CRISPR-Decryptr reveals cis-regulatory elements from noncoding perturbation screens

Anders Rasmussen[1,*], Tarmo Äijö[1,*], Mariano Ignacio Gabitto[1], Nicholas Carriero[3], Neville Sanjana[4], Jane Skok[5], Richard Bonneau[1,2,5,**]

[1] Center for Computational Biology, Flatiron Institute, Simons Foundation, New York, NY 10010, USA.
[2] New York University, Center for Data Science, New York, NY 10010, USA.
[3] Scientific Computing Core, Flatiron Institute, Simons Foundation, New York, NY, 10010, USA.
[4] New York Genome Center, New York, NY, 10013, USA.
[5] New York University, Department of Biology, New York, NY 10012, USA
* These authors contributed equally to this work
** To whom correspondence should be addressed

Abstract:

Clustered Regularly Interspace Short Palindromic Repeats (CRISPR)-Cas9 genome editing methods provide the tools necessary to examine phenotypic impacts of targeted perturbations in high-throughput screens. While these technologies have the potential to reveal functional elements with direct therapeutic applications, statistical techniques to analyze noncoding screen data remain limited. We present CRISPR-Decryptr, a computational tool for the analysis of CRISPR noncoding screens. Our method leverages experimental design: accounting for multiple conditions, controls, and replicates to infer the regulatory landscape of noncoding genomic regions. We validate our method on a variety of mutagenesis, CRISPR activation, and CRISPR interference screens, extracting new insights from previously published data.

Main:

Information garnered from pooled CRISPR perturbation screens impacts decisions that have therapeutic implications. Genome-wide knockout and noncoding screens have been used to identify new therapeutic targets, to reveal genes responsible for anti-cancer drug resistance, and to map functional elements in leukemia cell lines.[1,2,3,4] As researchers in academia and industry make greater use of improving gene editing technologies, computational approaches that tackle the unique challenges posed by their experimental design are of increasing importance. Methods employed for knockout screens are designed to assess the impact of perturbing a genome-wide set of pre-delineated coding regions. [5,6] However, analysis of CRISPR noncoding screens, which employ saturated guide libraries to reveal *cis*-regulatory elements, necessitate distinct experimental considerations. Most importantly, classification of functional elements without *a priori* knowledge of their location or size requires integrating information across perturbations within genomic proximity, an aspect that renders existing knockout methods inapplicable to these experimental designs. Literature on methods for analyzing noncoding screens is scarce, with only a single method published that addresses one of the many aspects of noncoding screen analysis.[7]

CRISPR-Decryptr utilizes techniques from Bayesian inference, signal processing, and latent variable models to integrate data and experimental design, allowing the end-user to make precise conclusions about their noncoding screen results (**Figure 1**; *Methods*). A Bayesian hierarchical generalized linear model (GLM) serves as the mathematical formulation from which perturbation-specific effect on phenotype are inferred[8, 9]. The model leverages experimental

conditions, controls, and replicates in a single numerical procedure implemented with Markov Chain Monte Carlo, allowing for rigorous statistical treatment of parameter uncertainty (*Methods 2.2*). Effects are mapped to a base-by-base level of granularity through a Gaussian process-based model (*Methods 2.4*)[10]. This deconvolution fully accounts for guide-specificity, off-target effects and, if applicable, double-strand break (DSB) repair uncertainty (*Methods 2.3*)[11, 12]. A hidden semi-Markov model (HsMM) incorporates spatial information to decode the latent regulatory landscape of interest, revealing enhancers and silencers in the noncoding genome (*Methods 2.5*)[13]. Regulatory element calls and guide-specific effects are exported in bioinformatics file formats such as Browser Extendable Data (.bed) and Wiggle (.wig) that can easily be explored in genomic visualization software such as the Integrative Genomics Viewer (IGV)[14].

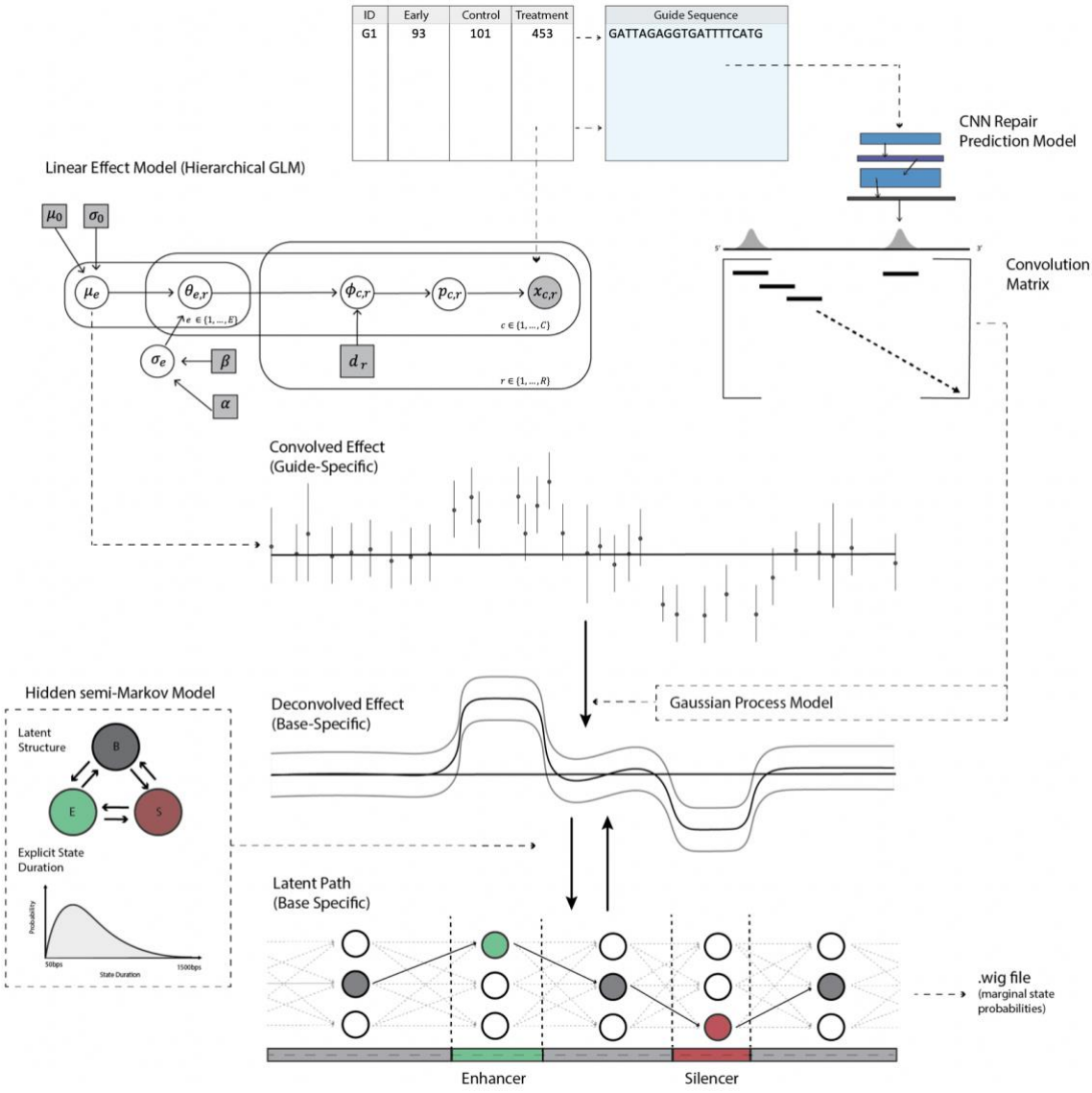


Figure 1: Overview of the CRISPR-Decryptr method for the analysis of noncoding screens. The hierarchical GLM infers pertubation-specific regulatory effect on phenotype from raw guide RNA (gRNA) counts (*top left*). Guide RNA sequences are used to construct a convolution matrix accounting for specificity, off-target effects, and repair uncertainty in the case of mutagenesis screens (*top right*). Finally, iterating between Gaussian Process deconvolution and HsMM training and prediction reveals base-specific effects and ultimately the latent state path of interest (*bottom half*).

We validated CRISPR-Decryptr on noncoding screens of distinct experimental designs, including CRISPR mutagenesis, CRISPR activation (CRISPRa), and CRISPR interference (CRISPRi) screens (**Figure 2**) [4, 15, 16]. In the CRISPR mutagenesis screen (Canver *et al.*), three intronic DNAse hypersensitivity sites (DHS) within *BCL11A* were perturbed in human umbilical cord blood-derived erythroid progenitor (HUDEP) cells[15]. These sites, termed DHS +62, +58, and +55, are known to impact fetal hemoglobin (HgF) levels from prior published research, with the enhancer identified in DHS +58 having proven a successful therapeutic target in two patients with hemoglobinopathies.[17] When applied to this dataset, CRISPR-Decryptr produced regulatory state calls in agreement with the original analysis (**Figure 2A** and **Supplementary Figure 3.1.1**). The CRISPR activation screen we re-analyzed (Simionov *et al.*) targeted the *IL2RA* and *CD69* gene loci in Jurkat T-cells.[16] To measure phenotypic change, the FACS sort cells into a "negative", "low", "medium", and "high" bins of IL2RA and CD69 based on expression levels. Analysis of the two gene loci with CRISPR-Decryptr recalls the enhancers from the original analysis, as well as novel putative enhancers that are correlated with DNAse-seq and H3K27ac from the Jurkat-T Cell line (**Supplementary Figures 3.2.2** and **3.2.3**). Finally, the re-analysis of the Fulco et al. CRISPRi screen of the GATA1 gene loci revealed similar regulatory element calls to the original analysis.[4] (**Figure 2C** and **Supplementary Figure 4.3.1**).

We have described a statistical technique for analyzing CRISPR noncoding screen data and illustrated the accuracy of CRISPR-Decryptr on three distinct perturbation technologies, demonstrating the method's ability to reveal novel insights from a diverse set of experimental designs. CRISPR-Decryptr will be a valuable component in future attempts to identify functional

genomic elements and their link to phenotypic traits, enabling target identification and synthetic biology in biomedical and biotechnological settings.

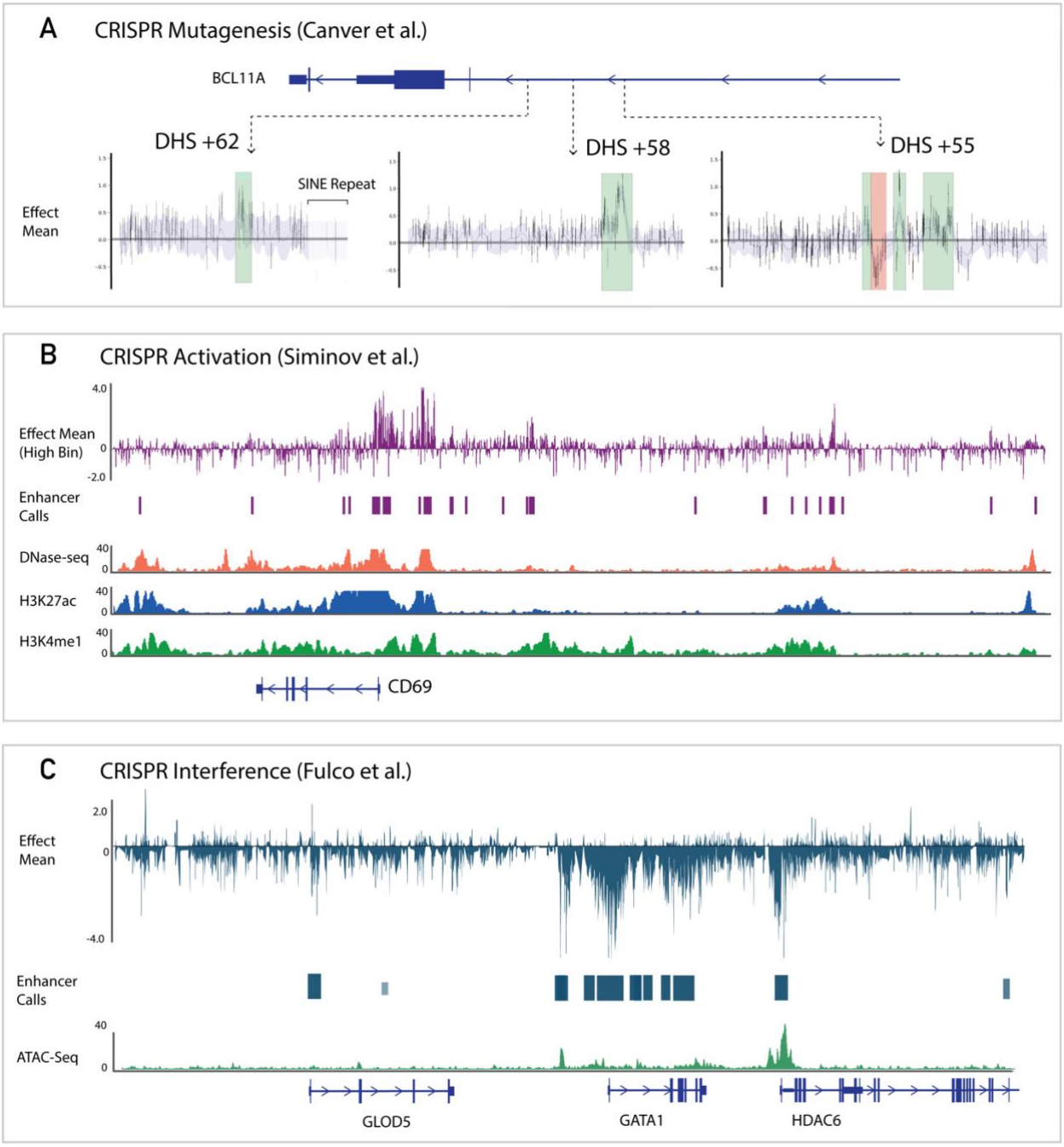


Figure 2: Regulatory elements classified by CRISPR-Decryptr for three published noncoding screens. **A**: Analysis mutagenesis screen targeting BCL11A DHS sites reveals similar enhancer and silencer locations as in the original publications. **B***:* Analysis CRISPRa screen targeting CD69 promoter region reveals novel enhancer calls. **C**: Analysis CRISPRi screen targeting GATA1 gene loci reveals the same enhancer calls as in the original analysis.

