Bioinformatics, 36(10), 2020, 2980–2985
doi: 10.1093/bioinformatics/btaa073
Advance Access Publication Date: 31 January 2020
Original Paper

OXFORD

Genome analysis

Coolpup.py: versatile pile-up analysis of Hi-C data

Ilya M. Flyamer [1,*], Robert S. Illingworth[2] and Wendy A. Bickmore[1]

[1]MRC Human Genetics Unit, Institute of Genetics and Molecular Medicine, The University of Edinburgh, Edinburgh, EH4 2XU, UK and [2]MRC Centre for Regenerative Medicine, Institute for Regeneration and Repair, The University of Edinburgh, Edinburgh, EH16 4UU, UK

*To whom correspondence should be addressed.

Associate Editor: Peter Robinson

Received on October 3, 2019; revised on January 16, 2020; editorial decision on January 25, 2020; accepted on January 27, 2020

Abstract

Motivation: Hi-C is currently the method of choice to investigate the global 3D organization of the genome. A major limitation of Hi-C is the sequencing depth required to robustly detect loops in the data. A popular approach used to mitigate this issue, even in single-cell Hi-C data, is genome-wide averaging (piling-up) of peaks, or other features, annotated in high-resolution datasets, to measure their prominence in less deeply sequenced data. However, current tools do not provide a computationally efficient and versatile implementation of this approach.

Results: Here, we describe *coolpup.py*—a versatile tool to perform pile-up analysis on Hi-C data. We demonstrate its utility by replicating previously published findings regarding the role of cohesin and CTCF in 3D genome organization, as well as discovering novel details of Polycomb-driven interactions. We also present a novel variation of the pile-up approach that can aid the 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 pile-ups.

Availability and implementation: *Coolpup.py* is cross-platform, open-source and free (MIT licensed) software. Source code is available from https://github.com/Phlya/coolpuppy and it can be installed from the Python Packaging Index.

Contact: ilya.flyamer@igmm.ed.ac.uk

1 Introduction

Major advances in the study of 3D genome organization have come from the development of a family of Chromosome Conformation Capture (3C) methods (Dekker, 2002). While these all rely on the same principle of *in situ* proximity ligation of cross-linked and digested chromatin, the scope of each method varies depending on experimental processing and the method of quantification of the 3C library (Barutcu *et al.*, 2016). Hi-C, a genome-wide 3C-derivative, is the method of choice to investigate the organization of the whole genome (Lieberman-Aiden *et al.*, 2009; Rao *et al.*, 2014).

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 (Lajoie *et al.*, 2015). This limits the resolution at which genomes can be analysed in 3D, since going beyond ~5 kb resolution requires billions of read pairs for a mammalian genome.

Looping interactions are among the most interesting features that can be studied using Hi-C. Chromatin loops bring distal regions in the genome into close proximity and are manifest in Hi-C data as foci of increased interaction frequency (Rao *et al.*, 2014). The majority of loops identified in Hi-C data from mammalian cells correspond to CTCF/cohesin-associated interactions, created by loop extrusion (Fudenberg *et al.*, 2016; Gassler *et al.*, 2017; Sanborn *et al.*, 2015). 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 are suggested to constrain enhancer-promoter communication in some cases (Franke *et al.*, 2016; Lupiáñez *et al.*, 2015; Williamson *et al.*, 2019) and might be related to genome stability (Canela *et al.*, 2017, 2019), while some loops have been suggested to correspond to enhancer-promoter contacts (Rao *et al.*, 2014). In addition, distal polycomb sites can be brought together in 'loops' (Joshi *et al.*, 2015; McLaughlin *et al.*, 2019).

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 (Rao *et al.*, 2014). This means that the vast majority of Hi-C datasets cannot 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, that have been annotated in a high-depth dataset (Rao *et al.*, 2014). This idea is very similar to 'average/aggregate profiles' 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 for quantification of their average prominence. Apart from quantifying the strength of

known features, the same analysis can be used to investigate whether certain regions, defined for example based on ChIP-seq peaks, tend to interact with each other on average above background. To our knowledge, the first application of pile-up-like analysis was to investigate clustering of pluripotency factor binding sites in mouse embryonic stem (ES) cells (de Wit *et al.*, 2013). Pile-up analysis can also aid in the discovery of novel drivers of interactions.

