GigaScience

Loop detection using Hi-C data with HiCExplorer

--Manuscript Draft--

Manuscript Number:	GIGA-D-21-00069R2
Full Title:	Loop detection using Hi-C data with HiCExplorer
Article Type:	Technical Note
Funding Information:	Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (031 A538A de.NBI-RBC) — Prof. Dr. Rolf Backofen Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (031 L0101C de.NBI-epi) — Dr. Björn Grüning Deutsche Forschungsgemeinschaft (CIBSS - EXC-2189 - Project ID 390939984) — Prof. Dr. Rolf Backofen
Abstract:	Background: Chromatin loops are an essential factor in the structural organization of the genome; however, their detection in Hi-C interaction matrices is a challenging and compute-intensive task. The approach presented here, integrated into the HiCExplorer software, shows a chromatin loop detection algorithm that applies a strict candidate selection based on continuous negative binomial distributions and performs a Wilcoxon rank-sum test to detect enriched Hi-C interactions. Results: HiCExplorer's loop detection has a high detection rate and accuracy. It is the fastest available CPU implementation and utilizes all threads offered by modern multi-core platforms. Conclusions: HiCExplorer's method to detect loops by using a continuous negative binomial function combined with the donut approach from HiCCUPS leads to reliable and fast computation of loops. All the loop-calling algorithms investigated provide differing results, which intersect by ~ 50% at most. The tested in-situ Hi-C data contains a large amount of noise; achieving better agreement between loop calling algorithms will require cleaner Hi-C data and therefore future improvements to the experimental methods which generate the data.
Corresponding Author:	Joachim Wolff Albert-Ludwigs-Universität Freiburg: Albert-Ludwigs-Universitat Freiburg Freiburg im Breisgau, Baden-Württemberg GERMANY
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Albert-Ludwigs-Universität Freiburg: Albert-Ludwigs-Universitat Freiburg
Corresponding Author's Secondary Institution:	
First Author:	Joachim Wolff
First Author Secondary Information:	
Order of Authors:	Joachim Wolff Rolf Backofen, Dr. Björn Grüning, Dr.
Order of Authors Secondary Information:	
Response to Reviewers:	Suggestions by Editor after consulting external reviewer

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

- The authors should add to their last figure a comparison of the HiCExplorer and HICUPS to the Peakachu run on the ICE corrected matrices stating it explicitly in the figure legend.

Anwer
We did this.

- Please include more comparisons showing what HiCExplorer calls and what is missed by other tools. Add at least one more locus-by-locus snapshot of loops called by the tools, including Fit-Hi-C method absent in Figure 4. It would be important to better illustrate the Venn Diagram results from figure 2.

Anwer
We did this. Please consider the additional new figures in the Supplementary material.

- Another potentially important figure would be to include calls before merging, displaying several examples of enriched pixels in all the three methods (Fit-Hi-C, HiCExplorer and HiCCUPS). This could help resolving the issue raised by Wolff et al and pertaining to the method used to merge significant pixels into a single loop. Perhaps this should make up an entire small chapter of the paper (depending on the data).
Anwer
We don't think intermediate data is important in this comparison. The issue that we raised with the merging of loops is a different one: We think that all tools should be compared based on their detected loops and this implies that the merging of loops is applied by individual tools. What we raised and what we are concerned about is a sentence in the publication of Peakachu stating they used the intermediate, unmerged loop candidates and applied for all their self-developed merging tool. This is biasing the results in a positive way towards Peakachu. However, this is an issue of the Peakachu publication and does not influence our comparison.

Reviewer #1

Reviewer #1: My main concern for the revised manuscript is the additional benchmarking the authors performed with Fit-Hi-C and Peackachu. Since Fit-Hi-C is one of the first algorithms for Hi-C loop prediction (published in 2014) and Peakachu is the only method that uses the supervised machine learning approach for such purpose, I suggested that these two software should be recognized. If the authors can perform a fair benchmarking and find out where the differences come from, the results would be really interesting.

The authors decided to test the aforementioned methods during the revision. Unfortunately, I believe there were some errors during the testing.

For Peakachu:

1. Most importantly, the authors used the wrong form of normalized Hi-C files for Peakachu. Peakachu model was trained and should be used with ICE-normalized Hi-C matrix. However, based on page 8 in the supplementary file, the input file is gm12878_KR.cool. The data range for ICE and KR normalization is very different, and therefore, the model trained in ICE file will not work with KR format and the prediction will wrong. Therefore, all the following evaluations and descriptions for the Peakachu prediction are not accurate and needs to be revised (such as Fig. 4, Table S1 ...).

Answer
If your tool is that sensible then one of the following should be considered:
Publish the information that you need an ICE corrected matrix ANYWHERE! It is not published on your GitHub repository where you provide the models, it is not written in the paper, nor in the supplementary material! How should someone know this?
ICE and KR are not that different, I suspect a bit that you trained on not upscaled

matrices. Can it be that you trained on a matrix with a value range of 0 - 1, as it is after the correction of ICE and KR if the correction tool does not scale the values back to the correct value range? If so, please make this information public, because it is quite common to work on scaled-back interaction data.
Even this is all not true: An observed/expected normalization should be able to catch most of the differences of ICE / KR correction. In the end, we just search for enriched interactions, the correction does not matter that much. See the comment of the external reviewer concerning this!
If the above still is wrong: Then I am sorry to say this, but I assume your model has a heavy overfitting issue.
I will not revise the results based on the KR corrected matrix. It seems to me that either you missed important normalization (e.g. obs/exp normalization.) in the pre-processing or your model has heavy overfitting. Concerning obs/exp normalization: we checked your publication and supplementary material, we did not find the term 'observed/expected matrix' or something similar anywhere. We, therefore, assume you don't apply this normalization.

Reviewer #1
2. In the response letter, there is another misunderstanding about merging. Because Fit-Hi-C predicted too many contacts, the authors of Peakachu merged "the top 140,000 interactions into 14,876 loops (Fig. 3a, b), with the same pooling algorithm used by Peakachu." The reason is that if multiple continuous bins on a Hi-C map are all predicted as loops, the merging/filtering step will use the bin with the most significant P-value as the chromatin loops (local minimal). As the authors noted, Fit-Hi-C by default will generate "significant contacts in the 100,000-ends." Therefore, this merging/filtering step is necessary if we want to compare the loops predicted by each method. This is also what the author did in this manuscript as well - I am quoting their own writing here, "This filtering step is necessary to address the candidate peak value as a singular outlier within the neighborhood." Therefore, I do not understand the authors are "irritated" by such approach.

Answer
Yes, we merged the significant detected candidates too. However, we used for all approaches the merging tool provided by the individual tool and not for all tools our own merging tool. This is a major difference! To use one merging tool, their own developed one (!!!), is the approach of Peakachu paper and this is what we criticize! We do not criticize the merging in general. Fit-HiC2 provides its own merging tool, so does HiCCUPS. Why have you not used it? That would have been a fair comparison!

