Predicting the Effect of Mutations on a Genome-Wide Scale

by

Alexey Strokach

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Predicting the Effect of Mutations on a Genome-Wide Scale

Alexey Strokach
Master of Science
Graduate Department of Computer Science
University of Toronto
2016

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Recent advances in DNA sequence technology have drastically lowered the cost and improved the accuracy of genome sequencing [1]. This has made exome and whole-genome sequencing a viable and cost-effective tool in both the laboratory and in the clinic to assist with the diagnosis and direct treatment of pediatric conditions [2] and cancers [3], and has led to an enormous growth in the amount of genomic data that is being generated. However, interpreting such genomic data to produce meaningful and actionable results remains a challenge.

In vitro and *in vivo* experiments remain the gold standard in elucidating the effect of mutations. However, evaluating experimentally the effect of all discovered mutants is not feasible. Computational techniques have been developed to predict the effect of different mutations and to prioritize them for experimental validation. Those techniques generally use conservation score describing the likelihood that a particular amino acid being found in the particular position in orthologous proteins.

The most widely-used program for predicting the deleteriousness of a mutation is Sorting Intolerant from Tolerant (SIFT) [4]. SIFT runs PSI-Blast to create a multiple sequence alignment for the query protein, and computes a conservation score by looking at the likelihood of the wildtype and mutant amino acids occurring at a given position in the alignment. While SIFT is a well-established tool in the field, it is difficult to compile and install on a local machine. Furthermore, multiple sequence alignments constructed by SIFT can be several megabytes in size, and caching this data for an entire proteome would require a non-trivial amount of storage space.

Another popular sequence-based algorithm is Provean [5]. Provean also creates a multiple sequence alignment for the query protein. However, instead of using the entire alignment, Provean runs CD-HIT to select under 50 representative sequences, referred to as the “supporting set”, which capture the diversity of the alignment. The supporting set for a particular protein can be precalculated and stored for future use. If a supporting set is available, calculating the Provean score takes several seconds per mutation. Provean is reported to achieve similar performance to SIFT [5]. However, unlike SIFT, it is freely available under the GPLv3 license, it compiles easily and it runs on all modern Linux distributions.

Many other tools have been developed that offer various advantages over SIFT / Provean. PolyPhen-2 [6] uses support vector machines to combine a conservation score with different sequential and structural features of the wildtype and mutant residue. However, since PolyPhen-2 is trained on a dataset of human deleterious mutations, it is difficult to use in downstream applications, as one would have to make sure to exclude the PolyPhen-2 training set throughout the training and validation process. FATHMM [7] constructs a hidden Markov model based on the alignment, and is reported to achieve marginally higher accuracy than SIFT / Provean. Other techniques offering various advantages over SIFT / Provean include MutPred [8], MutationAssesor [9], CADD [10], CONDEL [5], and others. Each of those tools uses

Despite the proliferation of tools predicting the deleteriousness of different SNPs, those tools remain limited in their accuracy and the type of information that they can provide. While millions of single nucleotide polymorphisms (SNPs) have been implicated in thousands of diseases, approaches for predicting the phenotypic effect of newly-discovered mutations are still in their infancy. One of the reasons is that while sequence-based tools achieve reasonably good performance at predicting whether or not a given mutation is going to be deleterious, they fall short in predicting *why* that mutation is deleterious. This lack of actionable predictions limits the usability of the vast DNA sequencing data that has been generated. However, the etiology by which the mutations cause or contribute to a disease are often unknown.

1.2 Predicting the structural effect of mutations

One reason for out lack of ability in interpreting is the focus on the sequence-level features, while in the majority of missense mutations, it is the alteration in the function of the transcribed protein which is responsible for the detrimental effect of mutations.

The field of protein science has generally been concerned with the broad questions of protein folding, protein design. Algorithms have been developed to predict the effect of mutations on protein folding and protein-protein affinity, but those tools are generally meant to be used on a case-by-case basis and have not been designed to be applied on a genome-wide scale to predict the effect of missense mutations from whole-genome sequencing studies.

While the growth in protein crystal structures has not seen the rapid rise that was observed in DNA sequencing, the number of resolved protein structures has also been increasing, with the Protein Data Bank (PDB) containing close to 125,000 structures as of 2016.

The most accurate class of computational techniques are alchemical free energy calculations, which involve modelling the structural transition from the wildtype to the mutant protein and using the Bennett acceptance ratio (BAR) or thermodynamic integration (TI) to calculate the energetics of the transition [11]. However, alchemical free energy calculations are computationally expansive, and are generally used only in cases where the experimental characterization of mutants is particularly difficult, as in the case of D-amino acid peptide design [12].

Many algorithms have been developed which attempt to predict the effect of mutations on protein stability and / or on protein-protein interaction affinity. Those techniques generally use a rigid backbone representation of protein and use statistical potentials. For a review see XXX.

Mixed strategies which utilize both sequence- and structure-based approaches. Such algorithms include PoPMuSiC,

Structure-based tools which predict the effect of mutations on protein structure and / or function using features describing the three-dimensional structure of the protein. mCSM [13] (graph-based signatures), MAESTRO [14] (multi-agent machine learning), CC/PBSA (Concoord/Poisson-Boltzmann surface area) [15],

1.3 Goals and objectives

- Evaluate how well we can predict the deleteriousness of a mutation by measuring the effect of protein folding on protein stability.
- Assessing the impact of missense mutations.
- Protein engineering. For example generating biological therapeutics that are more thermostable and have a higher affinity for their target.
- Basic science: characterizing the forces that are most important in protein folding and binding, and the effect of mutations on those forces.
- In this work we examine how much sequence-based features can aid in the prediction of traditionally structural realms such as the prediction of $\Delta\Delta G$ scores of mutations, and how much structure-based features can aid with the prediction of mutation pathogenicity–a traditionally sequence based

1.4 Acknowledgements

This is a continuation of the work performed by Niklas Berliner *et al.* [16]. In 2 we discuss how we expand ELASPIC to work on the genome-wide scale. In 3 we discuss how we retrained ELASPIC while leveraging the information we extracted from genome-wide analysis.

Chapter 2

Implementation

2.1 Profs

ELASPIC uses a domain-based approach for creating homology models of query proteins, and therefore requires access to accurate domain definitions. The most widely-used source of protein domain definitions is Pfam [17]. However, since Pfam domains definitions are based entirely on protein sequence, they correlate poorly with the structural fold of the protein. Using Pfam domain definitions when making homology models tends to produce unstable models of fragmented and / or truncated domains, and this would compromise our subsequent analysis of the structural impact of mutations.

In order to improve the structural accuracy of Pfam domains, Andres Felipe Giraldo Forero developed a pipeline that uses structural alignments and a set of heuristics to modify Pfam domain definitions and make them better aligned with the tertiary structure of the protein, as defined by CATH [18]. He named this pipeline Profs, for Protein families. A schematic of this pipeline is presented in Figure 2.1, and an R package implementing the pipeline is available at `https://bitbucket.org/afgiraldofo/profs`. Profs domains have an advantage over Pfam domains in that they have been corrected and expanded to match the structural fold of the protein. They have an advantage over CATH domains in that they are backed by large, manually-seeded alignments, and can be easily detected in any protein sequence using Pfam HMMs.

We used Andres' pipeline to annotate with Profs domains all proteins in the UniProt database. The resulting table of Profs domain definitions is available for download from the ELASPIC website (`http://elaspic.kimlab.org/static/download/`) and is included in the ELASPIC database (see **domain** and **domain_contact** tables in Figure 2.6 and Table 2.1). The following sections describe the procedure used to generate tables of Profs domain definitions and Profs domain-domain interactions that are used by ELASPIC.

2.1.1 Domains

We used Profs domain definitions, which had been calculated for all proteins in the PDB, to find Profs domains, and structural templates for those domains, for all proteins in UniProt (step 6 in Figure 2.1). To do this, we followed the same procedure that was used by the authors of Profs to annotate structures in the PDB that have Pfam domains but no CATH domains [19].

We started with Pfam domain definitions for all known protein sequences, which we download from

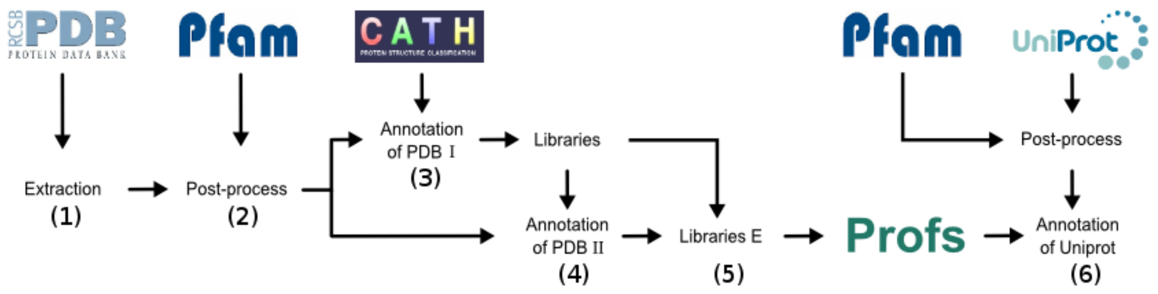


Figure 2.1: Flowchart illustrating steps in the Profs pipeline (courtesy of Andres Felipe Giraldo Forero). **(1)** All structures in the PDB are parsed to extract protein sequences, and *hmmscan* is ran to find Pfam domains in those sequences. **(2)** Pfam domains of proteins in the PDB are processed in order to join and / or remove overlapping and repeating domains. **(3)** Pfam domain definitions are altered in order to make them compatible with CATH definitions, for structures that have been annotated by CATH. **(4)** Pfam domain definitions are altered in order to make them compatible with CATH definitions, for structures that have not been annotated by CATH. This is done by performing pairwise alignments with structures that do have CATH annotations. **(5)** Libraries of Profs domain definitions, and Profs domain-domain interactions, are generated for all proteins in the PDB. **(6)** Libraries of Profs domain definitions, and Profs domain-domain interactions, are generated for all proteins in Uniprot.

the SIMAP website [20]. We mapped those protein sequences to Uniprot using the MD5 hash of each sequence, and we joined or removed overlapping and repeating domains using a mapping table supplied with the Profs R package. Next, we tried to find a Profs structural template for each Pfam domain by running *blastp* against libraries of Profs domains, which are included in the Profs R package. If a suitable template was found, we proceeded to do iterative global alignments using Muscle [21] while expanding domain boundaries of the Pfam domains to match domain boundaries of the Profs templates. If two Pfam domains were expanded to occupy the same region in the protein, that region was divided in half and attached to the preceding and the succeeding domains.

The results of this analysis are stored in the **uniprot_domain** and the **uniprot_domain_template** tables in the ELASPIC database (Figure 2.6). The **uniprot_domain** table contains all Pfam domains and supradomains that are obtained after removing repeating and overlapping domains, as outlined above. The *pdbfam_name* column contains the name of the Profs domain. The *alignment_def* column contains either the original Pfam domain definitions or, in the case of supradomains, the merged domain definitions of multiple Pfam domains. The **uniprot_domain_template** table contains information describing the alignment of the Pfam domain or supradomain with the corresponding Profs structural template, for domains for which a suitable Profs template could be found. The *cath_id* column identifies the Profs structural template that was selected, and the *domain_def* column contains the corrected and expanded domain definitions.

2.1.2 Comparison with Pfam and Gene3D

In order to ascertain the validity of Profs domain definitions, we compared Profs, Pfam and Gene3D in terms of sequence coverage (Figure 2.2) and domain size (Figure 2.3).

We downloaded Pfam and Gene3D domain definitions for all human proteins from SIMAP [20], and we calculated Profs domain definitions following the pipeline described above. The analysis was

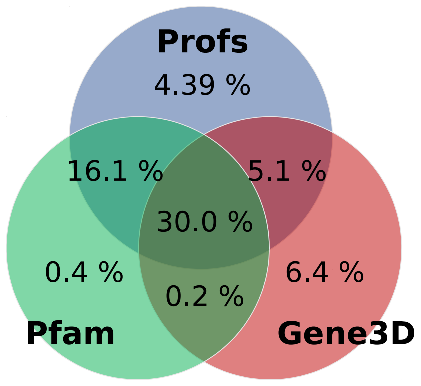


Figure 2.2: Venn diagram showing the overlap in domain definitions between Profs, Pfam, and Gene3D. Values represent the fraction of amino acids, of all human proteins in UniProt, which are covered by the particular domain or domains. A total of 18,828 human proteins and 10,868,810 amino acids were considered, after excluding proteins which had no predicted domains by any method. Profs has the highest coverage, with 55.7 % of amino acids being annotated by a Prof domain.

restricted to 18,828 human proteins from UniProt which are annotated with at least one Profs, Pfam or Gene3D domain.

In order to compare sequence coverage, we looked at the fraction of all protein sequences which are covered by each domain type (Figure 2.2). Overall, Profs has the highest sequence coverage, with 55.7 % of 10,868,810 amino acids in 18,828 proteins residing inside a Profs domain. Profs annotates $\approx$ 9% more amino acids than Pfam and $\approx$ 14% more amino acids than Gene3D, although the relatively low coverage by Gene3D is expected, as it can only detect domains which are represented in the PDB.

In order to compare domain size, we looked at the average number of domains per protein for each of the three methods (Figure 2.3). Profs has more proteins with only one domain per protein, while Pfam and Gene3D have more proteins with two or more domains per protein. This is consistent with Profs trying to join fragmented and repeating domains into consistent structural units. Gene3D does not detect domains in many proteins with Profs and Pfam domains, likely because those domains have not been crystallized.

The result of this analysis shows that, at least for human proteins, Profs achieves higher sequence coverage using fewer domains per protein than either Pfam or Gene3D. This makes Profs well-suited for the ELASPIC pipeline.