Hi-C is a cell population-based method, and only provides population average measurements. Several single-cell Hi-C approaches have been published (Flyamer *et al.*, 2017; Nagano *et al.*, 2017, 2013; Stevens *et al.*, 2017; Tan *et al.*, 2018; reviewed in Ulianov *et al.*, 2017); however, none of these provides data depth or resolution comparable to that which can be obtained from a population of thousands of cells (Díaz *et al.*, 2018): 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 (Flyamer *et al.*, 2017). These are all based on the 'pile-up' approach described above using data from single cells for the regions corresponding to specific features identified in population Hi-C data, to boost the amount of reads used in the analysis.

Since its inception in the current form (Rao *et al.*, 2014), originally termed APA ('Aggregate Peak Analysis'), pile-up analysis has been used both to analyse single-cell Hi-C data (Flyamer *et al.*, 2017; Gassler *et al.*, 2017; Nagano *et al.*, 2017) and as a general way of quantifying feature strength (Abdennur *et al.*, 2018; Bonev *et al.*, 2017; Díaz *et al.*, 2018; Fudenberg *et al.*, 2016; Hsieh *et al.*, 2019; Krietenstein *et al.*, 2019; Kruse *et al.*, 2019; McLaughlin *et al.*, 2019; Nora *et al.*, 2017; Rao *et al.*, 2017; Rowley *et al.*, 2019; Schwarzer *et al.*, 2017). A visual interactive tool to semi-manually classify and pile-up predefined regions has also been developed (Lekschas *et al.*, 2018). However, 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 (Table 1). At the same time, performing detailed analysis of Hi-C data remains difficult for non-specialists due to the absence of easy to use tools.

Here, we present a unified command-line interface tool written in **P**ython to **p**ile-**up** Hi-C data stored in the widely used and versatile **.cool** format (Abdennur and Mirny, 2019) (*coolpup.py*). A simple script for plotting the output of *coolpup.py* is provided in the package (*plotpup.py*), although for higher flexibility, we suggest directly using *matplotlib* or another library. We have extensively applied this tool to investigate the role of Polycomb group proteins in 3D genome organization of mouse ES cells (Boyle *et al.*, 2019; McLaughlin *et al.*, 2019).

Here, we have applied *coolpup.py* to published data to investigate the effect of different normalization strategies on the resulting pile-ups, and to replicate published results to verify *coolpup.py*'s algorithm. We also present a novel variation of the pile-up 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. Using published single-cell Hi-C data, we also investigate the dynamics of polycomb-associated looping revealing a different dynamics of looping across the cell cycle compared with CTCF loops.

2 Materials and methods

2.1 Sources of datasets and data analysis

As a proof of principal, we applied *coolpup.py* to publicly available Hi-C data (Bonev *et al.*, 2017; Nora *et al.*, 2017) using *distiller* (https://github.com/mirnylab/distiller-nf) to obtain *.cool* files filtered with a map quality (mapq) of $\geq$30. We used these data at 5 kb 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) (Gassler *et al.*, 2017; Rao *et al.*, 2014), were re-analysed at 10 kb resolution (without balancing and with coverage normalization and 10 random shifts). We also used single-cell Hi-C data for mouse ES cells grown in serum from Nagano *et al.* (2017) (*.cool* files were kindly shared by Aleksandra Galitsyna) at 5 kb resolution. We created pile-ups for each cell in the same manner as for snHi-C. The pile-ups with the coefficient of variation of values in their 5×5 upper left and lower right corners equal to 0.5 or above were not used further as too noisy. We used the average value of interactions in the central 3×3 pixel square to get the level of interaction enrichment. RING1B and H3K27me3 ChIP-seq peaks (Illingworth *et al.*, 2015) were lifted over to the mm9 mouse genome assembly. The coordinates of biochemically defined CpG islands were taken from (Illingworth *et al.*, 2010). CTCF ChIP-seq peaks were taken from Bonev *et al.* (2017) and, following liftOver to the *mm9* assembly, intersected with CTCF motifs found in the mm9 genome using Biopython's *motifs* module (Cock *et al.*, 2009). A human CTCF position-frequency matrix was downloaded from JASPAR (MA0139.1). We used only motifs with a score >7 and discounted peaks containing >1 motif.

