

---

# **pmlbeta Documentation**

***Release 1.0.10.dev0+gedba93e.d20200716***

**András Wacha**

**Jul 16, 2020**



---

## Contents:

---

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Requirements . . . . .	3
1.2	Installing from a pre-built package . . . . .	3
1.3	Manual installation . . . . .	3
<b>2</b>	<b>Quick start</b>	<b>5</b>
2.1	Command line . . . . .	5
2.2	Graphical user interface . . . . .	7
2.2.1	Betafab GUI . . . . .	7
2.2.2	Secondary structure editor . . . . .	8
<b>3</b>	<b><math>\beta</math>-amino acid residue naming</b>	<b>9</b>
3.1	General nomenclature . . . . .	9
3.2	Simplified nomenclature . . . . .	10
3.3	Secondary structure . . . . .	12
3.3.1	Single-letter side-chain codes . . . . .	12
<b>4</b>	<b>Secondary structure database</b>	<b>13</b>
4.1	Python API documentation . . . . .	14
<b>5</b>	<b>Commands</b>	<b>17</b>
5.1	$\alpha/\beta$ -peptide Construction and Manipulation . . . . .	17
5.2	Input / Output . . . . .	17
5.3	Secondary structure database manipulation . . . . .	17
5.4	Miscellaneous . . . . .	18
5.4.1	betafab2 . . . . .	18
5.4.2	fold_bp . . . . .	19
5.4.3	select_bbb . . . . .	19
5.4.4	save_crd . . . . .	20
5.4.5	save_g96 . . . . .	20
5.4.6	save_gro . . . . .	21
5.4.7	ssdb_add . . . . .	21
5.4.8	ssdb_del . . . . .	22
5.4.9	ssdb_list . . . . .	22
5.4.10	ssdb_dihedrals . . . . .	22
5.4.11	ssdb_resetdefaults . . . . .	23
5.4.12	gmx_beta_backbone_dihedrals_selection . . . . .	23

<b>6</b>	<b>Reporting bugs</b>	<b>25</b>
<b>7</b>	<b>Indices and tables</b>	<b>27</b>
	<b>Index</b>	<b>29</b>

*pmlbeta* is an extension for [PyMOL](#), containing utilities and algorithms for dealing with  $\beta$ -peptides. It has facilities for creating and manipulating geometries of natural  $\alpha$ - and artificial  $\beta$ - and even mixed  $\alpha/\beta$ -peptides.



This program has been developed by the [BioNano Platform](#) of the [Research Centre for Natural Sciences](#), Hungarian Academy of Sciences



### 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](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.





The main goal of *pmlbeta* is to provide tools for building molecular models of  $\beta$ -peptides (or even mixed  $\alpha/\beta$ -peptides). The idea is mostly taken from the *fab* command of PyMOL, which can create peptides in a desired secondary structure. The construction of  $\beta$ -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  $\alpha/\beta$ -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  $\beta^3$ -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”<sup>1</sup>, 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)

---

<sup>1</sup> 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  $\beta$ -peptide builder can be found in the Plugins menu of PyMOL. The main window is shown in the figure below.

