# Tautomatic – facile exploration and assessment of chemical tautomer space.
M.Forster June 2023.

## Introduction.

In the world of chemical structure data management, the concept of tautomerism is one that represents a significant challenge to automation and to unambiguous data management. If you are not familiar with the concept then Wikipedia offers a helpful starting point:
https://en.wikipedia.org/wiki/Tautomer

In essence chemical compounds contain mobile protons, which can 'hop' from between different heavy atom positions, changing the chemical sructure 'drawing' and also the bond orders. The question of which tautomer is 'correct' can often only be answered by experimentation, using methods such as NMR spectroscopy. However computational methods can often be well correlated with experiment, and do offer a systematic approach to addressing the issues.

Here we introduce some utility code known as 'Tautomatic'. This then supports the combination of some pre-existing open source tools to provide a facile workflow for tautomer exploration and scoring. The existing tools that are combined are as follows:

[1] Ambit-tautomers, created by Jeliazkova and co-workers and described here:
https://pubmed.ncbi.nlm.nih.gov/27481667/
This is used for enumerating possible tautomers but not for scoring them.

[2] Open Babel created by O'Boyle and co-workers and described here:
https://jcheminf.biomedcentral.com/articles/10.1186/1758-2946-3-33
This is used for convenient file format manipulation.

[3] the density functional tight binding code (xtb) created by Grimme and co-workers and deseibed here:
https://pubs.acs.org/doi/full/10.1021/acs.jctc.8b01176
This is used for re-scoring of the enumerated tautomers, in vacuo or in simulated solvents.

The two python language scripts that constitute the tautomatic code call these binaries using operating system calls. It is essential to have the correct mamba / conda environment running, and that the binary files called are available in the path. The which command (see below) is valuable for trouble shooting these aspects of your configuration.

## Installation.

Here we describe how to set up and run the code on the examples provided. All of this assumes a Linux or Unix type computing envrionment with the Conda or Mamba python related tools already installed and available. In this documentation we useapt-get install openjdk-19-jre mamba simply because it is quicker and more reliable. Here we describe the commands that need to be run in a Linux type shell to get started.

[1] Create Python environment called tautomatic, then activate that environment

```
mamba create --name tautomatic
mamba activate tautomatic
```

[2] Install needed tools into the environment

```
mamba install -c conda-forge openbabel
mamba install -c conda-forge xtb

which obabel
=> indicates path to obabel executable

which xtb
=> indicates path to xtb executable.
```

From here on we assume that the environment has been activated before trying to use the tools.

[3] Download the ambit tautomers java jar file from the sourceforge URL.
https://sourceforge.net/projects/ambit/files/Ambit2/AMBIT%20applications/tautomers/ambit-tautomers-2.0.0-SNAPSHOT.jar

From here on we assume that this jar file, or a link to it, is present in the directory where you are running the tools, or that a relevant PATH has been set up.

[4] Check if a Java run time environment is present on your machine, using this command

```
which java
=> indicates path to java executable
```

If no java run time is present then install one using a command such as

```
sudo apt-get install openjdk-19-jre
```

From here on we assume you have a working java environment available.

[5] Obtain the tautomatic scripts from Github

```
git clone https://github.com/xyzmjf/tautomatic.git
```

[6] Finish tidying up – move to directory ready to run scripts.

```
cd tautomatic
```

Ensure you have a link to the jar file in this directory (example)

```
ln -s ../ambit-tautomers-2.0.0-SNAPSHOT.jar .
```

## Running the code

The first of the tautomatic scripts uses the underlying Ambit tautomer code to enumerate a number of possible tautomers for the molecule Uracil. The input file is a 'one liner' containing the SMILES string and a convenient name for the molecule, in a file called uracil.smi. The file contants are as shown below:

```
Oc1nccc(O)n1 uracil
```

[1] generating tautomers.
To run the tautomer generation step use the following command:

```
./generate_tautomers.py uracil.smi > uracil_tautomers.smi
```

The output file uracil_tautomers.smi should look like this

```
Oc1nccc(O)n1 uracil
O=C1N=C(O)NC=C1 uracil_2
Oc1nccc(O)n1 uracil_3
O=C1C=CN=C(O)N1 uracil_4
O=C1N=C(O)C=CN1 uracil_5
O=C1C=CNC(=O)N1 uracil_6
Oc1nccc(O)n1 uracil_7
O=C1N=C(O)N=CC1 uracil_8
O=C1N=CC=C(O)N1 uracil_9
O=C1N=CCC(O)=N1 uracil_10
O=C1N=CCC(=O)N1 uracil_11
```

The first SMILES string represents the input molecule. In total a set of 12 tautomeric forms have been created. Some of these such as uracil_3 and uracil_7 are actually duplicates. If desired these can be removed manually, but they are easily recognised after the scoring step.

