Title

HiC-bench: comprehensive and reproducible Hi-C data analysis designed for parameter exploration and benchmarking

Authors

Charalampos Lazaris[1,2,3], Stephen Kelly[4,5], Panagiotis Ntziachristos[6], Iannis Aifantis[1,2*] and Aristotelis Tsirigos[1,2,3,4,5*]

1. Department of Pathology, NYU School of Medicine, New York, NY 10016, USA
2. Laura and Isaac Perlmutter Cancer Center and Helen L. and Martin S. Kimmel Center for Stem Cell Biology, NYU School of Medicine, New York, NY 10016, USA
3. Center for Health Informatics & Bioinformatics, NYU School of Medicine, NY 10016, USA
4. Applied Bioinformatics Center, Office of Science & Research, NYU School of Medicine, NY 10016, USA
5. Genome Technology Center, Office of Science & Research, NYU School of Medicine, NY 10016, USA
6. Department of Biochemistry and Molecular Genetics, Feinberg School of Medicine, Northwestern University, Chicago, IL 60611, USA

E-mail addresses

charalampos.lazaris@med.nyu.edu
stephen.kelly@nyumc.org
panos.ntz@northwestern.edu
ioannis.aifantis@nyumc.org
aristotelis.tsirigos@nyumc.org

* Address correspondence to: Aristotelis Tsirigos (AT) (aristotelis.tsirigos@nyumc.org) or Iannis Aifantis (IA) (ioannis.aifantis@nyumc.org)

Abstract

Chromatin conformation capture techniques have evolved rapidly over the last few years and have provided new insights into genome organization at an unprecedented resolution. Analysis of Hi-C data is complex and computationally intensive involving multiple tasks and requiring robust quality assessment at each step of the analysis. This has led to the development of several tools and methods for processing Hi-C data. However, most of the existing tools do not cover all aspects of the analysis and only offer few quality assessment options. Additionally, availability of a multitude of tools makes

scientists wonder how these tools and associated parameters can be optimally used, and how potential discrepancies can be interpreted and resolved. Most importantly, investigators need to be ensured that slight changes in parameters and/or methods do not affect the conclusions of their studies. Finally, any analysis, no matter how complex, should be reproducible by keeping track of the tool versions, parameters and input data. To address these issues (compare, explore and reproduce), we introduce HiC-bench, a configurable computational platform for comprehensive and reproducible analysis of Hi-C sequencing data. HiC-bench performs all common Hi-C analysis tasks, such as alignment, filtering, contact matrix generation and normalization, identification of topological domains, scoring and annotation of specific interactions using both published tools and our own. We have also embedded various tasks that perform quality assessment and visualization. HiC-bench is implemented as a data flow platform with an emphasis on analysis reproducibility. Additionally, the user can readily perform parameter exploration and comparison of different tools in a combinatorial manner that takes into account all desired parameter settings in each pipeline task. This unique feature facilitates the design and execution of complex benchmark studies that may involve combinations of multiple tool/parameter choices in each step of the analysis. To demonstrate the usefulness of our platform, we performed a comprehensive benchmark of existing and new TAD callers exploring different matrix correction methods, parameter settings and sequencing depths. Users can extend our pipeline by adding more tools as they become available. HiC-bench is distributed as free open-source software on GitHub and Zenodo, and our bioinformatics team offers installation and usage support.

Keywords

Hi-C, Chromosome Conformation, Computational pipeline, Data provenance, Parameter exploration, Benchmarking

Background

Nuclear organization is of fundamental importance to gene regulation. Over the last decade, proximity ligation assays have greatly enhanced our understanding of chromatin organization and its relationship to gene expression [1]. Here we focus on Hi-C, a powerful genome-wide chromosome conformation capture variant, which detects genome-wide chromatin interactions [2,3]. In Hi-C, chromatin is cross-linked and DNA is fragmented using restriction enzymes, the interacting fragments are ligated forming hybrids that are then sequenced and mapped back to the genome. Hi-C is a very powerful technique that has led to important discoveries regarding the organizational principles of the genome. More specifically, Hi-C has revealed that the mammalian genome is organized in active and repressed areas (A and B compartments) [2] that are further divided in “meta-TADs” [4], TADs [5] and sub-TADs [6]. TADs consist evolutionarily conserved, megabase-scale, non-overlapping areas with increased frequency of intra-domain compared to inter-domain chromatin interactions [5,7]. Despite the fact that Hi-C is very powerful, it is known to be prone to systematic biases [8-10]. Moreover, as the sequencing costs plummet allowing for increased Hi-C resolution, Hi-C poses formidable challenges to computational analysis in terms of data storage, memory usage and processing speed. Thus, various tools have been recently developed to mitigate biases in Hi-C data and make Hi-C analysis faster and more efficient in terms of resource usage. HiC-Box [11], hiclib [9] and HiC-Pro [12] perform various Hi-C analysis tasks, such as alignment and binning of Hi-C

sequencing reads into Hi-C contact matrices, noise reduction and detection of specific DNA-DNA interactions. Hi-Corrector [13] has been developed for noise reduction of Hi-C data, allowing parallelization and effective memory management, whereas Hi-Cpipe [14] offers parallelization options and includes steps for alignment, filtering, quality control, detection of specific interactions and visualization of contact matrices. Other tools that allow parallelization are HiFive [15], HOMER [16] and HiC-Pro [12]. Allele-specific Hi-C contact maps can be generated using HiC-Pro and HiCUP [17] (with SNPsplit [18]). TADbit can be used to map raw reads, create interaction matrices, normalize and correct the matrices, call topological domains and build three-dimensional (3D) models based on the Hi-C matrices [19]. HiCdat performs binning, matrix normalization, integration of other data (e.g. ChIP-seq) and visualization [20]. HIPPIE offers similar functionality with HiCdat and allows detection of specific enhancer-promoter interactions [21]. Other tools mainly focus on visualization of Hi-C data (e.g. Sushi [22] and HiCPlotter [23]). Despite the recent boom in the development of computational methods for Hi-C analysis, most of these tools only focus on certain aspects of the analysis, thus failing to encompass the entire Hi-C data analysis workflow. More importantly, these tools or pipelines are not extensible, and, for any given Hi-C task, they do not allow the integration of multiple alternative tools (use of alternative TAD calling methods for example) whose performance could then be qualitatively or quantitatively compared. Available tools do not support comprehensive reporting of the parameters used for each task and they do not enable reproducible computational analysis which is an imperative requirement in the era of big data [24], especially given the complexity of Hi-C analyses. The recently released HiFive is an

exception as it offers a Galaxy interface [15]. However, use of Galaxy [25] can become problematic for data-heavy analyses, especially when the remote Galaxy server is used. To facilitate comprehensive processing, reproducibility, parameter exploration and benchmarking of Hi-C data analyses, we introduce HiC-bench, a data flow platform which is extensible and allows the integration of different task-specific tools. Current and future tools related to Hi-C analysis can be easily incorporated into HiC-bench by implementing simple wrapper scripts. HiC-bench covers all current aspects of a standard Hi-C analysis workflow, including read mapping, filtering, quality control, binning, noise correction and identification of specific interactions (**Table 1**). Moreover, it integrates multiple alternative tools for performing each task (such as matrix correction tools and TAD-calling algorithms), while at the same time allowing simultaneous exploration of different parameter settings that are propagated from one task to all subsequent tasks in the pipeline. HiC-bench also generates a variety of quality assessment plots and offers other visualization options, such as generating genome browser tracks as well as snapshots using HiCPlotter. We have built this platform with reproducibility in mind, as all tools, versions and parameter settings are recorded throughout the analysis. HiC-bench is released as open-source software and the source code is available on GitHub and Zenodo (for details please refer to “Availability of data and material” section). Our team provides installation and usage support.

Methods

The HiC-bench workflow

128 HiC-bench is a comprehensive computational pipeline for Hi-C sequencing data analysis.
129 It covers all aspects of Hi-C data analysis, ranging from alignment of raw reads to
130 boundary-score calculation, TAD calling, boundary detection, annotation of specific
131 interactions and enrichment analysis. Thus, HiC-bench consists the most complete
132 computational Hi-C analysis pipeline to date (**Table 1**). Importantly, every step of the
133 pipeline includes summary statistics (when applicable) and direct comparative
134 visualization of the results. This feature is essential for quality control and facilitates
135 troubleshooting. The HiC-bench workflow (**Figure 1**) starts with the alignment of Hi-C
136 sequencing reads and ends with the annotation and enrichment of specific interactions.
137 More specifically, in the first step, the raw reads (fastq files) are aligned to the reference
138 genome using Bowtie2 [26] (*align*). The aligned reads are further filtered in order to
139 determine those Hi-C read pairs that will be used for downstream analysis (*filter*). A
140 detailed statistics report showing the numbers and percentages of reads assigned to the
141 different categories is automatically generated in the next step (*filter-stats*). The reads
142 that satisfy the filtering criteria are used for the creation of Hi-C contact matrices (*matrix-*
143 *filtered*). These contact matrices can either be directly visualized in the WashU
144 Epigenome Browser [27] as Hi-C tracks (*tracks*), or further processed using three
145 alternative matrix correction methods: (a) matrix scaling (*matrix-prep*), (b) iterative
146 correction (*matrix-ic*) [9] and (c) HiCNorm (*matrix-hicnorm*) [28]. As quality control, plots
147 of the average number of Hi-C interactions as a function of the distance between the
148 interacting loci are automatically generated in the next step (*matrix-stats*). The Hi-C
149 matrices, before and after matrix correction, are used as inputs in various subsequent
150 pipeline tasks. First, they are directly compared in terms of Pearson or Spearman

correlation (*compare-matrices* and *compare-matrices-stats*) in order to estimate the similarity between Hi-C samples. Second, they are used for the calculation of boundary scores (*boundary-scores* and *boundary-scores-pca*), identification of topological domains (*domains*) and comparison of boundaries (*compare-boundaries* and *compare-boundaries-stats*). Third, high-resolution Hi-C matrices are used for detection and annotation of specific chromatin interactions (*interactions* and *annotations*), enrichment analysis in transcription factors, chromatin marks or other segmented data (*annotation-stats*) and visualization of chromatin interactions in certain genomic loci of interest (*hicplotter*). We should note here that HiC-bench is totally extensible and customizable as new tools can be easily integrated into the HiC-bench workflow (see User Manual for more details). In addition to the multiple alternative tools that can be used to perform certain tasks, HiC-bench allows simultaneous exploration of different parameter settings that are propagated from one task to all subsequent tasks in the pipeline (for details please refer to "Main concepts and pipeline architecture" section). For example, after contact matrices are generated and corrected using alternative methods, HiC-bench proceeds with TAD calling using all computed matrices as inputs (**Figure 1** and **Figure 2A**). This unique feature enables the design and execution of complex benchmark studies that may include combinations of multiple tool/parameter choices in each step. HiC-bench focuses on the reproducibility of the analysis by keeping records of the source code, tool versions and parameter settings, and it is the only HiC-analysis pipeline that allows combinatorial parameter exploration facilitating benchmarking of Hi-C analyses.

Table 1. Comparison of HiC-bench with published Hi-C analysis or visualization tools.

Hi-C tasks	HiC-bench	HiFive	Hi-Cpipe	HiCNorm	hiclib	HiTC	HOMER	Hi-Corrector	HiC-Pro	TADbit	HiCUP	HiC-Box	HiCdat	HIPPIE	Sushi	HiCPlotter
alignment	x		x		x				x	x	x	x		x		
filtering	x	x	x		x				x	x	x	x		x		
genome browser tracks	x															
quality assessment plots	x	x	x			x			x		x		x	x		
contact matrices	x	x	x		x		x		x			x	x			
matrix correction	x	x		x	x	x	x	x	x	x		x	x			
matrix comparison	x												x			
boundary scores	x															
domains	x									x						
boundary comparison	x															
specific interactions	x		x		x		x		x		x	x	x	x		
annotations	x					x							x			
allele-specific interactions									x		x					
visualization	x	x	x			x	x						x		x	x
integration with ChIP-seq data	x												x	x		
parallelization	x	x	x				x	x	x	x						
integration of alternative tools	x															
parameter exploration	x															
reproducibility	x	x														

The HiC-bench toolkit

HiC-bench performs various tasks of Hi-C analysis ranging from read alignment to annotation of specific interactions and visualization. We have developed two new tools, *gtools-hic* and *hic-matrix*, to execute the multiple tasks in the HiC-bench pipeline, but we have also integrated existing tools to allow comparative and complementary analyses and

180 facilitate benchmarking. More specifically, the alignment task is performed either with
181 Bowtie2 [26] or with the “align” function of *gtools-hic*, our newest addition to
182 GenomicTools [29]. Likewise, filtering, creation of Hi-C tracks and generation of Hi-C
183 contact matrices are performed using the functions “filter”, “bin/convert” and “matrix” of
184 *gtools-hic* respectively. For advanced users, we have implemented a series of novel
185 features for these common Hi-C analysis tasks. For example, the operation “matrix” of
186 *gtools-hic* allows generation of arbitrary chimeric Hi-C contact matrices, a feature
187 particularly useful for the study of the effect of chromosomal translocations on chromatin
188 interactions. Another example is the generation of distance-restricted matrices (up to
189 some maximum distance off the diagonal) in order to save storage space and reduce
190 memory usage at fine resolutions. For matrix correction we use either published
191 algorithms (iterative correction (IC/ICE) [9], HiCNorm [28]) or our “naïve scaling” method
192 where we divide the Hi-C counts by (a) the total number of (usable) reads, and (b) the
193 “effective length” [8,28] of each genomic bin. We also integrated published TAD callers
194 like DI [5], Armatus [30], TopDom [31], insulation index (Crane) [32] and our own TAD
195 calling method (similar but not identical to contrast index [33,34]) implemented as the
196 “domains” operation in *hic-matrix*. Additionally, the “domains” operation produces
197 genome-wide boundary scores using multiple methods and allowing flexibility in choosing
198 parameters. Boundaries are simply defined as local maxima of the boundary scores. For
199 the detection of specific interactions, we introduce the “loops” function of *hic-matrix*, while
200 GenomicTools is used for annotation of these interactions with gene names, ChIP-seq
201 and other user-defined data. Finally, we implemented a wrapper for HiCPlotter, taking
202 advantage of its advanced visualization features in order to allow the user to quickly

generate snapshots of areas of interest in batch. The HiC-bench toolkit is summarized in **Table 2**. All the tools we developed appear in bold. Further information on the toolkit is provided in the User Manual found online and in the Supplemental Information section.

