UniDesign Manual

By Xiaoqiang Huang, Ph.D.

Updated: Dec/05/2020

Content

Overview and features

UniDesign is a command-line-based Unified platform for protein Design, aiming at all kinds of protein design and engineering tasks, as well as for protein structure modeling and scoring. UniDesign is mainly extended from EvoEF2 (https://zhanglab.ccmb.med.umich.edu/EvoEF/) and in part from EvoDesign (https://zhanglab.ccmb.med.umich.edu/EvoDesign/) but has many new features.

UniDesign can be used to do but is not limited to the following tasks:

- Protein design (de novo protein sequence design and redesign):
  *) design monomer proteins
  *) design protein-protein interactions
  *) design protein-ligand interactions
  *) design protein-nucleic acid interactions
  *) design enzymes
- Protein structure modeling applications:
  *) protein side-chain packing
  *) repair missing protein side chains
  *) protein structure energy minimization
  *) build mutant structural models
  *) add polar hydrogen atoms
  *) optimize hydrogen atom's position
- The scoring applications include:
  *) calculate protein fold stability score
  *) calculate protein-protein binding energy score

License

Software availability

The standalone UniDesign program and the source code are freely available to users.

Why UniDesign is developed?

UniDesign is developed based on two other protein design programs developed in the lab, i.e. EvoDesign[1,2] and EvoEF2[3].

EvoDesign is an evolutionary-profile-based approach to de novo protein sequence design for monomers and protein-protein interactions. The first version of EvoDesign was released in 2013[1]. At that time, for protein design simulations, EvoDesign utilized an external protein side-chain packing program named SCWRL4[4] to construct the structural model for a sequence decoy on a fixed protein backbone and then used a composite scoring function which combined an evolution-based potential (i.e. position-specific scoring matrix, PSSM) and an external empirical scoring function named FoldX[5] to score the models generated by SCWRL4. A Replica-Exchange Monte-Carlo (REMC) simulation was used to accept or reject the sequence decoys through the Metropolis criterion. A large number of low-energy (i.e. the FoldX score) sequence decoys generated in the REMC simulation were then subjected to clustering based on sequence similarity calculated using the BLOSUM62 matrix. The sequences at the largest cluster centers were selected, analyzed, and further screened for wet-lab experimental test. The first version of EvoDesign was only able to design monomers. In 2019, a newer version of EvoDesign has been developed to design protein complexes to enhance protein-protein binding interaction and the previous packer SCWRL4 was replaced by a much faster one called RASP[6].

EvoDesign calls a few third-party programs (i.e. SCWRL4, RASP, and FoldX) for the two major steps in computational protein design, i.e. model construction and scoring. Since the source code of these external programs is not available, it is difficult to extend them for specific tasks. For example, RASP cannot be used to deal with a protein-ligand interaction design because a custom ligand cannot be appropriately modeled. For protein design scoring, FoldX has been only optimized for predicting the protein stability change ($ddG_{stability}$) upon mutations[7], but its ability for de novo protein sequence design (or complete redesign) has not been widely evaluated and reported[8]. On the other hand, FoldX has not been fully optimized for modeling and designing protein-protein interactions[2]. Besides, the computational speed of FoldX is not satisfactory[2]. To address some of these issues, we developed a physics- and knowledge-based scoring function called EvoEF

(EvoDesign Energy Function)[2,3] to replace FoldX. The first version of EvoEF (EvoEF1) was optimized following a similar strategy to FoldX, i.e. to achieve the best performance for predicting the thermodynamic change data such as $ddG_{stability}$ and the protein-protein binding affinity change ($ddG_{binding}$) upon mutations. Based on our tests, EvoEF1 runs approximately three to five times faster than FoldX on protein stability and protein-protein binding affinity calculation, and the benchmark on two large-scale data sets showed that EvoEF1 could outperform FoldX on the ddG prediction by achieving a higher Pearson correlation coefficient (PCC) and a lower root-mean-square error (RMSE). However, we found that EvoEF1 exhibited a poor performance for de novo protein sequence design based on the so-called sequence recapitulation test and the designed sequences were not very native-like[3]. Therefore, a second version of EvoEF (EvoEF2) was developed and specifically optimized for de novo protein sequence design by incorporating more energy terms; based on our benchmark, EvoEF2 could achieve a comparable sequence recapitulation rate compared with the other state-of-the-art protein design approaches[3].

