pmlbeta Documentation

Release 1.0.10.dev10+g2841a2c.d20210927

András Wacha

Sep 27, 2021

Contents:

pmlbeta is an extension for PyMOL, containing utilities and algorithms for dealing with β-peptides. It has facilities for creating and manipulating geometries of natural α- and artificial β- and even mixed α/β-peptides.

This program has been developed by the BioNano Platform of the Hungarian Research Centre for Natural Sciences

CHAPTER 1

Installation

1.1 Requirements

pmlbeta is an extension for the PyMOL molecular graphics system. Only version 2.0 and above is supported. The additional dependencies are:

Python: at least version 3.0 is needed, 3.5 and above is tested. Python 2 is not supported.

PyQt5: for the graphical user interface

PyMOL: testing is done with the most up-to-date open source version, currently 2.3.0. Other versions above 2.0 should also work, but not guaranteed. However, if you find compatibility issues, *please file a bug report*.

1.2 Installing from a pre-built package

Download the latest binary package from https://gitlab.com/awacha/pmlbeta/raw/binaries/pmlbeta_latest.zip and install it with the plugin manager of PyMOL.

Please note that PyMOL (at least up to the open source version 2.3.0) only supports installing ZIP plugins by manually downloading them and selecting the downloaded file. Giving the above URL to the package manager does not work.

1.3 Manual installation

Check out the git repository:

```
$ git clone https://gitlab.com/awacha/pmlbeta.git
```

Create the plugin zip file with:

```
$ python makebundle.py
```

Then use the plugin manager of PyMOL to install the plugin from the just created zip file.

CHAPTER 2

Quick start

The main goal of *pmlbeta* is to provide tools for building molecular models of β-peptides (or even mixed α/β-peptides). The idea is mostly taken from the *fab* command of PyMOL, which can create peptides in a desired secondary structure. The construction of β-peptides can be done by two means: either using the command-line or a graphical user interface.

2.1 Command line

The command *betafab2* can be used for creating α/β-peptides in extended conformation:

```
betafab2 hp6, (2R3S)B23h(2A3A), (2R3S)B23h(2A3V), (S)B2hV, (S)B3hK, (2R3S)B23h(2A3A),␣
↪(2R3S)B23h(2A3L)

betafab2 valxval, (S)B3hV, (S)B3hA, (S)B3hL, (2S3S)B23h(2A3A), (S)B3hV, (S)B3hA,␣
↪(S)B3hL

betafab valval, (S)AV, (S)AA, (S)AL, (S)AV, (S)AA, (S)AL

betafab mixed, (S)AV, (S)B3hV, (R)B2hA, (S)AQ, (2R3S)B23h(2C3W)
```

The first argument is always the PyMOL object name, followed by the residue abbreviations following the notation in *Simplified nomenclature*

Folding a peptide into the desired secondary structure is possible by supplying the desired secondary structures for the amino-acids in *betafab2*:

```
betafab2 valxval, (S)B3hV{H14M}, (S)B3hA[-140.3 66.5 136.8], (S)B3hL{H14M}
```

Another option is to fold an already built peptide using the *fold_bp* command:

```
fold_bp H14M, valxval   # folds all residues the "valxval" model to the H14M helix

fold_bp (-140.3 66.5 136.8), valxval    # the same as above, just the torsion angles␣
↪are explicitly given
```

Instead of folding the entire peptide to the same secondary structure, this can be done residue-by-residue:

```
betafab2 test, (S)B3hA, (S)B3hA, (S)B3hA, (S)B3hA, (S)B3hV, (S)B3hA, (S)B3hA

fold_bp [ H14M H14M H14M H14M H14M H14M H14M], test

fold_bp [ H14M (-140.3 66.5 136.8) H14M H14M H14M (-140.3 66.5 136.8) H14M], test #␣
↪the same as above
```

Note that not every peptide sequence supports every secondary structure. For example, you will find steric clashes if you try to fold an (*R*)-homochiral β^3-peptide into the H14M helix:

```
betafab2 invalid, (R)B3hL, (R)B3hL, (R)B3hL, (R)B3hL, (R)B3hL

fold_bp H14M, invalid
```