Supplemental Notes for CRISPR-Decryptr reveals cis-regulatory elements from noncoding perturbation screens

Anders Rasmussen, Tarmo Äijö
Mariano Ignacio Gabitto, Nicholas Carriero, Jane Skok, Neville Sanjana, Richard Bonneau

Table of Contents:

1. Using CRISPR-Decryptr

2. Method

2.1 Defining the Regulatory Landscape

2.2 Inferring Regulatory Effect

2.3 Constructing a Convolution Matrix

2.4 Gaussian Process Deconvolution

2.5 Decoding Latent State Variables with Hidden semi-Markov Models

3. Analysis of Published Screens

3.1 Mutagenesis Screen of BCL11A DNase Hypersensitivity Sites (Canver et al.)

3.2 CRISPR Activation Screen at IL2RA and CD69 Gene Loci (Simeonov et al.)

3.3 CRISPR Interference Screen at the GATA1 Gene Locus (Fulco et al.)

4. Discussion

4.1 Discussion of Existing Methods

4.2 Discussion of Future Research

Technical Appendix:

A.1 Simulation Studies and Discussion

A.1.1 Examining Log-Fold Change

A.1.2 HsMM and window methods on simulated data

A.2 Notes on Gaussian Processes Models and Collapsed Multinomial

A.3 Notes on Repair Outcome Prediction

A.4 Notes on HsMM Hyperparameters

A.5 Brief Note on gRNA Specificity

Section 1: Using CRISPR-Decryptr

In this section, we present the usage and descriptions of each command in CRISPR-Decryptr for noncoding screens. https://github.com/anders-w-rasmussen/crispr_decryptr

Step 1: Infer perturbation-specific regulatory effect from gRNA count data.

Use: Decryptr infer <count> <design_matrix> <replicate_information> [options]

Positional Arguments:	**Description:**
count_file	Tab-delimited file with columns containing gRNA counts from different samples. The first row should contain sample names corresponding to each column. CRISPR-Decryptr will extract columns from this file according to the sample names included in the design_matrix file. You may include other columns for your own reference (such as gene IDs, etc) but they will be ignored.
design_matrix	Tab-delimited file of design matrix for specific experimental design. The first column of the file should contain the samples you are considering, all of which must correspond to an entry in the first row of the count_file. The first row contains the names of the 'effects' you are considering. The rest of the file is 0s and 1s indicating which effects are to be considered in the linear predictor (see section 2.2)
replicate_information	This file should have an identical first column to the design matrix. The second contains the word 'replicate' in the first row followed by integers indicating which replicate the sample in the first column is from.
Optional Arguments:	**Description:**
--n_chains	Number of chains to use in MCMC sampling (default is 4).
--batch_size	Number of guides per batch. Please see notes on the collapsed multinomial in section A.1. Larger batch sizes scale poorly. We recommend the default of 1000.
--n_batches	The number of batches to run simultaneously. The number of processes spawned will be n_chains * n_batches.
--n_samples	Number of samples for each chain. Default is 500.
--outfile	Out filename
--sample_file_prefix	Prefix for sample filenames to be saved. The default is "samples."
--logfilename	Specify the name for the .log file
--outfile_devs	Write replicate level standard deviations of effects to this file. If not specified this file will not be written. These are not used by CRISPR-Decryptr.

Step 2: Create the convolution matrix.

Use: Decryptr predict <targets> <cas9_alg> [options]

Positional Arguments:	**Description:**
targets	Single column file of target locations. First row should be chromosome, each subsequent row should be a genomic location on said chromosome. This file should have N_guides + 1 rows and must be in the same order as the count_file.
cas9_alg	True / False. Type True if you wish to use off-target and repair prediction (for mutagenesis screens) or False if you wish to use Gaussian windows to construct the convolution matrix. If you choose True, you must include the --spacers and --reference arguments. If you choose False, you must include the --filter_dev and ---filter_window arguments.
Other Arguments:	**Description:**
--species	Species for uniqueness. Only hg19 is supported right now but we may add other genomes in the future. You can use –reference and –uniqueness if you want to use custom files for another species.
--spacers	Single column file containing spacer sequences of the guide library. These must correspond to the intended targets in the 'targets' file.
--reference	.fasta file of the reference chromosome.
--uniqueness	Specify your own fixedStep .wig file of mappabilty/uniqueness of the region under consideration.
--ignore_specificity	If you set this to True, the algorithm will ignore guide-specificity in the convolution matrix.
--filter_dev	Standard deviation of Gaussian window. Larger perturbation profiles (such as for CRISPRa or CRISPRi) should have larger standard deviations. Please see *section 2.3* for more detailed discussion of this parameter.
--filter_window	Size of the Gaussian window. Larger windows will be more computationally expensive in the *deconvolve* step. Please see *section 2.3* for more detailed discussion of this parameter.
--n_processes	Number of processes Decryptr will spawn to create the convolution matrix (only applies if off_target is True).
--logfilename	Name of the logfile to write
--cmat_name	Name the output convolution matrix. Default is convolution_matrix.p

Step 3: Classify regulatory elements with GP deconvolution and HsMMs

Use: Decryptr classify <effect_file> <targets> <conv_mat>

Positional Arguments:	**Description:**
effect_file	Effect file as produced by the inference step (step 1).
targets	Single column file of target locations. First row should be chromosome, each subsequent row should be a genomic location on said chromosome. This file should have N_guides + 1 rows and must be in the same order as the count_file.
conv_mat	Convolution matrix as produced by the predict step (step 2).
Optional Arguments:	**Description:**
--hyperparameters	Hyperparameter file for HsMM (see Section A.4 for details)
--out_dir	Directory to write results files
--logfilename	Name of the .log file
--alpha	Signal deviation parameter for the Gaussian process (see section 2.4)
--rho	Length scale parameter for the Gaussian process (see section 2.4)
--sigma	Process noise parameter for the Gaussian process (see section 2.4)
--bed_threshold	Marginal state probability above which a state is called in the .bed file (default 0.80)
--flip_enhancer	If default hyperparameters are used, this argument if True will flip the enhancer state to emit NEGATIVE regulatory effect. May be appropriate for certain experimental designs such as CRISPRi screens.
--normalize	When the normalize argument is True, the incoming convolved signal will be normalized to a standard score (subtracts the mean of the effect and divides this difference by the effect's standard deviation). In application, this is very useful when effects in the region of interest are not of mean zero, or when we need to compare multiple effects that follow different distributions.
--hampel_filter	When this argument is True, the algorithm attempts to remove outliers from the incoming convolved signal using a Hample Filter.

Section 2: Methods

The objective of CRISPR-Decryptr is to reveal probabilistic locations of regulatory elements on the genome given gRNA counts from noncoding screens. We define a "regulatory landscape," the set of hidden states at each base of the genomic region of interest (*Section 2.1*). To arrive at these state probabilities, we begin by accounting for experimental design and readout, inferring regulatory effects along the genome using a hierarchical generalized linear model (GLM) (*Section 2.2*). Next, we account for off-target and repair prediction, creating a convolution matrix mapping "guide-specific" effects to "base-specific" effects (Section 2.3). Finally, we implement a Gaussian process (GP) model and hidden semi-Markov model (HsMM) to deconvolve effects and classify the "regulatory landscape" at the base-by-base level (*Section 2.4, 2.5, 2.6*).

2.1 Defining the Regulatory Landscape

We assume that perturbation of the noncoding region of interest can have any of E user defined effects on phenotype. As such, the probability mass function of phenotypic readout for any perturbation $x_{pertubation}$ can be written as a function of these effects, $f_{pertubation}(e_1, e_2, .., e_E)$. The goal of the CRISPR-Decryptr method is to take a dataset of phenotypic readouts for various targeted perturbations $\boldsymbol{X}$ and infer the probabilistic locations of regulatory elements within the targeted region that regulate these E effects. We posit that each base $b \in \{1, \dots, B\}$ within the genomic region of length B belongs to one of S_e states. Denote the state of each e, b pair $s_{e,b}$. A unique configuration of the entire region across effects is a set of states $\boldsymbol{s} = \bigcup_{e \,\epsilon\, \{1,\dots,E\}, b \,\epsilon\, \{1,\dots,B\}} \{s_{e,b}\}$. Given our screen data $\boldsymbol{x}$, we wish to determine the marginal probability of each (e, b) pair belonging to any given state, represented by the set of probability vectors

$$\mathbf{P} = \bigcup_{e \in \{1,\dots,E\},\ b \in \{1,\dots,B\}} \{[p(s_{e,b} = 1 \,|\boldsymbol{X}), p(s_{e,b} = 2 \mid \boldsymbol{X}), \dots, p(s_{e,b} = S_e \,|\boldsymbol{X})]\}.$$

In the following sections, we walk through the model step-by-step, moving from the input data $\boldsymbol{X}$ to the set of marginal state probability vectors $\mathbf{P}$.

2.2 Inferring Regulatory Effect

The first part of our algorithm implements a generalized linear model (GLM) and Markov chain Monte Carlo to infer posterior distributions for effects of each perturbation on cell phenotype, denoted $\boldsymbol{g_e} \in \mathbb{R}^I \; \forall \; e \in \{1, \dots, E\}$, where I is the number of guides.[1,2] Allow $c \in \{1, \dots, C\}$ to be condition within the set of C conditions, and $r \in \{1, \dots, R\}$ to be the replicate within the set of R replicates. gRNA count data for each c, r pair is denoted by the vector $\boldsymbol{x}_{c,r} \in \mathbb{N}_{\geq 0}^N$ consisting of individual guide specific counts $x_{c,r,i}$ where $i \in \{1, \dots, I\}$. We model $\boldsymbol{x}_{c,r}$ as being generated from a multinomial distribution with parameter $\boldsymbol{p}_{c,r}$**.** By using the standard unit *Softmax* function to map a real valued vector in $\mathbb{R}^N$ to the simplex Δ^{N-1}, we represent $\boldsymbol{p}_{c,r}$ as a function of a hierarchical linear predictor $\boldsymbol{p}_{c,r} = Softmax(\boldsymbol{\phi}_{c,r})$. The linear predictor takes the following form:

$$\boldsymbol{\phi}_{c,r} = \boldsymbol{d}_c\,[\boldsymbol{\theta}_{1,r}, \boldsymbol{\theta}_{2,r}, \dots \boldsymbol{\theta}_{E,r}], \;\; \forall\, c \in \{1, \dots, C\}, \;\; r \in \{1, \dots, R\}$$
$$\boldsymbol{\theta}_{e,r} \sim \mathrm{N}(\boldsymbol{\mu_e}, diag(\sigma_e)), \;\; \forall\, e \in \{1, \dots, E\}, r \in \{1, \dots, R\}$$
$$\boldsymbol{u_e} \sim \mathrm{N}(u_0, diag(\sigma_0)), \;\; \forall\, c \in \{1, \dots, E\},$$
$$\sigma_e^2 \sim \mathrm{InvGamma}(\alpha, \beta)\,, \;\; \forall\, e \in \{1, \dots, E\}$$

The predictor $\boldsymbol{\phi}_{c,r}$ is a linear combination of replicate-specific effect vectors $\boldsymbol{\theta}_{e,r} \in \{\boldsymbol{\theta}_{1,r}, \dots, \boldsymbol{\theta}_{E,r}\}$ with weights dictated by $\boldsymbol{d}_c$ row c of the design matrix $\boldsymbol{D}$. Each entry in the replicate-specific effect vector is distributed normally according to $u_{e,i}$ and σ_e: guide-specific regulatory effect and a single inferred standard deviation shared across guides. u_0, σ_0 , α, β are fixed hyperparameters. Allow $\boldsymbol{\vartheta} = \{\boldsymbol{\theta_{1,1}} \dots, \boldsymbol{\theta_{E,R}}, \boldsymbol{\mu_1}, \dots, \boldsymbol{\mu_E}, \sigma_1, \dots, \sigma_e\}$ to be the set of model parameters and $\boldsymbol{\zeta} = \{u_0, \sigma_0\,, \alpha, \beta\}$ to be the set of hyperparameters. We write the joint posterior distribution for the model variables using Bayes formula as,

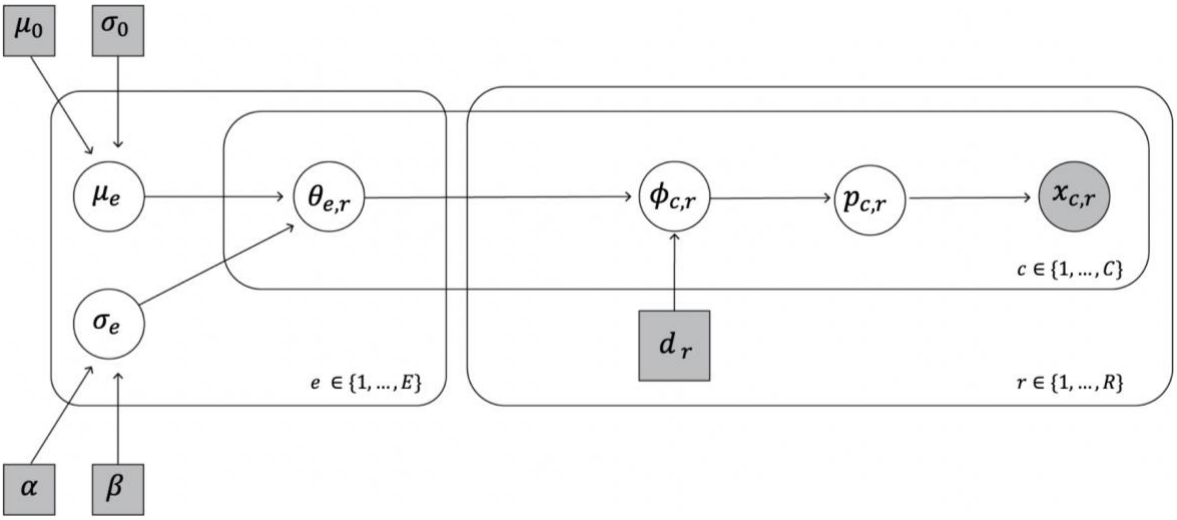
$$f_{posterior}(\boldsymbol{\vartheta} \mid \boldsymbol{x}, \boldsymbol{\zeta}) = \frac{L(\boldsymbol{x} \mid \boldsymbol{\vartheta})\, f_{prior}(\boldsymbol{\vartheta} \mid \boldsymbol{\zeta})}{\int L(\boldsymbol{x} \mid \boldsymbol{\vartheta})\, f_{prior}(\boldsymbol{\vartheta} \mid \boldsymbol{\zeta})\, d\boldsymbol{\vartheta}}$$

Where $f_{posterior}$ is the posterior density function of the model parameters given the data and hyperparameters, f_{prior} is the prior density function of the model parameters given the hyperparameters, and L is the likelihood of the data given model parameters. Closed-form representation of the joint posterior is not possible, as evaluation of the marginal likelihood is intractable. To arrive parameter estimates, we sample from the posterior distribution using Hamiltonian Monte Carlo (HMC) implemented in Stan, which will produce H sampled values of each parameter. Allow $\boldsymbol{\vartheta}_h$ to be the set of sampled parameters for sample $h \in \{1, \dots, H\}$. We calculate the posterior means of each $\boldsymbol{\mu_e}$, and σ_e^2 as follows:

$$\boldsymbol{\mu}_{\boldsymbol{\mu}_e} = \frac{1}{H}\sum_{h\in H}\boldsymbol{\mu}_{\boldsymbol{e,h}}\ \forall\ e\ \epsilon\ \{1,\dots,E\},\quad \mu_{\sigma_e^2} = \frac{1}{H}\sum_{h\in H}\sigma_{e,h}^2\ \forall\ e\ \epsilon\ \{1,\dots,E\}$$

We can now express the guide-specific regulatory effects as,

$$\boldsymbol{g}_e \sim N\left(\boldsymbol{\mu}_{\boldsymbol{\mu}_e}, diag\left(\mu_{\sigma_e^2}\right)\right)\forall\ e\ \epsilon\ \{1,\dots,E\}.$$



Supplementary Figure 2.2: Plate diagram of hierarchical Generalized Linear Model for effect inference. White circles represent latent model parameters to be inferred, while grey circles represent observed quantities. Grey squares are user-defined parameters, while circular nodes are model variables.

