The `standardiser` tool

Francis Atkinson

Chemogenomics Group

EMBL-EBI

Chemical Structure Standardisation

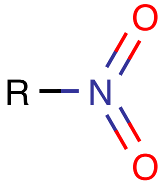
- Combining heterogeneous sources of chemical structures can cause problems due to differing representations
- The ChEMBL database's methods of dealing with these issues will be discussed first as background
- The `standardiser` tool will then be introduced
- Please note that, while this tool was inspired by the ChEMBL protocols, it is a separate project

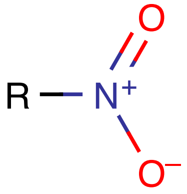
What is ChEMBL?

- A freely-available source of bioactivity data
  - Bulk of data is for small organic molecules
    - A small amount is for inorganics, biologicals *etc.*
- Core is data from key MedChem journals
  - J. Med. Chem, Bioorg. Med. Chem. Lett.
- Supplemented by…
  - Subset of data from PubChem
    - Only full-curve data taken at present
  - Other free databases
    - *e.g.* DrugMatrix, TPSearch
  - Deposited datasets
    - *e.g.* NTD consortia, GSK kinase set
- Heterogeneous sources of chemical structures!

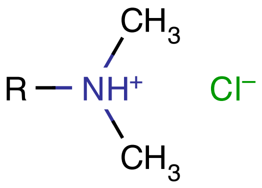
Example of issues

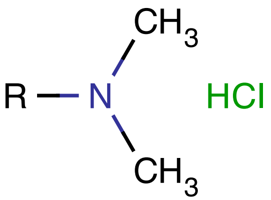
- Hypervalent *vs.* charge-separated





- Charged *vs.* neutral salts

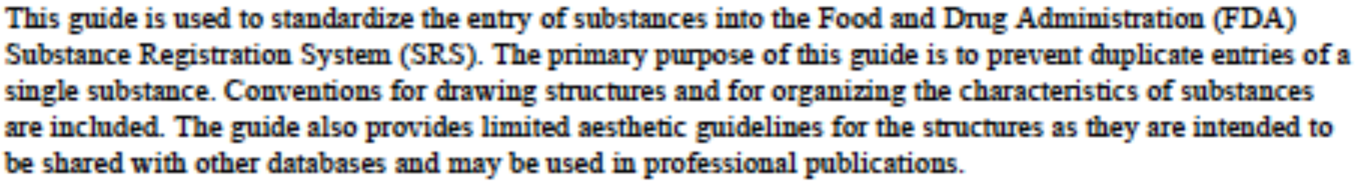




- Chemical databases cantreat these as different
  - Could cause SSS to fail if alternative depiction used as query
  - Standardization is required

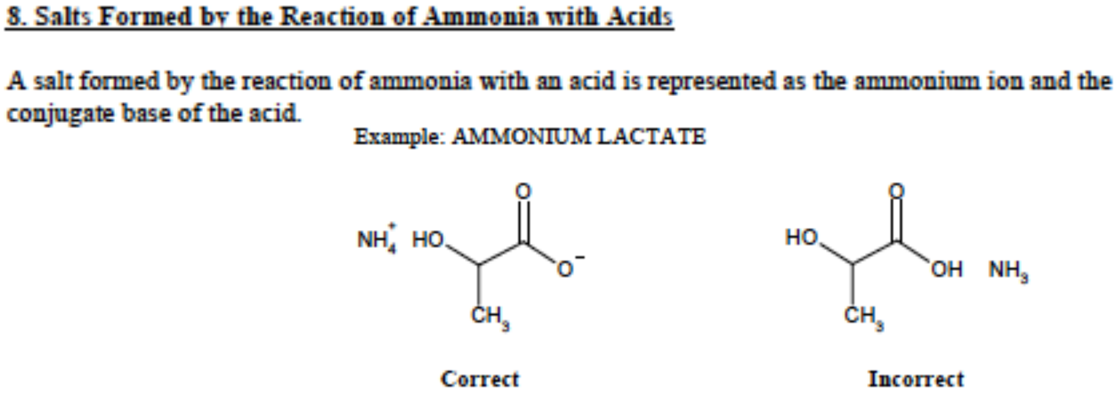
Standardisation

- The FDA have published a compound depiction SOP for their Substance Registration System



This guide is used to standardize the entry of substances into the Food and Drug Administration (FDA) Substance Registration System (SRS). The primary purpose of this guide is to prevent duplicate entries of a single substance. Conventions for drawing structures and for organizing the characteristics of substances are included. The guide also provides limited aesthetic guidelines for the structures as they are intended to be shared with other databases and may be used in professional publications.

- Used as basis for ChEMBL standardisation rules
  - with some modifications, *e.g.* ...



8. Salts Formed by the Reaction of Ammonia with Acids

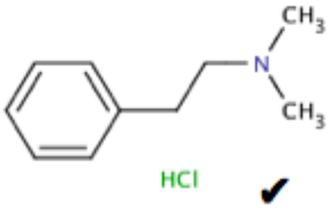
A salt formed by the reaction of ammonia with an acid is represented as the ammonium ion and the conjugate base of the acid.

