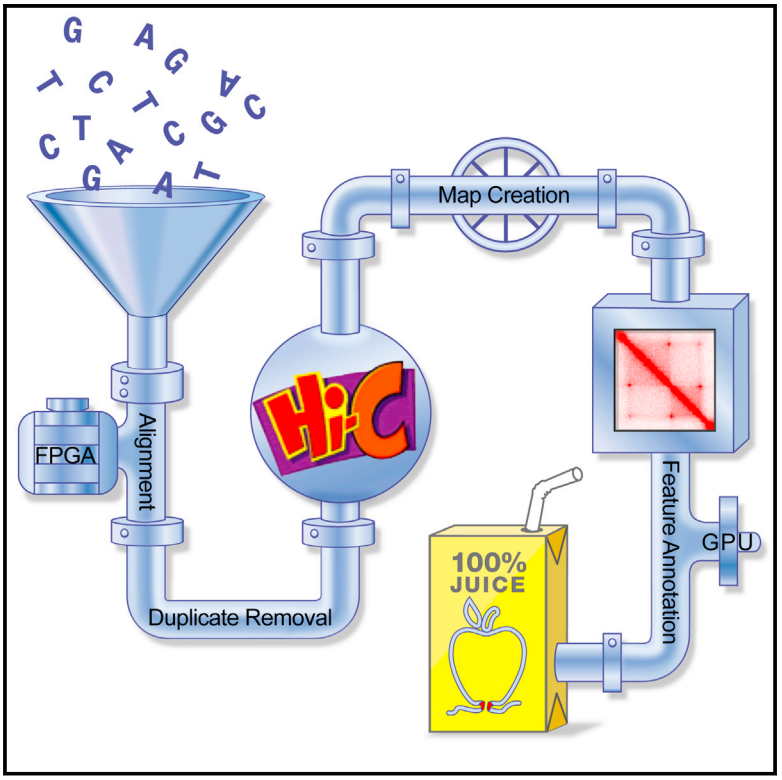
Tool

Cell Systems

Juicer Provides a One-Click System for Analyzing Loop-Resolution Hi-C Experiments

Graphical Abstract



Authors

Neva C. Durand,
Muhammad S. Shamim, Ido Machol,
Suhas S.P. Rao, Miriam H. Huntley,
Eric S. Lander, Erez Lieberman Aiden

Correspondence

erez@erez.com

In Brief

Durand et al. introduce Juicer, a one-click, end-to-end pipeline for analyzing data from Hi-C and other contact mapping experiments. Juicer generates contact matrices at multiple resolutions and identifies features including contact domains and loops.

Highlights

- Juicer enables users to process terabase scale Hi-C datasets with a single click
- Juicer automatically annotates loops and contact domains
- Juicer is available as open source software
- Juicer is compatible with multiple cluster operating systems and with Amazon Web Services

Durand et al., 2016, Cell Systems *3*, 95–98
July 27, 2016
http://dx.doi.org/10.1016/j.cels.2016.07.002

CellPress

Cell Systems
Tool

Juicer Provides a One-Click System for Analyzing Loop-Resolution Hi-C Experiments

Neva C. Durand,[1,2,3,4,10] Muhammad S. Shamim,[1,2,3,10] Ido Machol,[1,2,3] Suhas S.P. Rao,[1,2,3,5] Miriam H. Huntley,[1,2,3,6] Eric S. Lander,[4,7,8] and Erez Lieberman Aiden[1,2,3,4,9,*]
[1]The Center for Genome Architecture, Baylor College of Medicine, Houston, TX 77030, USA
[2]Department of Molecular and Human Genetics, Baylor College of Medicine, Houston, TX 77030, USA
[3]Department of Computer Science and Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005, USA
[4]Broad Institute of Harvard and Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA
[5]School of Medicine, Stanford University, Stanford, CA 94305, USA
[6]John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA
[7]Department of Biology, MIT, Cambridge, MA 02139, USA
[8]Department of Systems Biology, Harvard Medical School, Boston, MA 02115, USA
[9]Center for Theoretical Biological Physics, Rice University, Houston, TX 77030, USA
[10]Co-first author
*Correspondence: erez@erez.com
http://dx.doi.org/10.1016/j.cels.2016.07.002

SUMMARY

Hi-C experiments explore the 3D structure of the genome, generating terabases of data to create high-resolution contact maps. Here, we introduce Juicer, an open-source tool for analyzing terabase-scale Hi-C datasets. Juicer allows users without a computational background to transform raw sequence data into normalized contact maps with one click. Juicer produces a *hic* file containing compressed contact matrices at many resolutions, facilitating visualization and analysis at multiple scales. Structural features, such as loops and domains, are automatically annotated. Juicer is available as open source software at http://aidenlab.org/juicer/.