Reviewer #1
3. The authors of Peakach have released their prediction in 56 Hi-C datasets on their 3D Genome Browser website (http://3dgenome.fsm.northwestern.edu/publications.html), including the ones used in this manuscript. The authors used models trained at different sequencing depths for different datasets. Therefore, I would suggest the authors use this dataset for a fair evaluation.
Answer
We included this data. The very high agreement of loops with positions of HiChIP based H3K27ac and SMC1 confirms our suspicion of heavy overfitting. Your model does not learn the pattern of loops but the pattern of locations where these two proteins are present. In contrast to this, your performance considering CTCF with 50% is low, compared to HiCExplorer or HiCCUPS with 64% and 61%. Also the loci specific investigation shows that you miss certain clear loops like on chromosome 4 20.55 - 22.55 MB (Supplementary Figure 4) but you detect many loops which are not clearly visible and would need further orthogonal data to be confirmed, for example, the loops in the loci chromosome 1 15.00 - 18.00 MB (Supplementary Figure 6). We suggest you update your published models based on training where the Hi-C interaction matrices are obs / exp normalized and have different corrections as their base. Please take care that important and easily visible loops on chromosome 4 20.55 - 22.55 MB are identified.

Reviewer #1
Regarding Fit-Hi-C, what are the number of peaks the before and after filtering? The

author also needs to provide the loop locations so that reviewers can evaluate their claim independently. This information is critical. This manuscript might be helpful for the authors to evaluate Fit-Hi-C (Arya Kaul et al. Nature Protocol 2020).

Finally, the authors need to provide all the predicted chromatin loops in the cell lines as well as loops predicted by other software used in this manuscript as supplementary materials (loops in Supplementary Table 1).
Answer
We published the data on zenodo.

Reviewer 2

The authors have addressed some of my comments, but the majority of my comments have not been
satisfactorily addressed. The manuscript would still require major modification before publication.

Reviewer #2:
I am, of course, fully aware that a figure and its caption belong together, and my concern
referred to the figure and its caption text as a whole. However, the graphics, even in their
updated version, do not facilitate the readers' understanding of the method. Furthermore, it is
customary in scientific writing that the use of arrows or other markers in the graphics is accompanied by an explanation in the caption text (i.e. what does each arrow point to?). Also,
instead of the use of arrows, it would be much more informative to demarcate the borders of
those areas in the matrices that the authors would like to highlight.
Perhaps, a pragmatic solution could be to delete the confusing graphics altogether and just
use the caption text on its own as bullet points or numbered steps for the method.
Answer
We don't think we come here to any agreement with the reviewer. We dropped the graphic without any replacement. The algorithm itself is already described in the text and the graphical description was to have additional material to understand it maybe better. We reject the suggestion to use bullet points because it would be even more redundant.

Reviewer #2:

As pointed out, some of the methods search for loops genome-wide or only within 8Mb windows, while others use a custom distance for the loop search space. Currently the results show loops detected genome-wide for Juicer GPU, within 8Mb for Juicer CPU, genome-wide for HOMER and within 8Mb and 2Mb for the other methods. Comparison of
the methods should be restricted to those loops that are shorter than 8Mb/2Mb for all methods and the genome-wide results could be added in supplementary.
Answer:
We restrict all to 8 MB.

Reviewer #2:

Thank you for extending the section. However, the explanation about the chimeric reads is
incorrect. Read-pairs are indeed always come from ligation fragments, but chimeric reads is
a term used for those reads (single end) which overlap with the ligation site, and therefore
cannot be mapped to the genome without trimming or splitting.

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

Answer:
We added a sentence.

Reviewer #2:

I am not convinced that a passing introduction of a few methods that are not designed to
detect loops in Hi-C data will help the inexperienced reader to understand why they are not
appropriate. By the authors reasoning, several other methods should also be introduced e.g.
FIREcaller, SIP, Mustache, HiC-DC, HIPPIE.
Answer:
We removed the methods that are criticized in the introduction.

Reviewer #2:

As the authors themselves stated above in their reply to my question about the motivation
of developing the algorithm. They choose the continuous negative binomial distribution because the negative binomial allows for overdispersion of the data and the continuous version of it can be applied to normalized read counts.
What this comment referred to, was the following sentence: “In genome analysis, good experience has been made with negative binomial functions as proposed, for example, by
DESeq2.” This does not describe the overdispersion issue which is the main reason for using
a negative binomial based algorithm. In this case a continuous negative binomial.

Answer:

We never claimed that DESeq2 describes the overdispersion issue. The sentence was written why we consider a cNB distribution at all and to motivate that it has been successfully used in other areas of genome analysis. Anyhow, we removed the sentence to satisfy the reviewer.

Reviewer #2:

In this case, I would suggest comparing the tools for loops detected within 2Mb and 8Mb
and show the full genome results only in the supplementary. See my comment above about
the consistency of the comparison.

Answer:
We restrict the analysis results to 8Mb and dropped all other results.

Reviewer #1:
Fig 2A is showing the same data as Fig 2A in the Galaxy HiCExplorer publication (Wolff et al 2020), but the
detected loops indicated are different. What is the reason for that?
Answer:
The algorithm used in the Galaxy HiCExplorer 3 publication was based on HiCExplorer 3.2; with HiCExplorer 3.5
we changed the loop detection algorithm to its current form. For this reason, the detect loops differ. We changed
the algorithm because we were not happy with the performance in terms of accuracy of the detect loops and also
on the utilization of the threading of modern CPUs. For comparison of the algorithmic differences, please

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

	compare the manuscript to the bioRxiv publication of the loop detection. The authors should highlight these changes in current manuscript. Answer We added two sentences. Reviewer #2: The authors state that some methods are better correlated with CTCF binding sites than others based on the proportion of CTCF-bound loops. I did notice that the proportion was calculated for each method. However, the difference between these proportions would have needed to be statistically tested (with two-proportions Z-test) to claim differences in the methods' performance. Answer We added the requested z-test for the GM12878 data and the proportions of CTCF ChIA-PET, RAD21 ChIA-PET, H3K27ac HiChIP and SMC1 HiChIP. Given a p-value threshold of 0.05 all results are below this threshold for a H0 'the proportions are the same'. Given a p-value threshold of 0.001 only the differences in proportion between HiCExplorer and HiCCUPS for the CTCF ChIA-PET with 0.00216 and HiCExplorer and Homer for the SMC1 HiChIP data with 0.00299 are not significant.
Additional Information:	
Question	**Response**
Are you submitting this manuscript to a special series or article collection?	No
Experimental design and statistics Full details of the experimental design and statistical methods used should be given in the Methods section, as detailed in our Minimum Standards Reporting Checklist. Information essential to interpreting the data presented should be made available in the figure legends. Have you included all the information requested in your manuscript?	Yes
Resources A description of all resources used, including antibodies, cell lines, animals and software tools, with enough information to allow them to be uniquely identified, should be included in the Methods section. Authors are strongly encouraged to cite Research Resource Identifiers (RRIDs) for antibodies, model organisms and tools, where possible.	Yes

Have you included the information requested as detailed in our Minimum Standards Reporting Checklist?	
Availability of data and materials All datasets and code on which the conclusions of the paper rely must be either included in your submission or deposited in publicly available repositories (where available and ethically appropriate), referencing such data using a unique identifier in the references and in the “Availability of Data and Materials” section of your manuscript. Have you have met the above requirement as detailed in our Minimum Standards Reporting Checklist?	Yes

