Chapter 8

Step-by-Step Protocol to Generate Hi-C Contact Maps Using the rfy_hic2 Pipeline

Hideki Tanizawa, Claire Yik-Lok Chung, Shun-ichiro Fuse, Tomomi Hayashi, Peter Weisel, and Ken-ichi Noma

Abstract

The Hi-C method has emerged as an indispensable tool for analyzing the 3D organization of the genome, becoming increasingly accessible and frequently utilized in chromatin research. To effectively leverage 3D genomics data obtained through advanced technologies, it is crucial to understand what processes are undertaken and what aspects require special attention within the bioinformatics pipeline. This protocol aims to demystify the Hi-C data analysis process for field newcomers. In a step-by-step manner, we describe how to process Hi-C data, from the initial sequencing of the Hi-C library to the final visualization of Hi-C contact data as heatmaps. Each step of the analysis is clearly explained to ensure an understanding of the procedures and their objectives. By the end of this chapter, readers will be equipped with the knowledge to transform raw Hi-C reads into informative visual representations, facilitating a deeper comprehension of the spatial genomic structures critical to cellular functions.

Key words 3D genome structure, Hi-C, Software pipelines

1 Introduction

In the genomics field, understanding the three-dimensional (3D) structure of the genome is crucial for exploring gene regulation and cellular functions [1–3]. Hi-C [4], a chromosome conformation capture (3C)-based technique [5], has transformed our ability to investigate the spatial organization of genomes. This technique provides detailed insights into the genomic architecture associated with genetic and epigenetic regulations of gene expression [6, 7].

Analyzing the complex data generated by Hi-C requires specialized bioinformatics approaches. This chapter introduces a straightforward, publicly available computational workflow tailored for Hi-C data analysis. Designed for both beginners and

Ryuichiro Nakato (ed.), *Computational Methods for 3D Genome Analysis*, Methods in Molecular Biology, vol. 2856,
https://doi.org/10.1007/978-1-0716-4136-1_8,

experienced researchers, this workflow simplifies the process from sequencing read alignment to the final visualization of contact maps as heatmaps.

The workflow begins with aligning Hi-C sequence reads to a reference genome, addressing the challenges posed by ligation junctions through iterative trimming and alignment. Subsequent steps include the removal of reads arising from self-ligation, undigested fragments (or religation), PCR duplications, and low-quality reads, followed by constructing a contact matrix by counting reads assigned to combinations of fixed-size genomic bins. To correct known biases in Hi-C data related to fragment length, GC content, and mappability, we employ the iterative correction and eigenvector decomposition (ICE) normalization method, ensuring an accurate representation of genomic contacts. The schematic of this protocol is shown in Fig. 1.

The final output, a heatmap of the Hi-C contact map, offers a visually intuitive representation of genomic contacts. This protocol not only simplifies the complex analysis process but also equips researchers with the tools needed to delve into the dynamic structure of the genome.

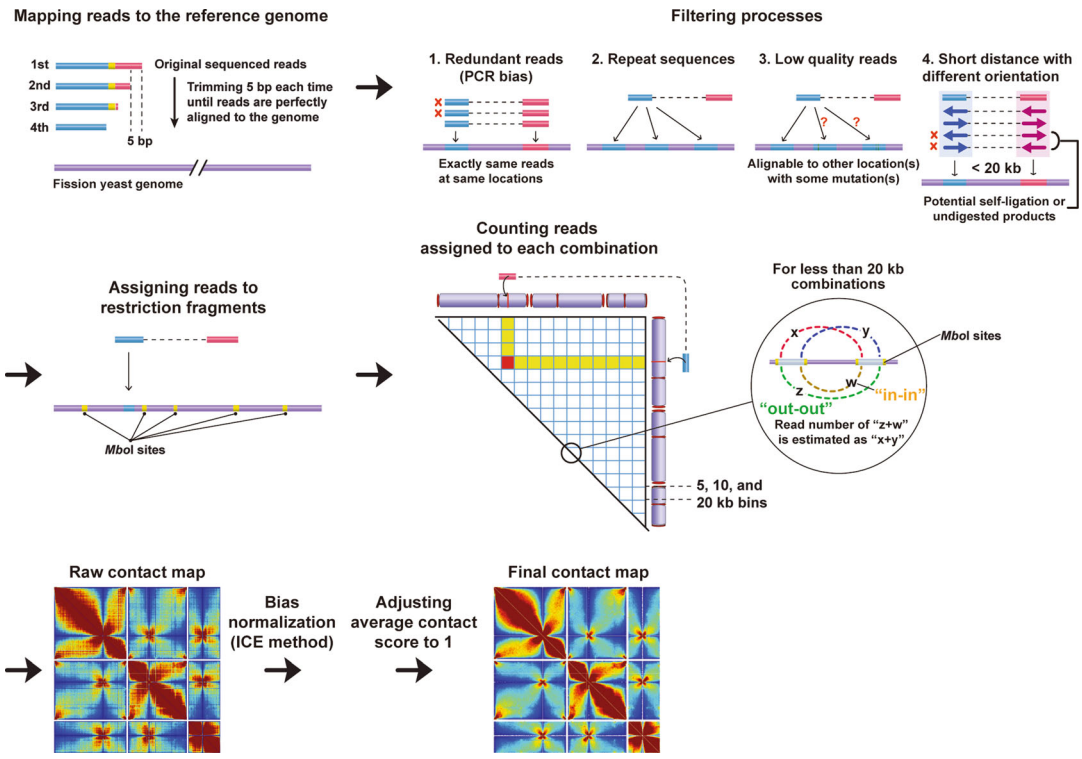

Fig. 1 Overview of the bioinformatics processing pipeline described in this article (reproduced from ref. 1 with permission from Springer Nature)

2 Materials

2.1 Linux Environment

This protocol is designed for use in a Linux environment, while it can also be run in other operation systems given a suitable setup. Windows users can use the pipeline by installing the Windows Subsystem for Linux (WSL). Refer to **Note 1** for setting up the WSL environment. While macOS and Linux are both UNIX-like operation systems, it is important to note that macOS users might encounter unexpected command behaviors. To provide broad compatibility while acknowledging potential limitations for non-Linux operating systems, we also prepare a Docker image equipped with all the prerequisites, including a Linux environment and necessary software, available in all the operation systems mentioned above. The usage instructions are described in Subheading 2.8.

Given the software prerequisites and the need to process large text files, a Linux system with at least a dual-core CPU and 4 GB of RAM is required. For a more comfortable analysis experience, an 8-core CPU and 32 GB of RAM are recommended to ensure efficient pipeline execution.

2.2 Software Prerequisites

The following software is required to execute the rfy_hic2 pipeline introduced in this protocol. The versions listed in parentheses are those used for verification during the creation of this protocol and are included in the Docker image we provide and describe in Subheading 2.8.

