Fragment Explorer: A Comprehensive Tool for Fragment-Based Drug Design

-PyQT Application -

Presented for Major-Project Second Review by

Anaghaa K – 124013063

(IV Year - B.Tech Bioinformatics, SCBT)

Under the guidance of

Mr. Udayakumar .M (Asst. Professor - III, SCBT)

Table of Contents

Background and Challenges

Background

Drug discovery is a complex and resource-intensive process, often involving the synthesis and screening of large compound libraries.

Challenges

Traditional methods are time-consuming and expensive, necessitating the exploration of more efficient approaches.

FBDD and Problem Statement

FBDD

FBDD Offers a promising alternative by breaking down molecules into smaller fragments & systematically building them up.

Problem

Drug discovery is a complex and resource-intensive process, often involving the synthesis and screening of large compound libraries.

Aim and Objective

Primary objective

Aim to identify efficient fragments that have the potential to become drug candidates

Secondary objective

1. Provide GUI for open scores library
2. Fragmentation, Fragment Filtering, 2D Visualization, Scaffold hopping, MCS.

Application

1. Fragment Library
2. Scaffold Library

Workflow

01 Fragmentation

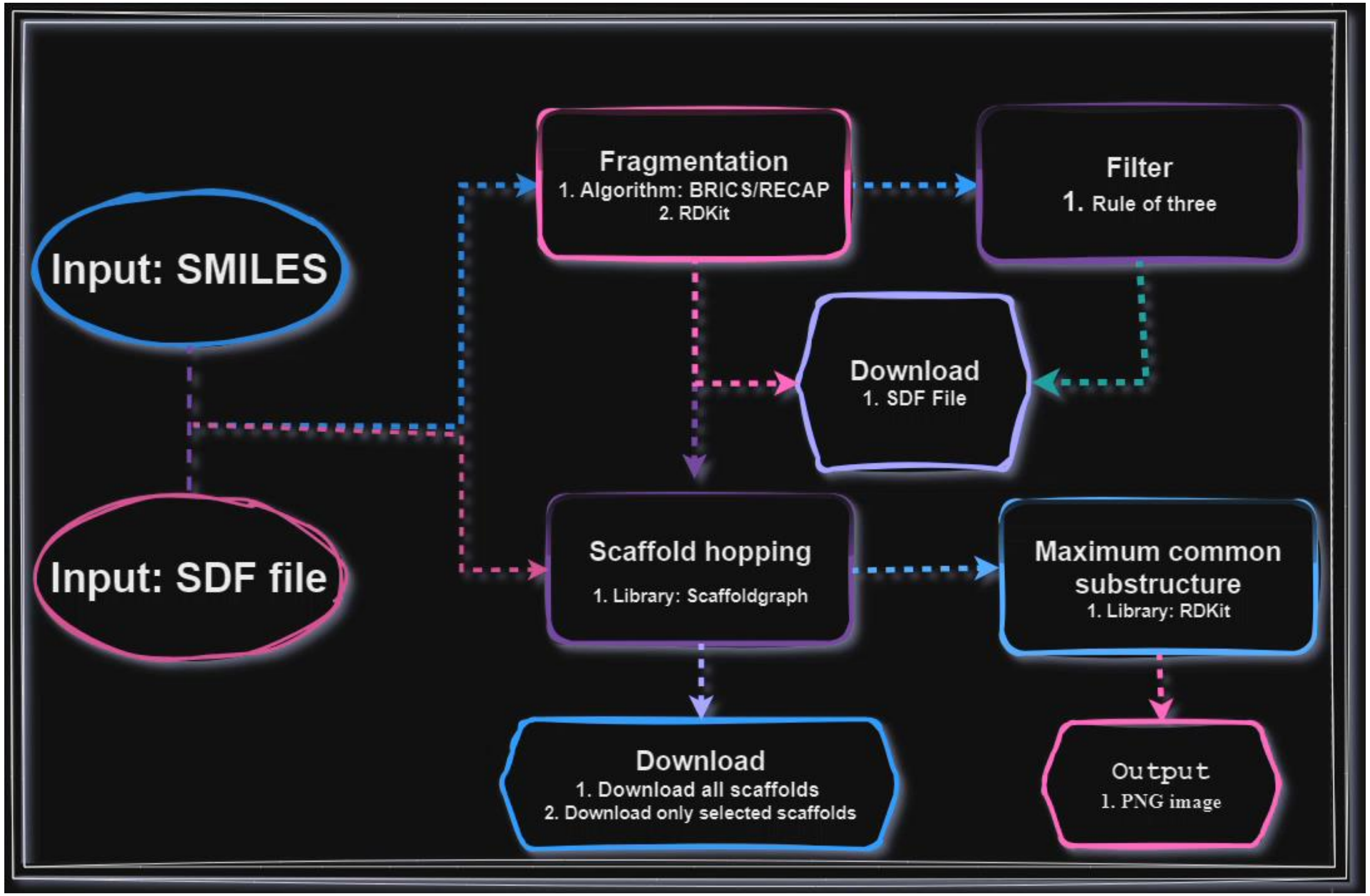
02 Fragment Filtering

03 Scaffold hopping

04 Maximum common substructure

05 2D Visualization

Fragment explorer (.exe)



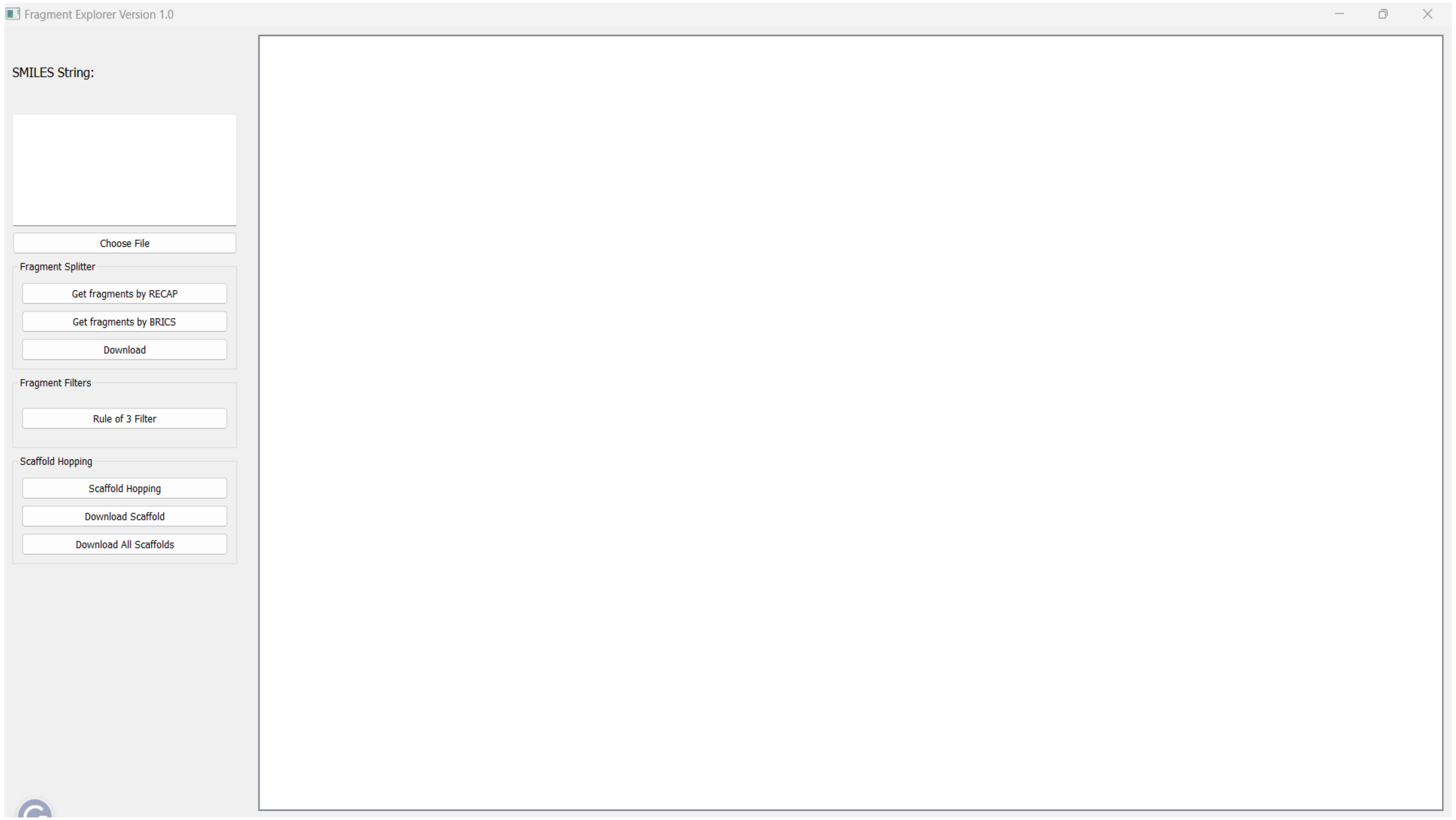
File Description

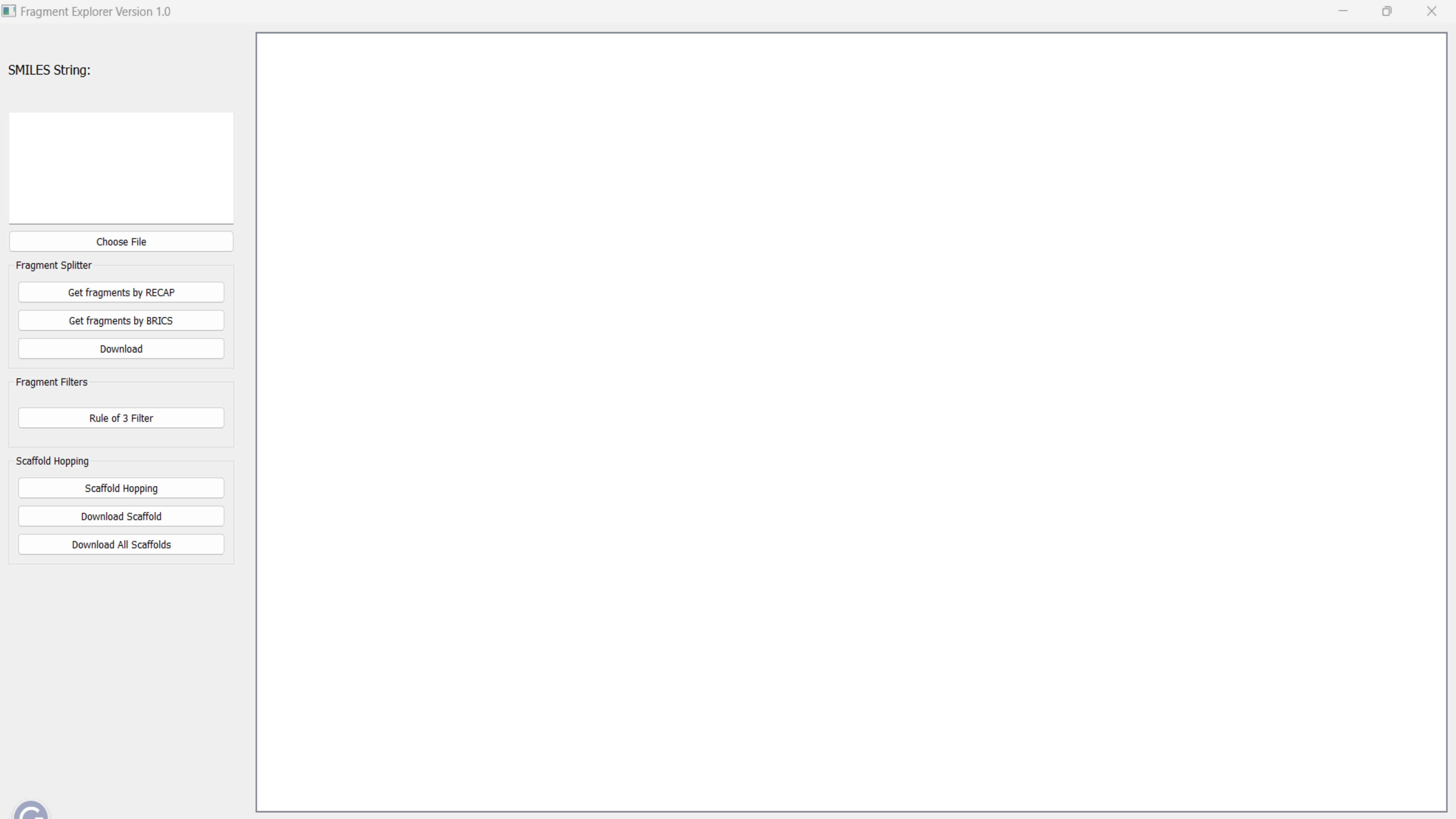
- Type: Application(.exe)
- Size: 143 MB