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

Click here to access/download;Manuscript;manuscript_loop.pdf

GigaScience, XXXX, 1–8

doi: xx.xxxx/xxxx
Manuscript in Preparation
Technical Note

TECHNICAL NOTE

Loop detection using Hi-C data with HiCExplorer

Joachim Wolff[1,*], Rolf Backofen[1,2] and Björn Grüning[1]

[1]Bioinformatics Group, Department of Computer Science, University of Freiburg, Georges-Köhler-Allee 106, 79110 Freiburg, Germany and [2]Signalling Research Centres CIBSS, University of Freiburg, Schänzlestr. 18, 79104 Freiburg, Germany

[*]wolffj@informatik.uni-freiburg.de

Abstract

Background: Chromatin loops are an essential factor in the structural organization of the genome; however, their detection in Hi-C interaction matrices is a challenging and compute-intensive task. The approach presented here, integrated into the HiCExplorer software, shows a chromatin loop detection algorithm that applies a strict candidate selection based on continuous negative binomial distributions and performs a Wilcoxon rank-sum test to detect enriched Hi-C interactions. **Results:** HiCExplorer's loop detection has a high detection rate and accuracy. It is the fastest available CPU implementation and utilizes all threads offered by modern multi-core platforms. **Conclusions:** HiCExplorer's method to detect loops by using a continuous negative binomial function combined with the donut approach from HiCCUPS leads to reliable and fast computation of loops. All the loop-calling algorithms investigated provide differing results, which intersect by ~ 50% at most. The tested in-situ Hi-C data contains a large amount of noise; achieving better agreement between loop calling algorithms will require cleaner Hi-C data and therefore future improvements to the experimental methods which generate the data.

Key words: Hi-C, Hi-C loop detection, DNA loops

Introduction

Many algorithms are currently available for loop detection in Hi-C data. HiCCUPS uses a *donut algorithm*, which considers all elements of a Hi-C interaction matrix as peaks and tests if the region around them is significantly different from the neighboring interactions. HiCCUPS is part of the software Juicer[1], and the implementation requires a general-purpose GPU (GPGPU), which imposes a barrier for users without access to Nvidia GPUs. However, an experimental CPU-based implementation has also been released. Algorithms such as iterative correction and eigenvector decomposition (ICE) [1], or Knight-Ruiz (KR) [2] are widely used in Hi-C data analysis for balancing Hi-C matrices, but the loop detection algorithm of HiCCUPS uses a different approach. HiCCUPS employs a Poisson model, which is a distribution for discrete data, to detect regions of interest. After balancing a Hi-C interaction matrix, the data is no longer discrete, but continuous. In order to work with the Poisson distribution, the balancing of the values is reverted. This procedure is methodologically questionable, as it involves manipulation of the data to fit the requirements of a particular distribution, rather than fitting on the distribution which is most probable or suitable. Moreover, the Poisson distribution on the raw Hi-C data tends to have an overdispersion, which suggests Poisson is not the best choice. HOMER [3] creates a relative contact matrix per chromosome and scans these for locally dense regions. HOMER does not support standard file formats for Hi-C matrices like *cool* [4], which forces the user to create all data from scratch, a time-consuming process and a potential source of errors and inaccuracies. Chromosight [5] detects loops based on a pattern-matching algorithm. Cooltools[2] uses a reimplementation of the HiCCUPS algorithm; Fit-Hi-C [6] detects significant Hi-C contacts and provides a merging algorithm to detect DNA loops. Peakachu

1 https://github.com/aidenlab/juicer

2 https://github.com/open2c/cooltools

Compiled on: November 5, 2021.
Draft manuscript prepared by the author.

[7] uses a random forest approach trained on CTCF or H3K27ac data. Chromosight, cooltools, Peakachu, and HiCExplorer support the *cooler* file format. HOMER, Fit-Hi-C, and Peakachu do not utilize parallelization techniques to improve runtime, running only on a single core.

Here we present an algorithm that can detect Hi-C loops. It is based on a continuous negative binomial distribution and is highly parallelized, assigning one thread per chromosome and parallelizing further using multiple threads within a chromosome. This approach makes full use of the resources available in the last generation of multi-core CPU platforms.

Methods

According to Rao [8], most of the anchor points of detected loops lie within a range of 2 Mb. This insight can be used to decrease the search space in a biologically meaningful way and also to reduce the computational burden, while at the same time maintaining a low memory footprint. Moreover, interaction pairs with genomic distances which are too close to each other, corresponding to points in the Hi-C matrix close to the main diagonal, already have high interaction counts. It is, in many cases, unlikely that these pairs contribute enrichments in the context of their neighborhood. The high interaction count can explain this observation between two loci; they are closer in one-dimensional space and close to the main diagonal. Specialized algorithms like FastHiC should be used to detect intra-TAD enrichments. A general problem for Hi-C interactions with few absolute counts is determining whether their interactions are true interactions or noise. These artifacts cannot be corrected by the commonly-used Hi-C interaction matrix correction algorithms such as iterative correction and eigenvector decomposition (ICE) [1], or Knight-Ruiz (KR) [2]. These algorithms perform a matrix balancing and correct for an uneven distribution of the interaction counts per genomic position. The correction algorithms are unable to distinguish and therefore filter true interactions from noise. All values below a given threshold are discarded, and noise is removed to account for these known problems in the Hi-C interaction data.

Algorithm

A strict candidate selection is critical to reducing the computational complexity of the loop detection algorithm. A maximum loop size can be defined to restrict the search space to take the previously-mentioned observation from Rao [8] into account. In Hi-C, the primary data structure is the symmetrical $n \times n$ interaction count matrix (ICM):

$$ICM = \begin{bmatrix} ic_{00} & \cdots & ic_{0n} \\ \vdots & \cdots & \vdots \\ ic_{n0} & \cdots & ic_{nn} \end{bmatrix} \quad (1)$$

The relative genomic distance is given by:

$$d = |i - j| \; for \; ic_{i,j} \quad (2)$$

where $ic_{i,j}$ is an element of Hi-C interaction matrix *ICM*.

As a first step, the interaction matrix *ICM* is transferred to an observed vs. expected matrix M^* to normalize the differing interaction heights per genomic distance. Each element m of M^* is defined as:

$$m^*_{i,j} = \frac{icm_{i,j}}{exp_d} \quad (3)$$

Different methods are offered to adjust differences in the samples introduced. Hi-C is, in comparison to techniques like RNA-seq, a two-dimensional approach; all reads are chimeric. The term chimeric in the context of Hi-C should be understood as reads which are ligated from two different locations in the genome. This is achieved by fixation of spatially close DNA fragments with formaldehyde, followed by digestion and ligation to create chimeric reads. These events should, in theory, happen uniformly in the whole genome; however, whether this is the case depends on the particular sample and genome studied. Therefore, three different ways to compute the expected value are offered. Note that the observed/expected matrix normalization step was not included in the initial version of this publication released on bioRxiv [9].