It should be noted that the previous versions of EvoEF are an energy function (rather than a protein design program) developed only for modeling/scoring proteins (monomers and protein-protein complexes) with the twenty canonical amino acids. And thus, the EvoEF and EvoDesign programs are only limited to design protein molecules. However, there is an increasing demand from ourselves and other users for extending them to perform a unified protein design task that can handle, ideally, all kinds of molecules, such as water molecules, ions, small-molecule ligands, and nucleic acids. In this regard, we extended EvoEF2 and EvoDesign into a more powerful protein design program (UniDesign) for Unified protein Design, aiming at different protein design and engineering tasks, such as designing monomers, protein-protein interactions, protein-ligand interactions, protein-nucleic acid interactions, and enzymes. In UniDesign, a simulated annealing Monte Carlo (SAMC) procedure was used to explore the sequence space for fixed protein backbones, where the MC movement consisted of exchanging one rotamer for another at a randomly chosen position. An MC move was accepted or rejected according to the Metropolis criterion. By doing this, we can completely remove the external packers such as SCWRL4 and RASP, and side-chain packing is realized by random rotamer substitution during the simulation. This architecture makes UniDesign extremely efficient for protein design, compared with EvoDesign. The SAMC procedure was very fast, and for instance, it took less than 15 minutes to completely redesign a protein that is about 200 amino acids long. The lowest-energy sequence is output as the best design for a single run of UniDesign. Users can run multiple independent design trajectories to obtain a pool of low-energy sequences for subsequent analysis and screening for experimental validation.

Platform and installation

Windows

Method 1

In the UniDesign package, we have provided an executable program named UniDesign.exe which is compiled in a Windows 7 x64 machine using the g++ compiler version 8.2.0. If you use a Windows 7 or higher machine, you should be able to directly run the precompiled UniDesign.exe in the Microsoft Windows command-line (cmd.exe).

Open the cmd.exe, and locate into the UniDesign directory, and run:

```
UniDesign.exe -h
```

If successful, the UniDesign help document will be printed out on your screen. Then you can run the other commands of UniDesign using “UniDesign.exe --command=CommandName [… other options …]”.

Method 2

To my knowledge, it is not very convenient to use the Windows command line. In this situation, it is better to install a third-party tool that can run UniDesign in command lines. Git (https://git-scm.com/) is a good choice and can be used for managing your code. With the g++ compiler installed, you can rebuild the UniDesign.exe for your system, which may bring you a higher computational efficiency and fewer bugs.

Other options

To achieve the highest running speed of UniDesign in Windows, you may want to build the executable UniDesign.exe program by yourself. it is necessary to install a g++ compiler first. You can make the installation through MinGW (http://www.mingw.org/) or Cygwin (https://www.cygwin.com/). After you complete the installation of the g++ compiler and set the environmental variable. You can test if the g++ compiler has been successfully installed with 'g++ -v' in the cmd.exe control console. If successful, it will show messages similar to:

```
"Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=c:/mingw/bin/../libexec/gcc/mingw32/8.2.0/lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-8.2.0/configure --build=x86_64-pc-linux-gnu \
--host=mingw32 --target=mingw32 --prefix=/mingw --disable-win32-registry \
--with-arch=i586 --with-tune=generic --enable-languages=c,c++,objc,obj-c++,fortran,ada \
--with-pkgversion='MinGW.org GCC-8
Thread model: win32
gcc version 8.2.0 (MinGW.org GCC-8.2.0-3)"
```

Otherwise, it will say that g++ cannot be found. You should check your installation. If you try many times but still cannot install the program successfully, please contact report problems to the emails listed below.