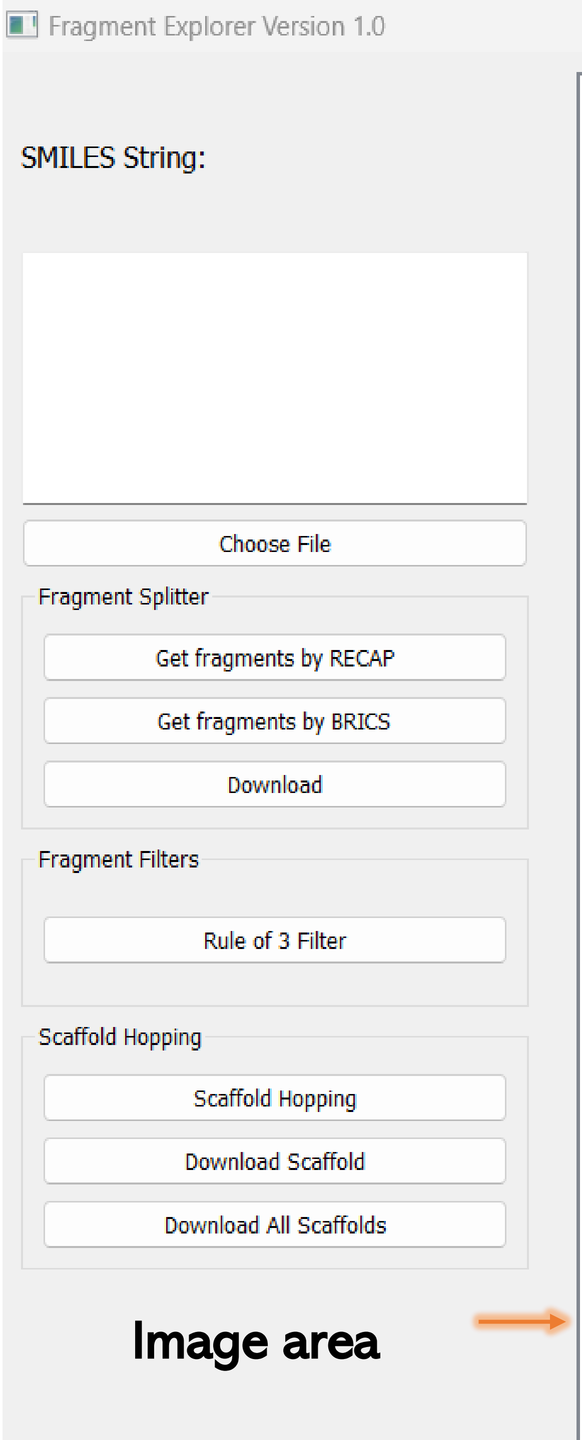
Note:

Working is not similar workflow to Fragment Explorer to compare.

Result and Implementation
Fragment explorer




Fragment Explorer Version 1.0
SMILES String:
Choose File
Fragment Splitter
Get fragments by RECAP
Get fragments by BRICS
Download
Fragment Filters
Rule of 3 Filter
Scaffold Hopping
Scaffold Hopping
Download Scaffold
Download All Scaffolds



Tool frame

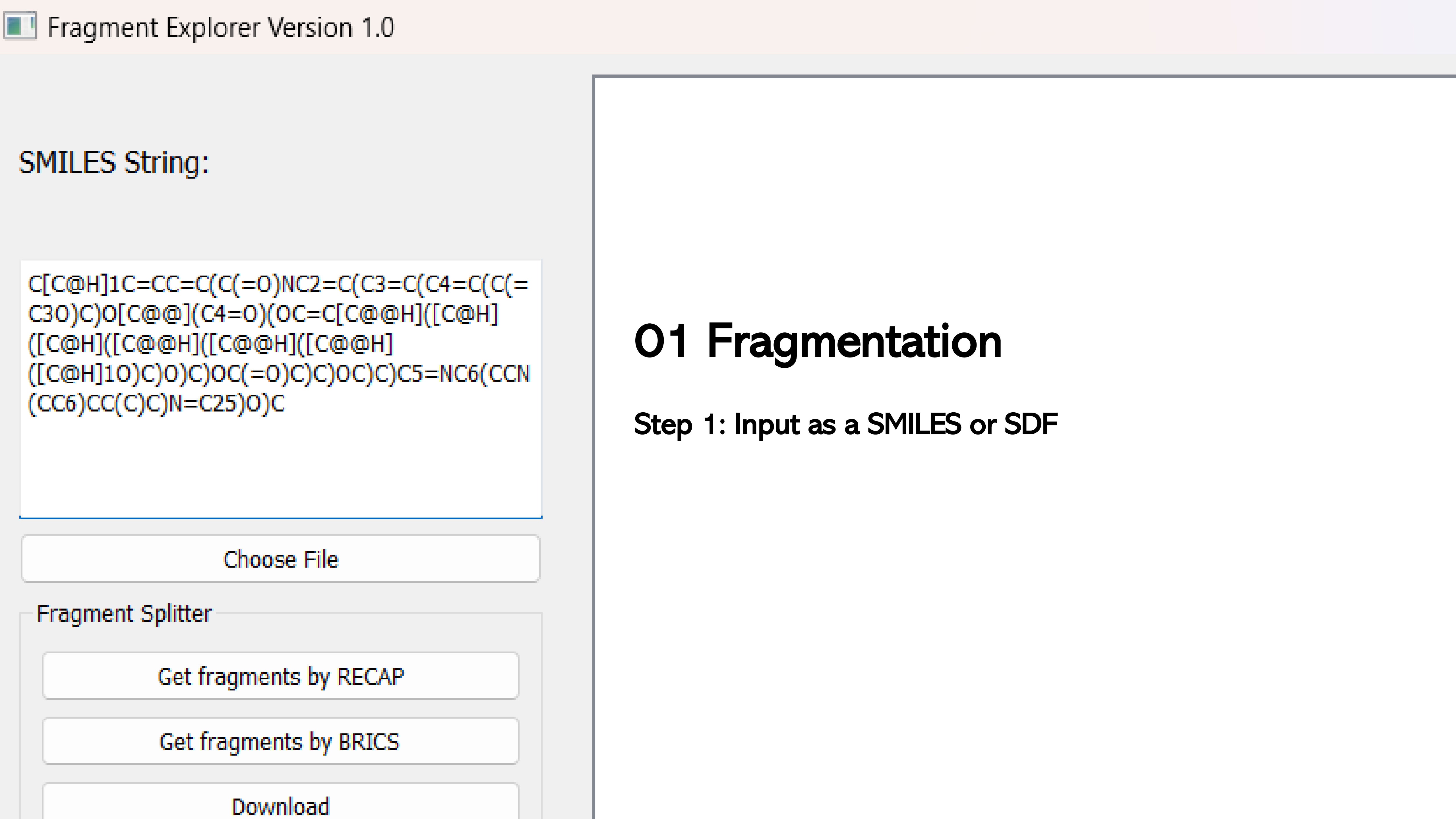
On the left side of the screen

```
def updateMolecule(self):
        # Slot to update the molecule visualization when the SMILES string changes
        smiles = self.smiles_textbox.toPlainText().strip()
        if smiles:
            molecule_image = self.loadMolecule(smiles)
            if molecule_image:
                self.image_display.setPixmap(molecule_image.scaled(self.image_display.size(), Qt.KeepAspectRatio))
```

Image area

Information frame

```
def createRightFrame(self):
        self.right_frame = QWidget()
        self.right_layout = QVBoxLayout(self.right_frame)
        self.table_scroll = QScrollArea()
        self.table_scroll.setWidgetResizable(True)
        self.table_widget = QTableWidget()
        self.table_widget.setColumnCount(9)
```



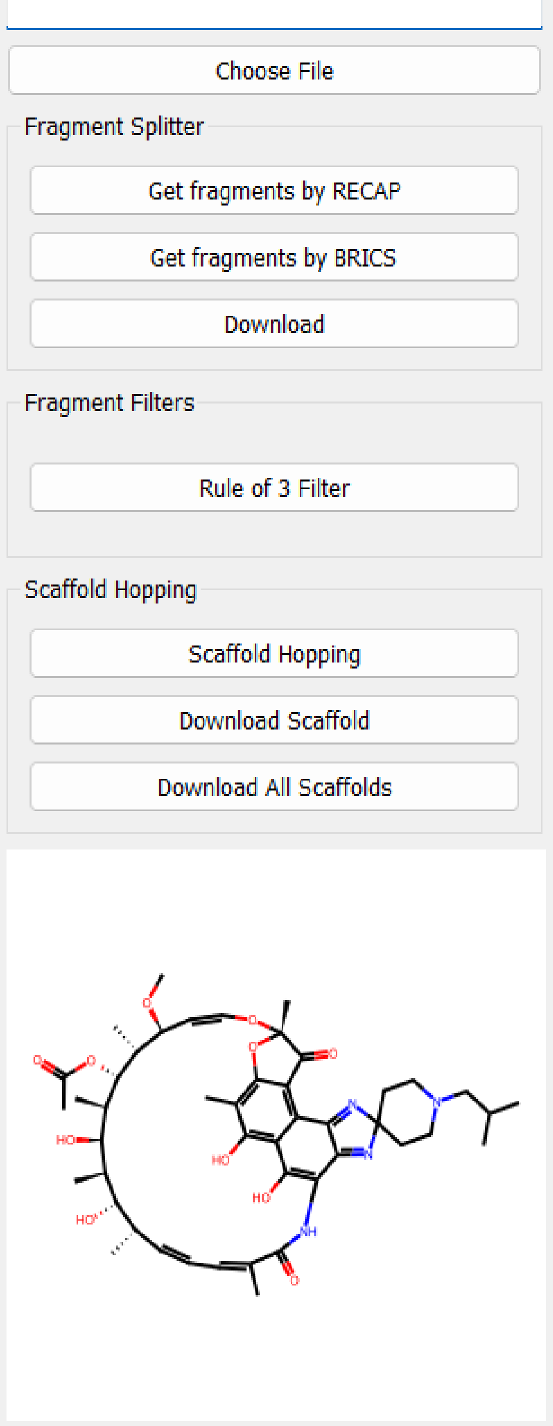
Fragment Explorer Version 1.0

SMILES String:

```
C[C@H]1C=CC=C(C(=O)NC2=C(C3=C(C4=C(C(=
C3O)C)O[C@@](C4=O)(OC=C[C@@H]([C@H]
([C@H]([C@@H]([C@@H]([C@@H]
([C@H]1O)C)O)C)OC(=O)C)C)OC)C)C5=NC6(CCN
(CC6)CC(C)C)N=C25)O)C
```

