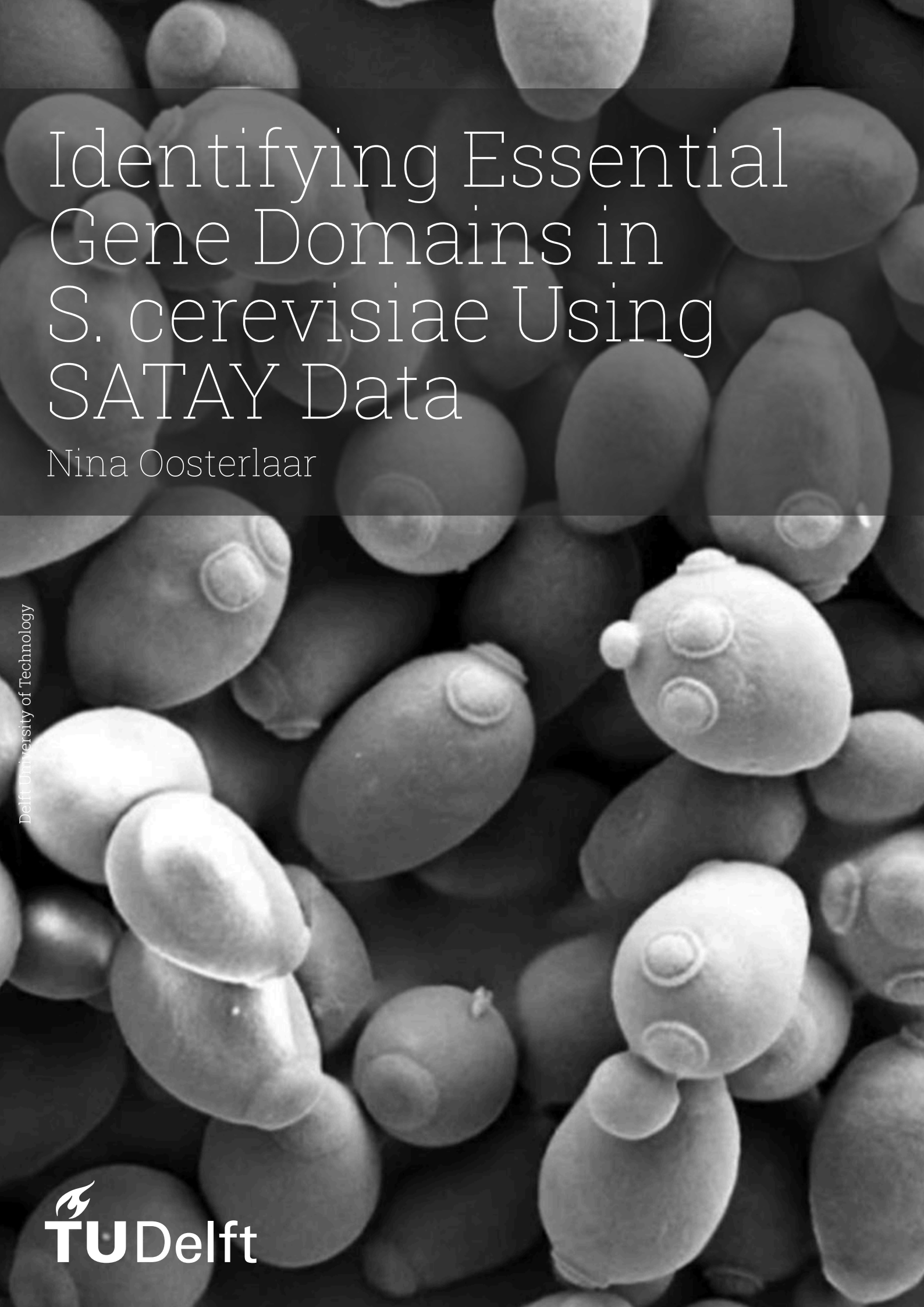


A scanning electron micrograph (SEM) of numerous S. cerevisiae (baker's yeast) cells. The cells are oval-shaped and appear in various stages of growth and budding. Some cells show distinct buds on their surface, while others are more rounded. The background is dark, and the cells are illuminated from the side, creating highlights and shadows that emphasize their three-dimensional structure.

Identifying Essential Gene Domains in *S. cerevisiae* Using SATAY Data

Nina Oosterlaar

Identifying Essential Gene Domains in *S. cerevisiae* Using SATAY Data

by

Nina Oosterlaar

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday May 18, 2026 at 15:00.

Project duration: September, 2025 – May, 2026

Thesis committee:
Dr. J. A. Baaijens
Dr. L. Laan
Dr. N. Šoštarić
Dr. S. Dumančić

Main supervisor
Co-supervisor
Committee member
Committee member

Abstract

Saccharomyces cerevisiae is a model organism used for studying fundamental eukaryotic processes and genome function. While essentiality studies provide important insight into gene function, biological pathways, and evolution, they often focus on entire genes, thereby ignoring variation in essentiality within and beyond gene boundaries. SATurated Transposon Analysis in Yeast (SATAY) is a Transposon Insertion Sequencing (TIS) technique that can measure essentiality across the entire genome of *S. cerevisiae*. However, the resulting data is noisy and sparse, and is affected by strong insertion biases. Most existing methods for analyzing TIS data either rely on known gene annotations or are optimized for bacterial datasets. Methods optimized for bacterial datasets are difficult to apply to SATAY data due to its higher sparsity and distinct insertion patterns. We therefore aim to develop a method tailored to SATAY data that does not rely on predefined annotations. In this study, we developed a change-point detection (CPD) algorithm to identify genomic regions with distinct levels of essentiality directly from SATAY data. We explored whether an autoencoder could improve the quality of SATAY data by denoising and imputing missing values. Although the autoencoder appeared to denoise the data, it did not perform meaningful imputation. Moreover, CPD applied to the AE output was outperformed by a CPD algorithm that modeled raw SATAY data as a zero-inflated negative binomial (ZINB) distribution. The ZINB-based CPD algorithm achieved better results by explicitly accounting for sparsity, overdispersion, and insertion biases. The regions produced by our CPD algorithm align with biological expectations, though they are oversegmented and remain affected by known biases in the data. Despite these limitations, our results show that CPD applied to SATAY data is a promising first step towards identifying essential genomic regions beyond predefined gene boundaries across the whole genome.

Contents

Abstract	i
1 Introduction	1
2 Related Works	4
2.1 Transposon Insertion Sequencing	4
2.2 Autoencoders for imputation	7
2.3 Change-Point Detection for data segmentation	9
3 Methods	11
3.1 Data Curation	11
3.2 Data Exploration	11
3.3 Zero-inflated Negative Binomial	11
3.4 Autoencoder	12
3.5 Synthetic Data Generation	15
3.6 Change-Point Detection	15
4 Results	19
4.1 Data Exploration	19
4.2 Autoencoder	20
4.3 Change-Point Detection applied on synthetic data	24
4.4 Change-point detection applied on SATAY data	26
5 Discussion	32
5.1 Summary of findings	32
5.2 Limitations and Future Work	33
5.3 Broader implications	34
6 Conclusion	36
7 Code and data availability	37
8 AI disclosure statement	37
References	38

1 Introduction

Saccharomyces cerevisiae, also commonly known as baker's yeast or budding yeast, is an important model organism in molecular biology, genetics, and evolution. *S. cerevisiae* is important because it shares genes and cellular processes with other eukaryotes. For example, more than 2000 genes of *S. cerevisiae* are shared with humans [1]. Additionally, *S. cerevisiae* is a simple organism with a small genome and fast reproduction, making genomic manipulation easily achievable [2]. These characteristics of yeast have led to important discoveries, including cell cycle regulation [3], aging mechanisms [4], and transcription [5].

Understanding the essentiality of different parts of the genome is a crucial part of eukaryotic research. We define essentiality as the necessity of a genomic region for an organism's survival and reproduction, and thus for its viability and growth. Determining which genes or genomic regions are essential gives insight into gene function [6], interactions [7], pathways [8], and evolution [9]. These insights can then be utilized to better understand human biology and diseases [10]. However, because genes are well-characterized and systematically annotated genomic units, essentiality is typically assessed at the gene level, leaving variation within genes less explored [11]. This overlooks the internal complexity of genes, which may contain domains with differing levels of essentiality, as well as essential functional elements outside annotated gene boundaries [12]. Achieving a higher-resolution understanding of essentiality, both within and beyond gene boundaries, may therefore provide a more complete view of the genome's functional landscape.

A variety of techniques are used to determine the essentiality of genomic regions in yeast. Common methods include traditional deletion knockouts [6], and CRISPR/Cas screening [13]. These methods inactivate a gene or a part of the genome to then study the consequences of this disruption. However, these techniques are low-throughput, meaning that only one or a few genes can be tested in each experiment. On the other hand, Synthetic Genetic Array analysis (SGA) [14] and bar-sequencing (bar-seq) [15] are high-throughput methods, meaning they can determine the essentiality of many genes in a single experiment. However, these methods are labor-intensive and expensive, limiting their accessibility and widespread use [16–18].

Transposon Insertion Sequencing (TIS) can overcome the limitations of these other techniques, as it can screen the whole genome in a single experiment [19, 20]. TIS introduces random transposon insertions into the genome, which disrupt the function of its surrounding region. If this region is essential for survival or growth, the cell will not be detected, while transposons inserted in non-essential regions will be detected. TIS can insert hundreds of thousands of transposons simultaneously across different genetic and environmental contexts, generating gene-disruption libraries that provide insight into the fitness landscape of an organism's genome (Fig. 1.1).

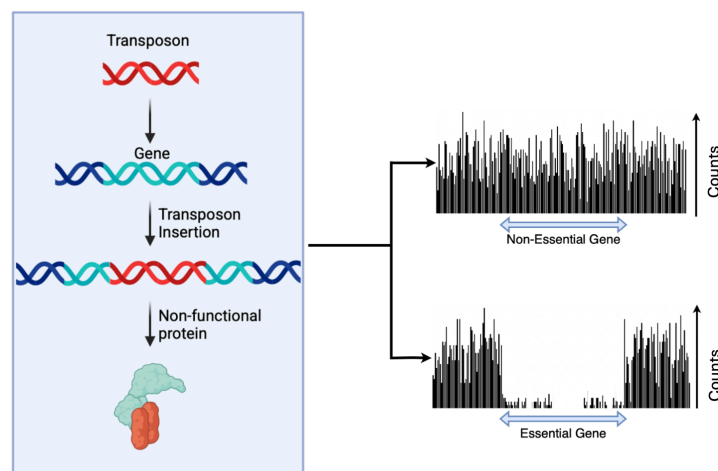


Figure 1.1: Illustration of the workflow of a TIS experiment. Transposons randomly insert into an organism's genome. When a transposon is inserted in a coding gene, a non-functional protein is created. All organisms are cultured and sequenced. Regions with many insertions are likely non-essential, while a region with few or no insertions suggests essentiality.

SATurated Transposon Analysis in Yeast (SATAY) is a TIS-derived method tailored for *S. cerevisiae* [16]. Like TIS, SATAY assesses gene fitness in various contexts by analyzing transposon insertion densities across the genome of *S. cerevisiae*. This enables comparison of the fitness landscape of *S. cerevisiae* across different genetic and environmental backgrounds and allows investigation of how this landscape changes under varying conditions. As such, SATAY may be used to study adaptive processes and track evolutionary change. So far, SATAY has been used to investigate gene function, genetic interactions, and cellular pathways in yeast under diverse genetic and environmental conditions [21–25]. These studies revealed essential genes, context-dependent regulators, and complex pathways, showing SATAY’s versatility.

SATAY does have limitations. The transposons are more likely to insert near centromeres and between nucleosomes, leading to significant biases in the data [16]. Furthermore, the insertion pattern is sparse, a unique characteristic of SATAY data compared to previous TIS methods, partly due to the larger genome size of yeast. This sparsity cannot be easily circumvented by improving the experimental approach [26]. If no insertion is observed, the genomic position may be essential, or simply untouched by transposon insertions. This uncertainty makes it challenging to analyze the data, particularly at higher resolutions.

Some methods for analyzing TIS data already exist. However, many of these methods were developed for bacterial TIS datasets, which differ from SATAY data in both sparsity and insertion patterns. For example, the transposons most commonly used in bacterial studies insert only at TA nucleotide sites, and this assumption is incorporated into several computational methods [27–33]. In contrast, SATAY insertions are not restricted to TA sites and are instead affected by other biases, such as centromere and nucleosome biases. Additionally, the methods either rely on known gene boundaries [28, 31, 34] or predefined essentiality states [27, 29]. We aim to develop a technique tailored to SATAY data that does not rely on predefined gene boundaries and can assign a continuous essentiality score to the resulting regions.

Thus, the aim of this project is to use SATAY data to identify sub-genic domains in the genome of *S. cerevisiae* and determine their essentiality. Specifically, we ask the following research question:

How do we identify essential genomic domains in *Saccharomyces cerevisiae* using SATAY data?

To address this research question, the following sub-questions are formulated:

1. How prevalent are the centromere and nucleosome biases in SATAY data?
2. Which autoencoder design can be utilized for denoising SATAY data?
3. How do we identify sub-genic regions and their corresponding essentiality levels?
4. How does denoised SATAY data change the resolution of essentiality predictions at the gene domain level compared to raw data?
5. How do essential domains inferred from SATAY data correspond to biological features, such as annotated protein domains?

To address these questions, we first analyze the data for known biases and their influence. Next, we propose a Denoising Autoencoder (DAE) architecture to improve SATAY data by denoising the data and imputing missing values. DAEs are commonly used for data imputation tasks in a wide range of applications due to their generative capabilities [35, 36]. Using raw data and AE reconstructions, we will identify genomic domains and their essentiality levels. We propose a window-based change-point detection (CPD) algorithm that detects regions with different transposon distributions and assigns an essentiality score to each region. We then compared our results to known annotations from the SGD database [11] to assess how our results align with existing biological knowledge. See figure 1.2 for an overview of the project.

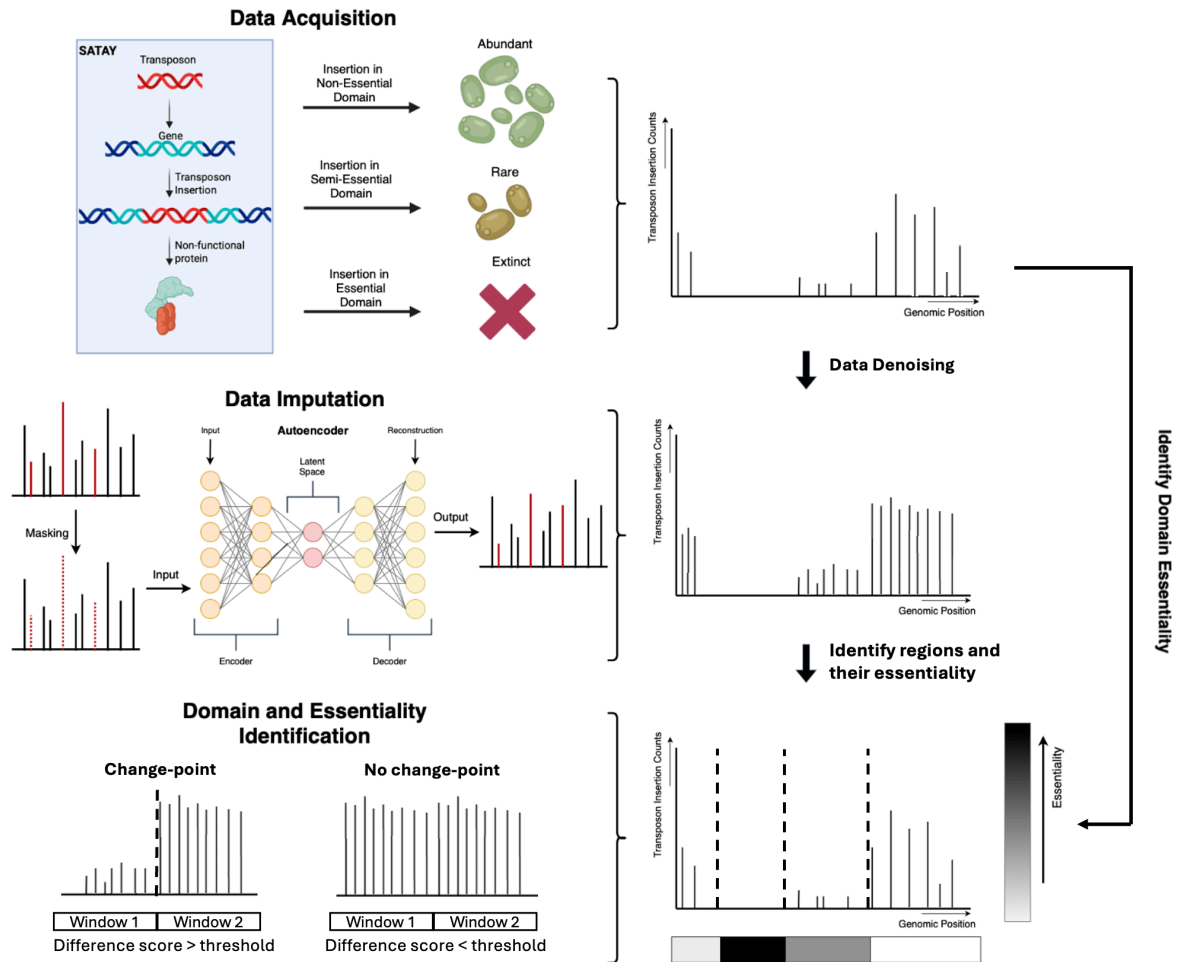


Figure 1.2: Overview of our proposed pipeline. **Data Acquisition:** After introducing random transposon insertions into the cells, the cells will be cultured. Cells with insertions in non-essential parts of the genome will grow normally and are thus abundantly present, resulting in high peaks in the transposon insertion profile. Cells with insertions in semi-essential regions of the genome will exhibit stunted growth and become less abundant, leading to smaller peaks in the transposon insertion profile. Cells with insertions in essential parts of the genome will go extinct and will not show any peaks in the transposon insertion profile. However, many positions will have no transposon insertions to begin with, resulting in a sparse dataset. **Data Denoising:** We will use SATAY data to train an Autoencoder model to denoise and impute the data. This process will be assisted by a masking algorithm that randomly removes data points, which the model should then learn to impute. After training the model, we will use it to denoise and impute SATAY data values, thereby overcoming data sparsity and noise. **Domain and Essentiality Identification:** Domains are identified using a window-based approach. A sliding window will traverse the genome, performing a likelihood-ratio test on the two windows. This will calculate a score that tests whether the two windows follow different underlying distributions. If the score exceeds a certain threshold, a change point is placed at this location.

2 Related Works

This chapter further elaborates on TIS and SATAY, explaining how these techniques work and highlighting their differences and limitations. It introduces existing computational methods to analyze TIS data and explains the need for a newly proposed algorithm. The chapter also introduces Autoencoders and their imputation capabilities, showcasing their potential to be used on SATAY data. Lastly, we introduce some common change-point detection concepts, which detect positions where the underlying distribution of the data shifts.

2.1. Transposon Insertion Sequencing

In 2009, four different TIS methods were introduced, all capable of measuring the essentiality of different genomic regions in a high-throughput manner. Those four methods are called transposon sequencing (Tn-Seq) [37], transposon-directed insertion site sequencing (TraDIS) [38], insertion sequencing (IN-Seq) [39], and high-throughput insertion tracking by deep sequencing (HITS) [40]. These four methods all follow these four general steps (Fig. 2.1):

1. A large library of cells containing a transposon insertion is created. This is done by introducing a plasmid containing a transposon fragment into the cell. The transposon can randomly insert itself into the host genome.
2. The cells are cultured and grow with a rate that depends on the essentiality of the genomic region with the transposon insertion. A cell with an insertion in a more essential region will grow more slowly than a cell with an insertion in a region that has no function. After culturing, the cells are pooled, and the DNA is extracted.
3. The extracted DNA is first fragmented, and adaptors are added to the DNA ends. Then, the DNA around the genome-transposon boundary is amplified by PCR using primers at the transposon end and an adaptor.
4. The amplified DNA fragments are sequenced, and the sequences surrounding the transposon sequence are mapped to a reference genome. From this mapping, the location of the transposon insertion can be determined.

The four methods mainly differ in the transposon and library preparation used. Tn-Seq and INSeq use a transposon containing a type IIS restriction site, allowing the adjacent genomic DNA to be cut at a fixed distance from the insertion site. The resulting sequence is usually sufficient to map the insertion to the reference genome. In contrast, HITS and TraDIS do not require a specific transposon and therefore rely on DNA fragmentation to recover transposon–genome junctions. This allows for the use of different types of transposons, but adds experimental steps and may introduce PCR bias due to variation in fragment size.

From these four methods, count data is generated, where each genomic position in the organism is associated with a certain transposon count. In theory, these counts represent the fitness effect of an insertion in that location, and a higher count would mean a higher reduction in fitness. When comparing two experiments with different genetic or environmental backgrounds, differences in transposon insertion patterns also indicate differences in fitness between the backgrounds. However, stochastic noise can result in the loss of mutants unrelated to their fitness. Additionally, technical noise, such as PCR bias, can result in count values that do not reflect fitness effects [41]. This should be taken into account in downstream analyses.

TIS has been applied to bacteria in various ways. It has been used to study antibiotic resistance [42], identify bacterial pathogens' virulence factors [43], analyze important genes in understudied bacterial species in a high-throughput manner [44], and understand how bacteria respond to different environments [45]. The method is particularly effective for bacteria due to their small, simple genomes. However, TIS methods for other species have also been developed, although each has its own caveats. Besides *S. cerevisiae*, there exist TIS workflows for *Schizosaccharomyces pombe* [46], another important yeast model system, as well as some archaeal species [47, 48], and even haploid human cells [49].

For more details on the different transposon techniques, their limitations, and their advances, we recommend reading the review paper written by Cain et al. [19].

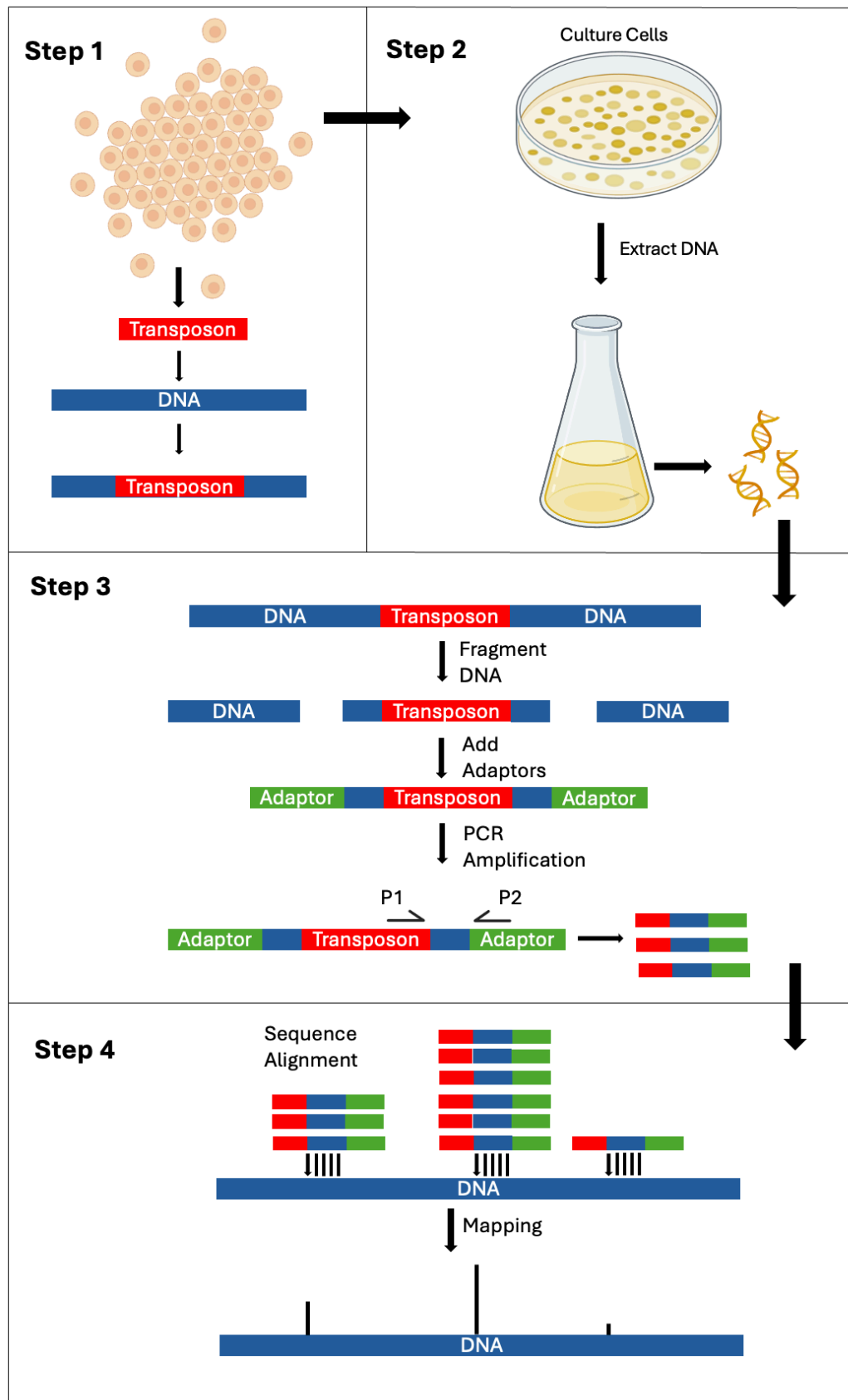


Figure 2.1: Overview of the transposon insertion sequencing workflow. **Step 1** A large library of cells is created that contains transposons (red) that insert themselves at a random location in the genome of the cell. **Step 2** Those cells are then cultured. Cells with disadvantageous insertions grow more slowly or not at all. After culturing, DNA is extracted from all the cells. The DNA of healthier cells will be present in a higher abundance than in unhealthy cells. **Step 3** After extracting all the DNA, the DNA is fragmented. Adaptors (green) are added to the fragments. Then the DNA is amplified with PCR. One primer (P1) will align the DNA from the transposon end, while the other primer (P2) will align the DNA from the adaptor, resulting in the amplification of genomic DNA next to the transposon. **Step 4** After sequencing, the genomic DNA is aligned to a reference genome, and from this, the location of the transposon insertion can be mapped to the genome. By counting reads at each position, the transposon count is determined.