Linux

It is much convenient to use UniDesign in a Linux system. Download the UniDesign package, go to the main directory of UniDesign and simply run the command “g++ -O3 --fast-math -o UniDesign src/*.cpp” or directly run the ‘build’ bash file to build the executable UniDesign program into the main directory. If the command does not work, try without the '--fast-math' option.

Mac

Building UniDesign in Mac is similar to that in Linux, try the command “g++ -O3 -fast-math -o UniDesign src/*.cpp” or simply “g++ -O3 -o UniDesign src/*.cpp” in the Mac terminal to build the executable UniDesign program.

Usage

Summary

In general, UniDesign is run in the following syntax:

```
./UniDesign --command=CommandName    [--options]
```

Run ‘./UniDesign -h’ or ‘./UniDesign -?’ or ‘./UniDesign --help’ to know more about the supported commands and options.

Design monomer protein

```
./UniDesign --command=ProteinDesign --monomer --pdb=proteinID.pdb --design_chains=A --
resfile=RESFILE.txt --ntraj=10
```

For monomer design, the option ‘--monomer’ and ‘--design_chains’ can be discarded. ‘--ntraj’ specifies how many independent protein design trajectories will be computed. RESFILE.txt designates the amino acid positions for designing, repacking, or fixing; more will be introduced about the format of RESFILE.txt in the Resfile section.

“proteinID.pdb” is the PDB file name. UniDesign will parse the “proteinID” and uses it as a prefix for the output file. If the above command is run successfully, 10 designs will be generated with file names:

ProteinID_bestseqs.txt	Collection of the designed lowest-energy sequences
ProteinID_bestsites0001.pdb **… …** **ProteinID_bestsites0010.pdb** **ProteinID_beststruct0001.pdb** **… …** **ProteinID_beststruct0010.pdb**	The design sites (*bestsites*) and the full protein structures (*beststruct*) for the corresponding lowest-energy designs; for de novo design, the corresponding bestsites file and beststruct file are identical.
ProteinID_desrots0001.txt … …	Intermediate/recording files for the design simulation; *desseqs* record the accepted

ProteinID_desrots0010.txt ProteinID_desseqs0001.txt … … ProteinID_desseqs0010.txt ProteinID_rotlist.txt ProteinID_rotlistSEC.txt ProteinID_selfenergy.txt	sequence decoys during simulated annealing Monte Carlo simulation, and *desrots* record the rotamer indices for the corresponding accepted sequences.

The files highlighted in bold are important design result files. A detailed explanation for the designed sequences is shown at the head of the proteinID_bestseqs.txt file.

Design protein-protein interaction

```
./UniDesign --command=ProteinDesign --ppint --pdb=proteinID_AB.pdb --design_chains=A --
resfile=RESFILE.txt    --ntraj=10
Or:
./UniDesign --command=ProteinDesign --ppint --pdb=proteinID_ABC.pdb --design_chains=AB
--resfile=RESFILE.txt --ntraj=10
```

Similar output files will be generated for protein-protein interaction design. The first example designs chain A for a dimer AB. The second example designs chains AB simultaneously for a trimer ABC.

Design protein-nucleic acid interaction

```
./UniDesign --command=ProteinDesign --ppint --pdb=protein-NA.pdb --design_chains=A --
resfile=RESFILE.txt --ntraj=10
```

Here, chain A is a protein chain. In the design process, the side-chain conformations of nucleotides are kept constant. Currently, protein-nucleic acid interaction design is treated in the same way as protein-protein interaction design.

Design protein-ligand interaction

For protein-ligand interaction design, the atom-parameter and residue-topology file must be generated before running the 'ProteinDesign' command:

To generate the two files, run:

```
./UniDesign --command=GenLigParamAndTopo --mol2=ligand.mol2 --
lig_atomparam=ligand_atomparam.prm --lig_topology=ligand_topology.inp
```

The ligand.mol2 file should include all atoms of the ligand, including hydrogen atoms. Please make sure that the ligand atomic charges are appropriately assigned to the last column in the @<TRIPOS>ATOM section. If the above command is run successfully, the atom-parameter and residue-topology file will be created into the specified files.

