Coolpup.py – a versatile tool to perform pile-up analysis of Hi-C data

Ilya M. Flyamer*, Robert S. Illingworth and Wendy A. Bickmore

MRC Human Genetics Unit, Institute of Genetics and Molecular Medicine, University of Edinburgh, Crewe Road South, Edinburgh EH4 2XU, UK

* To whom correspondence should be addressed.

Email: Ilya.Flyamer@igmm.ed.ac.uk

Present address: RSI, MRC Centre for Regenerative Medicine, Institute for Regeneration and Repair, The University of Edinburgh, Edinburgh BioQuarter, 5 Little France Drive, Edinburgh, EH16 4UU, UK

ABSTRACT

Hi-C is currently the method of choice to investigate the global 3D organisation of the genome. Of particular interest are looping interactions and topologically-associating domains (TADs), which have been implicated in genome functions, including both gene regulation and genome stability. However, a major limitation of Hi-C is the great sequencing depth required to robustly detect loops in the data. A popular approach used to mitigate this issue, even in extremely sparse single-cell Hi-C data, is genome-wide averaging (piling-up) of loops, or other features, annotated in high-resolution datasets, to measure their prominence in less deeply sequenced data. By carefully choosing the regions used for averaging, hypotheses about 3D genome organization can be tested genome-wide. Here we describe *coolpup.py* – a versatile tool to perform pile-up analysis on Hi-C data. We We demonstrate its utility by replicating previously published findings regarding the role of cohesin and CTCF in 3D genome organization, and to investigate the effect of different data normalization strategies. We also present a novel variation of the pile-up approach that can aid the in statistical analysis of looping interactions. We anticipate that *coolpup.py* will aid in Hi-C data analysis by allowing easy to use, versatile and efficient generation of pileups.

INTRODUCTION

One of the biggest advances in the study of 3D genome organization has come from the development of a family of Chromosome Conformation Capture (3C) methods (1). While they all rely on the same principle of *in situ* proximity ligation of crosslinked and digested chromatin, the scope of each method varies depending on experimental processing and method of quantification of the 3C library (2). Hi-C, a genome-wide 3C-derivative, is the method of choice to investigate the organization of the whole genome (3, 4). Despite the rapid development of ever cheaper next generation sequencing, one of the main challenges in Hi-C remains the required sequencing depth due to the extreme complexity of good quality Hi-C libraries. The output of Hi-C is a square matrix of interactions and therefore requires a vastly greater sequencing depth than most sequencing-based approaches that simply look for enrichment of reads linearly along the genome (5). This limits the resolution at which genomes can be analysed in 3D, since going beyond ~5 kbp resolution requires billions of read pairs for a mammalian genome.

Looping interactions are among the most interesting features that can be studied using C-methods, including Hi-C. Chromatin loops bring distal regions in the genome into close proximity and manifest themselves in Hi-C data as foci of increased interaction frequency (4). The majority of loops identified in Hi-C data from mammalian cells correspond to CTCF/cohesin associated interactions, created by loop

extrusion (6–8). CTCF/cohesin associated loops are closely related to topologically-associating domains (TADs), which in most cases are encompassed in a loop, and which can in turn contain loops. TADs have been reported to constrain enhancer-promoter communication (9, 10) and might be related to genome stability (11), while some loops have been suggested to correspond to enhancer-promoter contacts (4). In addition, distal polycomb sites can be brought together in 'loops' (12, 13).

To our knowledge, currently the only robust method to identify loops *de novo* requires very deep Hi-C libraries, on the order of over a billion Hi-C contacts (4). This means that the vast majority of Hi-C datasets can not be used to identify loops. However, they can be used to quantify the average loop strength (i.e. enrichment of contacts in those loops relative to their local background). To do this one can average (or "pile up") all areas of the Hi-C maps containing loops, annotated in a high-depth dataset (4). This idea is very similar to "metaplots" used, for example, in chromatin immunoprecipitation and sequencing (ChIP-seq) analysis to quantify signal in a subset of regions, except in Hi-C this is for a 2D matrix instead of a linear track. The same approach can of course be applied directly to the data where the loops were annotated. Apart from quantifying the strength of known features, the same analysis can be used to investigate whether certain regions, defined based on for example ChIP-seq peaks, tend to interact with each other on average. For example, to our knowledge the first ever application of pile-up-like analysis was used to investigate clustering of pluripotency factor binding sites in mouse ES cells (14). This can aid in discovery of novel drivers of interactions, although cautious interpretation is required due to the correlative nature of this analysis and additional functional validation is needed.

Another challenge is that Hi-C is a population-based method, and only provides population average measurements. Several single-cell Hi-C approaches have been published (15–19), (reviewed in 20), however none of these provides data depth or resolution comparable to that which can be obtained from a population of thousands of cells (21): the resulting matrices are too sparse to analyse individual regions, and only aggregate genome-wide metrics can be efficiently employed. Approaches to analyse strength of loops, TADs and genome compartmentalization from such data genome-wide have been developed (18). They are all based on the "pile-up" approach described above using data from single cells for regions corresponding to features identified in population Hi-C, to boost the amount of reads used in the analysis.

Since its inception in the current form (4), originally termed APA ("Aggregate Peak Analysis"), pile-up analysis has been used in numerous publications, both to analyse single-cell Hi-C data (8, 15, 18) and as a general way of quantifying feature strength (6, 13, 21–28). No single computational tool can perform all the various kinds of pile-up analyses that have been used in the literature, including local and rescaled (features of different size or shape are averaged, e.g. average TADs) and off-diagonal (e.g. average loops) pile-ups with different normalization strategies. At the same time, performing detailed analysis of Hi-C data remains difficult for non-specialists due to absence of easy to use tools. Here we present a unified command-line interface tool written in **Py**thon to **p**ile-**up** Hi-C data stored in the widely used and versatile .**cool** format (29) (*coolpup.py*). We applied *coolpup.py* to published data to investigate the effect of different normalization strategies on the resulting pileups, and to replicate previously shown results to verify *coolpup.py*'s algorithm. Then we also present a novel variation of the pileup approach implemented in *coolpup.py* that retains some of the locus-specific information and would allow more detailed statistical analysis of looping interactions in Hi-C data.

