HiCRep.py: Fast comparison of Hi-C contact matrices in Python

Dejun Lin∗1, Justin Sanders∗2, and William Stafford Noble†1,3

1Department of Genome Sciences, University of Washington
2Department of Computer Science, Brown University
3Paul G. Allen School of Computer Science and Engineering, University of Washington

Abstract

Hi-C is the most widely used assay for investigating genome-wide 3D organization of chromatin. When working with Hi-C data, it is often useful to calculate the similarity between contact matrices in order to assess experimental reproducibility or to quantify relationships among Hi-C data from related samples. The HiCRep algorithm has been widely adopted for this task, but the existing R implementation suffers from run time limitations on high resolution Hi-C data or on large single-cell Hi-C datasets. We introduce a Python implementation of HiCRep and demonstrate that it is much faster than the existing R implementation. Furthermore, we give examples of HiCRep’s ability to accurately distinguish replicates from non-replicates and to reveal cell type structure among collections of Hi-C data. HiCRep.py and its documentation are available with a GPL license at https://github.com/Noble-Lab/hicrep. The software may be installed automatically using the pip package installer.

1 Introduction

Hi-C is a powerful genomic assay for quantifying chromatin interactions across the whole genome [1]. It has been used extensively to study genome architecture and function in many different species and to understand how genome structure affects genetic diseases. The result of a Hi-C experiment is typically processed into a matrix, whose entries are contact counts between pairs of genomic loci. As Hi-C experiments become more popular, tools that are able to efficiently perform analysis on the resulting contact matrices are in increasing demand [2].

A common task in Hi-C data analysis is measuring the similarity between pairs of data sets. One application of Hi-C similarity is to assess experimental reproducibility. Low reproducibility may indicate low experiment quality or low sequencing depth. Low reproducibility may also warn against merging multiple Hi-C replicates, which is a common practice to boost the signal-to-noise ratio [3]. HiCRep is a tool for quantifying the similarity between pairs of Hi-C contact matrices based on their stratum adjusted correlation coefficients (SCCs) [4]. The SCC is a correlation score ranging from −1 and 1, where a higher score suggests higher similarity between the two input Hi-C matrices. Using high SCC scores as a proxy for high reproducibility, a number of published works have used HiCRep to assess the quality of replicate experiments and to make sure that merging them is sound [5, 6] or to validate that data from a novel assay closely resembles traditional Hi-C data [7, 8]. Beyond comparing replicates, HiCRep has also proved useful as a tool for measuring quantitative differences among samples. For example, Ray et al. used HiCRep to compare Hi-C contact maps of samples before and after undergoing heat shock in order to determine whether the shock had an effect on chromatin structure [9]. HiCRep can also be used to help interpret single-cell Hi-C (scHi-C) data. For example, Liu et al. demonstrated that the SCC values calculated by HiCRep can be used as the basis for a multidimensional scaling (MDS) visualization that accurately captures cell cycle structure in scHi-C data [10].

∗Equal contribution
†Correspondence to william-noble@uw.edu
The original implementation of HiCRep was released as an R package [4]. One of its biggest drawbacks is its inefficiency, mainly because of the dependence on dense contact matrix operations. In a head-to-head comparison among three other tools for measuring reproducibility, HiCRep was found to be the slowest by a significant margin [3]. This means that applying the R implementation to Hi-C data at high resolution or to large scHi-C data sets is prohibitively slow.

Here we present a Python implementation of the HiCRep algorithm that is much faster than its predecessor. Our Python version implements all operations using sparse matrices, which greatly reduce the memory consumption and computation time. Additionally, we have made efforts to make the software more accessible by providing a command line interface as well as a Python application programming interface.

## 2 Implementation

HiCRep takes as input two Hi-C contact matrices in either .cool or .mcool format [11]. First, matrices are normalized by the total contact counts and smoothed with a 2D mean filter of size set by the user. Then, corresponding diagonals of the two contact matrices are compared and used to calculate SCC scores, as described in the original HiCRep paper [4]. The software produces as output a list of SCC scores per chromosome. This output faithfully matches that produced by the existing R implementation (see Supplement for details). We provide thorough unit tests of the implementation covering most of its functionality.

## 3 Results

We used HiCRep to calculate SCC scores between 95 pairs of publicly available Hi-C matrices—19 pairs of biological replicates, 38 pairs of non-replicates of the same cell type, and 38 pairs of non-replicates of different cell types. As shown in Figure 1A, pairs of replicates consistently exhibit very high SCC scores (mean: 0.98, stdv: 0.02), which are markedly higher than the scores of both non-replicates of the same cell type (mean: 0.86, stdv: 0.10) and non-replicates of different cell types (mean: 0.61, stdv: 0.16). These results suggest that HiCRep does a good job of capturing the reproducibility of Hi-C datasets and is able to accurately separate replicates from non-replicates.

We also used HiCRep to evaluate the pairwise SCC scores of 90 Hi-C experiments conducted by the 4D Nucleome Consortium on a number of different cell types (Supplementary Table 1). Using these SCC scores as the distance metric for an MDS model, we show that HiCRep reveals structure among the experiments, with different cell types clustering separately (Figure 1B).

Finally, we compared the run times of our implementation of HiCRep to the R implementation. We selected five pairs of high resolution Hi-C experiments and ran both implementations of HiCRep on each
of them at a number of different resolutions (Figure 1C). Comparing the runtimes, we see that at higher resolutions the Python implementation of HiCRep is more than 20 times faster than the R version. This speed increase allows our version of HiCRep to be practically applied to data with much smaller bin sizes or to larger collections of scHi-C data than was previously possible.

