Peptide docking tutorial

Nawsad Alam, Ora Schueler-Furman
The Hebrew University of Jerusalem, Israel

EMBO Budapest

Welcome to the peptide-docking tutorial! This exercise will introduce you to our **peptide docking protocol Rosetta FlexPepDock** [1, 2] and its different flavors, and teach you how to generate accurate structures of peptide-protein complexes, starting from an approximate starting conformation. Time permitting, we will also show you how to generate such starting conformations when the peptide binding site is not known, using our **ClusPro/PIPER** FFT-based mapping protocols (developed in collaboration with Dima Kozakov): **PeptiMap** [3] uses solvent mapping to identify putative peptide binding sites on a protein surface, and **PeptiDock** (*unpublished)* uses the mapping of peptide fragments to *ab initio* dock motif-containing peptides.

Many signaling and regulatory processes involve peptide-mediated protein interactions, *i.e.* the binding of a short stretch in one protein to a domain in its partner. Computational tools that generate accurate models of peptide-receptor structures and binding improve characterization and manipulation of known interactions, help to discover yet unknown peptide-protein interactions and networks, and bring into reach the design of peptide-based drugs for targeting specific systems of medical interest.

Peptide docking differs from protein-protein docking mainly by the fact that the peptide usually changes its conformation considerably upon binding to the receptor. Consequently,

peptide docking protocols need to include conformational sampling of internal peptide degrees of freedom (phi, psi, chi), in addition to the sampling of the rigid body orientation. **Rosetta FlexPepDock** was built to generate precise models of protein-peptide complex structures, by effectively addressing the challenge of the considerable conformational flexibility of the peptide. **Rosetta FlexPepBind** is an extension of this protocol that allows characterizing peptide binding affinities and specificities of various biological systems, based on the structural models generated by Rosetta FlexPepDock [4, 5]. Here, we provide detailed descriptions and guidelines for the usage of these protocols and highlight the variety of different challenges that can be met and the questions that can be answered with Rosetta FlexPepDock.

Time (and interest) permitting, we will extend our exercise to include approaches for the full ab initio docking of peptides: Not always is the peptide binding site known, and not always is it possible to generate an approximate structure based on a homolog complex. In these cases global docking protocols are necessary to generate such a starting structure. Here we will present the protocols that we have optimized for this task using **ClusPro/PIPER** FFT-based mapping (in collaboration with Dima Kozakov): **PeptiMap** uses solvent mapping to identify putative peptide binding sites on a protein surface, and **PeptiDock** performs rigid *ab initio* docking of a set of motif-containing peptide fragments.

The tutorial consists of the following parts:

(1) **Local refinement of an approximate model of a peptide-protein complex, using the FlexPepDock server**: This part will provide you a quick introduction of peptide docking. We will look at the results of the prediction and learn to assess the quality and features of the modeled structures.

In many cases, an approximate model of an interaction is available from a homolog complex (e.g. peptide A bound to a peptide binding domain solved bound to peptide B), and local refinement will sample and identify an accurate conformation.

(2) **Local refinement, using command lines in LINUX**: This part will teach you how to run the prediction yourself, introduce you to the Rosetta Modeling Suite, and explain the different parameters that can be tuned to customize your simulation.

(3) ***Ab initio* peptide docking**: This part will describe how to model the structure of a peptide – protein complex when no information about the peptide conformation is known (but the binding site is given).

Here we will introduce more extended sampling of peptide degrees of freedom, using the Rosetta fragment sampling strategy. We will describe fragments and fragment sampling, and the parameters that are important for *ab initio* docking.

(4) **FlexPepBind - prediction of binding partners for a given peptide-binding domain:** using a template structure of a solved peptide-protein complex and a small set of known substrates/non-substrates, we will attempt to distinguish these two groups based on the structures generated by FlexPepDock simulations and their associated binding energies. We will learn the basics of this approach, and in particular experience the use of constraints to restrict the considered peptide conformations to conform to specified features. This, to provide a more precise description system and to prevent false positives.

Time permitting

(5) **Approaches for full blind docking – PeptiMap server: prediction of peptide binding sites.** Reliable identification of peptide binding sites can help focus sampling to one, or a few, sites on the receptor surface. The PeptiMap approach is based on experimental evidence that small organic molecules of various shapes and polarity tend to accumulate at sites that overlap with peptide and protein binding pockets. Using FFT based docking of solvent molecules and filtering according to characteristics of peptide binding sites, PeptiMap is able to predict peptide binding sites with high accuracy.

(6) **Approaches for full blind docking - PeptiDock server: prediction of approximate peptide-protein complexes by fast rigid docking of motif-containing peptide fragments extracted from the PDB.** We have recently developed a global peptide

docking protocol that uses motif information of the peptide sequence to extract a pool of fragments similar to the motif sequence and dock these fragments using the **PIPER** FFT-based docking protocol [6].

In summary, we introduce in this tutorial a wide range of tools which can be used to model and manipulate known, as well as unknown, novel peptide-protein interactions. On the way, you will learn how to run Rosetta simulations in the LINUX environment, and get to know different servers and what they can provide for your research.

Instructions and files needed for the different parts are provided in separate directories in the common **`EMBO_FlexPepDock_tutorials/`** directory.

Let's start:

➢ Determination of adequate amount of sampling needed

Before we start using the various FlexPepDock protocols we should know how to choose among various protocols.

- ❖ Can I use the **FlexPepDock refinement** protocol, which will perform small perturbations of the peptide backbone dihedrals with on the fly side-chain rotamer sampling, or
- ❖ will **simple minimization** do, or
- ❖ do I need **FlexPepDock *ab intio* docking**, which will sample the peptide backbone conformation very extensively using fragments derived from solved crystal structures?

The amount of sampling depends on complexity of the problem. The optimal parameters of the different FlexPepDock protocols depend on the amount of available information about the interaction studied.