Hi-C experiments probe the 3D structure of DNA and chromatin by ligating and sequencing DNA loci that are spatially proximate to one another (Lieberman-Aiden et al., 2009; Rao et al., 2014). The resulting maps reflect patterns of physical contact between loci, making it possible to deduce how loci are organized in 3D.

Efforts to improve the resolution of 3D maps have caused the amount of DNA sequence produced from Hi-C experiments to skyrocket. Our original maps, derived from 30 million reads and 16 Gb of DNA sequence, described the genome at 1 megabase resolution (Lieberman-Aiden et al., 2009). In contrast, we recently generated 6.5 billion reads and 1.6 Tb of DNA sequence in order to create a single 3D map of the genome at kilobase resolution (Rao et al., 2014).

Although pipelines for Hi-C data analysis exist (Lieberman-Aiden et al., 2009; Schmid et al., 2015; Servant et al., 2015; Sauria et al., 2015), these packages are not designed to process datasets at the terabase scale or to annotate the structural features that these maps reflect. Moreover, when designing tools that require high-performance computation, ensuring reliability and ease-of-use across software platforms and hardware instances becomes a crucial desideratum. Ensuring such compatibility can be a considerable engineering challenge.

Here, we introduce Juicer, an easy-to-use, fully-automated pipeline for the processing and annotation of data from Hi-C and other contact mapping experiments. Juicer is closely based on the algorithms that we recently developed to analyze and annotate our terabase-scale Hi-C experiments (Rao et al., 2014). In order to meet the engineering challenge of handling such massive datasets, Juicer supports the use of parallelization and hardware acceleration whenever possible, including CPU clusters, general-purpose graphics processing units (GPGPUs), and field-programmable gate arrays (FPGAs). Juicer is also compatible with a variety of cloud and cluster architectures.

Juicer comprises three tools, which are designed to be run one-after-another (Figure 1).

First, Juicer transforms raw sequence data into a list of Hi-C contacts (pairs of genomic positions that were adjacent to each other in 3D space during the experiment). To accomplish this, read pairs are aligned to the genome, both duplicates and near-duplicates are removed, and read pairs that align to three or more locations are set aside. When appropriate hardware is available, this procedure can be accelerated either by parallelizing across multiple CPUs or by using an FPGA (see Table 1).

Next, the catalog of contacts is used to create contact matrices. To do so, the linear genome is partitioned into loci of a fixed size, or "resolution," (e.g., 1 Mb or 1 kb). These loci correspond to the rows and columns of a contact matrix; each entry in the matrix reflects the number of contacts observed between the corresponding pair of loci during a Hi-C experiment. Due to factors such as chromatin accessibility, certain loci are observed more frequently in Hi-C experiments. Juicer can adjust for these biases in multiple ways. The options include our original normalization scheme (Lieberman-Aiden et al., 2009), as well as a matrix balancing scheme that ensures that each row and column of the contact matrix sums to the same value (Knight and Ruiz, 2012). A wide array of quality statistics is also calculated, making it possible to assess the success and reliability of a given experiment before the costly deep-sequencing step.

The contact matrices generated in this way are stored efficiently in a compressed format that is designed to facilitate all subsequent computations. For instance, 1 terabyte of raw sequencing data is