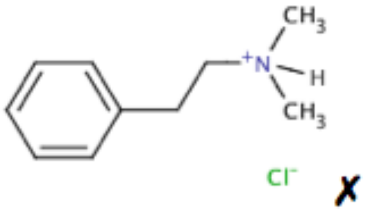
Example: AMMONIUM LACTATE

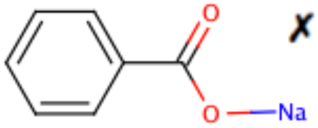
ChEMBL would treat these the same as other salts, *i.e.* they would be neutralized

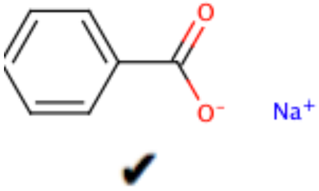
Examples of rules: charges

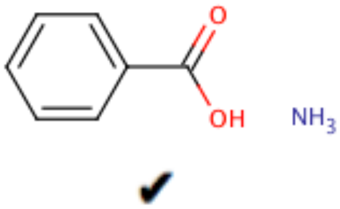
- Consistent rules for drawing salts

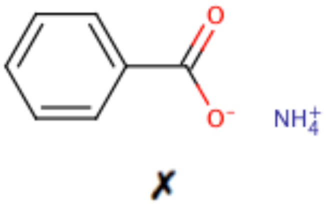


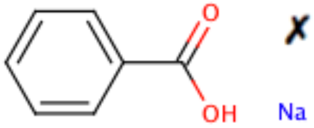








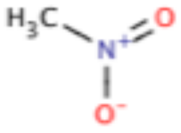




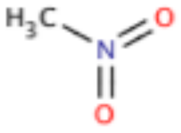
- Compounds to be charge-neutral if possible
  - quaternary N an exception if counterion unknown

Example of rules: functional groups

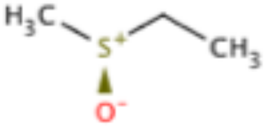
- Charge-separated preferred over hypervalent



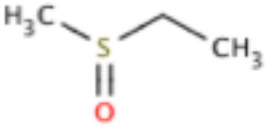
✓



✗

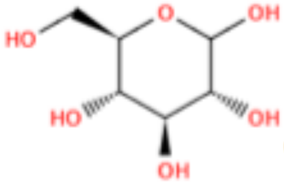


✓

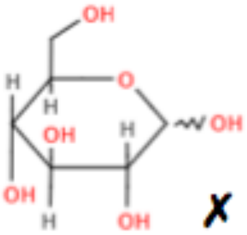


✗

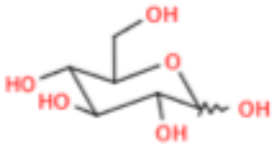
- Stylistic rules for sugars, peptides, steroids *etc.*



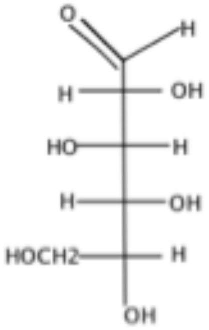
✓



✗



✗



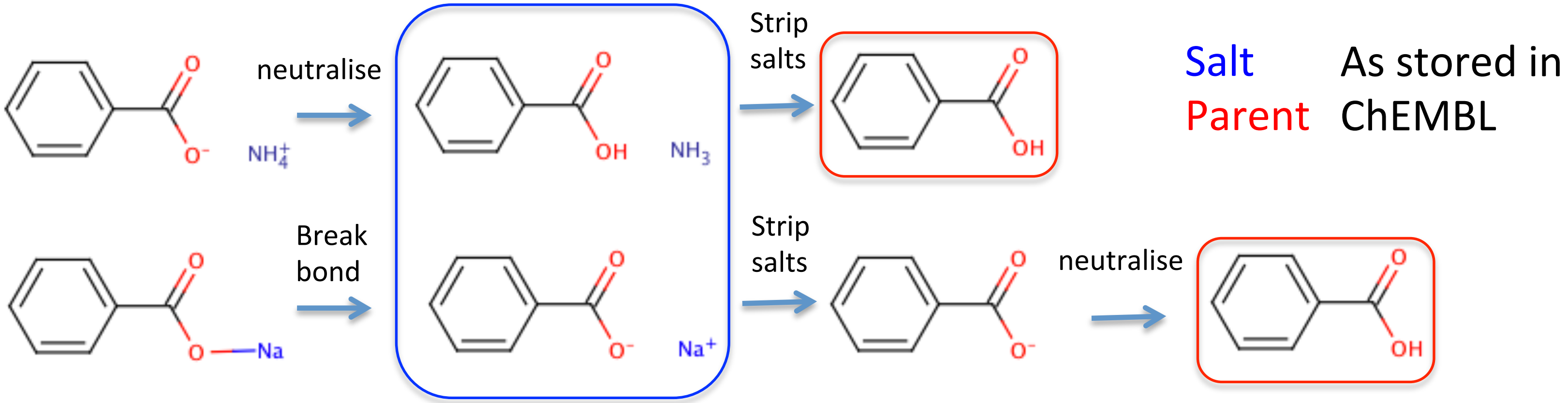
✗

Salt stripping

- Salt/solvate components should not affect the biological activity of a compound
  - There can be exceptions
    - *e.g.* salt form may affect solubility
  - However, any 3D or QSAR modelling should be done using only the bioactive 'parent'
- It is desirable to be able to view compounds with a common parent together
- It can be appropriate to aggregate data for the parent compound

Salt stripping (2)

- Counterions and solvent molecules are removed using a custom ‘salt dictionary’
  - Acids with inorganic cations may require a further round of charge-neutralization…