2.1.3 Domain interactions

We also created a table of domain-domain interactions for proteins that are known to interact and for which a homology model of the interaction can be created. We started by creating a comprehensive list of protein-protein interactions (PPIs), by taking the union of all PPIs listed in the HIPPIE database [22] and in the datasets hosted by the Harvard Center for Cancer Systems Biology (CCSB) [23]. The overlap in the PPIs obtained from each source is presented in Figure 2.4. We filtered those PPIs to select pairs of proteins where each protein has at least one domain with a structural template. This information is stored in the **uniprot_domain_pair** table. For each of those domains, we perform a Blast search

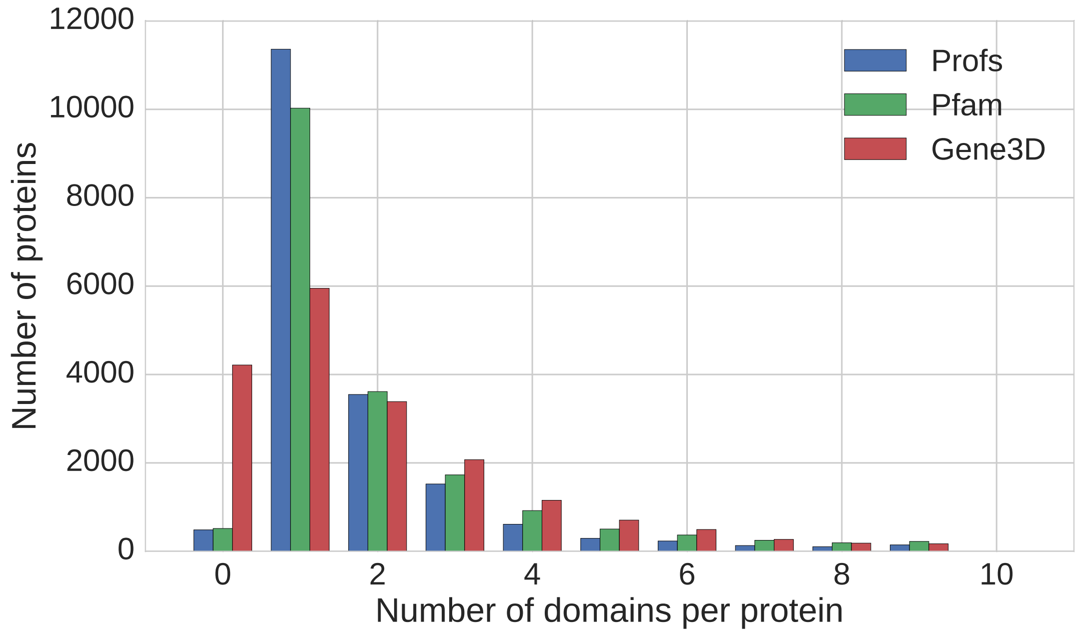


Figure 2.3: Average number of Profs, Pfam and Gene3D domains per protein, for all human proteins containing at least one domains. Profs tends to have fewer domains per protein then either Pfam or Gene3D, even though Profs domains have higher sequence coverage (see Figure 2.2). Gene3D lacks domain annotation for many proteins which contain at least one Pfam and Profs domain.

of the domain sequence against a library of Profs domains in the PDB (the **domain** table in Figure 2.6), and we selected only those templates that occur in the same crystal structure in both proteins and that interact according to the **domain_contact** table. In order to select the best template for the interaction, we calculate a quality score for each of the two domains using Equation 2.1, and chose the template with the highest geometric mean of the two scores (Equation 2.2).

$$alignment\_score = 0.95 \cdot seq\_identity \cdot coverage + 0.05 \cdot coverage \tag{2.1}$$

$$combined\_alignment\_score = \sqrt{alignment\_score\_1 \cdot alignment\_score\_2} \tag{2.2}$$

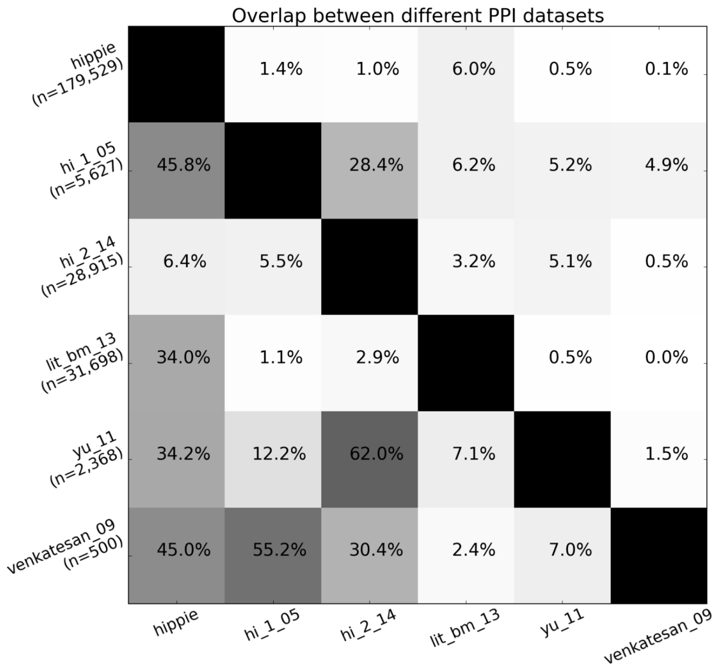


Figure 2.4: Overlap in protein-protein interaction (PPI) databases. The shade and value of each square denotes the percentage of PPIs in the database named on the y-axis that are also found in the database named on the x-axis. **hippie** is a meta-database, which integrates PPIs from many different sources [22]. **hi_1_05** contains PPIs discovered through a proteome-wide yeast two-hybrid experiment conducted by Rual *et al.* [24]. **hi_2_14** contains PPIs discovered through a proteome-wide yeast two-hybrid experiment conducted by Rolland *et al.* [23]. **lit_bm_13** contains PPIs obtained from the literature and supported by multiple pieces of evidence [23]. **yu_11** contains PPIs obtained using "stitch-seq", which combines PCR stitching with next-generation sequencing [25]. **venkatesan_09** corresponds to high-quality binary interactions found in repeat yeast two-hybrid assays conducted by Venkatesan *et al.* [26].
The **hippie** database was downloaded from the Hippie website: `http://cbdm-01.zdv.uni-mainz.de/~mschaefer/hippie/`. All other datasets were downloaded from the Harvard Center for Cancer Systems Biology: `http://interactome.dfci.harvard.edu/H_sapiens/`.

2.2 ELASPIC

The ELASPIC project was started by Niklas Berliner and others in 2014 [16]. ELASPIC uses Modeller [27] to construct homology models of domains and domain-domain interactions, FoldX to optimize those model and to introduce mutations [28], and the gradient boosting regressor algorithm [16], implemented in scikit-learn [29], to combine FoldX energy scores with sequence-based and other features and predict the energetic impact of a mutation on the stability of a single domain or the affinity between two domains.

Since the original publication, ELASPIC was updated in many ways in order to make it easy to configure and run on different systems, and to make it possible to run ELASPIC on a distributed cluster as a backend to the ELASPIC webserver. Instead of using SIFT, as discussed in the original publication, we switched to using Provean, as it proved to be much easier to configure and install, and gave an option of precalculating supporting sets which could be used to calculate the Provean score for other mutations in the same protein in seconds. We switched to using MSMS instead of NACCESS for calculating solvent-accessible surface area of residues because MSMS was able to handle better misformed PDB files. We created two libraries for running selaspic

An overview of the updated ELASPIC pipeline is presented in Figure 2.5. ELASPIC includes a Python library for construction sequence alignments, constructing Provean supporting sets and computing the Provean score, constructing homology models, running FoldX, and predicting the $\Delta\Delta G$ of the mutation. It also includes a “Standalone Pipeline” (Figure 2.5 right) and a “Database Pipeline” (Figure 2.5 left), which wrap the ELASPIC library and provide command line options for mutating local PDB files and proteins in the ELASPIC database.

The source code for the ELASPIC pipeline is hosted on GitHub (`https://github.com/kimlaborg/elaspic`), ELASPIC documentation is hosted on ReadtheDocs (`http://elaspic.readthedocs.io/`), and precalculated data can be downloaded from the ELASPIC website (`http://elaspic.kimlab.org/static/download/`).

2.2.1 Standalone pipeline

The standalone pipeline works without downloading and installing a local copy of the ELASPIC and PDB databases, but requires a PDB structure to be provided for every mutation. The output of the pipeline is saved as JSON files inside the *.elaspic* subfolder created in the working directory. The general overview of the local pipleine is presented on the right side of Figure 2.5.

2.2.2 Database pipeline

The database pipeline allows mutations to be performed on a proteome-wide scale, without having to specify a structural template for each protein. This pipeline requires a local installation of a relational database containing ELASPIC domain definitions and templates, as well as a local copy of the BLAST and PDB databases.

The general overview of the database pipleine is presented on the left side of Figure 2.5. A user runs the ELASPIC pipeline specifying the Uniprot ID of the protein being mutated, and one or more mutations affecting that protein. At each decision node, the pipeline queries the database to check whether or not the required information has been previously calculated. If the required data has not been calculated, the pipeline calculates it on the fly and stores the results in the database for later

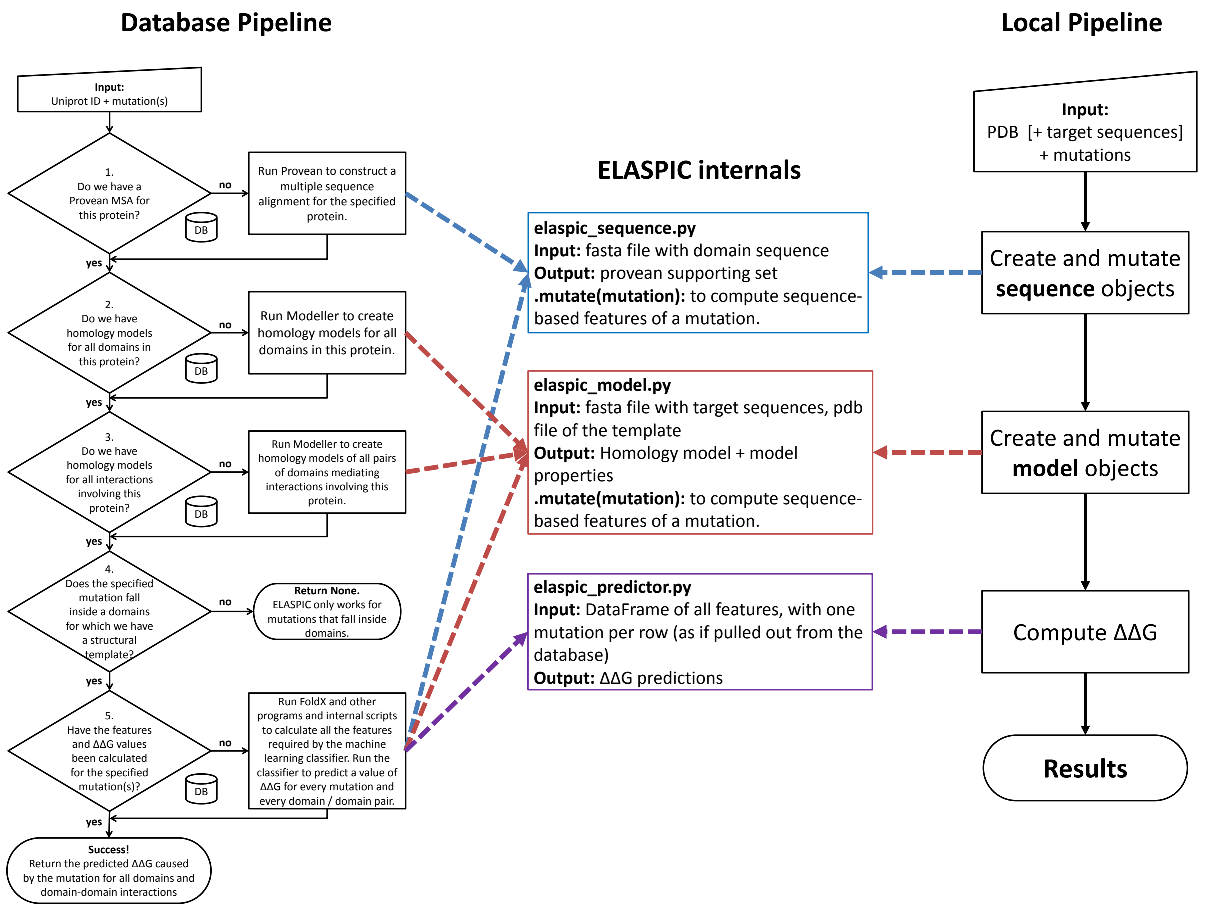

Database Pipeline
Local Pipeline
ELASPIC internals
Input:
Uniprot ID + mutation(s)
1. Do we have a Provean MSA for this protein?
no
Run Provean to construct a multiple sequence alignment for the specified protein.
yes
2. Do we have homology models for all domains in this protein?
Run Modeller to create homology models for all domains in this protein.
3. Do we have homology models for all interactions involving this protein?
Run Modeller to create homology models of all pairs of domains mediating interactions involving this protein.
4. Does the specified mutation fall inside a domains for which we have a structural template?
Return None. ELASPIC only works for mutations that fall inside domains.
5. Have the features and ΔΔG values been calculated for the specified mutation(s)?
Run FoldX and other programs and internal scripts to calculate all the features required by the machine learning classifier. Run the classifier to predict a value of ΔΔG for every mutation and every domain / domain pair.
Success! Return the predicted ΔΔG caused by the mutation for all domains and domain-domain interactions
DB
elaspic_sequence.py
Input: fasta file with domain sequence
Output: provean supporting set
.mutate(mutation): to compute sequence-based features of a mutation.
elaspic_model.py
Input: fasta file with target sequences, pdb file of the template
Output: Homology model + model properties
.mutate(mutation): to compute sequence-based features of a mutation.
elaspic_predictor.py
Input: DataFrame of all features, with one mutation per row (as if pulled out from the database)
Output: ΔΔG predictions
Input: PDB [+ target sequences] + mutations
Create and mutate sequence objects
Create and mutate model objects
Compute ΔΔG
Results

Figure 2.5: Overview of the ELASPIC pipeline. **Database Pipeline:** A user runs the ELASPIC pipeline by specifying the UniProt identifier of the protein being mutated, and one or more mutations affecting that protein. At each decision node, the pipeline queries the database (Figure 2.6) to check whether or not the required information has been calculated previously. If the required data has not been calculated, the pipeline calculates it on the fly and stores the results in the database for later retrieval. The pipeline proceeds until homology models of all domains in the protein, and all domain-domain interactions involving the protein, have been calculated, and the $\Delta\Delta G$ has been predicted for every specified mutation. **Local Pipeline:** A user runs the ELASPIC pipeline by specifying a PDB file with the structure of the protein that they wish to mutate and one or more mutations, or by specifying a FASTA file with the sequence of the protein that they wish to mutate, a PDB file with the structural template to be used for homology modelling and one or more mutations. ELASPIC runs Provean to calculate the supporting set, runs MODELLER to make the homology model, and runs FoldX to compute structural features describing the wildtype and mutant residues. Results are stored in a local *.elaspic* folder and are not recalculated if the user decides to run more mutations.

retrieval. The pipeline proceeds until homology models of all domains in the protein, and all domain-domain interactions involving the protein, have been calculated, and the $\Delta\Delta G$ has been predicted for every specified mutation.