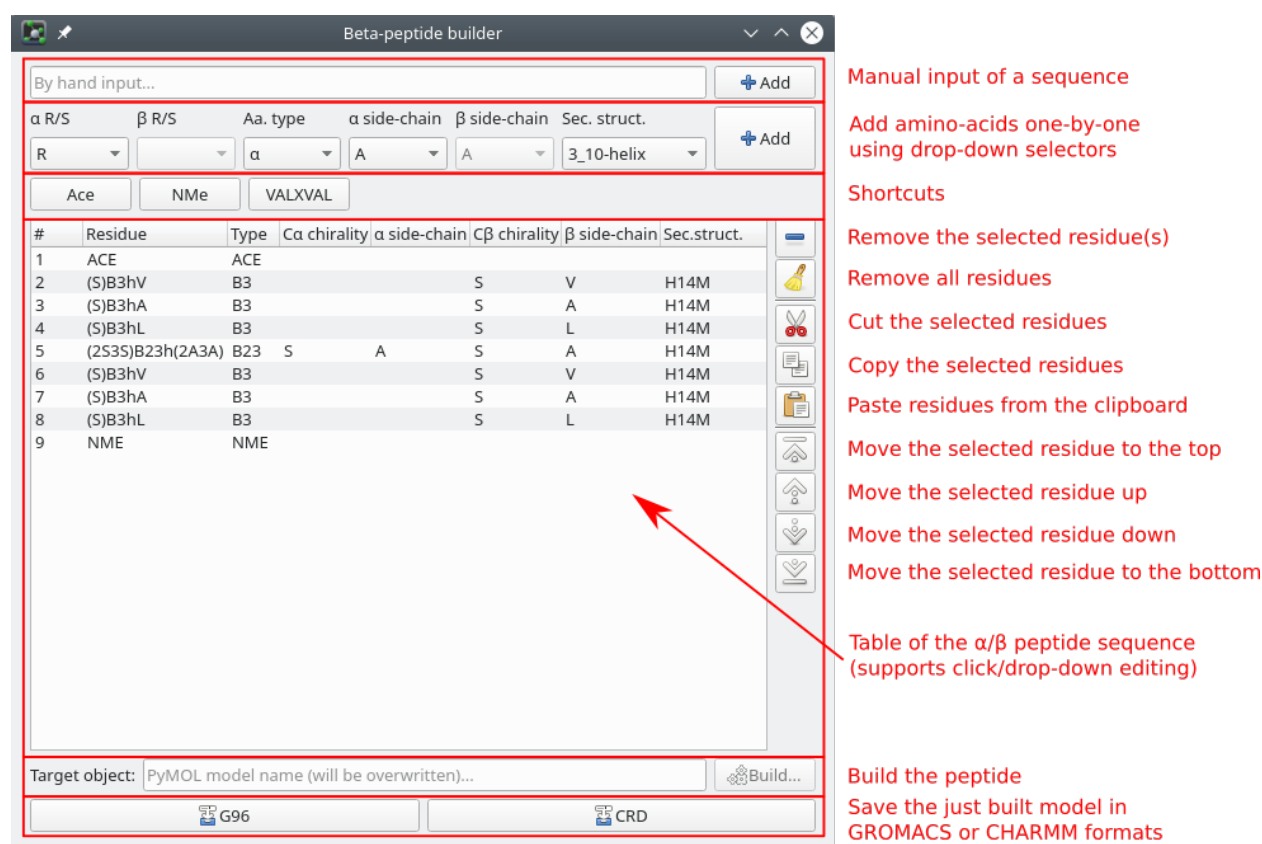


Fig. 1: The Betafab GUI window explained

Building blocks ( $\alpha$ - or  $\beta$ -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