Regions of high insulation (meaning low number of contacts crossing this regions) in the Bonev *et al.* Hi-C data were called using *cooltools diamond-insulation* from 25 kb 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 1500 kb were excluded 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; Rao *et al.*, 2014). The same loop annotations for mouse ES cells were used as in our recent publication (McLaughlin *et al.*, 2019).

All figure panels were created using *matplotlib* (Hunter, 2007) and assembled in Inkscape.

Table 1. Comparison of four tools for pile-up analysis across a set of features: Juicer Aggregate Peak Analysis (APA) (Rao *et al.*, 2014), HiCExplorer (hicAggregateContacts and hicAverageRegions) (Ramírez *et al.*, 2018) and GENOVA (APA, ATA and PE-SCAn) (van der Weide, 2019) and *coolpup.py*

Feature	Juicer	HiCExplorer	GENOVA	*coolpup.py*
Aggregate loops	+	−	+	+
Aggregate region pairs	−	+	+	+
Interactions between two region sets	−	+	−	+
Local pile-ups	−	+	−	+
(Local) rescaled pile-ups	−	−	+	+
Distance normalization	−	Expected (and z-score)	Fixed shifts (for pairwise analysis)	Expected or random shifts
Coverage normalization	−	−	−	+
Anchored pile-ups/loop-ability	−	−	−	+
Command-line interface	+	+	−	+
Simple text output of pile-ups	+	+	−	+
Hi-C file format	.hic	.cool, .h5, other?	HiC-pro	.cool

2.2 *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 pair bed ("bedpe": chrom1, start1, end1, chrom2, start2, end2) file to define the regions under investigation. The tool is implemented as a *python* package 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* (van der Walt *et al.*, 2011), *scipy* (Virtanen *et al.*, 2020) and *pandas* (*McKinney, 2010*). A separate CLI tool included in the package (*plotpup.py*) can be used to visualize the results and uses *matplotlib* (Hunter, 2007). The code is available on GitHub (https://github.com/Phlya/coolpuppy) and the package can be installed using *pip*, which then makes *coolpup.py* and *plotpup.py* 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 the 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 region 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 the 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*. With the assumption that the probability of interactions only depends on distance, the whole-chromosome expected matrix is a diagonal-constant matrix A with diagonal values d (also known as a Toeplitz matrix), such as: $A_{i,j} = A_{i+1,j+1} = d_{i-j}$ The simplicity of this expected model allows trivial creation of a matrix containing expected values for an arbitrary region of the intra-chromosomal Hi-C map without generating the whole matrix to avoid high-memory requirements, which is done for each ROI. All expected matrices are averaged to generate a normalizing matrix. 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, to generate the normalizing matrix, the whole pile-up procedure is repeated, but the coordinates are randomly shifted. In the end, the resulting matrix of averaged ROIs is divided element-wise by the normalizing matrix to remove effects of distance.

If not specified, balanced data with chromosome-wide expected normalization were used when creating pile-ups, except for the zygote and single-cell Hi-C datasets, where randomly shifted controls and coverage normalization were used instead. For the single-cell Hi-C (Nagano *et al.*, 2017) analysis, we only used pairs of convergent CTCF peaks within 100–800 kb of each other, since previous analysis (data not shown) indicated this as the distance range where CTCF-associated loops are most prominent; to reduce noise, RING1B-associated interactions were analysed for all distances above 100 kbp.

2.3 Performance profiling

Coolpup.py performance was tested on the University of Edinburgh Open Grid Scheduler cluster (Eddie3) using the Hi-C datasets for mouse ES cells (Bonev *et al.*, 2017; Nora *et al.*, 2017). To generate the required large number of coordinates for testing, we used coordinates of the B3 repeat from the 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, one compute core was utilized. The same procedure was performed for HiCExplorer *hicAggregateContacts*, except *shuf* was used to generate a random sample of required size. The following arguments were also provided to mimic *coolpup.py* behaviour as close as possible: *–range 105000:1000000000000 –avgType mean –transform obs/exp*. All measurements were performed five times. Plotted in Figure 4 are actual measured runtime values, the line shows mean values and shaded area — ±95% confidence interval, using the *seaborn* plotting package (Waskom *et al.*, 2018).

