ELASPIC Documentation

Release 1.0

kimlab

February 01, 2016

Contents

CHAPTER 1

Introduction

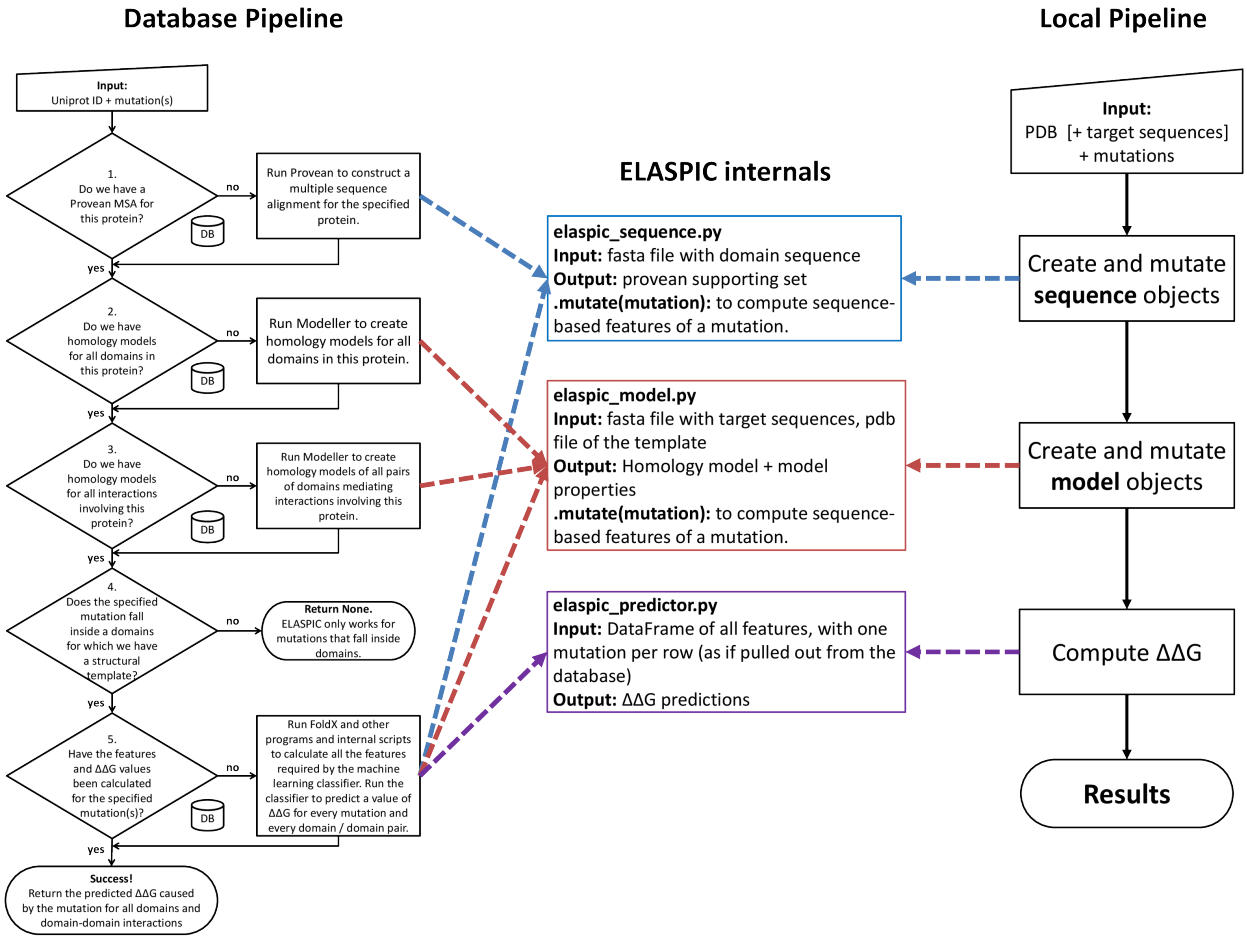
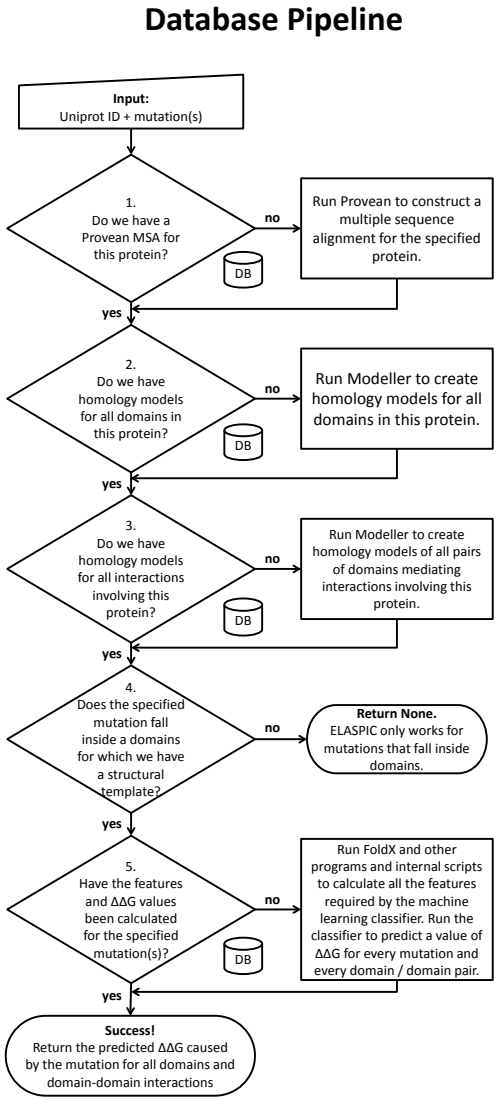


Fig. 1.1: Flowchart describing the ELASPIC pipeline.

ELASPIC can be run using two different pipelines: the *Local pipeline* and the *Database pipeline*.

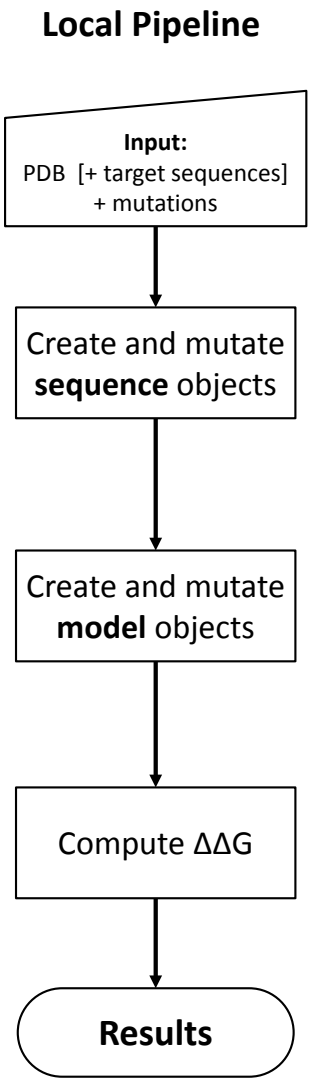
1.1 Database pipeline



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 copy of *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 in the `figure` to the right. 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 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.

1.2 Local pipeline



The local pipeline works without downloading and installing a local copy of the ELASPIC and PDB databases, but requires a PDB structure or template to be provided for every protein. Pipeline output is saves as *JSON* files inside the working directory, rather than being uploaded to the database as in the case of the database pipeline. The general overview of the local pipleine is presented in the `figure` to the right.

The local pipeline still requires a local copy of the *Blast* nr database.

CHAPTER 2

Installation Guide

In order to use the ELASPIC *Local pipeline* of your computer:

1. Install Python and ELASPIC (*Installing Python and ELASPIC*).
2. Download the BLAST database and preferrably also the PDB database to a local folder (*Downloading external datasets*).

In order to use the ELASPIC *Database pipeline*, in addition to the steps above:

1. Create a local database and modify the configuration file to match your system and database setting (*Updating the configuration file*).
2. Download Profs domain definitions for your organism of interest, and upload the data to a local database (*Importing precalculated data*).

2.1 Installing Python and ELASPIC

1. Download and install the Anaconda Python Distribution (Python 3) for Linux.
2. Add `bioconda`, `salilab`, and `ostrokach` channels to your ~/.condarc file:

```
conda config --add channels ostrokach
conda config --add channels salilab
conda config --add channels bioconda
```

3. Obtain a Modeller license, and export the license as `KEY_MODELLER` in your ~/.bashrc file:

```
# ~/.bashrc
export KEY_MODELLER=XXXXXXX
```

4. Install ELASPIC and all its dependencies into a new conda environment:

```
conda create -n elaspic elaspic
```

5. Activate the new environment and use elaspic:

```
source activate elaspic
elaspic --help
```

2.2 Downloading external datasets

2.2.1 Blast

Download and extract the *nr* and *pdbaa* databases from ftp://ftp.ncbi.nlm.nih.gov/blast/db/, and change the *blast_db_dir* variable in your *configuration file* to point to the directory containing the uncompressed files.

2.2.2 PDB

Download the contents of the ftp://ftp.wwpdb.org/pub/pdb/data/structures/divided/pdb/ folder, and change the *pdb_dir* variable in your *configuration file* to point to the directory containing the downloaded data.

2.3 Updating the configuration file

Edit the ELASPIC configuration file ./config/config_file.ini to match your system:

1. Settings in the *[SEQUENCE]* section should be modified to match the location of your local BLAST and PDB databases.
2. Settings in the *[DATABASE]* section should be modified to match the local MySQL, PostgreSQL, or SQLite database.
3. Settings in the *[DEFAULT]* and *[MODEL]* may be left unchanged, since the default values are good enough in most cases.

2.3.1 Configuration options

[DEFAULT]

global_temp_dir Location for storing temporary files. It will be used only if the `TMPDIR` environmental variable is not set. **Default = '/tmp/'**.

temp_dir string A folder in the *global_temp_dir* that will contain all the files that are relevant to ELASPIC. Inside this folder, every job will create its own unique subfolder. **Default = 'elaspic/'**.

debug Whether or not to show detailed debugging information. If True, the logging level will be set to `logging.DEBUG`. If False, the logging level will be set to `logging.INFO`. **Default = True**.

look_for_interactions Whether or not to compute models of protein-protein interactions. **Default = True**.

remake_provean_supset Whether or not to remake the Provean supporting set if one or more sequences cannot be found in the BLAST database. **Default = False**.

n_cores Number of cores to use by programs that support multithreading. **Default = 1**.

web_server Whether or not the ELASPIC pipeline is being run as part of a webserver. **Default = False**.

provean_temp_dir Location to store provean temporary files if working on any note other than *beagle* or *banting*. For internal use only. **Default = ''**.

copy_data Whether or not to copy calculated data back to the archive. Set to 'False' if you are planning to copy the data yourself (e.g. from inside a PBS or SGE script). **Default = True**.