Then run:

```
./UniDesign --command=ProteinDesign --protlig --pdb=proteinID.pdb --mol2=ligand.mol2 --
```

```
read_lig_ensemble=LIGAND_CONFORMERS.pdb    --resfile=RESFILE.txt --ntraj=10
```

The ligand conformers in the LIGAND_CONFORMERS.pdb file can be generated using a typical molecular docking tool such as AutoDock Vina. Unlike the ligand.mol2 file, the hydrogen atoms can be discarded in the LIGAND_CONFORMERS.pdb file. In the LIGAND_CONFORMERS.pdb file, each ligand conformer is shown in the field between the PDB keywords ‘MODEL’ and ‘ENDMDL’, e.g.:

```
MODEL 1
[The atomic coordinates of model1 here]
ENDMDL
MODEL 2
[The atomic coordinates of model2 here]
ENDMDL
….
MODEL 10
[The atomic coordinates of model10 here]
ENDMDL
```

For protein-ligand design and enzyme design, the selected ligand conformations will be output into a separate mol2 file. The following files will also be generated:

ProteinID_bestlig0001.mol2 **… …** **ProteinID_bestlig0010.mol2**	The corresponding ligand pose mol2 files for 10 designs.

Design enzyme

```
./UniDesign --command=ProteinDesign --enzyme --pdb=enzyme.pdb --mol2=ligand.mol2 --
read_lig_ensemble=LIGAND_CONFORMERS.pdb    --resfile=RESFILE.txt    --
cata_cons=CATALYTIC_CONSTRAINTS.txt    --ntraj=10
```

The ligand transition state conformers in the LIGAND_CONFORMERS.pdb file can be generated by UniDesign using an internal ligand placement approach implemented in UniDesign. It should be noted that the ligand conformers for enzyme design are usually transition state (TS) or near-attacking-conformation (NAC) conformers, which is different from the ligand conformers (substrate/Michaelis-complex-state conformers). Different from a regular protein-ligand design, enzyme design is more complicated because the chemical catalysis should be modeled appropriately. To model the putative catalysis for an enzyme, we use the so-called catalytic constraints to restrain the geometrical relationship between ligand and enzyme catalytic residues; these geometrical constraints are saved in the CATALYTIC_CONSTRAINTS.txt. More introduction will be made to enzyme design in future release. The specific format of catalytic constraints will be explained in detail later.

In some cases, you may want to redesign the active-site binding residues to improve the binding affinity between the enzyme and ligand. For this task, you can use the protein-ligand design command as described above and the ensemble of ligand conformers can be generated by a docking program.

Design protein side-chain packing

```
./UniDesign --command=ProteinDesign --pdb=proteinID.pdb --wildtype_only
```

UniDesign utilizes a stochastic simulated annealing algorithm to do protein design and protein side-chain packing, therefore distinct models may be generated for independent runs. To get a deterministic model, you can run FASPR[9] (https://zhanglab.ccmb.med.umich.edu/FASPR/) which is specifically designed for this task. FASPR is the most effective and efficient packer developed so far. But if you need multiple structural models for your protein (such as intrinsically disordered protein, IDP), UniDesign is a suitable choice.

Repair incomplete protein side chain

```
./UniDesign --command=RepairStructure --pdb=proteinID.pdb
```

UniDesign will only repair (and optimize the conformation of) the amino-acid side chains with missing non-hydrogen atoms without optimizing the other amino acids.

If successful, a new PDB file named proteinID_Repair.pdb will be output in the directory where you run the job.

Protein structure energy minimization

```
./UniDesign --command=Minimize --pdb=proteinID.pdb
```

UniDesign will do a fixed-backbone minimization to reduce/eliminate steric clashes. If your structure in proteinID.pdb has incomplete residues and also has many clashes, you can first run 'RepairStructure' and then run 'Minimize' to optimize the structure.