- Git (version 2.34.1): A distributed version control system with speed and efficiency (https://git-scm.com/).
- Bash (version 5.1.16): A Unix shell and command language (https://www.gnu.org/software/bash/).
- FastQC (version 0.11.9): A quality control tool for high-throughput sequence data (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/) [8].
- Samtools (version 1.13): A suite of programs for interacting with high-throughput sequencing data (https://www.htslib.org/) [9].
- SQLite3(version 3.37.2): A C-language library that implements a small, fast, self-contained, high-reliability, full-featured SQL database engine (https://www.sqlite.org/). For installation examples, refer to **Note 2**.
- R (version 4.1.2): A software environment for statistical computing and graphics (https://www.r-project.org/). For installation examples, refer to **Note 2**.

- Perl (version 5.34.0): A competent, feature-rich programming language with over 30 years of development (https://www.perl.org/).
- SRA-Toolkit (version 3.0.10): A collection of tools and libraries for using data in the Sequence Read Archive (SRA) (https://github.com/ncbi/sra-tools).
- Bowtie2 (version 2.4.4): An ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences (https://bowtie-bio.sourceforge.net/bowtie2/). For installation examples, refer to **Note 2** [10].

For installation of the software above in the macOS, please refer to **Note 3**.

2.3 Directory Structure

Establishing a structured folder system for organized and efficient analysis is beneficial although this is optional. This protocol assumes that the following separate directories have been prepared. Feel free to rename these directories. See examples of directory structure in Fig. 2.

– **library**: Stores scripts like rfy_hic2 that may be repeatedly used across several projects for consistency and efficiency. Each software or collection of scripts, such as rfy_hic2 and SRA-Toolkit, should reside in its own directory under the library folder. This file organization ensures clear organization and prevents potential conflicts during updates. For instance, the path to the rfy_hic2 scripts should be structured as */home/<username>/library/rfy_hic2*. Similarly, other software should follow the pattern */home/<username>/library/<other_software>*. This protocol assumes that the path to the library directory is stored in the variable *${DIR_LIB}*. An example of "DIR_LIB" could be */home/<username>/library*. To understand variables in the Linux environment, please refer to **Note 4**.

– **genome**: Create individual subfolders for each species (human, pombe, mouse, etc.). These should contain commonly used files such as reference sequences and index files for tools like Bowtie2 and rfy_hic2, facilitating reuse in multiple projects. This protocol assumes that the path to the fission yeast genome directory is stored in the variable *${GENOME_DIR}*. An example of setting "GENOME_DIR" is */home/<username>/genome/pombe*.

– **data_raw**: This directory should contain raw sequencing files, typically in FASTQ format (*.fastq* or *.fq*) or compressed FASTQ format (*.fastq.gz* or *.fq.gz*), along with associated experimental metadata, including index information. It is often easier to manage by creating a separate folder for each project. This protocol assumes that the path to the "data_raw" directory is

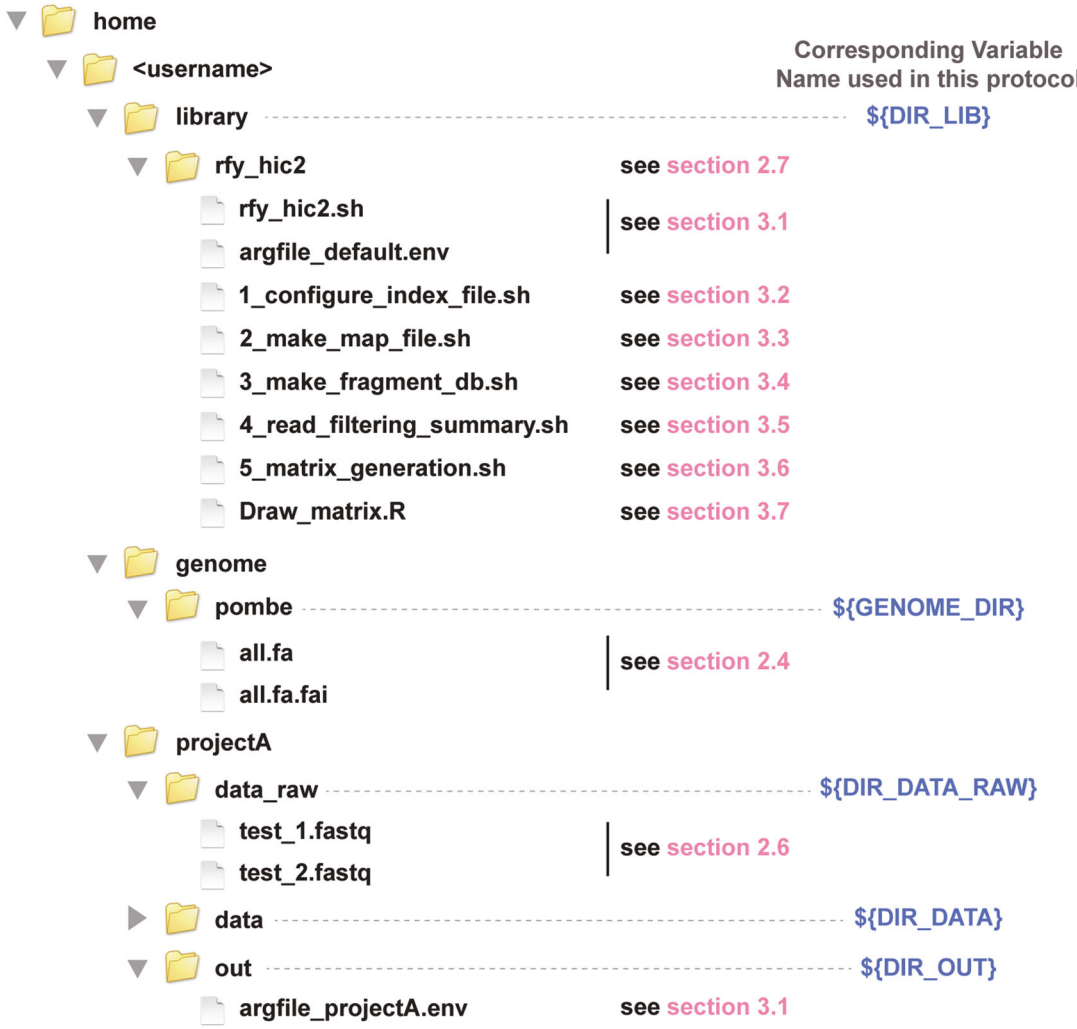

Fig. 2 Example of a directory structure. In this protocol article, we assume that readers have set up a directory structure similar to what is depicted in this figure. However, it is not mandatory to adhere strictly to this layout. Users are encouraged to modify folder names and hierarchies to match their personal preferences. It is important, though, to assign the full path of each folder to the variables shown in the figure in the configuration files. Also, note that not every folder and file is represented here. Only those of significant importance are highlighted

stored in the variable *${DIR_DATA_RAW}*. An example of setting "DIR_DATA_RAW" is */home/<username>/projectA/data_raw*.