[SEQUENCE]

blast_db_dir Location of the blast **nr** and **pdbaa** databases.

blast_db_dir_fallback Place to look for blast **nr** and **pdbaa** databases if *blast_db_dir* does not exist.

matrix_type Substitution matrix for calculating the mutation conservation score. **Default = 'blosum80'**.

gap_start Penalty for starting a gap when calculating the mutation conservation score. **Default = -16**.

gap_extend Penalty for extending a gap when calculating the mutation conservation score. **Default = -4**.

[MODEL]

modeller_runs Number of models that MODELLER should make before choosing the best one. Not implemented! **Default = 1**.

foldx_water

- `-CRYSTAL`: use water molecules in the crystal structure to bridge two protein atoms.
- `-PREDICT`: predict water molecules that make 2 or more hydrogen bonds to the protein.
- `-COMPARE`: compare predicted water bridges with bridges observed in the crystal structure.
- `-IGNORE`: don't predict water molecules. **Default**.

Source: http://foldx.crg.es/manual3.jsp.

foldx_num_of_runs Number of times that FoldX should evaluate a given mutation. **Default = 1**.

[DATABASE]

db_type The database that you are using. Supported databases are *MySQL*, *PostgreSQL*, and *SQLite*.

sqlite_db_dir Location of the SQLite database. Required only if *db_type* is *SQLite*.

db_schema The name of the schema that holds all elaspic data.

db_schema_uniprot The name of the database schema that holds uniprot sequences. Defaults to *db_schema*.

db_database The name of the database that contains *db_schema* and *db_schema_uniprot*. Required only if *db_type* is *PostgreSQL*. Defaults to *db_schema*.

db_username The username for the database. Required only if *db_type* is *MySQL* or *PostgreSQL*.

db_password The password for the database. Required only if *db_type* is *MySQL* or *PostgreSQL*.

db_url The IP address of the database. Required only if *db_type* is *MySQL* or *PostgreSQL*.

db_port The listening port of the database. Required only if *db_type* is *MySQL* or *PostgreSQL*.

db_socket Path to the socket file, if it is not in the default location. Used only if *db_url* is *localhost*. For example: `/usr/local/mysql5/mysqld.sock` for *MySQL* and `/var/lib/postgresql` for *PostgreSQL*.

schema_version Database schema to use for storing and retreiving data. **Default = 'elaspic'**.

archive_type

- extracted: all archive files are contained in an extracted directory tree.
- 7zip: archive is made of three compressed 7zip files (provean/provean.7z, uniprot_domain/uniprot_domain.7z, uniprot_domain_pair/uniprot_domain_pair.7z), provided on the elaspic downloads page.

archive_dir Location for storing and retrieving precalculated data.

pdb_dir Location of all pdb structures, equivalent to the "data/data/structures/divided/pdb/" folder in the PDB ftp site. Optional.

2.3.2 Environmental variables

PATH
A colon-separated list of paths where ELASPIC should look for required programs, such as BLAST, T-coffee, Modeller, and cd-hit.

TMPDIR
Location to store all temporary files and folders.

2.4 Importing precalculated data

2.4.1 ELASPIC downloads page

The ELASPIC downloads page contains all precalculated data that is required to run the ELASPIC pipeline on a local machine.

The `*.tsv.gz` files correspond to different tables of the *ELASPIC database*:

- The `domain.tar.gz` file in the root folder contains Profs domain definitions for files in the PDB, and corresponds to the *domain* table.
- The `domain_contact.tar.gz` file in the root folder contains a list of interactions between those domains, and corresponds to the *domain_contact* table.
- All other tables are split into separate folders according to the organism of origin. The files are named using the `{table_name}.tsv.gz` convention, where `table_name` is the name of the table in the database.

The `*.7z` files contain precalculated data:

- The *provean*, *uniprot_domain*, and *uniprot_domain_pair* subfolders contain precalculated provean supporting sets, and homology models of protein domains and domain-domain interactions, respectively.

Precalculated mutations:

- The *Homo_sapiens* folder contains an additional subfolder precalculated_mutations, which contains $\Delta\Delta G$ scores for mutations in various datasets.

Note: The configure_test.sh and run_test.sh scripts in the *./scripts* folder contain examples of how to download and set up a local copy of the database.

2.4.2 Downloading data

In order to run up ELASPIC on a local computer, you need to download precalculated data for your organism of interest. If your goal is to only test the pipeline, you can download a test dataset from the folder current_release/Homo_sapiens_test.

To download all precalculated data for a given organism, use the `wget` command:

```
# Download external files
wget -P "${TEST_DIR}/elaspic.kimlab.org" \
    http://elaspic.kimlab.org/static/download/current_release/domain.tsv.gz
wget -P "${TEST_DIR}/elaspic.kimlab.org" \
    http://elaspic.kimlab.org/static/download/current_release/domain_contact.tsv.gz
wget -P "${TEST_DIR}" \
    -r --no-parent --reject "index.html*" --cut-dirs=4  \
    http://elaspic.kimlab.org/static/download/current_release/Homo_sapiens_test/
```

You need to extract the provean supporting sets and domain homology models into a folder specified by the *archive_dir* variable in your *configuration_file*:

```
mkdir archive  # Set 'archive_dir' variable in the config file to this folder

7z x "${TEST_DIR}/elaspic.kimlab.org/provean/provean.7z" -o"archive"
7z x "${TEST_DIR}/elaspic.kimlab.org/uniprot_domain/uniprot_domain.7z" -o"archive"
7z x "${TEST_DIR}/elaspic.kimlab.org/uniprot_domain_pair/uniprot_domain_pair.7z" -o"archive"
```

2.4.3 Importing data into a database

You also need to create a local SQL database and fill it with precalculated data.

Modify the database variables in the ELASPIC *configuration file* to match your local *MySQL*, *PostgreSQL*, or *SQLite* database, and use the *elaspic database* CLI to create a new database and fill it with precalculated data.

First, you need to create an empty database:

```
elaspic database -c {your_configuration_file}.ini create
```

Next, you need to load all precalculated data for the organism in question to your database:

```
elaspic database -c {your_configuration_file}.ini load_data
```

To delete the database that you just created, run:

```
elaspic database -c {your_configuration_file}.ini delete
```

CHAPTER 3

Command Line Interface

After following instructions in the *Installation Guide*, you should be able to run ELASPIC from the command line using the `elaspic` command:

```
$ elaspic --help
usage: elaspic [-h] {run,database,train} ...

optional arguments:
  -h, --help            show this help message and exit

command:
  {run,database,train}
    run                 Run ELASPIC.
    database            Perform maintenance tasks on the ELASPIC database.
    train               Train the ELASPIC classifiers.
```

Type `--help` to see the options available for each subcommand:

- `elaspic run --help`
- `elaspic database --help`
- `elaspic database load_data --help`
- etc...

3.1 elaspic run

Run the ELASPIC pipeline.

If you wish to mutate an existing PDB, you should specify the name of the PDB file to be mutated, and the mutation(s):

```
elaspic run \
    --structure_file {structure_file} \
    --mutations {mutations}
```

If you wish to first create a homology model of a protein, you should provide a fasta file containing the sequence of the protein to be modelled, a PDB file containing the structural template, and the mutation(s):

```
elaspic run \
    --sequence_file {sequence_file} \
    --structure_file {structure_file} \
    --mutations {mutations}
```

If you wish to perform mutagenesis on a proteome-wide scale, you need to download protein domain definitions from the elaspic downloads page, and optionally a local copy of the PDB database. After saving your database information to a configuration file, you can run specify the uniprot id and mutation(s):

```
elaspic run \
  --config_file {config_file} \
  --uniprot_id {uniprot_id} \
  --mutations {mutations}
```

3.2 elaspic train

Train the machine learning predictor for the ELASPIC pipeline.

This is automatically done at install time, and you *do not* need to do this again unless you update your `scikit-learn` version.

3.3 elaspic database

Perform maintenance tasks on the ELASPIC database.

You must provide a configuration file containing the details of your database installation for any of these commands to work. For more information about configuration files, see *Updating the configuration file*.

3.3.1 elaspic database create

Create a new database schema.

3.3.2 elaspic database load_data

Load data to the database.

3.3.3 elaspic database delete

Delete the database schema.

CHAPTER 4

Benchmarks

Work in progres...

CHAPTER 5

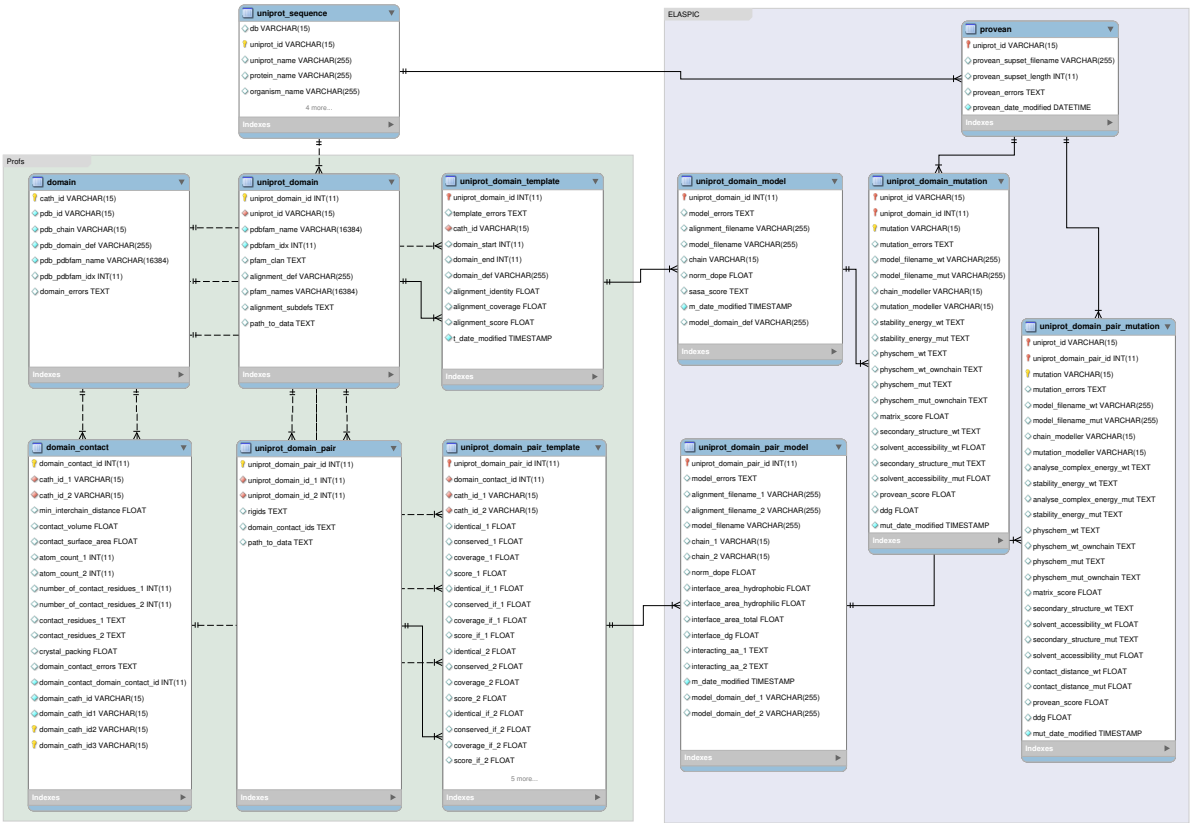
Statistics

Work in progres...

