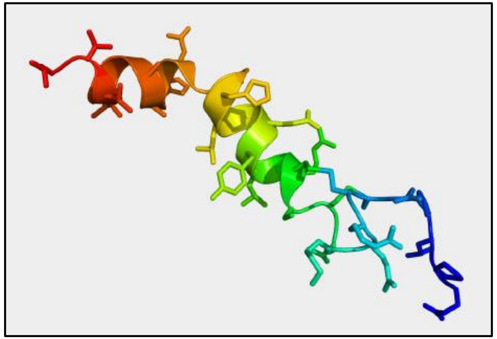


pyPept: a python library to analyze peptides using BILN and RDKit representations

Rodrigo Ochoa, J.B Brown, Alexander Weber, Thomas Fox

Computational Chemistry, Biberach

RDKit UGM – October 2022

Boehringer Ingelheim

Motivation: BILN and peptide-focused packages

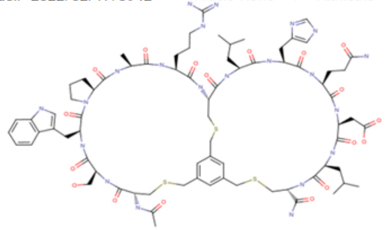
- BILN is an easy, human-readable line notation to describe even complex peptides
  - BILN allows all kind of text-based searches, comparisons, and analyses with peptides, in addition to generate canonical representations for complex branched molecules

RETURN TO ISSUE | < PREV | **APPLICATION NOTE** | NEXT >

BILN: A Human-Readable Line Notation for Complex Peptides

Thomas Fox*, Michael Bieler, Peter Haebel, Rodrigo Ochoa, Stefan Peters, and Alexander Weber

Cite this: *J. Chem. Inf. Model.* 2022, 62, 17, 3942– | Article Views | Altmetric | Citations



=

Ac-C(1,3)-S-W-P-A-R-C(2,3)-L-H-Q-D-L-C(3,3)-NH2.135TMB(1,1)(2,2)(3,3)

(Fox et. al., JCIM 2022)

https://github.com/rochoa85/BILN-converter

```
usage: BILN.py [-h]
               (--biln text | --helm text | --table_biln filename | --table_helm filename)
               [--logfile filename] [-v]
```

BILN <-> HELM conversion

A-C(1,3)-D-F-G-P-K(2,3)-L-M-C(1,3)-A.Ac-G(2,2)

⇕

PEPTIDE1{A.C.D.F.G.P.K.L.M.C.A}|PEPTIDE2{[Ac].G}$PEPTIDE1,PEPTIDE1,2:R3-10:R3|PEPTIDE1,PEPTIDE2,7:R3-2:R2$$$

BILN + RDKit for peptide analysis

BILN

H-Aib-E-G-T-A-K(1,3)-E-F-R-G.C18DA-gGlu-OEG-OEG(1,2)

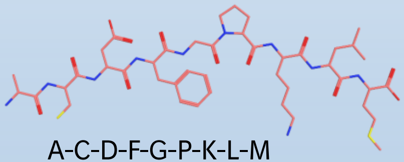
Package to analyze peptide and run additional analysis

⇩

pyPept

- Enriched peptide data
- Series annotation
- Peptide property calculation
- SAR interpretation
- MMP analysis of peptides
- Support peptide design