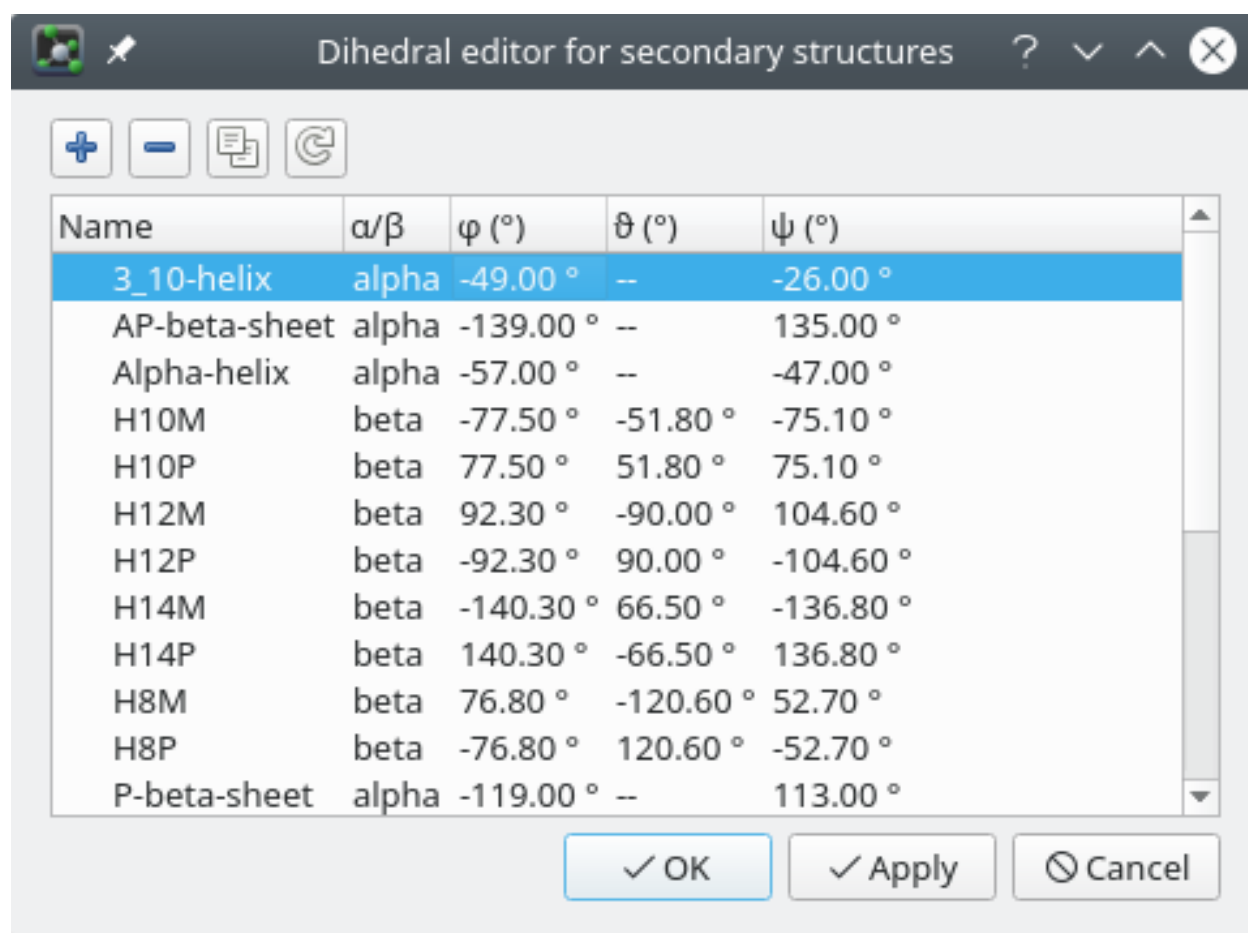


Fig. 2: The secondary structure dialog

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.

### 3.1 General nomenclature

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

$\beta^2$  Monosubstituted  $\beta$ -amino acid, the side-chain is on the  $\alpha$ -carbon.

$\beta^3$  Monosubstituted  $\beta$ -amino acid, the side-chain is on the  $\beta$ -carbon.

