Computationally-designed peptide macrocycle inhibitors of New Delhi metallo-β-lactamase 1

Supplementary Information

Vikram Khipple Mulligan[1,2], Sean Workman[3*], Tianjun Sun[3*], Stephen Rettie[2], Xinting Li[2], Liam J. Worrall[3], Timothy W. Craven[2], Dustin T. King[3], Parisa Hosseinzadeh[2], Andrew M. Watkins[4], P. Douglas Renfrew[1], Sharon Guffy[5], Jason W. Labonte[6,7], Rocco Moretti[8], Richard Bonneau[1,9,10], Natalie Strynadka[3], and David Baker[2]

1. Center for Computational Biology, Flatiron Institute, 162 Fifth Avenue, New York, NY 10010, USA.
2. Institute for Protein Design, Dept. of Biochemistry, University of Washington, Molecular Engineering and Sciences, 4000 15th Ave NE Room 420, Seattle, WA 98195, USA.
3. Department of Biochemistry and Molecular Biology and the Centre for Blood Research, University of British Columbia, Life Sciences Centre, 2350 Health Sciences Mall Vancouver, BC V6T 1Z3, Canada.
4. Department of Biochemistry, Stanford University School of Medicine, 279 Campus Drive, Stanford CA 94305, USA.
5. Department of Biochemistry and Biophysics, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.
6. Department of Chemistry, Franklin & Marshall College, 415 Harrisburg Avenue, Lancaster, PA 17604, USA.
7. Department of Chemical & Biomolecular Engineering, Johns Hopkins University, 3400 North Charles Street, Baltimore, MD 21218, USA
8. Center for Structural Biology, Department of Chemistry, Vanderbilt University, 465 21st Ave S., Nashville, TN 37240, USA.
9. Center for Genomics and Systems Biology, Department of Biology, New York University, 12 Waverly Pl., New York, NY, 10003, USA.
10. Courant Institute of Mathematical Sciences, Department of Computer Science, New York University, 251 Mercer St., New York, NY, 10012, USA.

* Sean Workman and Tianjun Sun contributed equally to this study.

Table of Contents

1. Enhancements to the Rosetta software suite **3**
1.1 Mainchain torsional potentials for non-canonical amino acids 3
1.2 Side chain torsional potentials for non-canonical amino acids 3
1.3 Design-centric guidance scoring terms 4
1.3.1 The netcharge scoring term 4
1.3.2 The buried_unsatisfied_penalty scoring term 5
1.3.3 The hbnet scoring term 6
1.3.4 The voids_penalty scoring term 7
1.4 Support for modelling metalloproteins 8
1.5 Other miscellaneous code improvements 9
1.5.1 Filtering efficiently based on internal hydrogen bond counts 9

1.5.2 Improved handling of cutpoints in polymer macrocycles 9
1.5.3 A more consistent set of interface conventions for non-canonical design 10
1.5.4 Ensemble analysis during peptide conformational sampling 11

2. Software protocols 11
2.1 Software protocols for the current version of Rosetta 11
2.1.1 Design protocol for NDM1i-1 peptides 12
2.1.2 Design protocol for NDM1i-3 peptides 27
2.1.3 Design protocol for NDM1i-4 peptides 40
2.1.4 Protocol for estimating binding free energies for the current version of Rosetta 49
2.1.5 Protocol for predicting fold propensity and folding free energies 54
2.1.6 Backbone conformational bin analysis 56
2.1.7 Comparison of NDM1i-3D design and crystal structures 56
2.2 Software protocols for legacy versions of Rosetta 60
2.2.1 Legacy code for early designs 60
2.2.2 Design protocol for NDM1i-1 peptides with Rosetta weekly build 2016.46 60
2.2.3 Design protocol for NDM1i-3 peptides with Rosetta weekly build 2018.19 74

3. Experimental methods 84
3.1 Peptide synthesis, cyclization, and purification 84
3.2 Determination of peptide concentration 85
3.3 NDM-1 expression and activity assays 85
3.4 Co-crystallization of NDM1i peptides with NDM-1 90
3.4.1 Expression of NDM-1 for crystallization 90
3.4.2 Crystallization and structure determination 90

4. Extended results 92
4.1 IC50 and KI values for all peptides 92
4.2 Analysis of NDM1i-2 peptides (variants of peptide NDM1i-1G) 94
4.3 Analysis of NDM1i-4 peptides (variants of peptide NDM1i-3D) 95

5. Supplementary figures 96

6. Supplementary references 103

1. Enhancements to the Rosetta software suite

The work described here involved substantial enhancements to the Rosetta software suite (1). A considerable amount of this work was carried out in 2017 at the Non-Canonical Amino Acids eXtreme Rosetta Workshop (NCAA-XRW), a hackathon hosted at the Flatiron Institute in New York City (2). Because these modifications were made incrementally and concurrently with the rounds of design work, different rounds of design were carried out with different versions of the Rosetta software. In this section, we describe the changes made to Rosetta. In **Section 2.1**, we provide RosettaScripts XML protocols (3) for each round of design, updated to run efficiently with current and future versions of Rosetta (all weekly releases following 2020.11, and all versioned releases from Rosetta 3.13 onward). In **Section 2.2**, we provide the legacy scripts to allow exact reproduction of the protocol used to produce the results reported, along with information about the legacy version of Rosetta used. In addition to being listed here, all scripts and input files are also available from the public Github repository **vmullig/ndm1_design_scripts** (https://github.com/vmullig/ndm1_design_scripts), and are made available under the permissive MIT licence. All changes and new modules have been incorporated into the public releases of Rosetta, and are documented on the Rosetta help wiki (https://www.rosettacommons.org/docs/latest/Home). Rosetta is licenced through UW CoMotion to for-profit users and corporations for a licencing fee, and is made freely available to academics, governments, and not-for-proft users. To obtain Rosetta, please visit https://els2.comotion.uw.edu/product/rosetta. For full information on the Rosetta project, see https://www.rosettacommons.org/.

1.1 Mainchain torsional potentials for non-canonical amino acids

Traditionally, Rosetta provided mainchain potentials for canonical amino acid residues. These were based on statistical information from the structures deposited in the Protein Data Bank. We added support for a library of non-canonical mainchain potentials, which can be generated using molecular mechanics or quantum mechanics computations. A mainchain potential for 2-aminoisobutyric acid (AIB) was generated using molecular mechanics methods. Full details of this work will be described elsewhere (4). In the case of L-α-amino acids with exotic side chains that resemble those of the canonical amino acids, such as L-2-aminomethyl phenylalanine (L-A34) or L-norleucine (L-Nlu), support was added for using the Ramachandran potential from a similar canonical amino acid (phenylalanine and methionine, respectively, for L-A34 and L-Nlu). Rosetta's **rama_prepo** scoring term, which is part of the default **ref2015** energy function (5, 6), now supports these mainchain potentials for non-canonical building blocks, allowing them to be used in any Rosetta protocol alongside the heavily optimized protein-centric energy function.

1.2 Side chain torsional potentials for non-canonical amino acids

Support for generating backbone-dependent side chain rotamer libraries for exotic non-canonical amino acids was first reported in 2012 (7). The **MakeRotLib** method generates such libraries by sampling conformations of an amino acid dipeptide, with limited

energy-minimization using a molecular mechanics force field. Although this method produces estimates of the conformational energy and breadth of each rotamer well, it has historically been used solely for conformational sampling, not for calculating energies. We added support for using the energies generated by **MakeRotLib** as a smoothly-interpolated lookup table to construct a backbone-dependent side chain potential for non-canonical amino acids (or other exotic building-blocks) comparable to the potentials that exist for canonical amino acids. Since rotamer wells for exotic side chains often do not fall neatly into ±gauche and anti bins, we used a Voronoi-based approach, in which rotamer well boundaries are defined as the hyper-plane midway between well centres. Rosetta's **fa_dun** scoring term, which is part of the default **ref2015** energy function (5, 6), now supports both canonical and non-canonical building-blocks, allowing both to be used together without altering the scoring function, in any Rosetta protocol. Considerable refactoring has ensured that this term and the rotamer code are fully general, without canonical-specific assumptions. Rotamer libraries for non-canonical amino acids can be downloaded from the Rosetta download links.

1.3 Design-centric guidance scoring terms

We previously described the amino acid composition (**aa_composition**) penalty term, which allows the user to define "sequence constraints" and to impose a nonlinearly-ramping penalty for deviation from a desired amino acid composition during rotamer optimization, allowing optimization of packing interaction energy subject to constraints regarding desired composition (8). This was the first of a suite of design-centric guidance scoring terms, which add non-physical penalties or bonuses to the energy function to encourage desired features or to discourage undesired features during design, converting an energy optimization problem into a simultaneous multi-objective optimization problem. This allows a designer either to impose prior knowledge about the properties that are likely to lead to success of the design objective, or to impose requirements for other properties needed for production, experimental characterization, or other parallel objectives. Since design algorithms typically select side chain rotamers to minimize the scoring function used, the addition of terms to the energy function provides a convenient means of controlling any Rosetta design algorithm that is based on rotamer optimization. The sole caveat is that many such terms are not pairwise-decomposable, so the challenge is to develop a means by which a design-centric guidance term can be fast to compute and to update during a rotameric search. Additional design-centric guidance terms that were added for the work described here include the **netcharge**, **buried_unsatisfied_penalty**, **hbnet**, and **voids_penalty** scoring terms. These are described below, and a full usage guide is provided on the Rosetta help wiki, with detailed descriptions of the interface and algorithm for each: https://www.rosettacommons.org/docs/latest/rosetta_basics/scoring/design-guidance-terms.

1.3.1 The netcharge scoring term

Where **aa_composition** allows a designer to control the desired amino acid composition in a design or a region within a design, the **netcharge** term equivalently allows a designer to control the net charge. Like **aa_composition**, the term applies a nonlinearly-ramping penalty for deviation from a desired net charge, which guides sequence optimization heuristics towards solutions with the desired net charge. Desired net charge, and the shape of the penalty function

as one moves to more positive or more negative net charges, are controlled by net charge constraints which are attached to the Rosetta pose (structure object) and respected by the **netcharge** scoring term. Special Rosetta movers (Rosetta modules which operate on a pose and alter it in some way) attach net charge constraints to a pose or remove them from a pose.

The code is organized modularly. The **NetChargeEnergy** class, which defines the energy method itself. is located in the Rosetta source code in **source/src/core/energy_methods/**, and is found in the **core::scoring::netcharge_energy** namespace (a rare violation of the matching relationship between namespace and directory structure usually enforced in Rosetta, necessitated by a recent code reorganization). The **NetChargeConstraint** class in the same namespace is located in **source/src/core/scoring/netcharge_energy/**, and defines an object that can be embedded in a Rosetta pose. A special mover, the **AddNetChargeConstraintMover**, takes as input a **.charge** file defining desired net charge and penalties for deviating from this desired charge, and constructs corresponding **NetChargeConstraint** objects, attaching them to a pose. This mover is located in **source/src/protocols/aa_composition/**, and in the **protocols::aa_composition** namespace. In the same directory and namespace, the **ClearCompositionConstraintsMover** removes all sequence constraints (**aa_composition** and **netcharge** constraints) from the pose; alternatively, the **ClearConstraintsMover**, located in the **source/src/protocols/constraint_movers/** directory and the **protocols::constraint_movers** namespace, provides a means of removing all sequence and geometric constraints from a pose. Full documentation of the **netcharge** term is provided here: https://www.rosettacommons.org/docs/latest/rosetta_basics/scoring/NetChargeEnergy.

1.3.2 The buried_unsatisfied_penalty scoring term

The NDM-1 active site features a number of hydrophobic amino acids on the inner face of the hinge loop, but many polar groups are also present on the front loop and deep within the pocket. Additionally, the peptide backbone presents polar amide protons and carbonyl oxygens. If any of these is unsatisfied in the bound configuration, the system could favour the unbound state, in which water can surround the peptide and fill the active site to satisfy all hydrogen bond donors and acceptors.

The pairwise-decomposable default Rosetta energy function cannot penalize unsatisfied hydrogen bond donors and acceptors, since identification of an unsatisfied polar group requires explicit consideration of everything that might be bonding to it. To address this, we implemented a non-pairwise scoring term, called the **buried_unsatisfied_penalty**, to provide a quadratically-ramping penalty for unsatisfied buried polar groups during design, guiding the packer to solutions with full satisfaction. This term carries out a precomputation prior to packing, in which a graph is constructed with nodes for each candidate rotamer and edges between each pair of rotamers that form one or more hydrogen bonds. Nodes store the buried polar groups associated with each rotamer, with burial defined by the method of side chain neighbors (9). Full hydrogen bonding calculations are performed a single time during this precomputation, representing the bulk of the computational cost. During the actual packing trajectory, each rotamer substitution triggers two iterations over all currently-selected rotamers. In the first pass, hydrogen bonds to each polar group stored in each node, based on edges

between currently-selected rotamers, are tallied. In the second pass, polar groups with tallies of zero are counted. The count is then squared and multiplied by a weighting factor, which is returned as the score associated with this term.

The **BuriedUnsatPenalty** class, which implements the energy method for this scoring term, is located in **source/src/core/pack/guidance_scoreterms/buried_unsat_penalty/**, and in the associated **core::pack::guidance_scoreterms::buried_unsat_penalty** namespace. Full documentation for this scoring term may be found on the Rosetta help wiki: https://www.rosettacommons.org/docs/latest/rosetta_basics/scoring/BuriedUnsatPenalty.

1.3.3 The hbnet scoring term

A polar group making a hydrogen bond to water is roughly isoenergetic to a polar group making a hydrogen bond to another polar group in a peptide or protein. When multiple polar groups can be pre-organized to form a network of hydrogen bonds, however, there is an entropic advantage to internal hydrogen bonding over satisfying the same polar groups with hydrogen bonds to water. This can help to stabilize the conformation (*e.g.* the bound state) associated with that hydrogen bonding network. In the past, specialized algorithms such as the **HBNet** algorithm were used to design hydrogen bond networks, followed by conventional packing of apolar side chains around the pre-designed networks (10). Although this can be effective, there are many situations in which one finds hydrogen bond networks that are incompatible with close packing of other side chains. Simultaneous optimization for hydrogen bond networks and for close packing would be preferable. Unfortunately, the identification of a hydrogen bond network is a fundamentally non-pairwise decomposable problem.

We implemented a design-centric guidance scoring term, called the **hbnet** term, to provide a quadratically-ramping bonus based on the size of a hydrogen bond network, where networks are defined as two or more residues connected by side chain-side chain or side chain-backbone hydrogen bonds. Like the **buried_unsatisfied_penalty**, the **hbnet** scoring term pre-computes all possible hydrogen bonds between all pairs of candidate rotamers, then constructs a graph representing hydrogen bonds between residues that it updates as rotamer substitutions are made. For each substitution, an $O(N+E)$ connected components calculation is performed on the updated hydrogen bond graph (where N is the number of nodes, corresponding to selected rotamers, and E is the number of edges, corresponding to hydrogen-bonding interactions between residues) to count the size of all hydrogen bond networks. Each count is squared, the squares are summed, and the sum multiplied by -1 and by the energy term's weight to produce the bonus. Adding this bonus to the energy function allows simultaneous optimization for hydrogen bonds and for conventional packing.

The scoring term is implemented in the **HBNetEnergy** class, which is located in **source/src/core/pack/guidance_scoreterms/hbnet_energy/** directory, and in the corresponding **core::pack::guidance_scoreterms::hbnet_energy** namespace. Full documentation is available here: https://www.rosettacommons.org/docs/latest/rosetta_basics/scoring/HBNetEnergy.

1.3.4 The voids_penalty scoring term

When designing interfaces, voids in an interface are undesirable features that result from poor shape complementarity between the two molecules. Voids in a protein core are similarly undesirable when performing *de novo* design. Since Rosetta's energy function was historically residue-level pairwise-decomposable in order to be fast to compute and to update during rotamer optimization, it was not possible to recognize voids, let alone to penalize them (since identification of a void fundamentally depends on explicit consideration of all residues to determine that no residue occupies the void). Techniques such as RosettaHoles were developed to allow *post-hoc* filtering of designs with buried voids (11, 12), but these techniques were too slow to apply them repeatedly during optimization in order to create designs optimized to have few buried voids, and were incompatible with simultaneous optimization for low overall energies.

We developed the **voids_penalty** scoring term to provide a means of optimizing designs for few buried voids. This term is able to achieve the speed necessary by carrying out all volumetric computations in a pre-calculation: prior to rotamer optimization, a voxel grid is superimposed onto the pose, and buried voxels are identified by the method of side chain neighbors described previously (9). Next, each candidate rotamer is superimposed in turn on the voxel grid, and the number of buried voxels that rotamer *i* overlaps is counted and converted to a volume (v_i) that is stored with the i^{th} rotamer. Any buried voxels that no rotamer can reach are discarded at this point, and the total number of voxels reachable by the rotamers under consideration is also converted to a volume (V) and stored. At this point, the voxel grid is purged from memory, and the most computationally-expensive stage of the calculation has completed.

During the rotamer optimization performed by the Rosetta packer, the sum of the volumes of the currently-considered rotamers is subtracted from the total buried accessible volume at each step and squared, and this value is added to overall energy function as a penalty (**Expression S1**). Intuitively, this penalty is minimized (zero) if the currently-selected rotamers have a total buried volume equal to the buried volume that we are attempting to fill; if the rotamer volume is greater or less, the penalty is positive. This can correspond to a void-free solution, but this can also occur if rotamers with the correct total volume overlap, leaving voids; however, the Rosetta energy function's **fa_rep** term strongly penalizes clashes, meaning that the lowest-energy solution has a high likelihood of being a clash-free solution with rotamer volume most closely matching the volume to fill: a voids-free or nearly voids-free solution.

$$E_{voids} = w_{voids}\left(V - \sum_{j=1}^{N} v_{S(j)} \right)^2 \qquad \textbf{(S1)}$$

In the above, E_{voids} is the energy associated with this scoring term, w_{voids} is the weight given to the term, V is the total buried volume to fill, $S(j)$ is the index of the rotamer currently selected at the j^{th} position, $v_{S(j)}$ is the buried volume of the rotamer currently selected at the j^{th} position, and N is the number of packable positions in the rotamer optimization problem. The **voids_penalty** energy method is defined in the VoidsPenaltyEnergy class, located in the

source/src/core/pack/guidance_scoreterms/voids_penalty_energy/ directory and the corresponding **core::pack::guidance_scoreterms::voids_penalty_energy** namespace. Full documentation for this energy term, including command-line flags or scriptable energy method options controlling the determination of which voxels are buried and the resolution of the voxel grid, is available on the Rosetta help wiki: https://www.rosettacommons.org/docs/latest/rosetta_basics/scoring/VoidsPenaltyEnergy.

1.4 Support for modelling metalloproteins

More than 30% of proteins are metalloproteins (13). Quantum mechanical (QM) effects give rise to the coordination geometry of metals and their liganding groups, but these are poorly captured by quasi-Newtonian energy functions like the Rosetta energy function (5). This hinders computational peptide and protein design efforts involving bound metals. In the past, case-specific approaches were used to model and to design metalloproteins, usually relying on metals represented as virtual atoms as part of custom residue types representing liganding side chains(14, 15) or custom constraints defining metal-binding geometry (13). Recognizing that a considerable subset of design cases were ones in which the metal-coordination geometry was already known from an input structure (as it was in the case of NDM-1 inhibitor design), we added support in Rosetta for automatically detecting metal-liganding side chain interactions and for automatically setting up constraints to preserve input metal coordination geometry during energy minimization, avoiding the need for QM calculations to preserve this geometry.

This functionality may be invoked from the command-line (where the option **-auto_setup_metals** triggers the functionality for any input PDB files) or from RosettaScripts or PyRosetta using the **SetupMetalsMover**. A new scoring term, the **metalbinding_constraint**, was added to penalize deviation from the input geometry at metal centres, based on constraints added to the pose by the **-auto_setup_metals** or **SetupMetalsMover** code. Core functions for detecting metal-metal liganding group interactions are located in the **core::util** namespace, and are in Rosetta C++ header file **source/src/core/util/metalloproteins_util.hh**. The **ConstraintGraph** and **ConstraintEnergyContainer** classes, both in namespace **core::scoring::constraints** and directory **source/src/core/scoring/constraints/**, have both been expanded to recognize the new **metalbinding_constraint** type. Residue type parameter files, located in the Rosetta database in **database/chemical/residue_type_sets/**, now contain **METAL_BINDING_ATOMS** lines listing the atoms that can potentially bind a metal ion; this functionality translates to additional data stored in the **ResidueTypeBase** and **ResidueType** classes (both in the **core::chemical** namespace and **source/src/core/chemical/** directory). Finally, the **SetupMetalsMover** is located in the **protocols::simple_moves** namespace and in the **source/src/protocols/simple_moves/** directory.

Full documentation for the **-auto_setup_metals** flag is available here: https://www.rosettacommons.org/docs/latest/rosetta_basics/non_protein_residues/Metals. The **SetupMetalsMover** is also documented in full, with usage examples, on this page: https://www.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/Movers/movers_pages/SetupMetalsMover.

1.5 Other miscellaneous code improvements

1.5.1 Filtering efficiently based on internal hydrogen bond counts

In past peptide design work (4, 8, 16, 17), we have observed that conformations with higher numbers of internal backbone-backbone hydrogen bonds are more readily stabilized through sequence design. During backbone conformational sampling of a poly-glycine chain in the context of the NDM-1 active site (the precursor step to sequence design), we therefore sought to discard sampled conformations with few internal backbone hydrogen bonds, excluding from our counts hydrogen bonds between residues that were close together in covalent connectivity. Early rounds of design relied on an inefficient approach (see **Section 2.2.2**) in which hydrogen bonds were recomputed repeatedly. Although a single count was fast (on the order of a half second on a single CPU), the fact that most sampled conformations would be discarded meant that this was the rate-limiting step in our design protocol, justifying work to make this count more efficient. We therefore implemented two new modules in Rosetta: the **PeptideInternalHbondsMetric** (https://www.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/SimpleMetrics/simple_metric_pages/PeptideInternalHbondsMetric), which efficiently counts backbone hydrogen bonds within a macrocycle and omits from the count hydrogen bonds between residues within a threshold covalent connectivity distance of one another, and the **PeptideInternalHbondsFilter** (https://www.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/Filters/filter_pages/PeptideInternalHbondsFilter), which invokes the **PeptideInternalHbondsMetric** and then accepts or discards the structure based on the internal backbone hydrogen bond count. These modules are more than tenfold faster, and considerably reduce the overall computation time while also reducing user error by providing a much simpler user interface. Scripts in **Section 2.1** have been updated to use the more efficient calculator, while scripts in **Section 2.2** use the legacy approach. Note that these modules are also compatible with non-cyclic geometry, and may be used in a conventional protein design context.

1.5.2 Improved handling of cutpoints in polymer macrocycles

Rosetta was originally developed to handle linear heteropolymers. The internal coordinate system used during conformational sampling and energy minimization depends on an directed acyclic graph (DAG) representation of the kinematic relationships between residues in a structure. Unfortunately, macrocyclic geometry necessarily means that at least one covalent

connection will be at a breakpoint in the kinematic DAG. During energy minimization, this means that this bond can stretch, compress, or otherwise adopt nonphysical conformations.

In earlier design scripts (see **Section 2.2.2**), we handled this by adding geometric constraints to the structure to ensure that ideal amide bond lengths, angles, and dihedrals were preserved for the N-to-C amide bond in a macrocycle during energy minimization. Since the carbonyl oxygen and amide proton for this bond had positions that did not correctly depend on the position of the connected residue, they would sometimes drift during energy minimization; our workaround was to snap these atoms back to their ideal positions periodically by re-declaring the chemical bond between the terminal atoms, which had the side-effect of updating positions of atoms that were dependent on this bond.

During the NCAA-XRW, we greatly refactored Rosetta's chainbreak code to allow this to be applied to macrocyclic geometry, removing assumptions about i → i+1 connections that had previously made it a term specific to protein loops while considerably refactoring the underlying functionality. The chainbreak code uses CUTPOINT variant types and the **chainbreak** energy term to impose a penalty for deviation from ideal amide bond geometry. Post-refactoring, these variant types add virtual atoms connected to the atoms flanking a break in the kinematic DAG, and the **chainbreak** energy term creates a harmonic potential holding these virtual atoms atop their real counterparts in the other residue. This has the advantage of both enforcing bond lengths and bond angles, while simultaneously allowing flanking dihedrals to rotate freely, with the position of atoms dependent on the bond (amide protons and carbonyl oxygens) tied to the position of the virtual atom. This ensures that these atoms do not drift to physically unrealistic positions without discrete updates to their positions or undue rigidity of the flanking residues.

1.5.3 A more consistent set of interface conventions for non-canonical design

In Rosetta, design is controlled by **TaskOperations**, which impose rules limiting the set of amino acids allowed on a position-specific basis. Historically, canonical and non-canonical amino acids had opposite conventions, with **TaskOperations** only allowed to *prohibit* canonical types and only allowed to *enable* non-canonical types. This contributed heavily to user confusion and user error, and necessitated undesirable workarounds that broke commutativity of these operations. We have greatly refactored **TaskOperation** code to make canonical and non-canonical design share the same set of conventions, and we have added the concept of **PackerPalettes**, which define the set of amino acids (canonical or non-canonical) with which one is designing in the absence of task operations to restrict the design set. These changes will be described more fully elsewhere (4).

1.5.4 Ensemble analysis during peptide conformational sampling

The metric P_{Near}, described previously, represents the estimated probability of finding a molecule in a region of conformation space near a desired conformation, where the region is defined with a soft boundary for numerical stability on repeated simulation (8, 16, 18). Given a large conformational sampling simulation producing N samples and reporting an energy E_i and an RMSD value from a desired conformation r_i for the *ith* sample, the definition of P_{Near} is shown in **Expression S2**.

$$P_{Near} = \frac{\sum_{i=1}^{N} e^{-\frac{r_i^2}{\lambda^2}} e^{-\frac{E_i}{k_B T}}}{\sum_{j=1}^{N} e^{-\frac{E_j}{k_B T}}} \quad \textbf{(S2)}$$

In the above, the parameters λ (in Å, or distance units matching r_i and r_j) and k_BT (in kcal/mol, or energy units matching E_i and E_j) govern the degree of allowed deviation from the desired conformation and the influence of higher-energy states, respectively. We added support for automatically computing the P_{Near} metric, based on the ensemble of sampled conformations, to the Message Passing Interface (MPI) build of the Rosetta **simple_cycpep_predict** application. We also added automatic conversion of this quantity to an estimate of the $\Delta G_{folding}$ of a peptide. Since:

$$K_{eq} = \frac{[folded]}{[unfolded]} \approx \frac{P_{Near}}{1-P_{Near}} \quad \textbf{(S3)}$$

It follows that:

$$\Delta G_{folding} = -RT\ ln(K_{eq}) = -nk_BT\ ln(K_{eq}) \approx -nk_BT\ ln\left(\frac{P_{Near}}{1-P_{Near}}\right) \quad \textbf{(S4)}$$

In the above, n is Avogadro's number.

2. Software protocols

2.1 Software protocols for the current version of Rosetta

All protocols listed in this section have been tested using Rosetta Git SHA1 cb360c57ed4ba63d45678afba4bf6a39d8dd958d, the current version at the time of this writing, compiled on CentOS Linux 7 with GCC 9.3.0. All updated code will be included in future weekly releases of Rosetta after weekly release 2020.11, and will be part of Rosetta 3.13. The RosettaScripts XML scripts below have been updated for this version of Rosetta. Changes

reflect changes to the user interface, or replacement of modules with others that solve the same problem with greater computational efficiency; no changes were made that were intended to alter the setup of the problem or to better bias Rosetta to produce more scientifically useful solutions. Nevertheless, for exact scientific reproducibility, the original scripts, written for the version of Rosetta that was state-of-the-art at the time that the design work was carried out, are provided in **Section 2.2** along with information identifying the legacy Rosetta version with which they are compatible. The Rosetta software suite is made freely available to academic, government, and not-for-profit users, and is licenced for commercial use. Academic and commercial licences are available from UW CoMotion through the Rosetta Licence and Download page at this URL: https://www.rosettacommons.org/software/license-and-download.

2.1.1 Design protocol for NDM1i-1 peptides

The design protocol for NDM1i-1 peptides used a manually-edited version of the .pdb version of Protein Data Bank structure 4EXS as input, with solvent atoms stripped and chain A removed., and zinc atoms renumbered to be residues 1 and 2. The L-captopril small molecule was converted into a D-cysteine-L-proline dipeptide by changing the following lines in the PDB file as follows. These changes retain the coordinates of equivalent atoms.

```
< HETATM 3496  O1  X8Z B 301 -43.856   6.045  -3.243  0.60 23.56           O
< HETATM 3497  C4  X8Z B 301 -43.040   5.919  -2.339  0.60 22.34           C
< HETATM 3498  C2  X8Z B 301 -42.880   7.070  -1.400  0.60 20.25           C
< HETATM 3499  C1  X8Z B 301 -42.578   8.303  -2.234  0.60 18.92           C
< HETATM 3500  S   X8Z B 301 -41.340   8.004  -3.518  0.60 11.30           S
< HETATM 3501  C3  X8Z B 301 -44.200   7.252  -0.664  0.60 20.21           C
< HETATM 3502  N   X8Z B 301 -42.276   4.832  -2.144  0.60 22.21           N
< HETATM 3503  C8  X8Z B 301 -42.409   3.649  -2.993  0.60 21.81           C
< HETATM 3504  C9  X8Z B 301 -43.581   2.796  -2.539  0.60 22.79           C
< HETATM 3505  O3  X8Z B 301 -43.934   1.791  -3.225  0.60 19.19           O
< HETATM 3506  O2  X8Z B 301 -44.175   3.139  -1.483  0.60 20.40           O
< HETATM 3507  C5  X8Z B 301 -41.301   4.628  -1.070  0.60 21.86           C
< HETATM 3508  C6  X8Z B 301 -40.513   3.393  -1.476  0.60 22.00           C
< HETATM 3509  C7  X8Z B 301 -41.111   2.889  -2.776  0.60 22.32           C

> ATOM   1720  N   DCS B 273 -44.200   7.252  -0.664  0.60 20.21           N
> ATOM   1721  CA  DCS B 273 -42.880   7.070  -1.400  0.60 20.25           C
> ATOM   1722  C   DCS B 273 -43.040   5.919  -2.339  0.60 22.34           C
> ATOM   1723  O   DCS B 273 -43.856   6.045  -3.243  0.60 23.56           O
> ATOM   1724  CB  DCS B 273 -42.578   8.303  -2.234  0.60 18.92           C
> ATOM   1725  SG  DCS B 273 -41.340   8.004  -3.518  0.60 11.30           S
> ATOM   1726  N   PRO B 274 -42.276   4.832  -2.144  0.60 22.21           N
> ATOM   1727  CA  PRO B 274 -42.409   3.649  -2.993  0.60 21.81           C
> ATOM   1728  C   PRO B 274 -43.581   2.796  -2.539  0.60 22.79           C
> ATOM   1729  O   PRO B 274 -43.934   1.791  -3.225  0.60 19.19           O
> ATOM   1730  CB  PRO B 274 -41.111   2.889  -2.776  0.60 22.32           C
> ATOM   1731  CG  PRO B 274 -40.513   3.393  -1.476  0.60 22.00           C
> ATOM   1732  CD  PRO B 274 -41.301   4.628  -1.070  0.60 21.86           C
> ATOM   1733  OXT PRO B 274 -44.175   3.139  -1.483  0.60 20.40           O
```

Listing 2.1.1.1: Difference between Protein Data Bank file **4EXS.pdb** and modified file **inputs/4EXS_Dcys_Lpro.pdb**.

The following RosettaScripts XML file defines a protocol to produce NDM1i-1 designs. This protocol has been modernized to use the noncanonical amino acid input format that is now

standard, and makes use of more efficient modules for internal hydrogen bond computation and for maintaining the N-to-C amide bond geometry during energy-minimization than were available in the original 2016 scripts provided in **Section 2.2.2**. The script shown here is the preferred way to produce these designs now; the legacy script in **Section 2.2.2** should only be used for reproduction of the exact protocol used to produce NDM1i-1A through 1G.