Conversion to 2D molecule

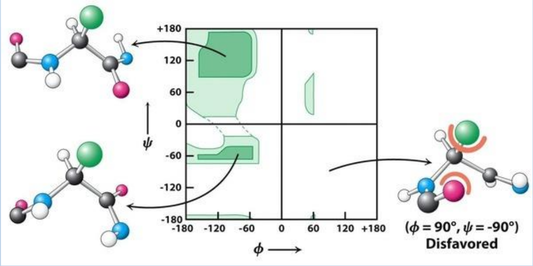


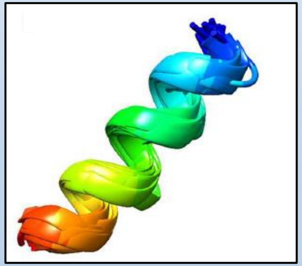
A-C-D-F-G-P-K-L-M

RDKit

Open-Source Cheminformatics and Machine Learning

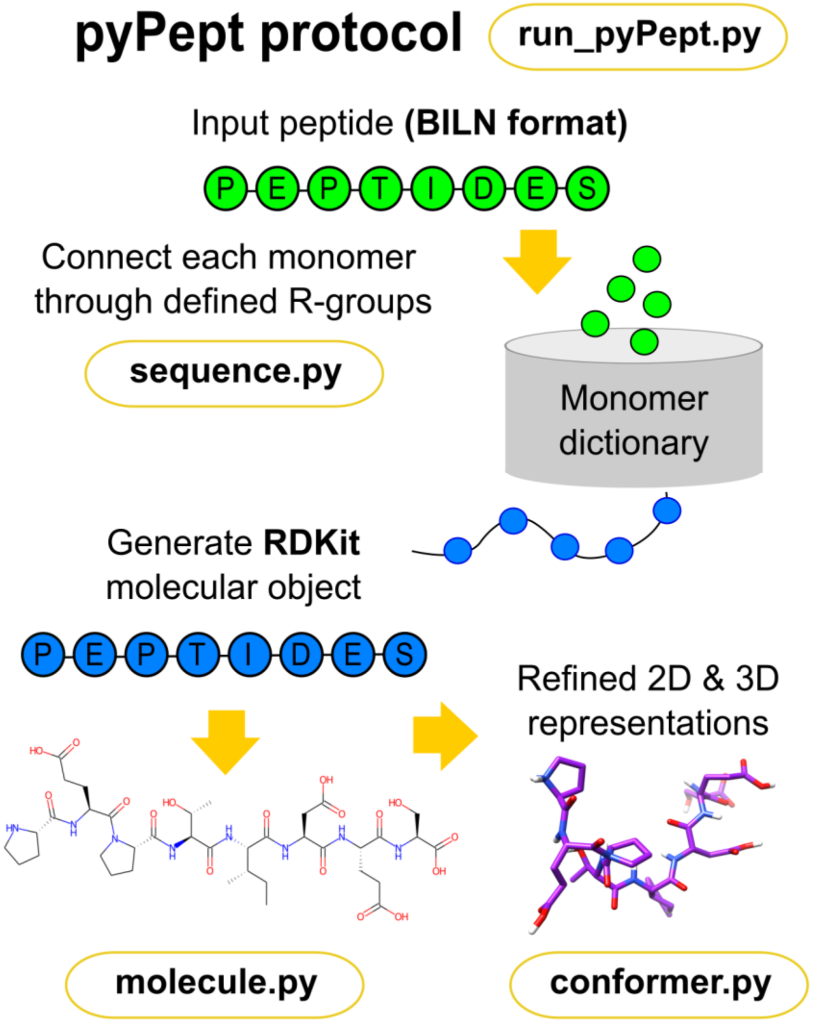
Generate conformers





pyPept

- A self-contained python package to read peptides using the BILN format and a monomer dictionary
- Complex and modified peptides can be analyzed, and 2D and 3D RDKit representations can be generated
- **The code will be released soon through the BI GitHub repository**

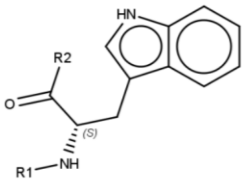


sequence.py: read the monomer dictionary

- A dictionary with RDKit representations of the monomers is used to assign the possible connection points using atom indexes

Monomer dictionary

(adapted from open HELM monomer set)



- Detailed R-groups
- Abbreviations (symbols and PDB codes)
- Atom indexes (derived from RDKit)

- **A public HELM monomer repository is adapted for pyPept**

https://github.com/PistoiaHELM/HELMMonomerSets

HELM Monomer Sets

This repository contains the monomer sets and file format definition that the HELM project uses.

The HELM project recommends that all HELM users adopt HELMCoreLibrary as the core of their monomer sets even if they then add monomers to their own copy. This will increase interoperability of HELM across organisations.

The HELMCoreLibrary was developed by analyising public datasets with particular help from Evan Bolton of PubChem and Anna Gaulton and Patricia Bento of EMBL-EBI.

```
Symbol,SMILES,Name,PDB_code
A,C[C@H](N)C(=O)O,Alanine,ALA
C,N[C@@H](CS)C(=O)O,Cysteine,CYS
D,N[C@@H](CC(=O)O)C(=O)O,Aspartic acid,ASP
E,N[C@@H](CCC(=O)O)C(=O)O,Glutamic acid,GLU
F,N[C@@H](Cc1ccccc1)C(=O)O,Phenylalanine,PHE
G,NCC(=O)O,Glycine,GLY
```

```
Pyr,O=C1CC[C@@H](C(=O)O)N1,Pyroglutamic acid,PYK
D_Pyr,O=C1CC[C@H](C(=O)O)N1,D-Pyroglutamic acid,DG5
Phe_3Cl,N[C@@H](Cc1cccc(Cl)c1)C(=O)O,3-Chloro-L-phenylalanine,PAA
Phe_4Cl,N[C@@H](Cc1ccc(Cl)cc1)C(=O)O,4-Chloro-L-phenylalanine,PBJ
Phe_4NH2,Nc1ccc(C[C@H](N)C(=O)O)cc1,p-Aminophenylalanine,PD0
Phg,N[C@H](C(=O)O)c1ccccc1,2-Phenylglycine,PEN
Ser_tBu,CC(C)(C)OC[C@H](N)C(=O)O,O-tert-Butyl-L-serine,SED
Tyr_Bn,N[C@@H](Cc1ccc(OCc2ccccc2)cc1)C(=O)O,O-Benzyl-L-tyrosine,TYH
Tza,N[C@@H](Cc1cscn1)C(=O)O,3-Thiazolylalanine,TZI
```