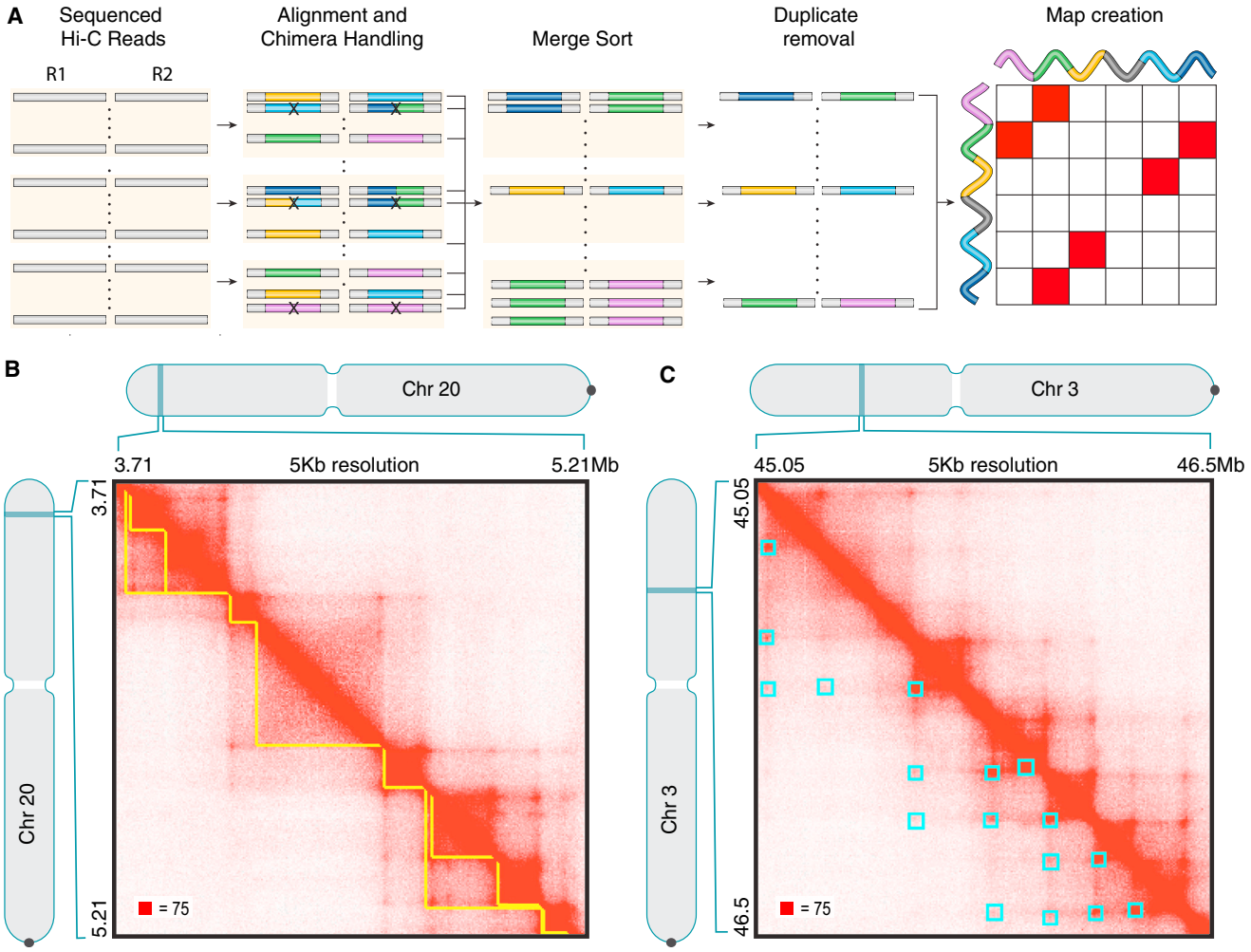


Figure 1. Juicer Analyzes Terabases of Hi-C Data with One Click

(A) Sequenced read pairs (horizontal bars) are aligned to the genome in parallel. Color indicates genomic position. Read pairs aligning to more than two positions are excluded. Those remaining are sorted by position and merged into a single list, at which point duplicate reads are removed. The *hic* file stores contact matrices at many resolutions and can be loaded into Juicebox for visualization. See Table S2.

(B) Contact domains (yellow) are annotated using the Arrowhead algorithm.

(C) Loops (cyan) are annotated using HiCCUPS.

represented as an 80 gigabyte *hic* file containing normalized and non-normalized contact matrices at 18 different resolutions, from 2.5 Mb resolution to single restriction fragment resolution for a 4-cutter restriction enzyme (~400 bp). Contact matrices in the *hic* format can also be visualized using Juicebox, which is described in the accompanying paper (Durand et al., 2016).

Finally, Juicer contains a suite of algorithms that are designed to annotate contact matrices and thus identify features of genome folding. These features include loops, loop anchor motifs, and contact domains.

Loops are identified using the HiCCUPS algorithm (Rao et al., 2014), which searches for clusters of contact matrix entries in which the frequency of contact is enriched relative to the local background. Because there are trillions of pixels in a kilobase-resolution Hi-C map, HiCCUPS is implemented using GPGPUs. Given CTCF and/or cohesin ChIP-seq tracks for the same cell type, HiCCUPS can frequently use FIMO (Grant et al., 2011) to identify the CTCF motif that serves as the anchor for each loop. We recently performed CRISPR experiments disrupting seven different CTCF motifs, each of which was identified by HiCCUPS as the anchor of one or more loops. In each case, disruption of the motif led to disruption of the corresponding loop, thus confirming the accuracy of HiCCUPS loop anchor annotations (Sanborn et al., 2015).