Results of the database pipeline are store in the ELASPIC database. An overview of the ELASPIC database schema is presented in Figure 2.6, and a description of each database table is provided in Table 2.1.

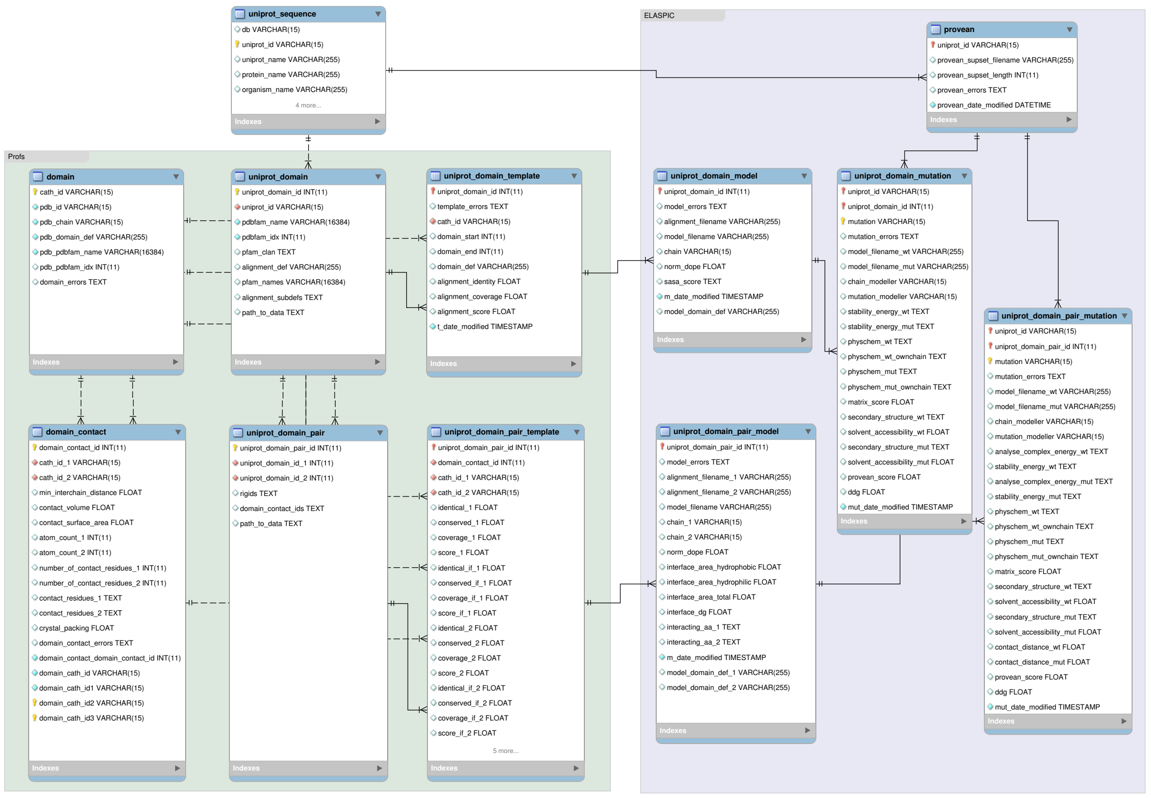

uniprot_sequence
db VARCHAR(15)
uniprot_id VARCHAR(15)
uniprot_name VARCHAR(255)
protein_name VARCHAR(255)
organism_name VARCHAR(255)
4 more...
Indexes
Profs
domain
cath_id VARCHAR(15)
pdb_id VARCHAR(15)
pdb_chain VARCHAR(15)
pdb_domain_def VARCHAR(255)
pdb_pdbfam_name VARCHAR(16384)
pdb_pdbfam_idx INT(11)
domain_errors TEXT
Indexes
uniprot_domain
uniprot_domain_id INT(11)
uniprot_id VARCHAR(15)
pdbfam_name VARCHAR(16384)
pdbfam_idx INT(11)
pfam_clan TEXT
alignment_def VARCHAR(255)
pfam_names VARCHAR(16384)
alignment_subdefs TEXT
path_to_data TEXT
Indexes
uniprot_domain_template
uniprot_domain_id INT(11)
template_errors TEXT
cath_id VARCHAR(15)
domain_start INT(11)
domain_end INT(11)
domain_def VARCHAR(255)
alignment_identity FLOAT
alignment_coverage FLOAT
alignment_score FLOAT
t_date_modified TIMESTAMP
Indexes
domain_contact
domain_contact_id INT(11)
cath_id_1 VARCHAR(15)
cath_id_2 VARCHAR(15)
min_interchain_distance FLOAT
contact_volume FLOAT
contact_surface_area FLOAT
atom_count_1 INT(11)
atom_count_2 INT(11)
number_of_contact_residues_1 INT(11)
number_of_contact_residues_2 INT(11)
contact_residues_1 TEXT
contact_residues_2 TEXT
crystal_packing FLOAT
domain_contact_errors TEXT
domain_contact_domain_contact_id INT(11)
domain_cath_id VARCHAR(15)
domain_cath_id1 VARCHAR(15)
domain_cath_id2 VARCHAR(15)
domain_cath_id3 VARCHAR(15)
Indexes
uniprot_domain_pair
uniprot_domain_pair_id INT(11)
uniprot_domain_id_1 INT(11)
uniprot_domain_id_2 INT(11)
rigids TEXT
domain_contact_ids TEXT
path_to_data TEXT
Indexes
uniprot_domain_pair_template
uniprot_domain_pair_id INT(11)
domain_contact_id INT(11)
cath_id_1 VARCHAR(15)
cath_id_2 VARCHAR(15)
identical_1 FLOAT
conserved_1 FLOAT
coverage_1 FLOAT
score_1 FLOAT
identical_if_1 FLOAT
conserved_if_1 FLOAT
coverage_if_1 FLOAT
score_if_1 FLOAT
identical_2 FLOAT
conserved_2 FLOAT
coverage_2 FLOAT
score_2 FLOAT
identical_if_2 FLOAT
conserved_if_2 FLOAT
coverage_if_2 FLOAT
score_if_2 FLOAT
5 more...
Indexes
ELASPIC
provean
uniprot_id VARCHAR(15)
provean_supset_filename VARCHAR(255)
provean_supset_length INT(11)
provean_errors TEXT
provean_date_modified DATETIME
Indexes
uniprot_domain_model
uniprot_domain_id INT(11)
model_errors TEXT
alignment_filename VARCHAR(255)
model_filename VARCHAR(255)
chain VARCHAR(15)
norm_dope FLOAT
sasa_score TEXT
m_date_modified TIMESTAMP
model_domain_def VARCHAR(255)
Indexes
uniprot_domain_mutation
uniprot_id VARCHAR(15)
uniprot_domain_id INT(11)
mutation VARCHAR(15)
mutation_errors TEXT
model_filename_wt VARCHAR(255)
model_filename_mut VARCHAR(255)
chain_modeller VARCHAR(15)
mutation_modeller VARCHAR(15)
stability_energy_wt TEXT
stability_energy_mut TEXT
physchem_wt TEXT
physchem_wt_ownchain TEXT
physchem_mut TEXT
physchem_mut_ownchain TEXT
matrix_score FLOAT
secondary_structure_wt TEXT
solvent_accessibility_wt FLOAT
secondary_structure_mut TEXT
solvent_accessibility_mut FLOAT
provean_score FLOAT
ddg FLOAT
mut_date_modified TIMESTAMP
Indexes
uniprot_domain_pair_model
uniprot_domain_pair_id INT(11)
model_errors TEXT
alignment_filename_1 VARCHAR(255)
alignment_filename_2 VARCHAR(255)
model_filename VARCHAR(255)
chain_1 VARCHAR(15)
chain_2 VARCHAR(15)
norm_dope FLOAT
interface_area_hydrophobic FLOAT
interface_area_hydrophilic FLOAT
interface_area_total FLOAT
interface_dg FLOAT
interacting_aa_1 TEXT
interacting_aa_2 TEXT
m_date_modified TIMESTAMP
model_domain_def_1 VARCHAR(255)
model_domain_def_2 VARCHAR(255)
Indexes
uniprot_domain_pair_mutation
uniprot_id VARCHAR(15)
uniprot_domain_pair_id INT(11)
mutation VARCHAR(15)
mutation_errors TEXT
model_filename_wt VARCHAR(255)
model_filename_mut VARCHAR(255)
chain_modeller VARCHAR(15)
mutation_modeller VARCHAR(15)
analyse_complex_energy_wt TEXT
stability_energy_wt TEXT
analyse_complex_energy_mut TEXT
stability_energy_mut TEXT
physchem_wt TEXT
physchem_wt_ownchain TEXT
physchem_mut TEXT
physchem_mut_ownchain TEXT
matrix_score FLOAT
secondary_structure_wt TEXT
solvent_accessibility_wt FLOAT
secondary_structure_mut TEXT
solvent_accessibility_mut FLOAT
contact_distance_wt FLOAT
contact_distance_mut FLOAT
provean_score FLOAT
ddg FLOAT
mut_date_modified TIMESTAMP
Indexes

Figure 2.6: ELASPIC database schema. Tables on the green plate titled Profs are calculated using the Profs pipeline, following the procedure outlined in Figure 2.1. Tables on the purple plate titled ELASPIC are calculated using the ELASPIC pipeline, following the "database pipeline" shown in Figure 2.5.
A detailed description of each table is provided in Table 2.1.

Table 2.1: Description of the tables in the ELASPIC database schema (Figure 2.6).

Table name	Table description
domain	Contains Profs domain definitions for all proteins in the PDB.
domain_contact	Contains information about interactions between Profs domains in the PDB. Only interactions that are predicted to be real by NOXclass [30] are included in this table.
uniprot_sequence	Contains protein sequences for all proteins that are annotated with Profs domains in the **uniprot_domain** table. This table is constructed by downloading and parsing *uniprot_sprot_fasta.gz*, *uniprot_trembl_fasta.gz* and *homo_sapiens_variation.txt* files from the Uniprot.
provean	Contains information about Provean [5] supporting set files. The construction of a supporting set is the longest part of running Provean. Thus, in order to speed up the evaluation of mutations, the supporting set is precalculated and stored for every protein.
uniprot_domain	Contains Profs domain definitions for proteins in the **uniprot_sequence** table. This table is obtained by downloading Pfam domain definitions for all known proteins from SIMAP [20], and mapping those proteins to Uniprot using the MD5 hash of each sequence. Overlapping and repeating domains are either merged or deleted, as described in [19].
uniprot_domain_template	Contains structural templates for domains in the **uniprot_domain** table. The *domain_def* column contains expanded and corrected domain definitions for every domain.
uniprot_domain_model	Contains information about the homology models that were created using structural templates in the **uniprot_domain_template** table.
uniprot_domain_mutation	Contains information about the structural impact of core mutations, calculated by introducing those mutations into homology models listed in the **uniprot_domain_model** table. The *ddg* column contains the predicted change in the Gibbs free energy of protein folding.
uniprot_domain_pair	Contains pairs of domains that are likely to mediate the interaction between pairs of proteins listed in Hippie [22] and Rolland *et al.* [23].
uniprot_domain_pair_template	Contains structural templates for domain pairs in the **uniprot_domain_pair** table.
uniprot_domain_pair_model	Contains information about homology models that were created using structural templates in the **uniprot_domain_pair** table.
uniprot_domain_pair	Contains information about the structural impact of interface mutations, calculated by introducing those mutations into homology models listed in the **uniprot_domain_pair_model** table. The *ddg* column contains the predicted change in the Gibbs free energy of protein-protein binding.

2.2.3 Jobsubmitter

In order to make ELASPIC accessible to a wider scientific audience, Daniel Witvliet created the ELASPIC webserver, which allows users to run ELASPIC for their protein and mutation of interest and to analyze interactively ELASPIC results [19].

One limitation of the webserver was that it spawned ELASPIC jobs on the same virtual machine as the webserver. This meant that only a few mutations could be analyzed at a time, and that the webserver could stall when running mutations in a protein lacking a precalculated Provean supporting set, since constructing a Provean supporting set could require more RAM than the virtual machine had available. In order to make the webserver scale to thousands of mutations, we decided to restructure the job execution backend to run ELASPIC on the local Sun Grid Engine (SGE) cluster. However, this design introduced several challenges.

First, since users can run multiple mutations affecting the same protein, we had to make sure that the Provean supporting sets and homology models are calculated first, before jobs for individual mutations are submitted to the cluster. Otherwise, each mutation would initiate the calculation of a Provean supporting set, which can require more than 5 GB of memory, and a homology model, which can take more than 30 minutes to complete. This would lead to many unnecessary jobs, drastically lowering our throughput, and could lead to inconsistent results, since different jobs can generate different supporting sets and homology models, even for the same protein, due to the inherent randomness of those tasks.

Second, jobs running on a SGE cluster can die unexpectedly, if, for example, they exceed allocated resources, or if the node on which they are executing experiences a hardware failure. In most cases, the jobs do not get an opportunity to send an error message before they are terminated. Therefore, we had to keep track of all running jobs, and resubmit jobs that do not finish successfully.

Third, in order to send a "Job Complete" email once all mutations submitted by a particular user have been evaluated, we had to keep track of the relationship between mutations and users that submitted those mutations.

One possible way to address those design requirements would be to use an asynchronous task queue, such as Celery. However, since different tasks inside the queue do not have a shared memory state, each task would have to periodically execute a *qstat* command on the SGE master node in order to monitor the status of the submitted jobs. Since we could have thousands of mutations running on the cluster at the same time, this would not be a scalable solution.