2.1.1. Saturated Transposon Analysis in Yeast

SATAY generally follows the same approaches as other TIS methods, but then applied to *S. cerevisiae*. SATAY uses the Maize Ac/Ds transposase/transposon system, a well-studied system that can be easily applied in *S. cerevisiae*. Ds is a transposon element that is often engineered into miniDs, a minimal version of the Ds transposon. Ac is the transposase that recognizes the miniDs sequence and can excise and reinsert this transposon into the genome. Another advantage is that the Ac/Ds system has a minimal sequence preference and can thus insert itself anywhere in the genome [50]. MiniDs is inserted within the ADE2 gene, which is important for the adenine biosynthesis pathway. Ac expression is induced in cells grown in an adenine-lacking medium, leading to an evolutionary incentive to excise MiniDs and perform homology repair of the ADE2 gene, which the cells need to form colonies. MiniDs will then reinsert itself into the genome of *S. cerevisiae* with a probability of 60% [51]. Besides these changes, SATAY follows a generic TIS strategy, where the DNA is extracted, and the genome sequence surrounding the transposons is amplified, sequenced, and mapped. For more details about the methodology, we recommend reading the paper by Michel et al. [16].

S. cerevisiae is a more complex organism than bacteria. The genome of bacteria is generally smaller than 6 Mbp, while *S. cerevisiae* has a genome size of around 12 Mbp. Additionally, the genome of *S. cerevisiae* is organized in chromosomes, unlike bacteria. The plasmid used to introduce the transposon into the cell is located near the centromeres, resulting in a bias toward transposon insertions near the centromeres. Additionally, transposons are less likely to insert themselves into nucleosomal regions, where DNA is tightly packed to the nucleosomes and therefore difficult to access [16]. Together with the larger genome size, this makes it challenging to achieve a high transposon insertion saturation across the whole genome. Many positions in the genome remain devoid of transposon insertions and will not be represented in SATAY data.

2.1.2. Algorithms to infer essentiality from TIS

A simple and common technique for inferring essentiality from TIS data is to compare the expected number of transposon insertions with the observed number of transposon insertions. If the expected number is significantly higher than the actual number of insertions, a gene is assumed to be essential and vice versa. ESSENTIALS is a software that implements this approach by modeling insertion counts as a negative binomial model and testing for the significant absence of insertions [34]. However, TIS data can exhibit insertion biases, leading to inconsistent estimates of expected counts. Kingma et al. [26] implemented an algorithm specifically tailored for SATAY data. The main difference is that they incorporated known SATAY biases into the estimation of the expected number of insertions and used that to calculate the mean transposon count for a gene. These techniques can provide reliable estimates of gene essentiality. The approach made by Kingma et al. can even provide the user with a specific essentiality score. However, these approaches are annotation-dependent, meaning they only work on known genes and cannot be used to discover new regions of interest. Additionally, some essential genes have only one essential domain, while other regions of the gene are non-essential. These approaches consider all insertions across a gene and can thus incorrectly classify an essential gene as non-essential when there are sufficient insertions in its non-essential region. To accurately classify these genes, these domains need to be known [52].

This last issue is solved by another approach taken by Griffin et al. [53] and DeJesus et al. [28]. Their algorithms identify the largest insertion-free gap within a gene. If the gap of that size is statistically unlikely to occur, they conclude the gene in question must be essential. This approach assumes that not all parts of an essential gene are equally essential, and a gene can be classified as essential if at least a significant portion of the gene is identified as essential. However, these methods remain annotation-dependent and do not yet identify the essentiality of genomic domains within and beyond genes.

Methods based on Hidden Markov Models, such as TRANSIT [27, 32] and ARTIST [29], can relax gene boundaries and identify domains within and beyond genes. Both methods define emission probabilities, which are the probabilities of an observation belonging to a certain essentiality state, and transition probabilities, which are the probabilities of transitioning from one state to another. The Viterbi algorithm can be used to compute the most probable state sequence for a given set of read counts, based on the emission and transition probabilities. The primary drawback of this method is that the states are categorical and must be predefined. It is thus difficult to determine essentiality scores across domains, limiting you to a fixed number of states. Additionally, the current implementations require high saturation levels, preferably above 50%, which is currently unachievable for SATAY data.

2.2. Autoencoders for imputation

The most common deep learning architecture for imputation is the autoencoder (AE), due to its noise and dimensionality reduction capabilities [36]. This section explains how AE architectures and their variants work and highlights examples of their use for biological data imputation.

2.2.1. Autoencoders

AEs are deep learning models tailored towards feature extraction, anomaly detection, and denoising. Deep learning models utilize multi-layer neural networks, where each layer processes inputs and passes them to the next via mathematical functions. The network is trained on large datasets by minimizing a loss function and adjusting connection weights. AEs are used for unsupervised learning, meaning that the model does not rely on labeled outputs (i.e., there are no corresponding labels provided during training). Instead, the AE learns patterns and structures from the input data, capturing the underlying distributions. AEs consist of an encoder that maps inputs to a low-dimensional latent space to reduce noise, while the decoder reconstructs the input (Fig. 2.2A). An AE is typically trained through a reconstruction loss that ensures the model produces an output similar to its input. When an AE is used for imputing values, missing values are sometimes excluded from the reconstruction loss so that the model is not penalized for imputing entries with unknown true values [36].

2.2.2. Variational Autoencoders

A Variational Autoencoder (VAE) is a probabilistic version of the AE. A VAE architecture maps inputs to a distribution, typically a multivariate Gaussian, rather than fixed points [54] (Fig. 2.2B). From this distribution, new points can be generated, making a VAE a generative model. To ensure that the latent space stays continuous and smooth, a VAE introduces an additional regularization loss: the Kullback-Leibler (KL) divergence term, which constrains the posterior to stay close to a standard normal prior. These additions enable VAEs to capture data uncertainty and regularize the model, thereby preventing overfitting on the training data. Overfitting occurs when a model becomes overly reliant on the training data, leading to poor performance on unseen data. Because of these advantages and a VAE's generative capabilities, a VAE is often chosen for tasks like data imputation. However, a VAE requires more training data than an AE to achieve high reconstruction accuracy [55].

2.2.3. Denoising Autoencoders

A Denoising Autoencoder (DAE) is a variant of the autoencoder that employs a denoising training principle: the model is presented with corrupted input but trained to reconstruct the original, clean data [56]. The model is trained on an input $\tilde{x}$, which is equal to x with some stochastic noise added. The DAE tries to reconstruct the original output x . By applying this training principle, DAEs become stable under noise and invariant to small perturbations, resulting in more robust and stable representations. When the goal is imputing missing data, a common approach is to randomly set a subset of input values to 0 and train the model to reconstruct those points again [36]. This forces the model to perform imputation on the input.

2.2.4. Applications in Genomics

AEs are commonly used to improve single-cell sequencing (sc-seq) data. Sc-seq is a high-resolution technology that can measure genomic features in individual cells. This reveals cell-to-cell variation and rare cell types that would go unnoticed in bulk sequencing. However, sc-seq suffers from large stochastic noise, leading to an excess of zero values that are difficult to distinguish between technical and biological zeroes [57], similar to SATAY data.

DCA [58] and scVI [59] are two methods that utilize an AE architecture to improve single-cell RNA-sequencing (scRNA-seq) data. Specifically, DCA employs a traditional AE architecture, whereas scVI utilizes a VAE architecture. They both model the output as Zero-Inflated Negative Binomials (ZINB), which is appropriate since scRNA-seq can be described as count data. Both the models' outputs and their latent representations can be used to improve downstream analyses of scRNA-seq data.

AE techniques have also been extended to other sc-seq techniques, such as single-cell Hi-c (scHi-c) and single-cell ATAC-sequencing (scATAC-seq). Zheng, Shen, and Keleş [60] adapted scVI to scVI-3D that denoises and imputes scHi-c data. SAILER [61] and SCALE [62] are both VAE models that learn representations of scATAC-seq data for imputation and downstream analyses.

Together, these methods demonstrate that AEs are a suitable choice for imputing biological data. This motivates the exploration of similar architectures for improving SATAY data.

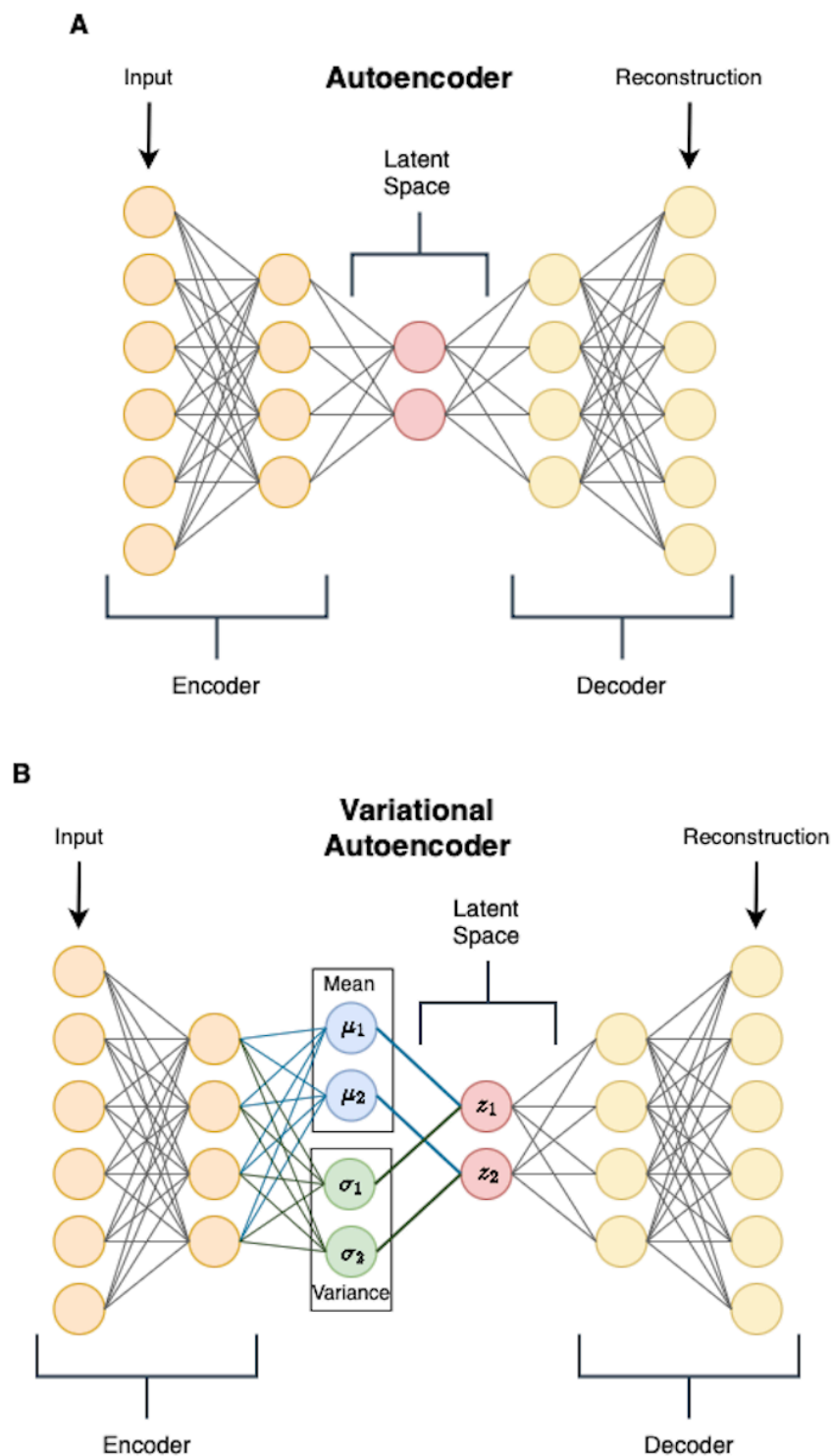


Figure 2.2: Autoencoder and Variational Autoencoder architecture. **A** Schematic of an Autoencoder (AE). The left part shows the encoder, which compresses the data into a latent representation. The right part represents the decoder, which decompresses the data into a denoised representation.

B Schematic of a Variational Autoencoder (VAE). The VAE differs from the AE because it maps the inputs to a distribution, often a multivariate Gaussian. This makes VAE a probabilistic technique.

2.3. Change-Point Detection for data segmentation

Signal processing is a field that focuses on representing, transforming, and interpreting noisy data. A classical problem in signal processing is change-point detection (CPD), which aims to detect points where the signal's underlying properties change. Offline CPD, which assumes the complete signal is available before analysis, is also often called signal segmentation. Offline CPD has been applied to a variety of biological problems. For example, it has been used to detect copy-number variation [63, 64], patterns in DNA sequences [65], and gene boundaries using sequencing data [66, 67].

2.3.1. Problem Description

A CPD problem can be described as follows: You observe a sequence of elements

$$x_{1:n} = (x_1, \dots, x_n) \quad (2.1)$$

where $x_i \in \mathbb{R}^d$ and $d \geq 1$. Assume that this sequence is generated by a piecewise-stationary process, and you want to infer K unknown breakpoints

$$1 < t_1 < t_2 < \dots < t_K < n, \quad (2.2)$$

where t_i is described as an index of sequence $x_{1:n}$, $t_0 = 0$, and $t_{K+1} = n$ are implicitly defined. These change points segment the sequence into $K + 1$ partitions within which the distribution is constant. In practice, “constant distribution” often means that the parameters of a chosen model are constant within each segment, for example, a constant mean and/or a constant variance [68].

2.3.2. Cost-based methods

One family of solutions to the CPD problem consists of cost-based optimization methods. As described by Truong, Oudre, and Vayatis [69], these methods can be described with a criterion function that estimates the change-point set $T = (t_1, \dots, t_K)$ by minimizing a penalized segmentation criterion:

$$\min_T V(T, x) = \sum_{k=0}^K c(x_{t_k, t_{k+1}}) + \text{pen}(T) \quad (2.3)$$

The term $c(\cdot)$ is a within-segment cost function that measures the lack-of-fit of the subsequence sub-signal $x_{t_k, t_{k+1}}$ which goes from change point k up until change point $k + 1$. The best segmentation chooses a set of change points T that minimizes this cost function, while simultaneously penalizing the number of change points to reduce the complexity and oversegmentation with a function $\text{pen}(T)$.

To build a CPD detection method, both the cost function and the penalty on the number of change points need to be defined, along with a search algorithm to solve these problems.

Cost function

The most common CPD methods use maximum likelihood estimation to estimate the goodness-of-fit of the sub-signals. These models can be parametric and non-parametric. When using a parametric method, a distribution needs to be chosen that best describes the underlying process. The cost function of a sub-signal $x_{a,b}$ can then be described with the following equation:

$$c(x_{a,b}) = - \sum_{t=a}^b \log f(x_t | \hat{\theta}) \quad (2.4)$$

where $f(\cdot | \theta)$ is the chosen parametric model, and $\hat{\theta}$ is the set of parameters that maximizes the log-likelihood of this segment. Traditionally, Gaussian distributions have been used to detect mean-shifts in data [68], including RNA-seq data [70]. However, this method has also been applied to a Poisson and a Negative Binomial distribution, which model count data [71]. Specifically, these count models have been used to perform genome annotation using RNA-seq data [72]. For more details on other cost functions, specifically non-parametric cost functions, check out the review by Truong, Oudre, and Vayatis [69].

Penalty

When the number of change points is unknown, a penalty function is needed to limit the number of change points. The most common penalty function is linear:

$$\text{pen}(T) = \lambda|T| \quad (2.5)$$

where T is the set of change points, and λ the weight factor of the penalization. Small λ leads to more change points, potentially causing oversegmentation, while a high λ limits the number of change points, potentially missing out on true change points [73].

To calibrate the weight factor λ the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) are often consulted. These criteria measure the goodness-of-fit of a model, while penalizing model complexity. In the context of an univariate Gaussian with fixed variance σ^2 and a piecewise constant mean, the BIC penalization is given by:

$$\text{pen}_{BIC}(T) = \sigma^2 \log N|T| \quad (2.6)$$

The AIC penalization is given by:

$$\text{pen}_{AIC} = \sigma^2|T| \quad (2.7)$$

where σ is the standard deviation, N the number of samples, and T the set of change points.

An alternative penalization is called fused lasso, or total variation regularization [74]. Fused lasso has been used for CPD of copy-number variation [64, 75]. This penalization penalizes differences between neighboring parameters, which promotes piecewise-constant parameter distributions. The fused lasso is defined as follows:

$$\text{pen}_{fused_lasso} = \lambda \sum_{t=2}^N |\theta_t - \theta_{t-1}| \quad (2.8)$$

where N is equal to the number of samples, θ_t equal to the parameter value at position t , and λ the weight factor of the fused lasso.

Search Method

The search method describes the algorithm used to optimize the cost function and the penalty. There are two categories of search methods: optimal and approximate. The most common optimal search method for an unknown number of change points is called Pruned Exact Linear Time (Pelt) [73]. Pelt considers each sample in the sequence and prunes candidate change-point locations that are guaranteed not to be optimal for any future samples. Specifically, it checks the following inequality:

$$F(\tau) + c(x_{\tau+1,t}) + \lambda \geq F(t) \quad (2.9)$$

where τ is the candidate location of the most recent change point, $F(\tau)$ is the minimal penalized cost for segmenting the data up to τ , $c(x_{\tau+1,t})$ the cost of the segmentation between τ and t , λ the penalty of adding an extra change point and $F(t)$ the cost of segmenting the data up to point t . In short, the formula checks whether using τ as the most recent change point, followed by a segment from $\tau + 1$ to t , would result in a lower penalized cost than the current best segmentation up to t . If this is not the case, τ can be discarded as a candidate change point. By discarding suboptimal change points, Pelt finds an exact solution in near-linear time.

Some examples of approximate search methods are window sliding, binary segmentation, and bottom-up segmentation. Window sliding computes the difference between the parameters of two adjacent windows that slide along the sequence. In practice, this problem often reduces to a hypothesis test: the two windows are either sampled from the same distribution or from different distributions, resulting in a change point [76]. Binary segmentation is a greedy algorithm that searches for the change point that minimizes the sum of costs. This binary segmentation is repeated until a stopping criterion is met [77]. Lastly, bottom-up segmentation starts by splitting the sequence into many small sub-signals. In each iteration, all change points are ranked by cost, and the segment associated with the lowest-cost change point is merged. This is repeated until a stopping criterion is met [78].

3 Methods

3.1. Data Curation

We obtained 35 SATAY datasets generated by the Laan lab at the Bionanoscience department of the TU Delft. A description of these datasets can be found in Appendix A. In general, only between 3%-10% of the positions have a transposon count associated with them. The experiments have been performed on the W303 strain of *S. cerevisiae*. The sequences have been aligned to the S288C strain. To calculate the saturation of the datasets, the number of non-zero counts in each dataset is divided by the total number of base pairs of the S288C strain.

The 35 datasets are created from 8 experiments, each producing multiple datasets from cells of the same genetic background. Some datasets are technical replicates, in which only the sequencing step was repeated, whereas most are biological replicates, in which the entire experiment was repeated. To increase data saturation, datasets from the same strain were combined. We combined datasets by averaging the non-zero counts at each position. Zero counts were excluded when averaging because they mostly represent technical zeros caused by the absence of an initial transposon insertion, and including them would artificially lower the averaged counts. Counts were averaged rather than summed to avoid giving disproportionate weight to positions with non-zero counts in multiple replicates. The resulting dataset saturation is shown in Appendix B. For this project, we will work with the combined datasets unless otherwise noted.

3.2. Data Exploration

The centromere bias was assessed by calculating the frequency of non-zero insertion counts and their mean value as a function of distance from the centromere midpoint. Centromere coordinates for the S288C reference genome have been retrieved from the Saccharomyces Genome Database (SGD) [11]. Transposon insertion rates were aggregated into bins of 10,000 bp to reduce local noise. Similarly, the nucleosome bias was assessed by calculating the frequency of non-zero insertion counts and their mean value as a function of distance from the closest nucleosome dyad (the midpoint of a nucleosome). Genome-wide nucleosome dyad coordinates were obtained from Brogaard et al. [79]. This analysis was applied to uncombined datasets, and the results were averaged across all datasets.

3.3. Zero-inflated Negative Binomial

For both the AE architecture and the CPD algorithm, we assume that SATAY data follows a zero-inflated negative binomial distribution (ZINB). A ZINB distribution consists of a negative binomial (NB) component, which models overdispersed count data where the variance exceeds the mean, and a zero-inflation component, which can model an excess of zeros. Previous work has shown that TIS data is well described by ZINB models [80]. The NB distribution is appropriate, since the variance exceeds the mean in TIS data. Additionally, many genomic positions lack transposon insertions. This results in more zeros than expected under an NB distribution, and thus requires the zero-inflation property of the ZINB distribution to be accurately modeled.

A ZINB distribution is a two-component mixture model. A zero count can either come from the zero-inflation component with a probability π , or come from the NB distribution with probability $(1 - \pi)$. Non-zero counts always come from the NB distribution. The ZINB distribution is defined as:

$$\text{ZINB}(y \mid \pi, \theta, \mu) = \begin{cases} \pi + (1 - \pi) \text{NB}(0 \mid \theta, \mu), & y = 0 \\ (1 - \pi) \text{NB}(y \mid \theta, \mu), & y > 0, \end{cases} \quad (3.1)$$

where y is the insertion count, π the zero-inflation probability, μ the mean count of the NB, and θ the dispersion parameter of the NB. $\text{NB}(y \mid \theta, \mu)$ models the Negative Binomial. The probability mass function of the NB is defined as:

$$P(Y = y \mid \mu, \theta) = \frac{\Gamma(y + \theta)}{\Gamma(\theta) \Gamma(y + 1)} \left(\frac{\theta}{\theta + \mu} \right)^\theta \left(\frac{\mu}{\theta + \mu} \right)^y, \quad (3.2)$$

where $\Gamma(\cdot)$ represents the Gamma function defined as: $\Gamma(n) = (n - 1)!$. The variance of the NB distribution is given by:

$$\text{Var}(Y) = \mu + \frac{1}{\theta}\mu^2 \quad (3.3)$$

For both the AE and CPD algorithms, model fit is evaluated using the ZINB log-likelihood, which is computed from the probability mass function of the ZINB distribution:

$$\log p(y \mid \mu, \theta, \pi) = \begin{cases} \log\left(\pi + (1 - \pi) \left(\frac{\theta}{\theta + \mu}\right)^\theta\right), & y = 0, \\ \log(1 - \pi) + \log \Gamma(y + \theta) - \log \Gamma(\theta) - \log \Gamma(y + 1) + \theta \log\left(\frac{\theta}{\theta + \mu}\right) + y \log\left(\frac{\mu}{\theta + \mu}\right), & y > 0 \end{cases} \quad (3.4)$$

3.4. Autoencoder

In this section, we describe the development of the AE model. First, we outline the preprocessing steps applied to the input data. Next, we describe the loss functions used in the model, followed by the model architecture. Finally, we explain how model performance was evaluated.