CHAPTER 6

Database

6.1 Database schema



6.2 Database tables

6.2.1 domain

Profs domain definitions for all proteins in the PDB.

Columns:

cath_id Unique id identifying each domain in the PDB. Constructed by concatenating the pdb_id, pdb_chain, and an index specifying the order of the domain in the chain.

pdb_id The PDB id in which the domain is found.

pdb_chain The PDB chain in which the domain is found.

pdb_domain_def Domain definitions of the domain, in PDB RESNUM coordinates.

pdb_pdbfam_name The Profs name of the domain.

pdb_pdbfam_idx An integer specifying the number of times a domain with domain name `pdb_pdbfam_name` has occurred in this chain up to this point. It is used to make every `(pdb_id, pdb_chain, pdb_pdbfam_name, pdb_pdbfam_idx)` tuple unique.

domain_errors List of errors that occurred when annotating this domain, or when using this domain to make structural homology models.

6.2.2 domain_contact

Interactions between Profs domains in the PDB. Only interactions that were predicted to be biologically relevant by NOXclass are included in this table.

Columns:

domain_contact_id A unique integer identifying each domain pair.

cath_id_1 Unique id identifying the first interacting domain in the *domain* table.

cath_id_2 Unique id identifying the second interacting domain in the *domain* table.

min_interchain_distance The closest that any residue in domain one comes to any residue in domain two.

contact_volume The volume covered by contacting residues.

contact_surface_area The surface area of the contacting regions of the first and second domains.

atom_count_1 The number of atoms in the first domain.

atom_count_2 The number of atoms in the second domain.

number_of_contact_residues_1 The number of residues in the first domain that come within 5 of the second domain.

number_of_contact_residues_2 The number of residues in the second domain that come withing 5 of the first domain.

contact_residues_1 A list of all residues in the first domain that come within 5 of the second domain. The residue number corresponds to the position of the residue in the domain.

contact_residues_2 A list of all residues in the second domain that come within 5 of the first domain. The residue number corresponds to the position of the residue in the domain.

crystal_packing The probability that the interaction is a crystallization artifacts, as defined by NOXclass.

domain_contact_errors List of errors that occurred when annotating this domain pair, or when using this domain as a template for making structural homology models.

6.2.3 uniprot_sequence

Protein sequences from the Uniprot KB, obtained by parsing *uniprot_sprot_fasta.gz*, *uniprot_trembl_fasta.gz*, and *homo_sapiens_variation.txt* files from the Uniprot ftp site.

Columns:

db The database to which the protein sequence belongs. Possible values are *sp* for SwissProt and *tr* for TrEMBL.

uniprot_id The uniprot id of the protein.

uniprot_name The uniprot name of the protein.

protein_name The protein name.

organism_name Name of the organism in which this protein is found.

gene_name Name of the gene that codes for this protein sequence.

protein_existence Evidence for the existence of the protein:

1. Experimental evidence at protein level
2. Experimental evidence at transcript level
3. Protein inferred from homology
4. Protein predicted
5. Protein uncertain

sequence_version Version of the protein amino acid sequence.

uniprot_sequence Amino acid sequence of the protein.

6.2.4 provean

Description of the Provean supporting set calculated for a protein sequence. The construction of a supporting set is the most lengthy step in running Provean. Therefore, the supporting set is precalculated and stored for every protein sequence.

Columns:

uniprot_id The uniprot id of the protein.

provean_supset_filename The filename of the Provean supporting set. The supporting set contains the ids and sequences of all proteins in the NCBI nr database that are used by Provean to construct a multiple sequence alignment for the given protein.

provean_supset_length The number of sequences in Provean supporting set.

provean_errors List of errors that occurred while the Provean supporting set was being calculated.

provean_date_modified Date and time that this row was last modified.

6.2.5 uniprot_domain

Pfam domain definitions for proteins in the *uniprot_sequence* table. This table was obtained by downloading Pfam domain definitions for all known proteins from the SIMAP website, and mapping the protein sequence to uniprot using the MD5 hash of each sequence.

Columns:

uniprot_domain_id Unique id identifying each domain.

uniprot_id The uniprot id of the protein containing the domain.

pdbfam_name The Profs name of the domain. In most cases this will be equivalent to the Pfam name of the domain.

pdbfam_idx The index of the Profs domain. `pdbfam_idx` ranges from 1 to the number of domains with the name `pdbfam_name` in the given protein. The `(pdbfam_name, pdbfam_idx)` tuple uniquely identifies each domain.

pfam_clan The Pfam clan to which this Profs domain belongs.

alignment_def Alignment domain definitions of the Profs domain. This field is obtained by removing gaps in the `alignment_subdefs` column.

pfam_names Pfam names of all Pfam domains that were combined to create the given Profs domain.

alignment_subdefs Comma-separated list of domain definitions for all Pfam domains that were merged to create the given Profs domain.

path_to_data Location for storing homology models, mutation results, and all other data that are relevant to this domain. This path is prefixed by *archive_dir*.

6.2.6 uniprot_domain_template

Structural templates for domains in the *uniprot_domain* table. Lists PDB crystal structures that will be used for making homology models.

Columns:

uniprot_domain_id An integer which uniquely identifies each uniprot domain in the *uniprot_domain* table.

template_errors List of errors that occurred during the process for finding the template.

cath_id The unique id identifying the structural template of the domain.

domain_start The Uniprot position of the first amino acid of the Profs domain.

domain_end The Uniprot position of the last amino acid of the Profs domain.

domain_def Profs domain definitions for domains with structural templates. Domain definitions in this column are different from domain definitions in the `alignment_def` column of the *uniprot_domain* table in that they have been expanded to match domain boundaries of the Profs structural template, identified by the `cath_id`.

alignment_identity Percent identity of the domain to its structural template.

alignment_coverage Percent coverage of the domain to its structural template.

alignment_score A score obtained by combining `alignment_identity` ($SeqId$) and `alignment_coverage` (Cov) using the following equation, as described by Mosca et al.:

$$Score = 0.95 \cdot \frac{SeqId}{100} \cdot \frac{Cov}{100} + 0.05 \cdot \frac{Cov}{100} \tag{6.1}$$

t_date_modified The date and time when this row was last modified.

6.2.7 uniprot_domain_model

Homology models for templates in the *uniprot_domain_template* table.

Columns:

uniprot_domain_id An integer which uniquely identifies each uniprot domain in the *uniprot_domain* table.

model_errors List of errors that occurred when making the homology model.

alignment_filename The name of the alignment that was given to Modeller when making the homology model.

model_filename The name of the homology model that was produced by Modeller.

chain The chain that contains the domain in question in the homology (this is now set to 'A' in all models).

norm_dope Normalized DOPE score of the model (lower is better).

sasa_score Comma-separated list of the percent solvent-accessible surface area for each residue.

m_date_modified The date and time when this row was last modified.

model_domain_def Domain definitions for the region of the domain that is covered by the structural template.

In most cases, this field is identical to the `domain_def` field in the *uniprot_domain_template* table. However, it sometimes happens that the best Profs structural template only covers a fraction of the Pfam domain. In that case, the `alignment_def` column in the *uniprot_domain* table, and the `domain_def` column in the *uniprot_domain_template* table, will contain the original Pfam domain definitions, and the `model_domain_def` column will contain domain definitions for only the region that is covered by the structural template.

6.2.8 uniprot_domain_mutation

Characterization of mutations introduced into structures in the *uniprot_domain_model* table.

Columns:

uniprot_id Uniprot ID of the protein that was mutated.

uniprot_domain_id Unique id which identifies the Profs domain that was mutated in the *uniprot_domain* table.

mutation Mutation that was introduced into the protein, in Uniprot coordinates.

mutation_errors List of errors that occured while evaluating the mutation.

model_filename_wt The name of the file which contains the homology model of the domain after the model was relaxed with FoldX but before the mutation was introduced.

model_filename_mut The name of the file which contains the homology model of the domain after the model was relaxed with FoldX and after the mutation was introduced.

chain_modeller The chain which contains the domain that was mutated in the `model_filename_wt` and the `model_filename_mut` structures.

mutation_modeller The mutation that was introduced into the protein, in PDB RESNUM coordinates. This identifies the mutated residue in the `model_filename_wt` and the `model_filename_mut` structures.

stability_energy_wt Comma-separated list of scores returned by FoldX for the wildtype protein. The comma-separated list can be converted into a DataFrame with each column clearly labelled using the `elaspic.predictor.format_mutation_features()`. The FoldX energy terms are:

- dg
- backbone_hbond
- sidechain_hbond
- van_der_waals
- electrostatics
- solvation_polar

- solvation_hydrophobic
- van_der_waals_clashes
- entropy_sidechain
- entropy_mainchain
- sloop_entropy
- mloop_entropy
- cis_bond
- torsional_clash
- backbone_clash
- helix_dipole
- water_bridge
- disulfide
- electrostatic_kon
- partial_covalent_bonds
- energy_ionisation
- entropy_complex
- number_of_residues

stability_energy_mut Comma-separated list of scores returned by FoldX for the mutant protein. FoldX energy terms are the same as in *stability_energy_wt*, but for the mutated amino acid rather than the wildtype.

physchem_wt Physicochemical properties describing the interaction of the wildtype residue with residues on the opposite chain. The terms are:

- number of atoms in interacting residues that have the same charge.
- number of atoms in interacting residues that have an opposite charge.
- number of hydrogen bonds (very rough calculation).
- number of carbons in interacting residues within 4 A of the mutated residue (rough measure of the van der Waals force).

physchem_wt_ownchain Physicochemical properties describing the interaction of the wildtype residue with residues on the same chain. The terms are the same as in *physchem_wt*.

physchem_mut Physicochemical properties describing the interaction of the mutant residue with residues on the opposite chain. The terms are the same as in *physchem_wt*.

physchem_mut_ownchain Physicochemical properties describing the interaction of the mutant residue with residues on the same chain. The terms are the same as in *physchem_wt*.

matrix_score Score assigned to the wt -> mut transition by the BLOSUM substitution matrix.

secondary_structure_wt Secondary structure of the wildtype residue predicted by stride.

solvent_accessibility_wt Percent solvent accessible surface area of the wildtype residue, predicted by msms.

secondary_structure_mut Secondary structure of the mutated residue predicted by stride.

solvent_accessibility_mut Percent solvent accessible surface area of the mutated residue, predicted by msms.

provean_score Score produced by Provean for this mutation.

ddg Change in the Gibbs free energy of folding that our classifier predicts for this mutation.

mut_date_modified Date and time that this row was last modified.