Contact domains are identified using a dynamic programming algorithm that relies on applying the Arrowhead transformation $[Ai,i + d = (M^*_{i,i-d} - M^*_{i,i+d})/(M^*_{i,i-d} + M^*_{i,i+d})]$ to a normalized contact matrix M^* (Rao et al., 2014). Many of these domains are associated with loops and can be disrupted by manipulating the corresponding loop anchors (Sanborn et al., 2015).

It is frequently useful to examine the cumulative signal from a large number of putative features at once, including both loops

Table 1. Using Juicer to Process 1.5 Billion Paired-End Hi-C Reads on Different Cluster Systems

System	Amazon Web Services g2.8 × Large			Broad Univa Grid Engine			Rice PowerOmics			Rice PowerOmics + FPGA		
CPU	Intel Xeon E5-2670 at 2.60 GHz			Intel Xeon X5650 at 2.66 GHz			IBM POWER8E at 2.061 GHz revision: 2.1			IBM POWER8E at 2.061 GHz revision: 2.1		
Cores/node	4 × 8 cores			4 × 6 cores			2 × 24 cores			2 x 24 cores		
RAM	60 GB			32 GB			256 GB			256 GB		
Cluster OS	OpenLava 2.2 (LSF compatible)			UGE 8.3.0			Slurm 14.11.8			Slurm 14.11.8		
GPU	NVIDIA Quadro K5000			none			NVIDIA Tesla K80			NVIDIA Tesla K80		
FPGA	none			none			none			Edico Genome DRAGEN Bio-IT Platform		
Max parallel cores	32			1,200			1,536			1,536		
	Core Hours (hr:min)	RAM (GB)	VM (GB)	Core Hours (hr:min)	RAM (GB)	VM (GB)	Core Hours (hr:min)	RAM (GB)	VM (GB)	Core Hours (hr:min)	RAM (GB)	VM (GB)
Align	8,744:49	12.3	13.5	11,614:07	10.8	11.9	4,221:29	13.1	14.0	1:29	0	0
Merge sort	35:36	9.9	10.1	117:03	8.7	198.1	452:13	14.0	120.0	426:30	30.0	120.0
Duplicate removal	12:21	0.5	0.5	17:04	0.4	0.5	3:12	0.4	0.0	1:28	0.4	0.0
.hic creation	112:43	21.8	34.9	209:43	13.4	19.5	139:17	19.3	8	177:04	19.3	8
Feature annotation	2:07	10.5	139.3	1:04	6.4	19.5	3:25	4.2	9.1	4:28	77.1	9.1
Total	8,906:11			11,959:01			4,819:36			608:59		

"RAM (Gb)" (resp., "VM(Gb)") are the maximum RAM (resp., virtual memory") used for each task. Loop annotation was not performed on the Broad cluster, which does not offer GPUs.
See also Table S1.

and domains. To this end, Juicer includes an implementation of Aggregate Peak Analysis (Rao et al., 2014).

Juicer is an open-source project. It is available at https://github.com/theaidenlab/juicer as a series of packages designed for a variety of hardware configurations: either a single machine, or clusters that run LSF, Univa Grid Engine, or SLURM. In addition, Juicer is available on the cloud at Amazon Web Services, and the test data used to review this paper is available at http://dx.doi.org/10.17632/c6bg4cbggn.1. Table 1 displays different performance metrics on each cluster system; the details of each setup are in the supplemental text. Once installed, Juicer can be executed using a single command by users without informatics experience.

EXPERIMENTAL PROCEDURES

All algorithms and data are drawn from Rao et al. (2014), except as described in the Supplemental Information.

SUPPLEMENTAL INFORMATION

Supplemental Information includes Supplemental Experimental Procedures and two tables and can be found with this article online at http://dx.doi.org/10.1016/j.cels.2016.07.002.

AUTHOR CONTRIBUTIONS

E.L.A. conceived of this project. N.C.D. created the pipeline. S.S.P.R. created HiCCUPS. M.H.H. created APA. M.H.H. and N.C.D. created Arrowhead. M.S.S. re-implemented all feature annotation algorithms in Java as fully-automated, end-to-end tools. I.M. ported the pipeline to SLURM and AWS. N.C.D., M.S.S., I.M., and E.S.L. contributed to tool development. N.C.D. and E.L.A. prepared the manuscript.