3.4.1. Data Preprocessing

SATAY data is used as input for the AE. Each strain was obtained from different experiments, resulting in datasets with varying library sizes. To make insertion counts comparable across datasets, we applied counts-per-million (CPM) normalization followed by a log transformation, similar to preprocessing commonly used for scRNA-seq data. For each dataset, the counts are divided by the total number of insertions computed across all chromosomes, multiplied by one million to obtain counts per million (CPM), and then subsequently transformed as $y = \log(1 + \text{CPM})$. To allow conversion back to raw counts, we also retain a dataset-specific scaling factor. The datasets are then split into a training, validation, and test set, based on chromosomes. For each position, the distance to the centromere and to the nearest nucleosome can be included as a feature. A `StandardScaler` from the `scikit-learn` was fitted on the training data of each feature to standardize the data and used on the validation and test data. To reduce sparsity, the data is either binned or smoothed using a moving average. The data is then divided into sequences of length 2000 using a sliding window, each corresponding to a single data sample. See Figure 3.1 for an overview of the preprocessing pipeline, and Appendix C for an overview of the corresponding hyperparameters.

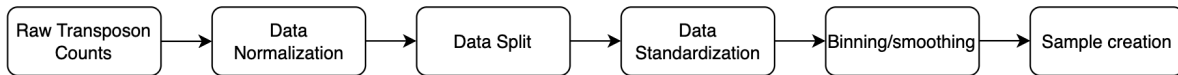


Figure 3.1: Overview of the preprocessing pipeline. First, the transposon counts are normalized using CPM normalization and log-transformation. Then the dataset is split. The included features are standardized using the training set. The data is either smoothed using a moving average or binned. Lastly, samples are created using a sliding window of size 2000.

3.4.2. Loss functions

Inspired by Lopez et al. [59], who used a ZINB distribution to model scRNA-seq data, we implemented a ZINB loss function. This approach allows the model to predict a mean μ , a dispersion parameter θ , and a zero-inflation parameter π for each position. In practice, π can be used as a probability of a zero count at each position. The loss function computes the negative log-likelihood of the observed raw counts under a Zero-Inflated Negative Binomial distribution and is defined as:

$$\mathcal{L}_{\text{ZINB}} = - \sum_{i=1}^N \log \text{ZINB}(y_i \mid \mu_i, \theta_i, \pi_i). \quad (3.5)$$

where N is the sequence length, y_i the raw count at position i , μ_i the predicted mean, θ_i the predicted dispersion, and π_i the predicted zero probability at position i .

In principle, the ZINB loss already encourages accurate reconstruction of masked values, since masked positions remain part of the raw counts used to compute the likelihood. Nevertheless, we also tested an additional loss function that focuses on correctly reconstructing masked values to emphasize

the model's imputation performance. Let $M = \{i \mid \text{mask}_i = 1\}$ denote the set of masked indices and let τ be the threshold applied to π . We define the reconstruction for each $i \in M$ as

$$\hat{x}_i = \mu_i \mathbb{I}(\pi_i < \tau),$$

where $\mathbb{I}(\cdot)$ is the indicator function. The masked reconstruction loss is then

$$\mathcal{L} = \frac{1}{|M|} \sum_{i \in M} |x_i - \mu_i \mathbb{I}(\pi_i < \tau)|. \quad (3.6)$$

The total loss is then defined as follows:

$$\mathcal{L} = - \sum_{i=1}^N \log \text{ZINB}(y_i \mid \mu_i, \theta_i, \pi_i) + \beta \frac{1}{|M|} \sum_{i \in M} |x_i - \mu_i \mathbb{I}(\pi_i < \tau)| \quad (3.7)$$

where β controls the weight of the masked reconstruction term.

3.4.3. Model Architecture

We implemented both a denoising AE and VAE to reconstruct and impute SATAY data. Before the input is passed to the network, we randomly mask a fixed percentage of non-zero values of the data. The encoder and decoder are implemented as a fully connected neural network, with a symmetrical architecture. Optionally, a one-dimensional convolutional layer can be applied before the encoder to capture local genomic patterns. Each hidden layer is followed by a ReLU activation function. The final layer of the decoder predicts a ZINB distribution for each position. It predicts a mean count μ and dispersion parameter θ through an exponential layer. It predicts a zero probability π using a sigmoid layer. An overview of the ZINBAE architecture is presented in Fig. 3.2.

We evaluated both the AE and VAE on a small dataset of 20 samples. Since we trained and tested it on the same small dataset, we expected the model to reconstruct the data with high accuracy. This was achieved by the AE architecture. In contrast, the VAE struggled to accurately reconstruct the input, likely because the stochastic latent representation introduced additional variability on top of the ZINB output distribution. We therefore chose to continue with the AE architecture.

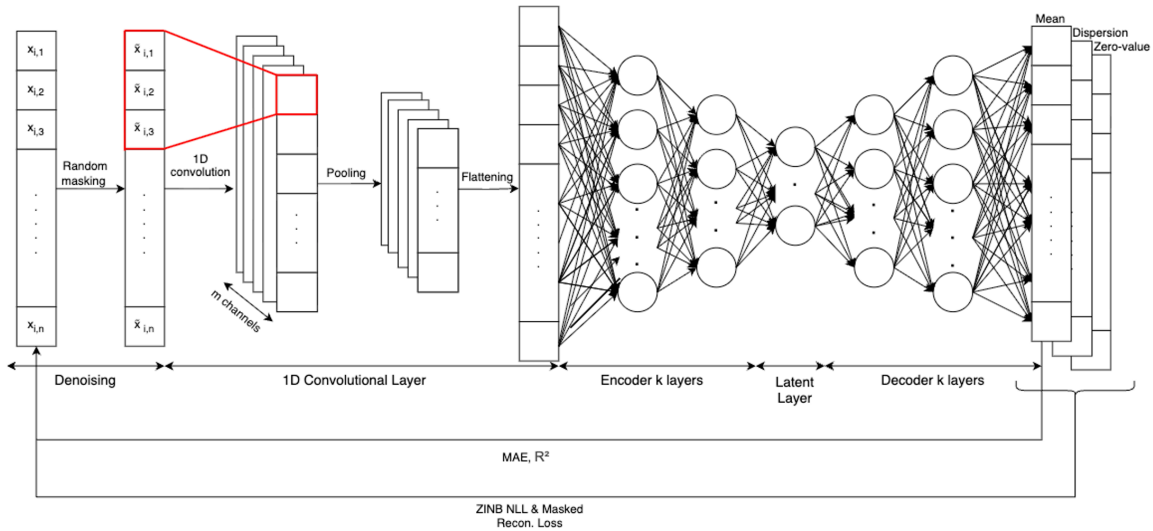


Figure 3.2: Denoising ZINBAE architecture. Random masking is applied to each sequence of length N before it is passed to the neural network. The first layer is a 1D convolutional layer with m channels that processes data in local neighborhoods. The resulting feature maps are pooled to reduce dimensionality and then flattened. The flattened representation is passed through an encoder with k layers to obtain the latent representation. The decoder then uses k layers to reconstruct the output. The final output layer of the ZINBAE predicts the mean μ , dispersion θ , and zero-inflation probability π for each position. Performance of the model is measured using the mean absolute error (MAE), coefficient of determination R^2 , and the two loss functions as described in subsection 3.4.5.

3.4.4. Hyperparameter Search

The data was split into a training, validation, and test set based on chromosomes. The training consists of chromosomes III, IV, VI, VII, IX, X, XI, XIII, and XVI. The validation set, used to select the best-performing hyperparameter configuration, consists of chromosomes IIV, XIV, and XV. The test set used for final evaluation includes chromosomes I, II, V, and XII. The chromosomes were randomly assigned to the three sets.

We applied Bayesian hyperparameter optimization (BHO) to identify the optimal hyperparameter configuration [81]. BHO prioritizes hyperparameter settings that consistently yield good performance, thereby accelerating the search. BHO was implemented using the `gp_minimize` function from the `scikit-optimize` Python package. We used 20 initialization points followed by 150 iterations. All other settings were kept to their default values. A more detailed overview of the hyperparameters, including their descriptions, ranges, and motivations, is provided in Appendix C.

We considered three different objective functions. One BHO configuration optimizes the ZINB loss (Eq. 3.5). This configuration tries to accurately reconstruct the data, without an explicit dependency on the masked reconstruction loss. The second approach applied BHO to optimize the masked reconstruction loss (Eq. 3.6). This setup should prioritize accurate reconstruction of the masked values. The last approach applied BHO to optimize a combination of the ZINB loss and the masked reconstruction loss with $\beta = 1$ (Eq. 3.7). This approach focuses on accurately reconstructing the entire dataset while still emphasizing the reconstruction of masked values. However, because $\beta = 1$ was chosen arbitrarily, it remained unclear whether equal weighting of the two terms would lead to optimal overall performance.

One of the hyperparameters considered in the BHO was the preprocessing strategy, specifically whether the data should be binned or smoothed using a moving average. However, because the initial results suggested that smoothing with a moving average makes imputation easier, we also performed a separate BHO in which only binning was allowed as a preprocessing option.

The optimal hyperparameter configuration obtained from the BHO across the different approaches is shown in Table 3.1.

Table 3.1: The optimal hyperparameter configuration obtained from the BHO for the different approaches.

Category	Hyperparameter	ZINB loss	Combined loss	Masked Recon.	Combined bins
Preprocessing	Features	Nucl	Centr	Centr & Nucl	Centr
	Step size	1.0	0.25	0.75	0.81
	Bin size	1	20	20	17
	Moving Average	NA	True	True	False
Model Architecture	Number of layers	3	3	3	3
	First layer size	1600	1600	752	1168
Convolutional Layer	Use Conv	True	True	False	True
	Conv Channels	128	118	NA	49
	Pool Size	8	8	NA	7
	Kernel Size	3	9	NA	5
Training	Epochs	150	144	93	116
	Batch size	128	32	64	128
	Learning rate	$1e - 5$	$2e - 5$	0.00012	$1.5e - 4$
	Reconstruction Loss β	0.001	0.078	0.0033	1.37
Regularization	Dropout	0.0	0.0	0.18	0.42
	Regularizer	None	None	None	l1
	Regularizer strength	NA	NA	NA	$3.5e - 4$

3.4.5. Evaluation

The resulting AE models are evaluated using four metrics. The first two are the loss functions used during training: the ZINB negative log-likelihood and the masked reconstruction loss, with lower values indicating better performance. The third metric is the coefficient of determination R^2 . This metric measures how well the reconstructed values match the original values, relative to a baseline model that always predicts the data's mean. An R^2 value of 1 indicates perfect reconstruction, whereas a value of 0 indicates performance equal to predicting the mean. The fourth metric is the mean absolute error (MAE), which measures the average difference between the model's predicted and original values. Both R^2 and MAE are used to measure performance across all data points, including masked values, and only on masked values. For these metrics, the predicted value was taken as the model-estimated mean, μ .

3.5. Synthetic Data Generation

To evaluate the performance of our change-point detection algorithm, we generated synthetic datasets from a ZINB distribution while incorporating centromere and nucleosome biases. Each generated dataset contains approximately 500,000 positions, similar in size to some of the chromosomes of *S. cerevisiae*. Each dataset consists of consecutive regions with varying region lengths L_r and mean values μ_r . The length of a region, L_r , is sampled independently from a uniform distribution $L_r \sim \text{Uniform}(40, 1000)$. The mean of each region μ_r is sampled from a Gaussian distribution $\mu_r \sim \mathcal{N}(4.4, 6.3^2)$, where the parameters were estimated from real SATAY data. Negative μ_r are truncated to zero. For each position i in a region r , a count is sampled from an NB distribution with mean μ_r and an overdispersion parameter $\theta = 0.5$, $x_{r,i} \sim \text{NB}(\mu_r, 0.5)$. The dispersion parameter θ was also estimated from real SATAY data, resulting in an average θ of 0.52 ± 0.20 .

To extend the synthetic data from an NB to a ZINB distribution, we estimated a value of π for each position. First, synthetic nucleosome positions were generated by sampling inter-nucleosome distances independently from a Gaussian distribution, $d \sim \mathcal{N}(167, 33.8^2)$. The mean and standard deviation were again estimated from real SATAY data. Starting from position 1, nucleosome positions were generated by cumulatively adding the sampled inter-nucleosome distances. The centromere position was set to the midpoint of the dataset. To determine π_i for each position, we first defined a global baseline $\pi = 0.6$. For each position, the distance to the centromere and to the nearest nucleosome was then calculated. Using the distance-dependent insertion rates obtained in Section 3.2, the baseline π was scaled according to the corresponding centromere and nucleosome insertion rates. Finally, each position was set to zero with probability π_i , yielding a synthetic dataset following a ZINB distribution.

3.6. Change-Point Detection

In this section, we describe the implementation of the change-point detection algorithm. We discuss how we evaluate our algorithm. Finally, we explain how we calculate essentiality levels for the resulting segments.

3.6.1. Window-based change-point detection

We use a window-based change-point detection approach to identify positions at which the underlying ZINB parameter changes. Because we are interested in changes in fitness, the parameter of interest is the mean of the NB distribution, μ . We scan the sequence using two consecutive, non-overlapping windows of size N , with a step size of $\frac{N}{2}$ (Fig. 3.3). At each step, we perform a hypothesis test. The null hypothesis assumes that the two windows have the same mean: $H_0 : \mu_1 = \mu_2$. The alternative hypothesis assumes that the two windows have a different mean: $H_1 : \mu_1 \neq \mu_2$. The zero-inflation parameter π is used to model the positions without an integrated transposon insertion, which depends on the distance to the centromere and nearest nucleosome. We assume a global dispersion parameter θ .

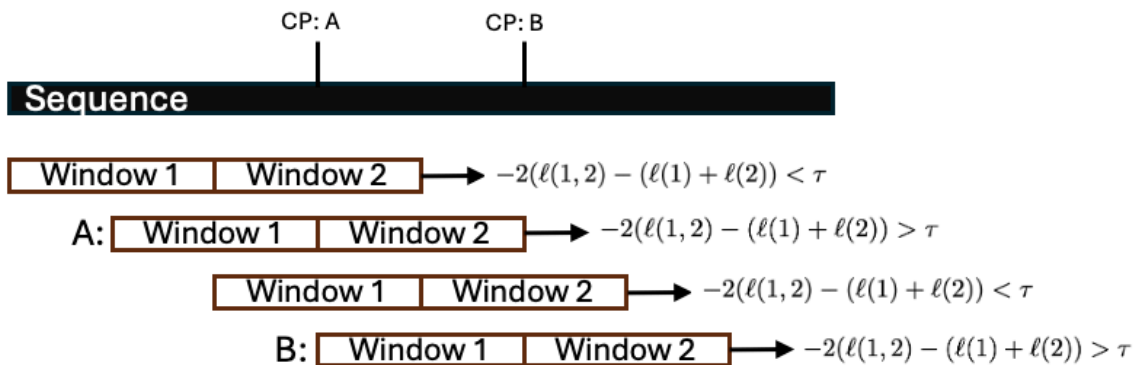


Figure 3.3: Overview of the change-point detection algorithm. Two windows slide over a data sequence. The optimal parameters are computed for both windows together and separately, and the corresponding log-likelihood values are computed. A likelihood ratio test is used to measure how much the two log-likelihood values differ from each other. If the likelihood ratio test exceeds a threshold τ , a change point is detected, as can be seen for examples A and B.

Under the null model, we estimate a shared mean μ across the two windows. Under the alternative model, we estimate μ separately for each window. In both cases, we estimate π separately for the two windows, since changes of π are not of interest. Using the estimated parameters, we compare the two models with a likelihood ratio test (LRT), which is commonly used for window-based change-point detection [69, 82]. The LRT is given with the following formula:

$$\lambda_{LRT} = -2 [(\ell(\mu, \pi_1, \theta) + \ell(\mu, \pi_2, \theta)) - (\ell(\mu_1, \pi_1, \theta) + \ell(\mu_2, \pi_2, \theta))], \quad (3.8)$$

The terms $\ell(\mu, \pi_1, \theta)$ and $\ell(\mu, \pi_2, \theta)$ are the log-likelihoods of the two windows under the null model, where both windows share the same mean μ , but have separate zero-inflation parameters π_1 and π_2 . The terms $\ell(\mu_1, \pi_1, \theta)$ and $\ell(\mu_2, \pi_2, \theta)$ are the log-likelihoods under the alternative model, where each window has its own mean, μ_1 and μ_2 . In all cases, θ denotes the dispersion parameter of the negative binomial distribution. A large value of λ_{LRT} indicates that the two windows separately provide a substantially better fit than the two windows combined. This suggests the presence of a change point. We evaluated the algorithm over a threshold range from 0 to 40.

The ZINB distribution does not have a closed-form. Thus, to estimate the parameters μ , π , and θ , we must rely on iterative numerical methods. For SATAY data, we estimate a global dispersion parameter θ on 2000 bp windows, since estimating it on the complete dataset tends to overestimate the global variance due to the additional variance introduced by regions with different distribution parameters. To estimate the θ for each window, we perform a grid search over values between 0 and 1. This range was chosen because the dispersion of SATAY data is expected to lie within it. The optimal value of θ was selected as the one that maximizes the log-likelihood after iterative estimation of π and μ . On zero counts, the NB distribution and the zero-inflation component π form a two-component mixture model. To estimate the contribution of each component, we use the Expectation-Maximization (EM) algorithm. We introduce a latent variable $z_i \in \{0, 1\}$, where $z_i = 0$ indicates that observation y_i was generated by the NB component and $z_i = 1$ indicates that it was generated by the zero-inflation component. For non-zero observations, the component membership is known, such that $P(z_i = 1 | y_i > 0) = 0$ and $P(z_i = 0 | y_i > 0) = 1$. For zero observations, the expectation step computes the posterior probability that the observation originated from the zero-inflation component:

$$P(z_i = 1 | y_i = 0) = \frac{\pi}{\pi + (1 - \pi) \text{NB}(0 | \theta, \mu)}. \quad (3.9)$$

In the maximization step, π and μ are updated as

$$\pi = \frac{1}{N} \sum_{i=1}^N P(z_i = 1 | y_i), \quad (3.10)$$

$$\mu = \frac{\sum_{i=1}^N P(z_i = 0 | y_i) y_i}{\sum_{i=1}^N P(z_i = 0 | y_i)}. \quad (3.11)$$

This EM algorithm is repeated until the parameters no longer change, or until 100 iterations. This process is repeated for each θ in the grid search to estimate the log-likelihood. The θ with the highest log-likelihood is chosen as the final parameter for the segment.

Estimation of μ and π using our EM algorithm becomes unstable for θ values lower than one, corresponding to high variance (Appendix F). However, this is exactly the range that SATAY data falls into according to our estimations. This is why we inform π using the centromere and nucleosome biases. First, we estimate a shared initial π_0 for both windows from the insertion rate corresponding to the centromere distance of the windows. We then scale π_1 and π_2 based on their distance to the closest nucleosome. The values of π_1 and π_2 are obtained by scaling $1 - \pi_0$ according to the ratio between the average expected insertion rate of the windows separately and the average expected insertion rate across the combined windows. This approach will not result in an exact estimation of π , however, it does provide a robust approximation that incorporates the main biological biases affecting zero inflation in SATAY data. Using the estimated value of π , the corresponding μ can then be estimated from the mean of the data, by correcting for the fraction of zeros attributed to zero inflation:

$$\hat{\mu} = \frac{\bar{x}}{1 - \hat{\pi}}. \quad (3.12)$$

The final estimates of μ , θ , and π are used to calculate the log-likelihood of the observed data under the ZINB distribution, as described in section 3.3.

3.6.2. Essentiality estimation

To estimate the essentiality of a region, we first estimate the mean parameter μ of the ZINB distribution for each region. We then add a pseudocount of 1 to stabilize regions with μ close to zero. This avoids extremely negative logarithmic values from near-zero estimates. The transformed values are then standardized as

$$z = \frac{\log(\mu + 1) - \overline{\log(\mu + 1)}}{\sigma_{\log(\mu + 1)}} \quad (3.13)$$

where $\overline{\log(\mu + 1)}$ is the length-weighted mean of the log-transformed regional μ values, and $\sigma_{\log(\mu + 1)}$ is the corresponding length-weighted standard deviation. We use the logarithm because otherwise the standard deviation exceeds the mean. After standardization, a z-score greater than 0 indicates that the region is less essential than average, whereas a z-score less than 0 indicates that it is more essential than average.

We consider two different approaches to estimate the mean value μ . The first approach jointly estimates μ and π using the EM algorithm explained in subsection 3.6.1, where the resulting μ serves as the segment estimate. However, the EM algorithm can be inaccurate for highly dispersed data. Thus, in the second approach, we estimate π using the centromere and nucleosome biases and then use π to estimate μ for each region. The advantage of this approach is that it avoids unstable EM-based estimates of π , although it relies on an approximate rather than directly inferred value of π , which may also introduce error into the estimate of μ .

3.6.3. Segment merging

Our window-based approach is a heuristic for detecting change points. The presence of outliers can lead to the detection of false change points. This can result in oversegmentation, where two consecutive windows have very similar essentiality scores, but are still separated by a change point. To overcome this issue, we decided to merge consecutive segments with an essentiality score difference below a specified threshold. In this project, we decided to use a threshold of 0.25, equivalent to 1/4 SD, since we expect a difference of this size to be too small to reflect a meaningful shift in essentiality.

We merge segments using a top-down approach. First, we identify the two consecutive segments within a chromosome with the smallest difference in essentiality score. These segments are then merged, and the essentiality score of the merged segment is computed as the length-weighted average of the two original segment scores. Next, we identify the consecutive pair of segments with the next-smallest difference and merge them in the same way. This process is repeated until no consecutive segments have a difference smaller than the threshold.

3.6.4. Gaussian change-point detection

When applying the AE on SATAY data, the resulting output becomes smoothed and continuous. This breaks the assumption of a ZINB distribution. Thus, to apply change-point detection on our AE output, another approach is required. We propose a Gaussian CPD algorithm, which assumes the AE output is continuous and normally distributed within each segment. This might not fully align with reality, but it provides a better approximation than our ZINB-based approach.

Similar to our ZINB-based approach, the Gaussian CPD algorithm uses a sliding window. For each window, the mean and the variance are estimated. The standardized mean difference between the windows is then calculated and tested against a threshold. If the difference exceeds the threshold, a change point is detected. This is similar to a two-sample t-test:

$$z = \frac{\mu_1 - \mu_2}{\sigma_{\text{diff}}}, \quad (3.14)$$

where $\sigma_{\text{diff}} = \sqrt{\frac{\sigma_1^2 + \sigma_2^2}{N}}$ and μ_i and σ_i are the mean and standard deviation of their respective windows, and N is the window size.

3.6.5. Evaluation

In this subsection, we first describe how we evaluate our CPD algorithm, then assess the estimation of essentiality, and finally compare similarities between CPD results from different strains.

Evaluation of the CPD algorithm

We use precision to measure the fraction of detected change points that match a true change point within a tolerance window equal to the window size.

$$\text{Precision} = \frac{\left| \left\{ \hat{t} \in \hat{T} : \exists t \in T \text{ s.t. } |\hat{t} - t| \leq N \right\} \right|}{|\hat{T}|}, \quad (3.15)$$

where T is the set of true change points, and $\hat{T}$ the set of detected change points, and N is the window size. Recall measures how many of the existing change points have been successfully detected:

$$\text{Recall} = \frac{\left| \left\{ t \in T : \exists \hat{t} \in \hat{T} \text{ s.t. } |\hat{t} - t| \leq N \right\} \right|}{|T|}. \quad (3.16)$$

When two true change points occur within a single window, we will count only one detected change point, as the window can distinguish at most a single change point, and the second would therefore not appear in the final output.

Evaluation of the essentiality level calculation

To evaluate the accuracy of the μ estimates of our essentiality level estimation in subsection 3.6.2, we measure the relative mean absolute error (RMSE) between the estimated μ and the true μ from the synthetic datasets. In addition, we also assess whether the predictions preserve the ordering of the segment essentiality. For all segment pairs detected by the change-point detection algorithm, we compared the predicted and true differences in μ and recorded whether their signs matched, thereby evaluating whether the relative ranking of segments was correctly recovered.

Comparison between strains

We use the Adjusted Rand Index (ARI) to measure the similarity between CPDs across strains. It quantifies how often pairs of genomic positions are assigned to the same or different segments in both segmentations, while correcting for agreement expected by chance. The ARI ranges from 1 to -1, where 1 indicates complete agreement between the two segmentations, 0 indicates agreement no better than random chance, and -1 indicates less agreement than expected by chance.

Given two segmentations with segments $T_1, \dots, T_m$ and $\hat{T}_1, \dots, \hat{T}_k$, the ARI is calculated as follows:

$$\text{ARI} = \frac{\sum_{i,j} \binom{n_{ij}}{2} - \frac{(\sum_i \binom{a_i}{2})(\sum_j \binom{b_j}{2})}{\binom{n}{2}}}{\frac{1}{2} \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \frac{(\sum_i \binom{a_i}{2})(\sum_j \binom{b_j}{2})}{\binom{n}{2}}}, \quad (3.17)$$

where $n_{ij} = |T_i \cap \hat{T}_j|$ is the number of genomic positions shared by segment T_i in the first segmentation and segment $\hat{T}_j$ in the second segmentation. Furthermore, $a_i = \sum_j n_{ij} = |T_i|$ is the size of segment i in the first segmentation, $b_j = \sum_i n_{ij} = |\hat{T}_j|$ is the size of segment j in the second segmentation, and n is the total number of positions.

