Documentation for the TOPRNA Coarse-Grained Model and *toprnaCreate.pl*

Anthony Mustoe
Hashim Al-Hashimi and Charles Brooks III Groups
University of Michigan
December 2013

Table of Contents

1. Introduction

TOPRNA (TOPological modeling of RNA) is a CHARMM-based coarse-grained model of RNA that uses three pseudo-atoms to represent the base (B), sugar (S), and phosphate (P) moieties of each nucleotide. The model was constructed to explore the hypothesis that topological constraints encoded by RNA secondary structure are significant determinants of 3D structure. Based on a specified secondary structure, TOPRNA simulations generate ensembles of 3D structures that report on the biases that arise from the connectivity and steric properties of the system. This is achieved by modeling RNAs as a collection of rigid body A-form helices linked by freely rotatable single-stranded regions. Electrostatics are ignored, and except for a small attractive van der Waals (VDW) interaction experienced between the base pseudo-atoms of paired nucleotides, all attractive interactions are also ignored.

This document provides technical details needed to run TOPRNA simulations. A complete discussion of the TOPRNA model, including parameterization details, may be found in the publication A.M. Mustoe, H.M. Al-Hashimi, and C.L. Brooks III, *J. Phys. Chem. B, in press* (2014).

Usage requires a good understanding of CHARMM. An overview of CHARMM can be found in the publication Brooks et al., *J. Comp. Chem.* 30, 1545-1615 (2009). Documentation and other information can also be found on the CHARMM website (http://www.charmm.org/). Some useful CHARMM tutorials can be found on the MMTSB website:
http://www.mmtsb.org/workshops/mmtsb-ctbp_workshop_2009/Tutorials/WorkshopTutorials_2009.html

2. TOPRNA usage in CHARMM

TOPRNA is implemented in CHARMM through the use of custom topology and parameter files. The first release versions of these files are named toprna_top1.rtf and toprna_par1.par. Subsequent releases will be named by incrementing numerals. TOPRNA models are compatible with most standard CHARMM functions. However, it should be noted that at the current time only marginal performance gains are observed when TOPRNA simulations are performed using OpenMM GPU acceleration. This is because of the current inability of the CHARMM OpenMM implementation to efficiently handle `NBFIX` energy terms, which feature heavily in the TOPRNA force field. We are hopeful that future versions of the CHARMM will fix this limitation.

2.1 Residue types

TOPRNA supports the four standard nucleotides of adenine, guanine, uridine, and cytosine. Each of these different residues can in turn have six different types, with the type dependent on whether the residue is base paired and on the neighboring sequence context. For example, adenine can have types `A`, `AA`, `AS`, `AAS`, `AL`, and `AAL`. Explanation of these types follows below.

2.1.1 Double and single letter types

A primary distinction between TOPRNA residue types is whether they have one or two letter prefixes (i.e. `A` vs. `AA`; `AS` vs. `AAS`; `AL` vs. `AAL`). In all cases, single and double letter types that share the same suffix have identical properties (i.e. `AS` is the same as `AAS`). The need for different prefixes arises because the RNA backbone is asymmetric; the additional carbon on the 5' side of the sugar ring means that the 5' S-P bond and angles involving this bond differ from those on the 3' side of the S pseudo-atom. Put another way, the intra-residue S-P bond is different than the inter-residue bond. As CHARMM does not have the capacity to treat bonds between the same atom types differently, preserving this asymmetry requires the sequence to have alternating residue types. Thus, a sequence should never be `A A A`, but instead be `A AA A`. Likewise, for single stranded residues, `AS AS AS` is incorrect. Instead it should be `AS AAS AS`. However, two consecutive residues that have different suffixes are allowed. For example, `AS A AS` is allowed sequence.

It is up to the user to ensure that the sequence follows these rules. Incorrect sequences will **not** necessarily result in errors or warnings from CHARMM, but will give incorrect results. One

feature of the *toprnaCreate.pl* utility is that it will automatically generate models that follow these rules.

2.1.2 Paired residue types: `N` and `NN`

Residue types `N` and `NN` (where N is any of the four base types) are reserved for residues participating in base pairs. Importantly, the B pseudo-atoms of `N` and `NN` type residues experience attractive VDW interactions with the B pseudo-atoms of other `N`, `NN`, `NL`, and `NNL` types. This is meant to simulate intra-helical base-stacking, although it should be noted that it also marginally favors inter-helical stacking across junctions. Due to these attractive interactions these types should only be used for residues actively participating in base pairs.