Table 2. The HiC-bench toolkit. The HiC-bench toolkit consists mostly of newly-developed tools (shown in bold) but we have also incorporated existing tools to allow comparisons and benchmarking.

Hi-C tasks	**HiC-bench toolkit**
alignment	bowtie2, **gtools-hic[align]**
filtering	**gtools-hic[filter]**
genome browser tracks	**gtools-hic[bin/convert]**
matrix generation	**gtools-hic[matrix]**
matrix correction	IC, HiCNorm, **hic-matrix[preprocess/normalize]**
boundary scores	**hic-matrix[domains]**
domain calling	DI, Armatus, TopDom, **hic-matrix[domains]**
interactions	**hic-matrix[loops]**
annotations	**genomic-tools**
visualization	HiCPlotter

Main concepts and pipeline architecture

We built our platform based on principles outlined in scientific workflow systems such as Kepler [35], Taverna [36] and VisTrails [37]. The main idea behind our platform is the ability to track data provenance [37,38], the origin of the data, computational tasks, tool versions and parameter settings used in order to generate a certain output (or collection of outputs) from a given input (or collection of inputs). Thus, our pipeline ensures reproducibility which is a particularly important feature for such a complex computational task. In addition, HiC-bench enables combinatorial analysis and parameter exploration by implementing the idea of computational “trails”: a unique combination of inputs, tools and parameter values can be imagined as a unique (computational) trail that is followed simultaneously with all the other possible trails in order to generate a collection of output

222 objects (**Figure 2A**). Our platform consists of three main components: (a) data, (b) code
223 and (c) pipelines. These components are organized in respective directories in our local
224 repository, and synchronized with a remote GitHub repository for public access. The data
225 directory is used to store data that would be used by any analysis, for example genome-
226 related data, such as DNA sequences and indices (e.g. Bowtie2), gene annotations and,
227 in general, any type of data that is required for the analysis. The code directory is used to
228 store scripts, source code and executables. More details about the directory structure can
229 be found in the User Manual. Finally, the "pipelines" directory is used to store the structure
230 of each pipeline. Here, we will focus on our Hi-C pipeline, but we have also implemented
231 a ChIP-seq pipeline, which is very useful for integrating CTCF and histone modification
232 ChIP-seq data with Hi-C data. The structure of the pipeline is presented to the user as a
233 numbered list of directories, each one corresponding to one operation (or task) of the
234 pipeline. As shown in **Figure 1**, our Hi-C pipeline currently consists of several tasks
235 starting with alignment and reaching completion with the identification and annotation of
236 specific DNA-DNA interactions and annotations with ChIP-seq and other genome-wide
237 data (see also **Table 2** and **Supplemental Table 1**). We will examine these tasks in detail
238 in the Results section of this manuscript.
239
240 **Parameter exploration, input and output objects**
241 In conventional computational pipelines, several computational tasks (operations) are
242 executed on their required inputs. However, in existing genomics pipelines, each task
243 generates a single result object (e.g. TAD calling using one method with fixed parameter
244 settings) which is then used by downstream tasks. To allow full parameter (and
245 method/tool) exploration, we introduce instead a data flow model, where every task may

accommodate an arbitrary number of output objects. Downstream tasks will then operate on all computed objects generated by the tasks they depend on. Pipeline tasks are implemented as shown in the diagram of **Figure 2B**. First, input objects are filtered according to user-specified criteria (e.g. TAD calling is only done for Hi-C contact matrices at 40kb resolution). Then, *pipeline-master-explorer* (implemented as an R script; see User Manual for usage and input arguments) generates the commands that create all desired output objects. In principle, all combinations of input objects with all parameter settings will be created, subject to user-defined filtering criteria. In the interest of extensibility, new pipeline tasks can be conveniently implemented using a single-line *pipeline-master-explorer* command (see **Supplemental Table 2**), provided that wrapper scripts for each task (e.g. TAD calling using TopDom) have been properly set up. In the simplest scenario, any task in our pipeline will generate computational objects for each combination of parameter file and input objects obtained from upstream tasks. For example, suppose the aligned reads from 12 Hi-C datasets are filtered using three different parameter settings, and that we need to create contact matrices at four resolutions (1Mb, 100kb, 40kb and 10kb). Then, the number of output objects (contact matrices in this case) will be 144 (i.e. 12 x 3 x 4). Although many computational scenarios can be realized by this simple one-to-one mapping of input-output objects, more complex scenarios are frequently encountered, as described in the next section.

Filtering, splitting and grouping input objects into new output objects

Oftentimes, a simple one-to-one mapping of input objects to output objects is not desirable. For this reason, we introduce the concepts of filtering, splitting and grouping of input objects which are used to modify the behavior of pipeline-master-explorer (see

270 **Figure 2B**). *Filtering* is required when some input objects are not relevant for a given
271 task, e.g. TAD calling is not performed on 1Mb-resolution contact matrices, and specific
272 DNA-DNA interactions are not meaningful for resolutions greater than 10-20kb. *Splitting*
273 is necessary in some cases: for example, we split the input objects by genome assembly
274 (hg19, mm10) when comparing contact matrices or domains across samples, since only
275 matrices or domains from the same genome assembly can be compared directly. In our
276 platform, the user is allowed to split a collection of input objects by any variable contained
277 in the sample sheet (except fastq files), thus allowing user-defined splits of the data, such
278 as by cell type or treatment. Complementary to the splitting concept, *grouping* permits the
279 aggregation of a collection of input objects (sharing the same value of a variable defined
280 in the sample sheet) into a single output object. For example, the user may want to create
281 genome browser tracks or contact matrices of combined technical and/or biological
282 replicates, or group all input objects (samples) together in tasks such as Principal
283 Components Analysis (PCA) or alignment/filtering statistics.
284
285 **Combinatorial objects**
286 Even after introducing the concepts described above, more complex scenarios are
287 possible as some tasks require the input of pairs (or triplets etc.) of objects. This feature
288 has also been implemented in our pipeline (tuples in **Figure 2B**) and is currently used in
289 the *compare-matrices* and *compare-boundaries* tasks. However, it should be utilized
290 wisely (for example in conjunction with filtering, splitting and grouping) because it may
291 lead to a combinatorial "explosion" of output objects.
292
293 **Parameter scripts**

294 The design of our platform is motivated by the need to facilitate the use of different
295 parameter settings for each pipeline task. For this reason, we have implemented wrapper
296 scripts for each tool/method used in each task. For example, we have implemented a
297 wrapper script for alignment, filtering, correcting contact matrices using IC or HiCNorm
298 (separate wrappers), TAD calling using Armatus [30], TopDom [31], DI [5] and insulation
299 index (Crane) [32] (separate wrappers). The main motivation is to hide most of the
300 complexity inside the wrapper script and allow the user to modify the parameters using a
301 simple but flexible parameter script. Unlike static parameter files, parameter scripts allow
302 for dynamic calculation of parameters based on certain input variables (e.g. enzyme
303 name, group name etc.). Within this framework, by adding and/or modifying simple
304 parameter scripts, the user can explore the effect of different parameters (a) on the task
305 directly affected by these parameters, and (b) on all dependent downstream tasks.
306 Additionally, these parameter scripts serve as a record of parameters and tool versions
307 that were used to produce the results, facilitating analysis reproducibility as well as
308 documentation in scientific reports and manuscripts.
309
310 **Results stored as computational trails**
311 All the concepts described above have been implemented in a single R script named
312 *pipeline-master-explorer*. This script maintains a database of input-output objects for each
313 task, stored in a hidden directory under results (results/.db). It also creates a “run” script
314 which is executed in order to generate all the desired results. All results are stored in the
315 results directory in a tree structure that reveals the computational trail for each object (see
316 examples shown in **Figure 2B** and **Supplementary Table 2**). Therefore, the user can

easily infer how each object was created, including what inputs and what parameters were used.

Initiating a new reproducible analysis

In the interest of data analysis reproducibility, any new analysis requires creating a copy of the code and pipeline structure into a desired location, effectively creating a branch. This way, any changes in the code repository will not affect the analysis and conversely, the user can customize the code according to the requirements of each project without modifying the code repository. Copying of the code and initiating a new analysis is done simply by invoking the script "*pipeline-new-analysis.tcsh*" as described in the User Manual.

Pipeline tasks

A pipeline consists of a number of (partially) ordered tasks that can be described by a directed acyclic graph which defines all dependencies. HiC-bench implements a total of 20 tasks as shown in the workflow of **Figure 1**. In the analysis directory structure, each task is assigned its own subdirectory found inside the pipeline directory starting from the top level. This directory includes a symbolic link to the inputs of the analysis (fastq files, sample sheet, etc.), a link to the code, a directory (*inpdirs*) containing links to all dependencies, a directory containing parameter scripts (see below) and a "*run*" script which can be used to generate all the results of this task. The "*run*" scripts of each task are executed in the specified order by the master "*run*" script located at the top level (see User Manual for details on pipeline directory structure).

341 **Input data and the sample sheet**

342 Before performing any analysis, a computational pipeline needs input data. All input data
343 for our pipeline tasks are stored in their own "*inputs*" directory accessible at the top level
344 (along with the numbered pipeline tasks) and via symbolic links from within the directories
345 assigned to each task to allow easy access to the corresponding input data. A "*readme*"
346 file explains how to organize the input data inside the inputs directory (see User Manual
347 for details). Briefly, the *fastq* subdirectory is used to store all fastq files, organized into
348 one subdirectory per sample. Then, the sample sheet needs to be generated. This can
349 be done automatically using the "*create-sample-sheet.tcsh*" script, but the user can also
350 manually modify and expand the sample sheet with features beyond what is required.
351 Currently required features are the sample name (to be used in all downstream analyses),
352 fastq files (R1 and R2 in separate columns), genome assembly version (e.g. hg19, mm10)
353 and restriction enzyme name (e.g. HindIII, NcoI). Adding more features, such as different
354 group names (e.g. sample, cell type, treatment), allows the user to perform more
355 sophisticated downstream analyses, such as grouping replicates for generating genome
356 browser tracks, or splitting samples by genome assembly to compare boundaries (see
357 previous section on grouping and splitting).

358

359 **Executing the pipeline**

360 The entire pipeline can be executed automatically by the "*pipeline-execute.tcsh*" script,
361 as shown below:

362 **code/code.main/pipeline-execute <project name> <user e-mail address>**

363 where <project name> will be substituted by the name of the project and <user e-mail
364 address> by the preferred e-mail address of the person who runs the analysis in order to

be notified upon completion. The "*pipeline-execute.tcsh*" script essentially executes the "run" script for each task (following the specified order). At the completion of every task, the log files of all finished jobs are inspected for error messages. If error messages are found, the pipeline aborts with an error message.

Timestamping

Besides creating the "*run*" script used to generate all results, the "*pipeline-master-explorer.r*" script, also checks whether existing output objects are up-to-date when compared to their dependencies (i.e. input objects and parameter scripts; can be expanded to include code dependencies as well). Currently, the pipelines are setup so that out-of-date objects are not deleted and recomputed automatically, but only presented to the user as a warning. The user can then choose to delete them manually and re-compute. The reason for this is to protect the user against accidentally repeating computationally demanding tasks (e.g. alignments) without given first the chance to review why certain objects may be out-of-date. From a more philosophical point of view, and in the interest of keeping a record of all computations (when possible), the user may never want to modify parameter files or the code for a given project, but instead only add new parameter files. Then, no object will be out-of-date, and only new objects will need to be recomputed every time.

