ARTICLES
https://doi.org/10.1038/s41587-021-01146-5

nature biotechnology

Learning protein fitness models from evolutionary and assay-labeled data

Chloe Hsu[1]✉, Hunter Nisonoff[2], Clara Fannjiang[1] and Jennifer Listgarten[1,2]✉

Machine learning-based models of protein fitness typically learn from either unlabeled, evolutionarily related sequences or variant sequences with experimentally measured labels. For regimes where only limited experimental data are available, recent work has suggested methods for combining both sources of information. Toward that goal, we propose a simple combination approach that is competitive with, and on average outperforms more sophisticated methods. Our approach uses ridge regression on site-specific amino acid features combined with one probability density feature from modeling the evolutionary data. Within this approach, we find that a variational autoencoder-based probability density model showed the best overall performance, although any evolutionary density model can be used. Moreover, our analysis highlights the importance of systematic evaluations and sufficient baselines.

Naturally occurring proteins serve many crucial functions in maintaining life, but have also been coopted for human endeavors such as gene editing[1,2]; lighting up specific parts of cells[3]; therapeutic drugs[4]; and herbicide-resistant crops[5]. Furthermore, in many cases we reengineer proteins to better serve our needs. For example, we might enhance the original function, such as when we increase enzyme activity[6] or make green fluorescent proteins (GFPs) brighter[7]. Alternatively, we might modify the original function to a related but different one, such as when we change an antibody to bind to a new target[8]. The two most common approaches to protein engineering are laboratory-based directed evolution[9] and computational, physics-based rational design[10,11]. Machine learning-based models that predict protein fitness from sequence can allow in silico screening to complement these approaches. Here, fitness is a broad term that refers to any protein property ranging from stability to enzyme activity and ligand binding. Depending on the size of the design space and the computational cost of inference of the fitness model, the model can be used either to systematically screen all protein variants in the design space[12–15], or in combination with an optimization algorithm to search through vast design spaces[16–19]. When in-depth knowledge of protein structure can be coupled to the phenotype of interest, physics-based methods such as FoldX[20], PoPMuSiC[21] and IMutant[22] can be used to model protein fitness. Applicable more generally, in the absence of such knowledge, machine learning models can learn from unlabeled, evolutionarily related sequences, or from variant sequences with experimentally measured labels, to predict protein fitness. Herein, we focus exclusively on machine learning methods for protein fitness prediction.

There have been two main machine learning-based strategies for estimating protein fitness models. The first strategy uses implicit fitness constraints present in naturally occurring protein sequences, so-called evolutionary data. Such evolutionary methods start from one query protein with the desired property (for example, a particular GFP that fluoresces at some wavelength), and search through databases of naturally occurring proteins to find a set of related proteins—typically by sequence homology—that are assumed to be enriched for the same property as the query sequence. Then, a probability density model of this set of protein sequences is estimated; finally, sequence density evaluations are used to predict the relative fitness of protein variants of interest[23,24]. Early methods for the related problem of pathogenicity prediction, such as SIFT[25] and Polyphen-2 (ref. [26]), used position-specific substitution matrices or Hidden Markov models[27]. In the context of protein fitness prediction, Potts models[13,23,28–30], which can model pairwise interactions in the sequence, have been shown to outperform SIFT and PolyPhen-2 at fitness prediction[23]. Deep learning models such as variational autoencoders (VAEs) can capture higher-order interactions, and may provide more accurate predictions still[24]. Most evolutionary methods assume that the evolutionarily related proteins have been aligned into a multiple sequence alignment (MSA)[13,23–30]. Although the set of evolutionarily related proteins does not typically have associated measurements for the property of interest, the homology search itself is assumed to implicitly provide a weak, positive labeling of all proteins in the set. Thus, we refer to these evolutionary methods as weak-positive only learning, rather than unsupervised learning. The number of sequences used for training in such cases can range from hundreds to hundreds of thousands when standard procedures are used to curate the set[23].

In the second main machine learning strategy for learning protein fitness models, supervised regression models are trained using variant sequences coupled with fitness labels measured in the laboratory. Depending on the protein and the property of interest, a supervised data set may comprise hundreds to hundreds of thousands of examples[31,32]. The laboratory assay may be a relatively direct measurement of the fitness of interest[33], or a crude proxy[34]. Additionally, the set of variant sequences may be restricted to one or two mutations away from a query sequence[31,32,34], or be more heterogeneously distributed[33], with the former setting being more common. The labeled variant sequences may vary at only a few sequence positions (for example, four positions[35]), or vary more broadly[32]. By their limited construction, these data sets are not typically sufficiently rich to comprehensively characterize protein fitness landscapes, but are a good starting point. As time progresses, more comprehensive data sets, revealing increasingly more nuances of fitness landscapes, will expand our understanding. The simplest supervised approach is a linear regression model with one-hot encoded site-specific amino

[1]Department of Electrical Engineering and Computer Science, University of California, Berkeley, USA. [2]Center for Computational Biology, University of California, Berkeley, USA. ✉e-mail: chloehsu@berkeley.edu; jennl@berkeley.edu

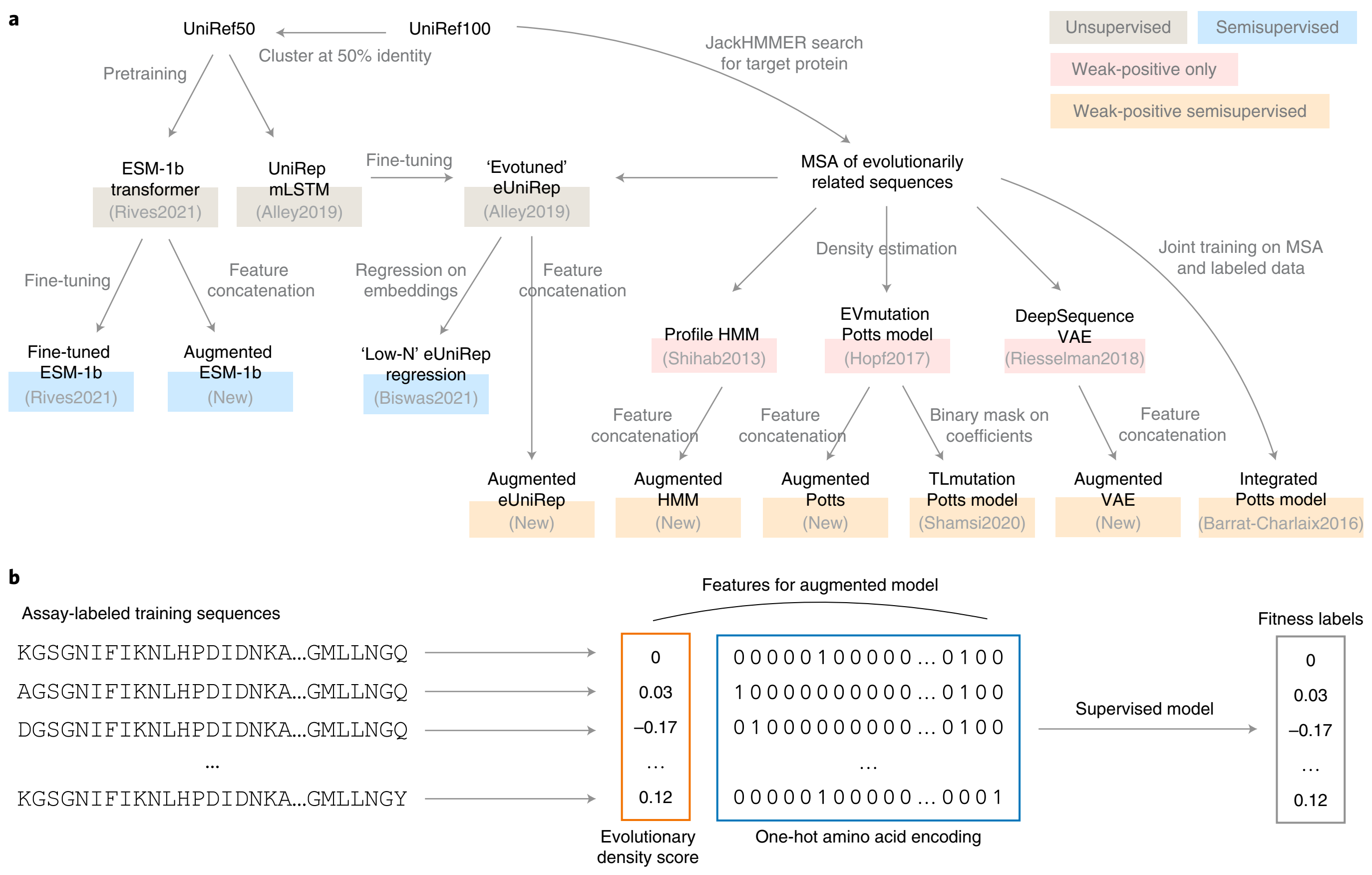

a
UniRef50
UniRef100
Cluster at 50% identity
Pretraining
JackHMMER search for target protein
Unsupervised
Semisupervised
Weak-positive only
Weak-positive semisupervised
ESM-1b transformer (Rives2021)
UniRep mLSTM (Alley2019)
Fine-tuning
'Evotuned' eUniRep (Alley2019)
MSA of evolutionarily related sequences
Fine-tuning
Feature concatenation
Regression on embeddings
Feature concatenation
Density estimation
Joint training on MSA and labeled data
Fine-tuned ESM-1b (Rives2021)
Augmented ESM-1b (New)
'Low-N' eUniRep regression (Biswas2021)
Profile HMM (Shihab2013)
EVmutation Potts model (Hopf2017)
DeepSequence VAE (Riesselman2018)
Feature concatenation
Feature concatenation
Binary mask on coefficients
Feature concatenation
Augmented eUniRep (New)
Augmented HMM (New)
Augmented Potts (New)
TLmutation Potts model (Shamsi2020)
Augmented VAE (New)
Integrated Potts model (Barrat-Charlaix2016)
b
Assay-labeled training sequences
KGSGNIFIKNLHPDIDNKA...GMLLNGQ
AGSGNIFIKNLHPDIDNKA...GMLLNGQ
DGSGNIFIKNLHPDIDNKA...GMLLNGQ
...
KGSGNIFIKNLHPDIDNKA...GMLLNGY
Features for augmented model
0
0.03
−0.17
...
0.12
0 0 0 0 0 1 0 0 0 0 0 ... 0 1 0 0
1 0 0 0 0 0 0 0 0 0 0 ... 0 1 0 0
0 1 0 0 0 0 0 0 0 0 0 ... 0 1 0 0
...
0 0 0 0 0 1 0 0 0 0 0 ... 0 0 0 1
Evolutionary density score
One-hot amino acid encoding
Supervised model
Fitness labels
0
0.03
−0.17
...
0.12

Fig. 1 | Machine learning methods for protein fitness prediction. a, Overview of methods considered, divided into three machine learning strategies: (1) weak-positive only learning methods that rely on probability density modeling of evolutionarily related sequences (profile HMM[27], EVmutation[23] and DeepSequence[24]); (2) semi-supervised learning methods that rely on unlabeled UniRef50 protein sequences, supervised data and optionally evolutionarily related sequences (fine-tuned ESM-1b[40] and eUniRep regression[42]); (3) weak-positive semi-supervised learning methods that rely on evolutionarily related sequences and supervised data (the integrated Potts model[44], TLmutation[43] and augmented models). **b**, Depiction of the augmented approach to predicting fitness wherein evolutionary and assay-labeled data are combined, and it is assumed that the probability density model has already been fully trained. In particular, the augmented approach uses linear regression with two kinds of features: first, evolutionary density evaluations (that is, the inferred sequence log-likelihoods or approximations thereof from a sequence density model trained on evolutionarily related sequences), and second, one-hot encoded site-specific amino acids, to perform supervised training on assay-labeled data.

acid features; this approach can be extended to include pairwise features, and a nonlinear transformation[36]. Richer supervised models used for protein fitness prediction include convolutional neural networks (CNNs)[15,37], long short-term memory networks (LSTMs)[38,39] and transformers[38,40,41]. In some cases, such as that of LSTMs and transformers, a large and broad set of unlabeled protein sequences is also used during training[38–42]; these data are thought to help find effective representations for the supervised learning component. Note that some of these classes of models, such as LSTMs and transformers, can be used either as probability density models or as supervised models.

Recently, several proposals have been made to combine these two machine learning strategies: using weak-positive only learning on evolutionarily related sequences together with supervised learning on assay-labeled variant sequences[42–44]. We refer to this setting as weak-positive semi-supervised learning. For many scenarios of practical importance, property labels can only be assayed for hundreds of protein sequences. Particularly in such a regime (but not limited to this), it is important to combine both sources of data. Barrat-Charlaix et al.[44] introduce an elegant approach to do so, the integrated Potts model, which assumes the Potts model energy function is identical to that in a supervised regression model; these are then trained jointly. Shamsi et al.[43] instead 'transfer' information from a learned evolutionary model to a supervised version of the model by learning a binary masking of parameters. Motivated by the field of natural language processing (NLP)[45,46], Biswas et al.[42] first train an unsupervised multiplicative LSTM (mLSTM) on a large database of naturally occurring protein sequences, then fine-tune it with evolutionary data to produce a fixed-length latent representation called eUniRep. This fixed representation is then used as the features in a regularized linear regression on the assay-labeled data.

Next we report on a systematic assessment of these 'combined' methods as well as 'pure' methods contained in them, such as probability density-only models, or supervised-only models. We also introduce a simple baseline combined approach that turns out to be competitive with and often outperform more sophisticated methods. In one instantiation, our approach achieves maximal performance on 15 of the 19 data sets used in our assessment—more than any other method—and is typically among the top competitors when it does not. Our simple approach can make use of any evolutionary probability density model, and adds little computational burden to it.

Results

Published machine learning methods assessed. We assessed a total of 13 published machine learning methods for protein fitness

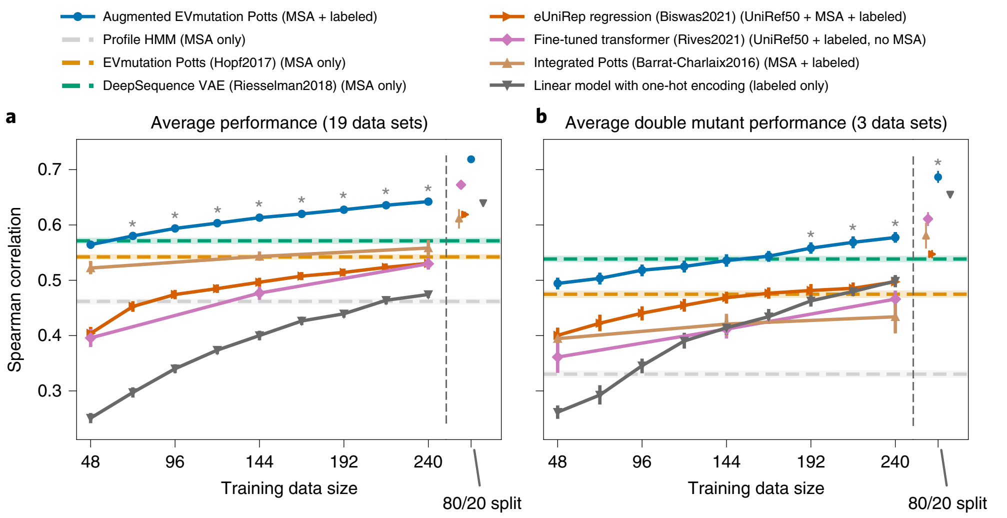


Fig. 2 | Performance of existing methods and the augmented Potts model. a, Average performance across all 19 data sets, as measured by Spearman correlation. The horizontal axis shows the number of supervised training examples used. Error bars are centered at the mean and indicate bootstrapped 95% confidence intervals estimated from 20 random splits of training and test data. Note some bars are so small as to be almost invisible. Asterisks (*) indicate that $P < 0.01$ among all two-sided Mann–Whitney U-tests that the augmented Potts model has different performance from each other method, at a given sample size. In particular, the largest such P values for each training set size were, respectively, $P = 9.0 \times 10^{-3}$, 3.9×10^{-7}, 9.2×10^{-8}, 7.7×10^{-4}, 6.8×10^{-8}, 6.8×10^{-8}, 6.8×10^{-8} and 7.7×10^{-4} for training data sizes $72, 96, 120, \cdots, 240$ and $P = 7.7 \times 10^{-4}$ for the 80/20 split. **b**, Average performance across all three data sets containing double mutant sequences (sequences that are two mutations away from the wild-type), and restricted to testing on only double mutants. Asterisks (*) indicate same as in **a**, with largest P values here, respectively, $P = 3.4 \times 10^{-4}$, 2.7×10^{-6} and 7.7×10^{-4} for training data sizes 192, 216, 240 and $P = 7.7 \times 10^{-4}$ for the 80/20 split. See Extended Data Fig. 1 for NDCG and Extended Data Figs. 2 and 3 and Supplementary Figs. 1 and 2 for performance on individual data sets.

prediction that use evolutionary data, assay-labeled data or both. Some[39,40,42] additionally make use of a large 'universal' set of unlabeled protein sequences, namely the UniRef50 database[47]. In addition to the three combined methods summarized earlier (eUniRep regression, the integrated Potts model and TLmutation), we also included methods that use only one or the other type of data so as to glean their relative importance across a range of settings. In particular, we included three representative purely evolutionary methods: a profile hidden Markov model (HMM)[27], a Potts model (EVmutation[23]) and a VAE (DeepSequence[24]). We also included two supervised methods that do not use evolutionary data: a linear regression model with one-hot encoded site-specific amino acid features that uses only assay-labeled data, and the Evolutionary Scale Modeling (ESM)-1b transformer[40] model that was pre-trained on UniRef50 (unsupervised), and then fine-tuned with the assay-labeled data (supervised). While several transformers have been developed and published for protein sequences[38,48], we chose the ESM-1b model because at the time of our assessment, it was shown to have had superior performance on a number of prediction tasks when compared to other transformers[40]. Figure 1a shows a graphical overview of all of these methods.

