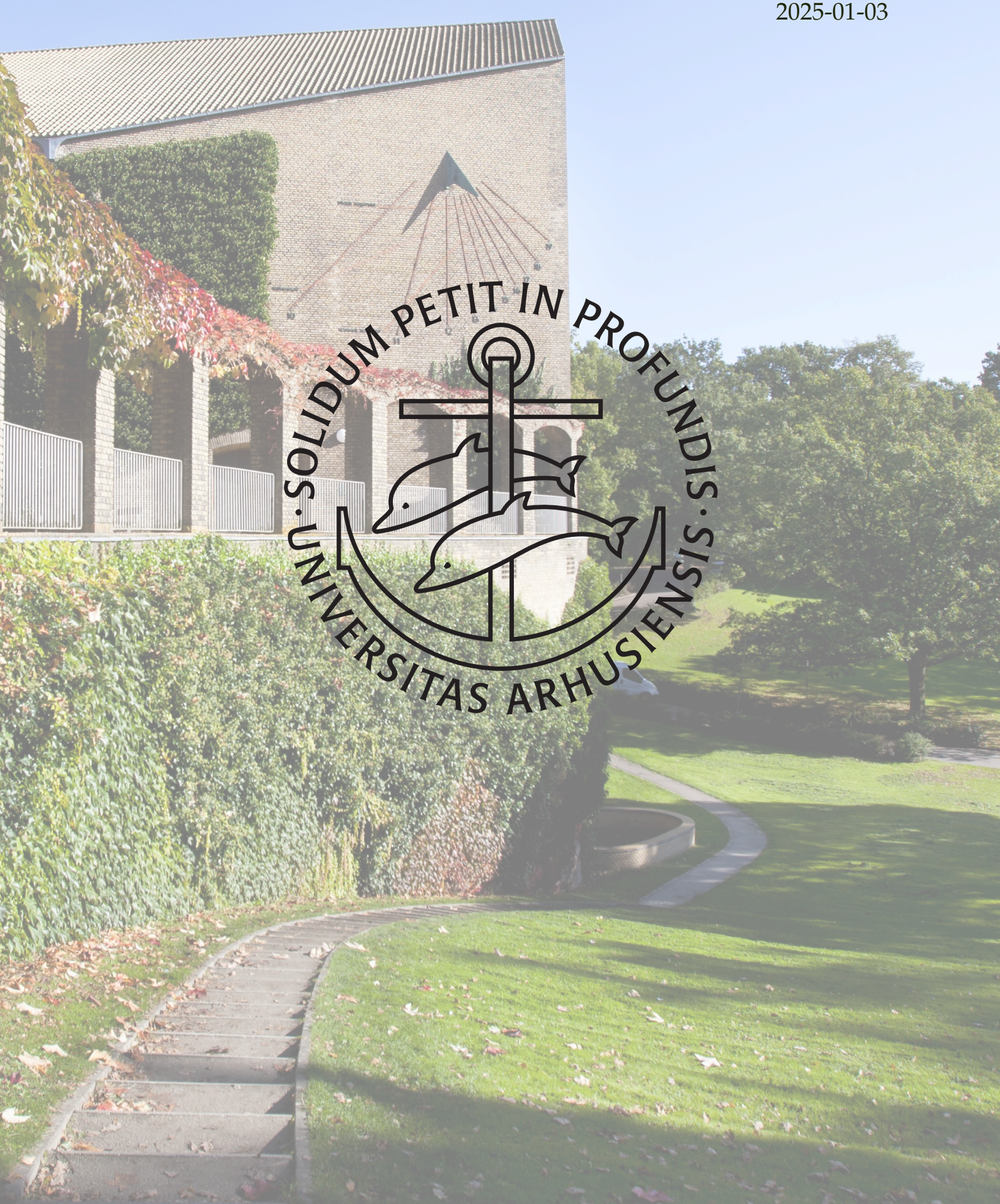
Thesis

Chromatin Compartments and Selection on X

Søren Jørgensen

2025-01-03



Chromatin Compartments and Selection on X

How Edges of Active Chromatin Align with Selection Regions in Primates

Søren Jørgensen

MSc. Bioinformatics

2025-01-03

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

“The hemizygosity of male mammals makes the X chromosome uniquely exposed to selective pressures, as the lack of a second copy provides no buffer against deleterious mutations. This, combined with a high density of essential genes related to reproduction and brain function, suggests the existence of biological mechanisms that protect the integrity of the X chromosome. One such mechanism is meiotic sex chromosome inactivation (MSCI) in males, while another could involve chromatin architecture.

In this study, the 3D chromatin architecture of the X chromosome in rhesus macaque (*Macaca mulata*) is investigated in the context of evolutionary pressures and genetic drivers. We redo Hi-C analyses from a 2019 paper on the latest reference genome (*rheMac10* or *Mmul_10*). We compare two Hi-C analysis frameworks, *HiCExplorer* and *cooler/cooltools* (Open2C), on a subset, finding Open2C to be most flexible and intuitive. The ICE method (Iterative Correction and Eigendecomposition) was used to infer conventional and refined A/B compartments for fibroblast and four stages of spermatogenesis.

We find that 200 kbp transition-zones between A/B-compartments in both fibroblasts and round spermatids align well with strong selective sweeps in humans (ECH-regions), and with strong negative selection in baboons (*Papio* spp.). We discuss the biological meaning of these findings, where conserved chromatin features may help to retain non-advantageous alleles, hinting to the role of selfish genetic elements in genome evolution.

- Søren Jørgensen”

Table of contents

1 Introduction

1.1 Sexual reproduction and Sex Chromosomes

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.

Sex chromosomes behave differently than autosomes for a long list of reasons. One reason is hemizygosity, where there is only one copy of a chromosome. The Y chromosome exist only in males, and (generally) never more than one copy is in the same individual. The copmlexity increases when we tend to the X chromosome, which exist both alone (in males) and as two copies (in females). This skewed ratio means that X chromosomes exist 2/3 of the time in females and only 1/3 of the time in males, and it is more exposed in males as there is no other copy to take over loss of function. It is also the underlying reasoning for Haldane's rule [ref], postulating that the heterogametic sex (XY, ZW) is the first to dissapear or have severely lowered fitness (e.g. sterility) when crossing different species. Even today, a hundred years later, there is only hypotheses as to why this is observed, including *Y-incompatibility* (Y has to be compatible to X or autosomes), *dosage compensation* (hybridisation deregulates crucial dosage compensation in heterogametes), *dominance* (recessive genes causes sterility), *faster-male* (male reproductive genes diverge faster than female), *faster-X* (X-linked loci diverge faster than autosomal ones), and *meiotic drive* (discrepancy between drivers and surpressors on sex chromosomes leads to sterility) [ref]. While is appears to be the same phenomenon across multiple *taxa* (even *kingdoms*), it has received different explanations, and consensus for different species are not the same. Often times even, several of the listed explanations are described to be acting in collaboration [ref].

- ☐ include something about conserved synteny on chrX here

The complexity of selection on the X chromosome therefore still an area of active investigation, and several studies infers selective strong selection on the X chromosomes across primates [ref]

1.2 Known selection on X

- ☐ Selective sweeps (or is it?) in humans (ECH90; Skov et al. (2023))

- Explain why it makes sense to compare these data sets

- Negative selection in baboons w.r.t. minor parent ancestry
  - Explain why it makes sense to compare these datasets
  - The low-diveristy in *p.hamadryas* are very wide - maybe chromatin

1.3 Gene drivers

- Talk about what makes a genetic driver, what criteria

Gene drive occur when a particular collection of genes is propagated through a population by increasing the probability of transmitting the genes to the offspring from random (Mendelian) inheritance, resulting in a biased gene transmission against its alternative. Two categories of gene drivers exist (Bravo Núñez, Nuckolls, and Zanders 2018); *class one drivers* affect chromosome segregation in meiosis, and *killer meiotic drivers* will sabotage meiotic product that have not inherited the driving allele [ref]. This happens regardless of the fitness effects of the developed organism, and is there often a factor offsetting classical selection. Even though the implications of such systems are potentially detrimental, they are notoriously difficult to detect. A circumstance contributing to the difficulty is that most genetic experiments are done in homozygotes. Bravo Núñez, Nuckolls, and Zanders (2018) states that the general choice of experimental system may have biased our understanding of sexual reproduction. A key point is; meiotic drivers can only be observed in heterozygotes, where a genetic driver has a competitor.

Interestingly, gene drive is much more well-documented on sex chromosomes than for autosomes. Possibly because the sex chromosome meiotic drive inherently causes a skewed sex ratio, more notably raising a flag for further analysis. Another point to consider is, in the case of sex chromosome drive, a fully driving gene will lead to the extinction of the species [ref], as only one (fertile) offspring will be produced.