Alignment and filtering

Paired-end reads were mapped to the reference genome (hg19 or mm10) using Bowtie2 [26]. Reads with low mapping quality (MAPQ<30) were discarded. Local alignments of input read pairs were performed as they consist of chimeric reads between two (non-

389 consecutive) interacting fragments. This approach yielded a high percentage of mappable
390 reads (> 95%) for all datasets (**Supplementary Figure 1**). Mapped read pairs were
391 subsequently filtered for known artifacts of the Hi-C protocol such as self-ligation,
392 mapping too far from the enzyme's known cutting sites etc. More specifically, reads
393 mapping in multiple locations on the reference genome (*multihit*), double-sided reads that
394 mapped to the same enzyme fragment (*ds-same-fragment*), reads whose 5'-end mapped
395 too far (*ds-too-far*) from the enzyme cutting site, reads with only one mappable end
396 (*single-sided*) and unmapped reads (*unmapped*), were discarded. Read pairs that
397 corresponded to regions that were very close (less than 25 kilobases, *ds-too-close*) in
398 linear distance on the genome as well as duplicate read pairs (*ds-duplicate-intra* and *ds-*
399 *duplicate-inter*) were also discarded. In **Supplementary Figure 1**, we show detailed
400 paired-end read statistics for the Hi-C datasets used in this study. We include the read
401 numbers (**Supplementary Figure 1A**) and their corresponding percentages
402 (**Supplementary Figure 1B**). Eventually, approximately 10-50% of paired-reads passed
403 all filtering criteria and were used for downstream analysis (**Supplementary Figure 1B**).
404 The statistics report is automatically generated for all input samples. The tools and
405 parameter settings used for the alignment and filtering tasks are fully customizable and
406 can be defined in the corresponding parameter files.
407
408 **Contact matrix generation, normalization and correction**
409 The read-pairs that passed the filtering task were used to create Hi-C contact matrices
410 for all samples. The elements of each contact matrix correspond to pairs of genomic
411 "bins". The value in each matrix element is the number of read pairs aligning to the
412 corresponding genomic regions. In this study, we used various resolutions, ranging from

413 fine (10kb) to coarse (1Mb). The resulting matrices either remained unprocessed
414 (filtered), or they were processed using different correction methods including HiCNorm
415 [28], iterative correction (IC or ICE) [9] as well as "naïve scaling". In **Supplementary**
416 **Figure 2**, we present the average Hi-C count as a function of the distance between the
417 interacting fragments, separately for each Hi-C matrix for not corrected (filtered) and IC-
418 corrected matrices.

419

420 **Comparison of contact matrices**

421 Our pipeline allows direct comparison and visualization of the generated Hi-C contact
422 matrices. More specifically, using our *hic-matrix* tool, all pairwise Pearson and Spearman
423 correlations were automatically calculated for each (a) input sample, (b) resolution, and
424 (c) matrix correction method. The corresponding correlograms were automatically
425 generated using the corrgram R package [39]. A representative example is shown in
426 **Supplementary Figure 3**. The correlograms summarizing the pairwise Pearson
427 correlations for all samples used in this study are presented before and after matrix
428 correction using the iterative correction algorithm. These plots are very useful because
429 the user can quickly assess the similarity between technical and biological replicates as
430 well as differences between various cell types. As shown before (Supplementary Figure
431 3 in [5]), iterative correction improves the correlation between enzymes at the expense of
432 a decreased correlation between samples prepared using the same enzyme.

433

434 **Boundary scores**

435 Topological domains (TADs) are defined as genomic neighborhoods of highly interacting
436 chromatin, with relatively more infrequent inter-domain interactions [5,40,41]. Topological

437 domains are demarcated by boundaries, i.e. genomic regions bound by insulators thus
438 hampering DNA contacts across adjacent domains. For each genomic position, in a given
439 resolution (typically 40kb or less), we define a "boundary score" to quantify the insulation
440 strength of this position. The higher the boundary score, the higher the insulation strength
441 and the probability that this region actually acts as a boundary between adjacent domains.
442 The idea of boundary scores is further illustrated in **Supplementary Figure 4**, where two
443 adjacent TADs are shown. The upstream TAD on the left (*L*) is separated from the
444 downstream TAD on the right (*R*) by a boundary (black circle). We define two parameters,
445 the distance from the diagonal of the Hi-C contact matrix to be excluded from the
446 boundary score calculation (δ) (not shown) and the maximum distance from the diagonal
447 to be considered (*d*). The corresponding parameter values can be selected by the user.
448 For this analysis, we used δ=0 and *d*=2Mb as suggested before [5]. In addition to the
449 published directionality index [5], we defined and computed the "*inter*", "*intra-max*" and
450 "*ratio*" scores, defined as follows:

451 $$\text{inter} = \text{mean}(X)$$
452 $$\text{intra}_{max} = \max(\text{mean}(L), \text{mean}(R))$$
453 $$\text{ratio} = \text{intra}_{max}/\text{inter}$$
454
455 Principal component analysis (PCA) of boundary scores across samples in this study,
456 before and after matrix correction, shows that biological replicates tend to cluster
457 together, either in the case of filtered or corrected (IC) matrices (**Supplementary Figure**
458 **5**).
459
460 **Topological domains**
461 Topologically-associated domains (TADs) are increasingly recognized as an important
462 feature of genome organization [5]. Despite the importance of TADs in genome

organization, very few Hi-C pipelines have integrated TAD calling (e.g. TADbit [19]). In HiC-bench, we have integrated TAD calling as a pipeline task and we demonstrate this integration using different TAD callers: (a) Armatus [30], (b) TopDom [31], (c) DI [5], (d) insulation index (Crane) [32] and (c) our own hic-matrix (domains). Our pipeline makes it straightforward to plug in additional TAD callers, by installing these tools and setting up the corresponding wrapper scripts. HiC-bench also facilitates the direct comparison of TADs across samples by automatically calculating the number of TAD boundaries and all the pairwise overlaps of TAD boundaries across all inputs, generating the corresponding graphs (as in the case of matrix correlations described in a previous section). We define boundary overlap as the ratio of the intersection of boundaries between two replicates (A and B) over the union of boundaries in these two replicates, as shown in the equation below:

$$\text{boundary_overlap} = (A \cap B)/(A \cup B)$$

For the boundary overlap calculation, we extended each boundary by 40kb on both sides (+/- 40kb flanking region, i.e. the size of one bin). The fact that HiC-bench allows simultaneous exploration of all parameter settings for all installed TAD-calling algorithms, greatly facilitates parameter exploration, optimization as well as assessment of algorithm effectiveness. Pairwise comparison of boundaries (boundary overlaps) across samples is shown in **Supplementary Figure 6** and **Figure 3**.

Visualization

In our pipeline, we also take advantage of the great visualization capabilities offered by the recently released HiCPlotter [23], in order to allow the user to visualize Hi-C contact

486 matrices along with TADs (in triangle format) for multiple genomic regions of interest. The
487 user can also add binding profiles in BedGraph format for factors (e.g. CTCF), boundary
488 scores, histone marks of interest (e.g. H3K4me3, H3K27ac) etc. An example is shown in
489 **Supplementary Figure 7**, where an area of the contact matrix of human embryonic stem
490 cells (H1) (HindIII) is presented along with the corresponding TADs (triangles), various
491 boundary scores, the CTCF binding profile and annotations of selected genomic
492 elements, before and after matrix correction (IC). The integration of HiCPlotter in our
493 pipeline, allows the user to easily create publication-quality figures for multiple areas of
494 interest simultaneously.
495
496 **Specific interactions, annotations and enrichments**
497 The plummeting costs of next-generation sequencing have resulted in a dramatic
498 increase in the resolution achieved in Hi-C experiments. While the original Hi-C study
499 reported interaction matrices of 1Mb resolution [2], recently 1kb resolution was reported
500 [42]. Thus, the characterization and annotation of specific genomic interactions from Hi-
501 C data is an important feature of a modern Hi-C analysis pipeline. HiC-bench generates
502 a table of the interacting loci based on parameters defined by the user. These parameters
503 include the resolution, the lowest number of read pairs required per interacting area as
504 well as the minimum distance between the interacting partners. The resulting table
505 contains the coordinates of the interacting loci, the raw count of interactions between
506 them, the number of interactions after “scaling” and the number of interactions between
507 the partners after distance normalization (observed Hi-C counts normalized by expected
508 counts as a function of distance). This table is further annotated with the gene names or
509 the factors (e.g. CTCF) and histone modification marks (e.g. H3K4me1, H3K27ac,

510 H3K4me3) that overlap with the interacting loci. The user can provide bed files with the
511 features of interest to be used for annotation. As an example, the enrichment of chromatin
512 marks in the top 50000 chromatin interactions in the H1 and IMR90 samples is presented
513 in **Supplementary Figure 8**.
514
515 **Software requirements**
516 The software requirements are: Bowtie2 aligner [26], Python (2.7 or later) (along with
517 Numpy, Scipy and Matplotlib libraries), R (3.0.2) [43], various R packages (lattice,
518 RColorBrewer, corrplot, reshape, gplots, preprocessCore, zoo, reshape2, plotrix,
519 pastecs, boot, optparse, ggplot2, igraph, Matrix, MASS, flsa, VennDiagram, futile.logger
520 and plyr) and HiCPlotter [23]. More details on the versions of the packages can be found
521 in the User Manual (sessionInfo()). In addition, installation of mirnylib Python library [44]
522 is required for matrix balancing based on IC (ICE). The pipeline has been tested on a
523 high-performance computing cluster based on Sun Grid Engine (SGE). The operating
524 system used was Redhat Linux GNU (64 bit).

525 ## Results

526 We used HiC-bench to analyze several published Hi-C datasets and the results of our
527 analysis are presented below. Additionally, we performed a comprehensive benchmark
528 of existing and new TAD callers exploring different matrix correction methods, parameter
529 settings and sequencing depths. Our results can be reproduced by re-running the
530 corresponding pipeline snapshot available upon request as a single compressed archive
531 file (too big to include as a Supplemental file).
532

533 **Comprehensive reanalysis of available Hi-C datasets using HiC-bench**

534 Our platform is designed to facilitate and streamline the analysis of a large number of
535 available Hi-C datasets in batch. Thus, we collected and comprehensively analyzed
536 multiple Hi-C samples from three large studies [5,42,45]. From the first study we analyzed
537 IMR90 (HindIII) samples, from the second we analyzed Hi-C samples from
538 lymphoblastoid cells (GM12878), human lung fibroblasts (IMR90 (MboI)),
539 erythroleukemia cells (K562), chronic myelogenous leukemia (CML) cells (KBM-7) and
540 keratinocytes (NHEK), and from the third one, we analyzed samples from human
541 embryonic stem cells (H1) and all the embryonic stem-cell derived lineages mentioned,
542 including mesendoderm, mesenchymal stem cells, neural progenitor cells and
543 trophectoderm cells. All datasets yielded at least 40 million usable intra-chromosomal
544 read pairs in at least two biological replicates. We performed extensive quality control on
545 all datasets, calculating the read counts and percentages per classification category
546 (**Supplementary Figure 1**), the attenuation of Hi-C signal over genomic distance
547 (**Supplementary Figure 2**), the correlation of Hi-C matrices before and after matrix
548 correction (**Supplementary Figure 3**), the similarity of boundary scores (**Supplementary**
549 **Figure 5**) and all pairwise boundary overlaps across samples (**Supplementary Figure**
550 **6**). In addition, we performed a comprehensive benchmarking of our own and published
551 TAD callers, across all reanalyzed Hi-C datasets. The results of our benchmark are
552 presented in the following sections.

553

554 **Iterative correction of Hi-C contact matrices improves reproducibility of TAD**
555 **boundaries**
556

557 Iterative correction has been shown to correct for known biases in Hi-C [9]. Thus, we
558 hypothesized that IC may increase the reproducibility of TAD calling. We performed a
559 comprehensive analysis calculating the TAD boundary overlaps for biological replicates
560 of the same sample (as described in Methods section), using different TAD callers and
561 different main parameter values for each TAD caller (**Figure 3A**). After comparing TAD
562 boundary overlaps between filtered (uncorrected) and IC-corrected matrices, we
563 observed an improvement in the boundary overlap when corrected matrices were used,
564 irrespective of TAD caller and parameter settings (**Figure 3B)**. The only exception was
565 DI. Careful examination of the overlaps per sample revealed that IC introduced outliers
566 only in the case of DI (in general, the opposite was true for the other callers). We
567 hypothesize that IC may occasionally negatively affect the computation of the
568 directionality index, especially because its calculation depends on a smaller number of
569 bins (1-dimensional line) compared to the rest of the methods (2-dimensional triangles).
570 In addition to the increase in the mean value of boundary overlap upon correction, we
571 observed that the standard deviation of boundary overlaps among replicates decreased
572 (again, with the exception of DI) (**Figure 3C**). While this seems to be the trend for almost
573 all TAD caller/parameter value combinations, the effect of correction in variance is more
574 profound in certain cases (e.g. hicintra-max) than others. It is also worth mentioning that
575 increased size of the insulation window (in the case of Crane), the resolution parameter
576 γ (Armatus) or the distance *d* (hicinter, hicintra-max, hicratio) may result in certain cases
577 in increased boundary overlap (e.g. Armatus), but this is not generalizable (e.g. hicintra-
578 max). Interestingly, increased TAD boundary overlap does not necessarily mean
579 increased consistency in the number of predicted TADs across sample types, as would

580 be expected since TADs are largely invariant across cell types [5]. For example, the TAD
581 calling algorithm which is based on insulation score (Crane), predicted similar TAD
582 overlaps and similar TAD numbers for different insulation windows (ranging from 0.5Mb
583 to 2Mb), whereas Armatus performed particularly well in terms of TAD boundary
584 reproducibility for a wide range of settings (**Figure 3A**) but the corresponding predicted
585 TAD numbers varied considerably (**Figure 3D**). This may be partly due to the nature of
586 the Armatus algorithm, as it has been built to reveal multiple levels of chromatin
587 organization (TADs, sub-TADs etc.). We conclude that while iterative correction improves
588 the reproducibility of TAD boundary detection across replicates, the number of predicted
589 TADs should be also taken into account when selecting TAD calling method for
590 downstream analysis. The method of choice should be the one that is robust in terms of
591 both reproducibility and number of predicted TADs.