If successful, a new file named proteinID_Minimize.pdb will be generated in the directory where you run the job.

Build mutant structure model

```
./UniDesign --command=BuildMutant --pdb=proteinID.pdb --mutant_file=individual_list.txt
```

The mutants are listed in the individual_list.txt file in the format of 'WT residue', 'Chain ID', 'residue position in chain', 'mutant residue' (for more explanation, please see https://zhanglab.ccmb.med.umich.edu/SSIPe/help.html#mutantformat):

```
QA22D;
HA18F,QA22D;
HA18F,MB20A;
```

The name of individual_list.txt file can be changed to any other legal name, such as "mutant_file.txt".

One mutation set per line and each line ends in semicolon.

If one mutation set has more than one mutation, separate them by commas.

In the above example, the first line represents a single mutation on Chain A where Glu22 is mutated to Asp. The second line represents a double mutation on the same chain: His18 on chain A is mutated to Phe and Glu22 on chain A is mutated to Asp. The third line represents a double mutation on different chains: His18 on chain A is mutated to Phe and Met20 on chain B is mutated to Ala.

If successful, files named in proteinID_Model_0001.pdb, proteinID_Model_0002.pdb, and proteinID_Model_0003.pdb will be generated. Each file corresponds to a mutant in the order listed in the mutant file.

Add polar hydrogen atom

```
./UniDesign --command=AddPolarHydrogen --pdb=proteinID.pdb --show_hydrogen
```

The option '--show_hydrogen' should be added for writing hydrogen atoms in the file. If successful, a new file name proteinID_PolH.pdb will be generated in the directory where you run the job.

Optimize hydrogen's position to maximize hydrogen bonding network

```
./UniDesign --command=OptimizeHydrogen --pdb=proteinID.pdb --show_hydrogen
```

The option '--show_hydrogen' should be added for writing hydrogen atoms in the file. If successful, a new file name proteinID_OptH.pdb will be generated in the directory where you run the job.

Calculate protein fold stability score

```
./UniDesign --command=ComputeStability --pdb=proteinID.pdb
```

If successful, the weighted total stability score and energy terms will be shown on the screen.

Note that the stability score value of a single protein is meaningless. So you should not directly compare two proteins' stability with this score. Instead, it is meaningful to calculate the $ddG_{stability}$ caused by mutations on a protein. For example, you can use the BuildMutant command on a wildtype protein to construct the structural model of a mutant. And then run the ComputeStability command for the wildtype and mutant protein separately to get two stability scores. Their difference can be regarded as $ddG_{stability}$. The more negative of $ddG_{stability}$, the higher the stability of a mutant is.

Calculate protein-protein binding energy score

```
./UniDesign --command=ComputeBinding --pdb=AB.pdb
```

If your protein has more than two chains, you need to do a chain splitting to calculate the binding energy between the two parts, e.g.:

```
./UniDesign --command=ComputeBinding --pdb=ABC.pdb --split_chains=AB,C
```

UniDesign will calculate the binding interaction energy between partners 'AB' and 'C'.

If you run UniDesign without splitting chains, UniDesign will calculate the interaction energy between 'A' and 'B', 'A' and 'C', and 'B' and 'C' separately. If successful, the weighted total binding score and energy terms will be shown on the screen.

Different from the stability score, you can directly compare two complexes' binding affinity by comparing the two binding scores[10]. It is also useful to calculate the $ddG_{binding}$ caused by mutations on a complex. For example, you can use the BuildMutant command on a wildtype protein to construct the structural model of a mutant. And then run the ComputeBinding command for the wildtype and mutant protein separately to get two binding scores. Their difference can be regarded as ddG_{bind}. The more negative of $ddG_{binding}$, the higher the stability of a mutant is. Sometimes, the mutation may take place at the protein-protein interface, and for these mutations, a specific approach named SSIPe[11] (https://zhanglab.ccmb.med.umich.edu/SSIPe/) can provide more accurate $ddG_{binding}$ prediction than UniDesign.