- ❖ **FlexPepDock refinement:** If the starting structure is obtained using homology modeling of a similar peptide sequence bound to the receptor or a homolog of the

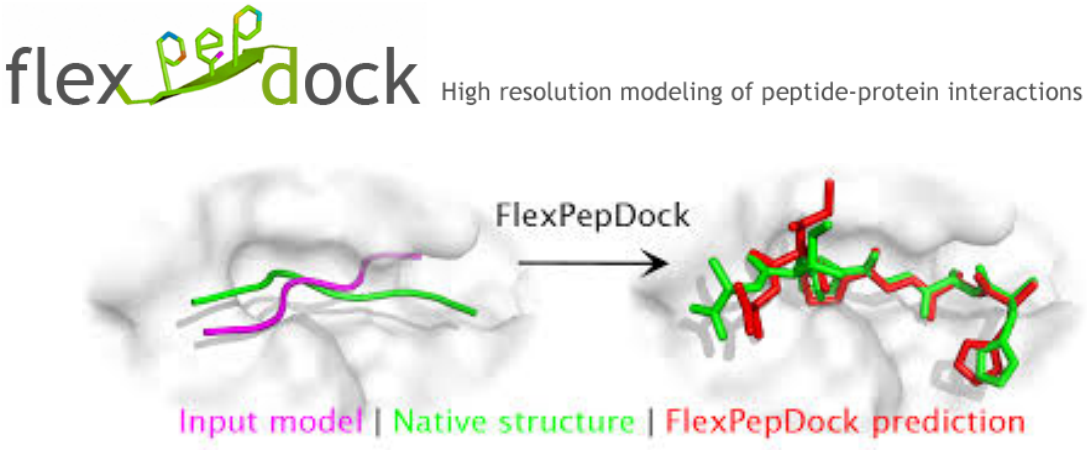
receptor, then it is likely that the starting conformation is close to the native conformation. In such cases one can use the refinement protocol to optimize the starting conformation to a near-native structure. As an example, various peptides binding to the PDZ domains show similar interaction features at the conserved binding site; also, peptide substrates binding to the active site of various enzymes show similar conformations. In such cases the refinement protocol can be used to generate a relatively small number of models (e.g. 200-1000), and even better, when constraints can be defined that characterize peptide interactions, such as the interactions critical for binding of a peptide at the active site of an enzyme; less models need to be generated to ensure the sampling of near-native conformations.

- **Simple minimization:** We have found that for enzymatic systems, in many cases a simple minimization helps to model peptide efficiently at the active site.
- **Ab initio FlexPepDock:** If only the binding site is known, but there is no previous knowledge about the peptide backbone conformation, then it is advised to use the FlexPepDock ab intio protocol to generate a considerably larger number of models (e.g. 50,000) to sample a larger space, and to guarantee convergence of the protocol. This helps to reliably identify more distant low-energy conformations in the energy landscape.

It is important to note that the refinement protocol can refine models which are close to the correct solution both in terms of Cartesian, as well as dihedral ϕ/ψ distance. Even if the peptide is placed in the correct rigid body orientation, but needs to undergo significant dihedral changes (e.g., transition from extended conformation to helix), it is advised to use the ab-initio protocol which will use fragments to sample large dihedral changes more efficiently.

1. Running FlexPepDock refinement using the FlexPepDock refinement webserver

The FlexPepDock refinement server allows refinement of coarse peptide-protein models to near-native accuracy. In this setting the input model should be close to the right solution both in the Cartesian and phi-psi space (effective range: ≤ 5.5Å peptide backbone-RMSD, ≤ 50° phi-psi angle RMSD).



In this exercise we will first go through the server webpage and learn how to submit jobs. We will then inspect the results of a demo run provided on the webpage. Then we will submit a refinement job for the complex of the peptide HAGPIA (from the HIV capsid protein) bound to cyclophilin A. The structure of the peptide-protein complex has been solved using X-ray crystallography (protein data bank [PDB] id 1AWR). The input structure to the server will be the bound receptor with the peptide bound in extended conformation in the binding site. The goal is to refine the peptide to near-native conformation.

- Location of files for this exercise:
  `1_Refinement_of_protein_peptide_complex_using_FlexPepDock_Server`
- Required input (located in **`input_files/`**):
  - **1AWR_ex.pdb** : The starting structure with the peptide in extended conformation
  - **1AWR.pdb** : The native structure (for comparison)

➢ Visit ***http://flexpepdock.furmanlab.cs.huji.ac.il*** and submit the refinement job (*Hands-on demonstration on the screen)*. See **Figure 1** for detailed instructions.

❖ **Inspect the model using Chimera or PyMOL (your choice)**: load the native structure, the starting structure, and the top-10 models. Are critical features of the interaction recovered?

❖ **Inspect the score vs. rmsd plot**. How does the energy landscape look like? Can you identify an energy funnel around the native conformation? Are we sampling enough, or should we sample more? Locate the model with the best energy on this plot. What RMSD to the native structure does it show? Is this an accurate prediction?

❖ For demonstration purposes, we have used here a starting structure of the bound receptor and an extended peptide. How would a more realistic starting structure look like? Explain.

Figure 1: Overview of FlexPepDock server pages:

(1) Submission of job

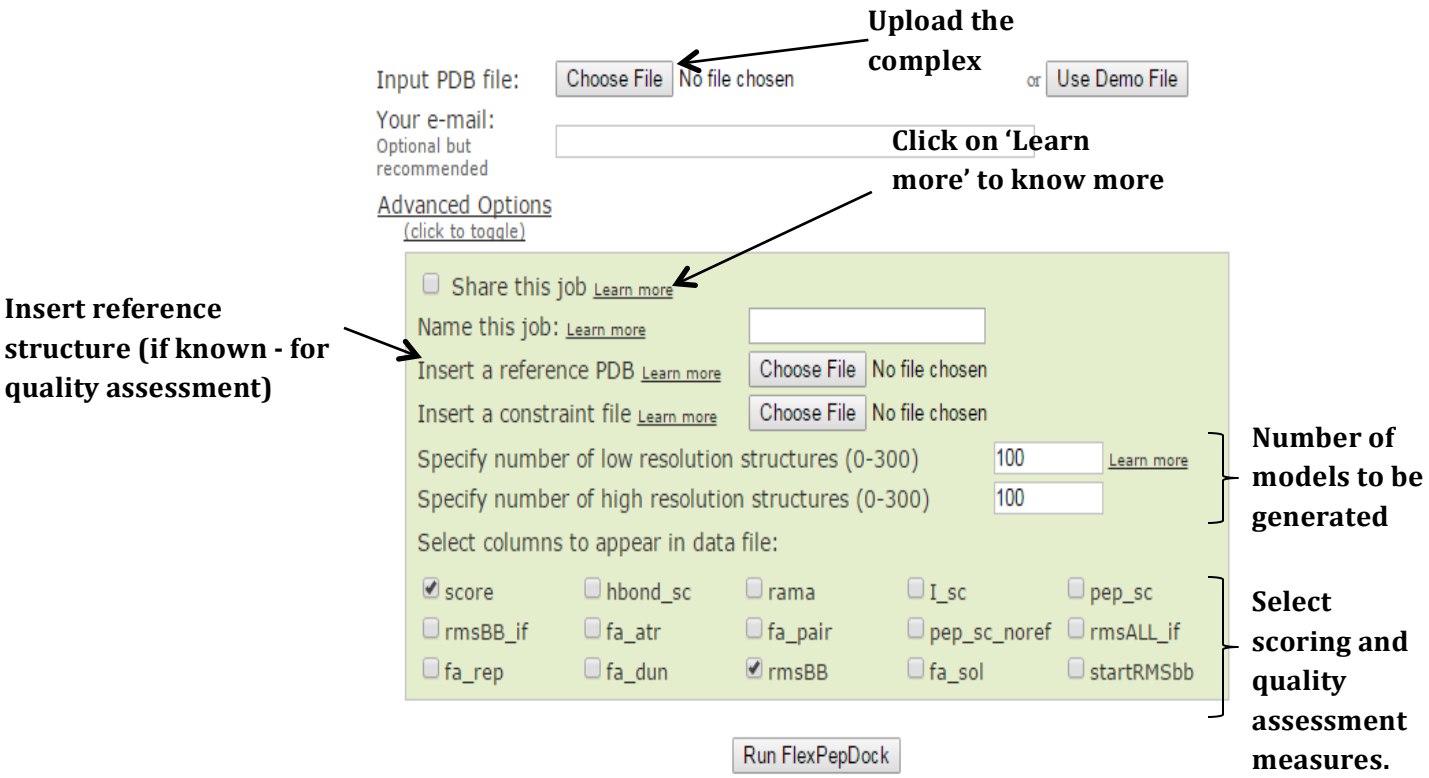
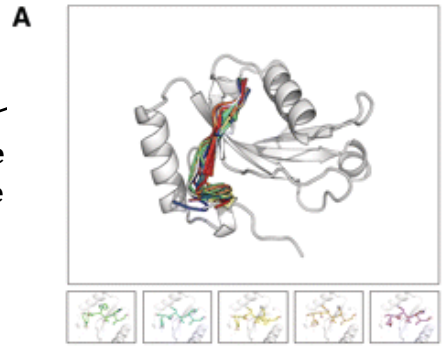