592

593 **Increased sequencing depth improves the reproducibility of TAD boundaries**

594

595 After selecting the parameter setting that optimized TAD boundary overlap between
596 biological replicates of the same sample per TAD caller, we also investigated the effect
597 of sequencing depth on the reproducibility of TAD boundary detection. Since some of the
598 input samples were limited to only 40 million usable intra-chromosomal Hi-C read pairs,
599 we resampled 10 million, 20 million and 40 million read pairs from each sample and
600 evaluated the effect of increasing sequencing depth on TAD boundary reproducibility. The
601 results are depicted in **Figure 4A**. We noticed that increased sequencing depth resulted
602 in increased TAD boundary overlap, regardless of the TAD calling algorithm used (**Figure**
603 **4A,C**). As far as the TAD numbers are concerned, increased sequencing depth
604 decreased TAD number variability for certain callers (e.g. hicratio) but not in all cases

605 (e.g. DI) (**Figure 4B**). In many cases, increased sequencing depth, decreased the
606 variance of TAD boundary overlap among replicates (**Figure 4C**). In summary, based on
607 this benchmark, we recommend that Hi-C samples should be sufficiently sequenced as
608 sequencing depth seems to affect TAD calling reproducibility.
609

610 Conclusions

611 Recently, several computational tools and pipelines have been developed for Hi-C
612 analysis. Some focus on matrix correction, others on detection of specific chromatin
613 interactions and their differences across conditions and others on visualization of these
614 interactions. However, very few of these tools offer a complete Hi-C analysis (e.g. HiFive,
615 HiCUP or HiC-Pro), addressed all tasks ranging from alignment to interaction annotation,
616 enrichment analysis and visualization. HiC-bench is a comprehensive Hi-C analysis
617 pipeline with the ability to process many samples in parallel, record and visualize the
618 results in each task, thus facilitating troubleshooting and further analyses. It integrates
619 both existing tools but also new tools that we developed to perform certain Hi-C analysis
620 tasks. In addition, HiC-bench focuses on parameter exploration, reproducibility and
621 extensibility. All parameter settings used in each pipeline task are automatically recorded,
622 while future tools can be easily added using the supplied wrapper template. More
623 importantly, HiC-bench is the only Hi-C pipeline so far that allows extensive parameter
624 exploration, thus facilitating direct comparison of the results obtained by different tools,
625 methods and parameters. This unique feature helps users test the robustness of the
626 analysis, optimize the parameter settings and eventually obtain reliable and biologically
627 meaningful results. To demonstrate the usefulness of HiC-bench, we performed a
628 comprehensive benchmark of popular and newly-introduced TAD callers, varying the

matrix preprocessing (filtered or corrected matrices with ICE method), the sequencing depth, and the value of the main parameter of each TAD caller, which is usually the window used for the calculation of directionality index or insulation score. We found that the matrix correction has a positive effect on the boundary overlap between replicates and that increased sequencing depth leads to higher boundary overlap.

In conclusion, HiC-bench is an easy-to-use framework for systematic, comprehensive, integrative and reproducible analysis of Hi-C datasets. We expect that use of our platform will facilitate current analyses and enable scientists to further develop and test interesting hypotheses in the field of chromatin organization and epigenetics.

List of abbreviations

DI: directionality index

IC or ICE: iterative correction

PCA: Principal Component Analysis

TAD: Topological Domain or Topologically Associating Domain

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and material

Published Hi-C data were downloaded from Gene Expression Omnibus, using the corresponding accession numbers: GSE35156 [5], GSE63525 [42] and GSE52457 [45].

HiC-bench source code is freely available on GitHub and Zenodo.

Project name: HiC-bench

Project home page: https://github.com/NYU-BFX/hic-bench/wiki

Archived version: https://zenodo.org/badge/latestdoi/20915/NYU-BFX/hic-bench

Operating system: Redhat Linux GNU (64 bit)

Programming language: R, C++, Python, Unix shell scripts

Other requirements: none

License: MIT

Any restrictions to use by non-academics: None

Competing interests

The authors have no competing interests to declare.

Funding

The study was supported by a Research Scholar Grant, RSG-15-189-01 - RMC from the American Cancer Society and a Leukemia & Lymphoma Society New Idea Award, 8007-17 to Aristotelis Tsirigos (AT). NYU Genome Technology Center (GTC) is a shared resource, partially supported by the Cancer Center Support Grant, P30CA016087, at the Laura and Isaac Perlmutter Cancer Center

Author contributions

674 CL performed computational analyses, generated figures and implemented certain
675 wrapper scripts. SK wrote the user manual. PN and IA offered biological insights and
676 helped with the interpretation of Hi-C data. AT designed and implemented the pipeline.
677 CL and AT wrote the manuscript.
678

679 ## Acknowledgements

680 Aristotelis Tsirigos was supported by a Research Scholar Grant, RSG-15-189-01 - RMC
681 from the American Cancer Society and a Leukemia & Lymphoma Society New Idea
682 Award, 8007-17. We would like to thank Dennis Shasha and Juliana Freire for inspiring
683 discussions on data flows. We would also like to thank Kadir Caner Akdemir for useful
684 discussions on the usage of HiCPlotter. We thank the NYU Genome Technology Center
685 (GTC) for expert library preparation and sequencing. This shared resource is partially
686 supported by the Cancer Center Support Grant, P30CA016087, at the Laura and Isaac
687 Perlmutter Cancer Center. We also like to thank the NYU Applied Bioinformatics Center
688 (BFX) for providing bioinformatics support and helping with the analysis and interpretation
689 of the data. This work has used computing resources at the High Performance Computing
690 Facility (HPCF) of the Center for Health Informatics and Bioinformatics at the NYU
691 Langone Medical Center.
692

693 ## Figure legends

694 **Figure 1. HiC-bench workflow.** Raw reads (input fastq files) are aligned and then filtered
695 (*align* and *filter* tasks). Filtered reads are used for the creation of Hi-C track files (*tracks*)
696 that can be directly uploaded to the WashU Epigenome Browser [27]. A report with a
697 statistics summary of filtered Hi-C reads, is also automatically generated (*filter-stats*).

698 Raw Hi-C matrices (*matrix-filtered*) are normalized using (a) scaling, (b) iterative
699 correction [9] or (c) HiCNorm [28]. A report with the plots of the normalized Hi-C counts
700 as function of the distance between the interacting partners (*matrix-stats*) is automatically
701 generated for all methods. The resulting matrices are compared across all samples in
702 terms of Pearson and Spearman correlation (*compare-matrices* and *compare-matrices-*
703 *stats*). Boundary scores are calculated and the corresponding report with the Principal
704 Component Analysis (PCA) is automatically generated (*boundary-scores* and *boundary-*
705 *scores-pca*). Domains are identified using various TAD calling algorithms (*domains*)
706 followed by comparison of TAD boundaries (*compare-boundaries* and *compare-*
707 *boundaries-stats*). A report with the statistics of boundary comparison is also
708 automatically generated. Hi-C visualization of user-defined genomic regions is performed
709 using HiCPlotter (*hicplotter*) [23]. Specific chromatin interactions (*interactions*) are
710 detected and annotated (*annotations*). Finally, enrichment of top interactions in certain
711 chromatin marks, transcription factors etc. provided by the user, is automatically
712 calculated (*annotations-stats*).

713 **Figure 2. (A) Computational trails.** Each combination of tools and parameter settings
714 can be imagined as a unique computational "trail" that is executed simultaneously with all
715 the other possible trails to create a collection of output objects. As an example, one of
716 these possible trails is presented in red. The raw reads were aligned, filtered and then
717 binned in 40kb resolution matrices. Our own naïve matrix scaling method was then used
718 for matrix correction and domains were called using TopDom [31]. **(B) HiC-bench**
719 **pipeline task architecture.** All pipeline tasks are performed by a single R script,
720 "*pipeline-master-explorer.r*". This script generates output objects based on all

721 combinations of input objects and parameter scripts while taking into account the split
722 variable, group variable and tuple settings. The output objects are stored in the
723 corresponding "*results*" directory. As an example, domain calling for IMR90 is presented.
724 The filtered reads of the IMR90 Hi-C sample (digested with HindIII) are used as input.
725 The pipeline-master-explorer script tests if TAD calling with these settings has been
726 performed and if not it calls the domain calling wrapper script (*code/hicseq-domains.tcsh*)
727 with the corresponding parameters (e.g. *params/params.armatus.gamma_0.5.tcsh*).
728 After the task is complete, the output is stored in the corresponding "results" directory.

729 **Figure 3. Comparison of topological domain calling methods subject to Hi-C**
730 **contact matrix preprocessing by simple filtering or iterative correction** (**IC**). The
731 methods were assessed in terms of boundary overlap between replicates (**A**), change
732 (%) in mean boundary overlap after matrix correction (**B**), change (%) in standard
733 deviation of mean overlap across replicates after matrix correction (**C**) and number of
734 identified topological domains per cell type (**D**). The different colors correspond to the
735 different callers. Gradients of the same color are used for the different values of the same
736 parameter, ranging from low (light color) to high (dark color) values. The TAD callers
737 along with the corresponding parameter settings are presented in the legend. For this
738 analysis all available read pairs were used.

739 **Figure 4**. **Comparison of topological domain calling methods for different**
740 **preprocessing method and sequencing depth**. TAD calling methods were assessed
741 in terms of boundary overlap between replicates (**A**), number of identified topological
742 domains (**B**) and boundary overlap across replicates upon increasing sequencing depth
743 (**C**) for different matrix preprocessing (filtered and IC corrected) and different sequencing

744 depths (10 million, 20 million and 40 million reads). For TAD calling, only the optimal
745 caller/parameter value pairs are shown (defined as the ones achieving the maximum
746 boundary overlap for IC and 40 million reads). The boxplot and line colors correspond to
747 the different TAD callers.
748
749 **Supplementary Figure 1. Hi-C reads filtering statistics.** Number **(A)** and percentage
750 (**B**) of the various read categories identified during filtering for all datasets used in the
751 study. Mappable reads were over 95% in all samples. Duplicate (*ds-duplicate-intra* and
752 *ds-duplicate-inter*; red and pink), non-uniquely mappable (*multihit*; light blue) and single-
753 end mappable (*single-sided*; dark blue) reads were discarded. Self-ligation products (*ds-*
754 *same-fragment*) and reads mapping too far (*ds-too-far*; light purple) from restriction sites
755 or too close to one another (*ds-too-close*; orange) were also discarded. Only double-sided
756 uniquely mappable *cis* (*ds-accepted-intra*; dark green) and *trans* (*ds-accepted-inter*; light
757 green) read pairs were used for downstream analysis. The *x* axis represents either the
758 raw read number (A) or the percentage of reads (B) falling within each of the categories
759 described in the legend. The *y* axis represents the samples.
760 **Supplementary Figure 2. Matrix statistics.** Normalized Hi-C counts are presented as a
761 function of the distance between the interacting partners for all samples and correction
762 methods. The Hi-C samples analyzed were GM12878 (light blue), hESCs (H1) (blue),
763 mesenchymal cells (light green), mesendoderm (dark green), neural progenitors (pink),
764 trophectoderm (red), IMR90 (light and dark orange), K562 (light purple), KBM7 (dark
765 purple) and NHEK (yellow). The matrices were either unprocessed (filtered) (top) or

766 corrected using IC (bottom). The *y* axis represents the normalized count of Hi-C
767 interactions and the *x* axis the distance between the interacting partners in kilobases.
768 **Supplementary Figure 3. Pairwise Pearson correlation of Hi-C matrices.**
769 Correlograms summarizing all pairwise Pearson correlations for all Hi-C samples used in
770 this study: raw (filtered) matrices (left panel) and matrices after iterative correction (right
771 panel). Dark red indicates strong positive correlation and dark blue strong negative. The
772 resolution of the matrices is 40kb.
773 **Supplementary Figure 4. Boundary score calculation.** Two adjacent topological
774 domains (red triangles) are depicted. The left domain (*L*) is separated from the right
775 domain (*R*) by a boundary (black circle). The areas of more-frequent intra-domain
776 interactions are in red. The area of less-frequent cross-domain (or inter-domain)
777 interactions is *X*. We also introduce parameter *d* which is the maximum distance from the
778 diagonal to be considered for the calculation of boundary scores (default value: *d*=2Mb).
779 **Supplementary Figure 5. Principal component analysis of boundary scores**.
780 Boundary scores were calculated using *ratio* score, for all samples either before (filtered)
781 (left panel) or after iterative correction (IC) (right panel).
782 **Supplementary Figure 6. Pairwise overlaps of TAD boundaries.** The pairwise
783 overlaps of TAD boundaries are shown for all samples of this study, after calling
784 boundaries using hicratio (all reads, *d*=0500). Before TAD calling, the Hi-C matrices were
785 either unprocessed (filtered) or corrected using iterative correction (IC) (resolution =
786 40kb).
787 **Supplementary Figure 7. Visualization of TADs and certain areas of interest.** HiC-
788 bench integrates HiCPlotter [23] and it offers the ability to easily prepare publication-

789 quality figures. We present the area surrounding *NANOG*, a gene of particular importance
790 for the maintenance of pluripotency. The Hi-C matrix corresponding to the
791 chr12:3940389-11948655 genomic region is presented for H1 cells, before and after
792 matrix correction. The matrix is also rotated 45° to facilitate TAD visualization. Various
793 boundary scores (intra-max, DI, ratio) are shown as individual tracks along with CTCF
794 binding. The location of *NANOG* is presented as a blue line.