In other cases when the sequence and the desired fold are compatible, you can still get steric clashes because the folding procedure does not touch the side-chains. Luckily these can easily be removed by the sculpting facility of PyMOL. For example, when the above peptide is produced in the default, extended conformation, there is a steric clash between the leucine side-chains and the amide oxygen. To resolve them, first fix the backbone atoms and initialize sculpting.

```
betafab2 clash, (R)B3hL, (R)B3hL, (R)B3hL, (R)B3hL, (R)B3hL

# select the backbone -> makes a new selection (bbone)
select_bbb bbone, invalid

# fix the backbone atoms
flag fix, (bbone), set

# Initialize sculpting
sculpt_activate invalid

# Do some iterations
sculpt_iterate invalid, cycles=1000

# Deactivate sculpting
sculpt_deactivate invalid

# make the backbone atoms free
flag fix, (bbone), clear
```

Or if your PyMOL installation "includes modelling capabilities"[1], you can get even better results with the *clean* command:

```
betafab2 clash, (R)B3hL, (R)B3hL, (R)B3hL, (R)B3hL, (R)B3hL

# select the backbone -> makes a new selection (bbone)
select_bbb bbone, invalid

# fix the backbone atoms
flag fix, (bbone), set

# Geometry optimization
clean invalid

```

(continues on next page)

[1] In other words, if the *freemol* extension is installed. It is installed by default in the Incentive and the Education version of PyMOL, but it can also be made to work in the open source version.

(continued from previous page)

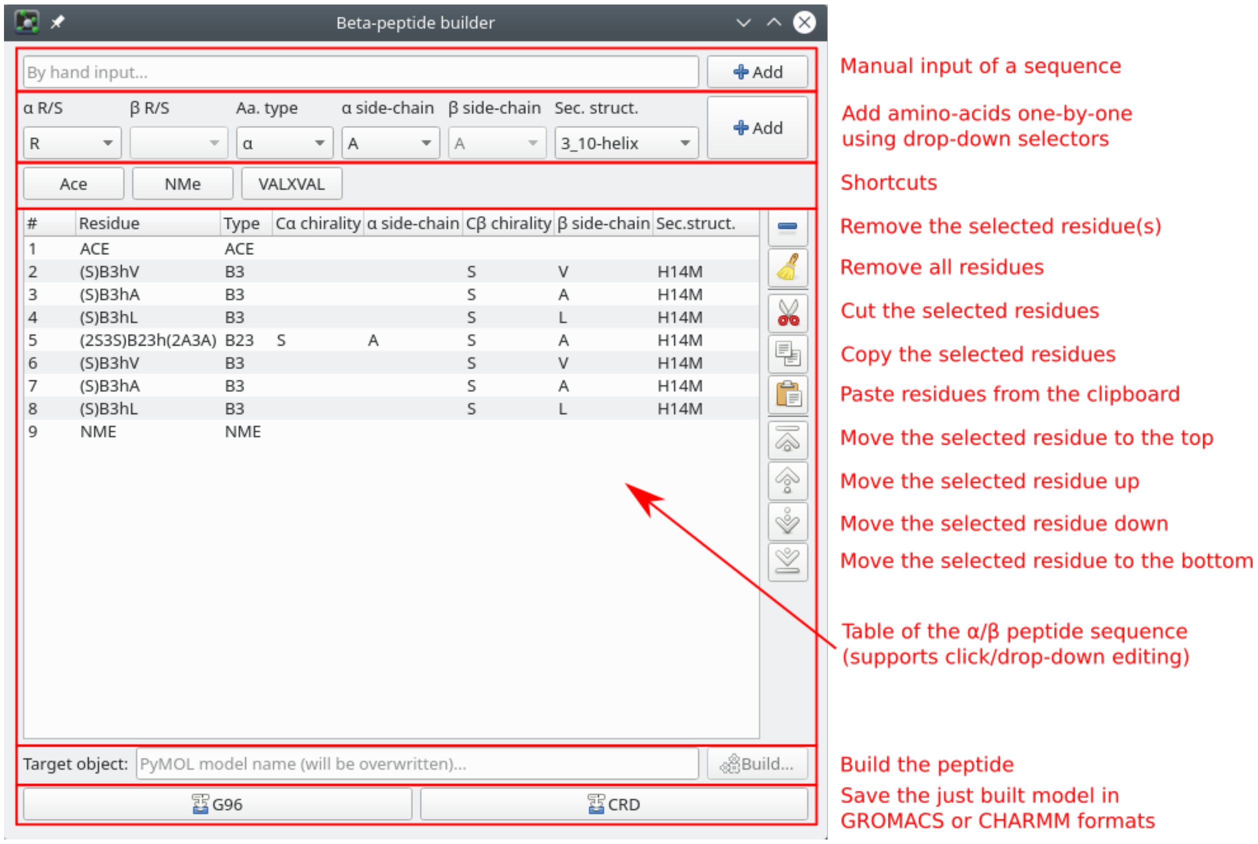
```
# make the backbone atoms free
flag fix, (bbone), clear
```

2.2 Graphical user interface

This plug-in supplies two GUI tools, available in the Plugin menu of the PyMOL main menu bar.