3 Results

3.1 Different normalization strategies implemented in *coolpup.py*

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 (Yaffe and Tanay, 2011). Cooler implements a matrix balancing (visibility equalization) approach to remove all potential biases (Imakaev *et al.*, 2012) and, when available, it is recommended to use balanced data for pile-ups. Sometimes, for example, in single-cell Hi-C, removing biases is impossible due to the 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 normalizing the matrix by the coverage (i.e. the total number of contacts of the bins in the chromosome) of the averaged regions (Flyamer *et al.*, 2017). To illustrate this, we integrated ChIP-seq datasets with Hi-C and analysed CTCF and polycomb (RING1B)-associated loops, and all potential intersections between high RING1B peaks, in mouse ES cell Hi-C data (Bonev *et al.*, 2017). This approach visibly reduces coverage variability between bins and removes sharp crosses from the central bin that is present with unbalanced data. This normalization seems to slightly over-correct, 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, it is often desirable to remove the distance-dependency of contact probability in Hi-C data, since sometimes it can obscure interesting properties, such as enrichment in the centre of the pile-up. The general approach to perform this normalization 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. 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 (Flyamer *et al.*, 2017). 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 contact probability 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 in our experience for local pile-ups (especially with rescaling; see below) random controls perform better than simple normalization to expected values (data not shown). It is worth bearing in mind that this normalization can also hide real signal in the data, such as enrichment of interactions in the lower-left corner, observed in particular for CTCF-anchored loops (Fig. 1B).

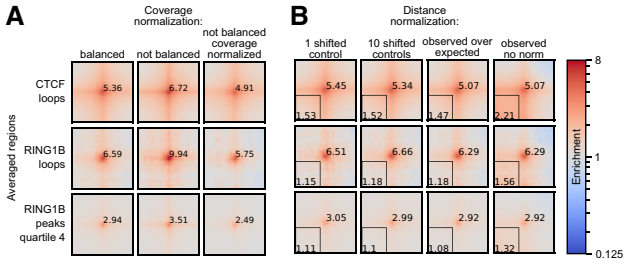

Fig. 1. Hi-C data normalization strategies. (**A**) Comparison of coverage normalization strategies for pile-up analyses using mouse ES cell Hi-C data (Bonev *et al.*, 2017). Normalization approaches are in columns: matrix balancing (iterative correction); no normalization; no balancing with coverage normalization of the pile-ups. The different averaged regions are shown in rows: loops associated with CTCF ($n = 6536$), loops associated with RING1B ($n = 104$) (see Materials and Methods section), all pairwise combinations of high RING1B peak regions from the fourth quartile (by RING1B ChIP-seq read count) ($n = 2660$ of peak regions). All pile-ups produced with 10 randomly shifted controls. All pile-ups are normalized to the average of the top-left and bottom-right corner pixels to bring them to same scale. Value of the central pixel is displayed. Five kilobytes 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; 10 randomly shifted control per ROI; normalization to chromosome-wide expected; no normalization. Same rows as in (A). Average enrichment of the lower-left corner of the pile-up is displayed

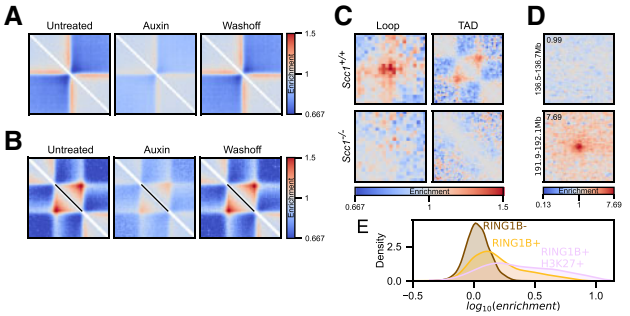

Fig. 2. Pile-up variations. (**A**) Local pile-ups of high-insulating regions in ES cells across untreated, auxin-treated and wash-off conditions in CTCF-AID Hi-C data (Nora *et al.*, 2017). Twenty-five kbp resolution data with 1000 kbp padding around the central pixel. (**B**) Local rescaled pile-ups of TADs (defined based on high-insulating regions) across same data as in (A) from 5 kbp resolution data. (**C**) Loop and rescaled TAD pile-ups for pooled single-cell Hi-C data showing loss of structures in *Scc1*$^{-/-}$ zygotes (Gassler *et al.*, 2017). (**D**) Two examples of anchored pile-ups from RING1B+/H3K27me3+ CpG islands on chr1, with no visible enrichment (top), or with very prominent enrichment (bottom). The anchored region is on the left side of the pile-up, and its coordinates (including the padding) are shown on the left. The value of the central pixel ('loopability') shown in top left corner. (**E**) Distribution of 'loopability' values of CpG islands not bound by RING1B, CpG islands bound by RING1B, and CpG islands bound by RING1B and also marked by H3K27me3

