# Why is it so difficult to replace the old with the new?

Learnings from dealing with legacy systems in AstraZeneca and how RDKit can help

Susan Leung and Nick Tomkinson

September 2023
RDKit UGM 2023, Mainz

# Outline

- Background/Motivation
- Studies
  1. Import/export
     - Methods
     - Results
  2. Molecular uniqueness checking
     - Methods
     - Results
- Conclusion/future work

# Background

- Chemistry toolkits and legacy systems can be in place for many years in companies as replacing these systems can be very challenging.
- AZ has systems in place to handle molecular data.
- Various software and chemistry toolkits are used. In some instances, workarounds have been built to handle edge and corner cases….
- So why would we consider anything else?

# Motivations

## Reasons to change

- Cost benefit
- Scalability
- Open-code based
- Portability
- Accuracy
- Other enhancements

## Reasons NOT to change

- Extensive evaluations are required
  - Extent of impact – how many molecules will be affected?
  - Need to assess implications on upstream/downstream processes
- Accuracy
- Unknown unknowns?
- People will need to become familiar with new system

# Motivations

## Reasons to change

- Cost benefit
- Scalability
- Open-code based
- Portability
- Accuracy
- Other enhancements

## Reasons NOT to change

- Extensive evaluations are required
  - Extent of impact – how many molecules will be affected?
  - Need to assess implications on upstream/downstream processes
- Accuracy
- Unknown unknowns?
- People will need to become familiar with new system

We have been embarking on this journey….

What would happen if we replace our current systems/toolkits with an RDKit-based one…

# Studies

1. **<u>Import/export</u>**
   - Can RDKit read and write AZ molecules without error?
   - Are any features lost?
2. **<u>Uniqueness checking</u>**
   - Main source of differences in uniqueness checking?

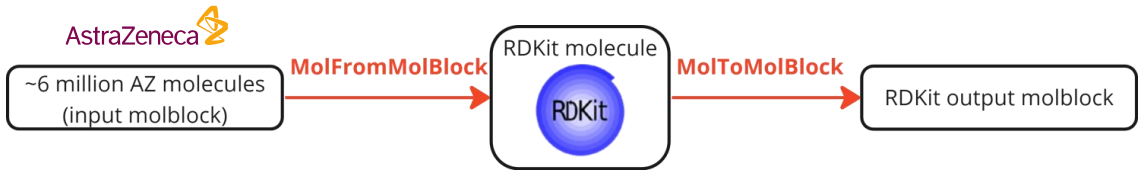
Note: all molecules in this presentation are public or made up or replaced with minimal examples for confidentiality.

# Study 1: Import/Export

# Import/export

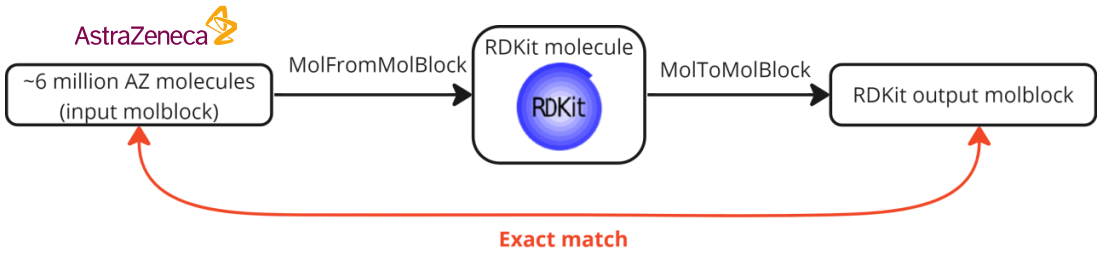
## Study 1.1:

- Q: Can RDKit read and write AZ molecules without error?