The term $\sum_{i,j} \binom{n_{ij}}{2}$ counts the number of pairs of positions that are assigned to the same segment in both segmentations. The term $\frac{(\sum_i \binom{a_i}{2})(\sum_j \binom{b_j}{2})}{\binom{n}{2}}$ represents the number of such agreements expected by chance. Therefore, the numerator measures the observed agreement above the chance level. The denominator normalizes this value by the maximum possible agreement above chance, making the ARI suitable for comparing similarity between segmentations with different numbers and sizes of segments.

To quantify the similarity in essentiality scores across strains, we use Pearson's correlation. Pearson correlation measures the strength and direction of the linear relationship between two sets of values. Its value ranges from -1 to 1, where 1 indicates a perfect positive linear relationship, 0 indicates no linear relationship, and -1 indicates a perfect negative linear relationship. In our case, a high positive Pearson correlation indicates that genomic positions in one strain have similar essentiality levels to those in the other strain.

4 Results

4.1. Data Exploration

In this section, we investigate the impact of two known biases in SATAY data: the centromere and nucleosome biases. We analyze how the distance to a nucleosome or centromere influences the transposon insertion rate and the mean transposon count.

4.1.1. SATAY insertion patterns are biased by their proximity to the centromere and nucleosomes

SATAY data is affected by two major biases. First, transposon insertions occur more frequently near the centromere. Second, transposon insertions are less likely in nucleosome-occupied regions. To examine these biases, we plotted the transposon insertion rate as a function of distance to the centromere (Fig. 4.1a) and distance to the nucleosome dyad, defined as the midpoint of the nucleosome (Fig. 4.1c).

Close to the centromere, the transposon insertion rate is, on average, higher than 10%, whereas farther away it decreases to below 5%. The centromere bias is most severe in the first 200 kbp but stabilizes farther from the centromere, although several peaks remain throughout the signal. These peaks are consistent across strains, suggesting that their genomic locations are biased towards a higher transposon insertion rate. Previous work has shown that the centromere bias is caused by a spatial bias, since the plasmid containing the transposon is located near the centromere [51]. These recurring peaks may therefore reflect chromosomal regions that are also spatially closer to the plasmid.

On average, nucleosomes repeat themselves approximately every 160bp [83]. This is consistent with our nucleosome data, which show a mean distance of 167 bp and a median of 163 bp between neighboring nucleosomes. Although nucleosome positions are not completely static, a clear bias in transposon insertion rate is still visible (Fig. 4.1c). The insertion rate almost doubles between a distance of 0 bp to 100 bp from the nucleosome dyad. At larger distances, the insertion rates increase by a factor of 3 compared to the insertion rate at a distance of around 0. However, the estimates from these distances also show more noise due to the low number of positions located far from the nucleosomes.

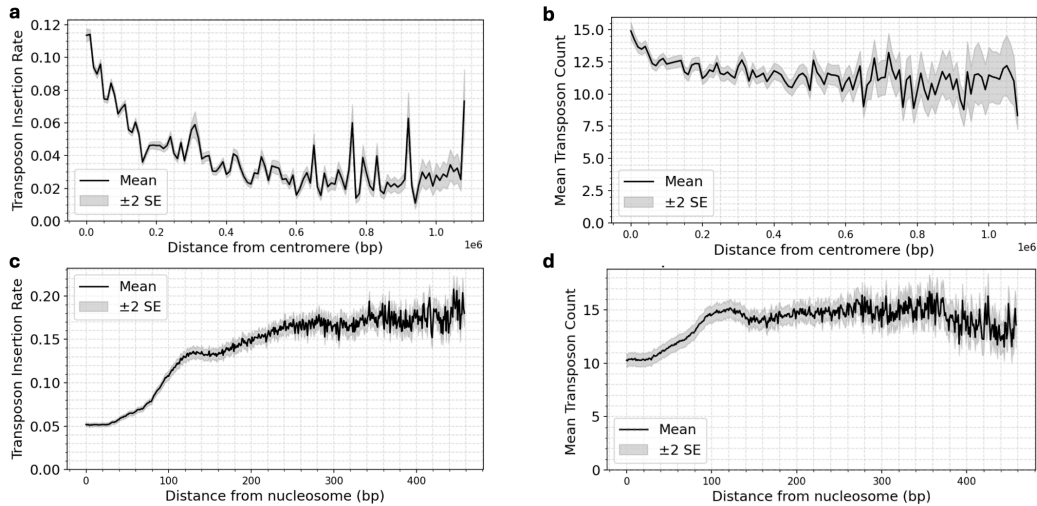


Figure 4.1: Centromere and nucleosome biases in SATAY data. **a** Transposon insertion rate plotted against the distance from the centromere. The data was averaged in bins of size 10,000 to reduce the noise. **b** The mean transposon count at non-zero positions, plotted against centromere distance. The data was averaged in bins of size 10,000 to reduce the noise. **c** Transposon insertion rate plotted against the distance from the nucleosome. At about 200 bp, the number of data points decreases, leading to increased noise. The plot was cut off at 453, which is the largest nucleosome distance that occurs on each chromosome. **d** The mean of transposon counts on non-zero positions plotted against nucleosome distance. At about 200 bp, the number of data points decreases, leading to increased noise. The plot was cut off at 453. These plots were created using uncombined datasets, and the results were averaged across all datasets. Overall, both the centromere and nucleosome distance affect SATAY insertion rates, indicating that transposon insertions are biased at both genome-wide and local scales.

We also examined whether proximity to the centromere and nucleosomes influences the transposon count at insertion sites. The centromere does not seem to have a large impact on the mean count (Fig. 4.1b). In contrast, the mean count of non-zero positions is approximately 50% higher at a nucleosome distance of 100 compared to a distance of 0. One possible explanation for this bias is that the same genomic position can be hit by transposons independently in different cells, leading to higher observed read counts after culturing. Because insertions are less likely to occur near nucleosomes, repeated insertions at the same position may also be less frequent in these regions.

Overall, both the centromere and nucleosome biases are apparent in our datasets. The centromere bias leads to a higher insertion density near the centromere than in regions closer to the telomeres, thereby increasing coverage near the centromere. However, since the centromere bias spans such a large distance, its influence when focusing on local patterns will most likely be minimal. In contrast, the nucleosome bias most of the time spans less than 100 bp but still significantly influences both the transposon insertion rate and the number of transposon reads. Combined with the fact that nucleosome positions are not static, this may make it difficult to distinguish between nucleosome-associated depletion and a truly essential local region when encountering a region with depleted transposon insertions.

4.2. Autoencoder

This section evaluates the trained AE models with respect to reconstruction quality, imputation behavior, and robustness across dataset saturation levels. First, we will go over their general reconstruction and imputation performance and characteristics. Then we will look into their imputation potential. Lastly, we will analyze whether the model's performance generalizes across different dataset saturation levels.

4.2.1. The combined-loss moving-average model achieves the strongest reconstruction

We performed hyperparameter optimization using three objective functions: the ZINB negative log-likelihood (ZINB NLL model), the masked reconstruction loss (masked recon. model), and the combination of these two losses (combined model). The combined model has the strongest overall reconstruction performance. On both the full dataset and the masked values, it attains the lowest MAE and masked reconstruction loss, as well as the highest R^2 (Fig. 4.2b,c,d,e,f). The only exception is the ZINB NLL. The model optimized for only the ZINB NLL achieves the best performance for this metric (Fig. 4.2a). In all other metrics, the ZINB NLL model performs the worst. This indicates that optimizing the ZINB NLL alone does not yield accurate reconstruction. The model optimizes ZINB NLL by predicting a high zero-inflation probability, π , for most values, which is effective for sparse datasets but does not produce meaningful reconstructions (Fig. 4.3a,d). Including the masked reconstruction loss in the objective function prevents this behavior.

Although the combined model performs similarly on masked values to the full dataset, it still assigns a high zero probability, π , to most zero values (Fig. 4.3d). Additionally, the predicted mean value μ of the zero-values is, on average, low as well, compared to the predicted mean μ of non-zero values (Fig. 4.3e). This contrasts with the zero probability and mean value assigned to the masked values, which is more similar to the predicted values for non-zero values (Fig. 4.3a,f). The strong reconstruction performance on masked values is therefore unlikely due to true imputation. When comparing the model's performance on the masked values with using the mean and median of its neighbors, the model achieves similar results for MAE and R^2 (Fig. 4.4). The optimal model configuration uses a moving average, which smooths the data and reduces fluctuations. This likely makes masked values easier to reconstruct, and also explains why using a combined objective function results in better masked value reconstruction compared to an objective function that only uses the masked value reconstruction loss: the reconstruction performance of the masked values is linked to the overall reconstruction performance (Fig. 4.2d,e,f). Training a model with higher percentages of masked values led to higher zero-probability values, π . However, it also reduced the performance of all metrics and increased estimation uncertainty, making the estimations unreliable (Appendix E). Thus, to achieve imputation while maintaining good reconstruction, increasing the percentage of masked values is insufficient.

We also optimized a model that bins the data instead of using a moving average. This model cannot rely on the moving average for imputation. However, this model performs poorly across all metrics (Fig. 4.2). It struggles to estimate accurate mean values μ , especially at higher counts (Appendix D). This is accompanied by a very low estimated dispersion parameter θ . A lower θ corresponds to higher

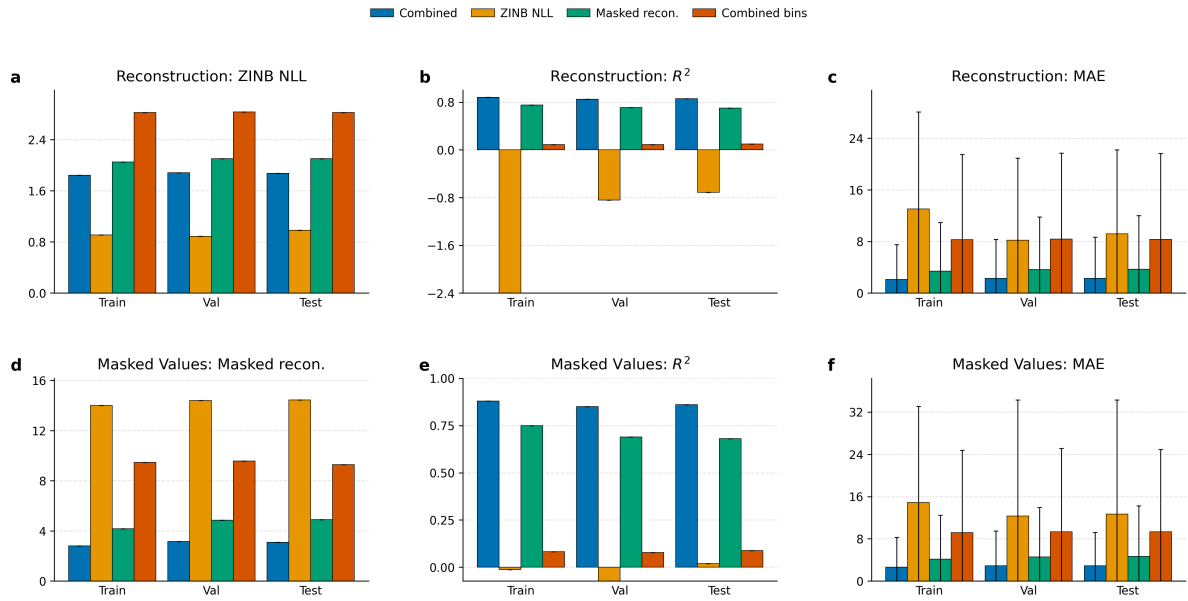


Figure 4.2: Performance of the four AE models on the full dataset and at the masked positions. **a** ZINB negative log-likelihood (ZINB NLL). **b** Coefficient of determination, R^2 , for the reconstructed μ values on the full dataset. **c** Mean absolute error (MAE) on the full dataset. **d** Masked reconstruction loss. **e** R^2 of the masked positions. **f** MAE of the masked positions. The combined model performs best across most metrics.

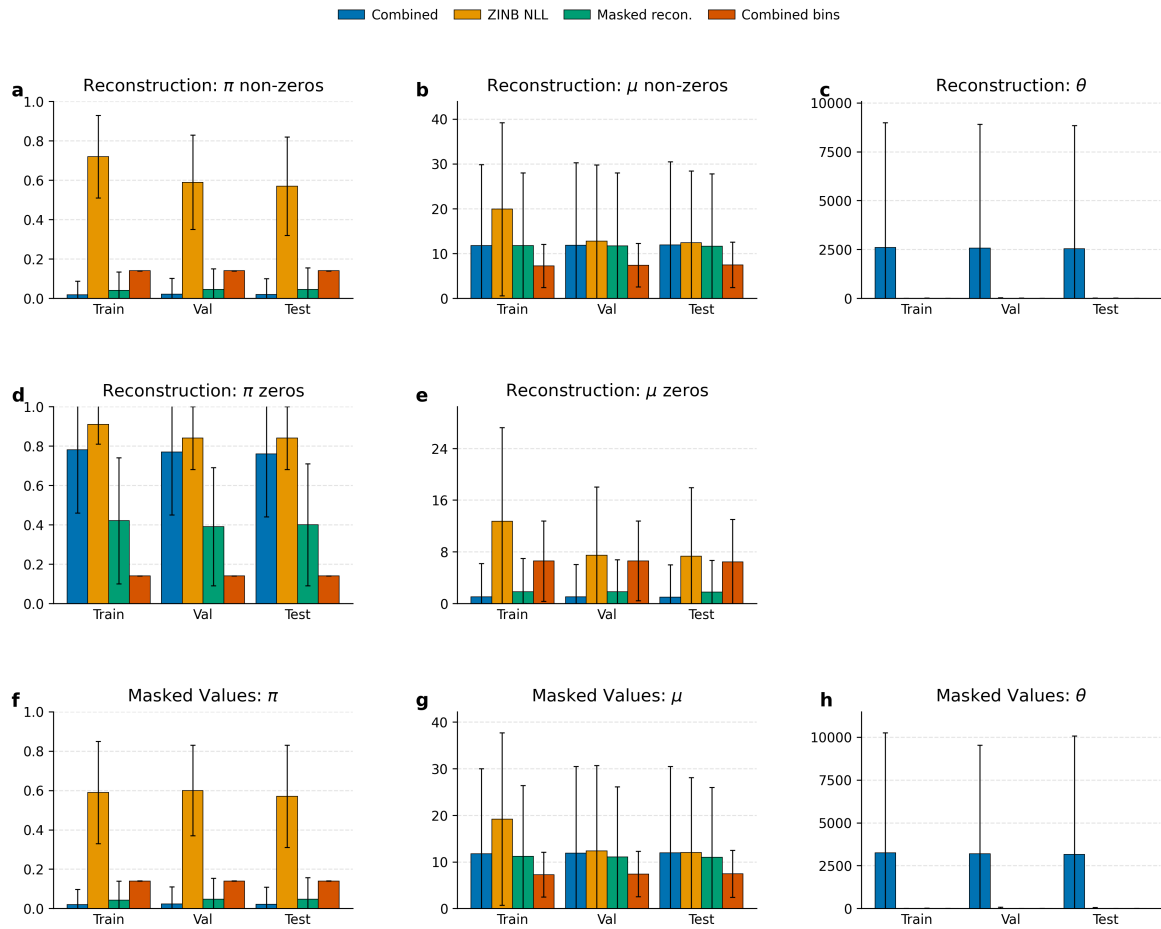


Figure 4.3: Average predicted ZINB parameters for the different models. **a** Average π for non-zero values. **b** Average μ for non-zero values. **c** Average θ across all values. **d** Average π for zero values. **e** Average μ for zero values. **f** Average π for masked values. **g** Average μ for masked values. **h** Average θ for masked values.

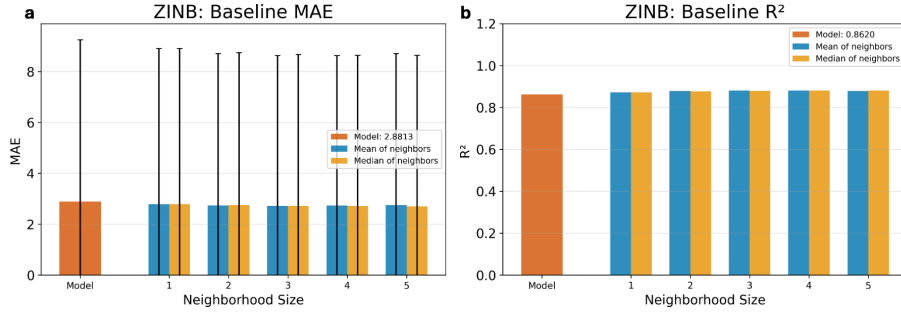


Figure 4.4: Comparison of AE imputation with local baseline methods Comparison of the masked-value performance of the moving-average model with the mean and median of reconstructed neighboring values for different neighborhood sizes. **a** MAE of the reconstructed masked positions. **b** R^2 of the reconstructed masked positions. The model does not perform better at imputation than the baseline methods.

variance, and the model estimated a θ as low as possible (Fig. 4.3b). It is unlikely that the binned model produces any meaningful output.

Although binning reduces data sparsity, it does not smooth the data or reduce fluctuations as a moving average does. Additionally, binning greatly reduces the number of training samples: with a bin size of 17, the number of samples decreases from 61992 to 4096. We also kept the sample size fixed at 2000. However, after binning, this may be too large for training, since a single sample now spans 340,000 bp. Together, these factors likely explain the binned model's poor performance.

Overall, producing meaningful imputation remains challenging. Currently, a moving average is required for strong reconstruction performance, which makes the reconstruction of masked values trivial. The combined model achieves a high R^2 and low MAE, but still predicts a high zero probability, π , for most zero values. Models with lower predicted zero probabilities for zero values, thus showing imputation potential, suffer from worse reconstruction performance and a low dispersion parameter θ , corresponding to high uncertainty (Fig. 4.3). However, a good reconstruction performance is also necessary to ensure accurate imputation values. In the next section, we will further explore the imputation potential of the combined and masked recon. model.

Even though the model does not impute missing values, the model might still denoise the data. SATAY data is highly noisy, and AE architectures are commonly used for denoising. However, from these metrics alone, it is difficult to determine whether the reconstructions denoise the data meaningfully. This should be analyzed in downstream analyses, where it can be measured whether the AE output yields meaningful results.

4.2.2. Strong reconstruction is necessary, but not sufficient, for meaningful imputation

To assess whether the model can distinguish between true zeros and technical zeros using the predicted zero-inflation probability π , we computed the average π for original zero values in essential and non-essential genes for both the combined model and the masked recon. model. We performed this analysis on all values from the train, validation, and test sets, across all strains. The strains have different genetic backgrounds, which could lead to differences in gene essentiality compared to the reference annotation used. Additionally, an essential gene can have non-essential domains. However, by aggregating all of the data together, we assume that meaningful imputation would at least lead to an on average higher zero-probability π for essential genes compared to non-essential genes.

For the combined model, the zero probability, π , of most original zero values is close to 1. This is true for both essential and non-essential genes. However, on average, the π -value of non-essential genes is lower than that of essential genes (Fig. 4.5a). In addition, the distribution for non-essential genes shows a stronger spread toward lower π -values, indicating that the model predicts lower zero probabilities at these positions than at positions within essential genes. This difference between positions within essential and non-essential genes is also significant according to the Mann-Whitney U test, which measures whether two independent samples differ significantly in their distributions. This indicates that the model can distinguish between excess zeros and biologically meaningful zeros to some extent. However, the π -values remain high overall, meaning that the model still assigns most zero values a high probability of being excess zeros. Ideally, biologically meaningful zeros would consistently receive lower π -values, allowing for a clearer separation between technical zeros and true biological signal.

In contrast, the masked reconstruction model predicts lower zero probabilities, π , but assigns higher

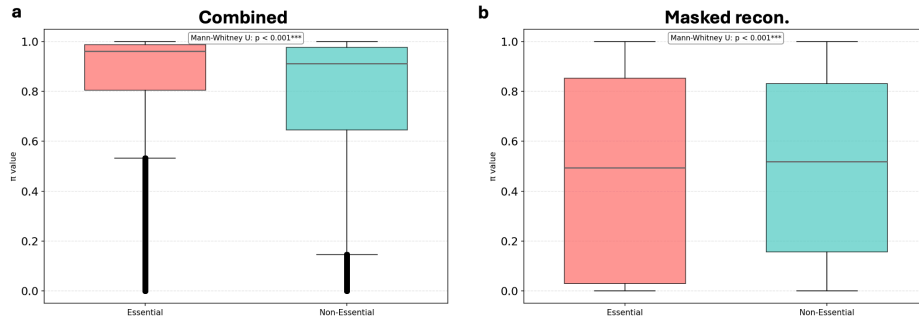


Figure 4.5: Zero-probability estimates by gene essentiality. Predicted zero probabilities, π , are shown for original zero values in essential and non-essential genes for the **a** combined model and **b** masked reconstruction model. Differences between essential and non-essential genes were significant in both models, as determined by the Mann–Whitney U test. The combined model assigned higher zero probabilities to zeros in essential genes, as expected, whereas this pattern was not observed for the masked reconstruction model.

π -values to non-essential genes (Fig. 4.5b). This difference is also significant according to the Mann–Whitney U test. This model also showed lower reconstruction performance. Together, these results suggest that the reduction in zero probabilities π does not reflect more meaningful imputation, but rather a poorer overall fit to the data. This further supports the idea that good reconstruction performance is necessary for meaningful imputation.

4.2.3. Higher saturation improves exact reconstruction

The model was trained on the combined strains, which differ in saturation level (Appendix B). To assess how dataset saturation affects model performance, we evaluated the trained combined model on datasets generated by combining different numbers of the original yEK23 replicates, resulting in a range of saturation levels. We then measured the ZINB NLL, MAE, R^2 , and masked reconstruction loss across these datasets (Fig. 4.10).

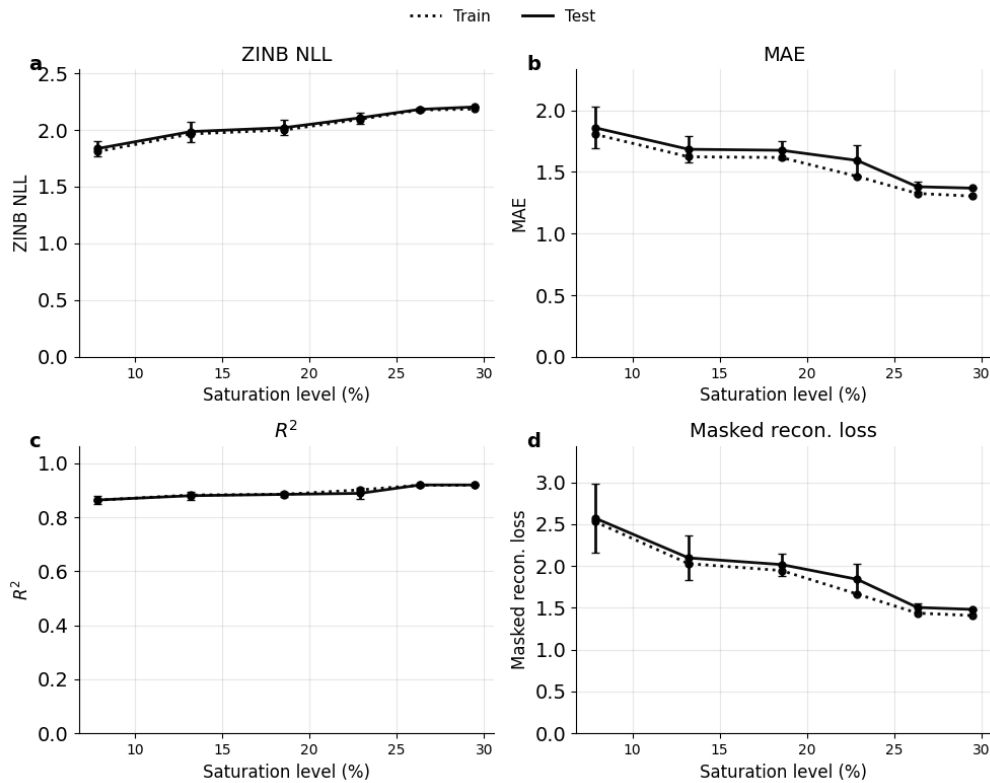


Figure 4.6: Performance metrics across datasets with different saturation levels. **a** ZINB negative log-likelihood. **b** MAE. **c** R^2 . **d** Masked reconstruction loss. The reconstruction performance slightly decreases for lower saturation levels.

In general, datasets with higher saturation appear to yield better reconstruction performance: as saturation increases, the MAE decreases (Fig. 4.10b), the R^2 slightly increases (Fig. 4.10c), and the masked reconstruction loss decreases (Fig. 4.10d). The only exception is the ZINB NLL loss, which increases for more highly saturated datasets (Fig. 4.10a). This suggests that although higher-saturation datasets are reconstructed more accurately, they are less well captured by the ZINB NLL loss. Notably, the MAE seems to benefit more from the increase in saturation than the R^2 , which only slightly increases. This suggests that higher saturation mainly improves the exact reconstruction of individual values, while the reconstruction of broader patterns in the data improves only slightly with increasing saturation, as reflected by the relatively stable R^2 , which measures how well overall variation in the data is captured.