2.3 Constructing a Convolution Matrix

In *Section 2.1*, we defined the latent regulatory landscape $\boldsymbol{s}$ as being the set of base and effect-specific states $\bigcup_{e \in \{1,\dots,E\}, b \in \{1,\dots,B\}} \{s_{e,b}\}$. In *Section 2.2* above, we described a numerical method to infer perturbation specific effects $\boldsymbol{g}_e \in \mathbb{R}^I$. However, to examine regulatory elements at the base-by-base level of granularity we must arrive at a base-specific effect vectors $\boldsymbol{b_e} \in \mathbb{R}^B \; \forall \; e \in \{1, \dots, E\}$. We do so by positing that there exists a linear map $f: \mathbb{R}^B \rightarrow \mathbb{R}^B$ that can be represented as an $I \times B$ convolution matrix $\boldsymbol{C}$, such that $\boldsymbol{g}_e = \boldsymbol{C}\boldsymbol{b_e} \; \forall \; e \in \{1, \dots, E\}$. We define the entries c_{ib} of matrix $\boldsymbol{C}$ as representing the probability of guide i perturbing base b. As such, we construct the convolution matrix guide-by-guide by calculating Cas9-binding and perturbation probabilities across the region of interest. The raw data, as well as our inferred effects, are specific to guide i with target sequence, $target_i$. We define the target DNA sequence as the sequence complementary to the guide RNA. For mutagenesis screens, we assume the probability of guide i inducing a mutation at base b can be written

$$c_{ib} = \sum_{b' \in \{1,\dots,B\}} p^{bind}_{i,b'}(target_i) \times p^{pert}_{b,b'}$$

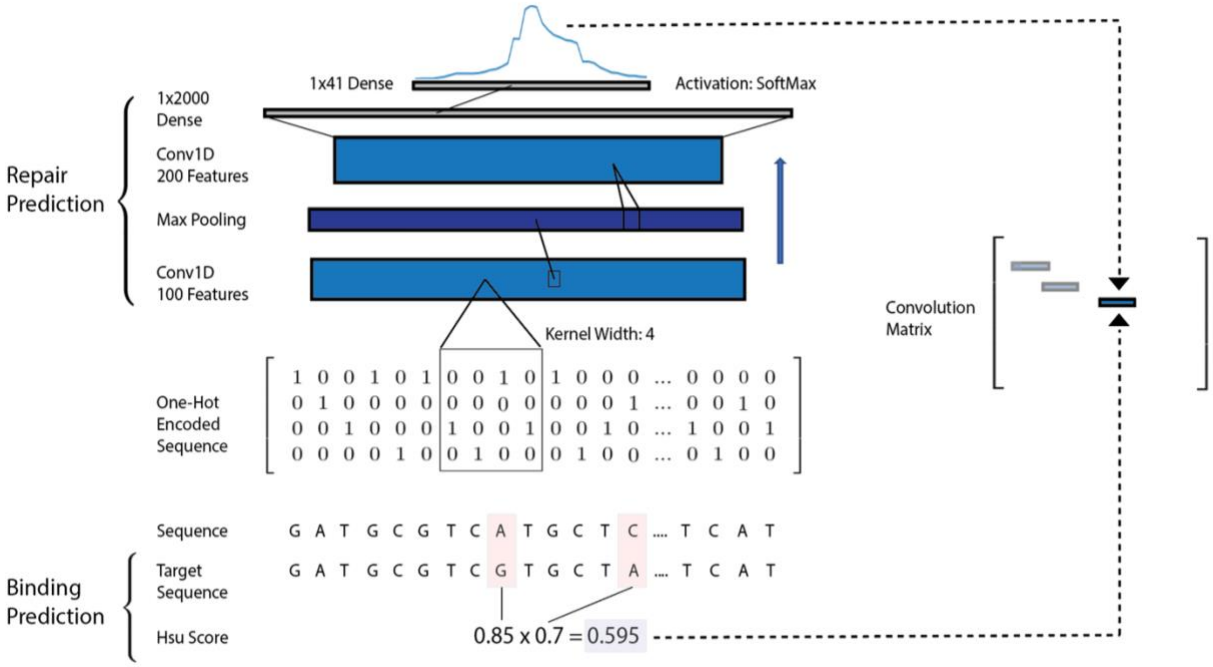
where b' is the base where Cas9 binds, defined as the most 3' nucleotide in the PAM sequence NGG on the both strands. $p^{bind}_{i,b'}$ is the probability that Cas9 fused with guide i binds at base b'. $p^{pert}_{b,b'}$ is the probability that a Cas9 binding event at base b' results in a genomic perturbation at base b.

We calculate each binding probability $p^{bind}_{i,b'}(target_i) \; \forall \; i \in \{0, \dots, I\}, \; b' \in \{0, \dots, B\}$ using the scoring methodology presented in Hsu *et al.* (2013)[3]. Any base that is not the last nucleotide in an NGG motif is assigned a Cas9 binding of zero. Allow $target_i[b]$ to be the b-th nucleotide in the target sequence and $candidate_{b'}[b]$ to be the b-th nucleotide in the 20 base-pair sequence beginning 23 bases in the 5' direction of b' and terminating 3 bases in the 5' direction of b'. We define the Hsu tensor $Hsu[t, c, b]$ as a $4 \times 4 \times 20$ tensor of Hsu scores defined in **Supplementary Table 2.3**.

$$p^{bind}_{i,b'}(seq_i) = \prod_{b=1}^{20} Hsu[target_i[b], candidate_{b'}[b], b]$$

The prediction of perturbation profile is dependent on experimental design, for CRISPR mutagenesis screens that induce cleavage events, we can calculate perturbation probabilities at each base around the binding site $p^{pert}_{b,b'}(target_i)$ using a Convolutional Neural Network trained on a database of double stranded break outcomes in Allen et al (2017)[4]. Please see

Section A.3 on more details on CNN model performance. Allow $\boldsymbol{seq_{b'}}$ to be a 4×41 one-hot encoding of the sequence beginning 20bps in the 5' direction of b' and ending 20bps in the 3' direction of b'. We allow $\boldsymbol{p^{mut}_{:,b'}}$ to be the 1×41 vector $[p^{mut}_{b'-20,b'}, p^{mut}_{b'-19,b'} \ldots. p^{mut}_{b'+20,b'}]$. As such, the CNN model approximates a function $f_{CNN}: \boldsymbol{seq_{b'}} \mapsto \boldsymbol{p^{mut}_{:,b'}}$.



Supplementary Figure 2.3: Illustration CRISPR-Decryptr's method for convolution matrix construction. For Cas9 mutagenesis screens, repair prediction is performed by a convolutional neural network model, taking one-hot encoded sequence as an input and outputting a sequence-specific pertubation profile. The guide is also aligned to the reference genome based on its Hsu score, determining the position and weighting the perturbation profile in the convolution matrix.

For CRISPRi and CRISPRa screens, we construct the convolution matrix out of Gaussian Windows of user defined standard deviation σ_{window} and odd integer window size W, centered on b'_i, defined as the most 3' nucleotide in the PAM sequence of the intended target of guide i. We write the entries of the convolution matrix as follows:

$$c_{ib} = \sum_{b' \in \{1,\ldots,B\}} p^{bind}_{i,b'}(seq_i) \times \begin{cases} \exp\left\{-\frac{1}{2}\left(\frac{b-b'_i}{\sigma_{window}\frac{W}{2}}\right)^2\right\} & \text{if } b \in \{b' - \frac{W}{2}, b' + \frac{W}{2}\} \\ 0 & otherwise \end{cases} \quad \forall\, i \in \{0,N\},\ b \in \{0,M\}$$

Distance from PAM

Mismatch Type (Guide:Target)	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
A:A	1.19	1.10	2.87	1.13	0.52	0.64	1.03	0.54	1.29	0.08	0.26	0.27	0.05	0.03	0.01	0.02	0.17	0.70	0.27
A:C	1.51	1.16	1.66	0.99	0.02	0.55	1.87	0.77	0.93	0.73	0.48	0.48	0.71	1.02	0.02	0.35	0.18	0.14	0.27
A:G	1.48	1.08	1.30	0.74	0.98	0.49	0.93	0.60	1.04	0.30	0.82	0.03	0.04	0.06	0.03	0.16	0.10	0.22	0.03
A:T	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
C:A	1.02	0.96	2.25	1.62	0.24	0.76	1.07	0.11	1.43	0.35	0.62	0.73	0.12	0.04	0.50	0.41	0.18	0.53	0.17
C:C	0.03	1.05	1.05	1.41	0.05	0.64	2.08	0.46	0.16	0.14	0.10	0.04	0.01	0.05	0.01	0.21	0.06	0.03	0.06
C:G	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
C:T	1.67	0.97	1.07	1.09	0.65	1.01	0.86	0.31	0.45	0.90	0.43	0.56	0.14	0.07	0.33	0.24	0.04	0.14	0.44
G:A	1.17	0.45	2.79	1.05	0.60	0.81	0.79	0.69	1.37	0.12	0.20	0.25	0.01	0.02	0.05	0.37	0.32	0.56	0.46
G:C	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
G:G	1.32	1.42	1.30	1.21	0.95	0.23	1.02	0.81	1.16	0.71	1.58	0.35	0.09	0.05	0.00	0.23	0.22	0.53	0.50
G:T	1.65	0.90	1.60	0.92	0.81	0.96	1.02	0.27	0.71	1.52	0.69	1.04	0.53	0.90	0.72	0.84	0.54	0.28	0.69
U:A	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
U:C	0.04	1.00	1.65	1.34	0.01	0.54	2.34	0.44	0.21	0.34	0.26	0.09	0.05	0.07	0.02	0.37	0.05	0.09	0.25
U:G	1.54	1.28	1.27	1.22	1.06	0.78	1.24	0.99	1.53	0.72	1.74	0.42	0.13	0.07	0.05	0.23	0.31	0.13	0.06
U:T	1.73	0.82	1.16	1.18	0.74	0.90	0.84	0.30	0.62	1.14	0.36	0.62	0.18	0.27	0.28	0.26	0.12	0.17	0.34

Supplementary Table 2: Illustration of the CRISPR-Decryptr method for convolution matrix construction. For Cas9 mutagenesis screens, repair prediction is performed by a convolutional neural network model, taking one-hot encoded sequence as an input and outputting a sequence-specific pertubation profile. The guide is also aligned to the reference genome based on its Hsu score, determining the position and weighting the perturbation profile in the convolution matrix.

2.4 Gaussian Process Deconvolution

In *section 2.3* we constructed a convolution matrix that satisfies the equation $\boldsymbol{g_e} = \boldsymbol{C b_e}$ under our assumptions. However, the problem of solving for a vector of base-specific effect $\boldsymbol{b_e}$, given a vector of guide-specific effect, is ill-posed, as the matrix $\boldsymbol{C}$ is singular. To solve the inverse problem, we implement a Gaussian Process model by defining a Gaussian process prior on $\boldsymbol{b_e}$ such that $\boldsymbol{b_e} \sim GP(\boldsymbol{0}, \boldsymbol{K})$[5]. $\boldsymbol{K}$ is a kernel matrix approximating the covariance structure as a squared exponential function of the distance between guides i and j, $K_{i,j} = \alpha^2 \exp\{-\frac{1}{2\rho^2}(q_i - q_j)\}$, where q_i, q_j are the genomic positions of i and j respectively. The characteristic-length scale ρ and signal variance α^2 are inferred parameters. Note that $\circ$ denotes the Hadamard or element-wise product.

Given the linear map $\boldsymbol{C}$ defined in 2.3, we write the likelihood for the guide specific effect variables as $\boldsymbol{g_e} \sim \boldsymbol{N}(\boldsymbol{C b_e}, diag(\sigma^2))$ where σ^2 is a process-noise term to be inferred. The joint distribution of $\boldsymbol{g_e}$ and $\boldsymbol{b_e}$ can now be written:

$$\begin{pmatrix} \boldsymbol{g_e} \\ \boldsymbol{b_e} \end{pmatrix} = N\left(\begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix}, \begin{pmatrix} \boldsymbol{CKC^T} + diag(\sigma^2) & \boldsymbol{CK} \\ \boldsymbol{KC^T} & \boldsymbol{K} \end{pmatrix}\right)$$

The posterior predictive distribution for $\boldsymbol{b_e}$ now takes the closed-form solution,

$$\boldsymbol{b_e} = \boldsymbol{KC^T}\,(\boldsymbol{CK\,C^T} + diag(\boldsymbol{\sigma^2}))^{\boldsymbol{T}}\boldsymbol{g_e}$$

The parameters ρ, α^2, σ^2 can optionally be set by the user. By default, CRISPR-Decryptr sets α^2 and σ^2 both to the average variance in the guide-specific effect signal and sets the length scale ρ to the average distance between saturated guides (defined as guides within 100bps).

By default, CRISPR-Decryptr does not normalize the incoming effect $\boldsymbol{g_e}$. However, if desired, the classify step can use the convolved effect's standard score $(\boldsymbol{g_e} - \boldsymbol{E}[\boldsymbol{g_e}]) / \sqrt{\boldsymbol{Var}[\boldsymbol{g_e}]}$ if the --normalize argument is set to True (see *section 1*). As the GP prior has a fixed mean of zero, this can influence the results of the deconvolution if the difference between zero and $\boldsymbol{E}[\boldsymbol{g_e}]$ is large.