6.2.9 uniprot_domain_pair

Potentially-interacting pairs of domains for proteins that are known to interact, according to Hippie, IRefIndex, and Rolland et al. 2014.

Columns:

uniprot_domain_pair_id Unique id identifying each domain-domain interaction.

uniprot_domain_id_1 Unique id of the first domain.

uniprot_domain_id_2 Unique id of the second domain.

rigids Phased out.

domain_contact_ids List of unique ids identifying all domain-domain pairs in the PDB, where one domain belongs to the protein containing `uniprot_domain_id_1` and the other domain belongs to the protein containing `uniprot_domain_id_2`. This was used as crystallographic evidence that the two proteins interact.

path_to_data Location for storing homology models, mutation results, and all other data that is relevant to this domain pair. This path is prefixed by *archive_dir*.

6.2.10 uniprot_domain_pair_template

Structural templates for pairs of domains in the *uniprot_domain_pair* table.

Columns:

uniprot_domain_pair_id Unique id identifying each domain-domain interaction.

domain_contact_id Unique id of the domain pair in the *domain_contact* table that was used as a template for the modelled domain pair.

cath_id_1 Unique id of the structural template for the first domain.

cath_id_2 Unique id of the structural template for the second domain.

identical_1 Fraction of residues in the Blast alignment of the first domain to its template that are *identical*.

conserved_1 Fraction of residues in the Blast alignment of the first domain to its template that are *conserved*.

coverage_1 Fraction of the first domain that is covered by the blast alignment.

score_1 Score obtained by multiplying `identical_1` by `coverage_1`.

identical_if_1 Fraction of interface residues [1] that are *identical* in the Blast alignment of the first domain.

conserved_if_1 Fraction of interface residues [1] that are *conserved* in the Blast alignment of the first domain.

coverage_if_1 Fraction of interface residues [1] that are *covered* by the Blast alignment of the first domain.

score_if_1 Score obtained by combining `identical_if_1` and `coverage_if_1` using (6.1).

identical_2 Fraction of residues in the Blast alignment of the second domain to its template that are *identical*.

conserved_2 Fraction of residues in the Blast alignment of the second domain to its template that are *conserved*.

coverage_2 Fraction of the second domain that is covered by the blast alignment.

[1] Interface residues are defined as residues that are within 5 of the partner domain.

score_2 Score obtained by multiplying `identical_2` by `coverage_2`.

identical_if_2 Fraction of interface residues [1] that are *identical* in the Blast alignment of the second domain.

conserved_if_2 Fraction of interface residues [1] that are *conserved* in the Blast alignment of the second domain.

coverage_if_2 Fraction of interface residues [1] that are *covered* by the Blast alignment of the second domain.

score_if_2 Score obtained by combining `identical_if_2` and `coverage_if_2` using (6.1).

score_total The product of `score_1` and `score_2`.

score_if_total The product of `score_if_1` and `score_if_2`.

score_overall The product of `score_total` and `score_if_total`. This is the score that was used to select the best Profs domain pair to be used as a template.

t_date_modified The date and time when this row was last updated.

template_errors List of errors that occured while looking for the structural template.

6.2.11 uniprot_domain_pair_model

Structural models of interactions between pairs of domains in the *uniprot_domain_pair* table.

Columns:

uniprot_domain_pair_id Unique id identifying each domain-domain interaction.

model_errors List of errors that occured while making the homology model.

alignment_filename_1 Name of the file containing the alignment of the first domain with its structural template.

alignment_filename_2 Name of the file containing the alignment of the second domain with its structural template.

model_filename Name of the file containing the homology model of the domain-domain interaction created by Modeller.

chain_1 Chain containing the first domain in the model specified by `model_filename`.

chain_2 Chain containing the second domain in the model specified by `model_filename`.

norm_dope The normalized DOPE score of the model.

interface_area_hydrophobic Hydrophobic surface area of the interface, calculated using POPS.

interface_area_hydrophilic Hydrophilic surface area of the interface, calculated using POPS.

interface_area_total Total surface area of the interface, calculated using POPS.

interface_dg Gibbs free energy of binding for this domain-domain interaction, predicted using FoldX. Not implemented yet!

interacting_aa_1 List of amino acid positions in the first domain that are within 5 of the second domain. Positions are specified using uniprot coordinates.

interacting_aa_2 List of amino acids in the second domain that are within 5 of the first domain. Position are specified using uniprot coordinates.

m_date_modified Date and time that this row was last modified.

model_domain_def_1 Domain boundaries of the first domain that are covered by the Profs structural template.

model_domain_def_2 Domain boundaries of the second domain that are covered by the Profs structural template.

6.2.12 uniprot_domain_pair_mutation

Characterization of interface mutations introduced into structures in the *uniprot_domain_pair_model* table.

Columns:

uniprot_id Uniprot ID of the protein that is being mutated.

uniprot_domain_pair_id Unique id identifying each domain-domain interaction.

mutation Mutation for which the $\Delta\Delta G$ score is being predicted, specified in Uniprot coordinates.

mutation_errors List of errors obtained when evaluating the impact of the mutation.

model_filename_wt Filename of the homology model relaxed by FoldX but containing the wildtype residue.

model_filename_mut Filename of the homology model relaxed by FoldX and containing the mutated residue.

chain_modeller Chain containing the domain that was mutated, in homology models specified by `model_filename_wt` and `model_filename_mut`.

mutation_modeller Mutation for which the $\Delta\Delta G$ score is being predicted, specified in PDB RESNUM coordinates.

analyse_complex_energy_wt Comma-separated list of FoldX scores describing the effect of the wildtype residue on the stability of the protein domain.

stability_energy_wt Comma-separated list of FoldX scores describing the effect of the wildtype residue on protein-protein interaction interface.

analyse_complex_energy_mut Comma-separated list of FoldX scores describing the effect of the mutated residue on the stability of the protein domain.

stability_energy_mut Comma-separated list of FoldX scores describing the effect of the mutated residue on protein-protein interaction interface.

physchem_wt Comma-separated list of physicochemical properties describing the interaction between the wildtype residue and other residues on the opposite chain.

physchem_wt_ownchain Comma-separated list of physicochemical properties describing the interaction between the wildtype residue and other residues on the same chain.

physchem_mut Comma-separated list of physicochemical properties describing the interaction between the mutated residue and other residues on the opposite chain.

physchem_mut_ownchain Comma-separated list of physicochemical properties describing the interaction between the mutated residue and other residues on the same chain.

matrix_score Score assigned to the wt -> mut transition by the BLOSUM substitution matrix.

secondary_structure_wt Secondary structure of the wildtype residue, predicted by stride.

solvent_accessibility_wt Percent solvent accessible surface area of the wildtype residue, predicted by msms.

secondary_structure_mut Secondary structure of the mutated residue, predicted by stride.

solvent_accessibility_mut Percent solvent accessible surface area of the mutated residue, predicted by msms.

contact_distance_wt Shortest distance between the wildtype residue and a residue on the opposite chain.

contact_distance_mut Shortest distance between the mutated reside and a residue on the opposite chain.

provean_score Provean score for this mutation.

ddg Predicted change in Gibbs free energy of binding caused by this mutation.

mut_date_modified Date and time when this row was last modified.

CHAPTER 7

Modules

7.1 elaspic package

7.1.1 Submodules

7.1.2 elaspic.call_foldx module

class elaspic.call_foldx.**FoldX**(*pdb_file*, *chain_id*, *foldx_dir=None*)
Bases: object

__call__(*whatToRun*, *mutCodes=[]*)
Select which action should be performed by FoldX by setting *whatToRun*.

Possible values are:

•AnalyseComplex

•Stability

•RepairPDB

•BuildModel

See the FoldX manual for an explanation on what they do.

7.1.3 elaspic.call_modeller module

7.1.4 elaspic.call_tcoffee module

class elaspic.call_tcoffee.**TCoffee**(*alignment_fasta_file*, *mode*, *pdb_file=None*)
Bases: object

Alignes sequences using t_coffee in expresso mode.

align(*GAPOPEN=-0.0*, *GAPEXTEND=-0.0*)
Calls t_coffee (make sure BLAST is installed locally!).

Parameters

- **alignment_fasta_file** (*string*) – A file containing the fasta sequences to be aligned
- **alignment_template_file** (*string*) – A file containing the structural templates for the fasta sequences described above

- **GAPOPEN** (*int or str*) – See t_coffee manual
- **GAPEXTEND** (*int or str*) – See t_coffee manual
- **Returns** –
- **--------** –
- **alignment_output_file** (*str*) – Name of file which contains the alignment in fasta format.

7.1.5 elaspic.conf module

class elaspic.conf.**Configs**
Bases: object

A singleton class that keeps track of ELASPIC configuration settings.

clear()

copy()

get(*key*, *fallback=None*)

items()

keys()

update(***kwargs*)

values()

class elaspic.conf.**Singleton**
Bases: type

instance = None

elaspic.conf.**get_temp_dir**(*global_temp_dir='/tmp'*, *elaspic_foldername=''*)
If a *TMPDIR* is given as an environment variable, the tmp directory is created relative to that. This is useful when running on banting (the cluster in the ccbr) and also on Scinet. Make sure that it points to '/dev/shm/' on Scinet.

elaspic.conf.**read_configuration_file**(*config_file*, *unique_temp_dir=None*)

elaspic.conf.**read_database_configs**(*configParser*)
[DATABASE]

elaspic.conf.**read_model_configs**(*configParser*)
[MODEL]

elaspic.conf.**read_sequence_configs**(*configParser*)
[SEQUENCE]

7.1.6 elaspic.database_pipeline module

7.1.7 elaspic.elaspic_database module

class elaspic.elaspic_database.**MyDatabase**(*echo=False*)
Bases: object

add_domain(*d*)

add_domain_errors(*t*, *error_string*)

add_uniprot_sequence(*uniprot_sequence*)
Add new sequences to the database. :param uniprot_sequence: UniprotSequence object :rtype: None

configure_session()
Configure the Session class to use the current engine.

autocommit and *autoflush* are enabled for the *sqlite* database in order to improve performance.

copy_table_to_db(*table_name*, *table_folder*)
Copy data from a `.tsv` file to a table in the database.

create_database_tables(*clear_schema=False*, *keep_uniprot_sequence=True*)
Create a new database in the schema specified by the `schema_version` global variable. If `clear_schema == True`, remove all the tables in the schema first.

> **Warning:** Using this function with an existing database can lead to loss of data. Make sure that you know what you are doing!