Figure 1, continued:

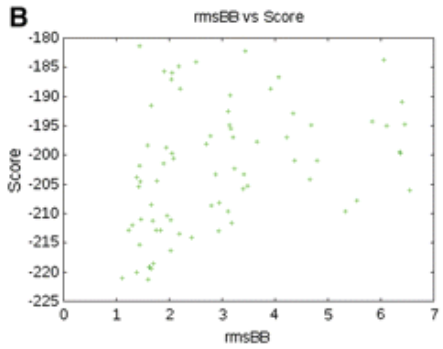
(2) Results:

Provide a constraints file to introduce e.g. distance constraints for biasing simulation to specific region



An overlay of the top 10 models created by FlexPepDock

Detailed structural views for the top 5 models



Plot of the energy landscape sampled by Rosetta FlexPepDock

Energy landscape can provide information about local funnels

Top 10 models (Zip file)

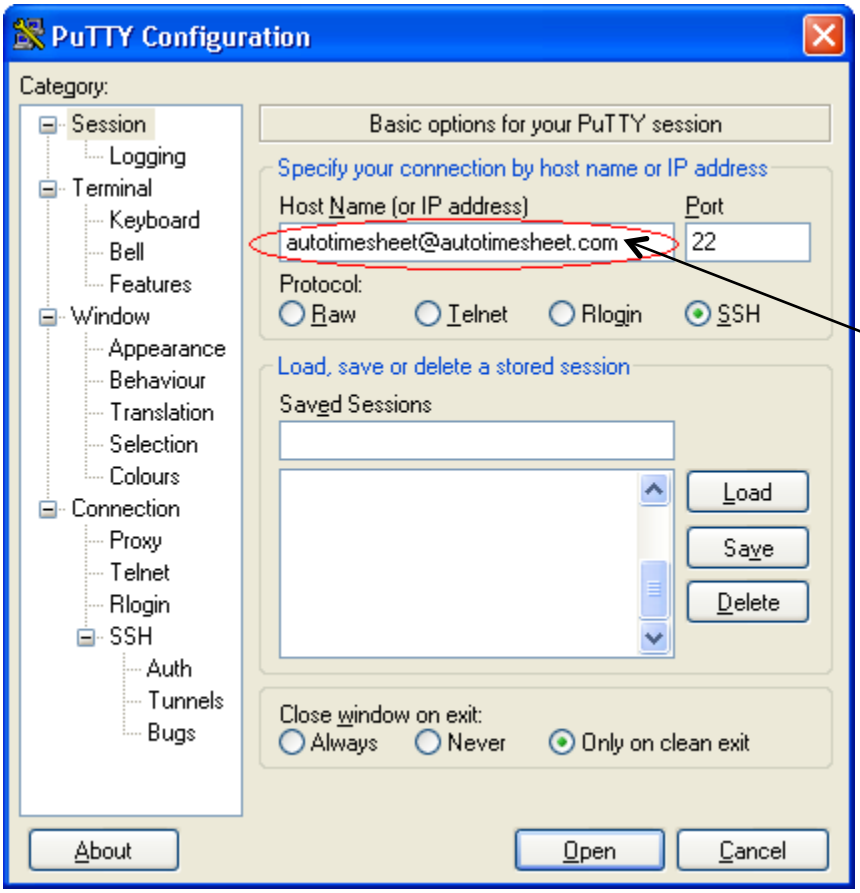
rmsBB and scores for top 10 models (TXT file)

Top 10 models and their corresponding scores

2. Running FlexPepDock refinement using the command line

Now that you have an idea of what a FlexPepDock simulation provides, we will proceed to your first run of a Rosetta simulation, within the LINUX command line environment. The tutorial directory named **`EMBO_FlexPepDock_tutorials`** has been copied to your account.

- Login to your account on the server (either cluster1 or cluster2) using the **Putty SSH** client.



Hosts

user@ cluster1.elte.hu
or
user@ cluster1.elte.hu

Running a Rosetta simulation

Before we begin, we need to know some basic things about Rosetta and how to use it. Rosetta is freely available to academic users using a licensing agreement that you need to sign the first time you use Rosetta.

Go to ***els.comotion.uw.edu/express_license_technologies/rosetta*** and proceed to the academic license link. This is a simple online form that is quickly completed.

In the following, we will only include the most basic instructions for performing Rosetta simulations. For the advanced/interested participants, we have included in **BOXES** additional details about the simulations. **Appendix 1** at the bottom of this document includes a detailed description of the different runline commands used with FlexPepDock

(for ease, the commands used in this tutorial are highlighted in bold). **Appendix 2** contains a brief description of different scoring terms specific to the FlexPepDock application.