To enable all of these methods to run on our entire suite of benchmarking data sets we made minor modifications to some of the training procedures so that they could run in a reasonable amount of time. For each modification used, we show that the resulting performance closely matches the performance of the original procedure on data sets used in the original papers (Methods).

A simple baseline approach. We developed a simple approach that makes use of both evolutionary and assay-labeled data, and that turns out to be competitive with much more expensive and complicated approaches. For any already-trained evolutionary probability density model, we use ridge regression on one-hot encoded site-specific amino acid features augmented with one other feature—the sequence density evaluation (Fig. 1b). We refer to this approach as 'augmenting' a probability density model. For example, when the evolutionary probability density model is a Potts model, our baseline yields an augmented Potts model. Beyond augmented Potts models, we also compare to augmented VAEs and so forth. The augmented Potts model in particular can be interpreted in a Bayesian light as updating a prior from the evolutionary data, with assay-labeled data (Methods). Because we use only site-specific amino acid features in the regression augmentation, the Bayesian update is focused on the site-specific parameters (as opposed to the pairwise coupling parameters). If we had included also pairwise amino acid features, then we would be more directly updating all parameters in the Potts model prior, which would yield an approach that is conceptually similar to the integrated Potts model[44], albeit with a different optimization objective. Given an already-trained probability density model, the augmentation entails only training a linear regression model, thus incurring little computational burden. Although incorporating richer evolutionary density-derived features, such as pairwise potentials, into the augmented regression could yield further improvements, we sought a baseline that did not require feature selection or specialized regularization[49]. We did, however, investigate extending our augmented regression features to include density evaluations from multiple models, such as from both a Potts model and a VAE, or from both a Potts model and a Transformer, and so forth. However, doing so resulted in only marginal gains on top of the augmented DeepSequence VAE model (data not shown). We speculate that this may be because the different probability density models were not providing sufficiently independent information.

Assay-labeled and evolutionary data sets. We assessed all methods on 19 labeled mutagenesis data sets, each comprising hundreds to tens of thousands of mutant sequences. Most related work[24,40,43,44] evaluates on a subset of, or all of the mutation effect data sets in EVmutation[23]. We obtained our 19 data sets by including all EVmutation[23] protein data sets with at least 100 entries (to have

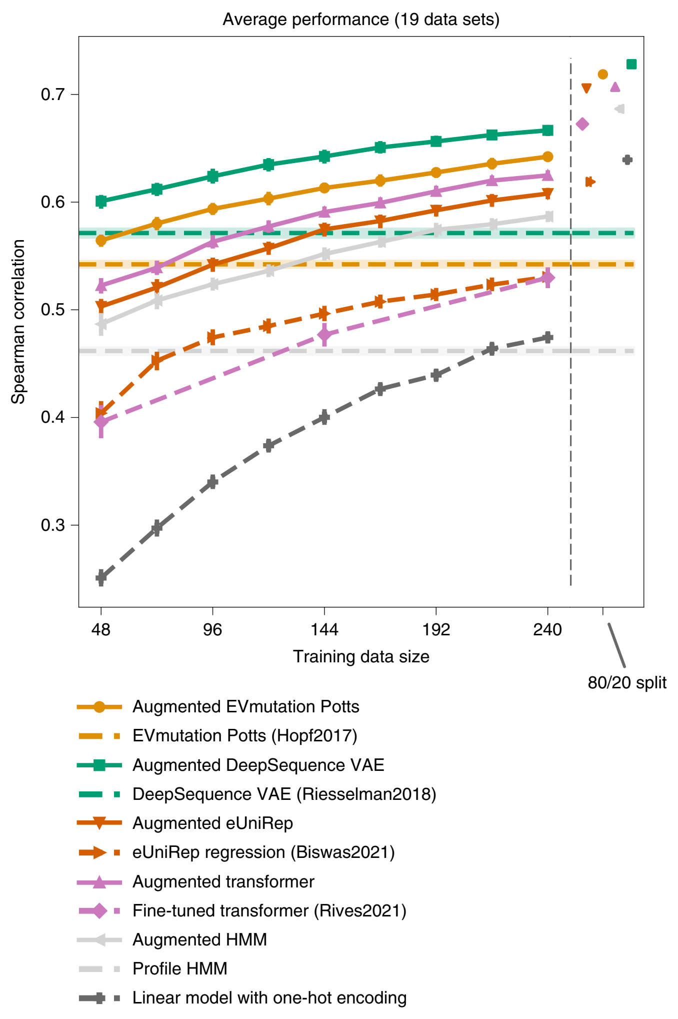


Fig. 3 | Augmented approach using different probability density models. Using the same evaluation setup as in Fig. 2a, methods are compared with their augmented counterpart, using matching colors on each pair. Flat, horizontal lines represent evolutionary density models that do not have access to assay-labeled data. Dashed lines indicate existing methods. Error bars are centered at the mean and indicate bootstrapped 95% confidence interval from 20 random data splits. See Extended Data Fig. 4 for performance measured by NDCG.

sufficient data to glean insights from), and also included a GFP fluorescence data set[33] as in Biswas et al.[42] (however, their exact data set was not available at the time of our assessment). Other data sets that might have been relevant had insufficient evolutionary data in the MSAs[32,35]. In each data set used, mutations were spread across either a domain or the whole protein. Although most (16 out of the 19) labeled data sets consisted only of sequences that were one mutation away from a fixed wild-type protein ('single mutants'), we also include detailed case studies of the three data sets[33,34,50] that contained sequences more than one mutation away from a wild-type protein ('higher-order mutants'). Supplementary Table 1 provides a detailed overview of these data sets. Each labeled data set was paired with an evolutionary data set found by searching through the UniRef100 database[47] for sequences similar to a single, relevant wild-type protein sequence, using Jackhmmer[23,24,42,51]. For ease of comparison with other papers[23,24,43], when available, we directly used MSAs curated in this way and provided by EVmutation[23]. Otherwise, we used Jackhmmer with parameters set as in EVmutation.

Experimental overview. For each data set we always used 20% of the available labeled data for test sets (which by definition are not used to train or perform hyper-parameter selection). Given a fixed test set, we systematically varied the supervised training data set size by randomly sampling each of $48, 72, \cdots, 216, 240$ training examples from the nontest sequences (each training data set size was increased in intervals of 24). In addition to these fixed training data set sizes, for more direct comparison with TLmutation[43] and ESM-1b[40], we also used all the remaining 80% of the available data not used for testing, which we refer to as an 80/20 train/test split. When computationally feasible, we used fivefold cross-validation on the training data set for hyper-parameter selection, and otherwise held out 20% of the training data as validation data. For each training data set size, including the 80/20 split, we averaged performance over 20 random seeds that randomized the data partitions.

We used two measures of predictive model performance: (1) a Spearman rank correlation between the true and predicted fitness values[13,23,24,40,44], and (2) a ranking measure from the information retrieval community called normalized discounted cumulative gain (NDCG)[15,52], which gives high values when the top predicted results are enriched for truly high fitness proteins—that is, when the model is predicting the ranking of the most fit sequences well. In contrast to the Spearman correlation, where errors at the top of the true ranked list carry the same weight as mismatches at the bottom, NDCG puts a higher weight on ranking the top of the list correctly and can be seen as a smoothed version of top *k* average fitness without a fixed *k*. No one metric is necessarily the most relevant, as this will depend on the task at hand. For example, in using the model to propose a list of candidates for protein engineering, we may be interested only in the best performance among the top *k* proteins which will be assessed in the laboratory. In contrast, if using the model to understand the fitness landscape of a protein, we may be interested in more general accuracy. By using both Spearman correlation and NDCG, we hope to be broadly informative across a wide variety of settings. For each Spearman correlation result in the main text, the corresponding result with NDCG is found in the Extended Data and the Supplementary Information. Generally, the two metrics yielded similar qualitative conclusions.

Results for TLmutation[43] are omitted from the main figures for several reasons. First, TLmutation is conceptually similar to our augmented Potts model, but is restricted to only allow for zeroing out of the evolutionary Potts model parameters by way of supervised learning. Second, TLmutation was much more computationally expensive than the augmented Potts model. And finally, in a set of exploratory experiments, TLmutation performed worse than our augmented Potts model (Supplementary Fig. 3).

Performance of existing methods and the augmented Potts model. When comparing existing methods to each other and to our augmented Potts model, averaged over all data sets, the augmented Potts model typically outperformed existing methods, including more sophisticated methods such as eUniRep regression[42] (Fig. 2). The sole exception was that the purely evolutionarily based VAE outperformed the augmented Potts model on the double-mutant data in the low data regime. However, later we show that the augmented VAE outperforms the VAE, suggesting that either the Potts model is not sufficiently expressive in some cases, or that the VAE has a more suitable inductive bias, or both. As expected, with increasing labeled training data, the supervised methods increase in accuracy. When breaking down the average performance into individual data sets, the augmented Potts model is competitive on most data sets (Extended Data Figs. 2 and 3 and Supplementary Figs. 1 and 2). The extremely simple, supervised-only, linear regression on one-hot encoded amino acid features, is also among the top performers in the 80/20 split setting. Due to limited availability of data sets that include double mutants, the reported double-mutant performance

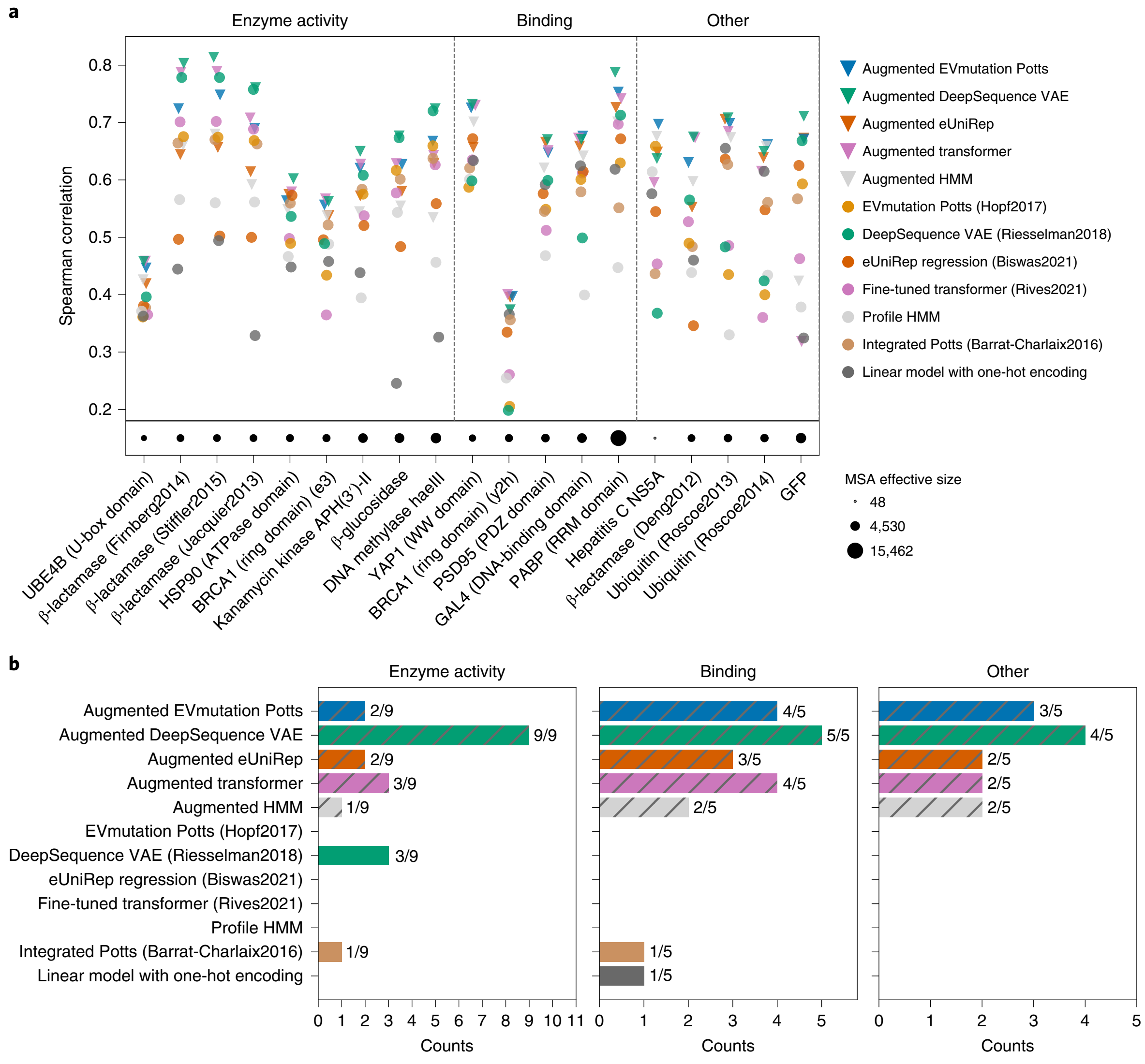


Fig. 4 | Performance on individual data sets. a, Other than the EVmutation Potts model, the DeepSequence VAE and the Profile HMM, none of which use supervised data, all other methods here used 240 labeled training sequences. Each colored dot is the average Spearman correlation from 20 random train/test splits. Random horizontal jitter was added for display purposes. The bottom row of black dots indicates the effective MSA size, determined by accounting for sequence similarity with sample reweighting at 80% identity cutoff. **b**, Summary of how often each modeling strategy had maximal Spearman correlation. Such modeling strategies were determined by first identifying the top-performing strategy for any given scenario, and then also identifying any other strategy that came within the 95% confidence interval of the top performer. See Extended Data Fig. 5 for analogous plots for NDCG and Extended Data Fig. 6 for evaluation on varying numbers of training examples.

is averaged over only the three relevant data sets and hence is heavily influenced by their characteristics. We refer the reader to Fig. 5 for the double-mutant performance breakdown.

Augmented models as a general, simple and effective strategy. Next we also augmented the profile HMM, the DeepSequence VAE, the eUniRep mLSTM and the ESM-1b transformer, comparing them to each other and to the augmented Potts model. Note that the transformer is the only method that does not make use of evolutionary data, but was pretrained on UniRef50. Overall, we found that the augmented VAE achieved the highest average performance among the augmented models, with the augmented Potts model a close second (Fig. 3). No matter which density model we augmented, the augmented model always outperformed the corresponding nonaugmented existing method, regardless of the training data set size.

Among all approaches evaluated here, and earlier, the augmented DeepSequence VAE tended to be the best performer on most data sets (Fig. 4b). It worked particularly well relative to others when predicting enzyme activity, achieving maximal performance in all nine of those protein data sets (Fig. 4b). For binding affinity, the different augmented models performed comparably (Fig. 4b).

Generally, all models that make use of evolutionary data achieved higher Spearman correlation with larger effective MSA size—the weighted number of sequences in the MSA after accounting for sequence similarity with sample reweighting. Meanwhile, the relative ranking of models appears to be unrelated to the effective MSA size (Fig. 4a). Although each data set had a fixed MSA, we chose the data set with the largest effective MSA size (poly(A)-binding protein) to systematically reduce the size to observe the consequences of decreasing size. In this case study, we considered the augmented Potts model and linear regression (Supplementary Fig. 4). The augmented

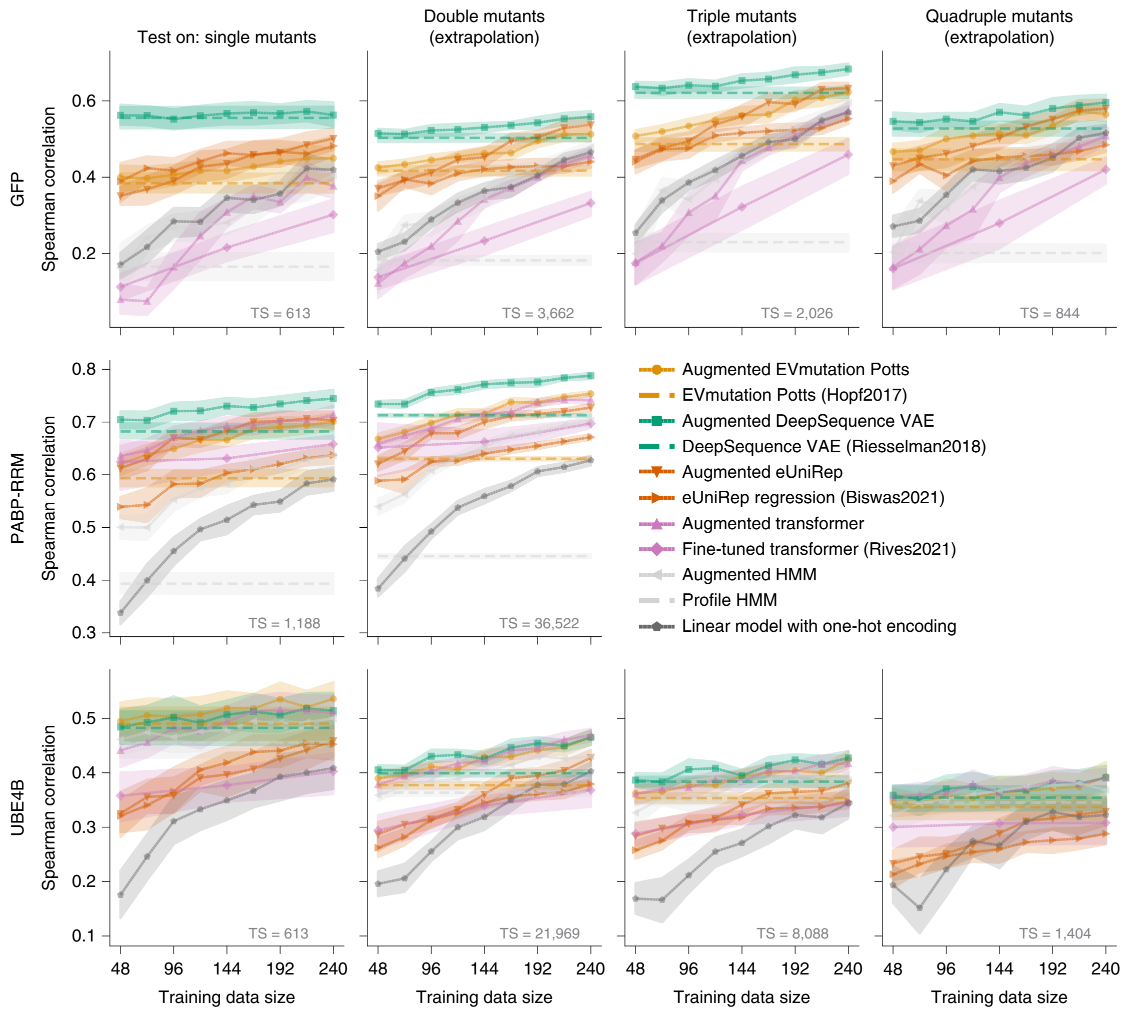


Fig. 5 | Extrapolative performance from single mutants to higher-order mutants. Each column shows the performance when training on randomly sampled single mutants and then separately testing on single, double or triple mutants, none of which were in the training data. The total size (TS) value indicates the total number of mutants of a particular order in all of the data. For example, 'TS = 613' for single mutants means there were 613 total single mutants in the data set that we sampled from. Error bars are centered at the mean and indicate bootstrapped 95% confidence interval from 20 random data splits. Supplementary Fig. 5 shows quantitatively similar results when measuring performance by NDCG, while Extended Data Fig. 7 and Supplementary Fig. 6 show results for extrapolation when training on more than just single mutants (and therefore for larger training data set sizes).

