EPIGENOMICS
2025, VOL. 17, NO. 17, 1233–1247
https://doi.org/10.1080/17501911.2025.2568368

Taylor & Francis
Taylor & Francis Group

METHOD

SLUR(M)-py: a SLURM powered Pythonic pipeline for parallel processing of 3D (Epi) genomic profiles

Cullen Roth, Vrinda Venu, Sasha Bacot, Shawn R. Starkenburg and Christina R. Steadman

Los Alamos National Laboratory, Genomics and Bioanalytics, Los Alamos, NM, USA

ABSTRACT

Epigenomics has become multi-faceted, with researchers exploring chromatin structure, nucleosome states, and epigenetic modifications, producing large, complex multi-omic datasets. Given this shift, there is a demand for bioinformatics that leverage high-performance computing (HPC) and parallelization to quickly process data. As such, we developed SLUR(M)-py, a pythonic computational platform that leverages the Simple Linux Utility for Resource Management system (SLURM) to process sequencing data. SLUR(M)-py is multi-omic and automates calls to SLURM for processing paired-end sequences from chromatin characterization experiments, including whole-genome, ChIP-seq, ATAC-seq, and Hi-C, thereby eliminating the need for multiple analytics pipelines. To demonstrate SLUR(M)-py's utility, we employ ATAC-seq and Hi-C data from viral infection experiments and the ENCODE project, and illustrate its processing speed and completeness, which outpaces current HPC pipelines. We explore the effect of dropping duplicate sequenced reads in ATAC-seq, demonstrate how SLUR(M)-py can be used for quality control, and how to detect artifacts in Hi-C from viral infection experiments. Finally, we show how features in SLUR(M)-py, like inter-chromosomal analysis, can be used to explore the dynamics of chromosomal contacts in mammalian cells. This multi-omic, system-agnostic platform eases the computational burden for researchers and quickly produces accurate and reliable data analytics for the epigenomics community.

PLAIN LANGUAGE SUMMARY

SLUR(M)-py (pronounced slurpy) is a computer program that makes it easier and faster to analyze DNA data on powerful supercomputers. Researchers often use DNA sequencing to study biological phenomena, like how genes are turned on or off, how proteins interact with DNA, or how viruses affect their hosts. Normally, this analysis requires juggling different tools, but SLUR(M)-py puts everything into one simple workflow.

What it does: SLUR(M)-py takes huge DNA data files from sequencing experiments, breaks them into smaller pieces, compares them to a reference genome, filters the results, and puts everything back together. It can handle experiments with multiple samples, pick up where it left off if something goes wrong, and automatically create quality check reports.

Why it is useful: SLUR(M)-py works for DNA data from any species, not just humans. It gives researchers one consistent, reliable way to process their data, saving time, and reducing errors. Because it is written in Python (a common coding language), it is easy to customize.

Bottom line: SLUR(M)-py is a supercomputer-friendly tool that helps scientists quickly and reliably turn raw DNA sequencing data into usable, high-quality results.

ARTICLE HISTORY
Received 16 April 2025
Accepted 25 September 2025

KEYWORDS
Bioinformatics; epigenetics; genomics; Hi-C; ATAC-seq; SLURM; Python; parallelization

1. Introduction

Within the nucleus, the shape, looping, modification, and interactions between chromatin are critical to cellular function, development, and for mounting responses to environmental stimuli [1–14]. A myriad of sequencing technologies, based on paired-end and short-read sequencing, are designed to probe various characteristics of chromatin, including conformation, chemical modification, and resulting gene expression. Assays including chromatin conformation capture followed by sequencing (Hi-C) and for transposase-accessible chromatin with sequencing (ATAC-seq) are used to study the three-dimensional (3D) architecture and the accessibility of DNA within cells, respectively [15–19]. Additionally, chromatin immunoprecipitation (ChIP-seq) assays probe the genome for specific chemical modifications made to histone proteins, simplistically indicative of permissive or repressive chromatin states [20–22]. Together, each of these sequencing paradigms provides complementary information important to researchers when exploring a hypothesis pertaining to epigenetics. Thus, the ability to quickly integrate, characterize, and standardize data from these sequencing approaches in epigenomic investigations (including their complete processing, analysis, and visualization) is critical to understand chromatin dynamics during cellular development and responses to external stimuli.

CONTACT Cullen Roth croth@lanl.gov Los Alamos National Laboratory, Genomics and Bioanalytics, PO Box 1663, Mail Stop C342, Los Alamos, NM 87545, USA

Supplemental data for this article can be accessed online at https://doi.org/10.1080/17501911.2025.2568368

Article highlights

- Introducing SLUR(M)-py (pronounced slurpy), a Pythonic command-line pipeline that leverages SLURM to process, align, and analyze paired-end sequencing from 3D genomics and epigenomic assays in high-performance computing (HPC) environments.
- HPC-ready by design: the platform auto-generates and submits bioinformatic jobs to SLURM that parallelize processing of large datasets.
- Open-source and installable on typical HPC clusters enabling reproducible deployment and extension by the scientific community.
- Delivers a single workflow with multi-omic capability, able to process whole-genome, ChIP-seq, ATAC-seq, and Hi-C—reducing the need for multiple disparate pipelines.
- Compatibility, SLUR(M)-py can easily generate common bioinformatic file formats (.bam, .hic, or .mcool) as outputs, allowing this tool to drop in to existing data analysis protocols.

While each genomic sequencing strategy has specific benefits, caveats, and requirements for analysis, the expected output of most genome-wide sequencing assays is the same: paired-end sequencing data. From the perspective of a bioinformatician, across experiments and assays, the processing of paired-end sequencing data is similar between the aforementioned genomic and epigenomic assays. Processing of both types of data requires trimming of adaptors from reads, removal of low-quality reads (such as reads with repetitive, low-complexity sequences), alignment to a reference genome, marking and removal of duplicate alignments, filtering of poorly aligned or non-unique alignments, and removal of artifacts. As such, co-processing these data through a single informatics pipeline can simplify downstream analysis, visualization, and biological interpretation. Individual single-omic software tools and pipelines have been written for this purpose. For example, Hi-C data is traditionally analyzed through the Juicer pipeline, which has automated the analysis and generation of interacting chromatin maps (.hic files) from paired-end sequencing files to characterize chromatin loops and topologically associated domains (TADs) [23,24], both important features to report when presenting Hi-C data. HiCExplorer and Fan-c, which are Python-based and have command-line functionality, offer the research community alternative pipelines for generating Hi-C maps, data analysis, and visualization [25–27]. For epigenomic assays like ATAC-seq and ChIP-seq, the ENCODE project has guidelines and recommended software for experiments and subsequent generation of sequencing files (.bam) for appropriate, high-quality analysis [18,22]. Additionally, software such as ATAC-graph provides automated processing, diagnostics, and quality control plots of ATAC-seq data [28]. Investigators will often cobble these tools together with other bioinformatic tools to analyze the myriad of sequencing data that is generated within a single study [29,30]; while sound, this can be challenging, time-consuming, inefficient, and fraught with the potential for (human) error. Moreover, the amalgamation of disparate pipelines for the analysis of data from the same set of experiments can generate inconsistencies and incompatibilities.

One feature of many of these pipelines is their "Pythonic" nature, that is, many are written in Python. Python is an ideal coding language to process paired-end sequencing reads from epigenomic and chromatin conformation experiments given its low barrier to entry, flexibility, and open-source nature. While these pipelines leverage Python – and are therefore accessible – they are only equipped to process single-omic data types. Further, in many instances, these pipelines do not scale with larger datasets. As such, using these tools to process numerous omics datasets that can be quite large (GB) from human cell lines or model systems can prove problematic, when more than one billion reads are recommended for fully resolved genomes [4,31].

To analyze large cohorts of sequencing data, like Hi-C from several experiments, or multi-layered omics data, researchers regularly rely on a high-performance computing (HPC) environment to quickly process data; these include popular HPC environments such as Amazon Cloud Computing and Nextflow [32]. Alternatively, web-based applications have also been developed for the processing of sequencing data [25,33–36]. One of the most popular open-source software tools, designed to manage a high-performance computing environment, is the Simple Linux Utility for Resource Management system (SLURM) [37]. As its name implies, SLURM is simple to use and scalable for large computational jobs [38]. Given these features and the user-friendly nature of SLURM, it has become an accepted and integral tool for HPC, capable of managing small to large servers. SLURM empowers users to submit and run parallel jobs while also establishing dependencies, allowing for the creation of truly end-to-end, automated pipelines [38]. For example, the Juicer pipeline leverages SLURM to manage, submit, and run parallel processes for generating .hic files from Hi-C experiments [23]. Additionally, the Hi-C-Pro pipeline contains a parallel mode which can submit jobs to SLURM to create and run parallel bioinformatic tasks [39]. To our knowledge, there is currently no other published HPC enabled pipeline for the processing and analysis of most other sequencing data from chromatin characterization experiments such as whole-genome, ChIP-seq, or ATAC-seq.

To fill this gap, we combine the accessibility of Python with the utility and scalability of the SLURM environment to provide the epigenomics community with a flexible pipeline for analyzing large (GB) multi-omics datasets. We call it SLUR(M)-py (pronounced slurpy; the M is silent) [40]: this Python-based pipeline is designed for an HPC environment that processes the alignments, filtering, and analysis of various types of paired-end sequencing data (genomics and epigenomics). This Pythonic pipeline auto-generates and submits scripts to SLURM; SLURM then schedules and parallelizes the bioinformatic processes underlying the analysis of omics data, greatly improving the speed of common bioinformatic tasks (by a factor of 8). The novelty of SLUR(M)-py is its ability to process paired-end sequencing data from several *different* sequencing strategies, including whole-genome, ATAC-seq, ChIP-seq, and Hi-C sequencing. While SLUR(M)py generates quality control metrics and useful images/graphs for diagnostic analysis and publication, the pipeline also includes several conversion functionalities; these features allow users to integrate this pipeline into current lab protocols and leverage the outputs from SLUR(M)-py for further analysis.

Here, we demonstrate the utility of our SLUR(M)-py platform in analyzing genomics and epigenomics data using our own

ATAC-seq and Hi-C data [41,42] and Hi-C data from the ENCODE project [43]. We compare the outputs generated by SLUR(M)-py to previous *in silico* efforts as well, including HiCExplorer and Juicer. We show that analysis with SLUR(M)-py greatly improves computational speeds, outpacing previous informatic analysis. Moreover, the data products produced from SLUR(M)-py qualitatively and quantitatively compare to previous outputs made using the Juicer pipeline. Finally, we show that outputs from SLUR(M)-py allow for granular interrogation of contacts between chromosomes as a function of viral infection. This finer resolution analysis reveals a new finding from these data: viral infection does not significantly impact intra-chromosomal contacts, which was missed using other pipelines. Re-analysis of our data also demonstrates additional tools that are democratized within the SLUR(M)-py pipeline, including inter-chromosomal contact frequency analysis.