– **data**: Dedicated to storing the processed data output from the rfy_hic2 pipeline. This protocol assumes the path to the data directory is stored in the variable *${DIR_DATA}*. An example of setting "DIR_DATA" is */home/<username>/projectA/data*.

– **out**: A folder for outputting various analysis results of each project, such as Hi-C heatmaps and significant contacts. This protocol assumes that the path to the out directory is stored in the variable *${DIR_OUT}*. An example of setting "DIR_OUT" is */home/<username>/projectA/out*.

This structured approach not only enhances project organization but also streamlines the process of managing scripts, genomics data, raw sequence reads, and processed data outputs.

2.4 Genome Sequence

Download the reference genome in FASTA format to initiate the analysis. For fission yeast (*Schizosaccharomyces pombe*), the reference genome is available at the PomBase (https://www.pombase.org) [11, 12]. Simplify the download and organization of the main three chromosomes (I, II, and III) with the following script:

```
cd ${GENOME_DIR}
for CHR in I II III
do
  curl  -O  https://www.pombase.org/data/genome_sequence_and_features/genome_sequence/Schizosaccharomyces_pombe_chromosome_${CHR}.fa.gz
done
```

Next, combine these chromosome files into one comprehensive file using:

```
zcat Schizosaccharomyces_pombe_chromosome_*.fa.gz > all.fa
```

Finally, calculate the length of each chromosome using the *samtools faidx* command as follows:

```
samtools faidx all.fa
```

Executing the above command will generate a file named *all.fa.fai*. This file contains a list of all chromosomes included in *all.fa* along with their length information.

2.5 Index File for Bowtie2

To align sequencing reads using Bowtie2, indexing a reference FASTA file is necessary. This process is accomplished using the *bowtie2-build* command as follows:

```
cd ${GENOME_DIR}
bowtie2-build -f all.fa <INDEX_NAME>
```

In this protocol, we will use "pombe" as the *<INDEX_NAME>* for subsequent procedures. This name will be used to reference the indexed genome in future Bowtie2 alignment commands. It is important to note that the *bowtie2-build* command generates multiple files with extensions such as *.bt2* or *.bt2l*, depending on the size of the genome. These files constitute the index required by Bowtie2 for alignment tasks. When running Bowtie2 to perform alignments, it is necessary to reference these index files by specifying their path and the common prefix (index name). This is done as follows: *${GENOME_DIR}/<INDEX_NAME>*.

2.6 Hi-C Datasets

To obtain a test dataset, download data examining the 3D structure of the fission yeast genome during the cell cycle process [13]. These datasets can be accessed from the Gene Expression Omnibus (GEO) using the accession number GSE93198 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE93198). For this protocol, focus on the asynchronous data, specifically GSM2446268. GEO provides links to the SRA, where sequence read information is stored. For these datasets, SRR5149274 is the accession ID for the sequence run. To download the FASTQ files for these data, use the *fastq-dump* command as follows:

```
cd ${DIR_DATA_RAW}
fastq-dump --split-files SRR5149274 -A wild_type_sample
```

The Hi-C libraries have been sequenced in paired-end mode, resulting in two files representing read1 and read2, named *wild_type_sample_1.fastq* and *wild_type_sample_2.fastq*, respectively. To facilitate the process in this protocol, we will craft the test datasets by extracting five million reads. The extraction can be performed using the following commands:

```
cd ${DIR_DATA_RAW}
head -n 20000000 wild_type_sample_1.fastq > test_1.fastq
head -n 20000000 wild_type_sample_2.fastq > test_2.fastq
```

Note that we use the *head -n 20000000* command to extract the first 20 million lines from each original file, which equates to

five million reads since the information of each read in a FASTQ file spans four lines.

For those who prefer to work directly with a prepared dataset, the files created through the above steps are also accessible for download with the following commands:

```
cd ${DIR_DATA_RAW}
  curl   -O   https://www.igm.hokudai.ac.jp/gacha/data/
test_1.fastq.gz
  curl   -O   https://www.igm.hokudai.ac.jp/gacha/data/
test_2.fastq.gz
```

The files are compressed, as indicated by the *.gz* extension. This protocol allows for the direct use of compressed files in the analysis, saving storage space and processing time. However, if there is a need to decompress these files, the following commands can be used:

```
cd ${DIR_DATA_RAW}
gunzip test_1.fastq.gz
gunzip test_2.fastq.gz
```

It is important to include the *.gz* extension when specifying compressed input FASTQ files. This ensures that the tools correctly recognize and process the files.

2.7 Setting Up the Pipeline

To set up the Hi-C automation pipeline, begin by downloading the rfy_hic2 using the following command. See **Note 5** for how to update the pipeline. In cases where a Docker image is being utilized, this step can be skipped and progress directly to Subheading 2.8:

```
cd ${DIR_LIB}
git clone https://github.com/rafysta/rfy_hic2.git
```

Next, to install the necessary R packages for execution, run *install_libraries.R*. See **Note 6** for troubleshooting installation issues:

```
Rscript --vanilla --no-echo ${DIR_LIB}/rfy_hic2/
install_libraries.R
```

After installing the software listed in Subheading 2.2, ensure it is operational by performing an environment check. See **Note 7** for caution on using double hyphens with command options:

```
bash ${DIR_LIB}/rfy_hic2/rfy_hic2.sh --env_check
```

If error messages appear, it indicates that not all required software is executable from the program. In such cases, either install the missing software or add the folder path containing the executable software to the execution path variable. See **Note 8** for instructions on setting the path in Linux.

2.8 Setting Up Docker Environment

1. **Install Docker Engine**: Follow the official manual to install Docker Engine for the suitable operating system (https://docs.docker.com/engine/install/).
2. **Obtain the rfy_hic2 Docker image**: On the Terminal (for Linux or macOS systems) or Command Prompt (for Windows), referred to as "the terminal" hereafter, execute the command *docker pull rafysta/rfy_hic2:latest*. Replace "latest" with a specific version tag from Docker Hub if necessary.
3. **Run a Docker container from the pulled image**: Execute the command *docker run -v <host_folder>:<container_folder> -it rafysta/rfy_hic2*. The *-v* parameter specifies the directory to be shared between the host machine and the container, facilitating the input and retrieval of files. The *-i* flag keeps the container running interactively to maintain the environment for pipeline execution, while *-t* creates a terminal interface for a better user experience.
4. **Prepare required files**: Copy necessary input files from the folder for data sharing folder to the respective folders described in Subheading 2.3.
5. **Start running the rfy_hic2 Hi-C analysis pipeline**: Follow the subsequent steps to initiate the rfy_hic2 Hi-C analysis pipeline within the Docker environment.