ACKNOWLEDGMENTS

This work was supported by NIH New Innovator Award 1DP2OD008540, NIH 4D Nucleome grant U01HL130010, National Science Foundation (NSF) Physics Frontier Center PHY-1427654, National Human Genome Research Institute (NHGRI) HG006193, Welch Foundation Q-1866, Cancer Prevention Research Institute of Texas Scholar Award R1304, an NVIDIA Research Center Award, an IBM University Challenge Award, a Google Research Award, a McNair Medical Institute Scholar Award, and the President's Early Career Award in Science and Engineering to E.L.A., an NHGRI grant (HG003067) to E.S.L., and a PD Soros Fellowship to S.S.P.R. The Rice PowerOmics cluster was a gift from IBM. The software and test data sets used to review this manuscript are available at http://dx.doi.org/10.17632/c6bg4cbggn.1.

Received: December 2, 2015
Revised: May 2, 2016
Accepted: July 1, 2016
Published: July 27, 2016

REFERENCES

Durand, N.C., Robinson, J.T., Shamim, M.S., Machol, I., Mesirov, J.P., Lander, E.S., and Aiden, E.L. (2016). Juicebox Provides a Visualization System for Hi-C Contact Maps with Unlimited Zoom. Cell Systems *3*, this issue, 99–101.

Grant, C.E., Bailey, T.L., and Noble, W.S. (2011). FIMO: scanning for occurrences of a given motif. Bioinformatics *27*, 1017–1018.

Knight, P.A., and Ruiz, D. (2012). A fast algorithm for matrix balancing. IMA J. Numer. Anal. *33*, 1029–1047.

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

Rao, S.S., Huntley, M.H., Durand, N.C., Stamenova, E.K., Bochkov, I.D., Robinson, J.T., Sanborn, A.L., Machol, I., Omer, A.D., Lander, E.S., et al. (2014). A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. Cell *159*, 1665–1680.

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

Sauria, M.E., Phillips-Cremins, J.E., Corces, V.G., and Taylor, J. (2015). HiFive: a tool suite for easy and efficient HiC and 5C data analysis. Genome Biol. *16*, 237.

Schmid, M.W., Grob, S., and Grossniklaus, U. (2015). HiCdat: a fast and easy-to-use Hi-C data analysis tool. BMC Bioinformatics *16*, 277.

Servant, N., Varoquaux, N., Lajoie, B.R., Viara, E., Chen, C.-J., Vert, J.-P., Heard, E., Dekker, J., and Barillot, E. (2015). HiC-Pro: an optimized and flexible pipeline for Hi-C data processing. Genome Biol. *16*, 259.

Cell Systems, Volume *3*

Supplemental Information

Juicer Provides a One-Click System for Analyzing Loop-Resolution Hi-C Experiments

Neva C. Durand, Muhammad S. Shamim, Ido Machol, Suhas S.P. Rao, Miriam H. Huntley, Eric S. Lander, and Erez Lieberman Aiden

SUPPLEMENTAL EXPERIMENTAL PROCEDURES

Contents

I. Supplemental Tables

	Juicer	Hi-C Pro	HiFive	HiCdat	hiclib
Processes raw sequence data?	✓	✓	✗	✓	✓
Chimeric reads	✓	✓	✗	✗	✓
Normalization	✓	✓	✓	✓	✓
Can handle large data sets	✓	✓	✓	✗	✗
Compact file format	✓	✗	✓	✗	✗
Diploid map capability	✓	✓	✗	✗	✗
Multi-scale map generation	✓	✗	✗	✗	✗
A/B compartment annotation	✓	✗	✗	✓	✓
Contact domain annotation	✓	✗	✗	✗	✗
Peak annotation	✓	✗	✗	✗	✗
Motif discovery	✓	✗	✗	✗	✗

Table S1. Comparison of Hi-C pipelines, Related to Table 1. Hi-C Pro (Servant et al., 2015) can handle the large datasets now typical of Hi-C experiments and can generate diploid maps given a SNP list but does not offer a compact file format or a method for multiple resolution generation in one step. HiFive (Suria et al., 2015) has a compact file format but does not start from raw sequencing data. HiCdat (Schmid et al., 2015) does not allow for chimeric reads, which are common to Hi-C experiments, and cannot handle large datasets. Hiclib is a Python library rather than a one-click pipeline. As such, it requires comfort with computer programming. Although hiclib has not been the subject of a peer-reviewed paper, it has been used in other peer-reviewed papers; we used these as our basis for its capabilities.