Resfile

Sometimes, you may not want to completely redesign a protein, and instead you want to fix the identity of a few residues in the scaffold while redesigning the other non-fixed residues. For example, suppose you are designing a protein-protein interface for a large protein dimer complex A/B. Probably you would like to redesign only the interface residues of A to achieve a stronger affinity for binding the partner B. In this situation, you can use a resfile to indicate the residues you want to design or fixed. Here, resfile stands for <u>res</u>traint <u>file</u> for specifying design sites.

In UniDesign, the residues for a designer protein can be divided into four groups: designable, repacked, fixed, and catalytic. The former three groups apply to any kind of protein design task, whereas the 'catalytic' group is only applicable to enzyme design.

As indicated by the name 'designable', the residues in this group can be substituted into other amino acid types; both the rotamer type and conformation can be changed in the design. The 'repacked' residues' conformation can be changed, but the rotamer type is constant. The 'fixed' residues have both their types and conformations fixed. The 'catalytic' group has a similar property to the 'repacked', but indicates which residues are subjected to catalytic constraints during enzyme design simulation.

The resfile has the following format:

```
SITES_DESIGN_START
A   782    ACDEFGHIKLMNPQRSTVWY
A   785    ACDEFGHIKLMNPQRSTVWY
… …
A   991    ACDEFGHIKLMNPQRSTVWY
A   993    ACDEFGHIKLMNPQRSTVWY
SITES_DESIGN_END
SITES_REPACK_START
A    37
A   783
```

```
… …
A 1022
A 1024
SITES_REPACK_END
SITES_FIX_START
A    806
A    819
… …
A    980
A    981
SITES_FIX_END
```

The designable residues are designated in the field between ‘SITES_DESIGN_START’ and ‘SITES_DESIGN_END’. Each row specify one position. For instance, the row ‘A 782 ACDEFGHIKLMNPQRSTVWY’ means that position 782 on the chain A is a designable site, which can be mutated into any of the 20 canonical residues. You can adjust the string ‘ACDEFGHIKLMNPQRSTVWY’ according to your own design requirements. Say, if you know this site must be a hydrophobic residue, you can set the types to ‘ACFGILMPVW’ (In some literature, tyrosine ‘Y’ is also regarded as partially hydrophobic). In each row, the rotamer-type string is adjustable and can be different. If you have some knowledge on the chemical property of some residues, this may help you better define your design scheme.

The repacked residues are specified in the field between ‘SITES_REPACK_START’ and ‘SITES_REPACK_END’. The fixed residues are specified in the field between ‘SITES_FIX_START’ and ‘SITES_FIX_END’. Similarly, the catalytic residues are listed in the field between ‘SITES_CATALYTIC_START’ and ‘SITES_CATALYTIC_END’. In fact, the fixed residues can be ignored because all the residues are initialized as ‘fixed’ when using a resfile to designate design sites.

There are some scenarios for defining which group a residue should be classified into. For example, for a de novo sequence design of a protein, all the residues are designable. For a protein-ligand interaction redesign, the residues in the binding pocket that are involved in direct contact with the ligand (the first shell) can be chosen as designable residues, and residues in the second shell (in close-contact with the first-shell residues but not in contact with the ligand) can be considered as repacked residues, while the others are regarded as fixed residues. For enzyme design to enhance binding and catalytic activity, the first-shell residues that are involved in chemical catalysis are regarded as catalytic, the first-shell non-catalytic residues are designable, the second-shell residues are repacked, and the others are fixed. For a protein-protein interaction redesign to increase binding, the interface positions are designable, the second-shell residues are repacked, and the others are fixed. For very large proteins, dividing residue positions into different groups appropriately can be very helpful to your design.

