Thesis

Chromatin Compartments and Selection on X

Søren Jørgensen

2024-12-19

Chromatin Compartments and Selection on X

How Edges of Active Chromatin Align with Selection Regions in Primates

Søren Jørgensen

MSc. Bioinformatics

2024-12-19

Submitted in fulfillment of the requirements of the degree of MSc. Bioinformatics

“This is a dummy abstract, dreamt up by chatGPT. This thesis investigates the 3D chromatin architecture of the X chromosome in baboons, macaques, and humans, focusing on chromatin compartments during spermatogenesis. Using publicly available Hi-C data, interaction maps were created to identify Principal Component 1 (PC1) compartments, revealing distinct compartmentalization patterns among species. The analysis included transition zones, where chromatin shifts between compartment types, and their correlation with positively selected regions. By comparing these zones with evolutionarily significant regions, the study explores how chromatin structure influences evolutionary pressures. Key findings include conserved chromatin features that may help retain non-advantageous alleles, suggesting a role for selfish genetic elements in genome evolution. This research offers new insights into the relationship between chromatin architecture and evolutionary dynamics across primate species.

- Søren Jørgensen ”

Table of contents

1 Introduction **1**
1.1 Sexual reproduction (spermatogenesis, meiosis) 1
1.2 Selfish genes (and randomness) 1
1.3 Our Organism of Interest, Wang et al., and the references 2
1.4 Extended Common Haplotypes Discoverved in Humans 2
1.5 Chromatin Conformation 2
1.6 High-Throughput Chromosome Conformation Capture (Hi-C) 2

2 Methods **3**
2.1 Initial Exploration with HiCExplorer 3
2.2 Downloading Data and Project Structure 4
2.3 Handling coolers (Or: preparing coolers) 5

3 Results **13**
3.1 Exploration with HicExplorer 13
3.2 Open2c ecosystem 18

4 Discussion **25**

Bibliography **27**

1 Introduction

1.1 Sexual reproduction (spermatogenesis, meiosis)

The production of gametes in a sexually reproducing organism is a highly complex process that involves numeruous elements. Spermatogenesis, the process of forming male gametes, involves four stages of differentiation from a germ cell through *spermatogonia*, *pachytene spermatocyte*, and *round spermatids* to *spermatozoa*, or *sperm* [Wang et al., 2019], and it is the very basis of male reproduction. The specialized cell division of meiosis neatly handles the pairing, recombination, and segregation of homologous chromosomes, thereby ensuring proper genetic distribution. Deeply understanding the steps of molecular steps of reproduction and how our genetic material is inherited is essential in biology, bringing insight to areas such as speciation, population diversity, and (male) infertility.

1.2 Selfish genes (and randomness)

The conventional story of meiosis in gametogenesis is one of random segregation of the sex chromosomes. They split into haploid gametes, where each chromosome has an equal chance of being passed on to a gamete. That seems like a fair game, but what if some genes are cheating the system by making others less viable. A meiotic driver is a selfish gene element that modulates meiosis and preferentially transmits its own allele through meiosis, regardless of the downstream fitness effects it may have (good or bad) on the organism it is part of. This phenomenon challenges the traditional understanding of selection, extending its scope beyond the fitness effects on an organism to include selective pressures at the molecular level. For example, if some genes on the X chromosome create a disadvantage for gametes that *do not* contain those genes, making sure the Y chromosome is not as viable as the X, resulting in a sex imbalance and possibly numerous other downstream effects. That is exactly what is coined *sex chromosome meiotic drive* [Jaenike, 2001], a result of selfish genetic elements.

Motivated by previous results in the Munch Research group [Munch, 2024] on hybrid incompatibility and extended common haplotypes [Skov et al., 2023, Sørensen et al., 2023] that could be explained by meiotic drive, we wanted to investigate how these patterns correlate with chromatin compartments.

1.3 Our Organism of Interest, Wang et al., and the references

1.4 Extended Common Haplotypes Discoverved in Humans

1.5 Chromatin Conformation

1.6 High-Throughput Chromosome Conformation Capture (Hi-C)

Our DNA can be divided into different orders of structure. *3C* focus on identifying the highest orders of organization inside the nucleus, that is, when the 30 nm thick coil of chromatin fibers folds into loops, Topologically Associating Domains (TADs), and chromatin compartments. Here, we narrow our focus on the largest of the structures, *compartments*, that is known to determine availability to transcription factors, thus making an *A* compartment *active*—and the *B* compartment *inactive*. The introduction of the Hi-C (high-throughput 3C) method [Lieberman-Aiden et al., 2009] opened new possibilities for exploring the three-dimensional organization of the genome.

1.6.1 Hi-C Library preparation

1.6.2 Hi-C Data Analysis

Aligning the Hi-C reads

Identifying and Storing Valid Hi-C Pairs

Interaction Matrices and Quality Control

Calling Compartments (ICE)

Edges

2 Methods

In this project, we formulate two objectives:

A: Reproduce the Hi-C interaction maps and eigendecomposition from [Wang et al., 2019], with some modifications. We briefly use *HiCExplorer*, but change the analyses to use the *Open2C Ecosystem* [Open Chromosome Collective, 2024] which have a Pyton API as well as command-line functions, which can be paired very well with Jupyter Notebooks. The majority of the data analysis was run with a *gwf* workflow, and the commands that were visually inspected were run in Jupyter Notebooks.

B Compare with regions of selection that are found in *human*. Investigate the biological meaning of the results.

All computations were performed on GenomeDK (GDK) [ref], an HPC cluster located on Aarhus Uninversity, and most of the processing of the data was nested into a *gwf* workflow [ref], a workflow manager developed at GDK. I would like to thank GDK and Aarhus University for providing computational resources and support that contributed to these research results.

The whole of this project is carried out with reproducibility in mind, so an effort (and quite a significant amount of time) has been put into documenting code and organizing the project for readbility and transparency through a Quarto project [ref]. Therefore, all code, virtual environments and text is made available as a Quarto book, rendered directly from the GitHub repository with GitHub Pages [ref]. To make this possible, the Quarto documentation has been extensively studied and discussed with *KMT* [ref, aknowledge].

2.1 Initial Exploration with HiCExplorer

Here moves most of the text about HiCExplorer…

For the initial exploration of methods with *HiCExplorer*, the 5 first samples in 'fibroblast' were chosen (Table 2.1).

Table 2.1: The samples chosen for initial data exploration with HiCExplorer. From NCBI SRA Portal.

	Run	Bases	Bytes	source_name
0	SRR6502335	73201141800	31966430779	fibroblast
1	SRR6502336	65119970100	24433383054	fibroblast
2	SRR6502337	52769196300	23015357755	fibroblast

Table 2.1: The samples chosen for initial data exploration with HiCExplorer. From NCBI SRA Portal.

	Run	Bases	Bytes	source_name
3	SRR6502338	52378949100	22999581685	fibroblast
4	SRR6502339	28885941600	10960123150	fibroblast

The goal was to replicate some of the figures from Wang et al. [2019] using *HiCExplorer*, especially to reconstruct interaction matrices and E1 graphs from macaque data.

The matrices are constructed with `hicBuildMatrix` from separately mapped read-pairs. Along with the matrix *.h5* file, a *.log* file it outputted, documenting the quality control for the sample. Multiple logs can be aggregated and visualized with `hicQC`.

Before correction (or balancing) of the interaction matrix, a pre-correction filter is applied, filtering out low-count bins and very high-count bins. A threshold for Mean Absolute Deviation (*MAD*) is estimated by `hicCorrect diagnostic_plot`, followed by iterative correction with `hicCorrect correct --correctionMethod ICE`.

The PCA was performed with `hicPCA` on the correcte matrices, yielding the first 3 PCs.