Table S2. Hi-C File Format, Related to Figure 1. Exact layout of the compressed *hic* file format used to store the contact maps at multiple resolutions. See Excel table.

II. Experimental Methods

In this paper, we describe Juicer, a one-click system for automatic Hi-C data analysis and feature annotation. The tools that comprise the system were largely described in the supplement to our 2014 *Cell* paper (Rao and Huntley et al., 2014). Here, we will summarize the pipeline and detail some small refinements made since that paper was published.

II.a. Sequence data to normalized Hi-C contact matrices

The first portion of the pipeline, where raw sequence data is converted to a *hic* file containing the normalized Hi-C contact matrices, proceeds exactly as described in the supplement to (Rao and Huntley et al., 2014). The raw sequence data is aligned in chunks in parallel as single end reads using an updated version of BWA, *bwa mem* (Li, 2013). Each chunk is then sorted by read name and the single end alignments are paired again. Ambiguous chimeric reads (those that align to more than two places in the genome) are discarded. Each read is assigned its corresponding restriction fragment based on the restriction enzyme. The read pairs are then sorted by position and the chunks are merged together into one large file for duplicate and near-duplicate removal. The result serves as input to the Juicebox tools "pre" command, which bins the reads at multiple different resolutions and normalizes each matrix at each resolution. The matrices are stored in the *hic* file format, a compressed binary format that allows Juicebox fast random access to different matrices at different resolutions. The *hic* file format is described in detail in section II.j. For an extensive discussion of the above steps, please refer to the supplement to (Rao and Huntley et al., 2014).

II.b. Arrowhead

The Arrowhead matrix transformation described in (Rao and Huntley et al., 2014) annotates genome-wide contact domains. The original algorithm was implemented in MATLAB; the Juicer implementation is in Java. With the exception of minor bug fixes and stability improvements, the details of the algorithm remain the same. As described by (Rao and Huntley et al., 2014), the Arrowhead transformation for annotating contact domains is defined as $A_{i,i+d} = (M^*_{i,i-d} - M^*_{i,i+d})/ (M^*_{i,i-d} + M^*_{i,i+d})$. This is equivalent to calculating (1 - observed/expected) for *(i,i+d)*, where the expected is the average of values at *(i,i+d)* and *(i,i-d)*. Positive values for $A_{i,i+d}$ indicate locus *(i-d)* is within the domain, whereas negative values indicate locus *(i+d)* is within the domain; values near zero indicate both loci are within or outside a domain. As in the original algorithm, the Juicer implementation leverages dynamic programming to calculate the domains in $O(n^2)$ time. Upon a successful run of the Arrowhead algorithm, a Juicebox-loadable file containing the contact domains will be produced in the output directory.

II.c. HiCCUPS

HiCCUPS (Hi-C Computational Unbiased Peak Search) is a sophisticated local peak caller that integrates information from multiple local neighborhoods in its peak calling procedure. The original algorithm was implemented using Python/pyCUDA; the Juicer implementation is in Java/JCUDA. With the exception of minor bug fixes and stability improvements, the details of the algorithm remain the same. As described by (Rao and Huntley et al., 2014), HiCCUPS examines each pixel in a Hi-C contact matrix and identifies those with enriched contact frequencies relative to local neighborhoods (pixels to its lower-left, pixels to its left and right, pixels above and below, and pixels within a "donut" surrounding the pixel of interest), after accounting for larger structural features and correcting for multiple hypothesis testing. Juicer provides a JCUDA-based (Yan et al., 2009) implementation of the HiCCUPS GP-GPU algorithm.

Clustering of enriched pixels found at 5kb, 10kb, and 25kb is conducted as described by (Rao and Huntley et al., 2014). The Juicer implementation allows extensive customization of parameters involved in locating enriched peaks, such as FDR thresholds and local window widths, and parameters involved in post processing enriched peaks, such as the initial clustering radius and observed/expected thresholds for filtering of peaks.