2.1.3 Single stranded residue types: `NS` and `NNS`

`NS` and `NNS` residue types are used for all residues not participating in base pairs. These residues only experience repulsive VDW forces and do not support pairing or backbone dihedral potentials. In addition, these residues possess softer backbone angle and bond potentials, consistent with the increased flexibility of single-stranded nucleotides.

2.1.4 Non-canonical paired residue types: `NL` and `NNL`

`NL` and `NNL` residue types are used for nucleotides in internal loops that participate in non-canonical base pairs. These types are similar to `N` and `NN` types, but with several differences that are meant to model the additional flexibility expected of these nucleotides compared to canonical base pairs. Like `N` and `NN` types, B pseudo-atoms of `NL` and `NNL` types experience attractive VDW forces to other `N`, `NN`, `NL`, and `NNL` types. Canonical pairing is not supported, but these residue types can accept a filler M atom through the `FILL` patch (see 2.3 below). All bond and angle potentials are identical to those of `N` and `NN` types, but the K_φ of backbone dihedral potentials are ¼ the K_φ of `N` and `NN` types. 'Stacking' impropers are not supported for `NL` and `NNL` types (see 2.2.2).

2.2 Implementation of canonical base pairs (AU, GC, and GU)

When a TOPRNA sequence in CHARMM is initialized all basic bond and angle terms are automatically generated and assigned. However, the user must manually specify all base pairs and backbone dihedrals. This is done using CHARMM patches, which are described below. Note that the *toprnaCreate.pl* utility will automatically generate a CHARMM script with the necessary patches and output a PSF file. For large RNAs many patches are required and thus for

simplicity it is recommended that one create the necessary PSF file once and then use this file to initialize subsequent calculations.

2.2.1 PAIR patches

PAIR creates a pairing bond between the two specified residues, establishing the bond and associated angles, impropers, and dihedrals across this bond. The residues must both be either N or NN types, and must both be on the same chain. In addition, the two residues must be either A and U, G and C, or G and U. The PAIR patch also creates a new 'filler' M pseudo-atom that is bonded in parallel to each of the paired B pseudo-atoms. As the M pseudo-atom lacks coordinates it must built using the IC tables before any energy evaluations.

Example (Place pair between residue 1 and 10 of CHNA and use IC tables to initialize M pseudo-atom coordinates):

```
PATCH PAIR CHNA 1 CHNA 10
IC GENE
IC PARA
IC BUILD
```

It is important to note that nucleotides should be apposed to one another and in a near-helical conformation before a PAIR patch is added. If initialized from an incorrect geometry the bond and angle energies will be very high and subsequent dynamics or minimizations are likely to encounter problems. This is typically not an issue when starting from coordinates derived from a pre-existing all-atom structure. However, when creating *de novo* TOPRNA models it is necessary to 'fold' the system into its secondary structure using backbone dihedral and NOE restraints before applying the patch. *toprnaCreate.pl* can automatically generate a CHARMM script that will perform such a heuristic 'folding' protocol.

It is also important to emphasize that PAIR patches do not place any backbone dihedral potentials on the paired nucleotides. For correct A-form helical structure STAC or TERM patches also need to be called to place such dihedrals (see 2.2.2 and 2.2.3).

2.2.2 STAC patches

STAC patches enforce A-form helical structure on two consecutive N or NN residues by adding backbone dihedral and improper potentials to the PSF. Two dihedral potentials are placed along the S_i-P_{i+1}-S_{i+1}-P_{i+2} and B_i-S_i-P_{i+1}-S_{i+1} dihedrals defined by the pseudo-atoms of residues i, i+1, and

i+2. A 'stacking' improper is also placed along the S_i-B_i-B_{i+1}-S_{i+1} improper dihedral. The user must provide three sequential residues, with the first two of these residues required to be `N` or `NN` types (any type is allowable for the third residue). If no third residue is available because the patch is at the end of a chain then a `TERM` patch should be used.