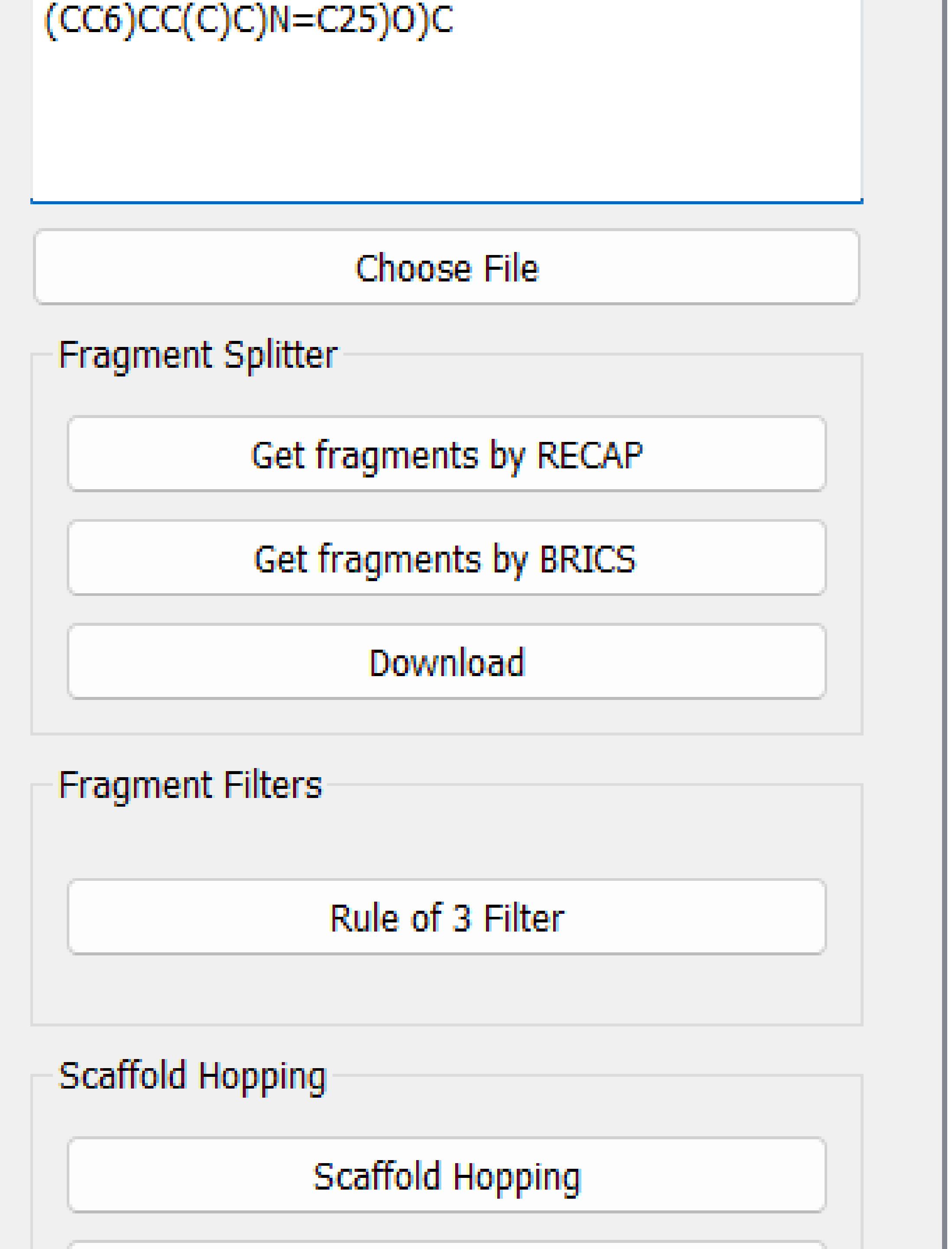
01 Fragmentation

Step 1: Input as a SMILES or SDF



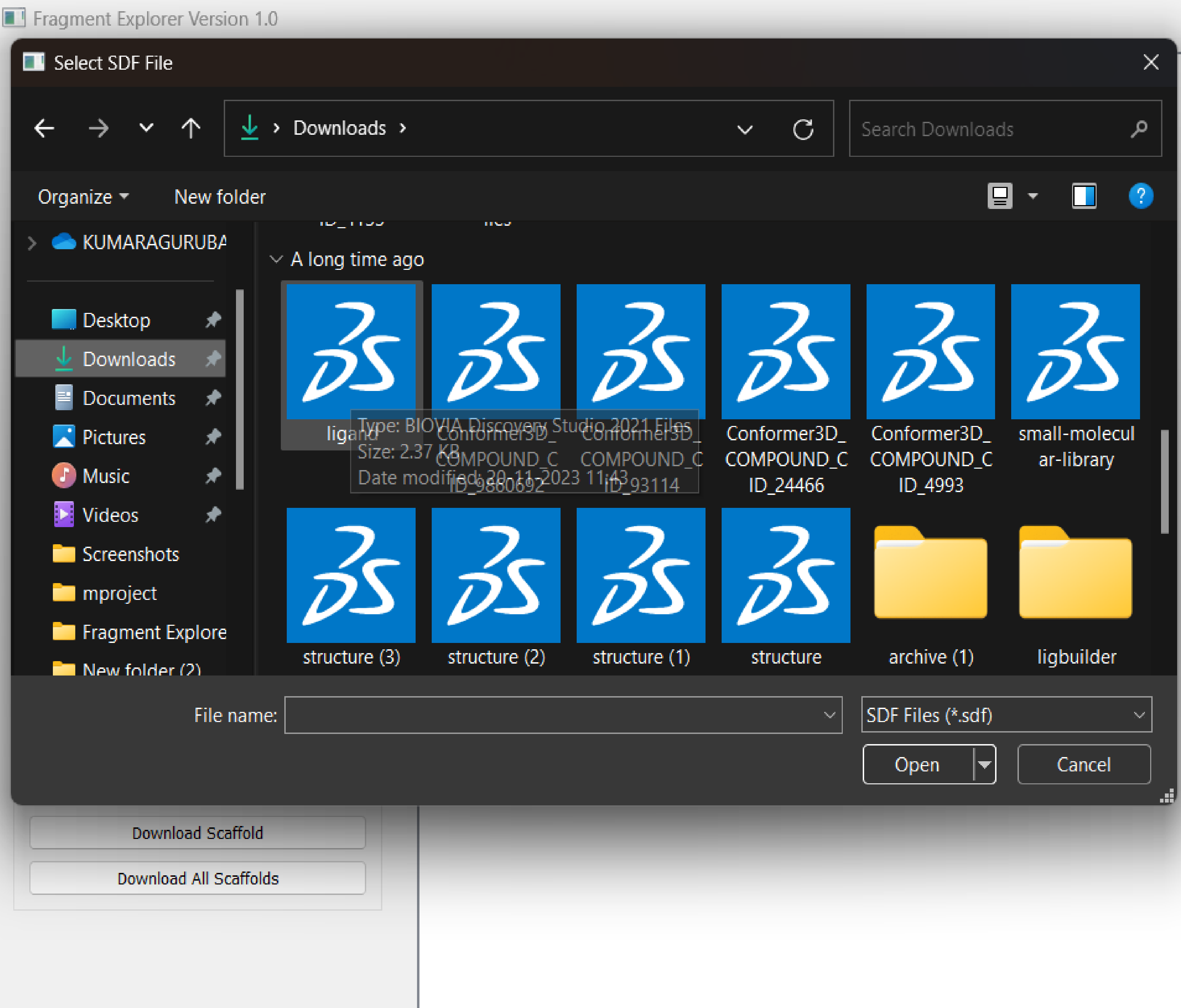
01 Fragmentation

Step 1: Input as a SMILES / SDF respective image will display


(CC6)CC(C)C)N=C25)O)C

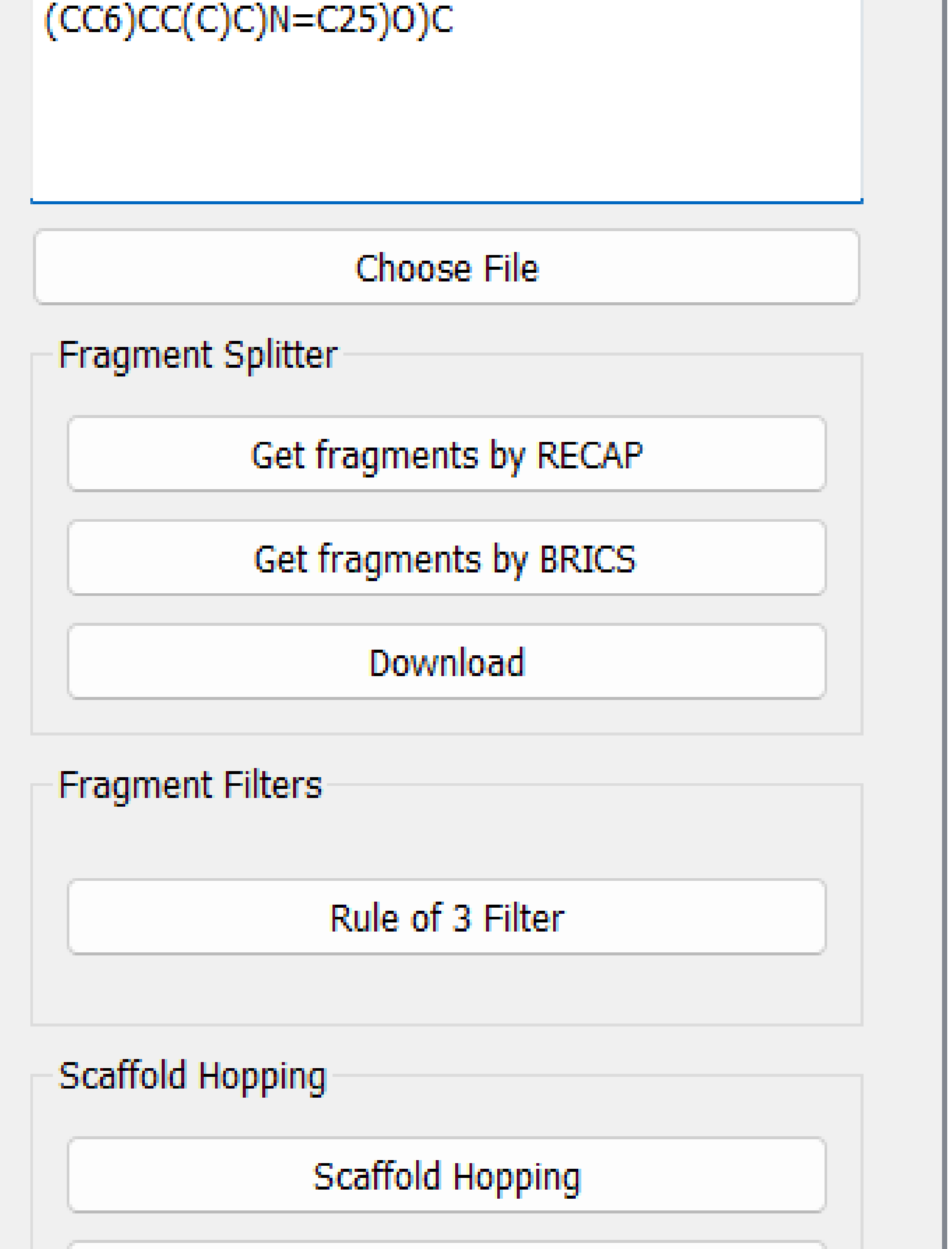
01 Fragmentation

Step 1: Select choose file
Step 2: Input as an SDF file



01 Fragmentation

Step 1: Select 'choose file' option
Step 2: Input as an SDF file

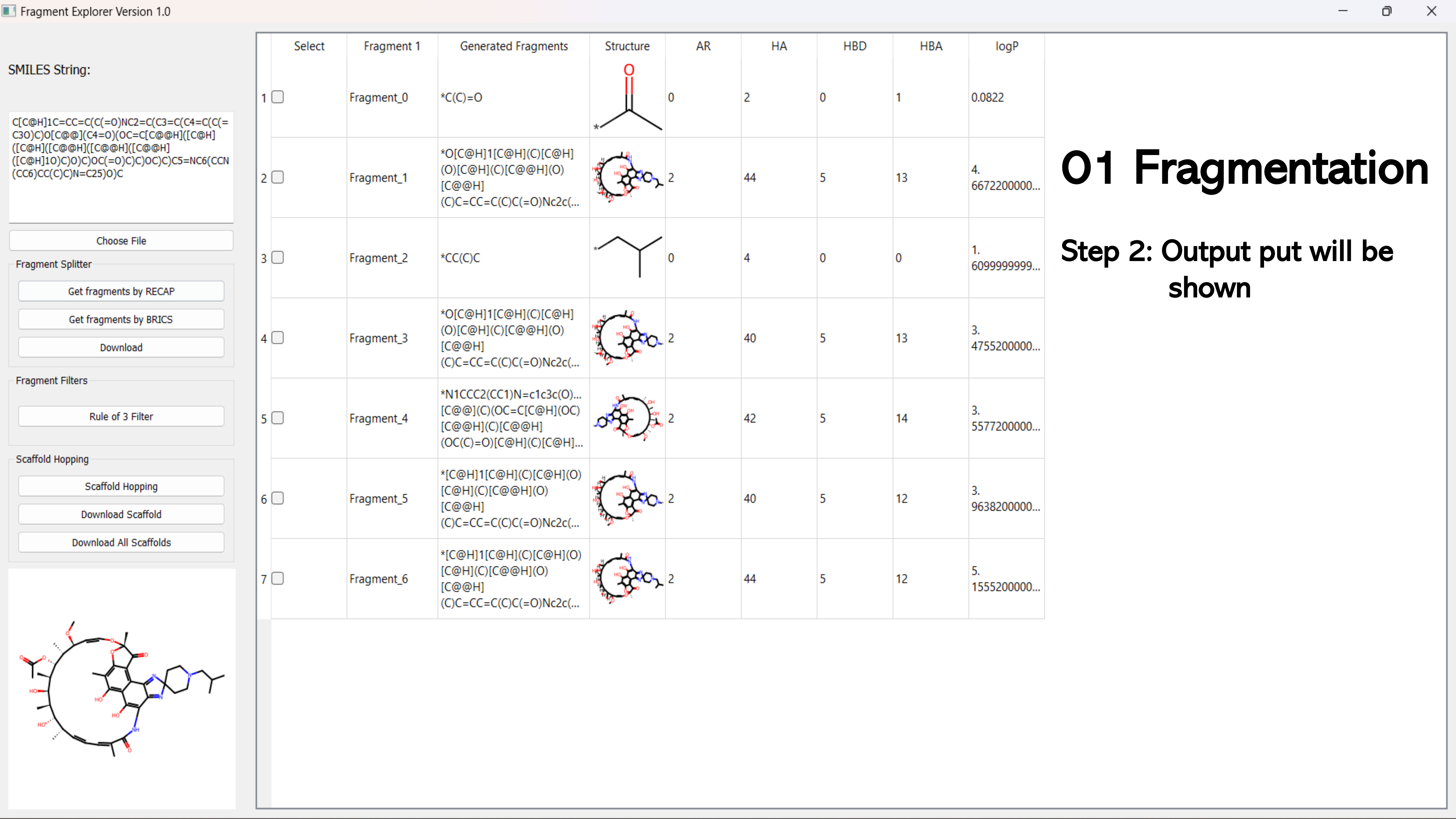

(CC6)CC(C)C)N=C25)O)C

01 Fragmentation

Step 1: Select any algorithm

01 Fragmentation

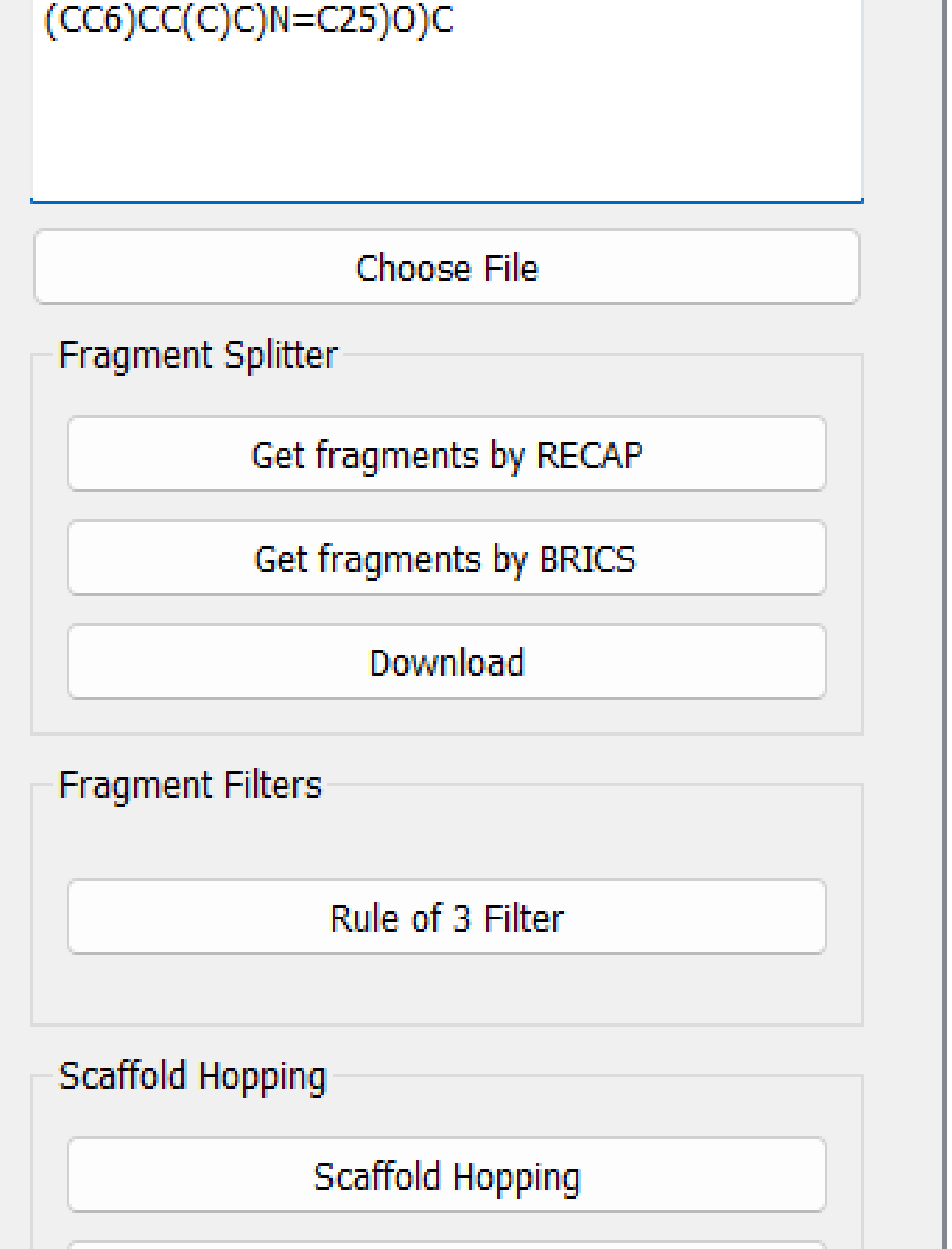
Step 2: Output put will be shown



SMILES String:

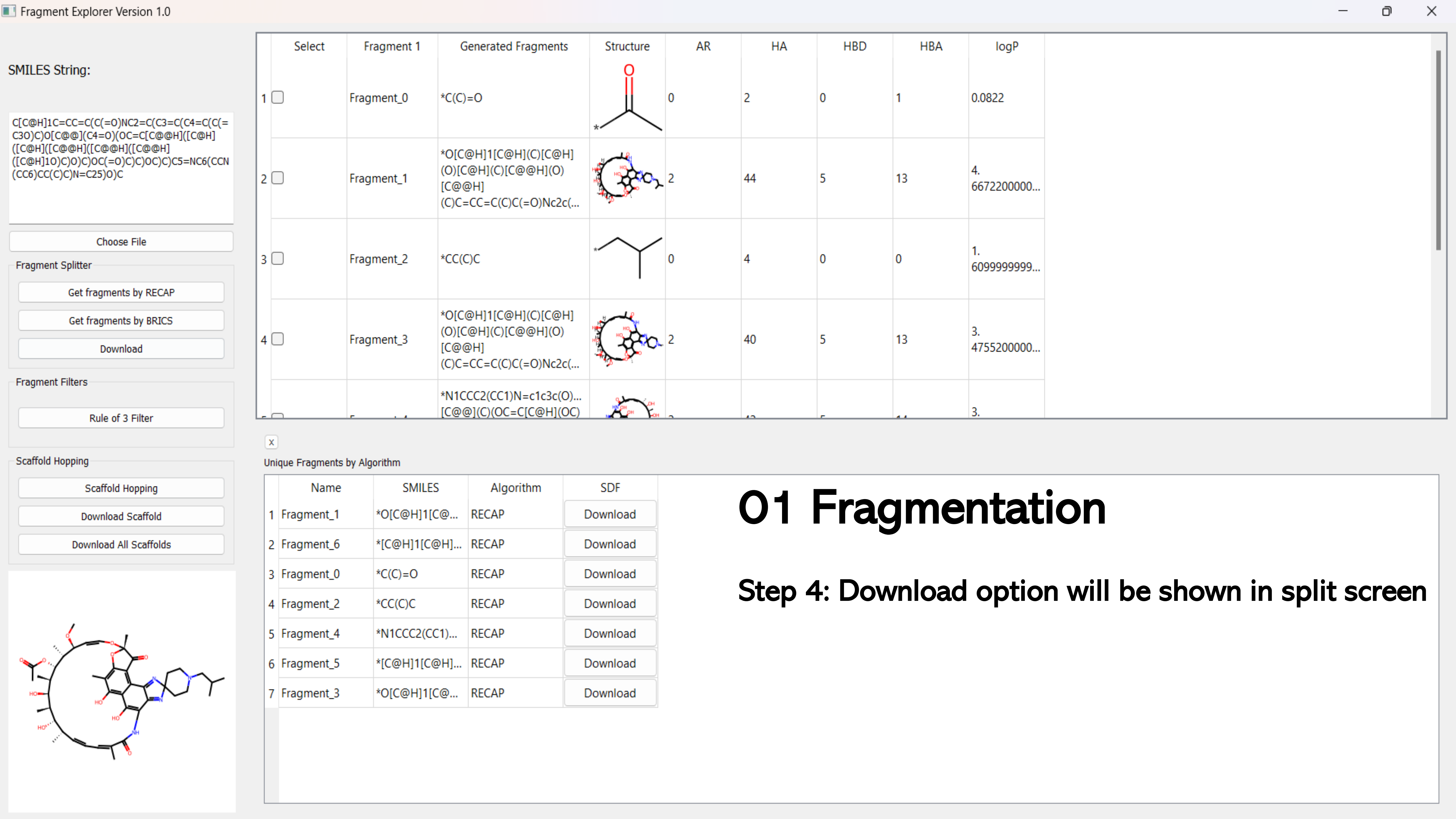
```
C[C@H]1C=CC=C(C(=O)NC2=C(C3=C(C4=C(C(=
C3O)C)O[C@@](C4=O)(OC=C[C@@H]([C@H]
([C@H]([C@@H]([C@@H]([C@@H]
([C@H]1O)C)O)C)OC(=O)C)C)OC)C)C5=NC6(CCN
(CC6)CC(C)C)N=C25)O)C
```

	Select	Fragment 1	Generated Fragments	Structure	AR	HA	HBD	HBA	logP
1	☐	Fragment_0	*C(C)=O		0	2	0	1	0.0822
2	☐	Fragment_1	*O[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	44	5	13	4.6672200000...
3	☐	Fragment_2	*CC(C)C		0	4	0	0	1.6099999999...
4	☐	Fragment_3	*O[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	40	5	13	3.4755200000...