3.2 Applications of pile-ups

As well as the basic pile-up procedure, 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 (Nora *et al.*, 2017) (Fig. 2A). In this case, the pile-ups are performed in the same way as previous off-diagonal pile-ups; however, the regions that are averaged lie on the main diagonal of the Hi-C map. A variation of this approach is local pile-ups with rescaling to analyse features of different size, for example, TADs (Flyamer *et al.*, 2017). As an example, TADs, based on aforementioned regions of high insulation annotated in data from (ref. Bonev *et al.*, 2017), were averaged to visualize changes in local interaction strength upon CTCF degradation (Nora *et al.*, 2017) (Fig. 2B). Here, all windows centred on regions of interest are rescaled to the same size, and then averaged.

Pile-ups 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 sparse 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 (Gassler *et al.*, 2017). Since the material is so limiting and data are based on single cells, the total number of contacts in this dataset is very low: 4.8 and 9.2 million contacts in *Scc1*$^{+/+}$ and *Scc1*$^{-/-}$, respectively. However, we successfully performed pile-ups, both with 'traditional' averaging of loops, and local pile-ups of TADs with rescaling, and observe the loss of both loops and TADs upon deletion of cohesin, comparable to the original study (Fig. 2C).

All pile-up approaches include averaging of multiple regions, a drawback of which is the 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 pile-ups 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 Fig. 2D). A simple proof of principle analysis highlights the interactions between sites bound by polycomb group proteins in mouse ES cells (data from Bonev *et al.*, 2017). By splitting the CpG islands (data from Illingworth *et al.*, 2010) — the main targets of polycomb binding in ES cells — in the mouse genome into RING1B (a core component of Polycomb Repressive Complex 1 — PRC1) negative, RING1B positive, and RING1B and H3K27me3 positive sets (data from Illingworth *et al.*, 2015), we observe high 'loop-ability' values for the two latter groups, while the RING1B negative CpG islands have close to no enrichment (Fig. 2E). We perform more detailed analysis of such loop-ability measurements in our recent report (Boyle *et al.*, 2019).

Pile-ups are an invaluable tool when analysing Hi-C data from single cells, since averaging features across the whole genome helps to circumvent the sparsity of the data. Here, we apply *coolpup.py* to analyse the looping interactions across the cell cycle using a published single-cell Hi-C dataset from hundreds of mouse ES cells (Nagano *et al.*, 2017). We compared the enrichment of interactions in different cell cycles stages for CTCF- and RING1B-associated interactions (see Fig. 3A and B). For convergent CTCF sites, we detected the loss of loop strength in pre- and post-mitotic cells until mid-G1 in the next cell cycle , consistent with the original publication (Nagano *et al.*, 2017).

In contrast, the interactions between RING1B-binding sites have a very different dynamic across the cell cycle. They are at their weakest during S phase, strengthen during G2 and do not reach their peak until early G1. This is consistent with the cell cycle kinetics of H3K27me3 abundance at polycomb marked sites with H3K27me3 levels lowest during S phase where they are diluted after the replication fork, and levels of H3K27me3 only accumulating slowly through G2 and not peaking again until G1 of the next cell cycle (Reverón-Gómez *et al.*, 2018).

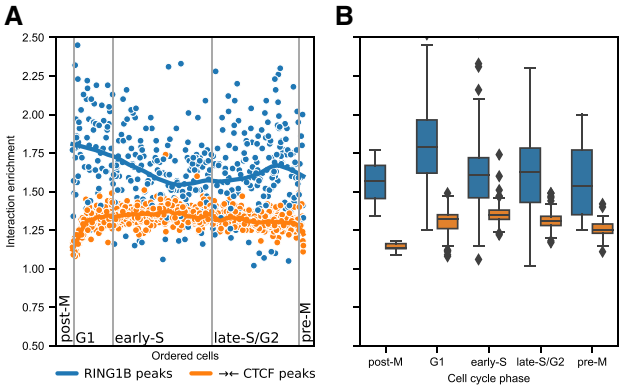

Fig. 3. Chromatin looping dynamics across cell cycle. (**A**) Hi-C interaction enrichment levels for single cells ordered along the cell cycle (Nagano *et al.*, 2017) for CTCF- and RING1B-associated interactions. The former is limited to 100–800 kb distance, while the latter is shown for all distances above 100 kb. Curves represent LOWESS-smoothed data for easier interpretation. (**B**) Distribution of enrichment values in all cell cycle stages from data in (A)