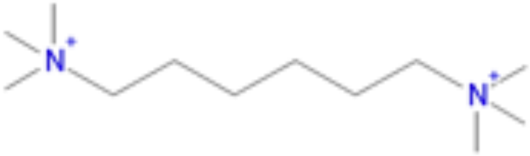


Salt / Parent — As stored in ChEMBL

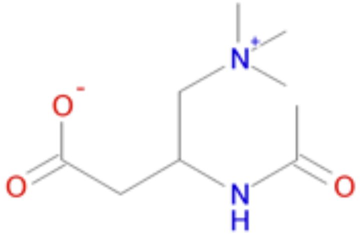
- The parent is registered as a separate structure
  - The parent and original salt are linked *via* the ‘molecule hierarchy’ table

Curator intervention

- Structures with permanent charges
  - *i.e.* counterion not recorded
  - Zero charge used as check on neutralization steps



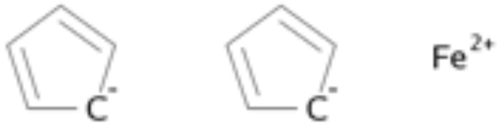
- Zwitterions
  - Naïve neutralization could introduce errors



- These types automatically routed for inspection
  - Possible manual standardisation

Inorganics & organometallics

- Handled poorly by current chemoinformatics tools…
  - multicenter bonding
  - coordination complexes
  - Non tetrahedral stereochemistry
    - Cannot distinguish *cis*- and *trans*-platins



ferrocene ?!

- Charge-balancing, salt-stripping *etc.* are difficult
- Cannot be properly searched for by cartridge
- Not put through main standardisation process
  - May be redrawn for clarity in some cases
    - *e.g.* approved drugs such as the platins
- Structures now excluded (post-CHEMBL17)
  - ~3200 structures affected
  - Bioactivities are retained

Implementation in ChEMBL

- Rules are applied *via* Pipeline Pilot protocols
  - Run by ChEMBL's chemical curator
  - Allow manual intervention in difficult cases
- The protocols are available on request
- However, they are…
  - tightly coupled with data-loading pipeline
  - subject to change as new issues identified
- Pipeline Pilot is commercial software

standardiser

- Tool to pre-process structures for modelling
  - Funded by IMI eTOX project
- Molecular representation can affect results…
  - Descriptor calculation
  - Docking
  - QM
- Need to standardise representation
  - Same for both training and application

standardiser

- ‘Inspired by’ ChEMBL curation strategy
  - An entirely separate project , however
- Several key differences to ChEMBL…
  - Only interested in parent (bioactive) component
  - Attempts to standardise tautomers
    - *i.e.* hydroxy-pyridine -> pyridone
    - *N.B.* Does *not* attempt tautomer canonicalization
  - No manual intervention
- Implemented in Python & RDKit
  - Fully open-source

Procedure

- Break bonds to Group I or II metals
- Neutralize charges by adding/removing protons
- Apply standardization rules
- Neutralize any charges exposed by rules
- Discard any salt/solvate components
- Return standardized parent

Limitations

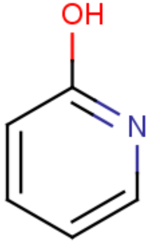
- Rule set could be expanded
  - *e.g.* will review RCS / CVSP rules
- No attempt to handle inorganics
  - Impractical with current tools
- No attempt to produce ‘canonical’ tautomer
  - Could flag tautomeric molecules for inspection?
- re-charging not handled

Key differences to ChEMBL

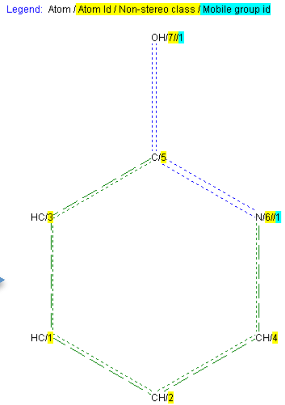
- ChEMBL makes no attempt to standardise tautomers
- InChI codes are used for registration (*i.e.* assignment of database identifiers/keys)
  - *i.e.* two molecules are the if they have the same InChI
- For any molecule, the first-encountered tautomer will always be used
  - to generate images, for searching and in downloadable SDF files *etc.*
- Thus, for example: if the first time a molecule encountered it is shown as the hydroxy-pyridine, this tautomer will be stored (as a molfile) and will be used for the depiction of this molecule even where an alternative tautomer is encountered in a later document
- By contrast, standardiser *does* attempt to standardise tautomers
  - *e.g.* the pyridone is preferred, as it is likely to be the lower-energy tautomer

ChEMBL

Encountered first: molfile stored in database against this InChI

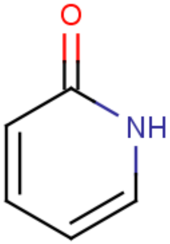


hydroxy-pyridine



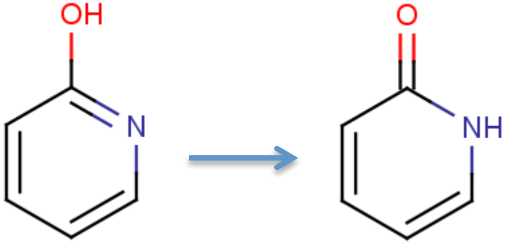
InChI=1S/C5H5NO/c7-5-3-1-2-4-6-5/h1-4H,(H,6,7)

