Scientific test: ligand_docking

SAMPLING FAILURES

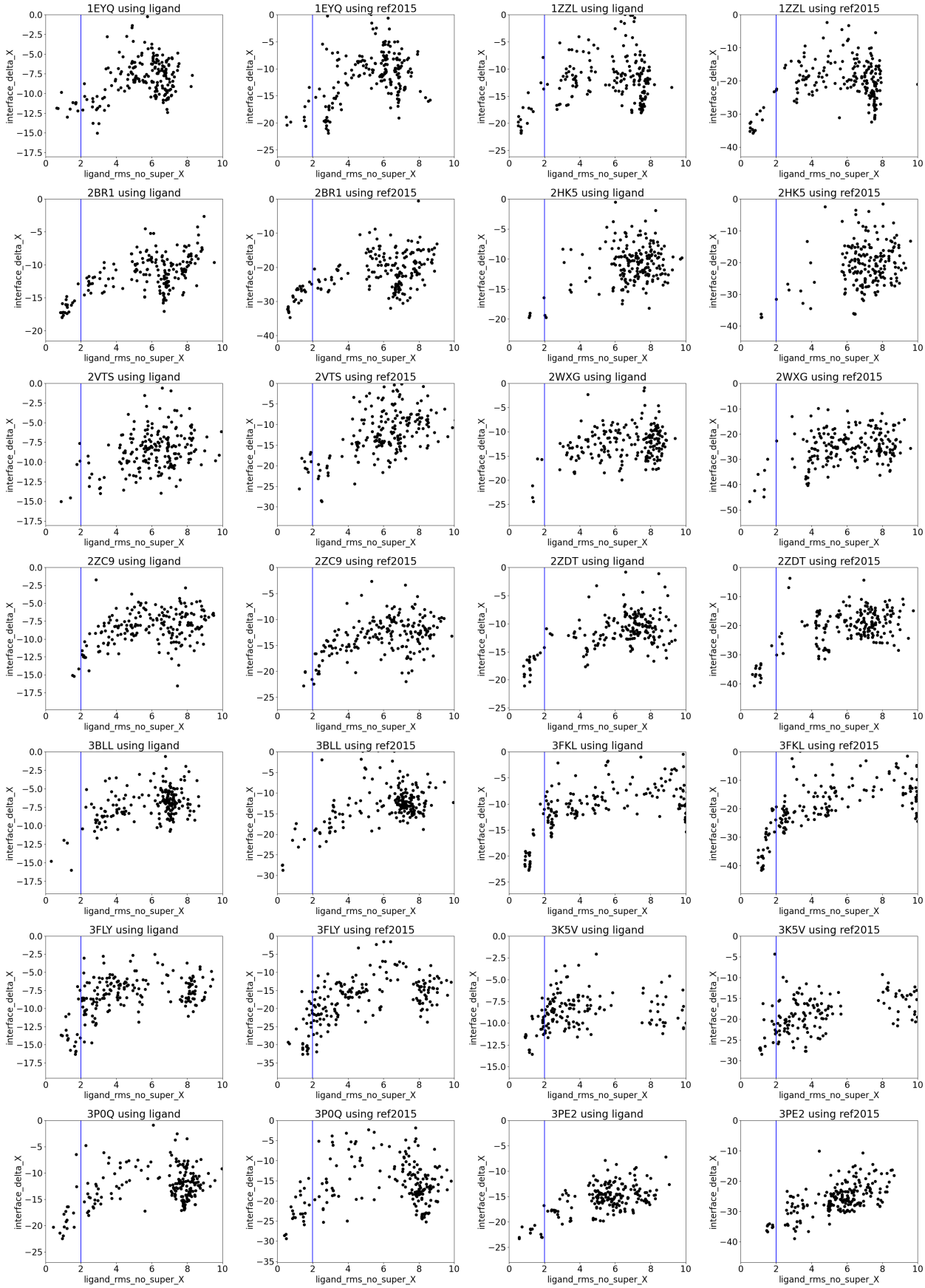
ligand: 0

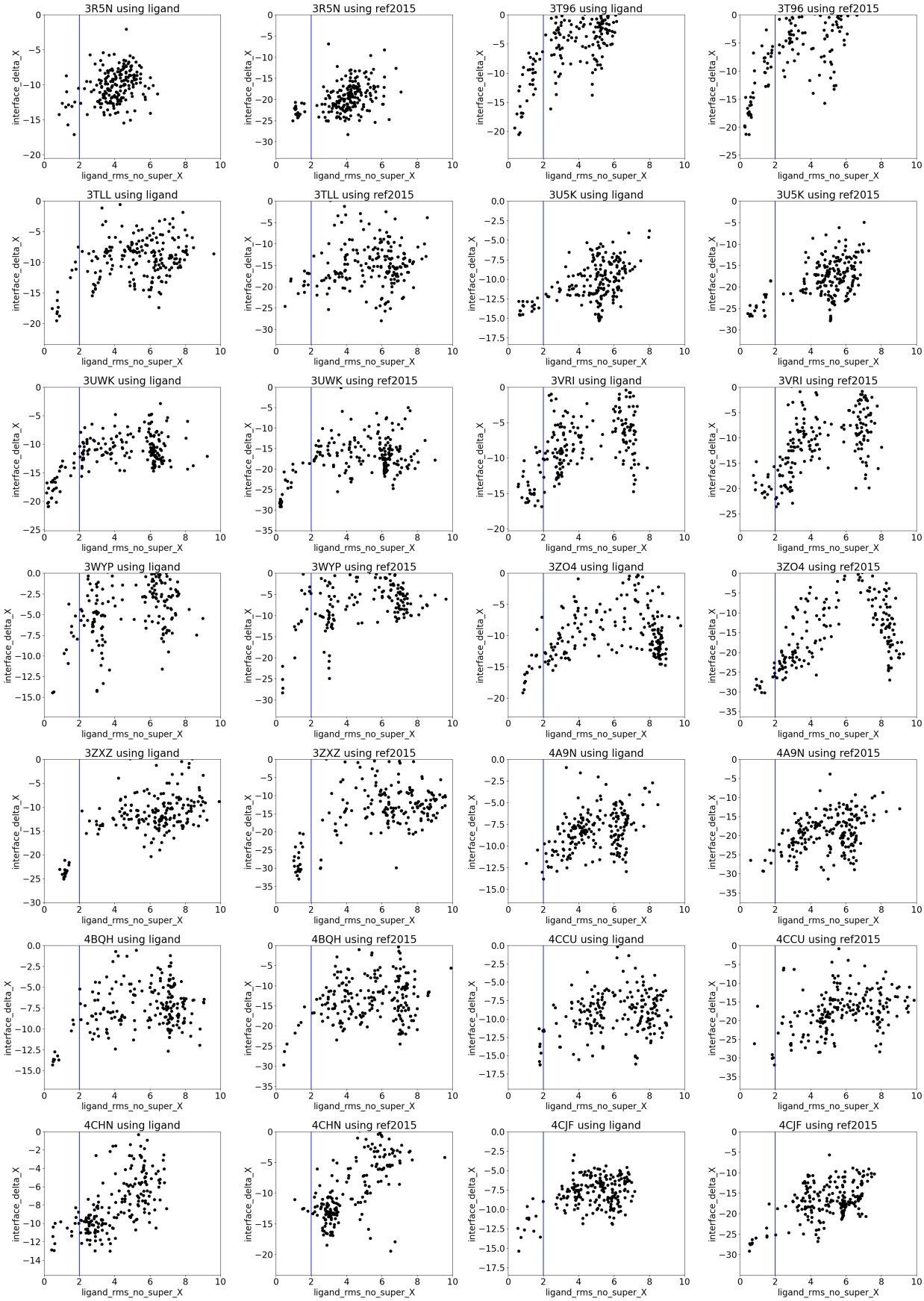
ref2015: 0

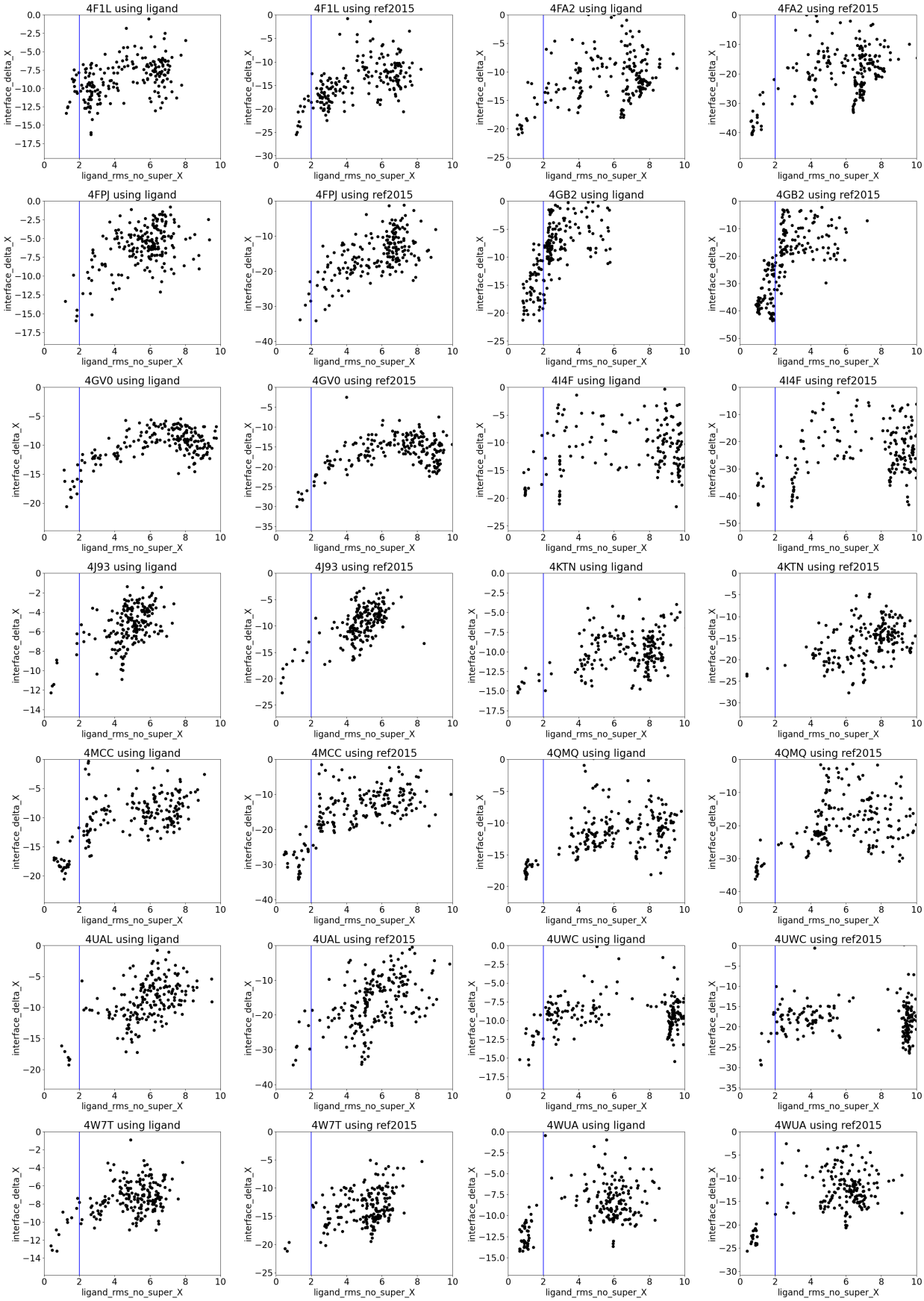
SCORING FAILURES

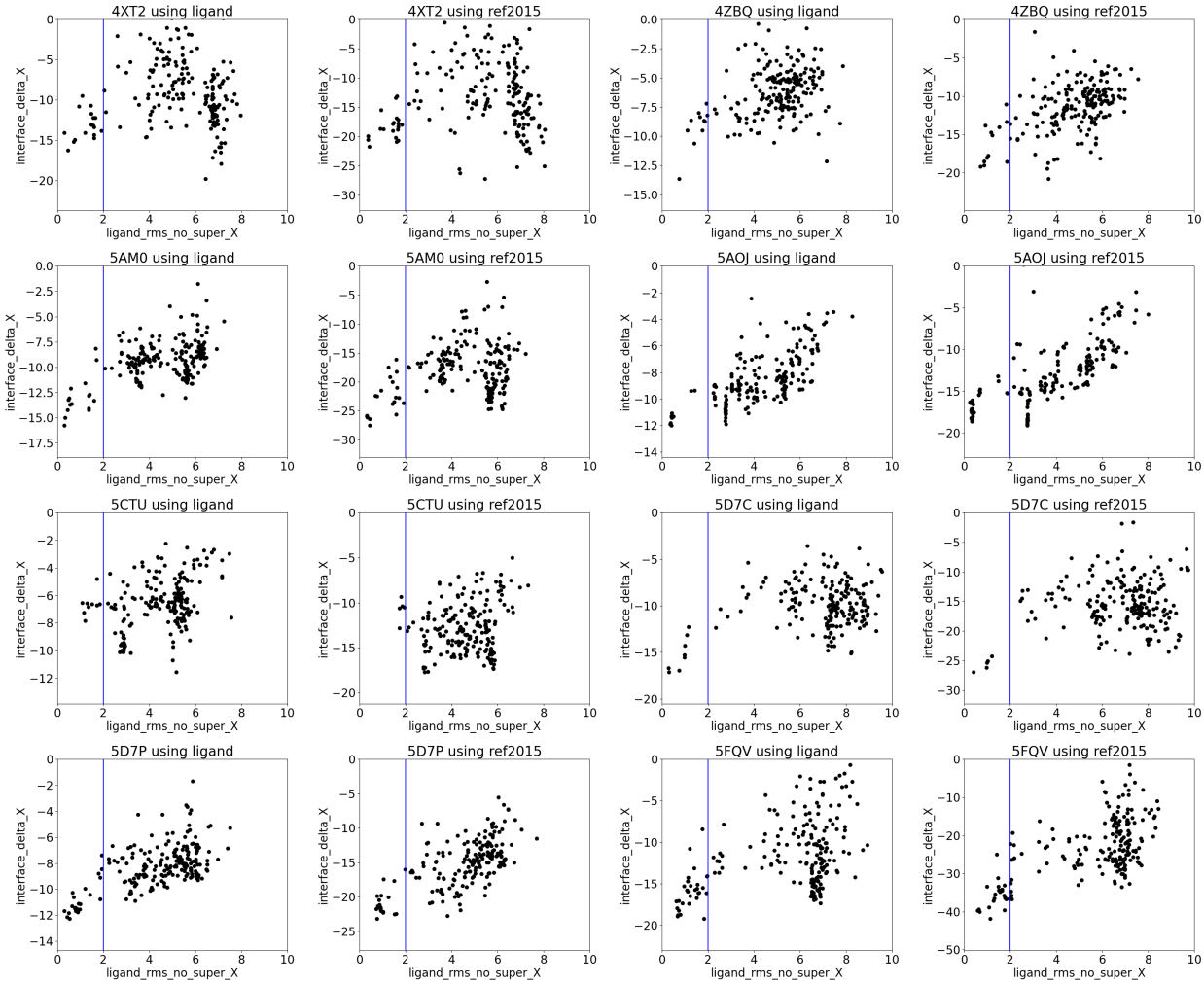
ligand: 20
1ZZL 2VTS 2ZDT 3BLL 3FKL 3K5V 3P0Q 3PE2 3U5K 3UWK 4A9N 4CCU 4FA2 4FPJ 4KTN 4UAL 4WUA 5CTU 5D7C 5FQV
ref2015: 25
1EYQ 1ZZL 2BR1 2HK5 2VTS 2WXG 3BLL 3FLY 3K5V 3PE2 3R5N 3U5K 3UWK 3ZO4 3ZXZ 4BQH 4CCU 4F1L 4I4F 4KTN 5AM0 5AOJ 5CTU 5D7C 5FQV

RESULTS


1EYQ using ligand
1EYQ using ref2015
1ZZL using ligand
1ZZL using ref2015
2BR1 using ligand
2BR1 using ref2015
2HK5 using ligand
