Published in final edited form as:

Methods Mol Biol. 2017 ; 1561: 213–232. doi:10.1007/978-1-4939-6798-8_13.

Enriching peptide libraries for binding affinity and specificity through computationally directed library design

Glenna Wink Foight^{1,2}, T. Scott Chen^{1,3}, Daniel Richman¹, and Amy E. Keating^{1,4}

¹Department of Biology, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

⁴Department of Biological Engineering Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Abstract

Peptide reagents with high affinity or specificity for their target protein interaction partner are of utility for many important applications. Optimization of peptide binding by screening large libraries is a proven and powerful approach. Libraries designed to be enriched in peptide sequences that are predicted to have desired affinity or specificity characteristics are more likely to yield success than random mutagenesis. We present a library optimization method in which the choice of amino acids to encode at each peptide position can be guided by available experimental data or structure-based predictions. We discuss how to use analysis of predicted library performance to inform rounds of library design. Finally, we include protocols for more complex library design procedures that consider the chemical diversity of the amino acids at each peptide position and optimize a library score based on a user-specified input model.

Keywords

Library design; integer linear programming; peptide engineering

1 Introduction

The increasing use of peptides as reagents for diagnostic, therapeutic, and basic research purposes highlights the need for engineered molecules with particular affinity and specificity profiles. Natural peptide interaction partners often do not have the high affinity or specificity required for such applications. Two main approaches exist for developing peptide reagents with desired binding characteristics: screening of large peptide libraries and computational design of peptide sequences.[1]

Library screens that use cell surface display, phage display, mRNA or ribosome display, and smaller, synthetic peptide libraries screened on beads are powerful techniques for discovering peptide reagents.[2–5] High throughput screening technologies can routinely survey 10^8 (yeast display) – 10^{15} (mRNA display) DNA sequences, but the enormous

*Corresponding author: Amy E. Keating, 77 Massachusetts Avenue, Building 68-622, Cambridge, MA 02139, USA. keating@mit.edu.

²Current address: Department of Chemistry, University of Washington, Seattle, WA 98195, USA

³Current address: Google Inc., Mountain View, CA 94043

theoretical sequence spaces of peptides greater than ~6–8 residues exceeds even those large numbers. Thus, randomly generated sequences may not sample the best molecules. A common alternative approach to identifying optimized peptide binders is to mutagenize a known interaction partner. When using random mutagenesis, whether by error-prone PCR or the use of NNK or other degenerate codons, the mutational load per sequence may be difficult to tune because different protein positions have different sensitivities to mutation. Too high of mutagenesis load will yield many non-binders, whereas low mutagenesis may not achieve sequences sufficiently diverged from the original sequence to meet challenging affinity or specificity goals.[6]

Meanwhile, computational modeling of protein-peptide interactions is advancing. Several methods have used structural information to successfully predict interacting peptide sequences in the proteome.[7–11] Physical detail in the models used for this ranges from high, in methods deriving from molecular mechanics calculations, to low in methods that use simple distance tabulations. However, relatively few examples of purely computational design of novel peptide binding partners have been reported.[12–15] This is in contrast to the field of protein-protein interaction design, in which computational design of novel interaction partners is becoming increasingly common.[16–18] It remains difficult to achieve adequate conformational sampling of peptide conformations, and inaccuracies in standard energy functions limit the accuracy of scoring complexes that involve only a small number of residue contacts.[19]

In recent years, the strengths of computational design and library screening technologies have been combined in methods that utilize computational algorithms to design libraries that reflect predictions about stability or binding made by a computational or data-based model. [1] Designing a large library that can be screened by high-throughput technologies, rather than just a small handful of sequences, overcomes the requirement for detailed and accurate information on all peptide positions. Researchers can make the best use of their screening capabilities by limiting variation to a productive sequence space, e.g., to peptide positions at which models predict affinity or specificity-enhancing mutations.

In the laboratory, libraries can be made using degenerate codons to include variation at different protein positions, can incorporate mixtures of defined codons at different positions, or can be composed of members of defined protein sequence. Libraries made with degenerate codons are the most economical and are the focus of this methods paper; such libraries are widely used. Degenerate codons include mixtures of nucleotides at each codon position that, collectively, encode a set of amino-acid residues. There are conventions for naming such codons, e.g. NNK stands for a codon with a mixture of A, C, G, T at the first and second positions, and a mixture of G and T at the last position. The NNK degenerate codon can code for any amino acid or a single stop codon. See the standard IUPAC nomenclature for definitions of other degenerate codons (<http://www.bioinformatics.org/sms/iupac.html>).

Much of the published work on computationally directed library design has been done in the context of enriching libraries for functional sequences of single-domain proteins such as green fluorescent protein (GFP), cytochrome P450, and β -lactamase.[20–24] A common

linear programming to optimize the library composition, with options to use pairwise energies and output a degenerate codon or defined-sequence library.

For the design of peptide libraries, flexibility in the ability to use many different types of input information is advantageous. Multiple sequence alignments, which are the preferred input for many previous library design methods, may not be useful for all protein-peptide interaction families, due to either too few validated binding partners, or extreme diversity in the binding site sequences. For many systems, there may also exist experimental data of varying types (SPOT arrays, alanine mutagenesis, deep sequencing data from single point mutant libraries, etc.) that a researcher would like to take into account. To incorporate diverse information sources that include both experimental and computationally derived data, we present a method for the computational design of peptide libraries enriched in sequences with a desired affinity or specificity profile. The basic method presented here was first used by Chen et al. to design libraries of Bcl-x_L variants with enhanced specificity for binding to BH3 peptides. Library design in that instance was based on Rosetta energies.[30] The method was then adapted to design BH3 peptide libraries enriched in specific binders of Bfl-1 based on SPOT array data, with a constraint imposed to ensure sampling of chemical diversity.[31] We have further demonstrated the utility of the method by designing two more BH3 peptide libraries with specificity for other Bcl-2 family members using SPOT array data and computational predictions from STATIUM.[32] Diverse, quantifiable experimental data or computational predictions can be used as input to this general framework, which applies integer linear programming to optimize library composition based on an easily modified set of parameters. The output is a degenerate codon library with size and characteristics tuned to the desired experimental screening strategy and end goal defined by the researcher.