Encountered second: hydroxy-pyridine molfile will be used for depiction *etc.*



pyridone

standardiser

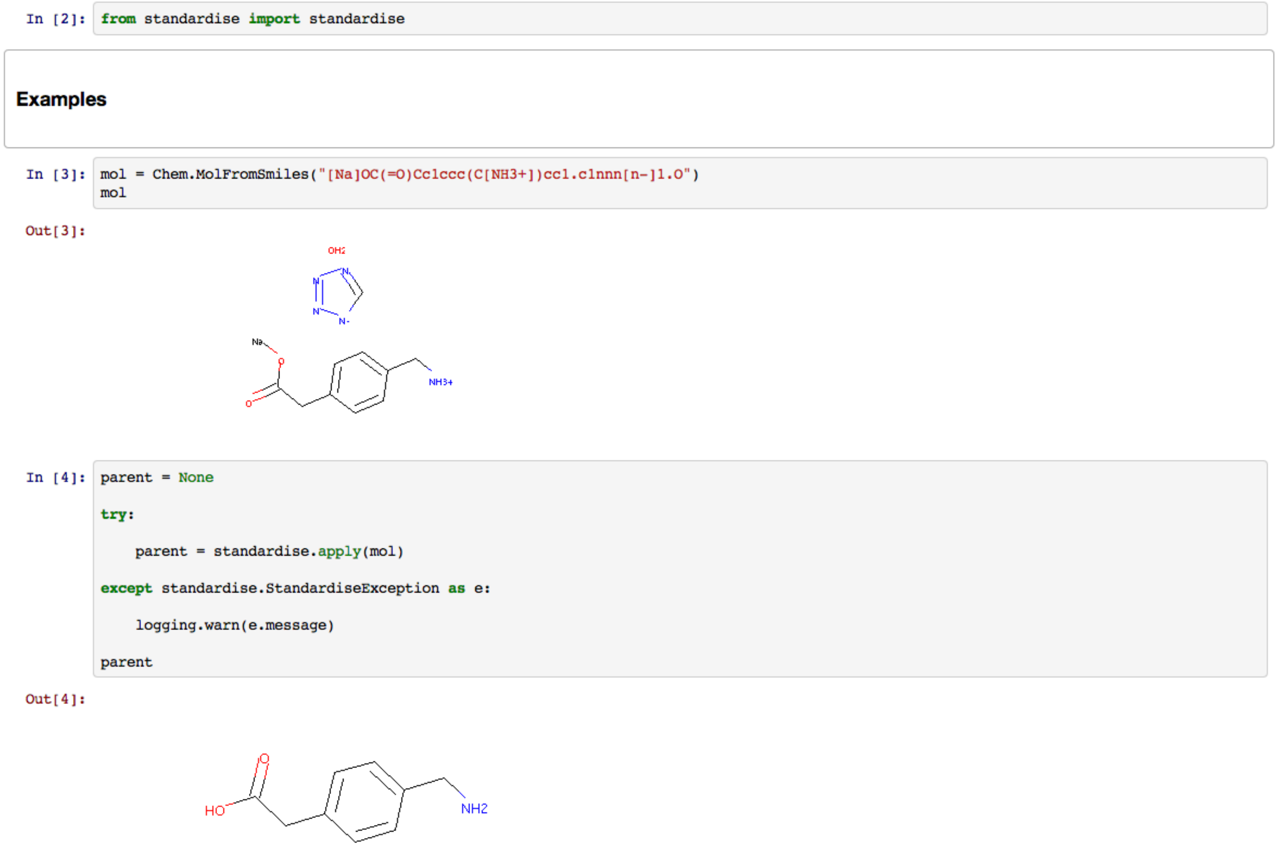


input

output

Example

- Provided as a Python package
- A simple driver program for batch processing is included
  - Accepts SDF or SMILES input



In [2]:
```
from standardise import standardise
```

Examples

In [3]:
```
mol = Chem.MolFromSmiles("[Na]OC(=O)Cc1ccc(C[NH3+])cc1.c1nnn[n-]1.O")
mol
```

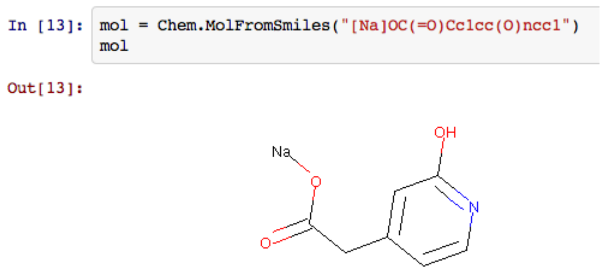
Out[3]:

In [4]:
```
parent = None
try:
    parent = standardise.apply(mol)
except standardise.StandardiseException as e:
    logging.warn(e.message)
parent
```

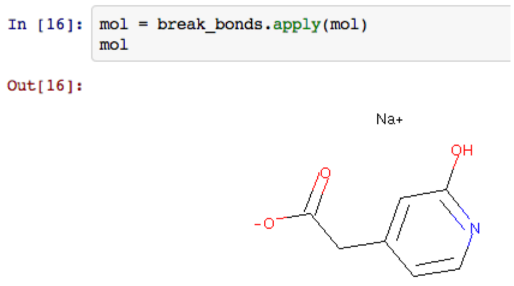
Out[4]:

Modules within package

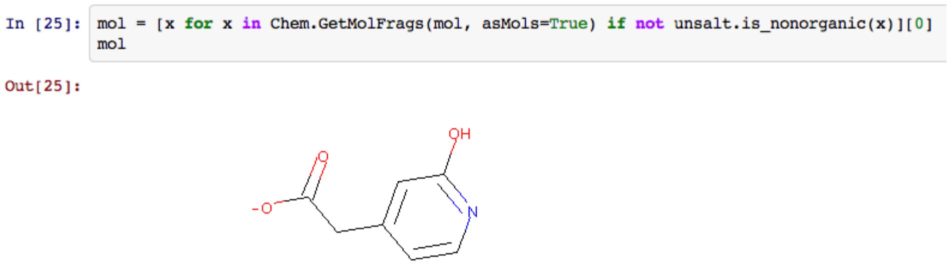
- Modules implementing different steps may be called independently
  - could be incorporated into different workflows



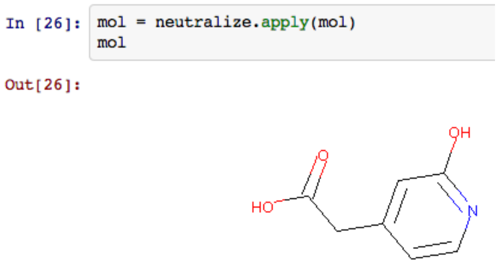
```
In [13]: mol = Chem.MolFromSmiles("[Na]OC(=O)Cc1cc(O)ncc1")
         mol

Out[13]:
```



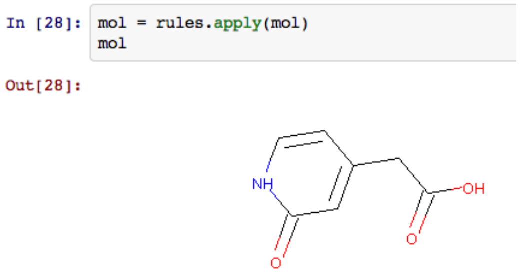
```
In [16]: mol = break_bonds.apply(mol)
         mol

Out[16]:
```



```
In [25]: mol = [x for x in Chem.GetMolFrags(mol, asMols=True) if not unsalt.is_nonorganic(x)][0]
         mol

Out[25]:
```



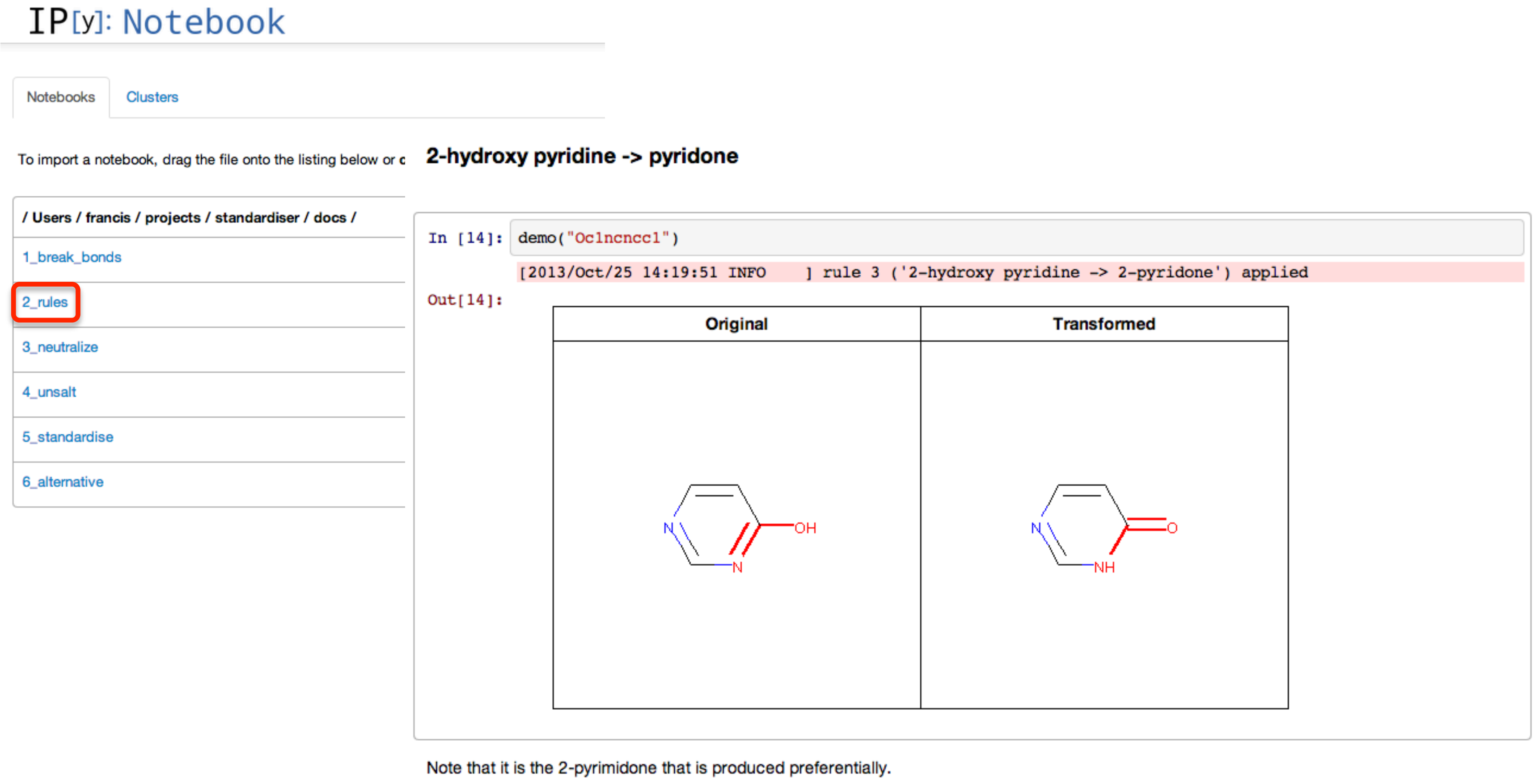
```
In [26]: mol = neutralize.apply(mol)
         mol

Out[26]:
```



```
In [28]: mol = rules.apply(mol)
         mol

Out[28]:
```