3 Methods

3.1 Automated Pipeline

The Hi-C data processing pipeline described in this protocol features two primary components: the automated matrix creation and visualization. Initially, it begins with sequencing reads, advances to alignment with the reference genome, incorporates various filtering steps, and ultimately generates a matrix in the automation section. The subsequent visualization segment employs this matrix to produce a heatmap illustrating Hi-C genomic contacts. Refer to Fig. 3 for the pipeline overview. The automation section includes five

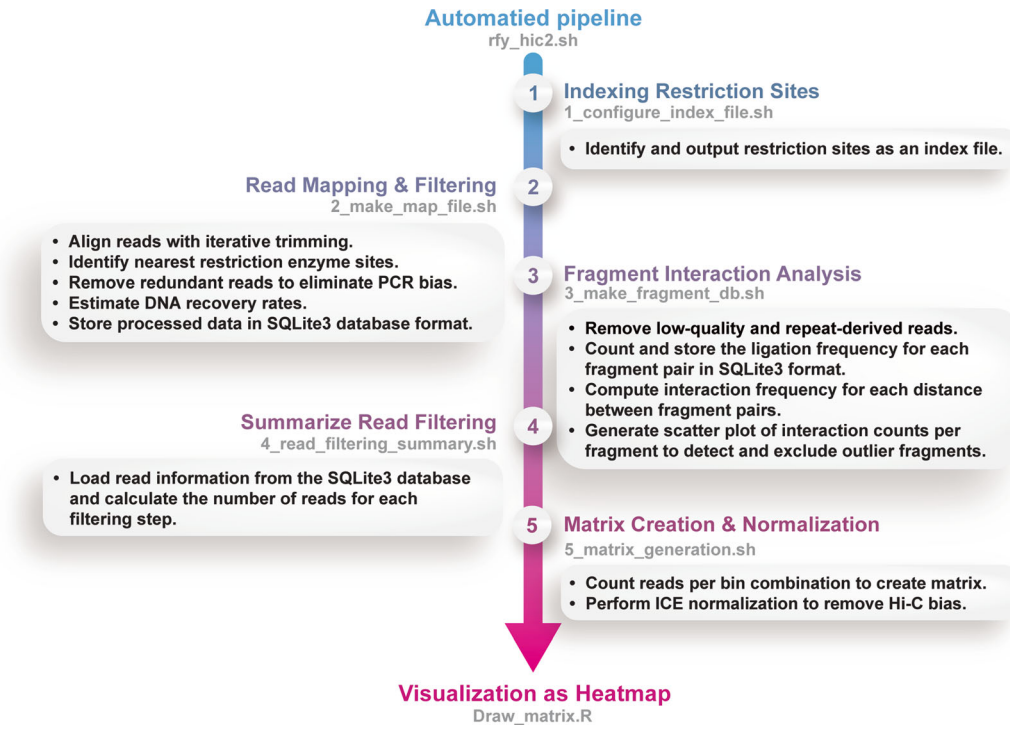

Fig. 3 An overview of the rfy_hic2 pipeline introduced in this protocol article. The pipeline consists of two main components: an automation section, which automatically executes five steps, including read alignment, various filtering processes, and the creation of matrices; and a program for visualization. This protocol article details the five steps that comprise the automated matrix creation, followed by a description of the heatmap visual representation

distinct programs capable of autonomously conducting the entire procedure. This is achieved by setting all parameters in a configuration file (*argfile*), a plain text document that lists the required parameter values for each stage of the analysis (see **Note 9** for details). After preparing the argfile, it is passed as a command-line argument to initiate the automated procedure in the following manner:

```
bash ${DIR_LIB}/rfy_hic2/rfy_hic2.sh --arg <argfile>
```

An example argfile, *argfile_default.env*, is provided in the rfy_hic2 pipeline to illustrate the pipeline configuration. This file can be copied using:

```
cp ${DIR_LIB}/rfy_hic2/argfile_default.env ${DIR_OUT}/
argfile_projectA.env
```

The copied file must be edited with a text editor before using it. Despite the *.env* extension, the content is in plain text format and thus is accessible with any preferred software such as Gedit, Nano, and Vim on Linux and Notepad or Visual Studio Code on Windows. To match the environment, it is necessary to amend the content, especially the full paths for each directory.

Upon editing, execute the command as follows:

```
bash ${DIR_LIB}/rfy_hic2/rfy_hic2.sh --arg ${DIR_OUT}/
argfile_projectA.env
```

The configuration file is applicable even when executing individual stages separately. Furthermore, a *--help* option is available for all subsequent programs, including *rfy_hic2.sh*. For instance, executing

```
bash ${DIR_LIB}/rfy_hic2/rfy_hic2.sh --help
```

allows verification of necessary parameters for command execution. Subsequent sections will explore the purpose and usage of specific programs involved in each step.

3.2 Indexing Restriction Sites

In the Hi-C experiment, DNA is cleaved by restriction enzymes, and ligated fragments are detected. The brief experimental procedure is shown in Fig. 4. The analyzable units are the interactions between fragments delineated by restriction enzymes. To avoid sequencing unnecessary parts, fragments containing ligation junctions are purified after sonication of ligation products. While sequencing reads from the ends of fragments, sonication results in

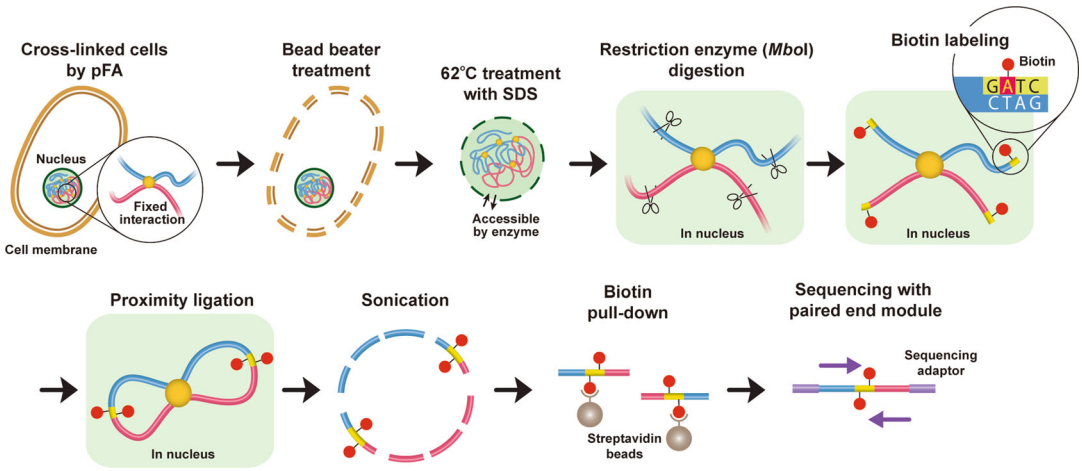

Fig. 4 Hi-C experimental procedure (reproduced from ref. 1 with permission from Springer Nature)

random breaks, meaning the sequenced locations do not always directly correspond to the ends of the original fragments. Therefore, inferring the original fragment information from the alignment location becomes necessary.