1.4 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.

1.5 Chromatin Architecture

- ☐ Chromatin organization is highly conserved between species
- ☐ Explain why take offset in Wang et al. (2019)
- ☐ Exlpain the rationale of reproducing results with the latest reference
- [ ]

1.6 3C: Chromatin Conformation Capture

- ☐ This might not be needed or will be merged with Hi-C section

1.7 Hi-C: High-Throughput 3C

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.

1.7.1 Hi-C Library preparation

A specialized protocol for preparing the DNA library is necessary [Lieberman-Aiden et al. (2009); Fig. 1a]. Briefly, formaldehyde is used to crosslink spatially adjecent chromatin. Restriction enzyme *HindIII* is used to digest the crosslinked chromatin, leaving sticky ends, `5-AGCT-3`, that are filled and biotinylated with a polymerase (using either biotinylated A, G, C, or T). The strands are ligated in highly dilute conditions, which is favoring the ligation of the two crosslinked strands, forming chimeric, biotinylated strands. Upon ligation, the restriction site is lost as a biotinylated `5-CTAG-3` site (also referred to as the *ligation junction*) is formed. Lastly, the ligation junctions are isolated with streptavidin beads and sequenced as a paired-end library.

To be able to create stage-resolved Hi-C library of spermatogenetis, several steps have to be performed on the samples before crosslinking. First, the samples have to be treated immediately after harvesting to ensure viable cells. Secondly, the samples have to be purified to accurately represent each stage og spermatogenesis. Specifically, the data for this project (Wang et al. (2019); acc. GSE109344) preluded the library preparation protocol by sedimentation-based cell sorting to separate live spermatogenic cells into different stages of differentiation, namely spermatogonia, pachytene spermatocyte, round spermatid, and spermatozoa. Then, the cells were fixed in their respective state before crosslinking. They use their own derived method for library preparation, termed small-scale *in-situ* Hi-C, allegedly producing a high-quality Hi-C libary from as little as 500 cells (capturing the variance of millions of cells).

1.7.2 Hi-C Data Analysis

The analysis of the read-pairs of a Hi-C library is divided into several smaller tasks, see [ref fig-hic-analysis-flow].

We must align the reads to the reference in such a way that the *intentional* chimeric read-pairs (as per above-mentioned protocol) are rescued, and *unintentional* read-pairs are discarded. That is, they represent ligation junction of adjecent chromatin segments, or they represent technical artefacts or unintentional fusions of unrelated DNA.

Aligning the Hi-C reads Any software for local alignment can be used for aligning reads from a Hi-C library. However, one should make sure to disable paired-end rescue mode if possible, otherwise each read in a pair (each mate) should be aligned separately (Lajoie, Dekker, and Kaplan 2015). This removes the assumption that the distance between mates fits a known distribution because the genomic sequences originate from a continuous DNA-fragment. For example, the *bwa-mem* [ref] implementation of this (the `-P` option) activates the Smith-Waterman algorithm to rescue missing hits, but disables the search of hits that fit a 'proper' pair. After alignment, each read is typically assigned to the nearest restriction fragment to enable categorization of pairs into different categories.

Interestingly, this last step is not included by default in *pairtools*, as [ref] observe very similar statistical properties on pairs that are either close or distant from the nearest restriction site. Thus, restriction fragment filters are not needed, and in stead, a simple filter is applied against short-distance pairs that is automatically calibrated.

Identifying and Storing Valid Hi-C Pairs One should be cautious when filtering invalid from valid pairs, as they are not easily distinguished. A ligation event will be categorized into one of five categories (see Table 1.1): *dangling-end*, *self-circle*, *weird*, *intrachromosomal* (cis), and *interchromosomal* (trans) (Bicciato and Ferrari 2022, Ch. 1). Either *dangling-end* or *self-circle* events are reported if

a read-pair maps to the same restriction fragment depending on the orientation, and deemed uninformative (Lajoie, Dekker, and Kaplan 2015). Usually, *weird* events are demeed uninformative as well, as it is challenging to distinguish a sequencing error from the result of a diploid fragment.

PCR duplicates should be discarded as well, having either identical genomic sequence, or sharing exact 5′ alignment positions of the pair (Lajoie, Dekker, and Kaplan 2015; Bicciato and Ferrari 2022, Ch. 1). The probability that such pairs are valid (i.e. there are multiple of the same pairs) is very low.

The *trans*/*cis*-ratio can be a good indicator of the noise level in the library, and additionally, the level of random ligation events can be quantified by counting the number of *trans* events occurring to mitochondrial genome. They should not occur naturally, as the mitochondrial genome is separated from the DNA in the nucleus. This method has some pitfalls that should be controlled for; some parts of the mitochondrial genome can be integrated into the host genome, and mitochondrial count may differ between states.

Table 1.1: Five categories of ligation events and a short explanation. *Hi-C Data Analysis: Methods and Protocols Ch. 1*.

Event name	Explanation
Dangling-end	Non-digested collinear fragments. Fraction can be high.
Self-circle	Collinear fragment(s) have circularized. Very low fraction could indicate unsuccesful ligation.
Weird	Mates have the same orientation on the reference. Is not possible with single copy fragment. Either sequencing errors or diploid fragments[1].
Cis	Pairs from the same chromosome (intrachromosomal)
Trans	Pairs from distinct chromosomes (interchromosomal)

Quality Control and Interaction Matrices To determine the quality of the Hi-C library, most tools generate quality control log files at some point during the filtering steps, which can then be aggregated and analyzed (with e.g. MultiQC [ref]). The ratios between the different ligation events can be informative about the quality of the Hi-C library. Here, both the distribution of discarded reads across categories, as well as the ratios between *cis*/*trans* interactions for a certain organism provide information about the library. For example, the biases of different aligners might be captured by comparing the reason why reads are discarded between two different aligners, as well as if there is a preference of *cis* or *trans* in an aligner itself. This allows for evaluating the mapping parameters as well as the filters applied downstream. Additionally, $P(s)$, the contact probability as a function of genomic separation can be inspected as it should decay with increasing distance.

[1] Bicciato and Ferrari (2022) mentions that this type of ligations had been used to infer interaction between sister-chromatids post-replication in *Drosophila*.

Typically, a filter against low mapping quality is applied on the data before constructing the interaction matrix (Hi-C matrix), and a conventional threshold is $mapq < 30$ (Bicciato and Ferrari 2022). However, a considerable amount of reads do not pass that threshold, and thus we risk discarding potential valid information and should make sure to have enough data. Consequently, *HicExplorer* defaults a lower threshold ($mapq < 15$), and *pairtools* enforces no filter by default, but recommends setting this manually (starting at $mapq < 30$).

A Hi-C interaction matrix simply maps the frequency of interactions between genomic positions in a sample. The maximum resolution of a Hi-C matrix is defined by the restriction enzyme, where the size of the restriction site (probabilistically) determines average space between each cut. With a 4 bp restriction site, the fragments will average $4^4 = 256bp$ and similarly $4^6 = 4096bp$ for a 6 bp restriction site. This leads to ~12,000,000 and ~800,000 fragments, respectively. Very deep sequencing is required to achieve enough coverage to analyze the interaction matrix at the restriction fragment resolution, but, usually, such high resolution is not required. Therefore, it is practice to bin the genome into fixed bin sizes, which also enables a more efficient handling of the data if the full resolution is not needed (e.g. when plotting large regions such as a whole chromosome). The conventional format to store a Hi-C matrix, consisting of large multidimensional arrays, is HDF5. Each HDF5 file can store all resolutions and metadata about the sample, resolutions typically ranging from 10kb to 1Mb. Typically, the stored resolutions should be multiples of the chosen base-resolution, as the lower resolutions are constructed by recursive binning of the base resolution. *cooler* [ref] neatly offers efficient storage with sparse, upper-triangle symmetric matrices and naming-conventions of the groups in their *.h5*-based file format, *.cool*, and they provide a Python class `Cooler` as well for efficiently fetching and manipulating the matrices in Python.