2.2.1 Betafab GUI

The graphical user interface of the β-peptide builder can be found in the Plugins menu of PyMOL. The main window is shown in the figure below.


Beta-peptide builder
By hand input...
Add
Add
Ace
NMe
VALXVAL
Target object:
PyMOL model name (will be overwritten)...
Build...
G96
CRD
Manual input of a sequence
Add amino-acids one-by-one using drop-down selectors
Shortcuts
Remove the selected residue(s)
Remove all residues
Cut the selected residues
Copy the selected residues
Paste residues from the clipboard
Move the selected residue to the top
Move the selected residue up
Move the selected residue down
Move the selected residue to the bottom
Table of the α/β peptide sequence (supports click/drop-down editing)
Build the peptide
Save the just built model in GROMACS or CHARMM formats

α R/S	β R/S	Aa. type	α side-chain	β side-chain	Sec. struct.
R		α	A	A	3_10-helix

#	Residue	Type	Cα chirality	α side-chain	Cβ chirality	β side-chain	Sec.struct.
1	ACE	ACE					
2	(S)B3hV	B3			S	V	H14M
3	(S)B3hA	B3			S	A	H14M
4	(S)B3hL	B3			S	L	H14M
5	(2S3S)B23h(2A3A)	B23	S	A	S	A	H14M
6	(S)B3hV	B3			S	V	H14M
7	(S)B3hA	B3			S	A	H14M
8	(S)B3hL	B3			S	L	H14M
9	NME	NME					

Fig. 1: The Betafab GUI window explained

Building blocks (α- or β-amino acids, as well as Ace and NMe capping groups) can be added in three different ways to the sequence (top of the window):

1. manual input of either a simple amino-acid or a whole sequence (comma separated amino acid abbreviations). The notation is the same as used by the command *betafab2*, see *Simplified nomenclature*
2. one-by-one addition of amino acids using the drop-down selectors.
3. using the quick-access buttons (currently for Ace and NMe capping groups and a test peptide)

The central part shows the peptide sequence and allows a simple means for editing. Amino acids can be removed, reordered and even modified. By double-clicking on cells in columns 3-8, properties of the residue (type, sidechains, stereoisomers) can be changed using drop-down selectors.