MATERIALS AND METHODS

Sources of datasets & data analysis

As a proof of principal, we applied *coolpup.py* to publicly available Hi-C data (23, 24) (GSE96107, GSE98671) using *distiller* to obtain *.cool* files filtered with a map quality (mapq) of ≥30. We used this data at 5 kbp resolution. In addition *.cool* files for single-nucleus Hi-C (snHi-C), together with coordinates of loops and TADs used in the original publication (kindly shared by Hugo Brandão) (4, 8), were reanalysed at 10 kbp resolution. RING1B and H3K27me3 ChIP-seq peaks were taken from (ref. 30) and lifted over to the mm9 mouse genome assembly. The coordinates of biochemically defined CpG islands were taken from (ref. 31). CTCF ChIP-seq peaks were taken from (ref. 23) and, following liftover to the *mm9* assembly, intersected with CTCF motifs found in the mm9 genome using Biopython's *motifs* module (32). A human CTCF position-frequency matrix was downloaded from JASPAR (MA0139.1). We used only motifs with a score above 7, and discounted peaks containing more than one motif. We used a simple RefSeq gene annotation, including intervals between single transcription start and end sites, and strand information (given the limitations of data resolution only genes > 30 kbp in length were interrogated).

Regions of high insulation (meaning low number of contacts crossing this regions) in the Bonev et al. data were called using *cooltools diamond-insulation* from 25 kbp resolution data and a window size of 1 Mb. The output was filtered to exclude boundaries with strength <0.1, and then pairs of consecutive boundaries were combined to create an annotation of TADs. TADs longer than 1,500 kbp were not used due to their likely artefactual nature (based on both visual inspection, and the fact that TAD sizes are reported to be on the order of a few hundred kbp in mammalian cells (4)). The same loop annotations for mouse embryonic stem (ES) cells were used as in our recent preprint (13).

All figure panels were created using *matplotlib (33)* and assembled in Inkscape.

Coolpup.py implementation

Coolpup.py is a versatile tool that uses *.cool* files as the main input together with a bed (chrom, start, end) or pairbed (chrom1, start1, end1, chrom2, start2, end2) file to define the regions under investigation. The tool is implemented as a single *python* script which parses all arguments via *argparse*, performs the computation and saves the output file(s). It leverages the scientific python environment, taking advantage of *numpy (34)*, *scipy (35)* and *pandas (36)*. The code is available on github (https://github.com/Phlya/coolpuppy) and the package can be installed using *pip*, which then makes *coolpup.py* script available in the command line. Alternatively, all main functions can be accessed directly from *python*.

The overall procedure for piling up a lot of small regions is the following. To minimize number of file reads (at the cost of required computer memory), a sparse representation of each chromosome's Hi-C contact matrix is loaded into memory. Then, using an iterator, each required location (on- or off-diagonal) is individually retrieved to generate a corresponding submatrix from the data (with some specified padding around the centre of the regions of interest (ROI), and added to the matrix of the same shape, initialized with zeros, while keeping track of the number of summed up regions. If specified, coverage of the window on each side is recorded. Similarly, if needed, the window (and the coverage) is rescaled to a required shape. This is done for all chromosomes (optionally, in parallel using *multiprocessing*), and then all of the results are summed and then divided by the total number of windows. If specified, coverage normalization is

applied at this stage. Then, unless otherwise specified, a normalization to remove distance-dependency of contact probability is applied. In most cases the best and most efficient way is to use a (chromosome-wide) expected value for each diagonal of the matrix, which can be obtained for a cooler file using, for example, *cooltools compute-expected*. This information is then used to create a matrix of expected interactions, under the assumption that the probability of interactions only depends on distance. Alternatively, if the expected values are not available, for example for single-cell Hi-C data, this normalization can be performed using randomly shifted control regions. In that case the whole pile-up procedure is repeated, but the coordinates are randomly shifted. In the end, the resulting matrix of averaged interactions is divided by the normalizing matrix to remove effects of distance.

If not specified, balanced data with chromosome-wide expected normalization was used when creating pileups, except for the zygote dataset, where randomly shifted controls and coverage normalization were used instead. For plotting average TADs, apart from observed/expected pileups in Figure 2B, we show the same matrices after re-introduction of slow decay of interaction probability with distance. We simply multiply the observed/expected matrix by a matrix of the same shape where each diagonal i has the value $i^{-0.25}$, starting from $0^{0.25}=1$.

Performance profiling

Coolpup.py performance was tested on the University of Edinburgh Open Grid Scheduler cluster (Eddie3). We used deep Hi-C data for mouse ES cells from (ref. 23) for testing. To generate the required large number of coordinates for testing, we used coordinates of the B3 repeat from RepeatMasker track available from UCSC Genome Browser. For coordinate pairs, we used all pairs of convergent CTCF sites, described above. A separate job was submitted for each measurement, and the runtime of the *coolpup.py* call was recorded. Subsets of different sizes were generated using *coolpup.py*'s `--subset` argument. Where not specified, 4 compute cores were utilized. All measurements were performed 5 times. Plotted everywhere are actual measured runtime values and means ±SD, using the *seaborn* plotting package (37).

RESULTS

Different normalization strategies implemented in *coolpup.py*

Coolpup.py is a tool designed to create pileups from Hi-C data. Hi-C data can be normalized in different ways to remove either technical biases, or uninteresting (in this context) biological signal of the decay of contact probability with genomic distance. *Coolpup.py* provides ways to deal with both of these problems.

Hi-C data are usually normalized to remove systematic biases, such as GC-content or restriction site frequency (38). *Cooler* implements a matrix balancing (visibility equalization) approach to remove all potential biases (39) and, when available, it is recommended to use balanced data for pile-ups. However sometimes, for example in single-cell Hi-C, removing biases is impossible due to sparsity of data. Therefore, using unbalanced data is also an option in *coolpup.py*. However, because of the averaging of multiple regions during the pile-up procedure, the effect of biases can be partially mitigated by collecting the coverage information of all averaged regions (i.e. the total number of contacts of the bins in the chromosome), and dividing the final matrix by the outer product of the coverage of two dimensions of the matrix (18). As illustrated in Figure 1A for both CTCF and polycomb (RING1B) loops, this approach reduces coverage variability between bins and removes sharp crosses from the central bin present with unbalanced data. This normalization seems to slightly overcorrect, i.e. the value of the central pixel is consistently

somewhat lower than when using balanced data. However, the results overall look more similar to balanced data than without coverage normalization.