2.5 Decoding Latent State Variables with Hidden semi-Markov Models

We model the latent state path as following a discrete time semi-Markov process, where the effect specific state of each base is denoted $s_{e,b}$. In a standard Markov process, the duration – or sojourn time – spent within any given state, follows a geometric distribution, an assumption we believe to be in conflict with the distribution in sizes of experimentally measured regulatory elements. As such, we implement a Hidden semi-Markov Model with state durations that follow a negative binomial distribution with state-specific parameters, written $D_{s,e} \sim NB(r_{s,e}, p_{s,e})$[6]. For CRISPR-Decryptr, we assume the HsMM has a latent model structure comprised of an arbitrary number of states n (default is two). The transition matrix and initial probability vector for each effect *e* take the form:

$$A_e = \begin{bmatrix} 0 & a_{12,e} & \dots & a_{1n,e} \\ a_{21,e} & 0 & & a_{2n,e} \\ \vdots & & \ddots & \\ a_{n1,e} & a_{n2,e} & & 0 \end{bmatrix}, \quad \pi_e = \begin{bmatrix} \pi_{1,e} \\ \pi_{2,e} \\ \vdots \\ \pi_{n,e} \end{bmatrix}$$

Where $a_{ij,e}$ represents the probability of transitioning from effect-specific state *i* to state *j* and $\pi_{i,e}$ represents the probability of base 0 being the effect-specific state *i*.

We now define the emission probabilities for each state. We assume the base-specific mean for effect e, $\mu_{e,b}$ follows a Normal distribution with inferred mean $\mu_{e,s}$ and precision $\tau_{e,i} = \frac{1}{\sigma_{e,i}{}^2}$, where $\sigma_{e,b}$ was the standard deviation of the effect mean at base *b.* We write this probability as follows:

$$p(\mu_{e,b} \mid s_b = s) = \sqrt{\frac{\tau_{e,i}}{2\pi}} \exp\{\frac{\tau_{e,i}(\mu_{e,i} - \mu_{e,s})^2}{2}\}$$

To maintain a Bayesian approach, we define prior distributions on model parameters as follows:

$$p_{e,s} \sim Beta(\alpha_{s,e}, \beta_{s,e}) \ \forall \ s \in \{1, \dots, S\}, e \in \{1, \dots, E\}$$
$$a_{e,s,:} \sim Dirichlet(\eta_s) \ \forall \ s \in \{1, \dots, S\}, \ e \in \{1, \dots, E\} \text{ where } \eta_s \in \mathbb{R}^S$$
$$\pi_e \sim Dirichlet(\eta_{\pi,e}) \text{ where } \eta_{\pi,e} \in \mathbb{R}^S$$
$$\mu_{e,s} \sim Normal(\mu_{e,s,0}, \frac{1}{\tau_{\mu,0}}) \ \forall \ s \in \{1, \dots, S\}, \ e \in \{1, \dots, E\}$$

Implementation of the Baum-Welch algorithm leads to estimates for marginal state-probability vectors $\mathbf{P}$ as described in *section 2.1.*

Section 3: Analysis of Published Screens

3.1 Mutagenesis Screen of BCL11A DNase Hypersensitivity Sites (Canver et al.)

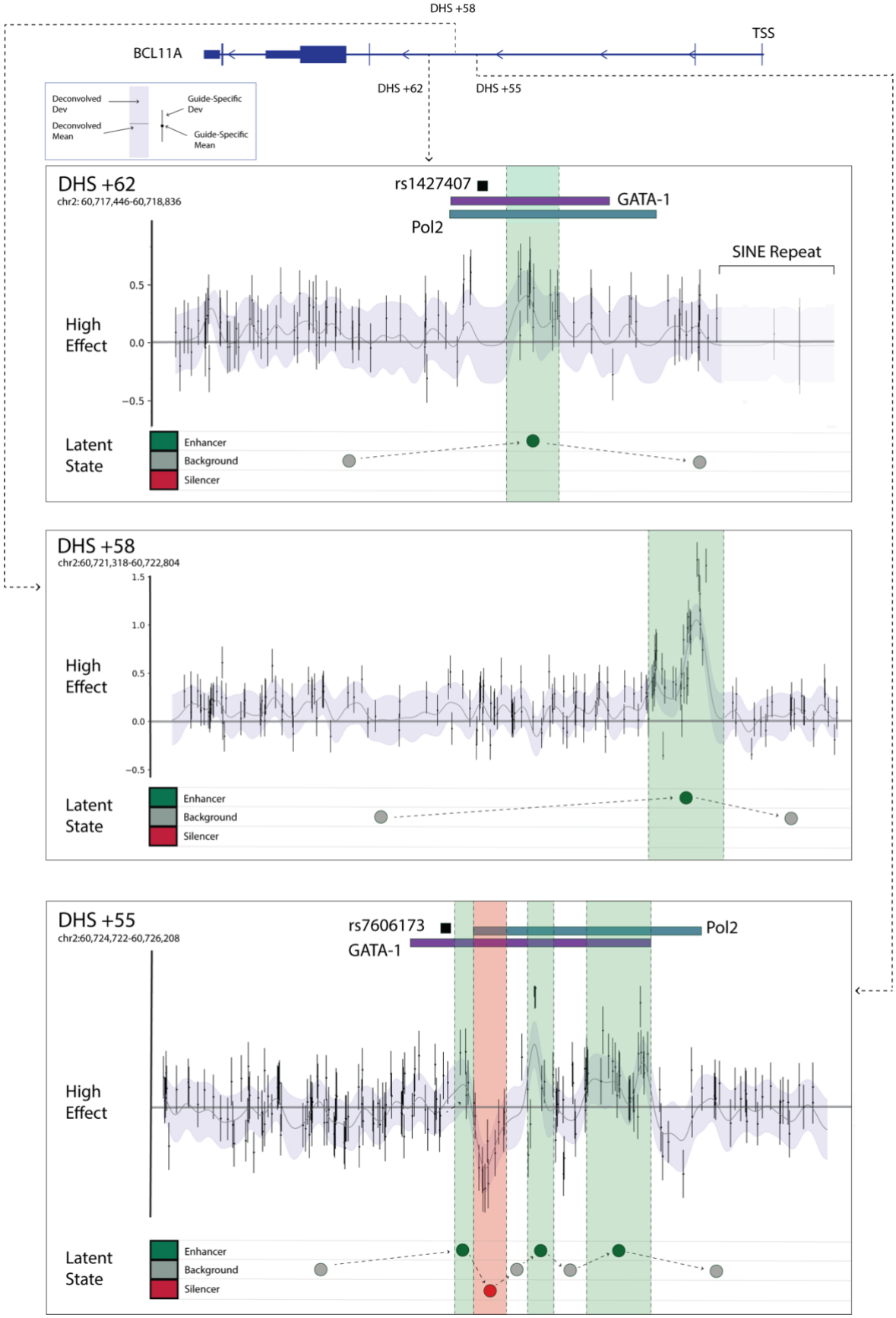
The BCL11A gene is a validated repressor of fetal hemoglobin (HbF) level. In their 2013 paper, Bauer et al. label three DNAse hypersensitivity sites (DHS) that overlap with single nucleotide polymorphisms (SNPs) that impact HbF levels in genome-wide association studies (GWAS)[7]. These sites, DHS +62, +58, and +55, are named according to their distance in kilobases from the TSS of BCL11a. The HbF-associated SNPs rs1427407 and rs7606173 are located within DHS +62 and DHS +55 respectively. DHS +58 has proven a successful therapeutic target in patients with hemoglobinopathies using CRISPR-Cas9 gene editing technology. Due to their function being known *a priori* and validated in application through therapeutic interventions, these three sites are ideal for exploring experimental and computational approaches to interrogating cis-regulatory elements. In Canver et al., these sites are perturbed in human umbilical cord blood-derived erythroid progenitor (HUDEP) cells CRISPR-Cas9[8]. The phenotypic readout for this experiment is gRNA counts from cells FACS sorted into a HbF high and HbF low bin and an early condition as control. The original analysis defines HbF enrichment as the $\log_2$ fold-change between the normalized HbF high and HbF low bins and utilizes a three-state Hidden Markov Model (HMM) to classify silencers and enhancers. We re-analyze this data with CRISPR-Decryptr, implementing a design matrix for effect inference as follows:

$$\begin{array}{r} \\ \text{High-Specific} \\ \text{Common} \\ \text{Early} \end{array} \begin{array}{c} \begin{array}{ccc} \text{High} & \text{Low} & \text{Early} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

We utilize the following hyperparameters for the Hidden semi-Markov Model:

STATE	E_MU	E_TAU_0	NB_R	NB_SUCC	NB_FAIL
BACKGROUND	0	1	2	1	500
SILENCER	-2	1	2	1	40
ENHANCER	2	1	2	1	40

From the high-specific effect, we arrive at a regulatory landscape in concordance with the original analysis. The common and early effects also yield insight into other phenomena impacting gRNA counts in the other bins. Targeting the Alu SINE repeat in the DHS +62 region impacts cell viability in both the early and later bins, a phenomenon that was also noted in the original analysis.



Supplementary Figure 3.1.1: Results from the analysis of guides targeting BCL11A DHS hypersensitivity sites in Canver et al. The high-specific effect reveals similar enhancer and silencer calls as in the original analysis.

3.2 CRISPR Activation Screen at IL2RA and CD69 Gene Loci (Simeonov et al.)

Simeonov et al. introduce the CRISPR activation screen, which utilizes a mutated version of Cas9 (dCas9) without endonuclease activity which - when fused with gRNA and transcriptional activators – is able to activate regulatory elements on the genome[9]. To validate the approach, the authors target IL2RA and CD69 gene loci in two different pooled screens, by transducing Jurkat-T cells with a dCas9-VP64 activator. To measure phenotypic change, the authors FACS sort cells into a “negative”, “low”, “medium”, and “high” bins of IL2RA and CD69 based on expression levels, as well as a non-transduced control group. We re-analyze this screen using the following design matrix:

	Background	Negative	Low	Medium	High
Background	1	0	0	0	0
Negative	1	1	0	0	0
Low	1	0	1	0	0
Medium	1	0	0	1	0
High	1	0	0	0	1

For the creation of the convolution matrix, we use the gaussian window method described in *section 2.3* with default parameters. For the HsMM, we utilized default hyperparameters.

CRISPR-Decryptr successfully recalls all the putative regulatory elements from the Simionov et al. analysis. The two putative regulators CaRE1 and CaRE6, revealed solely in the “low” bin in the original analysis, are now revealed “medium” bin as well. Additionally, CRISPR-Decryptr calls novel putative enhancers of varying strengths, many of which are highly correlated with DNAse-seq and the active enhancer mark H3K27ac from the Jurkat-T Cell line. These putative enhancers include the nearby RBM17 and PFKFB3 gene promoters.

Putative enhancers of different strengths (marginal state probability > 0.95, > 0.85, and > 0.75 for Strong, Moderate, and Faint calls respectively) included in **figure 3.2.2** illustrates how the method’s output of marginal state probabilities provides an easily interpretable metric by which to rank regulators.

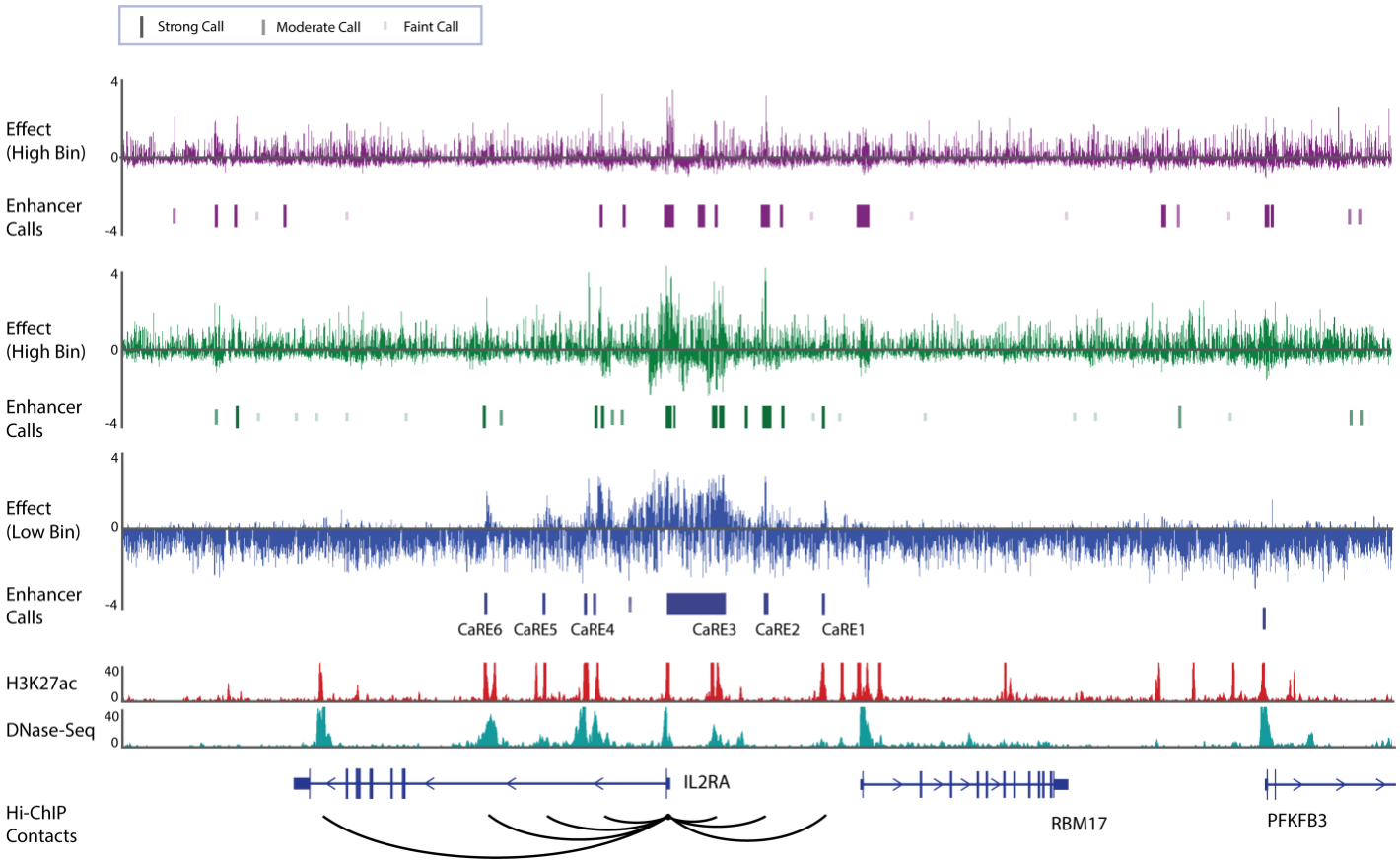


Figure 3.2.2: Analysis of Simenov et al. CRISPRa screen of the IL2RA gene locus. High, Medium, and Low effects are inferred from the inference step of CRISPR-Decryptr, with enhancer state calls from the HsMM of varying strengths displayed below respective bins (marginal state probability > 0.95, > 0.85, and > 0.75 for Strong, Moderate, and Faint calls respectively). H3K27ac, DNAse-seq, and HiChIP contacts, are correlated with active enhancers.