This script extends the D-Cys-L-Pro stub with a six-residue poly-glycine sequence, appending three glycine residues and prepending three glycine residues. It then closes the chain with generalized kinematic closure (GenKIC), sampling closed conformations to find conformations with a threshold number of backbone hydrogen bonds. Each closed conformation is then designed, first with a single low-cost call to the Rosetta packer though the **PackRotamersMover**, then, if that yields a design with sufficient shape complementarity to the target, with a more expensive call to **FastDesign**. If that yields slightly higher shape complementarity, the script then attempts a Monte Carlo search, in which each move perturbs the macrocycle using GenKIC and attempts redesign with the **PackRotamersMover**. At the end of this trajectory, the top sampled conformation and sequence is subjected to relaxation with the **FastRelax** mover and returned.

```
<ROSETTASCRIPTS>
        # This script takes as input the 4EXS structure with a 2-residue peptide in the
        # active site and extends it to generate an 8-residue peptide, in an open
        # conformation.  It uses generalized kinematic closure to close the peptide, forming
        # an N-to-C peptide macrocycle.  It then designs the sequence of the peptide, and
        # applies various filters.  If filters pass, it proceeds to carry out a Monte Carlo
        # search of local conformation space, designing at each step, to try to improve the
        # shape complementarity.
        #
        # This script was updated for the Rosetta version that was current as of 20 May 2020
        # (Git SHA1 cb360c57ed4ba63d45678afba4bf6a39d8dd958d).  Weekly releases after this
        # point, and Rosetta 3.13, will be able to run it.  This version differs from the
        # version used in 2016 to generate the initial designs insofar as the newer, more
        # efficient hydrogen bond counter is used, the cyclization constraints have been
        # replaced with the chainbreak energy term and cutpoint variants, and the updated,
        # simpler, and less error-prone interface for noncanonical design has been used.  For
        # the original version (to exactly reproduce the 2016 protocol with 2016 releases of
        # Rosetta), see section 2.2.2.

    # In the SCOREFXNS section, we define scoring functions used for design and energy
    # minimization.
    <SCOREFXNS>
        # A basic scoring function, with an added penalty discouraging sequences that promote
        # formation of aspartimide byproducts during peptide synthesis.  Note that at the time
        # this script was written, the energy function used was a beta version of the energy
        # function called "beta_nov15".  It has since been renamed "ref2015", and is the
        # current default energy function in newer versions of Rosetta:
        <ScoreFunction name="r15" weights="ref2015.wts" >
            <Reweight scoretype="aspartimide_penalty" weight="1.0" />
        </ScoreFunction>

        # A "soft" variation of the scoring function with more permissive atomic repulsive
        # potentials, useful for design steps.  This scoring function also activates geometric
        # and amino acid composition constraint terms.
        <ScoreFunction name="r15_soft" weights="ref2015_soft.wts">
            <Reweight scoretype="aspartimide_penalty" weight="1.0" />
            <Reweight scoretype="atom_pair_constraint" weight="1.0" />
            <Reweight scoretype="angle_constraint" weight="1.0" />
```

```
            <Reweight scoretype="dihedral_constraint" weight="1.0" />
            <Reweight scoretype="aa_composition" weight="1.0" />
        </ScoreFunction>

        # A basic scoring function, with an added penalty discouraging sequences that promote
        # formation of aspartimide byproducts during peptide synthesis.  This version also
        # activates geometric constraint terms and upweights the chainbreak term:
        <ScoreFunction name="r15_cst" weights="ref2015_cst.wts" >
            <Reweight scoretype="chainbreak" weight="40.0" />
            <Reweight scoretype="aspartimide_penalty" weight="1.0" />
        </ScoreFunction>

        # A variation on the constrained scoring function with the aspartimide penalty term
        # activated and hydrogen bonding, electrostatic, and chainbreak terms upweighted:
        <ScoreFunction name="r15_highhbond_cst" weights="ref2015_cst.wts" >
            <Reweight scoretype="chainbreak" weight="40.0" />
            <Reweight scoretype="aspartimide_penalty" weight="1.0" />
            <Reweight scoretype="hbond_sr_bb" weight="10.0" />
            <Reweight scoretype="hbond_lr_bb" weight="10.0" />
            <Reweight scoretype="hbond_bb_sc" weight="5.0" />
            <Reweight scoretype="hbond_sc" weight="3.0" />
            <Reweight scoretype="fa_elec" weight="2.0" />
        </ScoreFunction>

        # A variation on the previous scoring function that has the amino acid composition
        # penalty activated as well:
        <ScoreFunction name="r15_highhbond_aacomp_cst" weights="ref2015_cst.wts" >
            <Reweight scoretype="chainbreak" weight="40.0" />
            <Reweight scoretype="aspartimide_penalty" weight="1.0" />
            <Reweight scoretype="hbond_sr_bb" weight="10.0" />
            <Reweight scoretype="hbond_lr_bb" weight="10.0" />
            <Reweight scoretype="hbond_bb_sc" weight="5.0" />
            <Reweight scoretype="hbond_sc" weight="3.0" />
            <Reweight scoretype="fa_elec" weight="2.0" />
            <Reweight scoretype="aa_composition" weight="1.0" />
        </ScoreFunction>
    </SCOREFXNS>

    # The RESIDUE_SELECTORS section establishes residue selectors, which are rules for
    # selecting a subset of a structure for other modules to operate upon.
    <RESIDUE_SELECTORS>
        # These residue selectors select the peptide, the original stub residues, or just the
        # D-cysteine residue in the stub:
        <Index name="select_peptide" resnums="234-241" />
        <Index name="select_stub" resnums="237-238" />
        <Index name="select_stub_dcys" resnums="237" />

        # Select the start and end residues of the peptide:
        <Index name="select_pep_start" resnums="234" />
        <Index name="select_pep_end" resnums="241" />

        # This selector inverts the peptide selection, selecting the target protein:
        <Not name="select_target" selector="select_peptide" />

        # These selectors select residues based on backbone conformation, selecting residues
        # with negative and positive backbone phi angles, respectively:
        <Phi name="select_neg_phi" select_positive_phi="false" />
        <Not name="select_pos_phi" selector="select_neg_phi" />

        # Since the stub residues of the peptide are not designed, this selector selects only
        # the designable subset of positions:
        <Index name="select_design_positions" resnums="234-236,239-241" />

        # This selector selects positions within the peptide with positive phi values, which
        # will only be permitted to assume D-amino acid identities:
        <And name="select_D_positions" selectors="select_design_positions,select_pos_phi" />
```

```
    # This selector selects positions within the peptide with negative phi values, which
    # will only be permitted to assume L-amino acid identities:
    <And name="select_L_positions" selectors="select_design_positions,select_neg_phi" />

    # This selector selects positions within the peptide in backbone bin “A”
    # (corresponding to the left-handed alpha helical region of Ramachandran space:
    <Bin name="select_L_alpha" bin_params_file="ABBA.bin_params" bin="A" />

    # This selector selects positions within the peptide in backbone bin “X”, a.k.a.
    # “Aprime” (corresponding to the right-handed alpha helical region of Ramachandran
    # space:
    <Bin name="select_D_alpha" bin_params_file="ABBA.bin_params" bin="Aprime" />

    # This selector selects positions within the peptide in backbone bin “B”
    # (corresponding to the left-handed beta strand region of Ramachandran space:
    <Bin name="select_L_beta" bin_params_file="ABBA.bin_params" bin="B" />

    # This selector selects positions within the peptide in backbone bin “Y”, a.k.a.
    # “Bprime” (corresponding to the right-handed beta strand region of Ramachandran
    # space:
    <Bin name="select_D_beta" bin_params_file="ABBA.bin_params" bin="Bprime" />

    # This selector selects positions on the target near the peptide, which will be
    # allowed to repack during peptide sequence design:
    <Neighborhood name="select_interface" resnums="234-241" distance="8.0" />

    # This selector selects positions far from the interface:
    <Not name="not_interface" selector="select_interface" />

    # This selector selects positions far from the interface which are part of the target,
    # which will be fixed during design:
    <And name="select_target_far_from_interface"
        selectors="select_interface,select_target"
    />

    # This selector selects buried residues:
    <Layer name="select_core" select_core="true" select_boundary="false"
        select_surface="false"
    />

    # This selector selects buried positions that are also designable:
    <And name="select_hydrophobic_positions"
        selectors="select_design_positions,select_core"
    />

    # This selector selects exposed residues:
    <Not name="select_nonhydrophobic_positions" selector="select_hydrophobic_positions" />

    # This selector selects exposed positions that are also designable:
    <And name="select_nonhydrophobic_design_positions"
        selectors="select_nonhydrophobic_positions,select_design_positions"
    />

    # The following four selectors select every combination of (negative phi or positive
    # phi positions) and (buried or exposed positions).
    <And name="select_L_hydrophobic_positions"
        selectors="select_hydrophobic_positions,select_L_positions" />
    <And name="select_D_hydrophobic_positions"
        selectors="select_hydrophobic_positions,select_D_positions" />
    <And name="select_L_nonhydrophobic_positions"
        selectors="select_nonhydrophobic_design_positions,select_L_positions" />
    <And name="select_D_nonhydrophobic_positions"
        selectors="select_nonhydrophobic_design_positions,select_D_positions" />
</RESIDUE_SELECTORS>
```

```
    # The PACKER_PALETTES section defines palettes of amino acid types that will be used for
    # design by default.  Our default palette should include D-amino acids.
    <PACKER_PALETTES>
        # The CustomBaseTypePackerPalette defines the canonical amino acids as types that will
        # be used by default, and allows the user to add additional types.  We will add the D-
        # amino acid mirror images of the canonical amino acids.
        # (Note that the additional_residue_types line must be on a single line.)
        <CustomBaseTypePackerPalette name="design_palette"
            additional_residue_types="DALA,DASP,DGLU,DHIS,DILE,DLYS,DLEU,DMET,DASN,
            DPRO,DGLN,DARG,DSER,DTHR,DVAL,DTRP,DTYR"
        />
    </PACKER_PALETTES>

    # The TASKOPERATIONS section defines task operations, which are rules for controlling the
    # Rosetta packer.  The Rosetta packer optimizes side chain rotamers and carries out
sequence
    # design, so task operations define design problems.
    <TASKOPERATIONS>
        # Include the input rotamer, even if it is shifted from a rotamer well, in the set of
        # rotamers allowed at a position:
        <IncludeCurrent name="use_input_rotamer" />

        # Sample finer levels of discretization of rotamers:
        <ExtraRotamersGeneric name="extrarot" ex1="1" ex2="1" ex3="0" ex4="0"
            extrachi_cutoff="5"
        />

        # Prevent repacking of the NDM-1 target protein:
        <OperateOnResidueSubset name="no_repack_target" selector="select_target" >
            <PreventRepackingRLT />
        </OperateOnResidueSubset>

        # Prevent design of the NDM-1 target protein:
        <OperateOnResidueSubset name="no_design_target" selector="select_target" >
            <RestrictToRepackingRLT />
        </OperateOnResidueSubset>

        # Prevent repacking of the D-cysteine, L-proline stub within the peptide:
        <OperateOnResidueSubset name="no_repack_stub_dcys" selector="select_stub_dcys" >
            <PreventRepackingRLT />
        </OperateOnResidueSubset>

        # Prevent design of the D-cysteine, L-proline stub within the peptide:
        <OperateOnResidueSubset name="no_design_stub" selector="select_stub" >
            <RestrictToRepackingRLT />
        </OperateOnResidueSubset>

        # Don't design with cysteine or glycine.
        <ProhibitSpecifiedBaseResidueTypes name="no_cys_gly" base_types="CYS,DCYS,GLY"
            selector="select_design_positions"
        />

        # Apply restrictions on allowed residues at positive phi positions that are exposed.
        <ProhibitResidueProperties name="D_design" properties="L_AA"
            selector="select_D_nonhydrophobic_positions"
        />

        # Apply restrictions on allowed residues at positive phi positions that are buried:
        <RestrictToSpecifiedBaseResidueTypes name="D_hydrophobic_design"
            base_types="DPHE,DILE,DLEU,DMET,DPRO,DVAL,DTRP,DTYR"
            selector="select_D_hydrophobic_positions"
        />

        # Apply restrictions on allowed residues at negative phi positions that are exposed:
        <ProhibitResidueProperties name="L_design" properties="D_AA"
            selector="select_L_nonhydrophobic_positions"
```

```
        />

        # Apply restrictions on allowed residues at negative phi positions that are buried:
        <RestrictToSpecifiedBaseResidueTypes name="L_hydrophobic_design"
            base_types="PHE,ILE,LEU,MET,PRO,VAL,TRP,TYR"
            selector="select_L_hydrophobic_positions"
        />

        # Do not allow repacking of NDM-1 residues far from the peptide:
        <OperateOnResidueSubset name="no_repack_target_far_from_interface"
            selector="select_target_far_from_interface"
        >
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
    </TASKOPERATIONS>

    # The FILTERS section defines Rosetta filters, which are Rosetta algorithms to analyze a
    # structure, measure properties of that structure, and accept or reject that structure
    # based on the measured properties:
    <FILTERS>
        # These filters measure shape complementarity of the peptide to the target:
        <ShapeComplementarity name="shape1" min_sc="0.63" min_interface="150" jump="3" />
        <ShapeComplementarity name="shape2" min_sc="0.66" min_interface="150" jump="3" />
        <ShapeComplementarity name="shape3" min_sc="0.66" min_interface="150" jump="3" />

        # This filter is used in early stages to discard sampled peptide conformations in
        # which molecular geometry clashes egregiously with the target:
        <ScoreType name="low_stringency_clash" scorefxn="r15" score_type="fa_rep"
            threshold="400"
        />

        # Rosetta's pairwise-decomposable scoring function is unable to detect cases in which
        # three or more hydrogen bond donors all donate to the same acceptor.  This filter
        # detects and eliminates this pathology:
        <OversaturatedHbondAcceptorFilter name="oversat" scorefxn="r15"
            max_allowed_oversaturated="0" consider_mainchain_only="false"
        />

        # Count internal backbone-backbone hydrogen bonds, excluding hydrogen bonds between
        # adjacent residues, and discard structures with fewer than 3 internal hydrogen bonds.
        # Note that this filter, which is simpler to configure and faster to evaluate, was not
        # available in earlier releases of Rosetta:
        <PeptideInternalHbondsFilter name="total_hbonds" hbond_cutoff="3"
            exclusion_distance="1" residue_selector="select_peptide"
        />
        <PeptideInternalHbondsFilter name="total_hbonds_2" hbond_cutoff="3"
            exclusion_distance="1" residue_selector="select_peptide"
        />

        # This filter measures the energy, biased towards hydrogen bonds and electrostatic
        # terms, following a Monte Carlo move:
        <ScoreType name="mc_score" scorefxn="r15_highhbond_aacomp_cst"
            score_type="total_score" threshold="999999"
        />

        # This filter is used during the Monte Carlo simulation to compute the value passed to
        # the Metropolis evaluator to determine whether moves pass or fail:
        <CombinedValue name="mc_filter" threshold="99999">
            <Add filter_name="mc_score" factor="1.0" />
            <Add filter_name="shape2" factor="-100.0" />
        </CombinedValue>
    </FILTERS>

    # The MOVERS section sets up movers, which operate on a structure to alter it in some way.
    # Some sample conformations, others design sequences, others carry out energy
    # minimization, etc.
```

```
<MOVERS>
    # The PeptideStubMover appends or prepends residues to an existing structure.
    <PeptideStubMover name="extend" >
        <Prepend anchor_rsd="234" resname="GLY" />
        <Prepend anchor_rsd="234" resname="GLY" />
        <Prepend anchor_rsd="234" resname="GLY" />
        <Append anchor_rsd="238" resname="GLY" />
        <Append anchor_rsd="239" resname="GLY" />
        <Append anchor_rsd="240" resname="GLY" />
    </PeptideStubMover>

    # The AtomTree mover sets up the fold tree, which defines the kinematic relationships
    # between different parts of a structure:
    <AtomTree name="foldtree1" fold_tree_file="inputs/foldtree1.txt" />

    # This SetTorsion mover is used to initialize the torsions for the dihedral angles at
    # the start and end of the D-cysteine-L-proline stub.  It sets these to ideal values
    # and then adds a small random perturbation.  It also ensures that all omega angles
    # are 180 degrees.
    <SetTorsion name="initialize_tors">
        <Torsion residue="234" torsion_name="omega" angle="180.0" />
        <Torsion residue="235" torsion_name="omega" angle="180.0" />
        <Torsion residue="236" torsion_name="omega" angle="180.0" />
        <Torsion residue="237" torsion_name="phi"   angle="60.0" />
        <Torsion residue="237" torsion_name="phi"   angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="30.0"
        />
        <Torsion residue="237" torsion_name="psi"   angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="5.0"
        />
        <Torsion residue="237" torsion_name="omega" angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="5.0"
        />
        <Torsion residue="238" torsion_name="phi"   angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="5.0"
        />
        <Torsion residue="238" torsion_name="psi"   angle="-10.0" />
        <Torsion residue="238" torsion_name="psi"   angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="30.0"
        />
        <Torsion residue="238" torsion_name="omega" angle="180.0" />
        <Torsion residue="239" torsion_name="omega" angle="180.0" />
        <Torsion residue="240" torsion_name="omega" angle="180.0" />
    </SetTorsion>

    # This SetTorsion mover adds a smaller random perturbation to all backbone degrees of
    # freedom of the stub:
    <SetTorsion name="perturb_tors">
        <Torsion residue="237" torsion_name="phi"   angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="5.0"
        />
        <Torsion residue="237" torsion_name="psi"   angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="1.0"
        />
        <Torsion residue="237" torsion_name="omega" angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="1.0"
        />
        <Torsion residue="238" torsion_name="phi"   angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="1.0"
        />
        <Torsion residue="238" torsion_name="psi"   angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="5.0"
        />
        <Torsion residue="238" torsion_name="omega" angle="perturb"
            perturbation_type="gaussian" perturbation_magnitude="1.0"
        />
```

```
        </SetTorsion>

        # This mover adds a chemical bond between the first and last residues of the peptide.
        # It has the side-effect of correcting the placement of oxygen and hydrogen atoms
        # flanking the amide bond, and so is used repeatedly to update these placements.
        # Note that this is less likely in more recent versions of Rosetta, since the
        # cutpoint variant types have been updated to ensure that hydrogen and oxygen
        # placement remains reasonable at cutpoints during minimization:
        <DeclareBond name="connect_termini" atom1="N" res1="234" atom2="C" res2="241"
            add_termini="true"
        />

        # Add cutpoint variants to the cyclization point in the peptide.  The cutpoint variant
        # works with the chainbreak scoring term to ensure that good amide bond geometry is
        # maintained during energy minimization:
        <ModifyVariantType name="add_cutpoint_upper_pep" add_type="CUTPOINT_UPPER"
            residue_selector="select_pep_start"
        />
        <ModifyVariantType name="add_cutpoint_lower_pep" add_type="CUTPOINT_LOWER"
            residue_selector="select_pep_end"
        />

        # These movers add amino acid composition constraints to the whole peptide, to
        # negative-phi positions in the alpha-helix region of Ramachandran space, to
        # negative-phi positions in the beta-sheet region of Ramachandran space, to
        # positive-phi positions in the right-handed alpha-helix region of Ramachandran
        # space, and to positive-phi positions in the right-handed beta-sheet region of
        # Ramachandran space, respectively:
        <AddCompositionConstraintMover name="global_comp"
            filename="inputs/global_preferences.comp" selector="select_design_positions" />
        <AddCompositionConstraintMover name="L_alpha_comp"
            filename="inputs/L_alpha_preferences.comp" selector="select_L_alpha" />
        <AddCompositionConstraintMover name="D_alpha_comp"
            filename="inputs/D_alpha_preferences.comp" selector="select_D_alpha" />
        <AddCompositionConstraintMover name="L_beta_comp"
            filename="inputs/L_beta_preferences.comp" selector="select_L_beta" />
        <AddCompositionConstraintMover name="D_beta_comp"
            filename="inputs/D_beta_preferences.comp" selector="select_D_beta" />

        # The PackRotamersMover optimizes side chain rotamers, permitting structure
        # refinement or design.  Here, it is used to design the peptide and to refine
        # side chain conformations on the target.  Note that the task_operations line must
        # be on a single line:
        <PackRotamersMover name="softdesign" scorefxn="r15_soft"
            packer_palette="design_palette"
            task_operations="use_input_rotamer,no_design_target,no_repack_target,
            no_design_stub,no_repack_stub_dcys,no_cys_gly,D_design,L_design,
            D_hydrophobic_design,L_hydrophobic_design"
        />

        # The MinMover relaxes a structure through gradient-descent minimization.  This one
        # is configured to minimize only side chains of the peptide.  Note that more recent
        # versions of Rosetta permit more robust configuration of the minimizer through
        # move map factories:
        <MinMover name="min1" scorefxn="r15_cst" type="dfpmin" tolerance="0.001" bb="0"
            chi="0"
        >
            <MoveMap name="min1_mm" >
                <Jump number="1" setting="0" />
                <Jump number="2" setting="0" />
                <Jump number="3" setting="0" />
                <Jump number="4" setting="0" />
                <Jump number="5" setting="0" />
                <Jump number="6" setting="0" />
                <Jump number="7" setting="0" />
                <Jump number="8" setting="0" />
```

```
            <Jump number="9" setting="0" />
            <Span begin="1" end="999" chi="0" bb="0" />
            <Span begin="234" end="241" chi="1" bb="0" />
        </MoveMap>
    </MinMover>

    # This MinMover permits both side hain and backbone minimization of the peptide, but
    # keeps the rest of the structure rigid:
    <MinMover name="min2" scorefxn="r15_cst" type="dfpmin" tolerance="0.001" bb="0"
        chi="0"
    >
        <MoveMap name="min2_mm" >
            <Jump number="1" setting="0" />
            <Jump number="2" setting="0" />
            <Jump number="3" setting="0" />
            <Jump number="4" setting="0" />
            <Jump number="5" setting="0" />
            <Jump number="6" setting="0" />
            <Jump number="7" setting="0" />
            <Jump number="8" setting="0" />
            <Jump number="9" setting="0" />
            <Span begin="1" end="999" chi="0" bb="0" />
            <Span begin="234" end="241" chi="1" bb="1" />
        </MoveMap>
    </MinMover>

    # FastDesign performs alternating rounds of packing and minimization while ramping the
    # repulsive term in the scoring function from an initial low value.  This FastDesign
    # instance is configured to design the peptide and repack the target interface, and to
    # allow peptide side chains and backbone and target side chains to move during energy
    # minimization.  Note that the task_operations line must be on a single line:
    <FastDesign name="fdes" repeats="3" scorefxn="r15_highhbond_aacomp_cst"
        min_type="dfpmin" packer_palette="design_palette"
        task_operations="use_input_rotamer,no_repack_target_far_from_interface,
        no_design_target,no_design_stub,no_repack_stub_dcys,no_cys_gly,D_design,L_design,
        L_hydrophobic_design,D_hydrophobic_design"
    >
        <MoveMap name="fdes_mm" >
            <Jump number="1" setting="0" />
            <Jump number="2" setting="0" />
            <Jump number="3" setting="0" />
            <Jump number="4" setting="0" />
            <Jump number="5" setting="0" />
            <Jump number="6" setting="0" />
            <Jump number="7" setting="0" />
            <Jump number="8" setting="0" />
            <Jump number="9" setting="0" />
            <Span begin="1" end="999" chi="1" bb="0" />
            <Span begin="234" end="241" chi="1" bb="1" />
        </MoveMap>
    </FastDesign>

    # A ParsedProtocol encapsulates many previously-defined movers and/or filters into a
    # single mover.  This instance defines a series of steps carried out for every
    # solution found by the initial generalized kinematic closure attempts to close the
    # peptide. These steps include filtering based on total hydrogen bonds, presence of
    # oversaturated acceptors, and clashes, design and minimization, additional filtering,
    # and a more expensive FastDesign round if minimal shape complementarity requirements
    # are met after the low-cost steps carried out thus far.  The pattern is to go from
    # inexpensive to expensive computations, deciding whether to discard the attempt or to
    # continue after each step:
    <ParsedProtocol name="genkic_steps">
        <Add filter="oversat" />
        <Add filter="total_hbonds" />
        <Add filter="low_stringency_clash" />
        <Add mover="softdesign" />
```

```
        <Add mover="min1" />
        <Add mover="connect_termini" />
        <Add filter="oversat" />
        <Add mover="min2" />
        <Add mover="connect_termini" />
        <Add filter="oversat" />
        <Add filter="total_hbonds_2" />
        <Add filter="shape1" />
        <Add mover="fdes" />
        <Add mover="connect_termini" />
        <Add filter="oversat" />
        <Add filter="shape2" />
    </ParsedProtocol>

    # Generalized kinematic closure (GeneralizedKIC) allows efficient sampling of the
    # closed conformations of a chain of atoms, with rapid solution of a series of
    # equations to determine values of certain degrees of freedom in order to keep the
    # chain closed. This instance is configured to perform the initial closure of the
    # peptide macrocycle. For each closed conformation sampled, the steps listed in the
    # previous ParsedProtocol mover are carried out, and the conformation iis accepted
    # if and only if all steps pass:
    <GeneralizedKIC name="genkic" closure_attempts="100"
        pre_selection_mover="genkic_steps"
        stop_when_n_solutions_found="1" selector="lowest_energy_selector"
        selector_scorefunction="r15_highhbond_cst"
    >
        <AddResidue res_index="239" />
        <AddResidue res_index="240" />
        <AddResidue res_index="241" />
        <AddResidue res_index="234" />
        <AddResidue res_index="235" />
        <AddResidue res_index="236" />
        <SetPivots res1="239" res2="234" res3="236" atom1="CA" atom2="CA" atom3="CA" />
        <CloseBond atom1="C" res1="241" atom2="N" res2="234" torsion="180"
            bondlength="1.328685" angle1="116.2" angle2="121.7"
        />
        <AddPerturber effect="randomize_alpha_backbone_by_rama" >
            <AddResidue index="234" />
            <AddResidue index="235" />
            <AddResidue index="236" />
            <AddResidue index="239" />
            <AddResidue index="240" />
            <AddResidue index="241" />
        </AddPerturber>
    </GeneralizedKIC>

    # A second GeneralizedKIC mover is used to perturb the closed peptide macrocycle
    # during the Monte Carlo search.  For each perturbation, these steps (checking for
    # oversaturated hbond acceptors, counting total hbonds) are carried out:
    <ParsedProtocol name="genkic_perturb_steps">
        <Add mover="connect_termini" />
        <Add filter="oversat" />
        <Add filter="total_hbonds_2" />
    </ParsedProtocol>

    # This is the second GeneralizedKIC mover.  This one is configured to add a small,
    # random perturbation to an already-closed peptide macrocycle, ensuring that the
    # macrocycle remains closed after the perturbation.  It is used in the context of a
    # Monte Carlo search of local conformations:
    <GeneralizedKIC name="genkic_perturb" closure_attempts="5"
        pre_selection_mover="genkic_perturb_steps" stop_when_n_solutions_found="1"
        selector="lowest_rmsd_selector" selector_scorefunction="r15_highhbond_cst"
    >
        <AddResidue res_index="239" />
        <AddResidue res_index="240" />
        <AddResidue res_index="241" />
```

```
            <AddResidue res_index="234" />
            <AddResidue res_index="235" />
            <AddResidue res_index="236" />
            <SetPivots res1="239" res2="234" res3="236" atom1="CA" atom2="CA" atom3="CA" />
            <CloseBond atom1="C" res1="241" atom2="N" res2="234" torsion="180"
                bondlength="1.328685" angle1="116.2" angle2="121.7" />
            <AddPerturber effect="perturb_dihedral" >
                <AddAtoms res1="239" atom1="N"  res2="239" atom2="CA" />
                <AddAtoms res1="239" atom1="CA" res2="239" atom2="C"  />
                <AddAtoms res1="240" atom1="N"  res2="240" atom2="CA" />
                <AddAtoms res1="240" atom1="CA" res2="240" atom2="C"  />
                <AddAtoms res1="241" atom1="N"  res2="241" atom2="CA" />
                <AddAtoms res1="241" atom1="CA" res2="241" atom2="C"  />
                <AddAtoms res1="234" atom1="N"  res2="234" atom2="CA" />
                <AddAtoms res1="234" atom1="CA" res2="234" atom2="C"  />
                <AddAtoms res1="235" atom1="N"  res2="235" atom2="CA" />
                <AddAtoms res1="235" atom1="CA" res2="235" atom2="C"  />
                <AddAtoms res1="236" atom1="N"  res2="236" atom2="CA" />
                <AddAtoms res1="236" atom1="CA" res2="236" atom2="C"  />
                <AddValue value="2.5" />
            </AddPerturber>
        </GeneralizedKIC>

        # These movers are used only for debugging, not in production runs:
        <PDBTrajectoryRecorder name="record_traj" stride="1" filename="traj.pdb"
            cumulate_jobs="0" cumulate_replicas="0" />
        <PDBTrajectoryRecorder name="record_traj_accepted" stride="1" filename="accepted.pdb"
            cumulate_jobs="0" cumulate_replicas="0" />

        # This is the series of steps performed as the move in the Monte Carlo search.  First,
        # the stub and the macrocycle are perturbed slightly.  Next, a quick round of design
        # is carried out, followed by side chain minimization.  Then, filters are applied.
        # Then, the backbone is minimized, followed by more filtering.  If all of this
        # passes, the move is accepted or rejected by the Metropolis criterion:
        <ParsedProtocol name="mc_steps">
            Add mover=record_traj_accepted /> #COMMENTED OUT FOR PRODUCTION RUNS.
            <Add mover="perturb_tors" />
            <Add mover="genkic_perturb" />
            <Add mover="softdesign" />
            <Add mover="min1" />
            <Add mover="connect_termini" />
            Add mover=record_traj /> #COMMENTED OUT FOR PRODUCTION RUNS.
            <Add filter="oversat" />
            <Add mover="min2" />
            <Add mover="connect_termini" />
            <Add filter="oversat" />
            <Add filter="total_hbonds_2" />
        </ParsedProtocol>

        # This mover actually carries out the Monte Carlo search of local conformation space,
        # executing the series of steps in the previous ParsedProtocol as the moves in the
        # search:
        <GenericMonteCarlo name="mc_search" mover_name="mc_steps" filter_name="mc_filter"
            trials="500" temperature="0.5"
        />

        # FsstRelax performs alternating rounds of side chain packing and minimization,
        # keeping sequence fixed but ramping repulsive terms in the scoring function.  It
        # is used for final structural refinement:
        <FastRelax name="final_frlx" repeats="3" scorefxn="r15_cst" min_type="dfpmin"
            task_operations="use_input_rotamer,no_repack_target_far_from_interface"
        >
            <MoveMap name="final_frlx_mm" >
                <Jump number="1" setting="0" />
                <Jump number="2" setting="0" />
                <Jump number="3" setting="0" />
```

```
                <Jump number="4" setting="0" />
                <Jump number="5" setting="0" />
                <Jump number="6" setting="0" />
                <Jump number="7" setting="0" />
                <Jump number="8" setting="0" />
                <Jump number="9" setting="0" />
                <Span begin="1" end="999" chi="1" bb="0" />
                <Span begin="234" end="241" chi="1" bb="1" />
            </MoveMap>
        </FastRelax>
    </MOVERS>

    # The PROTOCOLS section strings together previously-defined movers and filters to
    # construct an overall protocol.  Our overall protocol is to initialize the structure
    # and add cutpoint variant types, perform an initial closure of the macrocycle in which
    # an initial sequence is designed, then carry out a Monte Carlo search of the local
    # conformational space, redesigning the macrocycle sequence at each step.  Final
    # relaxation and filtering to measure shape complementarity complete the protocol.
    <PROTOCOLS>
        <Add mover="extend" />
        <Add mover="foldtree1" />
        <Add mover="initialize_tors" />
        <Add mover="connect_termini" />
        <Add mover="add_cutpoint_upper_pep" />
        <Add mover="add_cutpoint_lower_pep" />
        <Add mover="global_comp" />
        <Add mover="L_alpha_comp" />
        <Add mover="D_alpha_comp" />
        <Add mover="L_beta_comp" />
        <Add mover="D_beta_comp" />
        <Add mover="genkic" />
        <Add mover="mc_search" />
        <Add mover="fdes" />
        <Add mover="final_frlx" />
        <Add mover="connect_termini" />
        <Add filter="oversat" />
        <Add filter="shape3" />
    </PROTOCOLS>

    # The OUTPUT section defines the scoring function that will be used to produce the final
    # score for the structure.
    <OUTPUT scorefxn="r15" />
</ROSETTASCRIPTS>
```

Listing 2.1.1.2: Modernized RosettaScripts XML for NDM1i-1 peptides (**xml/NDM1i_1_design.xml**). This script has been updated to work with 2020 releases of Rosetta. For the original design script, see **Section 2.2.2**.

This script requires certain additional files as input. The first is a fold tree definition, provided as **inputs/foldtree1.txt**. This defines a fold tree (kinematic relationships within the structure) that is rooted on residue 83 of the protein, progresses forward and backward through bonds to the first and last residues of the protein, has jumps (rigid-body transforms) to the two zinc atoms, and has a jump to the S_γ atom of the D-cysteine stub in the peptide. This file is shown below:

```
FOLD_TREE
EDGE 83 3 -1
EDGE 83 233 -1
EDGE 83 2 1
EDGE 85 1 2
JEDGE 1 237 3 ZN SG INTRA_RES_STUB
```

```
EDGE 237 234 -1
EDGE 237 241 -1
```

Listing 2.1.1.3: Fold tree definition file **inputs/foldtree1.txt**.

The next required inputs are five amino acid composition files, shown in the listings below. These establish desired amino acid compositions for the whole peptide (**inputs/global_preferences.comp**), and for D- and L-amino acid positions in the α-helical and β-strand regions of Ramachandran space (**inputs/D_alpha_preferences.comp**, **coinputsmp/L_alpha_preferences.comp**, **inputs/D_beta_preferences.comp**, and **inputs/L_beta_preferences.comp**).

```
# At least two hydrophobics:
PENALTY_DEFINITION
PROPERTIES HYDROPHOBIC
DELTA_START -2
DELTA_END 1
PENALTIES 25 10 0 0
ABSOLUTE 2
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

# At most one methionine:
PENALTY_DEFINITION
TYPE MET DME
DELTA_START -1
DELTA_END 1
ABSOLUTE 1
PENALTIES 0 0 25
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION

# Require L-glu or L-asp for on-bead cyclization (with the peptide tethered by a carboxyl-
# containing side chain):
PENALTY_DEFINITION
TYPE ASP GLU
DELTA_START -1
DELTA_END 1
PENALTIES 50 0 0
ABSOLUTE 1
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

# Prevent too many asp/d-asp:
PENALTY_DEFINITION
TYPE ASP DAS
DELTA_START -1
DELTA_END 1
ABSOLUTE 1
PENALTIES 0 2 5
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION

# At least one positive charge for mass spectrometry:
PENALTY_DEFINITION
TYPE LYS ARG DLY DAR
```

```
DELTA_START -1
DELTA_END 1
PENALTIES 50 0 0
ABSOLUTE 1
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

#Aim for 3 prolines:
PENALTY_DEFINITION
TYPE PRO DPR
DELTA_START -3
DELTA_END 1
PENALTIES 20 5 2 0 0
ABSOLUTE 3
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION
```

Listing 2.1.1.4: Global amino acid composition preferences file **inputs/global_preferences.comp**.

```
# Weakly encourage helix-favouring D-amino acids at at least 70% of positions in the
# positive-phi alpha-helix region of Ramachandran space.
PENALTY_DEFINITION
TYPE DAL DLE DAR
FRACT_DELTA_START -0.25
FRACT_DELTA_END 0.25
PENALTIES 3 0 0
FRACTION 0.7
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION
```

Listing 2.1.1.5: Amino acid composition preferences file **inputs/D_alpha_preferences.comp**, defining desired amino acid composition and positions in the positive-φ α-helix region of Ramachandran space.

```
# Weakly encourage helix-favouring L-amino acids at at least 70% of positions in the
negative-phi
# alpha-helix region of Ramachandran space.
PENALTY_DEFINITION
TYPE ALA LEU ARG
FRACT_DELTA_START -0.25
FRACT_DELTA_END 0.25
PENALTIES 3 0 0
FRACTION 0.7
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION
```

Listing 2.1.1.6: Amino acid composition preferences file **inputs/L_alpha_preferences.comp**, defining desired amino acid composition and positions in the negative-φ α-helix region of Ramachandran space.

```
# Weakly encourage D-threonine and D-valine at at least 70% of positions in the positive-phi
# beta-strand region of Ramachandran space.
PENALTY_DEFINITION
TYPE DTH DVA
FRACT_DELTA_START -0.25
FRACT_DELTA_END 0.25
```

```
PENALTIES 3 0 0
FRACTION 0.7
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION
```

Listing 2.1.1.7: Amino acid composition preferences file **inputs/D_beta_preferences.comp**, defining desired amino acid composition and positions in the positive-φ β-strand region of Ramachandran space.

```
# Weakly encourage L-threonine and L-valine at at least 70% of positions in the negative-phi
# beta-strand region of Ramachandran space.
PENALTY_DEFINITION
TYPE THR VAL
FRACT_DELTA_START -0.25
FRACT_DELTA_END 0.25
PENALTIES 3 0 0
FRACTION 0.7
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION
```

Listing 2.1.1.8: Amino acid composition preferences file **inputs/L_beta_preferences.comp**, defining desired amino acid composition and positions in the negative-φ β-strand region of Ramachandran space.

To run this script, the XML file shown in **Listing 2.1.1.2** should be saved as **xml/NDM1i_1_design.xml** (where **xml/** is a sub-directory of the working directory). Support files shown above should be placed in the **inputs/** sub-directory of the working directory. An additional file, **inputs/rosetta.flags**, should also be added to the **inputs/** directory. Its contents are shown below:

```
# The number of designs to generate.  On most computing clusters, this can be set to a high
# number, and jobs can be permitted to end when their time limit is reached:
-nstruct 1000

# Parameters adjusting the automatic setup of bonds to metal atoms:
-metals_distance_constraint_multiplier 5.0
-metals_angle_constraint_multiplier 5.0
-auto_setup_metals

# The input PDB file (interpreted as an all-atom representation):
-in:file:s inputs/4EXS_Dcys_Lpro.pdb
-in:file:fullatom

# Since glycine Ramachandran tables used for scoring are based on statistics from the Protein
# Data Bank, they are biased towards glycine residues in positive-phi regions of Ramachandran
# space.  When designing with D-amino acids, the following correction should always be
# applied:
-symmetric_gly_tables true

# Bonds between all covalently-connected atoms will be written to the output PDB file, to aid
# visualization in PyMol:
-write_all_connect_info

# The input RosettaScripts XML file:
-parser:protocol xml/NDM1i_1_design.xml

# Since some jobs are expected to fail due to filters that don't pass, we do not want the
# executable to exit with failure status if there are some failures:
```

```
-jd2:failed_job_exception false

# The following line should be uncommented for production runs, to prevent writing of a large
# output log:
#-mute all
```

Listing 2.1.1.9: Configuration file **inputs/rosetta.flags**, specifying command-line options for the **rosetta_scripts** application.

Now the RosettaScripts application can be run using the following command. Here, **<path_to_Rosetta>** should be replaced with the path to the user's **Rosetta/** directory. Additionally, the **.default.linuxgccrelease** portion of the executable file name should be replaced based on the user's build, operating system, and compiler. For example, if using the multi-threaded build on MacOS, the executable name becomes **rosetta_scripts.cxx11thread.macosclangrelease**:

```
<path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease
       \ @inputs/rosetta.flags
```

Listing 2.1.1.10: Command-line command to run the RosettaScripts application with the XML file shown in **Listing 2.1.1.2**.

2.1.2 Design protocol for NDM1i-3 peptides

The input for this round of design was the X-ray crystal structure of peptide NDM1i-1G bound to NDM-1, with residue L-Glu8 deleted; we did this to experiment with extensions of the peptide (9mers and 10mers, which turned out to be non-productive). This script adds back the missing residue, reconnects the termini, and re-closes the macrocycle using generalized kinematic closure, randomizing positions farthest from the D-Cys-L-Pro stub and allowing only small perturbations of positions close to the stub. It then carries out design using the new design-centric guidance terms described in **Section 1.3** to maximize shape complementarity (by minimizing voids in the interface), ensure a net positive charge, ensure satisfaction of buried hydrogen bond donors and acceptors, and promote hydrogen bond networks. Small motions of the hinge loop are allowed during energy-minimization. Since this script is intended to be quite conservative in how far it explores from the NDM1i-1G crystal structure, position 7 is manually mutated to hydroxyproline (based on the observation that this creates the opportunity for favourable hydrogen bonding interactions) and is not permitted to design.

In lieu of extensive backbone sampling, this round of design sought to use a much-expanded palette of non-canonical amino acid building blocks. **Table S1** shows the non-canonical achiral and L-amino acids considered during design at positions with negative backbone ϕ dihedral angles during NDM1i-3 or NDM1i-4 design. These residue types are described in greater detail in Renfrew *et al*. (2012) (7). The D-amino acid equivalents of the L-amino acids were considered at positions with positive ϕ values.

Rosetta 3-letter abbreviation	Name

A12	2,4-dimethyl-L-phenylalanine
A34	2-aminomethyl-L-phenylalanine
A43*	2-hydroxy-L-phenylalanine
A68	3-aminomethyl-L-phenylalanine
A78*	3-hydroxy-L-phenylalanine
A80*	L-DOPA (3-hydroxy-L-tyrosine)
A91	4,5-dehydro-L-leucine
A94	4-aminomethyl-L-phenylalanine
AIB†	2-aminoisobutyric acid
BB8*	3-phenyl-L-serine
B12*	4-carboxy-L-phenylalanine
B30	4-phenyl-L-phenylalanine
B36*	5-hydroxy-L-tryptophan
B44	9-anthryl-L-alanine
B67	β-(1-naphthyl)-L-alanine
B74	β-(2-naphthyl)-L-alanine
B96	β,β-diphenyl-L-alanine
C27	homophenyl-L-alanine
DAB	L-2,4-diaminobutyric acid
DPP	L-2,3-diaminopropionic acid
HYP‡	(4R)-4-hydroxy-L-proline
ORN	L-ornithine
NLU	L-norleucine
NVL	L-norvaline

Table S1: Non-canonical L-amino acid types considered during NDM1i-3 or NDM1i-4 peptide design process. †AIB is achiral, while all others listed are L-amino acids. ‡Hydroxyproline was only considered at position 7. *Only considered for NDM1i-4 design.