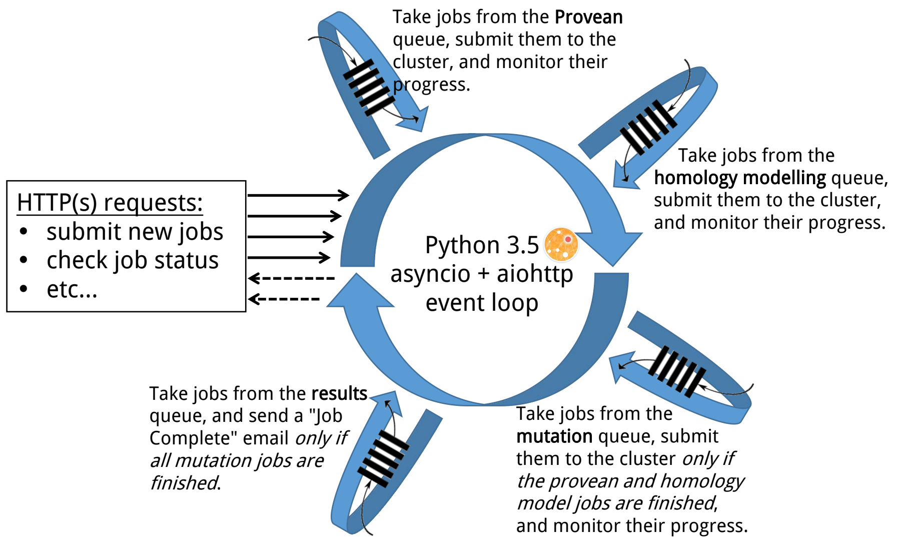
An alternative approach, which we used for the final design, was to create an independent web service responsible for submitting ELASPIC jobs to the SGE cluster and monitoring their progress. We called this web service the ELASPIC "jobsubmitter". It was implemented using the *aiohttp* library, which leverages the *asyncio* event loop and improved support for asyncronous programming present in Python 3.5 (Figure 2.7a). Once the jobsubmitter receives a *GET* or *POST* request containing a set of mutations, information concerning those mutations is distributed into the following queues:

- A "Provean" queue, which contains proteins for which a Provean supporting set has not been calculated.
- A "homology model" queue, which contains proteins for which a homology model has not been calculated.
- A "mutation" queue, which contains individual mutations.
- An "email" queue, which contains the set of mutations associated with each job.

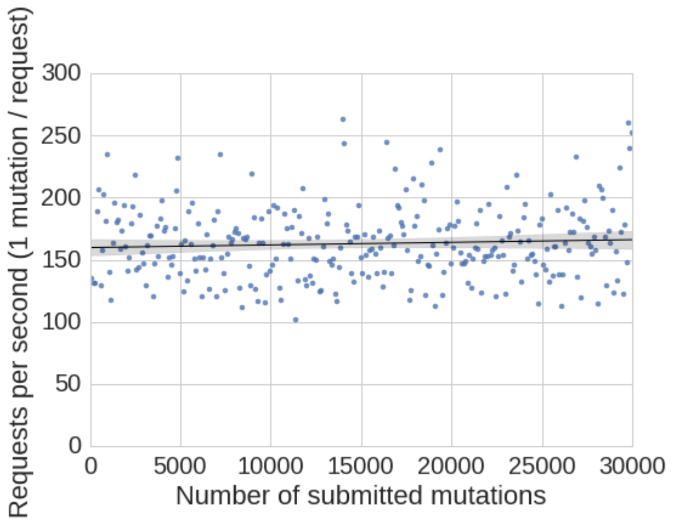
The information from those queues is then processed by the corresponding coroutines:

- For each protein in the "Provean" queue, a job is submitted to the SGE cluster, which calculates the Provean supporting set. If the Provean supporting set for the protein has already been calculated, the protein is taken of the "Provean" queue with no further action.
- For each protein in the "homology model" queue, a job is submitted to the SGE cluster, which calculates the homology model of the protein. If the homology model of the protein has already been calculated, the protein is taken of the "homology model" queue with no further action.
- For each mutation in the "mutation" queue, a job is submitted to the SGE cluster, which runs ELASPIC to calculate the $\Delta\Delta G$ of the mutation. This happens *only if the Provean supporting set and homology model for the protein have already been calculated!*
- For each job in the "email" queue, a "Job Complete" email is sent to the specified email address once all mutations for the associated job have been completed.

The ELASPIC jobsubmitter is highly performant. It is able to handle over 150 requests per second, even with 30,000 mutations already being processed by the web service (Figure 2.7b).


Take jobs from the **Provean** queue, submit them to the cluster, and monitor their progress.
Take jobs from the **homology modelling** queue, submit them to the cluster, and monitor their progress.
HTTP(s) requests:
• submit new jobs
• check job status
• etc...
Python 3.5
asyncio + aiohttp
event loop
Take jobs from the **results** queue, and send a "Job Complete" email *only if all mutation jobs are finished.*
Take jobs from the **mutation** queue, submit them to the cluster *only if the provean and homology model jobs are finished*, and monitor their progress.

(a) Overview of the "jobsubmitter" web service. The web service was implemented using Python 3.5 and the *aiohttp* library. It contains a central *asyncio* event loop, data structures holding information about the mutations being processed, and coroutines which submit jobs to the SGE cluster, monitor job progress, and perform other maintenance tasks.



(b) Plot showing the number of requests per second handled by the ELASPIC jobsubmitter as a function of the number of mutations that are already being processed.

Figure 2.7: Implementation (a) and performance (b) of the ELASPIC "jobsubmitter".

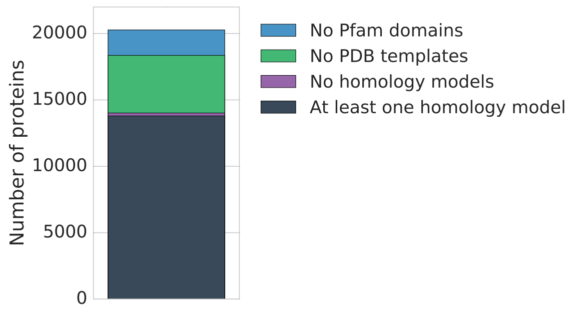
2.2.4 Precalculated data

In order to increase the speed with which the ELASPIC webserver could generate results, we attempted to precalculate homology models and Provean supporting sets for all human proteins, and to calculate mutations known to be involved in human disease.

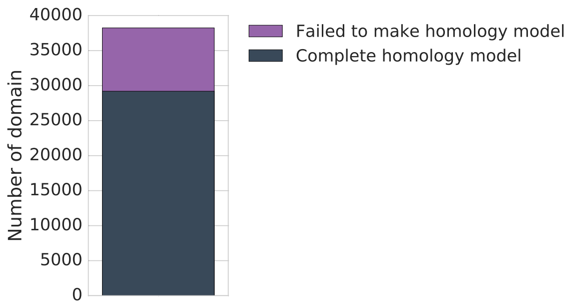
Out of 20,270 proteins in human SwissProt, 18,355 proteins have at least one Pfam domain, and 14,015 proteins have a Pfam domain for which we could find a structural template in the PDB (Figure 2.8a). We could create a homology model of at least one domain in 13,796 proteins, with the fraction of each protein covered by a homology model shown in Figure 2.8c. On the domain level, out of a total of 38,243 domains with a structural template, we could create a homology model for only 29,201 domains, or 76% (Figure 2.8b). The main reason for failing to calculate a homology model was low sequence identity between the domain being modelled and the structural template.

We also attempted to create a homology model for all protein pairs found in the HIPPIE [22] and CCSB [23] databases, keeping only the pairs where each protein has at least one domain with a homology model and where we would could find a structural template of the protein-protein interaction. Out of a total of 19,964 such protein pairs, we calculated a homology models for 18,956, or 95 % (Figure 2.9).

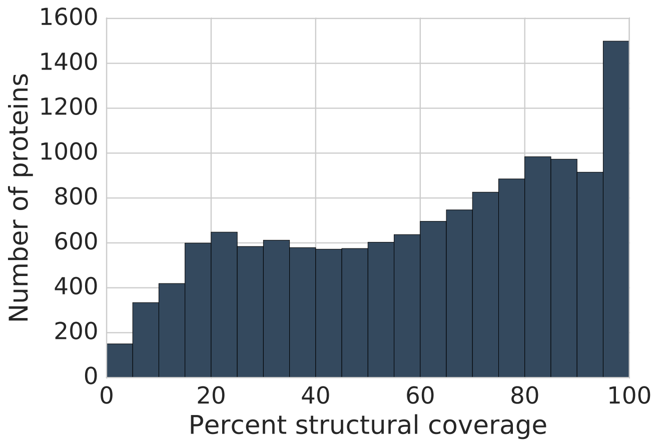
We successfully precalculated a Provean supporting set for all 14,015 human proteins with a Profs domain with a structural template. We also ran ELASPIC for over 990,000 million mutations implicated in human diseases or found in human cancers, including nearly 600,000 mutations in different protein-protein interfaces.



(a) Diagram showing the number of *proteins* in the human SwissProt database that have no Pfam domains (blue), that have Pfam domains but no structural templates (green), that have Pfam domains and structural templates but no homology models (purple), and proteins with a homology model of at least one domain (grey).



(b) Diagram showing the number of *domains* in all proteins in the human SwissProt database for which we failed to create a homology model (purple) and for which we successfully created a homology model (grey). The most common reason for failing to create a homology model was low sequence identity between the Profs domain and the structural template.



(c) The percentage of protein sequence covered by Profs domains with homology models, for all proteins in the human SwissProt database that have a homology model of at least one domain.

Figure 2.8: Plots showing the number of proteins for which we could create a homology model (a), the number of domains for which we could create a homology model (b), and the structural coverage of proteins with at least one modelled domain (c). Plots were generated using all human proteins in the SwissProt database.

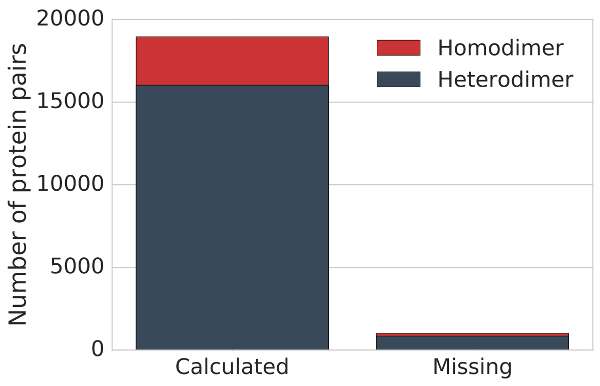


Figure 2.9: Number of homo-dimeric (red) and hetero-dimeric (grey) protein-protein interactions for which we created a homology model (left) and failed to create a homology model (right). In this figure, protein-protein interactions are defined as all pairs of proteins from the human SwissProt database that are found to interact according to one of the protein-protein interaction databases (see Figure 2.4) and that have at least one structural template of the interaction.

Chapter 3

Results

After making changes to the ELASPIC pipeline that are described in Section 2.2, we retrained ELASPIC core and interface predictors and validated them on new data. This involved curating high-quality training, validation and test datasets, selecting the best hyperparameters for the machine learning algorithm using grid-search, selecting the set of most informative features using feature elimination, and testing the final predictor on external datasets to compare our performance with competing methods.

The changes made to the ELASPIC pipeline and the large number of precalculated mutations associated with human disease.

3.1 Datasets

All datasets that we used to train, validate and test the predictors are described in Table 3.1. The dataset that we used to train the core predictor, labelled "Protherm" for simplicity, contains data from the Protherm database [31] combined with datasets curated by Kellogg *et al.* [32]. The dataset that we used to train the interface predictor, labelled "Skempi" for simplicity, contains data from the Skempi database [33] combined with the dataset curated by Kortemme and Baker [34].

We evaluated mutations in our training set using both the standalone and the database pipelines in order to make sure that both pipelines produce comparable results and that the resulting predictors perform equally well for both pipelines. The main difference between the standalone and database pipelines is that the standalone pipeline constructs the Provean supporting set using the sequence of the PDB chain that is being mutated, while the database pipeline constructs the Provean supporting set using the sequence of the entire protein containing the domain that is being mutated. For the database pipeline we attempted to construct several homology models ranging from low to high sequence identity for each mutation. We expected that including more proteins of lower sequence identity would improve the ability of the predictor to generalize to external datasets, since the training set is over-represented in proteins that have a crystal structure deposited in the PDB.

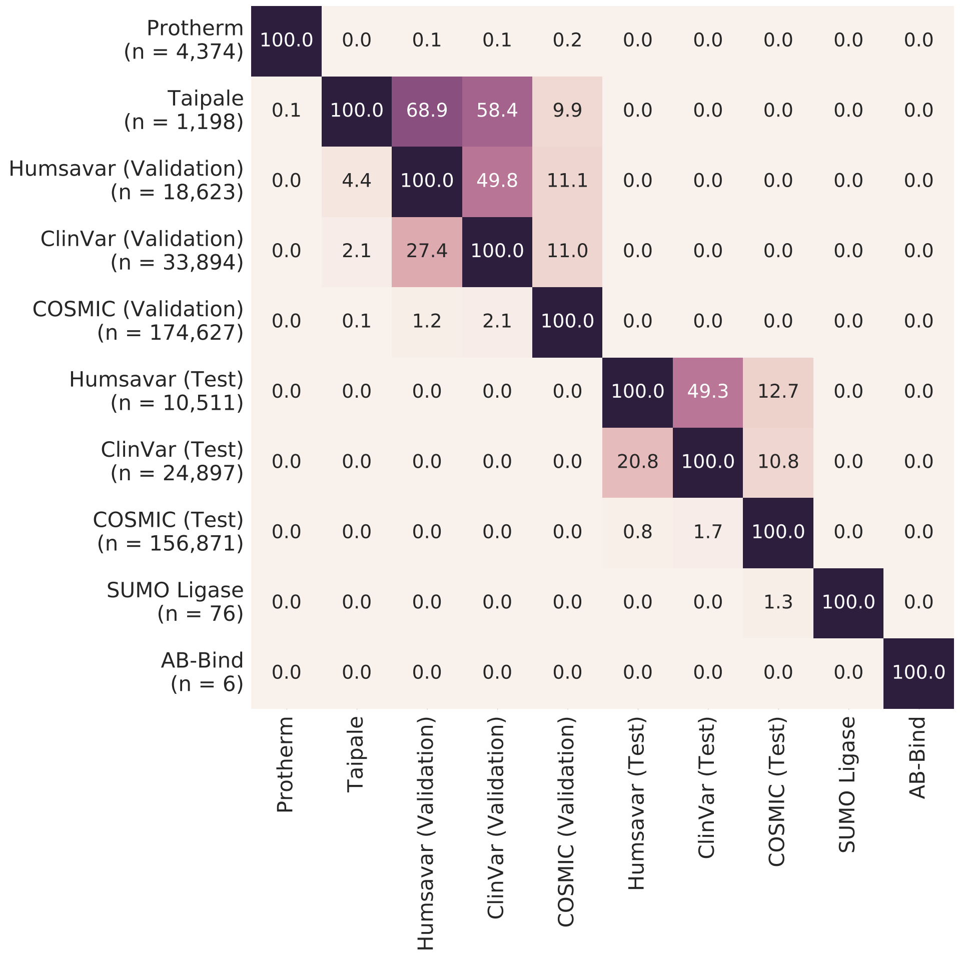
Previously, we had seen a low correlation between ELASPIC-predicted $\Delta\Delta G$ and the deleteriousness of mutations. To address this, we split the Humsavar, ClinVar, and COSMIC datasets into validation and test subsets, and used the performance on the validation subset as part of the scoring function for selecting the optimal hyperparameters and number of features.