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

**$\beta$ -Alanine** A simple  $\beta$ -backbone, without side-chains

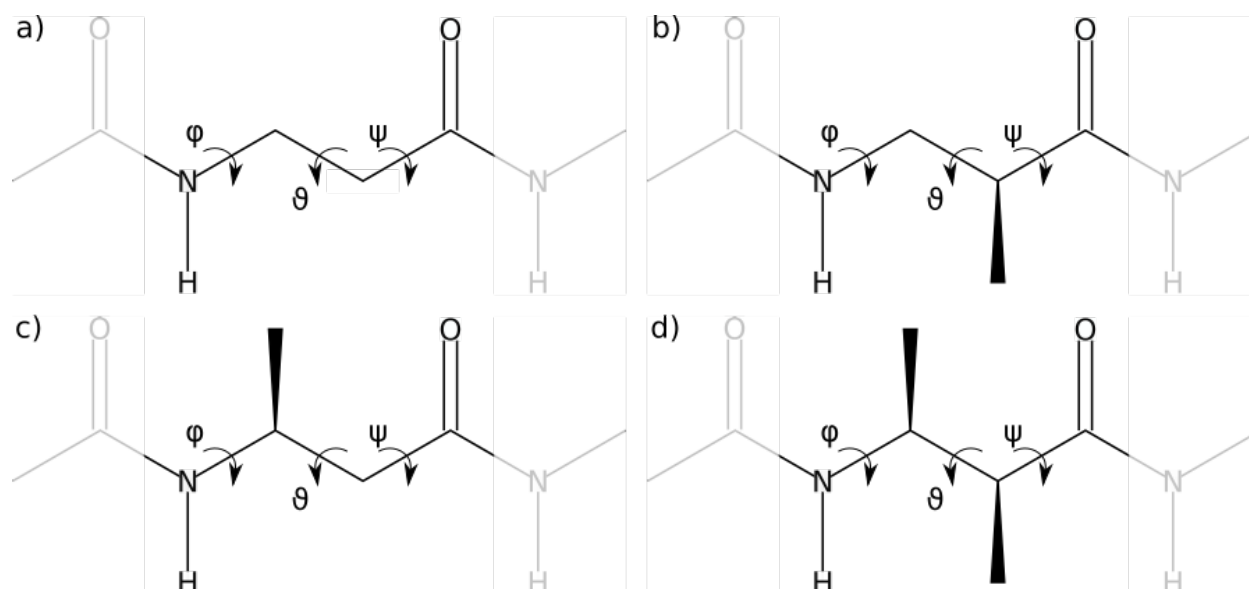


Fig. 1: The four kinds of  $\beta$ -amino acids: bare  $\beta$ -backbone or  $\beta$ -alanine (a),  $\beta^2$ -amino acid (b),  $\beta^3$ -amino acid (c) and  $\beta^{2,3}$ -amino acid (d)

In contrast to their natural counterparts,  $\beta$ -amino acids do not have a single, generally accepted nomenclature. We adopt here the following, widely used convention, which emphasizes the homology with  $\alpha$ -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  $\beta$ -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  $\beta^2$ ,  $\beta^3$  or  $\beta^{2,3}$ .

**<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  $\alpha$ -amino acids in this notation, too, with the following scheme: <chirality> $\alpha$ <side-chain designation>, where:

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

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

#### Examples:

##### monosubstituted:

- (*S*) $\beta^2$ hV: valine side-chain on the  $\alpha$ -carbon with *S* chirality
- (*R*) $\beta^3$ hR: arginine side-chain on the  $\beta$ -carbon with *R* chirality

##### disubstituted:

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

##### achiral (bare backbone):

- $\beta$ A

##### $\alpha$ -amino acids:

- (*S*) $\alpha$ V: L-valine
- (*R*) $\alpha$ W: 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  $\beta$ -peptide sequence, with the following simplification:

- superscripts are omitted
- instead of the greek letter  $\beta$  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

---

<sup>1</sup> Note that because for  $\beta$ -amino acids no internationally agreed convention exists on the *D*/*L* nomenclature, their chirality can only be specified using the unambiguous *S*/*R* notation.

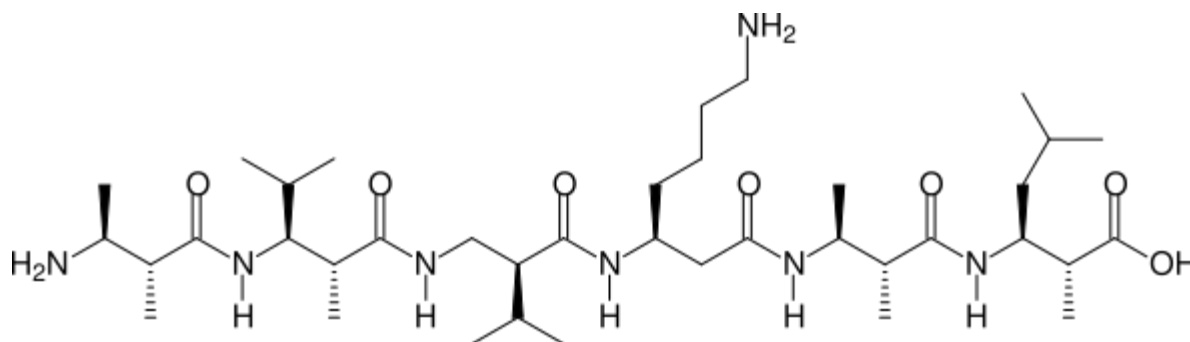


Fig. 2: (2*R*,3*S*) $\beta^{2,3}$ h(2*A*,3*A*) - (2*R*,3*S*) $\beta^{2,3}$ h(2*A*,3*V*) - (*S*) $\beta^2$ hV - (*S*) $\beta^3$ hK - (2*R*,3*S*) $\beta^{2,3}$ h(2*A*,3*A*) - (2*R*,3*S*) $\beta^{2,3}$ h(2*A*,3*L*)