Example (Place dihedrals and stacking improper between residues 1 and 2):

```
PATCH STAC CHNA 1 CHNA 2 CHNA 3
```

2.2.3 `TERM` patches

`STAC` patches enforce A-form helical structure on two consecutive type `N` or `NN` residues by adding several backbone dihedral and improper potentials to the PSF. Unlike `STAC` patches, it only requires the input of two sequential residues, sacrificing the dihedral defined by the atoms S_i-P_{i+1}-S_{i+1}-P_{i+2} of residues i, i+1, and i+2. Thus, it should only be used if residue i+2 does not exist, i.e. residue i+1 is the terminal residue of the molecule.

Example (Place dihedrals and stacking improper between residues 9 and 10):

```
PATCH TERM CHNA 9 CHNA 10
```

2.3 Implementation of non-canonical intra-helical base pairs

In some scenarios it may be desirable to enforce a non-canonical pair between nucleotides in an internal loop. For example, GA pairs embedded in an otherwise long continuous Watson-Crick helix are known to form relatively stable pairs. However, such non-canonical pairs cannot be enforced using the patches described in 2.2 above. There are several reasons for this, the most relevant being that it is hard to derive good potentials due to a dearth of structures in the PDB. In addition, most non-canonical pairs can form in multiple different geometries and are in general expected to be more flexible than canonical pairs. Thus, TOPRNA uses a different procedure that allows increased flexibility and does not assume a single pairing geometry. `NL` or `NNL` types are reserved for residues participating in such pairs, and the patches and NOEs listed below are used to enforce pairing.

Note that long-distance tertiary pairs are typically implemented using only NOE restraints, leaving the 'paired' nucleotides as `NS` and `NNS` types without using `FILL` or backbone patches.

2.3.1 `FILL` patches

`FILL` patches add an M pseudo-atom extension to the B pseudo-atom of the provided residue, which must be type `NL` or `NNL`. Similar to the M pseudo-atoms added to canonical pairs, this patch helps prevent interpenetrations of the non-canonical pair with neighboring pairs. As above for `PAIR` patches, the added M atom must be initialized from the IC tables before any energy evaluations can be performed.

Example (add an M atom to residue 4 and use IC to initialize M pseudo-atom coordinates):

```
PATCH FILL CHNA 4
IC GENE
IC PARA
IC BUILD
```

2.3.2 `BACK` patches

`BACK` patches perform the same operation as `STAC` patches above, except they do not place a 'stacking' S_i-B_i-B_{i+1}-S_{i+1} improper. Three residues must be provided, with the first two required to be `N`, `NN`, `NL`, or `NNL` types. As noted above, the K_ψ of dihedral potentials involving a `NL` or `NNL` type as residue i or i+1 of a `BACK` patch are ¼ the strength of those placed between two `N` or `NN` types.

Example (add backbone dihedrals between M atom to residue 4):

```
PATCH BACK CHNA 3 CHNA 4 CHNA 5
```

2.3.3 `NOE` restraints used to enforce pairing

CHARMM NOE restraints are used to keep the B pseudo-atoms of non-canonical pairs apposed. These are square-well potentials, allowing the two nucleotides to adopt multiple conformations within a given proximity radius, consistent with the multiple conformations expected of non-canonical pairs. An example of two restraints used to restrain a single pair is shown below; the two restraints are needed because one restraint by itself will not necessarily keep the B pseudo-atoms in a paired-like geometry. Note that unlike patches that modify the CHARMM PSF file (and are therefore preserved when the PSF is read in), NOE restraints need to be re-specified at the start of every CHARMM process. For this reason *toprnaCreate.pl* writes all NOE restraints out to a separate file so that they can be easily streamed-in by subsequent processes.

Example (add NOE restraints to non-canonically paired residues 4 and 10):

```
NOE
ASSIGN KMIN 2.0 RMIN 5.5 KMAX 2.0 RMAX 7.5 FMAX 2.0 -
```

```
SELE ATOM CHNA 4 B END SELE ATOM CHNA 10 B END
ASSIGN KMIN 2.0 RMIN 11.0 KMAX 2.0 RMAX 14.0 FMAX 2.0 -
SELE ATOM CHNA 4 S END SELE ATOM CHNA 10 S END
END
```

2.4 Renaming patches

A series of patches are available to change the type of a residue within a CHARMM process. These primarily exist to facilitate the 'folding' of linear chains to a given secondary structure, where it is desirable to initialize a molecule as having all `NS/NNS` types. Here, the lack of attractive interactions of `NS/NNS` types prevents the molecule from undergoing undesired, non-specific collapse. Residues can then be renamed to their appropriate types after the molecule has been folded with temporary restraints. This is the procedure used by *toprnaCreate.pl* when a molecule is initialized without preexisting coordinates. The below patches should always be accompanied by a rename statement that also updates the residue name (see example).