Inference involves searching for the nearest downstream restriction enzyme recognition site from the location where reads align, considering the orientation of the read alignment. To efficiently identify the restriction enzyme sites where sequence reads have been cleaved, recognition sites of the restriction enzyme (s) used in the Hi-C experiment are preregistered within the rfy_hic2 pipeline. An index file can be created using the *1_configure_index_file.sh* script as follows:

```
bash ${DIR_LIB}/rfy_hic2/1_configure_index_file.sh --fasta
<fasta file> --restriction <recognition sites for restric-
tion enzyme> --out_site <output file: recognition sites>
--out_site_index <output file: index file>
```

For this protocol, since MboI is used as the restriction enzyme for our test dataset, its recognition sequence, "GATC," should be used as the value for *<recognition sites for restriction enzyme>*. For instructions on specifying multiple restriction enzymes or enzymes that recognize multiple bases such as "N," refer to **Note 10**.

A specific example of input is as follows:

```
bash ${DIR_LIB}/rfy_hic2/1_configure_index_file.sh --fasta
${GENOME_DIR}/all.fa --restriction GATC --out_site ${GEN-
OME_DIR}/pombe_MboI_sites.txt  --out_site_index  ${GENO-
ME_DIR}/pombe_MboI_index.txt
```

3.3 Read Mapping and Filtering

In **step 2** (Read Mapping & Filtering in Fig. 3) of the rfy_hic2 pipeline, reference genome alignment of sequence reads is performed by the Bowtie2 software. A distinctive feature of Hi-C library sequence reads is their enrichment with ligation junctions, indicating that these reads may contain sequences from more than one genomic region. To address this, we implement the following alignment strategy:

We initiate the process by attempting to align reads in their entirety using Bowtie2. The alignment software generates a report called a Concise Idiosyncratic Gapped Alignment Report (CIGAR) string, which indicates how each read is aligned to the reference genome [14]. If a partial alignment is reported, we trim these unaligned sections and attempt alignment again. We iteratively trim five nucleotides from their ends for reads that still do not

align and retry the alignment. This process continues until the reads are fully aligned or until the trimmed read length reaches a minimum threshold of 20 nucleotides. This method maximizes mapping quality, which reflects the accuracy of the alignment. Generally, the longer the aligned segment and the fewer mismatches with the reference, the higher the mapping quality, indicating a more accurate alignment. By incorporating this iterative trimming strategy, we aim to minimize sequence removal, thereby increasing the yield of reads suitable for downstream analysis.

After alignment, we search for the nearest restriction enzyme recognition site to the aligned position, based on the index file prepared in **step 1** (Indexing Restriction Sites in Fig. 3). The details of this procedure are already described in Subheading 3.2 and are omitted here for brevity.

Next, we identify and remove redundant reads potentially derived by PCR. Next-generation sequencers, such as Illumina and MGI, amplify samples via PCR before sequencing. However, this amplification can introduce bias, as not all DNA in the library is equally amplified. Reads that are redundantly amplified do not reflect the 3D structure of the genome and thus must be removed. As mentioned earlier, the Hi-C library is randomly fragmented by sonication, making it highly unlikely for two fragments to be cut at the exact same location by chance. In Hi-C data analysis, we measure the ligation frequency between two fragments, but the probability that both fragments are cut at the same location is presumably extremely low. Therefore, multiple read pairs aligned to identical genomic locations can be deduced as PCR duplicates, where only one pair is thus retained.

The information of the remaining reads is converted and saved in an SQLite3 database format, allowing for fast access to reads matching various criteria later.

These operations can be executed using the following command with the *2_make_map_file.sh* script:

```
bash ${DIR_LIB}/rfy_hic2/2_make_map_file.sh --arg ${DIR_-
OUT}/argfile_projectA.env  --name  ${NAME}  --directory
${DIR_DATA} --f1 ${DIR_RAW}/${FILE_fastq1} --f2 ${DIR_-
RAW}/${FILE_fastq2}
```

Executing this command generates a file named *${NAME}.map.gz* in the *${DIR_DATA}* folder. This file contains information for each read pair, including the aligned chromosome name, coordinates, orientation, the closest restriction enzyme ID and coordinates, and the mapping quality, among other details. This information can be utilized for downstream analysis.

3.4 Fragment Interaction Analysis

In **step 3** (Fragment Interaction Analysis in Fig. 3) of the rfy_hic2 pipeline, reads aligned to repeats or of low quality are removed. This is achieved by excluding reads with a Mapping Quality (MapQ value) below a certain threshold. Mapping quality indicates confidence in the alignment of reads, where a higher value suggests a higher certainty of the read being correctly mapped to its position. Lower MapQ values reflect reads originating from repeats or containing bases that do not match the reference genome. The rfy_hic2 pipeline defaults to a MapQ threshold of 30, implying a 99.9% probability that the read is mapped correctly, or in other words, there is a possibility of incorrect mapping in 1 out of 1000 cases.

In the rfy_hic2 pipeline, we take into account the orientation of aligned reads to evaluate contacts between different DNA fragments. Understanding these orientations is essential to distinguish correct Hi-C ligation products from artifacts, such as self-ligation products. To simplify this, we introduce a more intuitive terminology: "left-left," "right-right," "in-in," and "out-out." See a schematic representation in Fig. 5.