Parameters

- **clear_schema** (*bool*) – Whether or not to delete all tables in the database schema before creating new tables.
- **keep_uniprot_sequence** (*bool*) – Whether or not to keep the *uniprot_sequence* table. Only relevant if *clear_schema* is *True*.

delete_database_tables(*drop_schema=False*, *keep_uniprot_sequence=True*)

Parameters

- **drop_schema** (*bool*) – Whether or not to drop the schema after dropping the tables.
- **keep_uniprot_sequence** (*bool*) – Wheter or not to keep the table (and schema) containing uniprot sequences.

get_alignment(*model*, *path_to_data*)

get_domain(*pfam_names*, *subdomains=False*)
Returns pdbfam-based definitions of all pfam domains in the pdb.

get_domain_contact(*pfam_names_1*, *pfam_names_2*, *subdomains=False*)
Returns domain-domain interaction information from pdbfam. Note that the produced dataframe may not have the same order as the keys.

get_engine(*echo=False*)
Get an SQLAlchemy engine that can be used to connect to the database.

get_rows_by_ids(*row_object*, *row_object_identifiers*, *row_object_identifier_values*)
Get the rows from the table *row_object* identified by keys *row_object_identifiers* with values *row_object_identifier_values*

get_uniprot_domain(*uniprot_id*, *copy_data=False*)

get_uniprot_domain_pair(*uniprot_id*, *copy_data=False*, *uniprot_domain_pair_ids=[]*)

get_uniprot_mutation(*d*, *mutation*, *uniprot_id=None*, *copy_data=False*)

get_uniprot_sequence(*uniprot_id*, *check_external=True*)

Parameters

- **uniprot_id** (*str*) – Uniprot ID of the protein

- **check_external** (*bool*) – Whether or not to look online if the protein sequence is not found in the local database.

Returns Contains the sequence of the specified uniprot

Return type SeqRecord

load_db_from_archive()
TODO: In the future I should move back to using json...

merge_model(*d*, *files_dict={}*)
Adds MODELLER models to the database.

merge_mutation(*mut*, *path_to_data=False*)

merge_provean(*provean*, *provean_supset_file*, *path_to_data*)
Adds provean score to the database.

merge_row(*row_instance*)
Adds a list of rows (*row_instances*) to the database.

mysql_command_template = "load data local infile '{table_folder}/{table_name}.tsv' into table {table_db_schema}.{ta

mysql_load_table_template = 'mysql –local-infile –host={db_url} –user={db_username} –password={db_password

psql_command_template = "\\\\copy {table_db_schema}.{table_name} from '{table_folder}/{table_name}.tsv' with cs

psql_load_table_template = 'PGPASSWORD={db_password} psql -h {db_url} -p {db_port} -U {db_username} -d

remove_model(*d*)
Remove a model from the database.

Do this if you realized that the model you built is incorrect or that some of the data is missing.

Raises `errors.ModelHasMutationsError` – The model you are trying to delete has pre-calculated mutations, so it can't be that bad. Delete those mutations and try again.

session_scope()
Provide a transactional scope around a series of operations. Enables the following construct: `with self.session_scope() as session:`.

sqlite_table_filename = '{table_folder}/{table_name}.tsv'

elaspic.elaspic_database.**check_exception**(*exc*, *valid_exc*)

elaspic.elaspic_database.**decorate_all_methods**(*decorator*)
Decorate all methods of a class with *decorator*.

elaspic.elaspic_database.**enable_sqlite_foreign_key_checks**(*engine*)

elaspic.elaspic_database.**get_uniprot_base_path**(*d*)
The uniprot id is cut into several chunks to create folders that will hold a manageable number of pdbs.

elaspic.elaspic_database.**get_uniprot_domain_path**(*d*)
Return the path to individual domains or domain pairs.

elaspic.elaspic_database.**pickle_dump**(*obj*, *filename*)

elaspic.elaspic_database.**retry_archive**(*fn*)
Decorator to keep probing the database untill you succeed.

elaspic.elaspic_database.**retry_database**(*fn*)
Decorator to keep probing the database untill you succeed.

elaspic.elaspic_database.**scinet_cleanup**(*folder*, *destination*, *name=None*)
zip and copy the results from the ramdisk to /scratch

7.1.8 elaspic.elaspic_database_tables module

Created on Thu Jun 11 16:52:31 2015

@author: ostrokach

class elaspic.elaspic_database_tables.**Domain**(**kwargs)
Bases: sqlalchemy.ext.declarative.api.Base

Profs domain definitions for all proteins in the PDB.

Columns:

cath_id Unique id identifying each domain in the PDB. Constructed by concatenating the pdb_id, pdb_chain, and an index specifying the order of the domain in the chain.

pdb_id The PDB id in which the domain is found.

pdb_chain The PDB chain in which the domain is found.

pdb_domain_def Domain definitions of the domain, in PDB RESNUM coordinates.

pdb_pdbfam_name The Profs name of the domain.

pdb_pdbfam_idx An integer specifying the number of times a domain with domain name pdb_pdbfam_name has occurred in this chain up to this point. It is used to make every (pdb_id, pdb_chain, pdb_pdbfam_name, pdb_pdbfam_idx) tuple unique.

domain_errors List of errors that occurred when annotating this domain, or when using this domain to make structural homology models.

cath_id

domain_errors

pdb_chain

pdb_domain_def

pdb_id

pdb_pdbfam_idx

pdb_pdbfam_name

class elaspic.elaspic_database_tables.**DomainContact**(**kwargs)
Bases: sqlalchemy.ext.declarative.api.Base

Interactions between Profs domains in the PDB. Only interactions that were predicted to be biologically relevant by NOXclass are included in this table.

Columns:

domain_contact_id A unique integer identifying each domain pair.

cath_id_1 Unique id identifying the first interacting domain in the *domain* table.

cath_id_2 Unique id identifying the second interacting domain in the *domain* table.

min_interchain_distance The closest that any residue in domain one comes to any residue in domain two.

contact_volume The volume covered by contacting residues.

contact_surface_area The surface area of the contacting regions of the first and second domains.

atom_count_1 The number of atoms in the first domain.

atom_count_2 The number of atoms in the second domain.

number_of_contact_residues_1 The number of residues in the first domain that come within 5 of the second domain.

number_of_contact_residues_2 The number of residues in the second domain that come withing 5 of the first domain.

contact_residues_1 A list of all residues in the first domain that come within 5 of the second domain. The residue number corresponds to the position of the residue in the domain.

contact_residues_2 A list of all residues in the second domain that come within 5 of the first domain. The residue number corresponds to the position of the residue in the domain.

crystal_packing The probability that the interaction is a crystallization artifacts, as defined by NOXclass.

domain_contact_errors List of errors that occurred when annotating this domain pair, or when using this domain as a template for making structural homology models.

atom_count_1

atom_count_2

cath_id_1

cath_id_2

contact_residues_1

contact_residues_2

contact_surface_area

contact_volume

crystal_packing

domain_1

domain_2

domain_contact_errors

domain_contact_id

min_interchain_distance

number_of_contact_residues_1

number_of_contact_residues_2

class elaspic.elaspic_database_tables.**Provean**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Description of the Provean supporting set calculated for a protein sequence. The construction of a supporting set is the most lengthy step in running Provean. Therefore, the supporting set is precalculated and stored for every protein sequence.

Columns:

uniprot_id The uniprot id of the protein.

provean_supset_filename The filename of the Provean supporting set. The supporting set contains the ids and sequences of all proteins in the NCBI nr database that are used by Provean to construct a multiple sequence alignment for the given protein.

provean_supset_length The number of sequences in Provean supporting set.

provean_errors List of errors that occurred while the Provean supporting set was being calculated.

provean_date_modified Date and time that this row was last modified.

provean_date_modified

provean_errors

provean_supset_filename

provean_supset_length

uniprot_id

uniprot_sequence

class elaspic.elaspic_database_tables.**UniprotDomain**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Pfam domain definitions for proteins in the *uniprot_sequence* table. This table was obtained by downloading Pfam domain definitions for all known proteins from the SIMAP website, and mapping the protein sequence to uniprot using the MD5 hash of each sequence.

Columns:

uniprot_domain_id Unique id identifying each domain.

uniprot_id The uniprot id of the protein containing the domain.

pdbfam_name The Profs name of the domain. In most cases this will be equivalent to the Pfam name of the domain.

pdbfam_idx The index of the Profs domain. `pdbfam_idx` ranges from 1 to the number of domains with the name `pdbfam_name` in the given protein. The `(pdbfam_name, pdbfam_idx)` tuple uniquely identifies each domain.

pfam_clan The Pfam clan to which this Profs domain belongs.

alignment_def Alignment domain definitions of the Profs domain. This field is obtained by removing gaps in the `alignment_subdefs` column.

pfam_names Pfam names of all Pfam domains that were combined to create the given Profs domain.

alignment_subdefs Comma-separated list of domain definitions for all Pfam domains that were merged to create the given Profs domain.

path_to_data Location for storing homology models, mutation results, and all other data that are relevant to this domain. This path is prefixed by *archive_dir*.

IS_TRAINING_SCHEMA = False

alignment_def

alignment_subdefs

path_to_data

pdbfam_idx

pdbfam_name

pfam_clan

pfam_names

uniprot_domain_id

uniprot_id

uniprot_sequence

class elaspic.elaspic_database_tables.**UniprotDomainModel**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Homology models for templates in the *uniprot_domain_template* table.

Columns:

uniprot_domain_id An integer which uniquely identifies each uniprot domain in the *uniprot_domain* table.

model_errors List of errors that occurred when making the homology model.

alignment_filename The name of the alignment that was given to Modeller when making the homology model.

model_filename The name of the homology model that was produced by Modeller.

chain The chain that contains the domain in question in the homology (this is now set to 'A' in all models).

norm_dope Normalized DOPE score of the model (lower is better).

sasa_score Comma-separated list of the percent solvent-accessible surface area for each residue.

m_date_modified The date and time when this row was last modified.

model_domain_def Domain definitions for the region of the domain that is covered by the structural template.

In most cases, this field is identical to the `domain_def` field in the *uniprot_domain_template* table. However, it sometimes happens that the best Profs structural template only covers a fraction of the Pfam domain. In that case, the `alignment_def` column in the *uniprot_domain* table, and the `domain_def` column in the *uniprot_domain_template* table, will contain the original Pfam domain definitions, and the `model_domain_def` column will contain domain definitions for only the region that is covered by the structural template.

alignment_filename

chain

m_date_modified

model_domain_def

model_errors

model_filename

norm_dope

sasa_score

template

uniprot_domain_id

class elaspic.elaspic_database_tables.**UniprotDomainMutation**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Characterization of mutations introduced into structures in the *uniprot_domain_model* table.