2. Method

2.1. SLUR(M)-py pipeline overview

SLUR(M)-py combines the best of currently available pythonic and bioinformatic tools and provides parallelization for quickly aligning and processing sequenced reads for studying chromatin structural dynamics and epigenomic modifications. These tools include bio-Python [44], Pandas and Dask Dataframes [45], MACS2 [46–48], fastp [49], and bwa [50]: all were specifically chosen for their speed, efficiency, community familiarity, and capabilities. We combined these tools to form a single pipeline to process whole-genome sequencing, ChIP-seq, ATAC-seq, and Hi-C data types. At the command line, SLUR(M)-py only requires a set of paired-end reads and a path to a reference file (in .fasta format) that has been indexed via bwa Figures 1(A) and 2(A). Like the Juicer pipeline, SLUR(M)-py encodes submissions of scripts to SLURM to efficiently manage parallelized processes within the pipeline (Table 1), first parallelizing across the input read pairs and then the input genome (Figure 1(A)). However, SLUR(M)-py includes additional scripts to quickly process information that is complementary to Hi-C, such as ATAC-seq data (Figure 1(A,B), Supplementary Figure S1). Quality control and diagnostic plots are also automatically generated when ATAC-seq or Hi-C datasets are provided (Figure 3, Supplementary Figure S4).

The computing environment and dependencies SLUR(M)-py requires are listed on the hosting GitHub repository and

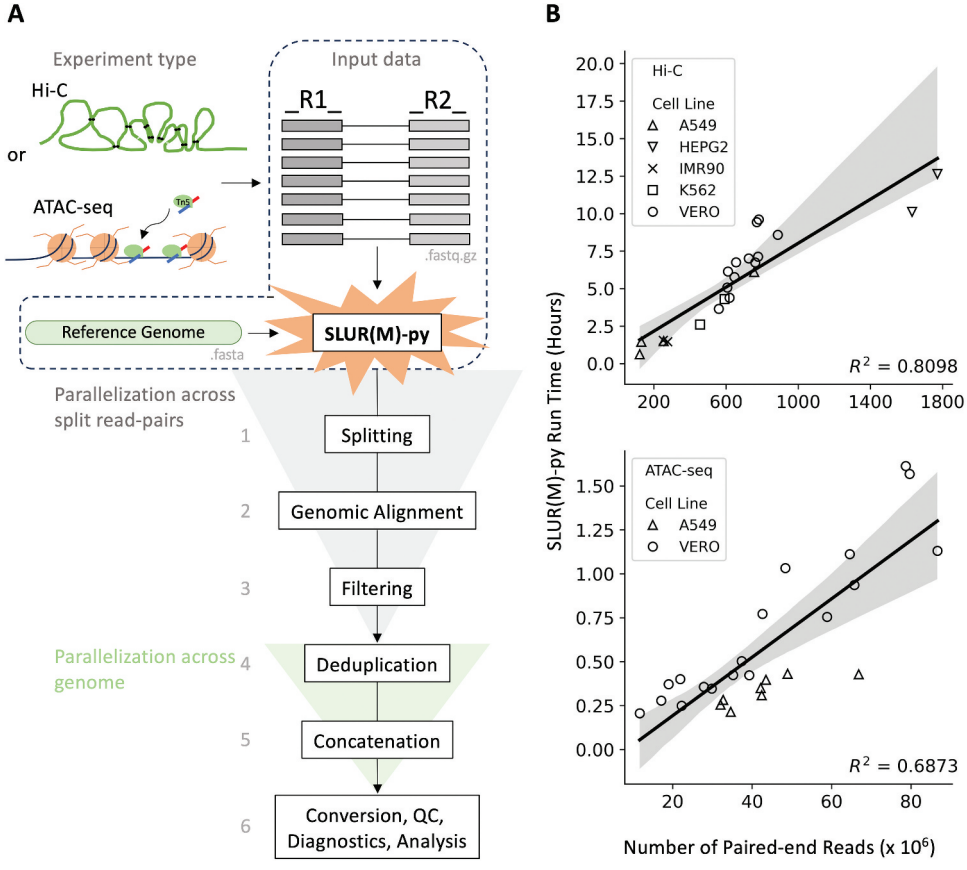

Figure 1. Overview of SLUR(M)-py, a high-performance computing, multi-omic platform for parallelized processing of paired-end reads to study chromatin dynamics. (A) the basic SLUR(M)-py workflow for processing paired-end sequenced reads from Hi-C or ATAC-seq or experiments. Hi-C measures interacting chromatin in 3D space while ATAC-seq targets open regions of the genome with Tn5, generating paired-end sequencing reads. Paired reads (in .fastq.gz format) and a reference genome (.fasta) are the minimum required inputs to SLUR(M)-py. The SLUR(M)-py pipeline generates jobs and submits them to SLURM which manages and runs tasks for splitting input reads, parallelized genomic alignments, and downstream processes, including filtering, deduplication, and file concatenation. SLURM also sets needed dependencies for file conversion, QC, diagnostics, and further analysis. (B) The run times (y-axis) of Hi-C and ATAC-seq processing (top and bottom, hours and minutes, respectively) as a function of the number of paired-end reads (x-axis). The cell lines of each sample are depicted as circles (Vero) or x's, triangles, and squares (human cell lines). Black lines and gray shaded regions represent linear regression models and their 95% confidence intervals (respectively); these models explain 80.98% and 68.73% (R^2) of the variation in run time (top and bottom, respectively).

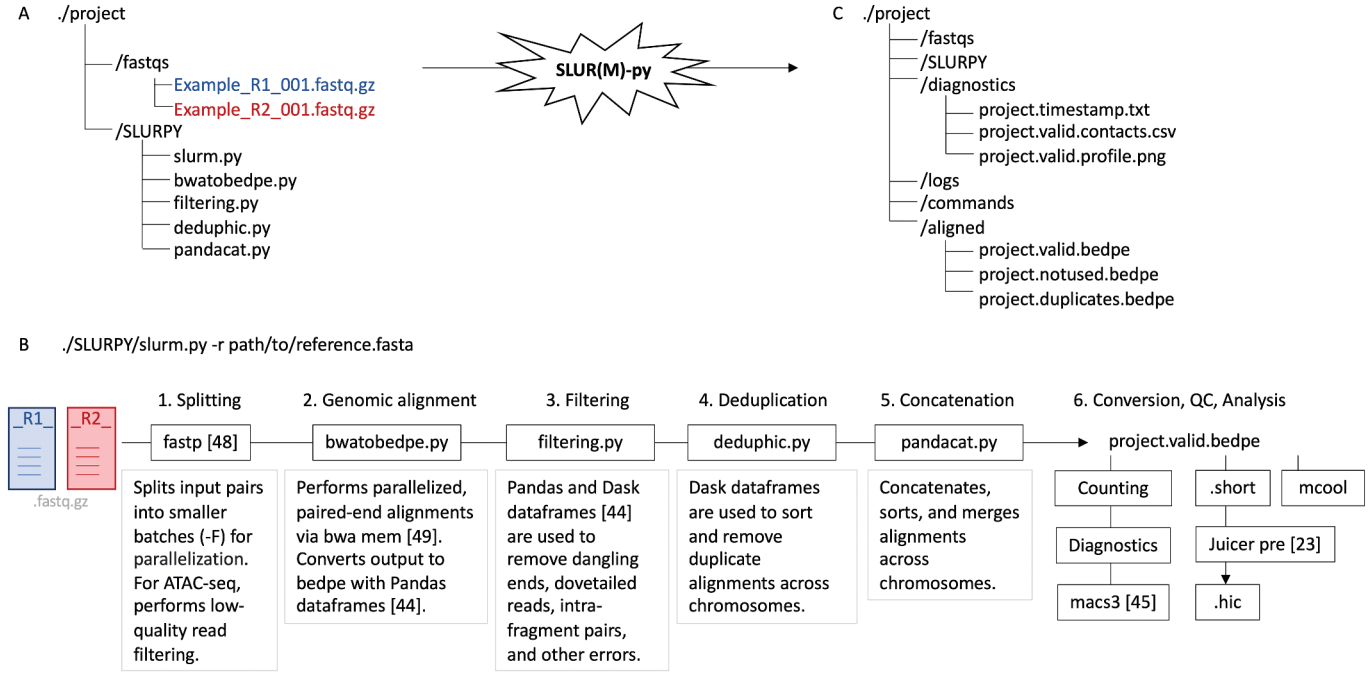

Figure 2. Example project directory and outline of SLUR(M)-py pipeline. A) Example project directory structure. Paired fastq.Gz files (within the fastqs subdirectory) contain paired-end reads as input into SLUR(M)-py. The directory contains all the scripts and functions associated with SLUR(M)-py and can be soft-linked as a subdirectory. B) The processing order of scripts within slurm.Py pipeline. To begin, paired fastqs.Gz files are split (with fastp) into smaller sets for parallelization. When processing ATAC-seq data, fastp conducts initial quality control, removing low-quality reads, and produces diagnostic reports. Across splits of paired fastqs.Gz files, paralleled calls to bwa mem align paired reads to a reference genome (set by the -r argument) are reformatted into bedpe mode (with bwatobedpe.Py). A filtering script (filtering.Py) segregates alignments into valid and error set pairs. Post alignment and filtering, deduphic.Py identifies and removes duplicates, sorting and merging outputs per chromosome. Genome-wide, files are concatenated (with pandacat.Py) into final outputs for post processing, including counting, file conversion, and analysis. C) After running SLUR(M)-py, diagnostic, logs, and aligned directories are generated containing error logs, quality control plots, and final outputs (respectively).

Table 1. Processing features within SLUR(M)-py and published Hi-C pipelines.

	SLUR(M)-py	Juicer [23]	HiCExplorer [27]	Fan-c [26]	HiC-Pro [39]
Multi-omic	Yes	No	No	No	No
SLURM management	Yes	Yes	No	No	Yes
Pythonic basis & filtering	Yes	No	Yes	Yes	Yes
Replicate separating	Yes	No	No	No	Yes
Arima compatible	Yes	Yes	Yes	No	Yes
Dovetail compatible	Yes	Yes	No	No	Yes
Enzymatic libraries[a]	Yes	Yes	Yes	Yes	Yes
Calls *bwa mem* with the –5SPM option[b]	Yes	Yes	No	Yes	No

[a]Restriction site based libraries: MboI, DpnII, Sau3AI, and HindIII.
[b]The –5SPM option sets the *bwa mem* command to ignore pairing, skip mate rescue, mark shorter split hits as secondary, and take the alignment with the smallest coordinate as the primary alignment. Often referred to as the"Hi-C option".

can be easily installed via Python's package, dependencies, and environment manager, conda [51]. When running SLUR(M)-py on a computing cluster, no specific node type is required. Scripts generated by SLUR(M)-py and submitted to SLURM are node agnostic and do not require a graphical processing unit (gpu) to perform tasks. By default, SLUR(M)-py will instruct SLURM to submit jobs to terabyte (tb) node(s); however, any type or combination of node(s) (or a list of nodes) may be specified with the -P argument in calls to SLUR(M)-py. Alternatively, specific nodes can be listed by name via the –node-list argument. After installation, SLUR(M)-py can be linked within a given project directory (Figure 2(A)). This project directory must contain a subdirectory named "fastqs" which holds the paired fastq files (in gzipped compressed format [52]) from a sequencing experiment. Multiple replicates (i.e., more than one pair of fastq files) may be held within this directory. Unlike other pipelines (for example, see Table 1), the SLUR(M)-py workflow can process multiple replicates (by adding the –nomerge flag at the command line) or merge replicates into a final output throughout processing, which is the default behavior. All of these address challenges we have experienced in analyzing (epi)genomics datasets and exist as improvements to currently available analysis pipelines.