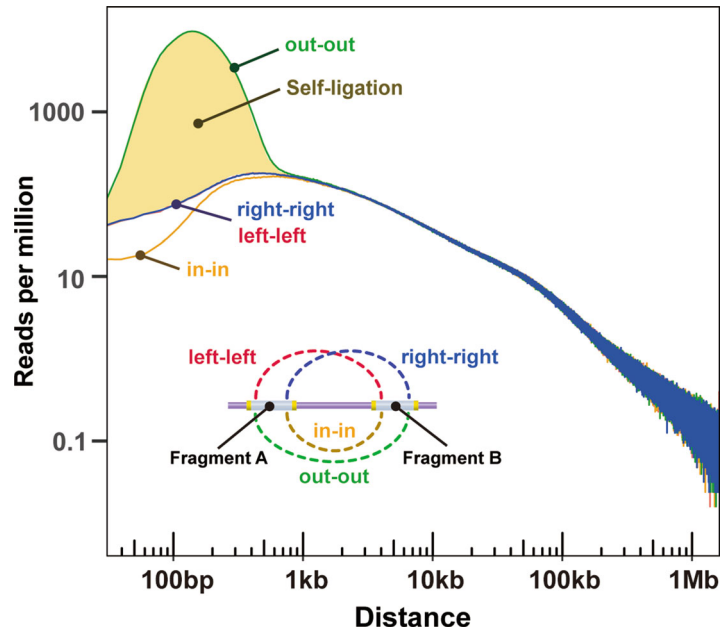

Fig. 5 Relationship between frequencies of the indicated read orientations against distance. This figure illustrates that the "out-out" orientation combinations are observed at higher frequencies compared to "left-left" or "right-right" combinations. This is attributed to the inclusion of both read pairs derived from self-ligation and correct Hi-C ligation products in the "out-out" configuration, whereas the "left-left" and "right-right" configurations are exclusive to correct Hi-C ligation products

"**Left-left**" configuration occurs when both reads of a fragment pair align to the left sides of their respective fragments. This configuration indicates the correct Hi-C ligation products.

"**Right-right**" configuration is observed when both reads align to the right sides of their fragments. This configuration also indicates the correct Hi-C ligation products.

"**In-in**" orientation indicates a read pair where one read aligns to the right side of its fragment and the other read aligns to the left side of its corresponding fragment, suggesting an inward orientation of paired reads. This configuration is a mixture of correct Hi-C products and undigested or religated fragments. The undigested fragments result from the failure of the restriction enzyme to cut the DNA.

"**Out-out**" orientation describes scenarios where one read aligns to the left side of its fragment and the other to the right side, suggesting an outward orientation of paired reads. This pattern indicates a combination of self-ligation products and correct Hi-C ligation products. Self-ligation does not reflect an actual genomic contact because the ends of the same DNA molecule are ligated.

By distinguishing these orientations, the rfy_hic2 pipeline effectively identifies and excludes self-ligation products and impurities, focusing the analysis on genuine Hi-C ligation products. Specifically, the "left-left" and "right-right" configurations generally indicate correct Hi-C ligation products, while the "in-in" and "out-out" configurations may include both correct products and artifacts.

Distinguishing between correct products and artifacts bioinformatically is challenging. Therefore, the rfy_hic2 pipeline employs the following strategy to estimate correct Hi-C products: The number of Hi-C products originating from the same fragment is assumed to be nearly the same across "in-in," "out-out," "left-left," and "right-right" configurations. Given that "in-in" and "out-out" configurations may contain a mixture of true and false products, whereas "left-left" and "right-right" are considered to be accurate, the values for "in-in" and "out-out" are estimated from those of "left-left" and "right-right."

However, beyond a certain distance, self-ligation products are no longer a concern, and reads from "in-in," "out-out," "left-left," and "right-right" configurations all derive from correct Hi-C products, making such estimations unnecessary. Therefore, the pipeline adopts a strategy of directly using the data for distances beyond the self-ligatable range and applying estimates based on "left-left" and "right-right" values for shorter, potentially contaminated distances. For fission yeast Hi-C data processed with MboI, the largest fragment size in a fully digested genome is 11 kb, which serves as

the upper limit of the maximum self-ligation length. However, fragments larger than this can occur if the restriction enzyme does not fully digest the DNA, potentially forming self-ligation products. Therefore, the rfy_hic2 pipeline, by default, conservatively uses 20 kb as the threshold for self-ligation to accommodate these larger, undigested fragments. This threshold may need adjustment based on organisms and restriction enzymes used in Hi-C analysis. Running *3_make_fragment_db.sh* generates a *${NAME}_distance.txt* file, detailing the distance between fragment pairs and their frequency, along with the type of fragment pair. This data can be used to plot graphs like Fig. 5, showing frequencies of observed read orientations against distance, to assess the extent of self-ligation.

The data for the total number of ligations per fragment pair is converted into SQLite3 format for fast access and storage.

To enhance the accuracy of data analysis, the rfy_hic2 pipeline is designed to identify and exclude fragments exhibiting abnormally high read counts, which may represent anomalies rather than reflecting actual genomic contacts. Here is a streamlined explanation of this process:

First, the rfy_hic2 pipeline calculates the total number of reads for each fragment. The pipeline sorts these fragments based on their total read counts, from highest to lowest. Following this sorting, a scatter diagram is plotted with the fragment index (sorted by total read count) on the x-axis and the total reads for each fragment on the y-axis. This visualization generally reveals an almost linear trend. However, within this general trend, a small number of fragments will be noticeable for their abnormally high read counts markedly diverged from the expected distribution. These outliers, typically fewer than ten fragments in total, are considered anomalies by the rfy_hic2 pipeline and are subsequently excluded from the analysis.

Their removal is crucial because their disproportionate read counts could potentially distort the global interpretation of the 3D genomics data, leading to inaccurate conclusions about an overall genomic contact pattern.

These operations can be executed by typing the following command using the script *3_make_fragment_db.sh*:

```
bash ${DIR_LIB}/rfy_hic2/3_make_fragment_db.sh --arg
${DIR_OUT}/argfile_projectA.env  --directory  ${DIR_DATA}
--in ${DIR_DATA}/${NAME}.map.gz --name ${NAME} --thresh-
old 20000 --mapq 30
```

Here, *${NAME}.map.gz* is the file produced by the script *2_make_map_file.sh* from **step 2** (Read Mapping & Filtering in Fig. 3). If data from biological replicates are available, specify all

the *.map.gz* files connected by commas, such as *name1.map.gz, name2.map.gz,name3.map.gz*. It is crucial to note that if these are technical replicates, data derived from repeated sequencing of the same Hi-C library, the files should be merged beforehand to prevent the same read pairs from being counted more than once. For instructions on merging, refer to **Note 11**. The *--threshold* option specifies the maximum distance for self-ligation reads, while *--mapq* sets the minimum mapping quality threshold.

3.5 Summarize Read Filtering

In **step 4** (Summarize Read Filtering in Fig. 3) of the rfy_hic2 pipeline, the number of sequence reads is analyzed, showing how they decrease through various filtering steps and how many reads are available for genomic contact analysis. This step is executed using the *4_read_filtering_summary.sh* script as follows:

```
bash ${DIR_LIB}/rfy_hic2/4_read_filtering_summary.sh --
arg ${DIR_OUT}/argfile_projectA.env --directory ${DIR_-
DATA} --out ${DIR_OUT}/read_summary.txt <NAME 1> <NAME
2> <NAME 3> ... <NAME n>
```

For the positional arguments in this command, specify the names of all samples separated by spaces.