In addition to normalization to remove biases in Hi-C data, it is often desirable to remove the distance-dependency of contact probability, since it can have a very strong effect on the resulting pileup and can obscure interesting properties, such as enrichment in its centre. The general approach to perform this is to create a vector of expected contact frequency, which usually corresponds to the averaged value of the Hi-C map at each diagonal per chromosome. A file with such information for a cooler file (for balanced data) can be obtained using *cooltools compute-expected*. This can then be used in *coolpup.py* to normalize the pileups. However sometimes the expected information is unavailable, for example, in single-cell Hi-C it can be too noisy. In that case an alternative approach to remove distance-dependency of contact frequency can be used: for each position in the Hi-C map being averaged, a matched set of randomly shifted control regions with the same distance separation is used (18). In this way, by creating many such control regions for each ROI, it is possible to estimate the expected frequency of interactions even for sparse single-cell Hi-C data. As shown in Figure 1B, both of these approaches are excellent at removing distance-dependency of interactions in the pileups and produce visually indistinguishable results. However, for a small set of regions (e.g. RING1B associated loops) a higher number of randomly shifted controls for each ROI is required to prevent noise. We note that for local pile-ups (especially with rescaling; see below) random controls perform better than simple normalization to expected values (data not shown).

Pile-up variations

While the basic pile-up procedure is simple enough, there are multiple variations built in to *coolpup.py* which are tailored to answer different biological questions. The following ones are trivial, but worth mentioning. For example, often it is desirable to restrict the minimal and/or maximal separation of analysed sites, either to remove short-range artefacts, or to analyse the distribution of enrichment signal across different distance scales. Only certain chromosomes might need to be included, or, with too many regions of interest, a random subset can be taken to speed up the computation.

A popular variation of the pile-up approach is “local” pile-up: an analysis which focuses on near-diagonal features. For example, we averaged regions of high insulation annotated in the deep ES cell Hi-C dataset to visualize insulation strength after Auxin-induced degradation of CTCF (24)(Figure 2A). In this case the pileups are performed in the same way as previous off-diagonal pileups, however the regions that are averaged lie on the main diagonal of the Hi-C map. A variation of this approach is local pileups with rescaling to analyse features of different size, for example, TADs (18). As an example, here TADs based on aforementioned regions of high insulation annotated in (ref. 23) were averaged to visualize changes in TAD strength upon CTCF degradation (24) (Figure 2B). Here all windows centred on regions of interest are rescaled to the same size, and then averaged.

Pileups are a particularly important approach to analysing very low depth datasets to uncover genome-wide average patterns, which are indiscernible when looking at individual regions in such shallow data. Here we apply *coolpup.py* to reproduce results from a dataset comprising pooled data from a few single cells, to show a loss of loops and TADs in mouse zygotes lacking SCC1 (RAD21), the kleisin subunit of cohesin (8). Since the material is so limiting and data are based on single cells, total number of contacts in this dataset is very low: 4.8 and 9.2 million contacts in *Scc1*$^{+/+}$ and *Scc1*$^{-/-}$, respectively. However, we can successfully

perform pileups, both with “traditional” averaging of loops, and local pileups of TADs with rescaling, similarly to the original paper to observe the loss of both loops and TADs upon deletion of cohesin (Figure 2C).

All pile-up approaches include averaging of multiple regions, a drawback of which is loss of locus-specific information. We therefore designed a novel approach that retains some information about the specific loci used in the analysis. In this approach, we pile-up a single region against multiple other regions; the same can be done for each of many regions in a set against all other regions. Then by extracting the value in the central pixel in pileups for each region, we can get a “loop-ability” value, which can then be related to other features of analysed regions, such as the level of occupancy by different factors. To confirm that this approach can work, we checked some example regions that displayed high or low level of “loop-ability”, to ensure that the values we observed were not due to noise from piling up interactions of a single region (see two examples in Figure 2D). A simple proof of principle analysis highlights the interactions between sites bound by polycomb group proteins in mouse ES cells. By splitting the CpG islands in the mouse genome into RING1B (a core component of PRC1) negative, RING1B positive, and RING1B and H3K27me3 positive sets, we observe high “loop-ability” values for the two latter groups, while the RING1B negative CpG islands have close to no enrichment (Figure 2E).

***Coolpup.py* can deal with huge numbers of regions**

Creating pileups from intersections of genomic regions can require averaging a huge number of 2D windows: the number of 2-combinations grows quickly with the number of regions. For example, with ~1000 regions per chromosome (which is approximately equivalent to the number of genes), requires averaging of ~10,000,000 regions, several orders of magnitude more than the number of regions usually averaged, such as number of annotated loops (~10,000). Therefore, a general-purpose tool for creating pileups should scale well with the number of averaged 2D windows. To facilitate this, *coolpup.py* performs a very low number of read operations on the Hi-C data – only once per chromosome (or twice, when using randomly shifted controls). Whilst in this way the whole Hi-C matrix of a chromosome has to be loaded into memory, it is only stored in a sparse format, and so conventional Hi-C datasets can be analysed on a regular desktop (although multi-billion contact datasets might require a high-memory machine; data not shown).

To test the performance of *coolpup.py* and how this depends on number of regions of interest, we measured the runtime with varying number of two-sided coordinate pairs (mimicking loop annotation) (Figure 3A), and varying the number of one-sided coordinate interactions being averaged (Figure 3B) in the deep Hi-C data from mouse ES cells (23). The runtime was almost constant up to ~1 million “loops”, with slow increase thereafter until surpassing 8 million “loops”, when it starts increasing much faster (Figure 3A). Similarly, in the latter analysis, runtime didn’t increase up to 1600 regions of interest, and started growing linearly after surpassing 6400 ROIs.

Since *coolpup.py* supports parallel processing to speed up analyses, we also tested how well it scales with the number of computer cores used. We measured the runtime of the same analysis performed with varying number of cores (Figure 3C) and showed that the runtime almost linearly shortens with additional processes. This means the parallelization strategy used in *coolpup.py* is efficiently utilizing available CPU cores and when available, we recommend using many cores to speed up computation.

DISCUSSION

With the large efforts being made in deciphering the structure and function of the genome in 3D, efficient, robust and versatile tools are required to facilitate quick hypothesis testing. Unlike for RNA-seq, ChIP-seq and other genome-wide methods, analysis of complex Hi-C data remains a challenge only readily accessible to specialists in the field due to an absence of easy to use informatics tools, with a few exceptions. One popular analysis applied to Hi-C data is pile-ups, which show an average genome-wide view of a selected set of regions in the 2D Hi-C interaction matrix: a very visual and intuitive approach at analysing data.