First, only non-zero contacts are considered:

$$exp\_nonzero_d = \frac{\sum ic_{i,j}}{|non - zero \; interactions \; d|} \quad (4)$$

Second, all contacts are considered:

$$exp\_with\_zero_d = \frac{\sum ic_{i,j}}{|all \; interactions \; d|} \quad (5)$$

And third, similar to HOMER's normalization, a correction for different occurring ligation events is offered:

$$exp\_ligation_d = exp\_nonzero_{i,j} * \frac{\sum(row_{ICM}(i)) * \sum(row_{ICM}(j))}{\sum(ICM)} \quad (6)$$

Candidate selection per genomic distance

To detect enriched Hi-C interactions, the observed/expected normalized Hi-C data is fitted per genomic distance d independently to a continuous negative binomial distribution. Supplementary Figure 1 shows the value density distribution of different genomic distances and provides evidence for the chosen distribution assumption. The negative binomial function, rather than the Poisson distribution, is used because the raw data of the genomic distances of chromosome 1 of GM12878 cell line at 10 kb indicate overdispersion [10] in a majority of the distances (80.1%); therefore, the negative binomial distribution with an additional free parameter is the better choice (Supplementary Figure 2).

$$X_d \sim cNB_d(r_d, p_d) \; \forall d = |i - j| \quad (7)$$

Gamma functions must replace the factorial in the binomial coefficient as used by edgeR [11, 12] to make the discrete negative binomial function continuous:

$$\binom{k + r - 1}{k} = \frac{(k + r - 1)!}{(k!) * (k + r - 1 - k)!} = \frac{(k + r - 1)!}{(k!) * (r - 1)!} \quad (8)$$

The gamma function is defined for any $n \in \mathbb{N}$:

$$\Gamma(n) = (n - 1)! \quad (9)$$

Moreover, the gamma function is defined for any $n \in \mathbb{R}_{>0}$:

$$\Gamma(n) = \int_0^\infty x^{n-1} * e^{-x} dx \quad (10)$$

With Equation (9), the binomial coefficient can be reformulated as:

$$\binom{k+r-1}{k} = \frac{\Gamma(k+r)}{\Gamma(k+1) * \Gamma(r)} \tag{11}$$

which leads to the probability mass function for a 'continuous negative binomial distribution' with $\forall k \in \mathbb{R}_{>0}$ and $\forall r \in \mathbb{R}_{>0}$:

$$f(k, r, p) = \frac{\Gamma(k+r)}{\Gamma(k+1) * \Gamma(r)} p^k (1-p)^r \tag{12}$$

The p-value of observing a specific observed vs. expected value at the genomic distance d is given by the continuous negative binomial cumulative density function:

$$pvalue\ of\ m^*_{i,j} = P(x \geq m^*_{i,j}) = \begin{cases} 1 - CDF_d(m^*_{i,j}) & \text{if } m^*_{i,j} > 0. \\ 1 & \text{if i = 0.} \end{cases} \tag{13}$$

Only the observed vs. expected values with p-values smaller than an individual threshold per genomic distance are accepted as candidates; these candidates are further filtered to remove candidates with too few absolute interactions. To reduce the amount of data to fit, the user can remove observed vs. expected values below a threshold before the continuous negative binomial function is fitted. Moreover, an option to remove candidates by their interaction height is also provided.

Loop peak detection

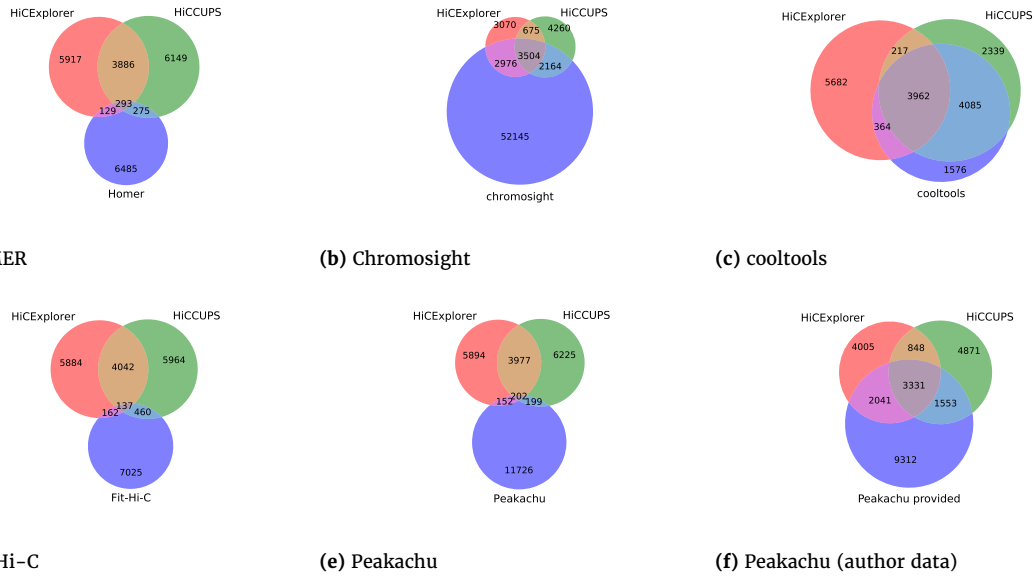
The entire neighborhood needs to be considered to detect enriched regions in a Hi-C interaction matrix. A neighborhood is a square of size n with the candidate element in its center. An enriched region needs to have an enriched interaction count in relation to the elements in its neighborhood. The neighborhood concept comes with a few issues: first, within a single neighborhood, there can be multiple candidate loops detected from different but adjacent genomic distances. Second, if a candidate is significant for its genomic distance, it is not necessarily an enriched value for its neighborhood. Third, a single enriched interaction in a neighborhood is possible, but is likely to be a false positive. Meaningful enriched interactions appear in groups and form a peak in the two-dimensional space. All candidates in one neighborhood are pooled together to handle the first issue, only the candidate with the highest observed vs. expected value for one neighborhood is considered a representative of its neighborhood; all others are removed. The neighborhood is split into a peak and a background region to cover the second and third issues by considering the square around the candidate as the peak region and the neighborhood's remaining elements as the background. The neighborhood is further divided into the vertical region left and right from the peak, the horizontal region above and below the peak, and the bottom left corner; this is a similar approach to HiCCUPS [8]. The peak and neighborhood square sizes are defined by their inradius values, *peakWidth* and *windowSize*. All candidates which fulfil the condition $mean(background) \geq mean(peak)$ are rejected as a loop. This filtering step is necessary to address the situation where a candidate peak value is a singular outlier within the neighborhood. Furthermore, the Wilcoxon rank-sum test is used, with the H0 hypothesis that the background and peak regions have the same distribution with significance level p. As background, the vertical and horizontal area mentioned above, and the bottom left corner, are independently tested against the peak region. Note in the initial version of this publication released on bioRxiv [9] only the peak vs. the entire neighborhood region was tested. The filter steps described guarantee that only neighborhoods with a centering peak value are considered.