2.2 Downloading Data and Project Structure

To reproduce the results from [Wang et al., 2019], I chose to use their raw data directly from the SRA portal [ref]. I filtered the data to contain all their paired-end Hi-C reads, and included only macaque samples. The data set also contains RNAseq data, and the same tissues for both macaque and mouse. The meta data for the data set was extracted into a runtable `SRA-runtable.tsv`. To get an overview of the data accessions used in this analysis, we will first summarize the runtable that contains the accession numbers and some metadata for each sample (Table 2.2). It adds up to ~1Tb of compressed `fastq` files, holding ~9.5 billion reads, roughly evenly spread on the 5 tissue types.

Table 2.2: Summary of the data accessions used in this analysis

	source_name	GB	Bases	Reads
0	fibroblast	211.403275	553,968,406,500	1,846,561,355
1	pachytene spermatocyte	274.835160	715,656,614,700	2,385,522,049
2	round spermatid	243.128044	655,938,457,200	2,186,461,524
3	sperm	164.131640	428,913,635,400	1,429,712,118
4	spermatogonia	192.794420	518,665,980,300	1,728,886,601

2.3 Handling coolers (Or: preparing coolers)

Figure 2.1: A flowchart showing the pipeline from `.fastq` to `.mcool`. The first 6 steps were done with a Probably BioRender or Inkscape.

2.3.1 The *gwf* workflow targets

A *gwf* workflow was created to handle the first part of the data processing, and each accesion number (read pair, mate pair) from the Hi-C sequencing was processed in parallel, so their execution was independent from each other.

Downloading the reads The reads were downloaded from NCBI SRA portal [ref] directly to GDK using `sra-downloader` [ref] as `.fastq.gz` files.

Handling the reference The latest reference genome for rhesus macaque (*macaca mulata*), *rheMac10* (or *Mmul_10*, UCSC or NCBI naming conventions, respectively) was downloaded to GDK from UCSC web servers with `wget` [ref]. To use `bwa` (Burrow Wheeler's Aligner) [ref] for mapping, rheMac10 needs to be indexed with both `bwa index` with the `--bwtsw` option and `samtools faidx`, which results in six indexing files for `bwa mem` to use.

Since [2019], the reference genome for rhesus macaque has changed several times from *rheMac2* to *rheMac10*, each time resulting in a much less fragmented reference assembly. Part of the reasoning for reproducing their results was doing so on the latest assembly of the Macaca mulata genome, which arguably will result in a more accurate mapping of the reads, and a better inference of the chromatin compartments as well.

Several mappers were used in different configurations (described in below), and `bowtie2` requires its own indexing of the reference, using `bowtie2-build --large-index`, which creates six index files for `bowtie2` to use. `--large-index` creates the special indexing format required for large genomes such as macaque.

Mapping Hi-C reads paragraph will be restructured.

The main difference between Hi-C libraries and standard paired-end libraries is the high fraction of chimeric reads in Hi-C. As a contact pair is crosslinked and ligated before sequencing, chimeric reads occur as a feature, and standard mapping techniques seeks to filter out this type of reads [ref]. Thus, we need specialized tools for rescuing chimeric reads. That said, we have to be cautious distinguishing the intended chimerism for Hi-C and that of technical artefacts.

It was not feasible to follow the same approach as [Wang et al., 2019] with either *HiCExplorer* or *Open2C*, as they use a third software, *HiC-Pro*. Hic-Pro uses bowtie2 in end-to-end mode, followed by remapping of 5'-ends of the unmapped reads to rescue chimeric fragments along with another approach. I mapped the reads using `bowtie2 --end-to-end` without the rescue-remapping, and it returned a very high fraction of discarded reads. I argue that even when trying to reproduce results, it is nonsensical to use methods that are not state-of-the-art. The HiC-Pro pipeline stops at a normalized contact map, and is thus not sufficient for downstream analysis.

HiCExplorer Initially, recommendations from HiCExplorer were used. According to their documentation [ref] it is crucial to 1) align reads locally, and 2) map mates separately. They recommend either of `bwa` or `bowtie2`, so I tested both with their recommended settings. `bowtie2` turned out to be a lot more resource-intensive and to produce almost no mapped reads [ref sup-fig-bowtie2-stats], so I suspect some settings was not set correctly. The mapped reads was converted to a Hi-C Matrix (`.h5`) with *HiCExplorers* `hicBuildMatrix`, which is extremely memory-intensive, using ~120 GB memory for the biggest matrix. I followed *HiCExplorer* pipeline to plot and explore the matrices created from this mapping. However, the work was laborious for experimentation, as, even though written in Python, `HicExplorer` only comes with a command-line interface and provided functions all write plots to files. I did not manage to make an efficient implementation for plotting the `.h5` files produced by the pipeline, as would be required for utilizing Jupyter Notebooks for customizing plots. I relatively quickly shifted to *Open2C* for their promises of the greener grass (a Python API).

Open2C Suspiciously, [Open Chromosome Collective, 2024] never mentions any problems with aligning the Hi-C reads, they just provide an example using `bwa mem` in paired-end mode and with the `-P` option set, which activates the Smith-Waterman [ref] algorithm to rescue missing hits, by focusing on assigning only of of the mates to a good mapping and escape mate-rescue. The documentation of `bwa` ref state that both bwa-mem and bwa-sw will rescue chimeric reads. Consequently, Open2C does not have a builtin way of pairing the reads after mapping, and I was left with two options: **1)** to re(-)pair the individually mapped read-mates (.bam) with `samtools-fixmate` into one of the specific input formats required for `cooler` to create an interaction matrix *cooler*, or **2)** re-map the reads using Open2C's recommendations and use their

established pipeline for producing a cooler. I chose the latter, where I mapped the fastq files to *rheMac10* in paired end mode for a pair (m1, m2) with `bwa mem -SP rheMac10 m1 m2`.

Parse and sort the reads **HiCExplorer** No action is needed, as this step is done implicitly when building the matrix

Open2C We need to convert the alignments into ligation events, and distinguish between several types of ligation events. The simplest event is when each side only maps to one unique segment in the genome 'UU'. Other events, where one or both sides map to multiple segments or the reads are long enough (>150bp) to contain two alignments (multiple ligations) have to be considered as well. Multiple ligations (walks) are treated according to the `--walks-policy` when parsing the alignments into valid pairs (or valid Hi-C contacts). Here, `mask` is the most conservative and masks all complex walks, whereas `5unique` reports the 5'-most unique alignment on each side. The pairs are piped directly into `pairtools sort` after parsing, as the deduplication step requires a sorted set of pairs. The *.pairs*-format produced by `pairtools` is an extension the 4DN Consortium-specified format, storing Hi-C pairs as in Table 2.3.

Table 2.3: Column specification of the .pairs format as extended by `pairtools` [ref].

Index	Name	Description
1	read_id	the ID of the read as defined in fastq files
2	chrom1	the chromosome of the alignment on side 1
3	pos1	the 1-based genomic position of the outer-most (5') mapped bp on side 1
4	chrom2	the chromosome of the alignment on side 2
5	pos2	the 1-based genomic position of the outer-most (5') mapped bp on side 2
6	strand1	the strand of the alignment on side 1
7	strand2	the strand of the alignment on side 2
8	pair_type	the type of a Hi-C pair
9	mapq1	mapq of the first mate
10	mapq2	mapq of the second mate

I initially used `--walks-policy mask`, reasoning I had plenty of data points and could handle complex walks in a conservative way. Only later I realized the recommendations from *pairtools*, specifically informing that longer reads might have a significant proportion of reads that contain complex walks. With this in mind, I decided to re-parse the alignments into a new set of pairs, and equally apply the recommended filter (next section). As both results are saved, we can compare the two approaches.