2 Materials

The library optimization method presented here can be run with a simple set of scripts in a terminal environment on a Mac or Linux machine. Two Perl scripts, writeCodon.pl and runILP.pl do the complete design process. Two additional files are needed to run these scripts. File codon_combos.txt includes a database of codons used by writeCodon.pl. File library_design.mod is the file formatted for use with the ILP solver, glpsol, and it is edited and run by the runILP.pl script. These files are included in file LibraryDesignScripts.tar.gz. All scripts and example files are available on the GitHub KeatingLab/LibraryDesign repository.

Additionally, example input and output files are included: file_pref, file_req, codons_output, library_design_output. The files are referred to by these names throughout the Methods section, and they are included in the file LibraryDesignExample.tar.gz. The tar files can be opened using the Archive Utility on a Mac or the command “tar -xvzf LibraryDesignExample.tar.gz” in the terminal.

The ILP problem is solved using the glpsol solver in the GNU Linear Programing Kit (GLPK). This is available as a free download from <https://www.gnu.org/software/glpk/>. Install on a Linux or Mac machine by following the installation instructions included with

the software. You may need to use the command “sudo make install” to install in the default location. Make note of the path where the glpsol solver is installed, as you will need to direct the runILP.pl script to its location (default is /usr/local/bin/glpsol).

Finally, for the advanced multi-option method presented in section 3.5, scripts and example input and output files are included in MultiOption.tar.gz. File names are as given in section 3.5.

3 Methods

The method for library design presented here proceeds through the three steps outlined in Figure 1, followed by optional analysis of predicted library performance and further rounds of design. Because the available input information will vary for every protein-peptide interaction study, we present a general framework for formalizing various input datasets. Likewise, library design objectives will vary. We describe and provide a basic framework for optimizing a library to include peptides with high affinity or specificity for one target interaction partner. After a library design is output by the ILP code, a researcher can analyze it for its predicted behavior based on any experimental or computational models available. The results of these analyses can then be used to manually alter the allowed substitution and codon choices to improve output for further rounds of design. We have found this iterative process of design and evaluation very useful in exploring the tradeoffs that a protein designer inevitably faces when devising a screen.

3.1 Formalization of prior knowledge of binding preferences

In deciding which positions to mutate, a researcher should use all information available (Note 1). This may include SPOT arrays, in which peptide positions are mutated to all 20 amino acids and binding is semi-quantitatively measured to the peptides synthesized on a membrane. Similarly, alanine scanning or hydrophile scanning can provide information on which positions are most important for binding.[33] Deep sequencing data from random mutagenesis libraries, or deep mutational scanning experiments can provide similar positional information.[34, 35] Lacking any experimental binding data, a structure of the peptide bound to the target is a valuable source of information (Note 2). Computational methods such as STATIUM or Rosetta can be used to generate scores for all possible peptide point mutants based on a structure of the complex, in effect generating a virtual SPOT array.[36, 37] In this section, we cover how to convert SPOT array intensities or deep sequencing data to a position specific scoring matrix (PSSM). We also discuss the use of computational tools for mutational scoring.[37]

3.1.1 Generation of a PSSM from SPOT array data—

1. SPOT array intensities can be quantified using imaging software. Several wild-type peptide spots (ideally distributed throughout the array to control for variation in exposure) can be used for normalization. In our protocols, we average the intensity of all wild-type peptide signals.
2. To compute a PSSM score for a mutation, use the following equation where M is the mutant SPOT array intensity, and W is the average wild-type intensity

$$\text{PSSM score} = \log_{10} \frac{M}{W}$$

Mutants showing weaker binding than wild type will have scores below zero, and mutants with tighter binding will have positive scores.

3. If SPOT arrays are available for the target and a competitor, a difference PSSM score can be calculated for use in a library designed for specificity. Because the range of intensities observed on SPOT arrays is likely to vary for different binding partners, each SPOT array-derived PSSM should be normalized and a Z-score calculated. The Z-score difference can then be used as a metric of specificity for each possible substitution. The equation below can be used to calculate the standardized PSSM difference, where μ is the mean intensity across each array (target, T or competitor, C), and σ is the standard deviation across each array.

$$\Delta\text{PSSM}_{T-C} = \log_{10} \left\{ \left(\frac{M - \mu}{\sigma} - \frac{W - \mu}{\sigma} \right)_T - \left(\frac{M - \mu}{\sigma} - \frac{W - \mu}{\sigma} \right)_C \right\}$$

3.1.2 Generation of a PSSM from deep sequencing data—The type of metric generated from deep sequencing data will depend on the type of experiment that it was produced from. For a single mutant dataset in which both the input and selected libraries were sequenced, variant frequencies can be converted into a PSSM-like matrix via a variety of previously published methods.[38–40] If multiple positions were mutated at once, a PSSM can be generated based on the frequencies of substitutions in unique sequences. Using unique sequences limits the biases that can arise in cell surface display or phage display datasets, e.g. from growth rate differences or background mutations. To further improve the quality of the dataset used for generating a sequencing-based model, one can limit the sequences included to those that had some minimum number of counts in the sequencing. This minimizes noise from sequencing errors. If similarly generated datasets are available for multiple binding partners, a difference PSSM score can be calculated and used as a specificity metric.

