the input contact matrix file, you may choose one of three algorithms "**`ic`**", "**`ic_mes`**" and "**`ic_mep`**" to run the program.

See our example bash scripts files for how to run these algorithms.

3. Get the output bias vector file

After running the algorithm code is done, you will get the output file which stores a bias vector of size *N* X 1, which is small and easy to save. Having this bias vector, it is simple to generate the normalized matrix, at any time if needed. In this file, each line contains an element of this vector.

Optional step: generate the normalized matrix file by using the bias vector file

If you want to have the normalized matrix, you may run the command "**`export_norm_data`**".

Usage of Commands

ic – the command of original iterative corrective algorithm

```
Usage: ic <input matrix file> <#rows/columns> <max iterations> <has header line in
input file?> <has header column input file?> <output file>
```

`input matrix file`:	The file storing the input chromatin contact frequency matrix to be normalized. This is a tab-delimited file.
`#rows/columns`:	The number of rows or columns of the input chromatin contact frequency matrix to be normalized.
`max iterations`:	Maximum number of iterations performed in the algorithm.
`has header line in input file?`	'1' indicates input matrix file has a header line; otherwise '0'.
`has header column input file?`	'1' indicates input matrix file has a header column; otherwise '0'.
`output file`:	The file storing the bias vector output of the algorithm. Each line contains an element of this vector.

It is recommended to include the directory in the filenames.

ic_mes – the command of the memory-efficient sequential version of IC algorithm

```
Usage: ic_mes <input matrix file> <memory size (MB)> <#rows/columns> <max iterations>
<has header line in input file?> <has header column input file?> <output file>
```

input matrix file:	The file storing the input chromatin contact frequency matrix to be normalized. This is a tab-delimited file.
memory size (MB):	The memory size. Its unit is Megabytes (MB).
#rows/columns:	The number of rows or columns of the input chromatin contact frequency matrix to be normalized.
max iterations:	Maximum number of iterations performed in the algorithm.
has header line in input file?	1 indicates input matrix file has a header line; otherwise 0.
has header column input file?	1 indicates input matrix file has a header column; otherwise 0.
output file:	The file storing the bias vector output of the algorithm. Each line contains an element of this vector.

It is recommended to include the directory in the filenames.

ic_mep – the command of the memory-efficient parallel version of IC algorithm

```
Usage: ic_mep [options] ...
```

-h or --help	Optional	Show help message.
--inputFile=FILE	Required	The file storing the input chromatin contact frequency matrix to be normalized. This is a tab-delimited file.
--hasHeaderRow=NUM	Optional	The input matrix file has header line or not. 1 means 'has'; 0 otherwise. default (0).
--hasHeaderColumn=NUM	Optional	The input matrix file has header column or not. 1 means 'has'; 0 otherwise. default (0).
--useSplitInputFiles	Optional	Use split contact matrix files.
--numRows=NUM	Required	The number of rows or columns of the input chromatin contact frequency matrix to be normalized.
--numTask=NUM	Required	Number of processors to be used. It includes both manager and worker tasks.
--memSizePerTask=NUM	Required	Size of memory used for each task. Its unit is Megabytes (MB).
--maxIteration=NUM	Required	Maximum number of iterations performed in the algorithm.
output file:	Required	The file storing the bias vector output of the algorithm. Each line contains an element of this vector.

It is recommended to include the directory in the filenames.

export_norm_data – the command of the sequential version of using bias vector to normalize data

```
Usage:
        export_norm_data <input raw matrix file> <#rows/columns> <has header line in
input file?> <has header column input file?> <memory size (MB)> <input bias vector file>
<fixed row sum after normalization> <output normalized matrix file>
```

input raw matrix file:	The file storing the input chromatin contact frequency matrix to be normalized. This is a tab-delimited file.
#rows/columns:	The number of rows or columns of the input chromatin contact frequency matrix to be normalized.
has header line in input file?	1 indicates input matrix file has a header line; otherwise 0
has header column input file?	1 indicates input matrix file has a header column; otherwise 0.
memory size (MB):	The memory size. Its unit is Megabytes (MB).
input bias vector file:	The file storing the bias vector.
fixed row sum after normalization:	The iterative correction algorithm can allow users to specify the row sum after the normalization, because this method is actually a matrix scaling approach that normalizes the matrix to be a doubly stochastic matrix (rows/columns sums equal to 1). Then we can multiple each element of this normalized matrix by the given value of this parameter, say 10.0 or 100.0. In such a way, the rows sums of normalized matrix becomes this number (10.0 or 100.0).
output normalized matrix file:	The file storing the normalized matrix. This is a tab-delimited file.

It is recommended to include the directory in the filenames.

Contacts

Please contact Professors Frank Alber and Xianghong Jasmine Zhou at alber@usc.edu or xjzhou@usc.edu . Visit http://zhoulab.usc.edu/HiCorrector/ for details.

References

Wenyuan Li, Ke Gong, Qingjiao Li, Frank Alber, and Xianghong Jasmine Zhou. (2014) **HiCorrector: A fast, scalable and memory-efficient package for normalizing large-scale Hi-C data**. *Bioinformatics*. 31(6): 960-962