SOFTWARE

# Software for the analysis and visualization of deep mutational scanning data

Jesse D. Bloom


## Abstract

**Background:** Deep mutational scanning is a technique to estimate the impacts of mutations on a gene by using deep sequencing to count mutations in a library of variants before and after imposing a functional selection. The impacts of mutations must be inferred from changes in their counts after selection.

**Results:** I describe a software package, `dms_tools`, to infer the impacts of mutations from deep mutational scanning data using a likelihood-based treatment of the mutation counts. I show that `dms_tools` yields more accurate inferences on simulated data than the widely used but statistically biased approach of calculating ratios of counts pre- and post-selection. Using `dms_tools`, one can infer the preference of each site for each amino acid given a single selection pressure, or assess the extent to which these preferences change under different selection pressures. The preferences and their changes can be intuitively visualized with sequence-logo-style plots created using an extension to `weblogo`.

**Conclusions:** `dms_tools` implements a statistically principled approach for the analysis and subsequent visualization of deep mutational scanning data.


**Keywords:** deep mutational scanning; sequence logo; amino-acid preferences

## Background

Deep mutational scanning is a high-throughput experimental technique to assess the impacts of mutations on a protein-coding gene [1]. Figure 1 shows a schematic of deep mutational scanning. A gene is mutagenized, and the library of resulting variants is introduced into cells or viruses, which are then subjected to an experimental selection that enriches for functional variants and depletes non-functional ones. Deep sequencing of the variants pre- and post-selection provides information about the functional impacts of mutations. Since the original description of deep mutational scanning by Fowler *et al* [2], the technique has been applied to a wide range of genes [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14], both to measure mutational tolerance given a single selection pressure as in Figure 1A, or to identify mutations that have different effects under alternative selections as in Figure 1B. New techniques to create comprehensive codon-mutant libraries of genes make it possible to profile all single amino-acid mutations [15, 16, 8, 9, 10], while new techniques for targeted mutagenesis of mammalian genomes enable deep mutational scanning to be applied across the biological spectrum from viruses and bacteria to human cells [17].

A key component of deep mutational scanning is analysis of the data: First, raw reads from the deep sequencing must be processed to count mutations pre- and post-selection. Next, the biological effects of mutations must be inferred from these counts. The first task of processing the reads is idiosyncratic to the specific sequencing strategy used. But the second task of inferring mutational effects from sequencing counts is amenable to more general algorithms. However, only a few such algorithms have been described [18, 19]. Here I present user-friendly software, `dms_tools`, that infers mutational effects from sequencing counts. Before describing the algorithms implemented in `dms_tools` and illustrating its use on existing and simulated data, I first discuss issues associated with inferring mutational effects from sequencing counts.

### The nature of deep mutational scanning data.

The data consist of counts of variants pre- and post-selection. The approach presented here treats each site in the gene separately, ignoring epistatic coupling among mutations. This aspect of the approach should not be construed as a suggestion that interactions among mutations are unimportant; indeed, several studies have used deep mutational scanning to examine pairwise epistasis [14, 20, 21], and techniques

Correspondence: jbloom@fredhutch.org
Division of Basic Sciences and Computational Biology Program, Fred Hutchinson Cancer Research Center, 1100 Fairview Ave N, 98109 Seattle, WA, USA
Full list of author information is available at the end of the article

have been described to obtain linkage between distant sites [22, 23]. However, the exploding combinatorics of multiple mutations (a 500-residue protein has only $19 \times 500 \approx 10^4$ single mutants, but $19^2 \times \frac{500 \times 499}{2} \approx 4 \times 10^7$ double mutants and $19^3 \times \frac{500!}{497! \times 3!} \approx 10^{11}$ triple mutants) make it currently plausible to comprehensively characterize only single mutations to all but the shortest genes. Treating sites independently is therefore not a major limitation for most current datasets. Eventually the approach here might be extended to include coupling among mutations.

The data for each site $r$ is characterized by the sequencing *depth* (total number of counts); let $N_r^{\text{pre}}$, $N_r^{\text{post}}$, $N_r^{s1}$, and $N_r^{s2}$ denote the depth at $r$ for each of the four libraries in Figure 1 (pre-selection, post-selection, selection $s1$, and selection $s2$). Typical depths for current experiments are $N \sim 10^6$. Denote the counts of character $x$ (characters might be nucleotides, amino acids, or codons) at $r$ as $n_{r,x}^{\text{pre}}$, $n_{r,x}^{\text{post}}$, $n_{r,x}^{s1}$, and $n_{r,x}^{s2}$. The values of $n_{r,x}$ for characters $x$ that differ from the wildtype identity $\text{wt}\,(r)$ depend on both the depth $N$ and the average per-site mutation rate. Typical experiments introduce about one mutation per gene, so the average mutation rate is $\overline{\mu} \sim 1/L$ where $L$ is the length of the gene. Therefore, if a 500-codon gene is sequenced at depth $N \sim 10^6$, we expect $N\overline{\mu} \sim 2000$ counts of non-wildtype codons at each site. Since there are 63 mutant codons, the average pre-selection counts for a mutation to a specific $x \neq \text{wt}\,(r)$ will be $n_{r,x}^{\text{pre}} \sim 30$, with counts for most mutations deviating from this average due to biases in creation of the mutant library and randomness in which molecules are sequenced. Counts in the post-selection libraries will further deviate from this average due to selection. Therefore, even at depths $N \sim 10^6$, the actual counts of most mutations will be quite modest.

The goal: inferring site-specific amino-acid preferences. The goal is to estimate the effects of mutations from changes in their counts after selection. Let $\mu_{r,x}$, $f_{r,x}$, $f_{r,x}^{s1}$, and $f_{r,x}^{s2}$ denote the *true* frequencies of $x$ at $r$ that would be observed for the four libraries in Figure 1 if we sampled at infinite depth in both the actual experiment and the sequencing. If we knew these frequencies, we could calculate parameters that reflect the effects of mutations. One parameter that characterizes the effect of mutating $r$ from $\text{wt}\,(r)$ to $x$ for the experiment in Figure 1A is the *enrichment ratio*, which is the relative frequency of mutations to $x$ after selection versus before selection:

$$\phi_{r,x} = \frac{f_{r,x}/f_{r,\text{wt}(r)}}{\mu_{r,x}/\mu_{r,\text{wt}(r)}}. \tag{1}$$

Beneficial mutations have $\phi_{r,x} > 1$, while deleterious ones have $\phi_{r,x} < 1$. A related parameter is the *preference* $\pi_{r,x}$ of $r$ for $x$. At each site, the preferences are simply the enrichment ratios rescaled to sum to one:

$$\pi_{r,x} = \frac{\phi_{r,x}}{\sum_y \phi_{r,y}} = \frac{f_{r,x}/\mu_{r,x}}{\sum_y f_{r,y}/\mu_{r,y}}, \tag{2}$$

or equivalently

$$f_{r,x} = \frac{\pi_{r,x} \times \mu_{r,x}}{\sum_y \pi_{r,y} \times \mu_{r,y}}, \tag{3}$$

where $y$ is summed over all character identities (all nucleotides, codons, or amino acids). The preferences can be intuitively visualized (Figure 1A) and interpreted as the equilibrium frequencies in substitution models for gene evolution [9, 24] (after accounting for uneven mutational rates [25, 26]).

The challenge of statistical inference from finite counts. Equations 1 and 2 are in terms of the true frequencies $\mu_{r,x}$, $f_{r,x}$, etc. But in practice, we only observe the counts in the finite sample of sequenced molecules. The computational challenge is to estimate the preferences (or enrichment ratios) from these counts.

The most naive approach is to simply substitute the counts for the frequencies, replacing Equation 1 with

$$\phi_{r,x} = \frac{\frac{n_{r,x}^{\text{post}} + \mathcal{P}}{n_{r,\text{wt}(r)}^{\text{post}} + \mathcal{P}}}{\frac{n_{r,x}^{\text{pre}} + \mathcal{P}}{n_{r,\text{wt}(r)}^{\text{pre}} + \mathcal{P}}} \tag{4}$$

where $\mathcal{P}$ (often chosen to be one) is a pseudocount added to each count to avoid ratios of zero or infinity.

However, Equation 4 involves ratios of counts with values $\sim 10$ to $100$ – and as originally noted by Karl Pearson [27, 28], ratios estimated from finite counts are statistically biased, with the bias increasing as the magnitude of the counts decrease. This bias can propagate into subsequent analyses, since many statistical tests assume symmetric errors. The problems caused by biases in uncorrected ratios have been noted even in applications such as isotope-ratio mass spectrometry [29] and fluorescent imaging [30], where the counts usually far exceed those in deep mutational scanning.

Taking ratios also abrogates our ability to use the magnitude of the counts to assess our certainty about conclusions. For instance, imagine that at a fixed depth, the counts of a mutation increase from a pre-selection value of 5 to a post-selection value of 10. While this doubling suggests that the mutation might be beneficial, the small counts make us somewhat uncertain of this conclusion. But if the counts increased

from 20 to 40 we would be substantially more certain, and if they increased from 100 to 200 we would be quite sure. So only by an explicit statistical treatment of the counts can we fully leverage the data.