NOTE: Throughout this tutorial, the input and output files of a run will be located in the sub-directories **`input`** and **`output`**, respectively (unless noted otherwise). The command lines are shown in bold. The commands follows the symbol '**$'** will represent the terminal prompt.

BOX 1: Basic features of a Rosetta run

1. **Setting up Rosetta**: In this workshop, Rosetta has been installed and is ready to use for you. It is located at ***/opt/rosetta_bin_linux_2015.38.58158_bundle/***. To set up Rosetta on your own machine, download Rosetta and build it. Detailed instructions on downloading Rosetta can be found at ***www.rosettacommons.org/software***. Build instructions can be found at ***www.rosettacommons.org/docs/latest/build_documentation/Build-Documentation.***

2. Any Rosetta run will use

- an executable, **$FlexPepDocking.{ext}** (here **FlexPepDocking.linuxccrelease**), located in the **$ROSETTA_BIN** directory (here **/opt/rosetta_bin_linux_2015.38.58158_bundle** ***/main/source/bin/***)
- a database (containing e.g. parameters for the scoring function, invoked as **-database $ROSETTA_DB** (here as ***/opt/rosetta_bin_linux_2015.38.58158_bundle/main/database/***).

3. The Rosetta command line is composed of two parts:

- The executable of the application
- A list of options for the particular Rosetta simulation.

Options can be listed with the command. Options, and arguments to the options, are separated by whitespace. A single or double colon is using to clarify options when there are multiple separate options with the same name.

$ROSETTA_BIN/FlexPepDocking.linuxgccrelease -database $ROSETTA_DB -s input/1AWR.ex.pdb –pep_refine

Options can also be written in a flag file and invoked to the command file using the '@' symbol. To inspect the content of the flags file:

$cat flags
-database $ROSETTA_DB
-s input/1AWR.ex.pdb
–pep_refine

$ROSETTA_BIN/FlexPepDocking.linuxgccrelease @flags

Depending on the system, you might need to change the parameters to the flags, such as the input file name and others.

4. Most Rosetta runs will provide as output:

- a **scorefile** that contains a one-line information about each model (decoy) generated (usually with file extension **.sc**)
- one or more **output structures** (usually with extension **.pdb**, or in silent format with extension **.silent**)
- a **log-file** that contains information about the performed run.

For more details, see the extensive documentation of the various Rosetta protocols at ***www.rosettacommons.org/docs/latest/Home.*** Queries related Rosetta can be posted to the Rosetta Forum at ***www.rosettacommons.org/forum***

Now that we are ready, let's proceed to our simulation.

Location of files for this exercise:
`2_Refinement_of_protein_peptide_complex_using_FlexPepDock/`
We will use the same input structures as above.

- Go to the tutorial directory

$cd 2_Refinement_of_protein_peptide_complex_using_FlexPepDock/

Running refinement involves several steps. Follow them one-by-one:

1. *Create an initial complex structure:* A coarse model of the peptide-protein interaction can be obtained from a low-resolution peptide-protein docking protocol (see below), or can be built using a homologous structure. As in **Section 1**, we will here use the bound

receptor - extended peptide conformation (**1AWR.ex.pdb**). Our aim is to generate a model similar to the native structure (**1AWR.pdb**). These input pdb files are located inside the ***input/*** directory.

2. *Prepack the input model*: This step involves the packing of the side-chains in each monomer to remove internal clashes that are not related to inter-molecular interactions. The prepacking guarantees a uniform conformational background in non-interface regions, prior to refinement. The ***run_prepack*** script located in the current directory will run prepacking of the input structure **input/1AWR.ex.pdb** (this notation mean the file 1AWR.ex.pdb located inside the **input/** directory), using parameters defined in the ***prepack_flags*** file located in the current directory. **BOX 2** provides details about the prepacking command lines listed inside the ***prepack_flags*** option list file.

- Run the ***run_prepack*** script

$ bash run_prepack

The output will be:

- a prepacked structure, **input/1AWR.ex.ppk.pdb**
- a score file, **output/ppk.score.sc**
- a log file, **output/prepack.log** file

The prepacked structure is usually very similar to the input starting structure (check it in Chimera or PyMOL).

BOX 2: The prepacking step command line

The **run_prepack** file will invoke the following command line:

$ROSETTA_BIN/FlexPepDocking.linuxgccrelease -database $ROSETTA_DB @prepack_flags >output/ppk.log

Where the variables **ROSETTA_BIN** and **ROSETTA_DB** have been defined as the paths corresponding to the locations of the executable directory and database directory, respectively. The **prepack_flags** is the flag file (See **BOX 1**) containing a list of options needed to run prepacking.

$cat prepack_flags

```
-s input/1AWR.ex.pdb          # the input peptide-protein complex structure
-native input/1AWR.pdb        # the ref structure to be used for RMSD calculations
-ex1                          # extra rotamers for chi-1 angles utilized during side-chain packing
-ex2aro                       # extra rotamers for aromatic chi-2 angles utilized during side-chain packing
-use_input_sc                 # adds the starting structure's rotamers to the rotamer library
#-unboundrot unbound.pdb      #add the rotamers of the unbound receptor to the rotamer library
-flexpep_prepack              # flag for running FlexPepDock prepack
-nstruct 1                    # create one pre-packed output structure
-scorefile ppk.score.sc       # name of the scorefile
-flexpep_score_only           # adds FlexPepDock specific scores to the scorefile
-out:path:pdb input           # location of the prepacked structure
-out:path:score output        # location of the scorefile
```

Where **–flexpep_prepack** is the prepacking specific flag that invokes the prepacking protocol. The options **-ex1 –ex2aro –use_input_sc** define the receptor side chain flexibility in this and the following simulation steps (see **Appendix 1** for a description of all the parameters). The flags **–unboundrot** adds the rotamers of the unbound receptor to the rotamer library. It is important to include this flag when you are using unbound receptor**.** The flags files for other FlexPepDock protocols will be very similar (see BOXES below).

Running the **run_prepack** script will generate **1AWR.ex_0001.pdb** which will be renamed to 1AWR.ex.ppk.pdb, and used as input for refinement in the next step. The logs related to the run will be saved to **output/ppk.log**.

3. *Refine the prepacked model*: This is the main part of the simulation. In this step, the peptide backbone and its rigid-body orientation are optimized relative to the receptor protein. This is done with iterative Monte-Carlo with Minimization steps that include periodic on-the-fly side-chain optimization. An optional low-resolution (centroid) pre-optimization will increase the sampling range and may improve performance further. The ***run_refine*** script located in the current directory will run refinement of the prepacked structure generated in the prepacking step located in the input directory,

using flags defined in the file ***refine_flags*** located in the current directory. **BOX 3** provides details about the flags listed inside the ***refine_flags*** flags file.