Upon a successful run of the HiCCUPS implementation, a directory will be created containing:

- Separate enriched pixel lists for every resolution specified
- Separate FDR threshold lists for every resolution specified
- Separate post-processed loop lists for every resolution specified
- A merged post-processed loop list

II.d. Motif Finder

Motif Finder determines the unique and inferred motifs responsible for creating a chromatin loop in the manner described in (Rao and Huntley et al., 2014) and validated using CRISPR Cas-9 editing by (Sanborn and Rao et al., 2015). By default, Motif Finder supports motifs for hg19, hg38, mm9, and mm10 using the M1 motif (Schmidt et al., 2012) and FIMO (Grant et al., 2011). Motif Finder requires a set of ChIP-Seq tracks in locating unique and inferred CTCF motifs for given loop list. (Rao and Huntley et al., 2014) describe in detail the process of using CTCF, RAD21, and SMC3 ChIP-Seq tracks to locate unique and inferred CTCF motifs.

Potential CTCF motifs across provided genomes are available at *http://hicfiles.s3.amazonaws.com/internal/motifs/*GENOME_ID*.motifs.txt* (e.g. *http://hicfiles.s3.amazonaws.com/internal/motifs/hg19.motifs.txt*). Motif Finder can also accept custom provided motif lists following FIMO output format (Grant et al., 2011).

The original algorithm used bedtools and STORM (Schones et al., 2007). The Juicer implementation includes a Java port of the required bedtools functions (merge and intersect; Quinlan and Hall 2010). FIMO (Grant et al., 2011) was used instead of STORM for improved capability in finding accurate motifs. Motif Finder results were validated biologically by (Sanborn and Rao et al., 2015).

II.e. APA

APA (Aggregate Peak Analysis) tests the aggregate enrichment of an entire set of putative peaks as described by (Rao and Huntley et al., 2014). APA is especially useful for assessing low-resolution Hi-C maps, where individual peaks may be difficult to discern, but where the aggregate signal should be detectable if the peak set is reliable and the Hi-C experiment was successful. The original algorithm was implemented using Python; the Juicer implementation is in Java. With the exception of minor stability improvements, the details of the algorithm remain the same.

II.f. Other improvements

HiCCUPSDiff is a Java wrapper to the HiCCUPS algorithm that allows users to find significant differences in loop lists given the original *hic* files and respective loop lists. As detailed in (Rao and Huntley et al., 2014), differential loops are only called if a loop is annotated for one file, no overlapping loop was identified in the other file, and the peak pixel displays less than 1.3-fold enrichment over all local neighborhoods in the other file. We have also added our script for creating diploid Hi-C maps from a variant call format (VCF) file containing phased SNPs. The details of the diploid algorithm are unchanged from (Rao and Huntley et al., 2014).

II.g. Cluster implementation

We compared the performance of three different cluster systems running the Juicer pipeline for processing Hi-C data: Amazon Web Services (AWS) Intel-based OpenLava cluster, Rice IBM Power8-based SLURM cluster, and Broad Institute Intel-based Univa GE (UGER) cluster. As detailed in Table 1, these clusters are each using different operating systems, hardware configurations, and cluster management software; performance is determined by the unique combination of software and hardware in each system. In particular, AWS is a dedicated system, whereas the performance of the clusters at Rice and Broad is affected by how many other users are sharing resources. We provide the code for Juicer on all three systems in our GitHub repository.

We used the public IMR90 dataset from (Rao and Huntley et al., 2014), which contains more than 1.5 billion paired-end reads. We generated contact maps down to 5kb resolution together with the list of contact domains returned by Arrowhead and the list of loops returned by HiCCUPS.

Juicer first splits the reads into subsets of 90 million reads each and aligns them in parallel to the human genome reference. Ligations are also counted during the alignment phase. After alignment, each pair is merged into single sorted files and these are then merged into one large file. In the next step, duplicates are removed in parallel. The Hi-C Creation phase consists of using the resulting file to calculate statistics and create the normalized contact maps, which are stored in the *hic* file. Features are then annotated using the *hic* file as input. On systems without GPUs, only contact domains are annotated.

To illustrate the capabilities of Juicer, we used it to re-analyze all 15,976,316,772 paired-end reads (7.3 TB) from GM12878 human lymphoblastoid cells generated in a recent Hi-C study by our lab (Rao and Huntley et

al., 2014). We processed this dataset using different aligners (*bwa sw* (Li et al., 2010) and *bwa mem* (Li, 2013)) and different reference genomes (hg19 and hg38), generating a map containing ~8.9B contacts in each case.