Here I describe a software package, `dms_tools`, that infers mutational effects in a Bayesian framework using a likelihood-based treatment of the counts. This software can be used to infer and visualize site-specific preferences from experiments like Figure 1A, and to infer and visualize differences in preferences under alternative selections from experiments like Figure 1B.

## Implementation and Results

### Algorithm to infer site-specific preferences.

`dms_tools` uses a Bayesian approach to infer site-specific preferences from experiments like those in Figure 1A. The algorithm calculates the likelihoods of the counts given the unknown preferences and mutation / error rates, placing plausible priors over these unknown parameters. The priors correspond to the assumption that all possible identities (e.g. amino acids) have equal preferences, and that the mutation and error rates for each site are equal to the overall average for the gene. MCMC is used to calculate the posterior probability of the preferences given the counts.

This algorithm is a slight modification of that in the *Methods* of [9]; here the algorithm is described anew to explain the implementation in `dms_tools`.

*Optional controls to quantify error rates.*
Some sequencing reads that report a mutation may actually reflect an error introduced during sequencing or PCR rather than an actual mutation that experienced selection. Errors can be quantified by sequencing an unmutated gene, so that any counts at $r$ of $x \neq \text{wt}\,(r)$ for this control reflect errors. In some cases (e.g. sequencing an RNA virus where the post-selection libraries must be reverse-transcribed), error rates for the pre- and post-selection libraries may differ and so be described by different controls. Let $N_r^{\text{errpre}}$ and $N_r^{\text{errpost}}$ be the depth and $n_{r,x}^{\text{errpre}}$ and $n_{r,x}^{\text{errpost}}$ be the counts of $x$ in the pre-selection and post-selection error controls, respectively. Define $\epsilon_{r,x}$ and $\rho_{r,x}$ to be the true frequencies of errors at $r$ from $\text{wt}\,(r)$ to $x$ in the pre- and post-selection controls, respectively.

*Likelihoods of observing specific mutational counts.*
Define vectors of the counts and frequencies for all characters at each site $r$, i.e. $\mathbf{n_r^{pre}} = (\cdots, n_{r,x}^{\text{pre}}, \cdots)$, $\mathbf{n_r^{post}} = (\cdots, n_{r,x}^{\text{post}}, \cdots)$, $\boldsymbol{\mu_r} = (\cdots, \mu_{r,x}, \cdots)$, $\mathbf{f_r} = (\cdots, f_{r,x}, \cdots)$, etc. Also define $\boldsymbol{\pi_r} = (\cdots, \pi_{r,x}, \cdots)$ of the preferences for each $r$, noting that Equation 3 implies $\mathbf{f_r} = \frac{\boldsymbol{\mu_r} \circ \boldsymbol{\pi_r}}{\boldsymbol{\mu_r} \cdot \boldsymbol{\pi_r}}$ where $\circ$ is the Hadamard product.

The likelihoods of some specific set of counts are:

$$\Pr\left(\mathbf{n_r^{errpre}} \mid N_r^{\text{errpre}}, \boldsymbol{\epsilon_r}\right) = \text{Multi}\left(\mathbf{n_r^{errpre}}; N_r^{\text{errpre}}, \boldsymbol{\epsilon_r}\right) \tag{5}$$

$$\Pr\left(\mathbf{n_r^{errpost}} \mid N_r^{\text{errpost}}, \boldsymbol{\rho_r}\right) = \text{Multi}\left(\mathbf{n_r^{errpost}}; N_r^{\text{errpost}}, \boldsymbol{\rho_r}\right) \tag{6}$$

$$\Pr\left(\mathbf{n_r^{pre}} \mid N_r^{\text{pre}}, \boldsymbol{\mu_r}, \boldsymbol{\epsilon_r}\right) = \text{Multi}\left(\mathbf{n_r^{pre}}; N_r^{\text{pre}}, \boldsymbol{\mu_r} + \boldsymbol{\epsilon_r} - \boldsymbol{\delta_r}\right) \tag{7}$$

$$\Pr\left(\mathbf{n_r^{post}} \mid N_r^{\text{post}}, \boldsymbol{\mu_r}, \boldsymbol{\pi_r}, \boldsymbol{\rho_r}\right) = \text{Multi}\left(\mathbf{n_r^{post}}; N_r^{\text{post}}, \frac{\boldsymbol{\mu_r} \circ \boldsymbol{\pi_r}}{\boldsymbol{\mu_r} \cdot \boldsymbol{\pi_r}} + \boldsymbol{\rho_r} - \boldsymbol{\delta_r}\right) \tag{8}$$

where Multi is the multinomial distribution, $\boldsymbol{\delta_r} = (\cdots, \delta_{x,\text{wt}(r)}, \cdots)$ is a vector with the element corresponding to $\text{wt}\,(r)$ equal to one and all other elements zero ($\delta_{xy}$ is the Kronecker delta), and we have assumed that the probability that a site experiences both a mutation and an error is negligibly small.

*Priors over the unknown parameters.*
We specify Dirichlet priors over the parameter vectors:

$$\Pr\left(\boldsymbol{\pi_r}\right) = \text{Dirichlet}\left(\boldsymbol{\pi_r}; \alpha_\pi \times \mathbf{1}\right) \tag{9}$$

$$\Pr\left(\boldsymbol{\mu_r}\right) = \text{Dirichlet}\left(\boldsymbol{\mu_r}; \alpha_\mu \times \mathcal{N}_x \times \mathbf{a_{r,\mu}}\right) \tag{10}$$

$$\Pr\left(\boldsymbol{\epsilon_r}\right) = \text{Dirichlet}\left(\boldsymbol{\epsilon_r}; \alpha_\epsilon \times \mathcal{N}_x \times \mathbf{a_{r,\epsilon}}\right) \tag{11}$$

$$\Pr\left(\boldsymbol{\rho_r}\right) = \text{Dirichlet}\left(\boldsymbol{\rho_r}; \alpha_\rho \times \mathcal{N}_x \times \mathbf{a_{r,\rho}}\right) \tag{12}$$

where $\mathbf{1}$ is a vector of ones, $\mathcal{N}_x$ is the number of characters (64 for codons, 20 for amino acids, 4 for nucleotides), the $\alpha$'s are scalar concentration parameters $> 0$ (by default `dms_tools` sets the $\alpha$'s to one). For codons, the error rate depends on the number of nucleotides being changed. The average error rates $\overline{\epsilon_m}$ and $\overline{\rho_m}$ for codon mutations with $m$ nucleotide changes are estimated as

$$\overline{\epsilon_m} = \frac{1}{L} \sum_r \frac{1}{N_r^{\text{errpre}}} \sum_x n_{r,x}^{\text{errpre}} \times \delta_{m, D_{x,\text{wt}(r)}} \tag{13}$$

$$\overline{\rho_m} = \frac{1}{L} \sum_r \frac{1}{N_r^{\text{errpost}}} \sum_x n_{r,x}^{\text{errpost}} \times \delta_{m, D_{x,\text{wt}(r)}} \tag{14}$$

where $D_{x,\text{wt}(r)}$ is the number of nucleotide differences between $x$ and $\text{wt}\,(r)$. Given these definitions,

$$\mathbf{a_{r,\epsilon}} = \left(\cdots, \sum_m \frac{\overline{\epsilon}_m}{\mathcal{C}_m} \times \delta_{m, D_{x,\text{wt}(r)}}, \cdots\right) \tag{15}$$

$$\mathbf{a_{r,\rho}} = \left(\cdots, \sum_m \frac{\overline{\rho}_m}{\mathcal{C}_m} \times \delta_{m, D_{x,\text{wt}(r)}}, \cdots\right) \tag{16}$$

where $\mathcal{C}_m$ is the number of mutant characters with $m$ changes relative to wildtype (for nucleotides $\mathcal{C}_0 = 1$ and $\mathcal{C}_1 = 3$; for codons $\mathcal{C}_0 = 1$, $\mathcal{C}_1 = 9$, $\mathcal{C}_2 = \mathcal{C}_3 = 27$).

Our prior assumption is that the mutagenesis introduces all mutant characters at equal frequency (this assumption is only plausible for codons if the mutagenesis is at the codon level as in [15, 16, 8, 9, 10]; if mutations are made at the nucleotide level such as by error-prone PCR then characters should be defined as nucleotides). The average per-site mutagenesis rate is estimated as

$$\overline{\mu} = \left(\frac{1}{L}\sum_r \frac{1}{N_r^{\text{pre}}} \sum_{x \neq \text{wt}(r)} n_{r,x}^{\text{pre}}\right) - \sum_{m \geq 1} \overline{\epsilon_m}, \quad (17)$$

so that

$$\mathbf{a_{r,\mu}} = \left(\cdots, \frac{\overline{\mu}}{\mathcal{N}_x - 1} + \delta_{x,\text{wt}(r)} \times [1 - \overline{\mu}], \cdots\right). \quad (18)$$

*Character types: nucleotides, amino acids, or codons.*
`dms_tools` allows four possibilities for the type of character for the counts and preferences. The first three possibilities are simple: the counts and preferences can both be for any of nucleotides, amino acids, or codons.