Documentation

- Provided as IPython Notebooks
  - Also available as static web pages



IP[y]: Notebook

To import a notebook, drag the file onto the listing below or c

/ Users / francis / projects / standardiser / docs /

- 1_break_bonds
- 2_rules
- 3_neutralize
- 4_unsalt
- 5_standardise
- 6_alternative

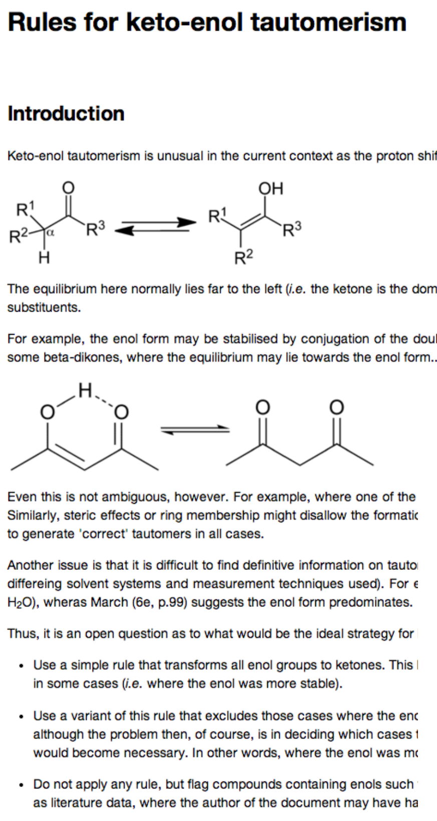
2-hydroxy pyridine -> pyridone

```
In [14]: demo("Oc1ncncc1")
         [2013/Oct/25 14:19:51 INFO     ] rule 3 ('2-hydroxy pyridine -> 2-pyridone') applied
Out[14]:
```

Note that it is the 2-pyrimidone that is produced preferentially.

Futher Documentation

- Also included are some pages describing various issues
  - Intended to stimulate debate on best practices



Rules for keto-enol tautomerism

Introduction

Keto-enol tautomerism is unusual in the current context as the proton shift involves a carboı

The equilibrium here normally lies far to the left (*i.e.* the ketone is the dominant tautomer), bı substituents.

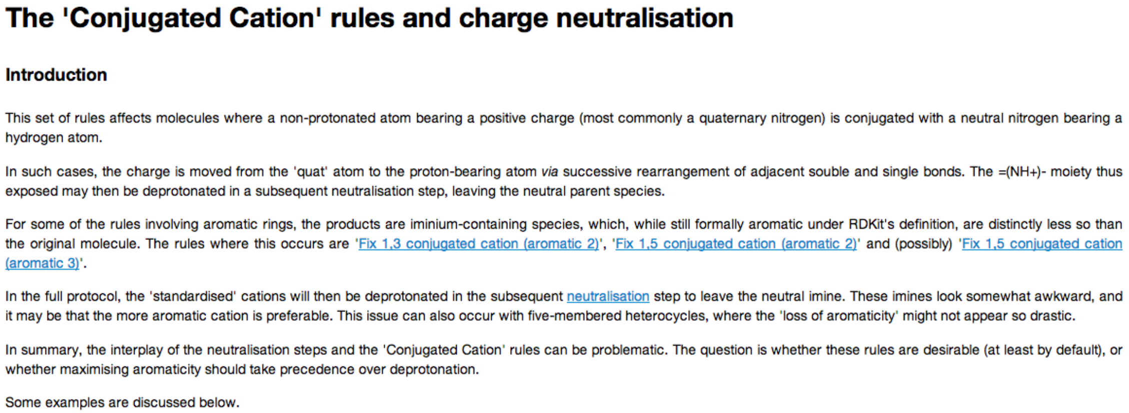
For example, the enol form may be stabilised by conjugation of the double-bond with an ε some beta-dikones, where the equilibrium may lie towards the enol form...

Even this is not ambiguous, however. For example, where one of the Similarly, steric effects or ring membership might disallow the formatic to generate 'correct' tautomers in all cases.

Another issue is that it is difficult to find definitive information on tauto differeing solvent systems and measurement techniques used). For ε H_2O), wheras March (6e, p.99) suggests the enol form predominates.

Thus, it is an open question as to what would be the ideal strategy for

- Use a simple rule that transforms all enol groups to ketones. This ı in some cases (*i.e.* where the enol was more stable).
- Use a variant of this rule that excludes those cases where the enc although the problem then, of course, is in deciding which cases ı would become necessary. In other words, where the enol was mc
- Do not apply any rule, but flag compounds containing enols such as literature data, where the author of the document may have ha



The 'Conjugated Cation' rules and charge neutralisation

Introduction

This set of rules affects molecules where a non-protonated atom bearing a positive charge (most commonly a quaternary nitrogen) is conjugated with a neutral nitrogen bearing a hydrogen atom.