Potts model performance degraded gradually when subsampling the MSA, while always outperforming linear regression, even with only 32 evolutionary sequences. In order to examine the effect of MSA size, we held the protein fixed and downsampled the MSA sequences. This is of course different than comparing different proteins with different MSA sizes. However, because different proteins have different fitness landscapes, there is no way to directly compare in such a manner.

Extrapolation from single to higher-order mutants. Because 16 of the 19 supervised data sets consisted of only single mutants of a wild-type, we next more closely examined the three data sets with higher-order mutants, namely, the GFP[33], Poly(A)-binding protein (PABP) RRM domain[34] and ubiquitination factor E4B (UBE4B) U-box domain[50] data sets. Here, we trained the models using only single-mutant labeled data, and then evaluated on single, double, triple and quadruple mutants separately (Fig. 5 and Supplementary Fig. 5). We also tried training on both the single- and double-mutant data, and the results were qualitatively similar to using just the single-mutant data for training, albeit with the relative difference in performance between any two models generally decreasing with more data (Extended Data Fig. 7 and Supplementary Fig. 6). The extrapolative performance should depend on how much epistasis contributes to the fitness landscape, and also on the ability of a model to capture such epistasis. The poor extrapolative performance on UBE4B may also stem from its evolutionary data not providing as much relevant information to the property of interest (that is, the assayed property).

In data sets where the assay-labeled variants comprise a variety of edit distances from the wild-type, we found that simply using the edit distance itself is predictive of fitness relative to the machine learning models evaluated. For example, for GFP fluorescence, edit distance as a predictive model achieves a Spearman correlation of 0.45 (Fig. 6), presumably because most mutations are deleterious and roughly additively so. This result indicates that correlation measures can be driven by simple features, even if captured by complex models. In contrast to GFP, mutation count is not as strongly predictive for the UBE4B U-box domain data (Extended Data Fig. 8), presumably because the mutational effects are more heterogeneous in the sense that they comprise more diverse (both deleterious and beneficial), or nonadditive effects, or both. Note that despite the bimodality of the experimentally measured fitness values, the predicted fitness values were unimodal from all five methods.

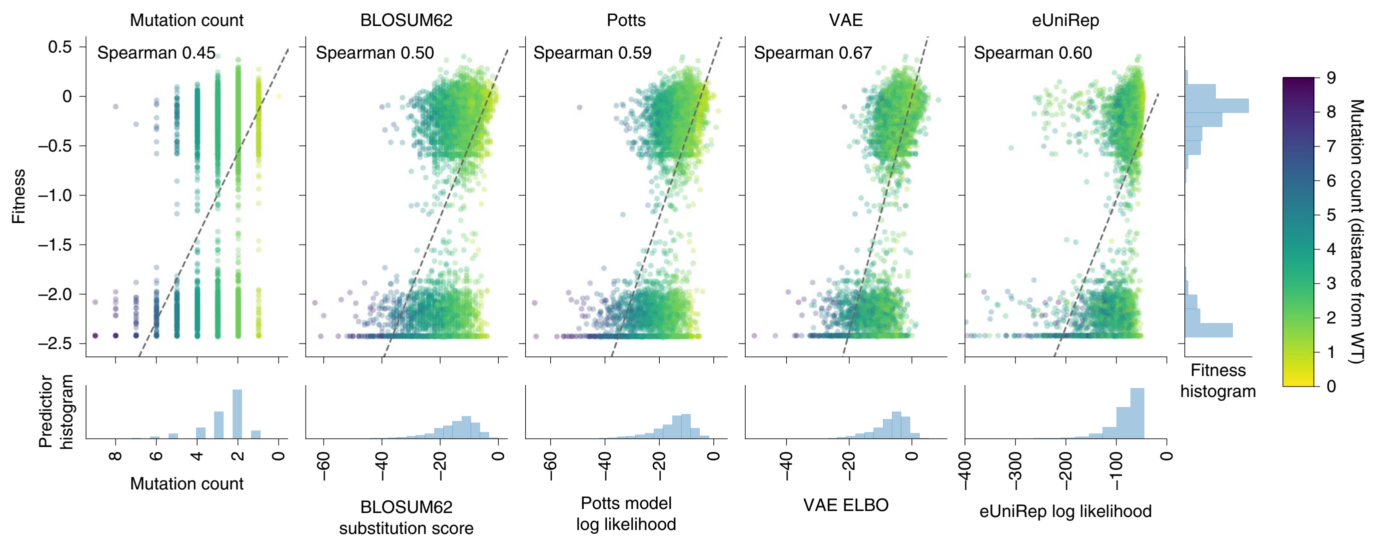


Fig. 6 | Edit distance from wild-type sequence as a predictive model. On the GFP fluorescence data set, we compared the performance of nonaugmented evolutionary density models to two predictive models that use only the edit distance of a sequence to the wild-type. In one version, the edit distance is defined as the number of mutations away from the wild-type. In the other version, we used BLOSUM62 to compute the distance from wild-type, which thus accounts not only for the number of mutations, but also the type of mutation. Each dot represents a GFP sequence, with darker colors indicating larger distances from the wild-type. See Extended Data Fig. 8 for the comparison on the UBE4B U-box domain data set.

Adding in structure-based information. The augmentation approach can easily incorporate structure-based features or predictions from existing methods as additional regression inputs. Recent studies[12,15,53–55] have shown that protein structures, Rosetta energy terms and molecular dynamics models can provide valuable information for fitness prediction. A comprehensive study of how incorporating such features may improve fitness prediction herein is beyond the scope of the present work. However, to get an initial sense of where such future work may lead, we added FoldX-derived stability features to our models on the three data sets with higher-order mutants (Extended Data Fig. 9 and Supplementary Fig. 7). Adding the structure-based FoldX-derived features improved performance on GFP relative to the augmented VAE, but such improvements were not observed on PABP-RRM or UBE4B. These initial results indicate that whether structure-based features provide complementary information to the evolutionary and assay-labeled data may depend on the property of interest and on the quality of available structures.

Discussion

We compared machine learning-based methods that make use of both evolutionary and assay-labeled data for protein fitness prediction. In addition to comparing published methods, we introduced a simple baseline approach, wherein evolutionary density models are augmented with supervised data in a linear regression model on site-specific amino acid features. We instantiated this approach using density models in the form of a profile HMM, a Potts model, a VAE, an LSTM and a transformer. Across a wide range of settings, the augmented Potts model was competitive with higher capacity and more computationally expensive approaches, such as the deep learning-based transformer and LSTMs. Even in the relatively larger data regimes—which were still small—the extremely simple linear regression using site-specific one-hot encoded amino acid features performed well.

One may initially be puzzled as to how linear models with only site-specific features can generalize at all to mutations not seen at train time, let alone surpass nonlinear models in this task. However, as detailed in the Methods, such generalization can emerge from a particular form of l_2-regularized linear regression with one-hot encoded features. In effect, through the regularization, the model learns about the importance of each position, even though each amino acid at each position has its own parameter. Thus, if the effects of different mutations at the same position are in the same direction, the regularized linear models can do a reasonable job of generalizing in such a manner. Although higher-capacity deep learning-based approaches can in principle also pick up on this predictive power, they may be struggling to do so effectively, possibly owing to more degrees of freedom combined with a less suitable inductive bias. Having said that, deep-learning models are not well understood, with even the classical bias-variance tradeoff now in question[56].

When comparing different augmented models, the augmented VAE was most effective at ranking mutational effects overall. Even though the augmented ESM-1b transformer does not use evolutionary data (but does use a large, 'global' set of proteins for unsupervised pretraining), the augmented ESM-1b transformer was competitive with methods that do have access to evolutionary data in the regime of relatively large supervised training data sets. Thus, an augmented transformer may be a promising choice for proteins with limited evolutionary data. More accurate approximations to the transformer sequence log-likelihood could improve performance further.

Approaches now considered standard in the mainstream machine learning community that use supervised data to fine-tune deep probability density models, such as transformers and LSTMs, were herein outperformed by the augmented transformers and LSTMs. These results highlight that state-of-the-art techniques from NLP should be carefully considered in the context of protein modeling. One substantial difference present in our setting is the existence of weak-positively labeled evolutionary data, which has no direct analog in traditional NLP data sets.

As larger and more variable (i.e., spanning more of protein space) assay-labeled data sets emerge, we expect that increasingly richer models will dominate. Moreover, different machine learning strategies are likely needed for smaller and larger labeled data regimes, both of which are likely to remain relevant in protein engineering into the future. Therefore, when making general methodological recommendations, researchers should either consider implications across the full range of labeled data sizes or restrict recommendations to the regime tested.

To get an initial sense of how bringing in structure-based prediction might alter our narrative, we compared augmentation approaches with and without FoldX-derived stability features on three data sets, finding that FoldX features could provide new, useful information. Going forward, it will be beneficial to examine this topic more comprehensively, including the use of other tools such as Polyphen-2 (ref. [26]). PoPMuSiC[21], IMutant[22] and AlphaFold2 (ref. [57]), for example. As more features go into the augmentation, there are likely opportunities to use higher-capacity models for the augmentation model, beyond linear regression. These could be based on, for example, regression trees or deep learning.

Although our goal was to assess different methods on real data rather than assuming certain simulation settings are representative, these real data sets have their limitations and are not necessarily representative of the test sets that one might encounter for protein engineering. In particular, in a protein engineering setting, one would like to understand how the model performs across a larger swath of protein space. Most importantly, one would like predictive models to extrapolate as accurately as possible to higher fitness areas than that observed in the data[58], and to be able to gauge when such extrapolations are unreliable[17]. While it may be possible to leverage techniques from covariate shift adaptation to help in this regard[58,59], such approaches come with their own problems. We thus focused on directly evaluating methods without such corrections.

While the Spearman rank correlation has been used in previous studies[23,24,38,40,43], it does not capture all aspects of predictive performance. The Spearman rank correlation reflects a monotonic association between predicted and experimental mutational effects, but does not necessarily speak to anything about the mean squared error of those predictions, their scale, or any distributional characteristics of the fitness landscape, such as bimodality. However, as demonstrated by Wittman et al.[15], even modest Spearman correlation can be useful for speeding up iterative directed evolution. Our auxiliary use of NDCG, which also reflects distributional characteristics of the fitness landscape beyond the rank, provided added insight into the robustness of the Spearman rank correlation to assess predictive accuracy, as conclusions drawn from NDCG were generally concordant with those based on the Spearman rank correlation. Ultimately, the specific task at hand will determine the suitability of any given metric, and any metric will have its shortcomings for a given task.

While we have focused on evaluating predictive performance, in cases where the predictive model is used in conjunction with a design strategy[15,17–19,58], one should additionally evaluate how design and prediction perform in tandem in addition to independently.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41587-021-01146-5.

Received: 9 April 2021; Accepted: 2 November 2021;
Published online: 17 January 2022

References

1. Doudna, J. A. & Charpentier, E. The new frontier of genome engineering with CRISPR–Cas9. *Science* **346**, 1258096 (2014).
2. Hsu, P. D., Lander, E. S. & Zhang, F. Development and applications of CRISPR–Cas9 for genome engineering. *Cell* **157**, 1262–1278 (2014).
3. Chalfie, M., Tu, Y., Euskirchen, G., Ward, W. W. & Prasher, D. C. Green fluorescent protein as a marker for gene expression. *Science* **263**, 802–805 (1994).
4. Leader, B., Baca, Q. J. & Golan, D. E. Protein therapeutics: a summary and pharmacological classification. *Nat. Rev. Drug Discov.* **7**, 21–39 (2008).
5. Pollegioni, L., Schonbrunn, E. & Siehl, D. Molecular basis of glyphosate resistance–different approaches through protein engineering. *FEBS J.* **278**, 2753–2766 (2011).
6. Joo, H., Lin, Z. & Arnold, F. H. Laboratory evolution of peroxide-mediated cytochrome P450 hydroxylation. *Nature* **399**, 670–673 (1999).
7. Heim, R. & Tsien, R. Y. Engineering green fluorescent protein for improved brightness, longer wavelengths and fluorescence resonance energy transfer. *Curr. Biol.* **6**, 178–182 (1996).
8. Binz, H. K., Amstutz, P. & Plückthun, A. Engineering novel binding proteins from nonimmunoglobulin domains. *Nat. Biotech.* **23**, 1257–1268 (2005).
9. Arnold, F. H. Design by directed evolution. *Acc. Chem. Res.* **31**, 125–131 (1998).
10. Alford, R. F. et al. The Rosetta all-atom energy function for macromolecular modeling and design. *J. Chem. Theory Comput.* **13**, 3031–3048 (2017).
11. Karplus, M. & Kuriyan, J. Molecular dynamics and protein function. *Proc. Natl Acad. Sci. USA* **102**, 6679–6685 (2005).
12. Rocklin, G. J. et al. Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science* **357**, 168–175 (2017).
13. Russ, W. P. et al. An evolution-based model for designing chorismate mutase enzymes. *Science* **369**, 440–445 (2020).
14. Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl Acad. Sci. USA* **110**, E193–E201 (2013).
15. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* **12**, 1026–1045 (2021).
16. Bryant, D. H. et al. Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotech.* **39**, 691–696 (2021).
17. Brookes, D., Park, H. & Listgarten, J. Conditioning by adaptive sampling for robust design. In *Proc. International Conference on Machine Learning* (eds Chaudhuri, K. & Salakhutdinov, R.) 773–782 (PMLR, 2019).
18. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16**, 687–694 (2019).
19. Sinai, S. et al. AdaLead: a simple and robust adaptive greedy search algorithm for sequence design. Preprint at https://arxiv.org/abs/2010.02141 (2020).
20. Schymkowitz, J. et al. The FoldX web server: an online force field. *Nucleic Acids Res.* **33**, W382–W388 (2005).
21. Dehouck, Y., Kwasigroch, J. M., Gilis, D. & Rooman, M. Popmusic 2.1: a web server for the estimation of protein stability changes upon mutation and sequence optimality. *BMC Bioinform.* **12**, 151 (2011).
22. Capriotti, E., Fariselli, P. & Casadio, R. I-mutant2. 0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic Acids Res.* **33**, W306–W310 (2005).
23. Hopf, T. A. et al. Mutation effects predicted from sequence co-variation. *Nat. Biotech.* **35**, 128–135 (2017).
24. Riesselman, A. J., Ingraham, J. B. & Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods* **15**, 816–822 (2018).
25. Sim, N.-L. et al. SIFT web server: predicting effects of amino acid substitutions on proteins. *Nucleic Acids Res.* **40**, W452–W457 (2012).
26. Adzhubei, I. A. et al. A method and server for predicting damaging missense mutations. *Nat. Methods* **7**, 248–249 (2010).
27. Shihab, H. A. et al. Predicting the functional, molecular, and phenotypic consequences of amino acid substitutions using hidden Markov models. *Human Mutation* **34**, 57–65 (2013).
28. Mann, J. K. et al. The fitness landscape of hiv-1 gag: advanced modeling approaches and validation of model predictions by in vitro testing. *PLoS Comput. Biol.* **10**, e1003776 (2014).
29. Cheng, R. R., Morcos, F., Levine, H. & Onuchic, J. N. Toward rationally redesigning bacterial two-component signaling systems using coevolutionary information. *Proc. Natl Acad. Sci. USA* **111**, E563–E571 (2014).
30. Figliuzzi, M., Jacquier, H., Schug, A., Tenaillon, O. & Weigt, M. Coevolutionary landscape inference and the context-dependence of mutations in beta-lactamase tem-1. *Mol. Biol. E* **33**, 268–280 (2016).
31. Araya, C. L. et al. A fundamental protein property, thermodynamic stability, revealed solely from large-scale measurements of protein function. *Proc. Natl Acad. Sci. USA* **109**, 16858–16863 (2012).
32. Olson, C. A., Wu, N. C. & Sun, R. A comprehensive biophysical description of pairwise epistasis throughout an entire protein domain. *Curr. Biol.* **24**, 2643–2651 (2014).
33. Sarkisyan, K. S. et al. Local fitness landscape of the green fluorescent protein. *Nature* **533**, 397–401 (2016).
34. 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**, 1537–1551 (2013).
35. Wu, N. C., Dai, L., Olson, C. A., Lloyd-Smith, J. O. & Sun, R. Adaptation in protein fitness landscapes is facilitated by indirect paths. *eLife* **5**, e16965 (2016).