795 **Supplementary Figure 8. Enrichment of chromatin interactions in human**
796 **fibroblasts (IMR90) and embryonic stem cells (H1).** The enrichment of certain
797 chromatin marks and CTCF in the top 50000 chromatin interactions in the IMR90 and H1
798 samples is shown. Deep blue and larger circle size indicate higher enrichment.

799
800

801 ## Table legends

802 **Table 1. Comparison of HiC-bench with published Hi-C analysis or visualization**
803 **tools.** HiC-bench is a comprehensive and feature-rich Hi-C analysis pipeline that
804 performs various Hi-C tasks by combining our newly-developed tools with existing tools.

805 **Table 2. The HiC-bench toolkit.** The HiC-bench toolkit consists mostly of newly-
806 developed tools (shown in bold) but we have also incorporated existing tools to allow
807 comparisons and benchmarking.

808 **Supplementary Table 1. HiC-bench task implementation.** The table summarizes how
809 the pipeline tasks are implemented, which are the requirements for their execution and
810 how they are handled by the *pipeline-master-explorer* script. The first column lists all the
811 tasks performed by pipeline ranging from alignment to annotation. The second column

812 lists the input directory required for each task while the third one lists the parameter files.
813 Certain tasks depend on the reference genome (human or mouse), thus the genome
814 assembly acts as split variable (column 4). In some tasks, replicates can be grouped
815 using the group variable (column 5). Pairwise comparisons between replicates or samples
816 are also allowed using tuples (column 6). The last column lists the full pipeline-master-
817 explorer command for each pipeline task.

818 **Supplementary Table 2. HiC-bench input-output objects.** The table summarizes the
819 inputs and outputs of the TAD-calling task using three different methods with parameter
820 values stored in the params files (column 2). The first column describes the tree structure
821 of the input directories that are essentially the different Hi-C matrices for each sample,
822 before (filtered) and after matrix correction using different methods (e.g. IC). The second
823 column lists all the different parameter scripts and the third column corresponds to the
824 tree structure of the generated output objects.

825

826 ## References
827

828 1. Dekker J, Marti-Renom MA, Mirny LA. Exploring the three-dimensional organization of
829 genomes: interpreting chromatin interaction data. Nat Rev Genet. 2013;14:390–403.

830 2. Lieberman-Aiden E, van Berkum NL, Williams L, Imakaev M, Ragoczy T, Telling A, et al.
831 Comprehensive mapping of long-range interactions reveals folding principles of the human
832 genome. Science. 2009;326:289–93.

833 3. Belton J-M, McCord RP, Gibcus JH, Naumova N, Zhan Y, Dekker J. Hi-C: a comprehensive
834 technique to capture the conformation of genomes. Methods. 2012;58:268–76.

835 4. Fraser J, Ferrai C, Chiariello AM, Schueler M, Rito T, Laudanno G, et al. Hierarchical folding
836 and reorganization of chromosomes are linked to transcriptional changes in cellular
837 differentiation. Molecular Systems Biology. 2015;11:852–2.

838 5. Dixon JR, Selvaraj S, Yue F, Kim A, Li Y, Shen Y, et al. Topological domains in mammalian

839 genomes identified by analysis of chromatin interactions. Nature. 2012;485:376–80.

840 6. Phillips-Cremins JE, Sauria MEG, Sanyal A, Gerasimova TI, Lajoie BR, Bell JSK, et al.
841 Architectural protein subclasses shape 3D organization of genomes during lineage
842 commitment. Cell. 2013;153:1281–95.

843 7. Vietri Rudan M, Barrington C, Henderson S, Ernst C, Odom DT, Tanay A, et al. Comparative Hi-
844 C Reveals that CTCF Underlies Evolution of Chromosomal Domain Architecture. Cell Rep.
845 2015;10:1297–309.

846 8. Yaffe E, Tanay A. Probabilistic modeling of Hi-C contact maps eliminates systematic biases to
847 characterize global chromosomal architecture. Nat Genet. 2011;43:1059–65.

848 9. Imakaev M, Fudenberg G, McCord RP, Naumova N, Goloborodko A, Lajoie BR, et al. Iterative
849 correction of Hi-C data reveals hallmarks of chromosome organization. Nat Meth. 2012;9:999–
850 1003.

851 10. Cournac A, Marie-Nelly H, Marbouty M, Koszul R, Mozziconacci J. Normalization of a
852 chromosomal contact map. BMC Genomics. 2012;13:436.

853 11. Koszul R. HiC-Box. https://github.com/rkoszul/HiC-Box. Accessed 20 February 2016.

854 12. Servant N, Varoquaux N, Lajoie BR, Viara E, Chen C-J, Vert J-P, et al. HiC-Pro: an optimized
855 and flexible pipeline for Hi-C data processing. Genome Biol. 2015;16:259.

856 13. Li W, Gong K, Li Q, Alber F, Zhou XJ. Hi-Corrector: a fast, scalable and memory-efficient
857 package for normalizing large-scale Hi-C data. Bioinformatics. 2015;31:960–2.

858 14. Castellano G, Le Dily F, Hermoso Pulido A, Beato M, Roma G. Hi-Cpipe: a pipeline for high-
859 throughput chromosome capture. bioRxiv. 2015; doi: 10.1101/020636

860 15. Sauria ME, Phillips-Cremins JE, Corces VG, Taylor J. HiFive: a tool suite for easy and efficient
861 HiC and 5C data analysis. Genome Biol. 2015;16:237.

862 16. Heinz S, Benner C, Spann N, Bertolino E, Lin YC, Laslo P, et al. Simple Combinations of
863 Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for
864 Macrophage and B Cell Identities. Molecular Cell. 2010;38:576–89.

865 17. Wingett S, Ewels P, Furlan-Magaril M, Nagano T, Schoenfelder S, Fraser P, et al. HiCUP:
866 pipeline for mapping and processing Hi-C data. F1000Res. 2015;4:1310.

867 18. Krueger F, Andrews SR. SNPsplit: Allele-specific splitting of alignments between genomes
868 with known SNP genotypes. F1000Res. 2016;5:1479.

869 19. Serra F, Baù D, Filion G, Marti-Renom MA. Structural features of the fly chromatin colors
870 revealed by automatic three-dimensional modeling. bioRxiv. 2016; doi: 10.1101/036764.

871

872 20. Schmid MW, Grob S, Grossniklaus U. HiCdat: a fast and easy-to-use Hi-C data analysis tool.
873 BMC Bioinformatics. 2015;16:390–6.

874 21. Hwang Y-C, Lin C-F, Valladares O, Malamon J, Kuksa PP, Zheng Q, et al. HIPPIE: a high-
875 throughput identification pipeline for promoter interacting enhancer elements. Bioinformatics.
876 2015;31:1290–2.

877 22. Phanstiel DH, Boyle AP, Araya CL, Snyder MP. Sushi.R: flexible, quantitative and integrative
878 genomic visualizations for publication-quality multi-panel figures. Bioinformatics.
879 2014;30:2808–10.

880 23. Akdemir KC, Chin L. HiCPlotter integrates genomic data with interaction matrices. Genome
881 Biol. 2015;16:198.

882 24. Editorial. Rebooting review. Nat Biotechnol. 2015;33:319–9.

883 25. Goecks J, Nekrutenko A, Taylor J, Galaxy Team. Galaxy: a comprehensive approach for
884 supporting accessible, reproducible, and transparent computational research in the life
885 sciences. Genome Biol. 2010;11(8):R86.

886 26. Ben Langmead, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Meth.
887 2012;9:357–9.

888 27. Zhou X, Lowdon RF, Li D, Lawson HA, Madden PAF, Costello JF, et al. Exploring long-range
889 genome interactions using the WashU Epigenome Browser. Nat Meth. 2013;10:375–6.

890 28. Hu M, Hu M, Deng K, Deng K, Selvaraj S, Selvaraj S, et al. HiCNorm: removing biases in Hi-C
891 data via Poisson regression. Bioinformatics. 2012;28:3131–3.

892 29. Tsirigos A, Haiminen N, Bilal E, Utro F. GenomicTools: a computational platform for
893 developing high-throughput analytics in genomics. Bioinformatics. 2012;28:282–3.

894 30. Filippova D, Patro R, Duggal G, Kingsford C. Identification of alternative topological domains
895 in chromatin. Algorithms for Molecular Biology. 2014;9:14.

896 31. Shin H, Shi Y, Dai C, Tjong H, Gong K, Alber F, et al. TopDom: an efficient and deterministic
897 method for identifying topological domains in genomes. Nucleic Acids Res. 2016;44(7):e70.

898 32. Crane E, Bian Q, McCord RP, Lajoie BR, Wheeler BS, Ralston EJ, et al. Condensin-driven
899 remodelling of X chromosome topology during dosage compensation. Nature. 2015;523:240–4.

900 33. Van Bortle K, Nichols MH, Li L, Ong C-T, Takenaka N, Qin ZS, et al. Insulator function and
901 topological domain border strength scale with architectural protein occupancy. Genome Biol.
902 2014;15:R82.

903 34. Alekseyenko AA, Walsh EM, Wang X, Grayson AR, Hsi PT, Kharchenko PV, et al. The
904 oncogenic BRD4-NUT chromatin regulator drives aberrant transcription within large topological
905 domains. Genes Dev. 2015;29:1507–23.

906 35. Ludäscher B, Altintas I, Berkley C, Higgins D, Jaeger E, Jones M, et al. Scientific workflow
907 management and the Kepler system. Concurrency and Computation: Practice and Experience.
908 2006;18:1039–65.

909 36. Oinn T, Addis M, Ferris J, Marvin D, Senger M, Greenwood M, et al. Taverna: a tool for the
910 composition and enactment of bioinformatics workflows. Bioinformatics. 2004;20:3045–54.

911 37. Freire J. Making Computations and Publications Reproducible with VisTrails. Computing in
912 Science & Engineering 2012;14:18–25.

913 38. Bavoil L, Callahan SP, Crossno PJ, Freire J, Scheidegger CE, Silva CT, et al. VisTrails: enabling
914 interactive multiple-view visualizations. VIS 05 IEEE; 2005. pp. 135–42.

915 39. Wright K. Plot a Correlogram. R package. http://CRAN.R-project.org/package=corrgram

916 40. Sexton T, Yaffe E, Kenigsberg E, Bantignies F, Leblanc B, Hoichman M, et al. Three-
917 Dimensional Folding and Functional Organization Principles of the Drosophila Genome. Cell
918 2012;148:458–72.

919 41. Nora EP, Lajoie BR, Schulz EG, Giorgetti L, Okamoto I, Servant N, et al. Spatial partitioning of
920 the regulatory landscape of the X-inactivation centre. Nature. 2012;485:381–5.

921 42. Rao SSP, Huntley MH, Durand NC, Stamenova EK, Bochkov ID, Robinson JT, et al. A 3D Map
922 of the Human Genome at Kilobase Resolution Reveals Principles of Chromatin Looping. Cell.
923 2014;159:1665–80.

924 43. R Core Team. R: A language and environment for statistical computing. R Foundation for
925 statistical Computing Vienna, Austria 2016. https://www.R-project.org/

926 44. mirnylib. https://bitbucket.org/mirnylab/mirnylib. Accessed on 20 May 2016.

927 45. Dixon JR, Jung I, Selvaraj S, Shen Y, Antosiewicz-Bourget JE, Lee AY, et al. Chromatin
928 architecture reorganization during stem cell differentiation. Nature. 2015;518:331–6.