Inferring from the matrix (Calling Compartments) The raw frequency matrices are generally not very informative, as the contact frequencies vary greatly between bins and contain biases in addition to the $P(s)$ decay, which results in a diagonal-heavy matrix with high amount of noise the further we travel from the diagonal. Therefore, to analyze the three-dimensional structure of the chromatin, a method for correcting (or balancing) the raw Hi-C matrix has to be applied. It is unadvisable to correct low-count bins as it will greatly increase the noise, or to correct very noisy bins, or very high-count bins. Therefore, some bin-level filters are applied before balancing (Lajoie, Dekker, and Kaplan 2015);

- Low-count bins are detected by comparing bin sums to the distribution of bin sums with a percentile cutoff,
- Noisy bins are detected by comparing bin variance to the variance distribution of all bins (and percentile cutoff), and
- Outlier point-interactions are removed (a top-percentile of bin-bin interactions)

A widely used balancing method is Iterative Correction and Eigendecomposition (ICE) (Imakaev et

al. 2012), which utilizes a data-driven approach for correcting multiplicative biases. Briefly, is based on an assumption of equal visibility of all loci, and uses the pairwise and genome-wide structure to generate a set of biases along with a map of relative interaction frequencies by iteratively dividing each row, then each column, by its mean until convergence. This results in a uniform coverage profile (corrected coverage), yielding a smoother interaction matrix with slower transitions, thus greatly reduces visibility-induced biases. It does not distinguish between the sources of biases, and thus calculates a collective bias for each position. Imakaev et al. (2012) show that *known* biases are factorizable by comparing their results to predictions of restriction fragment biases, GC content, and mappability from a computationally intensive probabilistic approach. By showing that the product of those known biases explain > 99.99 of the variability in their bias estimation, they argue both known and unknown biases will be captured with their iterative correction method (also denoted *matrix balancing*).

Even with a binned, filtered, and balanced matrix, we are still left with the challenge of translating the matrix into biologically relevant inferations. Importantly, we have to remember that the matrix arise from a collection of cells and that the interaction frequency cannot be translated to a fraction of cells. Additionally, the effect from averaging interaction patterns can cause both individual patterns to be burried and the average pattern to show a pattern that does not exist in any of the single cells. Therefore, when pooling matrices one must make sure that the samples are as similar as possible (e.g. the same differentiation stage and so on). We can also not distinguish interactions that either co-occur in the same cell or ones that are mutually exclusive. Lastly, the way interaction patterns are defined poses a challenge; we define the chromatin compartments to be the output of a method, the 'E' in 'ICE', eigendecomposition, not as a specific pattern that we can explicitly search for. Although experimentally verified to tightly correlate with chromatin states, the inferred compartments vary with different methods of calculating the eigenvector, as dicussed in Section 2.3.2 and Section 3.2.3. To further complicate the challenge, interaction patterns on different scales co-exist and are difficult to disentangle without simplifying assumptions such as small-scale interactions are not visible (or they are negligible) at a certain resolution, or restricting the viewframe to eliminate large-scale variance between chromosome arms. It is by definition a speculative exercise to interpret the biological relevance of an observed pattern, but the consensus is to call compartments on interacting regions that arise from the eigendecomposition of a Hi-C matrix without further modifications (Lajoie, Dekker, and Kaplan 2015). As the eigenvector is only unique up to a sign, a phasing track is used to orient the eigenvector, aiming for a positive correlation with GC content (in mammals), making A-compartments represent the active euchromatin, and B-compartments the closed heterochromatin.

Compartment Edges and Genomic Intervals As arbitrary as a compartment may be defined, we chose to define another genomic interval for analysis. It is well (Bicciato and Ferrari 2022, Ch. 3) known that CTCF and other structural proteins preferentially binds to Topologically Associating

Domains (TADs; they were initially defined as sub-Mb chromatin structures (Lajoie, Dekker, and Kaplan 2015), but now the definition seems to vary based on the method of extraction [ref cooltools]). Derived from this, we define a transition zone between A/B compartments to look for enrichment of specific regions of interest.

We can test if two sets of genomic intervals correlate (say, compartment edges and ECH regions) by either proximity of the non-intersecting parts of the sets, or by intersection over union (Jaccard index). When the underlying distribution of a statistic (or index) is unknown, a widespread method in bioinformatics for estimating a p-value is by bootstrapping. Here, one of the sets are bootstrapped (the intervals are placed at random positions) a number of times, b, and the fraction of statistics more extreme than the one we observe is reported as the p-value.

1.8 Reproducibility Infrastructure

First, describe the importance of reproducibility and the scientific method. Include a bit of scientific skeptiscism of there is time.

Thus, apart from the biological questions we seek to investigate and answer in this thesis, a major goal of the thesis is to create fully (and easily) reproducible results through a self-contained and version-controlled pipeline using git [ref], GitHub [ref], quarto [ref], Conda [ref], gwf [ref], and Jupyter [ref]. See Table 1.2 for a brief introduction.

Table 1.2: Overview of the tools used for reproducibility of this thesis.

Tool	Description
Jupyter	Interactive coding environment for analysis and development (notebooks are natively rendered with Quarto)
Quarto	A Quarto Manuscript project nested inside a Quarto Book for rendering html (website) and PDF (manuscript) from Markdown via Pandoc. Supports direct embedding of output from Jupyter Notebook cells (plots, tables).
Conda	For managing software requirements and dependency versions reproducibly.
git	Version control and `gh-pages` branch for automated render of Quarto project
GitHub	Action was triggered `on push` to render the project and host on munch-group.org
gwf	Workflow manager to automate the analysis on a HPC cluster, wrapped in Python code. `workflow.py` currently does everything from `.fastq` to `.cool`, but notebooks can be set to run sequentially as part of the workflow as well.

1.8.1 GWF: workflow management for High-Performance Computing (HPC)

To enable consistently reproducing the analyses, a workflow manager is used. Several exist, but the most well-known must be *snakemake* [ref]. However, we use the pragmatic (their own words),

lightweight workflow manager *gwf*, which is optimized for the GenomeDK insfrastructure, and has the benefit of in-house support.

Briefly, *gwf* works on a python script, conventionally `workflow.py`, that wraps all the jobs (*targets* in gwf lingo) you will submit to the HPC cluster. Each target is submitted from a template, written as a Python function, which includes `inputs` and `outputs` that *gwf* should look for when building the depency graph, `options` list of resources that is forwarded to the queueing system (*Slurm* in our case), and `specs`, specifying the submission code in Bash as a formatted Python string (meaning we can pass Python variables to the submission code), providing an extremely flexible framework for running large and intensive analyses in a high-performance computing environment.

1.8.2 Project Initialization

gwf The initialization of the project directory is the basis of reproducibility and transparency, together with `workflow.py` inhabiting the main directory. Specifically, it includes a subdirectory for (intermediate) files that are produced by the pipeline, `steps/`. Everything in this directory is reproducible simply by re-running the *gwf*-workflow. It is thus not tracked by `git`, as the large files (raw reads, aligned read-pairs, etc.) it contains are already indirectly tracked (`workflow.py` is tracked). It can be safely deleted if your system administrator tells you to free up disk space, although you would have to run the workflow again to continue the analysis. Several directories are created for files that are not produced by the pipeline, that is, files that the workflow uses, configuration files, figures edited by hand, etc. Ideally, as few files as possible should be outside of `steps/`, to be as close as possible to an automated analysis.

Jupyter Notebooks A `notebooks/` subdirectory contains Jupyter notebooks that are named chronologically, meaning they operate on data located in either `steps/` or generated from a previous notebook. This way, the workflow can also be set up to run the notebooks (in order) to produce the figures, tables, and their captions used in this manuscript.

Quarto Quarto is an open-source scientific and technical publishing system that uses (pandoc) markdown to create and share production quality output, integrating Jupyter Notebooks with Markdown and LaTeX and enabling embedding content across *.ipynb* and *.qmd*. In *.qmd*, code chunks in several programming languages can be executed and rendered, including Python, R, mermaid (JavaScript-based diagramming). A Quarto project is configured with a YAML configuration file (`_quarto.yml`) that defines how output is rendered. In this project, we use a nested structure, nesting a *Slides* project and a *Manuscript* project inside a *Book* project. To manage the directory as a Quarto Book project, a quarto configuration file was placed at the base, defining how the Book should be rendered. Additionally, configuration files were placed in `slides/` and `thesis/`, to render them as Quarto Slides and Quarto Manuscript, respectively. This nested structure lets us render different subprojects with different configurations than the main project, for example to

generate the manuscript, a single Quarto Markdown file, in both *.html* and *.pdf*, and only including embedded outputs from specified cells from notebooks in the parent directory. Although the Quarto framework is extensive, it is still under development and has several drawbacks worth mentioning. First, one can only embed the output of code cells from notebooks, meaning the only way to embed text with a python variable (e.g. you want the manuscript to reflect the actual value of a variable, sample sizes `n = [1000, 10000, 100000]`, and their respective outputs) is by converting a formatted python string into Markdown and send it to the output. Second, embedded figures will be copied as-is in the notebook, and thus cannot be post-processed with size or layout. This makes it impractical to e.g. use the same figures in slides and in the manuscript. Third, when rendering large projects that is tracked by git, some output files (that have to be tracked to publish the website) can exeed GitHub size limits. Especially if rendering in the *jats* format, producing a MECA Bundle that should be the most flexible way to exchange manuscripts [ref]. However, as not applicable to this thesis, the option was simply disabled. Fourth, some functionality relies on external dependencies that cannot be installed on a (linux) remote host (GenomeDK), such as relying on a browser for converting `mermaid` diagrams into png for the pdf-manuscript.