The fourth possibility is that the counts are for codons, but the preferences for amino acids. In this case, define a function mapping codons to amino acids,

$$\mathbf{A}(\mathbf{w}) = \left(\cdots, \sum_x \delta_{a,\mathcal{A}(x)} \times w_x, \cdots\right) \quad (19)$$

where $\mathbf{w}$ is a 64-element vector of codons $x$, $\mathbf{A}(\mathbf{w})$ is a 20- or 21-element (depending on the treatment of stop codons) vector of amino acids $a$, and $\mathcal{A}(x)$ is the amino acid encoded by $x$. The prior over the preferences $\boldsymbol{\pi}_\mathbf{r}$ is still a symmetric Dirichlet (now only of length 20 or 21), but the priors for $\boldsymbol{\mu}_\mathbf{r}$, $\boldsymbol{\epsilon}_\mathbf{r}$, and $\boldsymbol{\rho}_\mathbf{r}$ are now Dirichlets parameterized by $\mathbf{A}(\mathbf{a_{r,\mu}})$, $\mathbf{A}(\mathbf{a_{r,\epsilon}})$ and $\mathbf{A}(\mathbf{a_{r,\rho}})$ rather than $\mathbf{a_{r,\mu}}$, $\mathbf{a_{r,\epsilon}}$, and $\mathbf{a_{r,\rho}}$. The likelihoods are computed in terms of these transformed vectors after similarly transforming the counts to $\mathbf{A}(\mathbf{n_r^{pre}})$, $\mathbf{A}(\mathbf{n_r^{post}})$, $\mathbf{A}(\mathbf{n_r^{errpre}})$, and $\mathbf{A}(\mathbf{n_r^{errpost}})$.

*Implementation.*
The program `dms_inferprefs` in the `dms_tools` package infers the preferences by using `pystan` [31] to perform MCMC over the posterior defined by the product of the likelihoods and priors in Equations 5, 6, 7, 8, 9, 10, 11, and 12. The program runs four chains from different initial values, and checks for convergence by ensuring that the Gelman-Rubin statistic $\hat{R}$ [32] is $< 1.1$ and the effective sample size is $> 100$; the number of MCMC iterations is increased until these convergence is achieved. The program `dms_logoplot` in the `dms_tools` package visualizes the posterior mean preferences via an extension to `weblogo` [33].

## Inferring preferences with `dms_tools`.

*Application to actual datasets.*
Figures 2 and 3 illustrate application of `dms_tools` to two existing datasets [10, 11]. The programs require as input only simple text files listing the counts of each character identity at each site. As the figures show, the `dms_inferprefs` and `dms_logoplot` programs can process these input files to infer and visualize the preferences with a few simple commands. Error controls can be included when available (they are not for Figure 2, but are for Figure 3). The runtime for the MCMC depends on the gene length and character type (codons are slowest, nucleotides fastest) – but if the inference is parallelized across multiple CPUs (using the `--ncpus` option of `dms_inferprefs`), the inference should take no more than a few hours. As shown in Figures 2 and 3, the visualizations can overlay information about protein structure onto the preferences.

Figures 2 and 3 also illustrate use of `dms_correlate` to assess the correlation between preferences inferred from different biological replicates [34] of the experiment. The inclusion and analysis of such replicates provide the only sure way to fully assess the sources of noise associated with deep mutational scanning.

*Testing on simulated data.*
To test the accuracy of preference-inference by `dms_tools`, I simulated deep mutational scanning counts using the preferences in Figure 2, both with and without errors quantified by appropriate controls. I then used `dms_tools` to infer preferences from the simulated data, and also made similar inferences using simple ratio estimation (Equation 4). Figure 4 shows the inferred preferences versus the actual values used to simulate the data. For simulations with mutation counts (quantified by the product $N\overline{\mu}$ of the depth and average per-site mutation rate) $\sim$ 1000 to 2000, the inferences are quite accurate. Inferences made by `dms_tools` are always more accurate than those obtained by simply taking ratios of mutation counts.

## Algorithm to infer differential preferences.

As shown in Figure 1B, a useful extension to the experiment in Figure 1A is to subject the functional variants to two different selection pressures to identify mutations favored by one pressure versus the other. While this experiment could in principle by analyzed by simply comparing the initial unselected

mutants to the final variants after the two alternative selections, this approach is non-ideal. In experiments like Figure 1A, many mutations are enriched or depleted to some extent by selection, since a large fraction of random mutations affect protein function [35, 36, 37, 38, 39]. Therefore, the assumption that all mutations are equally tolerated (i.e. the preferences for a site are all equal, or the enrichment ratios are all one) is not a plausible null hypothesis for Figure 1A. For this reason, `dms_tools` simply infers the preferences given a uniform Dirichlet prior rather than trying to pinpoint some subset of sites with unequal preferences.

But in Figure 1B, the assumption that most mutations will be similarly selected is a plausible null hypothesis, since we expect to alternative selections to have markedly different effects on only a small subset of mutations (typically, major constraints related to protein folding and stability will be relatively conserved across different selections on the same protein). Therefore, `dms_tools` uses a different algorithm to infer the differential preferences under the two selections. This algorithm combines a prior that mildly favors differential preferences of zero with a likelihood-based analysis of the mutation counts to estimate posterior probabilities over the differential preferences.

*Definition of the differential preferences.*
Given an experiment like Figure 1B, let $f_{r,x}^{\text{start}}$ be the true frequency of character $x$ at site $r$ in the starting library (equivalent to the frequency $f_{r,x}^{\text{post}}$ in the figure), and let $f_{r,x}^{s1}$ and $f_{r,x}^{s2}$ be the frequencies after selections $s1$ and $s2$, respectively. The differential preference $\Delta\pi_{r,x}$ for $x$ at $r$ in $s2$ versus $s1$ is defined by:

$$f_{r,x}^{s1} = \frac{f_{r,x}^{\text{start}} \times \pi_{r,x}^{s1}}{\sum_y f_{r,y}^{\text{start}} \times \pi_{r,y}^{s1}} \tag{20}$$

$$f_{r,x}^{s2} = \frac{f_{r,x}^{\text{start}} \times \left(\pi_{r,x}^{s1} + \Delta\pi_{r,x}\right)}{\sum_y f_{r,y}^{\text{start}} \times \left(\pi_{r,y}^{s1} + \Delta\pi_{r,y}\right)} \tag{21}$$

where $\pi_{r,x}^{s1}$ is the "control preference" and is treated as a nuisance parameter, and we have the constraints

$$0 = \sum_x \Delta\pi_{r,x} \tag{22}$$

$$0 \le \pi_{r,x}^{s1} + \Delta\pi_{r,x} \le 1. \tag{23}$$

If there is no difference in the effect of $x$ at $r$ between selections $s1$ and $s2$, then $\Delta\pi_{r,x} = 0$. If $x$ at $r$ is more preferred by $s2$ than $s1$, then $\Delta\pi_{r,x} > 0$; conversely if $x$ at $r$ is more preferred by $s1$ than $s2$, then $\Delta\pi_{r,x} < 0$ (see Figure 5A).

*Likelihoods of observing specific mutational counts.*
Define vectors of the counts as $\mathbf{n_r^{start}} = \left(\cdots, n_{r,x}^{\text{start}}, \cdots\right)$ for the post-selection functional variants that are subjected to the further selections, and as $\mathbf{n_r^{s1}} = \left(\cdots, n_{r,x}^{s1}, \cdots\right)$ and $\mathbf{n_r^{s2}} = \left(\cdots, n_{r,x}^{s2}, \cdots\right)$ for selections $s1$ and $s2$. We again allow an error control, but now assume that the same control applies to all three libraries (since they are all sequenced after a selection), and define the counts for this control as $\mathbf{n_r^{err}} = \left(\cdots, n_{r,x}^{\text{err}}, \cdots\right)$; the true error frequencies are denoted by $\xi_{r,x}$. Define vectors of the frequencies, errors, control preferences, and differential preferences: $\mathbf{f_r^{start}} = \left(\cdots, f_{r,x}^{\text{start}}, \cdots\right)$, $\mathbf{f_r^{s1}} = \left(\cdots, f_{r,x}^{s1}, \cdots\right)$, $\mathbf{f_r^{s2}} = \left(\cdots, f_{r,x}^{s2}, \cdots\right)$, $\boldsymbol{\xi_r} = \left(\cdots, \xi_{r,x}, \cdots\right)$, $\boldsymbol{\pi_r^{s1}} = \left(\cdots, \pi_{r,x}^{s1}, \cdots\right)$, and $\boldsymbol{\Delta\pi_r} = \left(\cdots, \Delta\pi_{r,x}, \cdots\right)$. Equations 20 and 21 imply $\mathbf{f_r^{s1}} = \frac{\mathbf{f_r^{starat} \circ \pi_r^{s1}}}{\mathbf{f_r^{start} \cdot \pi_r^{s1}}}$ and $\mathbf{f_r^{s2}} = \frac{\mathbf{f_r^{start} \circ \left(\pi_r^{s1} + \Delta\pi_r\right)}}{\mathbf{f_r^{start} \cdot \left(\pi_r^{s1} + \Delta\pi_r\right)}}$.