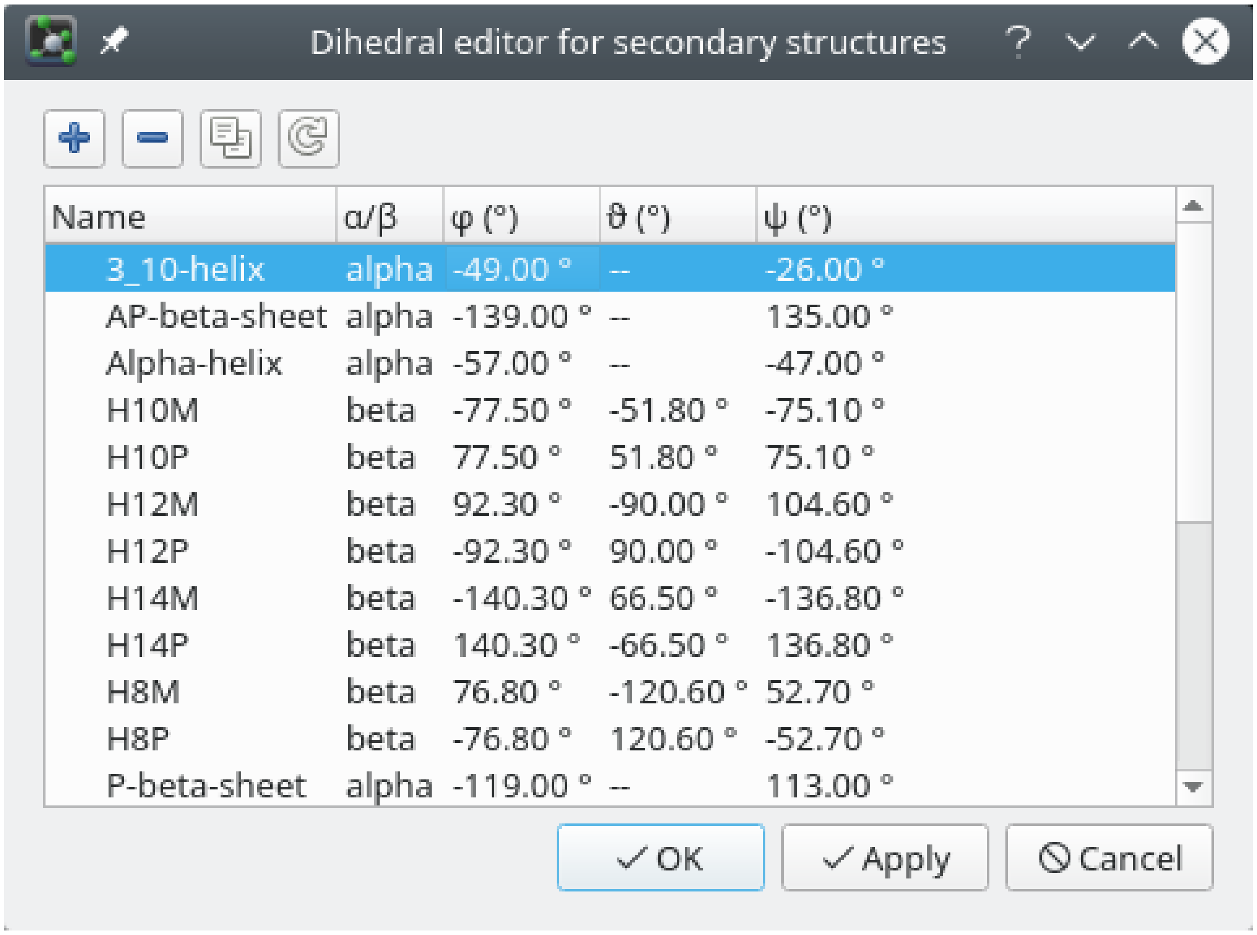
The desired secondary structure can also be given in the last column. The drop-downs display the list of the known secondary structures in the *Secondary structure database*.

The peptide sequence can be built by supplying a PyMOL model name for it and pressing the “Build” button.

The built model can be saved to .g96 or .crd files using the corresponding push buttons.

2.2.2 Secondary structure editor

pmlbeta provides a graphical utility for editing the secondary structure database. It can be reached via the Plugin menu of PyMOL



Name	α/β	φ (°)	ϑ (°)	ψ (°)
3_10-helix	alpha	-49.00 °	--	-26.00 °
AP-beta-sheet	alpha	-139.00 °	--	135.00 °
Alpha-helix	alpha	-57.00 °	--	-47.00 °
H10M	beta	-77.50 °	-51.80 °	-75.10 °
H10P	beta	77.50 °	51.80 °	75.10 °
H12M	beta	92.30 °	-90.00 °	104.60 °
H12P	beta	-92.30 °	90.00 °	-104.60 °
H14M	beta	-140.30 °	66.50 °	-136.80 °
H14P	beta	140.30 °	-66.50 °	136.80 °
H8M	beta	76.80 °	-120.60 °	52.70 °
H8P	beta	-76.80 °	120.60 °	-52.70 °
P-beta-sheet	alpha	-119.00 °	--	113.00 °

Fig. 2: The secondary structure dialog

The dialog is a simple list of secondary structures, with the possibility to add, duplicate and remove entries. Each entry can be edited by double-clicking on the cells in the table. Changes are updated only if the *OK* or *Apply* button is pressed. The default values can be restored by the *Add defaults* button on the toolbar.

Changes to the secondary structure database are saved for the next PyMOL session.

CHAPTER 3

β-amino acid residue naming

3.1 General nomenclature

Acyclic β-amino acids come in four varieties, according to the substitution site (see the figure)

β^2 Monosubstituted β-amino acid, the side-chain is on the α-carbon.

β^3 Monosubstituted β-amino acid, the side-chain is on the β-carbon.

$\beta^{2,3}$ Disubstituted β-amino acid, one side-chain on both the α- and the β-carbon.