At the CD69 gene loci, we consider the “high” expression bin as in the original analysis. Similar to the IL2RA gene loci, we recall the original enhancers and elucidate new enhancer calls in strong agreement with chromatin accessibility data and histone marks.

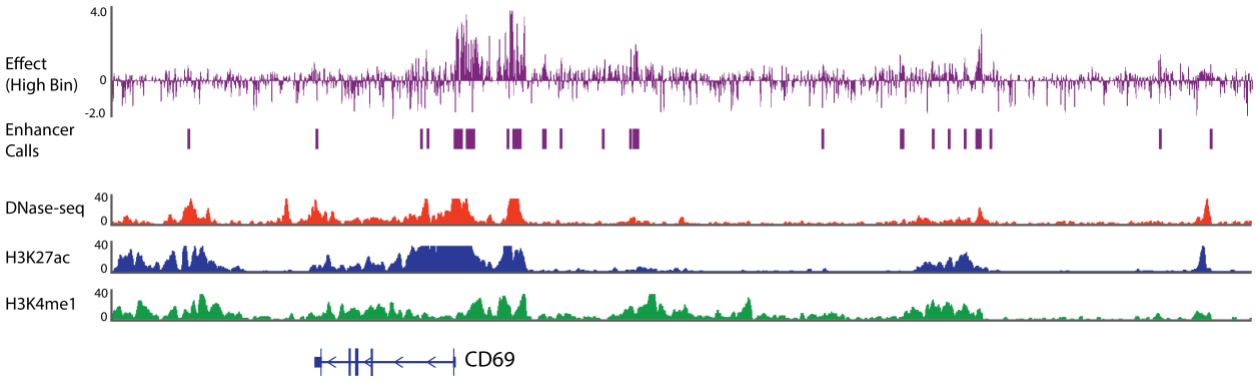


Figure 3.2.3: Analysis of Simenov et al. CRISPRa screen of the CD69 gene locus with the deconvolved effect from the “high” bin displayed above “strong” enhancer calls ($p > 0.95$). H3K27ac, DNAse-seq, and H3K4me1 are displayed to compare results with chromatin accessibility and active enhancer markers.

3.3 CRISPR Interference Screen at the GATA1 Gene Locus (Fulco et al.)

In a noncoding screen targeting the GATA1 loci, Fulco et al. fuse inactivated dCas9 with Krüppel associated box (KRAB) domain to repress transcriptional activation – an approach termed CRISPR Interference[10]. K562 erythroleukemia cells which express the KRAB-dCas9 doxycycline-inducible promoter were infected with a custom gRNA library. As GATA1 impacts K562 proliferation, phenotypic readout is gRNA counts, where depletion implies decreased gene expression. As such, the targeting of an enhancer should result in negative inferred regulatory effect around the site of perturbation. For the analysis, we utilize a filter standard deviation of 30, set the *--flip_enhancer* argument to True in the classify step, and use a design matrix of the following form:

$$\begin{array}{c} \\ \text{Early} \\ \text{Late} \end{array} \begin{array}{c} \begin{array}{cc} \text{Early} & \text{Late} \end{array} \\ \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{array}$$

In addition to calling the same putative regulators as in the original analysis, CRISPR-Decryptr reveals one faint novel enhancer at the GATA1 loci within the GLOD5 intron. From ChIPseq of K562 cells, we see this call is at the binding site of a CEBPB transcription factor, a known regulator of inflammatory processes which is also bound at other regions demonstrating gRNA depletion in the GATA1 loci.

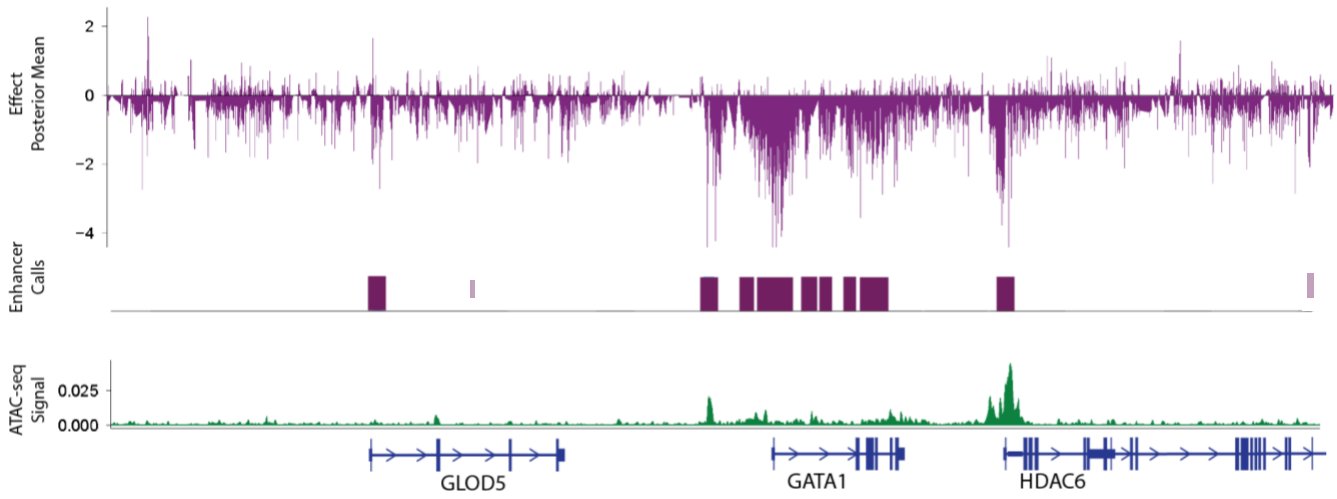


Figure 4.3.1: Analysis of gRNAs fused with KRAB-dCas9 targeting the GATA1 gene loci. In addition to enhancers identifies in the original analysis, CRISPR-Decryptr reveals a novel enhancer targeting an experimentally validated CEBPB transcription factor binding site in K562 cells.

Section 4: Discussion

4.1 Discussion of Existing Methods

CRISPR noncoding screens represent a relatively new method for interrogation of functional elements in the noncoding genome. We believe any method aiming to classify *de novo* regulatory elements must be able to accomplish two tasks: firstly, the algorithm should be able to take raw screen data and translate it into some quantity representing the effect of perturbation on phenotype. Second, the algorithm should classify regulatory element locations using this quantity. At the time of this submission, two methods have been published in the literature, both of which address some part of a complete noncoding screen analysis. CRISPR-SURF,[11] a deconvolution framework employing Lasso regularization, and MAUDE[13] a method for the analysis of gene expression changes in sorting-based CRIPSR screens, such as the CRISPRa screens discussed in *section 3.2*.

In CRISPR-SURF each gRNA's effect on phenotype is calculated as the Log_2 fold change (Log_2FC) across a pair of conditions and/or timepoints for a specific perturbation. When calculating the ratio between counts, small denominators lead to exaggerated fold changes. Similarly, large count differences have the tendency to be underexaggerated by the transformation. Please see *section A.1.1* for simulated illustrations of the drawbacks of fold changes when applied to simulated gRNA count data. Additionally, the fold change calculation provides no intuitive way to account for experimental design or combine information across replicates without crude calculations such as averaging fold changes. CRISPR-SURF primarily serves as a deconvolution method, modeling experiment specific perturbation profiles and mapping guide-specific quantities to base-specific quantities using a deconvolution operation implemented through Lasso, a common technique in machine learning for solving ill-posed inverse problems. In the case of CRISPR mutagenesis screens, CRISPR-SURF applies a user-defined perturbation profile to all guides, however, research on measured repair outcomes from Cas9 cleavage in Allen et. al. indicates that perturbation profile is sequence specific. Additionally, CRISPR-SURF does not account for off-target effects or sequence specificity, a major concern with any experiment or therapy using CRISPR technology. CRISPR-Decryptr explicitly models sequence specific repair profiles and off-target effects for CRISPR mutagenesis screens, predicting posterior effect variables at each base in the noncoding region using a Gaussian Process (GP) Model for deconvolution. In contrast with techniques such as Lasso or Tikhonov regularization, the GP framework is able to account for uncertainty estimates from the guide-specific inference step when predicting base-specific effect parameters.

MAUDE takes a statistical approach that is designed specifically for binned count readouts, an experimental design where mutated cells are FACS sorted based on their expression level as in the IL2RA/CD69 CRISPRa screen (Simionov et al) presented in *section 3.2*. The authors propose a model that accounts for gRNA count distributions across these FACS sorted bins, accounting for control guides to arrive at guide-specific Z-score which serves as their metric of effect on expression level. As this method is designed specifically to account for binned gRNA counts, it does not have the ability to account for other screen designs. Additionally, the statistical model does not provide a statistical treatment of parameter uncertainty that utilizes information across replicates. For classification of regulatory elements, MAUDE utilizes a sliding window method of arbitrary size to group guides, CRISPR-Decryptr's HsMM step decodes the marginal probabilities of regulatory state-space configurations of individual nucleotides, fully capturing spatial information in the deconvolved effect signal. In *section A.1.2*, we present simulations to demonstrate the limitations of sliding window analysis and relative outperformance of HsMMs.

Neither CRISPR-SURF nor MAUDE account for guide specificity, off-target effect, or repair outcome. We believe these are factors that should not be ignored when analyzing CRISPR noncoding screens. The authors of MAUDE claim their algorithm outperforms other methods due to the fact that it classifies 12 new regulatory elements when analyzing the CD69 CRISPRa screen (section 3.2). While 10 of these region appear to downregulate CD69, the authors do not account for the fact that these regions are overlapping extremely non-unique sequences, such as SINE repeats. As we discuss in a brief note in *section A5*, non-specific guides appear to be consistently depleted in some experimental designs, including the CD69 screen. By not accounting for guide-specificity, we believe the majority of MAUDEs regulatory element calls in their validation to be false positives. This highlights the importance of considering guide-specificity through all steps of the analysis of CRISPR noncoding screens.

CRISPR-Decryptr is the most complete framework for analyzing CRISPR noncoding screens of various experimental designs. As a fully generative model, CRISPR-Decryptr has an explicit model formulation and clear mathematical assumptions. The Generalized Linear Model implemented in *section 2.2* captures the compositional nature of gRNA counts, and able to model a diverse set of experimental designs through the user defined design matrix. The Gaussian Process utilized in *section 2.4* provides deconvolution framework similar to that in CRISPR-SURF, while also accounting for off-target effects, guide specificity, and probabilistic repair outcomes. The Hidden semi-Markov Model in CRISPR-Decryptr provides accurate state classification at the base-by-base level of granularity. Finally, by employing Bayesian inference techniques, CRISPR-Decryptr models parameter uncertainty at all stages of the analysis.

	CRISPR-SURF	MAUDE	CRISPR-DECRYPTR
MODEL FOR INFERENCE OF GUIDE-SPECIFIC EFFECT	Log2 Fold Change	Statistical Model for Binned gRNA Counts	Generalized Linear Model with Flexible Linear Predictor. MCMC Bayesian Inference.
MAPPING GUIDE-SPECIFIC EFFECT TO BASE-SPECIFIC EFFECT	Lasso-based Deconvolution	None	Gaussian Process Model with Off-Target Alignment, Specificity Adjustment, and Repair Outcome Prediction
CLASSIFICATION METHOD	None	Sliding Window Statistical Tests	Bayesian Hidden semi-Markov Model

Table 4.1.1: Table comparing how the three algorithms approach what we believe to be the three requisite aspects of a complete CRISPR noncoding screen analysis.

	CRISPR-SURF	MAUDE	CRISPR-DECRYPTR
ACCOUNTS FOR EXPERIMENTAL DESIGN	No	**Yes**	**Yes**
COMBINES INFORMATION ACROSS REPLICATES	No	No	**Yes**
STATISTICAL MODEL SPECIFICALLY DESIGNED FOR BINNED SCREENS	No	**Yes**	No
STATISTICAL TREATMENT OF PHENOTYPIC EFFECT UNCERTAINTY	No	No	**Yes**
DECONVOLUTION FRAMEWORK TO ACCOUNT FOR PERTURBATION PROFILE	**Yes**	No	**Yes**
OFF-TARGET PREDICTION	No	No	**Yes**
REPAIR OUTCOME PREDICTION	No	No	**Yes**
GUIDE-SPECIFICITY ADJUSTMENT	No	No	**Yes**
BASE-BY-BASE CLASSIFICATION OF REGULATORY ELEMENTS	No	No	**Yes**

Table 4.1.2: Table comparing CRISPR-SURF, MAUDE, and CRISPR-Decryptr on various attributes.

4.2: Discussion of Future Research

As each step in the CRISPR-Decryptr method has been designed as a modular component, they are readily adaptable as new research and technologies in gene editing emerge. Here we present our thoughts on some opportunities for future research:

Future research may support new generative model formulations that differ from the GLM implemented in *section 2.2*

In *section 2.2* we choose to implement a generalized linear model with a multinomial distribution due to the compositional nature of gRNA count data and simplicity of the canonical GLM framework. It is possible that different formulations could be proposed through future research and experience with the analysis of CRISPR Screens. The *infer* command's GLM model, implemented primarily in STAN, can be readily modified.

Research could help with more accurate construction of convolution matrices

In this paper, we use two published works (Allen et al. and Hsu et al.) in our construction of the convolution matrix from *section 2.3*[3, 4]. It is likely there are other current or future publications that can help in more accurately predicting the perturbations of gRNAs from sequence or other features. By simply replacing the *predict* step with another means of convolution matrix construction, the CRISPR-Decryptr method is readily adaptable.

Research has the potential to build upon the GP deconvolution step

We believe the GP deconvolution is a powerful framework for deconvolving guide-specific effects. Future research could explore alternative kernels to the squared exponential loss that may better capture the underlying process dynamics. Additionally, if computationally feasable, the Gaussian Process parameters could be fit in a Bayesian manner.

Classification of regulatory elements should leverage multiple technologies

We believe it is optimal to combine results from multiple technologies in the inference of regulatory element locations. Within the scope of the paper, we compare regulatory element state calls with other genomic signals, such as histone marks and ATAC-seq. However, in practice it would be ideal to leverage information from multiple signals in one inference procedure. This could be done by adding emissions for these technologies to the HsMM in *section 2.5*.