- Run the refinement protocol:

$ bash run_refine

The output will be:

- a prepacked structure, **output/1AWR.ex.ppk_0001.pdb**
- a score file, **output/refine.score.sc**
- a log file, **output/refine.log**

To save time, and for educational purposes, we have submitted a run that will generate one model only. In real life settings you will need to generate a pool of refined models (200-1000). It is recommended to run such a job either on a cluster or on multiple cores of a local system. Note that for running on a cluster of many nodes, the ***run_refine*** script needs to be modified (see **BOX 3** for more details).

We have performed such a run for you. The input files and the out are located in the **cluster_run**/ directory.

BOX 3: The refinement step command line

The **run_refine** file will invoke the following command line:

$ROSETTA_BIN/FlexPepDocking.linuxgccrelease -database $ROSETTA_DB @refine_flags >output/refine.log

Where **refine_flags** is the flag file (See **BOX 1**) containing the list of options needed to run refinement.

$cat refine_flags

```
-s input/1AWR.ex.ppk.pdb   # the input peptide-protein complex structure
-native input/1AWR.pdb     # the ref structure to be used for RMSD calculations
-ex1                       # extra rotamers for chi-1 angles utilized during side-chain packing
-ex2aro                    # extra rotamers for aromatic chi-2 angles utilized during side-chain packing
-use_input_sc              # adds the starting structure's rotamers to the rotamer library
```

```
#-unboundrot unbound.pdb #add the rotamers of the unbound receptor to the rotamer library
-pep_refine                 # flag for running FlexPepDock prepack
-lowres_preoptimize         # low-resolution optimization before full atom refinement
-nstruct 1                  # generates one refined output structure
-scorefile refine.score.sc  # name of the scorefile
-flexpep_score_only         # adds FlexPepDock specific scores to the scorefile
-out:path:pdb output        # location of the prepacked structure
-out:path:score output      # location of the scorefile
```

Where **–pep_refine** invokes the refinement protocol, and **–lowres_preoptimize** invokes low-resolution sampling before full atom refinement. The remaining parameters are identical to the previous prepack run (see **BOX 2** and **Appendix 1** for a description of all the parameters).

The logs related to the run will be saved to **output/refine.log** file.

In a regular production run, it is advised to run on a cluster of cores. In this case, the following modifications need to be added to the command line:

```
-nstruct 1000                  # generates 1000 refines structures
-out:file:silent decoys.silent # compresses the output structure into the silent file to save space
-out:file:silent_struct_type binary
```

For running the parallel job on the cluster you need to use mpirun command. The actual setup for a parallel run will depend on your cluster. Below is an example of how to run a parallel job on 10 cores.

mpirun -n 10 $ROSETTA_BIN/FlexPepDocking.mpi.linuxgccrelease -database $ROSETTA_DB @refine_flags >output/refine.log

Analysis of the results:

a. **The energy landscape sampled by our simulation:** In order to obtain a quick intuition about our simulation, it is advised to plot score *vs.* rmsd plots to show the energy landscape (as we looked at in the server run above).

- Use the **create_plots.sh** script to generate such plots:

$bash create_plots.sh

This script will extract the parameters relevant to score and RMSD, here **total score** (also written as **score** in older Rosetta versions), **interface score (I_sc)** and **reweighted score (reweighted_sc**; the sum of the total score, the interface score and the peptide score – up weights contributions by the peptide, and at the interface). It will generate plots of score *vs*. peptide backbone interface RMSD (rmsBB_if; calculated after aligning the receptor structure only).

The output plots are **rmsBB_if_vs_score.plot.png**, **rmsBB_if_vs_I_sc.plot.png** and **rmsBB_if_vs_reweighted_sc.plot.png**.

Compare these plots to the plot provided by the server.

- How accurate are our predictions? Extract the top-scoring models and analyze their structural features.

  The **extract_top10_pdbs.sh** script will extract top-scoring decoys according to reweighted score (reweighted_sc) into the **reweighted_sc_top10/** directory. **BOX 4** describes how to extract additional models.

- Run extract_top10_pdbs.sh as:

$bash extract_top10_pdbs.sh refine.score.sc reweighted_sc

BOX 4: Extract specific models for further visual inspection

We have here extracted the top-scoring models according to the scoring term **reweighted score.** This term was found to distinguish best near-native models from the rest. Additional terms may however be used, including interface score (I_sc) and peptide score (pep_sc).

Run the script **extract_top_pdbs.sh** with an additional parameter to obtain these, e.g.:

$bash extract_top10_pdbs.sh refine.score.sc I_sc

Here the top ten refined structures will be extracted and put into the **I_sc_top10/** directory.

You can also extract additional selected models from the decoys.silent file using the following command:

$ROSETTA_BIN/$extract_pdbs.{ext} -database $ROSETTA_DB -in:file:silent decoys.silent -in:file:fullatom -in:file:tags decoy_tag

Where the Rosetta executable **extract_pdbs** is used, and **decoy_tag** is the tag(s) of the desired decoy(s) to be extracted, e.g.:

-in:file:tags 1AWR_ex.ppk_0001 1AWR_ex.ppk_0002 1AWR_ex.ppk_0003

- Inspect the top-scoring models: how accurate are they? Compare the results to the server-results discussed in the previous section.

3. Running FlexPepDock *ab initio* using the command line.

When the peptide conformation is not known, we need to significantly increase the sampling of the peptide conformation space. This is done using fragments, similar to Rosetta ab initio protein folding. Here we will demonstrate how a native helical conformation is identified, starting from an extended peptide, on the example of the mineral corticoid receptor bound to a peptide from NRCOA-1 (nuclear receptor coactivator 1; PDB id 2A3I).

The files related to this exercise are located in **`EMBO_FlexPepDock_tutorials/3_Abinitio_fold_and_dock_of_peptides_using_flexpepdock/`**

Follow the following step to run FlexPepDock *ab initio* locally:

1. *Create an initial complex structure*: An initial model can be built by placing the peptide in close proximity to the binding site in an arbitrary conformation. In this demo, we have provided a starting structure with a peptide in extended conformation (**2A3I.ex.pdb**). Our goal is to optimize this structure using *ab-initio* FlexPepDock, towards a near-native model

with a helical peptide conformation. Both the native structures (**2A3I.pdb**), as well as the starting structure (**2A3I.ex.pdb**) are provided in the **input/** directory.

2. *Prepack the input model*: Run prepacking as in the refinement tutorial above.