β-Alanine A simple β-backbone, without side-chains

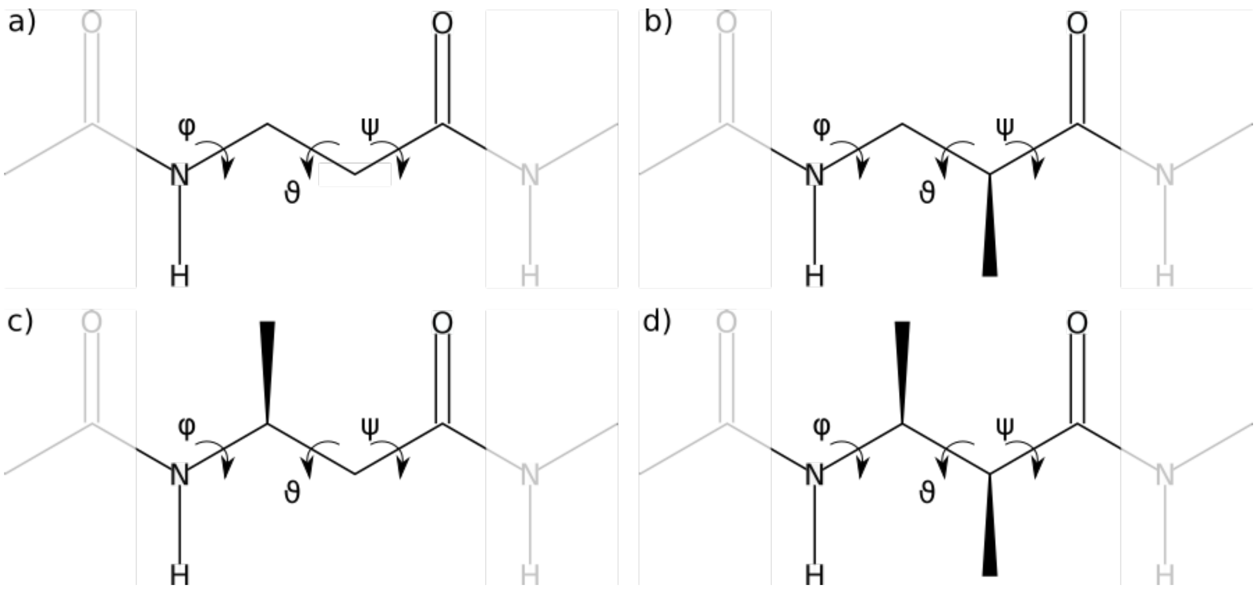


Fig. 1: The four kinds of β-amino acids: bare β-backbone or β-alanine (a), β^2-amino acid (b), β^3-amino acid (c) and $\beta^{2,3}$-amino acid (d)

In contrast to their natural counterparts, β-amino acids do not have a single, generally accepted nomenclature. We adopt here the following, widely used convention, which emphasizes the homology with α-amino acids and allows to account for the absolute conformation (chirality) as well. The general block format is :

`<chirality><substitution type>h<side-chain designation>`, where:

<chirality> is the designation of the chirality of the substitution site atoms. For monosubstituted β-amino acids, this is either (*S*) or (*R*), including parentheses. For disubstituted ones, the substitution site is also labelled to avoid ambiguity, i.e. (2*S*, 3*R*) etc.

<substitution type>: either β^2, β^3or $\beta^{2,3}$.

<side-chain designation>: single-letter abbreviation of the proteinogenic amino-acid whose side-chain is referred to. As for the chirality above, in the case of disubstituted amino-acids the substitution site must be explicitly given in order to avoid confusion, i.e. (2A,3Q), etc.

We include α-amino acids in this notation, too, with the following scheme: `<chirality>`α`<side-chain designation>`, where:

<chirality> is similar as for β^2- or β^3-amino acids, i.e. either (*S*) or (*R*). Additionally, for *D* and *L* are also supported for convenience[1].