The techniques presented in CRISPR-Decryptr may be of use in other analyses

The core model components of CRISPR-Decryptr also have applications to other problems in computational biology. The canonical GLM model can be extended to readouts following other distributions from the exponential family, a broad group of probability distributions that offer both discrete (binomial, poisson) and continuous (normal, exponential, gamma) supports. Additionally, the formulation of the linear predictor can be modified to account for new experimental designs or insights and account for *a priori* knowledge about model parameters. These factors establish Bayesian GLMs as a robust approach for inferring effects of genomic perturbations induced by CRISPR-Cas9 cleavage or future editing technologies. Outside of gene editing technologies, the Gaussian Process model can serve to deconvolve a variety of sparse genomic signals if the model assumptions hold on the data in question. Additionally, the HsMMs implemented in *section 2.5* can be used to decode latent state variables from a variety of genomic signals, including those with missing observations. We have implemented similar HsMM models to annotate accessible chromatin using ATAC-seq and DNAse-seq data and intend to continue the development of these models.[6]

Deep learning algorithms could learn generative models

Generative Bayesian models are readily interpretable and provide a statistical framework to utilize prior knowledge of uncertain systems. It is our belief that these models are invaluable for unsupervised learning tasks in computational biology, however, their proclivity to become computationally expensive can make their application to analyses of entire genomes difficult. We believe learning generative models with deep neural networks may allow for extremely fast applications of complicated models. This could be done by applying generative models to simulated or real data and using the results produced to train deep neural networks.

APPENDIX:

A.1. Simulation Studies

A.1.1 Log_2 Fold-Change on Simulated Data

The Log_2 Fold-Change is used the quantify the change in gRNA counts across conditions in some methods for noncoding and knockout screens (see section 9.1). This could be viewed as analogous to determining 'regulatory effect' of a targeted perturbation, as discussed further in *section 2.2* and *section 9.1*. Fold change calculations are commonly implemented in genomics and other sciences. However, the drawbacks of fold changes are well known when applied to other technologies such as RNA-seq for differential expression. The calculation is undefined with zero in the denominator and exhibits bias in that it emphasizes small denominators. Conversely, in cases where two counts are high, the ratio between them may understate the difference in counts. We demonstrate this visually in **figure 4.1** below. When considering gRNA counts, we do not believe the ratio between two measurements is an appropriate way to quantify the impact of targeted perturbations. The bias of the fold change transformation has a greater impact on correct identification of enriched guides at smaller read depths and signal strengths. We utilize forward simulations from the CRISPR-Decryptr generative model and evaluate performance of the Log_2 fold-change to demonstrate this phenomenon. We perform 7,000 forward simulations of 10,000 gRNA counts in two conditions from the CRISPR-Decryptr at three read depths (1, 2, and 4 million reads) and calculate the Log_2 fold change as follows:

$$\mathbf{x}_{\text{background}} \sim \text{Multinomial}(\text{Softmax}(\boldsymbol{\theta}_{\text{background}}), \text{Depth})$$
$$\mathbf{x}_{\text{experimental}} \sim \text{Multinomial}(\text{Softmax}(\boldsymbol{\theta}_{\text{background}} + \boldsymbol{\theta}_{\text{experimental}}), \text{Depth})$$
$$\text{Log}_2\text{FC}_i = \text{Log}_2\left(\frac{x_{\text{experimental},i}}{x_{\text{background},i}}\right) \forall\ i \in \{1, \ldots, 10000\},$$

Where $\mathbf{x}_{\text{background}}$, $\mathbf{x}_{\text{experimental}}$, $\boldsymbol{\theta}_{\text{background}}$, and $\boldsymbol{\theta}_{\text{experimental}} \in \mathbb{R}^{10000}$ and Depth $\in \{1000000,\ 2000000,\ 4000000\}$. We set $\boldsymbol{\theta}_{\text{background}}$ to 1, and allow 500 entries in $\boldsymbol{\theta}_{\text{experimental}}$ to be equal to $\mu \in \{0, 0.05, 0.10, \ldots\ldots 1\}$, while the other 9,500 entries are set to zero. We consider the non-zero entries to be effects of "enriched" guides, while the zero entries are effects of "control" guides. For each read depth and value of μ, we calculate the expected number of simulated control guides that are within the top 95th percentile of $\mathrm{Log}_2\mathrm{FC}$, while simultaneously have background counts ($x_{\text{background},i}$) in the bottom 5th percentile. This metric allows us to quantitatively demonstrate the tendency of the Log_2 fold-change to appear extreme in cases where background counts are very small, as well as examine how this

phenomenon is impacted by read depth and how strong the enrichment signal is in the data. In **figure 4.2**, we see that smaller read depths (Depth) and signal (μ) lead to higher numbers of small background counts leading to extreme Log_2 fold-changes in simulated data.

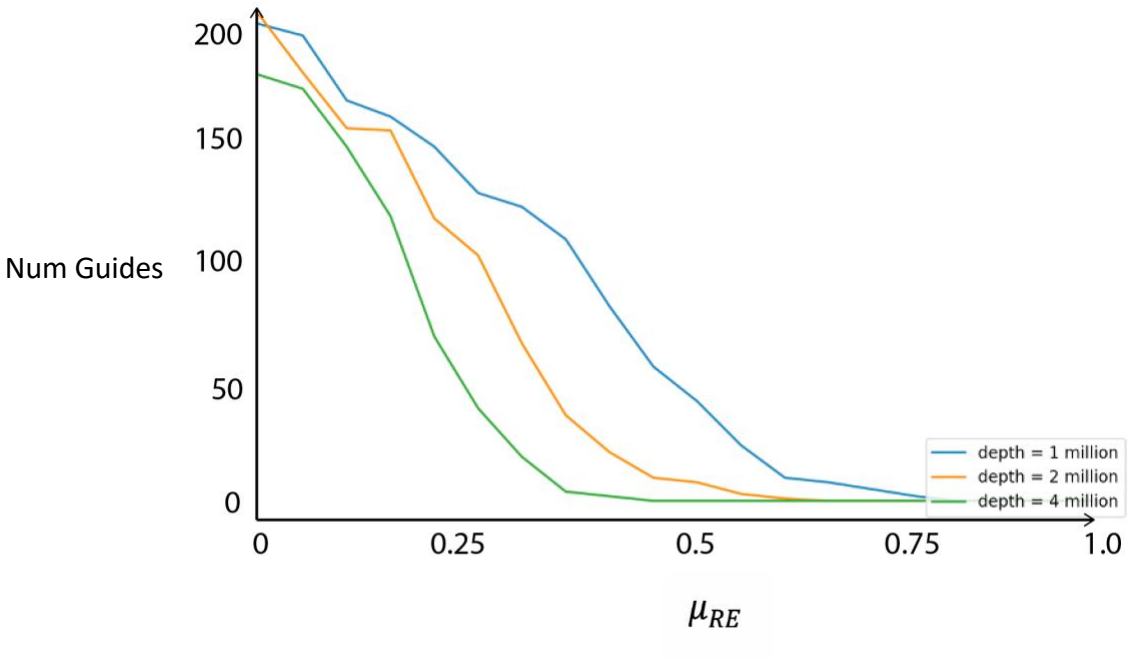
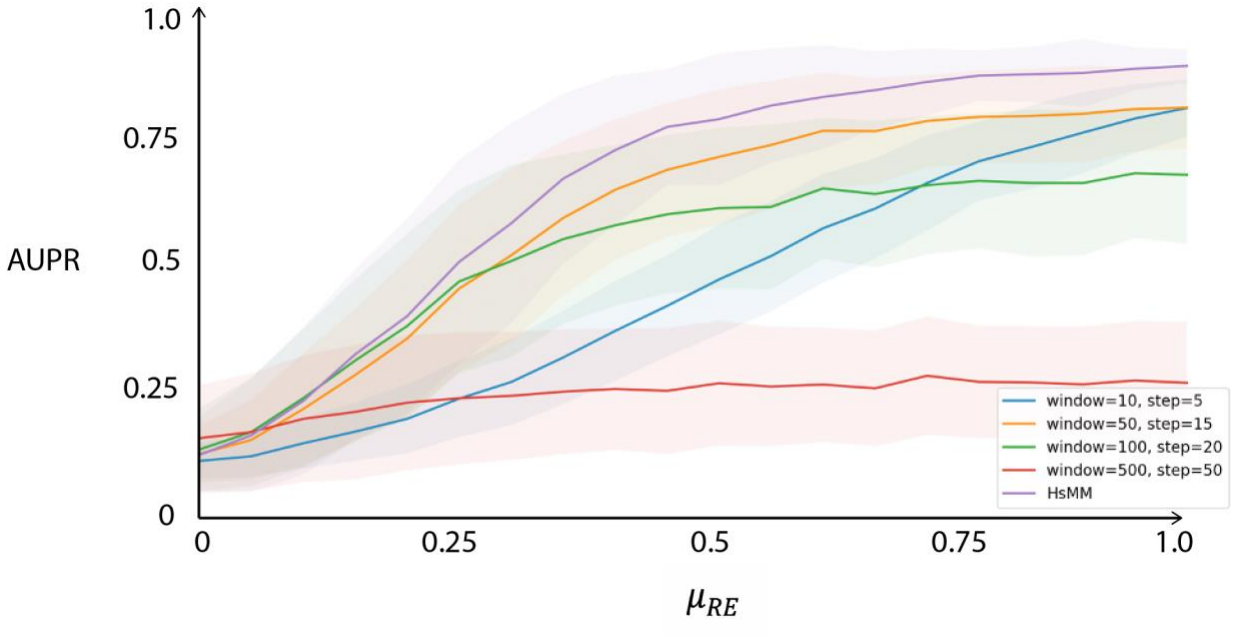


Figure 4.2: Mean number of simulated control guides that are within the top 95th percentile of $\mathrm{Log_2FC}$, while simultaneously have background counts $\mathrm{x_{background,i}}$ in the bottom 5th percentile vs. signal strength (μ_{RE}) for three read depths (1 million, 2 million, and 4 million reads).

A.1.2. Classification of Simulated Regulatory Elements

In this section, we will examine the performance of hidden semi-Markov models vs. sliding windows for classification of enriched regulatory elements. Our goal is to demonstrate the robust ability of HsMMs to fit and classify data and contrast this with the limitations of sliding-window methods with arbitrary parameters. An analysis of HsMM performance on data generated from the CRISPR-Decryptr generative model (as in the previous sections) would have been a task in classifying data using an identical model to that which it was generated from. As such, we do not generate data with a HMM or Negative Binomial state durations, but instead simulated 3kb stretches of chromatin, randomly placing up to three regulatory elements of uniform size $d_{\mathrm{RE}} \sim \mathrm{Uniform}(0, 200)$. Bases within regulatory elements emit a normally distributed regulatory effect $e_{\mathrm{RE}} \sim \mathrm{N}(\mu_{RE}, 1)$, while bases outside of regulatory elements emit a normally distributed regulatory effect $e_{\mathrm{RE}} \sim \mathrm{N}(0, 1)$. We classify regulatory elements using the HsMM model implemented in CRISPR-Decryptr, as well as a sliding window method which calculates a Z-score for each window via Stouffer's Method using four different parameter choices for window and step size. Iterating through 20 equally spaced $\mu_{\mathrm{RE}} \in \{0, 0.05, 0.1, \ldots, 1\}$, we perform 500 simulations at each μ_{RE} and compare AUPR values for each method.



Supplementary Figure 5: Performance of HsMM vs. Windowing Method on Simulated Data Average Precision Recall curves across varying μ_{RE} values, the mean of the regulatory elements effect distribution. Lines indicate the mean AUPR, while the shaded region represents +/- one standard deviation.

The hidden semi-Markov model demonstrates outperformance when compared to sliding window methods, quantified by average precision recall across signal strengths. Our simulation demonstrates that the arbitrary parameters, window size and step size, greatly impact classification performance. As window size becomes large in comparison to regulatory element size (best illustrated in the **window=500, step=50** series) classification performance becomes limited even given high signal strength. At the other extreme, as window size becomes too small (best illustrated by the **window=10, step=5** series) model performance is dramatically reduced at lower signal strengths, as the model is unable to incorporate spatial dependencies of bases outside the window. Even a sliding window with the same size as the expectation of regulatory element sizes (100bps) underperforms the HsMM model. With little a priori knowledge about the size of regulatory elements, parameters for sliding window methods are generally selected arbitrarily. The HsMM model implemented in CRISPR-Decryptr does not have user-defined parameters, but instead fits latent parameters using the expectation maximization algorithm.

A.2. Mathematical Notes

Gaussian process based deconvolution

A continuous-time stochastic process $\{\mathbf{f}_{\mathbf{x}}\}_{\mathbf{x}\in\mathcal{X}}$ is a Gaussian process if and only if for every finite set of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$ in the index set of $\mathcal{X}$

$$[\mathbf{f}_{\mathbf{x}_1}, \mathbf{f}_{\mathbf{x}_2}, \ldots, \mathbf{f}_{\mathbf{x}_M}]^{\mathrm{T}}$$

is a multivariate Gaussian random variable.

Let us consider a Gaussian process $\{\mathbf{f}_{\mathbf{x}}\}_{\mathbf{x}\in\mathcal{X}}$ and a finite index set $X = \{x_i \in \mathbb{R} | i = 1, 2, \ldots, M\}$, then

$$\mathbf{f}_X \sim \mathcal{N}(\mathbf{0}, K_{\mathbf{f}_{\mathbf{X}},\mathbf{f}_{\mathbf{X}}}),$$

where $\mathbf{f}_X \in \mathbb{R}^M$. Moreover, let us assume that the covariance between the random variables $\mathbf{f}_{x_i}$ and $\mathbf{f}_{x_j}$ can be written as a function of the indices x_i and x_j as follows

$$\mathrm{cov}\left(\mathbf{f}_{x_i}, \mathbf{f}_{x_j}\right) = \left[K_{\mathbf{f}_X,\mathbf{f}_X}\right]_{i,j} = k(x_i, x_j; \boldsymbol{\theta}_k) = \left[K_{X,X}\right]_{i,j},$$

where $k(\cdot,\cdot;\boldsymbol{\theta}_k)$ is a covariance function parameterized by the parameters in $\boldsymbol{\theta}_k$. Here, we use the squared exponential (SE) covariance function

$$k(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta}_k) = \alpha^2 \exp\left(-\frac{1}{2\rho^2}(x_i - x_j)^2\right),$$

where α^2 and ρ^2 are the signal variance and the squared length-scale, respectively.