1.8.3 git and GitHub

To track the project with git and GitHub, the abovementioned structure was initialized as a GitHub repository, including a workflow for GitHub Actions to publish and deploy the website on the `gh-pages` branch when pushing commits to `main`. Briefly, it sets up a virtual machine with Quarto and its dependencies, renders the project as specified in the `_quarto.yml` configuration file(s), and publishes the project on the group website munch-group.org.

1.9 Our research question

In this project, we formulate two main objectives:

A Redo the Hi-C analyses from (Wang et al. 2019) using the latest macaque reference genome, *rheMac10*, with some modifications. We decided to use *HiCExplorer*, a Python-based software for command line use, and supplement the analyses with the *Open2C Ecosystem* ("Open Chromosome Collective (Open2C)" n.d.) that 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 extended common haplotypes (strong selective sweeps) that are found in *human*, and with regions of negative selection of minor parent ancestry in baboons. Investigate the biological meaning of the results. We use in-house software to compare genomic intervals.

2 Methods

All computations were performed on GenomeDK (GDK) [ref], an HPC cluster located on Aarhus Uninversity, and most of the processing of the data was made into a custom *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.

We decided to 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. In hindsight, it would have been more sensical to use *HiC-Pro* to get normalized contact maps, then continue analyzing with *cooler*/*cooltools*, then compare the results evenly with the results achieved from using Open2C from start to finish.

2.1 Initial Exploration with 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.

mapping, parsing, filtering, build matrix

`hicBuildMatrix` both parse and filter the mapped reads. The default value was used, where alignments with $mapq < 15$ are discarded.

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
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

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

	source_name	GB	Bases	Reads
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 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.

Open2C Suspiciously, ("Open Chromosome Collective (Open2C)" n.d.) 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 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

Index	Name	Description
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 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) 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 decided 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 matrices *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 $\log 1\mathrm{p}(f_i) = \log\left(1 + f_i\right)$. 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.

I implemented a plotting utility, `plot_for_quarto` in notebook `07_various_plotting.ipynb` that is compatible with the YAML cell-options read by Quarto's `embed` shortcode. It will take an arbitrary number of samples and plot a chromosome (or region) with or without its respective E1 value for either of the three viewframes that has been created. The input is a (subsetted) *pandas DataFrame*, defined from a file search matching a pattern specified to the `glob` Python module.

2.3.3 Compartments and Their Edges (Transitional Regions)

From the eigenvectors, the A-compartments were extracted in bedgraph-format (`['chrom', 'start', 'end']`) and compared with ECH90 regions lifted to *rheMac10* from human [ref what reference?]. We perform visual inspection of the genomic intervals and test whether ECH90 regions are enriched near the edges of the compartments by defining a 200 kilobase transition-zone centered at each sign change of E1 (referred to as *compartment edge*). We compare genomic intervals (or sets) both visually by plotting the regions, and by a proximity test and bootstrapping the Jaccard index.

Proximity test Determines whether the non-overlapping parts of the sets are more proximal than expected by chance. We define the *annotation* set and the *query* set, and the distance from each interval on the *query* to the most proximal interval on the *annotation* is used to generate an index of proximity. It uses bootstrapping ($b = 100000$) to generate the null distribution, and finally, the fraction of the *null* as or more extreme as our observed proximity is reported as the p-value.

Jaccard test Measures the significance of the observed Jaccard index (intersection over union) between two sets. The index is a measure of *similarity* between two sets (between 0 and 1), which is very sensitive to the size difference between the sets, as even when comparing a set of intervals

to a small subset of itself will yield a very small Jaccard index. When we use bootstrapping to generate a null distribution (shuffling the intervals of the *query*), we generate the probability that the two sets (with their respective number and size of intervals), are as similar or more than what we observe. The ratio is reported as the p-value. However, this approach is still sensitive to flipping of query/annotation, as only the query is bootstrapped.

Multiple testing Careful considerations were made to avoid multiple testing biases (p-hacking): Performing tests on all combinations of variables (cell type, resolution, viewframe, flip annot, query, test) will yield 260 p-values, and we would have to adjust the significance threshold (with $\alpha = 0.05$, we expect 13 'significant' tests by chance alone). However, if we choose only

3 Results

3.1 Exploration with HicExplorer

3.1.1 Quality Control

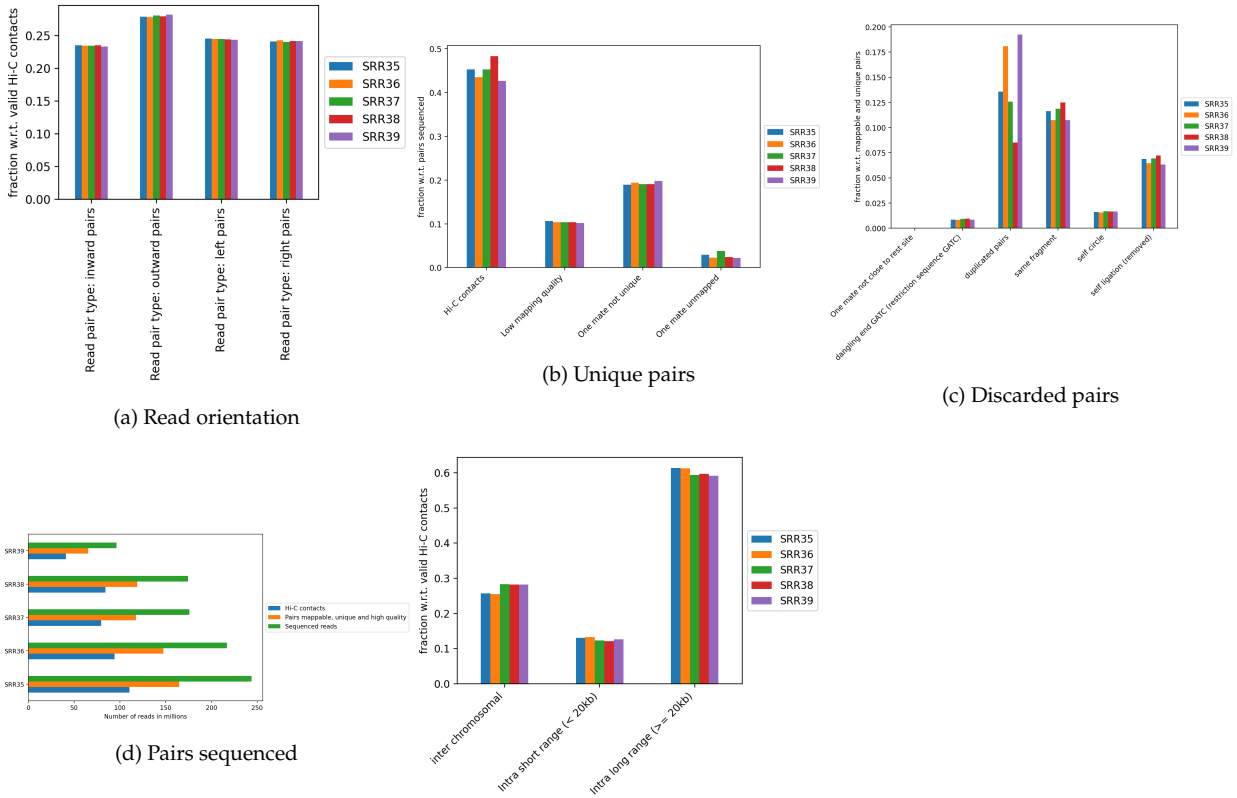
The separately mapped read-mates were parsed into a *.h5* interaction matrix by `hicBuildMatrix`, which include a *.log* file documenting the builtin quality control (hereafter, *QC*). Log files from the 5 samples were merged with `hicQC` (Figure 3.1). We observe showed equal fractions of the read-orientation 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, 2301:236; 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 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