To begin a run, the alignment script within SLUR(M)-py (./SLURPY/slurm.py) requires 1) a path to a reference file in . fasta format, set by the -r argument and 2) the presence of gzipped, fastq files (fastq.gz) within the fastqs directory (Figure 2(B)). By default, SLUR(M)-py expects paired-end reads that represent the sequenced material from Hi-C experiments. This behavior can be modified by passing the

additional flags –wgs, –atac-seq, or a path to control.bam file (via -c) to slurm.py at the command line, initiating processing of data from whole-genome sequencing, ATAC-seq, or ChIP-seq experiments, respectively (Supplementary Figure S1). A full list of command-line options and modifications is listed on the associated SLUR(M)-py GitHub or visible via the help menu (–h). After passing command-line arguments, the fastq preprocessing software fastp is used to quickly split input read pairs into smaller sets [49]. The number of paired reads per splits generated by fastp is controlled via the optional "–F" argument. Post splitting of input reads, SLUR(M)-py scripts conduct genomic alignments with the bwa mem algorithm including filtering, deduplication, and concatenation of generated files (Figure 2(B)); this processing is done in parallel per split. The basic SLUR(M)-py procedure ends with the counting of reads within these files and the removal of temporary files. The final output of this process is a text-based .bedpe file, representing the filtered, paired-end alignments.

In the event of SLURM crashing, node failure, or an error, SLUR(M)-py contains a checkpoint system within the workflow. This feature was included to save data generated at intermittent steps within the pipeline and allow for restarts at specific steps within the protocol. For example, when attempting to process Hi-C sequencing data, after splitting input read pairs with fastp – which is arguably one of the most time-consuming steps within SLUR(M)-py – users may need to restart alignments with bwa mem. This can be done without again splitting the input read pairs by passing "–R bwa," which would direct the slurm.py script to pick up at the step of alignment. In the event of an error, the SLUR(M)-py scripts also include a "checkwork.py" function to identify sources of errors saved within the "logs" directory (Figure 2(C)). For completely restarting runs, SLUR(M)-py can be passed to a "–restart" flag, should users wish to completely erase, reset, and rerun an attempt. After completing the processing with SLUR(M)-py, a "clean" functionality is also included to remove large intermittent and temporary files generated during processing and then compress the final, large text-based files (in .bedpe format) and other outputs with the gzip command.

2.2. Hi-C processing

SLUR(M)-py was initially designed to process sequenced, paired-end reads from Hi-C experiments. Given the underlying chemistry and preparation of Hi-C sequencing, the fragments from these experiments require additional, unique processing (compared to traditional, linear, paired-end reads from whole-genome sequencing) [31]. Briefly, alignments from bwa mem are reformatted at the output from the usual sequence alignment maps (.sam format) to a bed, paired-end file (.bedpe); these files are easily passed for further processing to Pandas and Dask Dataframes (Figure 2(B)). Across splits on input read pairs, alignments are filtered to select valid and informative Hi-C contacts, removing erroneous read pairs representing dangling ends from restriction digestion, pairs that dovetail or map to the same restriction fragment, or present as a self-circle/self-ligation event [31]. After filtering, alignments are segregated by chromosomes (taken from the input reference file or a list of chromosomes given by the -G parameter) into separate files. Per chromosomes, paired-end reads are analyzed for duplicate alignments using Pandas and Dask Dataframes (Figure 2(B)). SLUR(M)-py marks and removes duplicate read-pairs that have identical 5′ genomic coordinates and identical sequence signatures, imputed from cigar strings, as previously defined [53]. During this step, entries within each bedpe file representing Hi-C contacts are sorted (from left to right) by genomic position. Post duplicate marking, removal, and sorting, Hi-C contacts are concatenated into a set of final bedpe files within the "aligned" subdirectory, representing valid, unused, and duplicated Hi-C contacts (Figure 2(C)). This penultimate form of Hi-C data can be converted by SLUR(M)-py into an .mcool file or other file formats for compatibility with Juicer (.short format). Passing a jar file from Juicer tools will que the slurm.py command to auto-generate a .hic file upon completion of Hi-C processing. Our processing of Hi-C data finished in less than 13 hours across all samples reprocessed in this study with average run times of 6.69 and 4.23 hours from samples in Vero and human cell lines, respectively. These run times are faster than our previous run times with current, published pipelines (discussed later in the manuscript).

2.3. ATAC-seq and other processing

SLUR(M)-py is unique among analysis tools in its ability to process other forms of paired-end sequencing data (Table 1). For this purpose, we have encoded the –atac-seq and –wgs flags. Both these flags will configure a run of SLUR(M)-py to process, linear paired-end sequencing data, thereby altering alignment options in *bwa mem*, bypassing checks for Hi-C errors. Furthermore, with the –atac-seq option, output files are converted to be compatible with MACS3 (the most recent version of MACS2 [47]) which is used to identify "peaks" or genomic loci with significant pileup and overlap of mapped sequenced reads. Alternatively, the path to an .bedpe file from (SLUR(M)-py) or a .bam file can be passed to slurm.py at the command line (used as an input control) for processing and peak calling of ChIP-seq data rather than ATAC-seq data (Supplementary Figure S1). We used this capability to process the raw paired-end data representing three ChIP-seq samples, from 22Rv1, HAP-1, and HL-60 cells, downloaded from the ENCODE project (Supplementary Table S3).

To compare the processing of ATAC-seq data with SLUR(M)-py, we processed a single, test sample using the Nextflow enabled, NF-core-ATAC-seq pipeline [54]. While the data handling, software, and architectures of these pipelines are very different compared to SLUR(M)-py, making direct comparisons of their speeds difficult, both pipelines use MACS2 (or MACS3) for peak calling. On an example ATAC-seq sample from A549 cells, the NF-core-ATAC-seq and SLUR(M)-py pipelines are called nearly 68,582 and 65,030 peaks, respectively. Between the two pipelines, these calls produced similar signal tracks and locations of peaks, having an estimated 91% overlap (Supplementary Figure S2). Furthermore, downstream analysis (with deepTools [29]) demonstrated similar distributions of fragments within called peaks between the two pipelines (Supplementary Figure

S3A). Similar analysis was conducted with an ATAC-seq sample from the Vero cell line (Supplementary Figure S3B) and a ChIP-seq sample from HAP-1 cells (Supplementary Figure S3C) taken from the ENCODE project (Supplementary Table S3).

2.4. Automation of diagnostic plots for ATAC-seq and Hi-C

Upon successful completion of the run, a diagnostics directory is also generated which contains a timestamp (listing the start, end, and total run time) and a visualization (and .csv file) of the read alignment summary. Additionally, for ATAC-seq (and ChIP-seq experiments), read mapping summaries, fragment size distributions (Figure 3) and FRiP scores [18] (Supplementary Table S1) are also automatically calculated and reported to users within the same diagnostics directory. Similarly, for Hi-C analysis, the portion of valid Hi-C contacts (both inter- and intra-chromosomal contacts) and other mapped/processed read pairs are summarized automatically by SLUR(M)-py (Supplementary Figure S4). These diagnostic plots are helpful in determining the success and quality of an experiment.

2.5. Runtime analysis

To test the run time efficiency, flexibility, and fidelity of SLUR(M)-py, we processed both Hi-C and ATAC-seq data from two of our previous studies [41,42]. Experiments from Venu et al. in Vero cells include paired ATAC-seq and Hi-C samples taken at 12, 18, and 24 hours post infection with the vaccinia virus. In Roth et al. [41], ATAC-seq assays from A549 cells for several biological replicates ($n = 8$) were collected under nominal laboratory conditions (no perturbation or viral infection). Additional Hi-C data in human cell lines were gathered from the public ENCODE project and repository [43] in the form of raw fastq files and similarly reprocessed using the SLUR(M)-py pipeline. In total, these previously published datasets provided us with 26 ATAC-seq (Supplementary Table S1) and 22 Hi-C (Supplementary Table S2) samples for processing.

For ATAC-seq samples processed with SLUR(M)-py, the average run times for samples from the A549 simulation study ($n = 8$) and vaccinia virus infection experiments in Vero cells ($n = 18$) were 28 and 40 minutes, respectively, with three datasets taking a little over an hour to completely process the raw paired fastq.gz files into .bedpe files (Supplementary Table S1). Analysis of Hi-C data ($n = 22$) was also relatively quick, finishing in less than 16 hours across all the data analyzed here (Supplementary Table S2). The run times of both Hi-C and ATAC-seq were linear with respect to the number of paired-end reads (Figure 1(B), top and bottom, respectively). Indeed, modeling of SLUR(M)-py run time (as a function of counts of input read pairs) explains approximately 80.98% and 68.73% of the variation in run time for Hi-C sequencing and ATAC-seq experiments, respectively (p-value $< 0.1 \times 10^{-6}$, Linear Regression).

2.6. Comparing Hi-C processing

The Juicer pipeline and our SLUR(M)-py Hi-C script both use SLURM to manage jobs and possess Hi-C data; therefore, we wanted to compare our strategy for Hi-C processing against the current standard. After running both the Juicer and SLUR(M)-py Hi-C pipelines on the Vero Hi-C data, we generated .hic files using the processed results from these pipelines and the *pre* command from the suite of Juicer tools [23]. Visually, the Hi-C maps from Vero experiments look similar (Figure 4(A), Supplementary Figure S5). Comparing the counts of valid Hi-C contacts in Vero Hi-C maps, the Hi-C maps produced by the SLUR(M)-py pipeline tend to have more contacts retained compared to the number of valid contacts in Juicer produced Hi-C maps (Supplementary Figure S6A). On average, the SLUR(M)-py produced Hi-C

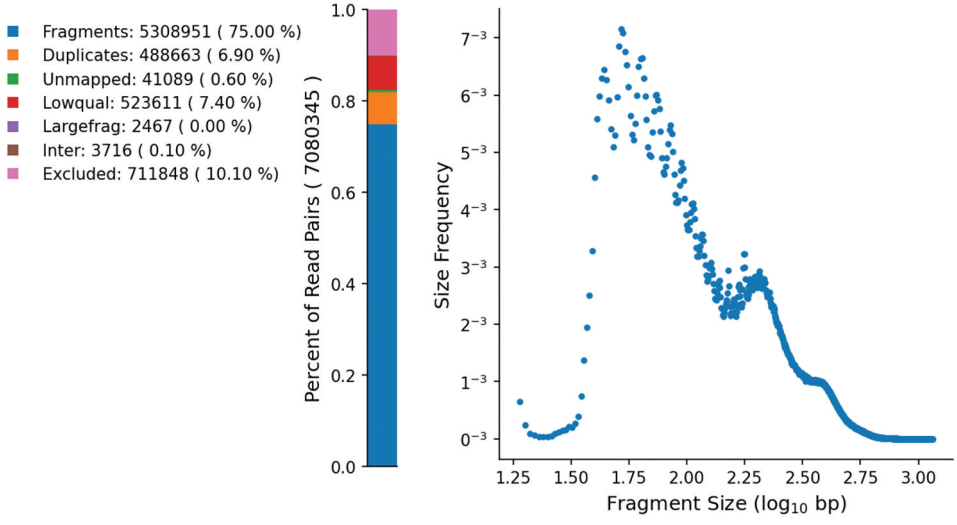

Figure 3. SLUR(M)-py generated diagnostic plot. Left) alignment counts plot (generated by slurm.Py) displaying mapping patterns of reads for a Vero ATAC-seq sample. Colors depict different categories of processed reads. Right) fragment size distribution plot autogenerated by SLUR(M)-py pipeline for ATAC-seq analysis. The x-axis denotes the size of fragment formed by aligned, paired-end reads (log10) while the y-axis marks the frequency of the fragment size out of the total aligned paired-end reads. Peaks in the curve at approximately 1.75 and 2.25 (along the x-axis) represent numerous fragments with an approximate size 56 and 178 bp respectively.