Filter (deduplicate) pairs With *HiCExplorer*, no action is needed, as this step is done implicitly when building the matrix.

pairtools comes with a de-duplication function, `dedup`, to detect PCR duplication artefacts. At this point we will remove all reads that are mapped to an unplaced scaffold. Even though the publication of *rhemac10* assembly states they have closed 99% of the gaps since *rhemac8* [ref], *rheMac10* still contain more than 2,500 unplaced scaffolds, which are all uninformative when calculating the chromatin compartments as is the goal of this analysis. Therefore, we simply only include the list of conventional chromosomes (1..22, X, Y) when doing the deduplication. Initially, the default values were used to remove duplicates, where pairs with both sides mapped within 3 base pairs from each other are considered duplicates.

`cooler` recommend to store the most comprehensive and unfiltered list of pairs, and then applying a filter it on the fly by piping from `pairtools select`. Initially, I missed this step and I did not filter for mapping quality. After reparsing the alignments and applying the same analysis, we compare the two pipelines.

A quality control report is generated by `pairtools dedup`, and the reports are merged with `MultiQC` [ref] for each cell type.

Create interaction matrices (coolers) **HiCExplorer** `hicBuildMatrix` both parse and filter the mapped reads. The default value was used, where alignments with $mapq < 15$ are discarded.

Open2C The final part of the *gwf* workflow takes `.pairs` as input and outputs a `.cool` file (*cooler*). Initially, we read directly from the newly generated deduplicated pairs without additional filtering, but here, the official recommendation is to filter out everything below $mapq = 30$ by piping the pairs through `pairtools select "(mapq1>=30) and (mapq2>=30)"` to `cooler cload pairs`.

We should have plenty of data to do the filtering, but I argue it is not strictly necessary. I will show a histogram of the *mapq* scores to convince you [ref].

I have re-parsed the alignments and created new coolers, including only the Hi-C contacts where $mapq \leq 30$, following the current recommendations from `cooler`.

2.3.2 Notebook edits

As `cooler` and `cooltools` have a Python API, the more experimental parts of the analysis were moved to Jupyter Notebooks. `cooltools` comes with a helper library for operations on

genomic intervals called `bioframe`.

Pooling samples (Merging coolers) The samples are grouped into *replicates* with a unique **BioSample** ID, but we chose to pool all the interaction matrices for each cell type. We argue that when Wang et al. [2019] determine compartments to be highly reproducible between replicates, by merging the replicates we can get a more robust signal.

`cooler merge` was used to merge all samples in each sub-folder (cell type) to just one interaction matrix for each cell type. The function merges matrices of the same dimensions by simply adding the interaction frequencies of each genomic position together, resulting in less empty positions by chance.

Create multi-resolution coolers (zoomify) A feature of working inside the ecosystem of *Open2C* [ref] is that it natively provides support for storing sparse interaction matrices in multiple resolutions in the same file by adding groups to the cooler [ref]. We can then efficiently store resolutions (i.e., different bin sizes) that is multiples of the smallest bin size. We chose to use 10kb, 50kb, 100kb, and 500kb bins, and the resolutions are made by recursively binning the base resolution. We call this process zoomifying.

Matrix balancing (Iterative correction) Finally, we balance the matrices using the cooler CLI. We use `cooler balance` with the default options which iteratively balances the matrix (Iterative Correction). It is first described as a method for bias correction of Hi-C matrices in [Imakaev et al., 2012], where it is paired with eigenvector decomposition, coining the combined analysis **ICE**. Here, the eigenvector decomposition of the obtained maps is experimentally validated to provide insights into local chromatin states.

[According to `cooler` documentation] We have to balance the matrices on each resolution, and thus it cannot be done prior to zoomifying. They state that the balancing weights are resolution-specific and will no longer retain its biological meaning when binned with other weights. Therefore, we apply `cooler balance` to each resolution separately. `cooler balance` will create a new column in the `bins` group of each cooler , `weight`, which can then be included or not in the downstream analysis. This means we will have access to both the balanced and the unbalanced matrix.

The default mode uses genome-wide data to calculate the weights for each bin. It would maybe be more suitable to calculate the weights for *cis* contacts only, and that is possible through the `--cis-only` flag, and that can be added to another column, so that we can compare the

difference between the two methods easily. However, we will only use the default mode for now.

Eigendecomposition The eigendecomposition of a Hi-C interaction matrix is performed in multiple steps. As value of the eigenvector is only *significant* up to a sign, it is convention [ref] to use GC content as a phasing track to orient the vector. E1 is arbitrarily defined to be positively correlated with GC content, meaning a positive E1 value signifies an active chromatin state, which we denote a A-type compartment (or simply A-compartment). We performed eigendecomposition of two resolutions, 100 Kbp and 500 Kbp. Wang et al. [2019] briefly describes their method to calculate the eigenvectors as a sliding window approach on the observed/expected matrix in 100 kb resolution summing over 400 kb bins with 100 kb step size, a method I was not able to replicate in the *Open2C* ecosystem. I decided to mimic this by smoothing the 100 kb E1 values by summing to 500 kb bins in steps of 100 kb, yielding a comparable resolution which I denote '*pseudo*-500 kb' resolution (*ps500kb*).

First, we calculate the GC content of each bin of the reference genome, *rheMac10*, which is binned to the resolution of the Hi-C matrix we are handling. It is done with `bioframe.frac_gc` (*Open2C*). To calculate the E1 compartments, we use only within-chromosome contacts (*cis*), as we are not interested in the genome-wide contacts. `cooltools.eigs_cis` will decorrelate the contact-frequency by distance before performing the eigendecomposition. `eigs_cis` needs a *viewframe* (view) to calculate E1 values, the simplest view being the full chromosome. However, when there is more variance between chromosome arms than within arms, the sign of the first eigenvector will be determined largely by the chromosome arm it sits on, and not by the chromatin compartments. To mitigate this, we apply a chromosome-arm-partitioned view of the chromosome (as a bedlike format, described in `bioframe` docs [ref]).

Additionally, to mimic the *Local PCA* from [Wang et al., 2019], I also defined a view of 10 Mb bins. Thoughout the project, I will compare results from each of the three views and resolutions.

Plotting *HiCExplorer* plots matrices to .png from the command-line. When plots were generated (with `hicPlotMatrix`), it produces a .png output that has to be loaded back into the notebook. There is limited support for modifying the plot (from command-line options), such as to add spacing for a bigWig track with E1 values, add plot titles, and define the size and resolution of the plot. I briefly tried to implement a plotting function on the .h5 matrices and bigWig tracks, but it could not fetch regions from a matrix on the fly and had to load the full matrix into memory (that is, all full-length chromosomes). To better visualise differences in the interaction matrix, the interaction frequency f_i, is transformed to $\log 1\mathrm{p}(f_i) = \log(1 + f_i)$. It is

required to visualize the normalized interaction frequency, as there are mostly values very close to 0, but it also aids the visibility in the raw counts matrix.

We use matplotlib and seaborn to plot in the *Open2C* framework. Utilizing the `cooler` class, we can fetch regions of the matrix without modifying the file. As my analysis is centered around the X chromosome, it is efficiently handled by simply fetching 'chrX' from the matrix with `cooler.Cooler.matrix().fetch('chrX')`. Many methods of the cooler class returns data selectors, which do not retrieve data before it is queried [ref]. This means we can create many selectors at once without overflowing memory, enabling us to plot multiple interaction matrices side-by-side, e.g. the corrected and un-corrected matrices. This is easily done with the `balance` parameter of the matrix selector (`.matrix()`), which determines if it should apply the balancing weights to the coordinates and defaults to `True`.