Analyses

The algorithm was tested on various cell types published by Rao 2014 [8] to verify the chromatin loop detection algorithm results: GM12878, K562, IMR90, HUVEC, KBM7, NHEK, and HMEC. First, the parameter setting for HiCExplorer is investigated, and second, the loop detection results of several algorithms are compared. HiCExplorer's implementation is tested against the HiCCUPS algorithm from the Juicer software, HOMER's loop detection, chromosight, cooltools' *call-dots*, Fit-Hi-C, and Peakachu. The algorithms of GOTHIC, cLoops, and FastHiC are not considered, due to the differing focus of these algorithms. The detected chromatin loop locations are correlated with binned protein peak locations of the 11-zinc finger protein CTCF identified by ChIP-Seq. CTCF is a known loop binding factor [8] although not all peaks need to have CTCF attached [13], especially in the case of a gene or a polycomb-mediated loop [14]. In order to test the algorithms mentioned above, the detected chromatin loops were accepted as true if CTCF was detected at both loci, otherwise rejected. CTCF was matched to the GM12878, HMEC, HUVEC, K562, and NHEK cell samples; for IMR90 and KBM7, no CTCF from the same source is provided. A downside of ChIP-Seq is the one-dimensionality. In addition, therefore, two-dimensional data for CTCF, H3K27ac, SMC1, and RAD21 created by HiChIP and ChIA-PET were tested for the GM12878 data set to investigate how one-dimensionality affects the results.

HiCExplorer parameters

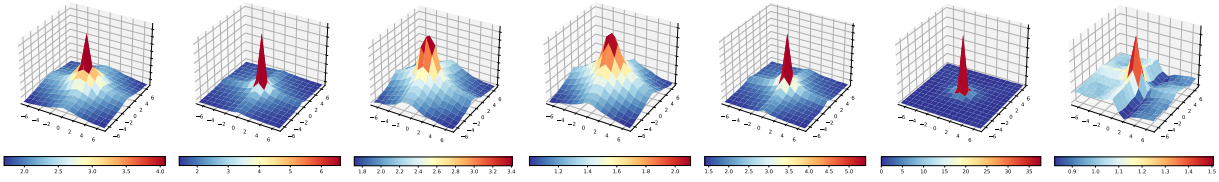
The parameters of HiCExplorer have an influence on the results of the algorithm. First, the threshold for the observed/expected values is negatively correlated with the number of detected loops. A threshold of 0.5 results in 12331 loops, a threshold of 1 in 12008, but a threshold of 1.5 and 2 results in 9147 and 6099 detected loops, respectively. The stricter the threshold, the more accurate the loops; however, the number of detected loops is lower. The p-value for the continuous negative binomial functions has the same effect: the stricter the threshold, the fewer loops are detected, but they become more accurate, as measured by CTCF correlation. Choosing good values for the peak window size and the neighborhood window size parameters presents some difficulty. The peak window size should be the smaller of the two, and the two values should not be too similar. A peak window size of 4 and a neighborhood window size of 5 leads to 2380 loops, but if the peak window size is reduced to 2, 9147 loops are detected. Increasing the two parameters by the same amount, to a peak window size of 4 and neighborhood size of 7, such that the same difference between the values is maintained, leads to a lower number of detected loops, 7269, with an equal level of accuracy, 0.70 vs. 0.69. The threshold for the peak region and the neighborhood test has an expected effect on loop detection. The stricter it is set, the fewer loops are detected, but the accuracy increases. The different methods provided for computing the expected value do not contribute to significant differences in the results. Supplementary Figure 3 and Supplementary Table 6 show the expected value based on all interactions (Equation 4) has the best accuracy (CTCF ChIP-Seq 0.71; CTCF ChIA-PET 0.64), the expected value based on the non-zero interactions (Equation 5) has the highest number of detected loops (14144) and provides more absolute correlated loop locations (CTCF ChIP-Seq 9352; CTCF ChIA-PET 7808). Last, the correction for ligation events



(a) HOMER **(b)** Chromosight **(c)** cooltools

(d) Fit-Hi-C **(e)** Peakachu **(f)** Peakachu (author data)

Figure 1. Intersection of detected loops of HiCExplorer, HiCCUPS and either HOMER, chromosight, cooltools, Fit-Hi-C or Peakachu. HiCExplorer, HiCCUPS, and cooltools have the highest relative intersection. Chromosight has the most intersected loops, but detects many false positives, predicting six times more interactions. Homer, Fit-Hi-C, and Peakachu have only a minor intersection. Last, the loop results of Peakachu, as published by the authors (subfigure **f**), shows a higher overlap with the detected loops of HiCExplorer and HiCCUPS compared to the results we computed.



(a) HiCExplorer **(b)** HiCCUPS **(c)** HOMER **(d)** Chromosight **(e)** Cooltools **(f)** Fit-Hi-C **(g)** Peakachu

Figure 2. Aggregated loop locations of detected loops on GM12878, 10 kb resolution for the different detection algorithms. Aggregation is performed with HiCExplorer's hicAggregateContacts.

as proposed by the HOMER (Equation 6) software shows the lowest accuracy (CTCF ChIP-Seq 0.58; CTCF ChIA-PET 0.48). The results depend on the data: the fewer reads a Hi-C matrix has, the sparser it is, and the fitted distributions are more biased towards zero. In this case, interactions with a lower interaction count have a lower p-value and are more likely to be detected. However, excluding the zero contacts from the distribution can lead to a bias in the other direction; interaction values that should be detected have a p-value which is too high and are therefore excluded from the computation.

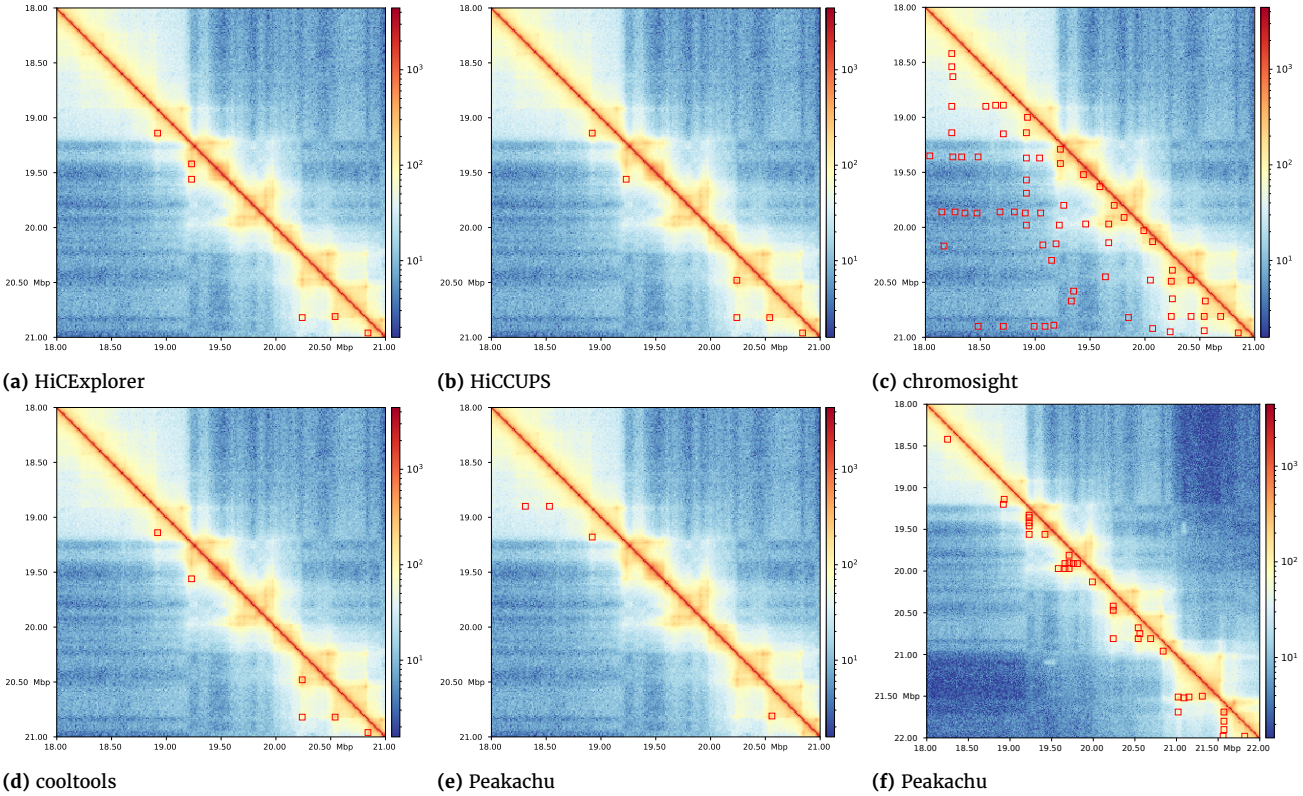
For other cell lines published by Rao 2014, the situation is comparable (Supplementary Table 1). The number of detected loops ranges between 3000 and 10000 loops. The non-zero values and implicitly the read coverage per bin help to explain this different detection behavior; the higher the read coverage, the more regions are detected (see Supplementary Tables 1, 4, and 5). The candidate selection approach via the definition of a neighborhood makes the algorithm sensitive to the Hi-C interaction matrix's resolution. The lower the resolution, the smaller the neighborhood needs to be. Otherwise, the chances of having elements in the neighborhood, peaks or TADs, or even the main diagonal, are too high. At the same time, decreasing the size of the neighborhood creates another issue: the number of elements in the peak and background regions becomes too low. This leads to non-significant test results and to the insight that firstly, the neighborhood size should be adjusted to the bin resolution of the Hi-C matrix, and secondly, that a neighborhood should contain at least around 250 - 300 elements to produce valuable results.