The likelihoods of the counts will be multinomially distributed around the "true" frequencies, so

$$\Pr\left(\mathbf{n_r^{err}} \mid N_r^{\text{err}}, \boldsymbol{\xi_r}\right) = \text{Multi}\left(\mathbf{n_r^{err}}; N_r^{\text{err}}, \boldsymbol{\xi_r}\right) \tag{24}$$

$$\Pr\left(\mathbf{n_r^{start}} \mid N_r^{\text{start}}, \mathbf{f_r^{start}}, \boldsymbol{\xi_r}\right) = \text{Multi}\left(\mathbf{n_r^{start}}; N_r^{\text{start}}, \mathbf{f_r^{start}} + \boldsymbol{\xi_r} - \boldsymbol{\delta_r}\right) \tag{25}$$

$$\Pr\left(\mathbf{n_r^{s1}} \mid N_r^{s1}, \mathbf{f_r^{start}}, \boldsymbol{\pi_r^{s1}}, \boldsymbol{\xi_r}\right) = \text{Multi}\left(\mathbf{n_r^{s1}}; N_r^{s1}, \frac{\mathbf{f_r^{start} \circ \pi_r^{s1}}}{\mathbf{f_r^{start} \cdot \pi_r^{s1}}} + \boldsymbol{\xi_r} - \boldsymbol{\delta_r}\right) \tag{26}$$

$$\Pr\left(\mathbf{n_r^{s2}} \mid N_r^{s2}, \mathbf{f_r^{start}}, \boldsymbol{\pi_r^{s1}}, \boldsymbol{\Delta\pi_r}, \boldsymbol{\xi_r}\right) = \text{Multi}\left(\mathbf{n_r^{s2}}; N_r^{s2}, \frac{\mathbf{f_r^{start} \circ \left(\Delta\pi_r + \pi_r^{s1}\right)}}{\mathbf{f_r^{start} \cdot \left(\Delta\pi_r + \pi_r^{s1}\right)}} + \boldsymbol{\xi_r} - \boldsymbol{\delta_r}\right) \tag{27}$$

where we have assumed that the probability that a site experiences a mutation and an error in the same molecule is negligibly small.

*Priors over the unknown parameters.*
We specify Dirichlet priors over the parameter vectors:

$$\Pr\left(\boldsymbol{\pi_r^{s1}}\right) = \text{Dirichlet}\left(\boldsymbol{\pi_r^{s1}}; \alpha_{\pi^{s1}} \times \mathbf{1}\right) \tag{28}$$

$$\Pr\left(\boldsymbol{\xi_r}\right) = \text{Dirichlet}\left(\boldsymbol{\xi_r}; \alpha_\xi \times \mathcal{N}_x \times \mathbf{a_{r,\xi}}\right) \tag{29}$$

$$\Pr\left(\mathbf{f_r^{start}}\right) = \text{Dirichlet}\left(\mathbf{f_r^{start}}; \alpha_{\text{start}} \times \mathcal{N}_x \times \mathbf{a_{r,start}}\right) \tag{30}$$

$$\Pr\left(\boldsymbol{\Delta\pi_r} \mid \boldsymbol{\pi_r^{s1}}\right) = \text{Dirichlet}\left(\boldsymbol{\Delta\pi_r}; \alpha_{\Delta\pi} \times \mathcal{N}_x \times \boldsymbol{\pi_r^{s1}}\right) - \boldsymbol{\pi_r^{s1}} \tag{31}$$

where `dms_tools` by default sets all the scalar concentration parameters ($\alpha$'s) to one except $\alpha_{\Delta\pi}$, which is set to two corresponding to a weak expectation that

the $\Delta\pi$ values are close to zero. The average error rate $\overline{\xi_m}$ for mutations with $m$ nucleotide changes is

$$\overline{\xi_m} = \frac{1}{L}\sum_r \frac{1}{N_r^{\rm err}} \sum_x n_{r,x}^{\rm err} \times \delta_{m,D_{x,{\rm wt}(r)}}, \quad (32)$$

and so

$$\mathbf{a_{r,\xi}} = \left(\cdots, \sum_m \frac{\overline{\xi}_m}{\mathcal{C}_m} \times \delta_{m,D_{x,{\rm wt}(r)}}, \cdots\right). \quad (33)$$

Our prior assumption is that all mutations are at equal frequency in the starting library (this assumption is unlikely to be true if the starting library has already been subjected to some selection, but we lack a rationale for a more informative prior). The average mutation frequency in the starting library is

$$\overline{f^{\rm start}} = \left(\frac{1}{L}\sum_r \frac{1}{N_r^{\rm start}} \sum_{x\neq {\rm wt}(r)} n_{r,x}^{\rm start}\right) - \sum_{m\geq 1} \overline{\xi_m}, \quad (34)$$

and so

$$\mathbf{a_{r,start}} = \left(\cdots, \frac{\overline{f^{\rm start}}}{\mathcal{N}_x - 1} + \delta_{x,{\rm wt}(r)} \times \left[1 - \overline{f^{\rm start}}\right], \cdots\right). \quad (35)$$

*Implementation.*
The program `dms_inferdiffprefs` in the `dms_tools` package infers the differential preferences by performing MCMC over the posterior defined by the product of the likelihoods and priors in Equations 24, 25, 26, 27, 28, 29, 30, and 31. The MCMC is performed as described for the preferences, and characters can again be any of nucleotides, amino acids, or codons. The program `dms_logoplot` visualizes the posterior mean differential preferences via an extension to `weblogo` [33]. In addition, `dms_inferdiffprefs` also creates text files that give the posterior probability that $\Delta\pi_{r,x} > 0$ or $< 0$ (in other words, the probability that there is differential selection).

### Inferring differential preference with `dms_tools`.

To test the accuracy of differential preference inference by `dms_tools`, I simulated an experiment like that in Figure 1B with the starting counts based on Melnikov *et al*'s actual deep mutational scanning data of a Tn5 transposon [10]. As show by Figure 5, `dms_inferdiffprefs` accurately infers the differential preferences at typical experimental depths. The results are easily visualized with `dms_logoplot`. To provide a second illustration of differential preferences, Figure 6 shows an analysis of the data obtained by Wu *et al* when they performed an experiment like that in Figure 1B on nucleotide mutants of the influenza NS gene in the presence or absence of interferon treatment.

## Conclusions

`dms_tools` is a freely available software package that uses a statistically principled approach to analyze deep mutational scanning data. This paper shows that `dms_tools` accurately infers preferences and differential preferences from data simulated under realistic parameters. As the figures illustrate, `dms_tools` can also be applied to actual data with a few simple commands. The intuitive visualizations created by `dms_tools` assist in interpreting the results. As deep mutational scanning continues to proliferate as an experimental technique [1], `dms_tools` can be applied to analyze the data for purposes such as guiding protein engineering [3, 10], understanding sequence-structure-function relationships [4, 5, 7, 14, 20], informing the development of better evolutionary models for sequence analysis [9, 24], and probing the biology of viruses and cells [6, 8, 11, 12, 13, 17].

## Availability and requirements

- **Project name:** `dms_tools`
- **Project home page:**
  - Documentation and installation instructions: http://jbloom.github.io/dms_tools/
  - Source code: https://github.com/jbloom/dms_tools
- **Operating system(s):** Linux
- **Programming language:** `Python`
- **Other requirements:** `pystan`, `weblogo`
- **License:** GNU GPLv3
- **Restrictions to use by non-academics:** None

## Data and code for figures in this paper.

The data and computer code used to generate the figures are in the tagged version of the `dms_tools` source code at https://github.com/jbloom/dms_tools in `examples` subdirectory. The `LaTex` source for this paper is in the `paper` subdirectory.

**Competing interests**
The author declares that he has no competing interests.

**Author's contributions**
JDB designed the algorithms, wrote the software, performed the analyses, and wrote the paper.

**Acknowledgements**
Thanks to Alec Heckert for assistance in testing `dms_tools`, to Erick Matsen for the excellent suggestion to use `pystan` for MCMC, to Nicholas Wu for providing the mutational counts data from [13], and to Orr Ashenberg and Hugh Haddox for helpful comments on the manuscript. This work was supported by the NIGMS of the NIH under grant R01GM102198.