The script shown below has been modernized for more efficient calculation of internal hydrogen bonds and to use updated syntax for noncanonical design. For the original script used in 2018, see **Section 2.2.3**.

```
<ROSETTASCRIPTS>
        # This script is used to design NDM1i-3 peptides, which include exotic non-canonical
        # amino acids in the palette of allowed amino acid types.  It was tested with
        # the most recent version of Rosetta available (Git SHA1
        # 83d83e41283ef44a6ae652f9450c57dc680ec32d) built with GCC 9.3.0 on Centos Linux 7.
        #
        # For full scientific reproducibility, please refer to the legacy script provided
        # in Section 2.2.3.

    # Scoring functions are defined in this section:
    <SCOREFXNS>
        # The default enegy function for Rosetta:
        <ScoreFunction name="r15" weights="ref2015.wts" />

        # A variant of the default energy function with constraint terms activated,
        # and the chainbreak term (which preserves amide bond geometry during
        # energy-minimization) upweighted.
        <ScoreFunction name="r15_cst" weights="ref2015_cst.wts" >
            <Reweight scoretype="metalbinding_constraint" weight="1.0" />
            <Reweight scoretype="chainbreak" weight="25.0" />
        </ScoreFunction>

        # A version of the energy function used for design.  This activates a number
        # of design-centric guidance terms, including the voids_penalty term (which
        # penalizes holes or voids in cores or interfaces), the hbnet term (which
        # encourages hydrogen bond networks), the netcharge term (which allows control
        # over the net charge of the peptide), and the buried_unsatisfied_penalty term
        # (which adds a penalty for buried hydrogen bond donors and acceptors that
        # are not involved in a hydrogen bond):
        <ScoreFunction name="r15_cst_voids" weights="ref2015_cst.wts" >
            <Reweight scoretype="metalbinding_constraint" weight="1.0" />
            <Reweight scoretype="voids_penalty" weight="1.0" />
            <Reweight scoretype="hbnet" weight="1.0" />
            <Reweight scoretype="hbond_sr_bb" weight="10.0" />
            <Reweight scoretype="hbond_lr_bb" weight="10.0" />
            <Reweight scoretype="hbond_bb_sc" weight="5.0" />
            <Reweight scoretype="hbond_sc" weight="3.0" />
            <Reweight scoretype="netcharge" weight="1.0" />
            <Reweight scoretype="aa_composition" weight="1.0" />
            <Reweight scoretype="aspartimide_penalty" weight="1.0" />
            <Reweight scoretype="chainbreak" weight="25.0" />
            <Reweight scoretype="buried_unsatisfied_penalty" weight="0.5" />
        </ScoreFunction>

        # A variation of the above energy function, used for scoring the structure when
        # selecting top structures from the generalized kinematic closure step.
        <ScoreFunction name="r15_cst_voids_scoring" weights="ref2015_cst.wts" >
            <Reweight scoretype="metalbinding_constraint" weight="1.0" />
            <Reweight scoretype="voids_penalty" weight="1.0" />
            <Set voids_penalty_energy_disabled_except_during_packing="false" />
            <Reweight scoretype="hbnet" weight="1.0" />
            <Reweight scoretype="netcharge" weight="1.0" />
            <Reweight scoretype="aa_composition" weight="1.0" />
            <Reweight scoretype="aspartimide_penalty" weight="1.0" />
            <Reweight scoretype="chainbreak" weight="25.0" />
            <Reweight scoretype="buried_unsatisfied_penalty" weight="0.05" />
        </ScoreFunction>
    </SCOREFXNS>
```

```
# This section defines residue selectors, which select sets of residues in a structure
# based on user-defined rules.
<RESIDUE_SELECTORS>
    # Select the peptide and the hinge loop, all of which will be permitted to
    # move during energy minimization:
    <Index name="movable_backbone" resnums="24-32,235-242" />

    # Select the hinge loop and the portion of the peptide that will be
    # weakly constrained to prevent major motions during energy minimization:
    <Index name="constrained_bb" resnums="24-32,236-241" />

    # Select the peptide:
    <Index name="peptide" resnums="235-242" />

    # Select the target:
    <Not name="not_peptide" selector="peptide" />

    # Select the positions in the peptide for which the amino acid identity
    # is not fixed:
    <Index name="peptide_designable_residues" resnums="235-237,240,242" />

    # Select positions that can be D-amino acids (positive-phi region of
    # Ramachandran space):
    <Phi name="positive_phi" select_positive_phi="true" />

    # Select positions that can be L-amino acids (negative-phi region of
    # Ramachandran space):
    <Not name="negative_phi" selector="positive_phi" />

    # Select designable positions that can be D-amino acids:
    <And name="designable_positive_phi"
        selectors="peptide_designable_residues,positive_phi" />

    # Select designable positions that can be L-amino acids:
    <And name="designable_negative_phi"
        selectors="peptide_designable_residues,negative_phi" />

    # Select positions for which the amino acid identity is fixed:
    <Not name="non_designable_residues" selector="peptide_designable_residues" />

    # Select target positions that are close to the peptide in space.  (Also
    # selects the peptide):
    <Neighborhood name="near_peptide" distance="8" selector="peptide" />

    # Select target positions that are far from the peptide in space:
    <Not name="not_near_peptide" selector="near_peptide" />

    # Select target positions that are within 4.5 A of the peptide in space.
    # Also selects the peptide:
    <Neighborhood name="very_near_peptide" distance="4.5" selector="peptide" />

    # Select target positions that are more than 4.5 A from the peptide in
    # space:
    <Not name="not_very_near_peptide" selector="very_near_peptide" />

    # Select positions for which the backbone is movable, or which are close to the
    # peptide:
    <Or name="movable_sidechains" selectors="movable_backbone,peptide,near_peptide" />

    # Select positions which should receive upper cutpoint variant types (within the
    # macrocycle and the hinge loop:
    <Index name="upper_cutpoints" resnums="28,235" />

    # Select positions which should receive lower cutpoint variant types (within the
    # macrocycle and the hinge loop:
    <Index name="lower_cutpoints" resnums="27,242" />
```

```
    # Select positions that are buried:
    <Layer name="select_buried" select_core="true" select_boundary="false"
        select_surface="false" core_cutoff="2" surface_cutoff="0.1" />

    # Select positions that are exposed:
    <Not name="select_not_buried" selector="select_buried" />

    # Select buried positions within the peptide:
    <And name="select_buried_and_peptide" selectors="select_buried,peptide" />

    # Select exposed positions within the peptide:
    <And name="select_not_buried_and_peptide" selectors="select_not_buried,peptide" />
</RESIDUE_SELECTORS>

# Packer palettes define the set of amino acid types with which we are
# designing, in the absence of any task operations.  Task operations
# can then prune allowed types away in on a position-specific basis:
<PACKER_PALETTES>
   # This packer palette activates a set of non-canonical amino acids with exotic
   # side chains.  Note that the additional_residue_types line must be on a single line:
   <CustomBaseTypePackerPalette name="design_palette"
        additional_residue_types="AIB,A12,A91,NLU,NVL,A34,A68,A94,B30,B96,C27,B44,ORN,DPP,
        B67,B74,DA12,DA91,DNLU,DNVL,DA34,DA68,DA94,DB30,DB96,DC27,DB44,DB67,DB74,DORN,
        DDPP,DAB,DDAB"
    />
</PACKER_PALETTES>

# Task operations, which control the Rosetta packer, are defined here:
<TASKOPERATIONS>
    # Only allow residues within 8 A of the peptide to repack or be designed:
    <OperateOnResidueSubset name="only_repack_near_peptide" selector="not_near_peptide" >
        <PreventRepackingRLT />
    </OperateOnResidueSubset>

    # Only allow residues within 4.5 A of the peptide to repack or be designed:
    <OperateOnResidueSubset name="only_repack_very_near_peptide"
        selector="not_very_near_peptide"
    >
        <PreventRepackingRLT />
    </OperateOnResidueSubset>

    # Restrict non-peptide positions to repacking; design only the peptide:
    <OperateOnResidueSubset
        name="only_design_peptide" selector="non_designable_residues"
    >
        <RestrictToRepackingRLT />
    </OperateOnResidueSubset>

    # Include the input rotamer among those to be considered:
    <IncludeCurrent name="include_current" />

    # Set the allowed residue types (D-amino acids) at positive phi positions.
    # Note that this uses resfiles that have been updated for resfile syntax
    # compatible with newer versions of Rosetta:
    <ReadResfile name="resfile_positive_phi" selector="designable_positive_phi"
        filename="inputs/pos_phi.resfile" />

    # Set the allowed residue types (L-amino acids) at negative phi positions.
    # See note above about the current resfile syntax and Rosetta versions:
    <ReadResfile name="resfile_negative_phi" selector="designable_negative_phi"
        filename="inputs/neg_phi.resfile" />

    # Allow more finely-sampled rotamers:
    <ExtraRotamersGeneric name="ex1_ex2" ex1="true" ex2="true" extrachi_cutoff="6" />
</TASKOPERATIONS>
```

```
# Filters, which analyze structures and reject or accept them based on measured
# properties, are defined here:
<FILTERS>
    # Measure the shape complementarity of the peptide to the binding pocket:
    <ShapeComplementarity name="shape_complementarity" min_sc="0.5" min_interface="400"
        write_int_area="true" residue_selector1="peptide" residue_selector2="not_peptide"
    />

    # Measure the total volume of buried cavities.  (Only used for measurement):
    <CavityVolume name="cavity_volume" />

    # The following lines are used to count internal backbone hydrogen bonds
    # within the peptide.  Note that the PeptideInternalHbondsFilter is much
    # more efficient than the legacy approach that was used in 2018:
    <PeptideInternalHbondsFilter name="total_hbonds" hbond_cutoff="2"
        exclusion_distance="1" residue_selector="peptide"
    />

    # Require at least 2 hydrogen bonds to residue 237:
    <HbondsToResidue name="hbonds_237_2" partners="2" energy_cutoff="-0.25" bb_bb="true"
        backbone="true" sidechain="false" residue="237"
        scorefxn="r15" from_same_chain="true" from_other_chains="false" />

    # Prohibit more than two hydrogen bonds to each acceptor:
    <OversaturatedHbondAcceptorFilter name="no_oversat_acceptor1"
        max_allowed_oversaturated="0" consider_mainchain_only="true"
        acceptor_selector="peptide" scorefxn="r15" />
    <OversaturatedHbondAcceptorFilter name="no_oversat_acceptor2"
        max_allowed_oversaturated="0" consider_mainchain_only="false"
        acceptor_selector="peptide" scorefxn="r15" />
</FILTERS>

# Jump selectors are used to select rigid-body transforms in the fold tree:
<JUMP_SELECTORS>
    # Jump 1 is the rigid-body transform from the start of the NDM-1 target to the
    # end.  The NDM-1 chain is split into two segments to allow the hinge loop to
    # move with both ends of the loop rooted to unmovable segments of the body of
    # the protein.  To keep the body of the protein fixed, the rigid-body transform
    # from the first residue to the last must be fixed:
    <JumpIndex name="fixed_jumps" jump="1" />

    # All other rigid-body transforms (to the zinc atoms, to the peptide) are
    # permitted to move during energy minimization:
    <Not name="movable_jumps" selector="fixed_jumps" />
</JUMP_SELECTORS>

# Move map factories define which parts of a structure can move and which parts are
# fixed during energy minimization:
<MOVE_MAP_FACTORIES>
    # Keep the structure fixed except for side chains near the peptide or hinge
    # loop, backbone of the peptide and hing loop, and rigid-body transforms
    # other than the transform relating the first and last residues of the
    # (split) NDM-1 chain:
    <MoveMapFactory name="frlx_mm_factory" bb="false" chi="false" jumps="false">
        <Backbone residue_selector="movable_backbone" />
        <Chi residue_selector="movable_sidechains" />
        <Jumps jump_selector="movable_jumps" />
    </MoveMapFactory>
</MOVE_MAP_FACTORIES>

# Movers are defined here.  They operate on a structure to alter it in some way:
<MOVERS>
    # This mover adds back the residue that was deleted in order to open up
    # the peptide to experiment with 9mers and 10mers.  Here, we're simply
    # restoring the peptide length to an 8mer:
```

```
        <PeptideStubMover name="extend_pep" >
            <Insert resname="GLY" anchor_rsd="241" />
        </PeptideStubMover>

        # This converts position 235 to a glycine to allow unbiased conformational
        # sampling of conformations favoured by both L- and D-amino acids:
        <MutateResidue name="mut_to_gly_235" new_res="GLY" target="235" />

        # The DeclareBond mover is used both to indicate that there is a chemical
        # bond between the first and last residues (preventing the proximity of
        # the N- and C-termini from being interpreted as a clash), and also to
        # update the position of O and H atoms that are dependent on the N-to-C
        # amide bonds following energy minimization.  This is less essential
        # since cutpoints had been updated by 2018.
        <DeclareBond name="declare_bond" atom1="C" atom2="N" res1="242" res2="235" />

        # Since the hinge loop is permitted to move as well, this mover allows the
        # positions of O and H atoms at the cutpoint in the HL to be updated, as
        # a precaution:
        <DeclareBond name="update_loop_O_H" atom1="C" atom2="N" res1="27" res2="28" />

        # Manually introduce the proline->hydroxyproline mutation at position 7.  This
        # position in the crystal structure has the potential to make a favourable
        # hydrogen bonding interaction with the target.
        <MutateResidue name="add_hyp" target="241" new_res="HYP" />

        # Add cutpoint variants in the hinge loop and in the peptide macrocycle:
        <ModifyVariantType name="upper_cutpoints" add_type="CUTPOINT_UPPER"
            residue_selector="upper_cutpoints" />
        <ModifyVariantType name="lower_cutpoints" add_type="CUTPOINT_LOWER"
            residue_selector="lower_cutpoints" />
        <ParsedProtocol name="add_cutpoint_variants" >
            <Add mover="upper_cutpoints" />
            <Add mover="lower_cutpoints" />
        </ParsedProtocol>

        # Set up the fold tree:
        <AtomTree name="foldtree1" fold_tree_file="inputs/foldtree1.txt" />

        # Auto-detect metal-ligand interactions and set up suitable chemical bonds and
        # constraints to preserve metal geometry:
        <SetupMetalsMover name="setup_metals" metals_detection_LJ_multiplier="1.0" />

        # Add net charge constraints to the peptide macrocycle requiring a net
        # positive charge:
        <AddNetChargeConstraintMover name="require_net_pos_charge"
            filename="inputs/net_positive.charge" selector="peptide" />

        # Add amino acid composition constraints to the peptide macrocycle as a whole,
        # the buried parts of the macrocycle, and the exposed parts of the macrocycle:
        <AddCompositionConstraintMover name="peptide_aa_composition"
            filename="inputs/peptide.comp" selector="peptide" />
        <AddCompositionConstraintMover name="peptide_buried_aa_composition"
            filename="inputs/peptide_buried.comp" selector="select_buried_and_peptide" />
        <AddCompositionConstraintMover name="peptide_exposed_aa_composition"
            filename="inputs/peptide_surf.comp" selector="select_not_buried_and_peptide" />

        # Carry out a round of design with energy-minimization.  Note that the
        # task_operations must be listed on a single line:
        <FastDesign name="fdes1" repeats="1" scorefxn="r15_cst_voids"
            movemap_factory="frlx_mm_factory" packer_palette="design_palette"
            task_operations="resfile_positive_phi,resfile_negative_phi,
            only_repack_very_near_peptide,only_design_peptide"
        />

        # Final relaxation (packing and energy-minimization), keeping amino acid
```

```
        # sequence fixed:
        <FastRelax name="frlx1" repeats="1" scorefxn="r15_cst"
            movemap_factory="frlx_mm_factory"
            task_operations="only_repack_near_peptide,include_current,ex1_ex2"
        />

        # Weakly hold the hinge loop and the peptide in place (allowing small motions)
        # during energy minimization:
        <AddConstraints name="add_bb_csts" >
            <CoordinateConstraintGenerator name="gen_csts" sd="1.0" sidechain="false"
                native="false" residue_selector="constrained_bb"
            />
        </AddConstraints>

        # Sample small motions of the hinge loop, perturbing it slightly from its
        # current conformation while ensuring that the loop remains closed:
        <GeneralizedKIC name="KIC_perturb_loop" selector="lowest_rmsd_selector"
            selector_scorefunction="r15_cst" closure_attempts="100"
            stop_when_n_solutions_found="25"
        >
            <AddResidue res_index="24" />
            <AddResidue res_index="25" />
            <AddResidue res_index="26" />
            <AddResidue res_index="27" />
            <AddResidue res_index="28" />
            <AddResidue res_index="29" />
            <AddResidue res_index="30" />
            <AddResidue res_index="31" />
            <AddResidue res_index="32" />
            <SetPivots res1="24" res2="27" res3="32" atom1="CA" atom2="CA" atom3="CA" />
            <CloseBond res1="27" res2="28" atom1="C" atom2="N" bondlength="1.328685"
                angle1="116.199993" angle2="121.699997" torsion="180.0" />
            <AddPerturber effect="perturb_dihedral" >
                <AddAtoms atom1="N" atom2="CA" res1="24" res2="24" />
                <AddAtoms atom1="N" atom2="CA" res1="25" res2="25" />
                <AddAtoms atom1="N" atom2="CA" res1="26" res2="26" />
                <AddAtoms atom1="N" atom2="CA" res1="27" res2="27" />
                <AddAtoms atom1="N" atom2="CA" res1="28" res2="28" />
                <AddAtoms atom1="N" atom2="CA" res1="29" res2="29" />
                <AddAtoms atom1="N" atom2="CA" res1="30" res2="30" />
                <AddAtoms atom1="N" atom2="CA" res1="31" res2="31" />
                <AddAtoms atom1="N" atom2="CA" res1="32" res2="32" />
                <AddAtoms atom1="CA" atom2="C" res1="24" res2="24" />
                <AddAtoms atom1="CA" atom2="C" res1="25" res2="25" />
                <AddAtoms atom1="CA" atom2="C" res1="26" res2="26" />
                <AddAtoms atom1="CA" atom2="C" res1="27" res2="27" />
                <AddAtoms atom1="CA" atom2="C" res1="28" res2="28" />
                <AddAtoms atom1="CA" atom2="C" res1="29" res2="29" />
                <AddAtoms atom1="CA" atom2="C" res1="30" res2="30" />
                <AddAtoms atom1="CA" atom2="C" res1="31" res2="31" />
                <AddAtoms atom1="CA" atom2="C" res1="32" res2="32" />
                <AddValue value="10" />
            </AddPerturber>
            <AddFilter type="loop_bump_check" />
            <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="24" />
            <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="27" />
            <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="32" />
        </GeneralizedKIC>

        # The series of steps performed for each round of design: carry out FastDesign,
        # update the O and H atoms on the macrocycle and the hinge loop, ensure that the
        # total hydrogen bond count is still over threshold and that residue 237 is
        # satisfied, and ensure that no hbond acceptor is oversaturated:
        <ParsedProtocol name="design_protocol" >
            <Add mover="fdes1" />
            <Add mover="declare_bond" /> # Update O and H atoms in peptide
```

```
        <Add mover="update_loop_O_H" />
        <Add filter="total_hbonds" />
        <Add filter="hbonds_237_2" />
        <Add filter="no_oversat_acceptor2" />
    </ParsedProtocol>

    # The series of steps performed for each round of final relaxation.  These are
    # similar to the design steps, but FastRelax replaces FastDesign (so that the
    # sequence does not change):
    <ParsedProtocol name="relax_protocol" >
        <Add mover="frlx1" />
        <Add mover="declare_bond" /> # Update O and H atoms in peptide
        <Add mover="update_loop_O_H" />
        <Add filter="total_hbonds" />
        <Add filter="hbonds_237_2" />
    </ParsedProtocol>

    # The series of steps performed on each closed solution that the peptide
    # macrocycle generalized kinematic closure mover finds.  Each conformation
    # is first subjected to some filtering steps.  The hinge loop is then perturbed,
    # and three rounds of design, followed by three rounds of relaxation, are carried
    # out.  Each round involves filtering steps that can abort the protocol and discard
    # the candidate GenKIC solution before the full computational expense has been
    # invested:
    <ParsedProtocol name="genkic_preselection_steps" >
        <Add filter="total_hbonds" />
        <Add filter="hbonds_237_2" />
        <Add filter="no_oversat_acceptor1" />
        <Add mover="KIC_perturb_loop" />
        <Add mover="design_protocol" />
        <Add mover="design_protocol" />
        <Add mover="design_protocol" />
        <Add mover="relax_protocol" />
        <Add mover="relax_protocol" />
        <Add mover="relax_protocol" />
        <Add filter="shape_complementarity" />
    </ParsedProtocol>

    # Generalized kinematic closure to close the peptide macrocycle.  Only the upper
    # half of the peptide is permitted to move; the lower half preserves the
    # conformation from the NDM1i-1G crystal structure:
    <GeneralizedKIC name="KIC_close_peptide" selector="lowest_energy_selector"
        selector_scorefunction="r15_cst_voids_scoring" closure_attempts="200"
        stop_when_n_solutions_found="1" pre_selection_mover="genkic_preselection_steps"
    >
        <AddResidue res_index="241" />
        <AddResidue res_index="242" />
        <AddResidue res_index="235" />
        <AddResidue res_index="236" />
        <SetPivots res1="241" atom1="CA" res2="242" atom2="CA" res3="236" atom3="CA" />
        <CloseBond res1="242" res2="235" atom1="C" atom2="N" bondlength="1.328685"
            angle1="116.199993" angle2="121.699997" torsion="180.0" />
        <AddPerturber effect="set_dihedral" >
            <AddAtoms res1="241" res2="242" atom1="C" atom2="N" />
            <AddAtoms res1="242" res2="235" atom1="C" atom2="N" />
            <AddValue value="180.0" />
        </AddPerturber>
        <SampleCisPeptideBond cis_prob="0.2">
            <AddResidue index="241" />
        </SampleCisPeptideBond>
        <AddPerturber effect="randomize_backbone_by_rama_prepro">
            <AddResidue index="241" />
            <AddResidue index="242" />
            <AddResidue index="235" />
        </AddPerturber>
        <AddPerturber effect="perturb_dihedral">
```

```
                <AddAtoms atom1="N" atom2="CA" res1="236" res2="236" />
                <AddAtoms atom1="CA" atom2="C" res1="236" res2="236" />
                <AddValue value="10" />
            </AddPerturber>
            <AddFilter type="loop_bump_check" />
            <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="241" />
            <AddFilter type="rama_prepro_check" rama_cutoff_energy="5.0" residue="242" />
            <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="236" />
        </GeneralizedKIC>
    </MOVERS>

    # This section strings together previously-defined movers and filters
    # to define the overall protocol that will be applied:
    <PROTOCOLS>
        <Add mover="extend_pep" />
        <Add mover="mut_to_gly_235" />
        <Add mover="declare_bond" />
        <Add mover="add_cutpoint_variants" />
        <Add mover="setup_metals" />
        <Add mover="foldtree1" />
        <Add mover="add_hyp" />
        <Add mover="require_net_pos_charge" />
        <Add mover="peptide_aa_composition" />
        <Add mover="peptide_buried_aa_composition" />
        <Add mover="peptide_exposed_aa_composition" />
        <Add mover="add_bb_csts" />
        <Add mover="KIC_close_peptide" />
        <Add filter="shape_complementarity" />
        <Add filter="cavity_volume" />
    </PROTOCOLS>

    # The final structure is re-scored with the ref2015 energy function:
    <OUTPUT scorefxn="r15" />
</ROSETTASCRIPTS>
```

Listing 2.1.2.1: Modernized design script **xml/NDM1i_3_design.xml** for generating NDM1i-3 peptides.

The script above makes use of the following resfiles, which define the amino acids used at D-amino acid and L-amino acid positions, respectively. Note that because backbone sampling was limited, certain positions that were always buried could be restricted to more limited subsets of the allowed amino acids. Note also that these resfiles use the updated syntax and conventions for non-canonical amino acids. For earlier versions of Rosetta, the resfiles in **Section 2.2.3** must be used instead.

```
PIKAA
FAILYVWDERKSTNQHPX[AIB]X[A12]X[A91]X[ORN]X[DAB]X[DPP]X[NLU]X[NVL]X[A34]X[A68]X[A94]X[B30]
    \ X[B96]X[C27]X[B44]X[B67]X[B74]
start
237 E PIKAA FAILYVWPX[AIB]X[A12]X[A91]X[NLU]X[NVL]X[A34]X[A68]X[A94]X[B30]X[B96]X[C27]X[B44]
    \ X[B67]X[B74]
240 E PIKAA FAILYVWPX[AIB]X[A12]X[A91]X[NLU]X[NVL]X[A34]X[A68]X[A94]X[B30]X[B96]X[C27]X[B44]
    \ X[B67]X[B74]
```

Listing 2.1.2.2: Resfile **inputs/neg_phi.resfile**, defining allowed amino acids at positions in the negative-φ region of Ramachandran space (accessible to L-amino acids). Note that truncated lines are indicated with a backslash (\), and must be on a single line for this file to work.

```
PIKAA
X[AIB]X[DPH]X[DAL]X[DIL]X[DLE]X[DTY]X[DVA]X[DTR]X[DAS]X[DGU]X[DAR]X[DLY]X[DSE]X[DTH]X[DAN]
    \ X[DGN]X[DHI]X[DPR]X[DA12]X[DA91]X[DORN]X[DDAB]X[DDPP]X[DNLU]X[DNVL]X[DA34]X[DA68]X[DA94]
    \ X[DB30]X[DB96]X[DC27]X[DB44]X[DB67]X[DB74]
start
237 E PIKAA X[DPH]X[DAL]X[DIL]X[DLE]X[DTY]X[DVA]X[DTR]X[DPR]X[AIB]X[DA12]X[DA91]X[DNLU]X[DNVL]
    \ X[DA34]X[DA68]X[DA94]X[DB30]X[DB96]X[DC27]X[DB44]X[DB67]X[DB74]
240 E PIKAA X[DPH]X[DAL]X[DIL]X[DLE]X[DTY]X[DVA]X[DTR]X[DPR]X[AIB]X[DA12]X[DA91]X[DNLU]X[DNVL]
    \ X[DA34]X[DA68]X[DA94]X[DB30]X[DB96]X[DC27]X[DB44]X[DB67]X[DB74]
```

Listing 2.1.2.3: Resfile **inputs/pos_phi.resfile**, defining allowed amino acids at positions in the positive-φ region of Ramachandran space (accessible to D-amino acids). Note that truncated lines are indicated with a backslash (\), and must be on a single line for this file to work.

The following amino acid composition files must also be included in the **inputs/** directory. They are used to describe the desired overall amino acid composition of the peptide, the desired composition at buried positions, and the desired composition at exposed positions, respectively.

```
# At most, one (or possibly two) exotic NCAAs:
PENALTY_DEFINITION
TYPE A12 A91 NLU NVL A34 A68 A94 B30 B96 C27 B44 B67 B74 A69 B12
DELTA_START -1
DELTA_END 3
PENALTIES 0 0 2 15 65
ABSOLUTE 0
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION

# At least 2 hydrophobic:
PENALTY_DEFINITION
PROPERITES HYDROPHOBIC
DELTA_START -1
DELTA_END 1
PENALTIES 20 0 0
ABSOLUTE 3
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

# At least 3 proline:
PENALTY_DEFINITION
TYPE PRO DPR HYP
DELTA_START -2
DELTA_END 1
PENALTIES 20 7 0 0
ABSOLUTE 3
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

# At least 4 proline or AIB:
PENALTY_DEFINITION
TYPE PRO DPR AIB HYP
DELTA_START -2
DELTA_END 1
PENALTIES 20 7 0 0
ABSOLUTE 4
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
```

```
END_PENALTY_DEFINITION

# At most 2 aromatic:
PENALTY_DEFINITION
PROPERTIES AROMATIC
DELTA_START -1
DELTA_END 1
PENALTIES 0 0 20
ABSOLUTE 2
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION

# At most 1 methionine:
PENALTY_DEFINITION
TYPE MET DME
DELTA_START -1
DELTA_END 1
PENALTIES 0 0 50
ABSOLUTE 1
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION

# At most 1 alanine:
PENALTY_DEFINITION
TYPE ALA DAL
DELTA_START -1
DELTA_END 1
PENALTIES 0 0 50
ABSOLUTE 1
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION

# At most 1 serine:
PENALTY_DEFINITION
TYPE SER DSE
DELTA_START -1
DELTA_END 1
PENALTIES 0 0 50
ABSOLUTE 1
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION
```

Listing 2.1.2.4: Amino acid composition file **inputs/peptide.comp**, defining the desired amino acid composition of the peptide as a whole.

```
# Ramping penalty for any buried residues
# that are not hydrophobic:
PENALTY_DEFINITION
PROPERITES HYDROPHOBIC
DELTA_START -1
DELTA_END 1
PENALTIES 20 0 0
FRACTION 1.0
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION
```

Listing 2.1.2.5: Amino acid composition file **inputs/peptide_buried.comp**, defining the desired amino acid composition of the buried peptide positions.

```
# Ramping penalty for any exposed residues
# that are hydrophobic:
PENALTY_DEFINITION
PROPERITES HYDROPHOBIC
DELTA_START -1
DELTA_END 1
PENALTIES 0 0 20
FRACTION 0
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION
```

Listing 2.1.2.6: Amino acid composition file **inputs/peptide_surf.comp**, defining the desired amino acid composition of the exposed peptide positions.

The net charge of the peptide was also constrained to be positive using the **netcharge** energy term. The following charge constraint file must be included in the **inputs/** directory.

```
DESIRED_CHARGE 1
PENALTIES_CHARGE_RANGE -1 2
PENALTIES 100 25 0 0
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
```

Listing 2.1.2.7: Net charge constraint file **inputs/net_positive.charge**, specifying that the net charge of the peptide should be positive with large, quadratically-ramping penalties for neutral or negative charges and no penalties for charges of +1 or higher.

A fold tree must also be defined in the file **inputs/foldtree1.txt**:

```
FOLD_TREE EDGE 1 27 -1 EDGE 1 231 1 EDGE 231 28 -1 EDGE 182 232 2 EDGE 79 234 3 EDGE 167 233 4
     \ EDGE 233 238 5 EDGE 238 235 -1 EDGE 238 242 -1
```

Listing 2.1.2.8: Fold tree file **inputs/foldtree1.txt**, defining the kinematic relationships between parts of the modelled structure. Note that this should be on a single line.

Finally, the file **inputs/rosetta.flags** contains command-line flags for running the RosettaScripts application:

```
# The number of designs to generate.  On most computing clusters, this can be set to a high
# number, and jobs can be permitted to end when their time limit is reached:
-nstruct 1000

# The XML file to pass to the RosettaScripts parser:
-parser:protocol xml/NDM1i_3_design.xml

# The input PDB file, interpreted as a full-atom model (as opposed to a centroid model):
-in:file:s inputs/Moriarty_xtal_chainB_pep_opened.pdb
-in:file:fullatom
```

```
# Output should include CONECT records for all bonded atoms, to aid visualization in PyMOL:
-write_all_connect_info

# Since glycine Ramachandran tables used for scoring are based on statistics from the Protein
# Data Bank, they are biased towards glycine residues in positive-phi regions of Ramachandran
# space.  When designing with D-amino acids, the following correction should always be
# applied:
-symmetric_gly_tables true

# Do not exit with error status if some jobs fail (e.g. due to filters failing):
-jd2:failed_job_exception false

# The following line should be uncommented for production runs, to prevent writing of a large
# output log:
#-mute all
```

Listing 2.1.2.9: Flags file **inputs/rosetta.flags**, containing command-line flags for running RosettaScripts. Note that the **-auto_setup_metals** flag is omitted since metal setup is handled with the **SetupMetalsMover** within the script.

The RosettaScripts application may be run with the following command-line command. Again, **<path_to_Rosetta>** should be replaced with the user's path to the Rosetta directory, and **.default.linuxgccrelease** may need to be updated depending on the user's build, compiler, and operating system.

```
<path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease
       \ @inputs/rosetta.flags
```

Listing 2.1.2.10: Command-line command to run the RosettaScripts application with the XML file shown in **Listing 2.1.2.1**.

2.1.3 Design protocol for NDM1i-4 peptides

NDM1i-4 designs used the X-ray crystal structure of peptide NDM1i-3D bound to NDM-1 as a starting point. The input PDB file was cleaned by removing duplicate chains and all but 12 water molecules, retaining only those closest to the active site. The script below was used to produce the NDM1i-4 designs. Note that there is no legacy version; this script is compatible as-is with the current version of Rosetta.