929

Figure 1

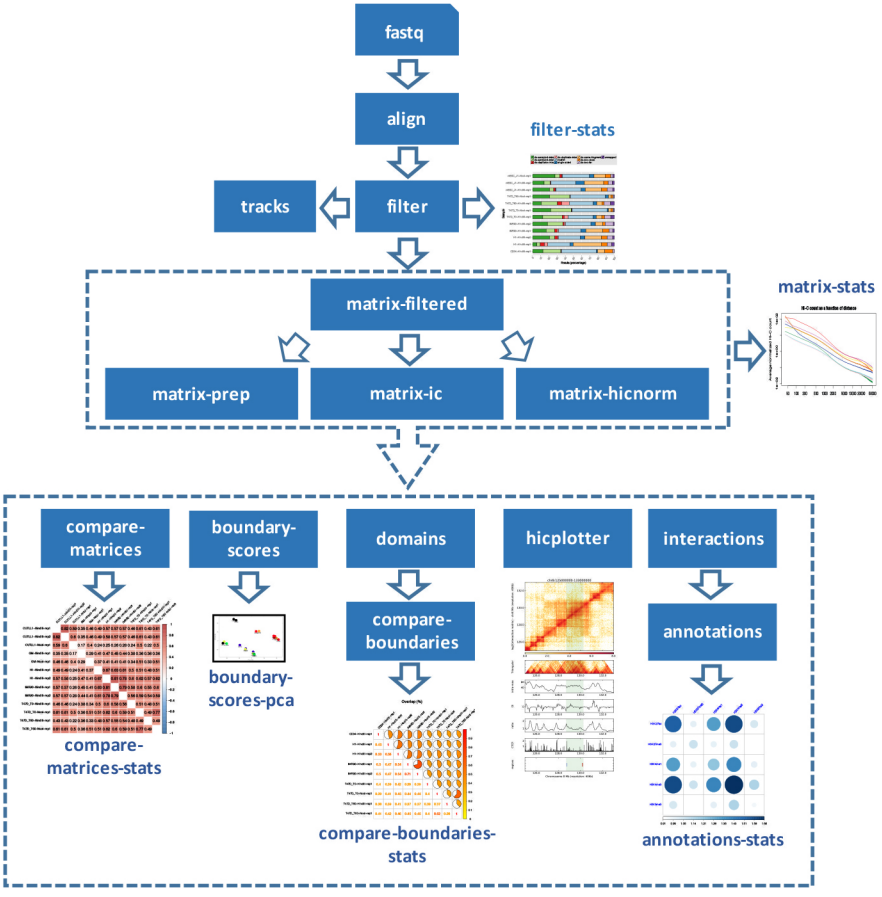
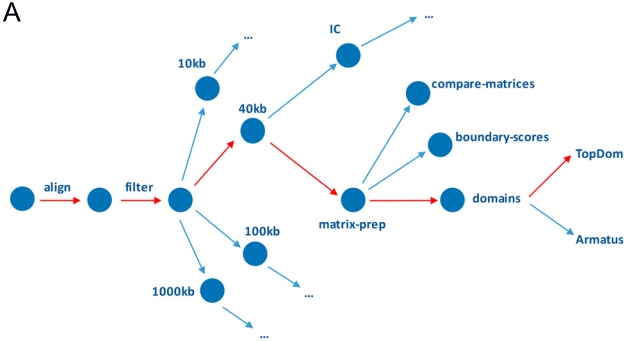

fastq
align
filter
tracks
filter-stats
matrix-filtered
matrix-prep
matrix-ic
matrix-hicnorm
matrix-stats
compare-matrices
boundary-scores
domains
hicplotter
interactions
compare-matrices-stats
boundary-scores-pca
compare-boundaries
compare-boundaries-stats
annotations
annotations-stats

Figure 2

A


IC
...
10kb
...
compare-matrices
40kb
boundary-scores
TopDom
align
filter
domains
matrix-prep
100kb
Armatus
1000kb
...
...

B

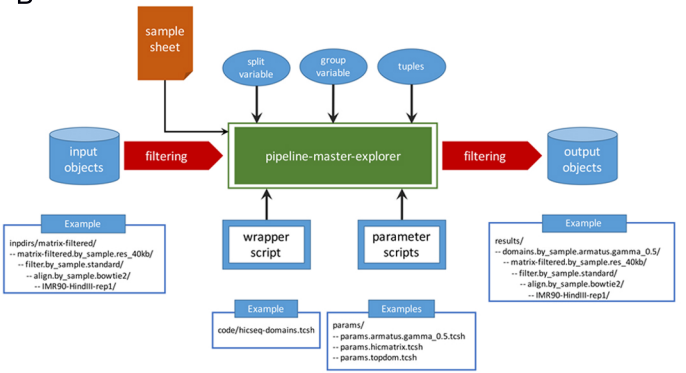

sample sheet
split variable
group variable
tuples
input objects
filtering
pipeline-master-explorer
filtering
output objects
Example
inpdirs/matrix-filtered/
-- matrix-filtered.by_sample.res_40kb/
-- filter.by_sample.standard/
-- align.by_sample.bowtie2/
-- IMR90-HindIII-rep1/
wrapper script
parameter scripts
Example
results/
-- domains.by_sample.armatus.gamma_0.5/
-- matrix-filtered.by_sample.res_40kb/
-- filter.by_sample.standard/
-- align.by_sample.bowtie2/
-- IMR90-HindIII-rep1/
Example
code/hicseq-domains.tcsh
Examples
params/
-- params.armatus.gamma_0.5.tcsh
-- params.hicmatrix.tcsh
-- params.topdom.tcsh

Figure 3

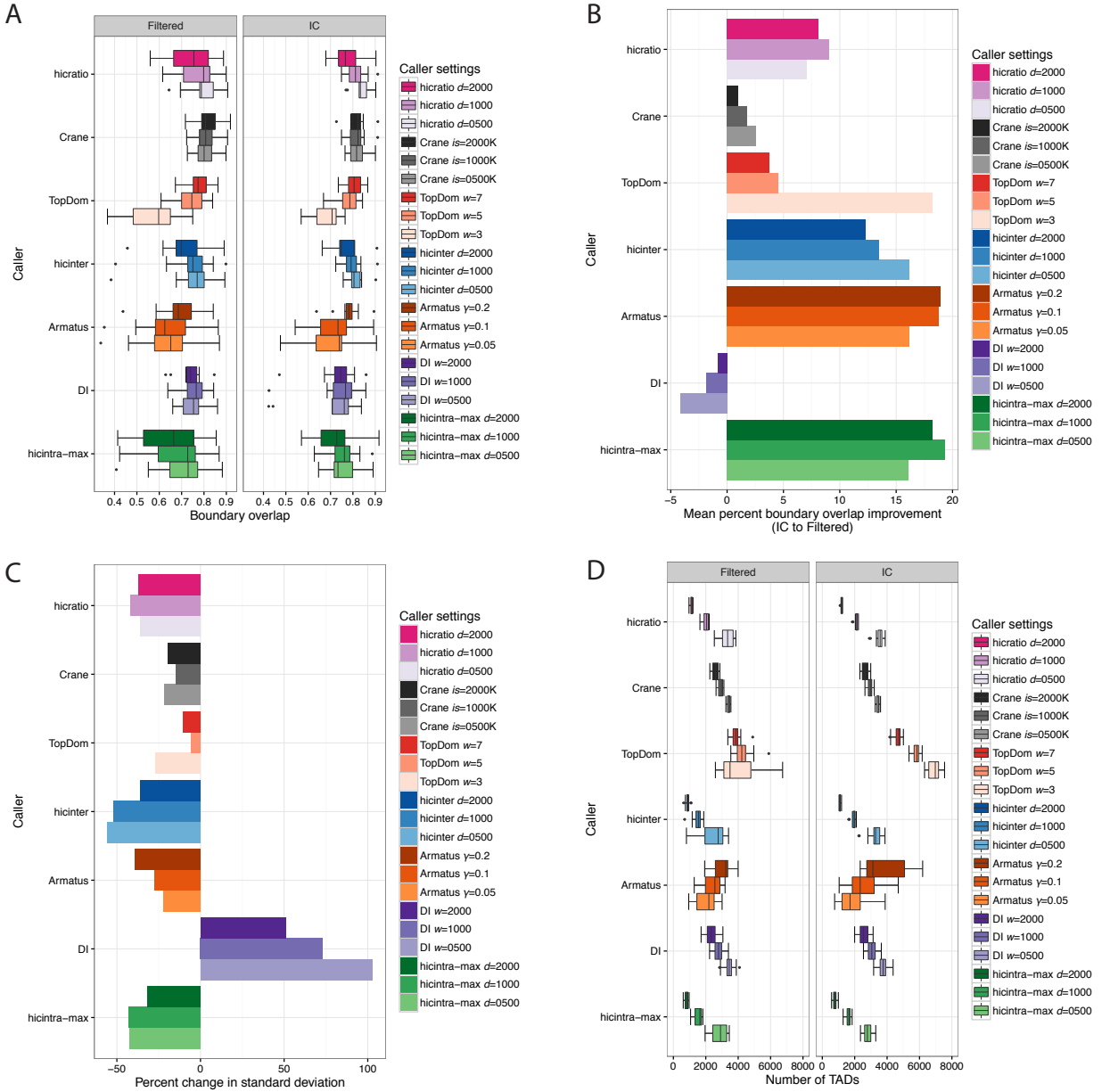
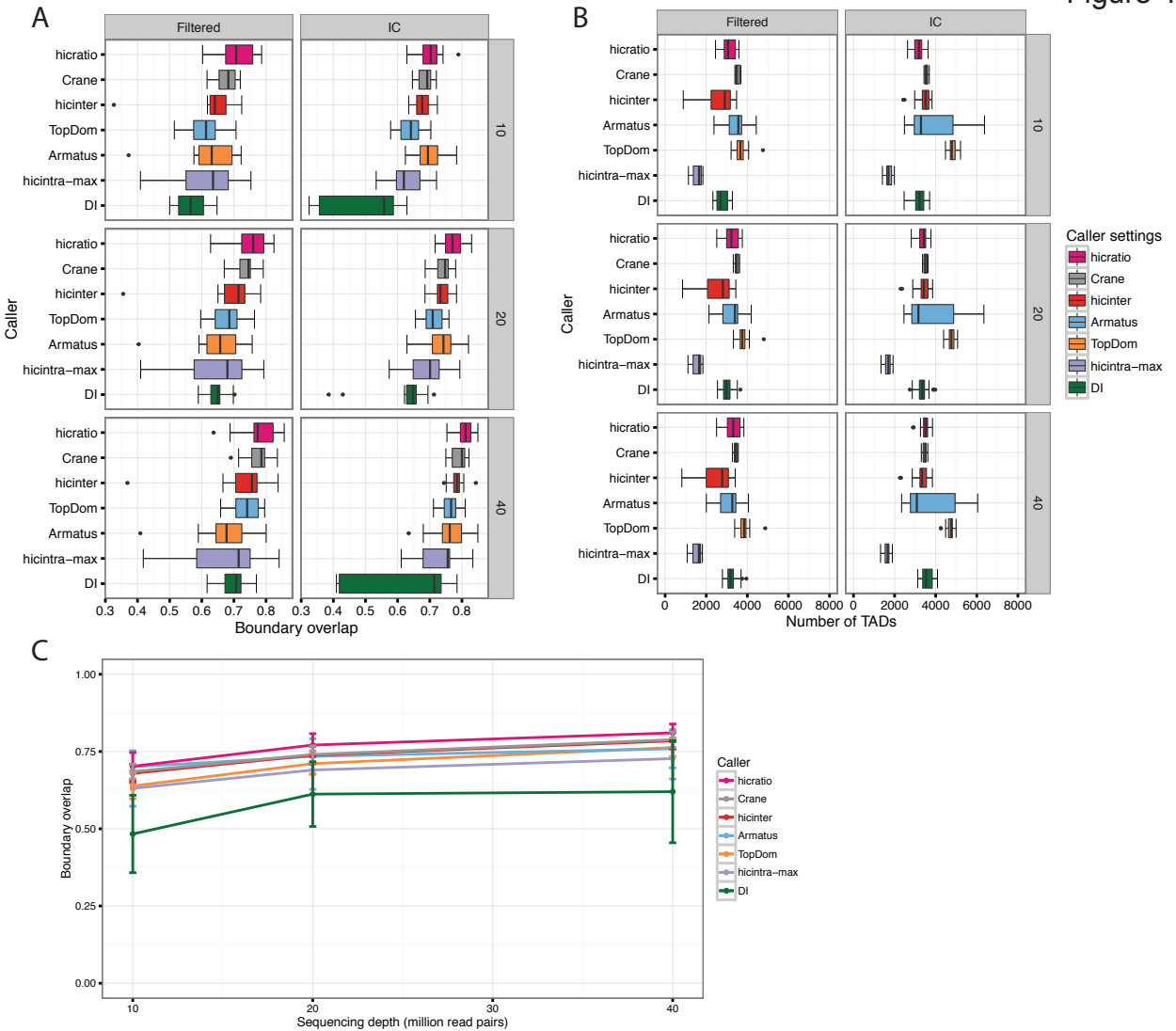
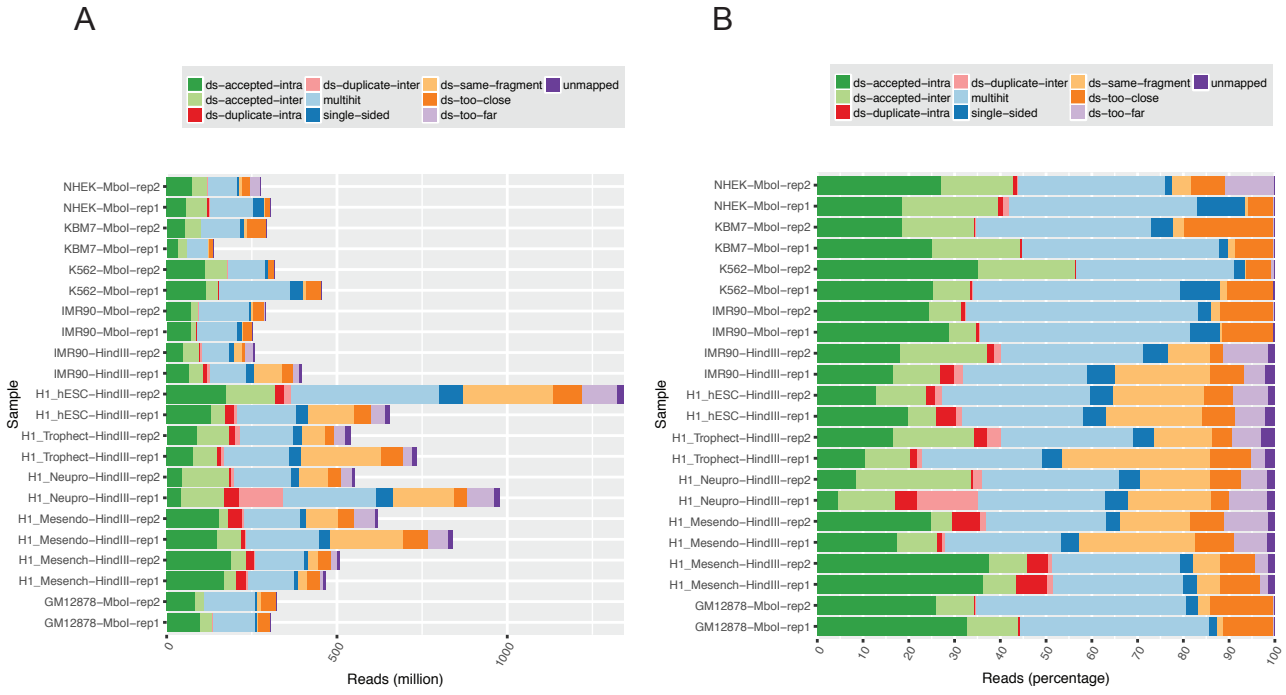