(a) Read orientation

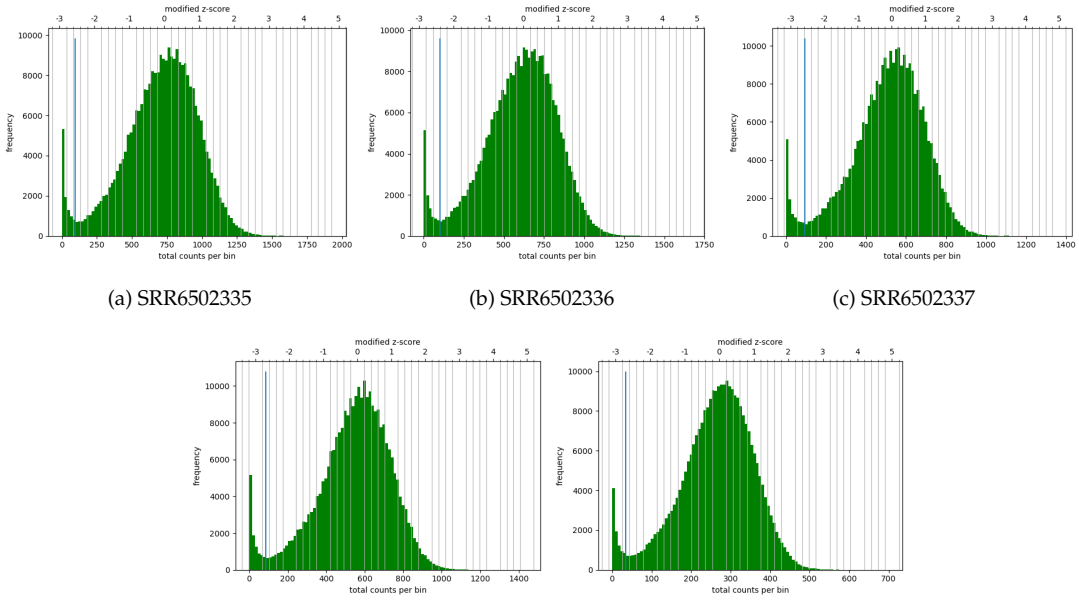
(b) Unique pairs

(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.

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.

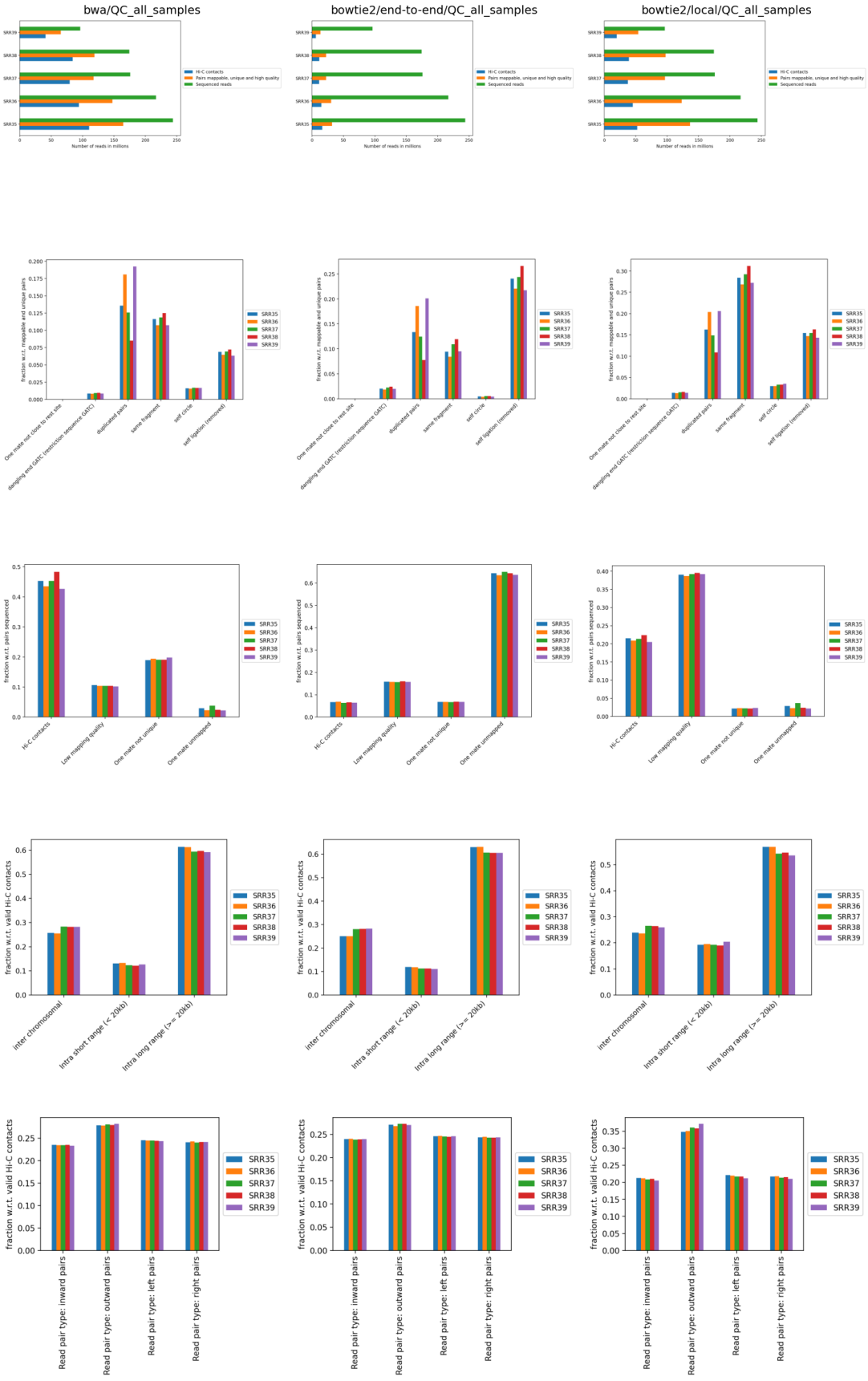
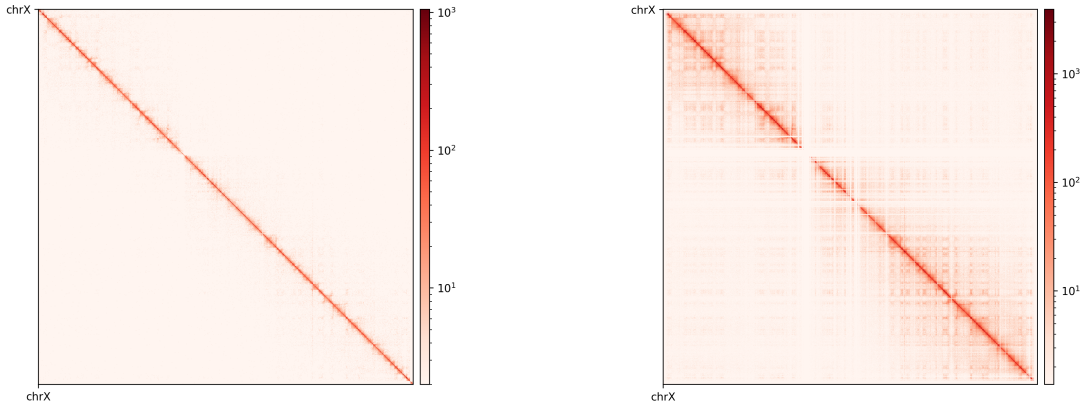


Figure 3.3: Comparison of HiCExplorer QC plots for all samples using different alignment tools.

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

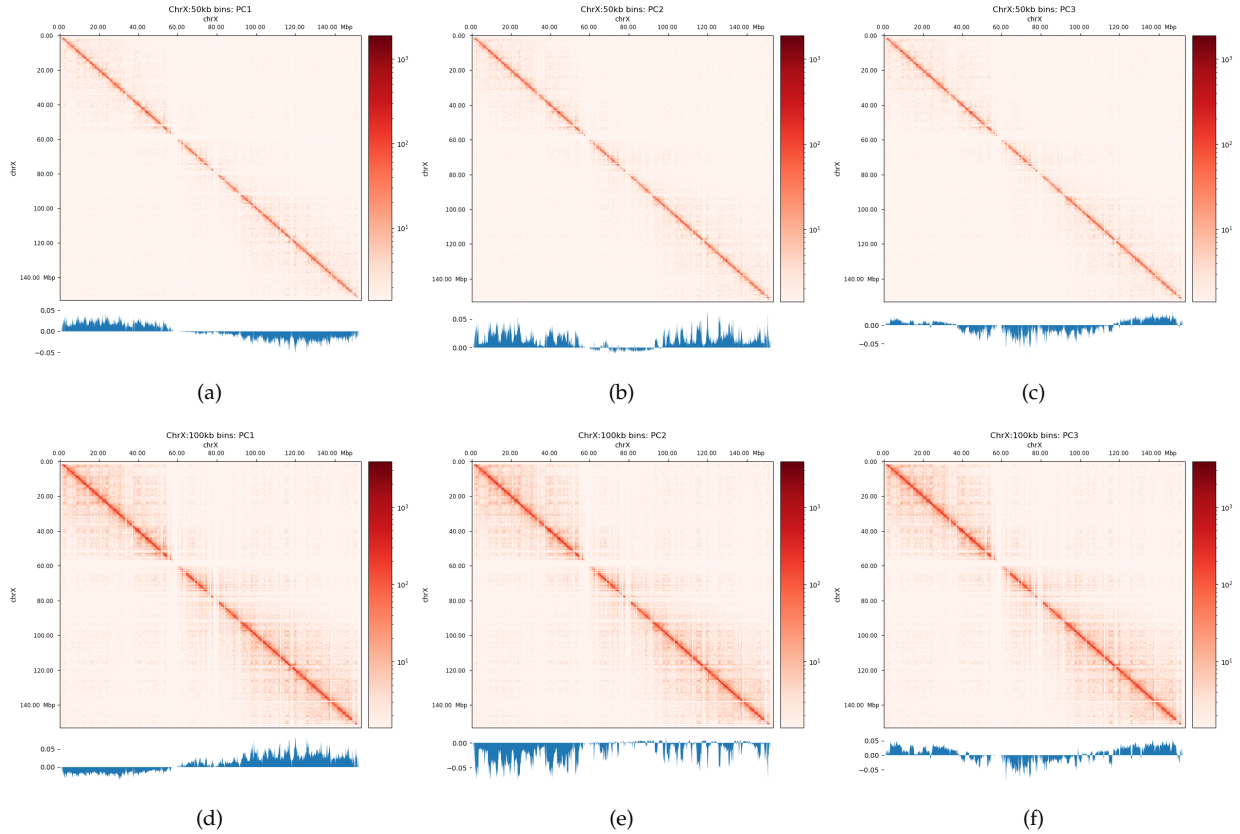
Figure 3.4: A comparison of interaction matrices before/after iterative correction (*HiCExplorer*).