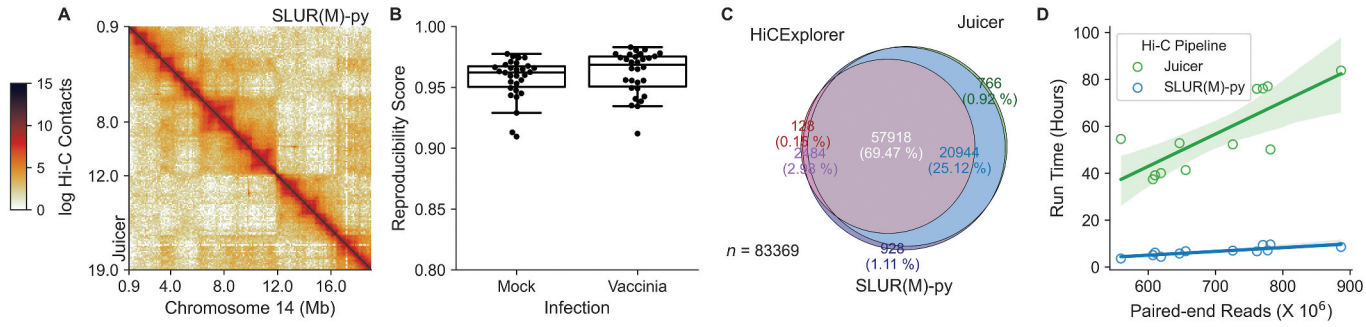

Figure 4. Comparison of Hi-C processing. A) Hi-C maps along Vero chromosome 14 from Juicer (lower, left diagonal) and SLUR(M)-py (upper, right diagonal) pipelines. Light to dark orange colors depict increasing Hi-C contact counts (log scale). B) Spector reproducibility score between Hi-C maps produced using Juicer versus SLUR(M)-py. The median reproducibility score (y-axis) is shown per chromosome, across Hi-C replicates from Venu et al. [40] separated by infection (x-axis). Scores above 0.9 are considered highly reproducible [55]. C) Overlap of valid Hi-C contacts across pipelines. A random sample of 100,000 read pairs from an A549 Hi-C experiment were processed through HiCExplorer, Juicer, and SLUR(M)-py. A total of 83,369 read pairs were designated as valid contacts by at least one pipeline while 57,918 contacts were designated as valid by all three pipelines. D). Distributions of run times (y-axis, hours) as a function of total paired-end reads when processing Vero Hi-C data from Venu et al. [40] with Juicer (green) pipeline versus SLUR(M)-py (blue). Green and blue lines and shaded regions denote regression models and relating run-time (hours) to total input read pairs.

maps had 25 million more read-pairs retained as valid contacts; this is on average 3.57% of the total read pairs across the Vero Hi-C samples (Supplementary Figure S6B). For additional quantitative comparisons, the Hi-C Spector reproducibility score [55] was used to quantify the similarity between SLUR(M)-py and Juicer processed Hi-C maps. Overall, the Spector reproducibility scores calculated between Hi-C data processed by SLUR(M)py and Juicer are biologically reproducible, with a genomic median score ranging from $_{0.944}$ to $_{0.976}$ across both infected and control samples ($n = 12$). Across these Hi-C samples, most individual chromosomes displayed reproducibility scores higher than $_{0.80}$ with the median reproducibility score across samples, per chromosomes all above $_{0.90}$ (Figure 4(B)) signifying high reproducibility between SLUR(M)-py and Juicer pipelines [55].

For two samples (both infected with the vaccinia virus), a single chromosome, chromosome 19 (NC_023660.1), displayed scores below 0.80 (Supplementary Figure S7). Examination of read pairs processed by SLUR(M)py and Juicer revealed that Juicer retains reads that are chimeric singletons (mapping to more than one chromosome in a single read) whose mate goes unmapped, while the scripting in SLUR(M)-py drops a chimeric mapping read(s), with an unmapped mate. These observations and differences in processing, along with viral infection, lead to the low reproducibility scores for these chromosomes in these samples. We discuss the detection of ligation events within Hi-C assays between the contig representing the vaccinia virus and the Vero genome later within this manuscript. Overall, visual inspection and quantitative measurements of Hi-C maps confirm the high reproducibility between results generated by our SLUR(M)-py pipeline and the Juicer pipeline.

Next, we compared the results generated by downstream tools often used in Hi-C analysis. These included JuiceBox, HiCCUPs, and the Arrowhead applications from the Juicer suite of tools [23,24]. The Hi-C maps produced using SLUR(M)-py are compatible with each of these applications (Supplementary Figure S8). Overall, there is a large agreement in the locations of loops and TADs called between the Juicer and SLUR(M)-py Hi-C maps (Supplementary Figure S9A). In total, there were more loops and TADs called using SLUR(M)-py Hi-C maps; this was on average (approximately) 17 and 9 (loops and TADs, respectively) across chromosomes and samples (Supplementary Figure S9B). Breaking that down further by genomic region, the difference in the number of called loops and TADs could range from −4 to 4 every 100 or 500 kb (for loops and TADs respectively) but attenuated toward a median of zero (Supplementary Figure S9C). Furthermore, the aggregates of average (by chromosome) Hi-C contact counts within called loops were also similar between SLUR(M)-py and Juicer produced Hi-C maps (Supplementary Figure S10).

For additional, direct comparison of SLUR(M)-py Hi-C processing to other pipelines, we generated a novel Hi-C dataset in the human, A549 cell line using the Arima Hi-C sequencing protocol. Over 700 million paired-end reads were sequenced and processed using our SLUR(M)-py pipeline (Supplementary Table S2). From these data, 100,000 paired-end reads were randomly subsampled and reprocessed using HiCExplorer, Juicer, and SLUR(M)-py. This was done to quickly process reads, mapping them to the human T2T reference genome, and to directly compare their processing (by read name) into Hi-C contacts across all three pipelines. Where possible, similar parameters, like mapping quality scores (≥30), were used in calls for all three pipelines. Across Juicer, HiCExplorer, and SLUR(M)-py, 83,369 out of 100,000 read pairs were retained as valid Hi-C contacts by at least one of the three pipelines (Figure 4(C)). The percent overlap of read pairs retained as valid Hi-C contacts by all three pipelines (post alignment, processing, and filtering) was 69.47% (Figure 4(C)). After examining the code-base (s), we postulate that the observed difference in Hi-C contacts is caused by one of the following: 1) differences in mapping approach (specifically options used in the bwa mem aligner, as shown in Table 1), 2) randomness introduced during mapping (specifically the lack of a random seed generator passed to bwa), 3) retention/processing of chimeric singletons (as previously discussed), 4) duplicate marking and handling, and 5) handling of read-pairs mapping to identical DNA restriction fragments (defined by restriction

sites and enzymatic digestion). The combined effect of these factors likely leads to large differences in results when the same data is processed by different pipelines and could extend to other Hi-C data processors. This analysis reveals that HiCExplorer had the least number of valid Hi-C contacts (post processing) and the most conservative definition of a "valid" Hi-C contact. By default, SLUR(M)-py retains more read pairs in analysis (like Juicer, those intra-chromosomal contacts that are nearly linear in mapping) compared to HiCExplorer. Thus, we have included an additional preset that may be passed to SLUR(M)-py such that during filtering of Hi-C contacts stricter filtering on alignments and contacts is provided (to better match results from HiCExplorer).

Finally, after compared mappings of Hi-C data (finding broad reproducible with the Juicer pipeline), we tested the run times of SLUR(M)-py compared to Juicer. Specifically, we used the early exit before .hic file creation in Juicer and the Hi-C data from the Vero experiments from Venu et al. Across the samples, all the runs with the Juicer pipeline took longer than 24 hours to converge to a pairs-txt file, while all the runs automated with SLUR(M)-py finished in less than 13 hours (Figure 4(D)). We attribute this increase in computational speed and reduction in run time when using SLUR(M)-py to a combination of fast software for read splitting (fastp [49]), Pandas and Dask Dataframes [45]), and a larger set of initial splits made on input read pairs, which decreased overall memory requirements and increased the number of parallel processes.

2.7. Duplicate marking

During initial library preparation, PCR amplification can lead to the propagation of duplicate-sequenced fragments. Alternatively, a single amplification cluster can be incorrectly identified as multiple clusters by the optical sensor, leading to the creation of optical duplicates. Most current practices suggest that duplicate fragments should be identified and removed to eliminate bias and the inclusion of artifacts in downstream analyses [18,22,23,41,56,57]. However, for some omics analyses, marking duplicates may be unwarranted. For example, during genetic variant calling from whole-genome sequencing, a recent study has suggested that marking and removing duplicates are memory intensive and unnecessary [58]. Similarly, another study examining duplicates within ATAC-seq data discovered that many duplicate fragments (identified by position) are not always the result of PCR amplification, but rather, true sequenced fragments created by the same, small open regions being sequenced multiple times [59]. For identifying duplicate alignments, SLUR(M)-py filters alignments using the 5′ genomic positions of fragments, strand orientation, and sequence signal/nucleotide variation; matching the definition of duplicate previously defined by other tools like Picard and Samblaster [53]. Given that users may wish to skip this step, duplicate marking and removal are optional within SLUR(M)-py, controlled by the addition of the –skip-dedup flag at the command line.

To test the effect of skipping duplicate markings and removal on processing times and results generated from ATAC-seq data, we ran several timing tests using the ATAC-seq data (in A549 cells) generated in Roth et al. [40]. Skipping duplicate marking and removal had no significant impact on overall run times (Supplementary Figure S11A). While we did see a slight increase in processing time – an insignificant 1.18 minutes on average (p-value > 0.05, Wilcoxon signed-rank test) – this was attributed to an increased data size processed by penultimate steps in the pipeline (such as MACS3) and not duplicate marking and removal. Similarly, while we saw small increases in the number of valid ATAC-seq fragment counts (Supplementary Figure S11B) and FRiP scores (Supplementary Figure S11C) post processing (when retaining duplicates), neither of these increases were deemed significant (p-value > 0.05, Wilcoxon signed-rank test).