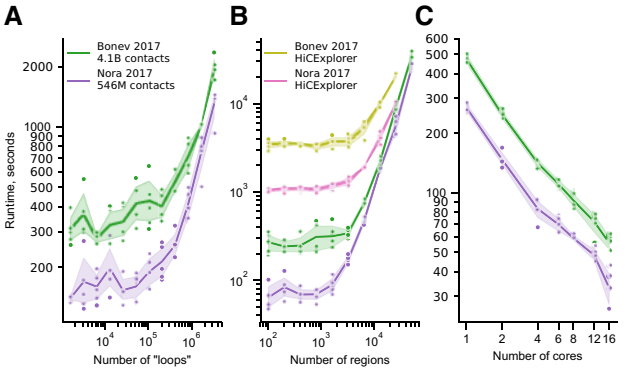

Fig. 4. Performance profiling. (**A**) Runtime (seconds) of *coolpup.py* with varying number of averaged 'loops' for two Hi-C datasets with different depth. (**B**) Same as (A), but for number of linear regions between which interactions are averaged. Also shown is runtime for HiCExplorer *hicAggregateContacts*. Note that the longest time-point for HiCExplorer required over 512 Gb RAM and was not computed. (C) Runtime of the same analysis with 5000 linear regions and a varying number of cores. Same colour coding as in (A)

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

Creating pile-ups from intersections of genomic regions can require averaging a huge number of 2D windows: the number of two 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 for the whole genome, several orders of magnitude more than the number of regions usually averaged, such as number of annotated loops (∼10 000). Therefore, it is important for a general-purpose tool for creating pile-ups to 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 this necessitates that 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) (Fig. 4A), and varying the number of one-sided coordinate interactions being averaged (Fig. 4B). We used both deep (Bonev *et al.*, 2017), and 'regular depth' Hi-C data (Nora *et al.*, 2017) from mouse ES cells. With both datasets, the runtime was almost constant (probably due to file system read speed limitations) up to a certain number of 'loops' ($\sim 1–2 \times 10^5$), where it starts quickly increasing (Fig. 4A). Notably, the best annotations that exist to date only contain <40 000 loops (Krietenstein *et al.*, 2019), and therefore this would fall within the flat part of the curve. Similarly, in the latter analysis, runtime did not increase up to 1600 and 3200 regions of interest for the Nora *et al.* and Bonev *et al.* datasets, respectively. Importantly, in both analyses the difference in time between datasets with almost 10-fold sequencing depth difference is not very large, and probably mostly driven by differences in time required to read the data from disk. When similar analysis was performed using HiCExplorer *hicAggregateContacts*, the runtime was >10-fold longer for each dataset with low numbers of regions (Fig. 4B), and the analysis required much more memory since the algorithm uses dense data structures and stores each submatrix of interest in memory (required >100 Gb for the Bonev *et al.* dataset; the longest time-point required over 512 Gb of RAM and was not computed, while *coolpup.py* only needed ∼8 Gb for any calculation). HiCExplorer implementation computes observed/expected matrix for every calculation and cannot use pre-computed expected values, which at least partially accounts for much longer runtimes.

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 (Fig. 4C) and showed that the runtime shortened linearly 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, although this would also significantly increase memory requirements.