Columns:

uniprot_id Uniprot ID of the protein that was mutated.

uniprot_domain_id Unique id which identifies the Profs domain that was mutated in the *uniprot_domain* table.

mutation Mutation that was introduced into the protein, in Uniprot coordinates.

mutation_errors List of errors that occured while evaluating the mutation.

model_filename_wt The name of the file which contains the homology model of the domain after the model was relaxed with FoldX but before the mutation was introduced.

model_filename_mut The name of the file which contains the homology model of the domain after the model was relaxed with FoldX and after the mutation was introduced.

chain_modeller The chain which contains the domain that was mutated in the `model_filename_wt` and the `model_filename_mut` structures.

mutation_modeller The mutation that was introduced into the protein, in PDB RESNUM coordinates. This identifies the mutated residue in the `model_filename_wt` and the `model_filename_mut` structures.

stability_energy_wt Comma-separated list of scores returned by FoldX for the wildtype protein. The comma-separated list can be converted into a DataFrame with each column clearly labelled using the `elaspic.predictor.format_mutation_features()`. The FoldX energy terms are:

- dg
- backbone_hbond
- sidechain_hbond
- van_der_waals
- electrostatics
- solvation_polar
- solvation_hydrophobic
- van_der_waals_clashes
- entropy_sidechain
- entropy_mainchain
- sloop_entropy
- mloop_entropy
- cis_bond
- torsional_clash
- backbone_clash
- helix_dipole
- water_bridge
- disulfide
- electrostatic_kon
- partial_covalent_bonds
- energy_ionisation
- entropy_complex
- number_of_residues

stability_energy_mut Comma-separated list of scores returned by FoldX for the mutant protein. FoldX energy terms are the same as in *stability_energy_wt*, but for the mutated amino acid rather than the wildtype.

physchem_wt Physicochemical properties describing the interaction of the wildtype residue with residues on the opposite chain. The terms are:

- number of atoms in interacting residues that have the same charge.
- number of atoms in interacting residues that have an opposite charge.
- number of hydrogen bonds (very rough calculation).
- number of carbons in interacting residues within 4 A of the mutated residue (rough measure of the van der Waals force).

physchem_wt_ownchain Physicochemical properties describing the interaction of the wildtype residue with residues on the same chain. The terms are the same as in *physchem_wt*.

physchem_mut Physicochemical properties describing the interaction of the mutant residue with residues on the opposite chain. The terms are the same as in *physchem_wt*.

physchem_mut_ownchain Physicochemical properties describing the interaction of the mutant residue with residues on the same chain. The terms are the same as in *physchem_wt*.

matrix_score Score assigned to the wt -> mut transition by the BLOSUM substitution matrix.

secondary_structure_wt Secondary structure of the wildtype residue predicted by stride.

solvent_accessibility_wt Percent solvent accessible surface area of the wildtype residue, predicted by msms.

secondary_structure_mut Secondary structure of the mutated residue predicted by stride.

solvent_accessibility_mut Percent solvent accessible surface area of the mutated residue, predicted by msms.

provean_score Score produced by Provean for this mutation.

ddg Change in the Gibbs free energy of folding that our classifier predicts for this mutation.

mut_date_modified Date and time that this row was last modified.

chain_modeller

ddg

matrix_score

model

model_filename_mut

model_filename_wt

mut_date_modified

mutation

mutation_errors

mutation_modeller

physchem_mut

physchem_mut_ownchain

physchem_wt

physchem_wt_ownchain

provean_score

secondary_structure_mut

secondary_structure_wt

solvent_accessibility_mut

solvent_accessibility_wt

stability_energy_mut

stability_energy_wt

uniprot_domain_id

uniprot_id

class elaspic.elaspic_database_tables.**UniprotDomainPair**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Potentially-interacting pairs of domains for proteins that are known to interact, according to Hippie, IRefIndex, and Rolland et al. 2014.

Columns:

uniprot_domain_pair_id Unique id identifying each domain-domain interaction.

uniprot_domain_id_1 Unique id of the first domain.

uniprot_domain_id_2 Unique id of the second domain.

rigids Phased out.

domain_contact_ids List of unique ids identifying all domain-domain pairs in the PDB, where one domain belongs to the protein containing `uniprot_domain_id_1` and the other domain belongs to the protein containing `uniprot_domain_id_2`. This was used as crystallographic evidence that the two proteins interact.

path_to_data Location for storing homology models, mutation results, and all other data that is relevant to this domain pair. This path is prefixed by *archive_dir*.

domain_contact_ids

path_to_data

rigids

uniprot_domain_1

uniprot_domain_2

uniprot_domain_id_1

uniprot_domain_id_2

uniprot_domain_pair_id

uniprot_id_1

uniprot_id_2

class elaspic.elaspic_database_tables.**UniprotDomainPairModel**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Structural models of interactions between pairs of domains in the *uniprot_domain_pair* table.

Columns:

uniprot_domain_pair_id Unique id identifying each domain-domain interaction.

model_errors List of errors that occured while making the homology model.

alignment_filename_1 Name of the file containing the alignment of the first domain with its structural template.

alignment_filename_2 Name of the file containing the alignment of the second domain with its structural template.

model_filename Name of the file containing the homology model of the domain-domain interaction created by Modeller.

chain_1 Chain containing the first domain in the model specified by `model_filename`.

chain_2 Chain containing the second domain in the model specified by `model_filename`.

norm_dope The normalized DOPE score of the model.

interface_area_hydrophobic Hydrophobic surface area of the interface, calculated using POPS.

interface_area_hydrophilic Hydrophilic surface area of the interface, calculated using POPS.

interface_area_total Total surface area of the interface, calculated using POPS.

interface_dg Gibbs free energy of binding for this domain-domain interaction, predicted using FoldX. Not implemented yet!

interacting_aa_1 List of amino acid positions in the first domain that are within 5 of the second domain. Positions are specified using uniprot coordinates.

interacting_aa_2 List of amino acids in the second domain that are within 5 of the first domain. Position are specified using uniprot coordinates.

m_date_modified Date and time that this row was last modified.

model_domain_def_1 Domain boundaries of the first domain that are covered by the Profs structural template.

model_domain_def_2 Domain boundaries of the second domain that are covered by the Profs structural template.

alignment_filename_1

alignment_filename_2

chain_1

chain_2

interacting_aa_1

interacting_aa_2

interface_area_hydrophilic

interface_area_hydrophobic

interface_area_total

interface_dg

m_date_modified

model_domain_def_1

model_domain_def_2

model_errors

model_filename

norm_dope

template

uniprot_domain_pair_id

class elaspic.elaspic_database_tables.**UniprotDomainPairMutation**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Characterization of interface mutations introduced into structures in the *uniprot_domain_pair_model* table.

Columns:

uniprot_id Uniprot ID of the protein that is being mutated.

uniprot_domain_pair_id Unique id identifying each domain-domain interaction.

mutation Mutation for which the $\Delta\Delta G$ score is being predicted, specified in Uniprot coordinates.

mutation_errors List of errors obtained when evaluating the impact of the mutation.

model_filename_wt Filename of the homology model relaxed by FoldX but containing the wildtype residue.

model_filename_mut Filename of the homology model relaxed by FoldX and containing the mutated residue.

chain_modeller Chain containing the domain that was mutated, in homology models specified by `model_filename_wt` and `model_filename_mut`.

mutation_modeller Mutation for which the $\Delta\Delta G$ score is being predicted, specified in PDB RESNUM coordinates.

analyse_complex_energy_wt Comma-separated list of FoldX scores describing the effect of the wildtype residue on the stability of the protein domain.

stability_energy_wt Comma-separated list of FoldX scores describing the effect of the wildtype residue on protein-protein interaction interface.

analyse_complex_energy_mut Comma-separated list of FoldX scores describing the effect of the mutated residue on the stability of the protein domain.

stability_energy_mut Comma-separated list of FoldX scores describing the effect of the mutated residue on protein-protein interaction interface.

physchem_wt Comma-separated list of physicochemical properties describing the interaction between the wildtype residue and other residues on the opposite chain.

physchem_wt_ownchain Comma-separated list of physicochemical properties describing the interaction between the wildtype residue and other residues on the same chain.

physchem_mut Comma-separated list of physicochemical properties describing the interaction between the mutated residue and other residues on the opposite chain.

physchem_mut_ownchain Comma-separated list of physicochemical properties describing the interaction between the mutated residue and other residues on the same chain.

matrix_score Score assigned to the wt -> mut transition by the BLOSUM substitution matrix.

secondary_structure_wt Secondary structure of the wildtype residue, predicted by stride.

solvent_accessibility_wt Percent solvent accessible surface area of the wildtype residue, predicted by msms.

secondary_structure_mut Secondary structure of the mutated residue, predicted by stride.

solvent_accessibility_mut Percent solvent accessible surface area of the mutated residue, predicted by msms.

contact_distance_wt Shortest distance between the wildtype residue and a residue on the opposite chain.

contact_distance_mut Shortest distance between the mutated reside and a residue on the opposite chain.

provean_score Provean score for this mutation.

ddg Predicted change in Gibbs free energy of binding caused by this mutation.

mut_date_modified Date and time when this row was last modified.

analyse_complex_energy_mut

analyse_complex_energy_wt

chain_modeller

contact_distance_mut

contact_distance_wt

ddg

matrix_score

model

model_filename_mut

model_filename_wt

mut_date_modified

mutation

mutation_errors

mutation_modeller

physchem_mut

physchem_mut_ownchain

physchem_wt

physchem_wt_ownchain

provean_score

secondary_structure_mut

secondary_structure_wt

solvent_accessibility_mut

solvent_accessibility_wt

stability_energy_mut

stability_energy_wt

uniprot_domain_pair_id

uniprot_id

class elaspic.elaspic_database_tables.**UniprotDomainPairTemplate**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Structural templates for pairs of domains in the *uniprot_domain_pair* table.

Columns:

uniprot_domain_pair_id Unique id identifying each domain-domain interaction.

domain_contact_id Unique id of the domain pair in the *domain_contact* table that was used as a template for the modelled domain pair.

cath_id_1 Unique id of the structural template for the first domain.

cath_id_2 Unique id of the structural template for the second domain.

identical_1 Fraction of residues in the Blast alignment of the first domain to its template that are *identical*.

conserved_1 Fraction of residues in the Blast alignment of the first domain to its template that are *conserved*.

coverage_1 Fraction of the first domain that is covered by the blast alignment.

score_1 Score obtained by multiplying `identical_1` by `coverage_1`.

identical_if_1 Fraction of interface residues [1] that are *identical* in the Blast alignment of the first domain.

conserved_if_1 Fraction of interface residues [1] that are *conserved* in the Blast alignment of the first domain.

coverage_if_1 Fraction of interface residues [1] that are *covered* by the Blast alignment of the first domain.

score_if_1 Score obtained by combining `identical_if_1` and `coverage_if_1` using (7.1).

identical_2 Fraction of residues in the Blast alignment of the second domain to its template that are *identical*.

conserved_2 Fraction of residues in the Blast alignment of the second domain to its template that are *conserved*.

coverage_2 Fraction of the second domain that is covered by the blast alignment.

score_2 Score obtained by multiplying `identical_2` by `coverage_2`.

identical_if_2 Fraction of interface residues [1] that are *identical* in the Blast alignment of the second domain.

conserved_if_2 Fraction of interface residues [1] that are *conserved* in the Blast alignment of the second domain.

coverage_if_2 Fraction of interface residues [1] that are *covered* by the Blast alignment of the second domain.

score_if_2 Score obtained by combining `identical_if_2` and `coverage_if_2` using (7.1).

score_total The product of `score_1` and `score_2`.

score_if_total The product of `score_if_1` and `score_if_2`.

score_overall The product of `score_total` and `score_if_total`. This is the score that was used to select the best Profs domain pair to be used as a template.

t_date_modified The date and time when this row was last updated.

template_errors List of errors that occured while looking for the structural template.

