Published in final edited form as:

Methods Mol Biol. 2017 ; 1529: 375–398. doi:10.1007/978-1-4939-6637-0_20.

EpiSweep: Computationally-driven Reengineering of Therapeutic Proteins to Reduce immunogenicity while Maintaining Function

Yoonjoo Choi*, Deeptak Verma*, Karl E. Griswold, and Chris Bailey-Kellogg⁺

Abstract

Therapeutic proteins are yielding ever more advanced and efficacious new drugs, but the biological origins of these highly effective therapeutics renders them subject to immune surveillance within the patient's body. When recognized by the immune system as a foreign agent, protein drugs elicit a coordinated response that can manifest a range of clinical complications including rapid drug clearance, loss of functionality and efficacy, delayed infusion-like allergic reactions, more serious anaphylactic shock, and even induced auto-immunity. It is thus often necessary to deimmunize an exogenous protein in order to enable its clinical application; critically, the deimmunization process must also maintain the desired therapeutic activity.

To meet the growing need for effective, efficient, and broadly applicable protein deimmunization technologies, we have developed the EpiSweep suite of protein design algorithms. EpiSweep seamlessly integrates computational prediction of immunogenic T cell epitopes with sequence- or structure- based assessment of the impacts of mutations on protein stability and function, in order to select combinations of mutations that make Pareto optimal trade-offs between the competing goals of low immunogenicity and high-level function. The methods are applicable both to the design of individual functionally deimmunized variants as well as the design of combinatorial libraries enriched in functionally deimmunized variants. After validating EpiSweep in a series of retrospective case studies providing comparisons to conventional approaches to T cell epitope deletion, we have experimentally demonstrated it to be highly effective in prospective application to deimmunization of a number of different therapeutic candidates. We conclude that our broadly applicable computational protein design algorithms guide the engineer towards the most promising deimmunized therapeutic candidates, and thereby have the potential to accelerate development of new protein drugs by shortening time frames and improving hit rates.

Keywords

Biologics; Therapeutic proteins; Computational protein design; Protein engineering; Immunogenicity; T cell epitope; Deimmunization; Combinatorial library; Pareto optimization

⁺corresponding author. Department of Computer Science, Dartmouth, USA. cbk@cs.dartmouth.edu.

*equally contributed

1. Introduction

Protein drugs are the most advanced tools in physicians' arsenal of therapeutic agents, and these complex molecular entities continue to improve outcomes for a range of familiar diseases as well as yield new treatment options for previously intractable illnesses [1]. One critical barrier to the development and clinical translation of therapeutic proteins is their susceptibility to immune surveillance within the patient's body, a process fundamentally driven by molecular recognition of immunogenic peptide fragments known as T cell epitopes. Upon eliciting an immune response, therapeutic proteins may cause a range of clinical complications including rapid drug clearance, loss of functionality and efficacy, delayed infusion-like allergic reactions, more serious anaphylactic shock, and even induced auto-immunity [2,3].

To mitigate protein immunogenicity, biomolecular engineers have previously sought to identify highly immunogenic T cell epitopes and delete them by mutagenic substitution of key amino acid residues [4–6]. However, experimental strategies for T cell epitope deletion are time- and labor-intensive, costly, and not universally successful. These limitations have led to the application of computational T cell epitope predictors as tools to accelerate the deimmunization process [7–9]. Yet prediction and mutagenic deletion of T cell epitopes is not sufficient for therapeutic protein deimmunization. Specifically, efficacious protein drugs require a folded, stable and active structure; thus combinations of epitope-deleting mutations must be selected for compatibility with each other and with the native protein architecture and function. Functional deimmunizing mutations can be selected purely experimentally [10], or guided by analysis of homolog sequences [11–14] and structural energies [15–18].

To create the next generation of computational tools for therapeutic protein deimmunization, we have integrated computational T cell epitope prediction with computational analysis of the structural and functional consequences of epitope-deleting mutations [11,18–20,13,21]. As opposed to serial application of T cell epitope predictors followed by bioinformatics-based or experimental mutation analysis, our protein design algorithms simultaneously optimize therapeutic candidates for both low immunogenicity and high-level stability and activity. Furthermore, they do so over an entire protein, considering the global implications of mutations on immunogenicity and function. Finally, by employing a powerful combinatorial optimization framework, our methods are guaranteed to generate globally optimal protein designs (with respect to the implemented predictors).

Here, we provide a step by step guide to the application of the EpiSweep suite of deimmunization algorithms, as introduced in our series of algorithmic papers [11,18,19,13,21] and prospectively applied in our series of experimental papers [12,22,14,23–25]. To assess immunogenicity, the software utilizes any pocket profile-based epitope predictor; we illustrate here with the publicly available ProPred matrices [26]. To gauge the structural and functional acceptability of epitope deleting mutations, it employs either sequence-based design or structure-based design, with score functions in the form of one- and two-body potentials based either on amino acid statistics or structural energy evaluations. To generate designs for experimental evaluation, it either optimizes single variants to be tested individually, or entire combinatorial libraries to be subjected to

screening or selection techniques. The design process is based on a “sweep” approach that maps the Pareto optimal frontier of the design space, identifying those designs that best optimize epitope score and multi-body potentials simultaneously, with no other single design better than the selected ones in terms of both objectives.