3.1.3 Use of structure-based scores—When a structure of the protein-peptide complex of interest is available, or a homology model built on a close homolog, a variety of computational scoring methods can be used to provide predictions of the effect of mutations on binding affinity. Scoring methods that can easily score all possible mutations in a peptide include STATIUM, Rosetta, FoldX, and Discovery Studio.[41] The effect of each point mutation can be calculated as the difference from the wild-type peptide score (e.g.,

$\text{STATIUM} = \text{STATIUM}_{\text{wt}} - \text{STATIUM}_{\text{mutant}}$). If structures are available for the peptide bound to the target protein and competitors, a specificity score can be computed as the difference between these relative scores (e.g., $\text{STATIUM} = \text{STATIUM}_{\text{target}} -$

$\text{STATIUM}_{\text{competitor}}$). As discussed above for the PSSM specificity scores, the range of scores for different structures may be different. Therefore, it is advisable to compute the

positional scores for all 20 amino acids at each peptide position and then normalize these scores for each structure. A Z-score difference can then be calculated as in 3.1.1.3. If the wild-type peptide binds to the target and competitors with different affinity, the score difference will be a difference in the relative effect on binding relative to wild-type, not a comparison the the affinity.

3.2 Categorization of mutations

Before designing a library on the DNA level, a researcher must first choose which peptide positions to vary and which amino-acid substitutions to favor. These choices will depend on the goal for the library screening experiment, particularly whether the goal is simply to obtain high affinity peptides for one protein target, or to obtain peptides that show both high affinity for the target and much lower binding to other proteins (competitors), in other words, specificity for the target. The length of the region to mutate can depend on physical considerations, such as how much of the peptide comes in contact with its binding partner, as well as on practical considerations, such as the length of oligonucleotides required for library assembly and the length of sequencing reads if the enriched library pools will be deep sequenced. Given advances in DNA synthesis and sequencing in recent years, most peptides will be well within standard length limits.

In our protocol, two categories of substitutions must be chosen: **required** and **preferred**. Substitutions categorized as required will always be included in the library design. Wild-type residues are generally included as required. Additional required residues may include substitutions for which there is strong evidence (experimental or computational) suggesting that they will have the desired effect on affinity or specificity. Preferred substitutions are included in the library as space and other criteria permit, as determined by the optimization algorithm. For a library designed to optimize binding affinity, preferred residues could include all residues predicted to be non-disruptive for binding to the target (neutral to beneficial). For specificity library design, the preferred set might be further narrowed to require that residues also weaken binding to competitors, according to some metric. Choices of how to define sets of required and preferred residues will depend on the data available for the protein-peptide interaction system and are ultimately made by the user. For example schemes used to designate preferred and required residues, please see references [30–32]. The number of positions input into the design process can exceed the number that will be varied in the output library design. Thus, a designer can be generous at this stage and provide information on more positions than they ultimately want to vary.

The following steps will define the two required sets of residues:

1. Make a plain text file for the required residues (see example file_req). Each line should include the wild-type residue and peptide position number followed by a list of the one-letter amino-acid codes of the residues that the designer wants to require at that position. For example, the line I4 IRY, would mean that in place of the wild-type isoleucine at position 4, the designer wants to require sampling of isoleucine, arginine, and tyrosine.

2. Make a plain text file for the preferred residues (see example file_pref). Each line lists the wild-type residue and position number followed by a list of the one-letter amino-acid codes of the preferred residues, each followed by the number 1. The preferred residues must include all of the required residues. For example, I4 I I K I R I T I A I V I L I M I Y I.

3.3 Library optimization

The choice of degenerate codons used to encode the library proceeds through two steps. First, for each peptide position, a list of all degenerate codons capable of encoding all of the required residues is output. This list is narrowed to exclude codons that encode fewer preferred residues but more trinucleotides than another codon in the list. Second, the list of possible degenerate codons at each position is fed into an ILP solver and a codon is chosen for each position such that the library score is maximized and the library size restraint is met. The default library score is the number of protein sequences encoded by the library that are composed entirely of preferred residues (i.e., the product of the number of preferred amino acids encoded by the chosen codons at each position). Users can define other scores that are linear functions of the codon choices, see section 3.5

3.3.1 Initial trimming of codons—

1. Put four files into one directory: the two files specifying the preferred and required residues (e.g. file_pref and file_req), a file containing all codons (codon_combos.txt), and the Perl script that makes the initial codon choices (writeCodon.pl).
2. In a terminal, in the directory with the four files, run the script with the following command, where “codons_output” is any name the designer chooses for the output file.

```
perl -w writeCodon.pl file_req file_pref codons_output
```

This produces a file (e.g. codons_output) that lists each peptide position specified in the required file (file_req) followed by a list of the degenerate codon choices for each position (Figure 2a). Each degenerate codon line includes the following information: degenerate codon, amino acids encoded, the number of trinucleotides encoded (codon size), the number of preferred residues encoded, and the percentage of trinucleotides that encode for preferred amino acids. Standard IUPAC nomenclature is used for the degenerate codons (<http://www.bioinformatics.org/sms/iupac.html>). Thus, the line: DNK ACDEFGIKLMNRSTVWYZ 24 9.00 0.50, is interpreted as the degenerate codon DNK encodes the amino acids ACDEFGIKLMNRSTVWYZ (Z is a stop codon) using 24 trinucleotides. This codon encodes 9 preferred amino acids, with 50% of the 24 trinucleotides encoding preferred amino acids. Note that some positions may have different codons encoding different amino-acid sets that have the same number of preferred residues and same size. The designer should look through the codons_output file for such examples and manually choose one codon to keep based on criteria such as chemical diversity or scores in the input models. If this is not done, the ILP script in the next step will simply use the first codon listed of a given size and score.