This script anchors the macrocycle to the target by the glutamate side chain that coordinates the active-site zinc. It then carries out small perturbations of the macrocycle and the hinge loop, ensuring closure of each using generalized kinematic closure. It proceeds to carry out several rounds of design and energy minimization, allowing both the macrocycle and the hinge loop backbone to move during minimization and making heavy use of guidance scoring terms described in **Section 1.3** to promote hydrogen bond networks, and to discourage voids in the interface and buried unsatisfied hydrogen bond donors and acceptors.

```
<ROSETTASCRIPTS>
        # This script was used to produce the NDM1i-4 peptides, using the NDM1i-3D crystal
        # structure as a starting point and designing with an expanded palette of amino
        # acid building-blocks.  Note that there is no legacy version; this is the version
        # that was used.

    # Energy functions are defined here:
    <SCOREFXNS>
        # The default Rosetta ref2015 energy function:
        <ScoreFunction name="r15" weights="ref2015.wts" />

        # The default energy function with the chainbreak term upweighted, used for
        # relaxing structures:
        <ScoreFunction name="r15_relax" weights="ref2015_cst.wts" >
            <Reweight scoretype="chainbreak" weight="20.0" />
        </ScoreFunction>

        # An energy function for the earlier rounds of design.  This heavily penalizes buried
        # unsatisfied hydrogen bond donors and acceptors and buried voids, and gives a large
        # bonus for hydrogen bond networks.  It also somewhat upweights backbone hydrogen
        # bond terms in order to encourage satisfaction of backbone hydrogen bond donors and
        # acceptors.
        <ScoreFunction name="r15_design" weights="ref2015_cst.wts" >
            <Reweight scoretype="buried_unsatisfied_penalty" weight="30.0" />
            <Reweight scoretype="hbnet" weight="1.0" />
            <Reweight scoretype="voids_penalty" weight="1.0" />
            <Reweight scoretype="aspartimide_penalty" weight="2.0" />
            <Reweight scoretype="aa_composition" weight="1.0" />
            <Reweight scoretype="chainbreak" weight="20.0" />
            <Reweight scoretype="hbond_sr_bb" weight="10.0" />
            <Reweight scoretype="hbond_lr_bb" weight="10.0" />
            <Set voids_penalty_energy_containing_cones_cutoff="4" />
            <Set voids_penalty_energy_cone_distance_cutoff="7.0" />
            <Set buried_unsatisfied_penalty_burial_threshold="1.5" />
            <Set buried_unsatisfied_penalty_cone_dist_midpoint="7.0" />
            <Set buried_unsatisfied_penalty_cone_angle_exponent="2.0" />
            <Set buried_unsatisfied_penalty_cone_angle_shift_factor="0.35" />
        </ScoreFunction>

        # An energy function for the later rounds of design.  This more lightly penalizes
        # buried unsatisfied hydrogen bond donors and acceptors and buried voids, and gives a
        # smaller bonus for hydrogen bond networks.
        <ScoreFunction name="r15_design2" weights="ref2015_cst.wts" >
            <Reweight scoretype="buried_unsatisfied_penalty" weight="10.0" />
            <Reweight scoretype="hbnet" weight="0.5" />
            <Reweight scoretype="voids_penalty" weight="0.1" />
            <Reweight scoretype="aspartimide_penalty" weight="2.0" />
            <Reweight scoretype="aa_composition" weight="1.0" />
            <Reweight scoretype="chainbreak" weight="20.0" />
            <Set voids_penalty_energy_containing_cones_cutoff="4" />
            <Set voids_penalty_energy_cone_distance_cutoff="7.0" />
            <Set buried_unsatisfied_penalty_burial_threshold="1.5" />
            <Set buried_unsatisfied_penalty_cone_dist_midpoint="7.0" />
            <Set buried_unsatisfied_penalty_cone_angle_exponent="2.0" />
            <Set buried_unsatisfied_penalty_cone_angle_shift_factor="0.35" />
        </ScoreFunction>
    </SCOREFXNS>

    # Residue selectors select subsets of a structure based on user-defined rules:
    <RESIDUE_SELECTORS>
        # Select macrocycle residues at the foldtree cutpoint:
        <Index name="select_pep_start" resnums="230" />
        <Index name="select_pep_end" resnums="237" />

        # Select loop residues at the foldtree cutpoint:
        <Index name="select_loop_cutpoint_upper" resnums="29" />
```

```
        <Index name="select_loop_cutpoint_lower" resnums="28" />

        # Select the peptide:
        <Index name="select_pep" resnums="230-237" />

        # Select the target:
        <Not name="select_not_pep" selector="select_pep" />

        # Select residues in the positive-phi and negative-phi regions of
        # Ramachandran space:
        <Phi name="select_pos_phi" select_positive_phi="true" />
        <Not name="select_neg_phi" selector="select_pos_phi" />

        # Select peptide residues in the positive-phi and negative-phi regions of
        # Ramachandran space:
        <And name="select_pos_phi_and_pep" selectors="select_pos_phi,select_pep" />
        <And name="select_neg_phi_and_pep" selectors="select_neg_phi,select_pep" />

        # Select the hinge loop:
        <Index name="select_ndm1_hingeloop" resnums="23-33" />

        # Select the peptide and the hinge loop:
        <Or name="select_movable_loops" selectors="select_ndm1_hingeloop,select_pep" />

        # Select residues near the peptide, including the peptide:
        <Neighborhood name="select_pep_and_vicinity" include_focus_in_subset="true"
            selector="select_movable_loops" distance="8.0" />

        # Select residues far from the peptide.  This does not include the peptide:
        <Not name="select_not_pep_or_vicinity" selector="select_pep_and_vicinity" />

        # Select the anchor glutamate:
        <Index name="select_anchor_res" resnums="234" />

        # Select the proline residue that we do not want to design:
        <Index name="select_preserved_pro" resnums="235" />
    </RESIDUE_SELECTORS>

    # Packer palettes define the set of amino acids with which we will be designing by
    # default.  Task operations can prune this list in a position-specific manner:
    <PACKER_PALETTES>
        # Activate various non-canonical amino acids.  Note that the amino acids must be
        # listed on a single line:
        <CustomBaseTypePackerPalette name="packer_palette"
            additional_residue_types="DALA,DASP,DGLU,DPHE,DHIS,DILE,DLYS,DLEU,DASN,DPRO,DGLN,
            DARG,DSER,DTHR,DVAL,DTRP,DTYR,NLU,NVL,DNLU,DNVL,ORN,DAB,DPP,DORN,DDAB,DDPP,AIB,
            B12,DB12,B96,DB96,BB8,DBB8,A43,DA43,A78,DA78,A80,DA80,B36,DB36"
        />
    </PACKER_PALETTES>

    # Task operations control the packer, which is used for rotamer optimization and sequence
    # design:
    <TASKOPERATIONS>
        # Allow additional task operations to be specified at the command-line:
        <InitializeFromCommandline name="init_from_commandline" />

        # Activate extra-fine sampling when building rotamers:
        <ExtraRotamersGeneric name="extrarot" ex1="true" ex2="false" />
        <ExtraRotamersGeneric name="extrarot2" ex1="true" ex2="true" />

        # Allow design only at peptide positions:
        <OperateOnResidueSubset name="only_design_pep" selector="select_not_pep" >
            <RestrictToRepackingRLT />
        </OperateOnResidueSubset>

        # Prohibit packing outside of the vicinity of the peptide:
```

```
        <OperateOnResidueSubset name="only_pack_near_pep"
            selector="select_not_pep_or_vicinity"
        >
            <PreventRepackingRLT />
        </OperateOnResidueSubset>

        # Do not design with glycine (too flexible), cysteine (forms unwanted disulphides),
        # or methionine (risks oxidation):
        <ProhibitSpecifiedBaseResidueTypes name="no_cys_gly_met" base_types="GLY,CYS,MET"
            selector="select_pep"
        />

        # Prohibit D-amino acids at negative phi positions, and L-amino acids at positive-phi
        # positions.  (Note that we're prohibiting rather than requiring, so as not to
        # exclude the achiral amino acid AIB).
        <ProhibitResidueProperties name="only_l_at_neg_phi" selector="select_neg_phi_and_pep"
            properties="D_AA"
        />
        <ProhibitResidueProperties name="only_d_at_pos_phi" selector="select_pos_phi_and_pep"
            properties="L_AA"
        />

        # Prevent the glutamate anchor residue from packing:
        <OperateOnResidueSubset name="no_pack_anchor" selector="select_anchor_res">
            <PreventRepackingRLT />
        </OperateOnResidueSubset>

        # Prevent all design:
        <RestrictToRepacking name="restrict_to_repacking" />
    </TASKOPERATIONS>

    # Move map factories define the moveable degrees of freedom during energy minimization:
    <MOVE_MAP_FACTORIES>
        # This move map factory specifies that the only moveable degrees of freedom will be
        # the backbone of the peptide and hinge loop, and the side chains near the peptide:
        <MoveMapFactory name="mm_factory" bb="false" chi="false" jumps="false" >
            <Backbone residue_selector="select_movable_loops" />
            <Chi residue_selector="select_pep_and_vicinity" />
        </MoveMapFactory>
    </MOVE_MAP_FACTORIES>

    # Simple metrics measure properties of a structure and allow those measurements to be
    # stored in the pose and written out with the PDB file.  Here, we use them to record
    # the peptide sequence to allow peptide structure prediction to be set up more easily:
    <SIMPLE_METRICS>
        <SequenceMetric name="capture_sequence" output_mode="basename"
            residue_selector="select_pep"
        />
    </SIMPLE_METRICS>

    # We include an additional MOVERS block here to define a mover that will be used by the
    # Ddg filter:
    <MOVERS>
        # A FastRelax mover for the Ddg filter.  Note that the task operations must be listed
        # on a single line:
        <FastRelax name="frelax_for_filter" repeats="3" scorefxn="r15_relax"
            movemap_factory="mm_factory"
            task_operations="extrarot2,restrict_to_repacking,
            only_pack_near_pep" relaxscript="MonomerRelax2019"
        />
    </MOVERS>

    # Filters measure properties of structures and make pass/fail decisions based
    # on their measurements:
    <FILTERS>
        # Measure shape complementarity of peptide to target:
```

```
        <ShapeComplementarity name="shape_complementarity" min_sc="0.4" min_interface="250"
            write_int_area="true" residue_selector1="select_pep"
            residue_selector2="select_not_pep"
        />

        # Estimate delta-G of binding:
        <Ddg name="ddg" jump="2" confidence="1" threshold="-10.0" scorefxn="r15"
            repeats="15" repack="true" repack_unbound="false" relax_unbound="true"
            repack_bound="false" relax_bound="true" relax_mover="frelax_for_filter"
            translate_by="10000" extreme_value_removal="true"
         />
    </FILTERS>

    # Movers modify a structure in some way:
    <MOVERS>
        # Set up cyclization:
        <DeclareBond name="cyclization_bond" res1="237" res2="230" atom1="C" atom2="N" />

        # Update H and O atoms at the cutpoint in the hinge loop:
        <DeclareBond name="loop_bond" res1="28" res2="29" atom1="C" atom2="N" />

        # Add cutpoint variants:
        <ModifyVariantType name="add_cutpoint_upper_pep" add_type="CUTPOINT_UPPER"
            residue_selector="select_pep_start"
        />
        <ModifyVariantType name="add_cutpoint_lower_pep" add_type="CUTPOINT_LOWER"
            residue_selector="select_pep_end"
        />
        <ModifyVariantType name="add_cutpoint_upper_loop" add_type="CUTPOINT_UPPER"
            residue_selector="select_loop_cutpoint_upper"
        />
        <ModifyVariantType name="add_cutpoint_lower_loop" add_type="CUTPOINT_LOWER"
            residue_selector="select_loop_cutpoint_lower"
        />

        # Set up the fold tree:
        <AtomTree name="foldtree1" fold_tree_file="inputs/foldtree1.txt" />

        # Add a small perturbation to the anchor's mainchain and side chain dihedral angles:
        <SetTorsion name="perturb_anchor" >
            <Torsion residue="234" torsion_name="phi" angle="perturb"
                perturbation_type="gaussian" perturbation_magnitude="5.0"
            />
            <Torsion residue="234" torsion_name="psi" angle="perturb"
               perturbation_type="gaussian" perturbation_magnitude="5.0"
            />
            <Torsion residue="pick_atoms" angle="perturb" perturbation_type="gaussian"
                perturbation_magnitude="5.0"
            >
                <Atom1 residue="234" atom="N" />
                <Atom2 residue="234" atom="CA" />
                <Atom3 residue="234" atom="CB" />
                <Atom4 residue="234" atom="CG" />
            </Torsion>
        </SetTorsion>

        # This GenKIC mover adds a small perturbation to the peptide conformation:
        <GeneralizedKIC name="genkic_pep" pre_selection_mover="cyclization_bond"
            closure_attempts="1000" correct_polymer_dependent_atoms="true"
            stop_when_n_solutions_found="20" selector="lowest_rmsd_selector"
        >
            <AddResidue res_index="235" />
            <AddResidue res_index="236" />
            <AddResidue res_index="237" />
            <AddResidue res_index="230" />
            <AddResidue res_index="231" />
```

```
        <AddResidue res_index="232" />
        <AddResidue res_index="233" />
        <SetPivots res1="235" res2="231" res3="233" atom1="CA" atom2="CA" atom3="CA" />
        <CloseBond atom1="C" atom2="N" res1="237" res2="230" bondlength="1.328685"
            angle1="116.199993" angle2="121.699997" torsion="180" />
        <AddPerturber effect="perturb_dihedral" >
            <AddAtoms res1="235" res2="235" atom1="N" atom2="CA" />
            <AddAtoms res1="236" res2="236" atom1="N" atom2="CA" />
            <AddAtoms res1="237" res2="237" atom1="N" atom2="CA" />
            <AddAtoms res1="230" res2="230" atom1="N" atom2="CA" />
            <AddAtoms res1="231" res2="231" atom1="N" atom2="CA" />
            <AddAtoms res1="232" res2="232" atom1="N" atom2="CA" />
            <AddAtoms res1="233" res2="233" atom1="N" atom2="CA" />

            <AddAtoms res1="235" res2="235" atom1="CA" atom2="C" />
            <AddAtoms res1="236" res2="236" atom1="CA" atom2="C" />
            <AddAtoms res1="237" res2="237" atom1="CA" atom2="C" />
            <AddAtoms res1="230" res2="230" atom1="CA" atom2="C" />
            <AddAtoms res1="231" res2="231" atom1="CA" atom2="C" />
            <AddAtoms res1="232" res2="232" atom1="CA" atom2="C" />
            <AddAtoms res1="233" res2="233" atom1="CA" atom2="C" />

            <AddValue value="5.0" />
        </AddPerturber>
        <AddFilter type="loop_bump_check" />
    </GeneralizedKIC>


    # This GenKIC mover perturbs the hinge loop:
    <GeneralizedKIC name="genkic_loop" pre_selection_mover="loop_bond"
        closure_attempts="1000" correct_polymer_dependent_atoms="true"
        stop_when_n_solutions_found="20" selector="lowest_rmsd_selector"
    >
        <AddResidue res_index="23" />
        <AddResidue res_index="24" />
        <AddResidue res_index="25" />
        <AddResidue res_index="26" />
        <AddResidue res_index="27" />
        <AddResidue res_index="28" />
        <AddResidue res_index="29" />
        <AddResidue res_index="30" />
        <AddResidue res_index="31" />
        <AddResidue res_index="32" />
        <AddResidue res_index="33" />
        <SetPivots res1="23" res2="28" res3="33" atom1="CA" atom2="CA" atom3="CA" />
        <CloseBond atom1="C" atom2="N" res1="28" res2="29" bondlength="1.328685"
            angle1="116.199993" angle2="121.699997" torsion="180" />
        <AddPerturber effect="perturb_dihedral" >
            <AddAtoms res1="23" res2="23" atom1="N" atom2="CA" />
            <AddAtoms res1="24" res2="24" atom1="N" atom2="CA" />
            <AddAtoms res1="25" res2="25" atom1="N" atom2="CA" />
            <AddAtoms res1="26" res2="26" atom1="N" atom2="CA" />
            <AddAtoms res1="27" res2="27" atom1="N" atom2="CA" />
            <AddAtoms res1="28" res2="28" atom1="N" atom2="CA" />
            <AddAtoms res1="29" res2="29" atom1="N" atom2="CA" />
            <AddAtoms res1="30" res2="30" atom1="N" atom2="CA" />
            <AddAtoms res1="31" res2="31" atom1="N" atom2="CA" />
            <AddAtoms res1="32" res2="32" atom1="N" atom2="CA" />
            <AddAtoms res1="33" res2="33" atom1="N" atom2="CA" />

            <AddAtoms res1="23" res2="23" atom1="CA" atom2="C" />
            <AddAtoms res1="24" res2="24" atom1="CA" atom2="C" />
            <AddAtoms res1="25" res2="25" atom1="CA" atom2="C" />
            <AddAtoms res1="26" res2="26" atom1="CA" atom2="C" />
            <AddAtoms res1="27" res2="27" atom1="CA" atom2="C" />
            <AddAtoms res1="28" res2="28" atom1="CA" atom2="C" />
```

```
                <AddAtoms res1="29" res2="29" atom1="CA" atom2="C" />
                <AddAtoms res1="30" res2="30" atom1="CA" atom2="C" />
                <AddAtoms res1="31" res2="31" atom1="CA" atom2="C" />
                <AddAtoms res1="32" res2="32" atom1="CA" atom2="C" />
                <AddAtoms res1="33" res2="33" atom1="CA" atom2="C" />

                <AddValue value="3.0" />
            </AddPerturber>
            <AddFilter type="loop_bump_check" />
        </GeneralizedKIC>

        # Add the amino acid composition constraints:
        <AddCompositionConstraintMover name="add_comp" filename="inputs/design.comp"
            selector="select_pep"
        />

        # A first round of design, with strong penalties for buried unsatisfied
        # hydrogen bond donors and acceptors or buried voids.  Note that the task_operations
        # must be listed on a single line:
        <FastDesign name="fdes1" packer_palette="packer_palette" repeats="3"
            relaxscript="InterfaceDesign2019" scorefxn="r15_design"
            task_operations="init_from_commandline,extrarot,
            no_cys_gly_met,no_pack_anchor,only_l_at_neg_phi,only_d_at_pos_phi,
            only_pack_near_pep,only_design_pep" movemap_factory="mm_factory"
        />

        # A second round of design, with slightly weaker penalties for buried unsatisfied
        # hydrogen bond donors and acceptors or buried voids.  Note that the task_operations
        # must be listed on a single line:
        <FastDesign name="fdes2" packer_palette="packer_palette" repeats="3"
            relaxscript="InterfaceDesign2019" scorefxn="r15_design2"
            task_operations="init_from_commandline,extrarot,
            no_cys_gly_met,no_pack_anchor,only_l_at_neg_phi,only_d_at_pos_phi,
            only_pack_near_pep,only_design_pep" movemap_factory="mm_factory"
        />

        # Store the amino acid sequence at the end for output in the PDB file:
        <RunSimpleMetrics name="run_metrics" metrics="capture_sequence" prefix="SEQ_" />
    </MOVERS>

    # Movers and filters that are defined above are listed here to define a protocol:
    <PROTOCOLS>
        <Add mover="cyclization_bond" />
        <Add mover="foldtree1" />
        <Add mover="add_cutpoint_upper_pep" />
        <Add mover="add_cutpoint_lower_pep" />
        <Add mover="add_cutpoint_upper_loop" />
        <Add mover="add_cutpoint_lower_loop" />
        <Add mover="perturb_anchor" />
        <Add mover="genkic_pep" />
        <Add mover="genkic_loop" />
        <Add mover="add_comp" />
        <Add mover="fdes1" />
        <Add mover="fdes2" />
        <Add mover="run_metrics" />
        <Add filter="shape_complementarity" />
        <Add filter="ddg" />
    </PROTOCOLS>
</ROSETTASCRIPTS>
```

Listing 2.1.3.1: RosettaScripts XML **xml/NDM1i_4_design.xml** defining the protocol used to produce NDM1i-4 designs.

The file **inputs/foldtree1.txt** defines the kinematic relationships in the structure. It contains 16 separate rigid-body transformations (jumps to the 12 bound water molecules, plus one to the peptide and three to the three bound zinc atoms (two at the active site, and one at a peripheral zinc-binding site). Note that the current default behaviour of the Rosetta packer is to consider "virtualized" water as an option during repacking or design, effectively allowing water molecules to disappear into bulk solvent if a polar polymeric group could occupy the same space. Also note that the unusual **JEDGE** syntax, below, specifies a through-space connection that tethers the peptide by its glutamate side chain instead of by a backbone atom.

```
FOLD_TREE
EDGE 1 28 -1
EDGE 23 35 1
EDGE 35 29 -1
EDGE 35 229 -1
JEDGE 209 234 2 C OE1 INTRA_RES_STUB
EDGE 234 230 -1
EDGE 234 237 -1
EDGE 209 238 3
EDGE 209 239 4
EDGE 209 240 5
EDGE 209 241 6
EDGE 209 242 7
EDGE 209 243 8
EDGE 209 244 9
EDGE 209 245 10
EDGE 209 246 11
EDGE 209 247 12
EDGE 209 248 13
EDGE 209 249 14
EDGE 209 250 15
EDGE 209 251 16
```

Listing 2.1.3.2: File **inputs/foldtree1.txt**, defining the kinematic directed acyclic graph for NDM1i-4 design.

Amino acid composition preferences were defined in the file **inputs/design.comp**, shown below.

```
# Since beta,beta-diphenyl-L-alanine is a very bulky amino acid residue, it can make many
# favourable interactions, and is artificially favoured by the energy function.  Here, we
# penalize more than one residue of this type in a design:
PENALTY_DEFINITION
TYPE B96
DELTA_START -1
DELTA_END 1
PENALTIES 0 5 25
ABSOLUTE 1
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION

# At least two polar residues:
PENALTY_DEFINITION
PROPERTIES POLAR
DELTA_START -1
DELTA_END 1
PENALTIES 50 0 0
```

```
ABSOLUTE 2
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

#At least one aromatic with absorption at 280 nm:
PENALTY_DEFINITION
PROPERTIES AROMATIC
NOT_TYPE PHE DPH
DELTA_START -1
DELTA_END 1
PENALTIES 50 0 0
ABSOLUTE 1
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

# At least two L- or D-proline residues:
PENALTY_DEFINITION
TYPE PRO DPR
DELTA_START -1
DELTA_END 1
PENALTIES 50 0 0
ABSOLUTE 2
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

# At least three L-proline, D-proline, or AIB residues:
PENALTY_DEFINITION
TYPE PRO DPR AIB
DELTA_START -1
DELTA_END 1
PENALTIES 50 0 0
ABSOLUTE 3
BEFORE_FUNCTION QUADRATIC
AFTER_FUNCTION CONSTANT
END_PENALTY_DEFINITION

# At most one alanine or D-alanine:
PENALTY_DEFINITION
TYPE ALA DAL
DELTA_START -1
DELTA_END 1
PENALTIES 0 0 10
ABSOLUTE 1
BEFORE_FUNCTION CONSTANT
AFTER_FUNCTION QUADRATIC
END_PENALTY_DEFINITION
```

Listing 2.1.3.2: File **inputs/design.comp**, defining the preferred amino acid composition for NDM1i-4 design.

Command-line options for running RosettaScripts can be listed in **inputs/rosetta.flags**, as follows. The RosettaScripts executable may then be run using the command-line command shown in **Listing 2.1.3.4**, again substituting **<path_to_Rosetta>** with the path to the Rosetta directory, and **.default.linuxgccrelease** with the suffix corresponding to the user's build, compiler, and operating system.

```
# On most clusters with queueing systems, it is simplest to set the number of output
# structures to a large number and to allow Rosetta to write structures until the job times
# out:
```

```
-nstruct 1000

# Certain versions of Rosetta between 2018 and 2020 had problems with name-clashes between
# Rosetta noncanonical types and residue types from the wwPDB Chemical Components Dictionary.
# Although this issue has been resolved, this flag was a workaround:
-load_PDB_components false

# Automatically detect bonds to metals and set up constraints to preserve metal coordination
# geometry:
-auto_setup_metals

# Write out virtual atoms to help with debugging.  These can be deleted in PyMOL by removing
# atoms with element type X ("rm e. X"):
-output_virtual true

# Since we expect some jobs to fail due to filters that fail, we don't want the executable
# to return an error status if some jobs fail:
-jd2:failed_job_exception false

# Since glycine Ramachandran tables used for scoring are based on statistics from the Protein
# Data Bank, they are biased towards glycine residues in positive-phi regions of Ramachandran
# space.  When designing with D-amino acids, the following correction should always be
# applied:
-symmetric_gly_tables true

# The input file, interpreted as a full-atom (as opposed to centroid) model:
-in:file:s inputs/NDM-1_P2_model_alt_conformation_trimmed.pdb
-in:file:fullatom

# For convenience, write CONECT records to the output PDB file for all bonded atoms:
-write_all_connect_info

# The XML file defining the protocol to run:
-parser:protocol xml/NDM1i_4_design.xml

# We do not want Rosetta to delete the water molecules in the input PDB file:
-ignore_waters false

# Uncomment the following to mute unneeded output for production runs, to reduce the size of
# log files:
#-mute all
```

Listing 2.1.3.3: File **inputs/rosetta.flags**, defining the preferred amino acid composition for NDM1i-4 design.

```
<path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default_linuxgccrelease
       \ @inputs/rosetta.flags
```

Listing 2.1.3.4: Command-line command to execute the RosettaScripts executable and to run the NDM1i-4 design script.

2.1.4 Protocol for estimating binding free energies for the current version of Rosetta

To estimate binding free energies, we used the inaccurately-named Rosetta **Ddg** filter, which measures the energy of a bound complex and a separated complex, reporting the difference as an estimate of $\Delta G_{binding}$. The script below relaxes the bound complex, then applies the **Ddg** filter. It is configured to repack and minimize only the peptide, hinge loop, and residues

close to the peptide in the bound state. Residue selectors and a **FastRelax** mover are provided to the **Ddg** filter that relax the separated peptide and target, ensuring that the same selection of residues (which defined the interface between peptide and target prior to their separation) is allowed to move during relaxation. We ran this script 20 times and took the average and standard error of the mean of the results. Note that this script does not perform extensive sampling of peptide or hinge loop conformations, but does allow both regions to move (relax) during energy-minimization.

```
<ROSETTASCRIPTS>
        # Given an input structure of a candidate inhibitor peptide bound to NDM-1, this
        # script estimates the delta-G of binding using the (poorly-named) ddG protocol.
        # It carries out relaxation of both bound and unbound structures, allowing small
        # motions of the hinge loop.
        #
        # This script was tested with the current version of Rosetta (Git SHA1
        # cb360c57ed4ba63d45678afba4bf6a39d8dd958d). It should be compatible with all Rosetta
        # versions after weekly release 2020.11.

    # Scoring functions are defined in this section:
    <SCOREFXNS>
        # The default Rosetta energy function, ref2015, with the metalbinding constraint
        # turned OFF explicitly.  We will use this for scoring the bound and unbound states,
        # to ensure that the metal coordination is not artificially inflating the score.
        <ScoreFunction name="r15" weights="ref2015" >
            <Reweight scoretype="metalbinding_constraint" weight="0.0" />
        </ScoreFunction>
        # The default energy function with the chainbreak term upweighted:
        <ScoreFunction name="r15_cst" weights="ref2015_cst" >
            <Reweight scoretype="metalbinding_constraint" weight="1.0" />
            <Reweight scoretype="chainbreak" weight="20.0" />
        </ScoreFunction>
        # The default energy function with the chainbreak term upweighted and the metalbinding
        # constraint term explicitly turned OFF.  See note above.
        <ScoreFunction name="r15_cst_nometal" weights="ref2015_cst" >
            <Reweight scoretype="metalbinding_constraint" weight="0.0" />
            <Reweight scoretype="chainbreak" weight="20.0" />
        </ScoreFunction>
    </SCOREFXNS>

    # Residue selectors, used to configure modules that operate on a subset of a pose, are
    # declared here:
    <RESIDUE_SELECTORS>
        # Select the peptide:
        <Index name="select_pep" resnums="234-241" />

        # Select the flexible hinge loop:
        <Index name="select_hinge_loop" resnums="25-33" />

        # Select the vicinity of the peptide (including the peptide):
        <Neighborhood name="select_near_pep" selector="select_pep" distance="10.0"
            include_focus_in_subset="true" />

        # Select residues that are not in the vicinity of the peptide.  This selection
        # excludes the peptide:
        <Not name="select_not_near_pep" selector="select_near_pep" />

        # Select a previously-stored selection of residues representing the interface between
        # peptide and target.
        <StoredResidueSubset name="selected_near_pep" subset_name="interface" />

        # Select those residues that are not in the stored selection above.
```

```
        <Not name="selected_not_near_pep" selector="selected_near_pep" />

        # Select the upper residue in the cutpoints in the peptide macrocycle and in the
        # hinge loop (to apply the cutpoint upper variant type):
        <Index name="select_upper_cutpoints" resnums="30,234" />

        # Select the lower residue in the cutpoints in the peptide macrocycle and in the
        # hinge loop (to apply the cutpoint lower variant type):
        <Index name="select_lower_cutpoints" resnums="29,241" />
    </RESIDUE_SELECTORS>

    # Task operations, which control the Rosetta packer during rotamer optimization, are
    # defined here.  Note that we do no design in this script, but still use the packer to
    # optimize rotamer conformation:
    <TASKOPERATIONS>
        # Prevent design everywhere:
        <RestrictToRepacking name="repack_only" />

        # Prevent repacking far from the peptide:
        <OperateOnResidueSubset name="only_near_pep" selector="selected_not_near_pep" >
            <PreventRepackingRLT />
        </OperateOnResidueSubset>

        # Ensure that the input rotamer is included in the set of rotamers considered during
        # rotamer optimization:
        <IncludeCurrent name="include_current" />

        # Enable finer discretization of rotamers:
        <ExtraRotamersGeneric name="ex1_ex2" ex1="true" ex2="true" extrachi_cutoff="0" />
    </TASKOPERATIONS>

    # Jump selectors select rigid-body transforms when configuring move map factories, which
    # control energy minimization steps:
    <JUMP_SELECTORS>
        # Select the rigid-body degree of freedom connecting the target to the peptide:
        <JumpIndex name="select_pep_jump" jump="4" />
    </JUMP_SELECTORS>

    # Move map factories control energy minimization steps by defining which degrees of
    # freedom are movable and which are fixed:
    <MOVE_MAP_FACTORIES>
        # The move map factory for the relaxation that is carried out on the bound complex.
        # Here, we allow only the side chains near the peptide, only the backbone of the
        # peptide and the hinge loop, and only the rigid-body transform from NDM-1 to peptide
        # to move during energy minimization:
        <MoveMapFactory name="mmf_prerelax" bb="false" chi="false" jumps="0" >
            <Chi enable="true" residue_selector="selected_near_pep" />
            <Jumps enable="true" jump_selector="select_pep_jump" />
            <Backbone enable="true" residue_selector="select_hinge_loop" />
            <Backbone enable="true" residue_selector="select_pep" />
        </MoveMapFactory>

        # The move map factory for the relaxation that is carried out on the separated
        # peptide and target.  Here, we allow only the side chains that had been at the
        # interface, and only the backbone of the peptide and the hinge loop to move.  All
        # rigid-body degrees of freedom are fixed during this relaxation:
        <MoveMapFactory name="mmf_post_separation" bb="false" chi="false" jumps="0" >
            <Chi enable="true" residue_selector="selected_near_pep" />
            <Backbone enable="true" residue_selector="select_hinge_loop" />
            <Backbone enable="true" residue_selector="select_pep" />
        </MoveMapFactory>
    </MOVE_MAP_FACTORIES>

    # Movers alter a structure in some way.  Here we define a mover that will be used by
    # the Ddg filter, which repacks and energy-minimizes the separated peptide and target:
    <MOVERS>
```

```
    <FastDesign name="frlx_for_filter" repeats="3" scorefxn="r15_cst_nometal"
        task_operations="repack_only,only_near_pep,include_current,ex1_ex2"
        movemap_factory="mmf_post_separation"
    />
</MOVERS>

# Filters are used to measure properties of a structure and to make pass/fail decisions.
# Here, we define the Ddg filter that will be used to measure the energy of the bound and
# separated structures, and to report the difference.  This filter takes as input a
# FastRelax mover, defined above, which is applied to the separated complex.
<FILTERS>
    <Ddg name="ddg" jump="4" confidence="0" threshold="0.0" scorefxn="r15" repeats="15"
        repack="true" repack_unbound="false" relax_unbound="true" repack_bound="false"
        relax_bound="false" relax_mover="frlx_for_filter" translate_by="10000"
        extreme_value_removal="true"
    />
</FILTERS>

# Here we define additional movers:
<MOVERS>
    # Connect the ends of the peptide to make an N-to-C cyclic macrocycle:
    <DeclareBond name="endbond" res1="234" res2="241" atom1="N" atom2="C" />

    # Set up the kinematic directed acyclic graph (DAG) for the structure:
    <AtomTree name="foldtree" fold_tree_file="inputs/foldtree.txt" />

    # Add cutpoint variant types to the residues flanking the cutpoints in the macrocycle
    # and in the hinge loop:
    <ModifyVariantType name="add_upper_cutpoints" add_type="CUTPOINT_UPPER"
        residue_selector="select_upper_cutpoints" />
    <ModifyVariantType name="add_lower_cutpoints" add_type="CUTPOINT_LOWER"
        residue_selector="select_lower_cutpoints"/>

    # Identify interface residues and store this residue selection in the pose for later
    # retrieval after the complex is separated:
    <StoreResidueSubset name="store_interface" residue_selector="select_near_pep"
        subset_name="interface" />

    # FastDesign with the RestrictToRepacking task operation does the same thing that
    # FastRelax would.  It is used here for convenience for the step of relaxing the
    # bound complex, since it reports the residues that it is repacking (while FastRelax
    # is less verbose):
    <FastDesign name="frlx" repeats="3" scorefxn="r15_cst"
        task_operations="repack_only,only_near_pep,include_current,ex1_ex2"
        movemap_factory="mmf_prerelax"
    />

    # Remove constraints from the pose:
    <ClearConstraintsMover name="clear_csts" />
</MOVERS>

# This section puts together previously-defined movers and filters to construct an
# overall protocol:
<PROTOCOLS>
    <Add mover="endbond" />
    <Add mover="foldtree" />
    <Add mover="add_upper_cutpoints" />
    <Add mover="add_lower_cutpoints" />
    <Add mover="store_interface" />
    <Add mover="frlx" />
    <Add mover="clear_csts" />
    <Add filter="ddg" report_at_end="false" />
</PROTOCOLS>

# At the end, the ref2015 energy function is used to score the output pose:
<OUTPUT scorefxn="r15" />
```

```
</ROSETTASCRIPTS>
```

Listing 2.1.4.1: Protocol **xml/ddgscript.xml**, used to estimate the $\Delta G_{binding}$ of candidate NDM-1 inhibitor peptides to NDM-1.

The script above requires file **input/foldtree.txt**, shown below.

```
FOLD_TREE
EDGE 3 29 -1
EDGE 3 1 1
EDGE 3 2 2
EDGE 20 40 3
EDGE 40 30 -1
EDGE 40 233 -1
EDGE 150 237 4
EDGE 237 234 -1
EDGE 237 241 -1
```

Listing 2.1.4.2: File **inputs/foldtree.txt**, defining the kinematic directed acyclic graph that establishes the kinematic relationships between parts of the peptide-target structure.

Flags for the RosettaScripts executable should be stored in file **inputs/rosetta.flags**, listed below. Input files PDB files should be listed, one per line, in a file called **inputs/pdbs.list**.

```
# For each structure, carry out 20 replicates:
-nstruct 20

# A list of input files:
-in:file:l inputs/pdbs.list

# Input files will be treated as full-atom representations, not centroid representations.
-in:file:fullatom

# The XML file to run:
-parser:protocol xml/ddgscript.xml

# Automatically set up bonds to metal atoms:
-auto_setup_metals

# Write virtual atoms in the output file.  This is useful for debugging:
-output_virtual

# The following line should be uncommented for production runs, to prevent writing of a large
# output log:
#-mute all
```

Listing 2.1.4.3: File **inputs/rosetta.flags**, defining the command-line flags to run the RosettaScripts application.

The script shown in **Listing 2.1.4.1** can be run using the following command at the command-line. As before, **<path_to_Rosetta>** should be replaced with the path to Rosetta, and **.default.linuxgccrelease** may have to be updated for the user's build, compiler, and operating system.

```
<path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default_linuxgccrelease
```

```
        \ @inputs/rosetta.flags
```

Listing 2.1.4.4: Command-line command to execute the RosettaScripts executable to estimate $\Delta G_{binding}$.

2.1.5 Protocol for predicting fold propensity and folding free energies

Validation of peptide designs was carried out with the Rosetta **simple_cycpep_predict** application described previously (8, 16). As described in **Section 1.5.4**, the Message Passing Interface (MPI) build of this application efficiently parallelizes its work and automatically computes P_{Near} and an estimate of the $\Delta G_{folding}$ value from the ensemble of peptide macrocycle conformations that it samples. To build the MPI version of the **simple_cycpep_predict** application, run the following command from the **Rosetta/main/source/** directory, replacing **<number_of_processes>** with the number of parallel compilation processes that you wish to launch (typically, equal to the number of cores available on the computing node used to compile):

```
./scons.py -j <number_of_processes> mode=release extras=mpi,serialization
    \ simple_cycpep_predict
```

Listing 2.1.5.1: Compilation command for the MPI version of the Rosetta **simple_cycpep_predict** application.

Inputs for this application include a sequence file, an (optional) PDB file representing the native structure, and a flags file listing command-line options. To use the NDM1i-1G peptide as an example, the sequence file, **inputs/seq.txt**, would contain the following:

```
DARG DARG LEU DCYS PRO ILE PRO GLU
```

Listing 2.1.5.2: Sequence file **inputs/seq.txt**, containing the sequence of peptide NDM1i-1G.

The file **inputs/native.pdb**, if provided, should contain only the peptide macrocycle. Note that LINK records must be stripped from the PDB file.

The flags used to generate the plots shown in **Fig. 2** are as follows:

```
# For an 8mer macrocycle, 80,000 samples is perhaps two to four times the minimum needed:
-nstruct 80000

# The structure of the desired conformation, interpreted as an all-atom structure:
-in:file:native inputs/native.pdb
-in:file:fullatom

# Automatically set up a master->slave process hierarchy in which all but one process will
# be used to compute structures:
-cyclic_peptide:MPI_auto_2level_distribution

# Send 25 jobs to each slave at a time.  (Overfrequent communication can create bottlenecks
# as the master struggles to field all of the requests for jobs):
-cyclic_peptide:MPI_batchsize_by_level 25

# Write output structures to this file, in the Rosetta binary silent file format:
```

```
-out:file:silent out.silent

# The sequence file:
-sequence_file inputs/seq.txt

# Options configuring the kinematic closure step and relaxation steps:
-genkic_closure_attempts 150
-genkic_min_solution_count 1
-min_genkic_hbonds 1
-min_final_hbonds 1
-fast_relax_rounds 3

# Since glycine Ramachandran tables used for scoring are based on statistics from the Protein
# Data Bank, they are biased towards glycine residues in positive-phi regions of Ramachandran
# space.  When designing with D-amino acids, the following correction should always be
# applied:
-symmetric_gly_tables true

# In addition to computing the RMSD and PNear value to the conformation of native.pdb, this
# option allows the application to compute the RMSD and PNear value to the lowest-energy
# conformation sampled:
-cyclic_peptide:compute_rmsd_to_lowest true

# Write the lowest-energy 0.02% of structures to disk:
-cyclic_peptide:MPI_output_fraction 0.0002

# Discard samples with a Ramachandran score greater than 5.0 kcal/mol for any residue.
-cyclic_peptide:rama_cutoff 5.0

# Set the lambda parameter (in Angstroms) for computing PNear.  This defines how much a
# structure can deviate from the desired conformation and still be considered to be "near" it:
-cyclic_peptide:MPI_pnear_lambda 1.5

# Set the value of the Boltzmann temperature to physiological temperature.  (37 degrees C or
# 310 K corresponds to 0.62 kcal/mol):
-cyclic_peptide:MPI_pnear_kbt 0.62

# Discard high-energy structures:
-cyclic_peptide:total_energy_cutoff 20.0

# The following two lines ensure that only the summary of results is written to the
# output log:
-mute all
-unmute protocols.cyclic_peptide_predict.SimpleCycpepPredictApplication_MPI
protocols.cyclic_peptide_predict.SimpleCycpepPredictApplication_MPI_summary
```

Listing 2.1.5.3: File **inputs/rosetta.flags** listing command-line options for running the **simple_cycpep_predict** application.

The **simple_cycpep_predict** application is run from the command-line using the following command. Note that **<processes_to_run>** must be replaced with the number of processes to launch (a number greater than or equal to 2), **<path_to_Rosetta>** must be replaced with the path to the user's Rosetta directory, and **linuxgccrelease** may need to be updated for the user's compiler and operating system.

```
nohup mpirun -np <processes_to_run> <path_to_Rosetta>/Rosetta/main/source/bin/
    \ simple_cycpep_predict.mpiserialization.linuxgccrelease
    \ @inputs/rosetta.flags >out.log 2>err.log &
```

Listing 2.1.5.3: File **inputs/rosetta.flags** listing command-line options for running the **simple_cycpep_predict** application.

Output will be written to the files **out.silent** (containing Rosetta binary representations of the lowest-energy structures, which can be converted to PDB files with the Rosetta **extract_pdbs** application), **out.log** (containing the output log), and **err.log** (containing the error log). A summary of energies and RMSD values for all samples will be in the output log, followed by the computed P_{Near} and $\Delta G_{folding}$ values.

2.1.6 Backbone conformational bin analysis

As a final consideration when choosing designs, we sought some diversity in the backbone conformations in the design pool, particularly for NDM1i-1 designs. To classify backbone conformations, we assigned a backbone conformational bin to each residue using the A, B, X, Y nomenclature described previously (8). Briefly, residues with backbone ϕ dihedral angles greater than 0 were assigned to bins X or Y, and those with ϕ angles less than 0° were assigned to A or B. Bin A contained residues with ψ angles between -125° and 50°, and bin B contained residues with ψ angles outside of this range. Similarly, bin X contained residues with ψ angles between -50° and 125°, and bin Y contained residues with ψ angles outside of this range. Among NDM1i-1 designs, the seven peptides chosen represented four unique bin strings (**Fig. 2** in the main text).

2.1.7 Comparison of NDM1i-3D design and crystal structures

Peptide NDM1i-3D folded into the designed conformation, but bound to the active site in a configuration rotated nearly 180° relative to the design model, as revealed by the crystal structure (**Fig. 5** in the main text). Although the inverted configuration was not considered during the design or validation process, we wanted to determine whether the inverted conformation was favoured by Rosetta's energy function -- that is, whether broader exploration of alternative docked configurations could conceivably have predicted it. There were several considerations in order to compare the design and crystal structures. First, X-ray crystal structures are often very close to local energy minima in the Rosetta energy function, but because small deviations from Rosetta's notion of ideality can result in very large increases in the energy, it was necessary to perform gradient-descent energy minimization on the crystal structure before scoring it. Second, we needed to ensure that the unusual elements of the structure (particularly the N-to-C amide bond in the peptide macrocycle and the bonds between the metal-liganding residue and the active-site metals) were preserved during energy minimization. Third, Rosetta is not able to perform the high-accuracy QM calculations that would be necessary to compare the favourability of coordinating the active-site metals with a cysteine side-chain (as in the design) versus a glutamate (as in the crystal structure). We therefore opted for a scoring protocol that omitted this interaction energy, scoring only the noncovalent interactions between amino acid residues and the internal conformational preferences of amino acid residues. And fourth, two additional C-terminal amino acid residues on NDM-1 were present in the design model that were

not resolved in the crystal structure, while water molecules were visible in the crystal structure that were not present in the design. In the interests of making an apples-to-apples comparison, the extra residues were deleted from the design model and the water molecules were deleted from the crystal structure.