Here we presented *coolpup.py*, a versatile tool to perform pile-up analysis on Hi-C data in *.cool* format. Apart from simple generation of pile-ups, *coolpup.py* can be used to explore different data normalization strategies. While we recommend using balanced data with normalization to chromosome-wide expected interaction frequency, in certain cases a different normalization strategy can be beneficial. Similarly, exploring other parameters of the algorithm (such as minimal separation between averaged loop bases, or minimal length of locally averaged features) is straight forward with *coolpup.py*. Since *coolpup.py* is designed as a command-line tool and allows reading the coordinates of regions from standard input, it is compatible with computational pipelines, and can be readily used in shared computing environments. Moreover, it remains accessible for non-specialists with minimal knowledge of the command line and no programming experience. *Coolpup.py* should aid in improving reproducibility by providing a standardised approach for pile-up analysis which is intuitive and therefore accessible to both specialists and non-specialist alike. We hope that it will facilitate research into the 3D organization of the genome by allowing easy to use, versatile and efficient generation of pileups.

AVAILABILITY

Coolpup.py is available on GitHub (https://github.com/Phlya/coolpuppy) and installable from the Python Package Index (PyPI). All code used to generate the figures for this paper is available on GitHub (https://github.com/Phlya/coolpuppy_paper).

ACKNOWLEDGEMENTS

We are grateful to Maksim Imakaev, Nezar Abdennur, Anton Goloborodko, Hugo Brandão, George Spracklin and Sergey Venev for discussions, help and advice about the *cooler* ecosystem. We also thank Sergey Ulianov for critically reading the manuscript.

FUNDING

I.M.F is funded by a PhD studentship from the Darwin Trust of Edinburgh. W.A.B is funded by a Medical Research Council University Unit programme grant [MC_ UU_00007/2]. RSI is supported by a seed award from the University of Edinburgh Simons Initiative for the Developing Brain (SIDB; a Simons Foundation Autism Research Initiative – SFARI). Funding for open access charge: [MC_ UU_00007/2].

CONFLICT OF INTEREST

Authors declare no conflict of interest.

REFERENCES

1. Dekker,J., Rippe,K., Dekker,M. and Kleckner,N. (2002) Capturing Chromosome Conformation. *Science*, **295**, 1306–1311.

2. Barutcu,A.R., Fritz,A.J., Zaidi,S.K., van Wijnen,A.J., Lian,J.B., Stein,J.L., Nickerson,J.A., Imbalzano,A.N. and Stein,G.S. (2016) C-ing the Genome: A Compendium of Chromosome Conformation Capture Methods to Study Higher-Order Chromatin Organization. *J. Cell. Physiol.*, **231**, 31–35.

3. Lieberman-Aiden,E., van Berkum,N.L., Williams,L., Imakaev,M., Ragoczy,T., Telling,A., Amit,I., Lajoie,B.R., Sabo,P.J., Dorschner,M.O., *et al.* (2009) Comprehensive mapping of long range interactions reveals folding principles of the human genome. *Science*, **326**, 289–293.

4. Rao,S.S.P., Huntley,M.H., Durand,N.C., Stamenova,E.K., Bochkov,I.D., Robinson,J.T., Sanborn,A.L., Machol,I., Omer,A.D., Lander,E.S., *et al.* (2014) A 3D Map of the Human Genome at Kilobase Resolution Reveals Principles of Chromatin Looping. *Cell*, **159**, 1665–1680.

5. Lajoie,B.R., Dekker,J. and Kaplan,N. (2014) The Hitchhiker's guide to Hi-C analysis: Practical guidelines. *Methods*, 10.1016/j.ymeth.2014.10.031.

6. Fudenberg,G., Imakaev,M., Lu,C., Goloborodko,A., Abdennur,N. and Mirny,L.A. (2016) Formation of Chromosomal Domains by Loop Extrusion. *Cell Reports*, **15**, 2038–2049.

7. Sanborn,A.L., Rao,S.S.P., Huang,S.-C., Durand,N.C., Huntley,M.H., Jewett,A.I., Bochkov,I.D., Chinnappan,D., Cutkosky,A., Li,J., *et al.* (2015) Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes. *Proc Natl Acad Sci U S A*, **112**, E6456–E6465.

8. Gassler,J., Brandão,H.B., Imakaev,M., Flyamer,I.M., Ladstätter,S., Bickmore,W.A., Peters,J.-M., Mirny,L.A. and Tachibana,K. (2017) A mechanism of cohesin□dependent loop extrusion organizes zygotic genome architecture. *The EMBO Journal*, 10.15252/embj.201798083.

9. Lupiáñez,D.G., Kraft,K., Heinrich,V., Krawitz,P., Brancati,F., Klopocki,E., Horn,D., Kayserili,H., Opitz,J.M., Laxova,R., *et al.* (2015) Disruptions of Topological Chromatin Domains Cause Pathogenic Rewiring of Gene-Enhancer Interactions. *Cell*, **161**, 1012–1025.

10. Franke,M., Ibrahim,D.M., Andrey,G., Schwarzer,W., Heinrich,V., Schöpflin,R., Kraft,K., Kempfer,R., Jerković,I., Chan,W.-L., *et al.* (2016) Formation of new chromatin domains determines pathogenicity of genomic duplications. *Nature*, **538**, 265–269.

11. Canela,A., Maman,Y., Jung,S., Wong,N., Callen,E., Day,A., Kieffer-Kwon,K.-R., Pekowska,A., Zhang,H., Rao,S.S.P., *et al.* (2017) Genome Organization Drives Chromosome Fragility. *Cell*, **170**, 507-521.e18.

12. Joshi,O., Wang,S.-Y., Kuznetsova,T., Atlasi,Y., Peng,T., Fabre,P.J., Habibi,E., Shaik,J., Saeed,S., Handoko,L., *et al.* (2015) Dynamic Reorganization of Extremely Long-Range Promoter-Promoter Interactions between Two States of Pluripotency. *Cell Stem Cell*, **17**, 748–757.

13. McLaughlin,K.A., Flyamer,I.M., Thomson,J.P., Mjoseng,H.K., Shukla,R., Williamson,I., Grimes,G.R., Illingworth,R.S., Adams,I.R., Pennings,S., *et al.* (2019) DNA methylation directs polycomb-dependent 3D genome re- organisation in naive pluripotency. *bioRxiv*, 10.1101/527309.