[1] Interface residues are defined as residues that are within 5 of the partner domain.

cath_id_1

cath_id_2

conserved_1

conserved_2

conserved_if_1

conserved_if_2

coverage_1

coverage_2

coverage_if_1

coverage_if_2

domain_1

domain_2

domain_contact

domain_contact_id

domain_pair

identical_1

identical_2

identical_if_1

identical_if_2

score_1

score_2

score_if_1

score_if_2

score_if_total

score_overall

score_total

t_date_modified

template_errors

uniprot_domain_pair_id

class elaspic.elaspic_database_tables.**UniprotDomainTemplate**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Structural templates for domains in the *uniprot_domain* table. Lists PDB crystal structures that will be used for making homology models.

Columns:

uniprot_domain_id An integer which uniquely identifies each uniprot domain in the *uniprot_domain* table.

template_errors List of errors that occurred during the process for finding the template.

cath_id The unique id identifying the structural template of the domain.

domain_start The Uniprot position of the first amino acid of the Profs domain.

domain_end The Uniprot position of the last amino acid of the Profs domain.

domain_def Profs domain definitions for domains with structural templates. Domain definitions in this column are different from domain definitions in the `alignment_def` column of the *uniprot_domain* table in that they have been expanded to match domain boundaries of the Profs structural template, identified by the `cath_id`.

alignment_identity Percent identity of the domain to its structural template.

alignment_coverage Percent coverage of the domain to its structural template.

alignment_score A score obtained by combining `alignment_identity` ($SeqId$) and `alignment_coverage` (Cov) using the following equation, as described by Mosca et al.:

$$Score = 0.95 \cdot \frac{SeqId}{100} \cdot \frac{Cov}{100} + 0.05 \cdot \frac{Cov}{100} \tag{7.1}$$

t_date_modified The date and time when this row was last modified.

alignment_coverage

alignment_identity

alignment_score

cath_id

domain

domain_def

domain_end

domain_start

t_date_modified

template_errors

uniprot_domain

uniprot_domain_id

class `elaspic.elaspic_database_tables.`**`UniprotSequence`**`(`***kwargs*`)`
Bases: `sqlalchemy.ext.declarative.api.Base`

Protein sequences from the Uniprot KB, obtained by parsing *uniprot_sprot_fasta.gz*, *uniprot_trembl_fasta.gz*, and *homo_sapiens_variation.txt* files from the Uniprot ftp site.

Columns:

db The database to which the protein sequence belongs. Possible values are *sp* for SwissProt and *tr* for TrEMBL.

uniprot_id The uniprot id of the protein.

uniprot_name The uniprot name of the protein.

protein_name The protein name.

organism_name Name of the organism in which this protein is found.

gene_name Name of the gene that codes for this protein sequence.

protein_existence Evidence for the existence of the protein:

1. Experimental evidence at protein level
2. Experimental evidence at transcript level
3. Protein inferred from homology
4. Protein predicted
5. Protein uncertain

sequence_version Version of the protein amino acid sequence.

uniprot_sequence Amino acid sequence of the protein.

db

gene_name

organism_name

protein_existence

protein_name

sequence_version

uniprot_id

uniprot_name

uniprot_sequence

elaspic.elaspic_database_tables.**get_db_specific_param**(*key*)

elaspic.elaspic_database_tables.**get_table_args**(*table_name*, *index_columns=[]*, *db_specific_params=[]*)

Returns a tuple of additional table arguments.

7.1.9 elaspic.elaspic_model module

7.1.10 elaspic.elaspic_predictor module

Created on Wed Sep 30 16:54:21 2015

@author: strokach

class elaspic.elaspic_predictor.**Predictor**

Bases: object

feature_name_conversion = {'normDOPE': 'norm_dope', 'seq_id_avg': 'alignment_identity'}

score(*df*, *core_or_interface*)

Parameters **df** (*DataFrame*) – One or more rows with all data required to predict $Delta Delta G$ score. Like something that you would get when you join the appropriate rows in the database.

Returns **df** – Same as the input dataframe, except with one additional column: *ddg*.

Return type Dataframe

elaspic.elaspic_predictor.**convert_features_to_differences**(*df*, *keep_mut=False*)
Creates a new set of features (ending in *_change*) that describe the difference between values of the wildtype (features ending in *_wt*) and mutant (features ending in *_mut*) features. If *keep_mut* is *False*, removes all mutant features (features ending in *_mut*).

elaspic.elaspic_predictor.**format_mutation_features**(*feature_df*, *core_or_interface*)
Converts columns containing comma-separated lists of FoldX features and physicochemical features into a DataFrame where each feature has its own column.

Parameters

- **feature_df** (*DataFrame*) – A pandas DataFrame containing a subset of rows from the *uniprot_domain_mutation* or the *uniprot_domain_pair_mutation* tables.
- **core_or_interface** (*int or str*) – If 0 or 'core', the *feature_df* DataFrame contains columns from the *uniprot_domain_mutation* table. If 1 or 'interface, the feature_df DataFrame contains columns from the *uniprot_domain_pair_mutation* table.

Returns Contains the same data as *feature_df*, but with columns containing comma-separated lists of features converted to columns containing a single feature each.

Return type DataFrame

7.1.11 elaspic.elaspic_sequence module

class elaspic.elaspic_sequence.**Sequence**(*sequence_file*, *provean_supset_file=None*)
Bases: object

Class for calculating sequence level features.

mutate(*mutation*)

provean_supset_exists

provean_supset_file

result

score_pairwise(*seq1*, *seq2*, *matrix=None*, *gap_s=None*, *gap_e=None*)
Get the BLOSUM (or what ever matrix is given) score.

elaspic.elaspic_sequence.**convert_basestring_to_seqrecord**(*sequence*, *sequence_id='id'*)

elaspic.elaspic_sequence.**download_uniport_sequence**(*uniprot_id*, *output_dir*)

7.1.12 elaspic.errors module

exception elaspic.errors.**AlignmentNotFoundError**(*save_path*, *alignment_filename*)
Bases: Exception

exception elaspic.errors.**Archive7zipError**(*result*, *error_message*, *return_code*)
Bases: Exception

exception elaspic.errors.**Archive7zipFileNotFoundError**(*result*, *error_message*, *return_code*)
Bases: *elaspic.errors.Archive7zipError*

exception elaspic.errors.**ChainsNotInteractingError**
Bases: Exception

exception elaspic.errors.**DataError**
Bases: Exception

exception elaspic.errors.**FoldXAAMismatchError**
Bases: Exception

exception elaspic.errors.**FoldxError**
Bases: Exception

exception elaspic.errors.**InterfaceMismatchError**
Bases: Exception

exception elaspic.errors.**LowIdentity**
Bases: Exception

exception elaspic.errors.**MSMSError**
Bases: Exception

exception elaspic.errors.**ModelHasMutationsError**
Bases: Exception

Don't delete a model that has precalculated mutations!

exception elaspic.errors.**ModellerError**
Bases: Exception

exception elaspic.errors.**MutationMismatchError**
Bases: Exception

exception elaspic.errors.**MutationOutsideDomainError**
Bases: Exception

exception elaspic.errors.**MutationOutsideInterfaceError**
Bases: Exception

exception elaspic.errors.**NoModelFoundError**
Bases: Exception

exception elaspic.errors.**NoSequenceFound**
Bases: Exception

exception elaspic.errors.**NoTemplatesFoundError**
Bases: Exception

exception elaspic.errors.**PDBChainError**
Bases: Exception

exception elaspic.errors.**PDBDomainDefsError**
Bases: Exception

PDB domain definitions not found in the pdb file

exception elaspic.errors.**PDBEmptySequenceError**
Bases: Exception

One of the sequences is missing from the alignment. The most likely cause is that the alignment domain definitions were incorrect.

exception elaspic.errors.**PDBError**
Bases: Exception

exception elaspic.errors.**PDBNotFoundError**
Bases: Exception

exception elaspic.errors.**PopsError**(*message*, *pdb*, *chains*)
Bases: Exception

exception elaspic.errors.**ProteinDefinitionError**
Bases: Exception

exception elaspic.errors.**ProveanError**
Bases: Exception

exception elaspic.errors.**ProveanResourceError**(*message*, *child_process_group_id*)
Bases: Exception

exception elaspic.errors.**ResourceError**
Bases: Exception

exception elaspic.errors.**TcoffeeBlastError**(*result*, *error*, *alignInFile*, *system_command*)
Bases: Exception

exception elaspic.errors.**TcoffeeError**(*result*, *error*, *alignInFile*, *system_command*)
Bases: Exception

exception elaspic.errors.**TcoffeePDBidError**(*result*, *error*, *alignInFile*, *system_command*)
Bases: Exception

exception elaspic.errors.**TemplateCoreError**
Bases: Exception

exception elaspic.errors.**TemplateInterfaceError**
Bases: Exception

exception elaspic.errors.**WrongConfigKeyError**
Bases: Exception

7.1.13 elaspic.helper module

class elaspic.helper.**WritableObject**(*logger*)
Bases: object

A class for collecting all the print statements from modeller in order to redirect them to the logger later on

write(*string*)

class elaspic.helper.**color**
Bases: object

BLUE = '\x1b[94m'

BOLD = '\x1b[1m'

CYAN = '\x1b[96m'

DARKCYAN = '\x1b[36m'

END = '\x1b[0m'

FAIL = '\x1b[91m'

GREEN = '\x1b[92m'

HEADER = '\x1b[95m'

OKBLUE = '\x1b[94m'

OKGREEN = '\x1b[92m'

PURPLE = '\x1b[95m'

RED = '\x1b[91m'

UNDERLINE = '\x1b[4m'

WARNING = '\x1b[93m'

YELLOW = '\x1b[93m'

elaspic.helper.**decode_aa_list**(*interface_aa*)

elaspic.helper.**decode_domain_def**(*domains*, *merge=True*, *return_string=False*)
Unlike split_domain(), this function returns a tuple of tuples of strings, preserving letter numbering (e.g. 10B)

elaspic.helper.**decode_text_as_list**(*list_string*)
Uses the database convention to decode a string, describing domain boundaries of multiple domains, as a list of lists.

elaspic.helper.**encode_domain**(*domains*, *merged=True*)

elaspic.helper.**encode_list_as_text**(*list_of_lists*)
Uses the database convention to encode a list of lists, describing domain boundaries of multiple domains, as a string.

elaspic.helper.**get_echo**(*system_constant*)

elaspic.helper.**get_hostname**()

elaspic.helper.**get_lock**(*name*)

elaspic.helper.**get_logger**(*do_debug=True*, *logger_filename=None*)

elaspic.helper.**get_path_to_current_file**()
Find the location of the file that is being executed

elaspic.helper.**get_username**()

elaspic.helper.**get_which**(*bin_name*)

elaspic.helper.**kill_child_process**(*child_process*)

elaspic.helper.**lock**(*fn*)
Allow only a single instance of function *fn*, and save results to a lock file.

elaspic.helper.**log_print_statements**(*logger*)
Channel print statements to the debug logger

elaspic.helper.**make_tarfile**(*output_filename*, *source_dir*)

elaspic.helper.**open_exclusively**(*filename*, *mode='a'*)

elaspic.helper.**print_heartbeets**()

elaspic.helper.**row2dict**(*row*)

elaspic.helper.**run_subprocess**(*system_command*, ***popen_argvars*)

elaspic.helper.**run_subprocess_locally**(*working_path*, *system_command*, ***popen_argvars*)

elaspic.helper.**slugify**(*filename_string*)

elaspic.helper.**subprocess_check_output**(*system_command*, ***popen_argvars*)

elaspic.helper.**subprocess_check_output_locally**(*working_path*, *system_command*, ***popen_argvars*)

elaspic.helper.**subprocess_communicate**(*child_process*)

elaspic.helper.**switch_paths**(*working_path*)

elaspic.helper.**underline**(*print_string*)

7.1.14 elaspic.local_pipeline module

7.1.15 elaspic.machine_learning module

elaspic.machine_learning.**cross_validate_predictor**(*data*, *features*, *clf_options*, *output_filename=None*)

elaspic.machine_learning.**get_final_predictor**(*data*, *features*, *options*)
Train a predictor using the entire dataset.

elaspic.machine_learning.**write_row_to_file**(*results*, *output_filename*)
TODO: Add a datetime column to each written row.