4.3. Change-Point Detection applied on synthetic data

In this section, we present the results of our CPD algorithm on synthetic data. We will assess how window sizes affect performance and how well our CPD algorithm performs. We evaluate the CPD without segment merging, since the synthetic datasets contain segments with randomly varying, and sometimes small, differences in mean values that could otherwise be merged incorrectly.

4.3.1. Larger window sizes increase precision but reduce recall

We evaluated 8 window sizes across a range of thresholds using a reference CPD algorithm that estimates μ from the true values of π and θ . For each window size and threshold, the precision and recall are computed, and a precision-recall curve is created (Fig. 4.7). A larger window size improves precision by smoothing over noise. However, larger window sizes are less sensitive to smaller segments. The minimum segment length in our synthetic data is 40. Larger window sizes are more likely to miss these smaller segments, since the signal gets diluted by surrounding positions. Additionally, a larger window size can fully encompass these regions, making it impossible to place two change points around it. We partly mitigate this loss of resolution by using a step size smaller than the window size. Nevertheless, recall decreases for the largest window sizes. For window sizes up to 100, the lower thresholds yield similar recall values, though precision varies. In contrast, the two largest window sizes show a clear decrease in recall at lower threshold values. The most likely explanation is that these window sizes start to miss change points corresponding to smaller segments. To balance performance and resolution, we therefore chose a window size of 100 for the remaining experiments, as it achieves a similar recall to the smaller window sizes while providing higher precision.

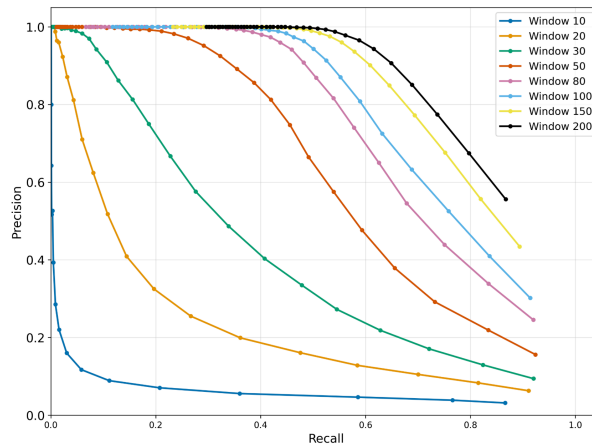


Figure 4.7: Precision-recall curve for different window sizes. The ZINB CPD reference algorithm was applied to 10 different synthetic datasets and averaged together to create this plot. Larger window sizes increase precision, but window sizes that are too large reduce recall.

4.3.2. Centromere- and nucleosome-informed π yields good and stable performance

We applied both the reference CPD algorithm and our CPD algorithm, which informs the zero-inflation probability π using centromere and nucleosome biases, to 10 synthetic datasets. For both methods, we used a window size of 100 and evaluated the precision and recall for a range of thresholds (Fig.

4.8). The precision-recall curve of the two methods is almost completely overlapping, showing that our approach performs almost exactly the same as when we know the exact zero-inflation values π . This is despite the fact that our π values are only approximations of the true values. However, the π values are biased similarly across both windows, while still capturing their relative differences correctly. As a result, the estimated μ values may also be biased, but the change in μ between windows is likely preserved, which is the quantity that matters for change-point detection.

Both our approach and the reference algorithm still detect many incorrect change points at high recall values. However, the dispersion parameter θ we estimated from SATAY data is around 0.5, which is very low. This means the data is highly overdispersed, making it difficult to accurately identify change points. We think this is the main reason for low precision at high recall values, since the high variance increases the likelihood that random fluctuations will be mistaken for true changes between segments.

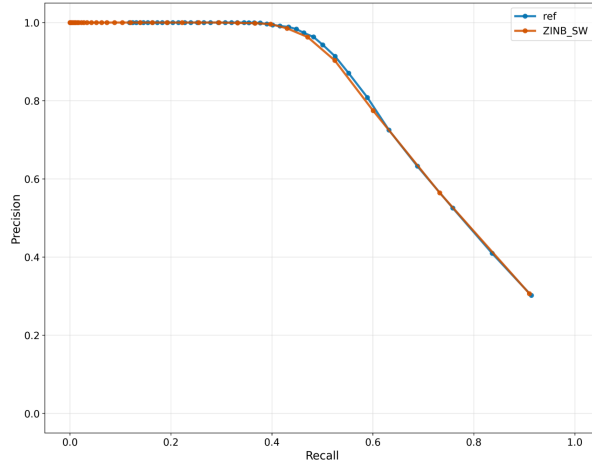


Figure 4.8: Precision-recall curve for ZINB-SW and the reference algorithm. Curves are shown for 10 synthetic datasets. The reference algorithm uses the true zero-inflation parameter π , whereas the ZINB sliding-window approach (ZINB-SW) estimates π from the centromere saturation of the two combined windows and further adjusts it using nucleosome distance. The ZINB-SW performs almost as well as the reference algorithm.

4.3.3. Estimation of essentiality level

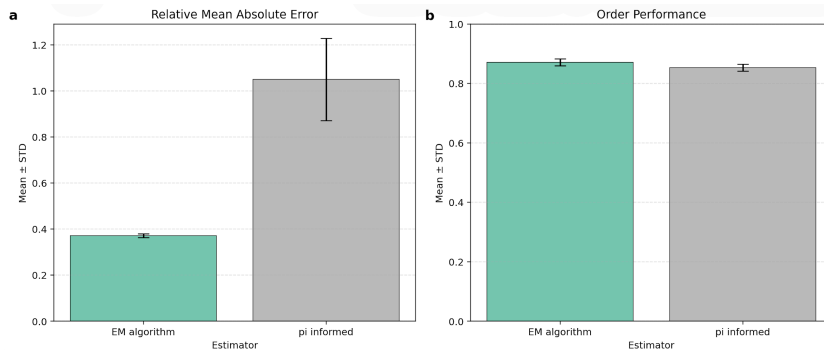


Figure 4.9: EM-based versus π -informed mean estimation. **a** Relative mean absolute error and **b** order performance, measuring whether the estimated means preserve the true ordering between pairs of μ values. In the EM algorithm approach, μ and the zero-inflation parameter π are estimated jointly using an EM algorithm. In the π -informed approach, π is first estimated from centromere and nucleosome distance, after which μ is estimated based on this π . The EM algorithm provides more accurate estimates of the mean parameter but does not always preserve the true ordering of pairs of μ values.

To estimate a segment's essentiality level, we need to estimate the mean μ . We tried two different approaches. The first approach uses the EM algorithm to jointly estimate π and μ . The second approach estimates π from centromere and nucleosome distances, then estimates μ . We applied both methods to the segments generated from synthetic data, compared their RMSEs, and assessed whether the estimated means preserve the true ordering of pairs of μ values (Fig. 4.9). The RMSE of the π -informed method is more than twice as high as the RMSE of the EM algorithm. This further

emphasizes that the π value obtained from the centromere distance is not an accurate representation of the true π value. However, for both methods, more than 80% of the pairs have the correct order, meaning that the estimated μ values still preserve the relative ranking between most pairs of true μ values. Still, the EM algorithm better preserves the order than the π -informed method. 20% of the pairs are ranked incorrectly, meaning that differences in estimated μ should be interpreted with some caution, especially when values are close together. Given the higher accuracy, we decided to use the EM algorithm to estimate the segment-level essentiality.

4.4. Change-point detection applied on SATAY data

In this section, we apply our CPD algorithm to real SATAY data. First, we assess the performance of our CPD algorithm on both raw data and AE output using hand-annotated data. We then compare the algorithm's output between different strains. Lastly, we look into how well the model aligns with known annotations.

4.4.1. ZINB-based CPD performs best despite AE denoising.

We tested our ZINB-based CPD algorithm on raw SATAY data and a Gaussian-based CPD algorithm on AE output. We created 6 datasets with increasing saturation levels by combining an increasing number of yEK23 strain datasets. We created two datasets with lower saturation by randomly removing count values. These datasets were provided to the AE to create the AE output datasets. A 1000-bp region surrounding the centromere of each chromosome was retrieved from the raw data and hand-annotated. This region and its AE output are segmented using the ZINB-based CPD (Fig. 4.10a) and Gaussian-based CPD (Fig. 4.10b) respectively. We also applied Gaussian-based CPD on the raw data processed by a moving average of window size 20, the same window size that the preprocessing algorithm of the AE uses (Fig. 4.10c). We do this to analyze whether the AE meaningfully denoises the data relative to the denoising already provided by a moving average. The precision and recall were calculated for each method over a range of thresholds and displayed in a precision-recall curve.

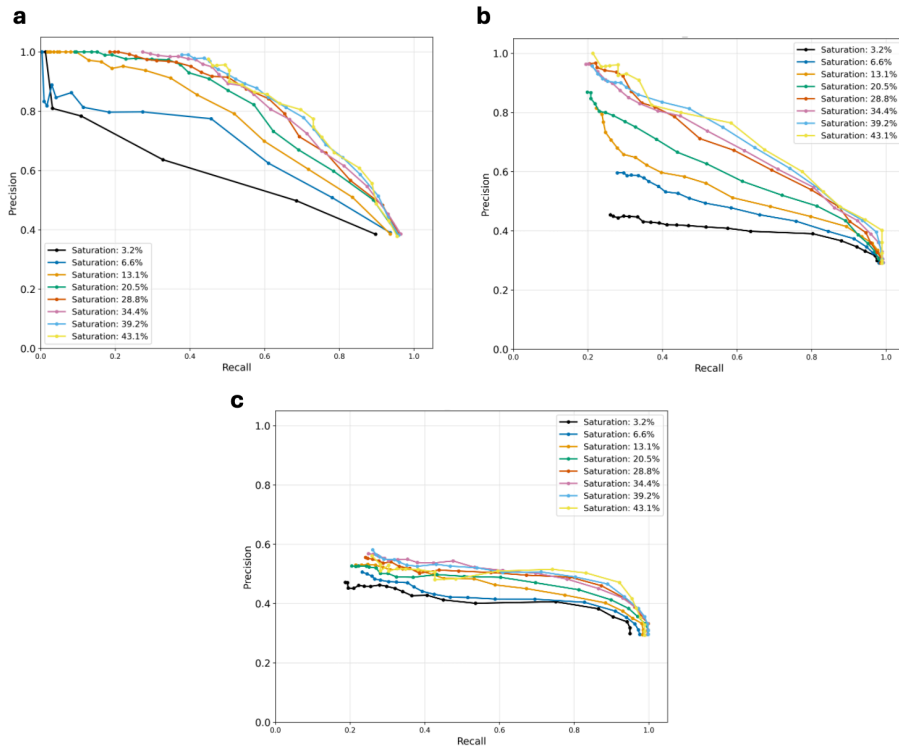


Figure 4.10: Comparison of CPD approaches across saturation levels. Precision–recall curves **a** of our ZINB-based CPD algorithm applied on raw data, **b** a Gaussian-based CPD algorithm applied on the AE output, and **c** a Gaussian-based CPD algorithm applied on raw data processed with a moving-average of window size 20. All algorithms are applied to hand-annotated regions spanning 1000 bp on either side of each centromere, shown across different saturation levels. Overall, the ZINB-based CPD algorithm achieves the best precision-recall trade-off, while the AE output improves performance compared with raw data processed with a moving average.

The Gaussian-based CPD produces better results on the AE output than the raw output processed with a moving average. The most likely reason is that the moving average alone is insufficient to denoise the data sufficiently to apply a Gaussian-based algorithm. This shows that the AE actually meaningfully denoises the data. However, despite the AE denoising the data, the ZINB-based CPD on raw data has the highest precision for all recall values. Additionally, the precision remains more stable at lower saturation levels than with the CPD applied to the AE output. This shows that our ZINB-based approach is more effective at identifying change points in SATAY data. However, the results for the AE output might be improved by developing a tailored CPD approach for the data rather than relying on a Gaussian CPD algorithm.

The ZINB-based CPD shows similar precision-recall curves for saturation levels between 20.5%-43.1%. However, lower saturation levels do show a clear decrease in precision. The zero-inflation component of the ZINB distribution can compensate for sparsity to some extent, but at some point, the information in the data becomes insufficient to make accurate predictions. It should also be noted that although the overall shape of the precision-recall curve remains consistent across different saturation levels, the same threshold yields different precision/recall values. Since a dataset shows different saturation levels at different distances from the centromere, this should be taken into account when interpreting the results.

4.4.2. Agreement of our CPD results depends mostly on the saturation level of the datasets

We applied our CPD algorithm to all the data across all strains. We compare the similarity between the predicted change points of different strains using the Adjusted Rand Index (ARI) (Fig. 4.11a), and the correlation between the estimated essentiality levels for each segment by measuring Pearson correlation between the different strains (Fig. 4.11b). For both metrics, the similarity between the datasets appears to be strongly influenced by their saturation levels. The three datasets with the highest agreement, FD, yEK19, and yEK23, all have a saturation level of around 30% (Appendix B). The datasets dnrp, yTW001, and ylic137 all have a saturation level of around 9%, and show similar ARI and correlation patterns. Lastly, datasets yWT04a and yWT03a have an even lower saturation level, below 6%. In turn, they have very low ARI and correlation values. These results indicate that the similarity between predicted change points and estimated essentiality levels is primarily driven by dataset saturation rather than biological differences between strains. This corresponds to our previous results, which show that the performance of the CPD algorithms begins to decrease at low saturation levels.

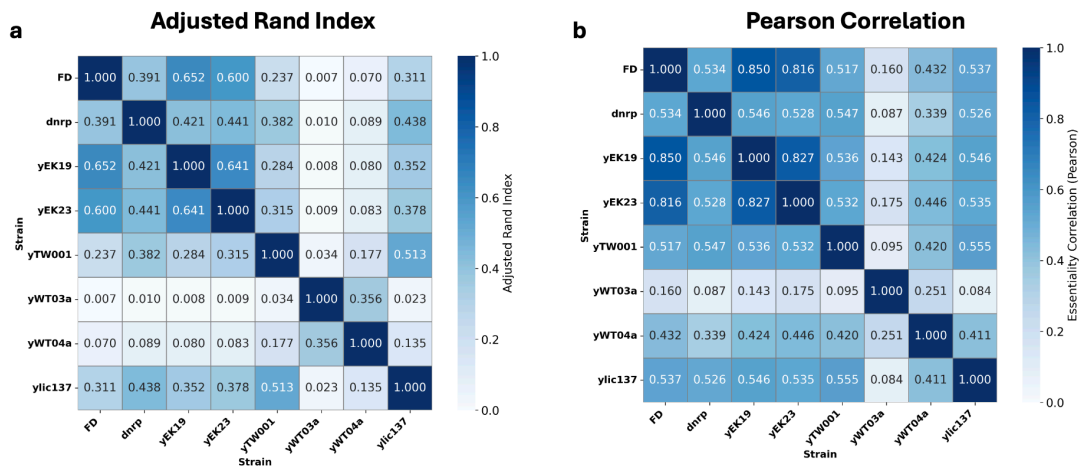


Figure 4.11: Agreement between CPD results of different strains. CPD results were compared across all strains using **a** the similarity of the predicted segmentations, measured by the Adjusted Rand Index, and **b** the correlation between the estimated essentiality levels, measured by the Pearson correlation coefficient. Agreement between strains appears to depend more strongly on data saturation than on biological differences between strains.

The FD and yEK19 strains are both generated from wild-type strains, while yEK23 is generated from a mutated strain. We assume that the mutated strain has a different fitness landscape, meaning that the genes and genomic regions important for survival and growth differ from those in the wild-type strains. The ARI and Pearson correlation between yEK23 and the two wild-type strains is also lower

than that between FD and yEK19. However, these differences are small compared to those between the other strains, so it is not entirely clear whether they are due to biological factors or technical noise.

There might be several reasons why no clear biological patterns are visible in these datasets. The changes in the fitness landscape might be too small relative to the size of the entire genome to detect interesting patterns in a global analysis. Additionally, the essentiality levels are calculated relative to a strain-specific mean value. It is currently unclear to what extent essentiality levels can be compared across strains, since each dataset is affected by its own noise and biases. When analyzing changes in the fitness landscape between strains, it is probably most effective to compare local regions within each strain, while accounting for potential differences in essentiality levels due to strain-specific noise and biases.

4.4.3. CPD-derived essentiality scores capture biologically meaningful genomic patterns

To assess whether the results of our CPD algorithm are associated with biologically meaningful genomic features, we examined how the essentiality scores are distributed across essential genes, non-essential genes, and regions outside annotated genes within the wild-type strain yEK19 (Fig. 4.12). Essential genes are strongly enriched at the lowest scores, which correspond to high essentiality. This enrichment drops rapidly as the score increases. This shows that, in general, our CPD algorithm correctly identifies essential genes.

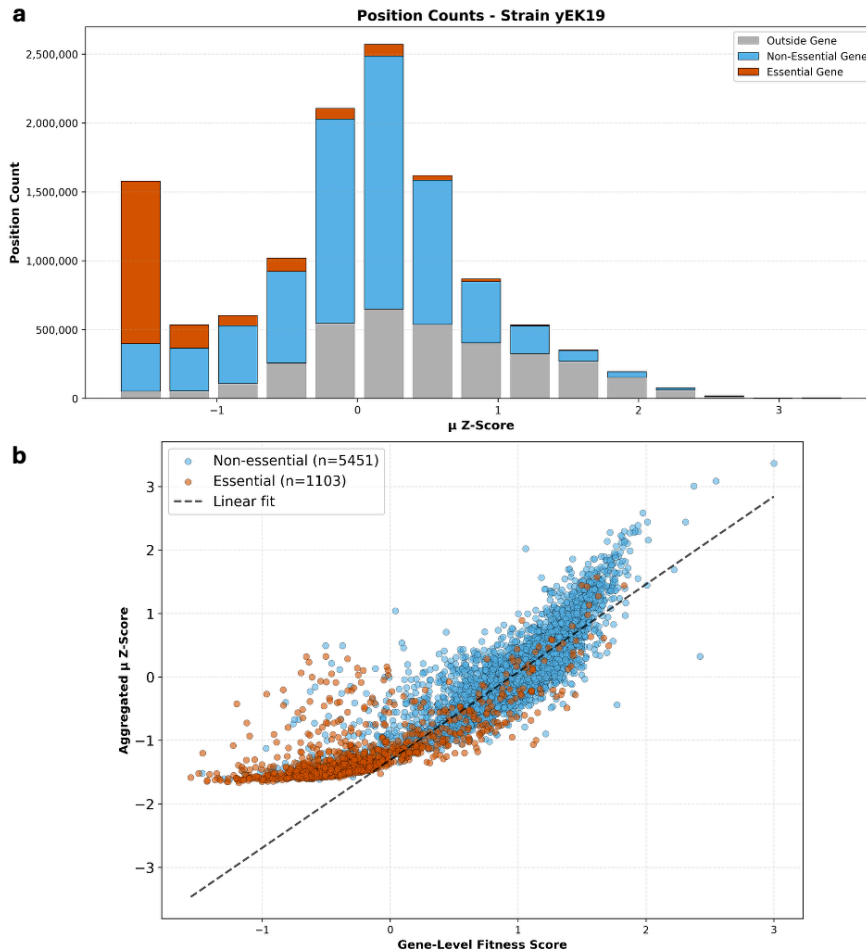


Figure 4.12: Estimated essentiality compared with SGD annotations and SATAY fitness scores. **a** Stacked histogram showing the number of positions across bins of essentiality scores (Z-scores), separated by essential genes (red), non-essential genes (blue), and regions outside of annotated genes (gray). **b** Gene-level comparison of our aggregated μ Z-scores with SATAY-derived fitness scores from the method of Kingma et al. [26]. Each point represents an annotated gene, with essential genes shown in red and non-essential genes shown in blue. The x-axis shows the fitness score obtained by Kingma et al. [26], whereas the y-axis shows our estimated μ Z-score. The dashed line shows a linear fit of the data. In general, the essentiality estimates from our method align with both SGD essentiality annotations and SATAY-derived fitness estimates from Kingma et al. [26], with essential genes showing lower scores than non-essential genes.

In contrast, non-essential genes and regions outside of gene boundaries are more common around essentiality scores of 0. This suggests that insertions in these regions tend to produce effects near the average, consistent with a more neutral fitness impact. Non-essential genes are slightly shifted toward lower scores, corresponding to higher essentiality, whereas regions outside gene boundaries are shifted toward higher scores, corresponding to lower essentiality or non-essentiality. This likely reflects the fact that, although genes annotated as non-essential are not required for survival or reproduction, their disruption can still impair growth, while insertions outside gene boundaries are generally better tolerated. Overall, these results are within our expectations and show that our CPD algorithm can assign meaningful essentiality scores.

We also compared our estimated essentiality scores with the fitness scores obtained using the method of Kingma et al. [26] to assess whether our scores are consistent with other SATAY-based measures of fitness (Fig. 4.12b). This method estimates gene-level fitness from SATAY data by averaging the mean transposon count in the central 80% of each annotated gene, adding unobserved expected insertions as zero-count sites. The resulting mean read count per gene is log-transformed and normalized by the median log-transformed mean read count across genes to obtain a relative fitness score. We computed a gene-level essentiality score for our method by taking a length-weighted average of the regional μ Z-scores overlapping the central 80% of each annotated gene. We fit a linear regression between the gene-level essentiality scores from our method and the SATAY-derived fitness scores from Kingma et al. [26] to quantify agreement between the two approaches.

Overall, our gene-level essentiality scores are consistent with the SATAY-derived fitness scores from Kingma et al. [26], although the agreement weakens for the most depleted genes. Annotated essential genes generally correspond to lower scores compared to non-essential genes in both methods. For fitness scores higher than 0, our μ -Z-score increases linearly with the fitness score, indicating that the methods produce similar gene-level essentiality estimates. However, for fitness scores lower than 0, the linear relationship starts to break down, since the pseudocount added to our μ estimates compresses differences between very small values. Kingma et al. [26] does not require a pseudocount, as their mean estimates are computed over whole genes and genes without transposon counts are filtered out, making near-zero mean estimates uncommon. In contrast, our CPD algorithm is more sensitive to locally depleted regions and can therefore produce regional μ estimates close to zero. Without the pseudocount, these near-zero estimates would produce extremely negative log values and dominate the logarithmic mean and standard deviation used for Z-score normalization. The pseudocounts reduce the resolution of the essentiality score for highly essential genes, but this compression mainly affects differences among the most strongly depleted regions, while still allowing essential genes to be distinguished from genes with higher transposon counts.

4.4.4. The CPD algorithm identifies biologically relevant essential regions, but interpretation requires caution

To assess how well the detected change points align with gene boundaries, we calculated the percentage of genes that have a change point within one window size (100 bp) of a boundary across different thresholds (Fig. 4.13a). At low thresholds, nearly all gene boundaries have a nearby change point because the data is oversegmented, but as the threshold increases, this percentage decreases. Essential genes consistently show a higher percentage of boundaries near change points than non-essential genes, indicating that the CPD algorithm preferentially detects changes near essential gene boundaries. Although higher thresholds reduce the total number of nearby change points, those retained are more often associated with essential genes, suggesting that the algorithm captures biologically relevant boundaries.

This is also reflected in the number of change points detected within gene boundaries ± 100 bp (Fig. 4.13b). Most essential genes have two boundaries, indicating that essentiality is often detected either across the full gene or within a distinct sub-genic region marked by change points on both sides. In comparison, non-essential genes are more often associated with zero or only one change point, indicating that these regions are less frequently identified as distinct from their surrounding regions. Still, many genes have more than two change points. This might indicate that within genes, there are regions with different levels of essentiality. However, these change points might also be incorrect and detected due to noise or biases.

We also examined example genes to assess how well our results align with their biological annotations. We have retrieved the location, essentiality, and known protein domains from the SGD and Pfam database [11], and aligned this to the segments generated by applying our CPD algorithm on the three

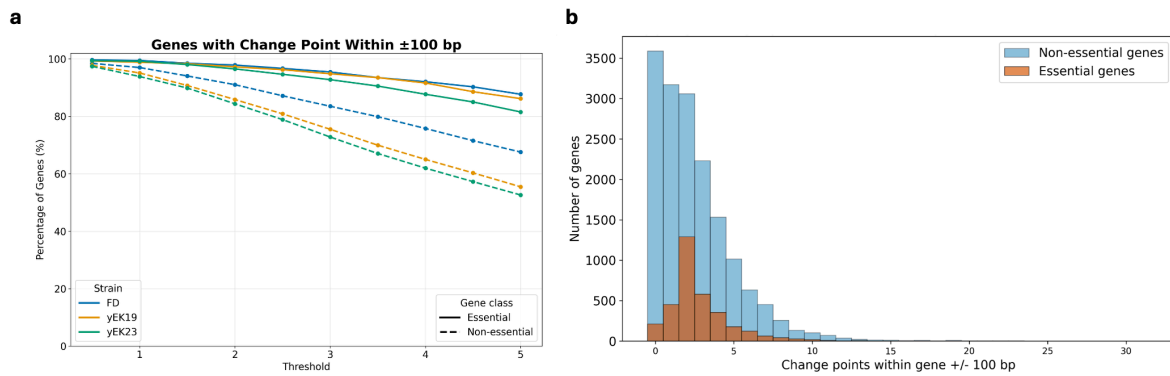


Figure 4.13: Change-point locations in essential and non-essential genes. **a** Percentage of genes with a change point within ± 100 bp of a gene boundary, shown separately for essential and non-essential genes. **b** The number of change points within a gene and ± 100 of its boundaries for both essential and non-essential genes. The CPD algorithm appears to capture biologically meaningful transitions in essentiality, especially near essential-gene boundaries.

strains with relatively high saturation levels: FD, yEK19, and yEK23 (Fig. 4.14). We used a threshold of 3 in all examples.