- Dataframe with key atom indexes

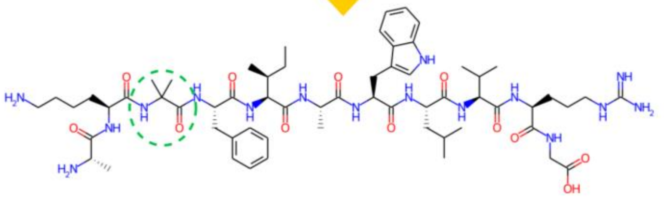
```
       m_abbr m_type    m_subtype  ...  m_attachmentPointIdx
                                   ...
            A     aa      natural  ...    [2, 3, None, None]
            C     aa      natural  ...       [0, 4, 3, None]
            D     aa      natural  ...       [0, 5, 3, None]
            E     aa      natural  ...       [0, 6, 4, None]
            F     aa      natural  ...    [0, 9, None, None]
          ...    ...          ...  ...                   ...
  D_hArg_Et2     aa  non-natural  ...  [10, 11, None, None]
    D_Met_S_O     aa  non-natural  ...    [6, 7, None, None]
```

molecule.py: connections and 2D representation

- The monomers are connected, and explicit bonds are assigned to the RDKit object

- For 2D representation, the 2D dihedrals are fixed. An in-house method and the rdDepictor modules are available

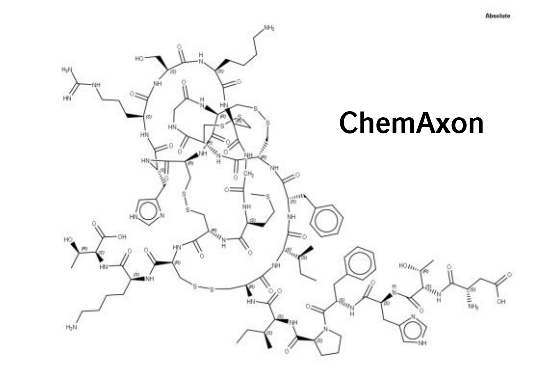
A-K-**Aib**-F-I-A-W-L-V-R-G **(BILN format)**

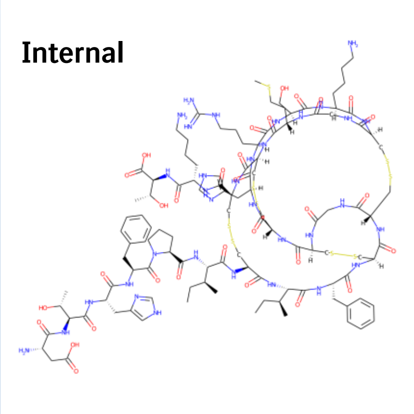


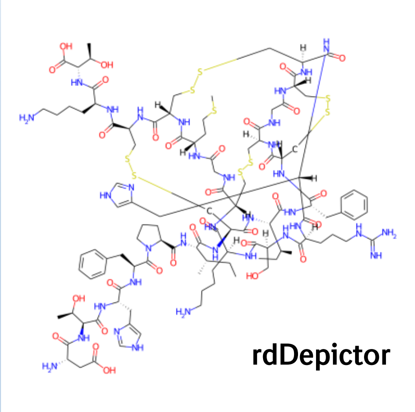
RDKit object (fixed 2D dihedrals)