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 by its change of sign. It seems the PCs capture variance from a bias that varies slowly and predictably along the chromosome. The first PC that is supposed to capture the compartments very suspiciously changes sign at the region of the centromere, a classic problem that could be solved by restricting the values from which the PC is

calculated along the chromosome. Unimpressed, I rationalize that the option `--extra-track` to provide a gene track or histone coverage should not affect this result 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.



(a) (b) (c)

(d) (e) (f)

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.

[ref if there is time, include a histogram of mapq values for pairs]

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].

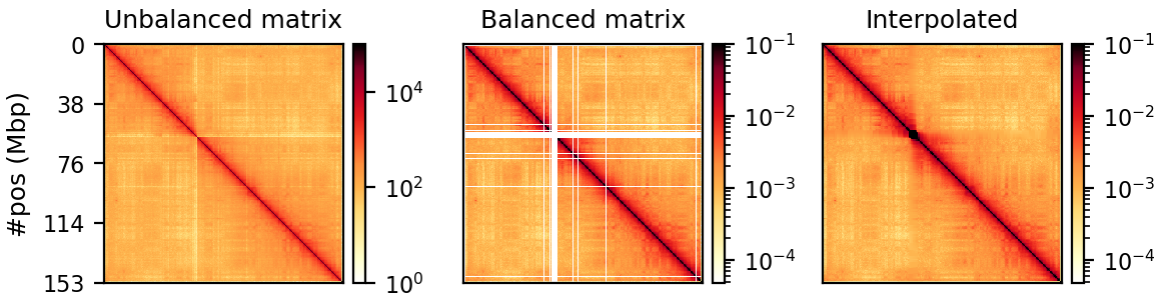


Figure 3.6: Raw, balanced, and interpolated chrX interaction matrix in 500kb resolution. The interpolation is done to make the matrix more visually appealing, but it is not necessary for the analysis.

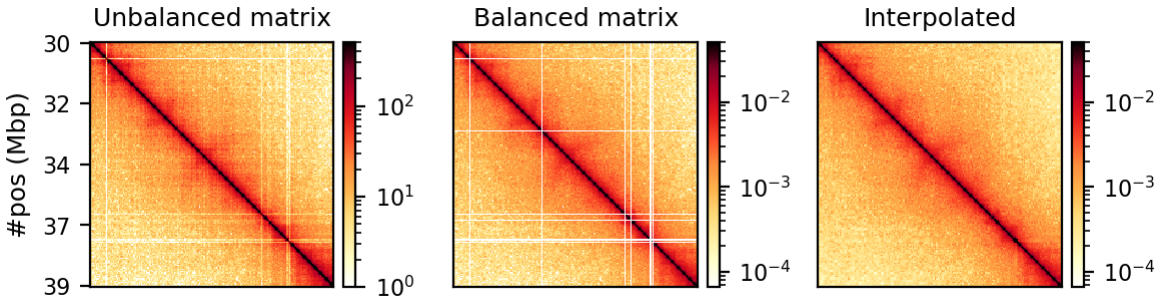


Figure 3.7: Raw, balanced, and interpolated chrX interaction matrix in 50kb resolution. The interpolation is done to make the matrix more visually appealing, but it is not necessary for the analysis.

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*.

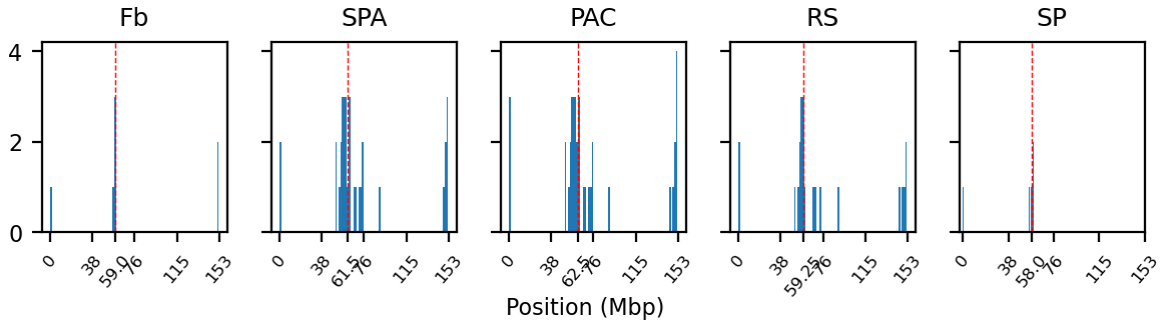
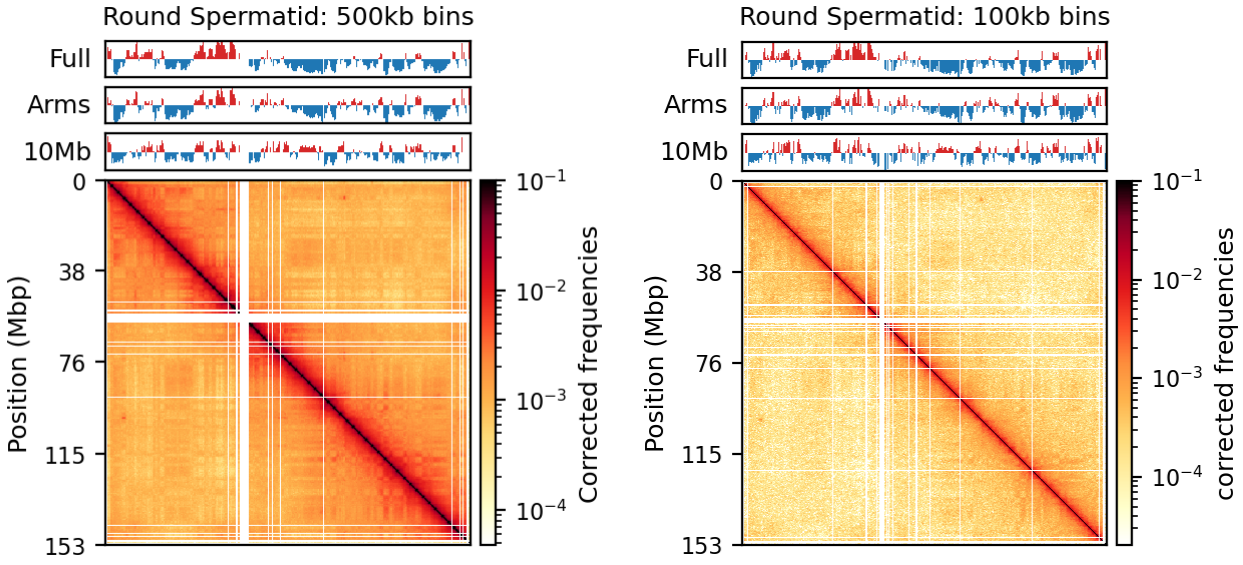


Figure 3.8: Histogram of NaN values in the E1 eigenvector for each cell type. Median position is marked with a red dashed line.

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 (colored red on the E1 tracks) 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).