II.h. Beta testing

The three cluster implementations of Juicer have been available since January 2016 on GitHub for beta testing. In the past month, we have also established a Google groups forum for answering questions. There were over 50 different users generating over 150 different posts, indicating a high level of interest and user adoption of our software. Beta users have been able to successfully run Juicer on all three different cluster implementations. We used the feedback we received through the forum to improve Juicer functionality and fix bugs.

II.i. Comparison to other Hi-C pipelines

There have been several Hi-C pipelines published recently (Castellano et al., 2015; Schmid et al., 2015; Servant et al., 2015; Suria et al., 2015). Most will map reads to a genome (though some do not properly handle chimeric reads), bin the Hi-C contacts at a single resolution, and normalize the resulting Hi-C contact map. However, the output of the pipelines is usually a single matrix at a single resolution in sparse or dense text format, which takes up a lot of space on disk and requires rerunning the pipeline to produce Hi-C contact maps at different resolutions. Most also cannot handle the large datasets that are now becoming standard for a Hi-C experiment. Finally, none of the other pipelines offer automatic feature annotation. Table S1 lists some of the other pipelines and their capabilities.

II.j. Specialized file format

The *hic* file format was created by Jim Robinson for Juicebox (Durand and Robinson et al., 2016) and is specially designed to provide fast random access to any contact matrix at any resolution. The footer of a *hic* file contains pointers to all the matrices, together with the size (in bytes) of the matrix. To quickly access a matrix at a particular resolution, the reader looks up the pointer in the footer and reads all the data from that matrix into main memory. Data is also stored in "blocks" of adjacent contacts, enabling fast visualization in two dimensions. This design makes it possible to visualize billions of Hi-C contacts quickly and to zoom in to extremely high resolution in real time in Juicebox. All of the subsequent tools in the Juicer pipeline run on *hic* files.

Table S2 contains the file format. The header portion starts with a field identifying the file as *hic* format and establishing the version number (currently 8). The master index position (the location in bytes in the file where the reader can look up the position of all the matrices) follows. The subsequent fields of genomeID, chromosome dictionary, base-pair resolution dictionary, and fragment resolution dictionary are important functional metadata used to properly read and render the matrices. The attribute dictionary is meant for pure metadata that describes the experiment and is completely customizable.

The body of the *hic* file contains the data for every chromosome-chromosome combination; if there are N chromosomes, there will be N^2 matrices represented. Each matrix starts with its unique chromosome-chromosome combination, followed by the number of resolutions stored. Then for each resolution, a header portion with the metadata necessary to read the matrix is followed by the matrix data, stored in block format to ensure good locality. That is, each matrix is subdivided into sub-matrices, called blocks. The "block column count" is the number of columns for the block grid. The "block bin count" is the size of each block in bins, or pixels. Since the blocks are square, their size is blockBinCount x blockBinCount. Finally, each block contains the cell data: the actual count data for that block at that resolution, stored in sparse upper triangular format.

The footer of the *hic* file format contains pointers to each matrix, ensuring fast lookup and direct access to the data without reading through the entire file. It also contains the expected value vectors, which enable calculation of observed/expected for all the matrices. Following the expected value vectors are the normalized expected value vectors, used for the observed/expected for normalized matrices, and the normalization vectors. There are two main types of normalization: VC, or vanilla coverage, and KR, or Knight-Ruiz balanced. However, the format is flexible, and additional types of normalization can easily be added. The normalization vectors are used with the count to calculate and display normalized matrices.

The code for reading and writing the *hic* file format is available at the Juicebox GitHub repository; consult the class DatasetReaderV2 in package juicebox.data for reading and the class Preprocessor in package juicebox.tools.utils.original for writing.

II.k. Availability

Juicer is an open source project available at http://github.com/theaidenlab/juicer. All of the *hic* files from our recent publications have been added to their corresponding GEO accession.

III. Supplemental References

Li H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997v1 [q-bio.GN].

Li H. and Durbin R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler Transform. Bioinformatics, Epub. [PMID: 20080505]

Quinlan, A. R., and Hall, I.M. (2010). BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics, 26(6), 841-842.

Schones, D. E., Smith, A.D., and Zhang, M.Q. (2007). Statistical significance of cis-regulatory modules. BMC bioinformatics 8(1), 19.

Yan, Y., Grossman, M., and Sarkar, V. (2009). JCUDA: A programmer-friendly interface for accelerating Java programs with CUDA. Euro-Par 2009 Parallel Processing. Springer Berlin Heidelberg, 887-899.