$ bash run_prepack

This will prepack the input structure **2A3I.ex.pdb** located in the input directory. The output will be:

- a prepacked structure, **input/2A3I.ex.ppk.pdb**
- a score file, **output/ppk.score.sc**
- a log file, **output/prepack.log**

3. *Create fragment libraries* **(**3mer, 5mer & 9mer (peptide length >=9): The scripts necessary for creating fragments are provided in the **`fragment_picking`** directory. We will provide the fragment files for this run in the input directory. See **BOX 5** for details for running this on your own.

BOX 5: Generating fragment files

Follow the steps below to generate the fragments:

A. Go to the **`fragment_picking`** directory.

B. Save the peptide sequence in the **xxxxx.fasta** file.

C. Run the **make_fragments.pl** script to generate the PSIPred secondary structure and PSI-Blast sequence profiles. You need to change the paths in the upper section of the **make_fragments.pl** file (here we have done it for you).

- Run as

$perl make_fragments.pl -verbose -id xxxxx xxxxx.fasta

This will create **xxxxx.psipred_ss2**, **xxxxx.checkpoint,** along with other files.

D. Run the executable **fragment_picker.linuxgccrelease** to create the fragments. The flags are provided in the **flags** file and the fragment scoring weights are provided in the **psi_L1.cfg** file.

- Run as

$ROSETTA_BIN/fragment_picker.linuxgccrelease -database $ROSETTA_DB @flags >log

E. Change the fragment numbering using the **shift.sh** script, and move the fragments to the input directory for the ab initio run:

- Run as

$bash shift.sh frags.500.3mer X >frags.3mers.offset

where X is the number of residues in the receptor. Repeat this for 5mer and 9mer fragments.

- Move the offset fragment files to the **input/frags** directory.

4. *Ab-initio folding and docking of the prepacked model*: This is the main part of the ***ab initio*** **FlexPepDock** protocol. In this step, the peptide backbone and its rigid-body orientation are optimized relative to the receptor protein using the Monte-Carlo with Minimization approach, including periodic on-the-fly side-chain optimization. The peptide backbone conformational space is extensively sampled using fragments derived from solved structures. The file **abinitio_flags** contains flags for running the ab-initio job. The **run_abinitio** script will run ab-initio modeling of the prepacked structure generated in the prepacking step located in the input directory. **BOX 6** provides details about the flags listed inside the ***abinitio_flags*** flags file.

- Run the run_abinitio script as:

$ bash run_abinitio

The output will be:

- a prepacked structure, **output/2A3I.ex.ppk_0001.pdb**
- a score file, **output/abinitio.score.sc**
- a log file, **output/abinitio.log**

This script has to be modified to run on a cluster during a production run (see below).

BOX 6: The *ab initio* step command line

The **run_abinitio** file will invoke the following command line:

$ROSETTA_BIN/FlexPepDocking.linuxgccrelease -database $ROSETTA_DB @abinitio_flags >output/abinitio.log

Where **abinitio_flags** is the flag file (See **BOX 1**) containing list of options needed to run *ab initio* docking. The logs related to the run will be saved to **output/abinitio.log**.

The following flags are identical to definitions in the refinement run (See **BOX 3** above for details):
-s input/2A3I.ex.ppk.pdb -native input/2A3I.pdb –ex1 -ex2aro -use_input_sc -nstruct 1 -scorefile abinitio.score.sc –pep_refine -flexpep_score_only -out:path:pdb output -out:path:score output

The additional *ab initio* protocol specific flags are listed below (See **Appendix 1** for a full list and details of runline parameters)

```
-lowres_abinitio                          # flag for running FlexPepDock prepack
-frag3 input/frags/frags.3mers.offset     # 3mer / 5mer / 9mer fragments files for ab-initio sampling
-flexPepDocking:frag5 input/frags/frags.5mers.offset
-frag9 input/frags/frags.9mers.offset
-flexPepDocking:frag5_weight 0.25         # setting weight for the different frags (3mers weights is 1.0)
-flexPepDocking:frag9_weight 0.1
```

–lowres_abinitio invokes the low resolution peptide fold and dock protocol, and the remaining flags are required to read various fragments files and set the relative weights of these fragments. Note that we invoke the **–pep_refine** flag (as in the refinement run), so that the folded and docked models are further refined using the refinement protocol.

As for refinement, in a regular production run we will use a cluster of cores. In this case, the following modifications need to be added to the command line:

```
-nstruct 50000                  # generates 50000 structures
-out:file:silent decoys.silent  # compresses the output structure into the silent file to save space
-out:file:silent_struct_type binary
```

See **BOX 4** for instructions to extract pdb files from this silent file, and **BOX 3** for how to use MPI for parallel running on a cluster.

In our script, we have generated one model only. In real life examples you will need to generate a large pool of refined decoys (1000-50000). We have provided the results for a cluster run of an *ab initio* job, which generated 5000 decoys. The input and output files are located in the **`cluster_run`**/ directory.

Before we proceed to look at our results, it is recommended to cluster the models to maximize diversity of the final model set, and to estimate the size of the local energy basin.

5. Cluster the top-scoring decoys using the Rosetta clustering application: To maximize the diversity of the final models, and to estimate the size of the local energy basin, we suggest clustering the results, using the Rosetta clustering application. We recommend to cluster the top-1% models (either based on reweighted score, or interface score) and to choose the clusters with lowest-energy representatives. We have found that a clustering radius of 2.0 Å (peptide CA atoms only) provides a diverse and representative set of conformations, among which good solutions are often found within the top 1-10 clusters (Note that in our script, the RMSD values have been modified to reflect RMSD of the whole complex as the clustering is performed on the whole complex and in that context the clustering radius will change).

A clustering script is provided in the **`clustering`** directory. It will cluster the top 500 decoys based on a cutoff radius of 2.0 Angstrom, and select for each the top-scoring

member (according to reweighted score, reweighted_sc). A top scoring member from each cluster is reported in the file cluster**_list_reweighted_sc_sorted**.

For detailed clustering command line options and parameter setting please visit **https://www.rosettacommons.org/docs/latest/application_documentation/utilities/cluster.**

- Go to the clustering directory and run the clustering script.