2.4.1 Change a residue to a base paired type

`PATCH NBP` or `PATCH NNBP` modifies the specified residue to a type N or NN residue type, where N indicates either A, U, G, or C.

Example (Change residue 10 of segid CHNA to type 'G' and rename to 'G'):

```
PATCH GBP CHNA 10
RENAME RESN G SELE RESID 10 END
```

2.4.2 Change a residue to a single stranded type

`PATCH NSS` or `PATCH NNSS` modifies the specified residue to a type NS or NNS residue type, where N indicates either A, U, G, or C.

2.4.3 Change a residue to non-canonical loop type

`PATCH NLP` or `PATCH NNLP` modifies the specified residue to a type NL or NNL residue type, where N indicates either A, U, G, or C.

2.5 Example setup script

An example CHARMM script that constructs a TOPRNA model of a 14-nt RNA hairpin is provided below (Figure 1). Note that this example is meant only to demonstrate the sequence and patching scheme used for this RNA. Presumed preexisting coordinates are read in as `coordinates.pdb`. See section 3.0 for options on how to use *toprnaCreate.pl* to generate TOPRNA coordinates.

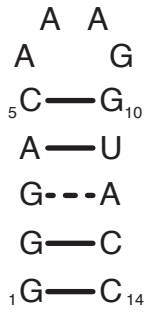

Figure 1: Example hairpin sequence. Solid lines indicate canonical base pairs and the dashed line indicates a non-canonical pair.

Example (initialize the PSF of the hairpin shown in Figure 1):

```
READ RTF CARD NAME TOPRNA_TOP1.RTF
READ PAR CARD NAME TOPRNA_PAR1.PAR

READ SEQU CARD
* The sequence for the hairpin
14
G GG GL A C AS AAS AS GS G U AL C CC

GENE CHNA SETUP
READ COOR PDB NAME COORDINATES.PDB

PATCH PAIR CHNA 1 CHNA 14
PATCH PAIR CHNA 2 CHNA 13
PATCH FILL CHNA 3
PATCH PAIR CHNA 4 CHNA 11
PATCH PAIR CHNA 5 CHNA 10


!! Place the NOE restraints for the non-canonical BP
NOE
ASSIGN KMIN 2.0 RMIN 5.5 KMAX 2.0 RMAX 7.5 FMAX 2.0 -
SELE ATOM CHNA 3 B END SELE ATOM CHNA 12 B END
ASSIGN KMIN 2.0 RMIN 11.0 KMAX 2.0 RMAX 14.0 FMAX 2.0 -
```

```
SELE ATOM CHNA 3 S END SELE ATOM CHNA 12 S END
END


!! Add the backbone dihedral potentials
PATCH STAC CHNA 1 CHNA 2 CHNA 3
PATCH BACK CHNA 2 CHNA 3 CHNA 4
PATCH BACK CHNA 3 CHNA 4 CHNA 5
PATCH STAC CHNA 4 CHNA 5 CHNA 6
PATCH STAC CHNA 10 CHNA 11 CHNA 12
PATCH BACK CHNA 11 CHNA 12 CHNA 13
PATCH BACK CHNA 12 CHNA 13 CHNA 14
PATCH TERM CHNA 13 CHNA 14

!! Use IC tables to build M-atoms added by pair patches
IC GENE
IC PARA
IC BUILD
```

2.6 Miscellaneous simulation details

TOPRNA models are in theory compatible with all standard CHARMM functionality. However, TOPRNA was primarily designed for Langevin dynamics simulations. Several details regarding these simulations are listed below.

The force field is optimized for simulations at 300 K using friction coefficients of fbeta = 5.0 ps^{-1} and 0.02 ps time step. Preliminary benchmarking simulations indicated that stability is maintained with time steps as long as 0.04 ps, but caution should be exercised if using longer time steps. As part of a temperature replica exchange protocol we have performed production simulations at temperatures as high as 500 K using the standard friction and time step parameters above. Given the low energy barriers inherent to the TOPRNA force field there is no practical reason to use higher temperatures for any production simulation scheme (and lower maximum temperatures should also be sufficient).