We made sure that no mutations in the test set appear in our training and validation sets (see

Table 3.1: Description of the datasets that were used in this study.

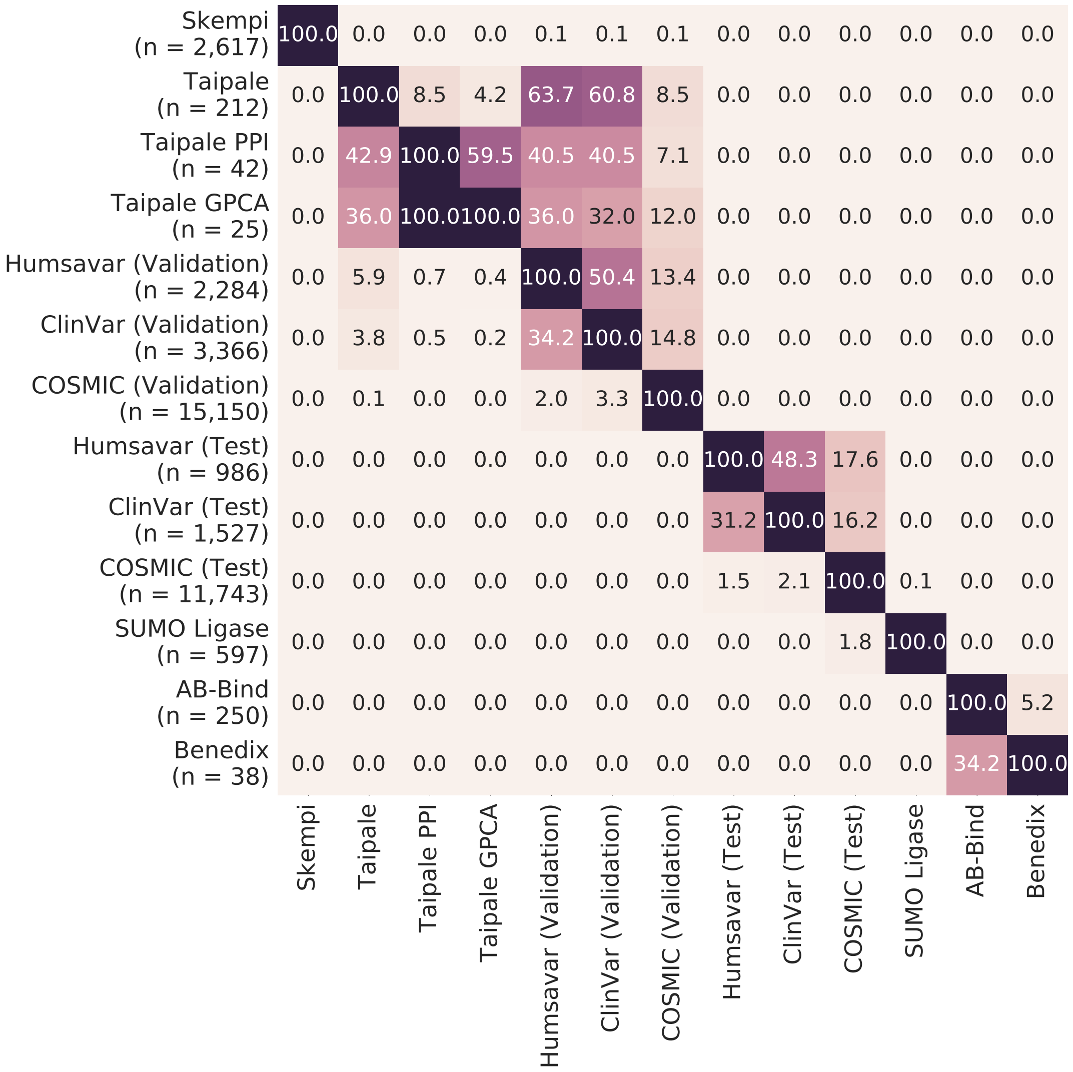
Name	Type	Description
Protherm	Train	Database of mutations-induced changes in the Gibbs free energy of protein folding ($\Delta\Delta G_{core}$) [35].
Skempi	Train	Database of mutations-induced changes in the Gibbs free energy of protein-protein interactions ($\Delta\Delta G_{interface}$) [33].
Taipale	Validation	Interaction between chaperones and wildtype or mutant proteins, quantified using the LUMIER assay [36].
Taipale PPI	Validation	Results of yeast two-hybrid experiments, measuring the presence or absence of protein-protein interactions for wild-type and mutant proteins [36].
Taipale GPCA	Validation	*Gaussia princeps* luciferase protein complementation assay, measuring the effect of mutations on protein affinity [36].
Humsavar	Validation & Test	Disease-causing mutations and polymorphisms obtained from the UniProt *humsavar.txt* file [37]. Mutations annotated with at least one disease were assigned a value of 1. Mutations annotated as "polymorphisms" were assigned a value of 0.
ClinVar	Validation & Test	Disease-causing mutations and polymorphisms obtained from ClinVar [38]. Mutations found in the ClinVar *clinvar_20160531.vcf* file were assigned a value of 1. Mutations found in the ClinVar *common_no_known_medical_impact_20160531.vcf* file were assigned a value of 0.
COSMIC	Validation & Test	Mutations found in cancer [39]. Mutations classified by FATHMM [7] as cancer drivers were assigned a value of 1. Mutations classified by FATHMM as cancer passengers were assigned a value of 0.
SUMO Ligase	Test	Mutations affecting the activity of SUMO ligase, measured using a cell viability assey [40].
AB-Bind	Test	Mutations explored in antibody affinity maturation experiments [41].
Benedix	Test	Mutations from alanine scanning of the TEM1 (β-lactamase) – BLIP (β-lactamase-inhibitor) interface [15].

Figures 3.1 and 3.2 for core and interface mutations, respectively). In the case of Humsavar, ClinVar and COSMIC datasets, we made sure that no *protein* in the test set appear in the training and validation sets.



	Protherm	Taipale	Humsavar (Validation)	ClinVar (Validation)	COSMIC (Validation)	Humsavar (Test)	ClinVar (Test)	COSMIC (Test)	SUMO Ligase	AB-Bind
Protherm (n = 4,374)	100.0	0.0	0.1	0.1	0.2	0.0	0.0	0.0	0.0	0.0
Taipale (n = 1,198)	0.1	100.0	68.9	58.4	9.9	0.0	0.0	0.0	0.0	0.0
Humsavar (Validation) (n = 18,623)	0.0	4.4	100.0	49.8	11.1	0.0	0.0	0.0	0.0	0.0
ClinVar (Validation) (n = 33,894)	0.0	2.1	27.4	100.0	11.0	0.0	0.0	0.0	0.0	0.0
COSMIC (Validation) (n = 174,627)	0.0	0.1	1.2	2.1	100.0	0.0	0.0	0.0	0.0	0.0
Humsavar (Test) (n = 10,511)	0.0	0.0	0.0	0.0	0.0	100.0	49.3	12.7	0.0	0.0
ClinVar (Test) (n = 24,897)	0.0	0.0	0.0	0.0	0.0	20.8	100.0	10.8	0.0	0.0
COSMIC (Test) (n = 156,871)	0.0	0.0	0.0	0.0	0.0	0.8	1.7	100.0	0.0	0.0
SUMO Ligase (n = 76)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	100.0	0.0
AB-Bind (n = 6)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0

Figure 3.1: Overlap in core mutations between all the datasets used in this study. The shade and value inside each square denotes the percentage of mutations in the dataset named on the y-axis that are also found in the database named on the x-axis. Core mutations are defined as mutations that do not occur within 6 Åof a neighbouring chain in the provided PDB or protein-protein homology model. A description of each dataset can be found in Table 3.1.



	Skempi	Taipale	Taipale PPI	Taipale GPCA	Humsavar (Validation)	ClinVar (Validation)	COSMIC (Validation)	Humsavar (Test)	ClinVar (Test)	COSMIC (Test)	SUMO Ligase	AB-Bind	Benedix
Skempi (n = 2,617)	100.0	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0
Taipale (n = 212)	0.0	100.0	8.5	4.2	63.7	60.8	8.5	0.0	0.0	0.0	0.0	0.0	0.0
Taipale PPI (n = 42)	0.0	42.9	100.0	59.5	40.5	40.5	7.1	0.0	0.0	0.0	0.0	0.0	0.0
Taipale GPCA (n = 25)	0.0	36.0	100.0	100.0	36.0	32.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0
Humsavar (Validation) (n = 2,284)	0.0	5.9	0.7	0.4	100.0	50.4	13.4	0.0	0.0	0.0	0.0	0.0	0.0
ClinVar (Validation) (n = 3,366)	0.0	3.8	0.5	0.2	34.2	100.0	14.8	0.0	0.0	0.0	0.0	0.0	0.0
COSMIC (Validation) (n = 15,150)	0.0	0.1	0.0	0.0	2.0	3.3	100.0	0.0	0.0	0.0	0.0	0.0	0.0
Humsavar (Test) (n = 986)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	48.3	17.6	0.0	0.0	0.0
ClinVar (Test) (n = 1,527)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	31.2	100.0	16.2	0.0	0.0	0.0
COSMIC (Test) (n = 11,743)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.5	2.1	100.0	0.1	0.0	0.0
SUMO Ligase (n = 597)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.8	100.0	0.0	0.0
AB-Bind (n = 250)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	5.2
Benedix (n = 38)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	34.2	100.0

Figure 3.2: Overlap in interface mutations between all the datasets used in this study. The shade and value inside each square denotes the percentage of mutations in the dataset named on the y-axis that are also found in the database named on the x-axis. Interface mutations are defined as mutations that occur within 6 Åof a neighbouring chain in the provided PDB or protein-protein homology model. A description of each dataset can be found in Table 3.1.

3.2 Hyperparameter optimisation

ELASPIC uses the gradient boosting regressor (GBR) algorithm, implemented in scikit-learn [29], to combine over 70 different sequential and structural features into a score that corresponds to the Gibbs free energy change of protein folding or protein-protein interaction. The GBR algorithm was selected because it achieved higher performance than linear regression, support vector machine and random forest algorithms, in 20-fold cross-validation over the training set [16].

Since the ELASPIC pipeline received many bugfixes since the original publication, and was restructured to work as a backend to a webserver, we retrained GBR predictors using features generated by the updated pipeline. In order to select the best set of GBR hyperparameters, we performed exhaustive “grid-search”, where we measured the performance of the GBR algorithm for 3,600 different combinations of hyperparameters (Table 3.2). For each set of parameters, we computed the Spearman correlation between predicted and experimental $\Delta\Delta G$ values for mutations in the training set, using 4-fold cross-validation. We also computed the Spearman correlation between predicted $\Delta\Delta G$ values and the measured values for our “Validation” datasets (datasets in Table 3.1 annotated as “Validation”). In the case of the “Taipale” dataset, the experimental value was the difference in the average LUMIER score between the wildtype and mutant proteins and the corresponding chaperones. In the case of the “Taipale PPI” dataset, the experimental value was 1 if the mutation led to the loss of the interaction and 0 if it led to the gain of interaction or if it had no effect. In the case of the “Taipale GPCA” dataset, the experimental value was the difference in *Gaussia princeps* luminosity between wild-type and mutant proteins. In the case of “Humsavar”, “ClinVar” and “COSMIC” datasets, the experimental value was 1 if the mutation was classified as deleterious and 0 if it was classified as benign. While mutation deleteriousness and $\Delta\Delta G$ are different metrics, it is expected that deleterious mutations, on average, should have a higher impact on protein structure that benign mutations. Therefore, accurate $\Delta\Delta G$ predictions should have a higher correlation with the deleteriousness score, defined as 1 for deleterious mutations and 0 for benign mutations.

We used the combined scores CS_{core} (Equations 3.1) and $CS_{interface}$ (Equation 3.2) to select the best set of hyperparameters for the core and interface predictors. The contribution of each dataset to the combined score was selected in an “ad-hoc” manner, assigning more weight to large datasets than to small datasets, and making sure that the performance on energy-based datasets, including training and Taipale datasets, had a bigger overall impact on the combined score than performance on a deleteriousness-based datasets, such as Humsavar, ClinVar and COSMIC. We used combined scores instead of cross-validation alone because we wanted to select predictors that not only perform well on the training set, but also generalize to external datasets. Since our training sets are limited and biased in the number and type of proteins and protein-protein interactions that they contain, the performance of the predictor in cross-validation may not be an accurate indicator of its performance in general. Since our validation datasets contain many more distinct proteins, protein-protein interactions and mutations than our training sets, they offer a helpful indication of how well predictors generalize to other proteins in the human genome.

$$CS_{core} = \frac{3 \cdot Cross\_validation + Humsavar + ClinVar + COSMIC + Taipale}{7} \tag{3.1}$$

$$CS_{interface} = \frac{3 \cdot Cross\_validation + Humsavar + ClinVar + COSMIC + \frac{Taipale\_PPI}{4} + \frac{Taipale\_GPCA}{4}}{6.5} \tag{3.2}$$

We plot the performance of core and interface predictors trained using different sets of hyperparameters and sorted according to the combined score in Figures 3.3 and 3.4. Cross-validation performance of the predictor is highly correlated with its performance on the validation datasets. However, selecting hyperparameters solely based on cross-validation performance would result in a predictor that substantially underperforms on the validation datasets.