The matrix is retrieved an plotted with `matplotlib.pyplot.matshow`, which automatically produces a heatmap image of the matrix. Here, in stead of transforming the interaction matrix, the color scale is log-transformed with `matplotlib.colors.LogNorm`. Additionally, `cooltools` comes with more tools to aid visualization: *adative coarsegrain* and *interpolation*, which can be chained. `adaptive_coarsegrain` iteratively coarsens an array to the nearest power of two and refines it back to the original resolution, replacing low-count pixels with NaN-aware averages to ensure no zeros in the output, unless there are very large regions that exceed the `max_levels` threshold, such as the peri-centromeric region.

3 Results

3.1 Exploration with HicExplorer

3.1.1 Quality Control

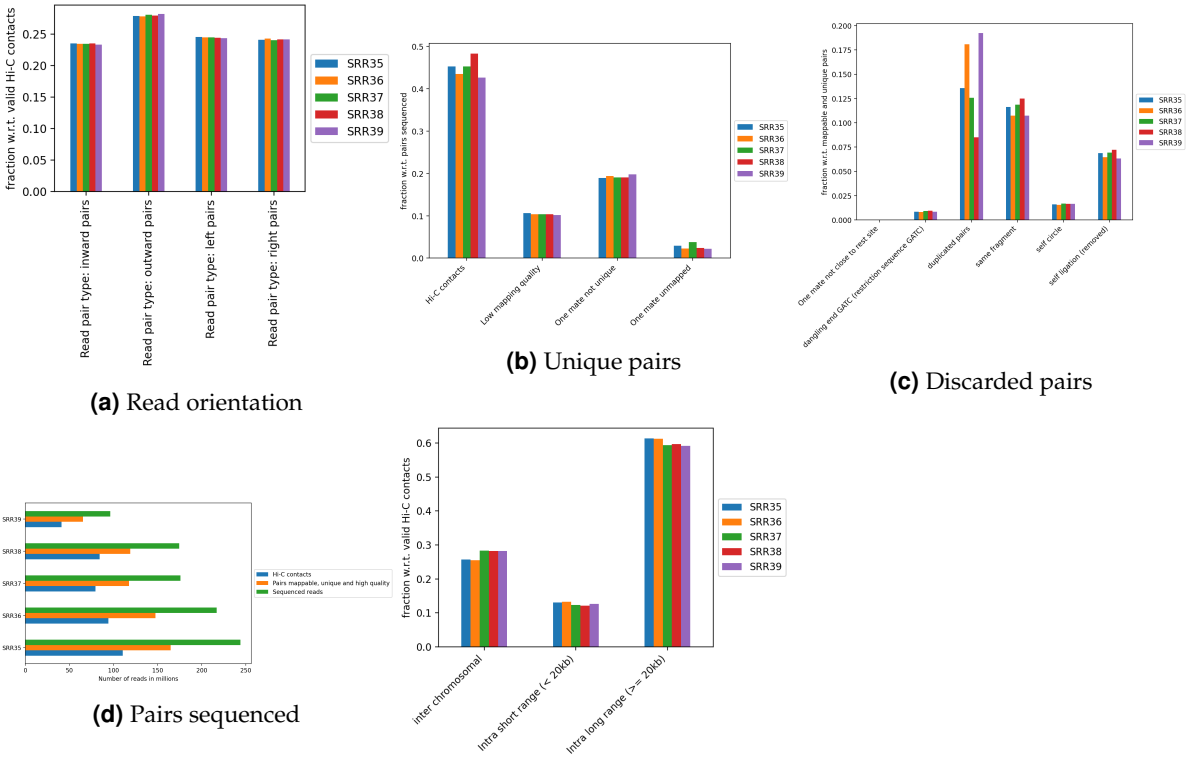
The separately mapped read-mates were parsed into a *.h5* interaction matrix by `hicBuildMatrix`, which include a *.log* file documenting the builtin quality control (hereafter, *QC*). Log files from the 5 samples were merged with `hicQC` (Figure 3.1). We observe showed equal fractions of the read-orientation of read-pairs (Figure 3.1a), which is expected for a good Hi-C library. Additionally, it determines between 40% to 50% of the total reads to be valid Hi-C contacts (Figure 3.1b), which is allegedly [ref] usually only 25%-40%. ~~Overall a solid set of Hi-C samples until now~~. Figure 3.1e shows, however, unusually high fractions of *inter*-chromosomal contacts (up to 30%) compared to *intra*-chromosomal contacts (also denoted *trans* and *cis* contacts, respectively). It is expected that *cis* contacts are orders of magnitude more frequent than *trans* contacts [Bicciato and Ferrari, 2022, p. 236; Lieberman-Aiden et al., 2009], and *HiCExplorer* states it should be below 10% [ref]. The high fraction may be mitigated by enforcing a stricter *mapq* threshold for a valid Hi-C pair, as we also observe higher-than expected valid contacts. However, we continue without the current matrices.

3.1.2 Correction

The correction diagnostic tool yielded a similar *mad* threshold within the range $[-3, -2]$. Even so, I followed the *HicExplorer* recommendation to set the lower threshold to at least -2 and the upper threshold to 5 in the pre-normalization filter. I argue that with a high number of valid contacts, it is safer to err on the side of caution and maybe filter out bad data.

> NB: when I say that a mapper performs poorly in finding Hi-C contacts, it is `hicBuildMatrix` that performs badly when reads are mapped with that mapper.

To compare these mappings with others, the QC results is an easy way. Therefore, the reads were mapped with *bowtie2* in both end-to-end- and local-mode followed by `hiCBuildMatrix`, and the QC from each method was plotted next to each other (Figure 3.3). Interestingly, *bowtie2* was much more computer-intensive in both modes, perhaps because of the `--very-sensitive` option. In any case, the QC reveals a major difference in the total number of reads that are



(a) Read orientation

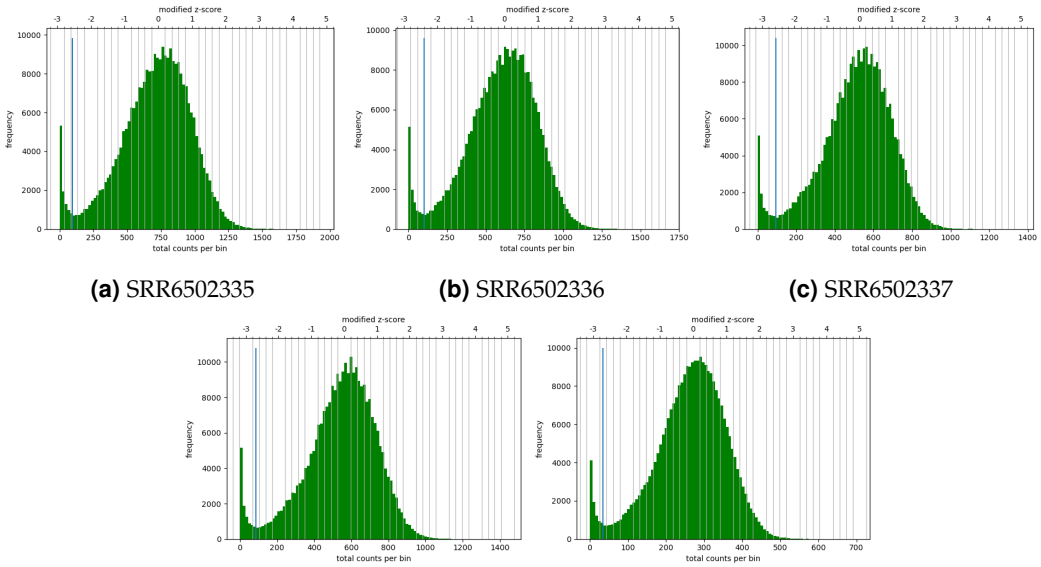
(b) Unique pairs

(c) Discarded pairs

(d) Pairs sequenced

(e) Contact distance

Figure 3.1: Quality control of the mapped Hi-C reads using *HiCExplorer* `hicQC`. The figures should be moved to Supplementary/Appendix because they are ugly and un-alignable. But that is the fault of HiCExplorer, not me. ~~Or I should spend a couple of hours to plot them manually~~.