**References**


1. Fowler, D.M., Fields, S.: Deep mutational scanning: a new style of protein science. Nature methods **11**(8), 801–807 (2014)
2. Fowler, D.M., Araya, C.L., Fleishman, S.J., Kellogg, E.H., Stephany, J.J., Baker, D., Fields, S.: High-resolution mapping of protein sequence-function relationships. Nat. Methods **7**(9), 741–746 (2010)
3. Traxlmayr, M.W., Hasenhindl, C., Hackl, M., Stadlmayr, G., Rybka, J.D., Borth, N., Grillari, J., Rüker, F., Obinger, C.: Construction of a stability landscape of the CH3 domain of human IgG1 by combining directed evolution with high throughput sequencing. J. Mol. Biol. (2012)
4. McLaughlin Jr, R.N., Poelwijk, F.J., Raman, A., Gosal, W.S., Ranganathan, R.: The spatial architecture of protein function and adaptation. Nature **491**(7422), 138 (2012)
5. Starita, L.M., Pruneda, J.N., Lo, R.S., Fowler, D.M., Kim, H.J., Hiatt, J.B., Shendure, J., Brzovic, P.S., Fields, S., Klevit, R.E.: Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis. Proc. Natl. Acad. Sci. USA **110**(14), 1263–1272 (2013)
6. Melamed, D., Young, D.L., Gamble, C.E., Miller, C.R., Fields, S.: Deep mutational scanning of an RRM domain of the *Saccharomyces cerevisiae* poly (A)-binding protein. RNA **19**(11), 1537–1551 (2013)
7. Roscoe, B.P., Thayer, K.M., Zeldovich, K.B., Fushman, D., Bolon, D.N.: Analyses of the effects of all ubiquitin point mutants on yeast growth rate. J. Mol. Biol. (2013)
8. Firnberg, E., Labonte, J.W., Gray, J.J., Ostermeier, M.: A comprehensive, high-resolution map of a gene's fitness landscape. Mol. Biol. Evol. **31**(6), 1581–1592 (2014)
9. Bloom, J.D.: An experimentally determined evolutionary model dramatically improves phylogenetic fit. Molecular Biology and Evolution **30**, 1956–1978 (2014). http://mbe.oxfordjournals.org/content/31/8/1956
10. Melnikov, A., Rogov, P., Wang, L., Gnirke, A., Mikkelsen, T.S.: Comprehensive mutational scanning of a kinase *in vivo* reveals context-dependent fitness landscapes. Nucleic Acids Research **42**, 112 (2014)
11. Thyagarajan, B., Bloom, J.D.: The inherent mutational tolerance and antigenic evolvability of influenza hemagglutinin. eLife **3**, 03300 (2014). http://elifesciences.org/content/3/e03300
12. Wu, N.C., Young, A.P., Al-Mawsawi, L.Q., Olson, C.A., Feng, J., Qi, H., Chen, S.-H., Lu, I.-H., Lin, C.-Y., Chin, R.G., *et al.*: High-throughput profiling of influenza A virus hemagglutinin gene at single-nucleotide resolution. Scientific reports **4**, 4942 (2014)
13. Wu, N.C., Young, A.P., Al-Mawsawi, L.Q., Olson, C.A., Feng, J., Qi, H., Luan, H.H., Li, X., Wu, T.-T., Sun, R.: High-throughput identification of loss-of-function mutations for anti-interferon activity in the influenza A virus NS segment. Journal of virology **88**(17), 10157–10164 (2014)
14. Olson, C.A., Wu, N.C., Sun, R.: A comprehensive biophysical description of pairwise epistasis throughout an entire protein domain. Current Biology **24**(22), 2643–2651 (2014)
15. Firnberg, E., Ostermeier, M.: PFunkel: efficient, expansive, user-defined mutagenesis. PLoS One **7**, 52031 (2012)
16. Jain, P.C., Varadarajan, R.: A rapid, efficient, and economical inverse polymerase chain reaction-based method for generating a site saturation mutant library. Analytical Biochemistry **449**, 90–98 (2014)
17. Findlay, G.M., Boyle, E.A., Hause, R.J., Klein, J.C., Shendure, J.: Saturation editing of genomic regions by multiplex homology-directed repair. Nature **513**(7516), 120–123 (2014)
18. Fowler, D.M., Araya, C.L., Gerard, W., Fields, S.: Enrich: software for analysis of protein function by enrichment and depletion of variants. Bioinformatics **27**(24), 3430–3431 (2011)
19. Bank, C., Hietpas, R.T., Wong, A., Bolon, D.N., Jensen, J.D.: A bayesian mcmc approach to assess the complete distribution of fitness effects of new mutations: uncovering the potential for adaptive walks in challenging environments. Genetics **196**(3), 841–852 (2014)
20. Araya, C.L., Fowler, D.M., Chen, W., Muniez, I., Kelly, J.W., Fields, S.: A fundamental protein property, thermodynamic stability, revealed solely from large-scale measurements of protein function. Proceedings of the National Academy of Sciences **109**(42), 16858–16863 (2012)
21. Bank, C., Hietpas, R.T., Jensen, J.D., Bolon, D.N.: A systematic survey of an intragenic epistatic landscape. Molecular biology and evolution **32**(1), 229–238 (2015)
22. Hiatt, J.B., Patwardhan, R.P., Turner, E.H., Lee, C., Shendure, J.: Parallel, tag-directed assembly of locally derived short sequence reads. Nat. Methods **7**(2), 119–122 (2010)
23. Wu, N.C., De La Cruz, J., Al-Mawsawi, L.Q., Olson, C.A., Qi, H., Luan, H.H., Nguyen, N., Du, Y., Le, S., Wu, T.-T., *et al.*: HIV-1 quasispecies delineation by tag linkage deep sequencing. PloS one **9**(5), 97505 (2014)
24. Bloom, J.D.: An experimentally informed evolutionary model improves phylogenetic fit to divergent lactamase homologs. Molecular Biology and Evolution **31**, 2753–2769 (2014). http://mbe.oxfordjournals.org/content/31/10/2753
25. Yampolsky, L.Y., Stoltzfus, A.: The exchangeability of amino acids in proteins. Genetics **170**(4), 1459–1472 (2005)
26. Stoltzfus, A., Yampolsky, L.Y.: Climbing mount probable: mutation as a cause of nonrandomness in evolution. Journal of heredity **100**(5), 637–647 (2009)
27. Pearson, K.: Mathematical contributions to the theory of evolution. On a form of spurious correlation which may arise when indices are used in the measurement of organs. Proceedings of the royal society of london **60**(359-367), 489–498 (1896)
28. Pearson, K.: On the constants of index-distributions as deduced from the like constants for the components of the ratio, with special reference to the opsonic index. Biometrika **7**(4), 531–541 (1910). doi:10.1093/biomet/7.4.531
29. Ogliore, R., Huss, G., Nagashima, K.: Ratio estimation in SIMS analysis. Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **269**(17), 1910–1918 (2011). doi:10.1016/j.nimb.2011.04.120
30. Van Kempen, G., Van Vliet, L.: Mean and variance of ratio estimators used in fluorescence ratio imaging. Cytometry **39**(4), 300–305 (2000)
31. Stan Development Team: PyStan: the Python interface to Stan, Version 2.5.0 (2014). http://mc-stan.org/pystan.html
32. Gelman, A., Rubin, D.B.: Inference from iterative simulation using multiple sequences. Statistical science, 457–472 (1992)
33. Crooks, G.E., Hon, G., Chandonia, J.-M., Brenner, S.E.: Weblogo: a sequence logo generator. Genome research **14**(6), 1188–1190 (2004). doi:10.1101/gr.849004
34. Blainey, P., Krzywinski, M., Altman, N.: Points of significance: Replication. Nature methods **11**(9), 879–880 (2014)
35. Shortle, D., Lin, B.: Genetic analysis of staphylococcal nuclease: identification of three intragenic "global" suppressors of nuclease-minus mutations. Genetics **110**, 539–555 (1985)
36. Rennell, D., Bouvier, S.E., Hardy, L.W., Poteete, A.R.: Systematic mutation of bacteriophage T4 lysozyme. J. Mol. Biol. **222**, 67–87 (1991)
37. Shafikhani, S., Siegel, R.A., Ferrari, E., Schellenberger, V.: Generation of large libraries of random mutants in *Bacillus subtilis* by PCR-based plasmid multimerization. BioTechniques **23**, 304–310 (1997)
38. Guo, H.H., Choe, J., Loeb, L.A.: Protein tolerance to random amino acid change. Proc. Natl. Acad. Sci. USA **101**, 9205–9210 (2004)
39. Bloom, J.D., Silberg, J.J., Wilke, C.O., Drummond, D.A., Adami, C., Arnold, F.H.: Thermodynamic prediction of protein neutrality. Proc. Natl. Acad. Sci. USA **102**, 606–611 (2005)