Comparison to state-of-the-art approaches

In the following section the detected loops by different tools on the Hi-C interaction matrices of the cell lines GM12878, HMEC, HUVEC, IMR90, K562, KBM7 and NHEK (by [8]) with the Knight-Ruiz correction [2] are compared. The search distance is restricted to 8 MB if the tool allows this; the results are post-processed for all others. The tools compared are: HiCExplorer, HiCCUPS, HOMER, chromosight, cooltools, Fit-Hi-C, and Peakachu.

Detected loop comparison

The detection rate is comparable for all tools and cell lines (Supplementary Table 1), except for chromosight and Peakachu. Chromosight detects significantly more loops with a very low p-value; however, as the loops' visualization (Figure 3c, chromosome 1 18.00 - 22.00 MB)) indicates, most detected loops are in very noisy regions, and it is questionable what exactly chromosight is detecting. This is supported by the analysis of additional regions, see Supplementary Figure 4c (chromosome 4 20.55 - 22.55 MB), 6c (chromosome 1 15.00 - 18.00 MB) and 8c (chromosome 10 90.00 - 92.00 MB). On the other hand, Peakachu detects much fewer loops than the other algorithms considered. After correspondence with the authors, it became clear that the models provided were trained on ICE-corrected



(a) HiCExplorer **(b)** HiCCUPS **(c)** chromosight

(d) cooltools **(e)** Peakachu **(f)** Peakachu

Figure 3. Plot of chr1 18 - 22 Mb on GM12878, with the detected loops highlighted from each software. HiCExplorer, HiCCUPS, and cooltools show similar results. Chromosight detects many loops in noisy regions and lacks specificity. The four loops of Peakachu show a general issue of this algorithm: The first two loops (18 Mb region) are in a region without enrichment, and the two others slightly miss the enriched interactions by a few kilobases. HOMER and Fit-Hi-C do not detect any loop in the area. Plots are produced using HiCExplorer hicPlotMatrix.

matrices, whereas we have used Knight-Ruiz corrected matrices. For this reason, the loops detected by Peakachu, as published by the authors, have also been taken into consideration. Nonetheless, a detailed analysis of loop loci shows that Peakachu misses important loops, regardless of whether the KR data or the author's own results are considered. For example, the region chromosome 4 20.55 - 22.55 Mb contains four visible loops: Peakachu on KR detects two of them, and misses one completely. Additionally, two locations are detected that slightly miss a loop (Supplementary Figure 4e). The Peakachu results provided by the authors miss two loops and detect the two others successfully (Supplementary Figure 4f). Supplementary Figure 6e shows another issue of Peakachu on KR data. Many loops are detected at the border of a faulty region; it seems the machine learning approach did not have access to this kind of data in training. The data provided by the authors of Peakachu do not have this kind of issue, but overall, while the provided data contain more correct locations, the detection sometimes detects too many loops, for example, in the region chromosome 10 90 - 92 Mb (Supplementary Material 8f). The third problematic tool is Fit-Hi-C. The number of detected loops is at first sight comparable to the other tools; the loci-specific analysis cannot confirm this. The regions chromosome 1 15 - 22 MB (Figure 3 and Supplementary Figure 6h), chromosome 4 20.55 - 22.55 MB (Supplementary Figure 4h) or chromosome 10 90.00 - 92.00 MB have no loops detected by Fit-Hi-C, while the other tools are able to detect loops in these regions. In comparison, the regions where Fit-Hi-C does detect loops are eye-opening. The regions chromosome 1 13.00 - 14.00 MB (Supplementary Figure 5) and chromosome 1 142.00 - 144.00 MB contain mostly very sparse or even faulty Hi-C data. Fit-Hi-C detects an overwhelming amount of enriched pixels in these regions and returns these as loops. While it might be true that these pixels are enriched in a local context, they are far from being a loop. The pattern of the accumulated loop locations (Figure 2) confirms that the detected pattern is usually a single enriched interaction. The other tools detect only very few or no loops in the regions chromosome 1 13.00 - 14.00 MB and chromosome 1 142.00 - 144.00 Mb. The intersection between the detected peaks of HiCExplorer, HiCCUPS, HOMER, chromosight, cooltools, Fit-Hi-C, and Peakachu is quite different (Figure 1). HiCExplorer, with a search distance of 8 Mb, shares ~ 46% of its loops with HiCCUPS. HiCExplorer has the highest intersection of detected loops with chromosight, but chromosight also provides the highest number of detected loops. The intersection of detected loops with cooltools is similar to HiCCUPS; the number of intersecting loops with HOMER, Fit-Hi-C, and Peakachu is lower. HiCCUPS and cooltools show the highest intersecting numbers, chromosight profits from its high detection rate, while HOMER shares only a few hundred loops with HiCCUPS, similar to its intersection with HiCExplorer. The intersection of Fit-Hi-C and Peakachu with HiCExplorer and HiCCUPS is very low, and the results of the Peakachu publication cannot be confirmed. Concerning Peakachu, it can be assumed that the performance is directly connected to the trained models and its inadequate generalization ability. In the publication describing Peakachu, the authors write they have used a probability threshold for a pixel between 90% and 97%. However, to detect a similar number of loops to have comparability, we had to use a score of 68%. For Fit-Hi-C, the authors of Peakachu have used a threshold of 10^{-5}, while we used 0.01 to enable detection of a few thousand loops.