Second, in a few very rare instances we have observed fatal instabilities in simulations using the above parameters. The cause of these instabilities has not been resolved. However, in all cases the instability did not reappear when the simulation was restarted from a time point prior to the appearance of the instability. Thus, if one encounters any fatal instability we simply recommend rerunning the simulation.

Finally, dynamics simulations often throw EIPHI warnings due to improper torsion angles being far from their minimum value. These warnings can be safely ignored. To give a brief explanation, these warnings arise from the weak harmonic dihedral potentials that are applied across base pair bonds, which as per the requirements of CHARMM are implemented as impropers. This use of impropers is counter to the original intention of the CHARMM designers, who intended impropers to be solely used to prevent inversions of chiral centers. For this purpose, large deviations of an improper from its minimum indicated a potential problem and thus generated a warning. However, for our simulations such deviations are consistent with the design parameters of TOPRNA and thus can be ignored.

An example dynamics function call for a simulation of a TOPRNA model is provided below. As emphasized previously, users of TOPRNA should have a strong working knowledge of CHARMM. This example is meant purely as an illustrative guide and users should make appropriate adjustments to match their specific needs. Users should refer to dynamc.doc of the CHARMM documentation for general information about running dynamics simulations.

Example (initialize fbeta and perform 5000 steps of dynamics at 300 K):

```
SCALAR FBETA SET 5
DYNA LEAP LANG NSTEP 5000 TIMESTEP 0.02 NTRFRQ 1000 -
IUNREA -1 IUNWRI -1 IUNCRD -1 KUNIT -1 -
IPRFRQ 0 IEQFRQ 0 INBFRQ -1 IHBFRQ 0 -
NPRINT 1000 NSAVC -1 ISVFRQ 1000 -
FIRSTT 300 FINALT 300
```

3. *toprnaCreate.pl* utility

For most molecules it is impractical to manually specify the sequence and list of patches necessary to create a TOPRNA model. It is also difficult to obtain initial coordinates. To address these challenges we constructed the *toprnaCreate.pl* perl utility, which has a variety of useful options for generating patch lists and initial coordinate files. At its most basic, *toprnaCreate.pl* takes an all-atom PDB file and a list of secondary structure base pairs and outputs a TOPRNA PDB file and a CHARMM script that creates the TOPRNA PSF.

3.1 Usage information

toprnaCreate.pl runs at the command line and has no external dependencies other than an install of perl version 5.10 or higher. While no setup is necessary, it is important to note that the paths of the TOPRNA parameter and topology files are hard coded as global variables within the script and will be output into generated CHARMM scripts. If desired these paths can be easily modified, which are located at the top of the script. Alternatively, the individual CHARMM scripts output by *toprnaCreate.pl* can be edited.

Command usage:

```
> toprnaCreate.pl [-help] [-seq SEQFILE] [-pdb PDBFILE]
    [-bp BPFILE] [-rmap STRING | MAPFILE]
    [-cons STRING | CONSFILE]
    {-init [prefix] | -fromc [prefix]}
```

Unless the `-help` option is specified, any *toprnaCreate.pl* function call must include `-seq` or `-pdb` input. Simultaneous input of both is also supported. Either the `-init` or `-fromc` output options must be specified.

3.2 Description of options

`-help`

Print out usage information.

`-seq SEQFILE`

Read in the specified sequence file.

`-pdb PDBFILE`

Read in the specified all-atom PDB file.

`-bp BPFILE`

Read in the specified base-pairing file.

`-rmap STRING | MAPFILE`

Read in a mapping of PDB residue numbers to TOPRNA residue numbers. The mapping can be either input at the command line as a string or as a text file. This option is helpful if the residue numbering of the PDB file is discontinuous or if one wishes to initialize a TOPRNA model from only a subset of coordinates in a PDB file.

`-cons STRING | CONSFILE`

Specify a list of TOPRNA residues that will be constrained to their initial PDB structure conformation through backbone dihedral constraints. The residue list can either be input at the command line as a string or as a text file. The dihedral constraints will be output as list in a separate file with the -cons postfix. This option is useful if there is a stable non-canonical structure that cannot be easily modeled using other TOPRNA functionality. This option is only available when both `-pdb` and `-fromc` options are also called.