**Figures**

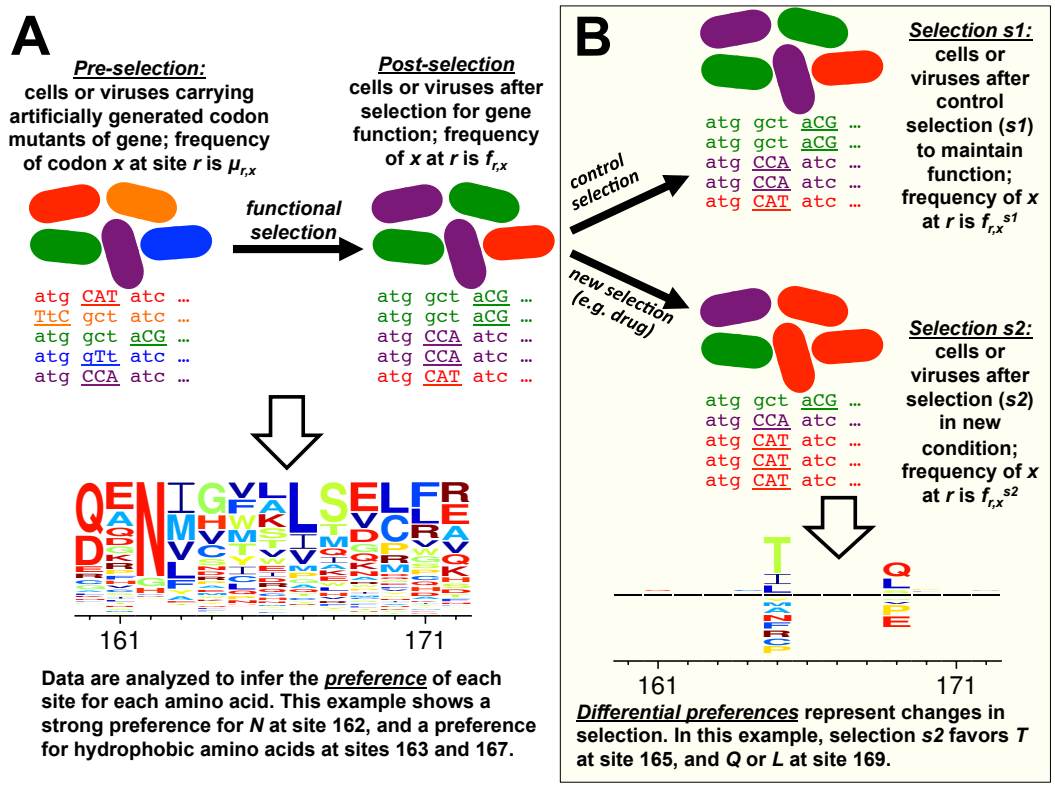


**Figure 1 A deep mutational scanning experiment.** (A) A gene is mutagenized to create a library that contains all single codon mutations. The mutant library is introduced into cells or viruses and subjected to a functional selection that enriches beneficial mutations and depletes deleterious ones. Deep sequencing is used to count mutations in a sample of the variants present pre- and post-selection. Using `dms_tools`, the data can be analyzed to infer the "preference" of each site for each amino acid; in the visualization, letter heights are proportional to the preference for that amino acid. (B) The experiment can be extended by subjecting the library of functional variants to two different selection pressures, and using deep sequencing to assess which variants are favored in one condition versus the other. Using `dms_tools`, the data can be analyzed to infer the "differential preference" of each site for each amino acid in the alternative selection $s2$ versus the control selection $s1$; in the visualization, letter heights above or below the line are proportional to the differential preference for or against that amino acid.

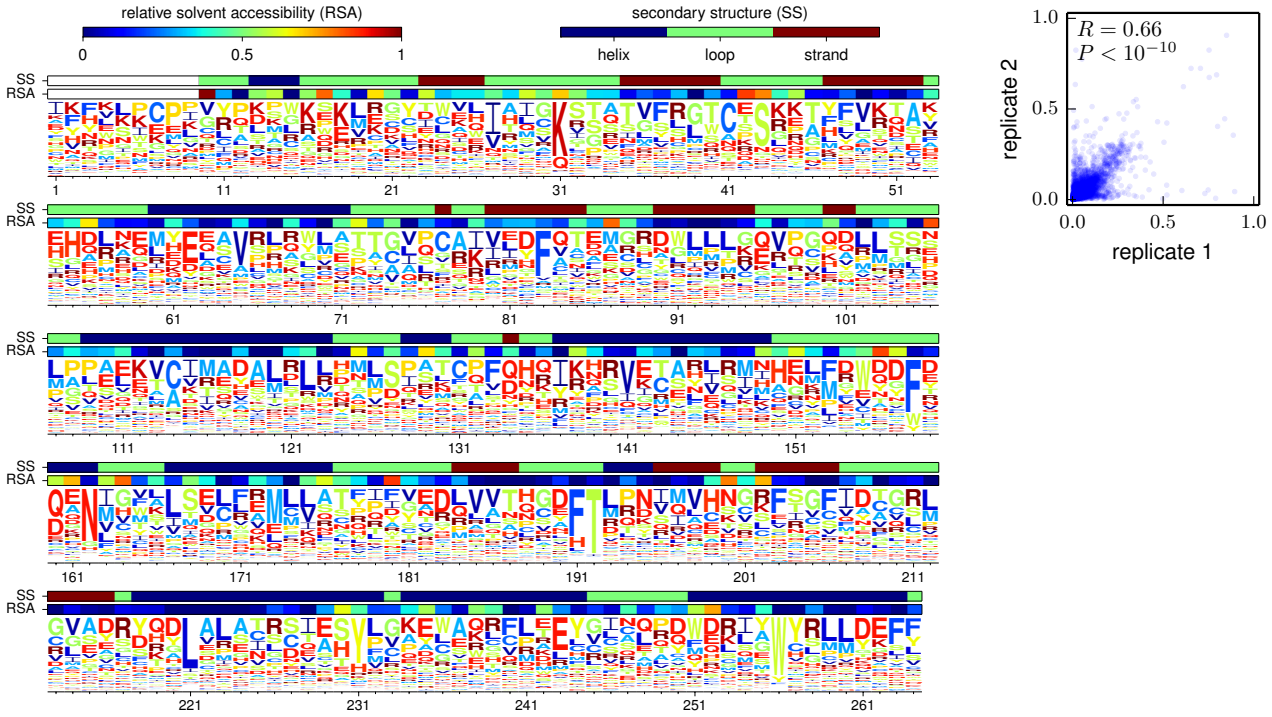


**Figure 2 Site-specific preferences from deep mutational scanning of a Tn5 transposon.** Melnikov *et al* [10] performed deep mutational scanning on a Tn5 transposon using kanamycin selection, and reported the counts of amino-acid mutations for two biological replicates of the experiment. Here I have used `dms_tools` to infer the preferences. (**A**) Visualization of the preferences averaged across the two replicates. (**B**) Correlation between the preferences inferred from each of the two replicates. Given files containing the mutation counts, the plots can be generated as `logoplot.pdf` and `corr.pdf` with the following commands:

```
dms_inferprefs pre_counts_1.txt post_counts_1.txt prefs_1.txt --excludestop --ncpus -1 --chartype aa
dms_inferprefs pre_counts_2.txt post_counts_2.txt prefs_2.txt --excludestop --ncpus -1 --chartype aa
dms_correlate prefs_1.txt prefs_2.txt corr --name1 "replicate 1" --name2 "replicate 2" --corr_on_plot
dms_merge prefs.txt average prefs_1.txt prefs_2.txt
dms_logoplot prefs.txt logoplot.pdf --nperline 53 --overlay1 RSAs.txt RSA "relative solvent accessibility"
    --overlay2 SSs.txt SS "secondary structure"
```