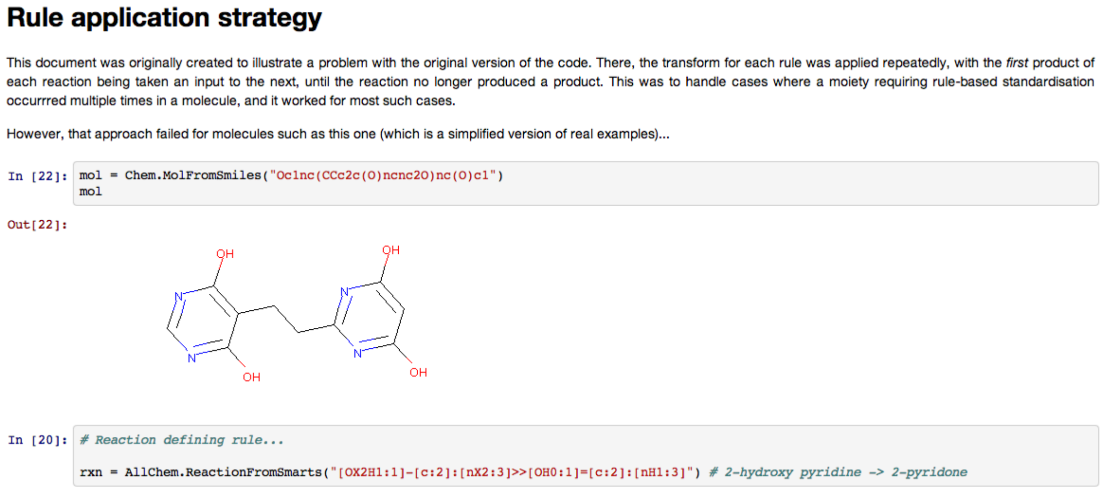
In such cases, the charge is moved from the 'quat' atom to the proton-bearing atom *via* successive rearrangement of adjacent souble and single bonds. The =(NH+)- moiety thus exposed may then be deprotonated in a subsequent neutralisation step, leaving the neutral parent species.

For some of the rules involving aromatic rings, the products are iminium-containing species, which, while still formally aromatic under RDKit's definition, are distinctly less so than the original molecule. The rules where this occurs are 'Fix 1,3 conjugated cation (aromatic 2)', 'Fix 1,5 conjugated cation (aromatic 2)' and (possibly) 'Fix 1,5 conjugated cation (aromatic 3)'.

In the full protocol, the 'standardised' cations will then be deprotonated in the subsequent neutralisation step to leave the neutral imine. These imines look somewhat awkward, and it may be that the more aromatic cation is preferable. This issue can also occur with five-membered heterocycles, where the 'loss of aromaticity' might not appear so drastic.

In summary, the interplay of the neutralisation steps and the 'Conjugated Cation' rules can be problematic. The question is whether these rules are desirable (at least by default), or whether maximising aromaticity should take precedence over deprotonation.

Some examples are discussed below.



Rule application strategy

This document was originally created to illustrate a problem with the original version of the code. There, the transform for each rule was applied repeatedly, with the *first* product of each reaction being taken an input to the next, until the reaction no longer produced a product. This was to handle cases where a moiety requiring rule-based standardisation occurrred multiple times in a molecule, and it worked for most such cases.

However, that approach failed for molecules such as this one (which is a simplified version of real examples)...

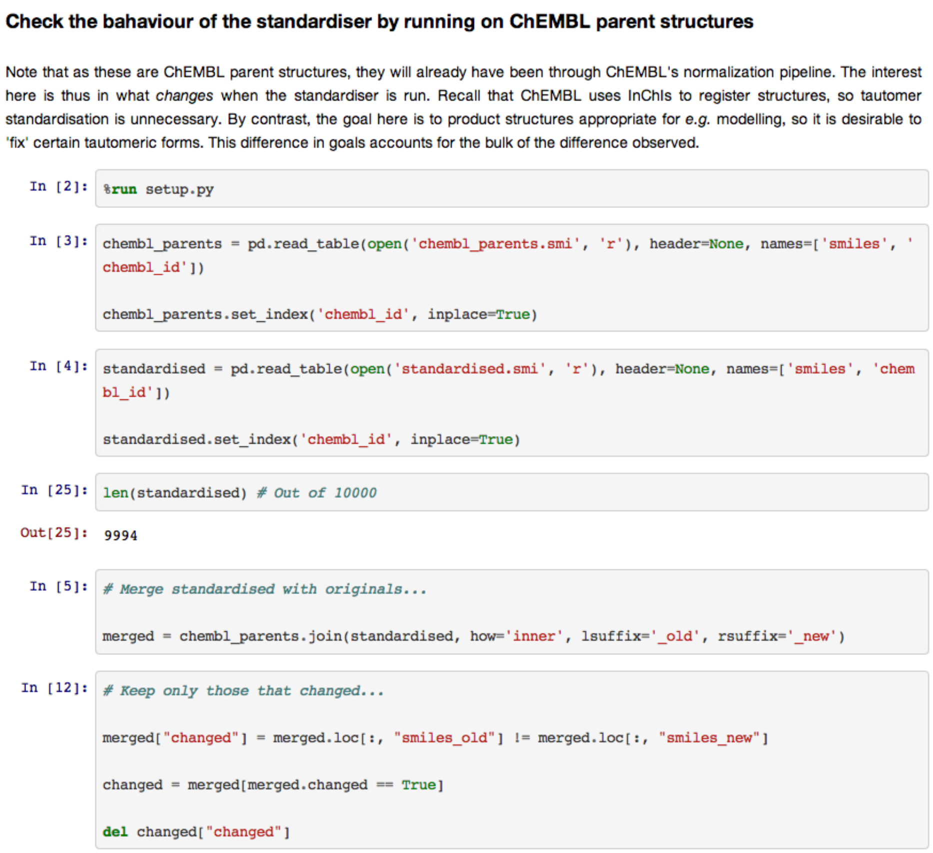
```
In [22]: mol = Chem.MolFromSmiles("Oc1nc(CCc2c(O)ncnc2O)nc(O)c1")
         mol

Out[22]:
```

```
In [20]: # Reaction defining rule...

         rxn = AllChem.ReactionFromSmarts("[OX2H1:1]-[c:2]:[nX2:3]>>[OH0:1]=[c:2]:[nH1:3]") # 2-hydroxy pyridine -> 2-pyridone
```