`-fromc [prefix]`

Output a CHARMM script that initializes a molecule from input PDB coordinates. Output files will be named by the provided prefix, or by the default prefix 'toprna'. This option is not supported in the absence of PDB input. Note that as an experimental feature, this option can also be used to construct TOPRNA models from a partial set of PDB coordinates. Here, an input sequence file specifies the sequence. The `-rmap` option can then be used to map PDB residues to the TOPRNA residue numbering, and these coordinates used to initialize part of the molecule. The remaining residues are built using a hierarchical folding procedure similar to that used by the `-init` option.

`-init [prefix]`

Output a CHARMM script that initializes TOPRNA coordinates from a linear chain. The script hierarchically folds the secondary structure using temporary restraints and short dynamics simulations, followed by removal of the constraints and imposition of necessary patches. Output files will be named by the provided prefix, or by the default prefix 'toprna'. This output option is supported for all input options. **It is important to note that this folding procedure has been derived heuristically and is not guaranteed to work. Initial models generated from this option should <u>always</u> be visually inspected to ensure that they have realistic structure. To ensure reproducibility it is recommended that several different initial models generated with this option and compared to one another.**

3.3 PDB input

The `-pdb` option accepts standard format PDB files. The input PDB can be used as a source of sequence input and/or initial coordinates. Only residue names of `GUA`, `G`, `CYT`, `C`, `URA`, `U`, `ADE`, or `A` are recognized, and all non-`ATOM` entries are skipped. In conjunction with information provided in the base-pairing file, the residues will be renamed to the correct TOPRNA type according to the conventions outlined in section 2. Unless overridden by `-seq` and `-rmap` options, the residue numbering of the PDB will be automatically renumbered beginning at 1, and incremented with each additional residue. Thus, discontinuities in the PDB residue numbering are not recognized. If the `-fromc` option is specified then the all-atom coordinates of the PDB file will be converted to TOPRNA pseudo-atoms and written to a new PDB file. Note that pseudoatoms are only built for moieties that have a complete set of component heavy atoms.

3.4 Sequence input

The `-seq` option accepts input of sequence files with the following format:

Example (the sequence shown in Figure 1; line identifiers are italicized on the left):

(Optional) Chain specification	`Chain B`
Sequence line 1	`G G G A C`
Sequence line 2	`A A A G`
Sequence line 3	`G U A C C`

The optional `Chain` keyword can be used to specify the chain letter of the molecule (`B` in the above example); by default the chain will be named `A`. If used, this command must be the first line of the sequence file. Subsequent lines list the sequence, with each residue separated by either a space or new line. Residues do not need to be typed according to TOPRNA conventions (i.e. single vs. double letter, single-stranded vs. base-paired) – this is handled automatically by *toprnaCreate.pl*. Thus only residues `G`, `C`, `A`, and `U` are supported.

3.5 Base pairing input

The `-bp` option accepts input of a base-pairing file with the following format:

Example (base pairing of molecule shown in Figure 1; line identifiers are italicized on the left):

Base pair line 1	`1 14`
Base pair line 2	`2 13`
Base pair line 3	`3 12 *`
Base pair line 4	`4 11`
Base pair line 5	`5 10`
Base pair line 6	`! 6 9`

Each line specifies two residues that are paired with one another, separated by a space. The residue numbering corresponds to the TOPRNA residue numbering, and due to renumbering may differ from that in the all-atom PDB file. In above the example the `!` flag at the beginning of base pair line 6 serves to comment out the line. The `*` flag at the end of the base pair line 3 tells *toprnaCreate.pl* to treat this pair as a non-canonical internal loop pair (see section 2.3). As in this example residue 3 is a G and residue 12 an A, *toprnaCreate.pl* will automatically recognize and treat this pair as non-canonical pair regardless of the `*` flag. However, this flag is required for cases where one wishes to specify a non-canonical pair between AU, GU, or GC residues.

3.6 Residue mapping input

The `-rmap` option allows one to select a set of coordinates from a PDB file and map them to a set of TOPRNA residues. Only selected residues will be mapped (i.e. non-selected PDB residues will be ignored and coordinates of unmapped TOPRNA residues uninitialized). The mapping can either be input as a string at the command line or as a text file. The PDB residue selection is specified first, followed by the destination TOPRNA residues, separated by a comma:

Example (map PDB residues 5 to 10 of chain A to TOPRNA residue numbers 1 to 5):

```
A.5:10,1:5
```

Several discontinuous mappings can also be made by separating selections with a '_':