3.3.2 ILP library optimization—The script runILP.pl reads in the codons_output file created by the step above and writes out a text file with the degenerate codon chosen for each peptide position and information on the library score and size (Note 3). To make the codon choices, the ILP solver is instructed by the file library_design.mod to optimize the library score, with the constraint that the library be smaller than a specified size. The library score is the number of sequences that are entirely composed of preferred residues.

1. Put the following three files into your design directory: the codons_output file created in the step above, library_design.mod, and the Perl script that directs the ILP solver and creates the output file, runILP.pl.
2. Edit the following line in runILP.pl to include the correct path to where the ILP solver, glpsol, was installed on your computer (replace path/to):

```
my $glpsol = "/path/to/glpsol"
```

If the GLPK package was installed on your local computer using default installation settings, it will likely be located in /usr/local/bin/glpsol.

3. Edit library_design.mod to set the library size constraint (Note 4). This file contains the constraints that go into the ILP solver. Find the line “subject to totalsize: sum {v in V} costVTOT[v] * X[v], <= 7.0;”, and change 7.0 to another number (e.g. for a library size constraint of 10^5 change to 5.0). This library size is the size in DNA sequences. A good rule-of-thumb is to set this 10-fold lower than the maximum transformation efficiency or screening throughput of the library-screening platform to be used, in order to sample most of the library.
4. Run the ILP optimization:

```
perl -w runILP.pl codons_output library_design_output
```

The file “library_design_output” is whatever the designer chooses to name their output file, and codons_output is the file output by writeCodon.pl.

If a solution is found, the standard output will say “Optimal solution found”, and a text file with the library design will be created (e.g. library_design_output_example). An example library design output is shown in Figure 2b, with an explanation of the outputted metrics given in the figure legend. This is a very fast process, completed in <1 second on a standard laptop. If a solution is not found, there is no solution possible that encodes all of the required residues within the library size constraint. The standard output will say “Problem has no feasible solution”. In this situation, the designer will need to go back to the categorization of mutations step and reduce the number of required mutations, or increase the library size constraint and run the codon trimming and ILP steps again.

Once a library design is output, the basic process is complete, and the designer can order oligonucleotides encoding the library from a DNA synthesis company using machine mixing of nucleotides to encode the degenerate codons. Below, we present further steps that a designer can take to analyze the predicted characteristics of the library to inform modifications for further rounds of the library design process. Additionally, we provide a more complex protocol that allows consideration of chemical diversity and optimization

based directly on a user-defined scoring system, rather than on the number of preferred residues.

3.4 Analysis of library designs

Designed libraries can be evaluated on several levels to get an idea of the predicted performance for the affinity or specificity objective. Adjustments can be made to the inputted lists of preferred and required residues, as well as the initial codon lists, in order to improve predicted performance. Predictors of performance or library quality include the simple statistics output by the library design script, or more in-depth analysis of the scores of all theoretical library sequences based on quantitative models available for the peptide interaction system.

A first-pass library design analysis would look at the statistics output by the library design script including the library score, the number of protein sequences encoded, and the percent of the library that is predicted to be useful (i.e., the percentage of the DNA sequences that encode only preferred residues). To maximize these statistics, you can change the amino acids you list as preferred and required, or alter the codon choices by manually editing the `codons_output` file before running the ILP script. Some combinations of required amino acids may necessitate the choice of large codons, which may include many amino acids that could be disruptive for binding, lowering the fraction of the library that is predicted to be useful. Consider whether all of the required residues are necessary, or if you can require a chemically similar residue that allows the choice of a smaller codon. Alternatively, if you are willing to use multiple oligonucleotides to construct your library, you can use more than one codon at a position, as done by Chen et al.[30]

It is also important to consider the mutational load of your library, or how many positions are varied. Previous studies have found that a higher mutational load in intelligently designed libraries correlates with a greater chance of success.[23] If some positions turn out to not contribute as much to affinity or specificity as predicted by an input model, then allowing a few amino-acid choices at many positions will provide a better chance of success than allowing a large diversity of amino acids at a few positions. However, it is also important to consider how many potentially disruptive mutations you are including. For example, if a given position includes a choice between just two amino acids, and one of these disrupts binding, then half of the library will not bind. To spread the diversity of your library across many positions, you can adjust the size of codons chosen. Manually edit the `writeCodons.pl` output (`codons_output`) to remove large codons (particularly codons that encode stop codons or potentially disruptive amino acids) at positions where a large amount of diversity is not a high priority, and then re-run the ILP optimization on the edited file. Focus diversity on positions that are most likely to impart high affinity or specificity, based on the information available for the peptide system.

If quantitative scoring models are available for your peptide system (e.g., PSSMs based on SPOT arrays, or fast-to-evaluate structure-based scores), these models can be used to score the theoretical library and predict how many sequences are likely to have the desired affinity or specificity characteristics. First, write out the theoretical library by creating sequences for all possible combinations of the amino acids at each position in the library design. Then,

compute the score for each sequence as the sum of the scores for each position in the peptide. If you are designing a library for specificity and have models for competitor interaction partners, you can score the theoretical library on those models for comparison and analysis of predicted specificity. The score of the wild-type peptide sequence that the library is based upon can be used as a cutoff to calculate the proportion of the library that has wild-type-like or greater affinity. For analysis of a library designed for affinity, a simple histogram can be used to visualize the distribution of library scores. For specificity library design, we use two-dimensional histograms (density plots) to compare the library sequence scores for the target and competitor interaction partners.[31]