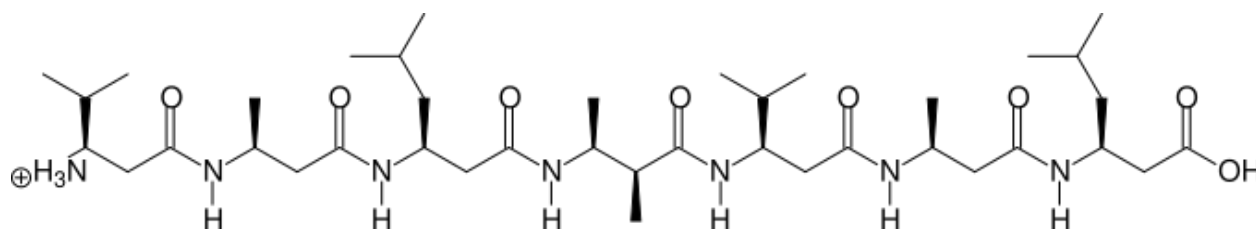


Fig. 3: (*S*) $\beta^3$ hV - (*S*) $\beta^3$ hA - (*S*) $\beta^3$ hL - (2*S*,3*S*) $\beta^{2,3}$ h(2*A*,3*A*) - (*S*) $\beta^3$ hV - (*S*) $\beta^3$ hA - (*S*) $\beta^3$ hL

- 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,  $\alpha$ -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 N $\delta$
HE	histidine protonated on N $\epsilon$
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



---

## 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°	from PyMOL
Straight alpha	180°	—	180°	
Alpha-helix	-57°	—	-47°	
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

Implement a database of secondary structures using the plugin preference store of PyMOL

**class** `pmlbeta.secstructdb.SecondaryStructureDB`

A database of the known secondary structures and the corresponding dihedral angles

**classmethod** `add(name: str, phi: float, theta: float, psi: float)`

Add or edit an entry in the secondary structure torsion angle database

### Parameters

- **name** (*str*) – the name of the secondary structure
- **phi** (*float*) – the first torsion angle
- **theta** (*float or None*) – the second torsion angle (None for alpha-amino acid secondary structures)
- **psi** (*float*) – the third torsion angle

**classmethod** `addDefaults()`

Add the default secondary structure types to the list

**classmethod** `dihedrals(name: str) → Tuple[float, Optional[float], float]`

Get the dihedral angles corresponding to a secondary structure.

**Parameters** **name** (*str*) – the name of the secondary structure

**Returns** the dihedral angles corresponding to that secondary structure, in degrees

**Return type** a tuple of three floats, the central one can be None

**Raises `KeyError`** – if the named entry does not exist

**classmethod `find`** (*phi*: float, *theta*: Optional[float], *psi*: float, *tolerance*: float = 0.5) → Optional[str]

Try to find a secondary structure in the database which matches the given torsion angles

#### Parameters

- **phi** (float) – the first torsion angle
- **theta** (float or None) – the second torsion angle (None for alpha-amino acids)
- **psi** (float) – the third torsion angle
- **tolerance** (float) – absolute tolerance in each angle

**Returns** the name of the closest matching secondary structure or None if no match

**Return type** str or None

**classmethod `getAll`** (*alpha*: bool = True, *beta*: bool = True) → Dict[str, Tuple[float, float, float]]

Get the list of secondary structures from the PyMOL plugin preferences

#### Parameters

- **alpha** (bool) – if secondary structures for alpha-peptides are requested
- **beta** (bool) – if secondary structures for beta-peptides are requested

**Returns** the known secondary structures and their backbone torsion angles

**Return type** a dict mapping names to triplets of floats

**classmethod `remove`** (*name*: str)

Remove an entry in the secondary structure torsion angle database

**Parameters** **name** (str) – the name of the secondary structure

pmlbeta.secstructdb.**ssdb\_add** (*entryname*: str, *phi*: Union[str, float], *theta*: Union[str, float], *psi*: Union[str, float, None] = None, *\_self*=None)

#### DESCRIPTION

Add/modify an entry in the secondary structure database

#### USAGE

ssdb\_add entryname, phi, theta, psi # for beta-amino acids

or

ssdb\_add entryname, phi, psi # for alpha-amino acids

#### ARGUMENTS

**entryname** = string: the name of the entry

**phi = float: the first (N-terminal) backbone dihedral angle: (C-, N, CA, C) for alpha-amino acids and (C-, N, CB, CA) for beta-amino acids**

**theta = float: the middle backbone dihedral angle for beta-amino acids: (N, CB, CA, C)**

**psi = float: the last (C-terminal) backbone dihedral angle: (N, CA, C, N+) for alpha-amino acids and (CB, CA, C, N+) for beta-amino acids**

#### NOTES

changes are automatically saved for further PyMOL sessions.