(a) SRR6502335 **(b)** SRR6502336 **(c)** SRR6502337

(d) SRR6502338 **(e)** SRR6502339

Figure 3.2: Histograms of the number of counts per bin (bottom x-axis) and the modified z-score (top x-axis) from which the *mad* threshold is defined.

determined to be valid Hi-C contacts by `hicBuildMatrix`. As expected, *end-to-end-bowtie2* performs worse at locating Hi-C contacts than the other methods [ref row1], finding a very low amount of mappable, unique pairs passing the quality threshold. In contrast, *local-bowtie2* performs similarly to *bwa* in finding mappable, unique, high-quality pairs, but calls only approximately half the number of valid Hi-C contacts (>20%), resulting in a fraction of valid Hi-C pairs that hits the expectation from *HicExplorer* docs [ref row3]. With *bwa*, the reads were discarded either due to low mapping quality or non-unique mates, whereas with *local-bowtie2*, the reads were almost exclusively filtered out due to low mapping quality. This must be a result of how the mappers assign mapping quality, and I believe *local-bowtie2* looks suspiciously selective in finding unique but low quality alignments. *end-to-end-bowtie* almost exclusively filters out read-pairs where one mate is unmapped, which is expected when the majority of reads are unmapped.

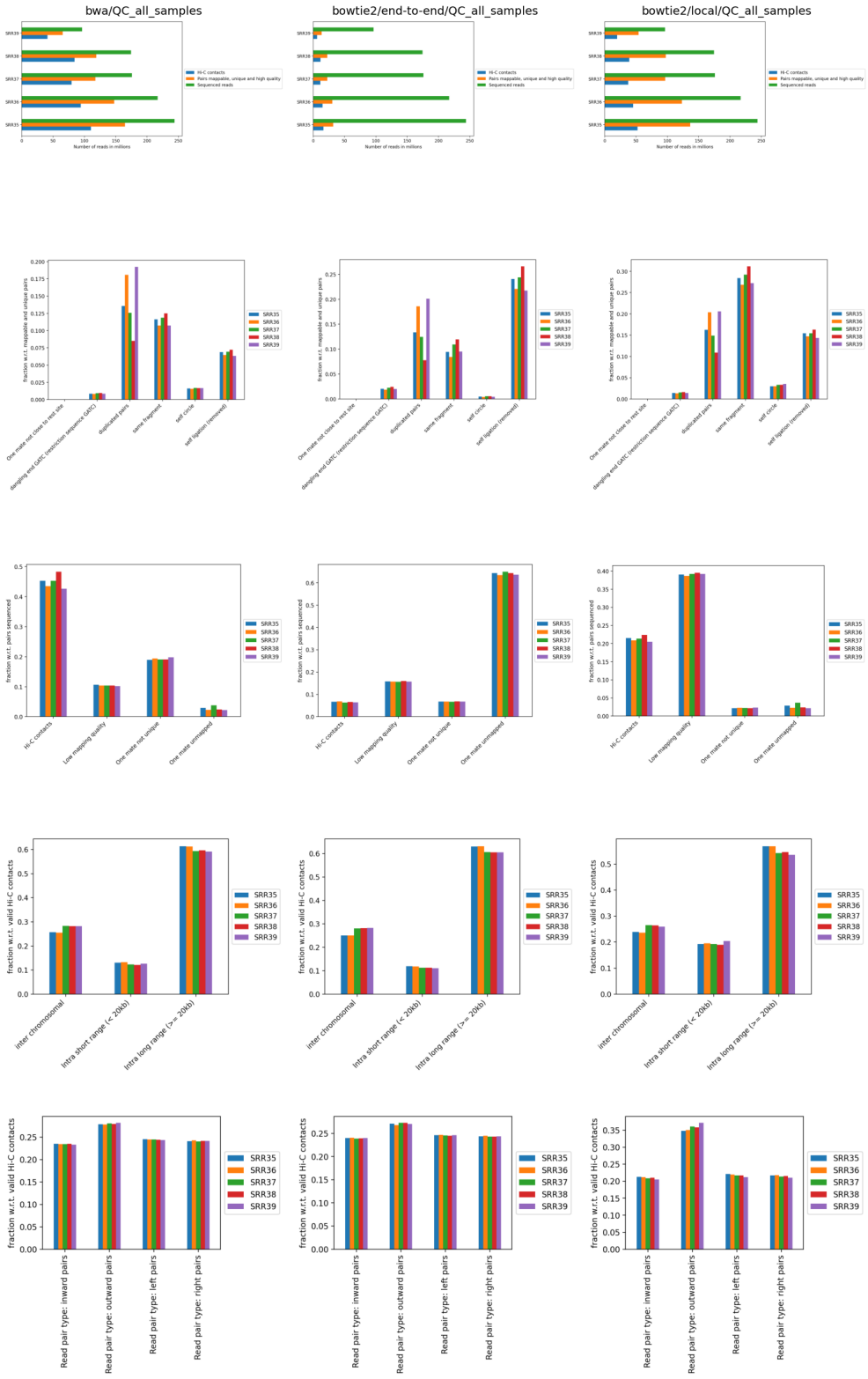
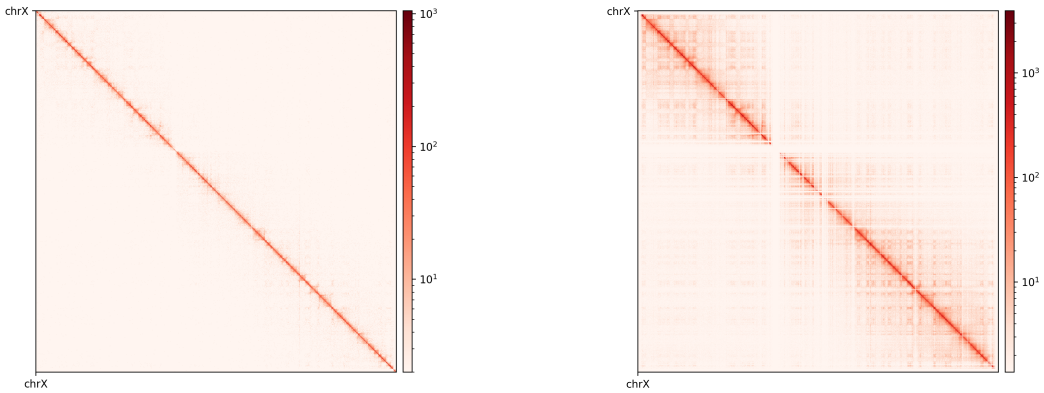


Figure 3.3: Comparison of HiCExplorer QC plots for all samples using different alignment tools.

As discussed, the five samples were pooled with `hicSumMatrices`, and the non-standard contigs (unplaced scaffolds) were filtered out, and the different resolutions were created (`hicMergeMatrixBins`). *HiCExplorer* also comes with a normalization function prior to correcting the matrix, which should be applied if different samples should have comparable bin counts. It has no effect when having only one matrix. Nevertheless, the pooled matrix was normalized and then corrected compared in Figure 3.4.



(a) Normalized matrix chrX

(b) Normalized and corrected chrX

Figure 3.4: A comparison of interaction matrices before/after iterative correction (*HiCExplorer*).

It is now obvious why we have to correct the matrix. The uncorrected (Figure 3.4a) has no signal apart from the diagonal. Even though some bins have been filtered out, the expected *plaid* pattern of a contact matrix is visible along the diagonal after the correction (Figure 3.4b), leaving evidence for chromatin structure, especially in the first 50 million bases of the chromosome. There is a wide region of empty values at the place of the centromere.