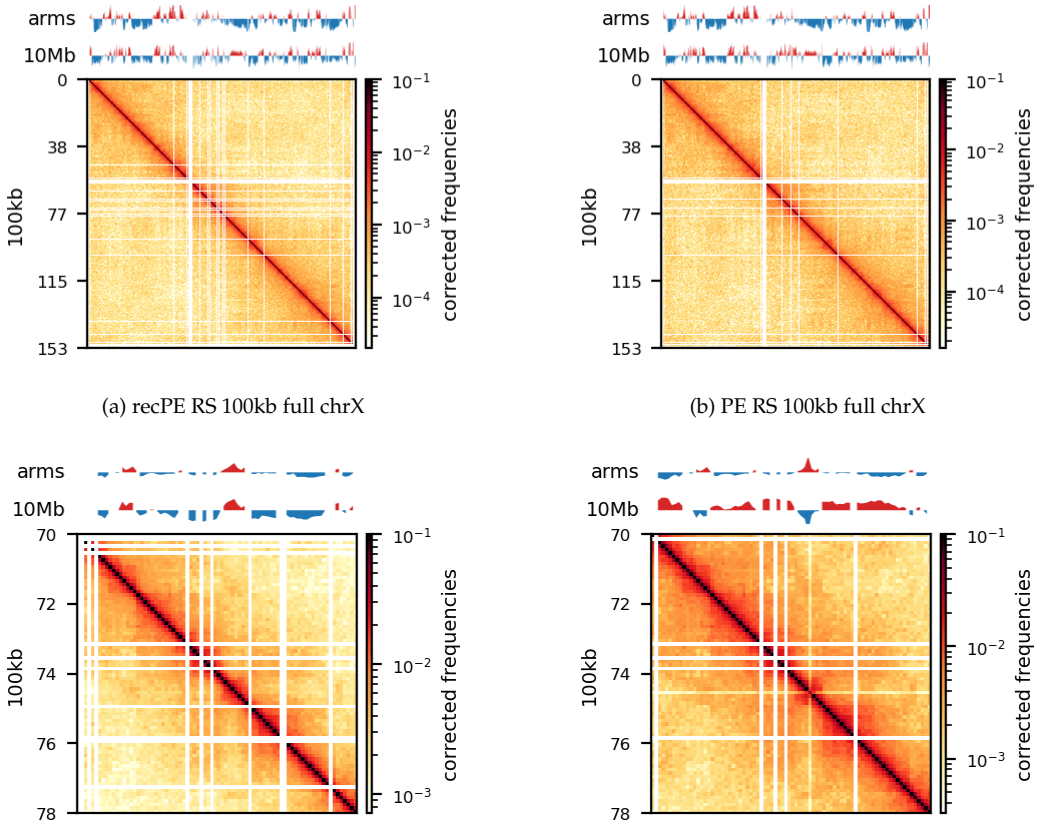
(b) E1 eigenvector values for merged round spermatid samples at 500kb resolution, as well as the interaction matrix. E1 was restricted to either Full-chromosome (top), Chromosome-arms (middle), or 10Mb windows (bottom).

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 (`07_various_plotting.ipynb`, with a new plotting function) should be same size as the following matrix plots.

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, we also keep *refined* compartments in the analysis.

Additionally, as I created coolers with two different sets of parsing parameters we will compare the resulting matrices and their compartments (Figure 3.10). As expected, we observe more empty bins in the Hi-C matrix, but otherwise, the interaction pattern is indestinguishable when going from the

initial run (PE) to the recommended parameters (redPE). The effect on the E1 is more noticable, where the absolute magnitude of the E1 values is generally smaller. There is, however, a small region that changes sign (from A to B) on the 10Mb-windowed ('refined') E1 track (Figure 3.10;c+d, zoomed-in region). This region is surrounded by added empty bins, which could mean that too many low quality pairs in PE were introducing bias and swapped the sign of E1. It is supported by the fact that the sign change *only* occured in *refined* E1, and that the sign after filtering weak pairs ($mapq < 30$) is consistent with the *arms* view. It supports my previous postulate that it is better to use a viewframe where with explicit molecular meaning than one of an arbitrary window size. That said, the `mapq` threshold should really be determined taking both coverage and resolution into account. For our purposes, and with the *arms* view, the mapping- and parsing parameters do not seem to be too sensitive.



(a) recPE RS 100kb full chrX

(b) PE RS 100kb full chrX

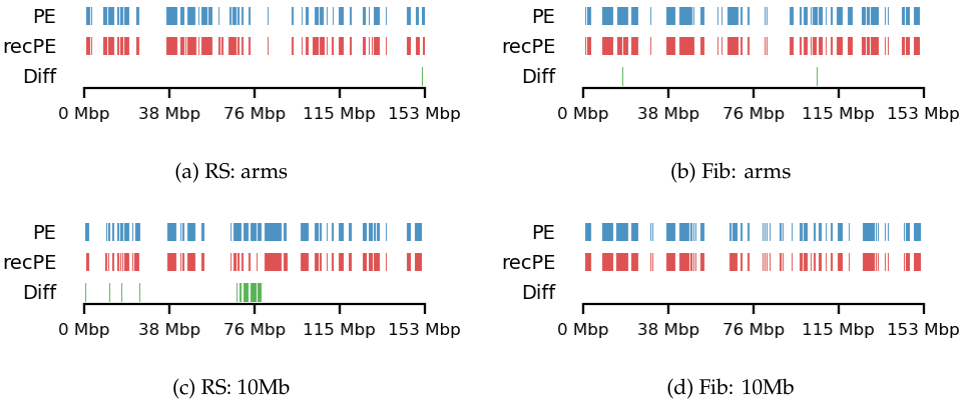
(c) recPE RS 100kb chrX:70Mb-78Mb

(d) PE RS 100kb chrX:70Mb-78Mb

Figure 3.10: Round Spermatid (RS) at 100kb, comparing the impact of parsing parameters

To emphasize the findings, the sets of A-compartments were compared between the two parsing runs, showing almost identical compartment calls. Additionally, the set difference was 8 bins between PE and recPE for round spermatid 100kb and 5 bins for fibroblast for *arms* viewframe (Figure 3.11; a+b, respectively). We observe a high number of differences around 76Mb for the

refined compartments (10Mb) of round spermatid, which is consistent with the sign-flip of E1 values discussed earlier. Anything else would be surprising, as it is the same data, but visualized in a different way.



(a) RS: arms

(b) Fib: arms

(c) RS: 10Mb

(d) Fib: 10Mb

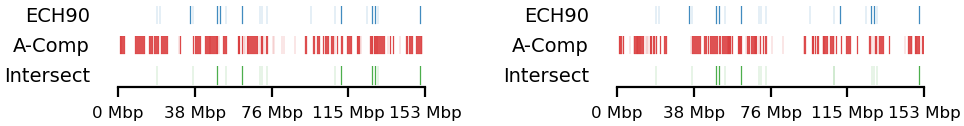
Figure 3.11: Round Spermatid (RS) and Fibroblast (Fb) at 100kb, comparing the impact of parsing parameters on A-compartment calling at different viewframes; *arms*, *10Mb*. PE: initial parse (masking complex walks); recPE: recommended parse (reporting the 5'most unique alignment of a complex walk).

The observed difference between the sets can for our data be attributed to chance, but we cannot draw general conclusions about the parameters in general. I argue that the quality and size of the Hi-C library will influence sensitive to parsing parameters. In that case, the most flexible approach is still to follow the recommendations from `cooler` to report more pairs as valid contacts, and then create coolers with different *mapq* filters if issues are encountered.

3.2.4 Compartment Edges (transition zones)

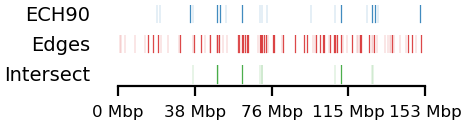
We compare how the ECH90 regions fit when queried on top of the A-compartments and equivalently for the edges, for fibroblasts and round spermatids at 100kb resolution. When queried against the edges in stead, the the total set size is reduced to less than 50%. Interestingly, some of the intersections between A-compartments and ECH90 remain, and new ones appear as we move to the outside edge of the compartment (Figure 3.12). This indicates that most, but not all, of the intersection between ECH90 regions and the A-compartments are within 100kb of the compartment edge, and additional overlap is gained if we define a transition zone on the outside of the edge as well. To visualize this (outside) edge enrichment, we find the set difference of the ECH-intersection to compartments and edges, respectively (Figure 3.13), thus removing all the 'inside' edges. We observe that in almost all of the of the regions of $ECH \cap Comp$ are accompanied by an edge also intersecting ECH ($ECH \cap Edge$), localized where the *Diff* track aligns (within 100kb) with both $CompInt$ and $EdgeInt$.