We recently designed and enriched a BH3 peptide library for specific binding to the viral Bcl-2 homolog KSBcl-2 over the competitor human Bcl-2 homologs.[32] In evaluating different library designs, we scored the libraries using PSSMs derived from SPOT arrays for KSBcl-2 or human Bcl-2 homologs binding to BH3 peptide mutants. Figure 3 shows the scores for the theoretical library in gray scale density plots. The wild-type peptide scores are marked with lines, creating quadrants. The proportion of the library that was predicted to have both greater affinity than wild type for KSBcl-2 and weaker binding to one of two human homologs (Mcl-1 in panel 3a and Bcl-x_L in panel 3b) falls in the lower right quadrant. We went through several rounds of refinement of our choice of preferred and required residues, and additionally edited the codons_output file in order to maximize the number of sequences that fell in this quadrant. Sequences from clones that survived experimental enrichment for KSBcl-2-specific binding are overlaid in red, and a small selection of peptides that were directly tested in solution binding assays and shown to bind preferentially to KSBcl-2 are shown in blue. These pools that are enriched in specific sequences cluster near the lower right quadrant lending support for this approach to library design optimization. For more details, see Foight & Keating.[32]

3.5 Alternative library optimization protocols

More complex library optimization strategies can be envisioned. In this section we present an additional set of scripts and example files (in MultiOption.tar.gz) that follow the same basic approach as in section 3.3, but allow consideration of chemical diversity and permit optimization based directly on positional scores. Chemical diversity criteria were used to favor codons with more chemically diverse sets of amino acids in the design of a peptide library to bind specifically to Bfl-1.[31] In analogy to optimizing the number of sequences that contain only preferred residues, optimization can be done using any score that can be converted to a PSSM, i.e, a table with peptide positions as columns and scores for all 20 amino acids as rows. In the example that we present here, we use a table with the frequencies of the 20 amino acids at 21 peptide positions in a multiple sequence alignment (MSA) of BH3 peptides. This library design favors sequences composed of residues with high frequencies, as outlined below, but users can choose their own PSSM. The library design protocol can be run in six different modes: there are three scoring modes, each of which can be used with or without consideration of chemical diversity. The scoring modes are: (1) the count of sequences composed entirely of preferred amino acids (“preferred score”, i.e, that used in section 3.3), (2) the MSA frequencies-based score (here referred to as the MSA score), or (3) a sum of the preferred and MSA scores. The scripts included

require input of four criteria (chemical diversity classes, preferred and required amino acids, and another score) regardless of which mode is being used.

3.5.1 Initial trimming of codons—

1. Make input files for writeCodon_MOp.pl, which does the initial trimming of codons based on size and score(s). Make two files of preferred and required residues as in section 3.2 (example files file_pref_MOp, file_req_MOp). Make a comma-separated value file (csv) of your PSSM-formatted scores (example file BSA_MSA_table.csv). This can be constructed in Excel and saved as a .csv file. The position names in the column header of the .csv file should be of the format: number, lowercase letter (e.g., '2a'). The position names in the required and preferred files should be of the format: capital letter of wild-type amino acid, number, lowercase letter (e.g. 'E2a'). The regular expressions that recognize the position names in both Perl scripts need to be changed if a different position naming convention is used.
2. Edit the chemical diversity classes set in writeCodon_MOp.pl. Open the file in a text editor and go to the section with the header “# settings for chemical diversity classes”. Replace the example positions and sets with your own position names and amino-acid sets. Create a line for each peptide position, again, using the same position nomenclature as used in step 1. When using the chemical diversity criteria, the ILP solver will count the number of “misses” in chemical diversity classes for each codon. The user will set a constraint on the maximum number of misses to allow across all positions, and the solver will fail if it can not find a solution that meets the library size and chemical diversity misses constraints.
3. Put all of the input files and the script into the same directory (file_pref_MOp, file_req_MOp, BSA_MSA_table.csv, codon_combos.txt, and writeCodon_MOp.pl). In a terminal, in that directory, run:

```
perl -w writeCodon_MOp.pl file_req_MOp file_pref_MOp
BSA_MSA_table.csv codons_output_MOp
```

The file codons_output_MOp is the outputted list of positions and codon selections. An example codon line is: NDS CDEFGHIKLMNQRSVWYZ 24 15.00 0.80 0 0.88. The information included is, from left to right: degenerate codon, amino acids encoded, number of trinucleotides, number of preferred amino acids, the sum of the MSA scores for all of the amino acids included, number of chemical diversity class misses, and the fraction of trinucleotides encoding preferred amino acids.

3.5.2 Run the ILP optimization—The library optimization script runILP_MOp.pl can be run in the six modes described above. As in section 3.3, it outputs the library design that maximizes the score while meeting the library size constraint set by the user, with an optional constraint of number of chemical diversity class misses. The “score” output by the runILP is the raw score optimized by the ILP, which varies depending on the mode that you are running. For preferred score only, the score is the same as in section 3.3, $\log_{10}(\text{number of sequences composed of preferred residues})$. For MSA score only, the score is

$\log_{10}$ (product of MSA score for each codon). The MSA score for each codon is the sum of the MSA scores for all amino acids included in that codon. When using both scores, “score” is the sum of both, each on a $\log_{10}$ scale. Note that the MSA score and preferred score may be on different scales, which will affect which score dominates the optimization when using both. If possible, rescale so that the magnitudes are similar, or only use one scoring method at a time.