A
Filtered
IC
hicratio
Crane
TopDom
hicinter
Armatus
DI
hicintra−max
Caller
0.4 0.5 0.6 0.7 0.8 0.9
Boundary overlap
B
Mean percent boundary overlap improvement
(IC to Filtered)
−5 0 5 10 15 20
C
Percent change in standard deviation
−50 0 50 100
D
Number of TADs
0 2000 4000 6000 8000
Caller settings
hicratio d=2000
hicratio d=1000
hicratio d=0500
Crane is=2000K
Crane is=1000K
Crane is=0500K
TopDom w=7
TopDom w=5
TopDom w=3
hicinter d=2000
hicinter d=1000
hicinter d=0500
Armatus γ=0.2
Armatus γ=0.1
Armatus γ=0.05
DI w=2000
DI w=1000
DI w=0500
hicintra−max d=2000
hicintra−max d=1000
hicintra−max d=0500

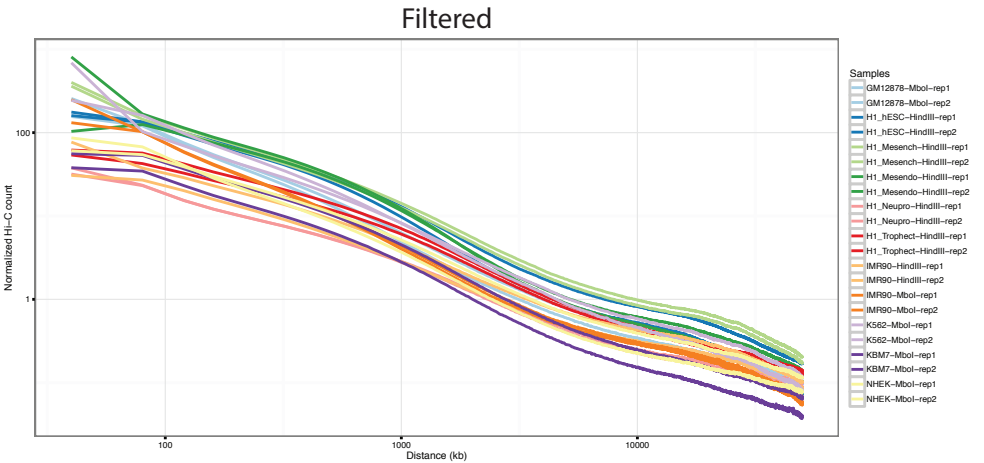
Figure 4

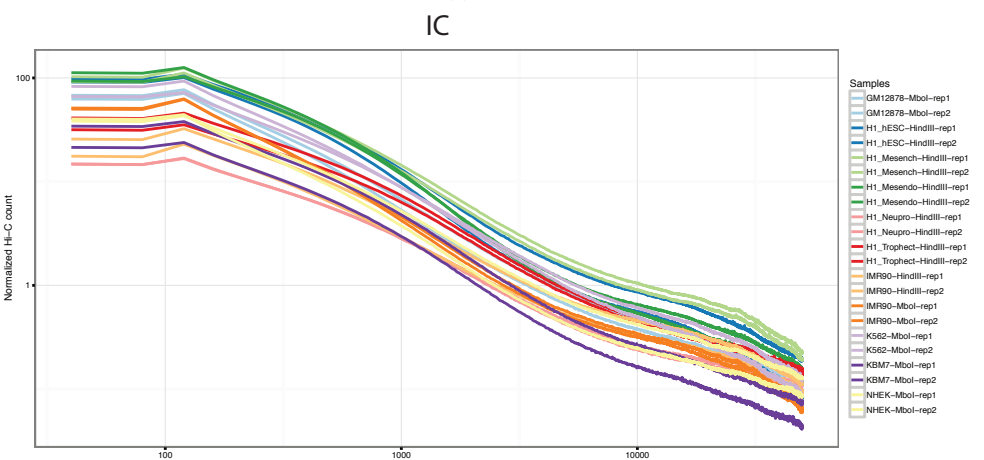

A
Filtered
IC
hicratio
Crane
hicinter
TopDom
Armatus
hicintra−max
DI
10
20
40
Caller
0.3 0.4 0.5 0.6 0.7 0.8
Boundary overlap
B
0 2000 4000 6000 8000
Number of TADs
Caller settings
C
1.00
0.75
0.50
0.25
0.00
Boundary overlap
10 20 30 40
Sequencing depth (million read pairs)
Caller