#### SEE ALSO

ssdb\_del, ssdb\_list, ssdb\_resetdefaults, ssdb\_dihedrals

pmlbeta.secstructdb.**ssdb\_del** (entryname: str, \_self=None)

#### DESCRIPTION

Remove an entry from the secondary structure database

#### USAGE

ssdb\_del entryname

#### ARGUMENTS

entryname = string: the name of the entry

#### NOTES

changes are automatically saved for further PyMOL sessions.

#### SEE ALSO

ssdb\_add, ssdb\_list, ssdb\_resetdefaults, ssdb\_dihedrals

pmlbeta.secstructdb.**ssdb\_dihedrals** (entryname: str, \_self=None)

#### DESCRIPTION

**Print the dihedral angles corresponding to a given entry** secondary structure database

#### USAGE

ssdb\_dihedrals entryname

#### ARGUMENTS

entryname = string: the name of the entry

#### SEE ALSO

ssdb\_add, ssdb\_del, ssdb\_list, ssdb\_resetdefaults

pmlbeta.secstructdb.**ssdb\_list** (\_self=None)

#### DESCRIPTION

Lists the entries in the secondary structure database

#### USAGE

ssdb\_list

#### SEE ALSO

ssdb\_add, ssdb\_del, ssdb\_resetdefaults, ssdb\_dihedrals

pmlbeta.secstructdb.**ssdb\_resetdefaults** (\_self=None)

#### DESCRIPTION

Add back the default entries to the secondary structure database

#### USAGE

ssdb\_resetdefaults

#### NOTES

changes are automatically saved for further PyMOL sessions.

#### SEE ALSO

ssdb\_add, ssdb\_del, ssdb\_list, ssdb\_dihedrals

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

## 5.1 $\alpha/\beta$ -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

## 5.2 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

## 5.3 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

## 5.4 Miscellaneous

*gmx\_beta\_backbone\_dihedrals\_selection* create selection files for beta-peptide backbone dihedrals, understandable by GROMACS

### 5.4.1 betafab2

`pmlbeta.betafab2.betafab2 (objname, *args)`

Construct an alpha/beta peptide

Amino acids can be given in the following way:

- 1) **beta-2 amino-acid:** (`{S|R}`)B2h<sc> e.g. (S)B2hV, (R)B2hA
- 2) **beta-3 amino-acid:** (`{S|R}`)B3h<sc> e.g. (S)B3hL, (R)B3hR
- 3) **beta-2,3 amino-acid:** (`2{S|R}3{S,R}`)B23h(2<sc1>3<sc2>) e.g. (2S3R)B23h(2A3L)
- 4) **bare beta-backbone:**  
**BA or (2S3R)B23h(2G3G) (with any combination of R and S in the first pair of parentheses)**
- 5) **alpha-amino acid:** (`{S|R|L|D}`)A<sc> e.g. (S)AL, (R)AV, (L)AA, (D)AC
- 6) **capping groups:** ACE: acetyl on the N-terminus (must be first) BUT: butyl on the N-terminus (must be first) NME: N-methylamide on the C-terminus (must be last)
- 7) **cyclic beta-residues:** (`2{S|R}3{S|R}`)ACHC (2-aminocyclohexanecarboxylic acid) e.g. (2S3R)ACHC (`2{S|R}3{S|R}`)ACPC (2-aminocyclopentanecarboxylic acid) e.g. (2S3R)ACPC

where <sc> is the single-letter abbreviation of a natural peptide sidechain. The following are supported:

A, C, D, E, F, G, I, K, L, M, N, O, Q, R, S, T, V, W, Y.

In addition, the following variants are also understood:

**CM :** deprotonated cysteine  
**DH :** aspartic acid  
**EH :** glutamic acid  
**HD :** histidine protonated on the delta-nitrogen  
**HE :** histidine protonated on the epsilon-nitrogen  
**HH :** histidine protonated on both nitrogens  
**KN :** neutral lysine  
**ON :** neutral ornithine

The desired secondary structure can be given as well after each amino-acid in two possible ways:

1. three (or two) floating point numbers in square brackets, e.g. (S)AQ[-57 -47] or (S)B3hV[-140.3 66.5 -136.8]
2. referencing an entry of the secondary structure database in curly braces, e.g. (S)AQ{Alpha-helix} or (S)B3hV{H14M}