Loop location correlation to protein locations

The detected loops are correlated with CTCF and cohesin factors (Supplementary Table 2) to investigate the amount of inter-

secting locations. This correlation is computed because it was shown that at the anchor points of loops, the proteins CTCF and cohesin are involved as loop binding factors [8, 14]. However, the loop structures representing gene or polycomb-mediated loops do not have CTCF at their anchor points, and the correlation can only be as good as the quality of the ChIP-Seq data from which it is derived. This measurement is, therefore, only an indicator of the accuracy of the detection.

The number of loops detected by HiCExplorer are comparable to HiCCUPS. On the GM12878 cell line and correlated to ChIA-PET based CTCF locations, HiCExplorer detects a similar amount of loops compared to HiCCUPS (6540 vs. 6564) but is more specific (0.64 vs. 0.61). Cooltools (5467 loops) and the loops provided by the Peakachu authors (8174 loops) have a similar relative value of 54% and 50%. Based on our computations, the loops detected with Peakachu have a match at only 686 loop locations and a relative value of 5%. The correlation for the other three tools is also low. Chromosight has 7205 loops correlated, a share of only 11%, Homer has 1349 loops and a share of 18%, and last, Fit-Hi-C has only 163 correlated loop locations with a share of 2%.

The correlation of locations for ChIA-PET RAD21, a cohesin subfactor, has overall significantly lower correlations. HiCExplorer has 2577 loops (25%), HiCCUPS 2385 loops (22%), cooltools 1781 loops (17%), and the loop locations provided by the Peakachu authors 2554 (15%). All other tools have a meager share of correlated locations of < 3%. As a second source of information, data from HiChIP experiments is also considered. The correlation values are overall much higher: for the histone H3K27ac, the highest correlation is achieved by the author-provided Peakachu results with 96%, followed by HiCCUPS with 92%, cooltools with 85% and HiCExplorer reaching only fourth place with 86%. The results of the other tools are also much higher than the results of CTCF and RAD21; for example, Fit-Hi-C had only 2% matches with RAD21, but has 29% with H3K27ac. The correlation based on SMC1, a cohesin subfactor, created with HiChIP indicates the same: again, the author-provided Peakachu results are the highest with 99% followed by HiCCUPS (96%), cooltools (94%), HiCExplorer (91%) and Homer (90%). The correlation of the low performing Fit-Hi-C 2 is 34% higher compared to other proteins, but is also low compared to all other tools. Last, the proportions of the detected locations of the different tools were tested for significant differences. A two-sided proportion z-score test was used, and given the H0 'the proportion is equal', the H0 was rejected for all datasets and tools, under a p-value of 0.05 (Supplementary Table 3).

Averaged loop structure comparison

The accumulated contacts of all detected loop locations on GM12878, displayed as a 3D plot in Figure 2, shows that all algorithms detect enriched regions, but the neighborhood structure is very different. HiCExplorer detects a sharp peak with an enriched direct neighborhood, while HiCCUPS and Fit-Hi-C have a very sharp peak with almost no neighborhood signal. HOMER and Chromosight detect broader peaks with a highly enriched neighborhood, and cooltools has a sharp peak and a neighborhood structure that is slightly more enriched than HiCCUPS and slightly less than HiCExplorer. Finally, Peakachu detects a sharp peak and has a neighborhood plateau on one side, similar to the other algorithms, but a sharp cliff on the other side. The visualizations indicate that a broad peak detection as provided by HOMER and Chromosight, or a very sharp peak with no neighborhood signal, have a low correlation to CTCF-based loops. The visualization of Peakachu's loop locations with the sharp cliff can be interpreted as locations with a TAD border. This can be explained in the context of a learned model based on CTCF locations, because CTCF is present at both loop locations and TAD boundaries.

Runtime and memory usage

The runtime and memory performance is a crucial factor in determining the quality of an algorithm, as well as its implementation. The performance was measured on the Hi-C interaction matrices of the cell lines by Rao [8] discussed above, with a 10 kb resolution for the tools HiCExplorer, HiCCUPS, Homer, chromosight, cooltools, Fit-Hi-C 2, and Peakachu. The measures was computed on an AMD 3700X with 128 GB memory and an Nvidia GTX 1070. For a fair comparison, the CPU implementations are considered, but for completeness, it should be mentioned that the GPU implementation of HiCCUPS with the search space restriction mode of 8 MB active was over all datasets the fastest approach.

On the 8 Mb search distance range, HiCExplorer is the fastest CPU implementation, except for GM12878 cell lines where the CPU-based version of HiCCUPS is faster. HiCExplorer is ~ 44% faster than chromosight (4:25 min vs. 6:22 min) on GM12878 with a 8 Mb search distance, and uses only 6.7 GB memory, while chromosight consumes 39 GB. Moreover, HiCExplorer is two times faster than cooltools if only loop detection is considered; if the necessary computation of expected values is added, it is almost 3.5 times faster (Supplementary Table 8). When considering the somewhat theoretical measure of single-core performance, Chromosight is the fastest algorithm (Supplementary Table 9); nonetheless, modern CPUs support up to 64 cores / 128 threads, and data analysis software should use the offered resources as well as possible. For this reason, HiCExplorers' hicDetectLoop supports parallelization by chromosomes as well as intra-chromosomal parallelization. The data structure allows this: each chromosome can be computed independently, as can each genomic distance normalization, distribution fitting, and p-value computation. For example, if 23 threads are used to compute each chromosome in parallel, and for each chromosome thread, ten other threads compute all intra-chromosomal computations in parallel, a total of 230 parallel threads are used. Not all threads are used at the same time; therefore, a good utilization is achieved. However, modern CPUs with core/thread counts of 64 / 128 can be fully utilized with this approach. Two of the algorithms, Fit-Hi-C and Peakachu, provide only a single-core implementation. Their runtimes are by far the slowest, taking 4:46 hours and 7:03 hours on the GM12878 data, and consume at the same time a high amount of memory. HOMER is, in all scenarios, the third slowest algorithm and also consumes the most memory. However, HOMER is also the only algorithm without any search space restriction parameter, so that all searches are performed genome-wide. This has the side effect that, for example, the GM12878 dataset could only be computed using a single core, because the memory consumption was already around 100 GB. The approach chosen by the developers of HOMER to not support any binary file format to store and access the Hi-C interaction matrix, such as Juicer's *hic* or the *cooler* [4] file format supported by many of the other investigated tools, results in a computation based on text files and raw data, and contributes, apart from the lack of a search space restriction, to the very poor runtime and memory performance.

Discussion

The search space of an algorithm is the dominant factor determining its accuracy and performance. Therefore, pruning it should be the primary goal when optimizing newly designed algorithms. In theory, brute force solutions which apply no restrictions to the search space, like HiCCUPS, can detect all possible enriched regions, but the result is an implementation

with very demanding hardware requirements. HiCCUPS solves this by utilizing massively parallel computational resources via GPGPU. On the other hand, HOMER also applies no limitations to the search space, yet detects a lower number of loops, and those which are detected have a significantly lower correlation over all samples to CTCF localization. HOMER does support a parallel computation per chromosome, like HiCExplorer but is significantly slower than all other solutions and uses significantly more memory per core. HOMER's poor runtime performance can be explained by the fact that computation is performed on raw data, while all other approaches use precomputed interaction matrices. Chromosight is a fast detection approach and provides the fastest single-core performance; however, it lacks specificity and detects many loops that should be considered noise, even though these loops may be provided with a high significance. Cooltools, with its reimplementation of the HiCCUPS approach, provides a genome distance search which makes it faster and more flexible. The results are good, but it is unclear why they are not more similar to Juicer's HiCCUPS results, given that the same algorithm is used. An overview of the properties of all algorithms is provided in Supplementary Table 10.