While marking and removing duplicates have little effect on processing times and diagnostics of these experiments, the overall effect on detected, significant peaks (from MACS3) was unclear. In most samples, the number of peaks detected by MACS3 increased when duplicate read pairs were retained (Supplementary Figure S11D), adding an average (median) of nearly 6,371 peaks to the overall peak counts across experiments in the A549 cell line. Yet in two (of the eight) A549 samples, fewer peaks were identified when retaining duplicates during processing (Supplementary Figure S11D). We hypothesize that in these two, outlying samples, duplicate reads (when retained in the analysis) increased the level of background noise, lowering the overall number of peaks deemed significant by MACS3. A note of caution to users, for ATAC-seq experiments, excluding duplicate marking and removal may not affect the total processing time, but it is likely to alter the number of significant peaks identified via peak callers (like MACS3).

2.8. Example using SLUR(M)-py data products: inter-chromosomal contact analysis

Within the cellular nucleus, chromosomes segregate into territories [60]. Consequently, the most sequenced, paired-end reads from Hi-C experiments will map within a given chromosome and are labeled as intrachromosomal contacts [31]. However, a small portion of read pairs map between chromosomes, representing interactions between chromatin from different, neighboring chromosomes; these valid Hi-C contacts are referred to as inter-chromosomal contacts. Traditionally, the ratio of inter- to intra-chromosomal contacts is monitored by studies, but methods of scoring inter-chromosomal contacts have not been developed [61].

To broadly quantify the interaction between chromosomes, SLUR(M)-py provides an estimate of the inter-chromosomal contact scores [15,62]. The inter-chromosomal contact score, as described in Lieberman-Aiden et al. [15] and Duan et al. [62], is a broad quantification of the interaction frequency between two chromosomes. Specifically, this score is a ratio calculated by dividing the number of observed contacts between two chromosomes by the expected number of contacts between them, given the total number of inter-chromosomal contacts involving said chromosomes. For visualization of these scores, SLUR(M)-py generates a heatmap using the Seaborn plotting package in python (Figure 5A) [63]. These calculations are also included within the diagnostic plot when processing Hi-C data (Supplementary Figure S4).

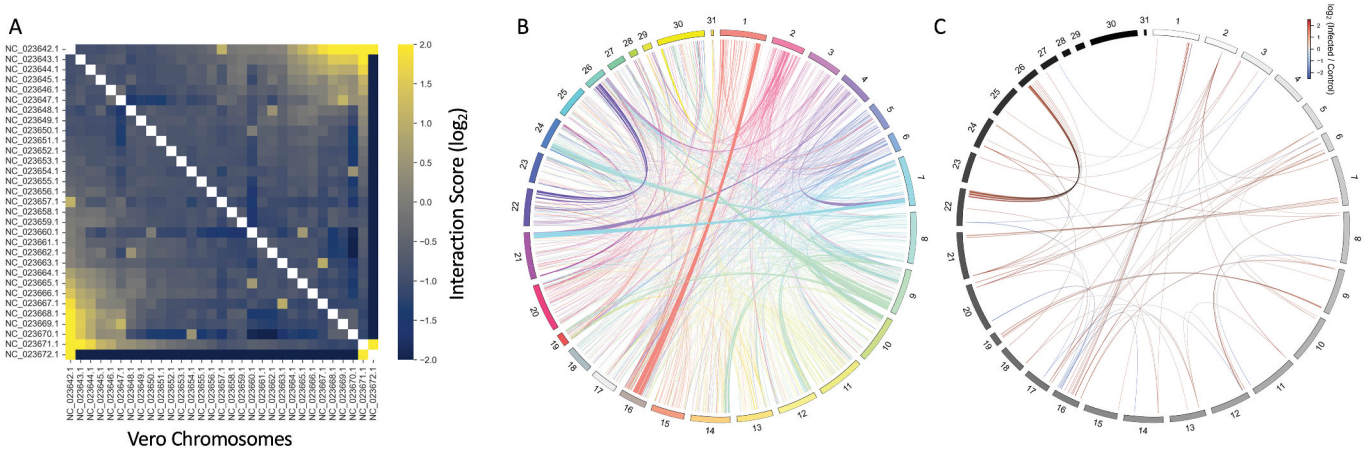

Figure 5. Inter-chromosomal analysis automated within the SLUR(M)-py pipeline. A) The interaction scores (log_2) between chromosomes (x- and y-axis,) are measured as the ratio of contacts between a pair of chromosomes and the expected number of inter-chromosomal contacts of those chromosomes. B) Significant inter-chromosomal interactions between Vero chromosomes. Colored arcs represent 500 kb regions with a significant number of Hi-C contacts between chromosomes as measured via a genome-wide binomial model. C) The fold-change of inter-chromosomal interactions with significant differences between control Hi-C map and the Hi-C map from Vero cells post 24 hours of exposure to the vaccinia virus.

When examining the inter-chromosomal contacts within Hi-C data from Venu et al., we detected an artifact of the ligation events between the vaccinia virus and the Green Monkey reference genome. The Green Monkey genome as a reference is much more complete and less complicated (with only 31 contigs/chromosomes) compared to the published Vero genome, which requires significant improvement [64]. Designed with host-pathogen infection experiments in mind, SLUR(M)-py can filter out alignments mapping to pathogen contigs, assuming the pathogen contig has been added as an additional contig within the reference genome. These artifacts were present in both the Hi-C and ATAC-seq data and the number of detected vaccinia virus fragments was proportional to the number of sequenced reads (p-value < 0.05, R^2 = 0.923, Supplementary Figure S12). Within the Hi-C experiments, these fragments make up a small portion (less than 1%) of the ligation events and only appeared in assays from infected cells (Supplementary Figure S13). Read pairs mapping to the vaccinia virus contig are recoverable from ATAC-seq experiments. However, no read pairs were identified that span the vaccinia virus and genome in this assay.

The apparent ligation between chromosomes and virus within the Hi-C experiment was further explored post processing with SLUR(M)-py. Specifically, we examined the anchoring points of paired reads, isolating pairs with one read mapping to the virus contig and the other belonging to the Green Monkey genome. In depicting the mapping of these coordinates, we discovered that the supposed ligation events between the virus and genome are uniformly distributed (Supplementary Figure S14). Furthermore, across chromosomes, the proportions of paired reads connecting the virus and genome are linear with respect to the size of the chromosome (p-value < 0.0001, R^2 = 0.92, Supplementary Figure S15). These observations, along with historical and experimental evidence that the vaccinia virus does not enter the nucleus, led us to conclude that these observed ligation events are false positive and inter-chromosomal contacts. From these events, we can predict a lower bound for the number of false ligation events in our experiments, making up approximately 0.345% of the observed inter-chromosomal ligations (Supplementary Figure S16A). Furthermore, these false-positive events were independent from the number of sequenced reads within Hi-C experiments (p-value > 0.05, linear regression, Supplementary Figure S16B).

Broadly, in the Vero experiments from Venu et al., the number of inter-chromosomal contacts was approximately 10% (of the total set of valid Hi-C contacts) within each Hi-C experiment (Supplementary Figure S13). This overall proportion of inter-chromosomal contacts did not significantly vary between controls and virus-infected samples. Previously, we had observed alterations in chromatin interaction frequencies between control and infected samples [42]. However, this analysis was only conducted at the scale of chromosomes. Here, we performed an additional, deeper analysis of the interactions captured between chromosomes using a combination of the output from our SLUR(M)-py, pipeline, and PyDESeq2 [65]. The contacts between chromosomes were quantified by generating contiguous bins, 500 kb in size, left to right, along each chromosome across samples, generating 841,610 unique, inter-chromosomal bins (Figure 5(B)). The number of contacts between pairs of chromosomes was counted per bin and used as inputs into PyDESeq2 to search for differentially altered inter-chromosomal contacts due to vaccinia virus infection.

Collapsing across time, we observed poor separation of infected and control samples when performing principal component analysis on the binned inter-chromosomal counts (Supplementary Figure S17A, left). Similarly, across the infection time-course only 66 of 841,610 bins (less than 1%) formed between interacting chromosomes were identified as significantly differential (adjusted p-value < 0.001, absolute foldchange > 1, Supplemental Figure S17A, right). Performing linear contrasts and examining differential inter-chromosomal contacts between chromosomes at each hour post infection (hpi), we saw no effect at 12 hpi (Supplementary Figure S17B), a minimal effect at 18 hpi (Supplementary Figure S17C), and the greatest separation of infected and control samples at 24 hpi (Supplementary Figure S17D). At this final time point (24 hpi), we identified only a small number of inter-chromosomal regions (13 and

138) with significant enrichment in fold-change, with respect to controls (negative fold-change) or infected (positive fold-change) samples (respectively) between chromosomes (p-value < 0.001, absolute fold-change > 1, Figure 5 (C), Supplementary Figure S17D). This result, the identification of only a small number of differential inter-chromosomal regions, suggests that changes in contacts between chromosomes are not a hallmark of vaccinia virus infection.

2.9. Memory usage

The amount of required random access memory (RAM) can range across calls of the SLUR(M)-py pipeline. We have experienced that the alignment of sequenced read pairs and creation of .hic files are often the most expensive processes in terms of RAM. For example, runs of *bwa mem* can require 5 to 20 GB of RAM (Supplementary Figure S18A) while creation of a .hic file can require anywhere from 10 to 40 GB, if not more, depending on the size of the input data and thread counts (Supplementary Figure S18B).

At the command line, SLUR(M)-py has two arguments to control RAM usage. These include the "–memory" and the "–xmx" inputs, which control the maximum memory across any processes submitted to SLURM, and the amount of memory allocated to Java, used in .hic file creation by juicer's *Pre* command (respectively). By default, the –memory parameter is not set.

Users of SLUR(M)-py may need to estimate the total amount of memory allocated and necessary for a run of SLUR(M)-py; this could be a challenge to most users. Should users want to set this parameter and cap the total amount of memory allocated, we have provided a script, named memoryprofile.py to aid in this endeavor. This module can be run on a completed SLUR(M)-py project, and using the job IDs from SLURM returns the amount of utilized RAM in Gigabytes across processes within the SLUR(M)-py pipeline (for example, see Supplementary Figure S18). This information can be used to set memory limits on future runs of the SLUR(M)-py pipeline.

2.10. Determining completeness and cleaning up

The success or failure of a run with SLUR(M)-py can be determined with the checkwork.py module. This script will report a successful completion of a run or identify logs (from within the log directory) containing warnings or errors and report those logs to the command line.

The number of parallel processes formatted by SLUR(M)-py can generate many temporary files post processing; their total size can be extensive and a burden on computer storage. For example, the processing of input paired fastq.gz files form an ATAC-seq experiment, with a total size of 6.7 GB, generated nearly 78 GB of data (in the form of final and temporary files) post processing with SLUR(M)-py. To help alleviate this burden on storage, we have provided a "clean up" functionality to the checkwork.py module, removing the temporary files produced by SLUR(M)-py. This script can be used after a successful run of SLUR(M)-py to remove large, unnecessary, and temporary files. Referring to the example above, this script reduced the size of the ATAC-seq project from 78 Gb to 17 Gb.