3.6 Matrix Creation and Normalization

In **step 5** (Matrix Creation and Normalization in Fig. 3), the rfy_hic2 pipeline generates a Hi-C matrix based on the number of genomic contacts per fragment, as determined in **step 3** (Fragment Interaction Analysis in Fig. 3). The genome is initially partitioned into fixed-length segments, known as bins. Given the potential alignments of reads, genomic contacts can occur in four distinct combinations: "in-in," "out-out," "left-left," and "right-right." Recognizing that interactions among fragments are nearly equal in frequency, the count of contacts is divided by four to distribute this total evenly across the four contact combinations. Subsequently, for each type of contact, the positions on the left and right sides of both fragments 1 and 2 are mapped to their corresponding bins based on the outermost coordinates of the fragments. The rfy_hic2 pipeline then aggregates the contact scores within each bin to construct a raw matrix.

The raw matrix is likely influenced by several biases derived from factors such as fragment length, GC content, and mappability [15], especially in high-resolution analyses. To ensure accurate data interpretation, these biases must be removed. The ICE method, as proposed by Imakaev et al. [16], is employed by the rfy_hic2 pipeline to iteratively normalize the data, ensuring each row and column of the matrix has the same total number of interactions. This method aims to remove potential over- and

underrepresentation of specific genomic regions. In the iterative process used by the rfy_hic2 pipeline, the change in bias factors is measured, typically converging after about 10 iterations for fission yeast Hi-C data. However, the pipeline default is set to perform 30 iterations to guarantee complete convergence.

The command for executing the operations above is as follows:

```
bash ${DIR_LIB}/rfy_hic2/5_matrix_generation.sh --arg
${DIR_OUT}/argfile_projectA.env  --directory  ${DIR_DATA}
--name ${NAME} --resolution 20kb
```

The matrix is made available in two formats: as a tab-separated text file and as an R object.

3.7 Visualization as Heatmap

For visualization of a Hi-C matrix, extracting data for specific regions of interest and representing the interaction strengths in a heatmap format is common. The rfy_hic2 pipeline provides a script named *Draw_matrix.R*, which can be used for visualization as follows:

```
Rscript --vanilla --no-echo ${DIR_LIB}/Draw_matrix.R -i
<matrices file> --chr <chromosome name> --start <start
coordinate> --end <end coordinate> --out <output png
image file name>
```

Here, <*matrices file*> refers to the matrix file outputted in **step 5** (Matrix Creation & Normalization in Fig. 3), which is either in text format or in an R object format that has been normalized using the ICE method (e.g., *${DIR_DATA}/${NAME}/20kb/ICE2/ALL.rds*). For <*chromosome name*>, specify the name of a specific chromosome (e.g., "II") or use "all" for interactions across all chromosomes. The <*end coordinate*> can specify a particular location or "all" to automatically use the end of the chromosome.

An example of a specific command is:

```
Rscript --vanilla --no-echo ${DIR_LIB}/Draw_matrix.R -i
${DIR_DATA}/${NAME}/20kb/ICE2/ALL.rds --chr all --start
1 --end all --out ${DIR_OUT}/${NAME}_20kb.png
```

An example of the output produced by this command is shown in Fig. 6.

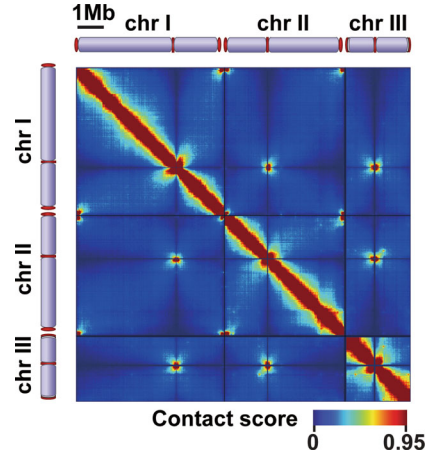

Fig. 6 Hi-C contact heatmap generated using this protocol

4 Notes

1. Linux system can be used on a Windows machine via the "Windows Subsystem for Linux" (WSL). To install WSL, search "Turn Windows features on or off" in the Start menu. Check both the "Windows Subsystem for Linux" and "Virtual Machine Platform" options. Reboot the system to apply the changes. Once the system is back on, open PowerShell as an administrator by right-clicking the Start menu and selecting the relevant option. In the PowerShell window, execute the command *wsl --install*. This command installs both WSL and the Ubuntu distribution by default.
2. Software prerequisites can be installed using the APT package management tool with the command *sudo apt install <software_name(s)>* on an Ubuntu Linux system with root access. An example command for installing *sqlite3*, *fastqc*, *bowtie2*, and *samtools* is *sudo apt install sqlite3 fastqc bowtie2 samtools*. For R, before running *apt install*, add the R repository GNU Privacy Guard (GPG) key by *sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E298A3A825C0D65DFD57CBB651716619E084DAB9* and include the R repository in the APT repository source list with "*sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu focal-cran40/'*." Before each *apt install*, update the package list by *sudo apt update*. SRA-Toolkit is not managed by a standard APT repository, but it can be downloaded from the git repository using the link we described in Subheading 2.2 prerequisites.
3. For direct execution of the rfy_hic2 pipeline in a macOS environment, prerequisite software can be installed via Homebrew with the command *brew install <software_name>*. Please refer to the official Homebrew website (https://brew.sh/) for an

explanation of installation and usage. We list the relevant formulae here: bash, perl, r, sqlite, sratoolkit, fastqc, and samtools.

4. In a UNIX-like environment (Linux or macOS), any string can be stored in a variable. When assigning a value to a variable, use the "=" operator without spaces before or after it, as in *VAR=CONTENTS*. This means that the string VAR stores the string CONTENTS. To use a variable in programs or the terminal, prefix it with a "$" symbol as *$VAR* or *${VAR}* to reference the content of the variable. Throughout this protocol, the expression *${VAR}* is frequently used; when this expression is found, replace it with the path defined in Subheading 2.3.
5. To update the rfy_hic2 package, there is no need to download it again. Instead, the library can be updated using the *git pull* command as follows:

```
cd ${DIR_LIB}/rfy_hic2
git pull origin main
```

This command synchronizes a local copy with the latest changes from the main branch, ensuring that the most recent version of the package is installed without a complete re-downloading.

6. If a message indicating failure to install due to lack of write permissions in the install directory occurs, execute the command with administrative privileges. Specifically, run the following command:

```
sudo Rscript --vanilla –no-echo ${DIR_LIB}/rfy_hic2/
install_libraries.R
```

The *sudo* command elevates permissions to the administrator level for the installation process, ensuring the necessary libraries are installed without permission issues.