The divergence between the Peakachu results based on our computations and the data published by the authors is high. Given that the machine learning-based model of Peakachu is trained using the locations of certain proteins, it is unsurprising that the H3K27ac and SMC1 locations have very high correlation values. Our understanding is that the published trained model does not detect loop locations themselves, but rather the locations of SMC1 and H3K27ac. Moreover, the poor performance on the KR corrected matrix used indicates heavy overfitting and a poor generalization ability onto different kinds of input matrices. Another explanation could be the lack of preprocessing to normalize the input data. The idea of training a model using the locations of proteins known to be correlated with loops is sensible, but is limited by the fact that not all loop locations have CTCF and cohesin at their anchors. Overall, the model is an interesting approach; nonetheless, the published model requires a more diverse training data to improve performance on varying input datasets.

Furthermore, it could be shown that the sparsity and thus the read coverage of a Hi-C interaction matrix significantly influences the detection of peaks in their neighborhood. The sparser a Hi-C interaction matrix is, the more likely that the possible valid regions detected by the continuous negative binomial distribution filtering are rejected by the Wilcoxon rank-sum test. The large number of differences between the detected loops and the high correlation rates to CTCF can be explained in multiple ways. The correlation to CTCF has its roots in biology. Not all loops have CTCF as a binding protein at its anchors; gene-loops or polycomb-mediated loops lack it. All the algorithms detect enrichments in the Hi-C data, which are interpreted as loops, but may also have other explanations. The enrichments can also be noise in the data, or interactions which are unrelated to CTCF. Secondly, the Hi-C data is created with in-situ Hi-C and has a higher noise level than newer approaches like Arima Hi-C[3]. Detections of loops in noisy areas is responsible for the low intersection values for the predictions of the competing algorithms, in particular for chromosight, which detects more noise than loops.

3 https://arimagenomics.com/

Availability of source code and requirements

HiCExplorer is licensed under GPLv3 and is available on Github (https://github.com/deeptools/HiCExplorer/) or as a conda package in the Bioconda channel [15]. HiCExplorer is implemented in Python 3.6, 3.7. and 3.8 for Linux and macOS.

Availability of supporting data and materials

Hi-C data: GSE63525; Rao et al. [8]. CTCF for: Gm12878 from GSM935611; Hmec from GSM749753; Huvec from GSM749749; K562 from GSM733719 and Nhek from GSM733636. CTCF ChIA-PET (GSM1872886); H3K27ac HiChIP (GSE101498), SMC1 HiChIP (GSE80820), and RAD21 ChIA-PET (GSM1436265). Result files are available via DOI: 10.5281/zenodo.5648500

Declarations

Competing Interests

The author(s) declare that they have no competing interests.

Funding

German Federal Ministry of Education and Research [031 A538A de.NBI-RBC awarded to R.B.]; German Federal Ministry of Education and Research [031 L0101C de.NBI-epi awarded to B.G.]. R.B. was supported by the German Research Foundation (DFG) under Germany's Excellence Strategy (CIBSS - EXC-2189 - Project ID 390939984).

Author's Contributions

JW: Designed and implemented the presented algorithm and wrote the manuscript. RB: contributed to the manuscript. BG: contributed to the manuscript.

Acknowledgements

We thank Simon Bray and Anup Kumar for proofreading the manuscript.

References

1. Imakaev M, Fudenberg G, McCord RP, Naumova N, Goloborodko A, Lajoie BR, et al. Iterative correction of Hi-C data reveals hallmarks of chromosome organization. Nature Methods 2012 sep;9(10):999–1003. http://www.nature.com/doifinder/10.1038/nmeth.2148, [PubMed:22941365] [PubMed Central:PMC3816492] [doi:10.1038/nmeth.2148].
2. Knight PA, Ruiz D. A fast algorithm for matrix balancing. IMA Journal of Numerical Analysis 2013;33(3):1029–1047. [doi:10.1093/imanum/drs019].
3. Heinz S, Benner C, Spann N, Bertolino E, Lin YC, Laslo P, et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Molecular Cell 2010 may;38(4):576–589. https://www.sciencedirect.com/science/article/pii/S1097276510003667?via{%}3Dihub, [PubMed:20513432] [PubMed Central:PMC2898526] [doi:10.1016/j.molcel.2010.05.004].
4. Abdennur N, Mirny LA. Cooler: scalable storage for Hi-C

data and other genomically labeled arrays. Bioinformatics 2020;36(1):311–316.
5. Matthey-Doret C, Baudry L, Breuer A, Montagne R, Guiglielmoni N, Scolari V, et al. Computer vision for pattern detection in chromosome contact maps. Nature communications 2020;11(1):1–11.
6. Kaul A, Bhattacharyya S, Ay F. Identifying statistically significant chromatin contacts from Hi-C data with FitHiC2. Nature protocols 2020;15(3):991–1012.
7. Salameh TJ, Wang X, Song F, Zhang B, Wright SM, Khunsriraksakul C, et al. A supervised learning framework for chromatin loop detection in genome-wide contact maps. Nature communications 2020;11(1):1–12.
8. Rao SSP, Huntley MH, Durand NC, Stamenova EK. A 3D Map of the Human Genome at Kilobase Resolution Reveals Principles of Chromatin Looping. Cell 2014;159(7):1665–1680. http://dx.doi.org/10.1016/j.cell.2014.11.021, [PubMed:25497547] [PubMed Central:PMC5635824] [doi:10.1016/j.cell.2014.11.021].
9. Wolff J, Backofen R, Gruening B. Loop detection using Hi-C data with HiCExplorer. bioRxiv 2020;.
10. Cameron AC, Trivedi PK. Regression-based tests for overdispersion in the Poisson model. Journal of econometrics 1990;46(3):347–364.
11. Robinson MD, McCarthy DJ, Smyth GK. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 2010;26(1):139–140.
12. McCarthy DJ, Chen Y, Smyth GK. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic acids research 2012;40(10):4288–4297.
13. Andrey G, Schöpflin R, Jerković I, Heinrich V, Ibrahim DM, Paliou C, et al. Characterization of hundreds of regulatory landscapes in developing limbs reveals two regimes of chromatin folding. Genome research 2017;27(2):223–233. [PubMed:27923844] [PubMed Central:PMC5287228] [doi:10.1101/gr.213066.116].
14. Bonev B, Cavalli G. Organization and function of the 3D genome. Nature Reviews Genetics 2016;17(11):661. [PubMed:28704353] [doi:10.1038/nrg.2016.147].
15. Grüning B, Dale R, Sjödin A, Chapman BA, Rowe J, Tomkins-Tinch CH, et al. Bioconda: sustainable and comprehensive software distribution for the life sciences. Nature methods 2018;15(7):475. [PubMed:29967506] [doi:10.1038/s41592-018-0046-7].

Click here to access/download
Supplementary Material
supplementary_material_loop.pdf