36. Otwinowski, J., McCandlish, D. M. & Plotkin, J. B. Inferring the shape of global epistasis. *Proc. Natl Acad. Sci. USA* **115**, E7550–E7558 (2018).
37. Shanehsazzadeh, A., Belanger, D. & Dohan, D. Is transfer learning necessary for protein landscape prediction? Preprint at https://arxiv.org/abs/2011.03443 (2020).
38. Rao, R. et al. Evaluating protein transfer learning with TAPE. In *Proc. Advances in Neural Information Processing Systems* (eds Wallach, H. et al.) 9689–9701 (Curran Associates, Inc., 2019).
39. Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M. & Church, G. M. Unified rational protein engineering with sequence-based deep representation learning. *Nat. Methods* **16**, 1315–1322 (2019).
40. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl Acad. Sci. USA* **118**, e2016239118 (2021).
41. Madani, A. et al. Deep neural language modeling enables functional protein generation across families. Preprint at *bioRxiv* https://doi.org/10.1101/2021.07.18.452833 (2021).
42. Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M. & Church, G. M. Low-N protein engineering with data-efficient deep learning. *Nat. Methods* **18**, 389–396 (2021).
43. Shamsi, Z., Chan, M. & Shukla, D. TLmutation: predicting the effects of mutations using transfer learning. *J. Phys. Chem. B.* **124**, 3845–3854 (2020).
44. Barrat-Charlaix, P., Figliuzzi, M. & Weigt, M. Improving landscape inference by integrating heterogeneous data in the inverse ising problem. *Sci. Rep.* **6**, 37812 (2016).
45. Howard, J. & Ruder, S. Universal language model fine-tuning for text classification. In *Proc. 56th Annual Meeting of the Association for Computational Linguistics, Vol. 1: long papers* (eds Gurevych, I. & Miyao, Y.) 328–339 (Association for Computational Linguistics, 2018).
46. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1: long and short papers*, 4171–4186 (2019).
47. Suzek, B. E. et al. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31**, 926–932 (2015).
48. Elnaggar, A. et al. ProtTrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing. Preprint at *bioRxiv* https://doi.org/10.1101/2020.07.12.199554 (2020).
49. Aghazadeh, A. et al. Epistatic net allows the sparse spectral regularization of deep neural networks for inferring fitness functions. *Nat. Commun.* **12**, 5225 (2021).
50. Starita, L. M. et al. Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis. *Proc. Natl Acad. Sci. USA* **110**, E1263–E1272 (2013).
51. Finn, R. D., Clements, J. & Eddy, S. R. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Res.* **39**, W29 (2011).
52. Järvelin, K. & Kekäläinen, J. Cumulated gain-based evaluation of ir techniques. *ACM Tran. Inf. Syst.* **20**, 422–446 (2002).
53. Gelman, S. et al. Neural networks to learn protein sequence-function relationships from deep mutational scanning data. *Proc. Natl Acad. Sci. USA* **118**, e2104878118 (2021).
54. Gray, V. E., Hause, R. J., Luebeck, J., Shendure, J. & Fowler, D. M. Quantitative missense variant effect prediction using large-scale mutagenesis data. *Cell Systems* **6**, 116–124 (2018).
55. Ingraham, J., Garg, V., Barzilay, R. & Jaakkola, T. Generative models for graph-based protein design. In *Proc. 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)* Vol. 32 (NeurIPS, 2019).
56. Hardt, M. & Recht, B.Patterns, predictions, and actions: A story about machine learning. Preprint at https://arxiv.org/abs/2102.05242 (2021).
57. Jumper, J. et al. Highly accurate protein structure prediction with alphafold. *Nature* **596**, 583–589 (2021).
58. Fannjiang, C. & Listgarten, J. Autofocused oracles for model-based design. In *Proc. 33rd Conference on Neural Information Processing Systems (NeurIPS 2020)* Vol. 33 (NeurIPS, 2020).
59. Sugiyama, M., Krauledat, M. & Müller, K.-R. Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res.* **8**, 985–1005 (2007).

Methods

Evolutionary sequences. We use the MSAs provided by EVmutation[23] whenever possible. For the GFP, the only exception, we follow the same procedure as EVmutation to gather sequences using the profile HMM homology search tool Jackhmmer[51]. We determine the bit score threshold in Jackhmmer search with the same criterion from EVmutation. In particular, for GFP, we started with 0.5 bits per residue and subsequently lowered the threshold to 0.1 bits per residue to meet the sequence number requirement (redundancy-reduced number of sequences $\geq 10L$, where L is the length of the aligned region). For sensitivity analysis, when using the bit score to 0.5 bits per residue or increasing the number of iterations to 10, the resulting MSAs on GFP still lead to similar downstream model performance (data not shown).

Mutation effect data sets. Hopf et al.[23] identified a list of mutation effect data sets generated by mutagenesis experiments of entire proteins, protein domains and RNA molecules. We exclude the data sets for RNA molecules and influenza virus sequences, as well as excluding data sets that contain fewer than 100 entries, to have meaningful train/test splits with at least 20 examples in test data. This leaves us with 18 data sets from EVmutation.

Following the convention in EVmutation and DeepSequence, we exclude sequences with mutations at positions that have more than 30% gaps in MSAs to focus on regions with sufficient evolutionary data. On most data sets, this excludes less than 10% of the data, although for a few proteins such as GFP this affects as much as half of the positions. For example, on GFP, out of 237 positions, only positions 15–150 pass the criterion of less than 30% gaps in the MSA. Coincidentally, the selected position 15–150 region covers the 81 amino region studied by Biswas et al.[42].

Among those 18 EVmutation data sets, only two on the poly(A)-binding protein activity[34] and the UBE4B auto-ubiquitination activity[50] include higher-order mutants (even though EVmutation Supplemental Table 1 indicates that the YAP1 WW domain 1 peptide binding data set also include higher-order mutants, the supplemental data for YAP1 only contain single-mutant sequences and no higher-order mutant data are available in the original publication[31] either.) While EVmutation only evaluates performance on the single mutants from the UBE4B U-box domain data[50], we also include higher-order mutants in evaluation. Additionally, we also include a GFP fluorescence data set[33], since GFP is a core example used by Biswas et al.[42]. The GFP data also include higher-order mutants. This results in 19 data sets total as listed in Supplementary Table 1.

Amino acid encodings. In addition to an overparameterized one-hot amino acid encoding (one binary feature for each amino acid possibility per position), we also experiment with the 19-dimensional physicochemical representation of the amino acid space developed by Georgiev[60], also used by Wittman et al.[15]. The features in Georgiev representation are principal components of over 500 amino acid indices from the AAIndex database[61]. As shown in Supplementary Fig. 8, the Georgiev encoding achieves better performance than the one-hot amino acid encoding on data sets with higher-order mutants when presented with sufficient training data (bottom right), agreeing with previous findings[15]. However, on single-mutant only data sets or on limited training examples, the Georgiev encoding leads to almost identical performance to the one-hot encoding. For the augmented Potts model, the Georgiev encoding also does not improve performance compared to the one-hot encoding in any setting.

Since the other evaluated methods, such as EVmutation[23], DeepSequence[24] and UniRep[39], all use the one-hot amino acid encoding as inputs to their models, we omit the Georgiev encoding results in the main text for fair comparison.

Linear model with one-hot encoded amino acid features. For a sequence $s=(s_1,\cdots,s_L)$ of length L, with elements $s_i \in \mathcal{A} = \{\text{ all amino acids }\}$, the linear model with one-hot encoded amino acid features maps the sequence to a scalar regression output by

$$f(s;\theta) = \theta_0 + \sum_{i=1}^{L}\sum_{a\in\mathcal{A}} \mathbb{1}_a(s_i)\theta_i(a) = \theta_0 + \sum_{i=1}^{L}\theta_i(s_i), \tag{1}$$

where $\theta_0 \in \mathbb{R}$ is the bias term (which we deem not to be part of the encoding), $\mathbb{1}_a(s_i)$ is an indicator function equal to 1 when $a=s_i$ and to 0 otherwise, and each $\theta_i \in \mathbb{R}^{|\mathcal{A}|}$ is an $|\mathcal{A}|$-dimensional vector. Together, we refer to all of these parameters as θ. Each coefficient, $\theta_i(a) \in \mathbb{R}$, corresponds to the effect of a amino acid a at position i, and we use $\theta_i(s_i)$ to denote the coefficient corresponding to the particular amino acid s_i.

This linear model is overparameterized by virtue of the one-hot encoding. In particular, there are $|\mathcal{A}| \times L + 1$ parameters for at most $(|\mathcal{A}|-1)\times L+1$ degrees of freedom. Additionally, if not all amino acids are observed at all positions in the training data, then the model will be correspondingly more overparameterized. As a consequence of this overparameterization, further constraints are needed to identify a unique solution. While there are different approaches for adding constraints, we chose to use ℓ_2-regularized regression (also known as ridge regression). We used fivefold cross-validation to determine the setting of the regularization parameter (λ below).

We deliberately chose to use this overparameterized approach over an approach with $|\mathcal{A}|-1$ nonredundant features, for reasons that will be explained shortly. Without regularization, this choice would be of no consequence. However, with regularization, the choice becomes consequential—both for the linear model with redundant one-hot encodings, and also for our augmented models that have a similar component. The main consequence of this choice is in how the trained model will make predictions on test sequences containing amino acids at positions that were not observed in the training data. Next we explain this in more detail.

Effects of ℓ_2 regularization on generalization. Minimizing the ℓ_2-regularized loss arising from the linear regression model in equation (1) is equivalent to minimizing the following objective function,

$$g(\theta) := \sum_{n=1}^{N}\left(y^{(n)} - f(s^{(n)};\theta)\right)^2 + \lambda\sum_{i=1}^{L}\sum_{a\in\mathcal{A}}\theta_i(a)^2, \tag{2}$$

where the scalar λ is the regularization hyper-parameter that dictates the strength of the regularization and $(s^{(n)}, y^{(n)})$ are the sequence-fitness pairs observed in the training data.

Generally, we think of ℓ_2-regularized linear regression as tantamount to putting a spherical Gaussian prior on the regression parameters and obtaining a point estimate of the parameters, if one is a Bayesian. Alternatively, one may view it as a shrinkage penalty that pushes the weights toward zero. However, in the specific context herein—of an overparameterized model with one-hot encoded features—the ℓ_2 regularization has some further interesting properties that are not widely known. First, the use of any positive value for λ in the ℓ_2 regularization in this context implies that at any given position, i, the optimal parameters, $\hat{\theta}_i(a)$ (that is, those that minimize equation (2)), at that position, sum up to zero. We refer to this as the zero-sum condition, which we now more formally define and prove, before discussing its consequences in more detail.

Lemma 1. For the zero-sum condition, let $\hat{\theta}$ be the optimal parameters that minimize the ℓ_2-regularized linear regression objective (equation (2)) on a linear model with one-hot encoded amino acid features (equation (1)). For any position i, the coefficients over all the amino acids always sum to zero. That is,

$$\sum_{a\in\mathcal{A}}\hat{\theta}_i(a) = 0.$$

Proof. By first-order optimality conditions, at a solution to equation (2), all of the partial derivatives of the ℓ_2-regularized objective (equation (2)) must all be equal to zero, namely,

$$\frac{\partial g(\theta)}{\partial\theta_0} = \left.\frac{\partial\sum_{n=1}^{N}\left(y^{(n)} - f(s^{(n)};\theta)\right)^2}{\partial\theta_0}\right|_{\theta=\hat{\theta}} = 0, \quad \text{and} \tag{3}$$

$$\frac{\partial g(\theta)}{\partial\theta_i(a)} = \left.\frac{\partial\sum_{n=1}^{N}\left(y^{(n)} - f(s^{(n)};\theta)\right)^2}{\partial\theta_i(a)}\right|_{\theta=\hat{\theta}} + 2\lambda\hat{\theta}_i(a) = 0 \ \text{ for all } \ i=1,\ldots,L \text{ and all } a\in\mathcal{A}. \tag{4}$$

Furthermore, the sum of the indicator functions appearing in equation (1), for position i, over all amino acids, is always equal to one,

$$\sum_{a\in\mathcal{A}}\mathbb{1}_a(s_i) = 1, \tag{5}$$

from which we see that

$$\sum_{a\in\mathcal{A}}\frac{\partial f(s;\theta)}{\partial\theta_i(a)} = \sum_{a\in\mathcal{A}}\frac{\partial\left(\theta_0 + \sum_{i=1}^{L}\theta_i(s_i)\right)}{\partial\theta_i(a)} = \sum_{a\in\mathcal{A}}\mathbb{1}_a(s_i) = 1. \tag{6}$$

It also straightforwardly holds that the partial derivative of the model with respect to the bias is always equal to one, and therefore by the previous result is also equal to the sum of partial derivatives,

$$\frac{\partial f(s;\theta)}{\partial\theta_0} = \frac{\partial\left(\theta_0 + \sum_{i=1}^{L}\theta_i(s_i)\right)}{\partial\theta_0} = 1 = \sum_{a\in\mathcal{A}}\frac{\partial f(s;\theta)}{\partial\theta_i(a)}. \tag{7}$$

It therefore follows by two applications of the chain rule that for any value of θ,

$$\frac{\partial\sum_{n=1}^{N}\left(y^{(n)} - f(s^{(n)};\theta)\right)^2}{\partial\theta_0} = -2\sum_{n=1}^{N}\left(y^{(n)} - f(s^{(n)};\theta_0)\right)\frac{\partial f(s^{(n)};\theta)}{\partial\theta_0} \tag{8}$$

$$= -2\sum_{n=1}^{N}(y^{(n)} - f(s^{(n)};\theta))\sum_{a\in\mathcal{A}}\frac{\partial f(s^{(n)};\theta)}{\partial\theta_i(a)} \quad (9)$$

$$= \sum_{a\in\mathcal{A}} -2\sum_{n=1}^{N}(y^{(n)} - f(s^{(n)};\theta))\frac{\partial f(s^{(n)};\theta)}{\partial\theta_i(a)} \quad (10)$$

$$= \sum_{a\in\mathcal{A}}\frac{\partial\sum_{n=1}^{N}(y^{(n)} - f(s^{(n)};\theta))^2}{\partial\theta_i(a)}. \quad (11)$$

Now, more specifically at the optimal $\hat{\theta}$ that minimizes the objective, combining equation (3) with equation (11), we have

$$\sum_{a\in\mathcal{A}}\left.\frac{\partial\sum_{n=1}^{N}(y^{(n)} - f(s^{(n)};\theta))^2}{\partial\theta_i(a)}\right|_{\theta=\hat{\theta}} = \left.\frac{\partial\sum_{n=1}^{N}(y^{(n)} - f(s^{(n)};\theta))^2}{\partial\theta_0}\right|_{\theta=\hat{\theta}} = 0, \quad (12)$$

which when combined with equation (4), shows that

$$2\lambda\sum_{a\in\mathcal{A}}\hat{\theta}_i(a) = 0. \quad (13)$$

Therefore, so long as $\lambda > 0$, the ℓ_2-regularized solution must satisfy the zero-sum condition stated above, $\sum_{a\in\mathcal{A}}\hat{\theta}_i(a) = 0$.

Generalization to mutations not seen at train time. If amino acid, α, at position i was not seen in the training data, then ridge regression sets $\hat{\theta}_i(\alpha) = 0$, so as to minimize the ℓ_2 penalty (Lemma 2, below). While we use a to generally denote amino acids, we use α to specifically refer to amino acids that are not seen at position i in the training data. A consequence of this property, jointly with the zero-sum condition, is that the effect, $\hat{\theta}_i(\alpha)$, of mutation α at position i on the predicted fitness, is precisely equal to the average effect of all amino acids seen at train time at that position, namely, $\hat{\theta}_i(\alpha) = \frac{1}{|\mathcal{A}_i^{\text{train}}|}\sum_{a\in\mathcal{A}_i^{\text{train}}}\hat{\theta}_i(a)$ (Proof below), where $\mathcal{A}_i^{\text{train}}$ denotes the set of all amino acids (including the wild-type) seen at position i in the training data.

Consequently, the model will generalize to unseen amino acids at a given position in a manner that 'understands' how mutable a site is, where mutability is the ability of the site to tolerate mutations while maintaining fitness. That is, if a given site tends to yield poor fitness when mutated, then its average training data effect will tend to be poor and the extrapolated fitness at test time to a new amino acid will be predicted to be poor. To explain how much that information (mutability at a site) might perform on its own, we also tried a predictive model consisting of nothing but the position of the mutation (0/1) in a linear regression model (Extended Data Fig. 10 and Supplementary Figs. 9–11). We found that this model is very predictive given that it is unaware of the specifics of any particular amino acids. In particular, it performs close to, but slightly worse than the linear model with one-hot encoded amino acid features. Notably, use of a nonredundant encoding in the same ℓ_2 modeling framework would not enable such predictions based on mutability to be made. In particular, in using one parameter per possible mutation from wild-type at each position, the model would predict zero effect for mutations not seen at train time.

The mechanism for the site-specific generalization described above is similarly present in the augmented models because of their ℓ_2-regularized linear regression with one-hot encoded features, which retain an analog to these properties even when augmented with a density feature. Note that one might consider altering the regularization to a more nuanced version wherein the emergent average amino acid prediction at a site is weighted in a manner to account for chemical similarity of amino acids, their frequency in the training data and so forth.