Example (map PDB residues 6 to 10 of chain A to TOPRNA residue numbers 1 to 5; PDB residues 14 to 19 to TOPRNA residues 6 to 11; and PDB residue 30 to TOPRNA residue 15):

```
A.6:10,1:5_A.14:19,6:11_A.30,15
```

The same string can also be saved in a text file, with the file path provided after the `-rmap` flag instead of a string. If desired, an alternative formatting is also available for residue mappings saved in text files; '`_`' can replaced be with a newline and '`,`' replaced with a space:

Example (map multiple sets of PDB residues to TOPRNA residues using alternative text file formatting; line identifiers are italicized on the left):

rmap line 1	`A.6:10 1:5`
rmap line 2	`A.14:19 6:11`
rmap line 3	`A.30 15`

Finally, in all of the above examples the chain letter does not need be specified if the PDB file has only one chain. If this is the case, the following string is sufficient:

Example (map PDB residues to TOPRNA residues when there is only a single PDB chain):

```
5:10,1:5_14:19,5:10_30,15
```

3.7 Backbone constraint selection input

The `-cons` option allows one to specify a range of backbone dihedrals to constrain to their PDB structure values. This option outputs a list of `CONS DIHE` commands to a separate script that can be streamed in by subsequent CHARMM processes. The list of residues can either be input at the command line as a string or as a text file. Note that the input should correspond to the TOPRNA residue numbering rather than PDB residue numbering. Residue selections follow similar conventions to that used for residue mappings:

Example (restrain TOPRNA residues 11 to 13 and residues 36 to 40):

```
11:13_36:40
```

4. tRNA example

tRNA^{Phe} is presented here as a more detailed example of the various options one has for building TOPRNA models with *toprnaCreate.pl*. Several files relating to this example are located in the directory *tRNAexample*: `6TNAe.pdb` is an edited version of the 6TNA PDB structure of tRNA where all modified residues where changed to their canonical analogs; `trnawt-seq.txt` and `trnamut-seq.txt` contain the wild-type and mutant tRNA sequences shown in Figure 2; `trna-bp.txt` contains the base pairs shown in Figure 2.

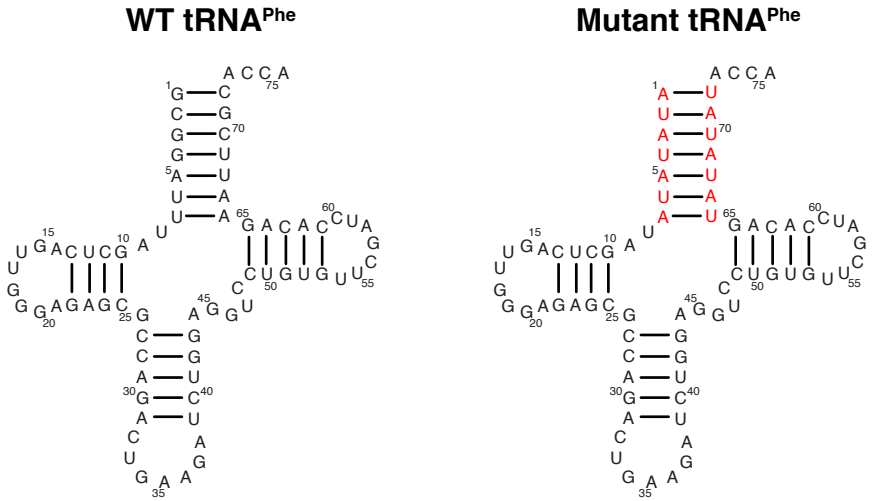

Figure 2: WT yeast tRNA^{Phe} and a hypothetical mutant tRNA^{Phe} sequence, with mutated acceptor stem shown in red. This mutant is meant to provide an example of the ability of *toprnaCreate.pl* to create new molecules from PDB structures that have different primary sequences.