7.1.16 elaspic.pipeline module

class elaspic.pipeline.**Foo**
Bases: object

run()

class elaspic.pipeline.**Pipeline**(*configurations*)
Bases: object

elaspic.pipeline.**execute_and_remember**(*f*)
Some basic memoizer.

elaspic.pipeline.**lock**(*fn*)
Allow only a single instance of function *fn*, and save results to a lock file.

7.1.17 elaspic.structure_analysis module

class elaspic.structure_analysis.**AnalyzeStructure**(*pdb_file*, *working_dir*, *vdw_distance=5.0*, *min_contact_distance=4.0*)
Bases: object

Runs the program pops to calculate the interface size of the complexes This is done by calculating the surface of the complex and the seperated parts. The interface is then given by the substracting.

__call__(*chain_id*, *mutation*, *chain_id_other=None*)
Calculate all properties.

get_dssp()
Not used because crashes on server.

get_interchain_distances(*pdb_chain=None*, *pdb_mutation=None*, *cutoff=None*)

get_interface_area(*chain_ids*)

get_physi_chem(*chain_id*, *mutation*)
Return the atomic contact vector, that is, counting how many interactions between charged, polar or "carbon" residues there are. The "carbon" interactions give you information about the Van der Waals packing of the residues. Comparing the wildtype vs. the mutant values is used in the machine learning algorithm.

'mutation' is of the form: 'A16' where A is the chain identifier and 16 the residue number (in pdb numbering) of the mutation chainIDs is a list of strings with the chain identifiers to be used if more than two chains are given, the chains not containing the mutation are considered as "opposing" chain

get_sasa(*program_to_use='pops'*)
Get Solvent Accessible Surface Area scores.

Note: deprecated

Use python:fn:*get_seasa* instead.

---

get_seasa()

get_secondary_structure()

get_stride()

get_structure_file(*chains*, *ext='.pdb'*)

working_dir = None
Folder with all the binaries (i.e. ./analyze_structure)

7.1.18 elaspic.structure_tools module

class elaspic.structure_tools.**MMCIFParserMod**(*temp_dir*)
Bases: Bio.PDB.MMCIFParser.MMCIFParser

get_structure(*structure_id*, *gzip_fh*)
Altered *get_structure* method which accepts gzip file handles as input.

class elaspic.structure_tools.**SelectChains**(*chain_letters*, *ns_chain_letters=None*, *ns=None*, *r_cutoff=None*)
Bases: Bio.PDB.PDBIO.Select

Only accept the specified chains when saving.

accept_residue(*residue*)

class elaspic.structure_tools.**StructureParser**(*pdb_file*, *chain_ids=None*, *domain_defs=[]*)
Bases: object

pdb_id
—

domain_boundaries
list of lists of lists

Elements in the outer list correspond to domains in each chain of the pdb. Elements of the inner list contain the start and end of each fragment of each domain. For example, if there is only one chain with pdb domain boundaries 1-10:20-45, this would correspond to domain_boundaries [[[1,10],[20,45]]].

extract()
Extract the wanted chains out of the PDB file. Removes water atoms and selects the domain regions (i.e. selects only those parts of the domain that are within the domain boundaries specified).

get_chain_seqres_sequence(*chain_id*, **args*, ***varargs*)
Call get_chain_seqres_sequence using chain with id chain_id

get_chain_sequence_and_numbering(*chain_id*, **args*, ***varargs*)
Call get_chain_sequence_and_numbering using chain with id chain_id

save_sequences(*output_dir=''*)

save_structure(*output_dir=''*, *remove_disordered=False*)

elaspic.structure_tools.**calculate_distance**(*atom_1*, *atom_2*, *cutoff=None*)
Calculate the distance between two points in 3D space.

Parameters **cutoff** (*float, optional*) – The maximum distance allowable between two points.

elaspic.structure_tools.**chain_is_hetatm**(*chain*)
Return True if the chain is made up entirely of HETATMs.

elaspic.structure_tools.**convert_aa**(*aa*, *quiet=False*)
Convert amino acids from three letter code to one letter code or vice versa

---

Note: Deprecated!

Use `''.join(AAA_DICT[aaa] for aaa in aa)` and `''.join(A_DICT[a] for a in aa)`.

---

elaspic.structure_tools.**convert_position_to_resid**(*chain*, *positions*, *domain_def_tuple=None*)
Convert mutation_domain to mutation_modeller. In mutation_modeller, the first amino acid in a chain may start with something other than 1.

elaspic.structure_tools.**convert_resnum_alphanumeric_to_numeric**(*resnum*)
Convert residue numbering that has letters (i.e. 1A, 1B, 1C...) to residue numbering without letters (i.e. 1, 2, 3...).

Note: Deprecated!

Use `get_chain_sequence_and_numbering()`.

---

elaspic.structure_tools.**download_pdb_file**(*pdb_id*, *output_dir*)
Move PDB structure to the local working directory.

elaspic.structure_tools.**euclidean_distance**(*a*, *b*)
Calculate the Euclidean distance between two lists or tuples of arbitrary length.

elaspic.structure_tools.**get_aa_residues**(*chain*)

elaspic.structure_tools.**get_chain_seqres_sequence**(*chain*, *aa_only=False*)
Get the amino acid sequence for the construct coding for the given chain.

Extracts a sequence from a PDB file. Usefull when interested in the sequence that was used for crystallization and not the ATOM sequence.

Parameters **aa_only** (*bool*) – If aa_only is set to *False*, selenomethionines will be included in the sequence. See: http://biopython.org/DIST/docs/api/Bio.PDB.Polypeptide-module.html.

elaspic.structure_tools.**get_chain_sequence_and_numbering**(*chain*, *domain_def_tuple=None*, *include_hetatms=False*)
Get the amino acid sequence and a list of residue ids for the given chain.

Parameters **chain** (*Bio.PDB.Chain.Chain*) – The chain for which to get the amino acid sequence and numbering.

elaspic.structure_tools.**get_interacting_residues**(*model*, *r_cutoff=5*, *skip_hetatm_chains=True*)
Returns all interactions between residues on different chains in *model*.

Returns A dictionary of interactions between chains i (0..n-1) and j (i+1..n). Keys are (chain_idx, chain_id, residue_idx, residue_resnum, residue_amino_acid) tuples. (e.g. (0, 'A', 0, '0', 'M'), (0, 1, '2', 'K'), ...) Values are a list of tuples having the same format as the keys.

Return type dict

You can reverse the order of keys and values like this:

```
complement = dict()
for key, values in get_interacting_chains(model):
    for value in values:
        complement.setdefault(value, set()).add(key)
```

You can get a list of all interacting chains using this command:

```
{(key[0], value[0])
 for (key, values) in get_interacting_chains(model).items()
 for value in values}
```

elaspic.structure_tools.**get_interactions**(*model*, *chain_id*, *r_cutoff=6*)

elaspic.structure_tools.**get_interactions_between_chains**(*model*, *chain_id_1*, *chain_id_2*, *r_cutoff=6*)

Calculate interactions between residues in pdb_chain_1 and pdb_chain_2. An interaction is defines as a pair of residues where at least one pair of atom is closer than r_cutoff. The default value for r_cutoff is 5 Angstroms.

Deprecated since version 1.0: Use python:fn:*get_interacting_residues* instead. It gives you both the residue index and the resnum.

Returns Keys are (residue_number, residue_amino_acid) tuples (e.g. ('0', 'M'), ('1', 'Q'), ...). Values are lists of (residue_number, residue_amino_acid) tuples. (e.g. [('0', 'M'), ('1', 'Q'), ...]).

Return type OrderedDict

elaspic.structure_tools.**get_interactions_between_chains_slow**(*model*, *pdb_chain_1*, *pdb_chain_2*, *r_cutoff=5*)

Calculate interactions between residues in pdb_chain_1 and pdb_chain_2. An interaction is defines as a pair of residues where at least one pair of atom is closer than r_cutoff. The default value for r_cutoff is 5 Angstroms.

Deprecated since version 1.0: Use *get_interacting_residues()* instead. It gives you both the residue index and the resnum.

elaspic.structure_tools.**get_pdb**

Parse a pdb file with biopythons PDBParser() and return the structure.

Parameters

- **pdb_code** (*str*) – Four letter code of the PDB file
- **pdb_path** (*str*) – Biopython pdb structure
- **temp_dir** (*str, optional, default='/tmp/'*) – Path to the folder for storing temporary files
- **pdb_type** (*'ent'/'pdb'/'cif', optional, default='ent'*) – The extension of the pdb to use

Raises PDBNotFoundError – If the pdb file could not be retrieved from the local (and remote) databases

elaspic.structure_tools.**get_pdb_file**(*pdb_id*, *pdb_database_dir*, *pdb_type='ent'*)

Get PDB file from a local mirror of the PDB database.

elaspic.structure_tools.**get_pdb_id**(*pdb_file*)

elaspic.structure_tools.**get_pdb_parser**(*pdb_type*, *temp_dir='/tmp'*)

Get PDB parser that can work with structures of the specified type.

elaspic.structure_tools.**get_pdb_structure**(*pdb_file*, *pdb_id=None*)
Set QUIET to False to output warnings like incomplete chains etc.

elaspic.structure_tools.**get_structure_sequences**(*file_or_structure*, *seqres_sequence=False*)
Convenience function returining a dictionary of sequences for a given file or a Biopython Structure, Model or Chain.

elaspic.structure_tools.**suppress_logger**(*fn*)

7.1.19 Module contents

7.2 Indices and tables

- modindex

e

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Y