Lemma 2. For the site-specific generalization effect, let $\mathcal{A}_i^{\text{train}}$ denote the set of all amino acids (including the wild-type (WT)) seen at position i in the training data $\mathcal{D}_{\text{train}}$. Let $\alpha \notin \mathcal{A}_i^{\text{train}}$ be an amino acid at position i that was not seen in the training data. For the optimal parameters $\hat{\theta}$ that minimize the ridge regression objective, we must have $\hat{\theta}_i(\alpha) = 0$. Furthermore, $\hat{\theta}_i(\alpha)$ is the average effect of all seen amino acids at that position in the sense that

$$\hat{\theta}_i(\alpha) = \frac{1}{|\mathcal{A}_i^{\text{train}}|}\sum\nolimits_{a\in\mathcal{A}_i^{\text{train}}}\hat{\theta}_i(a)$$

and

$$\hat{\theta}_i(\alpha) - \theta_i(a_{\text{WT}}) = \frac{1}{|\mathcal{A}_i^{\text{train}}|}\sum\nolimits_{a\in\mathcal{A}_i^{\text{train}}}(\hat{\theta}_i(a) - \theta_i(a_{\text{WT}}))$$

.

Proof. For an amino acid α that was not seen at position i in the training data, since $s_i^{(n)} \neq \alpha$ for any sequence $s^{(n)}$ in the training data, the first term in the objective function (equation (2)) does not depend on $\theta_i(\alpha)$. From the first-order optimality condition we therefore have

$$\left.\frac{\partial g(\theta)}{\partial\theta_i(\alpha)}\right|_{\theta=\hat{\theta}} = 2\lambda\hat{\theta}_i(\alpha) = 0 \Rightarrow \hat{\theta}_i(\alpha) = 0.$$

From the zero-sum condition,

$$\sum_{a\in\mathcal{A}_i^{\text{train}}}\hat{\theta}_i(a) + \sum_{a\notin\mathcal{A}_i^{\text{train}}}\hat{\theta}_i(a) = 0. \quad (14)$$

Since $\hat{\theta}_i(a) = 0$ for $a \notin \mathcal{A}_i^{\text{train}}$, we are then left with

$$\sum_{a\in\mathcal{A}_i^{\text{train}}}\hat{\theta}_i(a) = 0. \quad (15)$$

Therefore, for any $\alpha \notin \mathcal{A}_i^{\text{train}}$

$$\hat{\theta}_i(\alpha) = 0 = \frac{1}{|\mathcal{A}_i^{\text{train}}|}\sum_{a\in\mathcal{A}_i^{\text{train}}}\hat{\theta}_i(a). \quad (16)$$

If we prefer to consider instead the mutational effect relative to wild-type, then we can obtain an analogous result simply by subtracting $\theta_i(a_{WT})$ from both sides of the above equation (where a_{WT} denotes the wild-type amino acid), to get

$$\hat{\theta}_i(\alpha) - \theta_i(a_{\text{WT}}) = \frac{1}{|\mathcal{A}_i^{\text{train}}|}\sum_{a\in\mathcal{A}_i^{\text{train}}}(\hat{\theta}_i(a) - \theta_i(a_{\text{WT}})), \quad (17)$$

indicating that the results are true both for absolute and relative effects.

EVmutation Potts model. Briefly, EVmutation learns a sequence distribution under site-specific constraints and pairwise constraints for each protein family. Under a Potts model (also sometimes known as a generalized Ising model) with alphabet $\mathcal{A}$, the probability of a sequence $s = (s_1, \cdots, s_L) \in \mathcal{A}^L$ of length L is given by a Boltzmann distribution:

$$P(s) = \frac{1}{\mathcal{Z}}\exp\{-\mathcal{H}(s)\}, \quad \mathcal{H}(s) = -\sum_{i<j}^{L}J_{ij}(s_i, s_j) - \sum_{i}^{L}h_i(s_i)$$

where $\mathcal{Z}$ is a normalization constant, $h_i \in \mathbb{R}^{|\mathcal{A}|}$ are site-specific amino acid-specific parameters, $J_{ij} \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{A}|}$ are pair-specific amino acid-specific parameters and $s_i \in \mathcal{A}$ indicates the amino acid at position i. The h and J parameters are estimated by regularized maximum pseudo-likelihood. See ref. [23] for the full modeling fitting details of sequence reweighting, pseudo-likelihood approximation, regularization and optimization. We use the same plmc package (May 2018) as EVmutation to fit Potts models with default parameters.

Under the assumption that learned sequence probabilities are correlated with sequence fitness, EVmutation predicts the mutation effect of a mutant according to the log-odds ratios of sequence probabilities between the wild-type and mutant sequences, which is equivalent to the log-likelihood of the mutant up to a constant.

Profile HMMs. Profile HMMs[62] are probabilistic models that capture position-specific information about the amino acid distribution at each site, assuming that the amino acid at a particular position is independent of the amino acid at all other positions. Compared to Potts models, profile HMMs do not contain pairwise interactions and are also referred to as the independent model or the independent site model. We evaluated the HMMER suite profile HMM implementation[51] and the EVmutation independent model implementation[23] on the same evolutionary sequences. There is a minor performance difference (Supplementary Fig. 12) between the two implementations due to algorithmic differences, and the 'profile HMM' and 'augmented HMM' methods in this paper are based on the HMMER version.

DeepSequence VAE. Similar to EVmutation, DeepSequence[24] also models a sequence distribution for each protein family and predicts mutation effects according to approximations of log-odds ratios of sequence probabilities between the wild-type and mutant sequences. Here, the sequence distribution is modeled by a nonlinear VAE model with a multivariate Gaussian latent variable z. A neural network parameterizes the conditional distribution $P(s|z,\theta)$, and the sequence likelihood $P(s|\theta)$ is then

$$P(s|\theta) = \int P(s|z,\theta)P(z)\mathrm{d}z. \quad (18)$$

While the exact log-likelihoods are intractable, they are lower-bounded by the evidence lower bound (ELBO)

$$\mathcal{L}(\phi;s) = E_q[\log p(s|z,\theta)] - D_{KL}(q(z|s,\phi)||p(z)),$$

where $q(z|s,\phi)$ is a variational approximation for the posterior distribution $p(z|s,\theta)$ that is also modeled by a neural network. As an approximation to log-likelihoods, the ELBO can also be used to predict mutation effects. Whenever possible, we use the available ensemble predictions from DeepSequence[24]. For the GFP, the UBE4B U-box domain and the BRCA1 ring domain, we followed the DeepSequence code in Theano and parameters to train an ensemble of VAE models (five models with different random seeds). When computing the evidence-based lower bound (ELBO) as sequence log-likelihood estimations, we follow DeepSequence to take the average of 2,000 ELBO samples (400 samples from each of the five VAE models in the ensemble).

eUniRep mLSTM. The UniRep[39] mLSTM can also be viewed as a sequence distribution. Given the first i amino acids of a sequence, a neural network parameterizes the conditional probabilities of the next amino acid $P(s_{i+1}|s_1\cdots s_i)$. From the conditional probabilities we can also reconstruct the sequence probability as

$$P(s) = \Pi_{i=1}^{L} P(s_i|s_1\cdots s_{i-1}).$$

For convenience, artificial start and stop tokens are added at the two ends of sequences. Under this probabilistic interpretation, the unsupervised pretraining on UniRef50 sequences with next-step predictions is equivalent to finding model parameters that maximize likelihood on UniRef50 sequences. We omit the performance of the pretrained (not evo-tuned) UniRep model in our results since generally it is largely outperformed by the eUniRep described below.

To adapt the UniRep model to specific protein families, Biswas et al.[42] propose the 'evo-tuning' procedure, which fine-tunes the UniRep model by minimizing the same next-step prediction loss on evolutionary sequences in addition to on UniRef50 sequences. Following the naming convention[42], We refer to the model after 'evo-tuning' as eUniRep. After fine-tuning, Biswas et al.[42] then perform supervised regression with the 1,900-dimensional average embeddings (averaged over sequence length axis) from the final hidden layer in the eUniRep model as regression features.

While the original fine-tuning procedure for eUniRep[42] optimizes for next-step prediction loss on entire evolutionary sequences up to length 500 (discarding longer sequences), we find that next-step prediction on aligned portions of evolutionary sequences (with gaps included as gap tokens) works as well on GFP and beta-lactamase (Supplementary Fig. 13), and therefore fine-tune on only aligned portions to lower computational costs as the mLSTM memory usage scales quadratically with respect to sequence length. We used the open-source UniRep code in tensorflow.

For each protein family, we randomly split the evolutionary sequences from the MSA into an 80% training set and a 20% validation set to check for over-fitting. When training for 10,000 gradient steps with the same learning rate 1×10^{-5} as Biswas et al.[42], the validation loss eventually plateaus but does not increase. Since the validation loss typically plateaus before 10,000 steps, we chose to stop training at 10,000 steps.

ESM-1b transformer. The ESM-1b transformer[40] is pretrained on UniRef50 representative sequences with the masked language modeling objective, where a fraction (15%) of amino acids in each input sequence are masked and the model is trained to predict the missing tokens. We chose the ESM-1b model to represent transformers as it has the best performance on downstream secondary structure prediction and contact prediction tasks[40] among transformers from previous studies[38,48] and other transformer architectures tested. While the original mutation effect prediction results[40] were based on the older 34-layer ESM-1 model, we found that ESM-1b slightly improves performance over the ESM-1 model (data not shown).

When using transformer models for sequence-fitness predictions, Rives et al.[40] mask the mutated positions and used the difference in conditional log-likelihoods (conditioned on nonmutated amino acids) between the mutated amino acids and the wild-type amino acids as fitness prediction. We explain below that this could be viewed as an approximation for pseudo-log-likelihoods (PLLs).

Masked token language models can also be viewed as sequence distributions, where the sequence distribution is implicitly represented by conditional likelihoods $P(s_i|s_{-i})$, where s_{-i} indicates all other sequence positions excluding position i, that is, $s_{-i}=s_1\cdots s_{i-1}s_{i+1}\cdots s_L$.

Since exact likelihoods are too computationally expensive for Transformers, we resort to pseudo-likelihoods. In general, given a sequence s of length L, its pseudo-likelihood[63] is defined as the product of conditional likelihoods for each site. Hence, the pseudo-log-likelihood of a sequence s is

$$\text{PLL}(s) = \sum_{i=1}^{L} \log P(s_i|s_{-i}).$$

However, even the evaluation of PLLs is computationally expensive, since it requires L inferences for a sequence of length L. As a more computationally efficient approximation, we only compute conditional likelihoods on the mutated positions, and then use the difference between the conditional log-likelihoods of the mutated sequence and the wild-type sequence as an approximation for PLLs. More specifically, for a mutant sequence ϕ and wild-type sequence σ, we approximate the PLL difference by the following, as equivalent to the existing formula used by Rives et al.[40]:

$$\begin{aligned} PLL(\phi) - PLL(\sigma) &= \sum_{i=1}^{\ell} \log P(\phi_i|\phi_{-i}) - \sum_{i=1}^{\ell} \log P(\sigma_i|\sigma_{-i}) \\ &= \sum_{i:\phi_i\neq\sigma_i} (\log P(\phi_i|\phi_{-i}) - \log P(\sigma_i|\sigma_{-i})) \\ &+ \underbrace{\sum_{i:\phi_i=\sigma_i} (\log P(\sigma_i|\phi_{-i}) - \log P(\sigma_i|\sigma_{-i}))}_{\approx 0} \end{aligned}$$

The underlying assumption here is that the conditional log-likelihoods of wild-type amino acids on mutant backgrounds are roughly the same as on wild-type backgrounds. While this might be accurate for mutant sequences that are close enough to wild-type sequences, in general there is no support for this approximation on high-order mutant sequences and the full PLLs will likely be more accurate for mutation effect predictions.

For supervised learning, we evaluated two approaches with the same Transformer model. The first one ('fine-tuned transformer') is to fine-tune the entire Transformer model with fitness labels as done by Rives et al.[40], using the PLL difference between the mutant and the wild-type as a predictor for fitness. We perform 20 epochs of supervised fine-tuning with learning rate 3×10^{-5} and with early stopping according to validation Spearman correlation, using the open-source ESM code in PyTorch. For a given training data size, we keep 20% of the data for validation (early stopping) and use the remaining 80% for fine-tuning. Using Spearman correlation as opposed to validation loss for early stopping is crucial for performance, especially on small data sizes. Although the implementation details might not exactly match those from Rives et al. (since the fine-tuning code is not available), we show that our method is able to reproduce the same results on most of the Envision data sets used by Rives et al., as shown in Supplementary Fig. 14.

In the second approach ('augmented transformer'), while keeping the transformer model constant, we concatenate the PLL difference inferred from the pretrained (not fine-tuned) model together with one-hot amino acid encoding as features for regression.

We also attempted unsupervised fine-tuning ('evo-tuning') of the ESM-1b transformer model on evolutionarily related sequences from MSAs, although our preliminary efforts on β-lactamase and PABP-RRM do not result in improved performance (data not shown).

Integrated (tied-energy) Potts model. The integrative approach[44] for Potts models optimizes the joint log-likelihood for model parameters on both evolutionary data and labeled data. Given evolutionary sequences $\sigma_1,\cdots,\sigma_M$, the log-likelihood on evolutionary data is

$$\log P(\sigma^1,\cdots,\sigma^M|J,h) = -\sum_{k=1}^{M}\mathcal{H}(\sigma^k) - M\log\mathcal{Z},$$

where

$$\mathcal{H}(\sigma) = -\sum_{i<j}^{L} J_{ij}(\sigma_i,\sigma_j) - \sum_{i}^{L} h_i(\sigma_i).$$

On the other hand, given sequence-fitness pairs $(s_1,y_1),\cdots,(s_N,y_N)$, assuming independent and identically distributed Gaussian noise drawn from $\mathcal{N}(0,\Delta^2)$, the log-likelihood on labeled data is

$$\log P(s^1,\cdots,s^M,y^1,\cdots,y^M|J,h) = -\frac{1}{2\Delta^2}\sum_{k=1}^{N}[y^k - \mathcal{H}(s^k)]^2 - \frac{N}{2}\log(2\pi\Delta^2).$$

The integrated Potts model is learned by maximizing the joint log-likelihood $\log P(\sigma^1,\cdots,\sigma^M|J,h) + \log P(s^1,\cdots,s^M,y^1,\cdots,y^M|J,h)$ with ℓ_2-regularization.

The noise variance Δ^2 determines the relative weighting between the two losses in the joint log-likelihood. Using existing notation[44], the relative weighting parameter is

$$\lambda = \frac{1}{1+\Delta^2}.$$

In practice, since we do not know the noise variance, we use 20% of the training data for validation and choose the best λ according to Spearman correlation on validation set. Another practical complication is that we only have fitness labels that are up to some monotonic transformation from the energies. Following Figliuzzi et al.[30], we use a monotonic mapping between sorted energy values and sorted fitness labels.

Following recommendations from the authors, the Potts model parameters are initialized with parameters estimated by pseudo-likelihood. Before introducing labeled data, we first give the model a warm start by training 500 iterations only on evolutionary data. Then, we optimize the regularized joint log-likelihood for 100 iterations for each λ value. The only exception is for β-glucosidase, where the protein sequences are too long (>500 amino acids) and lead to very high memory consumption and long run-time (>2 days) for the integrated Potts model. Therefore, for β-glucosidase alone we use the Potts model performance without labeled data as a substitute for the integrated model performance.

We follow the publicly available code (https://github.com/PierreBarrat/DCATools/tree/master/src) with slight modifications to use zero-sum gauge instead of wild-type gauge for Potts models. In Potts models for categorical variables, there are more free parameters than independent constraints, and gauge fixing refers to reducing the number of independent parameters to match the number of independent constraints[64]. The wild-type gauge forces all parameters corresponding to wild-type amino acids to be zero, while the zero-sum gauge requires $\sum_{\omega\in\mathcal{A}} h_i(\omega) = 0$ and $\sum_{\omega\in\mathcal{A}} J_{ij}(\omega,\alpha) = \sum_{\omega\in\mathcal{A}} J_{ij}(\alpha,\omega) = 0$. When there is regularization involved, different gauge fixings lead to nonequivalent models. Although gradient calculations are easier in the wild-type gauge, we found that the zero-sum gauge led to better predictive performance (Supplementary Fig. 15) and hence adopted the zero-sum gauge.

Alternative tied-energy models. In addition to integrated Potts models, we also evaluate alternative ways to train a density model of a protein family and a predictive model of fitness in an integrated fashion (Supplementary Fig. 16). Rather than using a fixed and possibly incorrect monotonic mapping between sorted energy values and sorted fitness labels[30], we consider a differentiable proxy of the Spearman correlation[65] between the predicted and true fitness values. Specifically, given N sequence-fitness pairs, $(s_1, y_1), \cdots, (s_N, y_N)$, let $(s_1, r(y_1)), \cdots, (s_N, r(y_N))$ denote the corresponding sequence-ranking pairs, where $r(y_k) \in \{1, \ldots, N\}$ gives the rank in descending order of the value y_k among the values $\{y_1, \ldots, y_N\}$. We fit a Potts model by optimizing the joint loss

$$\frac{1}{M}\log P(\sigma^1, \cdots, \sigma^M | J, h) + \lambda\frac{1}{N}\sum_{k=1}^{N}[r(y^k) - \hat{r}(\mathcal{H}(s^k))]^2$$

with ℓ_2-regularization, where $\hat{r}(\mathcal{H}(s^k))$ is a differentiable proxy[65] of the rank $r(\mathcal{H}(s^k))$. For tractability, we optimize the pseudo-likelihood approximation of the log-likelihood term in the joint loss. We choose the value of $\lambda \in \{0.0001, 0.001, 0.01, 0.1, 0.9\}$ with fivefold cross-validation.