**Parameters**

- **objname** (*str*) – the name PyMOL will know about the resulting peptide
- **args** (*str*) – amino-acid abbreviations

**Returns** the constructed peptide

**Return type** BetaPeptide

### 5.4.2 fold\_bp

`pmlbeta.setbetahelix.fold_bp(sstype: str, selection: str = '(all)')`

**DESCRIPTION**

Adjust the torsion angles of an alpha- or beta-peptide to fold it into the desired secondary structure

**USAGE**

`fold_bp sstype [, selection]`

**ARGUMENTS**

**sstype = string: the desired secondary structure. The following possibilities exist:**

1. a name from the secondary structure database (see command `ssdb_list`)
2. two or three (for alpha- and beta-amino acids, respectively) space-separated floating point numbers in parentheses, corresponding to the backbone dihedral angles

The such given secondary structure will be applied to all residues in the selection.

Additionally, a space-separated list of the same length as the residues in the selection can be given in square brackets, containing entry names or angle tuples (or a mix of them), corresponding to the residues.

**selection = the selection to operate on. Must be a single peptide chain with** unique and consecutive residue IDs (default: all)

**EXAMPLES**

`fold_bp H14M, model valxval`

`fold_bp (-140.3 66.5 -136.8), model valxval`

`fold_bp [(-140.3 66.5 -136.8) (180 180 180) H14M], model tripeptide`

**SEE ALSO**

`ssdb_add`, `ssdb_del`, `ssdb_dihedrals`, `ssdb_resetdefaults`, `ssdb_list`

### 5.4.3 select\_bbb

`pmlbeta.utils.select_bbb(selectionname='bbone', originalselection='all')`

**DESCRIPTION**

Select the backbone of beta-peptides

**USAGE**

`select_beta_backbone name [, originalselection]`

**ARGUMENTS**

**name** = name of the new selection

originalselection = superset in which the backbone will be searched

#### 5.4.4 save\_crd

`pmlbeta.savegro.save_crd(filename, selection='(all)', sort=True)`

##### DESCRIPTION

“save\_crd” saves content to an expanded CHARMM CRD file in a similar fashion as “save”

##### USAGE

`save_crd filename [, selection ]`

##### ARGUMENTS

filename = string: file path to be written

selection = string: atoms to save {default: (all)}

##### NOTES

In contrast to “save”, only the current state can be saved as the CRD format is a single-structure format.

Atoms are written with increasing ID.

##### SEE ALSO

load, save

#### 5.4.5 save\_g96

`pmlbeta.savegro.save_g96(filename, selection='(all)', bx=None, by=None, bz=None, sort=False)`

##### DESCRIPTION

“save\_g96” saves content to a .g96 file in a similar fashion as “save”

##### USAGE

`save_g96 filename [, selection [, bx [, by [, bz ]]]]`

##### ARGUMENTS

filename = string: file path to be written

selection = string: atoms to save {default: (all)}

bx = float: rectangular box size X dimension in Angströms

by = float: rectangular box size Y dimension in Angströms

bz = float: rectangular box size Z dimension in Angströms

##### NOTES

In contrast to “save”, only the current state can be saved as the GROMOS-96 format is a single-structure format.

If any of bx, by and bz is not supplied, the box will be determined from the `get_symmetry()` command.

Atoms are written with increasing ID.

##### SEE ALSO

load, save



### 5.4.6 save\_gro

`pmlbeta.savegro.save_gro(filename, selection='(all)', bx=None, by=None, bz=None, sort=False)`

#### DESCRIPTION

“save\_gro” saves content to a .gro file in a similar fashion as “save”

#### USAGE

```
save_gro filename [, selection [, bx [, by [, bz ]]]]
```

#### ARGUMENTS

**filename** = string: file path to be written

**selection** = string: atoms to save {default: (all)}

**bx** = float: rectangular box size X dimension in Angströms

**by** = float: rectangular box size Y dimension in Angströms

**bz** = float: rectangular box size Z dimension in Angströms

#### NOTES

In contrast to “save”, only the current state can be saved as the GRO format is a single-structure format.

If any of bx, by and bz is not supplied, the box will be determined from the `get_symmetry()` command.

Atoms are written with increasing ID.