3. Discussion and conclusions

SLUR(M)-py is a SLURM powered, pythonic pipeline for performing parallel processing of paired-end sequencing profiles prepared from epigenomic and genomic experiments. SLUR(M)-py contains protocols and code written in Python for analyzing DNA sequences from many of the standard assays used within epigenomic studies. These pythonic scripts allow users to process paired-end sequenced reads from whole-genome, Hi-C, ATAC-seq, and ChIP-seq assays using just SLUR(M)-py. In this regard, SLUR(M)-py is a multi-omic pipeline, designed to run within an HPC, that eliminates the need for researchers to install and maintain multiple pipelines.

Underlying SLUR(M)-py is a unique Python code that interacts with the simple Linux utility for a resource management system, SLURM. With SLURM, one can parallelize, stack, and schedule dependencies necessary for end-to-end omics analysis. Unlike currently published pipelines, SLUR(M)-py is not dependent upon a gpu node for processing. SLUR(M)-py can schedule jobs to run on any node type integrated within a system managed by SLURM. By default, the runs of SLUR(M)-py will seek a tb node. All the results presented within this paper were generated using tb nodes. Our experience has shown that jobs submitted to gpu nodes tend to complete faster, which is expected given the difference between tb and gpu nodes. The flexibility of node choice allows users to run multiple submissions of SLUR(M)-py across several nodes, which is especially beneficial for high-performance computing environments with a heterogeneous node architecture.

We developed SLUR(M)-py using the sequencing data from Hi-C and ATAC-seq assays collected from mammalian cells during viral infection experiments [42]. Unlike current high-performance computing bioinformatics pipelines, the names of the contig representing the viral vector, mitochondrial DNA, or sex chromosomes (chrX and chrY, for example) can be passed to SLUR(M)-py as additional inputs. This feature allows users to monitor the level of contamination from these sources or to mask their alignments, which are removed early on in processing, saving computational time and cost. Conversely, a list of selected contigs from within a reference genome can be passed to SLUR(M)-py to limit analysis to those chromosomes of interest. Furthermore, these features provide flexibility to the SLUR(M)-py workflows which are not restricted to just human systems and cell lines with expected chromosome names such as chr1, chr2, and chrM, representing the first autosomes and the mitochondrial contig (just to name a few examples). Indeed, SLUR(M)-py is designed to be species agnostic. As an example of this, with SLUR(M)-py both Hi-C and ATAC-seq data were reprocessed from human cell lines (taken from the ENCODE consortium) and the African Green Monkey (*Chlorocebus sabaeus*), Vero cell line, produced by Venu et al. In both cases SLUR(M)-py auto-generated the metrics for quality control, produced the expected data packages for further analysis, and rendered figures for publication without error.

SLUR(M)-py demonstrates quick run-times relative to other pipelines. To our knowledge, Juicer and HiC-Pro are the only other Hi-C workflows that program and submit tasks to SLURM. For comparison, we reanalyzed our previously published Hi-C data [42], which was initially processed using the Juicer pipeline. The run times of SLUR(M)-py were faster than runs with the Juicer pipeline when reprocessing our data from viral infection experiments in the Vero cell line. Additionally, across the genome, the Hi-C maps generated with the outputs from the SLUR(M)-py pipeline match those Hi-C maps made using the Juicer pipeline with high reproducibility. However, SLUR(M)-py has the advantage over Juicer of being able to quickly and effectively analyze multiple types of epigenomics datasets, making bioinformatics analysis easier and more efficient for the end user.

Using SLUR(M)-py, we set out to explore the interactions between chromosomes within the Vero cell line. SLUR(M)-py provides scripts for calculating the interaction between chromosomes, and much like human cell lines [15], we see that smaller chromosomes interact more often with each other than with larger chromosomes in C. sabaeus. We expanded on this analysis along chromosomes (across 500 kb chromosome bins) to identify regions with differential inter-chromosomal contacts due to viral infection. Across the infection time course, this new analysis showed little difference suggesting that intra-chromosomal interactions are more impacted than inter-chromosomal interactions in response to vaccinia virus exposure. The largest difference (<1%) in the inter-chromosomal contact frequencies was seen at 24 hours post infection. Thus, unlike the dynamics in loop structures (due to infection) observed along chromosomes [42], the changes in contacts between chromosomes across time are not associated with viral infection. For other studies seeking to explore inter-chromosomal dynamics and their role in responding to viral infection, or other environmental perturbations, we have generated a set of scripts within SLUR(M)-py to aid in this analysis.

When analyzing contacts between chromosomes, we utilized SLUR(M)-py to explore artifacts that appear in those experiments from Venu et al. Specifically, we identified several ligation events between the contig representing the vaccinia virus and the chromosomes of the Green Monkey genome. The DNA of the vaccinia virus does not translocate into the nucleus, and therefore, its DNA does not directly interact with the host chromosomes [66–68]. It follows that crosslinking between the host and vaccinia virus genomes after formaldehyde exposure during Hi-C library preparation is impossible. Thus, we determined that these events (which are rare) are due to ligation errors produced after crosslinking. We estimated the rate of these false ligation events that occur within Hi-C experiments, is less than 1%. This finding should provide investigators with higher confidence that inter-chromosomal interactions are not dominated by random noise. For viral infection studies, SLUR(M)-py scripts have the option to remove these false contacts.

For the ATAC-seq data analyzed here – with an average of 21 million paired-end reads – the average run times were a little over half an hour (36.4 minutes). With these data and short runtimes, we investigated the effect on the average runtime when skipping duplicate marking and removal. The effects on runtimes were minimal, and skipping duplicate marking (and removal) is estimated to save users, on average, a little over 5 minutes. However, skipping duplicate filtering led to an increased count of significant loci detected within each ATAC-seq sample (an additional 6,000 peaks). It has been demonstrated that the nature of ATAC-seq leads to an overestimate of duplicate read pairs [59]. Briefly, small accessible regions create a bias in the final set of sequenced regions. While these regions are biologically reproducible, marking duplicates simply by removing fragments with duplicate positions (the traditional method) affects the total end counts of detected significant loci in ATAC-seq data [59]. Currently, the SLUR(M)-py pipeline is unable to retain duplicate read-pairs based on position and is only able to remove them. Future applications will try to estimate those read pairs that are marked as duplicates yet are biologically valid and worthy of retention. Thus, by retaining or removing duplicates within a run of SLUR(M)-py, we leave up to the users, controlled by the inclusion of the "–skip-dedup" flag at the command line.

In closing, SLUR(M)-py is designed to be a flexible pipeline for quickly processing data from epigenomic experiments. Additional tools and features are currently under development. For example, future versions of SLUR(M)-py will be able to process long read sequences from Oxford Nanopore or PacBio sequencing [69,70]. Protocols for processing sequencing data like RNA-seq, Cut&TAG sequencing [71–73], and genetic variant detection are also under development. While not explicitly tested, SLUR(M)-py should be able to process data from single-cell experiments too, assuming that the input data are in the form of paired-end sequences. SLUR(M)-py is publicly available and hosted on GitHub and able to be copied, forked, or modified by the scientific community. It is our hope that SLUR(M)-py contributes toward epigenomic and 3D genomic studies of chromatin conformation and architecture.

4. Materials, methods, and data

4.1. Data included, preparation, and processing

The raw data from ATAC-seq and Hi-C sequencing experiments generated by Venu et al. and used in this study are available in the SRA repository, under the Bio Project accession number PRJNA1037174. For details on the viral infection experiments in Vero cells infected with the vaccinia virus, see Venu et al. Using the SLUR(M)-py, pipeline the paired-end reads from ATAC-seq and Hi-C experiments were aligned to the Green Monkey (*Chlorocebus sabaeus*) reference genome from NCBI, named GCF_000409795.2_Chlorocebus_sabeus_1.1_genomic.fasta [42]. The sequence of the vaccinia virus (accession number U94848) was downloaded from GenBank and added to this reference genome as an additional contig. This reference file (in fasta file format) was then indexed using bwa [50]. For the Vero datasets (both ATAC-seq and Hi-C), the call to the slurm.py script was modified (using the -G parameter) to include only the 29 fully assembled autosomes (Chromosomes 1 through 29) and the two sex chromosomes (X and Y) with NCBI contig names NC_023642.1 – NC_023672.1. The calls to slurpy.py

were also modified to include the removal of reads mapping to the Green Monkey mitochondrial contig (-M NC_008066.1). For the analysis of ATAC-seq samples from the Vero cell line, the flag –atac-seq was added to calls of slurm.py. Apart from the inclusion of the -G and -M parameters, all other default settings in SLUR(M)-py scripts were utilized for analysis of the Vero samples.

Details on the creation, handling, and sequencing of ATAC-seq experiments using the A549 cell line presented here in timing and duplicate marking analysis are included within Roth et al. [40]. Briefly, raw, paired-end sequenced reads from eight biological replicates were downloaded from the sequence read archive from the Bio Project PRJNA975595 with accession numbers SRR24717527–SRR24717534. Using SLUR(M)-py with the –atac-seq flag and default settings, read pairs per sample were aligned to the telomere-to-telomere human reference genome, version 2 [74].

As part of this study, a Hi-C library was also prepared (using the Arima Hi-C library kit) from a sample of A549 cells. The methods for cellular culturing, handling, library preparation, and sequencing are exactly similar to those described previously in Venu et al. [40], however using A549 cells rather than Vero cells. Sequenced pair-end reads were also mapped to the telomere-to-telomere human reference genome, version 2 [74]. Raw fastq files from this sample are listed under the Bio Project PRJNA1250552, with SRA accession number SAMN47943572. For comparison of Hi-C processing software, 100,000 random paired-end reads were selected from this sample for processing.

4.2. Other data used

Nine sets of paired-end sequences representing Hi-C data collected on human cell lines were taken from the ENCODE project and data repository [22,43]. These included experiments conducted within the A549, HepG2, IMR90, and K562 cell lines. Please refer to Supplementary Table S2 for the list of ENCODE accession numbers for each sample. Additionally, three sets of paired-end sequencing data from ChIP-seq experiments in the 22Rv1, HAP-1, and HL-60 cell lines were gathered from the ENCODE repository (see Supplementary Table S3 for accession numbers). The Hi-C and ChIP-seq data were each aligned to the telomere-to-telomere human reference genome, version 2 [74] using SLUR(M)-py with default settings (for Hi-C data) or by passing the"-c" control flag to include input controls from ChIP-seq experiments. Input controls per experiment were processed via SLUR(M)-py by passing the –wgs flag.

4.3. Hi-C analysis and comparison

For comparative analysis, the Hi-C samples from experiments within the Vero cell line were aligned to the *C. sabaeus* reference genome modified with the addition of the vaccinia contig (as described above) using the Juicer pipeline [23]. For processing of each replicate with Juicer, the early exit parameter (–E) was also added to end the processing of Hi-C contacts after the creation of the merged, sorted no duplicate text file but prior to the creation of an .hic file. Post processing, .hic files were generated with the pre-command from Juicer tools (version = 1.22.01), using both the text-based outputs from Juicer and SLUR(M)-py workflows as inputs. Across both pipelines, a mapping quality score of 30 was used to filter alignments. To compare the .hic files made using the Juicer pipeline to those made using SLUR(M)-py, the Spector reproducibility score [55] was calculated for the main autosomes and sex chromosomes at 500 kb resolution across the sampled Hi-C maps ($n = 12$). Runs of SLUR(M)-py auto-generate timestamps and calculate the total run time(s). For runs with the Juicer pipeline, we estimated the total run time from start and end timestamps of logs within the Juicer debug output directory. These run times were brought into python for comparison, analysis, and plotting.