<side-chain designation>: is once again the single-letter abbreviation of proteinogenic amino acids

Examples:

monosubstituted:

- (*S*)β^2hV: valine side-chain on the α-carbon with *S* chirality
- (*R*)β^3hR: arginine side-chain on the β-carbon with *R* chirality

disubstituted:

- (2*S*,3*R*)$\beta^{2,3}$ h(2A,3L): disubstituted β-amino acid with an alanine side-chain on the α-carbon (with *S* chirality) and an arginine on the β-carbon (*R* chirality)

achiral (bare backbone):

- βA

α-amino acids:

- (*S*)αV: L-valine
- (*R*)αW: D-tryptophan

Two, more complicated examples are shown in the next two figures:

3.2 Simplified nomenclature

In the *betafab2* command, essentially the above defined notation is used for defining a β-peptide sequence, with the following simplification:

- superscripts are omitted
- instead of the greek letter β the capital “B” is used
- commas in amino-acid abbreviations are omitted, they are used instead for separating the subsequent residues in the peptide chain

[1] Note that because for β-amino acids no internationally agreed convention exists on the *D* / *L* nomenclature, their chirality can only be specified using the unambiguous *S* / *R* notation.

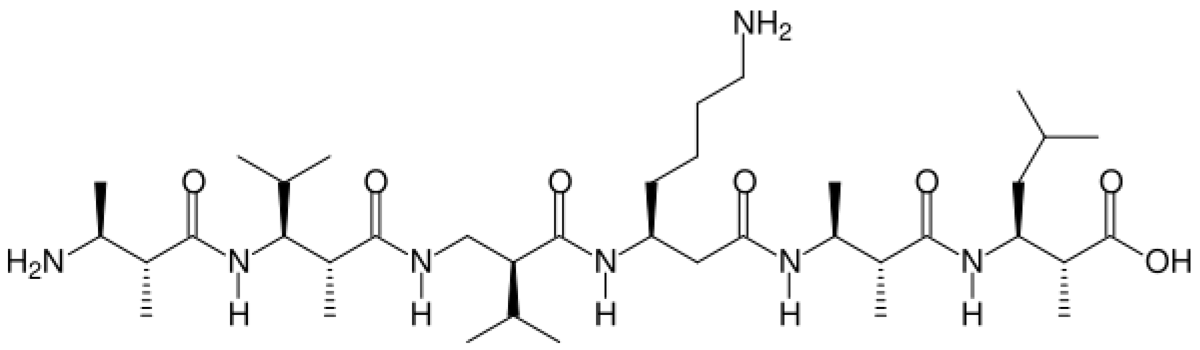


Fig. 2: (2*R*,3*S*)$\beta^{2,3}$h(2A,3A) - (2*R*,3*S*)$\beta^{2,3}$h(2A,3V) - (*S*)β^{2}hV - (*S*)β^{3}hK - (2*R*,3*S*)$\beta^{2,3}$h(2A,3A) - (2*R*,3*S*)$\beta^{2,3}$h(2A,3L)

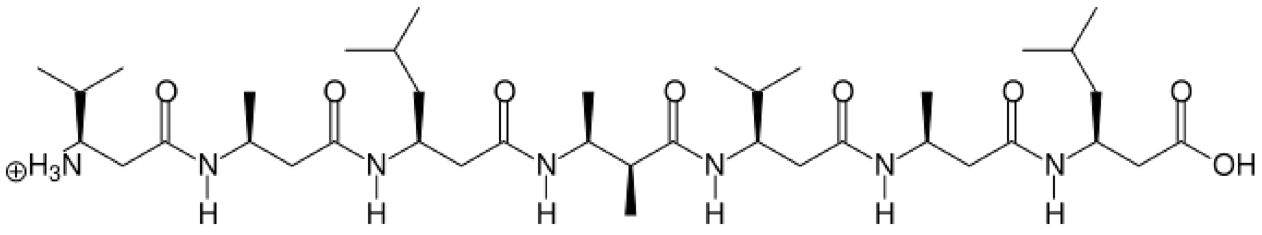


Fig. 3: (*S*)β^{3}hV - (*S*)β^{3}hA - (*S*)β^{3}hL - (2*S*,3*S*)$\beta^{2,3}$h(2A,3A) - (*S*)β^{3}hV - (*S*)β^{3}hA - (*S*)β^{3}hL

- letters *S* and *R*, describing the absolute conformation, are not italicized

For example, to construct the above two peptides, the following PyMOL commands can be used:

```
betafab2 hp6, (2R3S)B23h(2A3A), (2R3S)B23h(2A3V), (S)B2hV, (S)B3hK, (2R3S)B23h(2A3A),␣
↪(2R3S)B23h(2A3L)

betafab2 valxval, (S)B3hV, (S)B3hA, (S)B3hL, (2S3S)B23h(2A3A), (S)B3hV, (S)B3hA,␣
↪(S)B3hL
```