5	☐	Fragment_4	*N1CCC2(CC1)N=c1c3c(O)...[C@@](C)(OC=C[C@H](OC)[C@@H](C)[C@@H](OC(C)=O)[C@H](C)[C@H]...		2	42	5	14	3.5577200000...
6	☐	Fragment_5	*[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	40	5	12	3.9638200000...
7	☐	Fragment_6	*[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	44	5	12	5.1555200000...


(CC6)CC(C)C)N=C25)O)C

01 Fragmentation

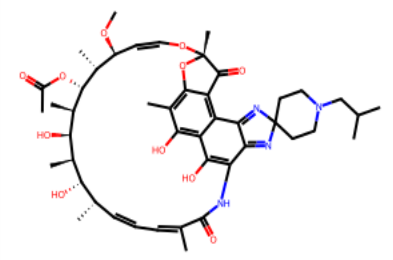
Step 3: Select Download option

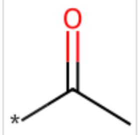
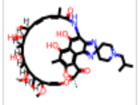
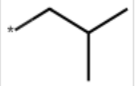
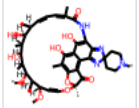


Fragment Explorer Version 1.0

SMILES String:

```
C[C@H]1C=CC=C(C(=O)NC2=C(C3=C(C4=C(C(=
C3O)C)O[C@@](C4=O)(OC=C[C@@H]([C@H]
([C@H]([C@@H]([C@@H]([C@@H]
([C@H]1O)C)O)C)OC(=O)C)C)OC)C)C5=NC6(CCN
(CC6)CC(C)C)N=C25)O)C
```



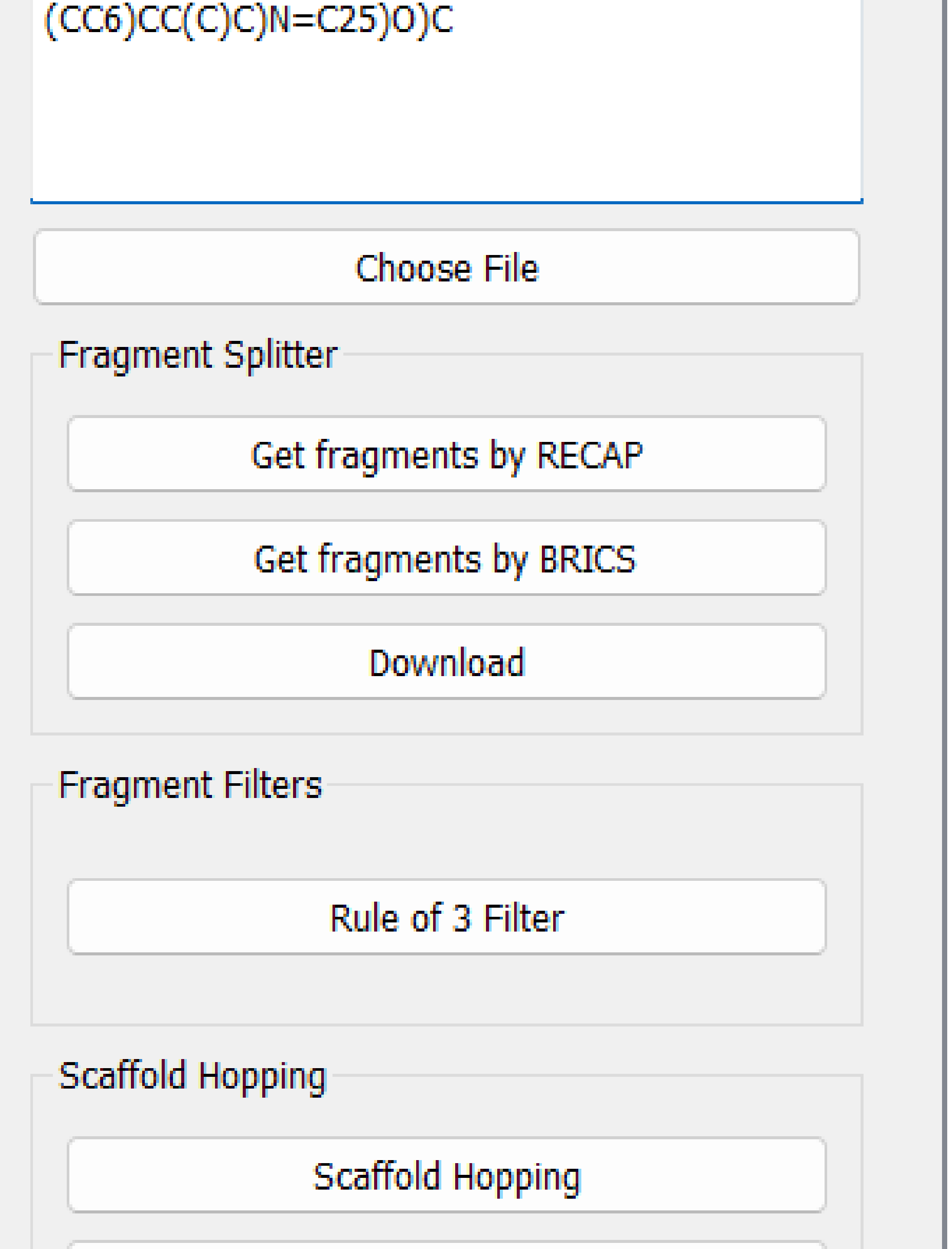
	Select	Fragment 1	Generated Fragments	Structure	AR	HA	HBD	HBA	logP
1	☐	Fragment_0	*C(C)=O		0	2	0	1	0.0822
2	☐	Fragment_1	*O[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	44	5	13	4.6672200000...
3	☐	Fragment_2	*CC(C)C		0	4	0	0	1.6099999999...
4	☐	Fragment_3	*O[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	40	5	13	3.4755200000...
			*N1CCC2(CC1)N=c1c3c(O)...[C@@](C)(OC=C[C@H](OC)						3.

