Bioinformatics, 31(6), 2015, 960–962
doi: 10.1093/bioinformatics/btu747
Advance Access Publication Date: 12 November 2014
Applications Note

Genome analysis

Hi-Corrector: 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 Program, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089, USA

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on September 2, 2014; revised on October 17, 2014; accepted on November 5, 2014

Abstract

Summary: 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 normalization of Hi-C data. We developed Hi-Corrector, an easy-to-use, open source implementation of the Hi-C data normalization algorithm. Its salient features are (i) scalability—the software is capable of normalizing Hi-C data of any size in reasonable times; (ii) memory efficiency—the sequential version can run on any single computer with very limited memory, no matter how little; (iii) fast speed—the parallel version can run very fast on multiple computing nodes with limited local memory.

Availability and implementation: 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/Hi-Corrector/.

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

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The recent development of genome-wide proximity ligation assays such as Hi-C (Lieberman-Aiden *et al.*, 2009) and its variant TCC (Kalhor *et al.*, 2012) has significantly facilitated the study of spatial genome organization. The raw chromatin interaction data obtained by Hi-C methods can have both technical and biological biases (Imakaev *et al.*, 2012). Therefore, correcting biases in the Hi-C data is an important preprocessing step. Among several recently developed methods (Hu *et al.*, 2012; Imakaev *et al.*, 2012; Yaffe and Tanay, 2011), the iterative correction (abbreviated as IC) algorithm (Imakaev *et al.*, 2012) has been used most widely by recent studies (Ay *et al.*, 2014; Le *et al.*, 2013; Naumova *et al.*, 2013; Varoquaux *et al.*, 2014) due to its conceptual simplicity, parameter-free algorithm and ability to account for unknown biases, although its assumption of the equal visibility across all loci may require further exploration. Mathematically, the IC algorithm is a matrix scaling or balancing method that transforms a symmetric matrix into one that is doubly stochastic, meaning that the row and column sums of the matrix are equal to 1. However, the Hi-C chromatin interaction matrix is of the massive size O(N^2), where N is the number of genomic regions. Thus, it requires expensive computing resources such as large memory and long computation time. This is especially problematic for high-resolution data at the kilobase level or beyond (Jin *et al.*, 2013; Le *et al.*, 2013). For example, at the resolution of 10K base pairs per region, the human genome has 303 640 regions and the matrix of the Hi-C data occupies about 343 GB of memory, which cannot be loaded into any common desktop computer even in the compressed format. Most scaling algorithms in the matrix computation field (Knight, 2008; Knight and Ruiz, 2012) suffer from this scalability issue, because their main focus is improving the convergence rate and numerical stability. Therefore, there is high demand for a fast and scalable IC algorithm that can work with massive Hi-C data matrices on common computing resources.

Here we propose a set of scalable algorithms (adapted from the original IC algorithm) to meet this need. Both sequential and parallel versions were implemented in the standard and efficient C language, which allows for precise memory control. The sequential implementation is memory efficient and can run on any single computer with limited memory, even for Hi-C datasets of large size. It is designed to overcome the memory limit by loading a portion of data into the memory at each time, so requires some extra time for file reading. The parallel implementation is both memory efficient and fast. It can run on one of the most popular parallel computing resources: a computer cluster (i.e. a distributed-memory computing environment). In this environment, a set of general-purpose processors or computers can be interconnected to share resources, and each computer retains its local and limited memory. The parallel algorithm is designed with very low communication overhead among computing nodes, so that it runs faster on clusters with more computers. Although the Hi-C analysis pipeline, ICE (Imakaev *et al.*, 2012), implements the IC algorithm, it works only on a single computer and cannot utilize as many computing resources as possible to speed up the computation.

Very few parallel matrix scaling or balancing algorithms have been developed prior to this work (Amestoy *et al.*, 2008; Zenios and Iu, 1990). However, none of them are suitable for the bias correction task of Hi-C data. Zenios and Iu (1990) parallelized the matrix balancing algorithm in 1990 for a shared-memory computer, which cannot address the memory shortage problem. Amestoy *et al.* (2008) designed a complicated data distribution strategy based on the partitions of non-zero elements. Their method is not applicable to the raw Hi-C contact map, which contains a high proportion of non-zero elements. We performed experiments on high-performance computing resources and clusters with different numbers of nodes and memory capacities. The results showed that this package could meet the strong demand for normalizing massive Hi-C data given limited computing resources.