Supplementary Figure 1

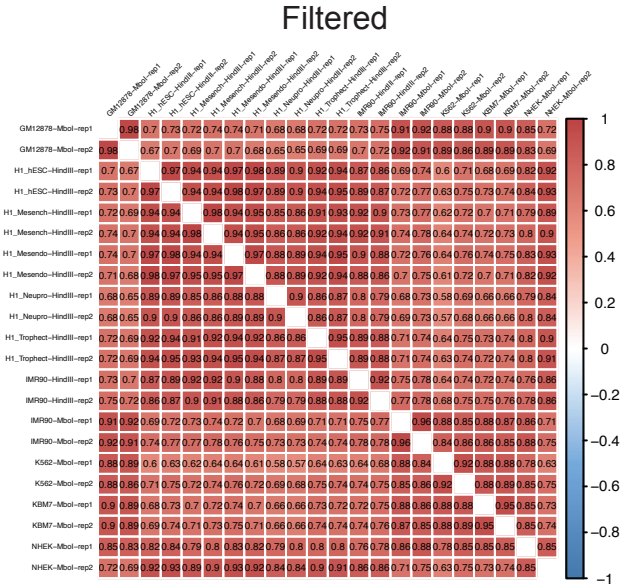


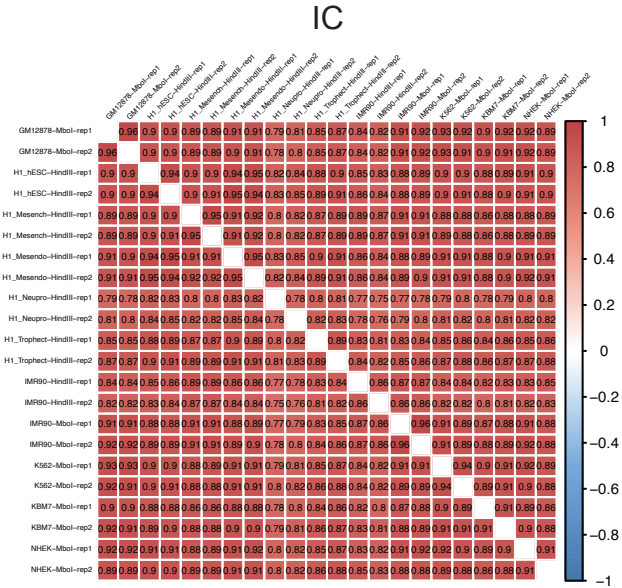
Supplementary Figure 2



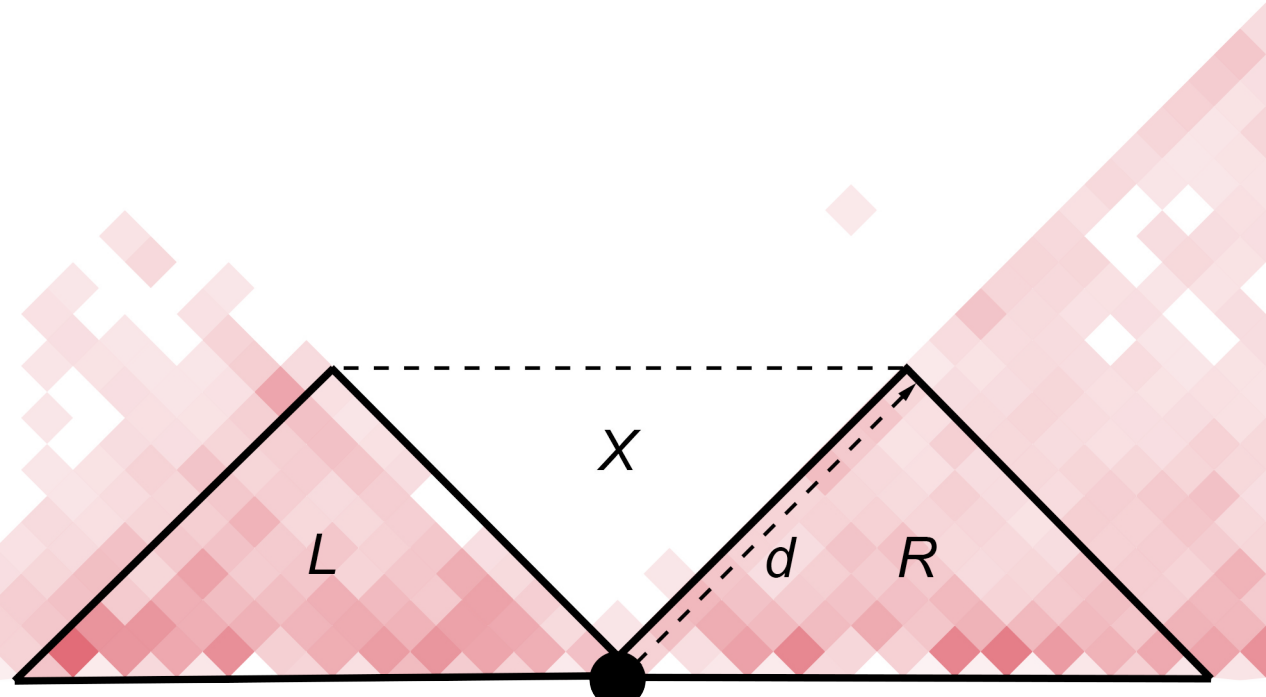


Supplementary Figure 3

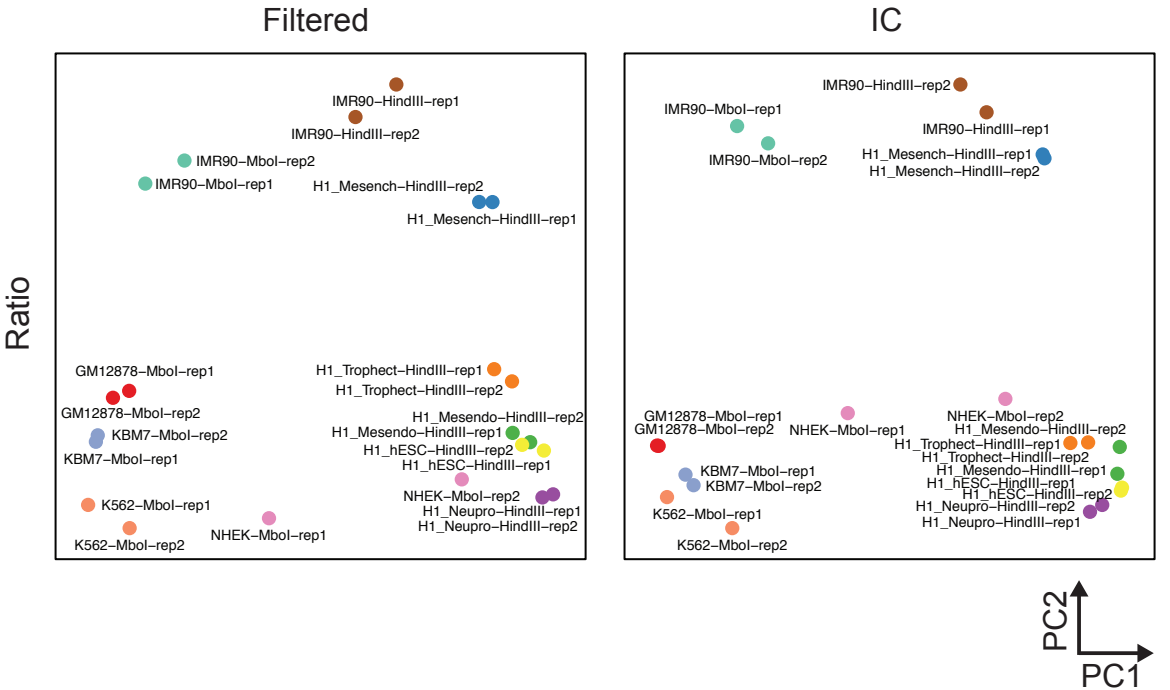




Supplementary Figure 4

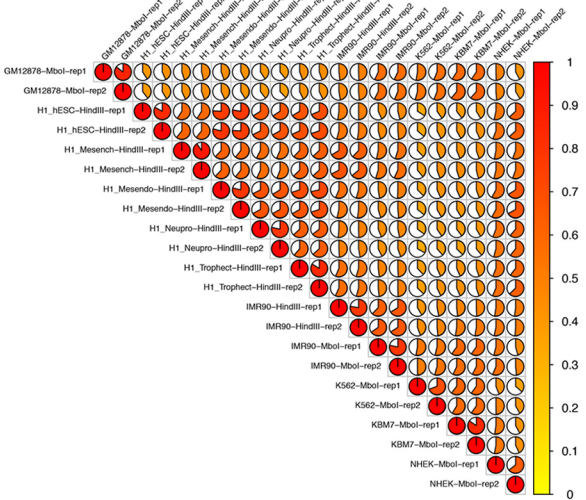


Supplementary Figure 5

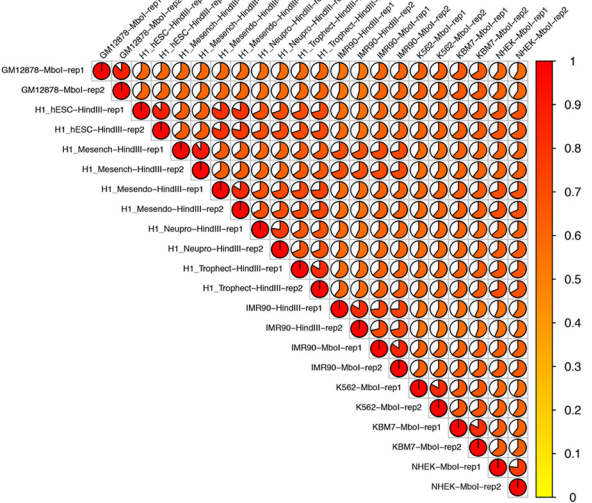


Supplementary Figure 6

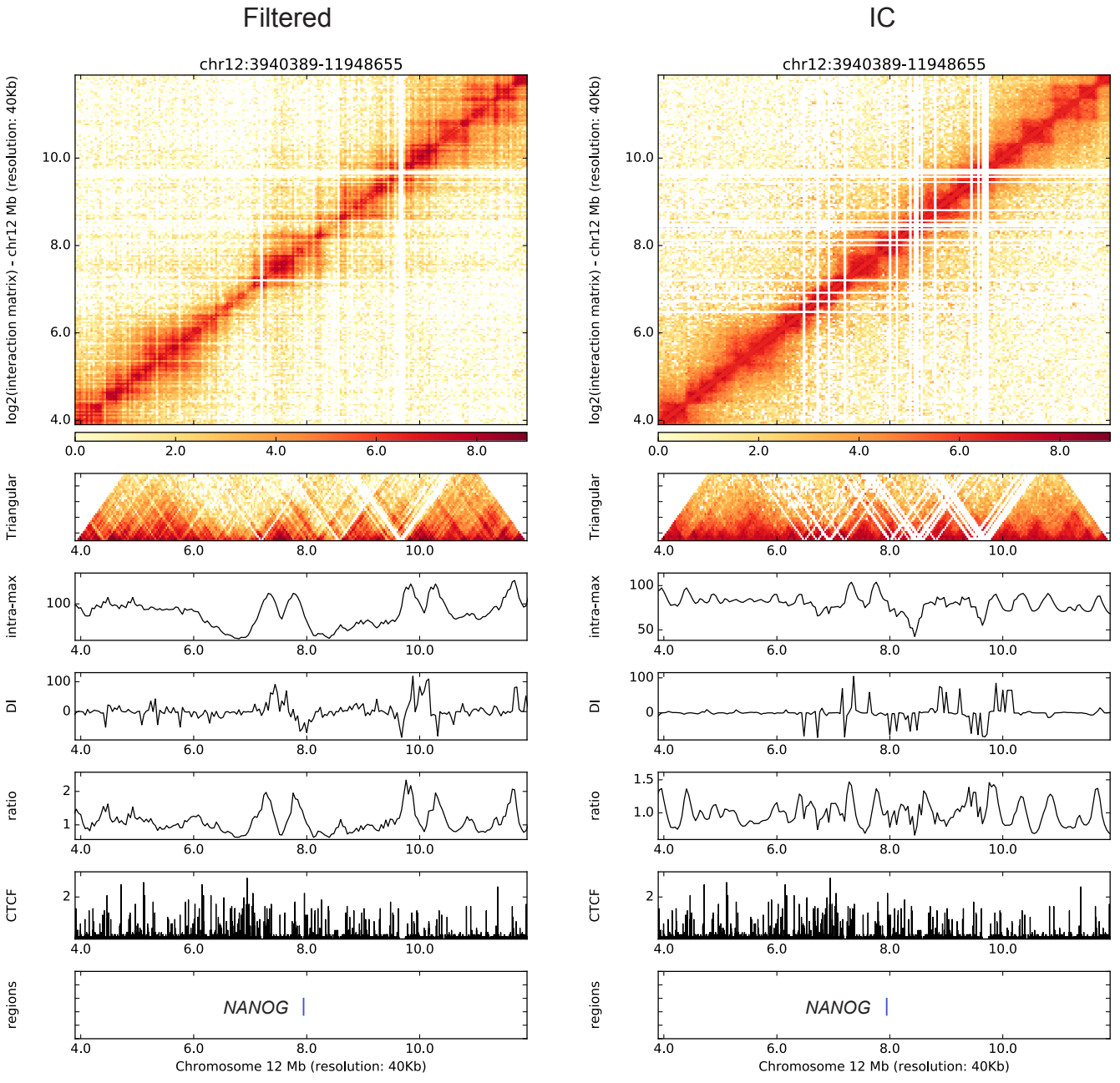
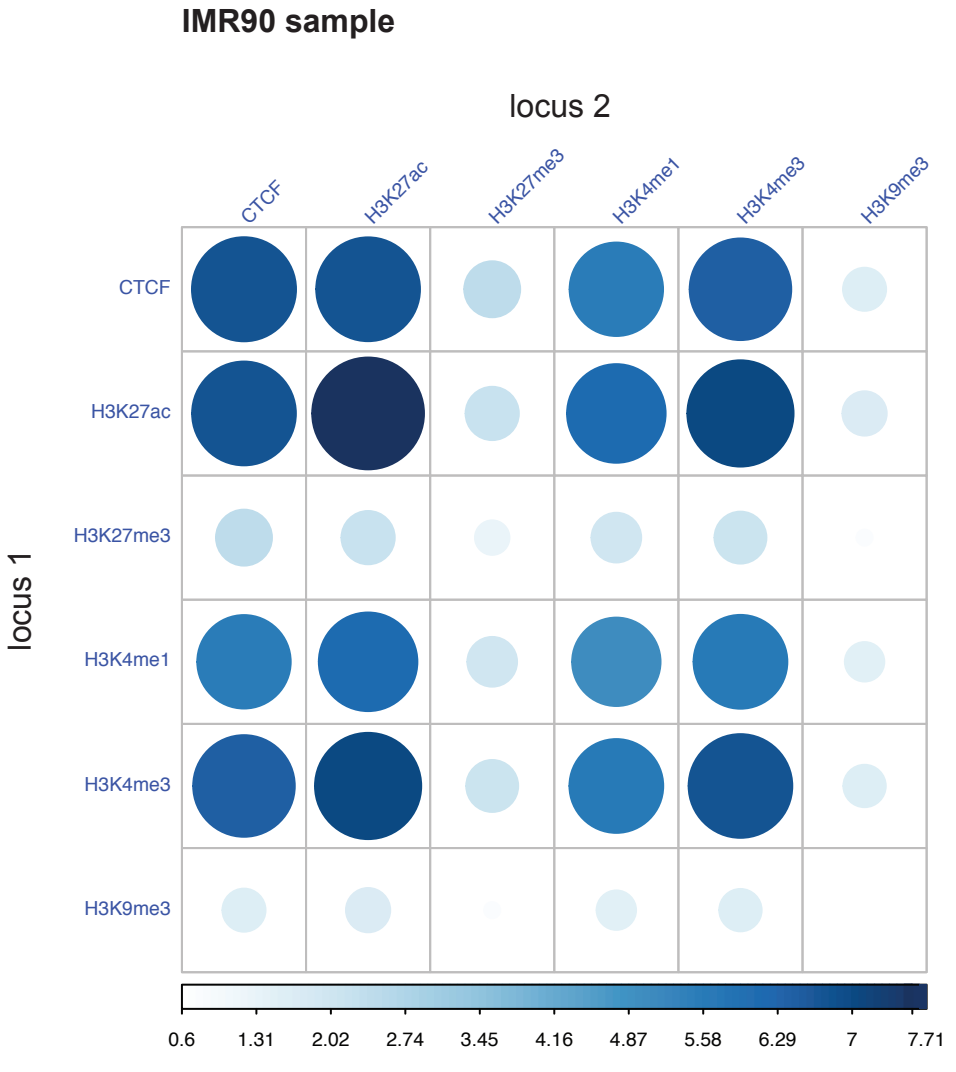
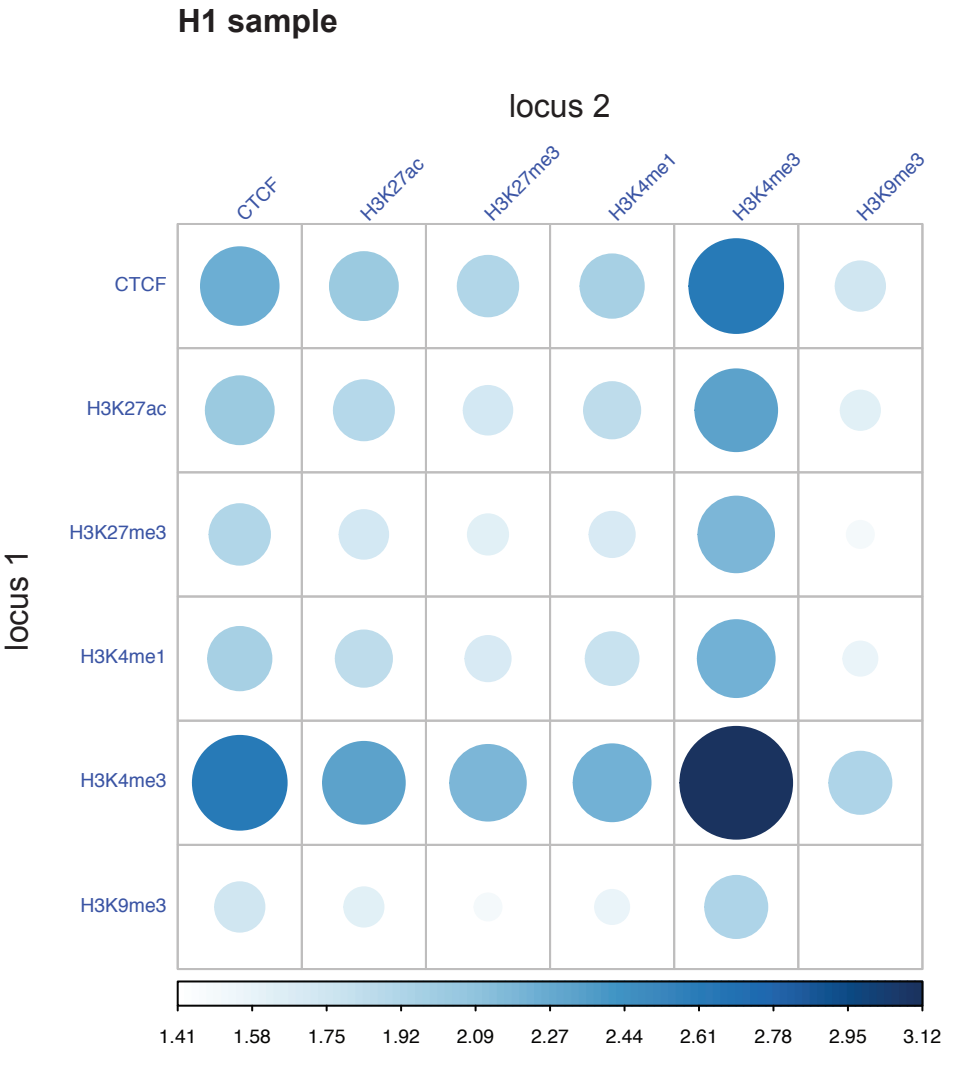
Filtered



IC



Supplementary Figure 7



Supplementary Figure 8