Unique Fragments by Algorithm

	Name	SMILES	Algorithm	SDF
1	Fragment_1	*O[C@H]1[C@...	RECAP	Download
2	Fragment_6	*[C@H]1[C@H]...	RECAP	Download
3	Fragment_0	*C(C)=O	RECAP	Download
4	Fragment_2	*CC(C)C	RECAP	Download
5	Fragment_4	*N1CCC2(CC1)...	RECAP	Download
6	Fragment_5	*[C@H]1[C@H]...	RECAP	Download
7	Fragment_3	*O[C@H]1[C@...	RECAP	Download

01 Fragmentation

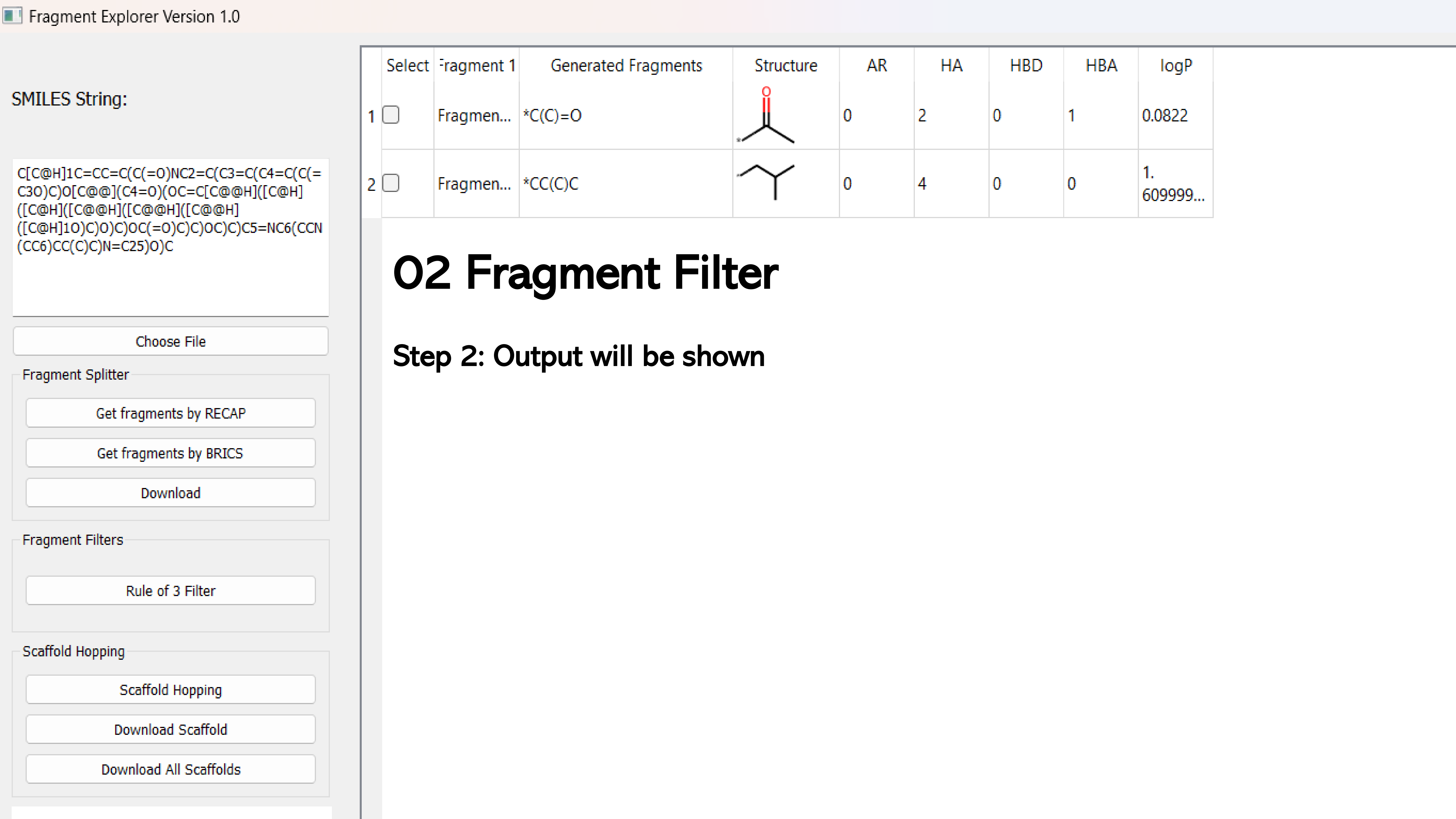
Step 4: Download option will be shown in split screen



(CC6)CC(C)C)N=C25)O)C

02 Fragment Filter

Step 4: Select rule of three



Fragment Explorer Version 1.0

SMILES String:

```
C[C@H]1C=CC=C(C(=O)NC2=C(C3=C(C4=C(C(=
C3O)C)O[C@@](C4=O)(OC=C[C@@H]([C@H]
([C@H]([C@@H]([C@@H]([C@@H]
([C@H]1O)C)O)C)OC(=O)C)C)OC)C)C5=NC6(CCN
(CC6)CC(C)C)N=C25)O)C
```

	Select	Fragment 1	Generated Fragments	Structure	AR	HA	HBD	HBA	logP
1	☐	Fragmen...	*C(C)=O		0	2	0	1	0.0822
2	☐	Fragmen...	*CC(C)C		0	4	0	0	1. 609999...

02 Fragment Filter

Step 2: Output will be shown

Fragment Explorer Version 1.0

SMILES String:

Choose File

Fragment Splitter

Get fragments by RECAP

Get fragments by BRICS

Download

Fragment Filters

Rule of 3 Filter

Scaffold Hopping

Scaffold Hopping

Download Scaffold

Download All Scaffolds

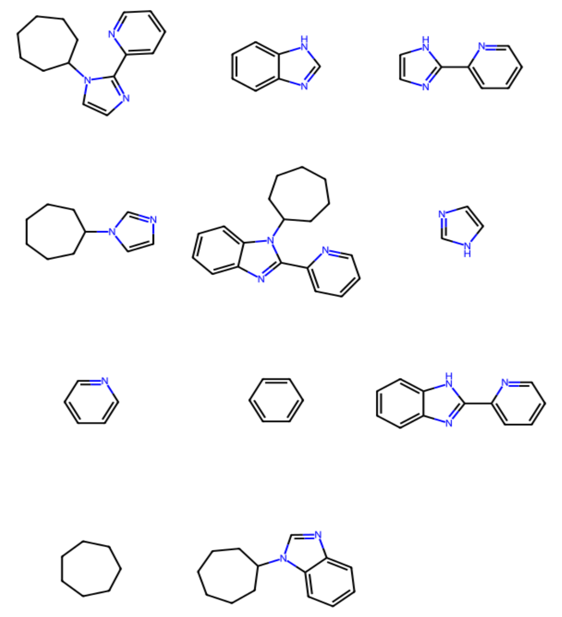
03 Scaffold hopping

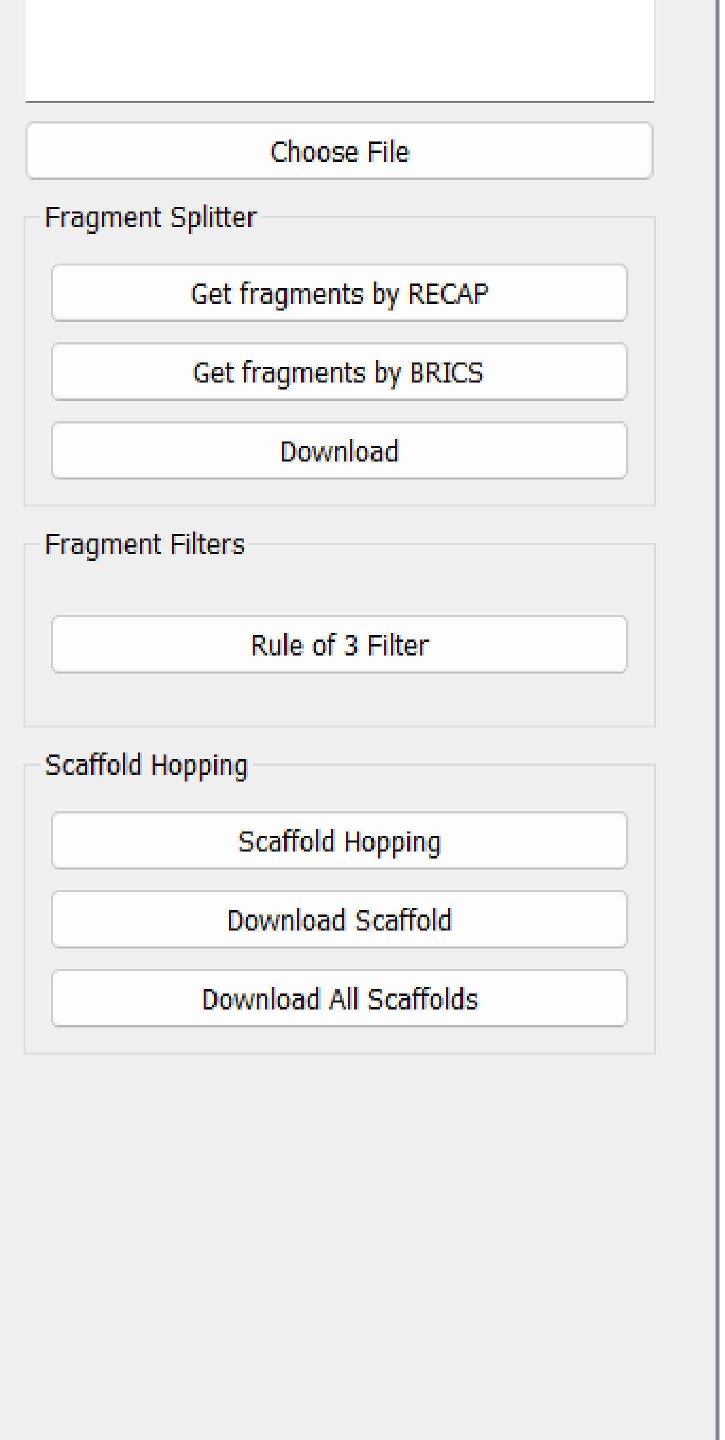
There are two major ways to use this option are as follows:

1. Direct import SDF or SMILES as input
2. Select any fragments generated by the algorithm or after using a filter rule of three.

```
# Import scaffoldgraph
import scaffoldgraph as sg
# Import rdkit
from rdkit.Chem import Draw
from rdkit import Chem
# Create a molecule from a SMILES string
mol = Chem.MolFromSmiles('O=C(O)c1ccc2c(c1)nc
        (-c1ccccn1)n2C1CCCCCC1')
# We can generate all possible murcko fragments using
#scaffoldgraph
# returned fragments are rdkit molecules
frags = sg.get_all_murcko_fragments(mol)
Draw.MolsToGridImage(frags)
```

Example output:





03 Scaffold hopping

1. Direct import SDF or SMILES as input

   Step 1: Select the scaffold hopping option

Fragment Explorer Version 1.0

SMILES String:

Choose File

Fragment Splitter

Get fragments by RECAP

Get fragments by BRICS

Download

Fragment Filters

Rule of 3 Filter

Scaffold Hopping

Scaffold Hopping

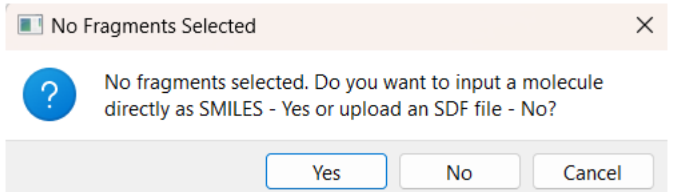
Download Scaffold

Download All Scaffolds

03 Scaffold hopping

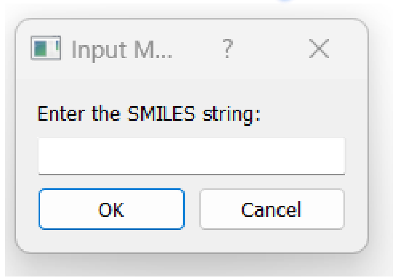
1. Direct import SDF or SMILES as input

Step 2: Select 'yes' means – SMILES input, 'no' means – SDF input



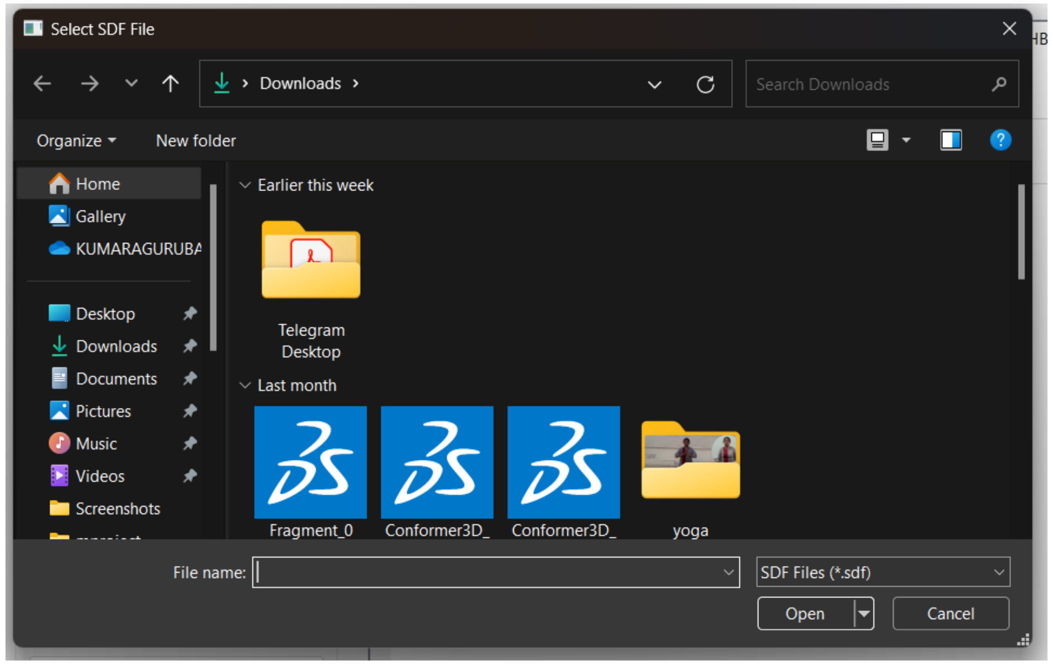
No Fragments Selected

No fragments selected. Do you want to input a molecule directly as SMILES - Yes or upload an SDF file - No?



Input M...

Enter the SMILES string:



Select SDF File

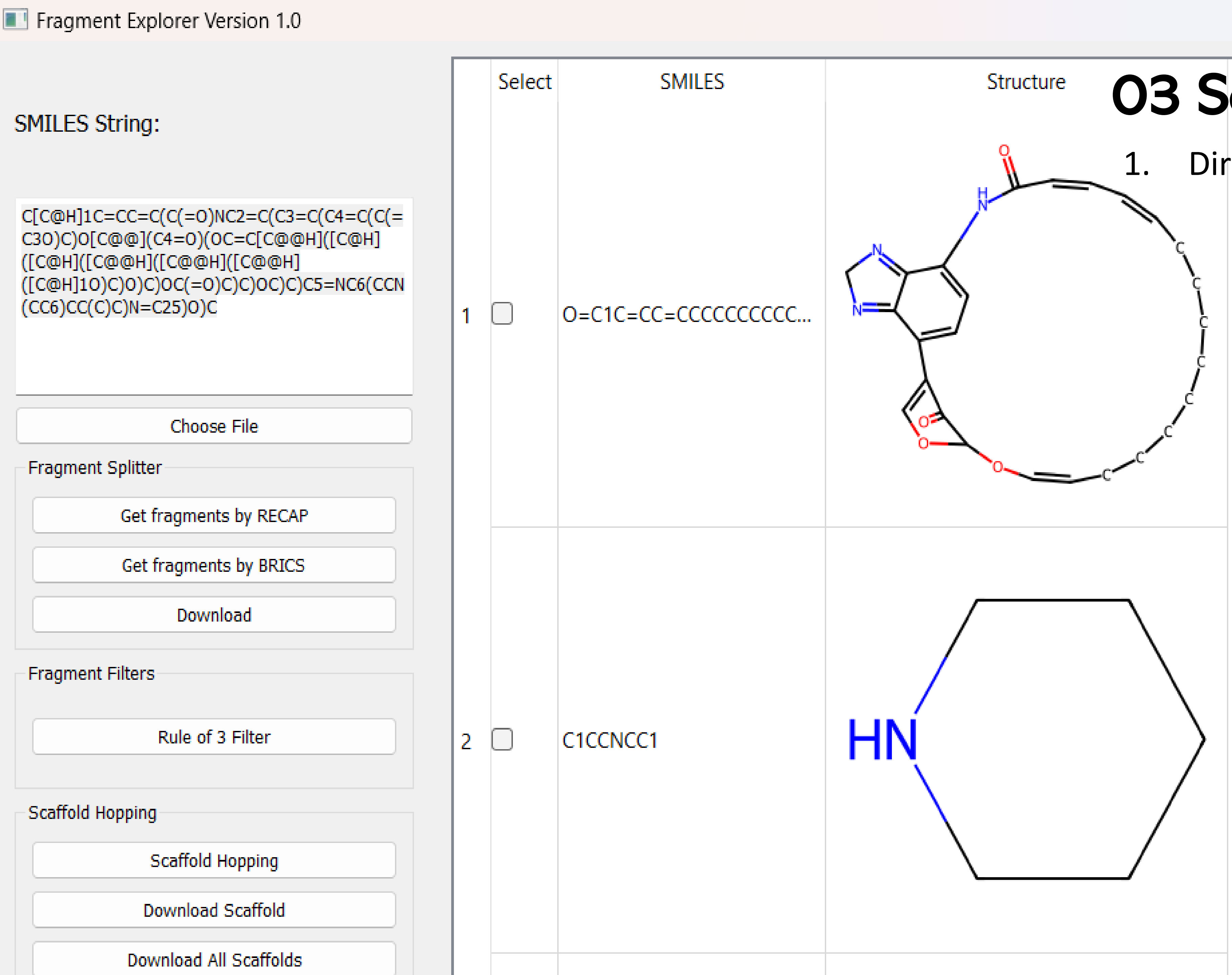
File name:

SDF Files (*.sdf)

03 Scaffold hopping

1. Direct import SDF or SMILES as input

Step 3: Output will be shown

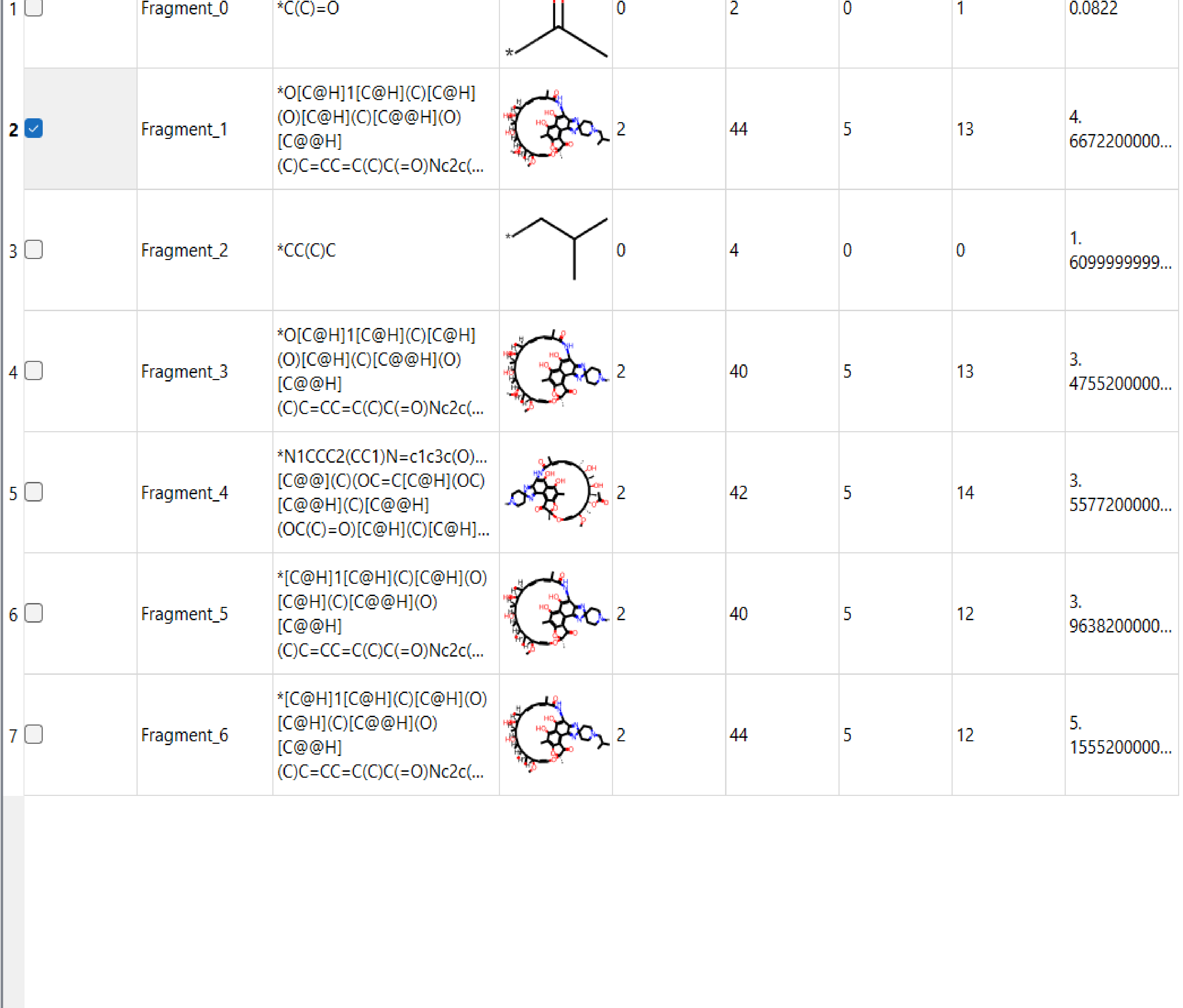


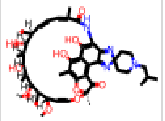
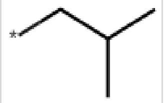
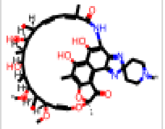
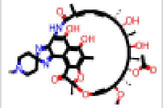
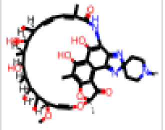
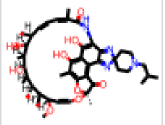
Fragment Explorer Version 1.0

SMILES String:

```
C[C@H]1C=CC=C(C(=O)NC2=C(C3=C(C4=C(C(=
C3O)C)O[C@@](C4=O)(OC=C[C@H]([C@H]
([C@H]([C@@H]([C@@H]([C@@H]
([C@H]1O)C)O)C)OC(=O)C)C)OC)C)C5=NC6(CCN
(CC6)CC(C)C)N=C25)O)C
```

	Select	SMILES	Structure
1	☐	O=C1C=CC=CCCCCCCCCC...	
2	☐	C1CCNCC1	



1	☐	Fragment_0	*C(C)=O		0	2	0	1	0.0822
2	☑	Fragment_1	*O[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	44	5	13	4. 6672200000...
3	☐	Fragment_2	*CC(C)C		0	4	0	0	1. 6099999999...
4	☐	Fragment_3	*O[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	40	5	13	3. 4755200000...
5	☐	Fragment_4	*N1CCC2(CC1)N=c1c3c(O)...[C@@](C)(OC=C[C@H](OC)[C@@H](C)[C@@H](OC(C)=O)[C@H](C)[C@H]...		2	42	5	14	3. 5577200000...
6	☐	Fragment_5	*[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	40	5	12	3. 9638200000...
7	☐	Fragment_6	*[C@H]1[C@H](C)[C@H](O)[C@H](C)[C@@H](O)[C@@H](C)C=CC=C(C)C(=O)Nc2c(...		2	44	5	12	5. 1555200000...