4.1 Create tRNA model using initial coordinates from PDB structure

The following commands generate a TOPRNA model of tRNA using initial PDB coordinates:

```
../toprnaCreate.pl -pdb 6TNAe.pdb -bp trna-bp.txt -fromc trna
```

This writes a CHARMM script to `trna.inp` and initial coordinates computed from the `6TNAe.pdb` PDB structure to `trna.pdb`. Execution of `trna.inp` by CHARMM will read in the coordinates, perform the necessary patches and a brief minimization, and then write the PSF

and final coordinate files to `trna-final.psf` and `trna-final.pdb`, respectively. `trna.inp` can be executed by CHARMM as follows:

```
$CHARMMEXEC <trna.inp
```

The final model can be visualized using VMD, inputting both the PSF and PDB to visualize the connectivity of the pseudo-atoms:

```
vmd trna-final.psf trna-final.pdb
```

4.2 Create tRNA model with an internal loop pair

In the `trna-bp.txt` file there is a line specifying a non-canonical pair between residues 26 and 44 that is commented out using a '!'. If this '!' is deleted and the file saved, execution of the same command will produce a script that gives a TOPRNA model containing this non-canonical pair:

```
../toprnaCreate.pl -pdb 6TNAe.pdb -bp trna-bp.txt -fromc trna
```

Once again, a CHARMM script named `trna.inp` and initial coordinates saved as `trna.pdb` will be output. In addition, the file `trna-cons.inp` that contains the two NOE restraints of the non-canonical pair will be output (see section 2.3). As in section 4.1, execution of `trna.inp` by CHARMM will output a PSF and PDB file. Any later CHARMM processes should both read in the PSF and PDB files as well as stream in `trna-cons.inp` to set up the necessary non-canonical pairing restraints.

4.3 Create tRNA model without initial PDB coordinates

The following command will output a CHARMM script that generates a TOPRNA model of tRNA using a hierarchical folding procedure starting from a linear chain:

```
../toprnaCreate.pl -seq trnawt-seq.txt -bp trna-bp.txt -init trna
```

Note that CHARMM execution of the output `trna.inp` script will take ~5 minutes.

4.4 Create tRNA model using initial PDB structure coordinates but new sequence

If both the `-seq` and `-pdb` options are specified than the sequence file will overwrite the sequence of the PDB file. However, the coordinates of the PDB file can still be used to provide a set of rough initial coordinates for the model. Such an option is useful if, for example, one wanted to explore the effects of different sequence-specific helix structures. In this example, we create a tRNA model from the `6TNAe.pdb` structure with the mutant sequence shown in Figure 2:

```
../toprnaCreate.pl -seq trnamut-seq.txt -pdb 6TNAe.pdb \
     -bp trna-bp.txt -fromc trna
```

4.5 Create tRNA model using only a subset of PDB structure coordinates

In some cases it may be desirable to build a TOPRNA model from a PDB structure that has an incomplete set of coordinates. Alternatively, one may wish to use a subset of PDB structure coordinates to initialize a molecule with an alternative secondary structure. An example of this later scenario would be if one were interested in the effects that inserting or deleting nucleotides in a junction would have on the topological constraints of a given RNA. The `-rmap` option is useful for both of these scenarios. Here, the sequence of the molecule is specified through a sequence file and `-rmap` is used to map a subset of PDB coordinates to a set of TOPRNA residues. PDB residues which are not mapped are ignored, and the set of TOPRNA residues without initial coordinates are built using a hierarchical folding procedure. In the following example, a tRNA model is initialized using all coordinates except for those of the anti-codon loop and the first base pair of the anti-codon stem; these residues are rebuilt by the output CHARMM script (saved as `trnapart.inp`):

```
../toprnaCreate.pl -seq trnawt-seq.txt -pdb 6TNAe.pdb \
     -bp trna-bp.txt -rmap 1:30,1:30_40:76,40:76 -fromc trnapart
```

4.6 Apply backbone dihedral restraints to model of tRNA

By design, TOPRNA treats single-stranded residues as freely rotatable chains. In some cases, this may be undesirable with the user wishing to restrain a subset of single-stranded residues to the conformation observed in a PDB structure. The user can specify these restraints manually in subsequent CHARMM scripts, but for long single-stranded regions many restraints may be

needed and specifying them can be cumbersome. To ease this process the `-cons` option allows one to generate a list of dihedral restraints that can then be streamed into later CHARMM scripts. The below example restrains the anti-codon loop:

```
../toprnaCreate.pl -pdb 6TNAe.pdb -bp trna-bp.txt \
     -cons 32:38 -fromc trna
```

Here, the restraints will be output to `trna-cons.inp`. Note that if the molecule has any non-canonical pairs that require NOE restraints these will also be output to the same –cons file.

5.0 Test Case

A short testcase to check the force field is provided in the directory *testcase*. CHARMM execution of `testFF.inp` should exit normally. If not, there is a problem with the force field and/or its compatibility with the installed version of CHARMM.