We developed an energy-minimization and scoring protocol that addressed all of these consideration, and applied it to both the design model and the X-ray crystal structure to compare the energies. During energy-minimization, the backbone changed by a backbone heavy-atom RMSD of 0.075 Å for the design and 0.165 Å for the crystal structure; the peptide's position and orientation relative to NDM-1 was negligibly affected. Rosetta calculated scores of -352.35 kcal/mol and -356.74 kcal/mol for the designed configuration and inverted configuration in the crystal structure, respectively, suggesting that had we explored alternative binding modes during validation with sufficiently extensive sampling to sample the inverted configuration, we would have been able to identify it as a more favourable configuration than the designed configuration.

Full scripts for performing the energy-minimization and scoring while maintaining cyclic and metal-bound geometry are shown below. The RosettaScripts xml file should be placed in an **xml/** sub-directory of the working directory:

```
<ROSETTASCRIPTS>
    <!-- This script takes as input the design model for peptide NDM1i-3D or the crystal
    structure of the same and performs some energy minimization.  It then scores the
    structure and writes the score.  This was to allow comparison of the energies of
    the two structures.

    This script was updated for the Rosetta version that was current as of 16 Dec 2020
    (Git SHA1 8ad9c9f1a9359f05c5c863b9e3c9ccfd2b30bc9a).  Weekly releases after this
    point, and Rosetta 3.13, will be able to run it. -->
    <SCOREFXNS>
        <!-- A scoring function with constraints: -->
        <ScoreFunction name="r15_cst" weights="ref2015_cst.wts">
            <Reweight scoretype="chainbreak" weight="20.0" />
        </ScoreFunction>
        <!-- A scoring function without constraints: -->
        <ScoreFunction name="r15" weights="ref2015.wts"/>
    </SCOREFXNS>
    <RESIDUE_SELECTORS>
        <!-- Select the first and last residues of the
        peptide macrocycle: -->
        <Index name="select_pep_end" resnums="240" />
        <Index name="select_pep_start" resnums="233" />
    </RESIDUE_SELECTORS>
    <SIMPLE_METRICS>
        <!-- Store the energy before and after minimization.
        This information will be written out in the PDB
        file produced at the end of the protocol: -->
        <TotalEnergyMetric name="pre_score" scorefxn="r15"
            scoretype="total_score" custom_type="PRE_MIN_SCORE_"
        />
        <TotalEnergyMetric name="post_score" scorefxn="r15"
            scoretype="total_score" custom_type="POST_MIN_SCORE_"
```

```
        />
    </SIMPLE_METRICS>
    <MOVERS>
        <!-- After reading in the PDB file, ensure that there is
        an amide bond connecting the termini of the peptide
        macrocycle: -->
        <DeclareBond name="join_termini"
            res1_selector="select_pep_end" atom1="C"
            res2_selector="select_pep_start" atom2="N"
        />
        <!-- Add the chainbreak variant types to the termini of the
        macrocycle.  This allows the energy function (with the
        chainbreak scoreterm activated) to be used to keep the
        macrocycle closed during energy-minimization: -->
        <ModifyVariantType name="add_chainbreak_1"
            add_type="CUTPOINT_UPPER" residue_selector="select_pep_start"
        />
        <ModifyVariantType name="add_chainbreak_2"
            add_type="CUTPOINT_LOWER" residue_selector="select_pep_end"
        />
        <!-- Automatically detect bonds to the zinc and cadmium
        ions, and add suitable constraints: -->
        <SetupMetalsMover name="setup_metals" />
        <!-- Set up the kinematic relationships in the structure. Note that
        this mover takes a commandline flag specifying the fold tree file, since
        the design model has a fold tree that passes through the cysteine SG atom,
        while the crystal structure model has a fold tree that passes through the
        glutamate OE2 atom: -->
        <AtomTree name="foldtree" fold_tree_file="%%foldtree_file%%" />
        <!-- Energy-minimize the structure: -->
        <MinMover name="minmover" scorefxn="r15_cst" type="linmin_iterated"
            tolerance="0.00000001" bb="true" chi="true" jump="ALL"
        />
    </MOVERS>
    <PROTOCOLS>
        <!-- This section lays out the sequence of events, calling
        previously-defined movers and simple metrics: -->
        <Add mover="join_termini" />
        <Add mover="add_chainbreak_1" />
        <Add mover="add_chainbreak_2" />
        <Add mover="setup_metals" />
        <Add mover="foldtree" />
        <Add metrics="pre_score" />
        <Add mover="minmover" />
        <Add metrics="post_score" />
    </PROTOCOLS>
    <!-- At the end, score the structure using the unconstrained scoring
    function, then write the PDB file: -->
    <OUTPUT scorefxn="r15" />
</ROSETTASCRIPTS>
```

Listing 2.1.7.1: File **xml/minimize_and_score.xml**, containing the RosettaScripts protocol for energy-minimizing the design model or X-ray crystal structure (while preserving the terminal amide bond and bonds to the metal ions) and then scoring it.

The above protocol requires fold tree files which are slightly different for the design model and the X-ray crystal structure. In the former, the directed acyclic graph defining the kinematic relationships passes from NDM1 through the active-site zinc and through the NDM1i-3D D-Cys4 residue's side-chain. In the latter, it passes from NDM1 through the active-site cadmium and through the NDM1i-3D L-Glu8 residue's side-chain. Both fold tree

files are listed below, and these should be placed in the **inputs/** sub-directory of the working directory:

```
FOLD_TREE
EDGE 209 1 -1
EDGE 209 229 -1
EDGE 209 230 1
EDGE 209 231 2
EDGE 209 232 3
JEDGE 231 237 4 ZN SG INTRA_RES_STUB
EDGE 237 233 -1
EDGE 237 240 -1
```

Listing 2.1.7.2: File **inputs/foldtree_design.xml**, defining the kinematic relationships in the design model.

```
FOLD_TREE
EDGE 209 1 -1
EDGE 209 229 -1
EDGE 209 230 1
EDGE 209 231 2
EDGE 209 232 3
JEDGE 231 237 4 CD OE2 INTRA_RES_STUB
EDGE 237 233 -1
EDGE 237 240 -1
```

Listing 2.1.7.3: File **inputs/foldtree_xtal.xml**, defining the kinematic relationships in the X-ray crystal structure.

Two Rosetta flags files are also required. These should also be placed in the **inputs/** sub-directory:

```
-in:file:s inputs/design_NDM1i_3D.pdb
-in:file:fullatom
-parser:protocol xml/minimize_and_score.xml
-script_vars foldtree_file=inputs/foldtree_design.txt
```

Listing 2.1.7.4: File **inputs/design.flags**, listing the commandline flags for energy-minimizing and scoring the design model.

```
-in:file:s inputs/xtal_NDM1i_3D_trimmed.pdb
-in:file:fullatom
-parser:protocol xml/minimize_and_score.xml
-script_vars foldtree_file=inputs/foldtree_xtal.txt
```

Listing 2.1.7.5: File **inputs/xtal.flags**, listing the commandline flags for energy-minimizing and scoring the X-ray crystal structure.

Input PDB files should also be placed in the **inputs/** directory. To run this script on the two models, the command is as follows:

```
<path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease
       \ @inputs/design.flags &&
       \ <path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease
       \ @inputs/xtal.flags
```

Listing 2.1.7.6: Commandline command to run the energy-minimization and scoring script on both the design model and the X-ray crystal structure.

As before, **<path to Rosetta>** should be replaced with the user's path to their Rosetta installation directory, and **.default.linuxgccrelease** should be updated as needed for the user's build extras, operating system, compiler, and build mode.

2.2 Software protocols for legacy versions of Rosetta

2.2.1 Legacy code for early designs

In late 2013, prior to creating the designs reported here, we synthesized ten early 7-residue cyclic peptides, which were designed with two Rosetta C++ pilot applications. The first application, **design_truecycpeptide_fromstub.cc**, used an early kinematic closure protocol (which was later generalized to produce the generalized kinematic closure mover (16)) for initial loop closure, carried out sequence design, then performed a Monte Carlo search of macrocycle conformation and sequence space to refine the design. This was a precursor of the script shown in **Listing 2.1.1.2**. The second, **design_cycpep_MPI.cc**, was developed in 2012 as an attempt to consider all alternative conformations of a macrocycle explicitly during sequence design in order to maximize the energy gap between binding-competent and alternative conformations, as opposed to the current approach of optimizing rotamers in a single binding-competent state and afterwards checking the energy gap through large-scale conformational sampling. Although these approaches yielded no high-affinity NDM-1 inhibitors, they led us to uncover and address major problems with our handling of D-amino acids and cyclic geometry, which in turn led to later successes (8, 16, 17). Although neither application is maintained, both are located in the Rosetta repository (in **source/src/apps/pilot/vmullig/**), and both are available on request should the need arise to resurrect either approach.

2.2.2 Design protocol for NDM1i-1 peptides with Rosetta weekly build 2016.46

NDM1i-1 designs were produced in 2016 with the then-current version of Rosetta. Improvements to the code and changes to the interface mean that the exact scripts used no longer work with the current (2020) version of Rosetta. These original NDM1i-1 design scripts have been tested against Rosetta weekly release 2016.46 (Git SHA1 47669d9d2e9f659f4889dc89ee6305575dd87c2e), compiled on CentOS Linux 7 with GCC 4.8.5. Unless exact reproduction of the original design protocol is desired, the updated design scripts in **Section 2.1.1**, prepared for the current version of Rosetta at the time of writing, are

recommended. Many bugs and efficiency issues have been addressed since the 2016 releases, and the current RosettaScripts interface has been streamlined for non-canonical amino acids.

The input into the process was the file described in **Section 2.1.1** and **Listing 2.1.1.1**: a modified version of Protein Data Bank file **4EXS.pdb** with chain A and solvent removed and the L-captopril small molecule converted to a D-cysteine-L-proline dipeptide. This file, called **4EXS_Dcys_Lpro.pdb**, was placed in an **inputs/** sub-directory of the working directory. We then ran the RosettaScripts script shown below to convert this to an octapeptide by prepending three glycine residues and appending three glycine residues:

```
<ROSETTASCRIPTS>
        # This script takes as input PDB structure 4EXS in which the L-captopril has been
        # converted to a D-cysteine, L-proline dipeptide stub.  It then extends this stub,
        # prepending three glycine residues and appending three glycine residues.  Finally, it
        # sets the omega torsion angles of all of these residues to 180 degrees.
        #
        # This script was tested against Rosetta weekly release 2016.46 (Git SHA1
        # 47669d9d2e9f659f4889dc89ee6305575dd87c2e).  It is NOT RECOMMENDED if the newer
        # scripts for Rosetta 3.13 can be used instead, and is provided ONLY for scientific
        # reproducibility!

    # Scorefunctions are defined in this section.  These are primarily energy functions,
    # though they can be modified or enhanced with additional scoring terms for particular
    # tasks.
    <SCOREFXNS>
        <bnv weights="beta_nov15.wts" />
        <bnv_cst weights="beta_nov15_cst.wts" />
        <bnv_highhbond_cst weights="beta_nov15_cst.wts" >
            <Reweight scoretype=hbond_sr_bb weight=10.0 />
            <Reweight scoretype=hbond_lr_bb weight=10.0 />
            <Reweight scoretype=hbond_bb_sc weight=5.0 />
            <Reweight scoretype=hbond_sc weight=3.0 />
            <Reweight scoretype=fa_elec weight=2.0 />
        </bnv_highhbond_cst>
    </SCOREFXNS>

    # Movers are declared in this section.  These modify a structure in some way.
    <MOVERS>
        # The PeptideStubMover appends or prepends residues to an existing structure.
        <PeptideStubMover name=extend >
            <Prepend anchor_rsd=234 resname="GLY" />
            <Prepend anchor_rsd=234 resname="GLY" />
            <Prepend anchor_rsd=234 resname="GLY" />
            <Append anchor_rsd=238 resname="GLY" />
            <Append anchor_rsd=239 resname="GLY" />
            <Append anchor_rsd=240 resname="GLY" />
        </PeptideStubMover>

        # The AtomTree mover is used to define the kinematic relationships in the structure,
        # and to indicate which sections will move and which will remain fixed when a bond is
        # rotated.
        <AtomTree name=foldtree1 fold_tree_file="inputs/foldtree1.txt" />

        # The SetTorsion mover sets mainchain torsions to user-defined values.
        <SetTorsion name=initialize_tors>
            <Torsion residue=234 torsion_name=omega angle=180.0 />
            <Torsion residue=235 torsion_name=omega angle=180.0 />
            <Torsion residue=236 torsion_name=omega angle=180.0 />
            <Torsion residue=237 torsion_name=phi   angle=-60.0 />
            <Torsion residue=238 torsion_name=psi   angle=-10.0 />
```

```
            <Torsion residue=238 torsion_name=omega angle=180.0 />
            <Torsion residue=239 torsion_name=omega angle=180.0 />
            <Torsion residue=240 torsion_name=omega angle=180.0 />
        </SetTorsion>

        # The DeclareBond mover is used to connect the first and last residues of the
        # newly-extended peptide with an amide bond.  Currently, this bond geometry will be
        # badly distorted.
        <DeclareBond name=connect_termini atom1=N res1=234 atom2=C res2=241
            add_termini=true
        />
    </MOVERS>

    # Previously-defined movers are listed here as a series of steps to perform to define a
    # protocol.
    <PROTOCOLS>
        <Add mover=extend />
        <Add mover=foldtree1 />
        <Add mover=initialize_tors />
        <Add mover=connect_termini />
    </PROTOCOLS>

    # This section indicates the energy function that will be used to return a final score on
    # output.
    <OUTPUT scorefxn=bnv />
</ROSETTASCRIPTS>
```

Listing 2.2.2.1: RosettaScripts XML defining a protocol for initial extension of the D-cysteine-L-proline dipeptide stub to create an octapeptide (file **xml/design_8res_setup.xml**).

This script may be run from the command line using the following command, where **<path_to_Rosetta>** should be replaced by the path to one's **Rosetta/** directory, and **.default.linuxgccrelease** should be updated as needed for one's operating system, compiler, and build:

```
<path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease
       \ -parser:protocol xml/design_8res_setup.xml -beta_nov15 -in:file:s
       \ inputs/4EXS_Dcys_Lpro.pdb -in:file:fullatom -auto_setup_metals
       \ -metals_distance_constraint_multiplier 5.0 -metals_angle_constraint_multiplier 5.0
       \ -symmetric_gly_tables true
```

Listing 2.2.2.2: Command-line command for running RosettaScripts XML script **design_8res_setup.xml**.

This script should execute quickly and produce output in a few seconds. The output file, which has the chain-extended poly-glycine peptide in an open confirmation, should be moved to **inputs/4EXS_Dcys_Lpro_start.pdb**; it becomes the input for the next step. Next, the following RosettaScripts script can be used to carry out chain closure and sequence design.

```
<ROSETTASCRIPTS>
        # This script takes as input the 4EXS structure with an 8-residue peptide in the
        # active site, in an open conformation.  It uses generalized kinematic closure to
        # close the peptide, forming an N-to-C peptide macrocycle.  It then designs the
        # sequence of the peptide, and applies various filters.  If filters pass, it proceeds
        # to carry out a Monte Carlo search of local conformation space, designing at each
        # step, to try to improve the shape complementarity.
```

```
        #
        # This script was tested with legacy Rosetta weekly release 2016.46 (Git SHA1
        # 47669d9d2e9f659f4889dc89ee6305575dd87c2e).  This script is ONLY included for
        # scientific reproducibility.  If your goal is to reproduce the work in this
        # publication precisely, use this script; if you want to use the most up-to-date
        # protocol, use the updated scripts for Rosetta 3.13.

    # In the SCOREFXNS section, we define scoring functions used for design and energy
    # minimization.
    <SCOREFXNS>
        # A basic scoring function, with an added penalty discouraging sequences that promote
        # formation of aspartimide byproducts during peptide synthesis.  Note that at the time
        # this script was written, the energy function used was a beta version of the energy
        # function called "beta_nov15".  It has since been renamed "ref2015", and is the
        # current default energy function in newer versions of Rosetta:
        <bnv weights="beta_nov15.wts" >
            <Reweight scoretype=aspartimide_penalty weight=1.0 />
        </bnv>
        # A "soft" variation of the scoring function with more permissive atomic repulsive
        # potentials, useful for design steps.  This scoring function also activates geometric
        # and amino acid composition constraint terms.
        <bnv_soft weights="beta_nov15_soft.wts">
            <Reweight scoretype=aspartimide_penalty weight=1.0 />
            <Reweight scoretype=atom_pair_constraint weight=1.0 />
            <Reweight scoretype=angle_constraint weight=1.0 />
            <Reweight scoretype=dihedral_constraint weight=1.0 />
            <Reweight scoretype=aa_composition weight=1.0 />
        </bnv_soft>
        # A basic scoring function, with an added penalty discouraging sequences that promote
        # formation of aspartimide byproducts during peptide synthesis.  This version also
        # activates geometric constraint terms:
        <bnv_cst weights="beta_nov15_cst.wts" >
            <Reweight scoretype=aspartimide_penalty weight=1.0 />
        </bnv_cst>
        # A variation on the constrained scoring function with the aspartimide penalty term
        # activated and hydrogen bonding and electrostatic terms upweighted:
        <bnv_highhbond_cst weights="beta_nov15_cst.wts" >
            <Reweight scoretype=aspartimide_penalty weight=1.0 />
            <Reweight scoretype=hbond_sr_bb weight=10.0 />
            <Reweight scoretype=hbond_lr_bb weight=10.0 />
            <Reweight scoretype=hbond_bb_sc weight=5.0 />
            <Reweight scoretype=hbond_sc weight=3.0 />
            <Reweight scoretype=fa_elec weight=2.0 />
        </bnv_highhbond_cst>
        # A variation on the previous scoring function that has the amino acid
        # composition penalty activated as well:
        <bnv_highhbond_aacomp_cst weights="beta_nov15_cst.wts" >
            <Reweight scoretype=aspartimide_penalty weight=1.0 />
            <Reweight scoretype=hbond_sr_bb weight=10.0 />
            <Reweight scoretype=hbond_lr_bb weight=10.0 />
            <Reweight scoretype=hbond_bb_sc weight=5.0 />
            <Reweight scoretype=hbond_sc weight=3.0 />
            <Reweight scoretype=fa_elec weight=2.0 />
            <Reweight scoretype=aa_composition weight=1.0 />
        </bnv_highhbond_aacomp_cst>
    </SCOREFXNS>

    # The RESIDUE_SELECTORS section establishes residue selectors, which are rules for
    # selecting a subset of a structure for other modules to operate upon.
    <RESIDUE_SELECTORS>
        # These selectors select the potential hydrogen bonding partners of residues in the
        # peptide, for backbone hydrogen bond counting.  Note that this has been replaced with
        # the PeptideInternalHbondsMetric and the PeptideInternalHbondsFilter in newer
        # versions of Rosetta:
        <Index name=partners_234 resnums=236-240 />
        <Index name=partners_235 resnums=237-241 />
```

```
        <Index name=partners_236 resnums=234,238-241 />
        <Index name=partners_237 resnums=234-235,239-241 />
        <Index name=partners_238 resnums=234-236,240-241 />
        <Index name=partners_239 resnums=234-237,241 />
        <Index name=partners_240 resnums=234-238 />
        <Index name=partners_241 resnums=235-239 />

        # These residue selectors select the peptide, the original stub residues, or just the
        # D-cysteine residue in the stub:
        <Index name=select_peptide resnums=234-241 />
        <Index name=select_stub resnums=237-238 />
        <Index name=select_stub_dcys resnums=237 />

        # This selector inverts the peptide selection, selecting the target protein:
        <Not name=select_target selector=select_peptide />

        # These selectors select residues based on backbone conformation, selecting residues
        # with negative and positive backbone phi angles, respectively:
        <Phi name=select_neg_phi select_positive_phi=false />
        <Not name=select_pos_phi selector=select_neg_phi />

        # Since the stub residues of the peptide are not designed, this selector selects only
        # the designable subset of positions:
        <Index name=select_design_positions resnums=234-236,239-241 />

        # This selector selects positions within the peptide with positive phi values, which
        # will only be permitted to assume D-amino acid identities:
        <And name=select_D_positions selectors=select_design_positions,select_pos_phi />

        # This selector selects positions within the peptide with negative phi values, which
        # will only be permitted to assume L-amino acid identities:
        <And name=select_L_positions selectors=select_design_positions,select_neg_phi />

        # This selector selects positions within the peptide in backbone bin “A”
        # (corresponding to the left-handed alpha helical region of Ramachandran space:
        <Bin name=select_L_alpha bin_params_file="ABBA.bin_params" bin="A" />

        # This selector selects positions within the peptide in backbone bin “X”, a.k.a.
        # “Aprime” (corresponding to the right-handed alpha helical region of Ramachandran
        # space:
        <Bin name=select_D_alpha bin_params_file="ABBA.bin_params" bin="Aprime" />

        # This selector selects positions within the peptide in backbone bin “B”
        # (corresponding to the left-handed beta strand region of Ramachandran space:
        <Bin name=select_L_beta bin_params_file="ABBA.bin_params" bin="B" />

        # This selector selects positions within the peptide in backbone bin “Y”, a.k.a.
        # “Bprime” (corresponding to the right-handed beta strand region of Ramachandran
        # space:
        <Bin name=select_D_beta bin_params_file="ABBA.bin_params" bin="Bprime" />

        # This selector selects positions on the target near the peptide, which will be
        # allowed to repack during peptide sequence design:
        <Neighborhood name=select_interface resnums=234-241 distance=8.0 />

        # This selector selects positions far from the interface:
        <Not name=not_interface selector=select_interface />

        # This selector selects positions far from the interface which are part of the
        # target, which will be fixed during design:
        <And name=select_target_far_from_interface selectors=select_interface,select_target />


        # This selector selects buried residues:
        <Layer name=select_core select_core=true select_boundary=false select_surface=false />
```

```
        # This selector selects buried positions that are also designable:
        <And name=select_hydrophobic_positions selectors=select_design_positions,select_core
        />

        # This selector selects exposed residues:
        <Not name=select_nonhydrophobic_positions selector=select_hydrophobic_positions />

        # This selector selects exposed positions that are also designable:
        <And name=select_nonhydrophobic_design_positions
            selectors=select_nonhydrophobic_positions,select_design_positions
        />

        # The following four selectors select every combination of (negative phi or positive
        # phi positions) and (buried or exposed positions).
        <And name=select_L_hydrophobic_positions
            selectors=select_hydrophobic_positions,select_L_positions />
        <And name=select_D_hydrophobic_positions
            selectors=select_hydrophobic_positions,select_D_positions />
        <And name=select_L_nonhydrophobic_positions
            selectors=select_nonhydrophobic_design_positions,select_L_positions />
        <And name=select_D_nonhydrophobic_positions
            selectors=select_nonhydrophobic_design_positions,select_D_positions />
    </RESIDUE_SELECTORS>

    # The TASKOPERATIONS section defines task operations, which are rules for controlling the
    # Rosetta packer.  The Rosetta packer optimizes side chain rotamers and carries out
    # sequence design, so task operations define design problems.
    <TASKOPERATIONS>
        # Include the input rotamer, even if it is shifted from a rotamer well, in the set of
        # rotamers allowed at a position:
        <IncludeCurrent name=use_input_rotamer />

        # Sample finer levels of discretization of rotamers:
        <ExtraRotamersGeneric name=extrarot ex1=1 ex2=1 ex3=0 ex4=0 extrachi_cutoff=5 />

        # Prevent repacking of the NDM-1 target protein:
        <OperateOnResidueSubset name=no_repack_target selector=select_target >
            <PreventRepackingRLT />
        </OperateOnResidueSubset>

        # Prevent design of the NDM-1 target protein:
        <OperateOnResidueSubset name=no_design_target selector=select_target >
            <RestrictToRepackingRLT />
        </OperateOnResidueSubset>

        # Prevent repacking of the D-cysteine, L-proline stub within the peptide:
        <OperateOnResidueSubset name=no_repack_stub_dcys selector=select_stub_dcys >
            <PreventRepackingRLT />
        </OperateOnResidueSubset>

        # Prevent design of the D-cysteine, L-proline stub within the peptide:
        <OperateOnResidueSubset name=no_design_stub selector=select_stub >
            <RestrictToRepackingRLT />
        </OperateOnResidueSubset>

        # Apply restrictions on allowed residues at positive phi positions that are exposed.
        # Note that in earlier versions of Rosetta, task operation conventions were different
        # for canonical and non-canonical amino acids.  Task operations could only prohibit
        # canonical types, and could only allow non-canonical types.  Later versions change
        # this convention to make it more uniform.  This script relies on the earlier
        # convention.
        <ReadResfile name=D_design filename="inputs/D_resfile.txt"
            selector=select_D_nonhydrophobic_positions />

        # Apply restrictions on allowed residues at positive phi positions that are buried:
        <ReadResfile name=D_hydrophobic_design filename="inputs/D_resfile_hydrophobic.txt"
```

```
            selector=select_D_hydrophobic_positions />

        # Apply restrictions on allowed residues at negative phi positions that are exposed:
        <ReadResfile name=L_design filename="inputs/L_resfile.txt"
            selector=select_L_nonhydrophobic_positions />

        # Apply restrictions on allowed residues at negative phi positions that are buried:
        <ReadResfile name=L_hydrophobic_design filename="inputs/L_resfile_hydrophobic.txt"
            selector=select_L_hydrophobic_positions />

        # Do not allow repacking of NDM-1 residues far from the peptide:
        <OperateOnResidueSubset name=no_repack_target_far_from_interface
            selector=select_target_far_from_interface >
            <PreventRepackingRLT />
        </OperateOnResidueSubset>
    </TASKOPERATIONS>

    # The FILTERS section defines Rosetta filters, which are Rosetta algorithms to analyze a
    # structure, measure properties of that structure, and accept or reject that structure
    # based on the measured properties:
    <FILTERS>
        # These filters measure shape complementarity of the peptide to the target:
        <ShapeComplementarity name=shape1 min_sc=0.63 min_interface=150 jump=3 />
        <ShapeComplementarity name=shape2 min_sc=0.66 min_interface=150 jump=3 />
        <ShapeComplementarity name=shape3 min_sc=0.66 min_interface=150 jump=3 />

        # This filter is used in early stages to discard sampled peptide conformations in
        # which molecular geometry clashes egregiously with the target:
        <ScoreType name=low_stringency_clash scorefxn=bnv score_type=fa_rep threshold=400 />

        # Rosetta's pairwise-decomposable scoring function is unable to detect cases in which
        # three or more hydrogen bond donors all donate to the same acceptor.  This filter
        # detects and eliminates this pathology:
        <OversaturatedHbondAcceptorFilter name=oversat scorefxn=bnv
            max_allowed_oversaturated=0 consider_mainchain_only=false
        />

        # The following filters are combined in the total_hbonds and total_hbonds_2 filter to
        # count the number of internal backbone hydrogen bonds.  Note that this results in an
        # unduly computationally expensive calculation (with intermediate steps repeated), as
        # well as an inconvenient user interface.  For this reason, it has been replace with
        # the PeptideInternalHbondsFilter in newer versions of Rosetta:
        <HbondsToResidue name=hbond1 scorefxn=bnv partners=0 energy_cutoff=-0.25 bb_bb=true
            sidechain=false backbone=true residue=234 from_other_chains=false
            residue_selector=partners_234 />
        <HbondsToResidue name=hbond2 scorefxn=bnv partners=0 energy_cutoff=-0.25 bb_bb=true
            sidechain=false backbone=true residue=235 from_other_chains=false
            residue_selector=partners_235 />
        <HbondsToResidue name=hbond3 scorefxn=bnv partners=0 energy_cutoff=-0.25 bb_bb=true
            sidechain=false backbone=true residue=236 from_other_chains=false
            residue_selector=partners_236 />
        <HbondsToResidue name=hbond4 scorefxn=bnv partners=0 energy_cutoff=-0.25 bb_bb=true
            sidechain=false backbone=true residue=237 from_other_chains=false
            residue_selector=partners_237 />
        <HbondsToResidue name=hbond5 scorefxn=bnv partners=0 energy_cutoff=-0.25 bb_bb=true
            sidechain=false backbone=true residue=238 from_other_chains=false
            residue_selector=partners_238 />
        <HbondsToResidue name=hbond6 scorefxn=bnv partners=0 energy_cutoff=-0.25 bb_bb=true
            sidechain=false backbone=true residue=239 from_other_chains=false
            residue_selector=partners_239 />
        <HbondsToResidue name=hbond7 scorefxn=bnv partners=0 energy_cutoff=-0.25 bb_bb=true
            sidechain=false backbone=true residue=240 from_other_chains=false
            residue_selector=partners_240 />
        <HbondsToResidue name=hbond8 scorefxn=bnv partners=0 energy_cutoff=-0.25 bb_bb=true
            sidechain=false backbone=true residue=241 from_other_chains=false
            residue_selector=partners_241 />
```

```
        <CombinedValue name=total_hbonds threshold=-2.9>
            <Add filter_name=hbond1 factor=-0.5 />
            <Add filter_name=hbond2 factor=-0.5 />
            <Add filter_name=hbond3 factor=-0.5 />
            <Add filter_name=hbond4 factor=-0.5 />
            <Add filter_name=hbond5 factor=-0.5 />
            <Add filter_name=hbond6 factor=-0.5 />
            <Add filter_name=hbond7 factor=-0.5 />
            <Add filter_name=hbond8 factor=-0.5 />
        </CombinedValue>
        <CombinedValue name=total_hbonds_2 threshold=-2.9>
            <Add filter_name=hbond1 factor=-0.5 />
            <Add filter_name=hbond2 factor=-0.5 />
            <Add filter_name=hbond3 factor=-0.5 />
            <Add filter_name=hbond4 factor=-0.5 />
            <Add filter_name=hbond5 factor=-0.5 />
            <Add filter_name=hbond6 factor=-0.5 />
            <Add filter_name=hbond7 factor=-0.5 />
            <Add filter_name=hbond8 factor=-0.5 />
        </CombinedValue>

        # This filter measures the energy, biased towards hydrogen bonds and electrostatic
        # terms, following a Monte Carlo move:
        <ScoreType name=mc_score scorefxn=bnv_highhbond_aacomp_cst score_type=total_score
            threshold=999999 />

        # This filter is used during the Monte Carlo simulation to compute the value passed
        # to the Metropolis evaluator to determine whether moves pass or fail:
        <CombinedValue name=mc_filter threshold=99999>
            <Add filter_name=mc_score factor=1.0 />
            <Add filter_name=shape2 factor=-100.0 />
        </CombinedValue>
    </FILTERS>

    # The MOVERS section sets up movers, which operate on a structure to alter it in some
    # way.  Some sample conformations, others design sequences, others carry out energy
    # minimization, etc.
    <MOVERS>
        # The AtomTree mover sets up the fold tree, which defines the kinematic relationships
        # between different parts of a structure:
        <AtomTree name=foldtree1 fold_tree_file="inputs/foldtree1.txt" />

        # This SetTorsion mover is used to initialize the torsions for the dihedral angles at
        # the start and end of the D-cysteine-L-proline stub.  It sets these to ideal values
        # and then adds a small random perturbation.  It also ensures that all omega angles
        # are 180 degrees.
        <SetTorsion name=initialize_tors>
            <Torsion residue=234 torsion_name=omega angle=180.0 />
            <Torsion residue=235 torsion_name=omega angle=180.0 />
            <Torsion residue=236 torsion_name=omega angle=180.0 />
            <Torsion residue=237 torsion_name=phi   angle=60.0 />
            <Torsion residue=237 torsion_name=phi   angle="perturb" perturbation_type=gaussian
                perturbation_magnitude=30.0 />
            <Torsion residue=237 torsion_name=psi   angle="perturb" perturbation_type=gaussian
                perturbation_magnitude=5.0 />
            <Torsion residue=237 torsion_name=omega angle="perturb" perturbation_type=gaussian
                perturbation_magnitude=5.0 />
            <Torsion residue=238 torsion_name=phi   angle="perturb" perturbation_type=gaussian
                perturbation_magnitude=5.0 />
            <Torsion residue=238 torsion_name=psi   angle=-10.0 />
            <Torsion residue=238 torsion_name=psi   angle="perturb" perturbation_type=gaussian
                perturbation_magnitude=30.0 />
            <Torsion residue=238 torsion_name=omega angle=180.0 />
            <Torsion residue=239 torsion_name=omega angle=180.0 />
            <Torsion residue=240 torsion_name=omega angle=180.0 />
        </SetTorsion>
```

```
# This SetTorsion mover adds a smaller random perturbation to all backbone degrees of
# freedom of the stub:
<SetTorsion name=perturb_tors>
    <Torsion residue=237 torsion_name=phi   angle="perturb" perturbation_type=gaussian
        perturbation_magnitude=5.0 />
    <Torsion residue=237 torsion_name=psi   angle="perturb" perturbation_type=gaussian
        perturbation_magnitude=1.0 />
    <Torsion residue=237 torsion_name=omega angle="perturb" perturbation_type=gaussian
        perturbation_magnitude=1.0 />
    <Torsion residue=238 torsion_name=phi   angle="perturb" perturbation_type=gaussian
        perturbation_magnitude=1.0 />
    <Torsion residue=238 torsion_name=psi   angle="perturb" perturbation_type=gaussian
        perturbation_magnitude=5.0 />
    <Torsion residue=238 torsion_name=omega angle="perturb" perturbation_type=gaussian
        perturbation_magnitude=1.0 />
</SetTorsion>

# This mover adds a chemical bond between the first and last residues of the peptide.
# It has the side-effect of correcting the placement of oxygen and hydrogen atoms
# flanking the amide bond, and so is used repeatedly to update these placements.
# Note that this is less likely in more recent versions of Rosetta, since the
# cutpoint variant types have been updated to ensure that hydrogen and oxygen
# placement remains reasonable at cutpoints during minimization:
<DeclareBond name=connect_termini atom1=N res1=234 atom2=C res2=241
    add_termini=true
/>

# This mover adds constraints to ensure that the amide bond geometry is preserved
# during energy-minimization.  Note that this can be accomplished more simply now
# using the chainbreak scoreterm, but this was necessary in the 2016 releases of
# Rosetta.
<ConstraintSetMover name=terminal_csts add_constraints=true
    cst_file="inputs/termini.cst" />

# These movers add amino acid composition constraints to the whole peptide, to
# negative- phi positions in the alpha-helix region of Ramachandran space, to
# negative-phi positions in the beta-sheet region of Ramachandran space, to
# positive-phi positions in the right-handed alpha-helix region of Ramachandran
# space, and to positive-phi positions in the right-handed beta-sheet region of
# Ramachandran space, respectively:
<AddCompositionConstraintMover name=global_comp
    filename="inputs/global_preferences.comp" selector="select_design_positions" />
<AddCompositionConstraintMover name=L_alpha_comp
    filename="inputs/L_alpha_preferences.comp" selector="select_L_alpha" />
<AddCompositionConstraintMover name=D_alpha_comp
    filename="inputs/D_alpha_preferences.comp" selector="select_D_alpha" />
<AddCompositionConstraintMover name=L_beta_comp
    filename="inputs/L_beta_preferences.comp" selector="select_L_beta" />
<AddCompositionConstraintMover name=D_beta_comp
    filename="inputs/D_beta_preferences.comp" selector="select_D_beta" />

# The PackRotamersMover optimizes side chain rotamers, permitting structure
# refinement or design.  Here, it is used to design the peptide and to refine
# side chain conformations on the target.  Note that the task_operations line, split
# over three lines here, must be placed on a single line for this script to run:
<PackRotamersMover name=softdesign scorefxn=bnv_soft
    task_operations=use_input_rotamer,no_design_target,no_repack_target,
    no_design_stub,no_repack_stub_dcys,D_design,L_design,D_hydrophobic_design,
    L_hydrophobic_design
/>

# The MinMover relaxes a structure through gradient-descent minimizatoin.  This one
# is configured to minimize only side chains of the peptide.  Note that more recent
# versions of Rosetta permit more robust configuration of the minimizer through
# move map factories:
```

```
        <MinMover name=min1 scorefxn=bnv_cst type=dfpmin tolerance=0.001 bb=0 chi=0 >
            <MoveMap name=min1_mm >
                <Jump number=1 setting=0 />
                <Jump number=2 setting=0 />
                <Jump number=3 setting=0 />
                <Jump number=4 setting=0 />
                <Jump number=5 setting=0 />
                <Jump number=6 setting=0 />
                <Jump number=7 setting=0 />
                <Jump number=8 setting=0 />
                <Jump number=9 setting=0 />
                <Span begin=1 end=999 chi=0 bb=0 />
                <Span begin=234 end=241 chi=1 bb=0 />
            </MoveMap>
        </MinMover>

        # This MinMover permits both side chain and backbone minimization of the peptide, but
        # keeps the rest of the structure rigid:
        <MinMover name=min2 scorefxn=bnv_cst type=dfpmin tolerance=0.001 bb=0 chi=0>
            <MoveMap name=min2_mm >
                <Jump number=1 setting=0 />
                <Jump number=2 setting=0 />
                <Jump number=3 setting=0 />
                <Jump number=4 setting=0 />
                <Jump number=5 setting=0 />
                <Jump number=6 setting=0 />
                <Jump number=7 setting=0 />
                <Jump number=8 setting=0 />
                <Jump number=9 setting=0 />
                <Span begin=1 end=999 chi=0 bb=0 />
                <Span begin=234 end=241 chi=1 bb=1 />
            </MoveMap>
        </MinMover>

        # FastDesign performs alternating rounds of packing and minimization while ramping the
        # repulsive term in the scoring function from an initial low value.  This FastDesign
        # instance is configured to design the peptide and repack the target interface, and to
        # allow peptide side chains and backbone and target side chains to move during energy
        # minimization.  Note that the task_operations line, split over three lines here, must
        # be placed on a single line for this script to run:
        <FastDesign name=fdes repeats=3 scorefxn=bnv_highhbond_aacomp_cst min_type=dfpmin
            task_operations=use_input_rotamer,no_repack_target_far_from_interface,
            no_design_target,no_design_stub,no_repack_stub_dcys,D_design,L_design,
            L_hydrophobic_design,D_hydrophobic_design
        >
            <MoveMap name=fdes_mm >
                <Jump number=1 setting=0 />
                <Jump number=2 setting=0 />
                <Jump number=3 setting=0 />
                <Jump number=4 setting=0 />
                <Jump number=5 setting=0 />
                <Jump number=6 setting=0 />
                <Jump number=7 setting=0 />
                <Jump number=8 setting=0 />
                <Jump number=9 setting=0 />
                <Span begin=1 end=999 chi=1 bb=0 />
                <Span begin=234 end=241 chi=1 bb=1 />
            </MoveMap>
        </FastDesign>

        # A ParsedProtocol encapsulates many previously-defined movers and/or filters into a
        # single mover.  This instance defines a series of steps carried out for every
        # solution found by the initial generalized kinematic closure attempts to close the
        # peptide.  These steps include filtering based on total hydrogen bonds, presence of
        # oversaturated acceptors, and clashes, design and minimization, additional filtering,
        # and a more expensive FastDesign round if minimal shape complementarity requirements
```

```
        # are met after the low-cost steps carried out thus far.  The pattern is to go from
        # inexpensive to expensive computations, deciding whether to discard the attempt or to
        # continue after each step:
        <ParsedProtocol name=genkic_steps>
            <Add filter=oversat />
            <Add filter=total_hbonds />
            <Add filter=low_stringency_clash />
            <Add mover=softdesign />
            <Add mover=min1 />
            <Add mover=connect_termini />
            <Add filter=oversat />
            <Add mover=min2 />
            <Add mover=connect_termini />
            <Add filter=oversat />
            <Add filter=total_hbonds_2 />
            <Add filter=shape1 />
            <Add mover=fdes />
            <Add mover=connect_termini />
            <Add filter=oversat />
            <Add filter=shape2 />
        </ParsedProtocol>

        # Generalized kinematic closure (GeneralizedKIC) allows efficient sampling of the
        # closed conformations of a chain of atoms, with rapid solution of a series of
        # equations to determine values of certain degrees of freedom in order to keep the
        # chain closed.  This instance is configured to perform the initial closure of the
        # peptide macrocycle.  For each closed conformation sampled, the steps listed in the
        # previous ParsedProtocol mover are carried out, and the conformation iis accepted if
        # and only if all steps pass:
        <GeneralizedKIC name=genkic closure_attempts=100 pre_selection_mover="genkic_steps"
            stop_when_n_solutions_found=1 selector="lowest_energy_selector"
            selector_scorefunction="bnv_highhbond_cst"
        >
            <AddResidue res_index=239 />
            <AddResidue res_index=240 />
            <AddResidue res_index=241 />
            <AddResidue res_index=234 />
            <AddResidue res_index=235 />
            <AddResidue res_index=236 />
            <SetPivots res1=239 res2=234 res3=236 atom1="CA" atom2="CA" atom3="CA" />
            <CloseBond atom1="C" res1=241 atom2="N" res2=234 torsion=180 bondlength=1.328685
                angle1=116.2 angle2=121.7 />
            <AddPerturber effect="randomize_alpha_backbone_by_rama" >
                <AddResidue index=234 />
                <AddResidue index=235 />
                <AddResidue index=236 />
                <AddResidue index=239 />
                <AddResidue index=240 />
                <AddResidue index=241 />
            </AddPerturber>
        </GeneralizedKIC>

        # A second GeneralizedKIC mover is used to perturb the closed peptide macrocycle
        # during the Monte Carlo search.  For each perturbation, these steps (checking for
        # oversaturated hbond acceptors, counting total hbonds) are carried out:
        <ParsedProtocol name=genkic_perturb_steps>
            <Add mover=connect_termini />
            <Add filter=oversat />
            <Add filter=total_hbonds_2 />
        </ParsedProtocol>

        # This is the second GeneralizedKIC mover.  This one is configured to add a small,
        # random perturbation to an already-closed peptide macrocycle, ensuring that the
        # macrocycle remains closed after the perturbation.  It is used in the context of a
        # Monte Carlo search of local conformations:
        <GeneralizedKIC name=genkic_perturb closure_attempts=5
```

```
        pre_selection_mover="genkic_perturb_steps" stop_when_n_solutions_found=1
        selector="lowest_rmsd_selector" selector_scorefunction="bnv_highhbond_cst"
    >
        <AddResidue res_index=239 />
        <AddResidue res_index=240 />
        <AddResidue res_index=241 />
        <AddResidue res_index=234 />
        <AddResidue res_index=235 />
        <AddResidue res_index=236 />
        <SetPivots res1=239 res2=234 res3=236 atom1="CA" atom2="CA" atom3="CA" />
        <CloseBond atom1="C" res1=241 atom2="N" res2=234 torsion=180 bondlength=1.328685
            angle1=116.2 angle2=121.7 />
        <AddPerturber effect="perturb_dihedral" >
            <AddAtoms res1=239 atom1="N"  res2=239 atom2="CA" />
            <AddAtoms res1=239 atom1="CA" res2=239 atom2="C"  />
            <AddAtoms res1=240 atom1="N"  res2=240 atom2="CA" />
            <AddAtoms res1=240 atom1="CA" res2=240 atom2="C"  />
            <AddAtoms res1=241 atom1="N"  res2=241 atom2="CA" />
            <AddAtoms res1=241 atom1="CA" res2=241 atom2="C"  />
            <AddAtoms res1=234 atom1="N"  res2=234 atom2="CA" />
            <AddAtoms res1=234 atom1="CA" res2=234 atom2="C"  />
            <AddAtoms res1=235 atom1="N"  res2=235 atom2="CA" />
            <AddAtoms res1=235 atom1="CA" res2=235 atom2="C"  />
            <AddAtoms res1=236 atom1="N"  res2=236 atom2="CA" />
            <AddAtoms res1=236 atom1="CA" res2=236 atom2="C"  />
            <AddValue value=2.5 />
        </AddPerturber>
    </GeneralizedKIC>

    # These movers are used only for debugging, and not in production runs:
    <PDBTrajectoryRecorder name="record_traj" stride=1 filename="traj.pdb" cumulate_jobs=0
        cumulate_replicas=0 />
    <PDBTrajectoryRecorder name="record_traj_accepted" stride=1 filename="accepted.pdb"
        cumulate_jobs=0 cumulate_replicas=0 />

    # This is the series of steps performed as the move in the Monte Carlo search.  First,
    # the stub and the macrocycle are perturbed slightly.  Next, a quick round of design
    # is carried out, followed by side chain minimization.  Then, filters are applied.
    # Then, the backbone is minimized, followed by more filtering.  If all of this
    # passes, the move is accepted or rejected by the Metropolis criterion:
    <ParsedProtocol name=mc_steps>
        Add mover=record_traj_accepted /> #COMMENTED OUT FOR PRODUCTION RUNS.
        <Add mover=perturb_tors />
        <Add mover=genkic_perturb />
        <Add mover=softdesign />
        <Add mover=min1 />
        <Add mover=connect_termini />
        Add mover=record_traj /> #COMMENTED OUT FOR PRODUCTION RUNS.
        <Add filter=oversat />
        <Add mover=min2 />
        <Add mover=connect_termini />
        <Add filter=oversat />
        <Add filter=total_hbonds_2 />
    </ParsedProtocol>

    # This mover actually carries out the Monte Carlo search of local conformation space,
    # executing the series of steps in the previous ParsedProtocol as the moves in the
    # search:
    <GenericMonteCarlo name=mc_search mover_name=mc_steps filter_name=mc_filter trials=500
        temperature=0.5 />

    # FsstRelax performs alternating rounds of side chain packing and minimization,
    # keeping sequence fixed but ramping repulsive terms in the scoring function.  It is
    # used for final structural refinement:
    <FastRelax name=final_frlx repeats=3 scorefxn=bnv_cst min_type=dfpmin
        task_operations=use_input_rotamer,no_repack_target_far_from_interface
```

```
        >
            <MoveMap name=final_frlx_mm >
                <Jump number=1 setting=0 />
                <Jump number=2 setting=0 />
                <Jump number=3 setting=0 />
                <Jump number=4 setting=0 />
                <Jump number=5 setting=0 />
                <Jump number=6 setting=0 />
                <Jump number=7 setting=0 />
                <Jump number=8 setting=0 />
                <Jump number=9 setting=0 />
                <Span begin=1 end=999 chi=1 bb=0 />
                <Span begin=234 end=241 chi=1 bb=1 />
            </MoveMap>
        </FastRelax>
    </MOVERS>

    # The PROTOCOLS section strings together previously-defined movers and filters to
    # construct an overall protocol.  Our overall protocol is to initialize the structure
    # and apply suitable constraints, perform an initial closure of the macrocycle in which
    # an initial sequence is designed, then carry out a Monte Carlo search of the local
    # conformational space, redesigning the macrocycle sequence at each step.  Final
    # relaxation and filtering to measure shape complementarity complete the protocol.
    <PROTOCOLS>
        <Add mover=foldtree1 />
        <Add mover=initialize_tors />
        <Add mover=connect_termini />
        <Add mover=terminal_csts />
        <Add mover=global_comp />
        <Add mover=L_alpha_comp />
        <Add mover=D_alpha_comp />
        <Add mover=L_beta_comp />
        <Add mover=D_beta_comp />
        <Add mover=genkic />
        <Add mover=mc_search />
        <Add mover=fdes />
        <Add mover=final_frlx />
        <Add mover=connect_termini /> #Snap O & H atoms at cutpoint back to ideal positions.
        <Add filter=oversat />
        <Add filter=shape3 />
    </PROTOCOLS>

    # The OUTPUT section defines the scoring function that will be used to produce the final
    # score for the structure.
    <OUTPUT scorefxn=bnv />
</ROSETTASCRIPTS>
```