1. Set up the runILP_MOp.pl script according to what mode you want to run it in. Six lines in the script are preceded by the header “###EDIT###”.
 - i. The first line includes the name of the library_design.mod file, which contains the library size and chemical diversity misses constraints; this directs the glpsol. Change this file name to library_design_CD_enabled.mod if using chemical diversity, or library_design_CD_disabled.mod if not using chemical diversity.
 - ii. The second line to edit tells the script to use chemical diversity or not. Set the variable “\$use_chemical_diversity” equal to 1 if using chemical diversity, or 0, if not.
 - iii. Edit the third line to contain the correct path to the glpsol on your machine.
 - iv. Set the scoring mode. A set of three lines preceded by “###EDIT###” start at line number 120 in the script. Comment out (add a “#” at the beginning of the line) the two scoring modes that you do not want to use. The choices from top to bottom are: preferred score only, MSA score only, or both.
2. If using chemical diversity, edit library_design_CD_enabled.mod to set the constraint on the maximum number of chemical diversity class misses to allow. Go to the line “subject to trs: sum { v in V } costVTRS[v] * X[v], <= 5.0;” and set the number at the end to the number of misses to allow. For example, for a peptide in which 10 positions are being varied, 5–10 misses would be a reasonable place to start. The number of misses will depend on how many chemical diversity classes you set in the writeCodon_MOp.pl script.
3. Run the ILP optimization. Include the following files in the directory you are working in: codons_output_MOp, library_design_CD_enabled/disabled.mod, and runILP_MOp.pl. In a terminal enter:

```
perl -w runILP_MultiOp.pl codons_output_MOp
library_design_output_MOp
```

If the ILP solver finds a solution, the library design will be output to library_design_output_MOp (or whatever you decide to name the output file). If no solution is possible, the library size constraint, the chemical diversity misses constraint, or the numbers of positions varied and required residues should be adjusted. The format of the library design output is the same as shown in Figure 2, with the exception that the “Score” value will correspond to one of the $\log_{10}$ values as described above, depending on which

scoring method is used. The number of sequences composed entirely of preferred amino acids (the “Score” for the original optimization method presented in section 3.3) is given as a separate value. Running this protocol without considering chemical diversity and using only the preferred score is equivalent to the protocol described in section 3.3.

Notes

- When the objective is to obtain a peptide with binding specificity for a target protein over competitor proteins, careful consideration of affinity and specificity trade-offs must be made at both the library design and experimental screening stages. To obtain specificity between very similar target and competitor proteins, you may need to include residues that impart specificity but are predicted to be somewhat disruptive for binding the target. However, at the screening stage, if the stringency for binding to the target is too great, these mutations may not make it through the screen.
- Computational models can provide hypotheses about binding at positions for which there is no experimental data. Modeling can be especially valuable for specificity predictions, because experimental data may be limited to the peptide positions that are most important for affinity. However, positions other than such conserved “motif” residues, including residues near peptide termini, are often important for specificity.[32, 42]
- The runILP.pl script creates temporary input and output files that are deleted at the end of the script. The temporary input file contains the codon scores and is used by the glpsol. The temporary output file generated by glpsol contains the information that is processed by runILP.pl to create the library design output. If you would like to see these files, comment out the line in runILP.pl below “#Include to delete temporary input and output files”.
- The library_design.mod script is based on a script for integer linear programming optimization of rotamer choice by Kingsford et al.[43]

Acknowledgments

This work was supported by the National Institutes of General Medical Sciences award R01 GM110048 to AEK.

References

1. Chen TS, Keating AE. Designing specific protein-protein interactions using computation, experimental library screening, or integrated methods. *Protein Sci.* 2012; 21:949–963. DOI: 10.1002/pro.2096 [PubMed: 22593041]
2. Liu BA, Engelmann BW, Nash PD. High-throughput analysis of peptide-binding modules. *Proteomics.* 2012; 12:1527–1546. DOI: 10.1002/pmic.201100599 [PubMed: 22610655]
3. Levin AM, Weiss GA. Optimizing the affinity and specificity of proteins with molecular display. *Molecular BioSystems.* 2006; 2:49–57. DOI: 10.1039/b511782h [PubMed: 16880922]
4. Olivos HJ, Bachhawat Sikder K, Kodadek T. Quantum Dots As A Visual Aid For Screening Bead-Bound Combinatorial Libraries. *Chembiochem.* 2003; 4:1242–1245. DOI: 10.1002/cbic.200300712 [PubMed: 14613120]