7. In command-line operations, the distinction between single and double hyphens before option names is crucial for accurate command execution. Follow these guidelines:
   - **Multi-character options**: Use two hyphens for options with names longer than one character, for example, *--env_check*.
   - **Single-character options**: Use one hyphen for options with a single-character name, for example, *-o*.

     This differentiation is important to observe as it directly affects how the command interprets the options provided.

8. In a Linux environment, to directly execute software without specifying its full path, adding the software installation path to the PATH environment variable is necessary. For example, only *bowtie2* is typed rather than */usr/local/Software/bowtie2/2.4.5/bowtie2*. This is done by the following command:

```
export PATH=/usr/local/Software/bowtie2/2.4.5:$PATH
```

In this command, *:$PATH* appends the existing content of the *PATH* variable, ensuring that previously added directories are preserved. This mechanism is crucial because if the *PATH* contains multiple directories with executables of the same name, the system searches these directories in the order listed in the *PATH*. The first executable found that matches the command is executed, reflecting a left-to-right search order in the *PATH*. By prepending a directory to the *PATH* as shown, it should be prioritized over default program locations in the system. This approach is particularly useful when working with specific versions of software that need to be prioritized.

To verify that the *PATH* has been correctly set and that the system recognizes a command without specifying its full path, the *which* command followed by the command name can be used, for example:

```
which bowtie2
```

This command returns the path to the executable if found in the directories listed in the PATH, confirming that the command can be executed directly.

Note on Making Changes Permanent

The export command shown above applies only to the current terminal session. To make the change permanent across all future sessions, add the export command to the shell initialization file, such as *~/.bashrc* or *~/.bash_profile*, depending on the shell in use. This ensures that the specified directory is always included in the *PATH* environment variable, simplifying software execution and workflow management in Linux environments.

9. The configuration file (*argfile.env*) is a plain text file that can be freely edited with a text editor. It lists the values for parameters required at each stage, following the method of setting variables in Linux as shown in **Note 4**. Parameters are listed on the left, an "=" is placed in the middle, and the content of the parameter to be set is on the right, with no spaces before or after the "=." If the same parameter is set more than once, the

parameter listed later in the file will be used. Even if there are parameters not used in a specific stage of the five-step rfy_hic2 pipeline, they are ignored. Therefore, there is no need to prepare separate parameter files for each stage. If the same parameter is provided as an option to the command, the value for the corresponding command option is used. Listing parameters commonly used across multiple projects in this file and providing frequently changed items like sample names as command options makes the pipeline easier to operate.

10. In setting up *1_configure_index_file.sh* with parameters for multiple restriction enzymes, every base sequence recognized by the enzymes should be listed, separating them with commas. For enzymes that recognize sequences with multiple possible bases (indicated by "N"), it is necessary to enumerate all combinations. For instance, the enzyme HinfI recognizes "GANTC," where "N" can be any nucleotide. Therefore, "GAATC,GATTC,GAGTC,GACTC" should be listed in the configuration as *--restriction*, covering all potential recognition sequences.
11. To merge two or more *.map.gz* files from technical replicates, sequences generated from the same Hi-C library, execute the following command:

```
bash ${DIR_LIB}/rfy_hic2/merge_map_technical_replicate.
sh --arg ${DIR_OUT}/argfile_projectA.env --in name1.map.
gz,name2.map.gz,name3.map.gz --directory ${DIR_DATA} --
name merged_replicates
```

In this example, *name1.map.gz,name2.map.gz,name3.map.gz* is the *.map.gz* files from technical replicates specified as input. These files should be listed in the command without spaces between the commas. The *--arg* flag points to the environment argument file, *--in* is used to specify the input files, *--directory* sets the directory where the output will be saved, and *--name* defines the name for the merged output data.

Acknowledgments

We would like to thank Michiko Kanbayashi, Makiko Fukuuchi, and Yuko Tsukamoto for their generous support in the laboratory. This work was supported by the National Institutes of Health/National Institute of General Medical Sciences R01GM124195, JSPS KAKENHI grant JP20K23376, Grant for Basic Science Research Projects from the Sumitomo Foundation, the Uehara Memorial Foundation, Takeda Science Foundation, the Mitsubishi

Foundation, the NOVARTIS Foundation (Japan) for the Promotion of Science, and the Photo-excitonix Project in Hokkaido University and the Grant for Joint Research Program of the Institute for Genetic Medicine, Hokkaido University (to KN).

References

1. Dekker J, Misteli T (2015) Long-range chromatin interactions. Cold Spring Harb Perspect Biol 7:a019356
2. Dixon JR, Gorkin DU, Ren B (2016) Chromatin domains: the unit of chromosome organization. Mol Cell 62:668–680
3. Noma K-I (2017) The yeast genomes in three dimensions: mechanisms and functions. Annu Rev Genet 51:23–44
4. Lieberman-Aiden E, van Berkum NL, Williams L et al (2009) Comprehensive mapping of long-range interactions reveals folding principles of the human genome. Science 326:289–293
5. Dekker J, Rippe K, Dekker M et al (2002) Capturing chromosome conformation. Science 295:1306–1311
6. Rowley MJ, Corces VG (2018) Organizational principles of 3D genome architecture. Nat Rev Genet 19:789–800
7. Szabo Q, Bantignies F, Cavalli G (2019) Principles of genome folding into topologically associating domains. Sci Adv 5:eaaw1668
8. Andrews S, others (2010) FastQC: a quality control tool for high throughput sequence data. Babraham Bioinforma Babraham Inst Camb U K
9. Danecek P, Bonfield JK, Liddle J et al (2021) Twelve years of SAMtools and BCFtools. GigaScience 10
10. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods 9:357–359
11. Rutherford KM, Lera-Ramírez M, Wood V (2024) PomBase: a Global Core Biodata Resource-growth, collaboration, and sustainability. Genetics iyae007
12. Wood V, Gwilliam R, Rajandream M-A et al (2002) The genome sequence of Schizosaccharomyces pombe. Nature 415:871–880
13. Tanizawa H, Kim K-D, Iwasaki O et al (2017) Architectural alterations of the fission yeast genome during the cell cycle. Nat Struct Mol Biol 24:965–976
14. Li H, Handsaker B, Wysoker A et al (2009) The sequence alignment/Map format and SAMtools. Bioinforma Oxf Engl 25:2078–2079
15. Yaffe E, Tanay A (2011) Probabilistic modeling of Hi-C contact maps eliminates systematic biases to characterize global chromosomal architecture. Nat Genet 43:1059–1065
16. Imakaev M, Fudenberg G, McCord RP et al (2012) Iterative correction of Hi-C data reveals hallmarks of chromosome organization. Nat Methods 9:999–1003