The gene *CDC42* shows an example of undersegmentation (Fig. 4.14a). *CDC42* is an essential gene, but the CPD algorithm failed to detect the change point at its right boundary for strain yEK19. This led to an incorrect estimate of the essentiality level. This shows why using a lower threshold can be more beneficial, since oversegmentation can be addressed in downstream analyses, whereas undersegmentation results in information loss.

As shown previously, essential genes are generally assigned low essentiality scores, indicating strong essentiality. Some essential genes have change points that correspond closely to the boundary of the genes (Fig. 4.14b). Others have change points that are more closely aligned to the protein domain of an essential gene (Fig. 4.14c). However, there are also some examples of essential genes that seem to have an essential region that does not align with a previously annotated protein domain (Fig. 4.14d). This shows that our CPD algorithm can identify previously annotated regions and may also provide new insights into important regions.

There is some discrepancy between annotated non-essential genes and the results from our CPD algorithm. *CLA4* is annotated as a non-essential gene, but in our results, some regions within these genes show very low scores, indicating essentiality (Fig. 4.14e). Additionally, *BOI1* is annotated as non-essential but shows a region within one of its protein domains with low essentiality scores (Fig. 4.14f). It should be taken into account that the distinction between essential and non-essential is not always that clear. In general, a gene is annotated as non-essential if a cell can survive and reproduce without this gene. However, this does not mean that the gene is unimportant, as its loss may still reduce fitness or become detrimental under specific conditions. This might be the case for *CLA4* and *BOI1*. The cells might be able to grow without those genes, but their growth rate can still be significantly reduced. A cell with a reduced growth rate will be quickly overtaken by cells with more optimal growth rates, resulting in a depletion of transposon insertions within these non-essential but growth-reducing genes. This might explain why genes annotated as non-essential show essentiality levels similar to those of essential genes in the SATAY data. However, other factors may also contribute to this discrepancy, such as context-dependent gene function, inconclusive annotations, or biases in SATAY data.

An example of a non-essential gene that also appears non-essential in our results is *BOI2* (Fig. 4.14e). In general, this gene shows essentiality levels around or above 0, indicating that this region is likely non-essential. However, small regions within the gene exhibit a sudden depletion of transposon insertions, indicating essentiality. However, these regions, due to their small size, are most likely caused by the presence of nucleosomes. In general, segments with a length of 100, the minimal segment length, are overrepresented in the data. These regions also tend to have lower essentiality levels (Appendix G). Even though we try to account for nucleosome positions in our CPD algorithm, not all nucleosome positions are known or correct. Additionally, as shown in subsection 3.2, transposon counts are lower near nucleosomes, which we do not account for in our CPD algorithm. For these reasons, the detection of small regions with low scores that appear highly essential but in fact correspond to nucleosome positions is unavoidable, and this should be taken into account in downstream analyses.

As mentioned before, strain yEK23 is generated from a mutant strain, while FD and yEK19 are both generated from a wild-type strain. A fitness landscape of a mutant strain is often different from

that of a wild-type. A possible example of this might be *AXL2* (Fig. 4.14h). Compared to FD and yEK19, yEK23 shows lower scores in this region, suggesting that this gene becomes more essential in this mutant. It should be noted that it is currently unclear how well results across different SATAY datasets can be compared, as they are affected by their own strain-specific noise and biases. Thus, our method shows potential for identifying strain-specific changes in gene essentiality and local differences in the fitness landscape between mutant and wild-type backgrounds. However, these findings should be further verified in the lab by comparing the effects of *AXL2* gene deletion in both the wild-type and mutant strains.

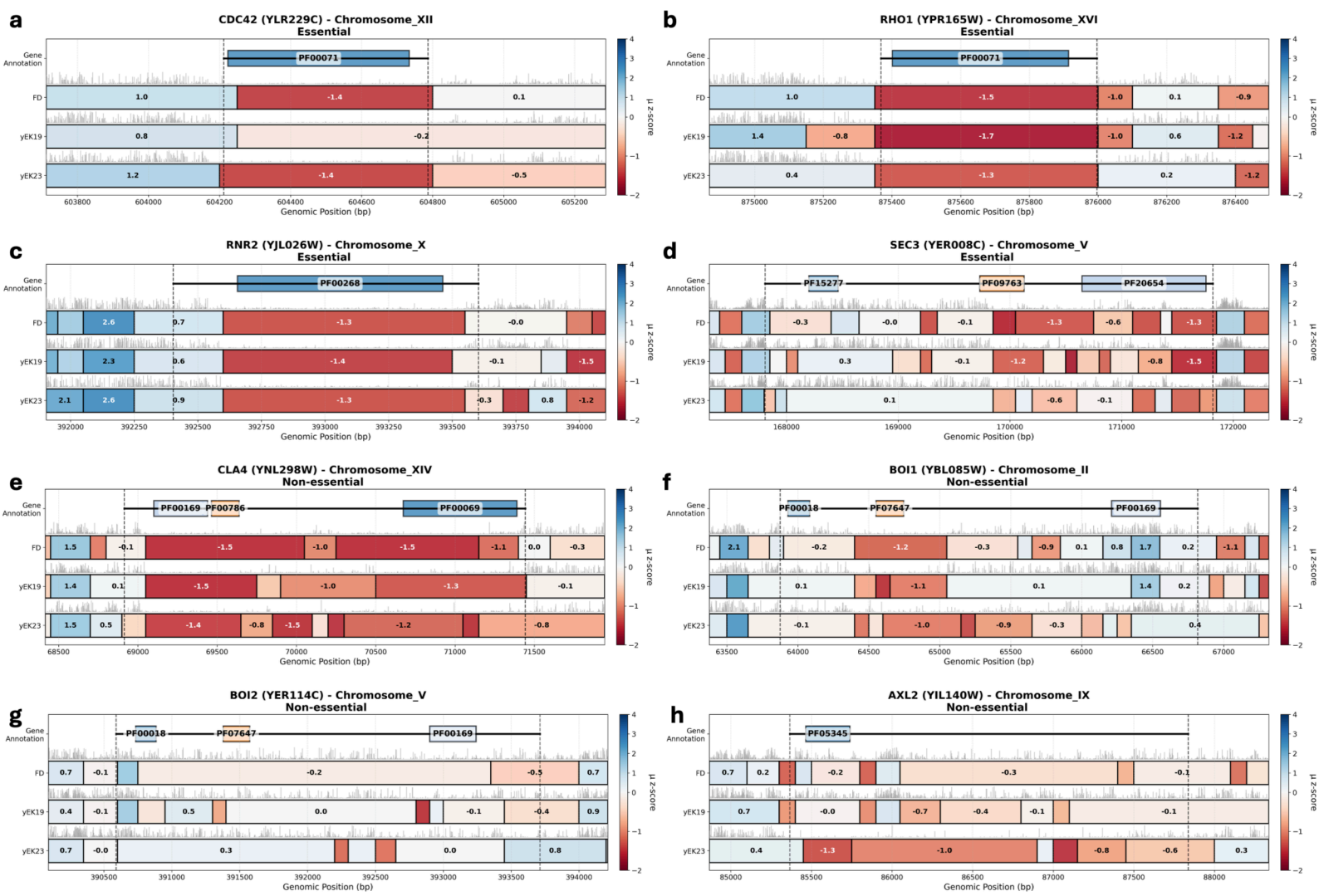


Figure 4.14: Example CPD segmentations across genes and strains. The segmentation results are shown for 8 different genes across three strains: the two wild-type strains FD and yEK19, and the mutant strain yEK23. For each gene, the top track shows the gene annotation and annotated Pfam protein domains. The lower tracks show the segments created by the CPD algorithm and their corresponding calculated essentiality level. The dashed line shows the annotated gene boundaries. 8 different genes are shown: **a** CDC42, **b** RHO1, **c** RNR2, **d** SEC3, **e** CLA4, **f** BOI1, **g** BOI2, and **h** AXL2.

5 Discussion

In this study, we explored the imputation and denoising of SATAY data using an AE architecture and detected changes in essentiality levels using a window-based ZINB CPD algorithm. On the one hand, meaningful imputation of SATAY data using an AE remains unsuccessful, but the AE does seem to denoise the data. On the other hand, the CPD algorithm detects biologically relevant patterns in SATAY data when sufficiently saturated. In this section, we will discuss these findings, their implications, the main limitations, and directions for future work.

5.1. Summary of findings

SATAY data is characterized by sparsity and noise. The sparsity of SATAY data is apparent from the centromere bias: close to the centromere, a single SATAY experiment typically reaches a transposon insertion rate of around 12%, whereas this decreases to around 3% closer to the telomeres. In addition to this genome-wide insertion pattern, insertion rates vary locally with distance to the nearest nucleosome. The data is noisy, as it shows a large variation in transposon counts. Together, the sparsity and noise of the data complicate the identification of genomic regions with different essentiality levels, which was the central goal of this project.

Our first approach was to develop an AE model to reconstruct and impute SATAY data to improve downstream analyses. Of the four AE models tested, only the moving-average combined-loss model achieved good reconstruction performance. The moving-average preprocessing helped reduce local noise, making the underlying patterns easier for the AE to capture. In addition, the ZINB loss was important for modeling the sparse, overdispersed count structure of SATAY data. The masked reconstruction loss was included to counteract the model's tendency to assign high zero probabilities to most positions.

Although the combined-loss model achieved good reconstruction performance, it was unable to perform meaningful imputation. The model still assigned high zero probabilities to many zeros, indicating that it mainly learned to preserve the data's sparsity rather than recover missing insertions. This is further supported by its stable performance across saturation levels: if the model were imputing more missing values in sparser datasets, reconstruction performance would be expected to decrease, but this was not observed. These results suggest that the ZINB loss may penalize imputed non-zero values too strongly. Although the masked reconstruction loss was intended to encourage recovery of masked values, it was not sufficient to counteract this effect. Nevertheless, the AE assigned higher zero probabilities to zeros in essential genes than to those in non-essential genes, suggesting that it captured some distinction between biologically meaningful and technical zeros. However, for most zero values, the predicted zero probability remained well above 0.5, meaning that the model still modeled them as true zeros rather than candidates for imputation.

Even though the AE did not meaningfully impute SATAY data, it did appear to denoise the data. We applied a Gaussian-based CPD algorithm to the AE output and compared it with the same algorithm applied to raw data processed with a moving average. While the moving-average baseline showed little potential for effective CPD, the AE output yielded better precision-recall trade-offs. This suggests that the AE denoised the data beyond the effect of the moving average alone.

However, the best CPD performance was achieved by the tailored ZINB-based CPD algorithm applied directly to raw SATAY data. This approach uses centromere and nucleosome biases to estimate local zero-inflation probabilities. On synthetic data, this bias-informed approach performed almost as well as a reference CPD algorithm with access to the true underlying zero-inflation probabilities. The zero-inflation component makes the method robust to sparsity, while the negative binomial component captures the overdispersion in insertion counts. As a result, the ZINB-based CPD algorithm outperformed the Gaussian-based CPD algorithm applied to the AE output and was also more robust across different saturation levels on hand-annotated SATAY data.

The robustness across saturation levels is particularly important because individual SATAY experiments are highly sparse. On real SATAY data, precision remained relatively stable above saturation levels of roughly 20%, whereas below this level, there may be insufficient information to reliably estimate the distribution parameters. Since a single SATAY experiment typically yields a saturation of about 3% to 12%, depending on the genomic region, multiple laboratory replicates should still be generated and combined to obtain sufficiently saturated datasets for reliable CPD. Nevertheless, by explicitly

accounting for zero inflation, the method reduces the number of experimental replicates required for interpretable segmentation.

The segments detected by our ZINB-based CPD algorithm and their estimated essentiality scores show agreement with known biological annotations in the SGD database. Positions corresponding to essential genes had significantly lower essentiality scores than positions in non-essential genes and regions outside genes. In addition, detected change points appeared to align more closely with the boundaries of essential genes than with those of non-essential genes. This indicates that the CPD algorithm captures biologically meaningful patterns. At the same time, the detected segments were not limited to whole-gene boundaries. In some cases, they aligned with smaller regions, such as protein domains, or with no known annotation. This suggests that the algorithm may identify biologically relevant sub-genic regions, but further validation is required to determine whether these finer-scale patterns reflect true biological substructure rather than oversegmentation or residual noise.

Taken together, these findings show that SATAY data contains biologically meaningful information, but that this information is difficult to recover with the current AE architecture. The AE can denoise the data, but it does not reliably impute missing values or improve downstream CPD. In contrast, the ZINB-based CPD algorithm performs well because it directly incorporates key properties of SATAY data, including sparsity, overdispersion, zero inflation, and insertion biases. This suggests that, for the current data and task, a domain-informed statistical approach is more effective than a more flexible but less constrained deep learning model.

5.2. Limitations and Future Work

5.2.1. Redesigning the AE objective for downstream segmentation

The AE model was designed to denoise and impute SATAY data to improve downstream identification of genomic regions with different essentiality levels, but its output did not improve this downstream analysis. One of the main limitations is the sparsity and noise of the data. The ZINB-only model mainly reduced loss by assigning high zero probabilities to most positions. Although the combined-loss model partly prevented this behavior, it still preserved too much of the sparse input structure for meaningful imputation. The improved performance after moving-average preprocessing suggests that prior denoising is needed for the AE to capture relevant patterns. The black-box nature of AEs makes it difficult to identify the main bottleneck in the current approach. Although the AE was based on successful imputation architectures, it remains unclear whether the limited downstream performance is due to the model design, the optimization objective, the sparse and noisy data structure, or the Gaussian CPD algorithm applied to the AE output. Together, these limitations make the current AE approach an uncertain route for improving downstream segmentation without substantial changes to the model objective.

A key limitation of the current setup is that the AE is optimized for reconstruction and imputation, whereas the ultimate goal is to identify genomic regions with different levels of essentiality. The current AE architecture may still prove useful as a starting point for this task, since it appears to denoise the data beyond the moving average alone. However, rather than improving AE performance in denoising and data imputation, future work could explore AE models whose outputs are more directly aligned with downstream segmentation. In the current model, the mean, zero-probability, and dispersion parameters are estimated separately for each position, making the model highly flexible but not directed toward identifying regional patterns. Instead, the model could be encouraged to estimate similar ZINB parameters for neighboring positions, so that regions with similar essentiality are represented by similar mean insertion levels. A Fused Lasso loss could be used for this purpose, as it can be used to penalize large differences between neighboring ZINB parameters. This type of penalty is closely related to approaches used in CPD, in which neighboring positions are encouraged to remain similar unless there is evidence of a change point. Incorporating this penalty could help the AE suppress local noise while preserving abrupt changes that may correspond to transitions between regions with different essentiality levels. This would better align the AE objective with the downstream goal of identifying regions with different levels of essentiality.

5.2.2. Refining the CPD framework and interpretation of segments

Our CPD algorithm uses a window-based approach, which approximates change-point locations by comparing neighboring windows. The main advantage of this heuristic is its computational efficiency compared with an exact CPD algorithm. However, it introduces a trade-off between precision and resolution. Larger windows improve ZINB parameter estimation and, in turn, CPD, but reduce resolution,

making it harder to detect small regions and exact change-point locations. Using a window-based approach also makes the CPD algorithm more sensitive to oversegmentation compared to an exact approach. Outliers can have a strong influence within a window, therefore leading to the detection of an incorrect change point. On the other hand, if a true change point is missed, the window-based approach may merge two distinct regions into one segment. This can happen, for example, when the transition between regions is gradual, although we assume that most essentiality changes are abrupt. Future work could explore applying the CPD algorithm with multiple window sizes to obtain segmentations at different scales. This would allow broad regions of differing essentiality, as well as smaller localized changes to be detected. In addition, an exact search method could be explored as an alternative to the window-based approach, although its improvement in CPD would need to be weighed against the increased computational cost.

The reliability of the CPD results also depends on the assumptions we made about the data. First, we assume that SATAY data follows a ZINB distribution. However, SATAY data may deviate from this assumption because biological and technical variation are unlikely to be fully captured by a ZINB distribution. This can affect both the precision of the detected change points and the accuracy of the estimated essentiality levels. In addition, essentiality estimation is based on an expectation-maximization algorithm, which is not guaranteed to yield the optimal solution, especially for highly overdispersed data such as SATAY. Second, we estimate the zero-inflation probability for each window based on nucleosome distance. However, nucleosome positions are not static and may deviate from the reference annotations used in this study, thereby reducing the accuracy of these estimates and leading to imprecise change points. These assumptions should therefore be taken into account when interpreting the algorithm's results.

Future work could explore alternatives to the ZINB distribution. One major limitation of SATAY data is its extreme overdispersion in transposon counts. A log-transformation could help stabilize the variance and improve model fit, but this would violate the discrete count assumptions of the ZINB distribution. Therefore, log-transformed data would require a different modeling framework. Specifically, non-parametric distributions might offer greater flexibility by avoiding distributional assumptions. However, any alternative distribution should still account for the nucleosome bias, which may be more difficult outside a ZINB distribution.

Finally, these methodological limitations complicate the biological interpretation of the detected segments. The results appear to suffer from oversegmentation, resulting in many small regions with abrupt changes in essentiality scores. Some of these regions may be explained by the presence of nucleosomes. Although the algorithm attempts to correct for the nucleosome bias, it is so pronounced in the data that removing it entirely appears infeasible. In addition, some predicted change points may reflect technical variation rather than true shifts in biological essentiality. This makes it difficult to determine whether individual segments represent genuine biological differences or technical effects.

5.3. Broader implications

5.3.1. Biological implications

Our ZINB-based CPD algorithm provides a first step toward identifying regions with different levels of essentiality directly from SATAY data. Instead of relying only on predefined gene annotations, the method segments the genome into data-driven regions based on local insertion patterns. This enables essentiality to be studied at higher resolution and may help identify important regions within and outside genes, rather than focusing only on whole-gene essentiality. In future applications, this approach could contribute to studies of functional genomics and evolution in *S. cerevisiae* by revealing how the importance of genomic regions varies across environmental conditions or genetic backgrounds. Still, comparing CPD results across different background strains remains an open challenge, as estimated essentiality levels are influenced by strain-specific insertion patterns and overall saturation, making direct comparison non-trivial.

Our ZINB-based CPD approach might also be transferable to other TIS settings. Most existing computational methods for TIS data rely on known annotations or known essentiality classes. Our method can identify regions without prior information or predetermined classes. This makes our method useful for identifying unknown essential regions within an organism. However, applying the method to other organisms or TIS technologies would likely require adaptation to their specific insertion biases and data distributions. Nevertheless, the general principle of combining a count-based statistical model with CPD may be useful for broader analyses of insertion patterns.

5.3.2. Computer science implications

Our work raises several questions about the use of AEs for imputing and denoising sparse, noisy data. First, it remains unclear how sparse a dataset can be before it lacks enough signal for tasks such as imputation. In our data, the easiest way for the model to reduce the ZINB loss was to assign high zero-probability values to most positions, requiring additional constraints to capture any signal at all. Sparsity also affects the optimization objective of an AE architecture. In our work, we used loss functions based on reconstruction performance, as is common in AE architectures. However, imputing non-zero values from zero values is penalized in these loss functions. When a dataset is highly sparse, this penalization might become too large to allow the model to recover missing values. At the same time, ignoring zero values in the reconstruction loss would remove the model's incentive to retain biologically meaningful zeros. Therefore, AE-based imputation of highly sparse data requires a careful balance between reconstruction and imputation, making loss-function design an important direction for future research on sparse data.

Because imputing highly sparse data is difficult, our work highlights the value of analyzing sparse data directly with statistical models tailored to the data structure, provided that the data is sufficiently saturated. By using a ZINB distribution within a CPD framework, our method accounts for both overdispersion and excess zeros in the data. This shows that zero-inflated models can be used as part of a signal-processing approach for sparse data, without requiring prior imputation of missing values. More broadly, this suggests that downstream methods for sparse data may benefit from explicitly modeling excess zeros rather than treating sparsity as a preprocessing problem.

Together, these results suggest that domain-informed models can offer clear advantages over more generic deep learning approaches. The AE architecture did not improve downstream analyses compared with the ZINB-based CPD algorithm applied directly to raw SATAY data. This algorithm performed well because it explicitly modeled key data properties, including zero inflation, overdispersion, and insertion bias. Although related information was also provided to the AE through centromere and nucleosome biases and a ZINB-based loss, the model still had to learn to use these inputs from sparse, noisy data. This is especially challenging when the available signal is limited. These results illustrate an important trade-off between model flexibility and model guidance: AEs can, in principle, learn complex patterns, but this flexibility can also allow the model to optimize the training objective without learning a structure that improves the downstream task. In contrast, the ZINB-based CPD algorithm was less flexible but better aligned with the structure of the SATAY data and the goal of detecting changes in essentiality. This made the method more constrained, but also more effective. Thus, this work goes against the current trend in computer science, which often focuses on deep learning solutions as a broadly applicable default, by showing that simpler, domain-informed statistical models may be more effective when better aligned with the data's structure.

6 Conclusion

In this work, we developed two different methods to process and analyze SATAY data. First, we developed an AE model to denoise the data and impute missing values. Although both a ZINB loss and a masked reconstruction loss were necessary for good reconstruction performance, they were not sufficient for meaningful imputation. The main obstacles to successful AE-based imputation were the extreme sparsity and noise of the SATAY data. As a result, the AE was able to denoise the data to some extent, but this did not lead to improved interpretation of the results compared with the raw data.

Second, we developed a change-point detection algorithm to segment the data into regions with different essentiality levels. This algorithm performed well on both synthetic and hand-annotated data and remained robust at saturation levels above 20%. These findings indicate that our ZINB-based approach is suitable for identifying regions of differing essentiality in SATAY data and can partially address data sparsity. However, substantial noise in the data still leads to false change points and limits the segmentation precision.

Despite these limitations, the regions identified by our CPD algorithm and their corresponding essentiality levels generally align well with known biological annotations. At the same time, the strong nucleosome bias continued to affect the data distribution, even after correction, complicating the interpretation of some predicted regions. Overall, our approach enables the detection of regions with varying levels of essentiality in SATAY data and provides a promising step toward the annotation-independent identification of functionally important genomic regions beyond predefined gene boundaries.

7 Code and data availability

The original data and code developed and used to create the results in this thesis are publicly available on GitHub: https://github.com/NinaOosterlaar/Transposon_Truths. In addition, our ZINB-based CPD algorithm is available as a tool with proper documentation on GitHub: <https://github.com/NinaOosterlaar/ZINB-CPD>.

8 AI disclosure statement

In this thesis, I used AI-based tools (ChatGPT, GitHub Copilot, and Grammarly) to support writing and coding. Specifically, I used ChatGPT and Grammarly to rephrase and improve the clarity and flow of the text. ChatGPT and GitHub Copilot were used to assist in writing and debugging code. All AI-generated outputs were critically reviewed, verified, and edited by me. I take full responsibility for the accuracy, originality, and academic integrity of all content in this thesis, including any material developed with AI assistance.