2HK5 using ref2015
2VTS using ligand
2VTS using ref2015
2WXG using ligand
2WXG using ref2015
2ZC9 using ligand
2ZC9 using ref2015
2ZDT using ligand
2ZDT using ref2015
3BLL using ligand
3BLL using ref2015
3FKL using ligand
3FKL using ref2015
3FLY using ligand
3FLY using ref2015
3K5V using ligand
3K5V using ref2015
3P0Q using ligand
3P0Q using ref2015
3PE2 using ligand
3PE2 using ref2015
interface_delta_X
ligand_rms_no_super_X


3R5N using ligand
3R5N using ref2015
3T96 using ligand
3T96 using ref2015
3TLL using ligand
3TLL using ref2015
3U5K using ligand
3U5K using ref2015
3UWK using ligand
3UWK using ref2015
3VRI using ligand
3VRI using ref2015
3WYP using ligand
3WYP using ref2015
3ZO4 using ligand
3ZO4 using ref2015
3ZXZ using ligand
3ZXZ using ref2015
4A9N using ligand
4A9N using ref2015
4BQH using ligand
4BQH using ref2015
4CCU using ligand
4CCU using ref2015
4CHN using ligand
4CHN using ref2015
4CJF using ligand
4CJF using ref2015
interface_delta_X
ligand_rms_no_super_X


4F1L using ligand
4F1L using ref2015
4FA2 using ligand
4FA2 using ref2015
4FPJ using ligand
4FPJ using ref2015
4GB2 using ligand
4GB2 using ref2015
4GV0 using ligand
4GV0 using ref2015
4I4F using ligand
4I4F using ref2015
4J93 using ligand
4J93 using ref2015
4KTN using ligand
4KTN using ref2015
4MCC using ligand
4MCC using ref2015
4QMQ using ligand
4QMQ using ref2015
4UAL using ligand
4UAL using ref2015
4UWC using ligand
4UWC using ref2015
4W7T using ligand
4W7T using ref2015
4WUA using ligand
4WUA using ref2015
interface_delta_X
ligand_rms_no_super_X



AUTHOR AND DATE

Adapted for the current benchmarking framework by Shannon Smith (shannon.t.smith.1@vanderbilt.edu; Meiler Lab), Feb 2019

PURPOSE OF THE TEST

This benchmark is meant to test how well we discriminate native small molecule binding orientations from decoys based on the interface score term by performing standard ligand docking experiments across 50 diverse protein-ligand complexes.