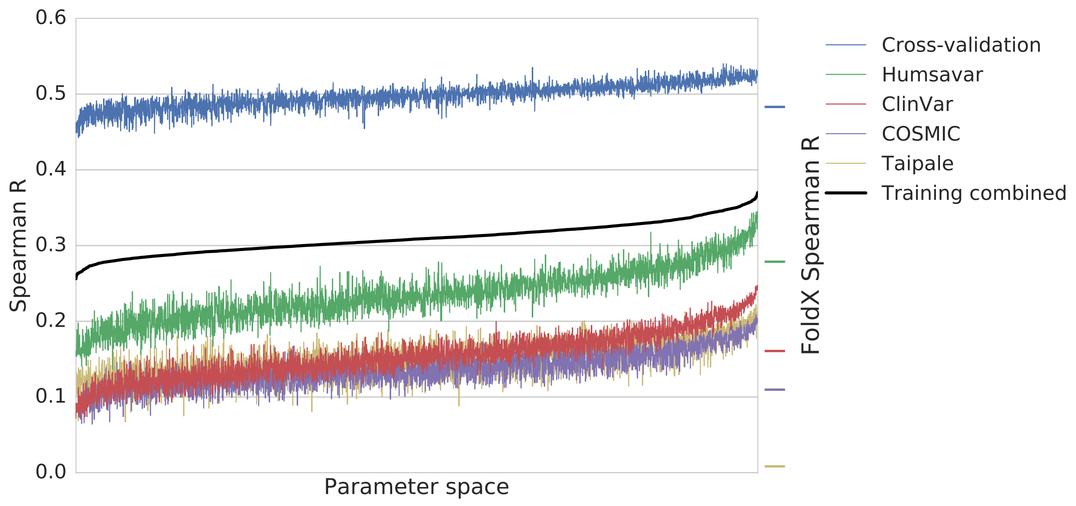


Figure 3.3: Core predictor hyperparameter optimization. The combined score (black line) was calculated using Equation 3.1.

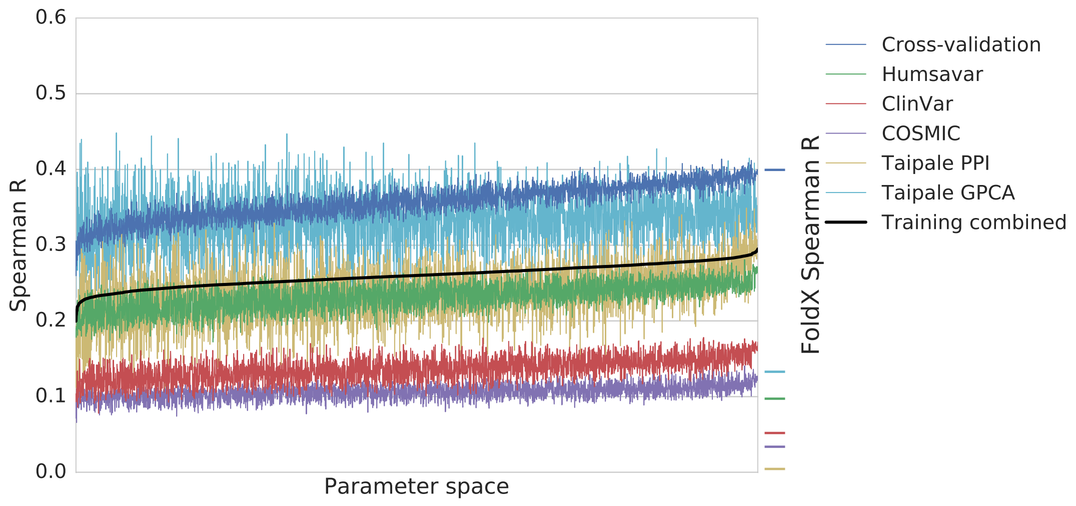


Figure 3.4: Interface predictor hyperparameter optimization. The combined score (black line) was calculated using Equation 3.2.

Table 3.2: Hyperparameter search space for tuning the gradient boosting regressor algorithm used in the core and interface predictors. An all-by-all combination of these hyperparameters was explored in order to find those that produce the best-performing predictors.

Parameter name	Parameter value
alpha	0.99, 0.95, 0.9, 0.8, 0.7, 0.5
learning_rate	0.1, 0.05, 0.02, 0.01
loss	huber
max_depth	10, 8, 6, 4
max_features	1.0, 0.8, 0.5, 0.3, 0.1,
min_samples_leaf	29, 21, 17, 13, 9, 5, 3
n_estimators	2000

Table 3.3: Hyperparameters selected for the core predictor.

Parameter name	Parameter value
alpha	0.5
learning_rate	0.01
loss	huber
max_depth	4
max_features	0.246
min_samples_leaf	17
n_estimators	2000

Table 3.4: Hyperparameters selected for the interface predictor.

Parameter name	Parameter value
alpha	0.9
learning_rate	0.01
loss	huber
max_depth	4
max_features	0.3
min_samples_leaf	13
n_estimators	2000

3.3 Feature elimination

After selecting the best set of hyperparameters for core and interface predictors, we performed feature elimination to evaluate the contribution of each feature to the overall accuracy of the predictor, and to select the sets of features that result in the most accurate predictions.

Feature elimination was performed using a recursive strategy, which involved:

- Leaving out each feature from the training set, one at a time.
- Training the predictor using all but the left out feature.
- Calculating the combined score (CS_{core} or $CS_{interface}$) evaluating the performance of the predictor.
- Discarding the feature that, when left out, produced the predictor with the highest combined score.
- Repeating the process until only one feature remains.

Performances of the core and interface predictors at every step of feature elimination are shown in Figures 3.5 and 3.6. The sets of features that produced the best-performing predictors are described in Tables 3.5 and 3.6.

Most features play a surprisingly small role in the performance of the ELASPIC predictor. In fact, we can achieve near-optimal performance with both core and interface predictors by using only 6 features (displayed in bold in Tables 3.5 and 3.6). This suggests either that most features are not informative in predicting the energetic effect of mutations, or that the training set is too noisy for the contribution of those features to make a significant impact on the accuracy of the predictor.

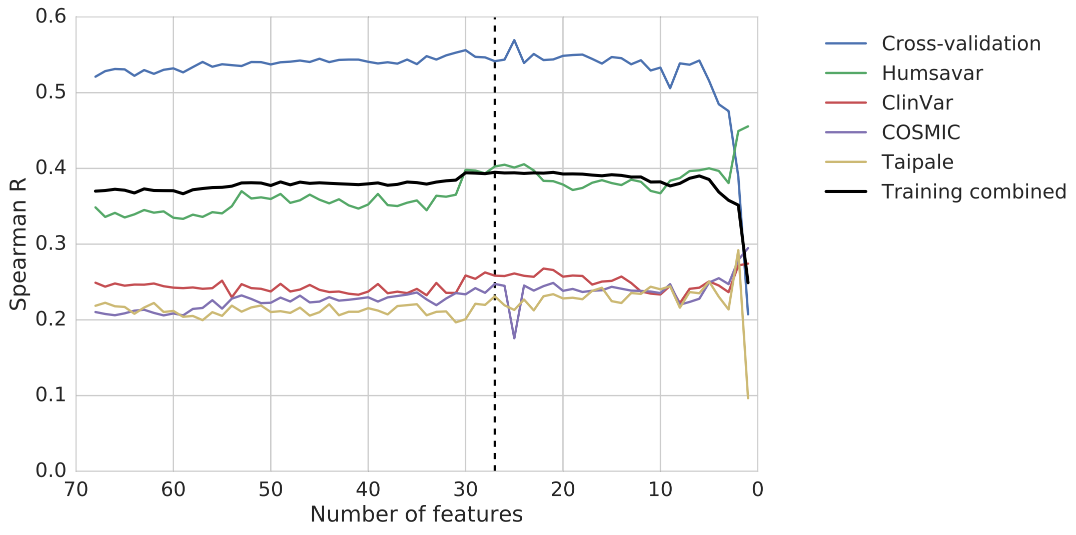


Figure 3.5: Performance of the core predictor at each step of feature elimination. The combined score (black line) was calculated using Equation 3.1. The set of features producing the predictor with the highest combined score are indicated by the vertical dashed line and are listed in Table 3.5.

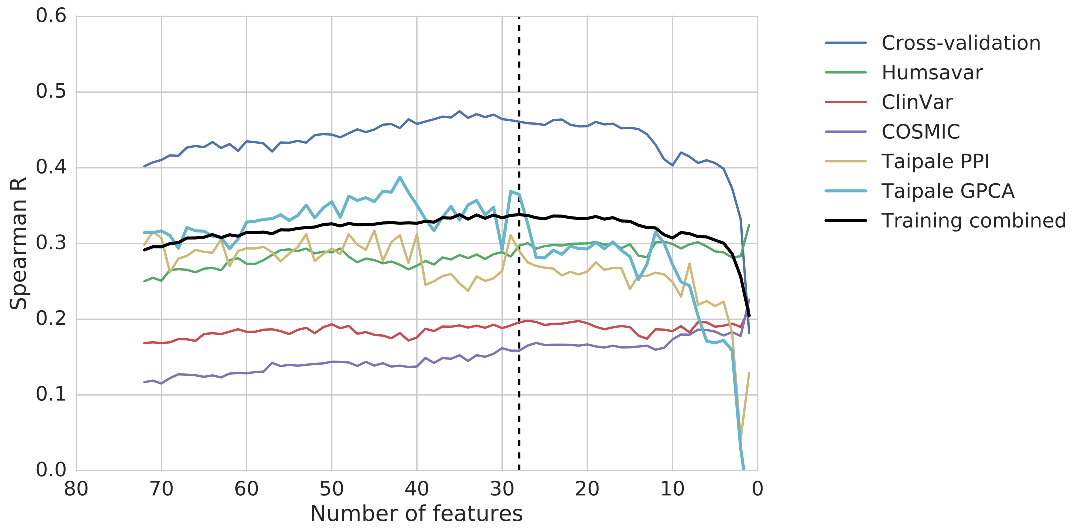


Figure 3.6: Performance of the interface predictor at each step of feature elimination. The combined score (black line) was calculated using Equation 3.2. The set of features producing the predictor with the highest combined score are indicated by the vertical dashed line and are listed in Table 3.6.

Table 3.5: Core predictor features. Rows in bold mark the 6 most important features. FoldX feature descriptions were taken from `http://foldxsuite.crg.eu/command/Stability`.

Feature name	Feature source	Feature description
alignment_coverage	ELASPIC	Structural template alignment coverage.
alignment_identity	ELASPIC	Structural template sequence identity.
alignment_score	ELASPIC	Structural template quality (Equation 2.1).
backbone_hbond_change	FoldX	Backbone hydrogen bond energy.
backbone_hbond_wt	FoldX	This the contribution of backbone hydrogen bonds.
cis_bond_wt	FoldX	Cis peptide bond energy.
disulfide_wt	FoldX	Contribution of disulfide bonds.
electrostatic_kon_change	FoldX	Electrostatic interaction between molecules in the pre-complex.
electrostatics_change	**FoldX**	**Electrostatic interactions.**
entropy_mainchain_change	FoldX	Entropy cost of fixing the main chain.
helix_dipole_wt	FoldX	Electrostatic contribution of the helix dipole.
matrix_score	ELASPIC	BLOSUM62 matrix score.
pcv_hbond_change	ELASPIC	Hydrogen-oxygen contacts involving atoms of the mutated residue and atoms of the interacting chain.
pcv_hbond_self_change	ELASPIC	Hydrogen-oxygen contacts involving atoms of the mutated residue and atoms of the mutated chain.
pcv_salt_equal_change	ELASPIC	Charge repulsions between atoms of the mutated residue and atoms of the interacting chain.
pcv_salt_equal_self_wt	ELASPIC	Charge repulsions between atoms of the mutated residue and atoms of the mutated chain.
pcv_salt_equal_wt	ELASPIC	Charge repulsions between atoms of the mutated residue and atoms of the interacting chain.
pcv_salt_opposite_change	ELASPIC	Charge attractions between atoms of the mutated residue and atoms of the interacting chain.
pcv_vdw_self_change	ELASPIC	Carbon carbon contacts between atoms of the mutated residue and atoms of the mutated chain.
provean_score	**Provean**	**Sequence conservation score.**
sloop_entropy_wt	FoldX	Entropic cost according to the SLoop database of loop conformations.
solvation_hydrophobic_change	**FoldX**	**Contribution of hydrophobic groups.**
solvation_polar_change	**FoldX**	**Energetic penalty for burying polar groups.**
solvent_accessibility_wt	**MSMS**	**Solvent-accessible surface area of the mutated residue.**
torsional_clash_change	FoldX	Intra-residue van der Waals torsional clashes.
van_der_waals_clashes_change	**FoldX**	**Energy penalization due to van der Waals clashes (interresidue).**
water_bridge_wt	FoldX	Contribution of water bridges.

Table 3.6: Interface predictor features. Rows in bold mark the 6 most important features. FoldX feature descriptions were taken from `http://foldxsuite.crg.eu/command/AnalyseComplex`.

Feature name	Feature source	Feature description
alignment_score	ELASPIC	Alignment quality (Equation 2.2).
backbone_clash_change	FoldX	Backbone-backbone van der Waals energy.
backbone_clash_wt	FoldX	Backbone-backbone van der Waals energy.
backbone_hbond_change	FoldX	Backbone hydrogen bond energy.
cis_bond_wt	FoldX	Cis peptide bond energy.
electrostatic_kon_wt	FoldX	Electrostatic interaction between molecules in the pre-complex.
energy_ionisation_wt	FoldX	Ionization energy.
entropy_complex_change	FoldX	Entropic cost of forming a complex.
entropy_sidechain_change	**FoldX**	**Entropic cost of fixing the side chain.**
intraclashes_energy_2_change	FoldX	van der Waals clashes of residues at the interface of the complex with their own molecule (type 2).
partial_covalent_bonds_wt	**FoldX**	**Interactions with bound metals.**
pcv_hbond_self_change	ELASPIC	Hydrogen-oxygen contacts involving atoms of the mutated residue and atoms of the mutated chain.
pcv_hbond_wt	ELASPIC	Hydrogen-oxygen contacts involving atoms of the mutated residue and atoms of the interacting chain.
pcv_salt_equal_self_change	ELASPIC	Charge repulsions involving atoms of the mutated residue and atoms of the mutated chain.
pcv_salt_equal_wt	ELASPIC	Charge repulsions involving atoms of the mutated residue and atoms of the interacting chain.
pcv_salt_opposite_change	ELASPIC	Charge attractions involving atoms of the mutated residue and atoms of the interacting chain.
pcv_salt_opposite_self_change	ELASPIC	Charge attractions involving atoms of the mutated residue and atoms of the mutated chain.
pcv_salt_opposite_self_wt	ELASPIC	Charge attractions involving atoms of the mutated residue and atoms of the mutated chain.
pcv_vdw_self_change	ELASPIC	Carbon carbon contacts involving atoms of the mutated residue and atoms of the mutated chain.
pcv_vdw_self_wt	**ELASPIC**	**Carbon carbon contacts involving atoms of the mutated residue and atoms of the mutated chain.**
pcv_vdw_wt	**ELASPIC**	**Carbon carbon contacts involving atoms of the mutated residue and atoms of the interacting chain.**
provean_score	**Provean**	**Sequence conservation score.**
sloop_entropy_change	FoldX	Entropic cost according to the SLoop database of loop conformations.
solvation_hydrophobic_change	FoldX	Contribution of hydrophobic groups.
solvation_polar_change	**FoldX**	**Energetic penalty for burying polar groups.**
solvation_polar_wt	FoldX	Energetic penalty for burying polar groups.
torsional_clash_change	FoldX	Intra-residue van der Waals torsional clashes.
water_bridge_change	FoldX	Contribution of water bridges.