n step...

Futher Documentation

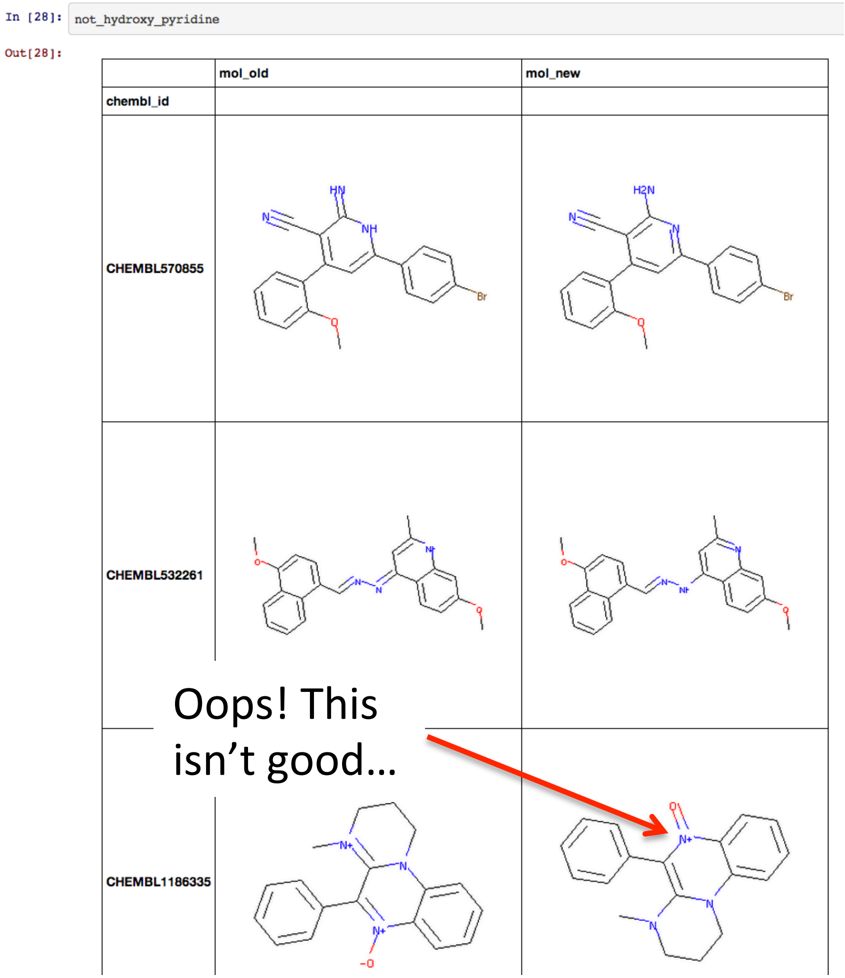
- The tool will be run on diverse test sets and the results posted
  - Again, designed to detect flaws and promote discussion



Check the bahaviour of the standardiser by running on ChEMBL parent structures

Note that as these are ChEMBL parent structures, they will already have been through ChEMBL's normalization pipeline. The interest here is thus in what *changes* when the standardiser is run. Recall that ChEMBL uses InChIs to register structures, so tautomer standardisation is unnecessary. By contrast, the goal here is to product structures appropriate for *e.g.* modelling, so it is desirable to 'fix' certain tautomeric forms. This difference in goals accounts for the bulk of the difference observed.

```
In [2]:  %run setup.py

In [3]:  chembl_parents = pd.read_table(open('chembl_parents.smi', 'r'), header=None, names=['smiles', '
         chembl_id'])

         chembl_parents.set_index('chembl_id', inplace=True)

In [4]:  standardised = pd.read_table(open('standardised.smi', 'r'), header=None, names=['smiles', 'chem
         bl_id'])

         standardised.set_index('chembl_id', inplace=True)

In [25]: len(standardised) # Out of 10000

Out[25]: 9994

In [5]:  # Merge standardised with originals...

         merged = chembl_parents.join(standardised, how='inner', lsuffix='_old', rsuffix='_new')

In [12]: # Keep only those that changed...

         merged["changed"] = merged.loc[:, "smiles_old"] != merged.loc[:, "smiles_new"]

         changed = merged[merged.changed == True]

         del changed["changed"]
```



```
In [28]: not_hydroxy_pyridine

Out[28]:
```

	mol_old	mol_new
chembl_id		
CHEMBL570855		
CHEMBL532261		
CHEMBL1186335		

Oops! This isn't good...