$cd clustering

$bash cluster.sh 2.0 ../input/2A3I.pdb ../output/decoys.silent

- **Inspect the score vs. rmsd plot**.
  - How does the energy landscape look like? Can you identify an energy funnel around the native conformation? Are we sampling enough, or should we sample more?
  - How much has the range of sampled conformations changes compared to a refinement run (compare the results to the refinement run; pay attention to the x-axis RMSD)
  - Locate the model with the best energy on this plot. What RMSD to the native structure does it show? Is this an accurate prediction?
- **Inspect the top-scoring cluster representatives using Chimera or PyMOL (your choice)**: load the native structure, the starting structure, and the top-10 models. Are critical features of the interaction recovered?
- For demonstration purposes, we have used here a starting structure in extended conformation, positioned in the binding site using one anchor residue. Suggest other ways to generate starting structures. How would the simulation be affected by the differences in the starting conformation?

4. Running FlexPepBind on Histone Deacetylase 8 (HDAC8)

Predicting the structure of a peptide-protein interaction (and an interaction in general) does not guarantee that the peptide indeed binds to the receptor. However, an accurate model of a peptide-protein interaction can be used to evaluate the binding ability of a given peptide sequence to a given receptor, e.g. relative to other peptides. We have found that this works particularly well when the characteristic binding features of interactions are defined and reinforced during modeling: sequences for which low-energy models can be generated that fulfill these constraints are assumed to be binders.

We have developed FlexPepBind - a protocol that uses the FlexPepDock framework to model the structure of a range of different peptide-receptor complexes and identifies binders / non-binders.

This demo illustrates how to run the FlexPepBind protocol to predict peptide binding for two systems: (1) substrates that are deacetylated by Histone Deacetylase 8 (HDAC8)[4], and (2) substrates that are farnesylated (a farnesyl moiety is attached to their c-terminus) by Farnesyl Transferase (FTase)[5].

Protocol overview: The FlexPepBind protocol predicts peptide binding by modeling different peptide sequences onto a template peptide-receptor complex. Below are the different stages of developing FlexPepBind protocol for a specific biological system:

1. Find structure of the receptor of interest in complex with a peptide. Go to the protein data bank webpage **www.rcsb.org/pdb/home/home.do** and search for your protein.
2. Gather a dataset of peptide which know binding affinity toward the receptor; a mixed set of binders or non-binders; or substrates or non-substrates. It is important that critical interactions need to be constrained by using Rosetta constraints. Below we show example constraints invoked during running FlexPepBind on the HDAC8 system.

AtomPair OD2 253 OH 366 HARMONIC 2.8 0.2 #constrain the distance between of atom OD2 of the residue number 253 to atom OH of the residue number 366 to a distance of 2.8 Å and use a harmonic potential to penalize if the distance changes.

3. Use the optimized protocol on unknown set of peptides.

System-specific calibration: Depending on the system, either simple minimization (of the full peptide conformation and rigid body orientation, and the receptor side chains) or extensive optimization using the FlexPepDock refinement protocol might be needed. Also, the optimal measure to rank the different peptides has to be determined (i.e., a score that highlights the interface energy; see below).

In this demo we show how to use the minimization only protocol to predict binders and non-binders in a set of peptide sequences.

The files related to the FlexPepBind tutorial are located in the ***6_peptide_specificity_using_FlexPepBind/*** directory.

- To run on the specific systems provided here, go to the relevant directory, and run the **fpbind_run.sh** script (located in the scripts directory).

$ cd HDAC8/

$ bash ../scripts/fpbind_run.sh

This will minimize, for each peptide in a list (**input_files/peptide.list)**, the peptide-protein complex generated by threading the peptide sequence onto the template.

We will use a constraint file that tethers the acetylated lysine into the binding pocket, as described in **Figure 2** below.

```
AtomPair OD2 253 OH  366 HARMONIC 2.8 0.2
AtomPair OD2 164 OH  366 HARMONIC 3.7 0.2
AtomPair NE2 129 NZ  366 HARMONIC 4.8 0.2
AtomPair ND1 166 OH  366 HARMONIC 3.3 0.2
AtomPair OD2  87 N   366 HARMONIC 3.2 0.2
AtomPair OD1  87 N   366 HARMONIC 3.0 0.2
AtomPair CE1 194 CG  366 HARMONIC 4.0 0.2
AtomPair CE1 138 CD  366 HARMONIC 3.4 0.2
AtomPair OD2 162 ND1 128 HARMONIC 2.5 0.2
AtomPair OD2 169 ND1 129 HARMONIC 2.8 0.2
AtomPair O   137 NZ  366 HARMONIC 3.0 0.2
AtomPair CZ  292 OH  366 HARMONIC 3.7 0.2
AtomPair OD2 164 OH  366 HARMONIC 3.7 0.2
Dihedral N   366 CA 366 C 366 N 367 CIRCULARHARMONIC -46.9 5.0
```

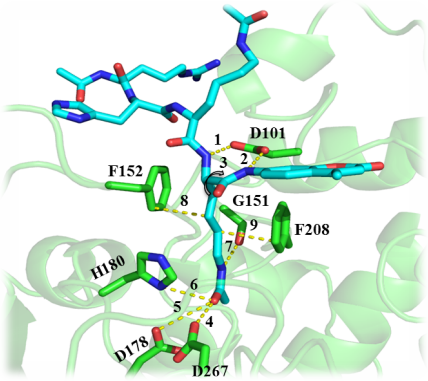


Figure 2: Constraints imposed in optimization run

- Analyze the data: run **fpbind_analysis.sh** (located in the scripts directory). It will extract the relevant scores of the minimized structures & save them in the **score_analysis/** directory.

$ bash ../scripts/fpbind_analysis.sh

Scores currently considered are:

1. **Interface score** - the energy contributed at the interface (**I_sc**)
2. **peptide score** - the energy contributed by the internal energy of the peptide + the interface score (**pep_sc**)
3. **peptide score without reference energy** - the same peptide score, but without the amino acid dependent energy terms (Eaa) that were optimized to generate designs with natural amino acid content (**pep_sc_noref**; we found that removing this term can significantly improve distinction of substrates from non-substrates in certain systems)
4. **Reweighted score**: = sum (total_score + peptide_score + interface score) (**reweighted_sc**)

The output contains a list of scores, e.g. **score_analysis/I_sc**

$ paste input_files/peptide.list score_analysis/I_sc

```
GYKFGC -16.7
GFKWGC -17.5
```

Substrates
Low scores

```
GFKFGC -16.4
GMKDGC -13.9 ┐
GDKDGC -13.2 ├ Non-substrates
GQKIGC -13.4 ┘ High scores
```

The above lines show the interface scores for 6 peptides (the top 3 are HDAC8 substrates & the bottom 3 are not HDAC8 substrates).