Listing 2.2.2.3: RosettaScripts design script **xml/NDM1i_1_design_legacy.xml**, used to produce NDM1i-1 peptide designs.

The file above uses as input the foldtree and amino acid composition constraint files listed in **Listings 2.1.1.3** though **2.1.1.8**. In addition, it uses legacy-format resfiles to define the L- and D-amino acid design tasks. These should be placed in the **inputs/** sub-directory of the working directory. They are listed below.

```
PIKAA ADEFHIKLMNPQRSTVWY
start
```

Listing 2.2.2.4: Resfile **inputs/L_resfile.txt**, defining L-amino acid types with which to design.

```
PIKAA FILMPVWY
start
```

Listing 2.2.2.5: Resfile **inputs/L_resfile_hydrophobic.txt**, defining L-amino acid types with which to design at buried positions.

```
EMPTY NC DAL NC DAS NC DGU NC DPH NC DHI NC DIL NC DLY NC DLE NC DME NC DAN NC DPR NC DGN NC
DAR NC DSE NC DTH NC DVA NC DTR NC DTY
start
```

Listing 2.2.2.6: Resfile **inputs/D_resfile.txt**, defining D-amino acid types with which to design. The command EMPTY, now deprecated, clears all designable types before appending D-amino acid types with the now-deprecated NC command. Note that since EMPTY was a non-commutative operation, the order in which this task operation was applied would matter.

```
EMPTY NC DPH NC DIL NC DLE NC DME NC DPR NC DVA NC DTR NC DTY
start
```

Listing 2.2.2.7: Resfile **inputs/D_resfile_hydrophobic.txt**, defining D-amino acid types with which to design at buried positions.

The constraints holding the ends of the macrocycle together during energy minimization, and preserving good amide bond geometry, must be defined in the file **inputs/termini.cst**, shown below.

```
AtomPair N 234 C 241 HARMONIC 1.328685 0.05
Angle CA 241 C 241 N 234 HARMONIC 2.02807246864 0.03
Angle C 241 N 234 CA 234 HARMONIC 2.12406564732 0.03
Dihedral CA 241 C 241 N 234 CA 234 CIRCULARHARMONIC 3.141592654 0.03
```

Listing 2.2.2.8: Constraints file **inputs/termini.cst**, defining constraints to preserve good N-to-C amide bond geometry during energy-minimization of the peptide macrocycle. Four constraints are defined: an atom pair constraint to preserve the amide bond length, two bond angle constraints to preserve the C_α-C-N and C-N-C_α bond angles, and a dihedral constraint to preserve the ω dihedral angle.

Finally, the file **inputs/rosetta.flags** defines command-line flags for running RosettaScripts:

```
# On most clusters with queueing systems, it is simplest to set the number of output
# structures to a large number and to allow Rosetta to write structures until the job times
# out:
-nstruct 1000

# These options control the automatic setup of chemical bonds to metal atoms:
-auto_setup_metals
-metals_distance_constraint_multiplier 5.0
-metals_angle_constraint_multiplier 5.0

# The input PDB file, generated in the previous step:
```

```
-in:file:s inputs/4EXS_Dcys_Lpro_start.pdb

# Since glycine Ramachandran tables used for scoring are based on statistics from the Protein
# Data Bank, they are biased towards glycine residues in positive-phi regions of Ramachandran
# space.  When designing with D-amino acids, the following correction should always be
# applied:
-symmetric_gly_tables true

# Interpret the input as a full-atom model (as opposed to a centroid representation):
-in:file:fullatom

# On output, write records of all connected atoms to facilitate visualization in PyMol:
-write_all_connect_info

# The input XML script to execute:
-parser:protocol xml/NDM1i_1_design_legacy.xml

# Do not exit with error status if some jobs fail (e.g. due to filters failing):
-jd2:failed_job_exception false

# In 2016, the energy function that would later become the default ref2015 energy function
# was known as "beta_nov15".  This flag activates that energy function:
-beta_nov15

# The following line should be uncommented for production runs, to prevent writing of a large
# output log:
#-mute all
```

Listing 2.2.2.9: File **inputs/rosetta.flags**, defining command-line flags for running RosettaScripts.

The script shown in **Listing 2.2.2.3** may now be run with the following command. As before, **<path_to_Rosetta>** should be replaced with the path to the Rosetta directory, and **.default.linuxgccrelease** may need to be updated for the user's build, operating system, and compiler:

```
<path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default_linuxgccrelease
       \ @inputs/rosetta.flags
```

Listing 2.2.2.10: Command-line command to execute the RosettaScripts executable and to run the design script.

2.2.3 Design protocol for NDM1i-3 peptides with Rosetta weekly build 2018.19

NDM1i-3 peptides were designed in 2018 with Rosetta weekly release 2018.19. The scripts in this section are meant *only* for full scientific reproducibility, and have been tested against this release of Rosetta (Git SHA1 44710f97b20e0d4310140acc1e62dc36a249f03f). This legacy version of Rosetta includes small bugs that have since been corrected, and lacks certain more efficient modules such as the **PeptideInternalHbondsFilter**. The updated scripts for the current version of Rosetta, listed in **Section 2.1.2**, are recommended unless the user's objective is exact reproduction of the protocol used to produce the NDM1i-3 peptides.

```
<ROSETTASCRIPTS>
       # This script is used to design NDM1i-3 peptides, which include exotic non-canonical
       # amino acids in the palette of allowed amino acid types.  It was tested with
```

```
    # Rosetta weekly build 2018.19 (Git SHA1 44710f97b20e0d4310140acc1e62dc36a249f03f).
    #
    # Note that this script is provided for full scientific reproducibility.  It is
    # recommended to sue the modernized script for the current Rosetta release unless
    # exact reproduction of the 2018 protocol, bugs and all, is desired

# Scoring functions are defined in this section:
<SCOREFXNS>
    # The default enegy function for Rosetta:
    <ScoreFunction name="r15" weights="ref2015.wts" />

    # A variant of the default energy function with constraint terms activated,
    # and the chainbreak term (which preserves amide bond geometry during
    # energy-minimization) upweighted.
    <ScoreFunction name="r15_cst" weights="ref2015_cst.wts" >
        <Reweight scoretype="metalbinding_constraint" weight="1.0" />
        <Reweight scoretype="chainbreak" weight="25.0" />
    </ScoreFunction>

    # A version of the energy function used for design.  This activates a number
    # of design-centric guidance terms, including the voids_penalty term (which
    # penalizes holes or voids in cores or interfaces), the hbnet term (which
    # encourages hydrogen bond networks), the netcharge term (which allows control
    # over the net charge of the peptide), and the buried_unsatisfied_penalty term
    # (which adds a penalty for buried hydrogen bond donors and acceptors that
    # are not involved in a hydrogen bond):
    <ScoreFunction name="r15_cst_voids" weights="ref2015_cst.wts" >
        <Reweight scoretype="metalbinding_constraint" weight="1.0" />
        <Reweight scoretype="voids_penalty" weight="1.0" />
        <Reweight scoretype="hbnet" weight="1.0" />
        <Reweight scoretype="hbond_sr_bb" weight="10.0" />
        <Reweight scoretype="hbond_lr_bb" weight="10.0" />
        <Reweight scoretype="hbond_bb_sc" weight="5.0" />
        <Reweight scoretype="hbond_sc" weight="3.0" />
        <Reweight scoretype="netcharge" weight="1.0" />
        <Reweight scoretype="aa_composition" weight="1.0" />
        <Reweight scoretype="aspartimide_penalty" weight="1.0" />
        <Reweight scoretype="chainbreak" weight="25.0" />
        <Reweight scoretype="buried_unsatisfied_penalty" weight="0.5" />
    </ScoreFunction>

    # A variation of the above energy function, used for scoring the structure when
    # selecting top structures from the generalized kinematic closure step.
    <ScoreFunction name="r15_cst_voids_scoring" weights="ref2015_cst.wts" >
        <Reweight scoretype="metalbinding_constraint" weight="1.0" />
        <Reweight scoretype="voids_penalty" weight="1.0" />
        <Set voids_penalty_energy_disabled_except_during_packing="false" />
        <Reweight scoretype="hbnet" weight="1.0" />
        <Reweight scoretype="netcharge" weight="1.0" />
        <Reweight scoretype="aa_composition" weight="1.0" />
        <Reweight scoretype="aspartimide_penalty" weight="1.0" />
        <Reweight scoretype="chainbreak" weight="25.0" />
        <Reweight scoretype="buried_unsatisfied_penalty" weight="0.05" />
    </ScoreFunction>
</SCOREFXNS>

# This section defines residue selectors, which select sets of residues in a structure
# based on user-defined rules.
<RESIDUE_SELECTORS>
    # Select the peptide and the hinge loop, all of which will be permitted to
    # move during energy minimization:
    <Index name="movable_backbone" resnums="24-32,235-242" />

    # Select the hinge loop and the portion of the peptide that will be
    # weakly constrained to prevent major motions during energy minimization:
    <Index name="constrained_bb" resnums="24-32,236-241" />
```

```
# Select the peptide:
<Index name="peptide" resnums="235-242" />

# Select the target:
<Not name="not_peptide" selector="peptide" />

# Select the positions in the peptide for which the amino acid identity
# is not fixed:
<Index name="peptide_designable_residues" resnums="235-237,240,242" />

# Select positions that can be D-amino acids (positive-phi region of
# Ramachandran space):
<Phi name="positive_phi" select_positive_phi="true" />

# Select positions that can be L-amino acids (negative-phi region of
# Ramachandran space):
<Not name="negative_phi" selector="positive_phi" />

# Select designable positions that can be D-amino acids:
<And name="designable_positive_phi"
    selectors="peptide_designable_residues,positive_phi" />

# Select designable positions that can be L-amino acids:
<And name="designable_negative_phi"
    selectors="peptide_designable_residues,negative_phi" />

# Select positions for which the amino acid identity is fixed:
<Not name="non_designable_residues" selector="peptide_designable_residues" />

# Select target positions that are close to the peptide in space.  (Also
# selects the peptide):
<Neighborhood name="near_peptide" distance="8" selector="peptide" />

# Select target positions that are far from the peptide in space:
<Not name="not_near_peptide" selector="near_peptide" />

# Select target positions that are within 4.5 A of the peptide in space.
# Also selects the peptide:
<Neighborhood name="very_near_peptide" distance="4.5" selector="peptide" />

# Select target positions that are more than 4.5 A from the peptide in
# space:
<Not name="not_very_near_peptide" selector="very_near_peptide" />

# Select positions for which the backbone is movable, or which are close to the
# peptide:
<Or name="movable_sidechains" selectors="movable_backbone,peptide,near_peptide" />

# Select positions which should receive upper cutpoint variant types (within the
# macrocycle and the hinge loop:
<Index name="upper_cutpoints" resnums="28,235" />

# Select positions which should receive lower cutpoint variant types (within the
# macrocycle and the hinge loop:
<Index name="lower_cutpoints" resnums="27,242" />

# Select positions that are buried:
<Layer name="select_buried" select_core="true" select_boundary="false"
    select_surface="false" core_cutoff="2" surface_cutoff="0.1" />

# Select positions that are exposed:
<Not name="select_not_buried" selector="select_buried" />

# Select buried positions within the peptide:
<And name="select_buried_and_peptide" selectors="select_buried,peptide" />
```

```
    # Select exposed positions within the peptide:
    <And name="select_not_buried_and_peptide" selectors="select_not_buried,peptide" />
</RESIDUE_SELECTORS>

# Task operations, which control the Rosetta packer, are defined here:
<TASKOPERATIONS>
    # Only allow residues within 8 A of the peptide to repack or be designed:
    <OperateOnResidueSubset name="only_repack_near_peptide" selector="not_near_peptide" >
        <PreventRepackingRLT />
    </OperateOnResidueSubset>

    # Only allow residues within 4.5 A of the peptide to repack or be designed:
    <OperateOnResidueSubset name="only_repack_very_near_peptide"
        selector="not_very_near_peptide"
    >
        <PreventRepackingRLT />
    </OperateOnResidueSubset>

    # Restrict non-peptide positions to repacking; design only the peptide:
    <OperateOnResidueSubset
        name="only_design_peptide" selector="non_designable_residues"
    >
        <RestrictToRepackingRLT />
    </OperateOnResidueSubset>

    # Include the input rotamer among those to be considered:
    <IncludeCurrent name="include_current" />

    # Set the allowed residue types (D-amino acids) at positive phi positions.
    # Note that this uses the outdated convention for non-canonicals, and will
    # fail for newer versions of Rosetta.  The modernized script should be used
    # instead with newer Rosetta releases:
    <ReadResfile name="resfile_positive_phi" selector="designable_positive_phi"
        filename="inputs/pos_phi.resfile" />

    # Set the allowed residue types (L-amino acids) at negative phi positions.
    # See note above about the legacy resfile syntax and Rosetta versions:
    <ReadResfile name="resfile_negative_phi" selector="designable_negative_phi"
        filename="inputs/neg_phi.resfile" />

    # Allow more finely-sampled rotamers:
    <ExtraRotamersGeneric name="ex1_ex2" ex1="true" ex2="true" extrachi_cutoff="6" />
</TASKOPERATIONS>

# Filters, which analyze structures and reject or accept them based on measured
# properties, are defined here:
<FILTERS>
    # Measure the shape complementarity of the peptide to the binding pocket:
    <ShapeComplementarity name="shape_complementarity" min_sc="0.5" min_interface="400"
        write_int_area="true" residue_selector1="peptide" residue_selector2="not_peptide"
    />

    # Measure the total volume of buried cavities.  (Only used for measurement):
    <CavityVolume name="cavity_volume" />

    # The following lines are used to count internal backbone hydrogen bonds within
    # the peptide.  They are a cumbersome and computationally inefficient way to do
    # this, and have been replaced with the PeptideInternalHbondsFilter and the
    # PeptideInternalHbondsMetric in newer versions of Rosetta.  As configured, the
    # filters will reject any structure in which there are fewer than 2 internal
    # hydrogen bonds.  (Note that each bond is counted twice, so the cutoff is 4).
    <HbondsToResidue name="hbonds_235" partners="0" energy_cutoff="-0.25" bb_bb="true"
        backbone="true" sidechain="false" residue="235"
        scorefxn="r15" from_same_chain="true" from_other_chains="false" />
    <HbondsToResidue name="hbonds_236" partners="0" energy_cutoff="-0.25" bb_bb="true"
```

```
            backbone="true" sidechain="false" residue="236"
            scorefxn="r15" from_same_chain="true" from_other_chains="false" />
        <HbondsToResidue name="hbonds_237" partners="0" energy_cutoff="-0.25" bb_bb="true"
            backbone="true" sidechain="false" residue="237"
            scorefxn="r15" from_same_chain="true" from_other_chains="false" />
        <HbondsToResidue name="hbonds_238" partners="0" energy_cutoff="-0.25" bb_bb="true"
            backbone="true" sidechain="false" residue="238"
            scorefxn="r15" from_same_chain="true" from_other_chains="false" />
        <HbondsToResidue name="hbonds_239" partners="0" energy_cutoff="-0.25" bb_bb="true"
            backbone="true" sidechain="false" residue="239"
            scorefxn="r15" from_same_chain="true" from_other_chains="false" />
        <HbondsToResidue name="hbonds_240" partners="0" energy_cutoff="-0.25" bb_bb="true"
            backbone="true" sidechain="false" residue="240"
            scorefxn="r15" from_same_chain="true" from_other_chains="false" />
        <HbondsToResidue name="hbonds_241" partners="0" energy_cutoff="-0.25" bb_bb="true"
            backbone="true" sidechain="false" residue="241"
            scorefxn="r15" from_same_chain="true" from_other_chains="false" />
        <HbondsToResidue name="hbonds_242" partners="0" energy_cutoff="-0.25" bb_bb="true"
            backbone="true" sidechain="false" residue="242"
            scorefxn="r15" from_same_chain="true" from_other_chains="false" />
        <CombinedValue name="total_hbonds" threshold="-4" >
            <Add filter_name="hbonds_235" factor="-1" />
            <Add filter_name="hbonds_236" factor="-1" />
            <Add filter_name="hbonds_237" factor="-1" />
            <Add filter_name="hbonds_238" factor="-1" />
            <Add filter_name="hbonds_239" factor="-1" />
            <Add filter_name="hbonds_240" factor="-1" />
            <Add filter_name="hbonds_241" factor="-1" />
            <Add filter_name="hbonds_242" factor="-1" />
        </CombinedValue>

        # Require at least 2 hydrogen bonds to residue 237:
        <HbondsToResidue name="hbonds_237_2" partners="2" energy_cutoff="-0.25" bb_bb="true"
            backbone="true" sidechain="false" residue="237"
            scorefxn="r15" from_same_chain="true" from_other_chains="false" />

        # Prohibit more than two hydrogen bonds to each acceptor:
        <OversaturatedHbondAcceptorFilter name="no_oversat_acceptor1"
            max_allowed_oversaturated="0" consider_mainchain_only="true"
            acceptor_selector="peptide" scorefxn="r15" />
        <OversaturatedHbondAcceptorFilter name="no_oversat_acceptor2"
            max_allowed_oversaturated="0" consider_mainchain_only="false"
            acceptor_selector="peptide" scorefxn="r15" />
    </FILTERS>

    # Jump selectors are used to select rigid-body transforms in the fold tree:
    <JUMP_SELECTORS>
        # Jump 1 is the rigid-body transform from the start of the NDM-1 target to the
        # end.  The NDM-1 chain is split into two segments to allow the hinge loop to
        # move with both ends of the loop rooted to unmovable segments of the body of
        # the protein.  To keep the body of the protein fixed, the rigid-body transform
        # from the first residue to the last must be fixed:
        <JumpIndex name="fixed_jumps" jump="1" />

        # All other rigid-body transforms (to the zinc atoms, to the peptide) are
        # permitted to move during energy minimization:
        <Not name="movable_jumps" selector="fixed_jumps" />
    </JUMP_SELECTORS>

    # Move map factories define which parts of a structure can move and which parts are
    # fixed during energy minimization:
    <MOVE_MAP_FACTORIES>
        # Keep the structure fixed except for side chains near the peptide or hinge
        # loop, backbone of the peptide and hing loop, and rigid-body transforms
        # other than the transform relating the first and last residues of the
        # (split) NDM-1 chain:
```

```
    <MoveMapFactory name="frlx_mm_factory" bb="false" chi="false" jumps="false">
        <Backbone residue_selector="movable_backbone" />
        <Chi residue_selector="movable_sidechains" />
        <Jumps jump_selector="movable_jumps" />
    </MoveMapFactory>
</MOVE_MAP_FACTORIES>

# Movers are defined here.  They operate on a structure to alter it in some way:
<MOVERS>
    # This mover adds back the residue that was deleted in order to open up
    # the peptide to experiment with 9mers and 10mers.  Here, we're simply
    # restoring the peptide length to an 8mer:
    <PeptideStubMover name="extend_pep" >
        <Insert resname="GLY" anchor_rsd="241" />
    </PeptideStubMover>

    # This converts position 235 to a glycine to allow unbiased conformational
    # sampling of conformations favoured by both L- and D-amino acids:
    <MutateResidue name="mut_to_gly_235" new_res="GLY" target="235" />

    # The DeclareBond mover is used both to indicate that there is a chemical
    # bond between the first and last residues (preventing the proximity of
    # the N- and C-termini from being interpreted as a clash), and also to
    # update the position of O and H atoms that are dependent on the N-to-C
    # amide bonds following energy minimization.  This is less essential
    # since cutpoints had been updated by 2018.
    <DeclareBond name="declare_bond" atom1="C" atom2="N" res1="242" res2="235" />

    # Since the hinge loop is permitted to move as well, this mover allows the
    # positions of O and H atoms at the cutpoint in the HL to be updated, as
    # a precaution:
    <DeclareBond name="update_loop_O_H" atom1="C" atom2="N" res1="27" res2="28" />

    # Manually introduce the proline->hydroxyproline mutation at position 7.  This
    # position in the crystal structure has the potential to make a favourable
    # hydrogen bonding interaction with the target.
    <MutateResidue name="add_hyp" target="241" new_res="PRO:pro_hydroxylated_case1" />

    # Add cutpoint variants in the hinge loop and in the peptide macrocycle:
    <ModifyVariantType name="upper_cutpoints" add_type="CUTPOINT_UPPER"
        residue_selector="upper_cutpoints" />
    <ModifyVariantType name="lower_cutpoints" add_type="CUTPOINT_LOWER"
        residue_selector="lower_cutpoints" />
    <ParsedProtocol name="add_cutpoint_variants" >
        <Add mover="upper_cutpoints" />
        <Add mover="lower_cutpoints" />
    </ParsedProtocol>

    # Set up the fold tree:
    <AtomTree name="foldtree1" fold_tree_file="inputs/foldtree1.txt" />

    # Auto-detect metal-ligand interactions and set up suitable chemical bonds and
    # constraints to preserve metal geometry:
    <SetupMetalsMover name="setup_metals" metals_detection_LJ_multiplier="1.0" />

    # Add net charge constraints to the peptide macrocycle requiring a net
    # positive charge:
    <AddNetChargeConstraintMover name="require_net_pos_charge"
        filename="inputs/net_positive.charge" selector="peptide" />

    # Add amino acid composition constraints to the peptide macrocycle as a whole,
    # the buried parts of the macrocycle, and the exposed parts of the macrocycle:
    <AddCompositionConstraintMover name="peptide_aa_composition"
        filename="inputs/peptide.comp" selector="peptide" />
    <AddCompositionConstraintMover name="peptide_buried_aa_composition"
        filename="inputs/peptide_buried.comp" selector="select_buried_and_peptide" />
```

```
        <AddCompositionConstraintMover name="peptide_exposed_aa_composition"
            filename="inputs/peptide_surf.comp" selector="select_not_buried_and_peptide" />

        # Carry out a round of design with energy-minimization.  Note that the
        # task_operations must be listed on a single line:
        <FastDesign name="fdes1" repeats="1" scorefxn="r15_cst_voids"
            movemap_factory="frlx_mm_factory" task_operations="resfile_positive_phi,
            resfile_negative_phi,only_repack_very_near_peptide,only_design_peptide"
        />

        # Final relaxation (packing and energy-minimization), keeping amino acid
        # sequence fixed:
        <FastRelax name="frlx1" repeats="1" scorefxn="r15_cst"
            movemap_factory="frlx_mm_factory"
            task_operations="only_repack_near_peptide,include_current,ex1_ex2"
        />

        # Weakly hold the hinge loop and the peptide in place (allowing small motions)
        # during energy minimization:
        <AddConstraints name="add_bb_csts" >
            <CoordinateConstraintGenerator name="gen_csts" sd="1.0" sidechain="false"
                native="false" residue_selector="constrained_bb"
            />
        </AddConstraints>

        # Sample small motions of the hinge loop, perturbing it slightly from its
        # current conformation while ensuring that the loop remains closed:
        <GeneralizedKIC name="KIC_perturb_loop" selector="lowest_rmsd_selector"
            selector_scorefunction="r15_cst" closure_attempts="100"
            stop_when_n_solutions_found="25"
        >
            <AddResidue res_index="24" />
            <AddResidue res_index="25" />
            <AddResidue res_index="26" />
            <AddResidue res_index="27" />
            <AddResidue res_index="28" />
            <AddResidue res_index="29" />
            <AddResidue res_index="30" />
            <AddResidue res_index="31" />
            <AddResidue res_index="32" />
            <SetPivots res1="24" res2="27" res3="32" atom1="CA" atom2="CA" atom3="CA" />
            <CloseBond res1="27" res2="28" atom1="C" atom2="N" bondlength="1.328685"
                angle1="116.199993" angle2="121.699997" torsion="180.0" />
            <AddPerturber effect="perturb_dihedral" >
                <AddAtoms atom1="N" atom2="CA" res1="24" res2="24" />
                <AddAtoms atom1="N" atom2="CA" res1="25" res2="25" />
                <AddAtoms atom1="N" atom2="CA" res1="26" res2="26" />
                <AddAtoms atom1="N" atom2="CA" res1="27" res2="27" />
                <AddAtoms atom1="N" atom2="CA" res1="28" res2="28" />
                <AddAtoms atom1="N" atom2="CA" res1="29" res2="29" />
                <AddAtoms atom1="N" atom2="CA" res1="30" res2="30" />
                <AddAtoms atom1="N" atom2="CA" res1="31" res2="31" />
                <AddAtoms atom1="N" atom2="CA" res1="32" res2="32" />
                <AddAtoms atom1="CA" atom2="C" res1="24" res2="24" />
                <AddAtoms atom1="CA" atom2="C" res1="25" res2="25" />
                <AddAtoms atom1="CA" atom2="C" res1="26" res2="26" />
                <AddAtoms atom1="CA" atom2="C" res1="27" res2="27" />
                <AddAtoms atom1="CA" atom2="C" res1="28" res2="28" />
                <AddAtoms atom1="CA" atom2="C" res1="29" res2="29" />
                <AddAtoms atom1="CA" atom2="C" res1="30" res2="30" />
                <AddAtoms atom1="CA" atom2="C" res1="31" res2="31" />
                <AddAtoms atom1="CA" atom2="C" res1="32" res2="32" />
                <AddValue value="10" />
            </AddPerturber>
            <AddFilter type="loop_bump_check" />
            <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="24" />
```

```
        <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="27" />
        <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="32" />
    </GeneralizedKIC>

    # The series of steps performed for each round of design: carry out FastDesign,
    # update the O and H atoms on the macrocycle and the hinge loop, ensure that the
    # total hydrogen bond count is still over threshold and that residue 237 is
    # satisfied, and ensure that no hbond acceptor is oversaturated:
    <ParsedProtocol name="design_protocol" >
        <Add mover="fdes1" />
        <Add mover="declare_bond" /> # Update O and H atoms in peptide
        <Add mover="update_loop_O_H" />
        <Add filter="total_hbonds" />
        <Add filter="hbonds_237_2" />
        <Add filter="no_oversat_acceptor2" />
    </ParsedProtocol>

    # The series of steps performed for each round of final relaxation.  These are
    # similar to the design steps, but FastRelax replaces FastDesign (so that the
    # sequence does not change):
    <ParsedProtocol name="relax_protocol" >
        <Add mover="frlx1" />
        <Add mover="declare_bond" /> # Update O and H atoms in peptide
        <Add mover="update_loop_O_H" />
        <Add filter="total_hbonds" />
        <Add filter="hbonds_237_2" />
    </ParsedProtocol>

    # The series of steps performed on each closed solution that the peptide
    # macrocycle generalized kinematic closure mover finds.  Each conformation
    # is first subjected to some filtering steps.  The hinge loop is then perturbed,
    # and three rounds of design, followed by three rounds of relaxation, are carried
    # out.  Each round involves filtering steps that can abort the protocol and discard
    # the candidate GenKIC solution before the full computational expense has been
    # invested:
    <ParsedProtocol name="genkic_preselection_steps" >
        <Add filter="total_hbonds" />
        <Add filter="hbonds_237_2" />
        <Add filter="no_oversat_acceptor1" />
        <Add mover="KIC_perturb_loop" />
        <Add mover="design_protocol" />
        <Add mover="design_protocol" />
        <Add mover="design_protocol" />
        <Add mover="relax_protocol" />
        <Add mover="relax_protocol" />
        <Add mover="relax_protocol" />
        <Add filter="shape_complementarity" />
    </ParsedProtocol>

    # Generalized kinematic closure to close the peptide macrocycle.  Only the upper
    # half of the peptide is permitted to move; the lower half preserves the
    # conformation from the NDM1i-1G crystal structure:
    <GeneralizedKIC name="KIC_close_peptide" selector="lowest_energy_selector"
        selector_scorefunction="r15_cst_voids_scoring" closure_attempts="200"
        stop_when_n_solutions_found="1" pre_selection_mover="genkic_preselection_steps"
    >
        <AddResidue res_index="241" />
        <AddResidue res_index="242" />
        <AddResidue res_index="235" />
        <AddResidue res_index="236" />
        <SetPivots res1="241" atom1="CA" res2="242" atom2="CA" res3="236" atom3="CA" />
        <CloseBond res1="242" res2="235" atom1="C" atom2="N" bondlength="1.328685"
            angle1="116.199993" angle2="121.699997" torsion="180.0" />
        <AddPerturber effect="set_dihedral" >
            <AddAtoms res1="241" res2="242" atom1="C" atom2="N" />
            <AddAtoms res1="242" res2="235" atom1="C" atom2="N" />
```

```
            <AddValue value="180.0" />
        </AddPerturber>
        <SampleCisPeptideBond cis_prob="0.2">
            <AddResidue index="241" />
        </SampleCisPeptideBond>
        <AddPerturber effect="randomize_backbone_by_rama_prepro">
            <AddResidue index="241" />
            <AddResidue index="242" />
            <AddResidue index="235" />
        </AddPerturber>
        <AddPerturber effect="perturb_dihedral">
            <AddAtoms atom1="N" atom2="CA" res1="236" res2="236" />
            <AddAtoms atom1="CA" atom2="C" res1="236" res2="236" />
            <AddValue value="10" />
        </AddPerturber>
        <AddFilter type="loop_bump_check" />
        <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="241" />
        <AddFilter type="rama_prepro_check" rama_cutoff_energy="5.0" residue="242" />
        <AddFilter type="rama_prepro_check" rama_cutoff_energy="1.0" residue="236" />
    </GeneralizedKIC>
</MOVERS>

# This section strings together previously-defined movers and filters
# to define the overall protocol that will be applied:
<PROTOCOLS>
    <Add mover="extend_pep" />
    <Add mover="mut_to_gly_235" />
    <Add mover="declare_bond" />
    <Add mover="add_cutpoint_variants" />
    <Add mover="setup_metals" />
    <Add mover="foldtree1" />
    <Add mover="add_hyp" />
    <Add mover="require_net_pos_charge" />
    <Add mover="peptide_aa_composition" />
    <Add mover="peptide_buried_aa_composition" />
    <Add mover="peptide_exposed_aa_composition" />
    <Add mover="add_bb_csts" />
    <Add mover="KIC_close_peptide" />
    <Add filter="shape_complementarity" />
    <Add filter="cavity_volume" />
</PROTOCOLS>

# The final structure is re-scored with the ref2015 energy function:
<OUTPUT scorefxn="r15" />
</ROSETTASCRIPTS>
```

Listing 2.2.3.1: Script **xml/NDM1i_3_design_legacy.xml** for legacy Rosetta weekly release 2018.19 for producing NDM1i-3 designs.