5. Rezaei Araghi R, Ryan JA, Letai A, Keating AE. Rapid Optimization of Mcl-1 Inhibitors using Stapled Peptide Libraries Including Non-Natural Side Chains. *ACS Chem Biol*. 11:1238–1244. DOI: 10.1021/acscchembio.5b01002
6. Goldsmith M, Tawfik DS. Enzyme engineering by targeted libraries. *Meth Enzymol*. 2013; 523:257–283. DOI: 10.1016/B978-0-12-394292-0.00012-6 [PubMed: 23422434]
7. Hou T, Li N, Li Y, Wang W. Characterization of domain-peptide interaction interface: prediction of SH3 domain-mediated protein-protein interaction network in yeast by generic structure-based models. *J Proteome Res*. 2012; 11:2982–2995. DOI: 10.1021/pr3000688 [PubMed: 22468754]
8. London N, Lamphear CL, Hougland JL, Fierke CA, Schueler-Furman O. Identification of a novel class of farnesylation targets by structure-based modeling of binding specificity. *PLoS Comput Biol*. 2011; 7:e1002170.doi: 10.1371/journal.pcbi.1002170 [PubMed: 21998565]
9. DeBartolo J, Taipale M, Keating AE. Genome-wide prediction and validation of peptides that bind human prosurvival Bcl-2 proteins. *PLoS Comput Biol*. 2014; 10:e1003693.doi: 10.1371/journal.pcbi.1003693 [PubMed: 24967846]
10. Sánchez IE, Beltrao P, Stricher F, Schymkowitz J, Ferkinghoff-Borg J, Rousseau F, Serrano L. Genome-wide prediction of SH2 domain targets using structural information and the FoldX algorithm. *PLoS Comput Biol*. 2008; 4:e1000052.doi: 10.1371/journal.pcbi.1000052 [PubMed: 18389064]
11. Fernandez-Ballester G, Beltrao P, Gonzalez JM, Song Y-H, Wilmanns M, Valencia A, Serrano L. Structure-based prediction of the *Saccharomyces cerevisiae* SH3-ligand interactions. *Journal of Molecular Biology*. 2009; 388:902–916. DOI: 10.1016/j.jmb.2009.03.038 [PubMed: 19324052]
12. Fu X, Apgar JR, Keating AE. Modeling backbone flexibility to achieve sequence diversity: the design of novel alpha-helical ligands for Bcl-xL. *Journal of Molecular Biology*. 2007; 371:1099–1117. DOI: 10.1016/j.jmb.2007.04.069 [PubMed: 17597151]
13. Roberts KE, Cushing PR, Boisguerin P, Madden DR, Donald BR. Computational design of a PDZ domain peptide inhibitor that rescues CFTR activity. *PLoS Comput Biol*. 2012; 8:e1002477.doi: 10.1371/journal.pcbi.1002477 [PubMed: 22532795]
14. Zheng F, Jewell H, Fitzpatrick J, Zhang J, Mierke DF, Grigoryan G. Computational design of selective peptides to discriminate between similar PDZ domains in an oncogenic pathway. *Journal of Molecular Biology*. 2015; 427:491–510. DOI: 10.1016/j.jmb.2014.10.014 [PubMed: 25451599]
15. Sammond DW, Bosch DE, Butterfoss GL, Purbeck C, Machius M, Siderovski DP, Kuhlman B. Computational Design of the Sequence and Structure of a Protein-Binding Peptide. *J Am Chem Soc*. 2011; 133:4190–4192. DOI: 10.1021/ja110296z [PubMed: 21388199]
16. Fleishman SJ, Whitehead TA, Ekiert DC, Dreyfus C, Corn JE, Strauch E-M, Wilson IA, Baker D. Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. *Science*. 2011; 332:816–821. DOI: 10.1126/science.1202617 [PubMed: 21566186]
17. Procko E, Hedman R, Hamilton K, Seetharaman J, Fleishman SJ, Su M, Aramini J, Kornhaber G, Hunt JF, Tong L, Montelione GT, Baker D. Computational design of a protein-based enzyme inhibitor. *Journal of Molecular Biology*. 2013; 425:3563–3575. DOI: 10.1016/j.jmb.2013.06.035 [PubMed: 23827138]
18. Strauch E-M, Fleishman SJ, Baker D. Computational design of a pH-sensitive IgG binding protein. *Proc Natl Acad Sci USA*. 2014; 111:675–680. DOI: 10.1073/pnas.1313605111 [PubMed: 24381156]
19. London N, Raveh B, Schueler-Furman O. Peptide docking and structure-based characterization of peptide binding: from knowledge to know-how. *Curr Opin Struct Biol*. 2013; 23:894–902. DOI: 10.1016/j.sbi.2013.07.006 [PubMed: 24138780]
20. Parker AS, Griswold KE, Bailey-Kellogg C. Optimization of combinatorial mutagenesis. *J Comput Biol*. 2011; 18:1743–1756. DOI: 10.1089/cmb.2011.0152 [PubMed: 21923411]
21. Pantazes RJ, Saraf MC, Maranas CD. Optimal protein library design using recombination or point mutations based on sequence-based scoring functions. *Protein Engineering Design and Selection*. 2007; 20:361–373. DOI: 10.1093/protein/gzm030
22. Chen MMY, Snow CD, Vizcarra CL, Mayo SL, Arnold FH. Comparison of random mutagenesis and semi-rational designed libraries for improved cytochrome P450 BM3-catalyzed hydroxylation