- **Inspect the scores of the substrates and the non-substrates;** How do the energies look like? Can you discriminate between the substrates and the non substrates based on score?
- Open the best-scoring and the worst-scoring models in Chimera (located in **XXXXXX/XXXXXXstart.ppk_0001.pdb**, where XXXXXX is the peptide sequence, upper case letters). Can you explain the difference between substrate and non-substrate?
- Choose a different system and discuss how you would implement FlexPepBind for that system.

Appendix 1: Runline options for FlexPepDock simulations

Parameters used in this tutorial are highlighted in bold.

(A) Common FlexPepDock flags

Flag	Description	Default
`-receptor_chain`	Chain id of receptor protein	first chain
`-peptide_chain`	Chain id of peptide protein	second chain
`-lowres_abinitio`	**Low-resolution ab-initio folding and docking mode**	**false**
`-pep_refine`	**Refinement mode**	**false**
`-lowres_preoptimize`	**Perform a preliminary round of centroid mode optimization before Refinement**	**false**
`-flexpep_prepack`	**Prepacking mode. Optimize the side-chains of each monomer separately (no docking)**	**false**
`-flexpep_score_only`	**Read in a complex, score it and output interface statistics**	**false**
`-flexPepDockingMinimizeOnly`	Minimization mode: Perform only a short minimization of the input complex	false
`-ref_startstruct`	Alternative start structure for scoring statistics (useful as reference for rescoring previous runs with the -flexpep_score_only flag.)	N/A
`-peptide_anchor`	Set the peptide anchor residue manually. Only recommended if one strongly suspects the critical region for peptide binding to be remote from its center of mass.	Residue nearest to the peptide center of mass

(B) Relevant Common Rosetta flags

Flag	Description
`-in::file::s` `-in:file:silent`	**Specify starting structure (PDB** or silent format, respectively)
`-in::file::silent_struct_type` `-out::file::silent_struct_type`	Format of silent file to be read in/out. For silent output, use the binary file type since other types may not support ideal form Models can be extracted using `extract_pdbs.`
`-native`	**Specify the native structure for which to compare in RMSD calculations. When the native is not given, the starting structure is used as reference.**
`-nstruct`	**Number of models to create in the simulation**
`-unboundrot`	**Add the position-specific rotamers of the specified structure to the rotamer library (usually used to**

	include rotamers of unbound receptor)
-use_input_sc	**Include rotamer conformations from the input structure during side-chain repacking. Unlike the -unboundrot flag, not all rotamers from the input structure are added each time to the rotamer library, only those conformations accepted at the end of each round are kept and the remaining conformations are lost.**
-ex1/-ex1aro -ex2/**-ex2aro** -ex3 -ex4	Adding extra side-chain rotamers (highly recommended). **The -ex1 and -ex2aro flags are recommended as default values.**
-database	**The Rosetta database**
-frag3 **-flexPepDocking:frag5** **-frag9**	**3mer / 5mer / 9mer fragments files for ab-initio peptide docking (9mer fragments for peptides longer than 9).**

(C) Expert flags

Flag	Description	Default
-rep_ramp_cycles	The number of outer cycles for the protocol. In each cycle, the repulsive energy of Rosetta is gradually ramped up and the attractive energy is ramped down, before inner-cycles of Monte-Carlo with Minimization (MCM) are applied.	10
-mcm_cycles	Number of inner-cycles for both rigid-body and torsion-angle Monte-Carlo with Minimization (MCM) procedures	8
-smove_angle_range	Defines the perturbation size of small/sheer moves	6.0
-extend_peptide	Start the protocol with the peptide in extended conformation (neglect original peptide conformation; extend from the anchor residue)	false
-frag3/5/9_weight	Relative weight of different fragment libraries in ab-initio fragment insertion cycles	1.0 / 0.25 / 0.1

Appendix II: Description of the various scoring and quality assessment measures

total_score *	**Total score of the complex**
reweighted_sc	Reweighted score of the complex, in which interface residues are given double weight, and peptide residues are given triple weight
I_bsa	Buried surface area of the interface
I_hb	Number of hydrogen bonds across the interface
I_pack	Packing statistics of the interface
I_sc	**Interface score (sum over energy contributed by interface residues of both partners)**
pep_sc	**Peptide score (sum over energy contributed by the peptide to the total score; consists of the internal peptide energy and the**

	interface energy)
pep_sc_noref	Peptide score without an amino acid dependent reference energy term E_{aa}, originally introduced to bias for natural protein sequences during protein design
I_unsat	Number of buried unsatisfied HB donors and acceptors at the interface.
rms (ALL/BB/CA)	RMSD between output model and the native structure, over all peptide (heavy/backbone/C-alpha) atoms
rms (ALL/BB/CA)_if	**RMSD between output model and the native structure, over all peptide interface (heavy/backbone/C-alpha) atoms**
startRMS(all/bb/ca)	RMSD between start and native structures, over all peptide (heavy/backbone/C-alpha) atoms

For the common Rosetta scoring terms, please also see

www.rosettacommons.org/docs/latest/rosetta_basics/scoring/score-types.

* For all interface terms, the interface residues are defined as those whose C-beta atoms (C-alpha for Glycines) are up to 8A away from any corresponding atom in the partner protein.

References

1. Raveh, B., London, N., and Schueler-Furman, O. (2010) Sub-angstrom modeling of complexes between flexible peptides and globular proteins, *Proteins 78*, 2029-2040.
2. Raveh, B., London, N., Zimmerman, L., and Schueler-Furman, O. (2011) Rosetta FlexPepDock ab-initio: simultaneous folding, docking and refinement of peptides onto their receptors, *PLoS One 6*, e18934.
3. Lavi, A., Ngan, C. H., Movshovitz-Attias, D., Bohnuud, T., Yueh, C., Beglov, D., Schueler-Furman, O., and Kozakov, D. (2013) Detection of peptide-binding sites on protein surfaces: the first step toward the modeling and targeting of peptide-mediated interactions, *Proteins 81*, 2096-2105.
4. Alam, N., Zimmerman, L., Wolfson, N. A., Joseph, C. G., Fierke, C. A., and Schueler-Furman, O. (2016) Structure-Based Identification of HDAC8 Non-histone Substrates, *Structure 24*, 458-468.
5. London, N., Lamphear, C. L., Hougland, J. L., Fierke, C. A., and Schueler-Furman, O. (2011) Identification of a novel class of farnesylation targets by structure-based modeling of binding specificity, *PLoS Comput Biol 7*, e1002170.
6. Kozakov, D., Grove, L. E., Hall, D. R., Bohnuud, T., Mottarella, S. E., Luo, L., Xia, B., Beglov, D., and Vajda, S. (2015) The FTMap family of web servers for determining and characterizing ligand-binding hot spots of proteins, *Nat Protoc 10*, 733-755.