2. Materials

Software

EpiSweep brings together a number of different protein analysis techniques within a powerful combinatorial optimization process. Many, and for some design problems, all of the steps may be performed on a standard desktop machine; for some larger design tasks, a computer cluster may be required to achieve satisfactory run times. Necessary software components are as follows:

1. The EpiSweep software
 - a. Make a root directory (e.g., [*/home/user/episweep*]). Henceforth, directories and file names are in [*italics*]; customize as desired.
 - b. Download the EpiSweep suite (available by registration at <http://www.cs.dartmouth.edu/~cbk/episweep/>). EpiSweep Python modules are in [*episweep/episweep*], data files required to run EpiSweep are in [*episweep/data*], all pre- and postprocessing scripts described here are in [*episweep/bin*], other files including configuration files of third party programs are in [*episweep/misc*] and lysostaphin protein example files are in [*episweep/targets/1st_cwbd*].
2. Python 2.6 or higher, not Python 3.x (<https://www.python.org>)
3. IBM ILOG CPLEX API; freely available for academic research through the Academic initiative program (<http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/>). Ensure that the Python modules are installed and accessible from the Python interpreter, setting the PYTHONPATH environment variable if necessary.
4. Set environment variables so that the EpiSweep modules can be imported. Under Bash, add the following lines to the [*.bashrc*] file; apply analogous commands for other shells.


```
export EPISWEEP=/home/user/episweep
export PYTHONPATH=$PYTHONPATH:$EPISWEEP
export PATH=$PATH:$EPISWEEP/bin
```
5. Rotamer-based design and structure-based library design use third-party programs to generate potentials used by EpiSweep.
 - a. For extracting a rotamer potential:

- i. OSPREY 2.x [27,28] (<http://www.cs.duke.edu/donaldlab/osprey.php>)
 - ii. Java JDK 1.6.x (or higher) (<http://www.java.com>)
 - iii. Compile OSPREY and add the OSPREY binary files to the Java CLASSPATH in [.bashrc]


```
export CLASSPATH=/home/user/OSPREY/bin:$CLASSPATH
```
 - iv. Compile the OSPREY energy converter


```
$ javac $EPISWEEP/bin/eMatrixConverter.java
```
 - v. Add the OSPREY energy converter to the Java CLASSPATH in [.bashrc]


```
export CLASSPATH=$EPISWEEP/bin:$CLASSPATH
```
 - b. For extracting a structure-based sequence potential:
 - i. CLEVER 1.0 [29,30] (<http://keatinglab.mit.edu>)

Add to [.bashrc] an environment variable referencing the CLEVER installation

```
export PATH=$PATH:/home/user/clever1.0/compiled
```
 - ii. Rosetta suite (3.4 or higher) [31,32] (<https://www.rosettacommons.org/software>), if using it to generate structures for CLEVER training
6. The epitope analysis and postprocessing procedures demonstrated here are performed with R scripts and, when a structure is available, PyMOL [33]. The EpiSweep output files are in comma-separated values (CSV) format, so custom analysis procedures can readily be developed in any programming language or with spreadsheet software.
 - a. R 3.x or higher (<http://www.r-project.org>)
 - b. PyMOL 1.4 or higher (<http://sourceforge.net/projects/pymol/>)

Background files

Background files are required by EpiSweep for preprocessing and performing epitope analysis on target specific files. The files can be downloaded from their respective sources and should be placed in [*episweep/data*]. The pre- and postprocessing scripts described below look for background files in this directory. For correct readability by EpiSweep, the files should follow specific formats described in the Notes.

1. propred.csv: T cell epitope score matrices, in CSV format (*see* Note 1); demonstrated here with ProPred [26] (<http://www.imtech.res.in/raghava/propred>)
2. mccaldon.csv: Amino acid background threshold frequencies for filtering mutations from the MSA, in CSV format (*see* Note 2). For the example presented here, the threshold values are obtained from the McCaldon scale [34].

Target specific files

Target specific data files include the sequence, structure, and other information for the protein of interest. For the example presented here, the files are stored in [*episweep/targets/Ist_cwbd*]:

1. Target amino acid sequence, in FASTA format (*see* Note 3)

¹The file for the epitope scoring matrices starts with a line indicating the name of MHC alleles to be evaluated, and then score entries followed by threshold values.

Name
Matrix
Threshold values

where

- a. *Name* is for the user's reference and can be an arbitrary string (e.g., HLA-DRB1*0101).
- b. *Matrix* has 20 rows (amino acids in alphabetical order by 1-letter code), each with 9 comma separated columns (peptide positions, P1 through P9) of position-specific floating point values.
- c. Given a query peptide, the position-specific values are summed and then the sum is compared to one of the *threshold values*. The peptide is considered a hit if the sum is higher than the assigned threshold value.
- d. Our previous studies have been performed at a 5% threshold and the epitope predictions shows good correlations with experimental IC50 values [22,14,23].
- e. Many studies have aimed to remove predicted T-cell epitopes of the eight most common human leukocyte antigen (HLA)-DR alleles (HLA-DR*0101, 0301, 0401, 0701, 0801, 1101, 1301, and 1501) [38].

²The background amino acid frequency file is in two-column CSV format, with each row listing an amino acid and its threshold amino acid composition (%).

Ex: A,0.083

³FASTA file format for the target is as follows:

```
>Sequence Name
ACDEFGHIKLMNPQRSTVWY
```

The first line is an identifier and the next lines contain a sequence of standard 20 amino acids without gaps (one letter code).

2. Target structure in the Protein Data Bank (PDB) format if structure-based design is to be performed (*see* Note 4)
3. Multiple sequence alignment (MSA) of target homologs, in FASTA format (*see* Note 5), if an amino acid potential is to be constructed or conservation is to be used to determine allowed mutations
4. Any additional prior knowledge regarding mutational constraints for the target protein, in CSV format (*see* Note 6)

3. Methods

EpiSweep can employ sequence potentials (“sequence-based” design) or rotamer potentials (“rotamer-based” design) for designing deimmunized protein variants. Sequence potentials can be generated from sequence analysis (for example, using an MSA) or structural analysis (for example, using CLEVER [29,30]), whereas rotamer potentials can be derived from rotameric structural analysis (for example, using OSPREY [27,28]). EpiSweep can also construct a combinatorial library of variants (“library-based design”) using multiple sets of amino acids at specific positions. These sets of amino acids are referred to as “tubes” and the potentials derived as tube potentials. Tube potentials are derived from averaged sequence potentials, which in turn can be either be MSA-based or structure-based. Once sequence, rotamer, or tube potentials are derived for a specific target, EpiSweep can design individual variants at specified mutational loads, or combinatorial libraries of specified sizes and numbers of mutated positions.