2 Algorithms and implementation

Given an observed chromosome contact frequency matrix $\mathbf{O} = (O_{ij})_{N\times N}$ over N genomic regions, the IC method eliminates biases so that all genomic regions have equal visibility (Imakaev *et al.*, 2012). To make this algorithm memory-efficient, we designed a strategy of breaking the matrix $\mathbf{O}$ into K equal partitions of complete rows and loading only one partition into memory at any given time. Therefore, the memory requirement can be very low when K is large. This strategy adapts the IC algorithm by adding two steps: (i) loading the kth matrix partition O^k into memory and (ii) updating this partition with the last updated bias vector $\mathbf{b}$. The new Memory Efficient Sequential algorithm (called IC-MES) works even for the extreme case of $K = N$, where only one row is loaded each time. IC-MES is memory efficient, but it is still a sequential algorithm that runs on a single computing machine. Therefore, it may be too slow when the machine has small memory. To normalize large Hi-C matrices in a short time, we also designed a fast, scalable and Memory-Efficient Parallel algorithm (called IC-MEP) that can maximally exploit the parallelism of the normalization problem and make use of many commonly available computing resources. In essence, the normalization problem is a data divisible task: a series of operations that can independently work on separate partitions of the data. This problem is perfectly suited to the data-parallel model in a distributed-memory computing environment such as a computer cluster, which consists of K independent processors (or nodes) that are loosely or tightly connected in high-speed networks and have limited local memory. We employed the manager–worker parallel programming paradigm. The manager task partitions the data into K blocks, then initiates K worker tasks in different nodes; each worker task processes a single data block. The manager coordinates all workers and synchronizes their calculations with updated bias vectors. The IC-MEP algorithm has very little network messaging overhead, because no communication exists between workers. Therefore, it is computationally efficient. Furthermore, in order for each worker to run its task on limited memory, we also used the memory-saving strategy of the IC-MES algorithm. That is, each worker further partitions its assigned data block into a set of sub-blocks and loads only one sub-block into memory at any given time. Theoretically, the IC-MEP algorithm can work on any number of processors with any local memory capacity. Details of these three algorithms and their flowchart figures are provided in the Supplementary materials. We used ANSI C to implement the two sequential algorithms IC and IC-MES, because of its maximum control and memory efficiency. We implemented the parallel algorithm IC-MEP using the popular message passing interface, which is a highly standardized and portable environment designed for solving large-scale scientific problems on distributed memory systems.

3 Results

We compared three algorithms (IC, IC-MES and IC-MEP) on the TCC/Hi-C data of two human cell types: GM12878 and hESC (Dixon *et al.*, 2012; Kalhor *et al.*, 2012). The whole genome is partitioned into the equal-size regions (or bins); the bin size is the main

Table 1. Running time of three algorithms on 10K and 20K bp resolution Hi-C data

Algorithm	IC	IC-MES	IC-MEP	
20K bp data (151 825 bins)				
#Processor	1	1	16	48
Memory	86 GB	4 GB	1 GB	1 GB
Time (gm12878)	0:36:50	3:58:14	0:19:50	0:6:38
Time (hESC)	0:35:01	3:49:18	0:19:48	0:6:47
10K bp data (303 640 bins)				
#Processor	1	1	16	48
Memory	343 GB	32 GB	2 GB	2 GB
Time (gm12878)	NA	47:27:32	4:50:03	0:26:02
Time (hESC)	NA	37:26:15	4:49:27	0:26:09

All algorithms were terminated after 10 iterations for the purpose of performance comparison, since each iteration has almost the same running time. 'Memory' includes only the memory allocated for computation in each processor, not system overhead. The elapsed time format is 'hours : minutes : seconds'.