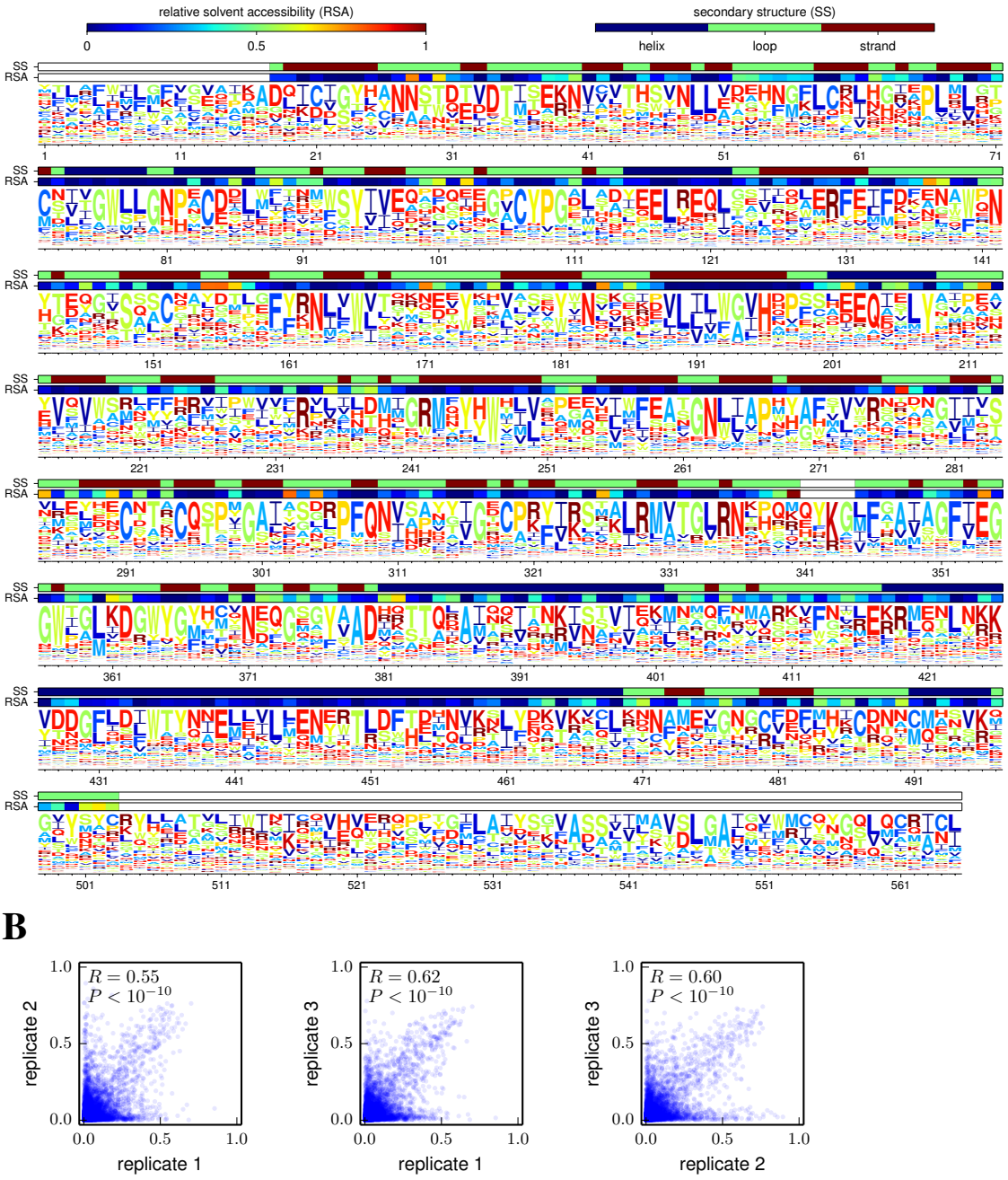


**Figure 3 Site-specific preferences from deep mutational scanning of influenza hemagglutinin.** Thyagarajan and Bloom [11] performed deep mutational scanning on influenza hemagglutinin, and reported the counts of codon mutations for three biological replicates of the experiment. Here I have used `dms_tools` to infer the preferences. (**A**) Visualization of the preferences averaged across the three replicates. (**B**) Correlations between the preferences from each pair of replicates. Given files containing the mutation counts, the plots can be generated as `logoplot.pdf`, `corr_1_2.pdf`, `corr_1_3.pdf`, and `corr_2_3.pdf` with the following commands:

```
dms_inferprefs mutDNA_1.txt mutvirus_1.txt prefs_1.txt --errpre DNA_1.txt --errpost virus_1.txt --ncpus -1
dms_inferprefs mutDNA_2.txt mutvirus_2.txt prefs_2.txt --errpre DNA_2.txt --errpost virus_2.txt --ncpus -1
dms_inferprefs mutDNA_3.txt mutvirus_3.txt prefs_3.txt --errpre DNA_3.txt --errpost virus_3.txt --ncpus -1
dms_correlate prefs_1.txt prefs_2.txt corr_1_2 --name1 "replicate 1" --name2 "replicate 2" --corr_on_plot
dms_correlate prefs_1.txt prefs_3.txt corr_1_3 --name1 "replicate 1" --name2 "replicate 3" --corr_on_plot
dms_correlate prefs_2.txt prefs_3.txt corr_2_3 --name1 "replicate 2" --name2 "replicate 3" --corr_on_plot
dms_merge prefs.txt average prefs_1.txt prefs_2.txt prefs_3.txt
dms_logoplot prefs.txt logoplot.pdf --nperline 71 --overlay1 RSAs.txt RSA "relative solvent accessibility"
     --overlay2 SSs.txt SS "secondary structure" --excludestop
```

Note that unlike in Figure 2, no `--chartype` option is specified since the `dms_inferprefs` default is already `codon_to_aa`.

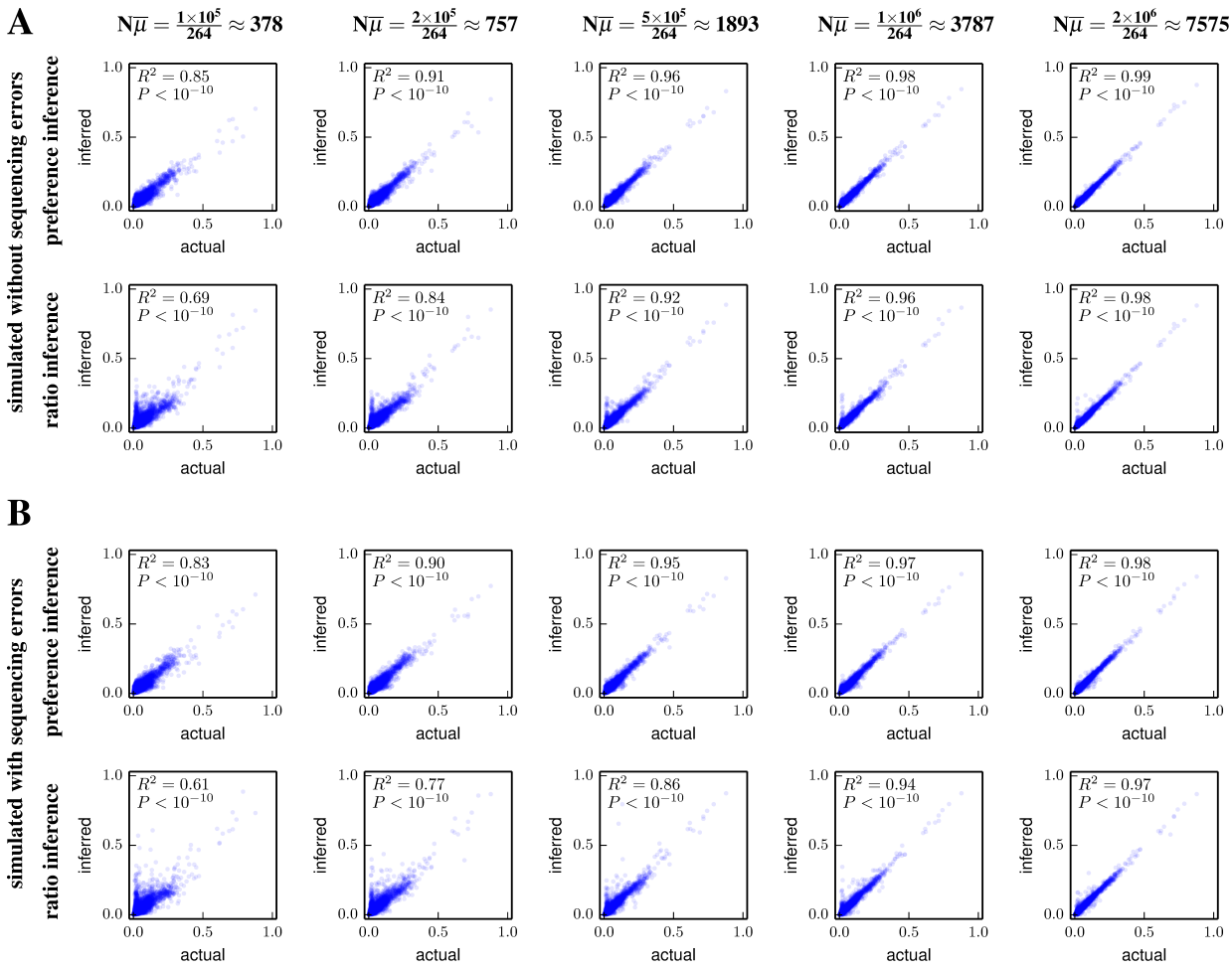


**Figure 4 Accuracy of preference inference on simulated data.** Deep mutational scanning counts were simulated using the preferences in Figure 2A and realistic parameters for mutation and error rates. The simulations were done $(\mathbf{A})$ without or $(\mathbf{B})$ with sequencing errors quantified by control libraries. Plots show the correlation between the actual and inferred preferences as a function of the product of the sequencing depth $N$ and the average per-site mutation rate $\overline{\mu}$; real experiments typically have $N\overline{\mu} \sim 1000$ to 2000 depending on the sequencing depth and gene length. Preferences are inferred using the full algorithm in `dms_tools` (top panels) or by simply calculating ratios of counts (bottom panels) using Equation 4 and its logical extension to include errors, both with a pseudocount of one. The `dms_tools` inferences are more accurate than the simple ratio estimation, with both methods converging to the actual values with increasing $N\overline{\mu}$. Given files with the mutation counts, the plots in this figure can be generated as `prefs_corr.pdf` and `ratio_corr.pdf` with commands such as:

```
dms_inferprefs pre.txt post.txt inferred_prefs.txt --ncpus -1
dms_inferprefs pre.text post.text ratio_prefs.txt --ratio_estimation 1
dms_correlate actual_prefs.txt inferred_prefs.txt prefs_corr --name1 "actual" --name2 "inferred"
     --corr_on_plot --r2
dms_correlate actual_prefs.txt ratio_prefs.txt ratio_corr --name1 "actual" --name2 "inferred" --corr_on_plot
     --r2
```