⁴The structure should be in standard Protein Data Bank file format (<http://www.wwpdb.org/documentation/file-format>). Since the starting position of the sequence assumed to be 1, ensure that the structure follows the same numbering and has exactly the same residues.

⁵The MSA contains multiple entries, each is in FASTA format (*see* Note 3).

```
>Sequence 1
ACDEFGHIKLMNPQRSTVWY
>Sequence 2
A-DEFGHIKLMNPQ--TVWY
```

All the sequences must be of the same length. The first sequence must be the target sequence (gaps “-” are allowed here). A set of homologs can be obtained from numerous resources such as PSI-BLAST (<http://blast.ncbi.nlm.nih.gov>) and Pfam (<http://pfam.xfam.org/>).

⁶A mutational constraints file specifies allowed/disallowed amino acids (one letter code) at each position. The file has four columns: position, exclusively allowed amino acids (only these are allowed for this position), allowed amino acids (listed amino acids are allowed but others are also possible) and disallowed amino acids (other amino acids are possible but the listed one are disallowed). For example, if tryptophan (W) is disallowed for the first position, only proline (P) is allowed for the second position, no restriction for the third position and asparagine (N) and phenylalanine (F) are allowed but not cysteine (C) for the tenth position, the file is written as follows:

```
1,,W
2,P,
3,,
4,,NFC
```

If a position is missing or left blank like position 3, it is considered unconstrained. If this file is not given, all the positions are considered unconstrained.

The methods are illustrated with case study application to a domain from a therapeutic protein that we have been developing [24,35,25]: the cell wall binding domain (CWBD) of *Staphylococcus simulans* lysostaphin (LST). Lysostaphin is a potent anti-staphylococcal enzyme, effective even against drug-resistant *S. aureus* strains [36]. It is a two-domain enzyme with a cell wall binding domain that targets the bacterial peptidoglycan and a catalytic domain that cleaves the cell wall. Unfortunately, due to its bacterial origin, its immunogenicity has kept it out of clinical application.

Using background and target specific files, the following steps design deimmunized protein variants using EpiSweep (Fig. 1). First, create design specifications for the type of design and experimental construction to be performed – sequence-based, rotamer-based, or library-based. The design specification indicates what to consider at each position (allowed amino acids, rotamers, or tubes), which pairs of positions to score, and the scoring potentials to use. It also specifies parameters controlling the EpiSweep design process – number of mutated sites, epitope scoring matrix, and so forth. The EpiSweep process generates sets (“frontiers”) of Pareto optimal and (if desired) increasingly sub-optimal designs, trading off epitope content for potential score. The resulting designs are analyzed with postprocessing scripts in order to help select those for experimental evaluation.

Design specification

An EpiSweep design problem is specified in a CSV-format file that lists the input files (both background and target-specific), the mutational choices, the scoring potential, and parameters controlling the design process. Example design specification files are provided for each lysostaphin example within corresponding folders located at [episweep/targets/].

1. EpiSweep can design individual variants and protein libraries based on sequence, structure or rotamer information. This specification starts by specifying the basis for design (sequence, rotamer, or tube).
design, rotamer
2. Target sequence in FASTA format (*see* Note 3)
wt_seq, lst_cwbd.fasta
3. Target structure in PDB format, if available (*see* Note 4):
structure, lst_cwbd.pdb
4. MSA of homologs in FASTA format, if available (*see* Note 5):
msa, lst_cwbd_MSA.fasta
5. Target-specific mutational constraints file, if available (*see* Note 6):
mut_constraint, lst_cwbd_mut_constraints.csv
6. Epitope score matrix file (background file, in [episweep/data/]; *see* Note 1) and threshold value. In this example, the threshold value of the scoring matrix is top 5%.
epitope_score, propred.csv

- epitope_threshold, 5
7. Position-specific design choices (*see* Note 7):
choices, lst_cwbd_rot_choices.csv
 8. Position pairs to evaluate in CSV format (*see* Note 8):
pairs, lst_cwbd_pairs.csv
 9. Scoring potential, in either CSV format for sequence potential and tube potential (*see* Note 9) or binary format for rotamer potential (*see* Note 10):
 - a. For sequence potential:
seq_potential, lst_cwbd_seq_pot.csv
 - b. For rotamer potential:
rot_potential, lst_cwbd_rot_pot.dat

⁷A choice file is a two or three-column CSV file.

- a. For sequence and tube choices, this file has to have two columns, position and choice.
Ex (sequence choice): 1,A
Ex (tube choice, degenerate oligonucleotide in lower case): 1,rvt
Ex (tube choice, combinatorial amino acids in upper case): 1,AV
- b. For rotamer choice, this file must have three columns: position, choice, and rotamer index.
Ex (rotamer choice): 1,A,2

⁸The pair information is used to specify residue-residue interactions to be scored. The pair information file is in two-column CSV format: position 1, position 2.

Ex:
2,3
3,5
...

Note that the row is commutative (e.g. one can write a row as '3,5' or '5,3'), yet only one must be given.

⁹Amino acid or tube score potential files can be prepared in the following CSV format:

- a. For one-body score potential values: position, amino acid or tube, score
Ex: 2,G,-0.34
- b. For two-body score potential values: position 1, amino acid or tube 1, position 2, amino acid or tube 2, score
Ex: 5,D,30,K,-0.377637

¹⁰A binary score potential file has the following binary format (Big-endian):

- a. [4-byte unsigned integer] The number of residues, n
 - i. The following elements are iterated for n times
 - ii. [4-byte unsigned integer] The number of mutations at this position, p
 1. [1-byte character] Amino acid, a
 2. [4 byte unsigned integer] The number of degenerate elements (e.g. rotamers), r
- b. [4-byte float] One-body score (p, a, r) values
- c. [4-byte float] Two-body score ($p_1, a_1, r_1; p_2, a_2, r_2$) values