However, there are complex cases …

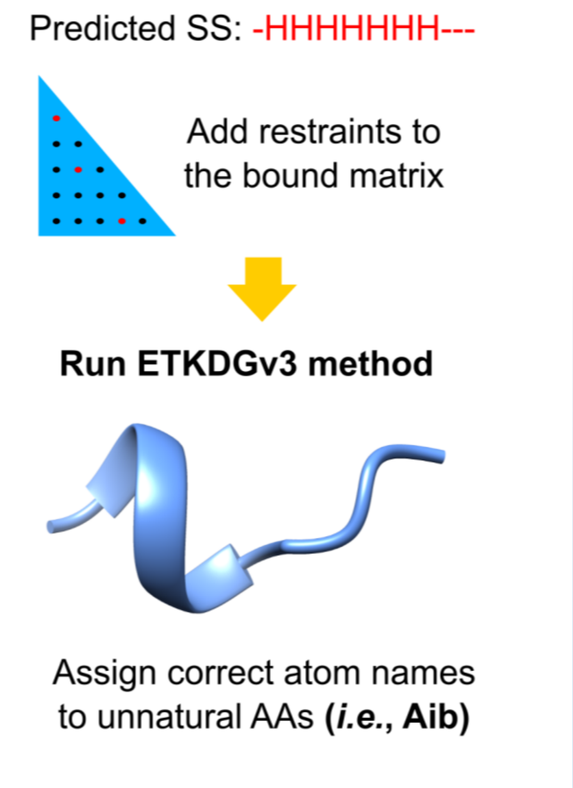
Hepcidin: D-T-H-F-P-I-C(1,3)-I-F-C(2,3)-C(3,3)-G-C(2,3)-C(4,3)-H-R-S-K-C(3,3)-G-M-C(4,3)-C(1,3)-K-T



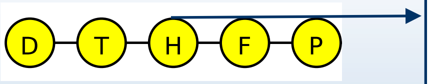




conformer.py: conformer prediction (ideal for peptides < 20 AA)



- A secondary structure (SS) predictor was developed by comparing fragments of the query with peptides having 3D structures in the PDB (8681 peptides with annotated SS using DSSP)



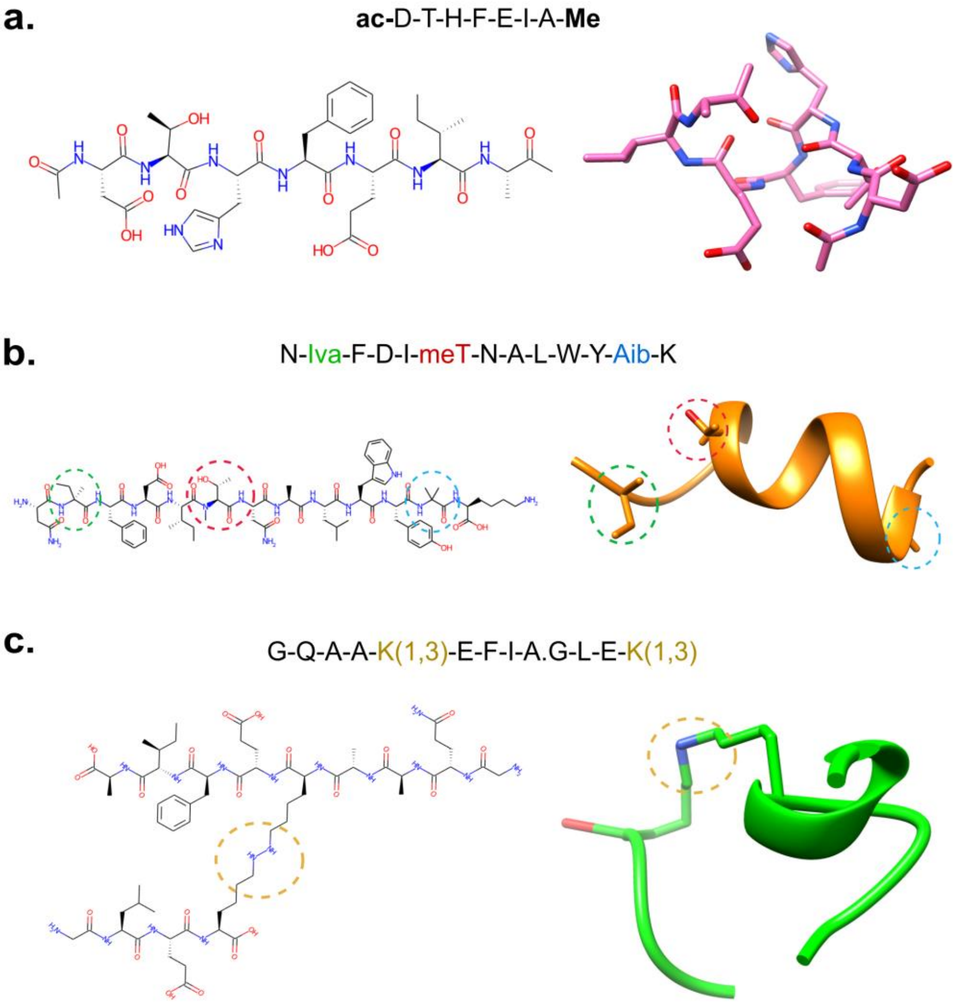
It will have multiple categories: Helix (H), Strand/Sheet (E), Coil (C), Turn (T), based on a profile with the most frequent elements