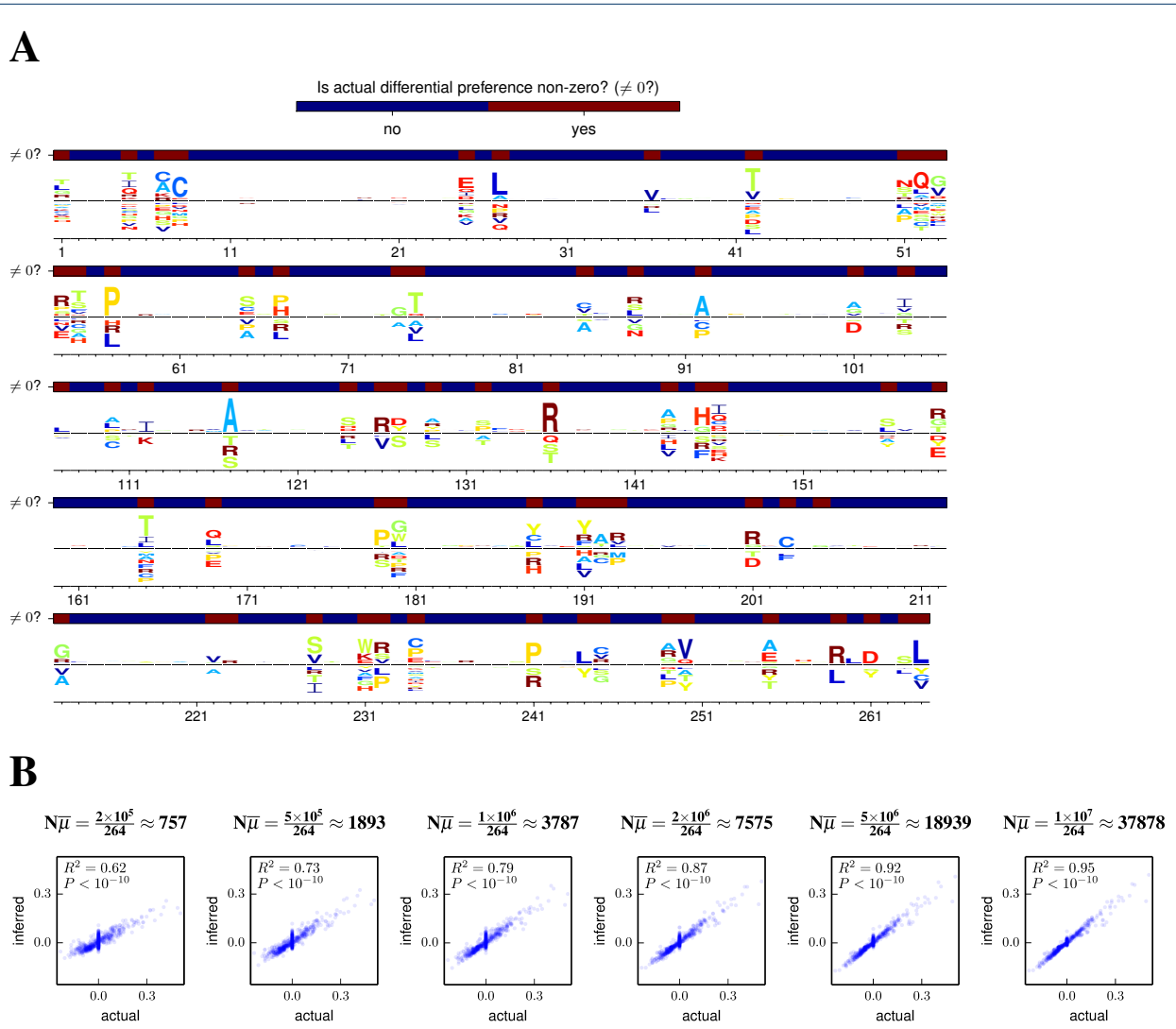


**Figure 5 Inference of differential preferences on simulated data.** To illustrate and test the inference of differential preferences, the experiment in Figure 1B was simulated at the codon level starting with the post-selection library that yielded the preferences in Figure 2. In the simulations, 20% of sites had different preferences between the control and alternative selection. **(A)**, `dms_tools` was used to infer the differential preferences from the data simulated at $N = 10^7$, and the resulting inferences were visualized. The overlay bars indicate which sites had non-zero differential preferences in the simulation. **(B)** The correlations between the inferred and actual differential preferences as a function of $N\overline{\mu}$ show that the inferred values converge to the true ones. Given files with the mutation counts, the plots in this figure can be generated as `logoplot.pdf` and `corr.pdf` with the following commands:

```
dms_inferdiffprefs start.txt s1.txt s2.txt diffprefs.txt --ncpus -1
dms_logoplot diffprefs.txt logoplot.pdf --nperline 53 --overlay1 actually_nonzero.txt "$\ne 0$?" "Is actual
    differential preference non-zero?" --diffprefheight 0.45
dms_correlate actual_diffprefs.txt diffprefs.txt corr --name1 "actual" --name2 "inferred" --corr_on_plot --r2
```

Note that no `--chartype` option is specified because the default for `dms_inferdiffprefs` is already `codon_to_aa`.

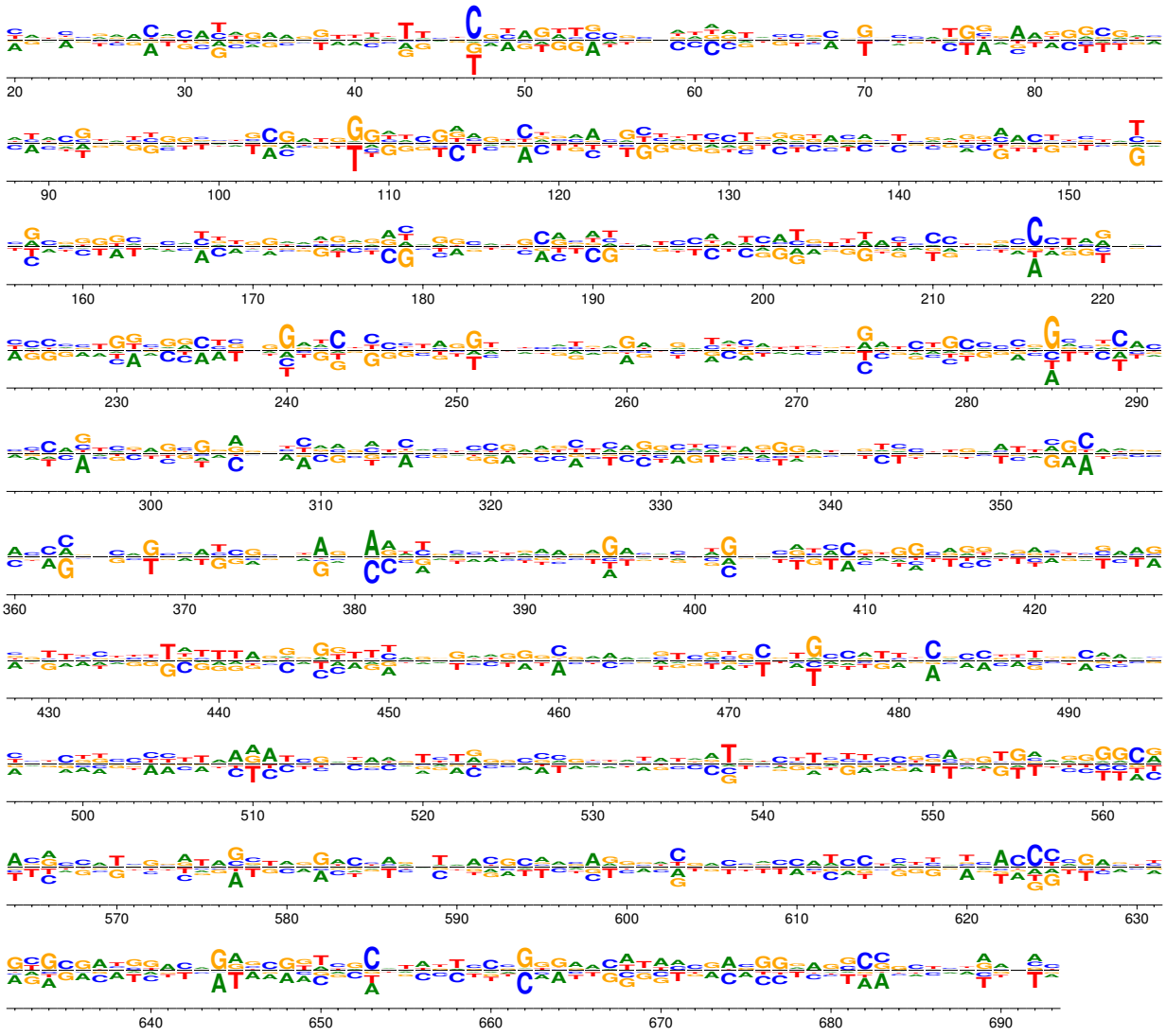


**Figure 6 Differential preferences following selection of influenza NS1 in the presence or absence of interferon.** Wu *et al* [13] generated libraries of influenza viruses carrying nucleotide mutations in the NS segment. They passaged these viruses in the presence or absence of interferon pre-treatment. Here, `dms_tools` was used to analyze and visualize the data to identify sites where different nucleotides are preferred in the presence versus the absence of interferon. Because the mutations were made at the nucleotide level, the data must also be analyzed at that level (unlike in Figures 2, 3, and 5, where codon mutagenesis means that the data can be analyzed at the amino-acid level). The plot can be generated as `logoplot.pdf` with the following commands:

```
dms_inferdiffprefs input.txt control.txt interferon.txt diffprefs.txt --ncpus -1 --chartype DNA
dms_logoplot diffprefs.txt logoplot.pdf --nperline 68 --diffprefheight 0.4
```