We apply both proximity test and Jaccard test, to see how well the results could occur by chance

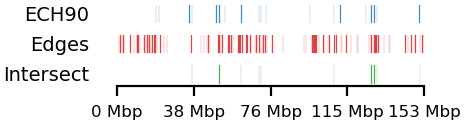


(a) Fibroblast A-compartments

(b) Round Spermatid A-compartments

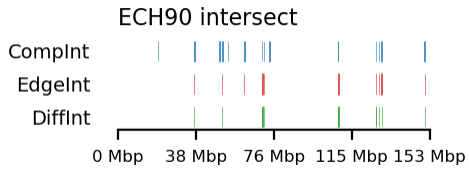


(c) Fibroblast edges

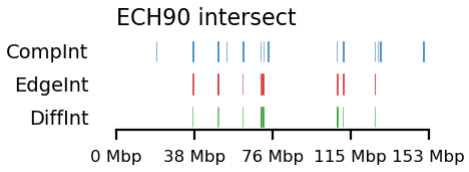


(d) Round Spermatid edges

Figure 3.12: Visual representation of the genomic intervals of ECH90, A-compartments (a+b), edges (c+d), and their intersections. Shown fibroblast (a+c) and round spermatid (b+d) at 100kb resolution and arms viewframe.



(a) Round Spermatid



(b) Fibroblast

Figure 3.13: Visual representation of the enrichment of edges in the intersection of ECH90 and A-compartments. Shown round spermatid (a) and fibroblast (b) at 100kb resolution and arms viewframe. Note that the edge-regions are too small to be distinguished visually from the compartment on the graph, making it look like they overlap, even though the difference is reported.

(Figure 3.14). For completeness, the tests are included for all cell types, but we only use 100kb resolution arms viewframe. We observe that both fibroblast and round spermatid have $p < 0.05$ for both tests, meaning the two cell type have both more intersection with ECH regions than expected by chance (Jaccard) *and* the non-overlapping intervals are more proximal to compartment edges than expected by chance (proximity test). I argue that a significant Jaccard statistic should be interpreted as a significant amount of overlap between the two sets, i.e. compartment edges and ECH90 regions, and the proximity test (when performed on the edges) gives us information about the potential of expanding or moving the transition window. That is, if the non-overlapping regions are *very* proximal, a larger (or shifted) window to only capture the 200kb region outside of the edge might be favourable.

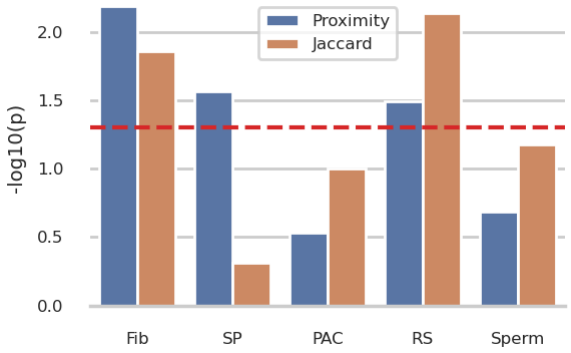
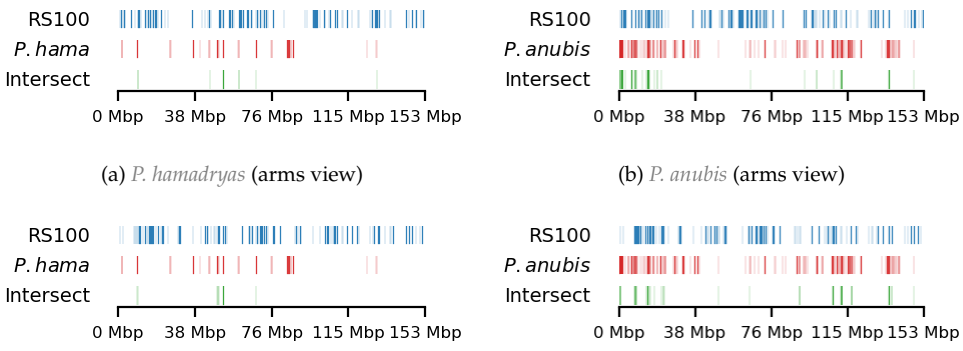


Figure 3.14: Proximity and Jaccard index p-values for ECH90 regions on compartment edges for all cell types at 100kb resolution at arms view $p = 0.05$ is marked as a red line.

3.3 Testing against hybrid incompatibility in baboons

Initially, the compartment edges of round spermatid at 100kb resolution (RS100) were plotted against the lifted coordinates from a *P.* anubis-hamadryas hybrid population, where either all the sampled individuals have *Papio anubis* ancestry or 95% of the sampled individuals have *Papio hamadryas* ancestry. Their respective intersections were plotted undeneath. We expect less intersection for *hamadryas* than for *anubis* as the total set size is much smaller. The compartment edges and *Papio anubis*-derived regions(Figure 3.15; b) seem to be highly enriched in the first 25 Mbp, and thus it has a high degree of intersection with the compartment edges. Interestingly, the ECH90 set is nearly empty in that region, making the finding outside the scope of this analysis, although it could be useful for determining the mechanism for selecting the *P.anubis* ancestral allele in the hybrid baboon population. The *P.hamadryas*-derived regions seem intersect the compartment edges more centered on the chromosome (Figure 3.15; a). The proximity test ruled out that the non-intersecting parts of the respective regions were this proximal by chance. However, the Jaccard test revealed that the intersection between the RS100 and both Hi-*P.hama* and Hi-*P.anu* can be explained by chance alone .



(a) *P. hamadryas* (arms view)

(b) *P. anubis* (arms view)

(c) *P. hamadryas* (10Mb view)

(d) *P. anubis* (10Mb view)

Figure 3.15: Comparing the A-compartments of round spermatid (RS) with regions in baboons from hybrid population where a) 95% of sampled individuals have *Papio hamadryas* ancestry or b) 100% of the samples have *Papio anubis* ancestry. The regions are extracted and lifted from panu4.0 reference to rhemac10, then collapsed to non-overlapping intervals with `genominterv`.

4 Discussion

Here is the discussion

Bibliography

Bicciato, Silvio, and Francesco Ferrari, eds. 2022. *Hi-C Data Analysis: Methods and Protocols*. Vol. 2301. Methods in Molecular Biology. New York, NY: Springer US. https://doi.org/10.1007/978-1-0716-1390-0.

Bravo Núñez, María Angélica, Nicole L. Nuckolls, and Sarah E. Zanders. 2018. "Genetic Villains: Killer Meiotic Drivers." *Trends in Genetics* 34 (6): 424–33. https://doi.org/10.1016/j.tig.2018.02.003.

Imakaev, Maxim, Geoffrey Fudenberg, Rachel Patton McCord, Natalia Naumova, Anton Goloborodko, Bryan R Lajoie, Job Dekker, and Leonid A Mirny. 2012. "Iterative Correction of Hi-C Data Reveals Hallmarks of Chromosome Organization." *Nature Methods* 9 (10): 999–1003. https://doi.org/10.1038/nmeth.2148.

Jaenike, John. 2001. "Sex Chromosome Meiotic Drive." *Annual Review of Ecology and Systematics* 32 (1): 25–49. https://doi.org/10.1146/annurev.ecolsys.32.081501.113958.

Lajoie, Bryan R., Job Dekker, and Noam Kaplan. 2015. "The Hitchhiker's Guide to Hi-C Analysis: Practical Guidelines." *Methods (San Diego, Calif.)* 72 (January): 65–75. https://doi.org/10.1016/j.ymeth.2014.10.031.

Lieberman-Aiden, Erez, Nynke L. Van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, et al. 2009. "Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome." *Science* 326 (5950): 289–93. https://doi.org/10.1126/science.1181369.

Munch, Kasper. 2024. "Munch-Group." https://munch-group.org/research.html.

"Open Chromosome Collective (Open2C)." n.d. Resource. *Open2C*. https://open2c.github.io/. Accessed November 13, 2024.

Skov, Laurits, Moisès Coll Macià, Elise Anne Lucotte, Maria Izabel Alves Cavassim, David Castellano, Mikkel Heide Schierup, and Kasper Munch. 2023. "Extraordinary Selection on the Human X Chromosome Associated with Archaic Admixture." *Cell Genomics* 3 (3): 100274. https://doi.org/10.1016/j.xgen.2023.100274.

Sørensen, Erik F., R. Alan Harris, Liye Zhang, Muthuswamy Raveendran, Lukas F. K. Kuderna, Jerilyn A. Walker, Jessica M. Storer, et al. 2023. "Genome-Wide Coancestry Reveals Details of Ancient and Recent Male-Driven Reticulation in Baboons." *Science* 380 (6648): eabn8153. https://doi.org/10.1126/science.abn8153.

Wang, Yao, Hanben Wang, Yu Zhang, Zhenhai Du, Wei Si, Suixing Fan, Dongdong Qin, et al. 2019. "Reprogramming of Meiotic Chromatin Architecture During Spermatogenesis." *Molecular Cell* 73 (3): 547–561.e6. https://doi.org/10.1016/j.molcel.2018.11.019.

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