**Funding:** This work was supported by National Institutes of Health award U54 DK107979.

**References**

[1] E. Lieberman-Aiden, N. L. van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo, M. O. Dorschner, R. Sandstrom, B. Bernstein, M. A. Bender, M. Groudine, A. Gnirke, J. Stamatoyannopoulos, L. A. Mirny, E. S. Lander, and J. Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950):289–293, 2009.

[2] F. Ay and W. S. Noble. Analysis methods for studying the 3D architecture of the genome. *Genome Biology*, 16(1):1–15, 2015.

[3] G. G. Yardımcı, H. Ozadam, M. E. G. Sauria, O. Ursu, K. Yan, T. Yang, A. Chakraborty, A. Kaul, B. R. Lajoie, F. Song, Y. Zhang, F. Ay, M. Gerstein, A. Kundaje, Q. Li, J. Taylor, F. Yue, J. Dekker, and W. S. Noble. Measuring the reproducibility and quality of Hi-C data. *Genome Biology*, 20(57), 2019.

[4] T. Yang, F. Zhang, G. G. Yardımcı, F. Song, R. C. Hardison, W. S. Noble, F. Yue, and Q. Li. HiCRep: assessing the reproducibility of Hi-C data using a stratum-adjusted correlation coefficient. *Genome Research*, 27(11):1939–1949, 2017.

[5] Y. Li, J. Haarhuis, Á. Sedeño Cacciatore, R. Oldenkamp, M. S. van Ruiten, L. Willems, H. Teunissen, K. W. Muir, E. de Wit, B. D. Rowland, and D. Panne. The structural basis for cohesin-CTCF anchored loops. *Nature*, 578:472–476, 2020.

[6] K. Pal, M. Forcato, D. Jost, T. Sexton, C. Vaillant, E. Salviato, E. Mazza, E. Lugli, G. Cavalli, and F. Ferrari. Global chromatin conformation differences in the *Drosophila* dosage compensated chromosome X. *Nature Communications*, 10(5355), 2019.

[7] D.-S. Lee, C. Luo, J. Zhou, S. Chandran, A. Rivkin, A. Bartlett, J. R. Nery, C. Fitzpatrick, C. O’Connor, J. R. Dixon, and J. R. Ecker. Simultaneous profiling of 3D genome structure and DNA methylation in single human cells. *Nature Methods*, 16:999–1006, 2019.

[8] G. Li, Y. Liu, Y. Zhang, N. Kubo, M. Yu, R. Fang, M. Kellis, and B. Ren. Joint profiling of dna methylation and chromatin architecture in single cells. *Nature Methods*, 16:991–993, 08 2019.

[9] J. Ray, P. R. Munn, A. Vihervaara, J. J. Lewis, A. Ozer, C. G. Danko, and J. T. Lis. Chromatin conformation remains stable upon extensive transcriptional changes driven by heat shock. *Proceedings of the National Academy of Sciences of the United States of America*, 116:19431–19439, 2019.

[10] J. Liu, D. Lin, G. Yardımcı, and W. S. Noble. Unsupervised embedding of single-cell Hi-C data. *Bioinformatics*, 34:96–104, 2018.

[11] N. Abdennur and L. A. Mirny. Cooler: scalable storage for Hi-C data and other genomically labeled arrays. *Bioinformatics*, 36(1):311–316, 2019.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
Supplement

Verifying the reimplementation  In order to to ensure that our Python implementation reproduces the results of the R version, we ran both versions on five pairs of Hi-C data sets at different resolutions and with different parameters, and verified that the resulting SCC scores matched perfectly. We also include in our implementation a comprehensive set of unit tests that ensure critical functions are working as expected.

Data  All of the Hi-C contact matrices presented here were obtained from the 4DN data portal, where they are available in Cooler format (https://data.4dnucleome.org). A list of accession numbers for the files we used are in Supplementary Table 1. No normalization was performed on any of the contact matrices.

Comparison of replicates  To assess how well HiCRep can separate replicates form non-replicates, we downloaded 19 pairs of replicates from 4DN. To produce Figure 1A, we used HiCRep to compute SCC scores for the 19 replicate pairs as well as 38 randomly selected pairs of data sets from the same “biosource” (i.e., cell type or tissue sample) and 38 randomly selected pairs from different biosources. HiCRep was run with a bin size of 500 kb, a smoothing factor $h = 2$, a maximum genomic distance of 5 Mb, and down-sampling set to false.

Multidimensional scaling  The MDS plot in Figure 1B was produced by using HiCRep to obtain pairwise SCC scores for 90 Hi-C datasets where replicates had already been merged. We then converted these correlation coefficients to Euclidean distance with the equation $d(A, B) = \sqrt{1 - SCC(A, B)}$. Finally, these distances were passed as input to the scikit-learn implementation of MDS [12] called with maximum iterations set to 30,000 and number of initializations set to 10. For this analysis, HiCRep was run with the same settings as listed above.

Timing  For the timing experiment, both versions of HiCRep were run on a computer with an Intel i7 CPU at 1.10 GHz with 16GB of RAM. Our Python implementation of HiCRep was installed from PyPI and run with Python 3.7.6, NumPy 1.18.5, and SciPy 1.5.1, all of which were installed through the conda package manager. The R implementation was installed from conda’s bioconda channel and run with R version 4.0.2, also installed from conda. In order to ensure a fair comparison, we undertook measures to ensure that NumPy was not parallelizing any operations and that R was not forced to use swap space.