03 Scaffold hopping

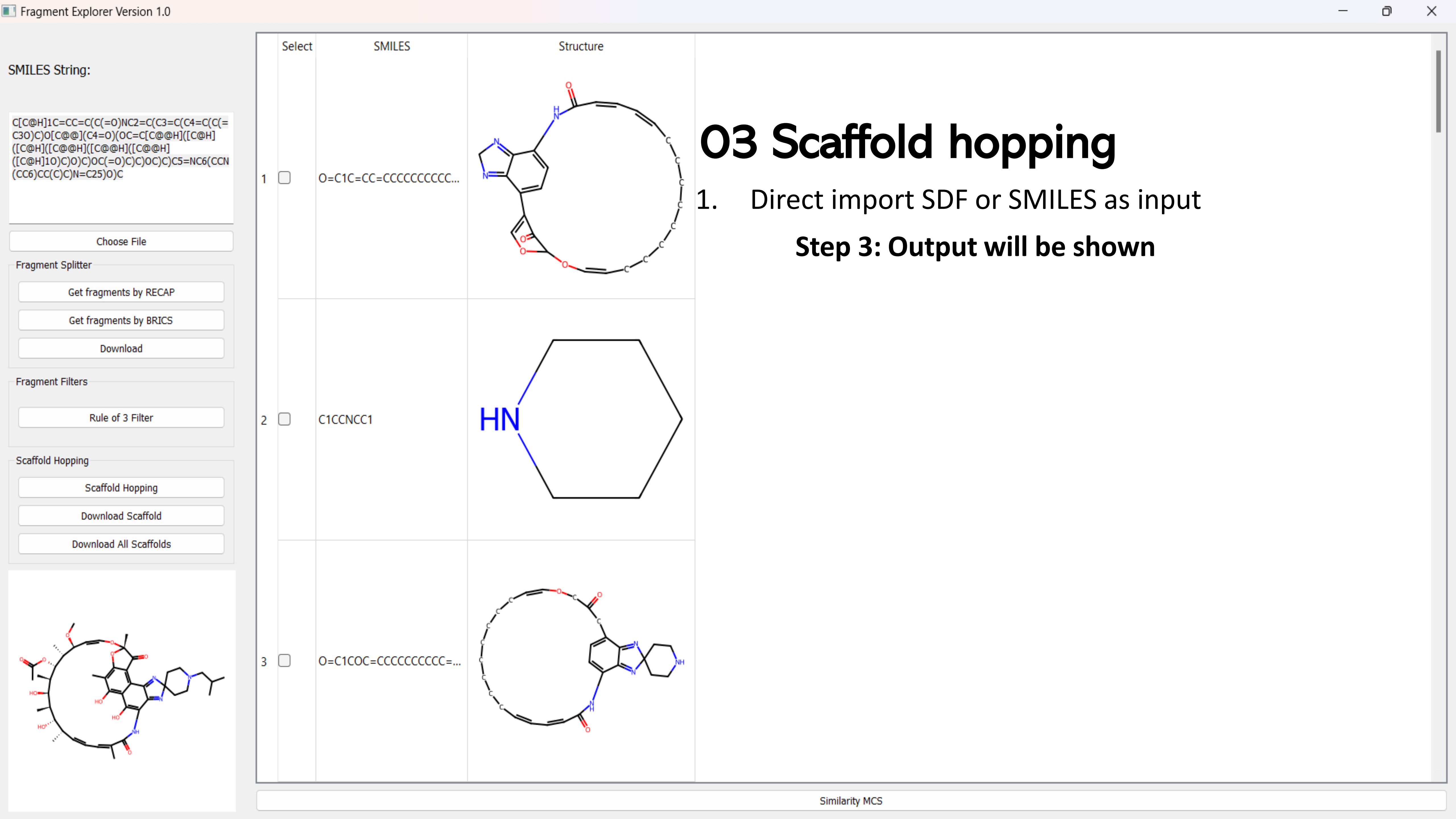
There are two major ways to use this option are as follows:

2. Select any fragments generated by the algorithm or after using a filter rule of three.

03 Scaffold hopping

1. Direct import SDF or SMILES as input

Step 3: Output will be shown



Fragment Explorer Version 1.0

SMILES String:

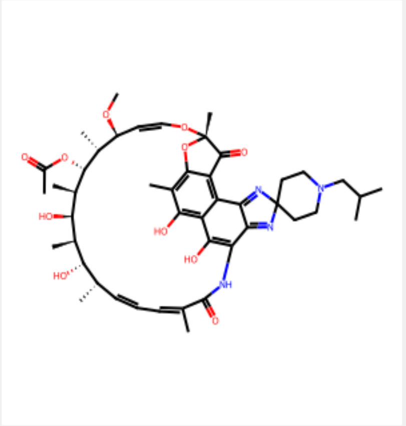
```
C[C@H]1C=CC=C(C(=O)NC2=C(C3=C(C4=C(C(=
C3O)C)O[C@@](C4=O)(OC=C[C@@H]([C@H]
([C@H]([C@@H]([C@@H]([C@@H]
([C@H]1O)C)O)C)OC(=O)C)C)OC)C)C5=NC6(CCN
(CC6)CC(C)C)N=C25)O)C
```

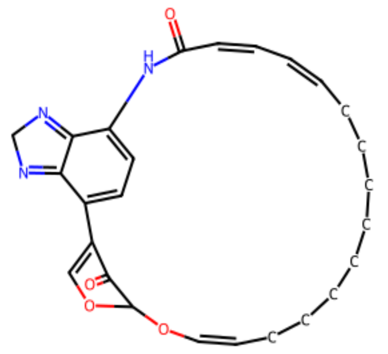
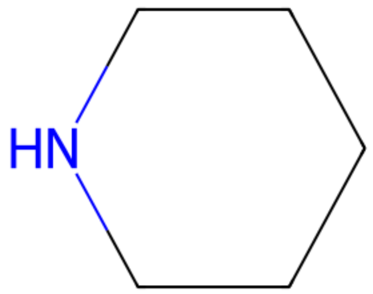
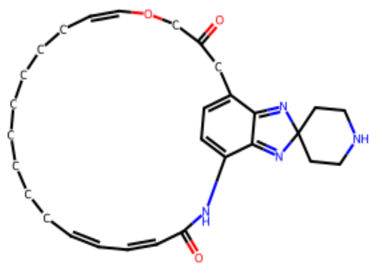
Choose File

Fragment Splitter
- Get fragments by RECAP
- Get fragments by BRICS
- Download

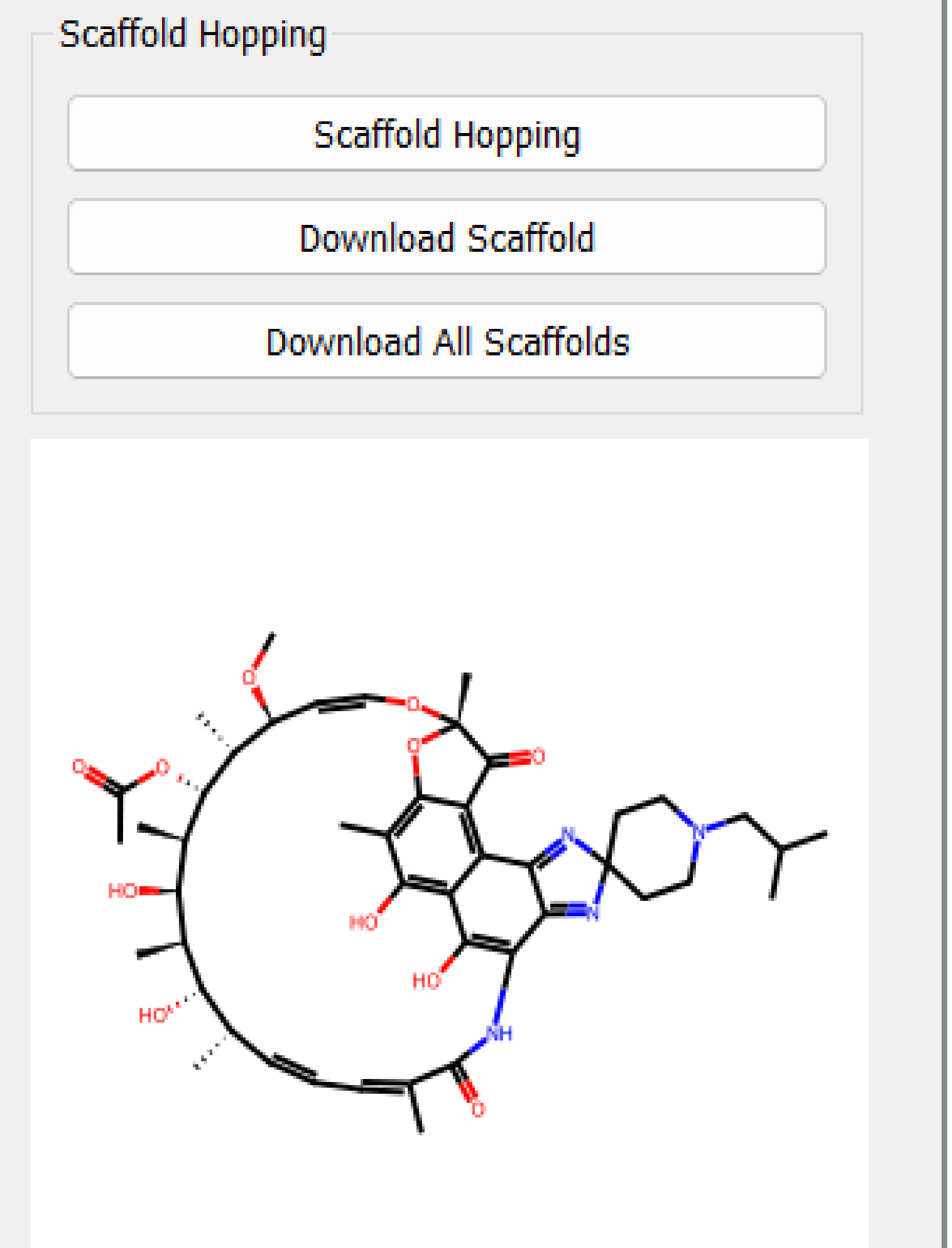
Fragment Filters
- Rule of 3 Filter

Scaffold Hopping
- Scaffold Hopping
- Download Scaffold
- Download All Scaffolds



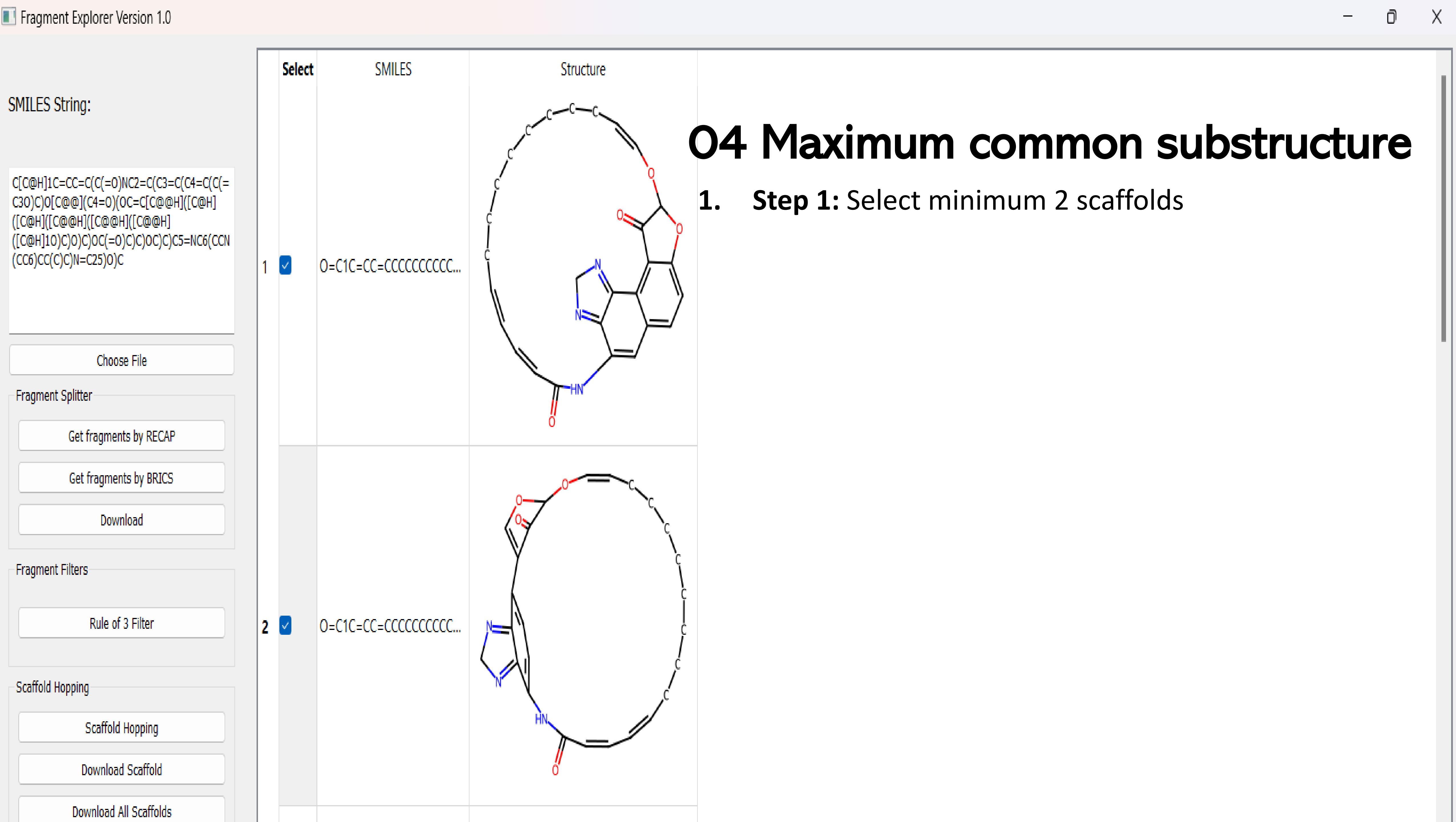
	Select	SMILES	Structure
1	☐	O=C1C=CC=CCCCCCCCCC...	
2	☐	C1CCNCC1	
3	☐	O=C1COC=CCCCCCCCCC=...	