The following resfiles, defining the allowed amino acid types that will be considered during design at each position, must be included in the **inputs/** directory. Note that these differ from the versions in **Listings 2.1.2.2** and **2.1.2.3**, since they use older resfile syntax that has been deprecated since the 2018 Rosetta releases.

```
PIKAA FAILYVWDERKSTNQHP NC AIB NC A12 NC A91 NC ORN NC DAB NC DPP NC NLU NC NVL NC A34 NC A68
    \ NC A94 NC B30 NC B96 NC C27 NC B44 NC B67 NC B74
start
237 E PIKAA FAILYVWP NC AIB NC A12 NC A91 NC NLU NC NVL NC A34 NC A68 NC A94 NC B30 NC B96
    \ NC C27 NC B44 NC B67 NC B74
240 E PIKAA FAILYVWP NC AIB NC A12 NC A91 NC NLU NC NVL NC A34 NC A68 NC A94 NC B30 NC B96
    \ NC C27 NC B44 NC B67 NC B74
```

Listing 2.2.3.2: Resfile **inputs/neg_phi.resfile**, defining allowed amino acid types at positions in the peptide with negative mainchain φ dihedral values (compatible with L-amino acids). Truncated lines are indicated with a backslash (\), and must be placed on a single line in order for this file to function.

```
EMPTY NC AIB NC DPH NC DAL NC DIL NC DLE NC DTY NC DVA NC DTR NC DAS NC DGU NC DAR NC DLY NC
DSE
    \ NC DTH NC DAN NC DGN NC DHI NC DPR NC DA12 NC DA91 NC DORN NC DDAB NC DDPP NC DNLU NC
DNVL
    \ NC DA34 NC DA68 NC DA94 NC DB30 NC DB96 NC DC27 NC DB44 NC DB67 NC DB74
start
237 E EMPTY NC DPH NC DAL NC DIL NC DLE NC DTY NC DVA NC DTR NC DPR NC AIB NC DA12 NC DA91
    \ NC DNLU NC DNVL NC DA34 NC DA68 NC DA94 NC DB30 NC DB96 NC DC27 NC DB44 NC DB67 NC DB74
240 E EMPTY NC DPH NC DAL NC DIL NC DLE NC DTY NC DVA NC DTR NC DPR NC AIB NC DA12 NC DA91
    \ NC DNLU NC DNVL NC DA34 NC DA68 NC DA94 NC DB30 NC DB96 NC DC27 NC DB44 NC DB67 NC DB74
```

Listing 2.2.3.3: Resfile **inputs/pos_phi.resfile**, defining allowed amino acid types at positions in the peptide with positive mainchain φ dihedral values (compatible with D-amino acids). Truncated lines are indicated with a backslash (\), and must be placed on a single line in order for this file to function.

Flags for running RosettaScripts should be placed in **inputs/rosetta.flags**, shown below.

```
# On most clusters with queueing systems, it is simplest to set the number of output
# structures to a large number and to allow Rosetta to write structures until the job times
# out:
-nstruct 1000

# The input RosettaScripts XML file:
-parser:protocol xml/NDM1i_3_design_legacy.xml

# The input PDB file (interpreted as an all-atom representation):
-in:file:s inputs/Moriarty_xtal_chainB_pep_opened.pdb
-in:file:fullatom

# Output PDB files should include CONECT records for all bonds to aid visualization:
-write_all_connect_info

# Since glycine Ramachandran tables used for scoring are based on statistics from the Protein
# Data Bank, they are biased towards glycine residues in positive-phi regions of Ramachandran
# space.  When designing with D-amino acids, the following correction should always be
# applied:
-symmetric_gly_tables true

# Since some jobs are expected to fail due to filters that don't pass, we do not want the
# executable to exit with failure status if there are some failures:
-jd2:failed_job_exception false

# The following line should be uncommented for production runs, to prevent writing of a large
# output log:
#-mute all
```

Listing 2.2.3.4: File **inputs/rosetta.flags**, listing command-line flags for running the RosettaScripts application. The **-auto_setup_metals** flag is omitted since metal setup is carried out with the **SetupMetalsMover**.

The **.comp** and **.charge** files shown in **Listings 2.1.2.4** through **2.1.2.7** should be included in the **inputs/** directory as well. To run this protocol, the following command-line command is used (again replacing **<path_to_Rosetta>** with the path to the Rosetta directory,

and **.default.linuxgccrelease** with the appropriate suffix for the user's build, compiler, and operating system).

```
<path_to_Rosetta>/Rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease
       \ @inputs/rosetta.flags
```

Listing 2.2.3.5: Command-line command to run the RosettaScripts application with the XML file shown in **Listing 2.2.3.1**.

3. Experimental methods

3.1 Peptide synthesis, cyclization, and purification

All peptide macrocycles reported here were synthesized from Fmoc-protected amino acids (P3 BioSystems, Louisville, KY, USA; 2-aminomethyl-L-phenylalanine and β,β-diphenyl-L-alanine from Chem-Impex, Wood Dale, IL, USA) on a Liberty Blue peptide synthesizer (CEM Corporation, Matthews, NC, USA). Coupling and deprotection steps were performed with microwave heating. Those peptides containing an L-aspartate or L-glutamate residue were synthesized on preloaded and side chain-linked Fmoc-L-Asp(Wang resin LL)-ODmab or Fmoc-L-Glu(Wang resin LL)-ODmab resin. Following synthesis of the linear peptide and the final Fmoc deprotection, the terminal Dmab group was removed using treatment with 2% (v/v) hydrazine monohydrate in dimethylformamide (DMF). A final coupling reaction then joined the N- and C-termini on-resin. Side chain deprotection and cleavage from the resin were then carried out using a trifluoroacetic acid (TFA):Water:TIPS:DODT mixture in a ratio of 92.5 : 2.5 : 2.5 : 2.5. TFA was removed by rotary evaporation and lyophilization.

Those peptides that did not contain L-aspartate or L-glutamate were synthesized as linear peptides, tethered by their C-terminus, on Cl-TCP(Cl) resin (CEM Corporation). Cleavage from this resin was carried out without side chain deprotection by treatment with 1% (v/v) TFA in dichloromethane (DCM), and the resulting protected peptide in DCM was mixed with an equal volume of 50:50 acetonitrile:water. DCM and TFA were removed by rotary evaporation, and residual DCM, residual TFA, acetonitrile, and water, by lyophilization. We then redissolved the peptide to 1 mM concentration (crude estimate based on amino acid usage) in DCM. The peptide was cyclized in solution by adding 2 equivalents of (7-Azabenzotriazol-1-yloxy)tripyrrolidinophosphonium hexafluorophosphate (PyAOP), stirring 30 minutes, then adding 10 equivalents of N,N-diisopropylethylamine (DIEA) dropwise and stirring overnight. Rotary evaporation and lyophilization were used to remove solvent.

Crude lyophilized peptides were resuspended in water and purified by reverse-phase high-performance liquid chromatography (HPLC) with an Agilent Zorbax SB-C18 column (9.4 mm X 250 mm). On an Agilent Infinity preparative HPLC system, we used a linear gradient of 1%/min, transitioning from water with 1% TFA to acetonitrile with 1% TFA at a flow rate of 5 mL/min. Elution peaks were detected by absorbance at 214 nm. Using a TSQ Quantum Access

mass spectrometer with electrospray ionization (Thermo Fisher Scientific, Waltham, MA, USA), we confirmed mass and purity of peptides.

3.2 Determination of peptide concentration

Lyophilized peptides were dissolved in 50 mM HEPES, pH 7.2, and tris(2-carboxyethyl)phosphine (TCEP) was added to a concentration of 0.5 mM. Due to the possibility of residual TFA from the peptide synthesis, the pH of each peptide solution was checked using pH indicator strips and was adjusted to ~7 using NaOH when necessary.

In the case of peptides containing a single cysteine residue (NDM1i-1, NDM1i-2, and NDM1i-3 peptides), peptide concentrations were determined using Ellman's assay (19). A standard curve was generated using cysteine hydrochloride monohydrate. Concentrated peptide was first diluted fivefold, tenfold, or twentyfold using reaction buffer (0.1 M sodium phosphate, pH 8, with 1 mM ethylenediaminetetraacetic acid [EDTA]) before a second 1 in 25 dilution was carried out by addition of 5 µL diluted peptide to 120 µL reaction buffer. The fully diluted peptide solution was added to 1.25 mL reaction buffer and the reaction was initiated by addition of 25 µL Ellman's Reagent Solution (4 mg 5,5'-dithiobis-(2-nitrobenzoic acid) [DTNB] dissolved in 1 mL reaction buffer). Reactions were allowed to proceed for 15 minutes before the absorbance of the solutions was measured at 412 nm. Concentrations were determined from the average absorbance determined from triplicate reactions. Since NDM1i-4 peptides contained no cysteine, peptide concentrations for this batch were based on masses of lyophilized powder, using the difference of empty vial mass and post-lyophilization peptide-containing vial mass.

3.3 NDM-1 expression and activity assays

The gene encoding residues 27-270 of NDM-1 was cloned into a pET28a vector encoding a thrombin cleavable N-terminal hexahistidine tag. NDM-1 was overexpressed in *E. coli* BL21 cells grown in ZYP-5052 autoinduction media at 37 °C for 3 hours before lowering the temperature to 23 °C and allowing growth to continue overnight (20). Cells were harvested by centrifugation and resuspended in 20 mM HEPES pH 7.5, 500 mM NaCl, and 10% glycerol. Resuspended cells were lysed 2x using an EmsulsiFlex-C5 homogenizer (Avestin). Unlysed cells and debris were pelleted by centrifugation at 185,000×*g* for 30 minutes. The resulting supernatant was loaded onto a 5 mL Ni-NTA Superflow (Qiagen) column pre-equilibrated with 20 mM HEPES pH 7.5, 500 mM NaCl, and 30 mM imidazole. It was washed with 60 mM imidazole, and NDM-1 was eluted with 300 mM imidazole. The purified protein was desalted into 20 mM HEPES pH 7.4 with 200 mM NaCl, and the His-tag was removed by thrombin cleavage overnight. Uncleaved protein was removed by passing the sample over a 5 mL Ni-NTA Superflow column. NDM-1 was further purified using a Superdex 200 10/300 GL column equilibrated in the same buffer used for desalting.

The NDM-1 inhibition assay conditions were adapted from previously published protocols (21). The reaction buffer contained 50 mM HEPES pH 7.2, 0.5 mM TCEP, 0.01% Triton X-100, 0.5 nM NDM-1, 1 µM $ZnSO_4$, and 1 µg/mL bovine serum albumin (BSA).

Peptides were allowed to incubate with NDM-1 in the reaction for 10 minutes before addition of 1.5 μM nitrocefin. Activity was measured for a total of 10 minutes, though the reaction was typically finished after 2-3 minutes. Initial velocities were calculated from the first 120 seconds of the reaction, with absorbance readings being taken every 10 seconds. The absorbance was measured at 492 nm. Blank runs of reaction buffer plus nitrocefin alone were carried out for subtraction of background nitrocefin degradation.

To determine the expression to which to fit the data, we start with the expressions for the initial velocity of the nitrocefin hydrolysis reaction in the absence (**Expression S5**) and presence (**Expression S6**) of a competitive peptide inhibitor (22):

$$v_0 = \frac{v_{max}[nitrocefin]}{K_M + [nitrocefin]} \quad \textbf{(S5)}$$

$$v_i = \frac{v_{max}[nitrocefin]}{K_M\left(1+\frac{[peptide]}{K_I}\right)+[nitrocefin]} \quad \textbf{(S6)}$$

In the expressions above, v_0 and v_i are the uninhibited and inhibited initial rates of hydrolysis, respectively, [*nitrocefin*] and [*peptide*] represent the concentrations of the nitrocefin substrate and the peptide inhibitor, respectively, v_{max} is the reaction rate at infinite substrate concentration, K_M is the Michaelis constant for hydrolysis of nitrocefin catalyzed by NDM-1, and K_I is the peptide's competitive inhibition constant. At a constant concentration of nitrocefin, by substituting v_i = ½ v_{max} and [*peptide*] = IC_{50} , **Expression S6** can be rearranged to yield the Cheng-Prusoff relationship (23) between IC_{50} (the peptide concentration of half-maximal effect) and the inhibition constant K_I:

$$K_I = \frac{IC_{50}}{1+\frac{[nitrocefin]}{K_M}} \quad \textbf{(S7)}$$

Dividing **Expression S6** by **Expression S5** and substituting using **Expression S7**, the fractional inhibition at a given peptide concentration simplifies to:

$$I = \frac{v_i}{v_0} = \frac{1}{1+[peptide]/IC_{50}} \quad \textbf{(S8)}$$

Since an unnormalized signal is measured experimentally, it is convenient to add parameters for the range and baseline of the signal:

$$I = (A - B)\ \frac{1}{1+[peptide]/IC_{50}} + B \quad \textbf{(S9)}$$

This can be rearranged to yield a more familiar form:

$$I = (B - A)\ \frac{1}{1+IC_{50}/[peptide]} + A \quad \textbf{(S10)}$$

Substituting $B = I_{inf}$ and $A = I_0$, representing the signals at infinite and zero peptide concentration, respectively, we obtain the Hill equation (**Expression S11**) modified with range and baseline parameters. Data were fitted to this expression during data collection with Origin 5.0 software. Final fitting was carried out using SciPy.

$$I = \frac{I_{inf} - I_0}{1 + IC_{50}/[peptide]} + I_0 \qquad \textbf{(S11)}$$

The parameters I_{inf}, I_0, and IC_{50} were all found by the fitter. **Listing 3.3.1** shows the Python 3 script used for data fitting. This script requires Numpy, MatPlotLib, and SciPy. Note that the more general form of the Hill equation includes an exponent n, the Hill coefficient, reflecting cooperativity of multiple binding events; in this case, the inhibitor is expected to bind with 1:1 stoichiometry, and the expression with $n = 1$ yielded good fits to the data, consistent with this expectation.

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
import sys
from scipy.optimize import least_squares
from scipy.optimize import curve_fit

# Read binding data:
def load_binding_data( filename ):
	concentrations = []
	activities = []
	activity_error = []
	n = -100
	y0 = -100
	yinf = -100
	IC50 = -100
	nerr = -100
	y0err = -100
	yinferr = -100
	IC50err = -100
	headerconc=-100
	headeravg=-100
	headererr=-100
	if filename != "xxx":
	with open(filename) as curfile:
		lines = curfile.readlines()
	print( "Read " + filename )
	indata=False
	finisheddata=False
	for line in lines:
		if finisheddata == False:
		if indata == False :
			#Assume first line is header.
			indata = True
			linesplit = line.split()
			for i in range(len(linesplit)) :
			if linesplit[i].startswith("[") :
				headerconc = i
			elif linesplit[i].startswith("Avg"):
				headeravg = i
			elif linesplit[i].startswith("StdErr") :
				headererr = i
```

```
                        assert headerconc != -100
                        assert headeravg != -100
                        continue
                else:
                        linesplit = line.split()
                        if( len(linesplit) == 0 ) :
                        indata = False
                        finisheddata = True
                        continue
                        assert len(linesplit) > headerconc, "Could not parse \"" + line + "\".
headerconc=" + str(headerconc)
                        assert len(linesplit) > headeravg, "Could not parse \"" + line + "\".
headeravg=" + str(headeravg)
                        if headererr != -100 :
                        assert len(linesplit) > headererr, "Could not parse \"" + line + "\".
headererr=" + str(headererr)
                        concentrations.append( float(linesplit[headerconc]) )
                        activities.append( float(linesplit[headeravg]) )
                        if headererr != -100 :
                        activity_error.append( float(linesplit[headererr]) )
                else:
                linesplit = line.split()
                if len(linesplit) == 3:
                        if( linesplit[0] == "n" ):
                        n = int(linesplit[1])
                        nerr = float( linesplit [2] )
                        elif( linesplit[0] == "y0"):
                        y0 = float(linesplit[1])
                        y0err = float(linesplit[2])
                        elif( linesplit[0] == "yinf"):
                        yinf = float(linesplit[1])
                        yinferr = float(linesplit[2])
                        elif( linesplit[0] == "IC50"):
                        IC50 = float(linesplit[1])
                        IC50err = float(linesplit[2])

        return ( concentrations, activities, activity_error, (n, nerr), (y0, y0err), (yinf,
yinferr), (IC50, IC50err) )

# Assert that we've been passed four argument: a datafile to fit:
assert len( sys.argv ) == 5, "Expected four arguments: a file to fit to the Hill equation, an
estimated y0, an estimated yinf, and an estimated IC50."

# Read the data:
binding_data = load_binding_data( sys.argv[1] )

# Ensure some tiny uncertainty in all points to avoid divide-by-zero errors during fitting:
for i in range( len( binding_data[2] ) ) :
        if binding_data[2][i] < 0.001 :
        binding_data[2][i] = 0.01

# Function to fit:
hilleq = lambda x, y0, yinf, IC50 : ( y0 - yinf ) / ( 1 + (IC50/ x ) ) + yinf

# Perform the fitting:
IC50 = float( sys.argv[4] )
IC50err = 0
y0 = float( sys.argv[2] )
y0err = 0
yinf = float( sys.argv[3] )
yinferr = 0
n = 1
nerr = 0

x_data = np.array( binding_data[0] )
y_data = np.array( binding_data[1] )
```

```
y_err = np.array( binding_data[2] )
print( x_data )
print( y_data )
print( y_err )

best_fit_vals, covar = curve_fit(hilleq, x_data, y_data, p0=[y0,yinf,IC50], sigma=y_err,
absolute_sigma=True )
print( "best_fit_vals:", best_fit_vals )
assert( len(best_fit_vals) == 3)
y0 = best_fit_vals[0]
yinf = best_fit_vals[1]
IC50 = best_fit_vals[2]

# Getting fit errors.
uncertainty_vals = np.sqrt(np.diagonal(covar))
print( "uncertainty_vals:", uncertainty_vals )
assert( len(uncertainty_vals) == 3 )
y0err = uncertainty_vals[0]
yinferr = uncertainty_vals[1]
IC50err = uncertainty_vals[2]

print( "n\t" + str(n) + "\t" + str(nerr) )
print( "y0\t" + str(y0) + "\t" + str(y0err) )
print( "yinf\t" + str(yinf) + "\t" + str(yinferr) )
print( "IC50\t" + str(IC50) + "\t" + str(IC50err) )

# Plot the data and the fit:
fig, ax = plt.subplots( nrows=1, ncols=1 )
plotmarker = matplotlib.markers.MarkerStyle( marker="o", fillstyle="full" )

ax.set_xlabel( "[peptide] (µM)" )
ax.set_ylabel( "Normalized activity" )
sc = ax.scatter( binding_data[0], binding_data[1], marker=plotmarker, edgecolors="none", s=10,
c='b' )
eb = ax.errorbar( binding_data[0], binding_data[1], yerr=binding_data[2], linestyle="None",
elinewidth=0.5  )
#Construct the best fit curve
xprime = []
interpolation_points = 50
for j in range(len(binding_data[0]) - 1) :
       for k in range(interpolation_points) :
       xprime.append( float(k)/float(interpolation_points) * ( binding_data[0][j+1] -
binding_data[0][j] ) + binding_data[0][j] )
xprime.append( binding_data[0][len(binding_data[0])-1] )
yprime = []
for x in xprime :
       if( x == 0  ) :
       yprime.append( yinf )
       else :
       yprime.append( (y0-yinf)/( 1 + (IC50/x)**n ) + yinf )
ax.plot( xprime, yprime, 'r-', linewidth=0.5 )

# Calculate and plot residuals:
resids = [ yinf - binding_data[1][0] ]
for j in range(1, len(binding_data[0]) ) :
       resids.append( -((y0-yinf)/( 1 + (IC50/ binding_data[0][j] )**n ) + yinf) +
binding_data[1][j] )
ax2 = inset_axes( ax, width="40%", height="25%", loc=1 )
insetplot = ax2.scatter( binding_data[0], resids, marker=plotmarker, edgecolors="none", s=2.0,
c='b' )
ebinsert = ax2.errorbar( binding_data[0], resids, yerr=binding_data[2], linestyle="None",
elinewidth=0.5, c='b' )
ax2.set_ylim( -0.1, 0.1 )
ax2.axhline(0, c='red', linewidth=0.5)