Next, let us consider the following linear transformation of $\mathbf{f}_X$

$$\begin{pmatrix} A \\ I \end{pmatrix} \mathbf{f}_X \sim \mathcal{N}\left(\begin{pmatrix} A \\ I \end{pmatrix}\mathbf{0}, \begin{pmatrix} A \\ I \end{pmatrix} K_{X,X} \begin{pmatrix} A^{\mathrm{T}} & I \end{pmatrix}\right),$$

where $A \in \mathbb{R}^{N\times M}$. The resulting distribution is

$$\begin{pmatrix} A\mathbf{f}_X \\ \mathbf{f}_X \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} AK_{X,X}A^{\mathrm{T}} & AK_{X,X} \\ K_{X,X}A^{\mathrm{T}} & K_{X,X} \end{pmatrix}\right),$$

which is a joint distribution of $A\mathbf{f}_X$ and $\mathbf{f}_X$. Note that $A\mathbf{f}_X \in \mathbb{R}^N$ and $AK_{X,X}A^{\mathrm{T}} \in \mathbb{R}^{N\times N}$.

Our convolution model is $\mathbf{m} = A\mathbf{f} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}^2)\right)$. Therefore, we can write the joint distribution of $\mathbf{m}$ and $\mathbf{f}_X$ as

$$\begin{pmatrix} \mathbf{m} \\ \mathbf{f}_X \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} AK_{X,X}A^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\sigma}^2) & AK_{X,X} \\ K_{X,X}A^{\mathrm{T}} & K_{X,X} \end{pmatrix}\right)$$

Conditioning the joint distribution on the observations $\mathbf{m}$ and their uncertainties $\boldsymbol{\sigma}^2$ leads to

$$\mathbf{f}_X | \mathbf{m}, X, A, \boldsymbol{\sigma}^2, \theta_k \sim \mathcal{N}\left(\bar{\mathbf{f}}_X, \mathrm{cov}(\mathbf{f}_X, \mathbf{f}_X)\right), \text{ where}$$
$$\bar{\mathbf{f}}_X = K_{X,X}A^{\mathrm{T}}\left(AK_{X,X}A^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\sigma}^2)\right)^{-1}\mathbf{m},$$
$$\mathrm{cov}(\mathbf{f}_X, \mathbf{f}_X)) = K_{X,X} - K_{X,X}A^{\mathrm{T}}\left(AK_{X,X}A^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\sigma}^2)\right)^{-1}AK_{X,X}.$$

The marginal likelihood $p(\mathbf{m}|X, A, \boldsymbol{\sigma}^2, \boldsymbol{\theta}_k)$ is

$$p(\mathbf{m}|X, A, \boldsymbol{\sigma}^2, \boldsymbol{\theta}_k) = \int p(\mathbf{m}|\mathbf{f}_X, X, A, \boldsymbol{\sigma}^2, \boldsymbol{\theta}_k) p(\mathbf{f}_X|X, \boldsymbol{\theta}_k)\, \mathrm{d}\mathbf{f}_X,$$

and the resulting log marginal likelihood is

$$\begin{aligned}\log p(\mathbf{m}|X, A, \boldsymbol{\sigma}^2, \boldsymbol{\theta}_k) = &-\frac{1}{2}\mathbf{m}^{\mathrm{T}}\left(AK_{X,X}A^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\sigma}^2)\right)^{-1}\mathbf{m} \\ &-\frac{1}{2}\log\det\left(AK_{X,X}A^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\sigma}^2)\right) - \frac{N}{2}\log 2\pi.\end{aligned}$$

Using the log marginal likelihood, we can formulate the type-II maximum likelihood estimation of $\boldsymbol{\theta}_k$ as follows

$$\hat{\boldsymbol{\theta}}_k = \arg\max_{\boldsymbol{\theta}_k} \log p(\mathbf{m}|X, A, \boldsymbol{\sigma}^2, \boldsymbol{\theta}_k).$$

The gradient of the log marginal likelihood with respect to the parameters α and ρ is

$$\begin{aligned}\nabla \log p(\mathbf{m}|X, A, \boldsymbol{\sigma}^2, \boldsymbol{\theta}_k) &= \begin{pmatrix}\frac{\partial}{\partial\alpha} \\ \frac{\partial}{\partial\rho}\end{pmatrix} \log p(\mathbf{m}|X, A, \boldsymbol{\sigma}^2, \boldsymbol{\theta}_k) \\ &= \frac{1}{2}\begin{pmatrix}\mathrm{tr}\left(\left(\boldsymbol{\beta}\boldsymbol{\beta}^{\mathrm{T}} - \left(AK_{X,X}A^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\sigma}^2)\right)^{-1}\right)A\frac{\partial K_{X,X}}{\partial\alpha}A^{\mathrm{T}}\right) \\ \mathrm{tr}\left(\left(\boldsymbol{\beta}\boldsymbol{\beta}^{\mathrm{T}} - \left(AK_{X,X}A^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\sigma}^2)\right)^{-1}\right)A\frac{\partial K_{X,X}}{\partial\rho}A^{\mathrm{T}}\right)\end{pmatrix},\end{aligned}$$

where $\boldsymbol{\beta} = \left(AK_{X,X}A^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\sigma}^2)\right)^{-1}\mathbf{m}$.

Collapsed multinomial

Let $\mathcal{I}_j$, $j = 1, 2, N_{\text{sets}}$ such that $\bigcup_{j=1}^{N_{\text{sets}}} \mathcal{I}_j = \{1, 2, \ldots, N\}$ and $\bigcap_{j=1}^{N_{\text{sets}}} \mathcal{I}_j = \emptyset$. Moreover, let $\mathbf{x} = \left(\mathbf{x}_{\mathcal{I}_1}^{\mathrm{T}}, \mathbf{x}_{\mathcal{I}_2}^{\mathrm{T}}, \ldots, \mathbf{x}_{\mathcal{I}_{N_{\text{sets}}}}^{\mathrm{T}}\right)^{\mathrm{T}}$ and $\mathbf{p} = \left(\mathbf{p}_{\mathcal{I}_1}^{\mathrm{T}}, \mathbf{p}_{\mathcal{I}_2}^{\mathrm{T}}, \ldots, \mathbf{p}_{\mathcal{I}_{N_{\text{sets}}}}^{\mathrm{T}}\right)^{\mathrm{T}}$. Let us consider collapsed versions of $\mathbf{x}$ and $\mathbf{p}$ as defined as follows

$$\mathbf{x}_{\text{collapsed}}^{\mathcal{I}_j} = \left(\mathbf{x}_{\mathcal{I}_j}^{\mathrm{T}}, \textstyle\sum_{k\in\{1,2,\ldots,N_{\text{sets}}\}\setminus\{j\}} \sum \mathbf{x}_{\mathcal{I}_k}\right)^{\mathrm{T}}$$

and

$$\mathbf{p}_{\text{collapsed}}^{\mathcal{I}_j} = \left(\mathbf{p}_{\mathcal{I}_j}^{\mathrm{T}}, \textstyle\sum_{k\in\{1,2,\ldots,N_{\text{sets}}\}\setminus\{j\}} \sum \mathbf{p}_{\mathcal{I}_k}\right)^{\mathrm{T}}.$$

That is, both $\mathbf{x}_{\text{collapsed}}^{\mathcal{I}_j}$ and $\mathbf{p}_{\text{collapsed}}^{\mathcal{I}_j}$ have $|\mathcal{I}_j| + 1$ elements.

If $\mathbf{x}$ follows a multinomial distribution with parameters M and $\mathbf{p}$

$$\mathbf{x}|M, \mathbf{p} \sim \text{Multinomial}(M, \mathbf{p}),$$

then $\mathbf{x}_{\text{collapsed}}^{j}$ follows the multinomial distribution with parameters M and $\mathbf{p}_{\text{collapsed}}^{\mathcal{I}_j}$

$$\mathbf{x}_{\text{collapsed}}^{j}|M, \mathbf{p}_{\text{collapsed}}^{\mathcal{I}_j} \sim \text{Multinomial}(M, \mathbf{p}_{\text{collapsed}}^{\mathcal{I}_j}).$$

A.3. Performance of CNN model for Repair Prediction

To demonstrate the predictive performance of the convolutional neural network used in the repair outcome prediction step, we benchmarked its performance on held-out data from the original dataset of repair outcomes from Allen et al. To do this, we randomly choose 80% of repair outcome profiles as training data, on which we trained the CNN, a 5-Nearest Neighbor Algorithm, and a predictor that guesses the average profile from the training set independent of sequence. Using remaining 20% of the dataset, we tested these three methods by predicting and evaluating their accuracy using two performance metrics: the sum of the mean squared error (MSE) and KL-divergence between predicted and realized profiles. As reported below (***Table A.3***), the CNN outperforms the other prediction methods as measured by both metrics. It is worth noting that the CNN was trained using an MSE loss function.

	MEAN SQUARED ERROR	KL-DIVERGENCE
CONVOLUTIONAL NEURAL NETWORK	8.83e-05	7.81 Bits
AVERAGE PROFILE	1.23e-04	9.24 Bits
5-NN PREDICTIONS	1.50e-04	11.46 Bits

Table A.3. Sum of Mean Squared Error and KL-Divergence of realized and predicted repair profiles for CNN, Average Profile, and 5-NN algorithms. The CNN outperforms both alternative prediction methods by both metrics.

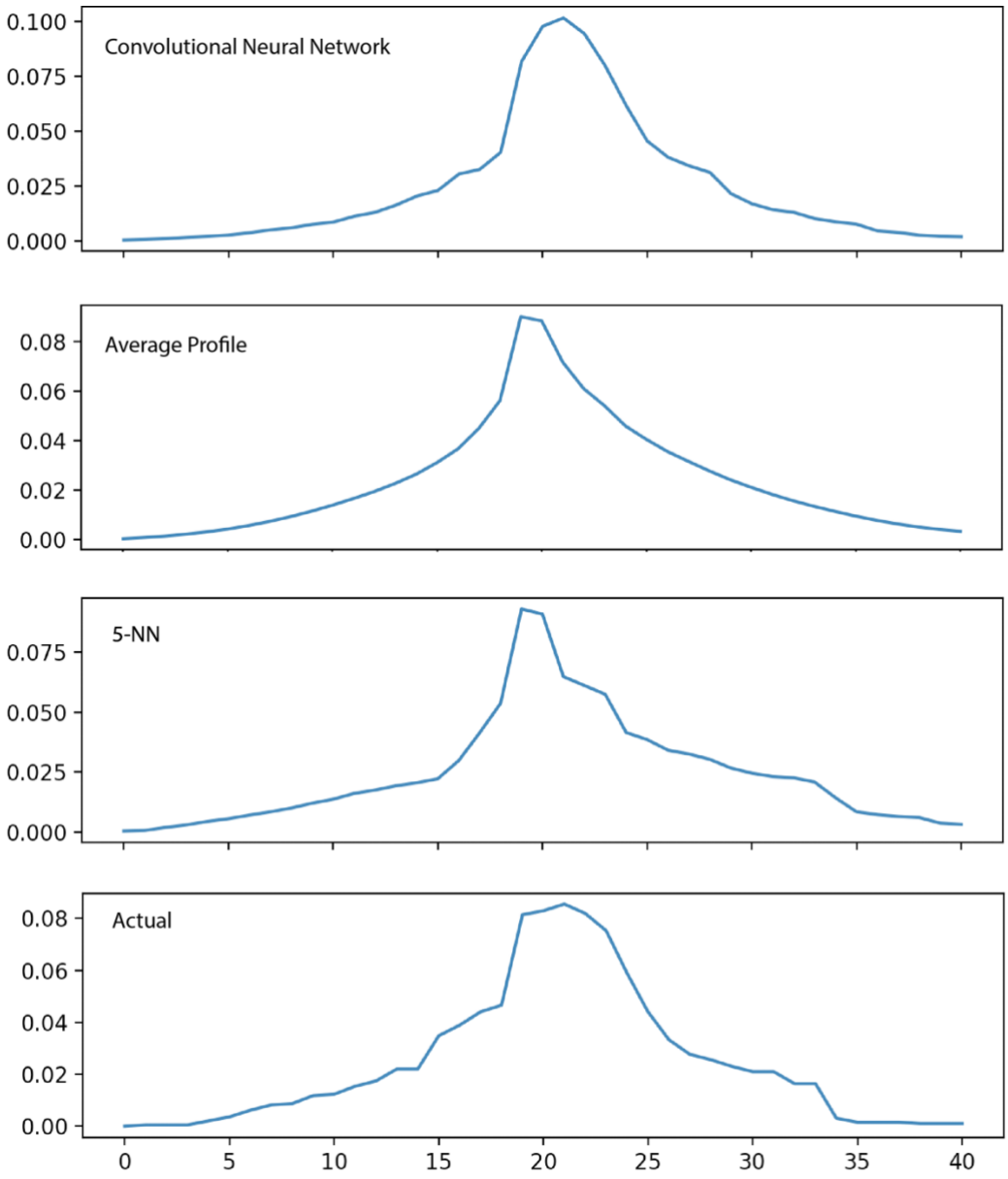


Figure A.3. Predicted profiles and realized repair profile for the target sequence CCAGACAACAAAGCTGCCCTCGGGTAAGGATGTAGGGAGGG from the CNN model, average profile prediction, and the 5-NN.

A.4. Notes on HsMM Hyperparameters

By default, CRISPR-Decryptr initializes a two-state HsMM with hyperparameters based on the length of the region being analyzed and the observed effect signal. Allow N_{bases} to be the length of the region and σ_e to be the standard deviation of the observed effect.

$$\mu_{e,background} \sim N(0, 200 \times N_{bases})$$
$$R_{e,background} = 3$$
$$p_{e,background} = Beta(0.002 \times N_{bases}, 0.501 \times N_{bases})$$
$$\mu_{e,enhancer} \sim N(\sigma_e, 200 \times N_{bases})$$
$$R_{e,background} = 3$$
$$p_{e,background} = Beta(0.002 \times N_{bases}, 0.201 \times N_{bases}), \textit{ or 1 success 200 failures per 1kb}$$

Expressed in terms of pseudocounts, hyperparameters for the parameter $p_{e,s}$ represent 1 success and 500 failures per 1kb for the background state and 1 success and 200 failures per 1kb for the enhancer state.

CRISPR-Decryptr provides an argument to specify prior hyperparameters for the emission and duration distributions of the model. Unlike the default HsMM initialization, these hyperparameters are not functions of the region length or observed signal variance. Recall that the HsMM is comprised of an arbitrary number of states with of negative binomial duration, with normal emission distributions **(Figure A.4)**. The parameter r of the negative binomial can be specified by the user if desired, with a default of three. The precision of the normal distribution emission is determined at each base (*Section 2.5*). As such, parameter posteriors that will be calculated by the EM procedure are the emission means $\mu_{e,s}$, negative binomial parameters $p_{e,s}$.