14. de Wit,E., Bouwman,B.A.M., Zhu,Y., Klous,P., Splinter,E., Verstegen,M.J.A.M., Krijger,P.H.L., Festuccia,N., Nora,E.P., Welling,M., *et al.* (2013) The pluripotent genome in three dimensions is shaped around pluripotency factors. *Nature*, **501**, 227–231.

15. Nagano,T., Lubling,Y., Várnai,C., Dudley,C., Leung,W., Baran,Y., Mendelson Cohen,N., Wingett,S., Fraser,P. and Tanay,A. (2017) Cell-cycle dynamics of chromosomal organization at single-cell resolution. *Nature*, **547**, 61–67.

16. Nagano,T., Lubling,Y., Stevens,T.J., Schoenfelder,S., Yaffe,E., Dean,W., Laue,E.D., Tanay,A. and Fraser,P. (2013) Single-cell Hi-C reveals cell-to-cell variability in chromosome structure. *Nature*, **502**, 59–64.

17. Stevens,T.J., Lando,D., Basu,S., Atkinson,L.P., Cao,Y., Lee,S.F., Leeb,M., Wohlfahrt,K.J., Boucher,W., O'Shaughnessy-Kirwan,A., *et al.* (2017) 3D structures of individual mammalian genomes studied by single-cell Hi-C. *Nature*, **544**, 59–64.

18. Flyamer,I.M., Gassler,J., Imakaev,M., Brandão,H.B., Ulianov,S.V., Abdennur,N., Razin,S.V., Mirny,L.A. and Tachibana-Konwalski,K. (2017) Single-nucleus Hi-C reveals unique chromatin reorganization at oocyte-to-zygote transition. *Nature*, 10.1038/nature21711.

19. Tan,L., Xing,D., Chang,C.-H., Li,H. and Xie,X.S. (2018) Three-dimensional genome structures of single diploid human cells. *Science*, **361**, 924–928.

20. Ulianov,S.V., Tachibana□Konwalski,K. and Razin,S.V. (2017) Single-cell Hi-C bridges microscopy and genome-wide sequencing approaches to study 3D chromatin organization. *BioEssays*, **39**, 1700104.

21. Díaz,N., Kruse,K., Erdmann,T., Staiger,A.M., Ott,G., Lenz,G. and Vaquerizas,J.M. (2018) Chromatin conformation analysis of primary patient tissue using a low input Hi-C method. *Nature Communications*, **9**, 4938.

22. Schwarzer,W., Abdennur,N., Goloborodko,A., Pekowska,A., Fudenberg,G., Loe-Mie,Y., Fonseca,N.A., Huber,W., H. Haering,C., Mirny,L., *et al.* (2017) Two independent modes of chromatin organization revealed by cohesin removal. *Nature*, **551**, 51–56.

23. Bonev,B., Mendelson Cohen,N., Szabo,Q., Fritsch,L., Papadopoulos,G.L., Lubling,Y., Xu,X., Lv,X., Hugnot,J.-P., Tanay,A., *et al.* (2017) Multiscale 3D Genome Rewiring during Mouse Neural Development. *Cell*, **171**, 557-572.e24.

24. Nora,E.P., Goloborodko,A., Valton,A.-L., Gibcus,J.H., Uebersohn,A., Abdennur,N., Dekker,J., Mirny,L.A. and Bruneau,B.G. (2017) Targeted Degradation of CTCF Decouples Local Insulation of Chromosome Domains from Genomic Compartmentalization. *Cell*, **169**, 930-944.e22.

25. Rao,S., Huang,S.-C., Hilaire,B.G.S., Engreitz,J.M., Perez,E.M., Kieffer-Kwon,K.-R., Sanborn,A.L., Johnstone,S.E., Bochkov,I.D., Huang,X., *et al.* (2017) Cohesin Loss Eliminates All Loop Domains, Leading To Links Among Superenhancers And Downregulation Of Nearby Genes. *bioRxiv*, 10.1101/139782.

26. Kruse,K., Diaz,N., Enriquez-Gasca,R., Gaume,X., Torres-Padilla,M.-E. and Vaquerizas,J.M. (2019) Transposable elements drive reorganisation of 3D chromatin during early embryogenesis. *bioRxiv*, 10.1101/523712.

27. Abdennur,N., Schwarzer,W., Pekowska,A., Shaltiel,I.A., Huber,W., Haering,C.H., Mirny,L. and Spitz,F. (2018) Condensin II inactivation in interphase does not affect chromatin folding or gene expression. *bioRxiv*, 10.1101/437459.

28. Rowley,M.J., Lyu,X., Rana,V., Ando-Kuri,M., Karns,R., Bosco,G. and Corces,V.G. (2019) Condensin II Counteracts Cohesin and RNA Polymerase II in the Establishment of 3D Chromatin Organization. *Cell Reports*, **26**, 2890-2903.e3.

29. Abdennur,N. and Mirny,L. (2019) Cooler: scalable storage for Hi-C data and other genomically-labeled arrays. *bioRxiv*, 10.1101/557660.

30. Illingworth,R.S., Moffat,M., Mann,A.R., Read,D., Hunter,C.J., Pradeepa,M.M., Adams,I.R. and Bickmore,W.A. (2015) The E3 ubiquitin ligase activity of RING1B is not essential for early mouse development. *Genes Dev*, **29**, 1897–1902.

31. Illingworth,R.S., Gruenewald-Schneider,U., Webb,S., Kerr,A.R.W., James,K.D., Turner,D.J., Smith,C., Harrison,D.J., Andrews,R. and Bird,A.P. (2010) Orphan CpG Islands Identify Numerous Conserved Promoters in the Mammalian Genome. *PLOS Genetics*, **6**, e1001134.

32. Cock,P.J.A., Antao,T., Chang,J.T., Chapman,B.A., Cox,C.J., Dalke,A., Friedberg,I., Hamelryck,T., Kauff,F., Wilczynski,B., *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.

33. Hunter,J.D. (2007) Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering*, **9**, 90–95.

34. Walt,S. van der, Colbert,S.C. and Varoquaux,G. (2011) The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, **13**, 22–30.

35. Jones,E., Oliphant,T., Peterson,P. and others (2001) SciPy: Open source scientific tools for Python.

36. McKinney,W. (2010) Data Structures for Statistical Computing in Python.

37. Michael Waskom, Olga Botvinnik, Drew O'Kane, Paul Hobson, Joel Ostblom, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, *et al.* (2018) mwaskom/seaborn: v0.9.0 (July 2018) Zenodo.