plt.show()
```

Listing 3.3.1: Python 3 script for fitting NDM-1 inhibition data to the Hill equation. This script requires Numpy, MatPlotLib, and SciPy.

3.4 Co-crystallization of NDM1i peptides with NDM-1

3.4.1 Expression of NDM-1 for crystallization

The same construct used for enzymatic assays was expressed in *E. coli* BL21(DE3) grown at 37 °C in M9 minimal media to better control the metal concentrations in the experiment, which had been observed to be required for optimal crystallization. Expression was induced with 0.5 mM isopropyl β-D-1-thiogalactopyranoside (IPTG) at culture OD_{600} of 0.6, and the growth medium was supplemented with 0.5 mM $ZnSO_4$. Cells were grown overnight at 23 °C. The cells were harvested and resuspended in 20 mM HEPES, pH 7.6, 500 mM NaCl, and 20% glycerol supplemented with 10 μg/mL DNase 1. Cells were lysed using a high-pressure homogenizer (Emulsiflex-C3; Avestin) and the insoluble material was removed by ultracentrifugation at 45,000 rpm for 30 min. The crude extract was loaded onto a Ni-Sepharose column equilibrated with buffer A (50 mM HEPES, pH 7.6, 500 mM NaCl, 10% glycerol), washed with buffer A plus 25 mM imidazole and 50 mM imidazole successively, and eluted with buffer B (50 mM HEPES, pH 7.6, 500 mM NaCl, 10% glycerol, 500 mM imidazole). The eluent was desalted using PD-10 column (GE Healthcare) in buffer A supplemented with 20μM $ZnSO_4$ and 2 mM β-mercaptoethanol. The desalted sample was mixed with thrombin for 40 h at 4 °C before further purification by gel filtration (Superdex 200 increase 10/300; GE Healthcare) in buffer containing 20 mM HEPES, pH 7.4, and 200 mM NaCl. Lastly, the peak fractions containing NDM-1 were pooled and concentrated to 36 mg/ml or higher using centricon ultrafiltration devices (Millipore, Bedford, MA, USA).

3.4.2 Crystallization and structure determination

Prior to crystallization, NDM-1 was supplemented with 2.5 mM $CdCl_2$ (final concentration) in order to saturate the second metal-binding site. NDM1i-1F, NDM1i-1G or NDM1i-3D were added to the protein at 5-7 mM final concentration and incubated on ice for 30 mins. Initial crystal hits were obtained at room temperature *via* hanging drop vapor diffusion by mixing equal volumes of NDM-1 with well solution containing 22-25% (w/v) polyethylene glycol (PEG) 2000 monomethyl ether, 0.1 M 2-(N-morpholino)ethanesulfonic acid (MES), pH 6.5, and 200 mM NaCl. Crystals suitable for structure determination were obtained using seeding into the similar conditions with 18% (w/v) polyethylene glycol (PEG) 2000 monomethyl ether. Crystals were cryoprotected with mother liquor supplemented with 25% glycerol and vitrified in liquid nitrogen.

Data for NDM1i-1F and NDM1i-1G were collected on beamline 08ID-1 at the Canadian Light Source, at 100 K using a wavelength of 0.979 Å. Data for NDM1i-3D were collected on beamline 5.0.1 at the Advanced Light Source, at 100 K using a wavelength of 0.977 Å. All data were processed with XDS and AIMLESS software (24, 25) using XIA2 as part of the CCP4 package (26, 27). The structures were phased by molecular replacement using PHASER (28) within the CCP4 package with unliganded NDM-1 (PDB 3SPU) as a search model. Model

building and refinement were carried out using COOT and REFMAC respectively (29, 30). Model validation was carried out with MOLPROBITY (31) with the NDM1i-1F complex having 99.02% and 0.22%, NDM1i-1G complex having 98.69% and 0%, and NDM1i-3D having 98.88% and 0% Ramachandran-favored and outliers, respectively. Data processing, refinement and model statistics are shown in **Table S2**. Structure factors and coordinates for the NDM1i-1F, NDM1i-1G and NDM1i-3D complexes have been deposited in the Protein Data Bank with PDB IDs 6XBE, 6XBF and 6XCI respectively.

	NDM1i-1F	**NDM1i-1G**	**NDM1i-3D**
Data collection			
Space group	P 1	P 1	P 21 21 21
Cell dimensions			
***a*, *b*, *c* (Å)**	42.70, 74.55, 77.31	42.61, 74.19, 76.65	71.17, 73.79, 77.65
α,β,γ (°)	118.11, 99.65, 95.47	95.75, 103.46, 106.43	90.00, 90.00, 90.00
Resolution (Å)	46.72 - 1.9 (1.96 - 1.9)*	46.67 - 2.2 (2.28 - 2.2)	42.76 - 1.65 (1.71 - 1.65)
R_{sym} **or** R_{merge}	0.0564 (0.5084)	0.0982 (0.7976)	0.0542 (0.2065)
CC1/2	0.997 (0.763)	0.992 (0.522)	0.999 (0.903)
I* / σ*I	8.14 (1.22)	6.02 (0.95)	20.81 (2.77)
Completeness (%)	95.00 (87.15)	94.82 (91.71)	95.69 (73.78)
Redundancy	1.9 (1.6)	1.9 (1.8)	5.7 (1.7)
Refinement			
Resolution (Å)	1.9	2.1	1.65
No. reflections	65658 (6059)	41536 (4018)	47715 (3615)
R_{work} / R_{free}	0.189 (0.350) / 0.225 (0.391)	0.201 (0.364) / 0.246 (0.367)	0.147 (0.199) / 0.164 (0.220)
No. atoms			
Protein/peptide	7062	7046	3453
Ligand/ion	12	12	10

Water	466	386	356
***B*-factors**			
Protein	38.3	49.7	15.0
Ligand/ion	29.4	42.6	22.3
Water	40.1	45.9	25.6
R.m.s. deviations			
Bond lengths (Å)	0.014	0.014	0.015
Bond angles (°)	1.73	1.73	1.78

Table S2: Data collection and refinement statistics for X-ray crystallography (molecular replacement).
* Values in parentheses are for highest-resolution shell.

4. Extended results

4.1 IC_{50} and K_I values for all peptides

Since all inhibition measurements in this study were carried out at a constant concentration of the nitrocefin substrate (1.5 μM), we directly compared IC_{50} values instead of K_I values. For a competitive inhibitor, the proportionality of K_I to IC_{50} is given by the Cheng-Prusoff relationship (23) derived previously in **Expression S7.**

Based on a substrate concentration of 1.5 μM and the previously-reported K_M value of 11±2 μM for NDM-1 hydrolysing nitrocefin (32), K_I values can be computed, with errors estimated by conventional uncertainty propagation. **Table S3** shows sequences, computed P_{Near}, experimentally-measured IC_{50}, and K_I values for all peptides reported here:

Peptide	Sequence	P_{Near}	P_{Near} std. error	IC_{50} (μM)	IC_{50} std. error (μM)	K_I (μM)	K_I error (μM)
NDM1i-1A	IcPVqpDk	0.9017	0.0010	11.55	0.85	10.16	1.99
NDM1i-1B	McPVtpDr	0.7593	0.0003	25.62	1.98	22.55	4.45
NDM1i-1C	vcPlArES	0.6428	0.0003	471.49	17.97	414.91	77.08
NDM1i-1D	DKKcPVPl	0.8801	0.0004	12.25	1.12	10.78	2.19
NDM1i-1E	DKKcPVPp	0.8286	0.0025	25.31	1.54	22.27	4.27
NDM1i-1F	rrLcPVPE	0.9584	0.0005	2.64	0.17	2.32	0.45

NDM1i-1G	rrLcPIPE	0.9607	0.0003	1.15	0.09	1.01	0.20
NDM1i-2A	trLcPIPE	0.9627	0.0001	9.21	0.36	8.10	1.51
NDM1i-2B	rrYcPIPE	0.9516	0.0002	2.93	0.18	2.58	0.49
NDM1i-2C	rrLcPLPE	0.9340	0.0025	7.16	0.31	6.30	1.18
NDM1i-2D	rrLcPIPX[AIB]	0.9314	0.0007	2.20	0.16	1.94	0.38
NDM1i-2E	trYcPIPE	0.9580	0.0002	7.38	0.79	6.49	1.37
NDM1i-2F	trLcPLPE	0.9505	0.0001	51.13	7.04	44.99	10.26
NDM1i-2G	trLcPIPX[AIB]	0.9495	0.0001	17.53	0.86	15.43	2.91
NDM1i-2H	rrYcPLPE	0.9271	0.0053	6.05	0.43	5.32	1.04
NDM1i-2I	trYcPLPE	0.9515	0.0001	18.78	2.33	16.53	3.64
NDM1i-2J	rrYcPLPX[AIB]	0.8375	0.0022	1.84	0.10	1.62	0.31
NDM1i-2K	trYcPLPX[AIB]	0.9330	0.0012	22.33	0.95	19.65	3.67
NDM1i-3A	pqX[NLU]cPX[A34]X[HYP]W	0.9679	0.0010	27.87	3.61	24.53	5.48
NDM1i-3B	pqX[NLU]cPX[A34]X[HYP]I	0.9728	0.0002	39.08	6.87	34.39	8.70
NDM1i-3C	pqX[NLU]cPX[A34]X[HYP]K	0.9719	0.0003	129.18	28.28	113.68	32.35
NDM1i-3D	pkX[NLU]cPX[A34]X[HYP]E	0.9700	0.0005	3.14	0.30	2.76	0.57
NDM1i-4A	pPX[B96]WEpnX[B96]	0.8496	0.0011	ND	ND	ND	ND
NDM1i-4B	pPX[B96]X[ORN]EeX[AIB]N	0.8750	0.0042	111.46	30.49	98.08	32.22
NDM1i-4C	pPX[B96]PEenH	0.9116	0.0012	274.74	88.75	241.77	89.62
NDM1i-4D	pPX[B96]X[ORN]EenI	0.8769	0.0021	ND	ND	ND	ND
NDM1i-4E	pPX[B96]PEenN	0.8372	0.0008	ND	ND	ND	ND
NDM1i-4F	pPX[B96]PEqx[DDAB]Q	0.8659	0.0022	173.05	34.90	152.28	41.35
NDM1i-4G	pPX[B96]X[ORN]EdX[AIB]N	0.8393	0.0017	303.12	70.10	266.75	78.47

Table S3: Sequences, computed P_{Near} values, experimentally-measured IC_{50} values, and K_I values for all peptides reported in this study. L-amino acids are shown as uppercase letters, and D-amino acids that are mirror images of canonical L-amino acids as lowercase letters, using one-letter abbreviations. Non-canonical amino acids that are not mirror images of canonical amino acids are listed as "X" or "x" (with the latter representing a D-amino acid) followed by the Rosetta name code shown in brackets. In the case of NDM1i-2 peptides, positions at which mutations have been introduced (from the NDM1i-1G starting point) are underlined. "ND" indicates no data.

4.2 Analysis of NDM1i-2 peptides (variants of peptide NDM1i-1G)

Whereas small-molecule drugs possess finite potential for chemical modification, peptides are extensively mutable. We carried out *in silico* mutational scanning to predict the

tolerance of the top inhibitor, NDM1i-1G, to mutation. We considered mutations to each of the 19 canonical L-amino acids plus the three non-canonical amino acids L-ornithine (L-ORN), L-2,4-diaminobutyric acid (L-DAB), and L-2,3-diaminopropionic acid (L-DPP). We also considered mutations to the corresponding D-amino acids, and to the achiral amino acids glycine and 2-aminoisobutyric acid (AIB). For each mutation considered, we carried out full conformational sampling using the Rosetta **simple_cycpep_predict** protocol, computing the fold propensity metric P_{Near} from the samples (**Fig. S3**). As expected, there is broad predicted positional tolerance to mutation, with mutations that preserve chirality predicted to be tolerated much better than mutations that invert chirality. Mutations to glycine were predicted to disrupt structure at most but not all sequence positions. Interestingly, several mutations that replaced proline with a less conformationally-constrained amino acid were predicted to be well-tolerated, suggesting that the enforcement of so high a proline content during design may not have been necessary.

This analysis considered only the effect of mutation on fold propensity of the peptide, but not on target interaction affinity. We modelled mutations predicted to have small effects on fold propensity to choose four that could plausibly interact favourably with the target for further characterization. The L3Y and I6L mutations were chosen since they introduce alternative bulky hydrophobic residues on the face of the peptide that interacts with the flexible HL, possibly allowing discovery of new interactions missed by the atomic-resolution modelling that did not take into account all possible HL conformations. The r1t and E8AIB mutations are at positions that make few interactions with the target, but which may influence the peptide conformation. We synthesized these four point mutants (peptides NDM1i-2A through 2D) and assayed inhibition of NDM-1 activity as before. We found that although each mutation in isolation is predicted to preserve a high P_{Near} value, all reduce inhibitory activity slightly (**Fig. S4**). This illustrates that although fold stability may be necessary for potent inhibition, it is not sufficient; favourable target interactions are of course critical as well. The r1t mutation resulted in a roughly 8-fold reduction in inhibition (from an IC_{50} value of 1.15±0.09 μM for NDM1i-1G to 9.21±0.36 μM for the r1t mutant NDM1i-2A) despite being at a position showing no interactions with the target. D-threonine is a β-branched amino acid that one would expect to favour β-strand conformations over the right-handed α-helical conformation at position 1. The fact that the r1t mutation did not alter predicted fold propensity considerably but did increase observed IC_{50} suggests that greater weighting on the intrinsic conformational preferences of amino acid residues may be necessary to improve the accuracy of predictions of fold propensity, and the precision when comparing very similar sequences. In contrast, mutation to the conformationally-constrained AIB likely helps to favour the conformation of position 8, and could account for the minimal disruption to the IC_{50} value (2.20±0.16 μM) observed. The hydrophobic face mutants both slightly reduce inhibitory activity, yielding IC_{50} values of 2.93±0.18 μM for the L3Y mutation and 7.16±0.31 μM for the I6L mutation, suggesting that no better binding mode is revealed for these residues when these mutations are made in isolation.

We next considered the combinatorial effects of these mutations, synthesizing four of the six possible double mutants, two of the four triple mutants, and the quadruple mutant (peptides NDM1i-2E through 2K, **Fig. S5**). The conformational mutations r1t and E8AIB (which have limited disruptive effects in isolation) proved to be incompatible, yielding a combined IC_{50} value

of 17.53±0.86 μM (considerably higher than either in isolation). The L3Y/I6L double mutant showed an IC_{50} value of 6.05±0.43 μM, slightly better than I6L in isolation, and this was further rescued by the L3Y/I6L/E8AIB triple mutation (peptide NDM1i-2J), which brought the IC_{50} value down to 1.84±0.10 μM -- slightly better than any of the mutations in isolation, and nearly as low as the 1.15±0.09 μM IC_{50} observed for NDM1i-1G. This benefit was lost in the r1t/L3Y/I6L/E8AIB quadruple mutant, however (peptide NDM1i-2K), which had an IC_{50} value of 22.33±0.95 μM, indicating the clearly nonlinear effects of combining the disruptive r1t mutation with the otherwise-tolerated triple mutant.

Although this small-scale screen of the sequence space near our top inhibitor did not reveal a more potent inhibitor, some tentative conclusions can be drawn. First, the mutability of the peptides allows discovery of many related sequences of comparable inhibitory potency from a computationally-discovered starting point. Second, although computational predictions of folding propensities of peptides NDM1i-1A through 1G, which were computationally designed to maximize favourable interactions with the target, correlated well with observed inhibitory activity, manually-selected mutations that were not optimized for target affinity disrupted this trend (**Fig. S6**). Most manually-selected mutations were worse inhibitors than would be predicted by P_{Near} analysis, but one, the L3Y/I6L/E8AIB triple mutation (peptide NDM1i-2J) showed potent inhibition despite a computed P_{Near} value of only 0.84, suggesting both that there is a role for manual screening in the vicinity of computationally-generated designs, and that the computational methods can potentially be improved further with energy functions that better capture the effects of single amino acid substitutions' effect on conformational preferences. It is possible that one factor in the limited accuracy of this prediction was the fact that AIB's conformational preferences were computed using MD methods, and not drawn from PDB statistics as natural amino acid residues' conformational preferences can be. Improved methods for de novo generation of conformational preferences of non-canonical amino acid residues, possibly using QM methods (33), may improve these predictions in the future.

4.3 Analysis of NDM1i-4 peptides (variants of peptide NDM1i-3D)

We attempted a final round of designs (**Fig. S9**), starting from the alternative binding mode observed in the crystal structure of NDM1i-3D. In this round, based on the diversity of HL conformations in the crystal structures obtained, we decided to allow more extensive *in silico* sampling of the HL conformation. We found that many designs were able to incorporate bulkier hydrophobic residues such as β,β-diphenyl-L-alanine (L-B96), though many of these had lower predicted fold propensity (P_{Near}) values than designs from previous rounds. Curious as to whether larger apolar surfaces could compensate for poorer fold propensity, we synthesized seven designs, designated NDM1i-4A through 4G, that incorporated at least one L-B96 residue. Unfortunately, NDM1i-4A (which incorporated two L-B96 residues) proved to be too insoluble to test, and many of the rest were of marginal solubility. None came close to the inhibition demonstrated by NDM1i-1G or to the D-captopril control, again suggesting that the most reliable strategy for finding high-affinity binders is to focus on peptides with high predicted P_{Near} values.

5. Supplementary figures

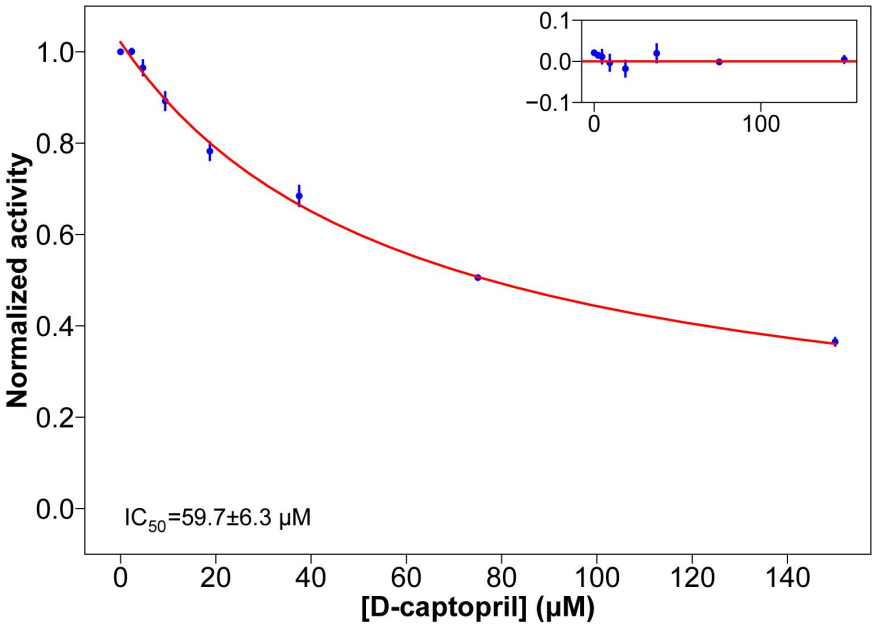

Figure S1: Inhibition of NDM-1 by D-captopril (positive control). Points, error bars, and curve of best fit are as in **Fig. 2**, column **v**.

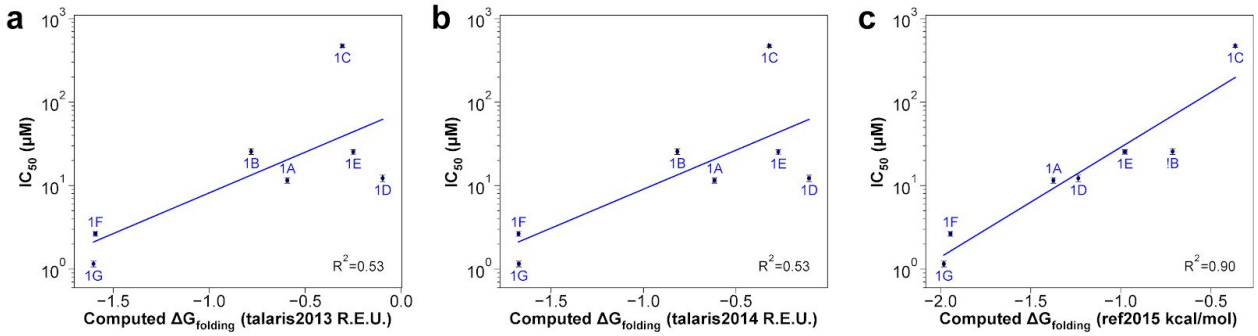

Figure S2: Effect of past improvements to the Rosetta energy function on correlation between computed $\Delta G_{folding}$ and experimentally-measured IC_{50} values. (**a** and **b**) Observed correlation using legacy energy functions talaris2013 and talaris2014, respectively, both of which were optimized against Protein Data Bank (PDB/) structures. R.E.U. represents Rosetta energy units, which did not correspond perfectly to physical units in older versions of the energy function. (**c**) Observed correlation with current default energy function ref2015, optimized using PDB structures and small-molecule fluid MD simulations.

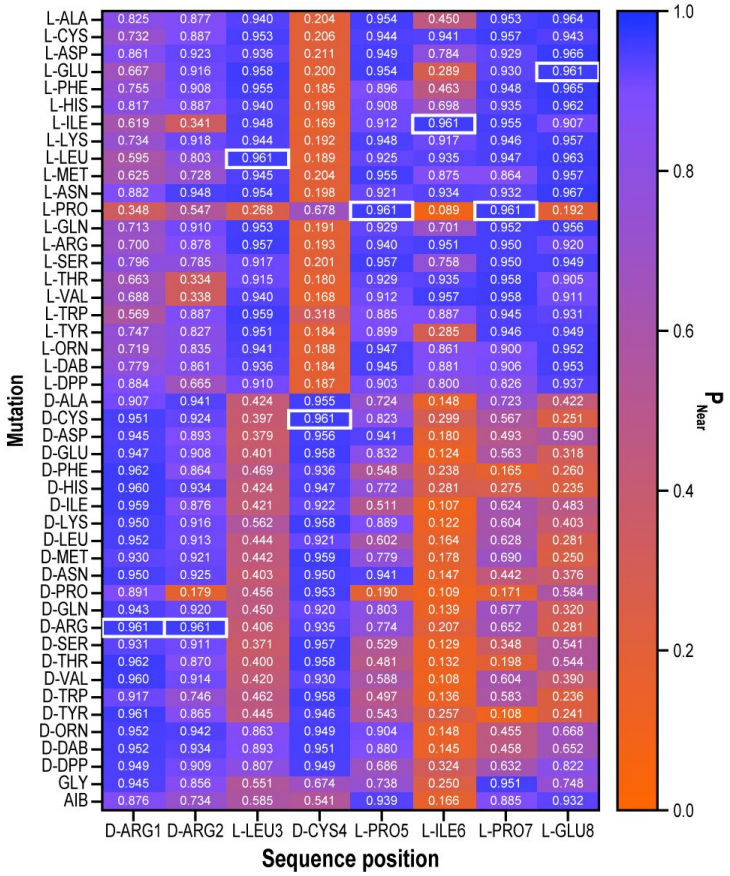

Figure S3: *In silico* mutation scanning of NDM1i-1G, using landscape analysis to identify mutants predicted to preserve the designed conformation. Computed P_{Near} values are shown in white, and mutations are shaded orange to blue to represent P_{Near} values from 0 to 1. White boxes indicate the wild-type sequence. Non-canonical amino acids ornithine (ORN), 2,4-diaminobutyric acid (DAB), 2,3-diaminopropionic acid (DPP), and 2-aminoisobutyric acid (AIB) were included in the analysis, and mirror-image equivalents of all L-amino acids were also included.

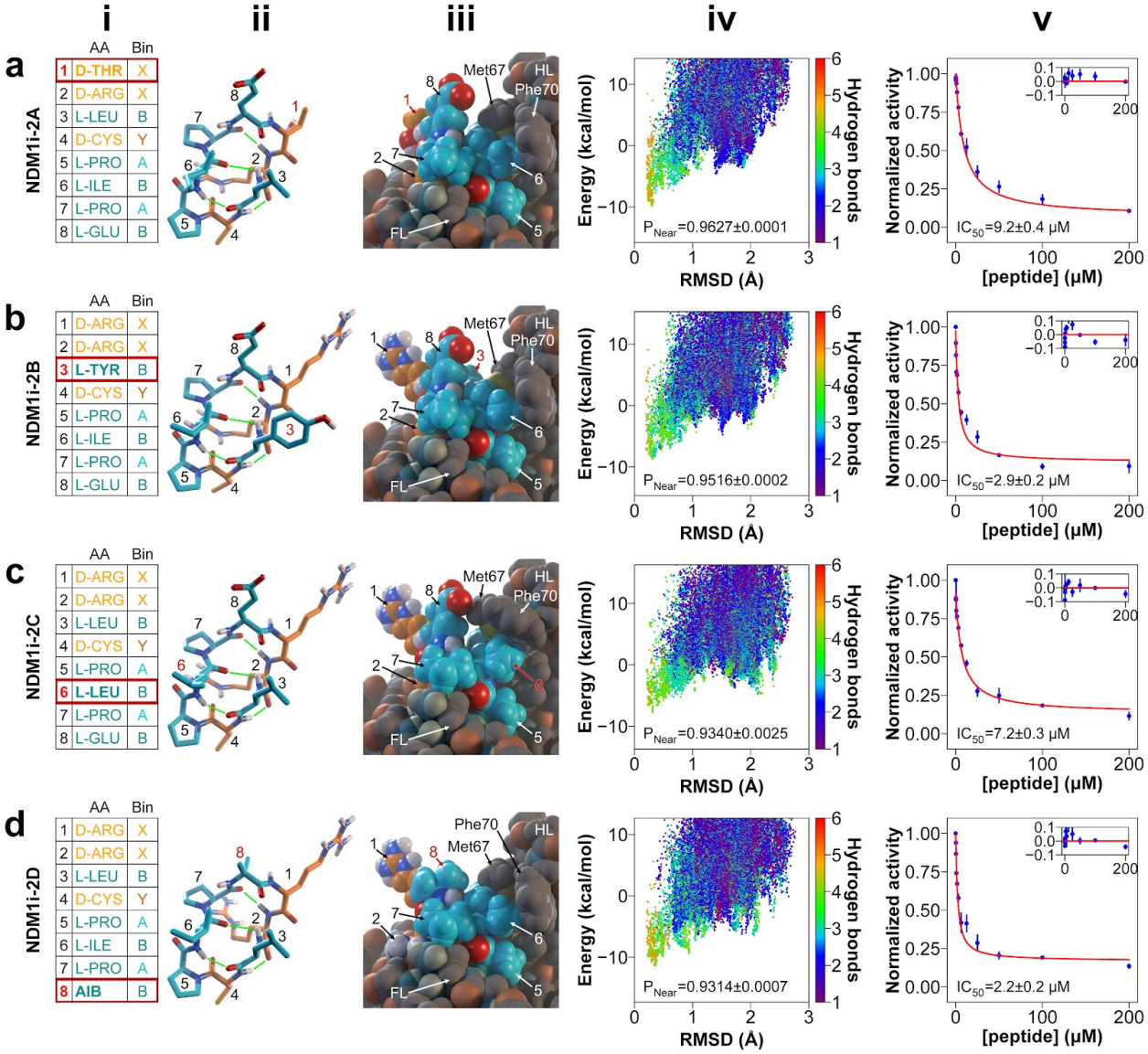

Figure S4: Single point mutations of NDM1i-1G. Columns and annotations are as in **Fig. 2**. Mutations r1t, L3Y, I6L, and E8AIB, designated peptides NDM1i-2A through 2D, are shown in rows **a** through **d**, respectively. Red boxes in column **i**, and red numbers in columns **ii** and **iii**, indicate mutated positions.

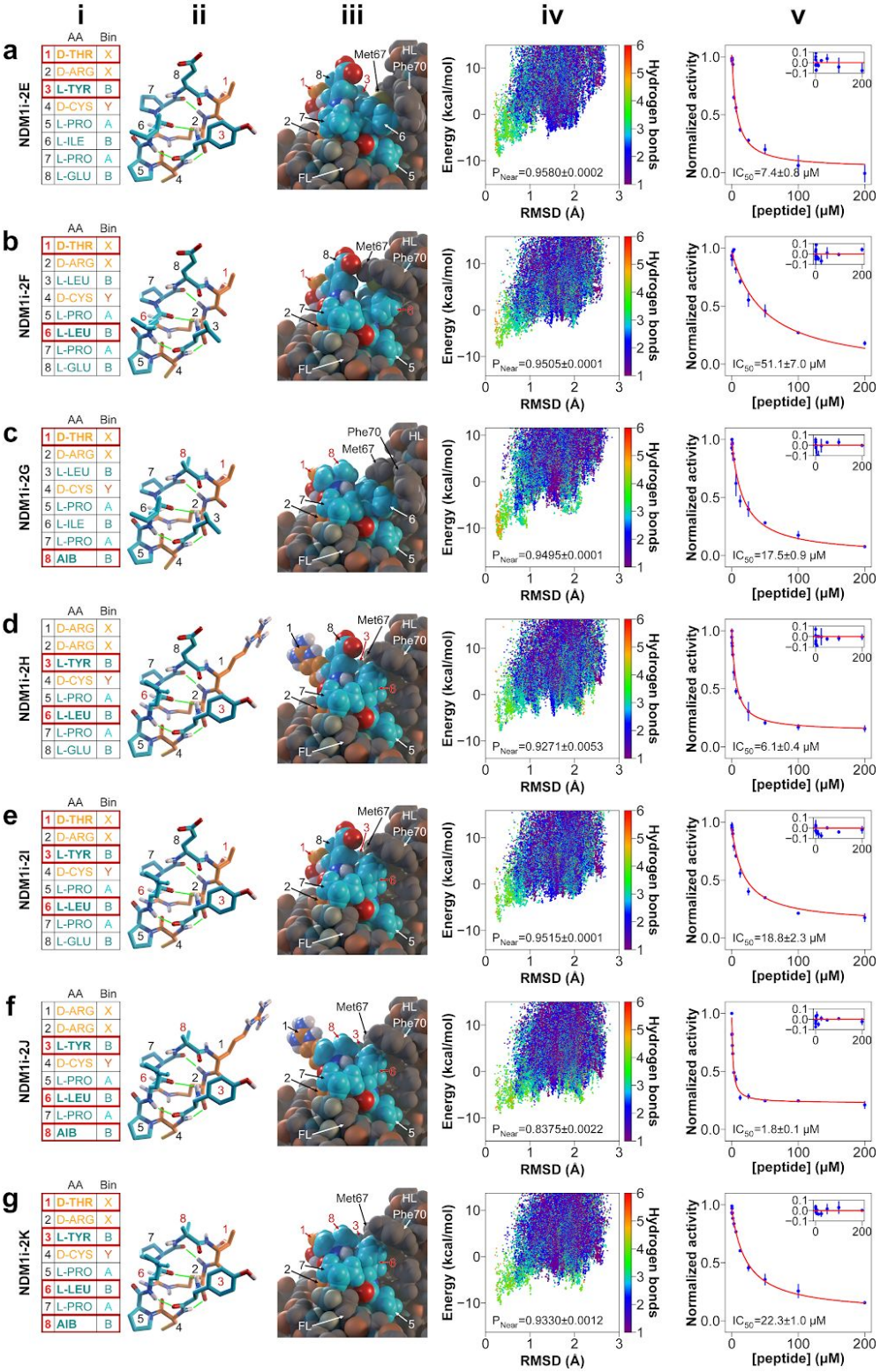

Figure S5: Multiple point mutations of NDM1i-1G. Columns and annotations are as in **Fig. 2** (main text) and **Fig. S4**, with red markings indicating mutated positions. Mutations r1t/L3Y, r1t/I6L, r1t/E8AIB, L3Y/I6L, r1t/L3Y/I6L, L3Y/I6L/E8AIB, and r1t/L3Y/I6L/E8AIB, designated peptides NDM1i-2E through 2K, are shown in rows **a** through **g**, respectively.

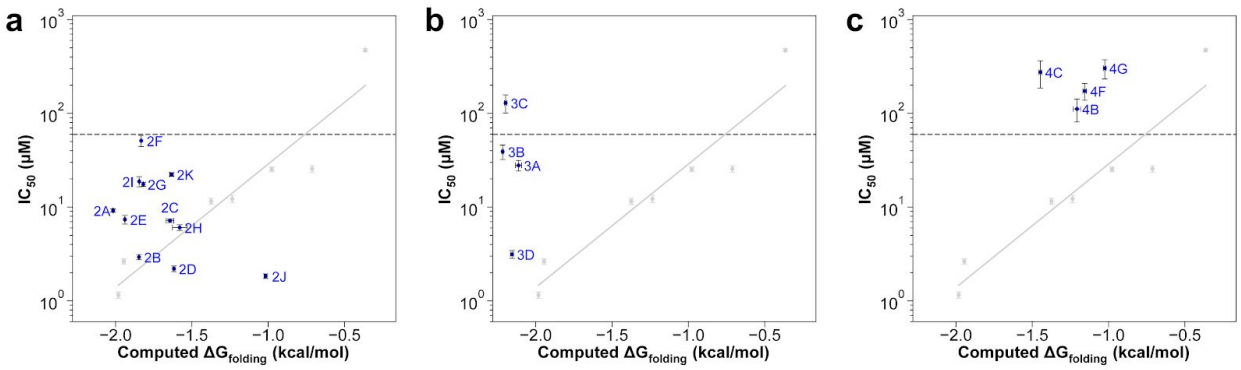

Figure S6: Comparison of observed IC_{50} *vs.* computed $\Delta G_{folding}$ for all designs. In all panels, grey points and the solid grey line represent NDM1i-1 peptides and the line of best fit, respectively. Dashed lines represent the measured IC_{50} value for D-captopril. Horizontal error bars represent the standard error of the mean of three replicates. Vertical error bars represent fitted IC_{50} value uncertainty. (**a**) NDMi1-2 peptides (blue points), which were variants of peptide NDM1i-1G with various combinations of the r15, L3Y, I6L, and E8AIB mutations. (**b**) NDM1i-3 peptides, based on redesign of peptide NDM1i-1G with an expanded palette of amino acid building-blocks. (**c**) NDM1i-4 peptides, based on the crystal structure of the NDM1i-3D peptide bound to NDM-1 in an alternative binding orientation, redesigned with an expanded amino acid palette. In most cases, the line of best fit for the optimized NDM1i-1 peptides provides a lower bound for predicting the IC_{50} of subsequent designs from computed P_{Near} or $\Delta G_{folding}$, but other factors can worsen binding.

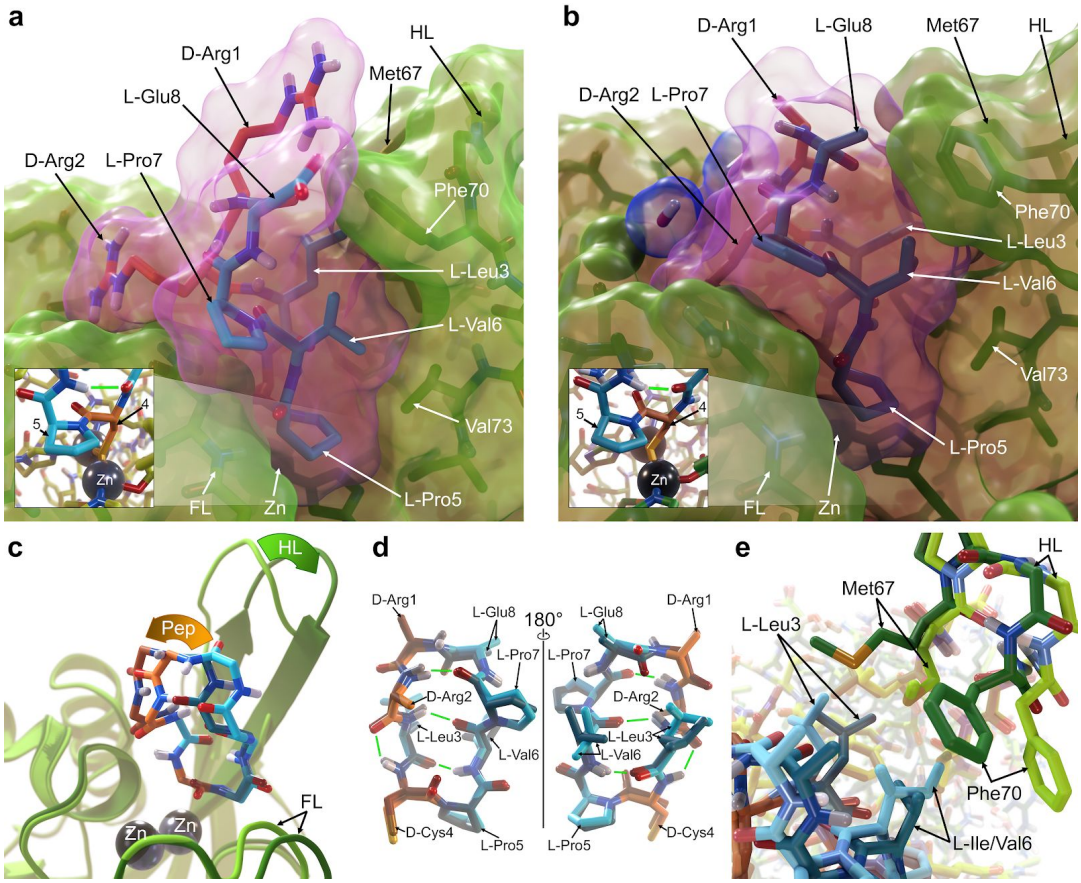

Figure S7: Comparison of design model and X-ray crystal structure (PDB ID 6XBE) of NDM1i-1F bound to NDM-1. (**a**) Design model of NDM1i-1F (pink surface, with D- and L-amino acids in orange and cyan respectively

in this and subsequent panels) bound to NDM-1 (green). HL and FL denote hinge loop and front loop, respectively. Inset: "stub" residues D-Cys4, binding active-site zinc atoms, and L-Pro5. (**b**) X-ray crystal of NDM1i-1F bound to NDM-1. Side chains for residues D-Arg1 and L-Glu8 were not resolved. Bound water molecules are shown as sticks with blue surfaces. Inset: "stub" residues, as in inset for panel **a**. (**c**) Overlay of backbone of design model (lighter colours) and X-ray crystal structure (darker colours). The hinge loop (HL) moves by 2.3 Å, while the peptide moves by 1.3 Å (backbone heavyatom RMSD). Active-site zinc atoms are shown as spheres. (**d**) Overlay of peptide portion of design model and crystal structure, with backbone heavyatoms aligned. Backbone heavyatom RMSD is 0.4 Å. (**e**) Overlay of NDM1i-1F crystal structure (dark colours) with NDM1i-1G crystal structure (light colours). Although the peptides differ by a single methylene group at position 6 (L-Ile in 1G, L-Val in 1F), this results in a considerable rearrangement of side chains of NDM-1 residues Met67 and Phe70.

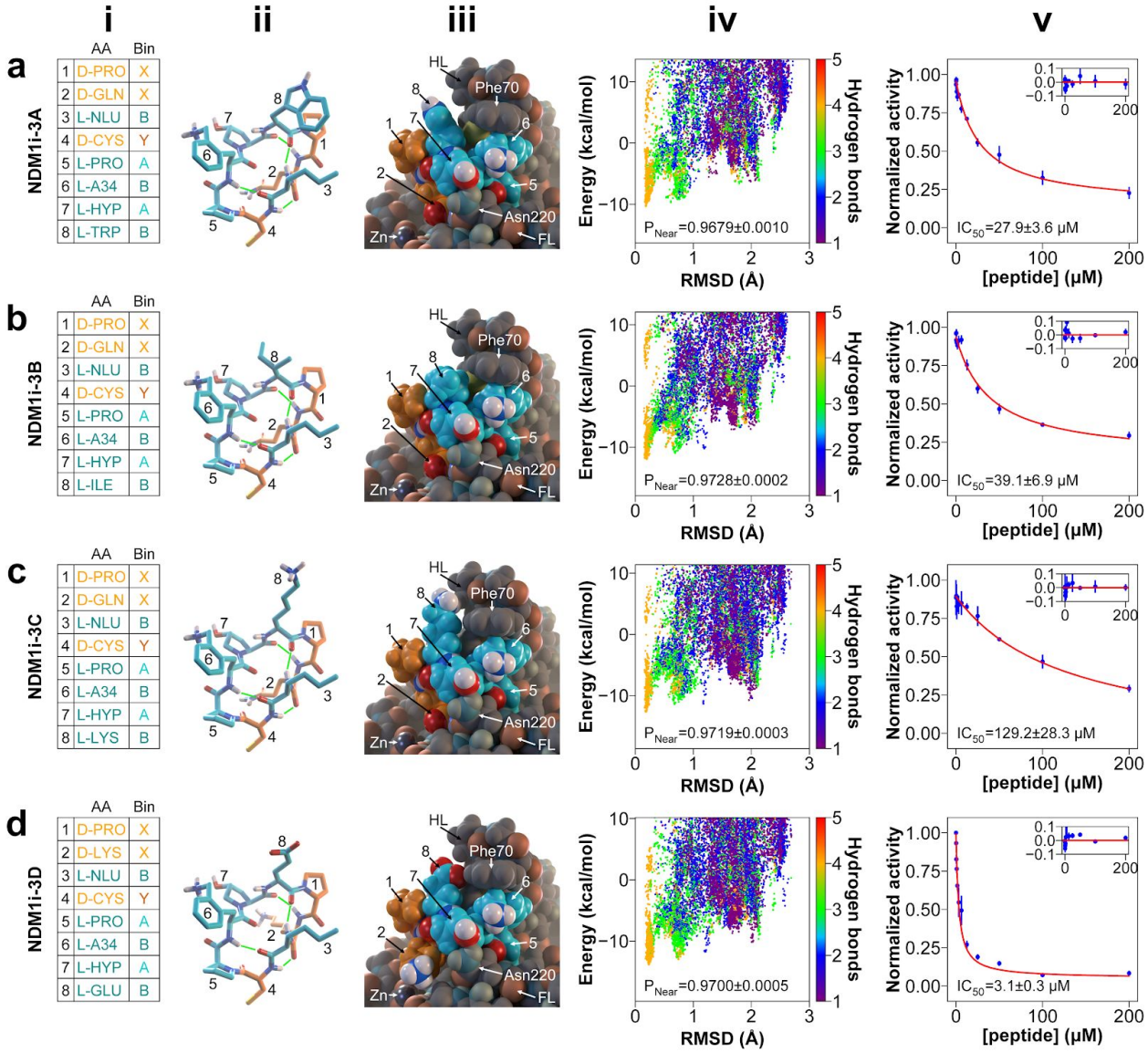

Figure S8: Third-round NDM-1 inhibitor peptides, designed based on the X-ray crystal structure of NDM1i-1G, and incorporating an expanded palette of amino acid building-blocks. All designs incorporate L-norleucine (L-Nlu), L-2-aminomethyl phenylalanine (L-A34), and L-hydroxyproline (L-Hyp) at positions 2, 6, and 7, respectively. Rows **a** through **d** show peptides designated NDM1i-3A through 3D. Columns are as in **Fig. 2**.

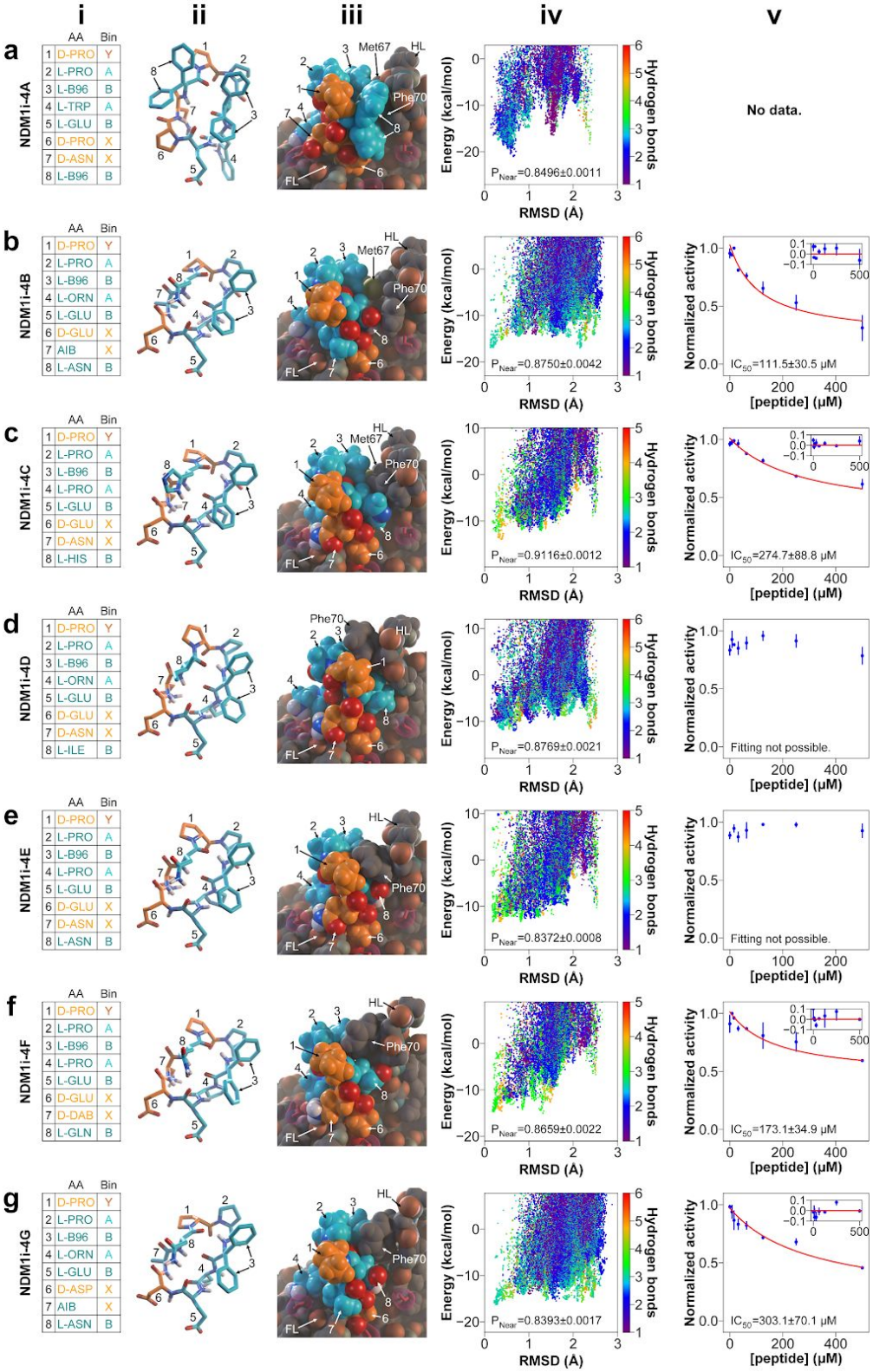

Figure S9: Fourth-round NDM-1 inhibitor peptides. Rows **a** through **g**: peptides NDM1i-4A through NDM1i-4G. Columns are as in **Fig. 2**. Non-canonical amino acids incorporated in this round include ornithine (Orn), 2-aminoisobutyric acid (AIB), and biphenylalanine (B96). In the case of NDM1i-4A, insolubility prevented the inhibition assay. Peptides NDM1i-4D and 4E showed too little inhibitory activity to measure.

6. Supplementary references

1. A. Leaver-Fay, *et al.*, "Rosetta3" in *Methods in Enzymology*, (Elsevier, 2011), pp. 545–574.
2. J. Koehler Leman, *et al.*, Better together: Elements of successful scientific software development in a distributed collaborative community. *PLoS Comput. Biol.* **16**, e1007507 (2020).
3. S. J. Fleishman, *et al.*, RosettaScripts: a scripting language interface to the Rosetta macromolecular modeling suite. *PLoS ONE* **6**, e20161 (2011).
4. V. K. Mulligan, *et al.*, Computational design of mixed chirality peptide macrocycles with internal symmetry. *Manuscript under review* (2020).
5. R. F. Alford, *et al.*, The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design. *Journal of Chemical Theory and Computation* **13**, 3031–3048 (2017).
6. H. Park, *et al.*, Simultaneous Optimization of Biomolecular Energy Functions on Features from Small Molecules and Macromolecules. *Journal of Chemical Theory and Computation* **12**, 6201–6212 (2016).
7. P. D. Renfrew, E. J. Choi, R. Bonneau, B. Kuhlman, Incorporation of Noncanonical Amino Acids into Rosetta and Use in Computational Protein-Peptide Interface Design. *PLoS ONE* **7**, e32637 (2012).
8. P. Hosseinzadeh, *et al.*, Comprehensive computational design of ordered peptide macrocycles. *Science* **358**, 1461–1466 (2017).
9. G. J. Rocklin, *et al.*, Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science* **357**, 168–175 (2017).
10. S. E. Boyken, *et al.*, De novo design of protein homo-oligomers with modular hydrogen-bond network-mediated specificity. *Science* **352**, 680–687 (2016).
11. W. Sheffler, D. Baker, RosettaHoles: Rapid assessment of protein core packing for structure prediction, refinement, design, and validation. *Protein Sci* **18**, 229–239 (2009).
12. W. Sheffler, D. Baker, RosettaHoles2: A volumetric packing measure for protein structure refinement and validation. *Protein Sci* **19**, 1991–1995 (2010).
13. S. L. Guffy, B. S. Der, B. Kuhlman, Probing the minimal determinants of zinc binding with computational protein design. *Protein Eng Des Sel* **29**, 327–338 (2016).
14. J. H. Mills, *et al.*, Computational design of an unnatural amino acid dependent metalloprotein with atomic level accuracy. *J. Am. Chem. Soc.* **135**, 13393–13399 (2013).
15. J. H. Mills, *et al.*, Computational design of a homotrimeric metalloprotein with a trisbipyridyl core. *Proceedings of the National Academy of Sciences* **113**, 15012–15017 (2016).
16. G. Bhardwaj, *et al.*, Accurate de novo design of hyperstable constrained peptides. *Nature* **538**, 329–335 (2016).
17. B. Dang, *et al.*, De novo design of covalently constrained mesosize protein scaffolds with unique tertiary structures. *Proceedings of the National Academy of Sciences* **114**, 10852–10857 (2017).
18. V. K. Mulligan, The emerging role of computational design in peptide macrocycle drug discovery. *Expert Opinion on Drug Discovery* **15**, 833–852 (2020).
19. G. L. Ellman, Tissue sulfhydryl groups. *Archives of Biochemistry and Biophysics* **82**, 70–77

(1959).
20. F. W. Studier, Protein production by auto-induction in high density shaking cultures. *Protein Expr. Purif.* **41**, 207–234 (2005).
21. S. S. van Berkel, *et al.*, Assay Platform for Clinically Relevant Metallo-β-lactamases. *J Med Chem* **56**, 6945–6953 (2013).
22. D. Voet, J. G. Voet, C. W. Pratt, *Fundamentals of Biochemistry Upgrade Edition*, Rev. ed. (Wiley, 2002).
23. Y.-C. Cheng, W. H. Prusoff, Relationship between the inhibition constant (KI) and the concentration of inhibitor which causes 50 per cent inhibition (I50) of an enzymatic reaction. *Biochemical Pharmacology* **22**, 3099–3108 (1973).
24. W. Kabsch, XDS. *Acta Crystallogr D Biol Crystallogr* **66**, 125–132 (2010).
25. P. R. Evans, G. N. Murshudov, How good are my data and what is the resolution? *Acta Crystallogr D Biol Crystallogr* **69**, 1204–1214 (2013).
26. G. Winter, *et al.*, DIALS : implementation and evaluation of a new integration package. *Acta Crystallogr D Struct Biol* **74**, 85–97 (2018).
27. M. D. Winn, *et al.*, Overview of the CCP 4 suite and current developments. *Acta Crystallogr D Biol Crystallogr* **67**, 235–242 (2011).
28. A. J. McCoy, *et al.*, Phaser crystallographic software. *J Appl Crystallogr* **40**, 658–674 (2007).
29. P. Emsley, B. Lohkamp, W. G. Scott, K. Cowtan, Features and development of Coot. *Acta Crystallogr D Biol Crystallogr* **66**, 486–501 (2010).
30. G. N. Murshudov, *et al.*, REFMAC 5 for the refinement of macromolecular crystal structures. *Acta Crystallogr D Biol Crystallogr* **67**, 355–367 (2011).
31. C. J. Williams, *et al.*, MolProbity: More and better reference data for improved all-atom structure validation. *Protein Science* **27**, 293–315 (2018).
32. A. Makena, *et al.*, Biochemical characterization of New Delhi metallo-β-lactamase variants reveals differences in protein stability. *J Antimicrob Chemother* **70**, 463–469 (2015).
33. V. Mironov, Y. Alexeev, V. K. Mulligan, D. G. Fedorov, A systematic study of minima in alanine dipeptide. *J Comput Chem* **40**, 297–309 (2019).