[2] Scoring tautomers in vacuo.
Next we use the xtb code to score the relative energies of these tautomers, in-vacuo, using the density functional tight binding formalism.

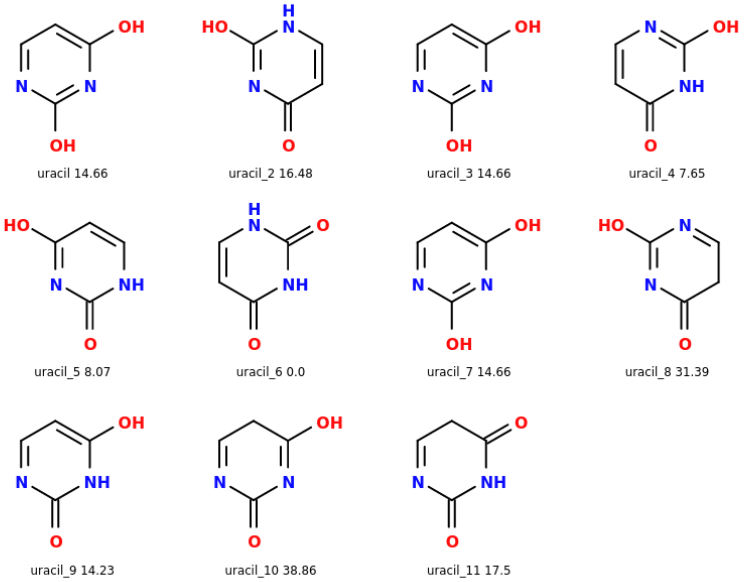
Run the following command:

```
./score_tautomers.py uracil_tautomers.smi > uracil_tautomers.txt
```

The output file should look like this:

```
Oc1nccc(O)n1 uracil -15431.44
O=C1N=C(O)NC=C1 uracil_2 -15429.62
Oc1nccc(O)n1 uracil_3 -15431.44
O=C1C=CN=C(O)N1 uracil_4 -15438.45
O=C1N=C(O)C=CN1 uracil_5 -15438.03
O=C1C=CNC(=O)N1 uracil_6 -15446.10
Oc1nccc(O)n1 uracil_7 -15431.44
O=C1N=C(O)N=CC1 uracil_8 -15414.71
O=C1N=CC=C(O)N1 uracil_9 -15431.87
O=C1N=CCC(O)=N1 uracil_10 -15407.24
O=C1N=CCC(=O)N1 uracil_11 -15428.60
```

The numeric values are the xtb computed total energies in Kcal/mol.
The lowest energy tautomer according to this scheme is therefore uracil_6.
The relative energies and structures are shown in the image below.




It is clear that the input structure (uracil), uracil_3 and uracil_7 are duplicates.
It may be possible to remove these duplicates in a future software update.

It is remarkable that such a small structure with just 8 heavy atoms can lead to a significant number of tautomeric forms. This leads to many complications in areas such as storing chemical structures in databases, or indeed in structure based modelling for drug design.

[3] Comparison with some literature.
The calculation of the relative energies of uracil tautomers by quantum mechanics is a subject that has been previously studied in the scientific literature, over a number of years. A thorough review cannot be performed here. However, we can make a comparison to the recent work of Szatylowicz publsihed in 2022.

https://www.mdpi.com/1420-3049/27/21/7240

This paper is open access so can be readily studied. These authors utilise density functional calculations with the B97-D3/aug-cc-pVDZ method and the Gaussian-16 code. In Figure 1 they report the four most stable tautomers denoted u1,u2,u3,u4. These correspond to the structures uracil_6, uracil_4, uracil_5 and uracil_7 (or uracil_3 or uracil as these are identical) described here. Hence most stable structures found here, and indeed their energy ordering are consistent with. Note that these authors do not report a structure corresponding to uracil_9 which here has slightly lower energy than uracil_7. However the latter structure does possess different rotameric forms which can perturb the energy, and that complexity is not considered here.

[4] Scoring tautomers in solvent
The xtb code provides tools to compute relative molecular energies in implicit solvent.
For more details see the xtb documentation here:

https://xtb-docs.readthedocs.io/en/latest/gbsa.html

To run such a scoring calculation using water as the implicit solvent use the commands

```
./score_tautomers.py uracil_tautomers.smi water > uracil_water.txt
```

Note that the relative computed energies of tautomers can and do change with solvent.
Further comparisons of xtb computed energies in implicit solvents, and comparison
with experimental data are under sonsideration.

Hope this is useful. :-)