4.4. Testing of ATAC-seq analysis

For the analysis of processing times when analyzing ATAC-seq and Hi-C data using SLUR(M)-py, the autogenerated timestamps across runs were transformed into total minutes and hours then regressed against the number of input, paired-end reads using a linear model. For testing the effect of duplicate marking and removal on the overall runtime and processing of ATAC-seq samples, the slurm.py script within SLUR(M)-py was re-run twice for each sample, with and without marking duplicates, post alignment with bwa mem by passing the –skip-dedup flag to the slurm.py. Autogenerated time stamps from these runs were brought into Python for calculating the change in processing time. The auto-generated diagnostic files, which contain information on the number of peaks and summits detected with MACS3, for each run were similarly analyzed.

4.5. Inter-chromosomal analysis

Using the SLUR(M)-py script, the number of Hi-C contacts between chromosomes was counted and used to calculate the inter-chromosomal interaction score (IS) following methods from and Lieberman-Aiden et al. and Duan et al. [62]. Briefly, this is calculated for a pair of chromosomes i and j using the following formula: $IS(i,j) = N_{ij}/((N_i/N)(N_j/N)(N))$. Where N_i and N_j are the number of inter-chromosomal contacts involving chromosomes i and j (respectively), N_{ij} is the number of inter-chromosomal contacts between chromosomes i and j, and N is the total number of inter-chromosomal contacts [15,62]. These scores were visualized using Seaborn's heatmap function in Python on a $\log_2$ scale [63].

For visualization of inter-chromosomal contacts, contacts between chromosomes were counted every 500 kb. These inter-chromosomal bins were used in a binomial model from Kaufmann et al. (2015) to identify significant interactions every 500 kb between chromosomes [75]. The Circos, circle plot library [76] in Python, was used to plot significant 500 kb bins (binomial test, p-value > 0.001) between chromosomes.

For differential and inter-chromosomal analysis of vaccinia infected Vero samples, the inter-chromosomal contacts every 500 kb were counted across replicates and samples and

compared between vaccinia infected and control Hi-C maps at 12, 18, and 24 hours post infection. Specifically, the counts of binned inter-chromosomal contacts were used as input into PyDESeq2 [65]. Linear contrasts were performed at each time-point post infection. An adjusted, *p*-value threshold of *p*-value < 0.001 and absolute fold-change greater than one were used to establish significant differential contacts between chromosomes.

Acknowledgments

We would like to thank members of the Genomics and Bioanalytics Group and the Biosystems Administrators at Los Alamos National Laboratory for their helpful comments on this manuscript. We would also like to thank members of the Genomics Sequencing Core at Los Alamos National Laboratory for sequencing of genomic data presented in this manuscript. We would also like to acknowledge and thank the ENCODE Consortium and the members of Dr. Job Dekker's lab (University of Massachusetts), Dr. Erez Aiden's lab (Baylor University), Dr. Tim Reddy's lab (Duke University), and Dr. Michael Snyder's lab (Stanford University) for generating the Hi-C and ChIP-seq datasets used within this study.

Author contributions

Cullen Roth develops and maintains SLUR(M)-py and its associated software. Vrinda Venu tested, proposed user features, provided feedback, and assisted in the development of SLUR(M)-py software. Vrinda Venu conducted cell culture and library preparation for Hi-C sequencing. Cullen Roth and Vrinda Venu interpreted and summarized Hi-C results. Sasha Bacot added code, performed debugging, ran testing, and made edits to the SLUR(M)-py pipeline and documentation. Cullen Roth performed the analysis and generated figures. Christina R. Steadman and Shawn R. Starkenburg provided funding and materials for the research presented within this paper. Cullen Roth and Christina R. Steadman wrote the manuscript. All authors provided edits and comments on the manuscript.

Disclosure statement

No potential conflict of interest was reported by the author(s).

No writing assistance was utilized in the production of this manuscript.

Reviewer disclosures

Peer reviewers on this manuscript have no relevant financial or other relationships to disclose.

Funding

This material is based upon work supported by a Laboratory Directed Research and Development grant [20210082DR] from Los Alamos National Laboratory (Triad National Security, LLC), awarded to Shawn R. Starkenburg. This work is also funded by the U.S. Department of Energy, Office of Science, through the Office of Biological and Environmental Research (BER) and the Office of Advanced Scientific Computing Research (ASCR) as part of the BRaVE (Biopreparedness Research Virtual Environment) program under contract number 89233218CNA000001 to Los Alamos National Laboratory (Triad National Security, LLC), awarded to Shawn R. Starkenburg and Christina R. Steadman. The funding organizations had no role in study design, data collection and analysis, decision to publish, or preparation of this manuscript.

Data availability statement

The code base for SLUR(M)-py is currently under development and hosted on GitHub at:

https://github.com/SLUR-m-Py/SLURPY

References

Papers of special note have been highlighted as either of interest (•) or of considerable interest (••) to readers.

1. Ferrari R, Pellegrini M, Horwitz GA, et al. Epigenetic reprogramming by adenovirus E1A. Science. 2008;321(5892):1086–1088. doi: 10.1126/science.1155546
2. Kleinjan DA, Lettice LA. Long-range gene control and genetic disease. Adv Genet. 2008;61:339–388. doi: 10.1016/S0065-2660(07)00013-2
3. Turner BM. Epigenetic responses to environmental change and their evolutionary implications. Phil Trans R Soc B: Biolsci. 2009;364(1534):3403–3418. doi: 10.1098/rstb.2009.0125
4. Rao SS, Huntley MH, Durand NC, et al. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. Cell. 2014;159(7):1665–1680. doi: 10.1016/j.cell.2014.11.021
5. Teferi WM, Desaulniers MA, Noyce RS, et al. The vaccinia virus K7 protein promotes histone methylation associated with heterochromatin formation. PLOS ONE. 2017;12(3):e0173056. doi: 10.1371/journal.pone.0173056
6. Menachery VD, Schäfer A, Burnum-Johnson KE, et al. MERS-CoV and H5N1 influenza virus antagonize antigen presentation by altering the epigenetic landscape. Proc Natl Acad Sci USA. 2018;115(5):E1012–E1021. doi: 10.1073/pnas.1706928115
7. Zheng H, Xie W. The role of 3D genome organization in development and cell differentiation. Nat Rev Mol Cellbiol. 2019;20(9):535–550. doi: 10.1038/s41580-019-0132-4
8. Tsai K, Cullen BR. Epigenetic and epitranscriptomic regulation of viral replication. Nat Rev Microbiol. 2020;18(10):559–570. doi: 10.1038/s41579-020-0382-3
9. Oudelaar AM, Higgs DR. The relationship between genome structure and function. Nat Rev Genet. 2021;22(3):154–168. doi: 10.1038/s41576-020-00303-x
10. Balnis J, Madrid A, Hogan KJ, et al. Persistent blood DNA methylation changes one year after SARS-CoV-2 infection. Clin Epigenet. 2022;14(1):94. doi: 10.1186/s13148-022-01313-8
11. Loda A, Collombet S, Heard E. Gene regulation in time and space during X-chromosome inactivation. Nat Rev Mol Cellbiol. 2022;23(4):231–249. doi: 10.1038/s41580-021-00438-7
12. Kee J, Thudium S, Renner DM, et al. SARS-CoV-2 disrupts host epigenetic regulation via histone mimicry. Nature. 2022;610(7931):381–388. doi: 10.1038/s41586-022-05282-z
13. Wang R, Lee JH, Kim J, et al. SARS-CoV-2 restructures host chromatin architecture. Nat Microbiol. 2023;8(4):679–694. doi: 10.1038/s41564-023-01344-8
14. Dekker J, Alber F, Aufmkolk S, et al. Spatial and temporal organization of the genome: current state and future aims of the 4D nucleome project. Mol Cell. 2023;83(15):2624–2640. doi: 10.1016/j.molcel.2023.06.018
15. Lieberman-Aiden E, Van Berkum NL, Williams L, et al. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. Science. 2009;326(5950):289–293. doi: 10.1126/science.1181369
16. Buenrostro JD, Wu B, Chang HY, et al. Atac-seq: a method for assaying chromatin accessibility genome-wide. CP Mol Biol. 2015;109(1):21–29. doi: 10.1002/0471142727.mb2129s109
17. Paulsen J, Sekelja M, Oldenburg AR, et al. Chrom3D: three-dimensional genome modeling from Hi-C and nuclear lamin-genome contacts. Genome Biol. 2017;18(1):1–15. doi: 10.1186/s13059-016-1146-2
18. Yan F, Powell DR, Curtis DJ, et al. From reads to insight: a hitchhiker's guide to ATAC-seq data analysis. Genome Biol. 2020;21(1):1–16. doi: 10.1186/s13059-020-1929-3
19. Reiff SB, Schroeder AJ, Kırlı K, et al. The 4D Nucleome data portal as a resource for searching and visualizing curated nucleomics data. Nat Commun. 2022;13(1):2365. doi: 10.1038/s41467-022-29697-4