Disclaimer and Copyright

Citation

The UniDesign paper is in preparation and has not been published formally in a peer-reviewed journal. If UniDesign is useful to your work, please cite the following papers:

- Xiaoqiang Huang, Robin Pearce, Yang Zhang. EvoEF2: accurate and fast energy function for computational protein design. Bioinformatics (2020), 36: 1135-1142. [EvoEF2]
- Robin Pearce, Xiaoqiang Huang, Dani Setiawan, Yang Zhang. EvoDesign: Designing protein-protein binding interactions using evolutionary interface profiles in conjunction with an optimized physical energy function. Journal of Molecular Biology (2019), 431: 2467-2476. [EvoEF1 and EvoDesign]

History

- UniDesign, an extension from EvoEF2 and EvoDesign, for unified protein design.
  Version 1.0 released.
- EvoEF2 for de novo protein sequence design (2020), handling only proteins.
- EvoDesign for PPI design (2019), EvoEF1 developed from scratch for physical scoring.
- EvoDesign for monomer design (2013).

Reporting bugs, comments, and suggestions

If you find bugs or have comments and suggestions for improving UniDesign, please contact Dr. Xiaoqiang Huang (tommyhuangthu@foxmail.com or xiaoqiah@umich.edu). We will try our best to improve the program.

References

(1) Mitra, P.; Shultis, D.; Zhang, Y., EvoDesign: de novo protein design based on structural and evolutionary profiles. *Nucleic Acids Res.* **2013**, *41*, W273-280.

(2) Pearce, R.; Huang, X.; Setiawan, D.; Zhang, Y., EvoDesign: Designing protein-protein binding interactions using evolutionary interface profiles in conjunction with an optimized physical energy function. *J. Mol. Biol.* **2019**, *431*, 2467-2476.

(3) Huang, X.; Pearce, R.; Zhang, Y., EvoEF2: accurate and fast energy function for computational protein design. *Bioinformatics* **2020**, *36*, 1135-1142.

(4) Krivov, G. G.; Shapovalov, M. V.; Dunbrack, R. L., Jr., Improved prediction of protein side-chain conformations with SCWRL4. *Proteins* **2009**, *77*, 778-795.

(5) Schymkowitz, J.; Borg, J.; Stricher, F.; Nys, R.; Rousseau, F.; Serrano, L., The FoldX web server: an online force field. *Nucleic Acids Res.* **2005**, *33*, W382-388.

(6) Miao, Z.; Cao, Y.; Jiang, T., RASP: rapid modeling of protein side chain conformations. *Bioinformatics* **2011**, *27*, 3117-3122.

(7) Guerois, R.; Nielsen, J. E.; Serrano, L., Predicting Changes in the Stability of Proteins and Protein Complexes: A Study of More Than 1000 Mutations. *J. Mol. Biol.* **2002**, *320*, 369-387.

(8) Leaver-Fay, A.; O'Meara, M. J.; Tyka, M.; Jacak, R.; Song, Y.; Kellogg, E. H.; Thompson, J.; Davis, I. W.; Pache, R. A.; Lyskov, S.; Gray, J. J.; Kortemme, T.; Richardson, J. S.; Havranek, J. J.; Snoeyink, J.; Baker, D.; Kuhlman, B., Scientific benchmarks for guiding macromolecular energy function improvement. *Methods Enzymol.* **2013**, *523*, 109-143.

(9) Huang, X.; Pearce, R.; Zhang, Y., FASPR: an open-source tool for fast and accurate protein side-chain packing. *Bioinformatics* **2020**, *36*, 3758-3765.

(10) Huang, X.; Zhang, C.; Pearce, R.; Omenn, G. S.; Zhang, Y., Identifying the Zoonotic Origin of SARS-CoV-2 by Modeling the Binding Affinity between the Spike Receptor-Binding Domain and Host ACE2. *J. Proteome Res.* **2020**, 10.1021/acs.jproteome.0c00717.

(11) Huang, X.; Zheng, W.; Pearce, R.; Zhang, Y., SSIPe: accurately estimating protein-protein binding affinity change upon mutations using evolutionary profiles in combination with an optimized physical energy function. *Bioinformatics* **2020**, *36*, 2429-2437.