3.1.3 Eigenvectors

The PCA performed by `hicPCA` on the pooled samples at both 50kb and 100kb resolution yielded the first 3 principal components. For PC1 on both resolutions (Figure 3.5a, Figure 3.5d) we observe only a single sign change which occurs at around 60 Mbp, the region of the centromere. It means the PCA has captured more variance between the chromosome arms than within them, making it uninformative about chromatin compartments. Upon visual inspection, it is clear that neither of the PC graphs capture the pattern of the interaction matrix. Unimpressed, I rationalize that the option `--extra-track` to provide a gene track or histone

coverage should not affect this reult much. It should be provided as a phasing track to orient the eigenvector to positively correlate with gene density or histone marks, and could possibly muddle the compartments if not included. At this point, I stoppped using *HiCExplorer*, as I assessed that a more flexible tool was needed.

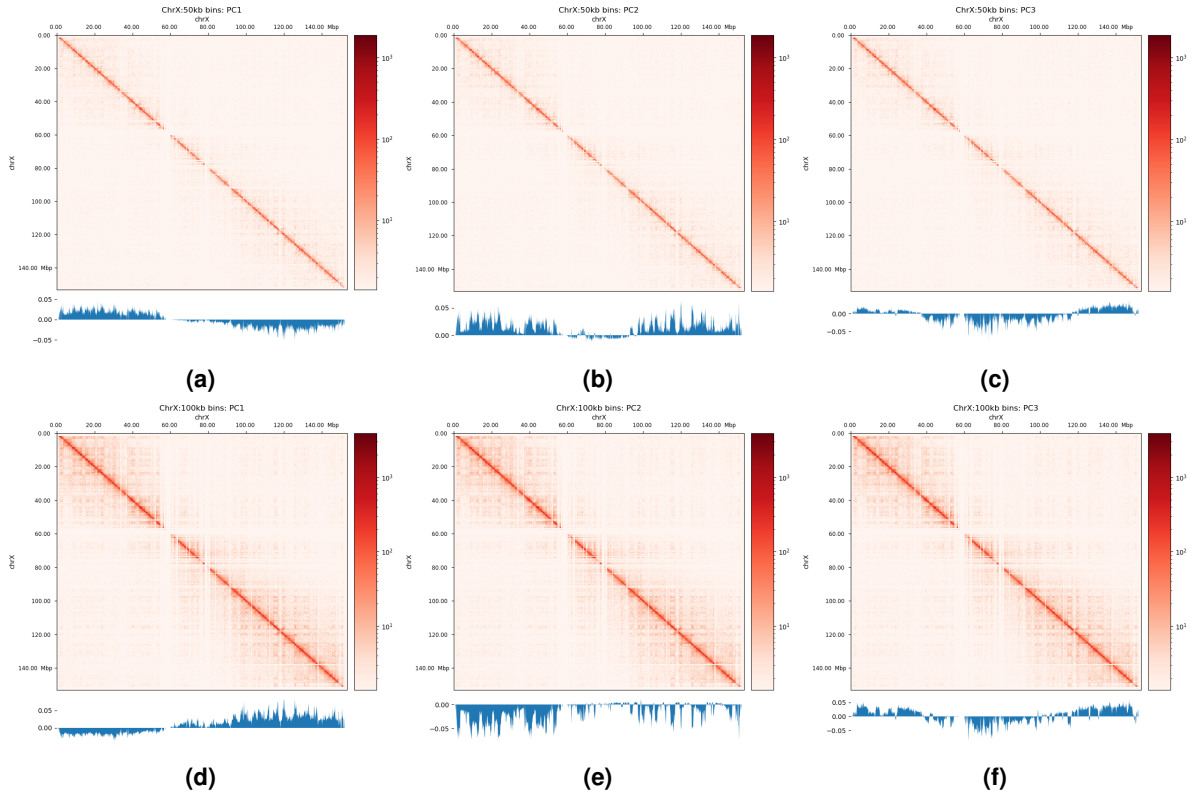


Figure 3.5: Corrected interaction matrix for chromosome X along with PC1, 2, or 3, respectively. a-c: 50kb resolution, d-f: 100kb resolution. *HiCExplorer*.

3.2 Open2c ecosystem

3.2.1 Quality Control

Initial run (`--walks-policy mask`) Comparing the multiQC report for each of the cell sources show similar distributions of *unmapped* (both sides unmapped), *one-sided* (one side mapped), *two-sided* (both sides mapped), and *duplicated* (w.r.t. total mapped) reads. The percentage of *cis* pairs w.r.t. mapped pairs is around 70% for all samples. [ref supptbl-qc-all-samples-mask]. The valid pairs also show similar distributions of pair types divided into 10 categories. The $P(s)$ curve looks similar as well, peaking between 270 bp and 320 bp separation

(ref suppfig-multiqc-ps-curve). The QC does not show any information about mapping quality of the reads. Note that the $P(s)$ ccurve arise from pre-filtered pairs, and should be compcared with the $P(s)$ of the cooler after filtering.

Recommended (`--walks-policy 5unique`) Parsing alignments with the recommended walks-policy aproximately halves the percentage of *unmapped* reads, and *one-* and *two-sided* reads as well *duplicated* reads are slightly increased. Overall number of unique pairs are increased with more than 20% increase. The percentage of *cis* pairs are only decreased by a percentage point at most [ref supptbl-qc-all-samples-5unique].

3.2.2 Correction

Matrix balancing did not show major improvement in the plaid pattern, as it was already pretty good. It does, however, filter out bins that are deemed too low-count to be informative, for example peri-centromeric regions. The matrix was expected to be smoother after balancing (for chromosomes), as regions along a chromosome should only vary slowly in contact frequency with other regions [ref].

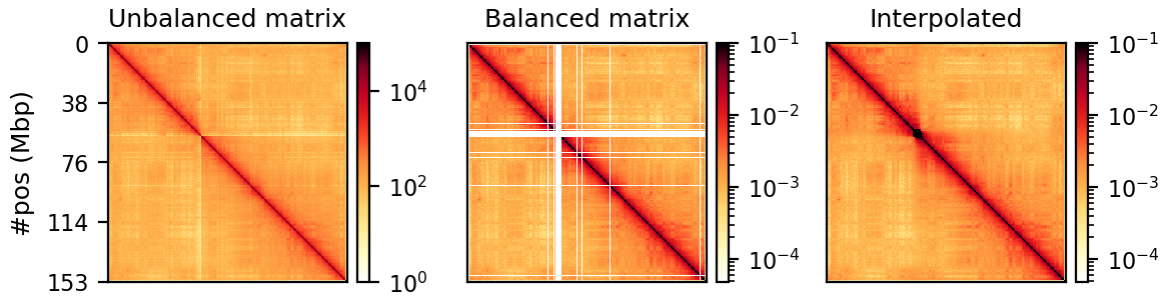


Figure 3.6: Raw, balanced, and interpolated chrX interaction matrix in 500kb resolution. The interpolation is done to make the matrix more visually appealing, but it is not necessary for the analysis.

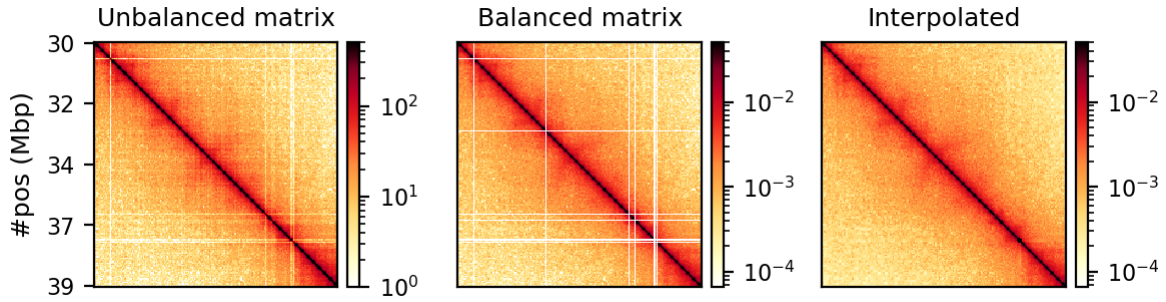


Figure 3.7: Raw, balanced, and interpolated chrX interaction matrix in 50kb resolution. The interpolation is done to make the matrix more visually appealing, but it is not necessary for the analysis.