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

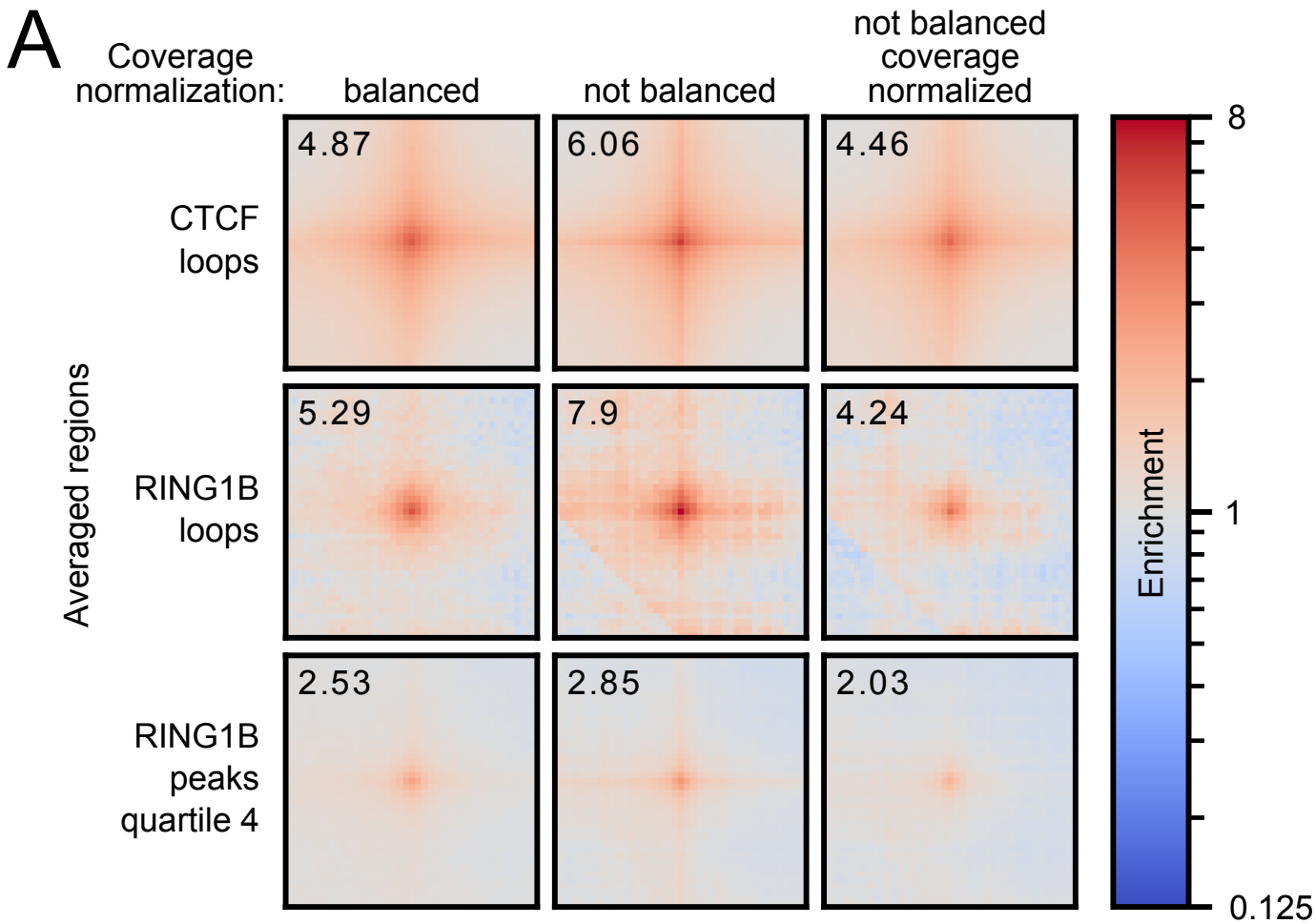
39. Imakaev,M., Fudenberg,G., McCord,R.P., Naumova,N., Goloborodko,A., Lajoie,B.R., Dekker,J. and Mirny,L.A. (2012) Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nature Methods*, **9**, 999–1003.