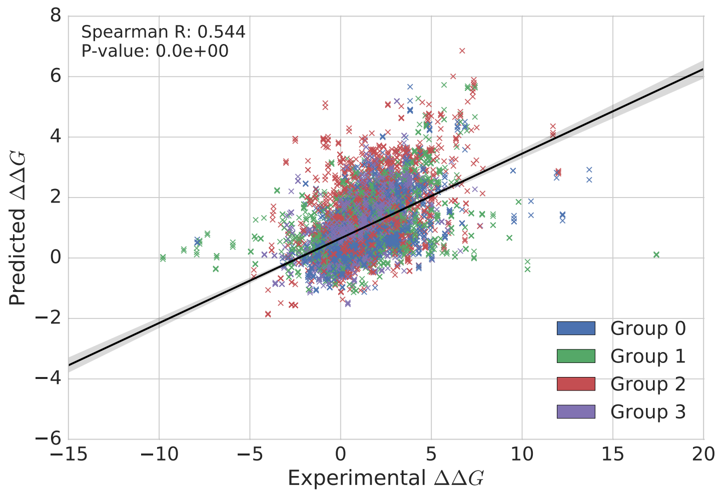
3.4 Validation

The final ELASPIC core and interface predictors were trained using gradient boosting regressor hyper-parameters listed in Tables 3.3 and 3.5 and sets of features described in Tables 3.4 and 3.6. Performance of the resulting predictors on the training, validation and test datasets are presented in Figures 3.7 and 3.8, for core and interface predictors, respectively.

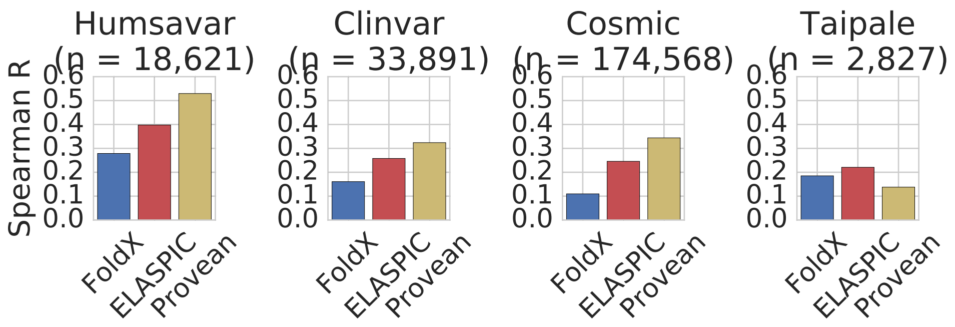
The ELASPIC core predictor outperforms FoldX and Provean on the Taipale dataset, which is the only validation dataset explicitly measuring the effect of mutations on protein stability rather than whether or not the mutation is associated with a disease (Figure 3.7b). It also outperforms FoldX and Provean on the core subsets of the SUMO and AB-Bind test datasets (Figure 3.7c). The core subsets of those datasets only contain mutations located more than 6 Å away from another chain in the PDB.

The ELASPIC interface predictor also outperforms FoldX and Provean on the Taipale GPCA dataset (Figure 3.8b). It performs marginally worse than Provean on the Taipale PPI dataset, but this is likely because the Taipale PPI dataset is mostly made up of deleterious mutations, which are predicted well by Protherm. The ELASPIC interface outperforms Protherm and FoldX on the SUMO Ligase and AB-Bind test datasets (Figure 3.8c) both alone and in combination with the core predictor. The ELASPIC interface predictor shows slightly lower performance than FoldX on the very small Benedix dataset.

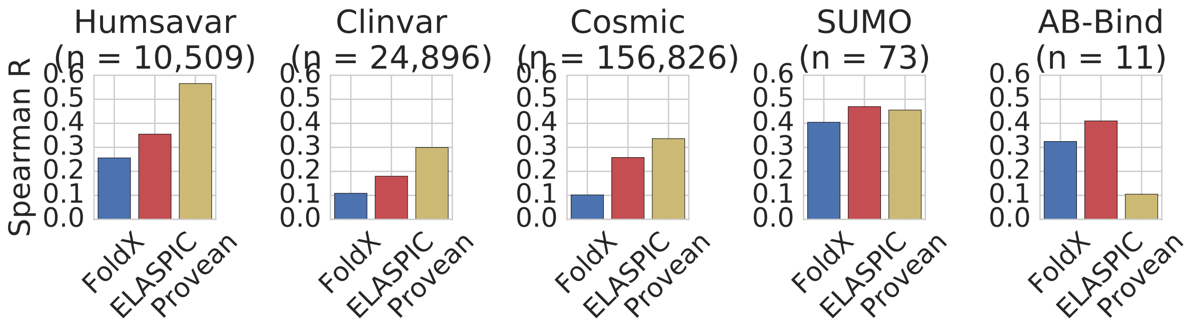
Both the core and interface predictors performs better than FoldX but worse than Provean on the validation and test subsets of the Humsavar, ClinVar and COSMIC (Figures 3.7c and 3.8c). The low performance of the ELASPIC compared to Provean was expected, since the ELASPIC core predictor attempts to model the effect of mutations on protein stability, and does not account for other reasons that may lead to a mutation being deleterious. For example, mutations could affect the active site or the signal sequence of a protein, which may prove to be highly deleterious to the organism but have only a marginal effect on protein stability.



(a) Performance of the ELASPIC core predictor on the training dataset, evaluated using four-fold cross-validation. Colours indicate different cross-validation bins.

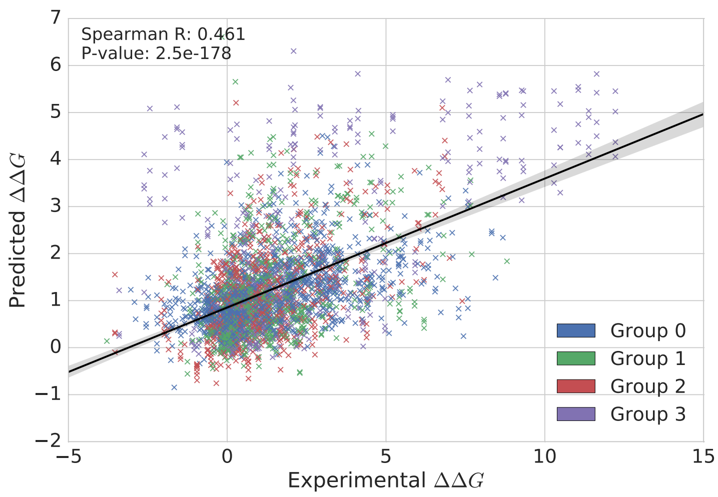


(b) Performance of the ELASPIC core predictor, FoldX and Provean on the validation datasets.

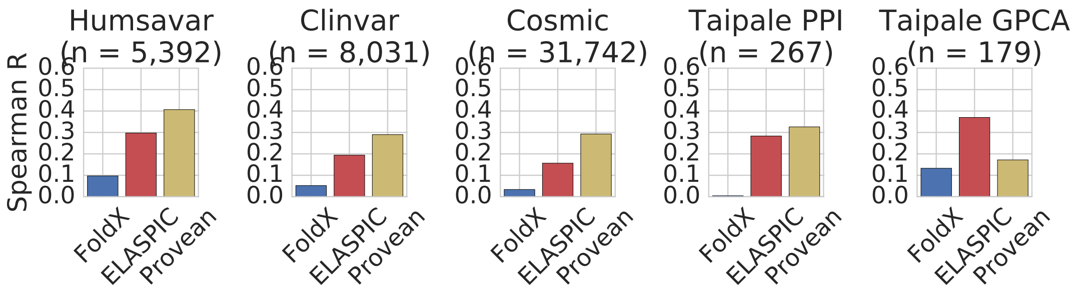


(c) Performance of the ELASPIC core predictor, FoldX and Provean on the test datasets. There is no overlap in mutations (or proteins for Humsavar, ClinVar and COSMIC) between the test datasets, and the training and validation datasets (see Figure 3.1).

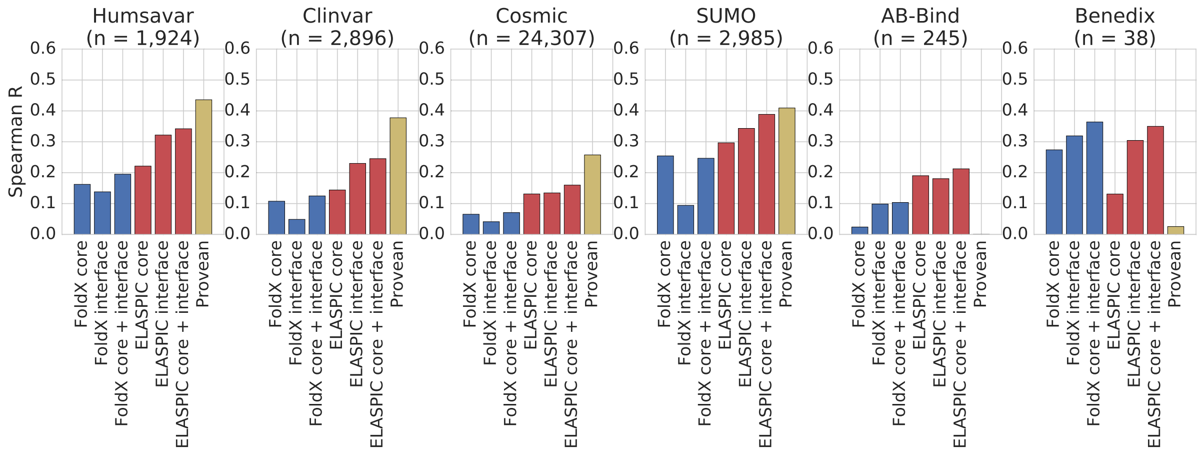
Figure 3.7: Performance of the ELASPIC core predictor on the training (a), validation (b) and test (c) datasets.



(a) Performance of the ELASPIC interface predictor on the training dataset, evaluated using four-fold cross-validation. Colours indicate different cross-validation bins.



(b) Performance of the ELASPIC interface predictor, FoldX and Provean on the validation datasets.



(c) Performance of ELASPIC, FoldX and Provean on the test datasets. Correlations are provided for core predictors, interface predictors, and the sum of core and interface predictors, for ELASPIC and FoldX. There is no overlap in mutations (or proteins for Humsavar, ClinVar and COSMIC) between the test datasets, and the training and validation datasets (see Figure 3.2).

Figure 3.8: Performance of the ELASPIC interface predictor on the training (a), validation (b) and test (c) datasets.

Chapter 4

Discussion

The results of this work are mixed.

The main goal of this work was to improve our ability to predict whether or not a mutations is going to be deleterious and predicting the phenotypic effect of mutations by evaluating the structural impact of mutations instead of looking solely at how conserved the particular is in different organisms and species.

In the set of features selected through feature elimination, there are both sequence-based features and structure-based features. The most important sequence-based feature is the Provean score.

Results of feature elimination support the view that electrostatics, van der Waals forces and entropy are the main forces determining the effect of mutations, as proposed by Benedix *et al.* in the Concoord/Poisson-Boltzmann surface area model (Equation 4.1).

$$\Delta G_{CC/PBSA} = \Delta G_{electrostatic} + \Delta G_{van\ der\ Waals} + \Delta G_{entropy} \tag{4.1}$$

Since the publication of the ELASPIC pipeline [16] and webserver [19], several other algorithms have been published which use a similar approach as ELASPIC to either predict mutation deleteriousness [42] or the $\Delta\Delta G$ [43].

4.1 Adding support for multiresidue mutations

ELASPIC can easily be extended to calculate the $\Delta\Delta G$ for mutations involving multiple amino acids. The main problem that would have to be solved

The tricky part is that the number of features changes with the number of amino acids that are mutated. We could address this by treating a mutation affecting multiple amino acids as a set of single amino acid mutations. For example, we could use the following recursive strategy:

- Introduce each of the single amino acid mutations, one at a time.
- Select the single amino acid mutation with the most stabilizing effect.
- Repeat for the remaining mutations, using the structure containing the mutation selected in Step 2.

About one third on mutations in the Protherm and Skempi databases affect multiple amino acids. We could include those mutations in the training set by dividing them into single amino acid mutations and assigning to them a $\Delta\Delta G$ proportional to their contribution to the overall mutation score, as determined by the multiple amino acid substitution version of ELASPIC. This would require "bootstrapping" the ELASPIC predictor using single amin acid mutations, using the "bootstrapped" predictor to approximate the contribution of single amino acid mutaitons to the $\Delta\Delta G$ affecting mulitple amino acids, adding those mutations to the training set, and repeating.

In the case of the ELASPIC core predictor, we could create a dataset of multiple amino acid polymorphisms (MAAMs) from a thermophilic bacterium and it's closest non-thermophilic relative (maybe such a database already exists?). Cross-validate ELASPIC making sure that we predict those MAAMs to be stabilizing. Incorporate those MAAMs into our training set, weighting them accordingly.

In the case of the ELASPIC interface predictor, we could construct a dataset from phage-display read counts, and cross-validate ELASPIC while keeping track of its performance on phage display counts. Could then recursively incorporate the phage display data into the training set, weighting it by how well the ELASPIC predictor does on those mutations, as determined through cross-validation.

It is likely that the performance of the ELASPIC predictor would be lower for mutations affecting multiple amino acids than for mutations affecting a single amino acids, as the former is more likely to induce changes in the conformation of the protein that are not modelled by ELASPIC. This drop in performance could in-part be ameliorated by including a backbone relaxation step between each mutation, using molecular dynamics [44], Rosetta Backrub [45], or other algorithms [46].

If the ELASPIC predictor can achieve reasonable results for mutations affecting multiple amino acids, it could be used "in reverse" to design protein domains with increased stability and protein interfaces with increased affinity.

4.2 Adding support for more interaction types

4.3 Multitask learning of mutation deleteriousness and energetic effects

In this work, we attempted to improve the generalizability of ELASPIC core and interface predictors by keeping track of their performance on mutation deleteriousness datasets throughout cross-validation and feature elimination (Figures 3.3, 3.4, 3.5 and 3.6). While this approach prevents us from selecting predictors that overfit the training set, it does not improve the accuracy of any individual predictor.