4 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 to 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 straightforward with *coolpup.py*. Using our tool, we reproduced published results on the role of CTCF and cohesin in generating chromatin loops and TADs. We have shown application of *coolpup.py* to both low coverage Hi-C data (merged snHi-C data), and extremely sparse single-cell Hi-C data. The latter analysis not only replicated published data on CTCF-mediated looping changes across the cell cycles, but also revealed novel cell cycle dynamics of polycomb-associated interactions with highest contact enrichment around the time of mitosis. We note that these observations are generally consistent with the dilution and slow recovery of the H3K27me3 mark after the replication fork (Alabert *et al.*, 2015; Reverón-Gómez *et al.*, 2018), as well as an antagonistic relationship between cohesin-mediated loop extrusion and looping between RING1B target sites, reported previously (Rhodes *et al.*, 2020). These observations also pose a question of whether polycomb-associated interactions persist in metaphase chromosomes—a possibility since components of CBX2-containing PRC1 remain associated with metaphase chromosomes (Zhen *et al.*, 2014). These novel insights highlight the exploratory power of pile-up analysis.

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 standardized 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 pile-ups.

Acknowledgements

We are grateful to Maksim Imakaev, Nezar Abdennur, Anton Goloborodko, Hugo Brandão and Sergey Venev for discussions, help and advice about the *cooler* ecosystem, and to Aleksandra Galitsyna for sharing *.cool* files for the Nagano *et al.* (2017) dataset. This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF) (http://www.ecdf.ed.ac.uk/). 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]. R.S.I. is supported an MRC Career Development Award (MR/S007644/1) and 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: none declared.

References

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

Abdennur,N. *et al.* (2018) Condensin II inactivation in interphase does not affect chromatin folding or gene expression. *bioRxiv*, doi: 10.1101/437459.

Alabert,C. *et al.* (2015) Two distinct modes for propagation of histone PTMs across the cell cycle. *Genes Dev.*, **29**, 585–590.

Barutcu,A.R. *et al.* (2016) C-ing the genome: a compendium of chromosome conformation capture methods to study higher-order chromatin organization. *J. Cell. Physiol.*, **231**, 31–35.

Bonev,B. *et al.* (2017) Multiscale 3D genome rewiring during mouse neural development. *Cell*, **171**, 557–572.e24.

Boyle,S. *et al.* (2019) A central role for canonical PRC1 in shaping the 3D nuclear landscape. *bioRxiv*, doi: 10.1101/2019.12.15.876771.

Canela,A. *et al.* (2017) Genome organization drives chromosome fragility. *Cell*, **170**, 507–521.e18.

Canela,A. *et al.* (2019) Topoisomerase II-induced chromosome breakage and translocation is determined by chromosome architecture and transcriptional activity. *Mol. Cell*, **75**, 252–266.e8.

Cock,P.J.A. *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.

de Wit,E. *et al.* (2013) The pluripotent genome in three dimensions is shaped around pluripotency factors. *Nature*, **501**, 227–231.

Dekker,J. (2002) Capturing chromosome conformation. *Science*, **295**, 1306–1311.

Díaz,N. *et al.* (2018) Chromatin conformation analysis of primary patient tissue using a low input Hi-C method. *Nat. Commun.*, **9**, 4938.

Flyamer,I.M. *et al.* (2017) Single-nucleus Hi-C reveals unique chromatin reorganization at oocyte-to-zygote transition. *Nature*, **544**, 110–114.

Franke,M. *et al.* (2016) Formation of new chromatin domains determines pathogenicity of genomic duplications. *Nature*, **538**, 265–269.

Fudenberg,G. *et al.* (2016) Formation of chromosomal domains by loop extrusion. *Cell Rep.*, **15**, 2038–2049.

Gassler,J. *et al.* (2017) A mechanism of cohesin-dependent loop extrusion organizes zygotic genome architecture. *EMBO J.*, **36**, 3600–3618.

Hsieh,T.-H.S. *et al.* (2019) Resolving the 3D landscape of transcription-linked mammalian chromatin folding. *bioRxiv*, doi: 10.1101/638775.

Hunter,J.D. (2007) Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.*, **9**, 90–95.

Illingworth,R.S. *et al.* (2010) Orphan CpG islands identify numerous conserved promoters in the mammalian genome. *PLOS Genet.*, **6**, e1001134.

Illingworth,R.S. *et al.* (2015) The E3 ubiquitin ligase activity of RING1B is not essential for early mouse development. *Genes Dev.*, **29**, 1897–1902.

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

Joshi,O. *et al.* (2015) Dynamic reorganization of extremely long-range promoter-promoter interactions between two states of pluripotency. *Cell Stem Cell*, **17**, 748–757.