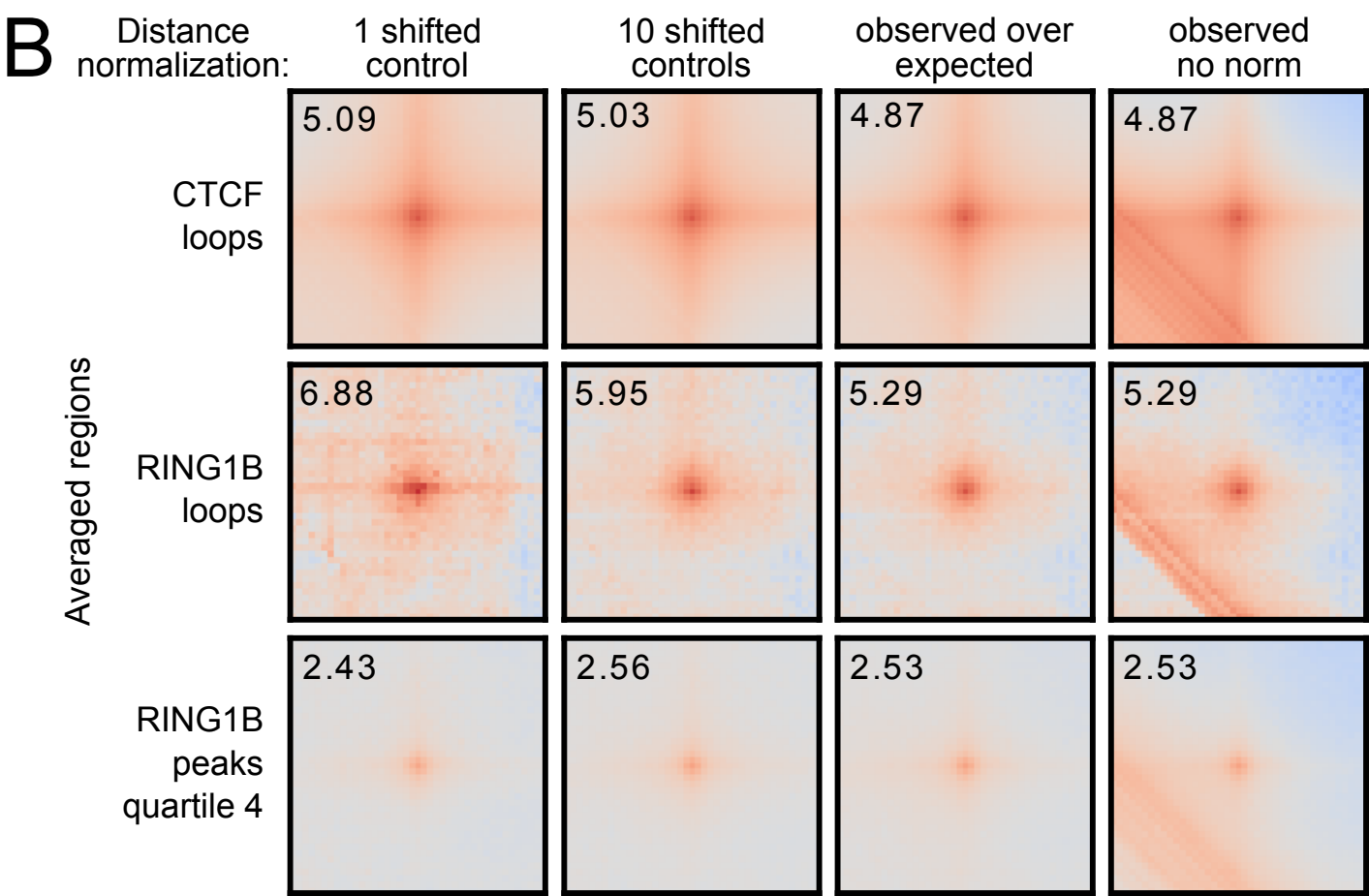
FIGURE LEGENDS

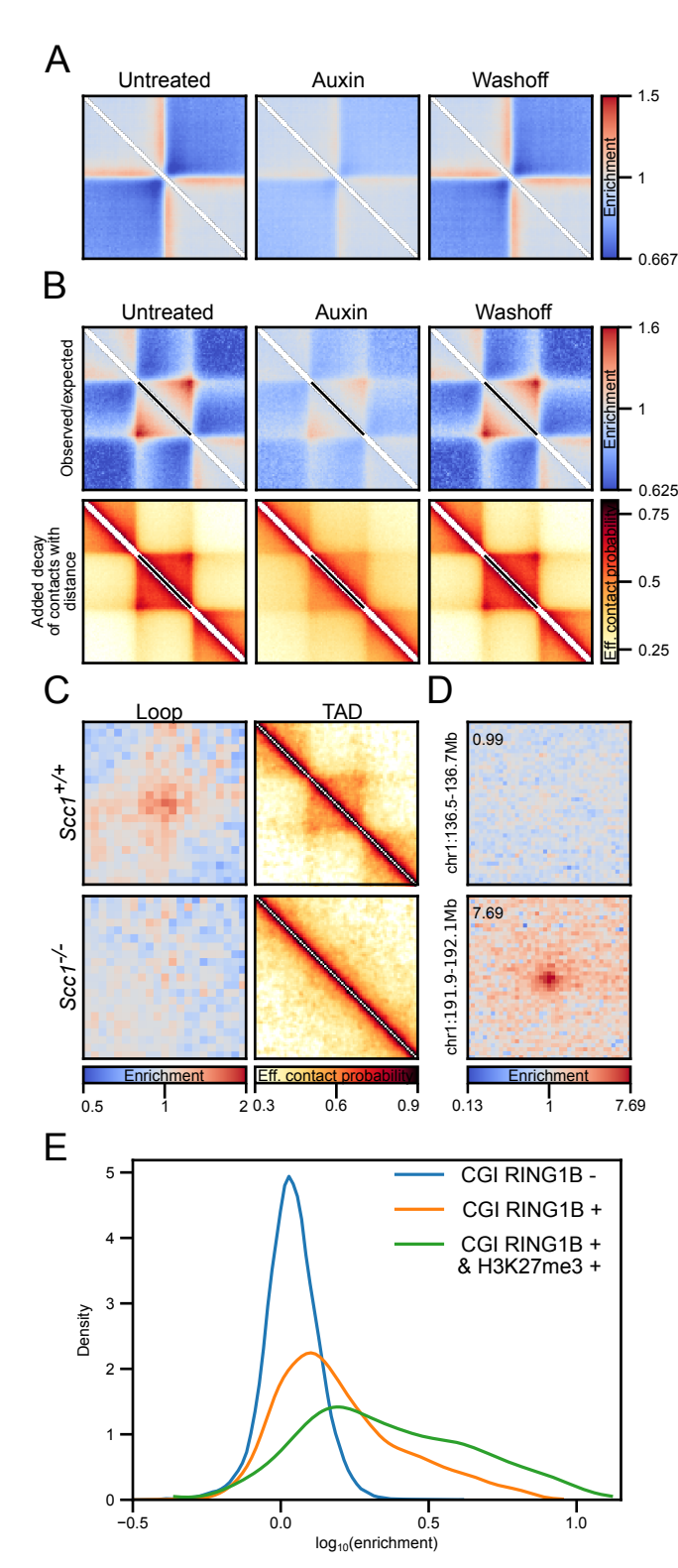
Figure 1. Hi-C data normalization strategies. (**A**) Comparison of coverage normalization strategies for pile-up analyses using mouse ES cell Hi-C data (23). In columns, normalization approaches: matrix balancing (iterative correction); no normalization; no balancing with coverage normalization of the pileups. In rows, different averaged regions: loops, associated with CTCF (n=6536); loops, associated with RING1B (n=104) (see Methods); all pairwise combinations of high RING1B peak regions from the 4th quartile (by RING1B ChIP-seq read count) (n=2660 of peak regions). All pileups normalized to average of top-left and bottom-right corner pixels to bring them to same scale. Number in top left corner is value of the central pixel. 5 kb resolution with 100 kb padding around the central pixel. Colour is shown in log-scale and shows enrichment of interactions. (**B**) Same as A, but for different approaches to remove distance-dependency of contact probability with balanced data. In columns: single randomly shifted control regions per ROI; ten randomly shifted control per ROI; normalization to chromosome-wide expected; no normalization. Same rows as in (A).