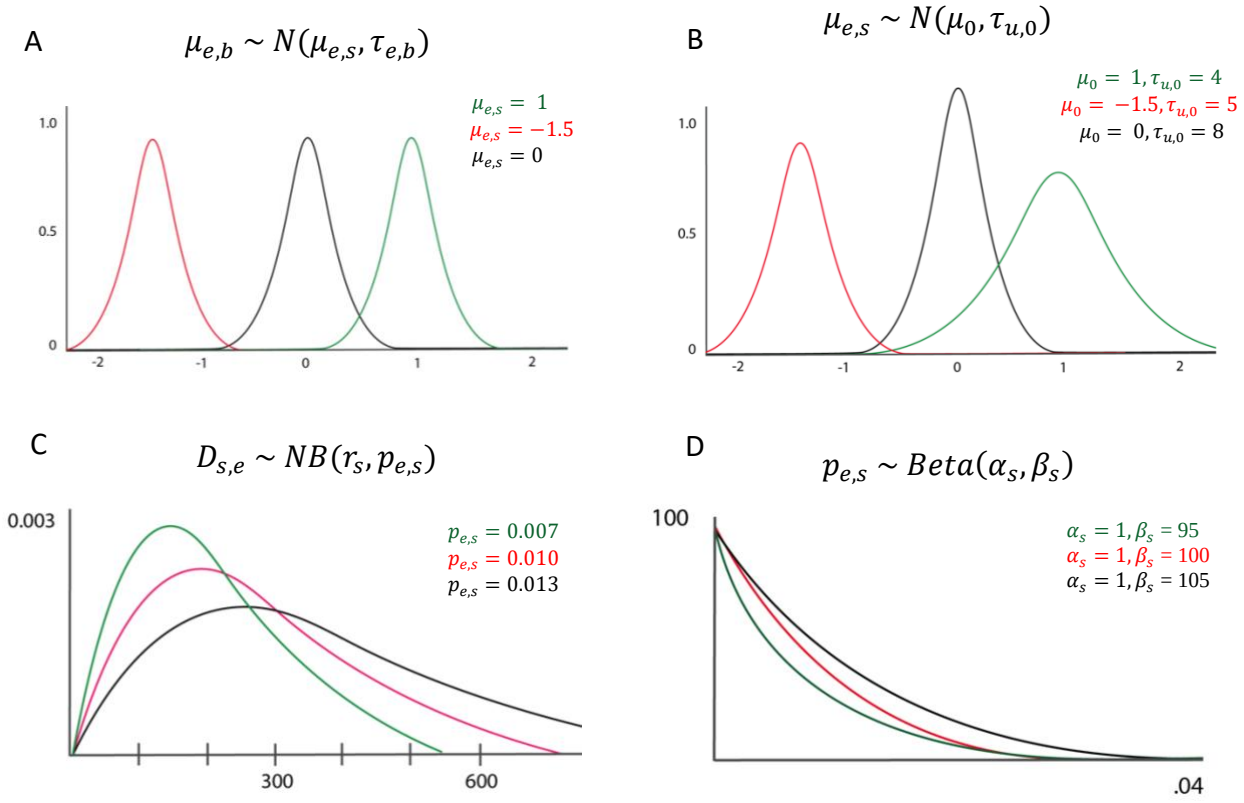


Figure A.4. Example distributions for model parameters and priors with different priors/hyperparameters. **A:** Distribution of effect specific mean at base *b* $\mu_{e,b}$ follows a normal distribution with fixed precision at each base. **B:** Prior distribution of $\mu_{s,e}$ is normally distributed with hyperparameters $\mu_0, \tau_{u,0}$ which can be specified by the user. **C:** State duration is Negative Binomially distributed with user defined parameter r_s and inferred parameter $p_{e,s}$. **D:** Prior distribution for parameter $p_{e,s}$ follows a beta distribution.

The optional *--priors* argument takes a tab delimited file following the format of **Table A.4**. The columns of the table are defined as follows:

Column Name:	**Description:**
mu_0	mean hyperparameter μ_0 (figure A.4.B) of prior distribution $\mu_{s,e}$.
tau_0	precision hyperparameter $\tau_{u,0}$ (figure A.4.B) of prior distribution $\mu_{s,e}$.
NB_r	r parameter for the negative binomial duration distribution (figure A.4.C)
NB_succ	Think of NB distribution (with parameter r) as the distribution of number of succeed/fail coin flips performed (where success probability is $p_{e,s}$) before reaching r successes. This parameter represents the pseudocount of number of successes "pseudo-observed" which constructs the prior. From figure A.4.D $\alpha_s = NB_{succ,s}$. The expected state duration will be $\frac{NB_{fail}+NB_{succ}}{NB_{succ}} \times r$.
NB_fail	This parameter represents the pseudocount of number of failures "pseudo-observed" which constructs the prior (see above).

state	e_mu_0	e_tau_0	NB_r	NB_succ	NB_fail
background	0	5000000	3	1000	500000
silencer	1	5000000	3	1000	200000
enhancer	-1	5000000	3	1000	200000

Table A.4: Example of .tsv file containing hyperparameter information for a three-state model.

A.5. Brief note on gRNA specificity

In the course of constructing CRISPR-Decryptr, we have made note of guide depletion when targeting certain repeat elements or regions of low sequence specificity. To our knowledge, this phenomenon has not been significantly researched, but was noted in the Canver et al. paper where the authors targeted an Alu SINE repeat in BCL11a DHS +62 (**Figure 3.1.1**). CRISPR-Decryptr accounts for guide-specificity by default, however, in our analysis of the CRISPRa CD69 screen (section 3.2), we also see gRNA depletion at repeat elements with low specificity if we disable all guide-specificity options. In this paper, we are unable to provide a biological argument as to why this would be the case in both CRISPR mutagenesis and CRISPR activation screens. However, we are confident that this phenomenon has great potential to lead to false positives. In *section 4.1* we note the propensity of the MAUDE algorithm to pick up on this "non-specificity effect," and subsequently erroneously classify repeat regions as putative regulators. Future research appears indicated if this phenomenon is to be modeled in future methods.

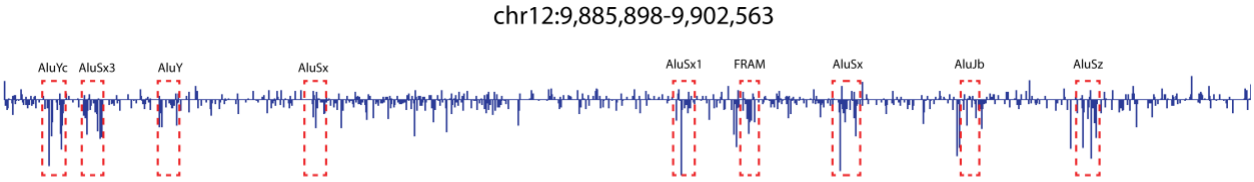


Figure A.4: Selected region downstream of CD69 showcasing gRNA depletion at repeat elements (red boxes) as seen in the inferred regulatory effect (low bin effect, see section 3.2).

Code Availability:

CRISPR-Decryptr code and readme are located at:
https://github.com/anders-w-rasmussen/crispr_decryptr

Data Availability:

gRNA count data and input files for the screens analyzed can be found at the following link:

Acknowledgements:

This research was made possible by the Simons Foundation. RB and MG acknowledge support from the following sources: NIH R01DK103358, Simons Foundation, NSF-IOS-1546218, R35GM122515, NSF CBET-1728858, and NIH R01AI130945

Contributions:

A.R. conceived of the model with guidance and oversight from R.B. and T.Ä. The inference step of the method and Gaussian Process deconvolution were formulated and coded by T.Ä. Off-target, repair outcome prediction, and the Gaussian Process / HsMM iterative procedure were formulated by A.R. The majority of HsMM code is adapted from previous work by done by M.I.G and A.R. in developing the ChromA algorithm. Rules for HsMM parameter updates and variational methods were developed by M.I.G. N.C. was instrumental in advising on high-performance computing considerations. N.S. and J.S. provided important insight into noncoding screens from the viewpoint of experimentalists and were central in bringing the need for this statistical method to A.R. and R.B.'s attention. A.R. did analyses of published data, wrote the paper supplement, and wrote the CRISPR-Decryptr software. All authors contributed to the writing of the manuscript.

Competing Interests:

A.R. owns stock in Editas medicine and 10x Genomics. T.Ä. owns stock in 10x Genomics.

R.B. has ongoing or recent consulting or advisory relationships with Eli Lily, Merus, Merck and Epistemic AI.

References from Main Text:

[1] Wei, L., Lee, D., Law, C. *et al.* Genome-wide CRISPR/Cas9 library screening identified PHGDH as a critical driver for Sorafenib resistance in HCC. *Nat Commun* **10,** 4681 (2019).

[2] Lau, M., Ghazanfar, S., Parkin, A. *et al.* Systematic functional identification of cancer multi-drug resistance genes. *Genome Biol* **21,** 27 (2020).

[3] Fellmann, C., Gowen, B., Lin, P. *et al.* Cornerstones of CRISPR–Cas in drug discovery and therapy. *Nat Rev Drug Discov* **16,** 89–100 (2017).

[4] Fulco, C.P, Munschauer, M. Anyoha, R. Systematic mapping of functional enhancer–promoter connections with CRISPR interference. *Science* **354**, 769–773 (2016).

[5] Li, W., Xu, H., Xiao, T. *et al.* MAGeCK enables robust identification of essential genes from genome-scale CRISPR/Cas9 knockout screens. *Genome Biol* **15,** 554 (2014).

[6] Allen, F., Khodak, A. Behan, F. *et al.* JACKS: joint analysis of CRISPR/Cas9 knockout screens. *Genome Research* **29**, 464-471 (2019)

[7] Hsu, J.Y., Fulco, C.P., Cole, M.A. *et al.* CRISPR-SURF: discovering regulatory elements by deconvolution of CRISPR tiling screen data. *Nat Methods* **15,** 992–993 (2018).

[8] Gelman A, Hill J. *Data analysis using regression and multilevel/hierarchical models* (Cambridge university press, Cambridge, 2006).

[9] Gelman, Andrew, et al. *Bayesian data analysis* (CRC press, Boca Raton, FL, 2013).

[10] Rasmussen, C.E., Williams, C.K. *Gaussian Processes for Machine Learning* (The MIT Press, Cambridge, MA, 2006)

[11] Hsu, P., Scott, D., Weinstein, J. *et al.* DNA targeting specificity of RNA-guided Cas9 nucleases. *Nat Biotechnol* **31,** 827–832 (2013).

[12] Allen, F., Crepaldi, L., Alsinet, C. *et al.* Predicting the mutations generated by repair of Cas9-induced double-strand breaks. *Nat Biotechnol* **37,** 64–72 (2019).

[13] Gabitto, M.I., Rasmussen, A., Wapinski, O. *et al.* Characterizing chromatin landscape from aggregate and single-cell genomic assays using flexible duration modeling. *Nat Commun* **11,** 747 (2020).

[14] Robinson, J., Thorvaldsdóttir, H., Winckler, W. *et al.* Integrative genomics viewer. *Nat Biotechnol* **29,** 24–26 (2011).

[15] Canver, M., Smith, E., Sher, F. *et al. BCL11A* enhancer dissection by Cas9-mediated *in situ* saturating mutagenesis. *Nature* **527,** 192–197 (2015).

[16] Simeonov, D., Gowen, B., Boontanrart, M. *et al.* Discovery of stimulation-responsive immune enhancers with CRISPR activation. *Nature* **549,** 111–115 (2017).

[17] Bauer DE, et al. An Erythroid Enhancer of BCL11A Subject to Genetic Variation Determines Fetal Hemoglobin Level. *Science* **342**, 253–257 (2013).

Reference Genome Data:

CRISPR-Decryptr utilizes the following resources for the hg19 reference genome.

(hg19, GRCh37 Genome Reference Consortium Human Reference 37 (GCA_000001405.1))
http://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/

Encode Mapability Tracks[12]:
http://hgdownload.soe.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeMapability/

Supplement References:

[1] Gelman A, Hill J. *Data analysis using regression and multilevel/hierarchical models* (Cambridge university press, Cambridge, 2006).

[2] Gelman, Andrew, et al. *Bayesian data analysis* (CRC press, Boca Raton, FL, 2013).

[3] Hsu, P., Scott, D., Weinstein, J. *et al.* DNA targeting specificity of RNA-guided Cas9 nucleases. *Nat Biotechnol* **31,** 827–832 (2013).

[4] Allen, F., Crepaldi, L., Alsinet, C. *et al.* Predicting the mutations generated by repair of Cas9-induced double-strand breaks. *Nat Biotechnol* **37,** 64–72 (2019).

[5] Rasmussen, C.E., Williams, C.K. *Gaussian Processes for Machine Learning* (The MIT Press, Cambridge, MA, 2006)

[6] Gabitto, M.I., Rasmussen, A., Wapinski, O. *et al.* Characterizing chromatin landscape from aggregate and single-cell genomic assays using flexible duration modeling. *Nat Commun* **11,** 747 (2020).

[7] Bauer DE, et al. An Erythroid Enhancer of BCL11A Subject to Genetic Variation Determines Fetal Hemoglobin Level. *Science* **342**, 253–257 (2013).

[8] Canver, M., Smith, E., Sher, F. *et al. BCL11A* enhancer dissection by Cas9-mediated *in situ* saturating mutagenesis. *Nature* **527,** 192–197 (2015).

[9] Simeonov, D., Gowen, B., Boontanrart, M. *et al.* Discovery of stimulation-responsive immune enhancers with CRISPR activation. *Nature* **549,** 111–115 (2017).

[10] Fulco, C.P, Munschauer, M. Anyoha, R. Systematic mapping of functional enhancer–promoter connections with CRISPR interference. *Science* **354**, 769–773 (2016).

[11] Hsu, J.Y., Fulco, C.P., Cole, M.A. *et al.* CRISPR-SURF: discovering regulatory elements by deconvolution of CRISPR tiling screen data. *Nat Methods* **15,** 992–993 (2018).

[13] de Boer C.G., Ray J.P. , Hacohen, N. MAUDE: Inferring expression changes in sorting-based CRISPR screens, bioRxiv 819649;

[14] Davis CA, Hitz BC, Sloan CA, et al. The Encyclopedia of DNA elements (ENCODE): data portal update. *Nucleic Acids Res*. 2018;46(D1):D794-D801. doi:10.1093/nar/gkx1081

The followings were referenced in the construction of the CRISPR-Decryptr code:

Bob Carpenter, Andrew Gelman, et al. *Stan: A probabilistic programming language*. Journal of Statistical Software 76(1) 2017.

J Aitchison. 1986. *The statistical analysis of compositional data*. Chapman & Hall, Ltd., GBR.

Fabian Pedregosa, Gaël Varoquaux, et al.. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, **12**, 2825-2830 (2011)

Pauli Virtanen, Ralf Gommers, et al. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, in press.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511790492