References

- [1] Aashiq H. Kachroo et al. "Humanized yeast to model human biology, disease and evolution". In: *Disease Models & Mechanisms* 15.6 (June 2022). DOI: 10.1242/dmm.049309. URL: <https://doi.org/10.1242/dmm.049309>.
- [2] Carrie Baker Brachmann et al. "Designer deletion strains derived from *Saccharomyces cerevisiae* S288C: A useful set of strains and plasmids for PCR-mediated gene disruption and other applications". In: *Yeast* 14.2 (Jan. 1998), pp. 115–132. DOI: 10.1002/(sici)1097-0061(19980130)14:2%3C115::aid-yea204%3E3.0.co;2-2. URL: [https://doi.org/10.1002/\(sici\)1097-0061\(19980130\)14:2%3C115::aid-yea204%3E3.0.co;2-2](https://doi.org/10.1002/(sici)1097-0061(19980130)14:2%3C115::aid-yea204%3E3.0.co;2-2).
- [3] L H Hartwell. "Twenty-five years of cell cycle genetics." In: *Genetics* 129.4 (Dec. 1991), pp. 975–980. DOI: 10.1093/genetics/129.4.975. URL: <https://doi.org/10.1093/genetics/129.4.975>.
- [4] Valter D. Longo et al. "Replicative and Chronological Aging in *Saccharomyces cerevisiae*". In: *Cell Metabolism* 16.1 (July 2012), pp. 18–31. DOI: 10.1016/j.cmet.2012.06.002. URL: <https://doi.org/10.1016/j.cmet.2012.06.002>.
- [5] N F Lue and R D Kornberg. "Accurate initiation at RNA polymerase II promoters in extracts from *Saccharomyces cerevisiae*." In: *Proceedings of the National Academy of Sciences* 84.24 (Dec. 1987), pp. 8839–8843. DOI: 10.1073/pnas.84.24.8839. URL: <https://doi.org/10.1073/pnas.84.24.8839>.
- [6] Guri Giaever et al. "Functional profiling of the *Saccharomyces cerevisiae* genome". In: *Nature* 418.6896 (July 2002), pp. 387–391. DOI: 10.1038/nature00935. URL: <https://www.nature.com/articles/nature00935>.
- [7] Michael A. DeJesus et al. "Statistical analysis of genetic interactions in Tn-Seq data". In: *Nucleic Acids Research* 45.11 (Feb. 2017), e93. DOI: 10.1093/nar/gkx128. URL: <https://doi.org/10.1093/nar/gkx128>.
- [8] Nathaniel Charest et al. "Discovering pathways through ribozyme fitness landscapes using information theoretic quantification of epistasis". In: *RNA* 29.11 (Aug. 2023), pp. 1644–1657. DOI: 10.1261/rna.079541.122. URL: <https://doi.org/10.1261/rna.079541.122>.
- [9] Daniel M. Weinreich et al. "Darwinian evolution can follow only very few mutational paths to fitter proteins". In: *Science* 312.5770 (Apr. 2006), pp. 111–114. DOI: 10.1126/science.1123539. URL: <https://doi.org/10.1126/science.1123539>.
- [10] Aashiq H. Kachroo et al. "Systematic humanization of yeast genes reveals conserved functions and genetic modularity". In: *Science* 348.6237 (May 2015), pp. 921–925. DOI: 10.1126/science.aaa0769. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC4718922/>.
- [11] Stacia R Engel et al. "Saccharomyces Genome Database: advances in genome annotation, expanded biochemical pathways, and other key enhancements". In: *Genetics* 229.3 (Nov. 2024). DOI: 10.1093/genetics/iyae185. URL: <https://doi.org/10.1093/genetics/iyae185>.
- [12] Stacia R Engel et al. "Saccharomyces Genome Database: advances in genome annotation, expanded biochemical pathways, and other key enhancements". In: *Genetics* (Nov. 2024). DOI: 10.1093/genetics/iyae185. URL: <https://doi.org/10.1093/genetics/iyae185>.
- [13] João Rainha, Joana L. Rodrigues, and Lígia R. Rodrigues. "CRISPR-Cas9: A Powerful Tool to Efficiently Engineer *Saccharomyces cerevisiae*". In: *Life* 11.1 (Dec. 2020), p. 13. DOI: 10.3390/life11010013. URL: <https://doi.org/10.3390/life11010013>.
- [14] Amy Hin Yan Tong and Charles Boone. "Synthetic Genetic Array Analysis in *Saccharomyces cerevisiae*," in: *Humana Press eBooks* (Aug. 2005), pp. 171–192. DOI: 10.1385/1-59259-958-3:171. URL: <https://doi.org/10.1385/1-59259-958-3:171>.
- [15] Andrew M. Smith et al. "Quantitative phenotyping via deep barcode sequencing". In: *Genome Research* 19.10 (July 2009), pp. 1836–1842. DOI: 10.1101/gr.093955.109. URL: <https://doi.org/10.1101/gr.093955.109>.

- [16] Agnès H Michel et al. "Functional mapping of yeast genomes by saturated transposition". In: *eLife* 6 (May 2017). DOI: 10.7554/elife.23570. URL: <https://doi.org/10.7554/elife.23570>.
- [17] Dalin Rifat et al. "Genome-Wide Essentiality Analysis of *Mycobacterium abscessus* by Saturated Transposon Mutagenesis and Deep Sequencing". In: *mBio* 12.3 (June 2021). DOI: 10.1128/mbio.01049-21. URL: <https://doi.org/10.1128/mbio.01049-21>.
- [18] Kelly M. Wetmore et al. "Rapid quantification of mutant fitness in diverse bacteria by sequencing randomly Bar-Coded transposons". In: *mBio* 6.3 (May 2015). DOI: 10.1128/mbio.00306-15. URL: <https://doi.org/10.1128/mbio.00306-15>.
- [19] Amy K. Cain et al. "A decade of advances in transposon-insertion sequencing". In: *Nature Reviews Genetics* 21.9 (June 2020), pp. 526–540. DOI: 10.1038/s41576-020-0244-x. URL: <https://doi.org/10.1038/s41576-020-0244-x>.
- [20] Tim Van Opijnen and Andrew Camilli. "Transposon insertion sequencing: a new tool for systems-level analysis of microorganisms". In: *Nature Reviews Microbiology* 11.7 (May 2013), pp. 435–442. DOI: 10.1038/nrmicro3033. URL: <https://doi.org/10.1038/nrmicro3033>.
- [21] Arun T John Peter et al. "Rewiring phospholipid biosynthesis reveals resilience to membrane perturbations and uncovers regulators of lipid homeostasis". In: *The EMBO Journal* 41.7 (Feb. 2022). DOI: 10.15252/embj.2021109998. URL: <https://doi.org/10.15252/embj.2021109998>.
- [22] Nataliia Serbyn et al. "The aspartic protease DDI1 contributes to DNA-Protein crosslink repair in yeast". In: *Molecular Cell* 77.5 (Jan. 2020), 1066–1079.e9. DOI: 10.1016/j.molcel.2019.12.007. URL: <https://doi.org/10.1016/j.molcel.2019.12.007>.
- [23] Raffaele Nicastro et al. "Malonyl-CoA is a conserved endogenous ATP-competitive mTORC1 inhibitor". In: *Nature Cell Biology* 25.9 (Aug. 2023), pp. 1303–1318. DOI: 10.1038/s41556-023-01198-6. URL: <https://doi.org/10.1038/s41556-023-01198-6>.
- [24] Riko Hatakeyama et al. "Spatially distinct pools of TORC1 balance protein homeostasis". In: *Molecular Cell* 73.2 (Dec. 2018), 325–338.e8. DOI: 10.1016/j.molcel.2018.10.040. URL: <https://doi.org/10.1016/j.molcel.2018.10.040>.
- [25] Elodie Caudal et al. "Loss-of-function mutation survey revealed that genes with background-dependent fitness are rare and functionally related in yeast". In: *Proceedings of the National Academy of Sciences* 119.37 (Sept. 2022). DOI: 10.1073/pnas.2204206119. URL: <https://doi.org/10.1073/pnas.2204206119>.
- [26] Enzo Kingma et al. "Saturated Transposon Analysis in Yeast as a one-step method to quantify the fitness effects of gene disruptions on a genome-wide scale". In: *PLoS ONE* 20.2 (Feb. 2025), e0312437. DOI: 10.1371/journal.pone.0312437. URL: <https://doi.org/10.1371/journal.pone.0312437>.
- [27] Michael A. DeJesus et al. "Bayesian analysis of gene essentiality based on sequencing of transposon insertion libraries". In: *Bioinformatics* 29.6 (Jan. 2013), pp. 695–703. DOI: 10.1093/bioinformatics/btt043. URL: <https://doi.org/10.1093/bioinformatics/btt043>.
- [28] Michael A. DeJesus et al. "Bayesian analysis of gene essentiality based on sequencing of transposon insertion libraries". In: *Bioinformatics* 29.6 (Jan. 2013), pp. 695–703. DOI: 10.1093/bioinformatics/btt043. URL: <https://doi.org/10.1093/bioinformatics/btt043>.
- [29] Justin R. Pritchard et al. "ARTIST: High-Resolution Genome-Wide Assessment of Fitness using Transposon-Insertion sequencing". In: *PLoS Genetics* 10.11 (Nov. 2014), e1004782. DOI: 10.1371/journal.pgen.1004782. URL: <https://doi.org/10.1371/journal.pgen.1004782>.
- [30] Yanjia J. Zhang et al. "Global Assessment of Genomic Regions Required for Growth in *Mycobacterium tuberculosis*". In: *PLoS Pathogens* 8.9 (Sept. 2012), e1002946. DOI: 10.1371/journal.ppat.1002946. URL: <https://doi.org/10.1371/journal.ppat.1002946>.
- [31] Sina Solaimanpour, Felipe Sarmiento, and Jan Mrázek. "TN-SEQ Explorer: A tool for analysis of High-ThroughPut Sequencing data of Transposon Mutant Libraries". In: *PLoS ONE* 10.5 (May 2015), e0126070. DOI: 10.1371/journal.pone.0126070. URL: <https://doi.org/10.1371/journal.pone.0126070>.
- [32] Michael A. DeJesus et al. "TRANSIT - a software tool for Himar1 TNSEQ analysis". In: *PLoS Computational Biology* 11.10 (Oct. 2015), e1004401. DOI: 10.1371/journal.pcbi.1004401. URL: <https://doi.org/10.1371/journal.pcbi.1004401>.

- [33] Michael C. Chao et al. "High-resolution definition of the *Vibrio cholerae* essential gene set with hidden Markov model-based analyses of transposon-insertion sequencing data". In: *Nucleic Acids Research* 41.19 (July 2013), pp. 9033–9048. DOI: 10.1093/nar/gkt654. URL: <https://doi.org/10.1093/nar/gkt654>.
- [34] Aldert Zomer et al. "ESSENTIALS: Software for rapid analysis of high throughput transposon insertion sequencing data". In: *PLoS ONE* 7.8 (Aug. 2012), e43012. DOI: 10.1371/journal.pone.0043012. URL: <https://doi.org/10.1371/journal.pone.0043012>.
- [35] Najmeh Abiri et al. "Establishing strong imputation performance of a denoising autoencoder in a wide range of missing data problems". In: *Neurocomputing* 365 (July 2019), pp. 137–146. DOI: 10.1016/j.neucom.2019.07.065. URL: <https://doi.org/10.1016/j.neucom.2019.07.065>.
- [36] Ricardo Cardoso Pereira et al. "Reviewing Autoencoders for missing data Imputation: Technical trends, applications and Outcomes". In: *Journal of Artificial Intelligence Research* 69 (Dec. 2020), pp. 1255–1285. DOI: 10.1613/jair.1.12312. URL: <https://doi.org/10.1613/jair.1.12312>.
- [37] Tim Van Opijnen, Kip L Bodi, and Andrew Camilli. "Tn-seq: high-throughput parallel sequencing for fitness and genetic interaction studies in microorganisms". In: *Nature Methods* 6.10 (Sept. 2009), pp. 767–772. DOI: 10.1038/nmeth.1377. URL: <https://doi.org/10.1038/nmeth.1377>.
- [38] Gemma C. Langridge et al. "Simultaneous assay of every *Salmonella* Typhi gene using one million transposon mutants". In: *Genome Research* 19.12 (Oct. 2009), pp. 2308–2316. DOI: 10.1101/gr.097097.109. URL: <https://doi.org/10.1101/gr.097097.109>.
- [39] Andrew L. Goodman et al. "Identifying genetic determinants needed to establish a human gut symbiont in its habitat". In: *Cell Host & Microbe* 6.3 (Sept. 2009), pp. 279–289. DOI: 10.1016/j.chom.2009.08.003. URL: <https://doi.org/10.1016/j.chom.2009.08.003>.
- [40] Jeffrey D. Gawronski et al. "Tracking insertion mutants within libraries by deep sequencing and a genome-wide screen for *Haemophilus* genes required in the lung". In: *Proceedings of the National Academy of Sciences* 106.38 (Sept. 2009), pp. 16422–16427. DOI: 10.1073/pnas.0906627106. URL: <https://doi.org/10.1073/pnas.0906627106>.
- [41] Michael C. Chao et al. "The design and analysis of transposon insertion sequencing experiments". In: *Nature Reviews Microbiology* 14.2 (Jan. 2016), pp. 119–128. DOI: 10.1038/nrmicro.2015.7. URL: <https://www.nature.com/articles/nrmicro.2015.7>.
- [42] Kathryn A. Coe et al. "Multi-strain Tn-Seq reveals common daptomycin resistance determinants in *Staphylococcus aureus*". In: *PLoS Pathogens* 15.11 (Nov. 2019), e1007862. DOI: 10.1371/journal.ppat.1007862. URL: <https://doi.org/10.1371/journal.ppat.1007862>.
- [43] Luchang Zhu et al. "Novel Genes Required for the Fitness of *Streptococcus pyogenes* in Human Saliva". In: *mSphere* 2.6 (Nov. 2017). DOI: 10.1128/mspheredirect.00460-17. URL: <https://doi.org/10.1128/mspheredirect.00460-17>.
- [44] Samuel J. Willcocks et al. "High-throughput analysis of *Yersinia pseudotuberculosis* gene essentiality in optimised in vitro conditions, and implications for the speciation of *Yersinia pestis*". In: *BMC Microbiology* 18.1 (May 2018), p. 46. DOI: 10.1186/s12866-018-1189-5. URL: <https://doi.org/10.1186/s12866-018-1189-5>.
- [45] Adam Deutschbauer et al. "Towards an informative mutant phenotype for every bacterial gene". In: *Journal of Bacteriology* 196.20 (Aug. 2014), pp. 3643–3655. DOI: 10.1128/jb.01836-14. URL: <https://doi.org/10.1128/jb.01836-14>.
- [46] Yabin Guo et al. "Integration profiling of gene function with dense maps of transposon integration". In: *Genetics* 195.2 (July 2013), pp. 599–609. DOI: 10.1534/genetics.113.152744. URL: <https://doi.org/10.1534/genetics.113.152744>.
- [47] Changyi Zhang et al. "The essential genome of the crenarchaeal model *Sulfolobus islandicus*". In: *Nature Communications* 9.1 (Nov. 2018), p. 4908. DOI: 10.1038/s41467-018-07379-4. URL: <https://doi.org/10.1038/s41467-018-07379-4>.
- [48] Felipe Sarmiento, Jan Mrázek, and William B. Whitman. "Genome-scale analysis of gene function in the hydrogenotrophic methanogenic archaeon *Methanococcus maripaludis*". In: *Proceedings of the National Academy of Sciences* 110.12 (Mar. 2013), pp. 4726–4731. DOI: 10.1073/pnas.1220225110. URL: <https://doi.org/10.1073/pnas.1220225110>.

- [49] Jan E. Carette et al. “Haploid genetic screens in human cells identify host factors used by pathogens”. In: *Science* 326.5957 (Nov. 2009), pp. 1231–1235. DOI: 10.1126/science.1178955. URL: <https://doi.org/10.1126/science.1178955>.
- [50] Erik Vollbrecht et al. “Genome-Wide Distribution of Transposed Dissociation Elements in Maize”. In: *The Plant Cell* 22.6 (June 2010), pp. 1667–1685. DOI: 10.1105/tpc.109.073452. URL: <https://doi.org/10.1105/tpc.109.073452>.
- [51] Agnès H. Michel et al. “Exploiting homologous recombination increases SATAY efficiency for loss- and gain-of-function screening”. In: *bioRxiv (Cold Spring Harbor Laboratory)* (Dec. 2019). DOI: 10.1101/866483. URL: <https://doi.org/10.1101/866483>.
- [52] A. S. M. Zisanur Rahman et al. “Identification of putative essential protein domains from high-density transposon insertion sequencing”. In: *Scientific Reports* 12.1 (Jan. 2022), p. 962. DOI: 10.1038/s41598-022-05028-x. URL: <https://doi.org/10.1038/s41598-022-05028-x>.
- [53] Jennifer E. Griffin et al. “High-Resolution phenotypic profiling defines genes essential for mycobacterial growth and cholesterol catabolism”. In: *PLoS Pathogens* 7.9 (Sept. 2011), e1002251. DOI: 10.1371/journal.ppat.1002251. URL: <https://doi.org/10.1371/journal.ppat.1002251>.
- [54] Diederik P Kingma and Max Welling. *Auto-Encoding variational Bayes*. Dec. 2013. URL: <https://arxiv.org/abs/1312.6114>.
- [55] Ruoqi Wei et al. “Variations in Variational Autoencoders - A Comparative evaluation”. In: *IEEE Access* 8 (Jan. 2020), pp. 153651–153670. DOI: 10.1109/access.2020.3018151. URL: <https://doi.org/10.1109/access.2020.3018151>.
- [56] Pascal Vincent et al. “Extracting and Composing Robust Features with Denoising Autoencoders”. In: *Proceedings of the 25th International Conference on Machine Learning (ICML)*. ACM, 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294. URL: <https://doi.org/10.1145/1390156.1390294>.
- [57] David Lähnemann et al. “Eleven grand challenges in single-cell data science”. In: *Genome biology* 21.1 (Feb. 2020). DOI: 10.1186/s13059-020-1926-6. URL: <https://doi.org/10.1186/s13059-020-1926-6>.
- [58] Gökçen Eraslan et al. “Single-cell RNA-seq denoising using a deep count autoencoder”. In: *Nature Communications* 10.1 (Jan. 2019). DOI: 10.1038/s41467-018-07931-2. URL: <https://doi.org/10.1038/s41467-018-07931-2>.
- [59] Romain Lopez et al. “Deep generative modeling for single-cell transcriptomics”. In: *Nature Methods* 15.12 (Nov. 2018), pp. 1053–1058. DOI: 10.1038/s41592-018-0229-2. URL: <https://doi.org/10.1038/s41592-018-0229-2>.
- [60] Ye Zheng, Siqi Shen, and Sündüz Keleş. “Normalization and de-noising of single-cell Hi-C data with BandNorm and scVI-3D”. In: *Genome biology* 23.1 (Oct. 2022). DOI: 10.1186/s13059-022-02774-z. URL: <https://doi.org/10.1186/s13059-022-02774-z>.
- [61] Yingxin Cao et al. “SAILER: scalable and accurate invariant representation learning for single-cell ATAC-seq processing and integration”. In: *Bioinformatics* 37.Supplement_1 (Apr. 2021), pp. i317–i326. DOI: 10.1093/bioinformatics/btab303. URL: <https://doi.org/10.1093/bioinformatics/btab303>.
- [62] Lei Xiong et al. “SCALE method for single-cell ATAC-seq analysis via latent feature extraction”. In: *Nature Communications* 10.1 (Oct. 2019). DOI: 10.1038/s41467-019-12630-7. URL: <https://doi.org/10.1038/s41467-019-12630-7>.
- [63] A. B. Olshen et al. “Circular binary segmentation for the analysis of array-based DNA copy number data”. In: *Biostatistics* 5.4 (Oct. 2004), pp. 557–572. DOI: 10.1093/biostatistics/kxh008. URL: <https://doi.org/10.1093/biostatistics/kxh008>.
- [64] Zhongyang Zhang et al. “Reconstructing DNA copy number by penalized estimation and imputation”. In: *The Annals of Applied Statistics* 4.4 (Dec. 2010), pp. 1749–1773. DOI: 10.1214/10-aas357. URL: <https://doi.org/10.1214/10-aas357>.
- [65] Jerome V. Braun and Hans-Georg Müller. “Statistical methods for DNA sequence segmentation”. In: *Statistical Science* 13.2 (May 1998). DOI: 10.1214/ss/1028905933. URL: <https://doi.org/10.1214/ss/1028905933>.

- [66] Alice Cleynen, Sandrine Dudoit, and Stéphane Robin. “Comparing segmentation methods for genome annotation based on RNA-SeQ data”. In: *Journal of Agricultural Biological and Environmental Statistics* 19.1 (Oct. 2013), pp. 101–118. DOI: 10.1007/s13253-013-0159-5. URL: <https://doi.org/10.1007/s13253-013-0159-5>.
- [67] Alice Cleynen et al. “Segmentor3lsBack: an R package for the fast and exact segmentation of Seq-data”. In: *Algorithms for Molecular Biology* 9.1 (Jan. 2014), p. 6. DOI: 10.1186/1748-7188-9-6. URL: <https://doi.org/10.1186/1748-7188-9-6>.
- [68] Steven M. Kay. *Fundamentals of Statistical Signal Processing: Detection theory*. Pearson, Jan. 1998.
- [69] Charles Truong, Laurent Oudre, and Nicolas Vayatis. “Selective review of offline change point detection methods”. In: *Signal Processing* 167 (Sept. 2019), p. 107299. DOI: 10.1016/j.sigpro.2019.107299. URL: <https://doi.org/10.1016/j.sigpro.2019.107299>.
- [70] Arnaud Liehrmann et al. “DiffSegR: an RNA-seq data driven method for differential expression analysis using changepoint detection”. In: *NAR Genomics and Bioinformatics* 5.4 (Oct. 2023), lqad098. DOI: 10.1093/nargab/lqad098. URL: <https://doi.org/10.1093/nargab/lqad098>.
- [71] Alice Cleynen and Emilie Lebarbier. “Segmentation of the Poisson and negative binomial rate models: a penalized estimator”. In: *arXiv (Cornell University)* (Jan. 2013). DOI: 10.48550/arxiv.1301.2534. URL: <http://arxiv.org/abs/1301.2534>.
- [72] Alice Cleynen, Sandrine Dudoit, and Stéphane Robin. “Comparing segmentation methods for genome annotation based on RNA-SeQ data”. In: *Journal of Agricultural Biological and Environmental Statistics* 19.1 (Oct. 2013), pp. 101–118. DOI: 10.1007/s13253-013-0159-5. URL: <https://doi.org/10.1007/s13253-013-0159-5>.
- [73] R. Killick, P. Fearnhead, and I. A. Eckley. “Optimal detection of changepoints with a linear computational cost”. In: *Journal of the American Statistical Association* 107.500 (Oct. 2012), pp. 1590–1598. DOI: 10.1080/01621459.2012.737745. URL: <https://doi.org/10.1080/01621459.2012.737745>.
- [74] Robert Tibshirani et al. “Sparsity and smoothness via the fused lasso”. In: *Journal of the Royal Statistical Society Series B (Statistical Methodology)* 67.1 (Dec. 2004), pp. 91–108. DOI: 10.1111/j.1467-9868.2005.00490.x. URL: <https://doi.org/10.1111/j.1467-9868.2005.00490.x>.
- [75] Toby Dylan Hocking et al. “Learning smoothing models of copy number profiles using breakpoint annotations”. In: *BMC Bioinformatics* 14.1 (May 2013), p. 164. DOI: 10.1186/1471-2105-14-164. URL: <https://doi.org/10.1186/1471-2105-14-164>.
- [76] Kurt S. Riedel. “Detection of Abrupt Changes: Theory and application”. In: *Technometrics* 36.3 (Aug. 1994), pp. 326–327. DOI: 10.1080/00401706.1994.10485821. URL: <https://doi.org/10.1080/00401706.1994.10485821>.
- [77] Kuo-Mei Chen, Arthur Cohen, and Harold Sackrowitz. “Consistent multiple testing for change points”. In: *Journal of Multivariate Analysis* 102.10 (May 2011), pp. 1339–1343. DOI: 10.1016/j.jmva.2011.04.012. URL: <https://doi.org/10.1016/j.jmva.2011.04.012>.
- [78] E. Keogh et al. “An online algorithm for segmenting time series”. In: *IEEE Xplore* (Nov. 2002), pp. 289–296. DOI: 10.1109/icdm.2001.989531. URL: <https://doi.org/10.1109/icdm.2001.989531>.
- [79] Kristin Brogaard et al. “A map of nucleosome positions in yeast at base-pair resolution”. In: *Nature* 486.7404 (June 2012), pp. 496–501. DOI: 10.1038/nature11142. URL: <https://www.nature.com/articles/nature11142>.
- [80] Siddharth Subramaniam et al. “Statistical analysis of variability in TnSeq data across conditions using zero-inflated negative binomial regression”. In: *BMC Bioinformatics* 20.1 (Nov. 2019). DOI: 10.1186/s12859-019-3156-z. URL: <https://doi.org/10.1186/s12859-019-3156-z>.
- [81] Fernando Nogueira. *Bayesian Optimization: Open source constrained global optimization tool for Python*. 2014–. URL: <https://github.com/bayesian-optimization/BayesianOptimization>.
- [82] Samaneh Aminikhanghahi and Diane J. Cook. “A survey of methods for time series change point detection”. In: *Knowledge and Information Systems* 51.2 (Sept. 2016), pp. 339–367. DOI: 10.1007/s10115-016-0987-z. URL: <https://doi.org/10.1007/s10115-016-0987-z>.

-
- [83] Guo-Cheng Yuan et al. “Genome-Scale Identification of Nucleosome Positions in *S. cerevisiae*”. In: *Science* 309.5734 (June 2005), pp. 626–630. DOI: 10.1126/science.1112178. URL: <https://doi.org/10.1126/science.1112178>.
- [84] Diederik P. Kingma and Jimmy Ba. *Adam: A method for stochastic optimization*. Dec. 2014. URL: <https://arxiv.org/abs/1412.6980>.

