iced: fast and memory efficient normalization of contact maps

Nelle Varoquaux[1] **and Nicolas Servant**[3, 4, 5]

1 University of California, Berkeley **3** Institut Curie **4** INSERM U900 **5** Mines ParisTech

DOI: 10.21105/joss.01286

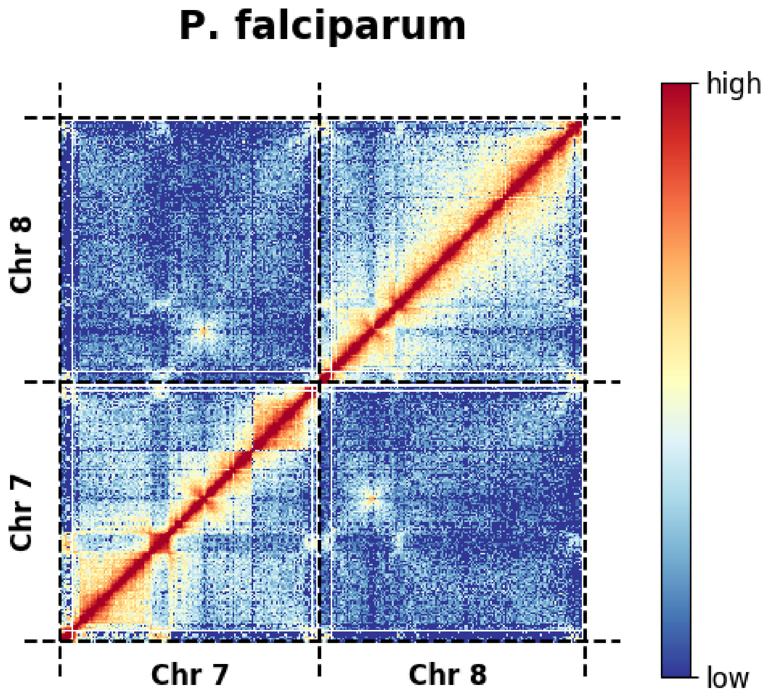
Software

- Review
- Repository
- Archive

Submitted: 11 February 2019
Published: 03 April 2019

Summary

The three-dimensional structure of the genome is thought to play an important role in many biological processes, such as gene regulation and replication. Recent technological advances allow the measurement, in a single experiment, of the frequencies of physical contacts among pairs of genomic loci at a genome-wide scale. The Hi-C protocol results in a noisy and indirect measurement of the 3D structure of the genome, yielding a symmetric matrix where each row and each column correspond to a genomic window, and each entry to the number of times those windows have been seen interacting with one another. As with any genomics experiments, the resulting matrix contains unwanted variations depending on the GC-content, mappability, and the details of the protocol used. Before any downstream analysis, this matrix needs to be appropriately normalized.



Iced implements fast and memory efficient normalization methods, such the ICE normalization strategy or the SCN algorithm. It is included in the HiC-pro pipeline, that processes data from raw fastq files to normalized contact maps (Servant et al., 2015). iced eventually grew bigger than just being a normalization packages, and contains a number of utilities functions that may be useful if you are analyzing and processing Hi-C data.

Moving from sequencing reads to a normalized contact map is a challenging task. Hi-C usually requires several millions to billions of paired-end sequencing reads, depending on genome size and on the desired resolution. Managing these data thus requires optimized bioinformatic workflows able to extract the contact frequencies in reasonable computational time and with reasonable resource and storage requirements. The final step of such pipeline is typically a normalization step, essential to ensure accurate analysis and proper interpretation of the results.

We propose here fast implementations of the iterative correction method (Imakaev et al., 2012) and SCN (Cournac, Marie-Nelly, Marbouty, Koszul, & Mozziconacci, 2012) in Python. iced emphasizes ease-of-use, performance, maintainability, and memory-efficiency. This implementation leverages a memory-efficient data format of Hi-C maps, and outperforms both in speed and memory usage HiCorrector (W. Li, Gong, Li, Alber, & Zhou, 2015), a parallelized C++ implementation of the same algorithm.

References

Cournac, A., Marie-Nelly, H., Marbouty, M., Koszul, R., & Mozziconacci, J. (2012). Normalization of a chromosomal contact map. *BMC Genomics*, *13*, 436. doi:10.1186/1471-2164-13-436

Imakaev, M., Fudenberg, G., McCord, R. P., Naumova, N., Goloborodko, A., Lajoie, B. R., Dekker, J., et al. (2012). Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nature Methods*, *9*, 999–1003. doi:10.1038/nmeth.2148

Li, W., Gong, K., Li, Q., Alber, F., & Zhou, X. J. (2015). Hi-Corrector: a fast, scalable and memory-efficient package for normalizing large-scale Hi-C data. *Bioinformatics*, *31*(6), 960–962. doi:10.1093/bioinformatics/btu747

Servant, N., Varoquaux, N., Lajoie, B. R., Viara, E., Chen, C. J., Vert, J. P., Heard, E., et al. (2015). HiC-Pro: an optimized and flexible pipeline for Hi-C data processing. *Genome Biol.*, *16*. doi:10.1186/s13059-015-0831-x