#### SEE ALSO

load, save

### 5.4.7 ssdb\_add

`pmlbeta.secstructdb.ssdb_add(entryname: str, phi: Union[str, float], theta: Union[str, float], psi: Union[str, float, None] = None, _self=None)`

#### DESCRIPTION

Add/modify an entry in the secondary structure database

#### USAGE

```
ssdb_add entryname, phi, theta, psi # for beta-amino acids
```

or

```
ssdb_add entryname, phi, psi # for alpha-amino acids
```

#### ARGUMENTS

**entryname** = string: the name of the entry

**phi** = float: the first (N-terminal) backbone dihedral angle: (C-, N, CA, C) for alpha-amino acids and (C-, N, CB, CA) for beta-amino acids

**theta** = float: the middle backbone dihedral angle for beta-amino acids: (N, CB, CA, C)

**psi** = float: the last (C-terminal) backbone dihedral angle: (N, CA, C, N+) for alpha-amino acids and (CB, CA, C, N+) for beta-amino acids

#### NOTES

changes are automatically saved for further PyMOL sessions.

SEE ALSO

ssdb\_del, ssdb\_list, ssdb\_resetdefaults, ssdb\_dihedrals

### 5.4.8 ssdb\_del

pmlbeta.secstructdb.**ssdb\_del** (entryname: str, \_self=None)

DESCRIPTION

Remove an entry from the secondary structure database

USAGE

ssdb\_del entryname

ARGUMENTS

entryname = string: the name of the entry

NOTES

changes are automatically saved for further PyMOL sessions.

SEE ALSO

ssdb\_add, ssdb\_list, ssdb\_resetdefaults, ssdb\_dihedrals

### 5.4.9 ssdb\_list

pmlbeta.secstructdb.**ssdb\_list** (\_self=None)

DESCRIPTION

Lists the entries in the secondary structure database

USAGE

ssdb\_list

SEE ALSO

ssdb\_add, ssdb\_del, ssdb\_resetdefaults, ssdb\_dihedrals

### 5.4.10 ssdb\_dihedrals

pmlbeta.secstructdb.**ssdb\_dihedrals** (entryname: str, \_self=None)

DESCRIPTION

**Print the dihedral angles corresponding to a given entry** secondary structure database

USAGE

ssdb\_dihedrals entryname

ARGUMENTS

entryname = string: the name of the entry

SEE ALSO

ssdb\_add, ssdb\_del, ssdb\_list, ssdb\_resetdefaults

### 5.4.11 ssdb\_resetdefaults

`pmlbeta.secstructdb.ssdb_resetdefaults(_self=None)`

#### DESCRIPTION

Add back the default entries to the secondary structure database

#### USAGE

`ssdb_resetdefaults`

#### NOTES

changes are automatically saved for further PyMOL sessions.

#### SEE ALSO

`ssdb_add`, `ssdb_del`, `ssdb_list`, `ssdb_dihedrals`

### 5.4.12 gmx\_beta\_backbone\_dihedrals\_selection

`pmlbeta.gmxselections.gmx_beta_backbone_dihedrals_selection` (*outputfile*: *str*,  
*startresi*: *int*,  
*endresi*: *int*,  
*nterm\_ace*: *bool*  
= *True*, *cterm\_nme*:  
*bool* = *True*)

Write a selection file for GROMACS containing the backbone torsion angles of a beta-peptide

#### Parameters

- **outputfile** (*str*) – output file name
- **startresi** (*int*) – first residue number
- **endresi** (*int*) – last residue number



---

### 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 7

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`





## B

`betafab2()` (in module *pmlbeta.betafab2*), 18

## F

`fold_bp()` (in module *pmlbeta.setbetahelix*), 19

## G

`gmx_beta_backbone_dihedrals_selection()`  
(in module *pmlbeta.gmxselections*), 23

## S

`save_crd()` (in module *pmlbeta.savegro*), 20

`save_g96()` (in module *pmlbeta.savegro*), 20

`save_gro()` (in module *pmlbeta.savegro*), 21

`select_bbb()` (in module *pmlbeta.utils*), 19

`ssdb_add()` (in module *pmlbeta.secstructdb*), 21

`ssdb_del()` (in module *pmlbeta.secstructdb*), 22

`ssdb_dihedrals()` (in module *pmlbeta.secstructdb*),  
22

`ssdb_list()` (in module *pmlbeta.secstructdb*), 22

`ssdb_resetdefaults()` (in module *pml-*  
*beta.secstructdb*), 23