A Dataset description

Table A.1: Description of the different strains and their corresponding datasets. The datasets in the third column are all biological replicates of each other. In the replicates column, it further elaborates whether the different datasets are technical or biological replicates.

Strain	Strain description	Dataset	Replicates
dnrp	NRP1 knockout	dnrp1-1 dnrp1-2	technical replicates a–b technical replicates a–b
FD	Wild-type	FD7 FD9 FD11 FD12	technical replicates 1-6 technical replicates 1-3 1 replicate 1 replicate
yEK19	Wild-type	yEK19a yEK19b yEK19c	biological replicates 6–8 biological replicate 12 biological replicates 7, 9
yEK23	BEM1, BEM3, NRP1 knock-out, modification CAN1	yEK23a yEK23b yEK23c	biological replicates 23, 27, 32 biological replicates 1–2 biological replicate 10
yLIC137	unknown	yLIC137	biological replicates 7–8
yTW001	unknown	yTW001	biological replicates 4, 6
yWT03a	Bem1-AID construct	yWT03a	biological replicates 16, 21
yWT04a	BEM3 knockout, Bem1-AID construct	yWT04a	biological replicates 14, 23

B Dataset saturation

Table B.1: Comparison of transposon insertion densities for uncombined, technical-replicate combined, and strain-combined datasets.

Strain	Sample	Saturation: (Uncombined)	(Tech. Combined)	(Strain Combined)
FD	FD7_1	7,94%	17,27%	33,95%
	FD7_2	7,76%	17,27%	33,95%
	FD7_3	7,81%	17,27%	33,95%
	FD7_4	7,87%	17,27%	33,95%
	FD7_5	8,34%	17,27%	33,95%
	FD7_6	8,08%	17,27%	33,95%
	FD9_1	9,13%	14,97%	33,95%
	FD9_2	8,75%	14,97%	33,95%
	FD9_3	9,49%	14,97%	33,95%
	FD11	8,32%	8,31%	33,95%
	FD12	9,63%	9,62%	33,95%
dnrp	dnrp1-1-a	3,84%	5,38%	9,21%
	dnrp1-1-b	3,82%	5,38%	9,21%
	dnrp1-2-a	4,20%	5,60%	9,21%
	dnrp1-2-b	4,20%	5,60%	9,21%
yEK19	yEK19a_6	7,78%	7,78%	32,03%
	yEK19a_7	8,48%	8,47%	32,03%
	yEK19a_8	8,59%	8,59%	32,03%
	yEK19b_12	9,80%	9,79%	32,03%
	yEK19c_7	8,86%	8,85%	32,03%
	yEK19c_9	8,41%	8,40%	32,03%
yEK23	yEK23a_23	5,43%	5,42%	29,27%
	yEK23a_27	9,16%	9,15%	29,27%
	yEK23a_32	7,38%	7,38%	29,27%
	yEK23b_1	8,19%	8,19%	29,27%
	yEK23b_2	8,70%	8,69%	29,27%
	yEK23c_10	7,98%	7,98%	29,27%
yTW001	yTW001_4	5,62%	5,61%	9,82%
	yTW001_6	5,54%	5,53%	9,82%
yWT03a	yWT03a_16	1,30%	1,30%	2,86%
	yWT03a_21	1,78%	1,78%	2,86%
yWT04a	yWT04a_14	2,86%	2,86%	5,97%
	yWT04a_23	3,61%	3,60%	5,97%
yLIC137	yLIC137_7	4,92%	4,92%	9,15%
	yLIC137_8	5,31%	5,30%	9,15%

C Hyperparameters

Table C.1: Hyperparameters and their descriptions for the AE model. Tested values are the values that have been tested.

Category	Hyperparameter	Description	Tested values
Preprocessing	Features	The features to include with the input	Nucl, Centr
	Step size	Step size of the sliding window	0.25-1.0
	Bin size	Number of positions in each bin	1-50
	Moving Average	Apply moving average with bin size	True, False
Model Architecture	Number of layers	Number of layers for the encoder/de-coder	1-4
	First layer size	The size of the first layer.	64-1600
Convolutional Layer	Use Conv	Whether to apply a Conv1D layer before the encoder	True / False
	Conv Channels	Number of output channels of the convolutional layer	16-128
	Pool Size	Pooling kernel size	2-8
	Kernel Size	Size of the convolution kernel	3, 5, 7, 9
Training	Epochs	Number of epochs the model is trained	30-150
	Batch size	Number of data points in a batch	32, 64, 128
	Learning rate	Step size of the optimizer	10^{-5} - 10^{-2}
	Reconstruction Loss	Weight of the reconstruction loss	10^{-3} -10
Regularization	Dropout	Dropout rate	0.0-0.5
	Regularizer	Which type of regularizer to use	l1, l2, None
	Regularizer strength	Weight of the regularization loss	10^{-5} -100

The preprocessing step includes 4 hyperparameters in the search. The first one is which features to include. The only known relevant features are the distance to the centromere and the distance to the closest nucleosome. Including these features can either provide the model with more context for reconstruction or unnecessarily increase the data's dimensionality. The second feature is the step size, which determines the overlap between different data samples. The range is set to 0.25-1.0, where a lower step size corresponds to more data samples but also more repetition between them. The lower bound is set to 0.25 to prevent the dataset size from exploding. The third and fourth features are the bin size, which determines the number of data points averaged together, and whether a moving average is used to compute each data point or if the data is separated into independent bins. The bin size range is 1 to 20, with 20 chosen as the maximum bin size, since a higher bin size would impact the resolution of the data too much.

The model architecture includes two hyperparameters in the search. The first one is the number of layers, which is set between 1-4. In the literature [36], the number of layers of (denoising) autoencoders ranges from 1 to 13. However, the most common value is 3, and values higher than that are not standard. The second hyperparameter is the size of the first layer. In autoencoders designed for scRNA-seq data, 64 is a common layer size [58, 59]. However, since my data samples are high-dimensional and larger layer sizes are common in other applications, we set the range to 64-1600.

For the convolutional layer, we included 4 hyperparameters in the search. The first one tests whether using a convolutional layer is beneficial at all. The other 3 hyperparameters, the number of convolutional channels, pool size, and kernel size, are set to a wide range of commonly used values.

The model's training configuration has 4 hyperparameters to search over. In the literature [36], the number of epochs used for autoencoders ranges from 50 to 10000. However, in initial experiments, we trained the model for 100 epochs, and it did not improve further in the last epochs. Thus, we chose to set the number of epochs between 30 and 150. The batch size is set to either 32, 64, or 128, and the learning rate is set between 10^{-5} - 10^{-2} , since those are the values used in literature [36]. The weight of the reconstruction loss is set between 10^{-3} – 10, which we think is a large enough range to find the optimal value.

Three regularization hyperparameters are included in the search. In neural networks, dropout randomly masks a fraction of neurons during training to reduce overfitting and improve generalization. For

autoencoders, the dropout rate is generally set between 0% and 50% [36]. There are two regularizers that are most often used in deep learning: L1 (lasso) and L2 (ridge). Both regularizers are tested, as well as using no regularizer. The weight of the regularizer is set between $10^{-5} - 100$. This is a large range, which is necessary for regularizers, since the optimal weight is problem-specific.

Table C.2: Fixed hyperparameters and their descriptions for the AE model.

Category	Hyperparameter	Description	Value
Preprocessing	Sample size	Number of positions in one data sample	2000
Model Architecture	Model type	The model architecture used	ZINBAE
Convolutional Layer	Padding	The padding around the input	same
	Stride	Step size of convolutional kernel	1
Training	noise level	Number of data points masked as zero-value	0.15
	Optimizer	The optimizer used to train the model	Adam

Some of the hyperparameters we kept fixed (table C.2). The sample size, meaning the number of data points included in one data sample, was kept fixed at 2000. We chose 2000, since a gene in *S. cerevisiae* is on average around 1500 bp long. By choosing a sample size slightly larger than 1500, we hope to include enough context for a model to learn meaningful patterns. The model type was set to be ZINBAE, since it was the only model that showed potential in the initial experiments. Padding and stride control the spatial resolution within the convolutional layer. They are kept fixed to preserve consistent tensor dimensions across layers, and it is not expected that they have a significant impact on reconstruction performance compared to other hyperparameters. The noise level, which determines the percentage of values being masked to zero to push the model to imputation, is set to 15%. Since masking values make reconstruction inherently more difficult, BHO will push it to the smallest possible value, making it unnecessary to include it in the hyperparameter search. After the hyperparameter search, the influence of the noise level is tested separately on the most optimal model. The Adam algorithm is chosen as optimizer, since it is currently the most common optimizer used in deep learning due to its effectiveness [84].

D Reconstruction Accuracy

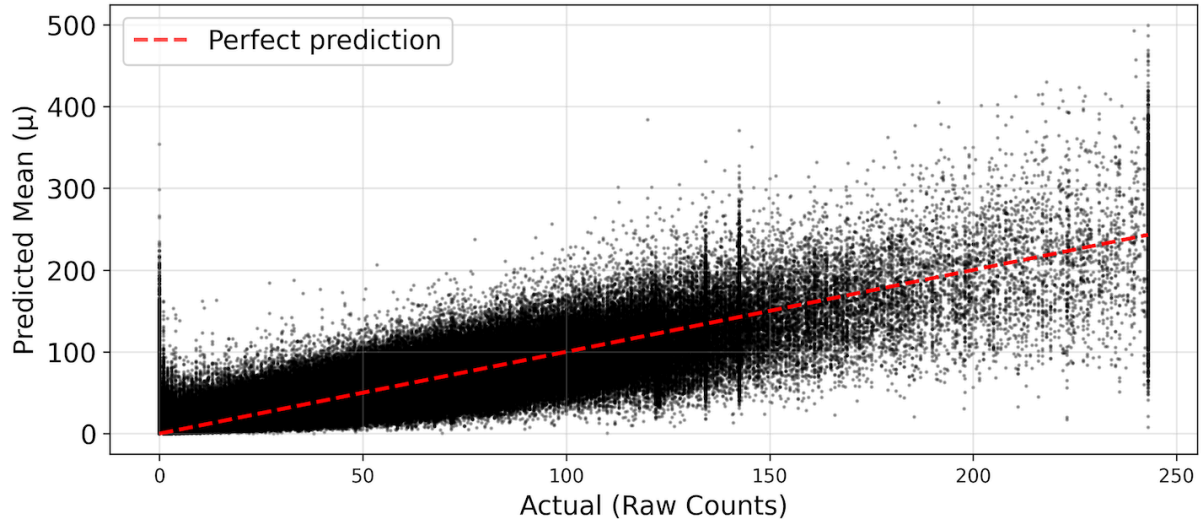


Figure D.1: Predicted value for the mean μ against the original count value of the test set for the combined model using a moving average.

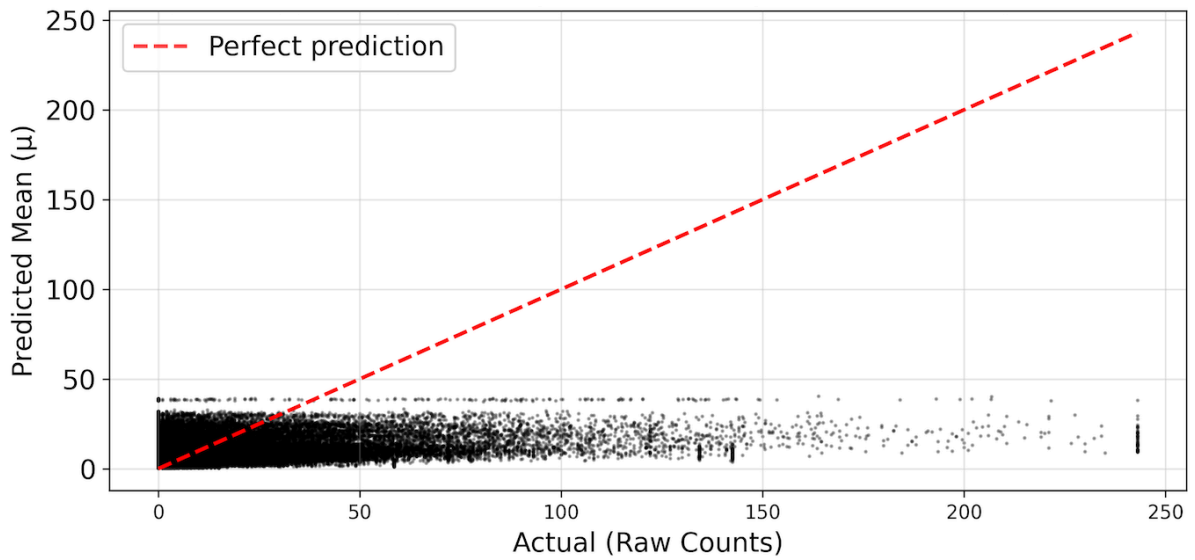


Figure D.2: Predicted value for the mean μ against the original count value of the test set for the combined model using bins.

E Noise Level

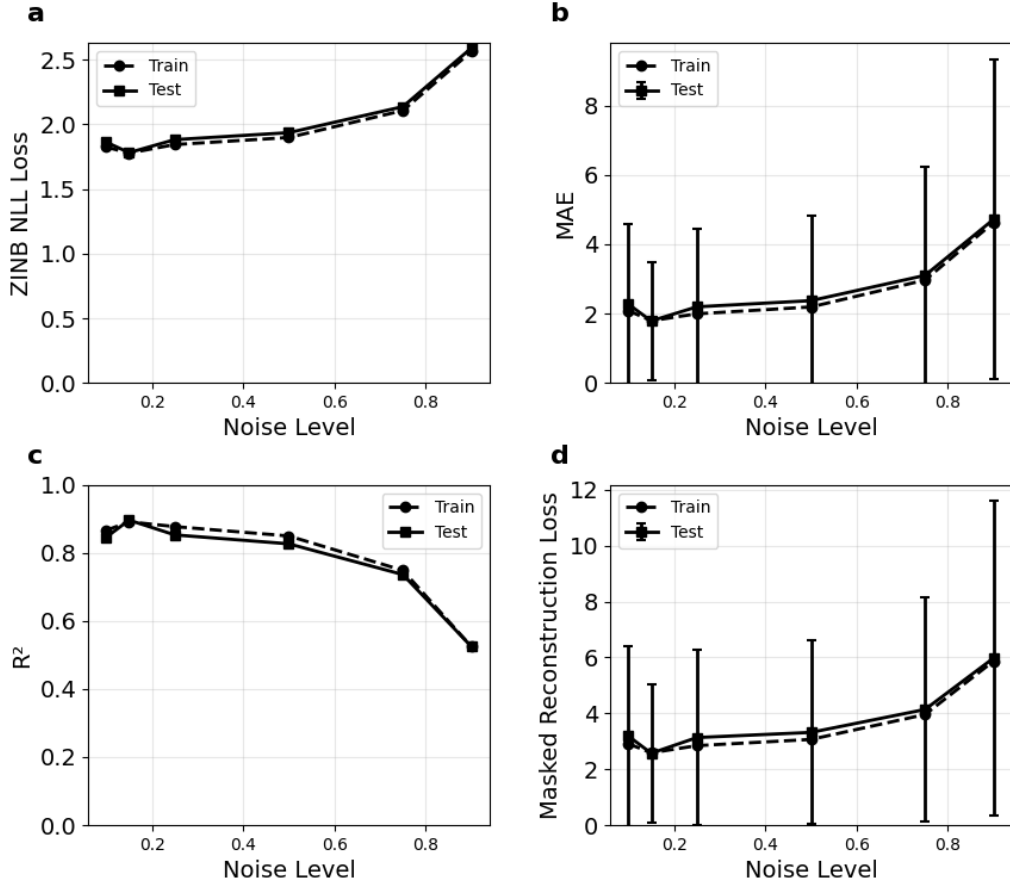


Figure E.1: The **a** ZINB NLL loss, **b** MAE, **c** R^2 , and **d** masked reconstruction loss for different noise levels. The models were trained and tested on these noise levels.

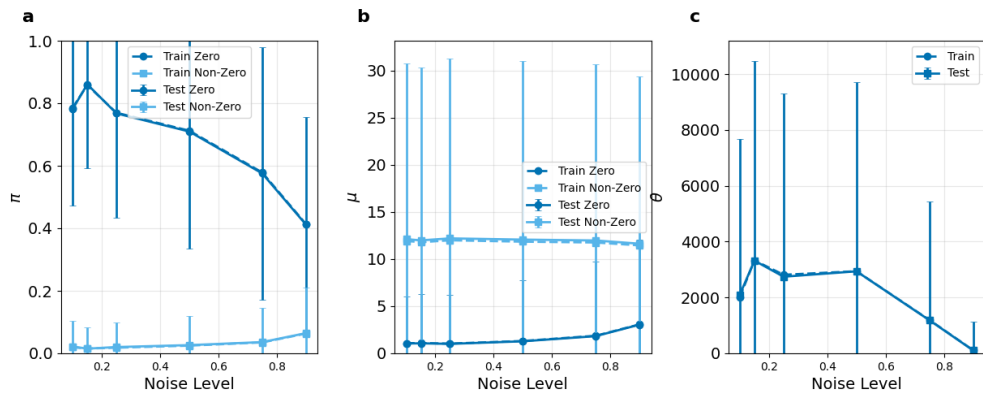


Figure E.2: The average **a** zero-inflation value π for zero and non-zero values, **b** mean value μ for zero and non-zero values, **c** and dispersion parameter θ for different noise levels. The models were trained and tested on these noise levels.

F ZINB parameter estimation accuracy

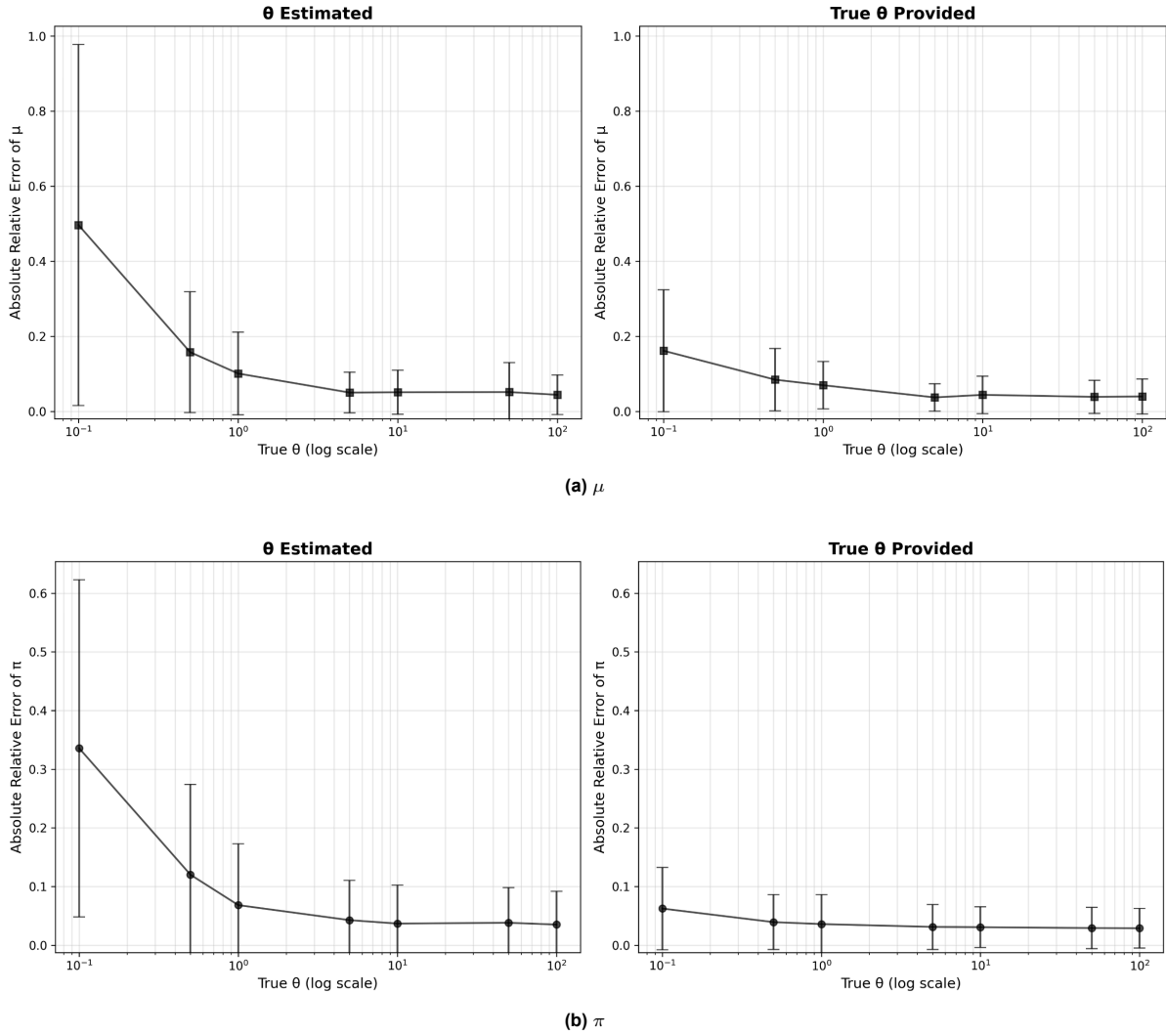


Figure F.1: Relative error of the estimation using our ZINB EM algorithm of **a** μ with an estimated value of θ , **b** μ with a provided value of θ , **c** π with an estimated value of θ , **d** π with a provided value of θ .

G Segment Lengths

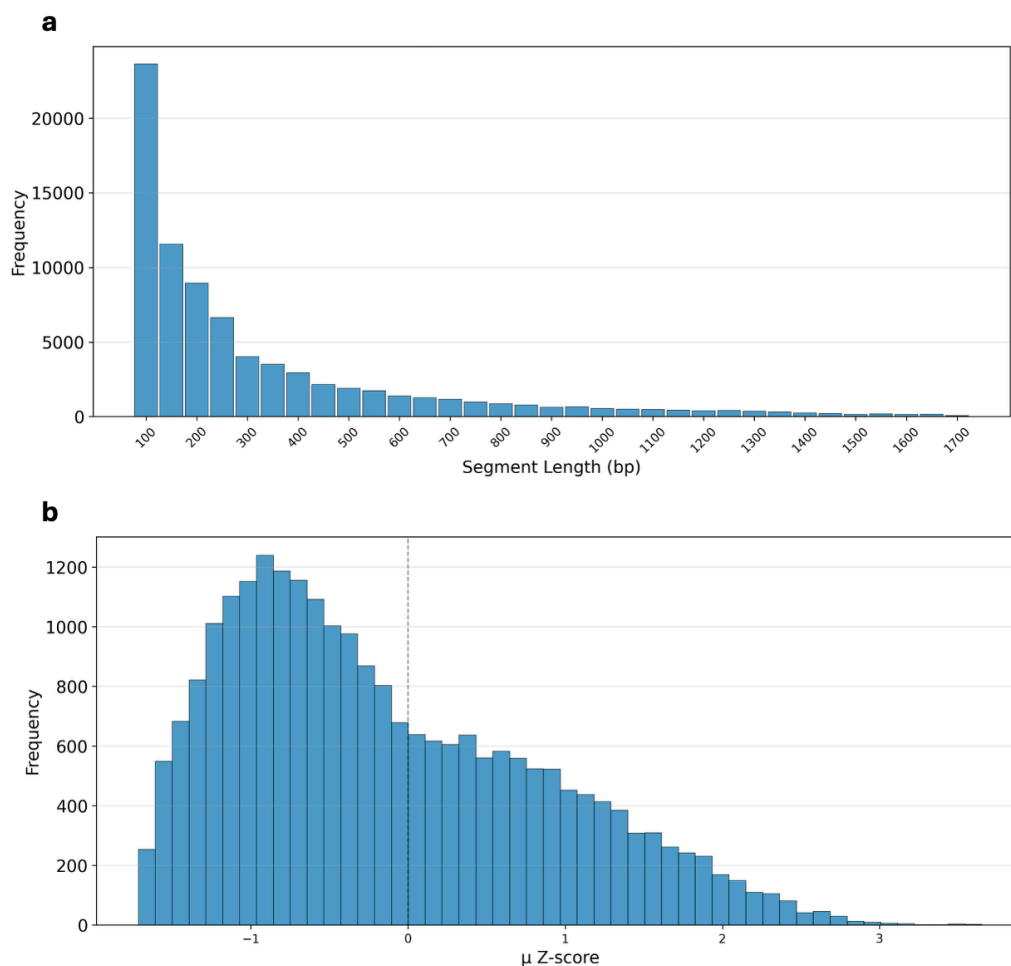


Figure G.1: **a** Histogram showing the distribution of segment length of the FD, yEK23, and yEK19 datasets combined. **b** Distribution of the estimated essentiality levels of segments with a region length equal to 100, the minimal region length.