To explore the effect of training energy-based models other than the Potts model in this fashion, we use the same procedure to fit an energy-based model with an energy function parameterized by a two-layer feed-forward neural network with hidden layer sizes of (300, 100) and rectified linear activation function activations. As with the Potts model, we optimize the pseudo-likelihood approximation of the log-likelihood term in the joint loss.

BLOSUM62 substitution scores. For a mutant sequence, we sum the BLOSUM62 substitution scores between every amino acid in the wild-type sequence and the mutant sequence. The scores are then offset by the a constant such that a score of zero is obtained for the wild-type sequence.

Augmented models. For augmentation, we rely on an already-trained probability density model on a set of evolutionarily related sequences, that does not get altered. To augment this existing density model, we concatenate the sequence density estimation from the original model together with one-hot encoded site-specific amino acid features, as illustrated in Fig. 1b. Mathematically, following the same notation for linear models with one-hot encoded features, given a density model with sequence probability distribution $p(s;\varphi)$ and fixed density model parameters φ, the corresponding augmented model is

$$f(s;\varphi,\theta,\beta) = \beta\log P(s;\varphi) + \theta_0 + \sum_{i=1}^{L}\theta_i(s_i),$$

where the parameters $\theta_i \in \mathbb{R}^{|\mathcal{A}|}$, $\theta_0 \in \mathbb{R}$ and $\beta \in \mathbb{R}$ are learned from assay-labeled data. When performing ridge regression with this augmented model, the regularization strength for β was set to make this feature practically unregularized (regularization strength, 10^{-8}), while the other parameters, θ, were regularized using a common strength determined by cross-validation. For density models where computing exact log-likelihoods is challenging, we use one of various approximations. For Potts models and transformers, we use the pseudo-log-likelihood, and for VAEs, the ELBO. Similar to the setup of a linear model with one-hot encoded site-specific amino acid features, we again used the overparameterized one-hot site-specific amino acid encoding with $|\mathcal{A}|$ features per position (one binary feature for each amino acid possibility per position) in combination with ℓ_2 regularization. See the section on Effects of ℓ_2 regularization on generalization above for more details on how this choice influences model prediction on mutations not seen in the training data.

The augmented Potts model as an evolutionary prior for linear regression. One can view the augmented Potts model as a tailoring of the implicit prior in the ℓ_2-regularized (ridge) linear regression with site-specific amino acid features, where the tailoring is based on evolutionary information from the density model. First, note that in the Bayesian, maximum a posteriori (MAP) interpretation of ridge regression, a zero-mean Gaussian prior probability has been given to the ridge regression parameters, $\theta_i(\alpha) \approx \mathcal{N}(0, \tau^2)$. Now consider the augmented Potts model, with already-fitted Potts model, $p(s;\varphi \equiv \{h_i, J_{ij}\})$, with site-specific parameters, $\{h_i(\alpha)\}$ at position i for amino acid α and analogous coupling parameters, $\{J_{ij}(\alpha_1, \alpha_2)\}$. If in the augmented Potts model, the Potts model density feature was independent from the site-specific amino acid features, one could in a two-step procedure first fit the density-associated parameter, $\beta = \hat{\beta} \in \mathbb{R}$, and having fit that, then fit the remaining parameter vector, θ. Under the independence assumption, the resulting parameter estimates from the two-step procedure would be identical to having estimated them jointly as in regular ridge regression (with no regularization on the density feature, as done for the augmented Potts model). In such a two-step fitting procedure, the second step would correspond to performing ridge regression on the residuals $y' = y - \hat{\beta}\log p(s;\varphi)$. Correspondingly, the implicit prior of ridge regression gets tailored by shifting its mean by an amount, $\hat{\beta}\tilde{h}_i(\alpha)$, corresponding to the Potts model site-specific parameters, to back-interpret the prior over the parameters to be $\theta_i(\alpha) \approx \mathcal{N}(\hat{\beta}\tilde{h}_i(\alpha), \tau^2)$. Note that the effect of the coupling parameters in the Potts model is entirely mediated through the estimate of $\hat{\beta}$. Also note that the assumption of independence of the Potts model density feature from the other features is unlikely to be true in most practical cases; however, this does not necessarily detract from the overall intuitive interpretation.

Model evaluation. For each data set, we randomly sample 20% of the data set as held-out test data. Among the remaining 80% data, we randomly sample $N = 24, 48, 72, 96, \cdots$, or 240 single-mutant sequences as training data in separate experiments, or use all single-mutant sequences in the 80% training data in the 80/20 split experiments. When computationally feasible (for the linear model, eUniRep regression, and all augmented models), we use fivefold cross-validation to determine hyperparameters. Otherwise, for the fine-tuned transformer and the integrated Potts model, we set aside 20% of the training data to determine hyperparameters (that is, the number of fine-tuning epochs for the transformer and the relative weighting between evolutionary and assay-labeled data for the integrated Potts model). For each fixed sample size, the model evaluation procedure is repeated 20 times with different random seeds for uncertainty estimation. The only exception is for the integrated Potts model, where we only use five random seeds due to the long computation time. The 95% confidence intervals of the means are estimated via bootstrapping.

NDCG. DCG and its normalized version, NDCG, are both widely used measures of ranking quality in information retrieval. NDCG has also recently been used in the protein engineering community to assess fitness prediction[15]. The NDCG score can be seen as a smoothed version of the mean fitness of the top k predictions, without having to fix an arbitrary k. In the top k mean metric, the true fitness values of variants are summed with a binary weight of either $1/k$ or 0 depending on whether the variant is in the top k or not. Instead of the binary weight, NDCG sums over the variants with a smoothed logarithmic weight.

Given a model's prediction on M protein variants, to calculate DCG we first sort all variants by predicted scores into an ordered rank list $\hat{y}_1 \geq \hat{y}_2 \geq \cdots \geq \hat{y}_M$. Then, DCG sums the true fitness values y with a logarithmic discount according to the rank order, where a variant's true fitness value contributes more to the sum if it is more highly ranked,

$$\mathrm{DCG} = \sum_{i=1}^{M}\frac{y_i}{\log(i+1)}.$$

The best (highest) possible DCG on a set of M protein variants occurs when all protein variants are predicted to be in the exact same rank order as the true fitness. On different data sets, the value of the best possible DCG is different, depending on the true fitness values in the data set. Consequently, the DCG is often normalized to make the values more comparable between data sets. To do so, we first standardize the true scores y to have zero mean and unit variance. Then, we normalize the DCG by dividing by the best possible DCG (that is, obtained for a perfect ranking) to obtain the NDCG, which therefore lies between 0 and 1 for all data sets.

As an example, consider three protein variants A, B and C with the respective true fitness values, −0.8, 0.6 and 0.2, and predicted fitness values, 0.1, 0.9 and −0.1. First we sort them in predicted order B > A > C. Then the DCG from the predicted model is $0.6/\log(2) + (-0.8)/\log(3) + 0.2/\log(4) = 0.195$, while the ideal DCG from the perfect ranking is $0.6/\log(2) + 0.2/\log(3) + (-0.8)/\log(4) = 0.326$. The ratio between the DCG and the ideal DCG, 0.195/0.326 = 0.598, is then the NDCG.

Mann–Whitney *U*-test. We use the two-sided nonparametric Mann–Whitney *U*-test for comparing average performance between different methods. The null hypothesis of the *U*-test is that, for randomly selected values *X* and *Y* from two populations, the probability of *X* being greater than *Y* is equal to the probability of *Y* being greater than *X*. The alternative hypothesis is that one population is stochastically greater than the other. In the context of Fig. 2, since we are comparing the average performance over all data sets, the population consists of the average performance of a given method computed from different random seeds. This satisfies the assumption that the observations in each population are independent of each other.

FoldX. The FoldX suite[20] evaluates the effect of mutations on the stability of proteins. To derive stability features from FoldX, we use the 'total energy' output from the BuildModel command in FoldX 5.0. We used Protein Data Bank (PDB) structure 2WUR for the GFP following Sarkisyan et al.[33], and PDB structures 6R5K and 2KR4 for the poly(A)-binding protein RRM domain and the ubiquitination factor E4B U-box domain, respectively. Unfortunately, among these three proteins, only the GFP has atomic resolution structure determined by X-ray crystallography. The only structures available for the other two domains are determined by nuclear magnetic resonance and electron microscopy at lower resolutions. Higher-resolution structures could potentially make FoldX-derived features more useful on those two domains.

Reporting Summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability
All protein fitness data were publicly available through citations available in the paper. A processed version of these data and our evaluation results are available on Dryad with https://doi.org/10.6078/D1K71B. All protein structures used in the study are available publicly with PDB IDs 2WUR, 6R5K and 2KR4.

Code availability
The code to reproduce the results is available at https://github.com/chloechsu/combining-evolutionary-and-assay-labelled-data.

References
60. Georgiev, A. G. Interpretable numerical descriptors of amino acid space. *J. Comput. Biol.* **16**, 703–723 (2009).
61. Kawashima, S. et al. Aaindex: amino acid index database, progress report 2008. *Nucleic Acids Res.* **36**, D202–5 (2007).
62. Eddy, S. R. Profile hidden Markov models. *Bioinformatics* **14**, 755–763 (1998).
63. Besag, J. Statistical analysis of non-lattice data. *J. Royal Stat. Soc.: Ser. D. Statistician* **24**, 179–195 (1975).
64. Stein, R. R., Marks, D. S. & Sander, C. Inferring pairwise interactions from biological data using maximum-entropy probability models. *PLoS Comput. Biol.* **11**, e1004182 (2015).
65. Blondel, M., Teboul, O., Berthet, Q. & Djolonga, J. Fast differentiable sorting and ranking. In *Proc. International Conference on Machine Learning* (eds Hal, D., III & Aarti, S.) 950–959 (PMLR, 2020).

Acknowledgements
We thank A. Aghazadeh, P. Almhjell, F. Arnold, A. Busia, D. Brookes, M. Jagota, K. Johnston, L. Schaus, N. Thomas, Y. Wang and B. Wittmann for helpful discussions. We also thank P. Barrat-Charlaix, S. Biswas, J. Meier and Z. Shamsi for providing helpful details about their methods and implementations. Partial support was provided by the US Department of Energy, Office of Biological and Environmental Research, Genomic Science Program Lawrence Livermore National Laboratory's Secure Biosystems Design Scientific Focus Area under grant award no. SCW1710 (J.L., C.H.), the Chan Zuckerberg Investigator program (J.L.) and C3.ai (J.L., H.N.). Research reported in this publication was supported by the National Library Of Medicine of the National Institutes of Health under grant award no. T32LM012417 (H.N.). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. This material is based on work supported by the National Science Foundation Graduate Research Fellowship Program under grant no. DGE 2146752 (C.F.).

Author contributions
C.H. and J.L. conceptualized the study and developed the methodology. C.H. implemented models and analyzed data, with contributions from H.N. and C.F. All authors wrote the paper.

Competing interests
J.L. is on the Scientific Advisory Board of Patch Biosciences and Foresite Laboratories. The remaining authors declare no competing interests.

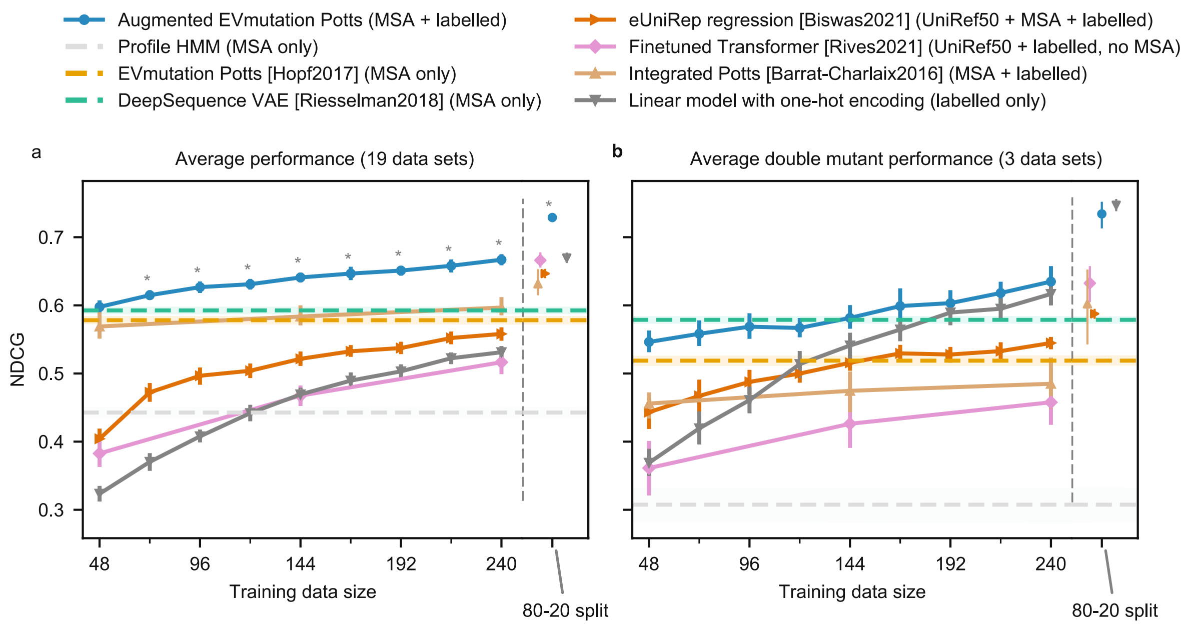
Additional information
Extended data are available for this paper at https://doi.org/10.1038/s41587-021-01146-5.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41587-021-01146-5.

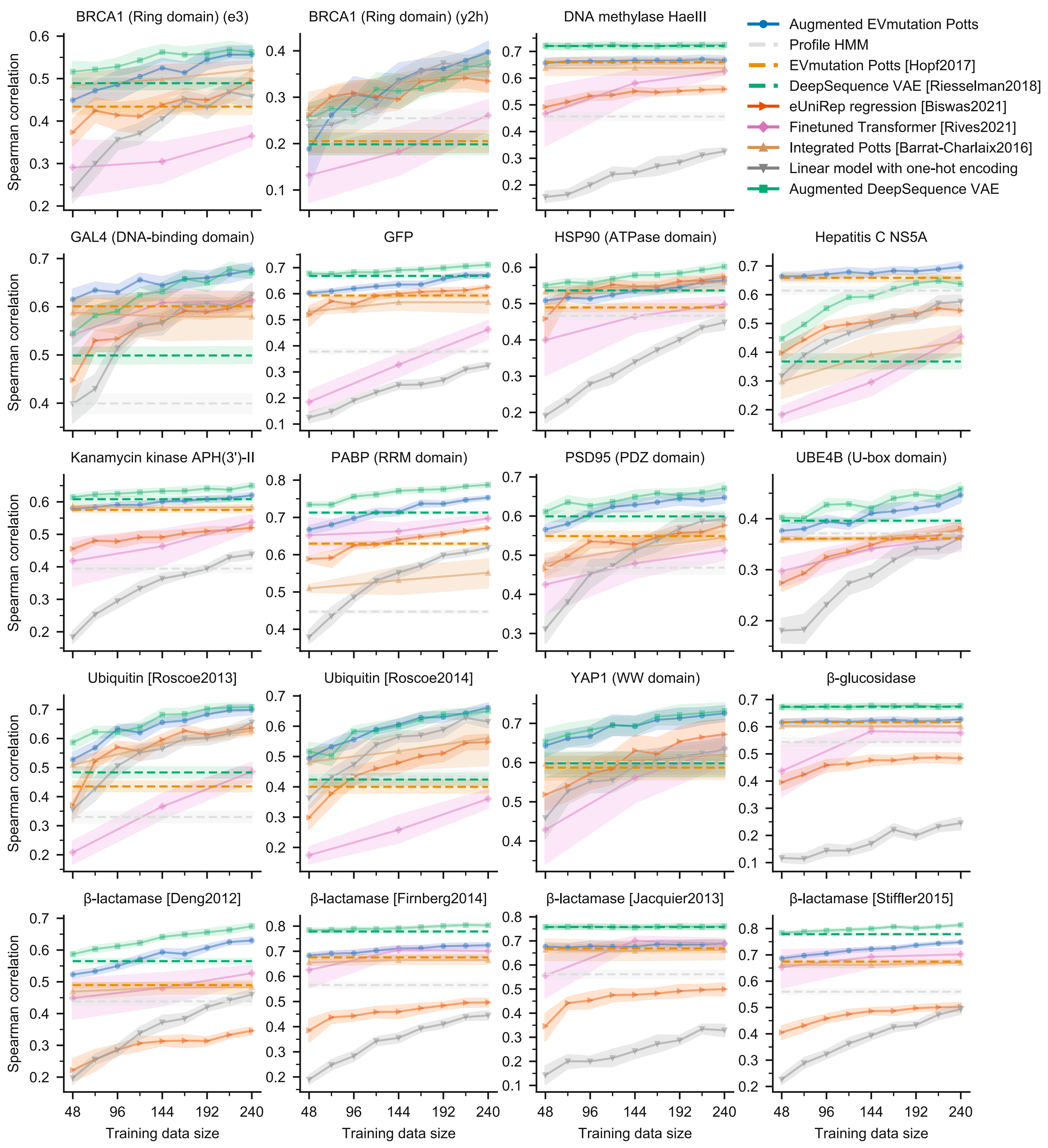
Correspondence and requests for materials should be addressed to Chloe Hsu or Jennifer Listgarten.

Peer review information *Nature Biotechnology* thanks the anonymous reviewers for their contribution to the peer review of this work.