The coarsegrained and interpolated matrix is useful to make a good-looking interaction matrix, but is not that useful for analysis purposes. It might get easier to visually inspect the matrix, but it is not clear how well the interpolated matrix reflects the structure of the chromatin. The regions that are coarsgrained are small zero- or low-count bins which are averaged, effectively reducing the resolution of those regions until the count is sufficient. They get more frequent the longer genomic distance (the further we travel from the diagonal), and effectively enables us to get some intuition about the interactions. The coarsegrain, however, does not interpolate the `NaN`s created when filtering out whole bins in the balancing step (horisontal and vertical lines in Figure 3.6 and Figure 3.7; middle). This is done in a subsequent step by linearly interpolating the `NaN`s. Examining the interpolated matrix on full chrX (Figure 3.6; right) gives the impression that the pericentromeric (at ~60 Mbp) region harbours a *very* strong compartment, but that is clearly an artefact of the interpolation on the very large empty region of the centromere, where the diagonal is somehow extended in a square. On the thinner lines, the interpolation seem to be more smooth, and barely noticable on the diagonal.

`NaN` histograms As expected, most of the low quality bins are located on the edges of the chromosome arms, especially the region around the centromere [ref litterature], as they contain many repetitive sequences. The low-quality bins are filtered out by the balancing algorithm, those bins are `NaN` in the Hi-C matrix. The median position of the `NaN` values (Figure 3.8) ranges between 58 and 63.5, which is within the estimate of the centromeric region of *rhemac10*.

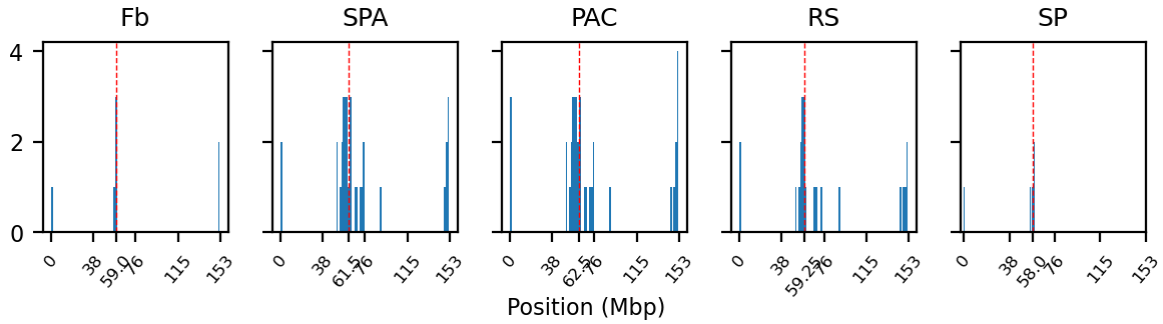
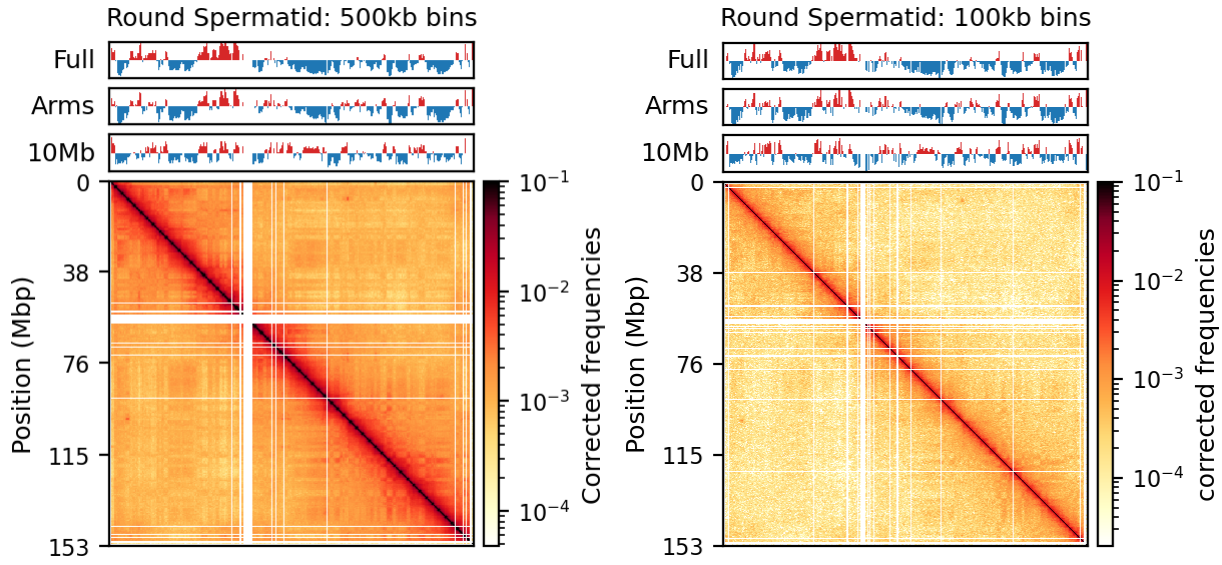


Figure 3.8: Histogram of NaN values in the E1 eigenvector for each cell type. Median position is marked with a red dashed line.

The fact that the medians lie within the centromeric region on all cell sources shows both that the majority of the bad bins are in the (peri)centromeric region *and* there are approximately equally many on each side.

3.2.3 Compartments (Eigenvectors)

The three viewframes (*Full*, *Arms*, *10Mb*) for the calculation of the eigenvectors captured different variability in the data (Figure 3.9a), and as expected, the inferred compartments (red) are more abundant and smaller with smaller viewframes. To determine how well each of the E1 tracks capture the pattern in the interaction matrix, we can overlay the matrix with the E1 sign-change and visually determine if the squares reflect the E1 sign change (Figure 3.9a).