Krietenstein,N. *et al.* (2019) Ultrastructural details of mammalian chromosome architecture. *bioRxiv*, doi: 10.1101/639922.

Kruse,K. *et al.* (2019) Transposable elements drive reorganisation of 3D chromatin during early embryogenesis. *bioRxiv*, 523712.

Lajoie,B.R. *et al.* (2015) The Hitchhiker's guide to Hi-C analysis: practical guidelines. *Methods*, 72, 65–75.

Lekschas,F. *et al.* (2018) HiPiler: visual exploration of large genome interaction matrices with interactive small multiples. *IEEE Trans. Visual. Comput. Graphics*, **24**, 522–531.

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

Lupiáñez,D.G. *et al.* (2015) Disruptions of topological chromatin domains cause pathogenic rewiring of gene-enhancer interactions. *Cell*, **161**, 1012–1025.

McKinney,W. (2010) *Data Structures for Statistical Computing in Python*, In *Proc. of the 9th Python in Science Conf.*, 51–56.

McLaughlin,K. *et al.* (2019) DNA methylation directs polycomb-dependent 3D genome re-organization in naive pluripotency. *Cell Rep.*, **29**, 1974–1985.e6.

Waskom,M. *et al.* (2018) mwaskom/seaborn: v0.9.0 (July 2018) Zenodo, doi: 10.5281/zenodo.1313201.

Nagano,T. *et al.* (2013) Single-cell Hi-C reveals cell-to-cell variability in chromosome structure. *Nature*, **502**, 59–64.

Nagano,T. *et al.* (2017) Cell-cycle dynamics of chromosomal organization at single-cell resolution. *Nature*, **547**, 61–67.

Nora,E.P. *et al.* (2017) Targeted degradation of CTCF decouples local insulation of chromosome domains from genomic compartmentalization. *Cell*, **169**, 930–944.e22.

Ramírez,F. *et al.* (2018) High-resolution TADs reveal DNA sequences underlying genome organization in flies. *Nat. Commun.*, **9**, 189.

Rao,S. *et al.* (2017) Cohesin loss eliminates all loop domains. *Cell*, 171, 305–320.e24.

Rao,S.S.P. *et al.* (2014) A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell*, **159**, 1665–1680.

Reverón-Gómez,N. *et al.* (2018) Accurate recycling of parental histones reproduces the histone modification landscape during DNA replication. *Mol. Cell*, **72**, 239–249.e5.

Rhodes,J.D.P. *et al.* (2020) Cohesin disrupts polycomb-dependent chromosome interactions in embryonic stem cells. *Cell Reports*, 30, 820–835.e10.

Rowley,M.J. *et al.* (2019) Condensin II counteracts cohesin and RNA polymerase II in the establishment of 3D chromatin organization. *Cell Rep.*, **26**, 2890–2903.e3.

Sanborn,A.L. *et al.* (2015) Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes. *Proc. Natl. Acad. Sci. USA*, **112**, E6456–E6465.

Schwarzer,W. *et al.* (2017) Two independent modes of chromatin organization revealed by cohesin removal. *Nature*, **551**, 51–56.

Stevens,T.J. *et al.* (2017) 3D structures of individual mammalian genomes studied by single-cell Hi-C. *Nature*, **544**, 59–64.

Tan,L. *et al.* (2018) Three-dimensional genome structures of single diploid human cells. *Science*, **361**, 924–928.

Ulianov,S.V. *et al.* (2017) Single-cell Hi-C bridges microscopy and genome-wide sequencing approaches to study 3D chromatin organization. *BioEssays*, **39**, 1700104.

van der Walt,S. *et al.* (2011) The NumPy array: a structure for efficient numerical computation. *Comput. Sci. Eng.*, **13**, 22–30.

van der Weide,R. (2019) GENome Organisation Visual Analytics. GitHub, https://github.com/robinweide/GENOVA.

Virtanen,P. *et al.* (2020) SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods*.

Williamson,I. *et al.* (2019) Developmentally regulated Shh expression is robust to TAD perturbations. *Development*, **146**, dev179523.

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.

Zhen,C.Y. *et al.* (2014) Cbx2 stably associates with mitotic chromosomes via a PRC2- or PRC1-independent mechanism and is needed for recruiting PRC1 complex to mitotic chromosomes. *Mol. Biol. Cell*, **25**, 3726–3739.