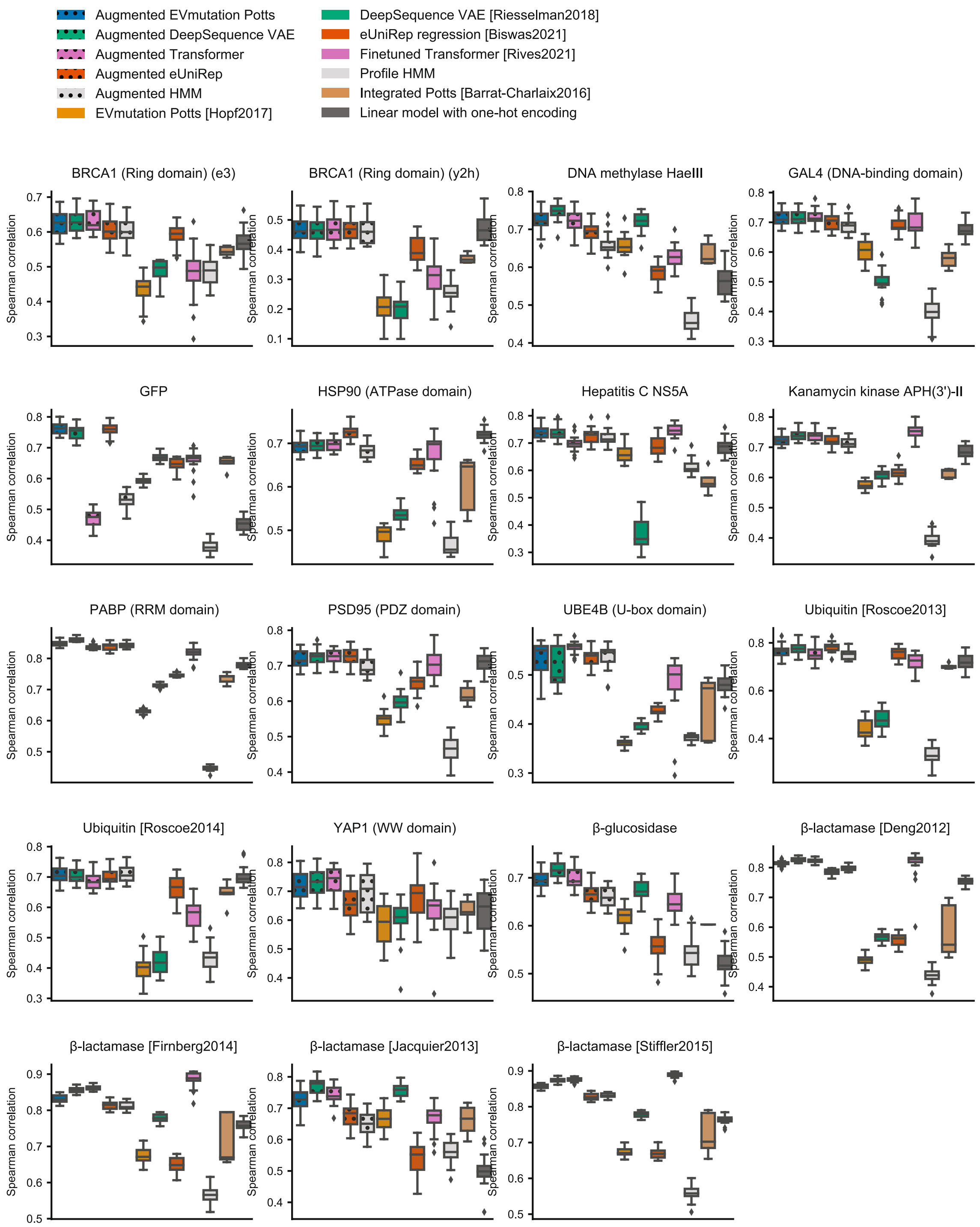
Reprints and permissions information is available at www.nature.com/reprints.



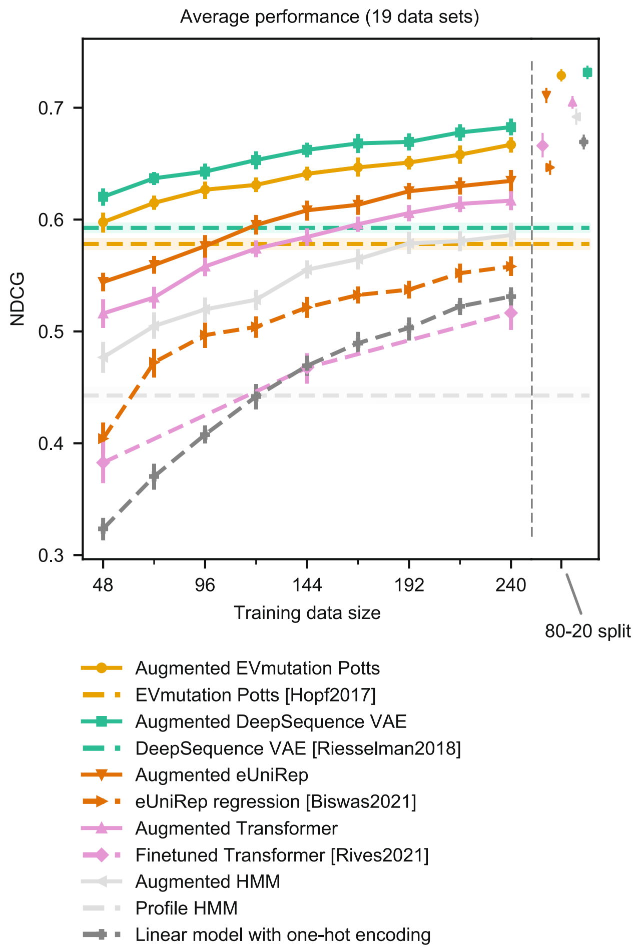
Extended Data Fig. 1 | Performance of existing methods and the augmented Potts model with NDCG. Analog of Fig. 2, but using NDCG instead of Spearman correlation. (a) Average performance across all 19 data sets, as measured by NDCG. The horizontal axis shows the number of supervised training examples used. Error bars are centered at the mean and indicate bootstrapped 95% confidence intervals estimated from 20 random splits of training and test data. Asterisks (*) indicate that $P < 0.01$ among all two-sided Mann-Whitney U tests that the augmented Potts model has different performance from each other method, at a given sample size. In particular, the largest such p-value for each training set size was respectively, $P = 3.9 \times 10^{-2}$, 6.9×10^{-7}, 2.2×10^{-7}, 7.9×10^{-8}, 7.7×10^{-4}, 6.8×10^{-8}, 6.8×10^{-8}, 6.8×10^{-8} and $P = 7.7 \times 10^{-4}$ for the 80-20 split. (b) Average performance across all three data sets containing double mutant sequences (sequences that are two mutations away from the wild-type), and restricted to testing on only double mutants.



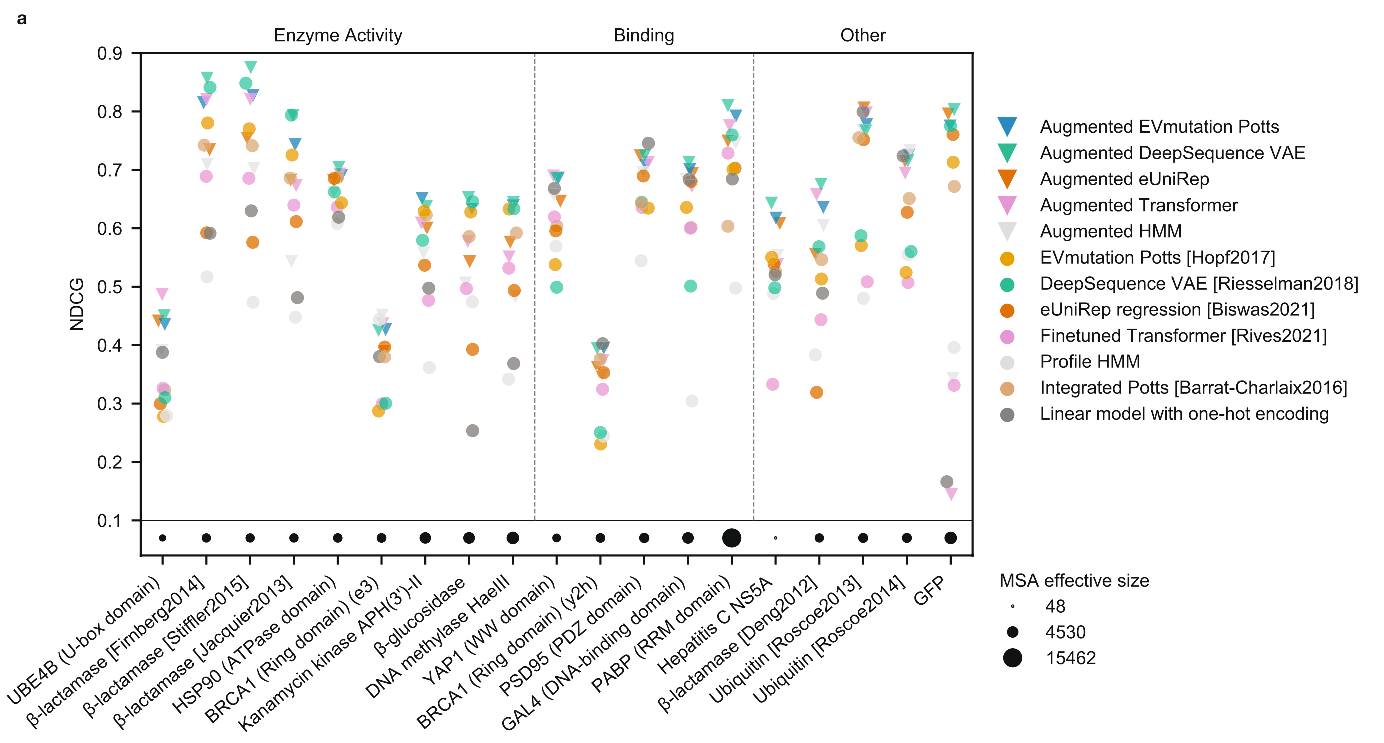
Extended Data Fig. 2 | Performance on individual data sets when trained on limited labeled data. A breakdown of averaged Spearman correlation results presented in Fig. 2a by individual data set. See Supplementary Fig. 1 for the analogous plot using NDCG. Error bands are centered at mean and indicate bootstrapped 95% confidence interval from 20 random data splits.

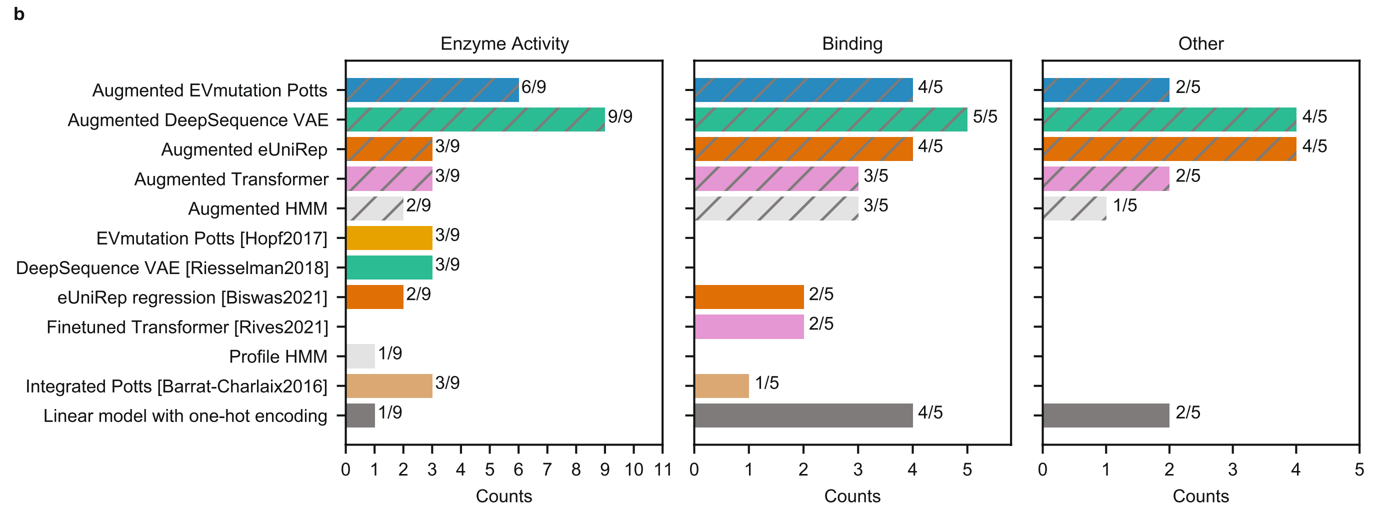


Extended Data Fig. 3 | Performance on individual data sets when trained on 80% data. A breakdown of averaged Spearman correlation results presented in the right-side mini-panel in Fig. 2a, on 80-20 splits, by individual data set. See Supplementary Fig. 2 for the analogous plot using NDCG. Error bars indicate bootstrapped 95% confidence interval from 20 random data splits. Box-and-whisker plots show the first and third quartiles as well as median values. The upper and lower whiskers extend from the hinge to the largest or smallest value no further than 1.5 x interquartile range from the hinge.

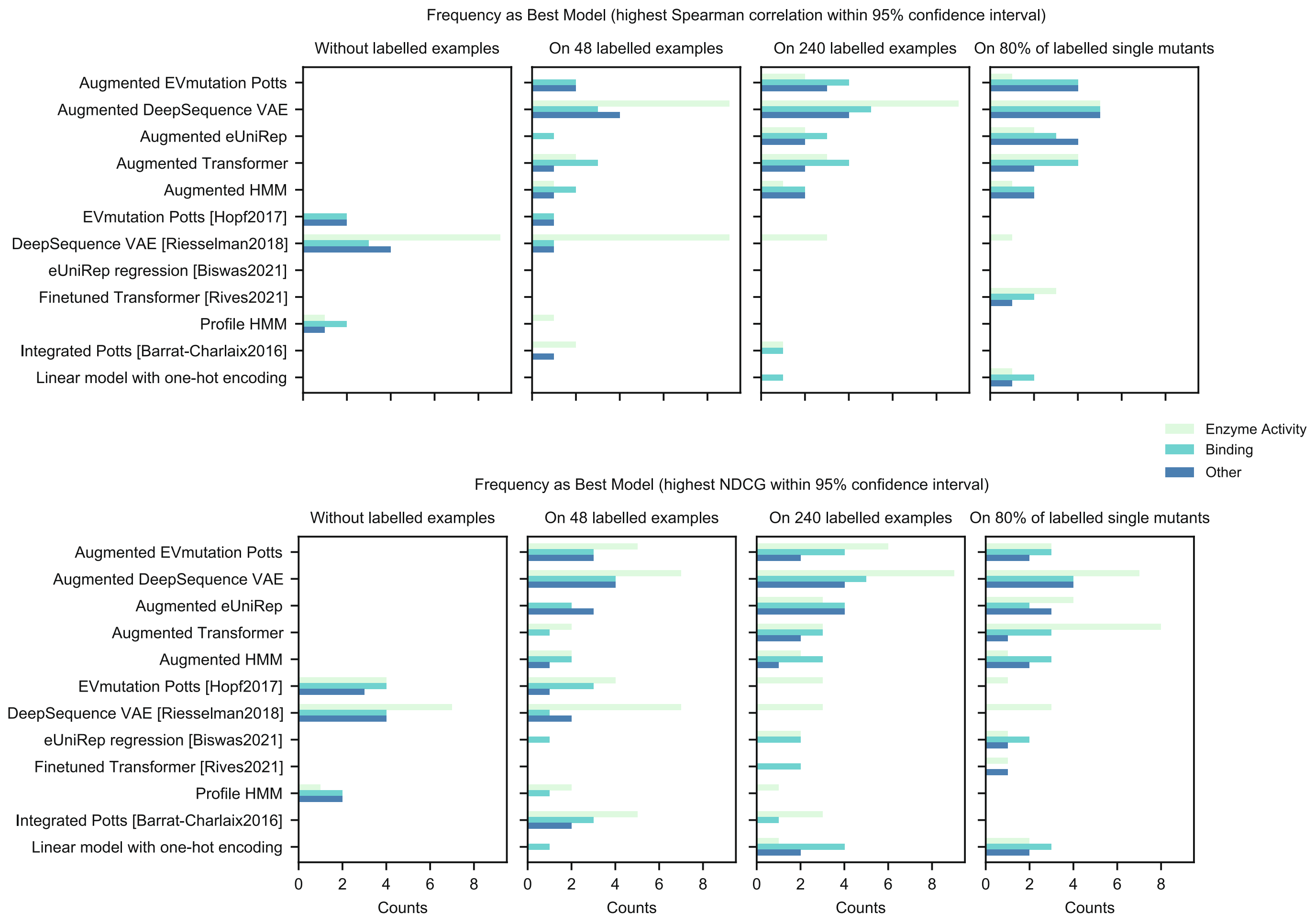


Extended Data Fig. 4 | Augmented approach using different probability density models, with NDCG. Analogous to Fig. 3, but using NDCG. Methods are compared with their augmented counterpart, using matching colors on each pair. Flat, horizontal lines represent evolutionary density models that do not have access to assay-labeled data. Dashed lines indicate existing methods. Error bars are centered at the mean and indicate bootstrapped 95% confidence interval from 20 random data splits.