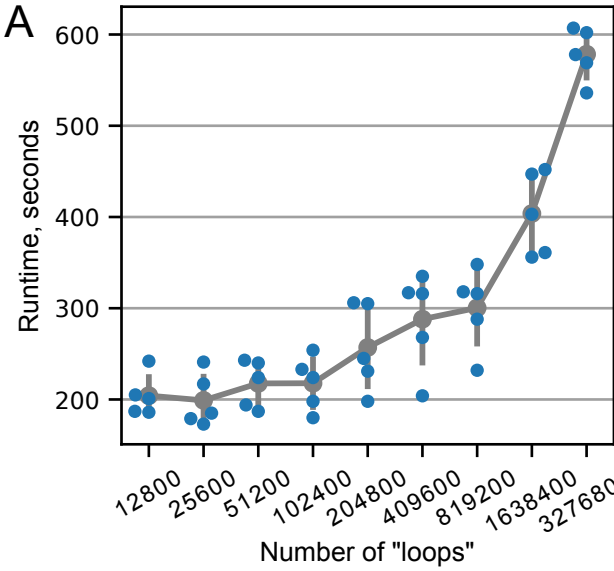
Figure 2. Pileup variations. (**A**) Local pileups of high insulating region in ES cells across untreated, auxin-treated and wash-off conditions in CTCF-AID Hi-C data (24). 25 kb resolution data with 1,000 kb padding around the central pixel. (**B**) Local rescaled pileups of TADs (defined based on high insulating regions) across same data as in (A) from 5 kbp resolution data. Top row: raw observed over expected pileups; bottom row: pileups after artificial re-introduction of shallow decay of contact probability (P_c=$s^{-0.25}$, see Methods) for easier visual interpretation of the data (18). (**C**) Loop and rescaled TAD pileups for pooled single-cell Hi-C data showing loss of structures in Scc1$^{-/-}$ zygotes (8) (**D**) Two examples of anchored pileups from RING1B+/H3K27me3+ CpG islands, with no visible enrichment (top), or with very prominent enrichment (bottom). The anchored region is on the left side of the pileup, and its coordinates (including the padding) are shown on the left. The value of the central pixel ("loop-ability") shown in top left corner. (**E**) Distribution of "loop-ability" values of CpG islands not bound by RING1B, CpG islands bound by RING1B, and CpG islands bound by RING1B and also marked by H3K27me3.

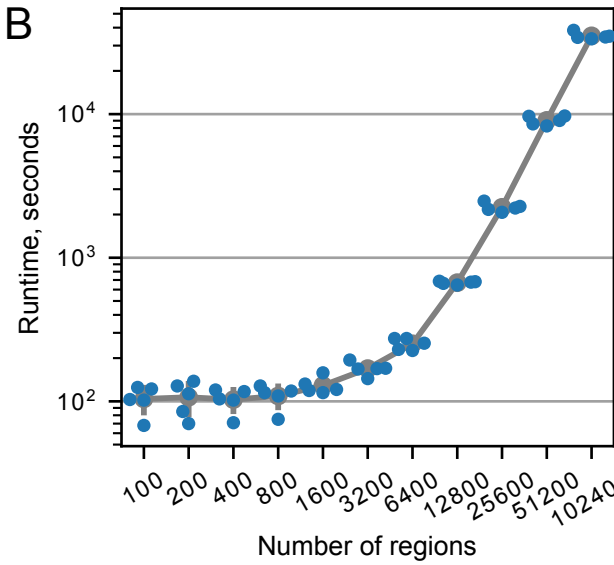
Figure 3. Performance profiling. (**A**) Runtime (seconds) of *coolpup.py* with varying number of averaged "loop", using 4 cores. (**B**) Same as (A), but for number of linear regions for between which interactions are averaged. (**C**) Runtime of the same analysis with 5000 linear regions and a varying number of cores.

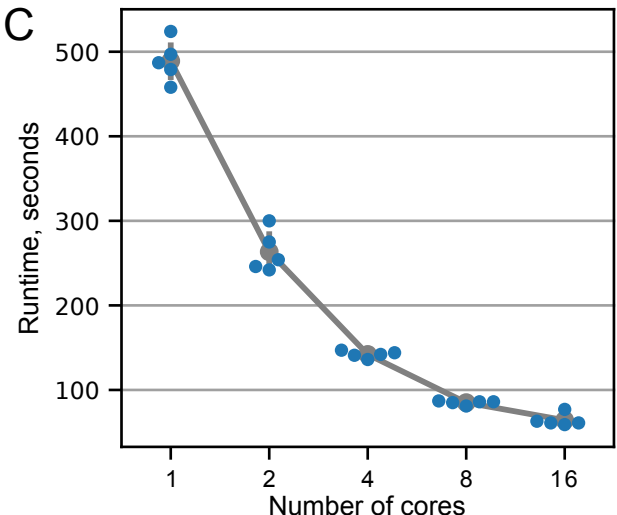

A
Coverage normalization:
balanced
not balanced
not balanced coverage normalized
Averaged regions
CTCF loops
RING1B loops
RING1B peaks quartile 4
4.87
6.06
4.46
5.29
7.9
4.24
2.53
2.85
2.03
Enrichment
8
1
0.125


B
Distance normalization:
1 shifted control
10 shifted controls
observed over expected
observed no norm
Averaged regions
CTCF loops
RING1B loops
RING1B peaks quartile 4
5.09
5.03
4.87
4.87
6.88
5.95
5.29
5.29
2.43
2.56
2.53
2.53


A
Untreated
Auxin
Washoff
Enrichment
1.5
1
0.667
B
Untreated
Auxin
Washoff
Observed/expected
Enrichment
1.6
1
0.625
Added decay of contacts with distance
Eff. contact probability
0.75
0.5
0.25
C
Loop
TAD
Scc1+/+
Scc1-/-
Enrichment
0.5
1
2
Eff. contact probability
0.3
0.6
0.9
D
chr1:136.5-136.7Mb
0.99
chr1:191.9-192.1Mb
7.69
Enrichment
0.13
1
7.69
E
CGI RING1B -
CGI RING1B +
CGI RING1B + & H3K27me3 +
Density
0
1
2
3
4
5
−0.5
0.0
0.5
1.0
log10(enrichment)


A
Runtime, seconds
600
500
400
300
200
12800
25600
51200
102400
204800
409600
819200
1638400
3276800
Number of "loops"


B
Runtime, seconds
10^4
10^3
10^2
100
200
400
800
1600
3200
6400
12800
25600
51200
102400
Number of regions


C
Runtime, seconds
500
400
300
200
100
1
2
4
8
16
Number of cores