- of small alkanes. *Protein Eng Des Sel*. 2012; 25:171–178. DOI: 10.1093/protein/gzs004 [PubMed: 22334757]
23. Treynor TP, Vizcarra CL, Nedelcu D, Mayo SL. Computationally designed libraries of fluorescent proteins evaluated by preservation and diversity of function. *Proceedings of the National Academy of Sciences*. 2007; 104:48–53. DOI: 10.1073/pnas.0609647103
24. Verma D, Grigoryan G, Bailey-Kellogg C. Structure-based design of combinatorial mutagenesis libraries. *Protein Sci*. 2015; 24:895–908. DOI: 10.1002/pro.2642 [PubMed: 25611189]
25. Jacobs TM, Yumerefendi H, Kuhlman B, Leaver-Fay A. SwiftLib: rapid degenerate-codon-library optimization through dynamic programming. *Nucleic Acids Res*. 2015; 43:e34–e34. DOI: 10.1093/nar/gku1323 [PubMed: 25539925]
26. Mena MA, Daugherty PS. Automated design of degenerate codon libraries. *Protein Engineering Design and Selection*. 2005; 18:559–561. DOI: 10.1093/protein/gzi061
27. Allen BD, Nisthal A, Mayo SL. Experimental library screening demonstrates the successful application of computational protein design to large structural ensembles. *Proceedings of the National Academy of Sciences*. 2010; 107:19838–19843. DOI: 10.1073/pnas.1012985107
28. Hayes RJ, Bentzien J, Ary ML, Hwang MY, Jacinto JM, Vielmetter J, Kundu A, Dahiyat BI. Combining computational and experimental screening for rapid optimization of protein properties. *Proc Natl Acad Sci USA*. 2002; 99:15926–15931. DOI: 10.1073/pnas.212627499 [PubMed: 12446841]
29. Guntas G, Purbeck C, Kuhlman B. Engineering a protein-protein interface using a computationally designed library. *Proc Natl Acad Sci USA*. 2010; 107:19296–19301. DOI: 10.1073/pnas.1006528107 [PubMed: 20974935]
30. Chen TS, Palacios H, Keating AE. Structure-based redesign of the binding specificity of anti-apoptotic Bcl-x(L). *Journal of Molecular Biology*. 2013; 425:171–185. DOI: 10.1016/j.jmb.2012.11.009 [PubMed: 23154169]
31. Dutta S, Chen TS, Keating AE. Peptide ligands for pro-survival protein Bfl-1 from computationally guided library screening. *ACS Chem Biol*. 2013; 8:778–788. DOI: 10.1021/cb300679a [PubMed: 23363053]
32. Foight GW, Keating AE. Locating Herpesvirus Bcl-2 Homologs in the Specificity Landscape of Anti-Apoptotic Bcl-2 Proteins. *Journal of Molecular Biology*. 2015; doi: 10.1016/j.jmb.2015.05.015
33. Boersma M, Sadowsky J, Tomita Y. Hydrophile scanning as a complement to alanine scanning for exploring and manipulating protein-protein recognition: Application to the Bim BH3 domain. *Protein Sci*. 2008; 17:1232–1240. DOI: 10.1110/ps.032896.107 [PubMed: 18467496]
34. Fowler DM, Araya CL, Fleishman SJ, Kellogg EH, Stephany JJ, Baker D, Fields S. High-resolution mapping of protein sequence-function relationships. *Nat Methods*. 2010; 7:741–746. DOI: 10.1038/nmeth.1492 [PubMed: 20711194]
35. Fowler DM, Fields S. Deep mutational scanning: a new style of protein science. *Nat Methods*. 2014; 11:801–807. DOI: 10.1038/nmeth.3027 [PubMed: 25075907]
36. London N, Gullá S, Keating AE, Schueler-Furman O. In silico and in vitro elucidation of BH3 binding specificity toward Bcl-2. *Biochemistry*. 2012; 51:5841–5850. DOI: 10.1021/bi3003567 [PubMed: 22702834]
37. DeBartolo J, Dutta S, Reich L, Keating AE. Predictive Bcl-2 family binding models rooted in experiment or structure. *Journal of Molecular Biology*. 2012; 422:124–144. DOI: 10.1016/j.jmb.2012.05.022 [PubMed: 22617328]
38. Araya CL, Fowler DM, Chen W, Muniez I, Kelly JW, Fields S. A fundamental protein property, thermodynamic stability, revealed solely from large-scale measurements of protein function. *Proc Natl Acad Sci USA*. 2012; 109:16858–16863. DOI: 10.1073/pnas.1209751109 [PubMed: 23035249]
39. Fowler DM, Araya CL, Gerard W, Fields S. Enrich: software for analysis of protein function by enrichment and depletion of variants. *Bioinformatics*. 2011; 27:3430–3431. DOI: 10.1093/bioinformatics/btr577 [PubMed: 22006916]

40. Starita LM, Young DL, Islam M, Kitzman JO, Gullingsrud J, Hause RJ, Fowler DM, Parvin JD, Shendure J, Fields S. Massively Parallel Functional Analysis of BRCA1 RING Domain Variants. *Genetics*. 2015; doi: 10.1534/genetics.115.175802
41. Sirin S, Apgar JR, Bennett EM, Keating AE. AB-Bind: Antibody binding mutational database for computational affinity predictions. *Protein Sci* n/a–n/a. 2015; doi: 10.1002/pro.2829
42. Stein A, Aloy P. Contextual specificity in peptide-mediated protein interactions. *PLoS ONE*. 2008; 3:e2524.doi: 10.1371/journal.pone.0002524 [PubMed: 18596940]
43. Kingsford CL, Chazelle B, Singh M. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*. 2005; 21:1028–1036. DOI: 10.1093/bioinformatics/bti144 [PubMed: 15546935]

Figure 1.

Flowchart of the library design process. The first two steps of gathering information from binding experiments or structure-based models and prioritizing substitutions will depend on the information available for the protein-peptide interaction of interest, so we present some general guidelines. The optimization of a degenerate codon library to encode the desired substitutions then proceeds in two parts: initial trimming of the codon choices based on codon size and score, followed by ILP library optimization to yield a library of a desired size with an optimal score. Finally, suggestions are given for analysis of the predicted behavior of the library based on input models, which can inform further rounds of library design to improve predicted library characteristics.

Figure 2.

Output of the initial codon trimming step and the ILP library optimization step. (a) An example of the degenerate codon choices for one position as output by writeCodon.pl. The columns are labeled with their corresponding properties. (b) An example library design output by runILP.pl. The three columns are position, degenerate codon, and amino acids encoded. The total size in DNA sequences is under the limit set by the user (in this case 10^7). The total size in protein sequences is the product of the number of amino acids encoded by each chosen codon. The score is the optimized value, the number of protein sequences composed entirely of preferred amino acids. The useful fraction is the product of the fraction of trinucleotides encoding preferred amino acids for each chosen codon.

Figure 3.

Analysis of sequence scores for a library designed and then screened for specificity. Predicted KSBcl-2 binding is shown on the x-axis. Predicted binding to competitors Mcl-1 (a) or Bcl-x_L (b) are shown on the y-axis. A density plot of scores for the theoretical library is shown in gray scale. Scores for sequences from a library pool enriched for binding specificity to KSBcl-2 are overlaid in red. The blue points are for peptides that were tested in solution binding experiments and showed at least some margin of specificity for KSBcl-2 binding.