indicator of Hi-C data resolution. The results are listed in Table 1. In the experiment with 20K bp resolution data, the basic IC algorithm requires a minimum memory of 86 GB. The algorithm IC-MES can run with just 4 GB memory (a common memory configuration in office computers) and complete the same work in reasonable time (within 4 h). IC-MEP can dramatically speed up the computation using more processors (about 6 min with 48 processors), while using only 1 GB of memory in each processor. For the 10K bp data, none of HPC computer nodes (with 128 GB memory limit) can load the full matrix (about 343 GB) for the basic IC algorithm. But IC-MES and IC-MEP can use 2 GB memory to quickly get the results (even in half hour using 48 processors). Details are provided in the Supplemental materials.

4 Conclusion

With the rapidly increasing resolution of Hi-C datasets, the size of the chromatin contact map will soon exceed the memory capacity of general computers. We developed Hi-Corrector, a scalable and memory-efficient package for bias removal in HiC data. Hi-Corrector can run on any single computer or a computer cluster with limited memory size to complete the task. We performed experiments on high-resolution HiC data from two human cell types to show that the package can process very large data sets in reasonable time using the single processor, and in very short time with multiple processors. The experiments further demonstrate the scalability of our package with the observation shown in Table 1 that the more processors used, the faster it is. Therefore, Hi-Corrector is a timely resource addressing the challenge of normalizing high-resolution Hi-C data.

Funding

National Science Foundation Grant CAREER 1150287 and the Arnold and Mabel Beckman foundation (BYI program) (to F.A.); and National Institutes of Health Grant (NHLBI MAPGEN U01HL108634 to X.J.Z.); and F.A. is a Pew Scholar in Biomedical Sciences, supported by the Pew Charitable Trusts.

Conflict of Interest: none declared.

References

Amestoy,R.P. *et al.* (2008) A parallel matrix scaling algorithm. In: J.M.L.M.,Palma *et al.* (eds.) *High Performance Computing for Computational Science*. Springer, Berlin, pp. 301–313.

Ay,F. *et al.* (2014) Three-dimensional modeling of the *P. falciparum* genome during the erythrocytic cycle reveals a strong connection between genome architecture and gene expression. *Genome Res.*, **24**, 974–988.

Dixon,J.R. *et al.* (2012) Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, **485**, 376–380.

Hu,M. *et al.* (2012) HiCNorm: removing biases in Hi-C data via Poisson regression. *Bioinformatics*, **28**, 3131–3133.

Imakaev,M. *et al.* (2012) Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nat. Methods*, **9**, 999–1003.

Jin,F. *et al.* (2013) A high-resolution map of the three-dimensional chromatin interactome in human cells. *Nature*, **503**, 290–294.

Kalhor,R. *et al.* (2012) Genome architectures revealed by tethered chromosome conformation capture and population-based modeling. *Nat. Biotechnol.*, **30**, 90–98.

Knight,P.A. (2008) The Sinkhorn–Knopp algorithm: convergence and applications. *SIAM J. Matrix Anal. Appl.*, **30**, 261–275.

Knight,P.A. and Ruiz,D. (2012) A fast algorithm for matrix balancing. *IMA J. Numer. Anal.*, **33**, 1029–1047.

Le,T.B.K. *et al.* (2013) High-resolution mapping of the spatial organization of a bacterial chromosome. *Science*, **342**, 731–734.

Lieberman-Aiden,E. *et al.* (2009) Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, **326**, 289–293.

Naumova,N. *et al.* (2013) Organization of the mitotic chromosome. *Science*, **342**, 948–953.

Varoquaux,N. *et al.* (2014) A statistical approach for inferring the 3D structure of the genome. *Bioinformatics*, **30**, i26–i33.

Yaffe,E. and Tanay,A. (2011) Probabilistic modeling of Hi-C contact maps eliminates systematic biases to characterize global chromosomal architecture. *Nat. Genet.*, **43**, 1059–1065.

Zenios,S.A. and Iu,S.-L. (1990) Vector and parallel computing for matrix balancing. *Ann. Oper. Res.*, **22**, 161–180.