One way to improve overall accuracy of the predictors would be to leverage information contained in mutation deleteriousness datasets to discover better and more useful features. Mutation deleteriousness datasets are much larger than the $\Delta\Delta G$ datasets, and they may allow sequential and structural features to "mix" in a more general environment, and produce combined features that are less noisy and better correlated with the actual effect of mutations.

We could learn those features by first training a boosted decision tree algorithm to predict mutation deleteriousness, and then use the output of those trees as input to a logistic regression model trained to predict mutation $\Delta\Delta G$ (Figure 4.1). The resulting predictor should not only have better accuracy, but also have a better ability to extrapolate than the currently-used gradient boosted regressor algorithm, which never predicts values that are higher than the highest values observed in the training set.

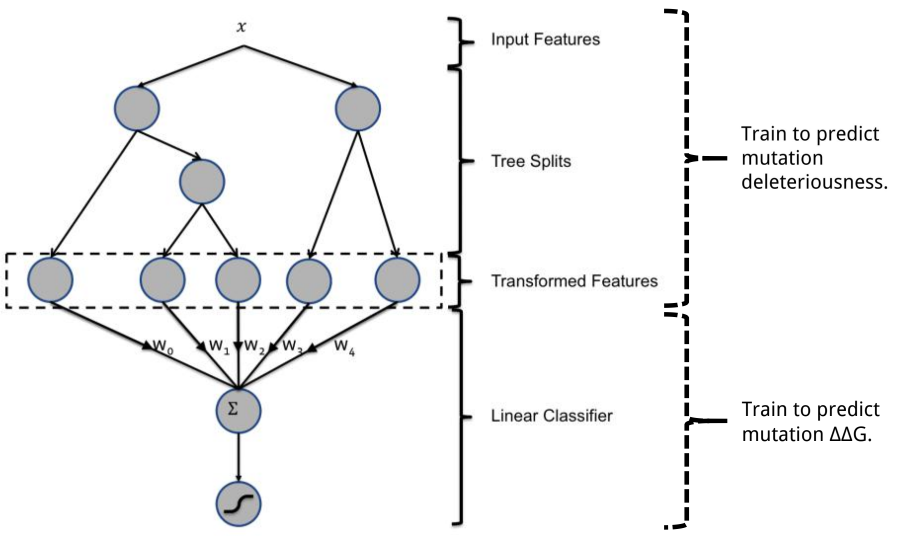


Figure 4.1: Multitask learning of mutation deleteriousness and $\Delta\Delta G$. The figure is adapted from He *et al.* [47], where it is used to describe an algorithm that couples boosted decision trees and linear regression to predict add click-trough rate. Boosted decision trees are used to learn a feature "manifold" that is provided as input to the linear classifier, which in turn makes the final predictions.
We propose to use a similar design, but train boosted decision trees to predict mutation deleteriousness, and fit a linear regressor to predict mutation $\Delta\Delta G$. We anticipate that the large training set of benign and deleterious mutations would allow the boosted decision tree algorithm to learn useful and generalizable features.

Bibliography

[1] KA. Wetterstrand. *DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP).* May 24, 2016.

[2] Caitlin C. Chrystoja and Eleftherios P. Diamandis. “Whole Genome Sequencing as a Diagnostic Test: Challenges and Opportunities”. In: *Clinical Chemistry* 60.5 (May 2014), pp. 724–733. DOI: `10.1373/clinchem.2013.209213`.

[3] Serena Nik-Zainal et al. “Landscape of Somatic Mutations in 560 Breast Cancer Whole-Genome Sequences”. In: *Nature* 534.7605 (June 2, 2016), pp. 47–54. DOI: `10.1038/nature17676`.

[4] Pauline C. Ng and Steven Henikoff. “SIFT: Predicting Amino Acid Changes that Affect Protein Function”. In: *Nucleic Acids Research* 31.13 (July 1, 2003), pp. 3812–3814.

[5] Yongwook Choi et al. “Predicting the Functional Effect of Amino Acid Substitutions and Indels”. In: *PLoS ONE* 7.10 (October 8, 2012). 00256, e46688. DOI: `10.1371/journal.pone.0046688`.

[6] Ivan Adzhubei et al. “Predicting Functional Effect of Human Missense Mutations Using PolyPhen-2”. In: *Current Protocols in Human Genetics.* John Wiley & Sons, Inc., 2001.

[7] Hashem A. Shihab et al. “Ranking Non-Synonymous Single Nucleotide Polymorphisms Based on Disease Concepts”. In: *Human Genomics* 8.1 (June 30, 2014). 00000, p. 11. DOI: `10.1186/1479-7364-8-11`.

[8] Biao Li et al. “Automated Inference of Molecular Mechanisms of Disease from Amino Acid Substitutions”. In: *Bioinformatics* 25.21 (January 11, 2009), pp. 2744–2750. DOI: `10.1093/bioinformatics/btp528`.

[9] The Cancer Genome Atlas Research Network. “Integrated Genomic Analyses of Ovarian Carcinoma”. In: *Nature* 474.7353 (June 30, 2011), pp. 609–615. DOI: `10.1038/nature10166`.

[10] Martin Kircher et al. “A General Framework for Estimating the Relative Pathogenicity of Human Genetic Variants”. In: *Nature Genetics* 46.3 (March 2014), pp. 310–315. DOI: `10.1038/ng.2892`.

[11] MichaelR. Shirts and DavidL. Mobley. “An Introduction to Best Practices in Free Energy Calculations”. In: *Biomolecular Simulations.* Ed. by Luca Monticelli and Emppu Salonen. Methods in Molecular Biology 924. Humana Press, January 1, 2013, pp. 271–311. DOI: `10.1007/978-1-62703-017-5_11`.

[12] Brett D. Welch et al. “Potent D-Peptide Inhibitors of HIV-1 Entry”. In: *Proceedings of the National Academy of Sciences* 104.43 (October 23, 2007), pp. 16828–16833. DOI: `10.1073/pnas.0708109104`.

[13] Douglas E. V. Pires et al. “mCSM: Predicting the Effects of Mutations in Proteins Using Graph-Based Signatures”. In: *Bioinformatics* 30.3 (January 2, 2014), pp. 335–342. DOI: `10.1093/bioinformatics/btt691`.

[14] Josef Laimer et al. “MAESTRO - Multi Agent Stability Prediction upon Point Mutations”. In: *BMC Bioinformatics* 16 (2015), p. 116. DOI: `10.1186/s12859-015-0548-6`.

[15] Alexander Benedix et al. “Predicting Free Energy Changes Using Structural Ensembles”. In: *Nature Methods* 6.1 (January 2009), pp. 3–4. DOI: `10.1038/nmeth0109-3`.

[16] Niklas Berliner et al. “Combining Structural Modeling with Ensemble Machine Learning to Accurately Predict Protein Fold Stability and Binding Affinity Effects upon Mutation”. In: *PLoS ONE* 9.9 (September 22, 2014), e107353. DOI: `10.1371/journal.pone.0107353`.

[17] Marco Punta et al. “The Pfam Protein Families Database”. In: *Nucleic Acids Research* 40 (D1 January 1, 2012). 00002, pp. D290–D301. DOI: `10.1093/nar/gkr1065`.

[18] Alison L. Cuff et al. “Extending CATH: Increasing Coverage of the Protein Structure Universe and Linking Structure with Function”. In: *Nucleic Acids Research* 39 (Database issue January 2011). 00100, pp. D420–D426. DOI: `10.1093/nar/gkq1001`.

[19] Daniel K. Witvliet et al. “ELASPIC Web-Server: Proteome-Wide Structure-Based Prediction of Mutation Effects on Protein Stability and Binding Affinity”. In: *Bioinformatics* 32.10 (May 15, 2016), pp. 1589–1591. DOI: `10.1093/bioinformatics/btw031`.

[20] Thomas Rattei et al. “SIMAP—a Comprehensive Database of Pre-Calculated Protein Sequence Similarities, Domains, Annotations and Clusters”. In: *Nucleic Acids Research* 38 (suppl 1 January 1, 2010). 00031, pp. D223–D226. DOI: `10.1093/nar/gkp949`.

[21] Robert C. Edgar. “MUSCLE: A Multiple Sequence Alignment Method with Reduced Time and Space Complexity”. In: *BMC Bioinformatics* 5.1 (August 19, 2004). 02783, p. 113. DOI: `10.1186/1471-2105-5-113`.

[22] Martin H. Schaefer et al. “HIPPIE: Integrating Protein Interaction Networks with Experiment Based Quality Scores”. In: *PLoS ONE* 7.2 (February 14, 2012), e31826. DOI: `10.1371/journal.pone.0031826`.

[23] Thomas Rolland et al. “A Proteome-Scale Map of the Human Interactome Network”. In: *Cell* 159.5 (November 20, 2014). 00006, pp. 1212–1226. DOI: `10.1016/j.cell.2014.10.050`.

[24] Jean-François Rual et al. “Towards a Proteome-Scale Map of the Human Protein–protein Interaction Network”. In: *Nature* 437.7062 (October 20, 2005). 02009, pp. 1173–1178. DOI: `10.1038/nature04209`.

[25] Haiyuan Yu et al. “Next-Generation Sequencing to Generate Interactome Datasets”. In: *Nature Methods* 8.6 (June 2011). 00070, pp. 478–480. DOI: `10.1038/nmeth.1597`.

[26] Kavitha Venkatesan et al. “An Empirical Framework for Binary Interactome Mapping”. In: *Nature Methods* 6.1 (January 2009). 00427, pp. 83–90. DOI: `10.1038/nmeth.1280`.

[27] Benjamin Webb and Andrej Sali. “Comparative Protein Structure Modeling Using MODELLER”. In: *Current Protocols in Bioinformatics*. John Wiley & Sons, Inc., 2002.

[28] Joost Schymkowitz et al. “The FoldX Web Server: An Online Force Field”. In: *Nucleic Acids Research* 33 (suppl 2 January 7, 2005), W382–W388. DOI: `10.1093/nar/gki387`.

[29] F. Pedregosa et al. "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[30] Hongbo Zhu et al. "NOXclass: Prediction of Protein-Protein Interaction Types". In: *BMC Bioinformatics* 7.1 (January 19, 2006), p. 27. DOI: 10.1186/1471-2105-7-27.

[31] K. Abdulla Bava et al. "ProTherm, Version 4.0: Thermodynamic Database for Proteins and Mutants". In: *Nucleic Acids Research* 32 (suppl 1 January 1, 2004). 00169, pp. D120–D121. DOI: 10.1093/nar/gkh082.

[32] Elizabeth H. Kellogg et al. "Role of Conformational Sampling in Computing Mutation-Induced Changes in Protein Structure and Stability". In: *Proteins* 79.3 (March 2011). 00094, pp. 830–838. DOI: 10.1002/prot.22921.

[33] Iain H. Moal and Juan Fernández-Recio. "SKEMPI: A Structural Kinetic and Energetic Database of Mutant Protein Interactions and Its Use in Empirical Models". In: *Bioinformatics* 28.20 (October 15, 2012), pp. 2600–2607. DOI: 10.1093/bioinformatics/bts489.

[34] Tanja Kortemme and David Baker. "A Simple Physical Model for Binding Energy Hot Spots in Protein–protein Complexes". In: *Proceedings of the National Academy of Sciences of the United States of America* 99.22 (October 29, 2002), pp. 14116–14121. DOI: 10.1073/pnas.202485799.

[35] M. D. Shaji Kumar et al. "ProTherm and ProNIT: Thermodynamic Databases for Proteins and Protein–nucleic Acid Interactions". In: *Nucleic Acids Research* 34 (suppl 1 January 1, 2006), pp. D204–D206. DOI: 10.1093/nar/gkj103.

[36] Nidhi Sahni et al. "Widespread Macromolecular Interaction Perturbations in Human Genetic Disorders". In: *Cell* 161.3 (April 23, 2015), pp. 647–660. DOI: 10.1016/j.cell.2015.04.013.

[37] The UniProt Consortium. "UniProt: A Hub for Protein Information". In: *Nucleic Acids Research* 43 (D1 January 28, 2015), pp. D204–D212. DOI: 10.1093/nar/gku989.

[38] Melissa J. Landrum et al. "ClinVar: Public Archive of Interpretations of Clinically Relevant Variants". In: *Nucleic Acids Research* 44 (D1 April 1, 2016), pp. D862–D868. DOI: 10.1093/nar/gkv1222.

[39] Simon A. Forbes et al. "COSMIC: Exploring the World's Knowledge of Somatic Mutations in Human Cancer". In: *Nucleic Acids Research* 43 (D1 January 28, 2015), pp. D805–D811. DOI: 10.1093/nar/gku1075.

[40] J. Weile et al. "An Atlas of Functional Amino Acid Changes in Human SUMO and SUMO Ligase." In: (In preparation).

[41] Sarah Sirin et al. "AB-Bind: Antibody Binding Mutational Database for Computational Affinity Predictions". In: *Protein Science* 25.2 (February 1, 2016), pp. 393–409. DOI: 10.1002/pro.2829.

[42] Evan H. Baugh et al. "Robust Classification of Protein Variation Using Structural Modelling and Large-Scale Data Integration". In: *Nucleic Acids Research* 44.6 (July 4, 2016), pp. 2501–2513. DOI: 10.1093/nar/gkw120.

[43] Minghui Li et al. "MutaBind Estimates and Interprets the Effects of Sequence Variants on Protein–protein Interactions". In: *Nucleic Acids Research* 44 (W1 August 7, 2016), W494–W501. DOI: 10.1093/nar/gkw374.

[44] Mark James Abraham et al. "GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers". In: *SoftwareX* 1–2 (September 2015), pp. 19–25. DOI: 10.1016/j.softx.2015.06.001.

[45] Colin A. Smith and Tanja Kortemme. "Predicting the Tolerated Sequences for Proteins and Protein Interfaces Using RosettaBackrub Flexible Backbone Design". In: *PLOS ONE* 6.7 (July 18, 2011), e20451. DOI: 10.1371/journal.pone.0020451.

[46] Mark G. F. Sun et al. "Protein Engineering by Highly Parallel Screening of Computationally Designed Variants". In: *Science Advances* 2.7 (July 1, 2016), e1600692. DOI: 10.1126/sciadv.1600692.

[47] Xinran He et al. "Practical Lessons from Predicting Clicks on Ads at Facebook". In: *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising.* ADKDD'14. New York, NY, USA: ACM, 2014, 5:1–5:9. DOI: 10.1145/2648584.2648589.