20. Barski A, Cuddapah S, Cui K, et al. High-resolution profiling of histone methylations in the human genome. Cell. 2007;129 (4):823–837. doi: 10.1016/j.cell.2007.05.009
21. Barski A, Zhao K. Genomic location analysis by ChIP-seq. J Cellular Bio Chem. 2009;107(1):11–18. doi: 10.1002/jcb.22077
22. Landt SG, Marinov GK, Kundaje A, et al. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. Genome Res. 2012;22(9):1813–1831. doi: 10.1101/gr.136184.111
23. Durand NC, Shamim MS, Machol I, et al. Juicer provides a one-click system for analyzing loop-resolution Hi-C experiments. Cell Syst. 2016;3(1):95–98. doi: 10.1016/j.cels.2016.07.002
24. Robinson JT, Turner D, Durand NC, et al. Juicebox.js provides a cloud-based visualization system for Hi-C data. Cell Syst. 2018;6 (2):256–258.e1. doi: 10.1016/j.cels.2018.01.001
25. Wolff J, Rabbani L, Gilsbach R, et al. Galaxy HiCExplorer 3: a web server for reproducible Hi-C, capture Hi-C and single-cell Hi-C data analysis, quality control and visualization. Nucleic Acids Res. 2020;48(W1):W177–W184. doi: 10.1093/nar/gkaa220
26. Kruse K, Hug CB, Vaquerizas JM. Fan-C: a feature-rich framework for the analysis and visualisation of chromosome conformation capture data. Genome Biol. 2020;21(1):1–19. doi: 10.1186/s13059-020-02215-9
    - **A python based toolkit for Hi-C data visualization and figure generation; generates beautiful plots and contains several analysis tools.**
27. Wolff J, Backofen R, Grüning B. Loop detection using Hi-C data with HiCExplorer. Giga Sci. 2022;11:giac061. doi: 10.1093/gigascience/giac061
28. Lu RJH, Liu YT, Huang CW, et al. Atacgraph: profiling genome-wide chromatin accessibility from ATAC-seq. Front Genet. 2021;11:618478. doi: 10.3389/fgene.2020.618478
29. Ramírez F, Dündar F, Diehl S, et al. Deeptools: a flexible platform for exploring deep-sequencing data. Nucleic Acids Res. 2014;42(W1): W187–W191. doi: 10.1093/nar/gku365
30. Lopez-Delisle L, Rabbani L, Wolff J, et al. Pygenometracks: reproducible plots for multivariate genomic datasets. Bioinformatics. 2021;37(3):422–423. doi: 10.1093/bioinformatics/btaa692
    - **An in-depth guide to processing, analyzing, and understanding Hi-C data.**
31. Lajoie BR, Dekker J, Kaplan N. The hitchhiker's guide to Hi-C analysis: practical guidelines. Methods. 2015;72:65–75. doi: 10.1016/j.ymeth.2014.10.031
32. Di Tommaso P, Chatzou M, Floden EW, et al. Nextflow enables reproducible computational workflows. Nat Biotechnol. 2017;35 (4):316–319. doi: 10.1038/nbt.3820
33. Ramírez F, Ryan DP, Grüning B, et al. Deeptools2: a next generation web server for deep-sequencing data analysis. Nucleic Acids Res. 2016;44(W1):W160. doi: 10.1093/nar/gkw257
34. Li PE, Lo CC, Anderson JJ, et al. Enabling the democratization of the genomics revolution with a fully integrated web-based bioinformatics platform. Nucleic Acids Res. 2017;45(1):67–80. doi: 10.1093/nar/gkw1027
35. Kerpedjiev P, Abdennur N, Lekschas F, et al. Higlass: web-based visual exploration and analysis of genome interaction maps. Genome Biol. 2018;19(1):1–12. doi: 10.1186/s13059-018-1486-1
36. Afgan E, Baker D, Batut B, et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. Nucleic Acids Res. 2018;46(W1):W537–W544. doi: 10.1093/nar/gky379
37. Yoo AB, Jette MA, Grondona M. Slurm: simple Linux utility for resource management. In: Feitelson D, Rudolph L, Schwiegelshohn U, editors. Workshop on job scheduling strategies for parallel processing. Springer; 2003. p. 44–60. doi: 10.1007/10968987_3
38. Jette MA, Wickberg T. Architecture of the Slurm Workload Manager. In: Klusáček D, Corbalán J, Rodrigo GP, editors. Workshop on job scheduling strategies for parallel processing. Springer; 2023. p. 3–23. doi: 10.1007/978-3-031-43943-8_1
39. Servant N, Varoquaux N, Lajoie BR, et al. HiC-Pro: an optimized and flexible pipeline for Hi-C data processing. Genome Biol. 2015;16 (1):1–11. doi: 10.1186/s13059-015-0831-x
40. Roth C, Venu V, Bacot S, et al. Slur (M)-py: a SLURM powered pythonic pipeline for parallel processing of 3D (Epi) genomic profiles. bioRxiv. 2024. doi: 10.1101/2024.05.18.594827
41. Roth C, Venu V, Job V, et al. Improved quality metrics for association and reproducibility in chromatin accessibility data using mutual information. BMC Bioinf. 2023;24(1):441. doi: 10.1186/s12859-023-05553-0
42. Venu V, Roth C, Adikari SH, et al. Multi-omics analysis reveals the dynamic interplay between Vero host chromatin structure and function during vaccinia virus infection. Commun Biol. 2024;7 (1):721. doi: 10.1038/s42003-024-06389-x
    - **A multi-omic, host-pathogen study that includes paired, sequenced samples of RNA-seq, ATAC-seq, and Hi-C sequencing data.**
43. Consortium EP, others. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012;489(7414):57. doi: 10.1038/nature11247
44. Chapman B, Chang JB. Biopython: python tools for computational biology. ACM Sigbio Newsl. 2000;20(2):15–19. doi: 10.1145/360262.360268
45. McKinney W. Data structures for statistical computing in Python. SciPy. 2010;445:51–56. doi: 10.25080/Majora-92bf1922-00a
46. Zhang Y, Liu T, Meyer CA, et al. Model-based analysis of ChIP-seq (MACS). Genome Biol. 2008;9(9):1–9. doi: 10.1186/gb-2008-9-9-r137
47. Feng J, Liu T, Qin B, et al. Identifying ChIP-seq enrichment using MACS. Nat Protoc. 2012;7(9):1728–1740. doi: 10.1038/nprot.2012.101
48. Gaspar JM. Improved peak-calling with MACS2. bioRxiv. 2018. doi: 10.1101/496521
49. Chen S, Zhou Y, Chen Y, et al. Fastp: an ultra-fast all-in-one FASTQ preprocessor. Bioinformatics. 2018;34(17):i884–i890. doi: 10.1093/bioinformatics/bty560
    - **An extremely fast fastq preprocessor for performing quality control, data filtering, and data splitting. Contains similar (and more) features compared to FASTQC while being lightweight and magnitudes faster.**
50. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv preprint. 2013. doi: 10.48550/arXiv.1303.3997
51. Rolon-Mérette D, Ross M, Rolon-Mérette T, et al. Introduction to anaconda and python: installation and setup. Quant Methods Phychol. 2016;16(5):S3–S11. doi: 10.20982/tqmp.16.5.S003
52. Deutsch P. Gzip file format specification version 4.3. Internet Engineering Task Force; 1996. https://www.rfc-editor.org/rfc/rfc1952
53. Faust GG, Hall IM. Samblaster: fast duplicate marking and structural variant read extraction. Bioinformatics. 2014;30(17):2503–2505. doi: 10.1093/bioinformatics/btu314
    - **A fast bioinformatics tool for deduplicating alignments from genomic sequencing data.**
54. Ewels PA, Peltzer A, Fillinger S, et al. The nf-core framework for community-curated bioinformatics pipelines. Nat Biotechnol. 2020;38 (3):276–278. doi: 10.1038/s41587-020-0439-x
55. Yan KK, Yardımcı GG, Yan C, et al. HiC-spector: a matrix library for spectral and reproducibility analysis of Hi-C contact maps. Bioinformatics. 2017;33(14):2199–2201. doi: 10.1093/bioinformatics/btx152
    - **A reproducibility statistic that is designed for the unique structures and distributions within Hi-C data. Performs better than the standard Pearson correlation statistic or coefficient of determination.**
56. McKenna A, Hanna M, Banks E, et al. The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 2010;20(9):1297–1303. doi: 10.1101/gr.107524.110
57. Danecek P, Bonfield JK, Liddle J, et al. Twelve years of SAMtools and BCFtools. Giga Sci. 2021;10(2):giab008. doi: 10.1093/gigascience/giab008
58. Ebbert MT, Wadsworth ME, Staley LA, et al. Evaluating the necessity of PCR duplicate removal from next-generation sequencing data

and a comparison of approaches. BMC Bioinf. 2016;17(S7):491–500. doi: 10.1186/s12859-016-1097-3
59. Zhu T, Liao K, Zhou R, et al. Atac-seq with unique molecular identifiers improves quantification and footprinting. Commun Biol. 2020;3(1):675. doi: 10.1038/s42003-020-01403-4
•• **ATAC-seq analysis demonstrating sequenced read duplicates in ATAC-seq are often biologically reproducible and due to resampling of small, open DNA regions and not necessarily because of PCR artifacts.**
60. Zouari YB, Molitor AM, Sexton T. Sailing the Hi-C's: benefits and remaining challenges in mapping chromatin interactions. Nucl Archit Dyn. 2018;2:457–473. doi: 10.1016/B978-0-12-803480-4.00019-3
61. Hansen P, Gargano M, Hecht J, et al. Computational processing and quality control of Hi-C, capture Hi-C and Capture-C data. Genes. 2019;10(7):548. doi: 10.3390/genes10070548
62. Duan Z, Andronescu M, Schutz K, et al. A three-dimensional model of the yeast genome. Nature. 2010;465(7296):363–367. doi: 10.1038/nature08973
63. Waskom ML. Seaborn: statistical data visualization. J Open Source Softw. 2021;6(60):3021. doi: 10.21105/joss.03021
64. Sène MA, Kiesslich S, Djambazian H, et al. Haplotype-resolved de novo assembly of the Vero cell line genome. NPJ Vaccines. 2021;6(1):106. doi: 10.1038/s41541-021-00358-9
65. Muzellec B, Teleńczuk M, Cabeli V, et al. PyDESeq2: a Python package for bulk RNA-seq differential expression analysis. Bioinformatics. 2023;39(9):btad547. doi: 10.1093/bioinformatics/btad547
66. Hruby DE, Guarino LA, Kates JR. Vaccinia virus replication i. Requirement for the host-cell nucleus. J Virol. 1979;29(2):705–715. doi: 10.1128/jvi.29.2.705-715.1979
67. Moyer RW. The role of the host cell nucleus in vaccinia virus morphogenesis. Virus Res. 1987;8(3):173–191. doi: 10.1016/0168-1702(87)90014-1
68. Greseth MD, Traktman P. The life cycle of the vaccinia virus genome. Annu Rev Virol. 2022;9(1):239–259. doi: 10.1146/annurev-virology-091919-104752
69. Schöpflin R, Melo US, Moeinzadeh H, et al. Integration of Hi-C with short and long-read genome sequencing reveals the structure of germline rearranged genomes. Nat Commun. 2022;13(1):6470. doi: 10.1038/s41467-022-34053-7
70. Zhong JY, Niu L, Lin ZB, et al. High-throughput pore-C reveals the single-allele topology and cell type-specificity of 3D genome folding. Nat Commun. 2023;14(1):1250. doi: 10.1038/s41467-023-36899-x
71. Skene PJ, Henikoff S. An efficient targeted nuclease strategy for high-resolution mapping of DNA binding sites. Elife. 2017;6:e21856. doi: 10.7554/eLife.21856
72. Zheng X-Y, Gehring M. Low-input chromatin profiling in Arabidopsis endosperm using CUT&RUN. Plant Reprod. 2019;32(1):63–75. doi: 10.1007/s00497-018-00358-1
73. Wu SJ, Furlan SN, Mihalas AB, et al. Single-cell CUT&Tag analysis of chromatin modifications in differentiation and tumor progression. Nat Biotechnol. 2021;39(7):819–824. doi: 10.1038/s41587-021-00865-z
74. Nurk S, Koren S, Rhie A, et al. The complete sequence of a human genome. Science. 2022;376(6588):44–53. doi: 10.1126/science.abj6987
75. Kaufmann S, Fuchs C, Gonik M, et al. Inter-chromosomal contact networks provide insights into mammalian chromatin organization. PLOS ONE. 2015;10(5):e0126125. doi: 10.1371/journal.pone.0126125
76. Krzywinski M, Schein J, Birol I, et al. Circos: an information aesthetic for comparative genomics. Genome Res. 2009;19(9):1639–1645. doi: 10.1101/gr.092759.109