In addition, α-amino acids are also supported with the following notation:

```
betafab valval, (S)AV, (S)AA, (L)AL, (D)AV, (S)AA, (S)AL

betafab mixed, (S)AV, (S)B3hV, (R)B2hA, (S)AQ, (2R3S)B23h(2C3W)
```

Additionally, we now support the two most common cyclic beta-residues: 2-aminocyclopentanecarboxylic acid (ACPC) and 2-aminocyclohexanecarboxylic acid (ACHC) with the syntax:

```
betafab transachc, (2S3R)ACHC
betafab cisacpc, (2R3S)ACPC
```

Capping groups can also be added:

N-terminal:

- ACE (acetyl)
- BUT (butyryl)

C-terminal:

- NME (N-methylamide)

```
betafab ace_valxal_nme, ACE, (S)B3hV, (S)B3hA, (S)B3hL, (2S3S)B23h(2A3A), (S)B3hV,␣
↪(S)B3hA, (S)B3hL, NME
```

3.3 Secondary structure

In addition to designating the amino-acids, the desired fold can also be given in the above notation. Each residue can be followed by a secondary structure descriptor in one of the following forms:

1. A square bracket-enclosed, space-separated triplet (pair) of floating point numbers, e.g. *(S)AQ[-57 -47]* or *(S)B3hA[-140.3 66.5 -136.8]*
2. The name of an entry in the *secondary structure database*, enclosed in curly braces, e.g. *(S)AQ{Alpha-helix}* or *(S)B3hA{H14M}*

3.3.1 Single-letter side-chain codes

This is a list of extended single-letter amino-acid sidechain codes recognized by *pmlbeta*

Letter	Side-chain
A	alanine
C	cysteine (neutral)
CM	cysteine (anionic)
D	aspartate (ionic)
DH	aspartic acid (protonated)
E	glutamate (ionic)
EH	glutamic acid (protonated)
G	glycine (i.e. no side-chain)
HD	histidine protonated on Nδ
HE	histidine protonated on Nϵ
HH	doubly protonated histidine
H	unprotonated histidine
I	isoleucine
K	lysine (charged)
KN	lysine (neutral)
L	leucine
M	methionine
N	asparagine
O	ornithine (charged)
ON	ornithine (neutral)
P	proline (only alpha-amino acid)
Q	glutamine
R	arginine
S	serine
T	threonine
V	valine
W	tryptophan
Y	tyrosine

CHAPTER 4

Secondary structure database

Under the hood, *pmlbeta* maintains a database of secondary structures, which is essentially a bunch of backbone dihedral angle triplets with labels. This is stored in the form of a Python *dict*, mapping string labels to tuples of floating point numbers.

By default the following are defined:

```
DEFAULT_HELIXTYPES = {
    'Z6M': (126.7, 62.6, 152.7),
    'Z6P': (-126.7, -62.6, -152.7),
    'Z8M': (47.5, 53.5, -104.3),
    'Z8P': (-47.5, -53.5, 104.3),
    'H8M': (76.8, -120.6, 52.7),
    'H8P': (-76.8, 120.6, -52.7),
    'H10M': (-77.5, -51.8, -75.1),
    'H10P': (77.5, 51.8, 75.1),
    'H12M': (92.3, -90.0, 104.6),
    'H12P': (-92.3, 90.0, -104.6),
    'H14M': (-140.3, 66.5, -136.8),
    'H14P': (140.3, -66.5, 136.8),
    'SM': (70.5, 176.2, 168.9),
    'SP': (-70.5, -176.2, -168.9),
    'Straight': (180, 180, 180),
    'Straight alpha': (180, None, 180),
    'Alpha-helix': (-57, None, -47),
    '3_10-helix': (-49, None, -26),
    'P-beta-sheet': (-119, None, 113),
    'AP-beta-sheet': (-139, None, 135),
}
```

Corresponding to the following:

Abbreviation	phi	theta	psi	Comments
Z6M	126.7°	62.6°	152.7°	from Beke et. al.(2006)
Z6P	-126.7°	-62.6°	-152.7°	
Z8M	47.5°	53.5°	-104.3°	
Z8P	-47.5°	-53.5°	104.3°	
H8M	76.8°	-120.6°	52.7°	
H8P	-76.8°	120.6°	-52.7°	
H10M	-77.5°	-51.8°	-75.1°	
H10P	77.5°	51.8°	75.1°	
H12M	92.3°	-90.0°	104.6°	
H12P	-92.3°	90.0°	-104.6°	
H14M	-140.3°	66.5°	-136.8°	
H14P	140.3°	-66.5°	136.8°	
SM	70.5°	176.2°	168.9°	
SP	-70.5°	-176.2°	-168.9°	
Straight	180°	180°	180°	
Straight alpha	180°	—	180°	
Alpha-helix	-57°	—	-47°	from PyMOL
P-beta-sheet	-119°	—	113°	
AP-beta-sheet	-139°	—	135°	
3_10-helix	-49°	—	-26°	

Editing this list is possible by the *corresponding GUI utility*, or by the *ssdb_** commands.