Similarity MCS



04 Download scaffold

1. Download scaffold – selected scaffold only downloads
2. Download all scaffolds – all scaffolds are downloads



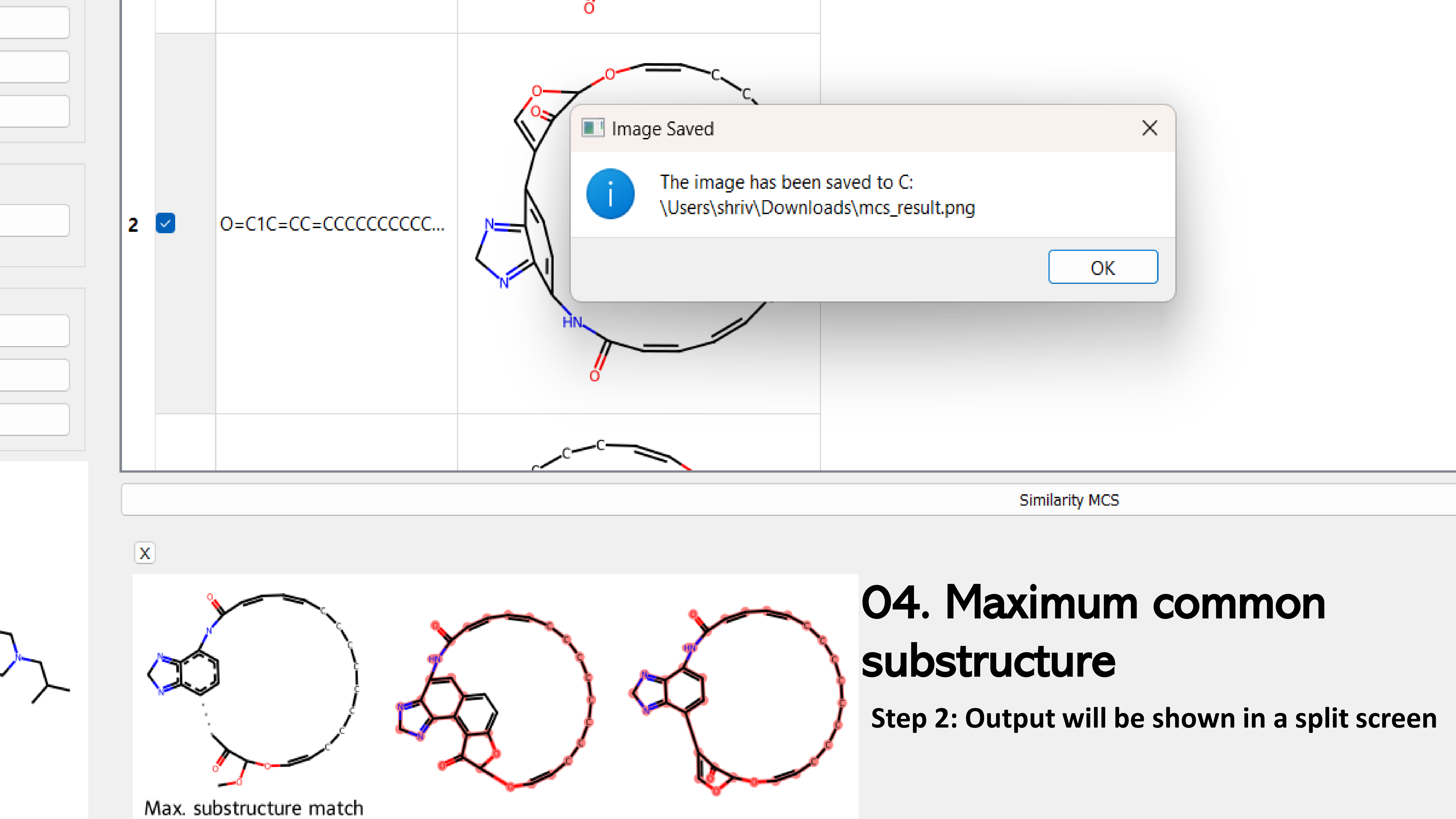
SMILES String:

C[C@H]1C=CC=C(C(=O)NC2=C(C3=C(C4=C(C(=C3O)C)O[C@@](C4=O)(OC=C[C@@H]([C@H]([C@@H]([C@@H]([C@@H]([C@H]1O)C)O)C)OC(=O)C)C)OC)C)C5=NC6(CCN(CC6)CC(C)C)N=C25)O)C

	Select	SMILES	Structure
1	✓	O=C1C=CC=CCCCCCCCC...	
2	✓	O=C1C=CC=CCCCCCCCC...	

04 Maximum common substructure

1. **Step 1:** Select minimum 2 scaffolds



2 ✓ O=C1C=CC=CCCCCCCCCC...

Image Saved

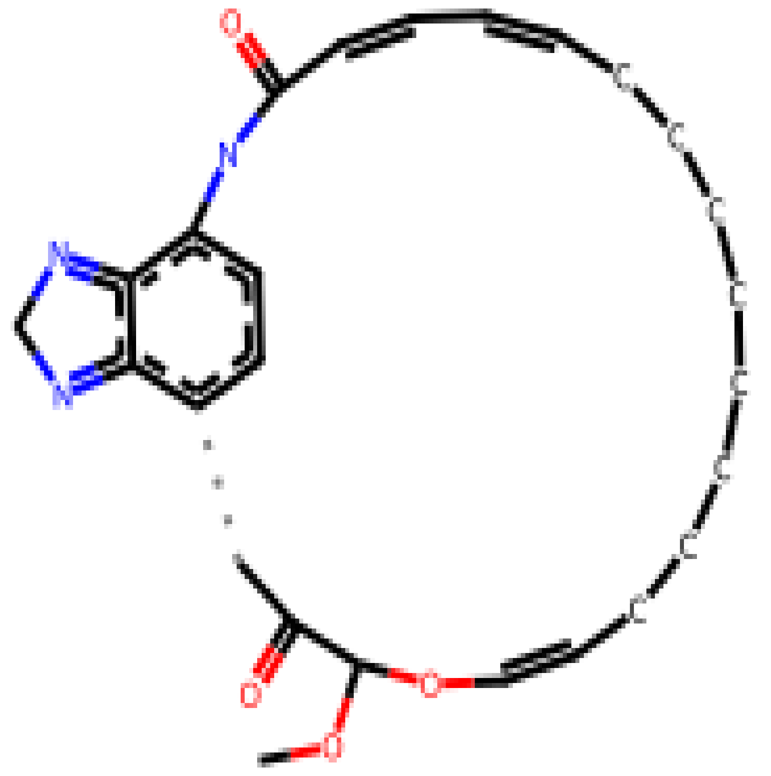
The image has been saved to C:\Users\shriv\Downloads\mcs_result.png

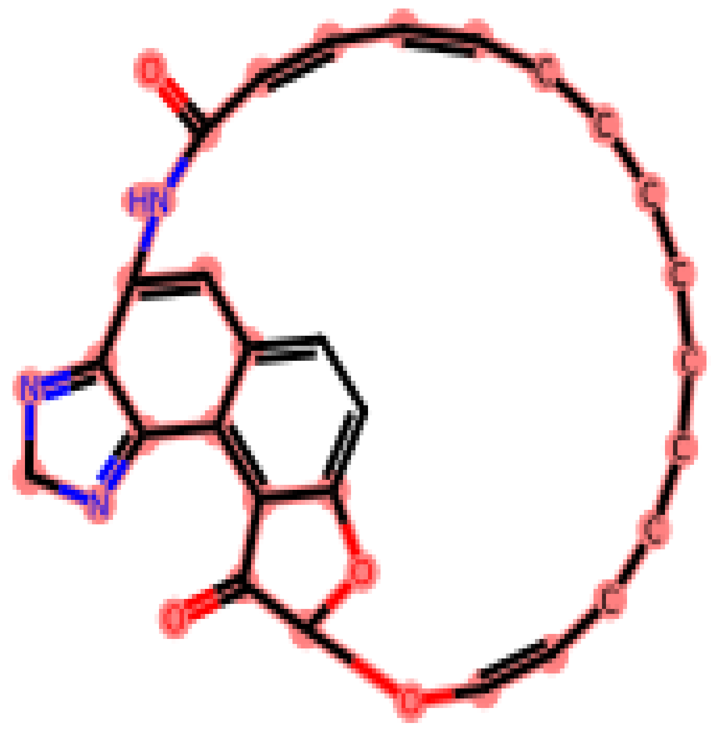
04. Maximum common substructure

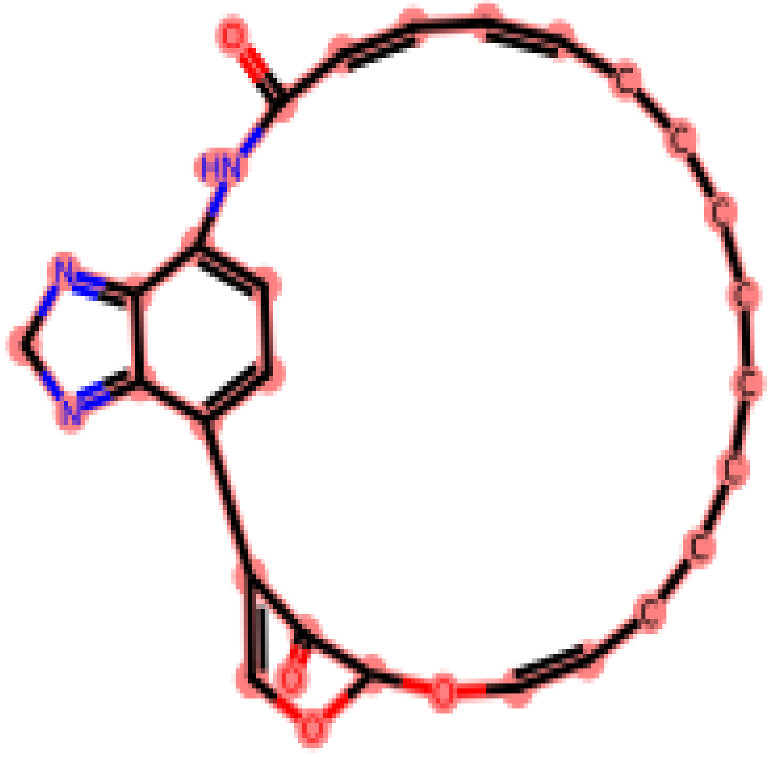
Step 2: Output will be shown in a split screen

04 Maximum common substructure

1. **Step 2: Output will be shown in a split screen**


Max. substructure match





Conclusion

1. **Educational Tool:** Designed to help researchers and students understand fragment-based drug design.
2. **Fragmentation:** Enables the study of fragmentation and its benefits.
3. **Rule of Three:** Demonstrates the advantages of filters like the rule of three.
4. **Scaffold Hopping:** Facilitates scaffold hopping for identifying new drug candidates.
5. **Maximum Common Substructure:** Explains the concept and its application in drug design.

Conclusion

6 **Library Creation:** Generates scaffold and fragment libraries in SDF format for future research.

7 **User-Friendly Interface:** Enables the study of fragmentation and its benefits.

Reference

1. Degen, J., Wegscheid-Gerlach, C., Zaliani, A., & Rarey, M. (2008, October 10). On the Art of Compiling and Using “Drug-Like” Chemical Fragment Spaces. ChemMedChem. https://doi.org/10.1002/cmdc.200800178
2. Lewell, X. Q., Judd, D. B., Watson, S. P., & Hann, M. M. (1998, April 11). *RECAPRetrosynthetic Combinatorial Analysis Procedure: A Powerful New Technique for Identifying Privileged Molecular Fragments with Useful Applications in Combinatorial Chemistry*. Journal of Chemical Information and Computer Sciences. https://doi.org/10.1021/ci970429i
3. Cao, Y., Jiang, T., & Girke, T. (2008, July 1). *A maximum common substructure-based algorithm for searching and predicting drug-like compounds*. Bioinformatics. https://doi.org/10.1093/bioinformatics/btn186
4. Kralj, S., Jukič, M., & Bren, U. (2023, April 18). *Molecular Filters in Medicinal Chemistry*. Encyclopedia. https://doi.org/10.3390/encyclopedia3020035
5. Li, Q. (2020, August 5). *Application of Fragment-Based Drug Discovery to Versatile Targets*. Frontiers in Molecular Biosciences. https://doi.org/10.3389/fmolb.2020.00180
6. Ivanov, N. N., Shulga, D. A., & Palyulin, V. A. (2023, May 24). *Decomposition of Small Molecules for Fragment-Based Drug Design*. Biophysica. https://doi.org/10.3390/biophysica3020024
7. Konteatis, Z. (2021, March 26). What makes a good fragment in fragment-based drug discovery? Expert Opinion on Drug Discovery. https://doi.org/10.1080/17460441.2021.1905629
8. Scott, O. B., & Chan, A. W. E. (2020, March 31). *ScaffoldGraph: an open-source library for the generation and analysis of molecular scaffold networks and scaffold trees*. Bioinformatics. https://doi.org/10.1093/bioinformatics/btaa219

Acknowledgement

- ScaffoldGraph from the paper: Scott, O. B., & Chan, A. W. E. (2020, March 31). *ScaffoldGraph: an open-source library for the generation and analysis of molecular scaffold networks and scaffold trees*. Bioinformatics. https://doi.org/10.1093/bioinformatics/btaa219

- RDKit: Open-source cheminformatics. https://www.rdkit.org/

Thank you