- **A PDB file with correct residue and atom assignation can then be subjected to simulations to improve the prediction**

```
ATOM     33  CB1 AIB A   3      -2.089  -3.998   5.040
ATOM     34  CA  AIB A   3      -0.753  -3.655   4.352
ATOM     35  N   AIB A   3       0.254  -3.327   5.379
ATOM     36  C   AIB A   3      -0.997  -2.464   3.442
ATOM     37  O   AIB A   3      -1.208  -2.644   2.209
ATOM     38  CB2 AIB A   3      -0.282  -4.890   3.554
ATOM     39 HB11 AIB A   3      -1.971  -4.885   5.700
ATOM     40 HB12 AIB A   3      -2.875  -4.221   4.288
ATOM     41 HB13 AIB A   3      -2.433  -3.147   5.665
ATOM     42  H   AIB A   3      -0.001  -3.448   6.387
ATOM     43 HB21 AIB A   3      -1.069  -5.227   2.846
ATOM     44 HB22 AIB A   3      -0.048  -5.728   4.246
ATOM     45 HB23 AIB A   3       0.632  -4.663   2.967
```

Examples

- The code provides a list of monomers that can be embedded, including capping groups
- For modified peptides, natural analogs are used to predict the secondary structure
- The BILN format allows the inclusion of branches and other chemical extensions



Code insights: waiting for release soon

pyPept

A python library to analyze peptides using BILN and RDKit representations

- From publication "pyPept: a python library to analyze peptides using BILN and RDKit representations"
- Journal of Cheminformatics, 2022
- Authors: Rodrigo Ochoa, J.B Brown, Alexander Weber, Thomas Fox

```
python run_pyPept.py --biln 'ac-D-T-H-F-E-I-A-am' --depiction rdkit
```

```
# Create the Sequence object
biln = args.biln
logger and logger.info("1. Processing the BILN sequence {}".format(biln))
s = Sequence(biln, path=args.path)

# Loop wit the included monomers
mm = s.s_monomers
for i, monomer in enumerate(mm):
    mol = monomer['m_romol']

# Generate the RDKit object
logger and logger.info("2. Creating the RDKit object")
m = Molecule(s,args.depiction)
romol = m.getMolecule(fmt='ROMol')
print("The SMILES of the peptide is: {}".format(Chem.MolToSmiles(romol)))
Draw.MolToFile(romol, '{}.png'.format(args.output), size=(1200, 1200))

# Create the peptide conformer with correct atom names and SS
logger and logger.info("3. Predicting the peptide conformer with correct atom names and secondary structure")
s = correct_PDB_atoms(s, biln, path=args.path)
romol = generate_conformer(s, biln, generatePDB=True, output_name=args.output, path=args.path)
```

Quick installation

We recommend to create a conda environment where RDKit can be installed using the following command

```
conda install -c rdkit rdkit
```

After that, pyPept can be easilly installed using the `setup.py` with:

```
python setup.py install
```

How to run it

Using the command-line script provided

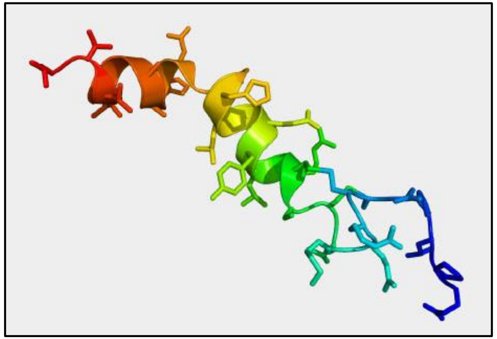
The script `run_pyPept.py` has the following arguments:

```
usage: run_pyPept.py [-h] --biln text [--depiction text] [--path text] [--output text] [--logfile filename] [

optional arguments:
  -h, --help          show this help message and exit

Main options:
  --biln text         BILN string with the peptide information (REQUIRED).
  --depiction text    Method to generate the 2D image. Only two options: local or rdkit.
  --path text         Path to the data folder if it is called externally. By the default it will look into th
  --output text       Name used in the output files.

Logging options:
  --logfile filename  Output messages to given logfile, default is stderr.
  -v, --verbose       Increase output verbosity
```



For more information:

rodrigo.ochoa@boehringer-ingelheim.com

Boehringer Ingelheim