Changes to the secondary structure database are remembered in the next PyMOL session.

4.1 Python API documentation

CHAPTER 5

Residue Topology Recognition

Another useful functionality of *pmlbeta* is residue topology recognition. For actual work with molecular models an accurate designation of various portions of the molecule (“residues”, e.g. amino-acids or other building blocks) and atom naming is often required. While the models constructed by *betafab* are labeled according to the nomenclature of the CHARMM force field, re-labeling for a different molecular mechanics force field or labeling an unlabeled molecule (e.g. obtained from crystallography or NMR experiments) is sometimes needed. This package has the following methods:

- Given the N-terminal nitrogen and the C-terminal carbon atoms, command *renumber_peptide_chain* finds and marks the residues in a peptide chain (either α or β).
- Command *residue_topology_candidates* finds matching building block topologies from a GROMACS residue topology database file for each residue in the molecule using graph isomorphishm.
- Command *assign_residue_topology* assigns atom properties (name, residue name, partial charge) from a corresponding building block topology in a GROMACS topology database
- The functionality of the above two commands is implemented in a user-friendly graphical tool accessible from the menu.

CHAPTER 6

Commands

The following commands are defined by the *pmlbeta* plugin:

6.1 α/β-peptide Construction and Manipulation

betafab2 build a beta-peptide

fold_bp fold a beta-peptide into the desired secondary structure

select_bbb select the beta-backbone

6.2 Residue Topology Recognition

renumber_peptide_chain recognize and mark residues in a peptide chain

residue_topology_candidates find matching building block topologies from a GROMACS residue topology database file

assign_residue_topology assign atom properties from a matching building block topology

6.3 Input / Output

save_crd output to the extended CHARMM CRD format

save_gro output to the GRO format of GROMACS

save_g96 output to the G96 format used by GROMOS and GROMACS

6.4 Secondary structure database manipulation

ssdb_add add/edit an entry

ssdb_del remove an entry

ssdb_list list the entries

ssdb_dihedrals get the dihedral angles corresponding to an entry

ssdb_resetdefaults reset the secondary structure database to its default state

6.5 Miscellaneous

gmx_beta_backbone_dihedrals_selection create selection files for beta-peptide backbone dihedrals, understandabe by GROMACS

6.5.1 betafab2

6.5.2 fold_bp

6.5.3 select_bbb

6.5.4 save_crd

6.5.5 save_g96

6.5.6 save_gro

6.5.7 ssdb_add

6.5.8 ssdb_del

6.5.9 ssdb_list

6.5.10 ssdb_dihedrals

6.5.11 ssdb_resetdefaults

6.5.12 gmx_beta_backbone_dihedrals_selection

6.5.13 renumber_peptide_chain

6.5.14 residue_topology_candidates

6.5.15 assign_residue_topology

CHAPTER 7

Citing pmlbeta

If you find this plugin useful in your research work and publish a paper, the authors would be grateful if you could cite the corresponding article:

Wacha, András, and Tamás Beke-Somfai. 2021. "PmlBeta: A PyMOL Extension for Building β-Amino Acid Insertions and β-Peptide Sequences." SoftwareX 13 (January): 100654. https://doi.org/10.1016/j.softx.2020.100654.

CHAPTER 8

Reporting bugs

If you find bugs, errors or other unintentional/clumsy behaviour of the program, please file a bug report at:

https://gitlab.com/awacha/pmlbeta/issues/new

Before filing the report, please take care that you use the most up-to-date version: the bug might have already been fixed.

A helpful bug report should include:

- Operating system
- PyMOL version
- Python version
- Description of the error
- Expected behaviour
- Detailed steps on how to reproduce the bug
- If applicable, the error messages printed by PyMOL

CHAPTER 9

Indices and tables

- genindex
- modindex
- search