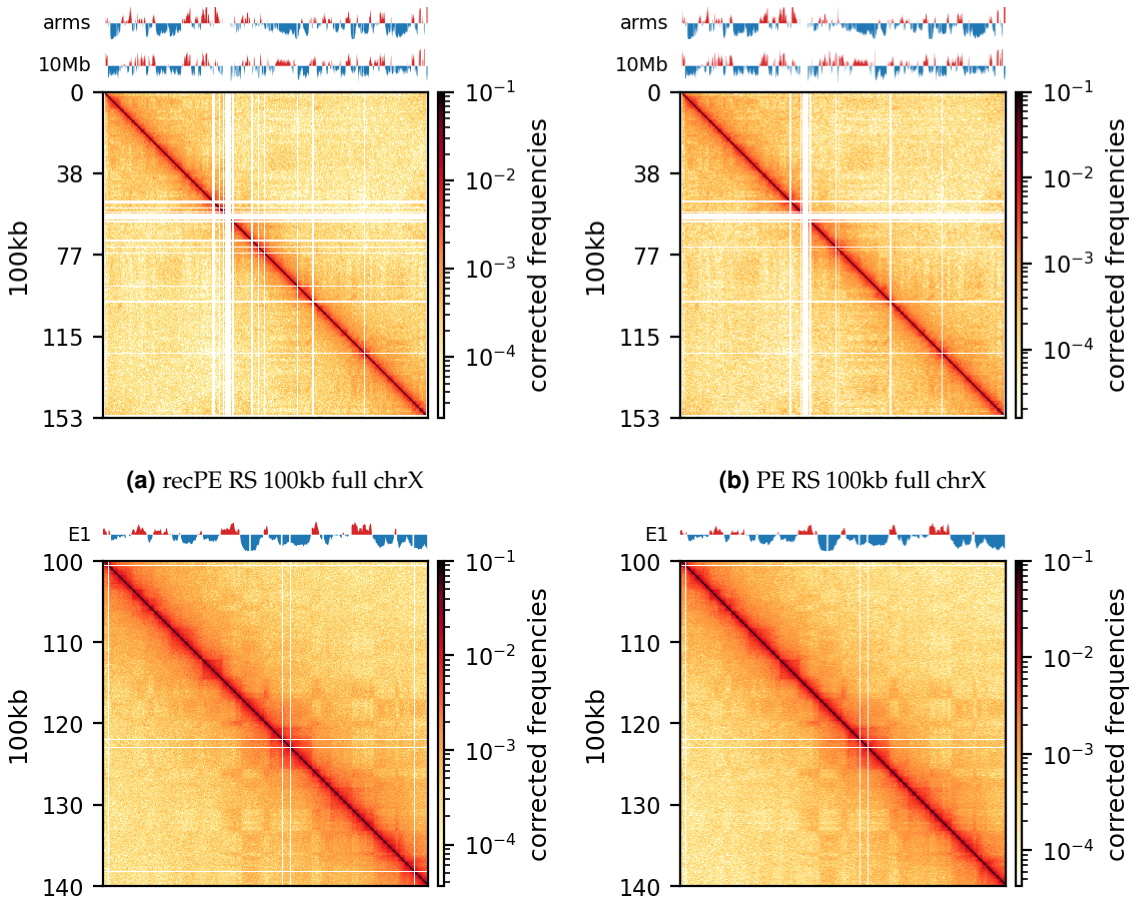
(a) E1 eigenvector values for merged round spermatid samples at 500kb resolution, as well as the interaction matrix. E1 was restricted to either Full-chromosome (top), Chromosome-arms (middle), or 10Mb windows (bottom).

(b) E1 eigenvector values for merged round spermatid samples at 500kb resolution, as well as the interaction matrix. E1 was restricted to either Full-chromosome (top), Chromosome-arms (middle), or 10Mb windows (bottom).

Figure 3.9: Super caption, subcaptions should be moved here (from notebook). They now fit on the page, but it would be nice to make this as one plt.subplots with shared axis title etc. *Update:* The notebook is ready for making these plots (bottom section).

I argue that without more finescaled knowledge than the position of the centromeres, the arbitrary size of the 10 Mb windowed E1 can not fully be justified. Also, Wang et al. [2019] concludes that only the pachytene spermatocyte showed local interactions in that viewframe (what they refer to as *refined A/B-compartments*), and all the other stages of spermatogenesis were consistent with the conventional A/B compartments. The reasonable thing to do is therefore to continue the analysis, focusing on the arms-restricted eigendecomposition. Nevertheless, all three viewframes were used for further analysis.

Additionally, as I created coolers with two different sets of parsing parameters we will compare the resulting matrices and their compartments



(a) recPE RS 100kb full chrX

(b) PE RS 100kb full chrX

(c) recPE RS 100kb chrX:100Mb-140Mb

(d) PE RS 100kb chrX:100Mb-140Mb

Figure 3.10: Comparing two Hi-C matrices

4 Discussion

Here is the discussion

Bibliography

Silvio Bicciato and Francesco Ferrari, editors. *Hi-C Data Analysis: Methods and Protocols*, volume 2301 of *Methods in Molecular Biology*. Springer US, New York, NY, 2022. ISBN 978-1-07-161389-4 978-1-07-161390-0. doi: 10.1007/978-1-0716-1390-0. URL https://link.springer.com/10.1007/978-1-0716-1390-0.

Maxim Imakaev, Geoffrey Fudenberg, Rachel Patton McCord, Natalia Naumova, Anton Goloborodko, Bryan R Lajoie, Job Dekker, and Leonid A Mirny. Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nature Methods*, 9(10):999–1003, October 2012. ISSN 1548-7091, 1548-7105. doi: 10.1038/nmeth.2148. URL https://www.nature.com/articles/nmeth.2148.

John Jaenike. Sex Chromosome Meiotic Drive. *Annual Review of Ecology and Systematics*, 32(1): 25–49, November 2001. ISSN 0066-4162. doi: 10.1146/annurev.ecolsys.32.081501.113958. URL https://www.annualreviews.org/doi/10.1146/annurev.ecolsys.32.081501.113958.

Erez Lieberman-Aiden, Nynke L. Van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R. Lajoie, Peter J. Sabo, Michael O. Dorschner, Richard Sandstrom, Bradley Bernstein, M. A. Bender, Mark Groudine, Andreas Gnirke, John Stamatoyannopoulos, Leonid A. Mirny, Eric S. Lander, and Job Dekker. Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome. *Science*, 326(5950):289–293, October 2009. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1181369. URL https://www.science.org/doi/10.1126/science.1181369.

Kasper Munch. Munch-group, November 2024. URL https://munch-group.org/research.html.

Open Chromosome Collective. Open Chromosome Collective (Open2C), 2024. URL https://open2c.github.io/.

Laurits Skov, Moisès Coll Macià, Elise Anne Lucotte, Maria Izabel Alves Cavassim, David Castellano, Mikkel Heide Schierup, and Kasper Munch. Extraordinary selection on the human X chromosome associated with archaic admixture. *Cell Genomics*, 3(3):100274, March 2023. ISSN 2666979X. doi: 10.1016/j.xgen.2023.100274. URL https://linkinghub.elsevier.com/retrieve/pii/S2666979X23000344.

Erik F. Sørensen, R. Alan Harris, Liye Zhang, Muthuswamy Raveendran, Lukas F. K. Kuderna, Jerilyn A. Walker, Jessica M. Storer, Martin Kuhlwilm, Claudia Fontsere, Lakshmi Seshadri, Christina M. Bergey, Andrew S. Burrell, Juraj Bergman, Jane E. Phillips-Conroy, Fekadu

Shiferaw, Kenneth L. Chiou, Idrissa S. Chuma, Julius D. Keyyu, Julia Fischer, Marie-Claude Gingras, Sejal Salvi, Harshavardhan Doddapaneni, Mikkel H. Schierup, Mark A. Batzer, Clifford J. Jolly, Sascha Knauf, Dietmar Zinner, Kyle K.-H. Farh, Tomas Marques-Bonet, Kasper Munch, Christian Roos, and Jeffrey Rogers. Genome-wide coancestry reveals details of ancient and recent male-driven reticulation in baboons. *Science*, 380(6648):eabn8153, June 2023. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.abn8153. URL https://www.science.org/doi/10.1126/science.abn8153.

Yao Wang, Hanben Wang, Yu Zhang, Zhenhai Du, Wei Si, Suixing Fan, Dongdong Qin, Mei Wang, Yanchao Duan, Lufan Li, Yuying Jiao, Yuanyuan Li, Qiujun Wang, Qinghua Shi, Xin Wu, and Wei Xie. Reprogramming of Meiotic Chromatin Architecture during Spermatogenesis. *Molecular Cell*, 73(3):547–561.e6, February 2019. ISSN 10972765. doi: 10.1016/j.molcel.2018.11.019. URL https://linkinghub.elsevier.com/retrieve/pii/S1097276518309894.

Bioinformatics Research Centre
Department of Molecular Biology and Genetics
Aarhus University
Universitetsbyen 81
8000 Aarhus C
Denmark