BENCHMARK DATASET

The dataset consists of 50 protein-ligand complexes extracted from the Diverse Platinum Dataset (Friedrich, N. et al. J. Chem. Inf. Model, 2017). In addition to the filters described thoroughly in Friedrich et al., the maximum resolution allowed was reduced to 2.0A, drug-like ligands were chosen using standard Lipinski Rules (Lipinski, C.A., et al. Advanced Drug Delivery Reviews, 1997), and visually inspected for unrealistic orientations and crystallographic artefact. This dataset was also filtered to eliminate cofactors or multiple ligands witin the binding site. The full list of protein-ligand complexes is given in the 1.submit.py file by PDB ID.

PROTOCOL

Structure preparation:

Proteins were extracted directly from the PDB according to their PDBIDs, underwent minimal cleaning to remove the ligand. Note that no prior relax or other structural manipulations were performed prior to beginning the docking run.

Ligand preparation:

Initial ligand structures were downloaded from the Protein Data Bank using the ligand ideal SDF. Note that this is not the same as the structure from the input cocrystal structure in order to minimize conformational bias towards a particular pre-generated pose. Ligand files were cleaned using OpenBabel (J. Cheminf. 2011, 3, 33), run through BCL Conformer Generator (Kothiwale, Meiler. J. Cheminform., 2015) and generated Rosetta-readable parameter files according to the following scripts:

BCL Conformer Generator

bcl.exe molecule:ConformerGenerator -rotamer_library cod -top_models 100 -ensemble_filenames NAME.sdf -conformers_single_file NAME_conformers.sdf -conformation_comparer 'Dihedral(method=Max)' 30 -max_iterations 1000

Rosetta Parameter File Generation:

/programs/x86_64-linux/rosetta/3.8/main/source/scripts/python/public/molfile_to_params.py -n $NAME -p $NAME --mm-as-virt --conformers-in-one-file NAME_conformers.sdf --chain X

This protocol currently uses 50 protein-ligand complexes, each containing the input file containing the cleaned protein + input ligand PDB (target_input.pdb), the native protein-ligand complex for RMSD calculations (target_native.pdb), the ligand params file (target_ligand.params) and the ligand conformer library file pointed to by the params file (target_ligand_conformers.pdb).

NOTE: the protein and ligand preparation steps were performed previously and are given as the input in the data/ directory. In other words, these steps are not performed each time this benchmark is run.

Each test takes ~8 CPU hours x 50 tests = ~400 CPU hours.

PERFORMANCE METRICS

The big question that this benchmark intends to test is how well we discriminate native versus non-native binding poses. A run is determined successful if there is an near-native (<2A) structure within the top 1 percent of models based on the interface_delta_X score.

Sampling failure is defined as having no sub-2A output structures.

Scoring failure is defined as not having a sub-2A output structure within the top 10% ranked by interface score (this is a pretty large margin and may adjust accordingly).

A pass/fail is defined in the following way: Each score function has a number of allowed failures, i.e. for how many targets a scoring failure exists, out of 50 targets. The cutoffs for these were derived by looking at 10 runs, taking the maximum number of targets that fail, plus 2. The cutoffs for these are defined in the 9.finalize.py step. If any scorefunction has more targets failing than this cutoff, the entire test fails.

KEY RESULTS

This benchmark is meant to be a longitudinal test to determine how changes in the scorefunction impact small-molecule docking performance.

Performance varies greatly across different test cases, so I am not sure how to go about grading overall performance. This also makes it difficult to define a binary pass/fail criteria to the entire benchmark.

DEFINITIONS AND COMMENTS

There is a bit of noise in the runs, so you may need to compare several runs to get better sense of how different scorefunctions compare.

LIMITATIONS

The run-to-run variablility is rather high, which might indicate that the benchmark could be improved by running several output structures for each input, rather than just the current one.

This benchmark currently utilizes the ligand scorefunction (an off-shoot of the score12 environment), as this is currently the best protocol for ligand docking in Rosetta. Could be updated for talaris2014, but performance has been shown to deteriorate in the newer scoring environments, notably ref2015 and later.