- c.** For tube potential, which also needs the originating sequence potential:
seq_potential, lst_cwbd_seq_pot.csv
tube_potential, lst_cwbd_tube_pot.csv
- 10.** Optimization parameters. For all design problems, give the number of optimal/suboptimal curves (default optimal curves = 1) and the mutational load.
num_curves, 5
mut_load, 8

In this example, the number of curves to be generated is 5 (one Pareto optimal and four suboptimal) and the number of allowed mutations (mutated positions for library design) is 9. The mutational load can also be given as a range, in which case all loads in the range will be designed:
mut_load, 4–8

For library design, also specify the minimum and maximum number of variants.
library_size, 10000–20000
- 11.** Finally, the filename for the EpiSweep output, in which the designs will be listed (*see* Note 11):

¹¹The result output file is in CSV format and columns are as follows:

- a.** Design index (sequential integer numbers, 0 for wild type)
b. Curve identification (0: wild type, 1: Pareto optimal and > 1 for suboptimal designs)
c. Epitope score
d. Potential score (the lower the better by convention)
e. Mutations made (“|” separated)
Ex. Empty for the wild type
D13E (single mutation, D to E at position 13)
D13E|F45W (double mutation)
Y5rvt (a tube with the degenerate codon ‘rvt’, lower-case, encoding for amino acids N, T, S, D, A and G)
Y5YV (a combinatorial tube YV encoding for combinatorial amino acids Y and V).
- f.** Sequence. For individual designs, a full sequence is shown. For library designs, ‘*’ is present where tubes are incorporated.
Ex:

```
KTNKHGTLTKSE*GSFTPNTDIIITRTTGPFTSMPQ*GV*KAGQT*HYDEVKQ*GHVWVGTYGN*G*RIYLPV
KTNKHGTLTKSE*E*GSFTPNTDIIITRTTGPFTSMPQSGV*KAGQT*HYDEVKQ*GHVWVGTYGN*G*RIYLPV
KTNKHGTLTKSE*GSFTPNTDIIITRTTGPFTSMPQ*GV*KAGQT*HYDEVKQ*GHVWVGTYGN*G*RIYLPV
```

design_output, lst_cwbd_plans.csv

Design specification generation

While all of the files in the design specification can be generated manually, EpiSweep provides a rich set of tools to help. An EpiSweep design specification generator is a simple Python script, in which a Design object is created and its methods are invoked in order to identify allowed mutations, generate scoring potentials, generate input files for external programs, etc. The following Python commands, from [*lst_cwbd_design_spec_gen.py*], illustrate the case study example.

1. Import EpiSweep library and initialize a Design object


```
from episweep import *
design = Design()
```
2. Load the design specification CSV file


```
design.load_spec('design_spec.csv')
```
3. Allowed mutation choices (*see* Note 7 for format) can be extracted from the MSA of the target using the following filters.
 - a. Initialize the filter based on the wild type sequence and the MSA FASTA file.


```
design.initialize_filter()
```
 - b. Filter the MSA to a subset of diverse, non-gappy sequences, with at least a specified identity to the wild type, at most another specified identity to others in the set, and at most a given fraction of gaps:


```
design.msa_filter(low_thresh=0.30, high_thresh=0.95,
fraction_gap_allowed=0.25)
```
 - c. Only keep amino acids that are sufficiently frequent in the (filtered) MSA relative to background frequencies. In this example, the McCaldon scale [34] of amino acid composition is used, according to the background file in [*episweep/data*].


```
design.background_frequency_filter(input='mccaldon.csv'
)
```
 - d. Only keep amino acids that by themselves contribute to removing predicted epitopes. In this example, the epitope prediction allows only those mutations which can remove at least one predicted epitope.


```
design.epitope_score_filter(min_epi_del=1)
```

- e. Apply mutational constraints (the file specified in step 5 in Design Specification; *see* Note 6).
`design.apply_mutational_constraints()`
4. Pairs whose interactions should be scored can be extracted either from the target MSA or the target protein structure (*see* Note 8).
 - a. Use MSA amino acid pair frequencies and χ^2 statistics [37].
`design.chisquare_pair_calculator()`
 - b. Use contacts in the structure. Residues whose C β atom (C α for glycine) distances are within a distance cut-off value are defined as a pair. In this example (and by default), the distance cut-off value is 8Å.
`design.structure_pair_calculator(cutoff=8)`
5. Construct the scoring potential file, which is in CSV format for sequence-based design (*see* Note 9) or in binary format for rotamer-based design (*see* Notes 10). For library-based design, first construct a sequence potential, and then construct the tube potential from it.
 - a. MSA-based sequence potentials
 - i. Generate a sequence potential from the target MSA.
`design.generate_seq_potential()`
 - ii. Finish editing the [*lst_cwbd_design_spec_gen.py*] file. Save and run the script.
`$ python lst_cwbd_design_spec_gen.py`
 - iii. The resulting potentials will be stored in [*lst_cwbd_seq_pot.csv*].
 - b. Structure-based sequence potentials, via CLEVER
 - i. Prepare the CLEVER configuration file. A target name must be specified and the output file is [*lst_cwbd_CLEVER_design.dat*] (*see* Note 12), which contains a list of amino acid choices and pair information.