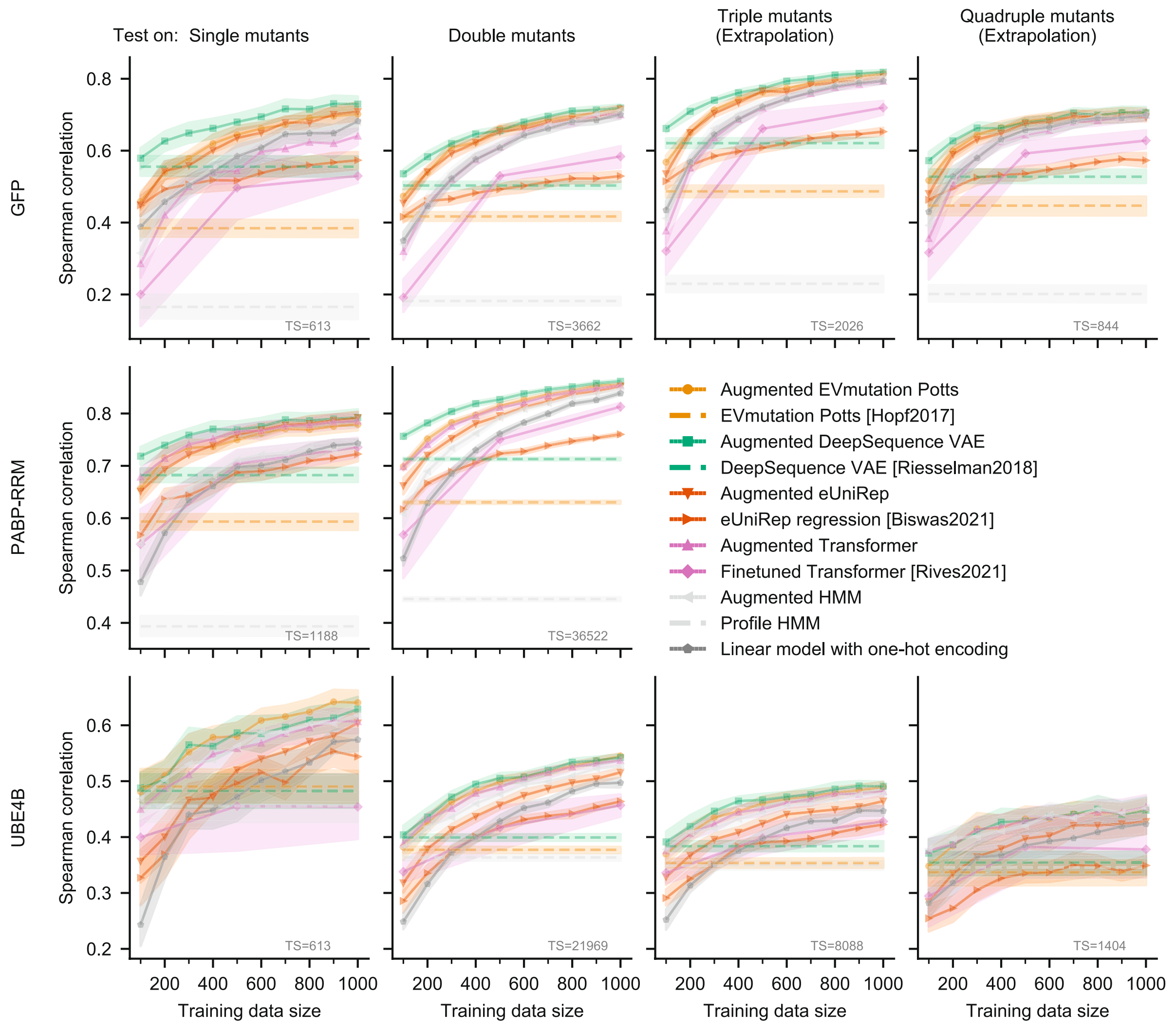




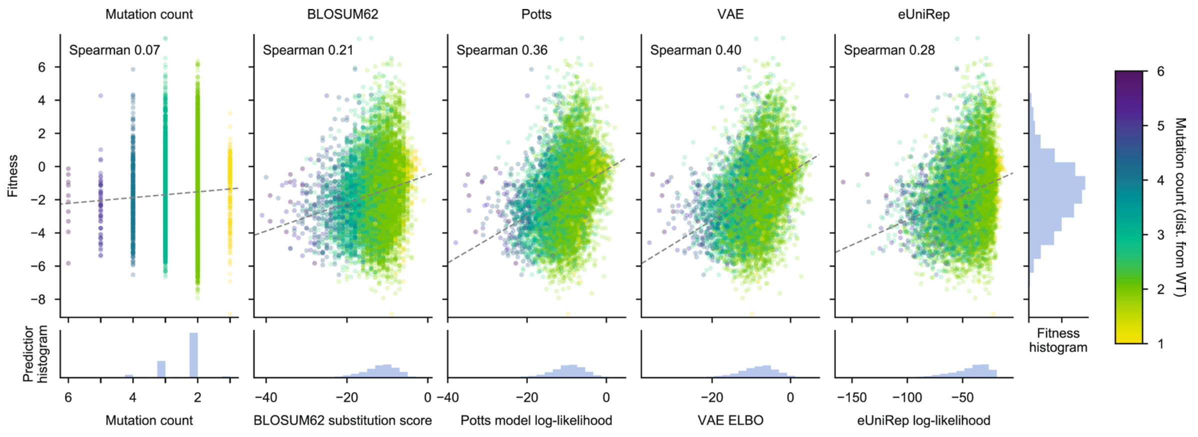
Extended Data Fig. 5 | Performance on individual data sets with NDCG. Analogous to Fig. 4, but using NDCG. (a) Other than the EVmutation Potts model, the DeepSequence VAE, and Profile HMM, none of which use supervised data, all other methods here used 240 labeled training sequences. Each colored dot is the average NDCG from 20 random train-test splits. Random horizontal jitter was added for display purposes. The bottom row of black dots indicates the effective MSA size determined by accounting for sequence similarity with sample reweighting at 80% identity cutoff. (b) Summary of how often each modeling strategy had maximal NDCG. Such modelling strategies were determined by first identifying the top-performing strategy for any given scenario, and then also identifying any other strategy that came within the 95% confidence interval of the top performer.



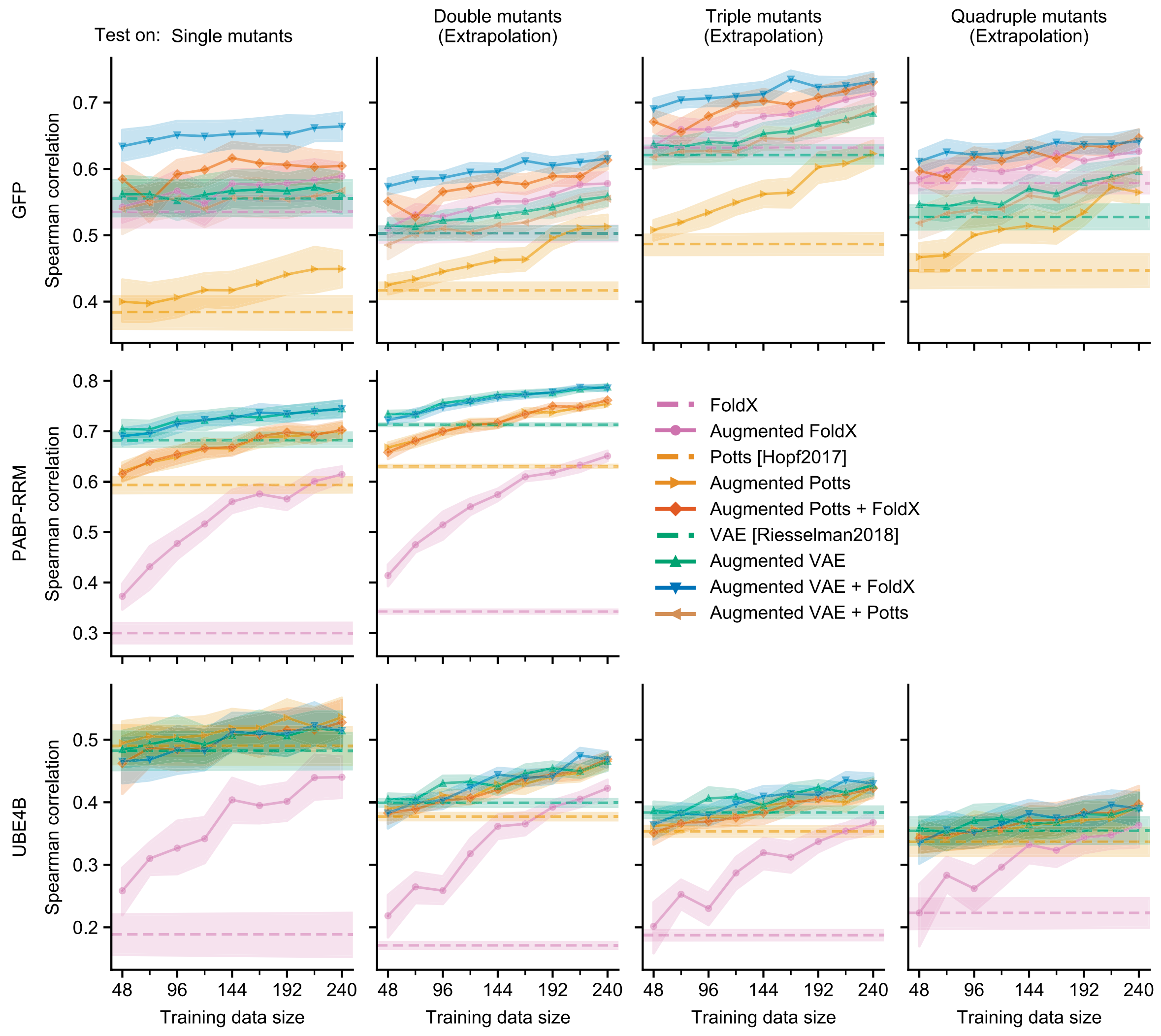
Extended Data Fig. 6 | The distribution of best model(s) on each data set. Analogous to Fig. 4b, but varying the number of assay-labeled training examples. (a) Summary of how often each modelling strategy had maximal Spearman correlation. Such modelling strategies were determined by first identifying the top-performing strategy for any given scenario, and then also identifying any other strategy that came within the 95% confidence interval of the top performer. Four settings are used: with no assay-labeled data, when training on 48 or 240 assay-labeled single-mutant examples, and in the 80-20 train-test split setting. (b) Summary of how often each modelling strategy had maximal NDCG.



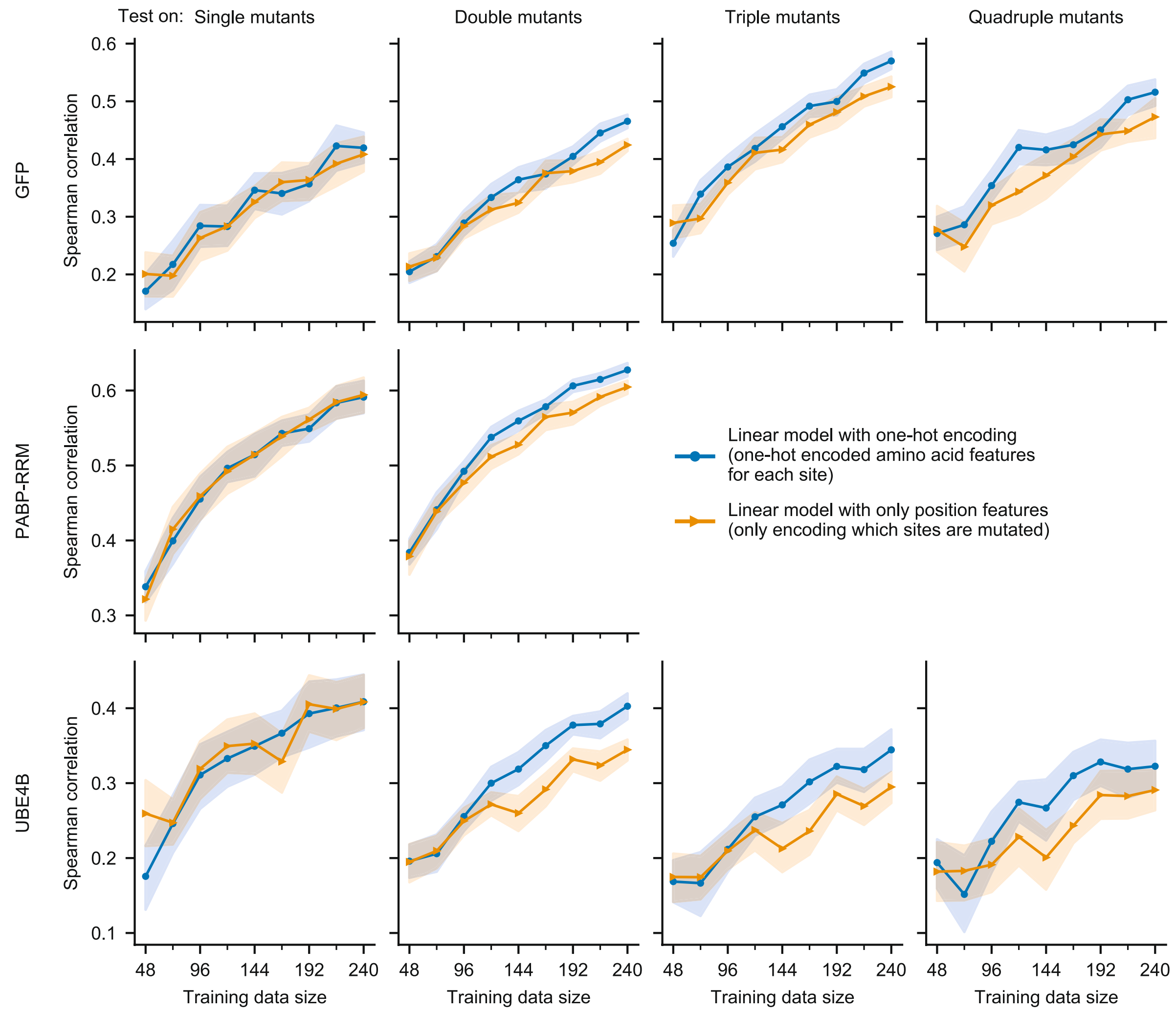
Extended Data Fig. 7 | Extrapolation performance from single and double mutants to higher-order mutants. Analogous to Fig. 5, but training on a random sample from both single and double mutants. Each column shows the performance when training on randomly sampled single mutants and then separately testing on single, double, or triple mutants, none of which were in the training data. The total size (TS) indicates the total number of mutants of a particular order in all of the data. For example, 'TS=613' for single mutants means there were 613 total single mutants in the data set that we sampled from. Error bars are centered at the mean and indicate bootstrapped 95% confidence interval from 20 random data splits. See Supplementary Fig. 6 for analogous plot using NDCG.



Extended Data Fig. 8 | Edit distance from wild-type sequence as a predictive model (UBE4B U-box domain). Analogous to Fig. 6, but on the UBE4B U-box domain data set. We compared the performance of non-augmented evolutionary density models to two predictive models that use only the edit distance of a sequence to the wild type. In one version, the edit distance is defined as the number of mutations away from the wild type. In the other version, we used BLOSUM62 to compute the distance from wild type, which thus accounts not only for the number of mutations, but also the type of mutation. Each dot represents a UBE4B U-box domain sequence, with darker colors indicating larger distances from the wild-type.



Extended Data Fig. 9 | FoldX predictions as additional features in augmented models. Each column shows the performance of augmented models with a single FoldX-derived stability feature added, when training on randomly sampled single mutants and then separately testing on single, double, or triple mutants. It also shows augmentation of two density models at the same time, without FoldX, as in "Augmented VAE + Potts". Error bars are centered at the mean and indicate bootstrapped 95% confidence interval from 20 random data splits. See Supplementary Fig. 7 for analogous evaluation with NDCG.



Extended Data Fig. 10 | Performance of linear model using only one feature per site (not per amino acid at each site). In addition to the linear model with one-hot encoded, site-specific amino acid features, we also evaluated a simpler linear model with position-only features that encode which sites are mutated. The evaluation uses Spearman correlation. Each column shows the performance when training on randomly sampled single mutants and then separately testing on single, double, or triple mutants, none of which were in the training data. Error bars are centered at the mean and indicate bootstrapped 95% confidence interval from 20 random data splits.

nature research

Corresponding author(s): Jennifer Listgarten

Last updated by author(s): Oct 26, 2021

Reporting Summary

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☐	☒	A description of all covariates tested
☐	☒	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☐	☒	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Open source code used: HMMER 3.3.1. The custom code for MSA generation is available at https://github.com/chloechsu/combining-evolutionary-and-assay-labelled-data.
Data analysis	Open source code: plmc (May 2018); scikit-learn (0.23.1); theano (1.0.5); PyTorch (1.4.0); tensorflow (2.2.0); esm (0.3.1); UniRep (March 2020); DeepSequence (Jul 2018); FoldX 5.0; The custom code for data analysis is available at https://github.com/chloechsu/combining-evolutionary-and-assay-labelled-data.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

All protein fitness data were publicly available through citations available in the paper. A processed version of these data and our evaluation results are available on Dryad with doi:10.6078/D1K71B. All protein structures used in the study are available publicly with PDB IDs 2WUR, 6R5K, and 2KR4.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	The size of each evaluated data set is based on the available data from existing literature.
Data exclusions	Following the convention in EVmutation and DeepSequence, we exclude sequences with mutations at positions that have more than 30% gaps in MSAs, to focus on regions with sufficient evolutionary data.
Replication	For each evaluation, we repeat the evaluation with 20 different random seeds for uncertainty quantification. The results are qualitatively similar from the different random seeds.
Randomization	For each data set, we randomly sample 20% of the data set as held-out test data. Among the remaining 80% data, we randomly sample N = 24, 48, 72, 96, ... or 240 single mutant sequences as training data in separate experiments, or use all single mutant sequences in the 80% training data in the 80-20 split experiments. Samples were allocated into the training data and the test data randomly.
Blinding	We performed our analysis with automated scripts and therefore were blinded to the data group allocation.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Human research participants
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging