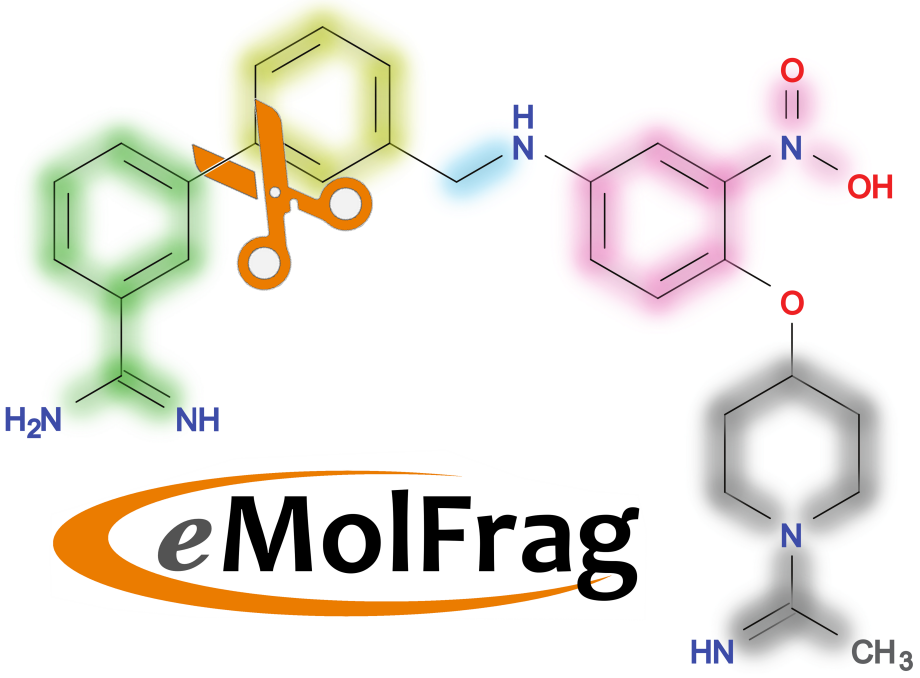

*e*MolFrag User Guide

Tairan Liu

Misagh Naderi

Feb 09, 2017

Disclaimer and Acknowledgements

---

*e*MolFrag was developed by Tairan Liu as part of a collaboration between the Computational Systems Biology Group in the Department of Biological Sciences and the Division of Computer Science and Engineering at Louisiana State University.

Email address:

Tairan Liu: tliu7@lsu.edu
Misagh Naderi: mnader5@lsu.edu
Chris Alvin: calvin@bradley.edu
Supratik Mukhopadhyay: supratik@csc.lsu.edu
Michal Brylinski: supratik@csc.lsu.edu

Contents

1 Introduction **1**

2 Installation and Setup **2**

3 Using *e*MolFrag **3**

4 Output **5**

5 Online Resources **11**

1 Introduction

*e*MolFrag [5] is a new open source software to decompose organic compounds into non-redundant fragments retaining molecular connectivity information.

The code has been developed in Python. In order to perform the fragmentation process, *e*MolFrag utilizes BRICS algorithm [2] implemented in the RDKit [4] Python module. Although the resulting fragments can be paired with a variety of virtual molecular synthesis tool, *e*MolFrag is specfically optimized to work with the software *e*Synth [6].

The following sections will give an overview of the different aspects of *e*MolFrag.

2 Installation and Setup

Prerequisites:
1. Python (either 2 or 3)
2. RDKit 2015.09.2 or newer (2016.03.3 has been tested). It is recommended to use Anaconda to install RDKit and use the following command: "conda install -c rdkit rdkit=2015.09.2".
3. pkcombu [3]
4. *e*MolFrag scripts
5. Openbabel 2.3.1 [7] (Optional)

Installation:
Use ConfigurePath.py to configure paths. The paths are only needed to be set prior to first run as long as the actual paths are unchanged. After the script starts, instructions will be given for setting paths.

1. The first path is for eMolFrag scripts. The absolute path to the scripts folder is needed.
2. The second path is for pkcombu. The absolute path to pkcombu to be used is needed.

For example:

```
$ python /.../ConfigurePath.py # run Configure path,
   use absolute path
$ # step 1: assuming that path to eMolFrag.py is /.../
   eMolFrag_201x_xx_xx_xx/eMolFrag.py, type: /.../
   eMolFrag_201x_xx_xx_xx/
$ # step 2: assuming that path to pkcombu is /.../
   pkcombu, type: /.../pkcombu
```

3 Using *e*MolFrag

Run scripts to process data:

```
$ /Path_to_Python/python /Path_to_scripts/eMolFrag.py -
    i /Path_to_input_directory/ -o /
    Path_to_output_directory/ -p Number-Of-Cores -m
    Output-selection -c Output-format
```

Table 1: Input segments description.

Term	Description
/Path_to_Python/python	May be simplified to python.
/Path_to_scripts/eMolFrag.py	Main path to the scripts, relative path is also acceptable.
/Path_to_input_directory/	Path of the directory which contains input mol2 files, relative path is also acceptable.
/Path_to_output_directory/	Path of the directory for output, relative path is also acceptable.
Number-Of-Cores	Number of processes created in parallel step. It is better to set this parameter no larger than the number of cores of the system/node/cluster.
Output-selection	Output selection: removing redundancy (or not), keeping temporary files (or not).
Output-format	Extensive or simple output formats.

Table 2: Input arguments description.

Parameter	Optional	Default argument	Example of argument	Description
-i	N	No default	/.../test-set100/	Input path
-o	N	No default	/.../output-100-1/	Output path
-p	Y	1	16	Parallel cores to be used
-m	Y	0	1	Output selection: 0: extensive process and output 1: extracted fragments without removing redundancy 2: non-redundant fragment sets
-c	Y	0	1	Output format: 0: exhaustive format 1: all linkers in one file, all bricks in one file, all logs in one folder 2: remove log folder

Example:

```
$ python /.../eMolFrag_201x_xx_xx_xx/eMolFrag.py -i
  /.../TestEMolFrag/test-set100/ -o /.../TestEMolFrag/
  outputp-testset100-1/ -p 16 -m 0 -c 0
$ # Check output.
```

4 Output

Fragments extracted with *e*MolFrag are categorized as Bricks and Linkers. An example is shown below in Figure 1.

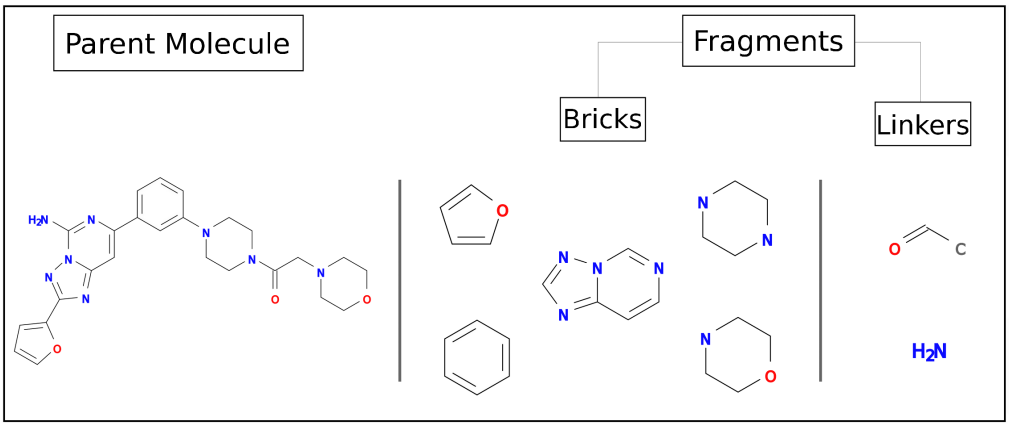

Figure 1: Example of a parent molecule and the extracted fragments.

As in Figure 1, the first column is the parent molecule, the second and third columns are all the fragments generated from this parent molecule.

Now take a look at the details of fragments.
A brick fragment is stored in standard Structure Data format [1] followed by auxiliray information, as shown in Figure 2. "> <ATOMTYPES>" section provides the sybyl atom types [1] following the original sdf atom order. "> <BRANCH @atom-number eligible-atmtype-to-connect>" section shows the connections that the fragment was extracted from. First column is the atom number and the following columns include all the atoms that were observed to be connected to the atom in the fragment. For example, in the brick fragment in Figure 2, the 6^{th} atom which is an N.3 was disconnected from a C.3 atom. "> <fragments similar>" provides the name of same fragments extracted from different molecule or molecular contexts.

```
b-CHEMBL175476.mol2-000.sdf
     RDKit          3D

  6  6  0  0  0  0  0  0  0  0999 V2000
    1.2268  -10.0020   -0.0554 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.4759  -10.3733    1.5879 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.8909   -8.5086   -0.0838 C   0  0  0  0  0  0  0  0  0  0  0  0
   -0.8291   -8.8837    1.5761 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0595  -10.7412    0.3138 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.3682   -8.1042    1.2294 N   0  0  0  0  0  0  0  0  0  0  0  0
  1  3  1  0
  1  5  1  0
  2  4  1  0
  2  5  1  0
  3  6  1  0
  4  6  1  0
M  END

> <ATOMTYPES>
C.3
C.3
C.3
C.3
O.3
N.3

> <BRANCH @atom-number eligible-atmtype-to-connect>
6 C.3


> <fragments similar>
synth-mol-output/b-CHEMBL175476.mol2-000.sdf
$$$$
```

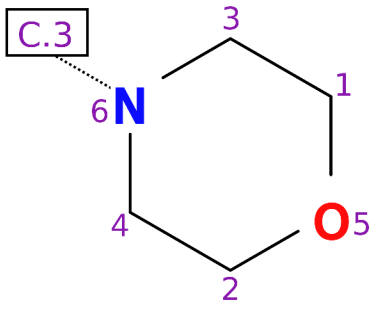

Figure 2: Example of a brick fragment in sdf format.

In linker fragments, on the other hand, only one auxiliary information section "> <MAX-NUMBER-Of-CONTACT ATOMTYPES>" is added. The first column in this section shows the maximum number of observed connections at every atom following the original order of atoms in the linker's sdf file. The atom type is mentioned in the second column. For example, the second line "1 C.3" means that the second atom is a C.3 and it can connect 1 other atom at most.

```
l-CHEMBL175476.mol2-001.sdf
     RDKit          3D

  3  2  0  0  0  0  0  0  0  0999 V2000
    2.4295   -6.4901    1.0459 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.0858   -6.6629    1.2625 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.3774   -5.8952    1.1464 C   0  0  0  0  0  0  0  0  0  0  0  0
  1  3  2  0
  2  3  1  0
M  END

> <MAX-NUMBER-Of-CONTACTS ATOMTYPES>
0 O.2
1 C.3
1 C.2

$$$$
```

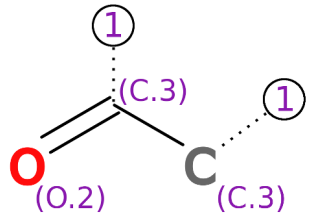

Figure 3: Example of a linker fragment in sdf format.

*e*MolFrag provides options to control the format of output. Here is a list of output folders/files and their corresponding descriptions for choosing different options.

A. Output format 0: Traditional format

Table 3: Output description.

Directory	File Name	Description
../output-chop-comb	*.sdf	All brick and linker fragments before removing redundancy.
../output-brick	b-*.sdf	All unique brick fragments.
../output-linker	l-*.sdf	All unique linker fragments.
../output-log	InputList	File contains all the input *.mol2 file names
	ListAll	File contains all the fragments before reconnecting small linkers, and total/carbon/nitrogen/oxygen atom numbers in each fragment
	BrickListAll.txt	File contains all the brick fragments and total/carbon/nitrogen/oxygen atoms in each fragment.
	LinkerListAll.txt	File contains all the linker fragments after reconnect and total/carbon/nitrogen/oxygen atoms in each fragment.
	BrickGroupList.txt	Brick fragments are grouped by the total number of C, N and O heavy atoms. The number shows the frequency of fragments with similar combination of atoms.
	LinkerGroupList.txt	Linker fragments are grouped by the total number of C, N and O heavy atoms. The number shows the frequency of fragments with similar combination of atoms.
	brick-log.txt	Log file for removing redundancy of brick fragments and similarity information.
	linker-log.txt	Log file for removing redundancy of linker fragments and similarity information.
	Process.log	Log file for the whole process.

B. Output format 1: Simple output format with log and statistics

Table 4: Output description.

Directory	File Name	Description
../	BrickFull.sdf	All brick fragments before removing redundancy in one file.
../	BrickUnique.sdf	All unique brick fragments in one file.
../	LinkerFull.sdf	All linker fragments before removing redundancy in one file.
../	LinkerUnique.sdf	All unique linker fragments in one file.
../output-log	InputList	File contains all the input *.mol2 file names
	ListAll	File contains all the fragments before reconnecting small linkers, and total/carbon/nitrogen/oxygen atom numbers in each fragment
	BrickListAll.txt	File contains all the brick fragments and total/carbon/nitrogen/oxygen atoms in each fragment.
	LinkerListAll.txt	File contains all the linker fragments after reconnect and total/carbon/nitrogen/oxygen atoms in each fragment.
	BrickGroupList.txt	Brick fragments are grouped by the total number of C, N and O heavy atoms. The number shows the frequency of fragments with similar combination of atoms.
	LinkerGroupList.txt	Linker fragments are grouped by the total number of C, N and O heavy atoms. The number shows the frequency of fragments with similar combination of atoms.
	brick-log.txt	Log file for removing redundancy of brick fragments and similarity information.
	linker-log.txt	Log file for removing redundancy of linker fragments and similarity information.
	Process.log	Log file for the whole process.

C. Output format 2: Simple output format.

Table 5: Output description.

Directory	File Name	Description
../	BrickFull.sdf	All brick fragments before removing redundancy in one file.
../	BrickUnique.sdf	All unique brick fragments in one file.
../	LinkerFull.sdf	All linker fragments before removing redundancy in one file.
../	LinkerUnique.sdf	All unique linker fragments in one file.

5 Online Resources

The most recent version of *e*MolFrag is available at:
`https://github.com/liutairan/eMolFrag`
`http://brylinski.cct.lsu.edu/content/emolfrag-standalone-package`

If you meet any problems or find any bugs with *e*MolFrag, please raise an issue on GitHub issues page:
`https://github.com/liutairan/eMolFrag/issues`

A web server which can provide fragmentation service:
`http://brylinski.cct.lsu.edu/content/emolfrag-webserver`

Molecular synthesis tool, *e*Synth, is available at :
`http://brylinski.cct.lsu.edu/content/molecular-synthesis`

References

[1] Matthew Clark, Richard D Cramer, and Nicole Van Opdenbosch. Validation of the general purpose tripos 5.2 force field. *Journal of Computational Chemistry*, 10(8):982–1012, 1989.

[2] Jörg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem*, 3(10):1503–1507, 2008.

[3] Takeshi Kawabata. Build-up algorithm for atomic correspondence between chemical structures. *Journal of chemical information and modeling*, 51(8):1775–1787, 2011.

[4] Greg Landrum. Rdkit: Open-source cheminformatics. *Online). http://www. rdkit. org. Accessed*, 3(04):2012, 2006.

[5] Tairan Liu, Misagh Naderi, Chris Alvin, Supratik Mukhopadhyay, and Michal Brylinski. Break down in order to build up: Decomposing small molecules for fragment-based drug design with emolfrag. Submitted.

[6] Misagh Naderi, Chris Alvin, Yun Ding, Supratik Mukhopadhyay, and Michal Brylinski. A graph-based approach to construct target-focused libraries for virtual screening. *J Cheminform*, 8:14, 2016 2016.

[7] Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 3(1):33, 2011.