¹²CLEVER [29,30] can be used to develop a sequence potential that represents structure-based energy terms, via a cluster expansion technique. CLEVER requires a list of possible amino acids at each position with the pair information. After training with energy data, CLEVER generates an output log file listing one-body and two-body scores, which can be converted into EpiSweep sequence potentials. Example configuration files are provided in [*episweep/misc/clever*] and a converter to EpiSweep format is in [*episweep/*

```
design.CLEVER_design_file_maker(output=lst_cwbd_CLEVER_design.n.dat')
```

- ii. Finish editing the *[lst_cwbd_design_spec_gen.py]* file. Save and run the script to generate the *[lst_cwbd_CLEVER_design.dat]* file.

```
$ python lst_cwbd_design_spec_gen.py
```
- iii. Generate random sequences by providing the design configuration file *[lst_cwbd_CLEVER_design.dat]* and using CLEVER sequence generator script *[GenSeqs]*. In this example, 6,000 random sequences (*see* Note 13) are generated and saved in *[train6000.seq]*. This step also assigns zero energy values to each protein next to its sequence (*see* Note 14).

```
$ GenSeqs -n 6000 -d  
lst_cwbd_CLEVER_design.dat -s  
train6000.seq
```
- iv. Use *[mutate_pdb.py]*, providing the design spec file *[design_spec.csv]* and the random sequence file *[train6000.seq]*. This script creates a PDB file for each sequence, with

bin]. The CLEVER configuration file is a plain text file that contains allowed amino acids at each position and pair information. The file has the following format:

```
#design_start  
[Space separated allowed amino acids for all positions] Ex: 1 A C D  
#design_end  
#cluster_start  
[A list of single positions line by line] Ex: 1  
[A list of pairs line by line] Ex: 1 3  
#cluster_end
```

¹³The following equation [21] provides an approximate number n of sequences to generate for CLEVER model training, based on the number of positions l , choices per position m , and pairs p . Please note that more sequences results in a better-trained model.

$$n = l \times m + p \times m^2$$

¹⁴The random sequence file is a plain text file each of which line has the following format:

[Energy value] [A space-separated sequence of amino acids]

Ex: 0.000 A C D E F G H I K L M N P Q R S T V W Y

Initially an energy value of 0.000 is assigned to each sequence. After calculating energy values for each sequence using modeling software, the zero entries need to be substituted for new energy values. For example, if the calculated energy value of the designed structure for the above sequence is -5.000, the substituted value would be:

Ex: -5.000 A C D E F G H I K L M N P Q R S T V W Y

file names in a sequential order (here *1.pdb, 2.pdb, ..., 6000.pdb*).

```
$ mutate_pdb.py -i design_spec.csv -s
train6000.seq
```

- v. Design structures and extract energies for the resulting sequences using any protein design program. For example, if employing Rosetta, use the fixbb program to design each structure and relax to minimize them. The calculated energies will be used in CLEVER model training in the following steps.

- vi. Enter calculated structure energy values into [*train6000.seq*] in the specified format (see Note 14), i.e., replace the zero values for each designed sequence in the initial file with their calculated energy values. Save this new file as [*train6000_energy.seq*]. An example file with this name and substituted energy values is provided for reference.

- vii. Estimate amino acid energies using the CLEVER script [*CETrFile*], providing the sequence list with energy values and the configuration file. The output file [*lst_cwbd_CLEVER.log*] is the standard output of the script.

```
$ CETrFile -d
lst_cwbd_CLEVER_design.dat -s
train6000_energy.seq -l
lst_cwbd_CLEVER.log
```

- viii. Convert the output file [*lst_cwbd_CLEVER.log*] to a structure-based sequence potential using the script [*clever_log_converter.py*], providing the log file. The output is a sequence potential file [*lst_cwbd_seq_pot.csv*] in CSV format (see Note 9).

```
$ clever_log_converter.py -i
lst_cwbd_CLEVER.log -o
lst_cwbd_seq_pot.csv
```

- c. Rotamer potentials, via OSPREY

- i. Prepare the OSPREY configuration files (see Note 15). [*KStar.cfg*] contains path information of OSPREY and rotamer libraries, and has to be separately prepared. An example file is provided in [*episweep/misc/osprey*]. The other two configuration files can be generated by the design object. The ‘target’ argument to the method is used to indicate final output file names. The following example generates [*lst_cwbd.DEE.cfg*], which contains amino acid choices at each position, and [*lst_cwbd.System.cfg*], which has structure information.

```
design.OSPREY_config_maker(target='lst_cwbd')
```

- ii. Finish editing the [*lst_cwbd_design_spec_gen.py*] file. Save and run the script.

```
$ python lst_cwbd_design_spec_gen.py
```

- iii. Run OSPREY, providing all the configuration files. For example, the following script runs OSPREY using 8 threads and 30 GB of memory. The standard output including rotamer choices after rotamer pruning will be saved in [*lst_cwbd_DEE.log*].

```
$ java -Xmx30000M KStar -t 8 -c
KStar.cfg lst_cwbd.System.cfg
lst_cwbd.DEE.cfg doDEE >
lst_cwbd_DEE.log
```

- iv. Extract rotamer choices from the standard output file using the script

¹⁵OSPREY 2.0 or higher [27,28] can be used to calculate structure-based rotamer energies. Note that OSPREY distinguishes protonation of histidine (HIP: double protonated histidine; positively charged, HID: delta protonated, HIE: epsilon protonated). The PDB structure file must contain all hydrogen atoms. Many programs can add hydrogen atoms to a PDB format structure, e.g., PyMol [33], AMBER [39], and TINKER [40]. The OSPREY rotamer-energy matrix needs to be converted to an appropriate format (see Note 10). There are three configuration files required to run OSPREY; example configuration files are provided in [*episweep/misc/osprey*], and the design generator can produce [*System.cfg*] and [*DEE.cfg*] for a design spec.

- a. KStar.cfg: contains the system path of OSPREY and the rotamer library
- b. System.cfg: contains information of the protein (structure filename, protein length, etc.)
- c. DEE.cfg: specifies output file names and detailed parameters including force field and mutation choices generated from the filtering process

[*extract_rotamer_choice.py*], providing the output log file [*lst_cwbd_DEE.log*]. The output file [*lst_cwbd_rotamer_choice.csv*] lists rotamer choices in EpiSweep CSV format (see Note 7).

```
$ extract_rotamer_choice.py -i
lst_cwbd_DEE.log -o
lst_cwbd_rotamer_choice.csv
```

- v. OSPREY generates an energy matrix file [*lst_cwbd_0.dat*] and a reference energy file [*lst_cwbd.eref.dat*] in Java binary format. Convert the output energy matrix files to a binary file [*lst_cwbd_rot_pot.dat*] (see Note 10) using the converter [*eMatrix_converter.class*].
- ```
$ java eMatrix_converter -i lst_cwbd_0.dat
-r lst_cwbd.eref.dat -o
lst_cwbd_rot_pot.dat
```

d. Tube potentials

- i. Generate a sequence potential via step 5.a or 5.b.
- ii. Generate “tubes” [20] from the allowed mutations, using either degenerate oligonucleotides or combinatorial sets of amino acids. To specify tube generation parameters, see Note 16.

<sup>16</sup>Library parameters that can be specified for tube construction are discussed with respect to this example:  
 design.generate\_tube(method='degenerate', restrict\_tube\_size=50, avoid\_aa=['C', 'P'], avoid\_stop\_codon=True, junk\_percent=0.34, junk\_type='deimmunizing')

- a. Method: EpiSweep designs combinatorial libraries to be constructed by incorporating at specific positions either degenerate oligonucleotides (method='degenerate') or combinations of specific mutations (method='combinatorial'). A “tube” considered at a position is the corresponding set of amino acids, either encoded by the degenerate oligonucleotide or listed as point mutations.
- b. Tube size restriction: The number of amino acids encoded within a tube can be restricted by a user specified value. Content in the tubes from the ‘combinatorial’ method should be restricted to a small number (e.g., 3) since possible combinations grow exponentially with increasing number of amino acid choices. Content in ‘degenerate’ tubes is restricted by the genetic code and thus the tube size restriction can be relaxed (e.g., 50).
- c. Amino acids restriction: A tube can be excluded if it encodes for undesired amino acids (e.g., ‘C’ or ‘P’).
- d. Stop codon allowance: A degenerate tube encoding for a stop codon can be eliminated.
- e. Junk percent: Degenerate tubes that encode for additional amino acids (due to degeneracy) beyond the desired ones are removed if there are too many undesired ones (called *junk*). For example, if the desired choices are {D,G,V} then a tube encoding for {D,E,G,V}, due to degeneracy in the genetic code, includes E as junk. The default fraction of junk allowed is 1/3; this can be changed using the junk\_percent parameter. The junk within a degenerate tube can be further restricted so that only ‘deimmunizing’ junk (i.e., deletes an epitope, as in design.epitope\_score\_filter) is allowed, or that ‘nondeimmunizing’ junk is also allowed.

```
design.generate_tube()
```

- iii. Generate tube potentials from the sequence potential and the previously-generated tubes.

```
design.generate_tube_potential()
```

- iv. Finish editing the [*lst_cwbd_design_spec_gen.py*] file. Save and run the script to generate and save tubes in [*lst_cwbd_tube_choices.csv*] and tube potentials in [*lst_cwbd_tube_pot.csv*].

```
$ python lst_cwbd_design_spec_gen.py
```

EpiSweep execution

EpiSweep execution only requires the design specification [*design_spec.csv*] file. The optimizer will set up an integer program to design variants or libraries according to the specification, and save the designs in the specified output file.

```
$ sweep.py design_spec.csv
```

Design analysis

1. Assess overall predicted immunogenicity of the wild type.
  - a. Predict epitopes, using script [*epitopes.py*], providing the design spec file [*design_spec.csv*], design index number (0 for wild type) and output filename. The output file is saved in CSV format (see Note 17).

```
$ epitopes.py -i design_spec.csv -n 0 -o
lst_cwbd_epitope.csv
```

- b. Plot the epitope “hits” (Fig. 2A), using script [*epitope_map.R*], providing the epitope prediction output file.

```
$ epitope_map.R -i lst_cwbd_epitope.csv -o
lst_cwbd_epitope.pdf
```

¹⁷The output file of epitope prediction is saved in CSV format and columns are as follows:

- a. Start position
- b. Peptide (nonamer)
- c. End position
- d. Hits (Binary, Allele specific)
- e. Total hits

- c. If a structure is available, visualize epitopes as “sausage” in PyMOL (Fig. 2B), using the script [*sausage_epitope.py*]; see Note 18.

PyMOL> from episweep import *

PyMOL> sausage(spec=“design_spec.csv”, palette=“white red”, overlap=True)
2. Plot the Pareto optimal frontiers (Fig. 3A for a rotamer-based individual design and Fig. 4A for a structure-based library design), using [*pareto_curve.R*] with the design spec file [*design_spec.csv*].

\$ pareto_curve.R -i design_spec.csv -o lst_cwbd_design_pareto.pdf
3. Predict epitopes for a variant of interest and compare to wild type.
  - a. Generate an epitope map [*lst_cwbd_var.csv*], using script [*epitopes.py*], providing the design spec file [*design_spec.csv*], design index number (design number 4 in this case) and output filename.

\$ epitopes.py -i design_spec.csv -n 4 -o
lst_cwbd_var_epitope.csv
  - b. Plot differential epitope maps of individual designs (Fig. 3B). Use script [*epitope_deletion_map.R*], providing epitope prediction files of the wild type and the particular variant.

\$ epitope_deletion_map.R -w lst_cwbd_epitope.csv -v
lst_cwbd_var_epitope.csv -o epitope_deletion_map.pdf
4. Enumerate and analyze the variants comprising a designed combinatorial library.
  - a. Extract a specific design from the Pareto frontier and enumerate a random subset or all designs (complete enumeration is potentially very time consuming), generating a CSV format file of individual variant designs [*lib_plan_enumerated.csv*] (see Note 11). In this example, library design index 10 is selected from the library Pareto frontier (Fig. 4A) and 5,000 variants are randomly sampled.

¹⁸The script runs inside the PyMOL command line interface. The main function of the script is ‘sausage’, which takes three arguments.

- a. design spec: the EpiSweep design specification file (mandatory)
- b. palette: color set (optional). Default: “white red”, i.e. the redder, the higher the epitope score
- c. overlap: if this argument is false, only the first residue of each epitope is highlighted, colored by how many epitopes start at that frame; else, all residues in each epitope are highlighted, with each position colored by how many epitopes include it. Default: True.

```
$ enumerate_library.py -i design_spec.csv -n 10 -e 5000 -
o lib_plan_enumerated.csv
```

- b.** Score and plot the variants (Fig. 4B), using script
[*plot_enumerated_lib.R*].

```
$ plot_enumerated_lib.R -i lib_plan_enumerated.csv -o
lib_plan_enumerated.pdf
```

Acknowledgments

This work was supported by NIH grant R01-GM-098977 to KEG and CBK. We also gratefully acknowledge computational resources provided by NSF grant CNS-1205521.

References

1. Aggarwal SR. What's fueling the biotech engine-2012 to 2013. *Nat Biotechnol.* 2014; 32:32–39. [PubMed: 24406926]
2. De Groot AS, Scott DW. Immunogenicity of protein therapeutics. *Trends in immunology.* 2007; 28(11):482–490. [PubMed: 17964218]
3. Schellekens H. Bioequivalence and the immunogenicity of biopharmaceuticals. *Nature reviews Drug discovery.* 2002; 1(6):457–462. [PubMed: 12119747]
4. Cizeau J, Grenkow DM, Brown JG, Entwistle J, MacDonald GC. Engineering and biological characterization of VB6-845, an anti-EpCAM immunotoxin containing a T-cell epitope-depleted variant of the plant toxin bouganin. *Journal of Immunotherapy.* 2009; 32(6):574–584. [PubMed: 19483652]
5. Harding FA, Liu AD, Stickler M, Razo OJ, Chin R, Faravashi N, Viola W, Graycar T, Yeung VP, Aehle W. A β -lactamase with reduced immunogenicity for the targeted delivery of chemotherapeutics using antibody-directed enzyme prodrug therapy. *Molecular cancer therapeutics.* 2005; 4(11):1791–1800. [PubMed: 16276001]
6. Warmerdam PA, Plaisance S, Vanderlick K, Vandervoort P, Brepoels K, Collen D, De Maeyer M. Elimination of a human T-cell region in staphylokinase by T-cell screening and computer modeling. *Thromb Haemost.* 2002; 87(4):666–673. [PubMed: 12008950]
7. De Groot A, Knopp P, Martin W. De-immunization of therapeutic proteins by T-cell epitope modification. *Developments in biologicals.* 2004; 122:171–194.
8. De Groot AS, Moise L. Prediction of immunogenicity for therapeutic proteins: state of the art. *Current Opinion in Drug Discovery and Development.* 2007; 10(3):332. [PubMed: 17554860]
9. Moise L, Song C, Martin WD, Tassone R, De Groot AS, Scott DW. Effect of HLA DR epitope de-immunization of Factor VIII in vitro and in vivo. *Clinical Immunology.* 2012; 142(3):320–331. [PubMed: 22222093]
10. Cantor JR, Yoo TH, Dixit A, Iverson BL, Forsthuber TG, Georgiou G. Therapeutic enzyme deimmunization by combinatorial T-cell epitope removal using neutral drift. *Proceedings of the National Academy of Sciences.* 2011; 108(4):1272–1277.
11. He L, Friedman AM, Bailey-Kellogg C. A divide-and-conquer approach to determine the Pareto frontier for optimization of protein engineering experiments. *Proteins: Structure, Function, and Bioinformatics.* 2012; 80(3):790–806.
12. Osipovitch DC, Parker AS, Makokha CD, Desrosiers J, Kett WC, Moise L, Bailey-Kellogg C, Griswold KE. Design and analysis of immune-evading enzymes for ADEPT therapy. *Protein Engineering Design and Selection.* 2012; 25(10):613–624.
13. Parker AS, Zheng W, Griswold KE, Bailey-Kellogg C. Optimization algorithms for functional deimmunization of therapeutic proteins. *BMC bioinformatics.* 2010; 11(1):180. [PubMed: 20380721]

14. Salvat RS, Parker AS, Choi Y, Bailey-Kellogg C, Griswold KE. Mapping the Pareto Optimal Design Space for a Functionally Deimmunized Biotherapeutic Candidate. *PLoS computational biology*. 2015; 11(1):e1003988. [PubMed: 25568954]
15. Choi Y, Griswold KE, Bailey-Kellogg C. Structure-based redesign of proteins for minimal T-cell epitope content. *Journal of computational chemistry*. 2013; 34(10):879–891. [PubMed: 23299435]
16. King C, Garza EN, Mazor R, Linehan JL, Pastan I, Pepper M, Baker D. Removing T-cell epitopes with computational protein design. *Proceedings of the National Academy of Sciences*. 2014; 111(23):8577–8582.
17. Mazor R, Eberle JA, Hu X, Vassall AN, Onda M, Beers R, Lee EC, Kreitman RJ, Lee B, Baker D. Recombinant immunotoxin for cancer treatment with low immunogenicity by identification and silencing of human T-cell epitopes. *Proceedings of the National Academy of Sciences*. 2014; 111(23):8571–8576.
18. Parker AS, Choi Y, Griswold KE, Bailey-Kellogg C. Structure-guided deimmunization of therapeutic proteins. *Journal of Computational Biology*. 2013; 20(2):152–165. [PubMed: 23384000]
19. Parker AS, Griswold KE, Bailey-Kellogg C. Optimization of therapeutic proteins to delete T-cell epitopes while maintaining beneficial residue interactions. *Journal of bioinformatics and computational biology*. 2011; 9(02):207–229. [PubMed: 21523929]
20. Parker AS, Griswold KE, Bailey-Kellogg C. Optimization of combinatorial mutagenesis. *Journal of Computational Biology*. 2011; 18(11):1743–1756. [PubMed: 21923411]
21. Verma D, Grigoryan G, Bailey-Kellogg C. Structure-based design of combinatorial mutagenesis libraries. *Protein Science*. 2015; 24(5):895–908. [PubMed: 25611189]
22. Salvat RS, Choi Y, Bishop A, Bailey-Kellogg C, Griswold KE. Protein deimmunization via structure-based design enables efficient epitope deletion at high mutational loads. *Biotechnology and bioengineering*. 2015; 71(24):4869–4880.
23. Salvat RS, Parker AS, Williams A, Choi Y, Bailey-Kellogg C, Griswold KE. Computationally driven deletion of broadly distributed T cell epitopes in a biotherapeutic candidate. *Cellular and Molecular Life Sciences*. 2014; 71(24):4869–4880. [PubMed: 24880662]
24. Blazanovic K, Zhao H, Choi Y, Li W, Salvat RS, Osopovitch DC, Fields J, Moise L, Berwin BL, Fiering SN, Bailey-Kellogg C, Griswold KE. Structure-based Design of Lysostaphin Yields Potent and Deimmunized Anti-Staphylococcal Therapies. *Molecular Therapy - Methods & Clinical Development*. 2:15021.
25. Zhao H, Verma D, Li W, Choi Y, Fiering SN, Bailey-Kellogg C, Griswold KE. Depletion of T cell epitopes in lysostaphin mitigates anti-drug antibody response and enhances antibacterial efficacy in vivo. *Chemistry and Biology*. 22(5):629–639.
26. Singh H, Raghava G. ProPred: prediction of HLA-DR binding sites. *Bioinformatics*. 2001; 17(12):1236–1237. [PubMed: 11751237]
27. Chen C-Y, Georgiev I, Anderson AC, Donald BR. Computational structure-based redesign of enzyme activity. *Proceedings of the National Academy of Sciences*. 2009; 106(10):3764–3769.
28. Gainza P, Roberts KE, Donald BR. Protein design using continuous rotamers. *PLoS computational biology*. 2012; 8(1):e1002335. [PubMed: 22279426]
29. Grigoryan G, Reinke AW, Keating AE. Design of protein-interaction specificity gives selective bZIP-binding peptides. *Nature*. 2009; 458(7240):859–864. [PubMed: 19370028]
30. Grigoryan G, Zhou F, Lustig SR, Ceder G, Morgan D, Keating AE. Ultra-fast evaluation of protein energies directly from sequence. *PLoS computational biology*. 2006; 2(6):e63. [PubMed: 16789811]
31. Rohl CA, Strauss CE, Misura KM, Baker D. Protein structure prediction using Rosetta. *Methods in enzymology*. 2004; 383:66–93. [PubMed: 15063647]
32. Simons KT, Kooperberg C, Huang E, Baker D. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *Journal of molecular biology*. 1997; 268(1):209–225. [PubMed: 9149153]
33. Schrödinger. The PyMOL Molecular Graphics System.

34. McCaldon P, Argos P. Oligopeptide biases in protein sequences and their use in predicting protein coding regions in nucleotide sequences. *Proteins: Structure, Function, and Bioinformatics*. 1988; 4(2):99–122.
35. Zhao H, Blazanovic K, Choi Y, Bailey-Kellogg C, Griswold KE. Gene and protein sequence optimization for high-level production of fully active and aglycosylated lysostaphin in *Pichia pastoris*. *Applied and environmental microbiology*. 2014; 80(9):2746–2753. [PubMed: 24561590]
36. Kokai-Kun JF. 10 Lysostaphin: a Silver Bullet for Staph. *Antimicrobial Drug Discovery: Emerging Strategies*. 2012; 22:147.
37. Larson SM, Di Nardo AA, Davidson AR. Analysis of covariation in an SH3 domain sequence alignment: applications in tertiary contact prediction and the design of compensating hydrophobic core substitutions. *Journal of molecular biology*. 2000; 303(3):433–446. [PubMed: 11031119]
38. Southwood S, Sidney J, Kondo A, del Guercio M-F, Appella E, Hoffman S, Kubo RT, Chesnut RW, Grey HM, Sette A. Several common HLA-DR types share largely overlapping peptide binding repertoires. *The Journal of Immunology*. 1998; 160(7):3363–3373. [PubMed: 9531296]
39. Case D, Babin V, Berryman J, Betz R, Cai Q, Cerutti D, Cheatham T III, Darden T, Duke R, Gohlke H. *Amber*. 2014:14.
40. Ponder, JW. TINKER: Software tools for molecular design. Washington University School of Medicine;

Figure 1.

Overview of the EpiSweep design strategy. (A) Collect target-specific information, including protein sequence, structure, homologs and knowledge-based mutational constraints. (B–D) Specify the details of the design problem. (B) Identify design choices and scoring potentials according to design strategy: (B.1) Sequence, (B.2) Rotamer or (B.3) Library. (C) Restrict pair potentials to positions deemed worthy of scoring, according to sequence/structure analysis. (D) Provide other optimization parameters controlling mutational load, epitope scoring, etc. (E) Apply EpiSweep to design Pareto frontiers trading off epitope score and potential score. (G) Analyze designs to select those worthy of experimental construction and evaluation.

Figure 2. Representation of epitope map: (A) sequence representation and (B) structure representation (rendered by PyMOL). The target protein shown here is the cell-wall binding domain of lysostaphin (PDB code: 4LXC chain A 403~493). In the sausage representation, the redder, the higher the epitope content.

Figure 3.

Postprocessing results plots. Default scripts with default options were used. (A) Pareto optimal variants of the lysostaphin cell-wall binding domain from rotamer-based design (5 mutations, 1 Pareto optimal and 4 suboptimal curves). The wild type is the black solid circle. The Pareto frontier is depicted with red solid circles and suboptimal designs are in gray circles. A particular design (in the blue circle, highlighted only for this figure) is further analyzed for epitope mapping. (B) An example epitope map of the design compared to the target wild type. Red lines are epitopes of the wild type and blue lines are deleted epitopes. The broken red line is the newly introduced epitope by the mutation. The mutated positions are marked at the bottom.

Figure 4. Structure-based library designs. (A) Pareto curve, with each red point representing a library. The wild type is shown by a black solid circle. (B) Enumeration of the blue circled library from (A), with 5,000 randomly selected variants illustrated by small black solid circles.