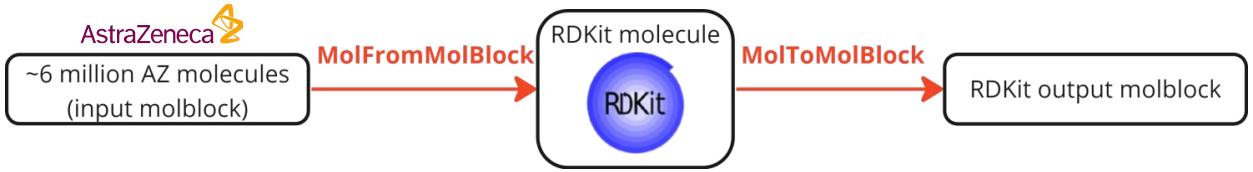

## Study 1.2:

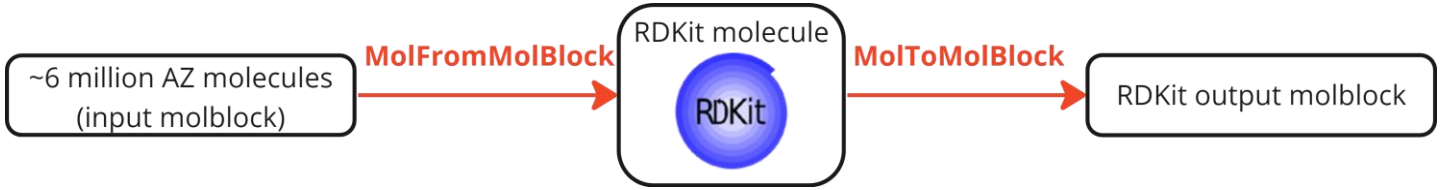
- Q: Are any features lost?



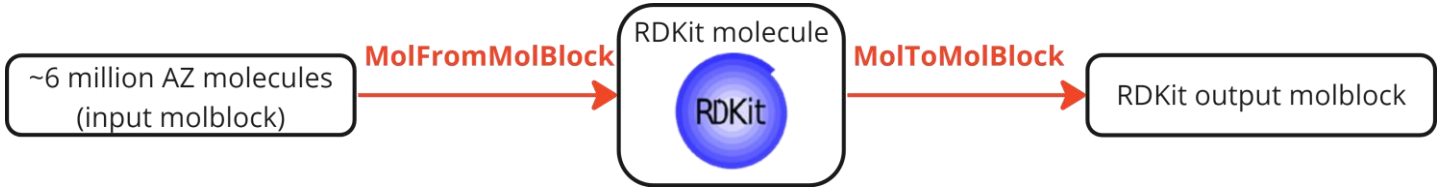
# Study 1.1: Import/Export

Read in, read out



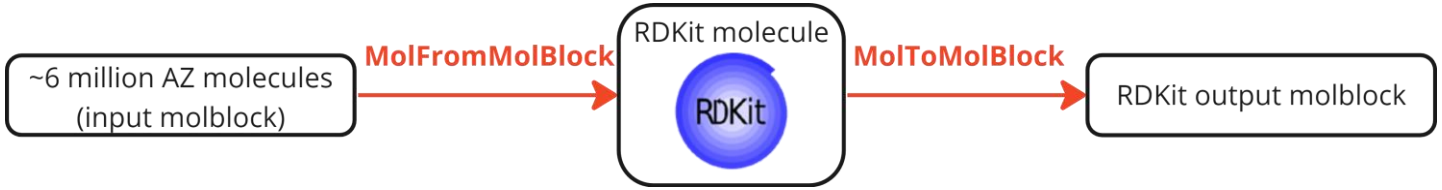



| Warning/error from MolFromMolBlock | Molblock written from MolToMolBlock | % |
|---|---|---|
| | | |
| | | |




- Most molecules (>99.9%) were successfully read/written.

| Warning/error from MolFromMolBlock | Molblock written from MolToMolBlock | % |
|---|---|---|
| x | ✓ | >99.9 |
| | | |
| | | |

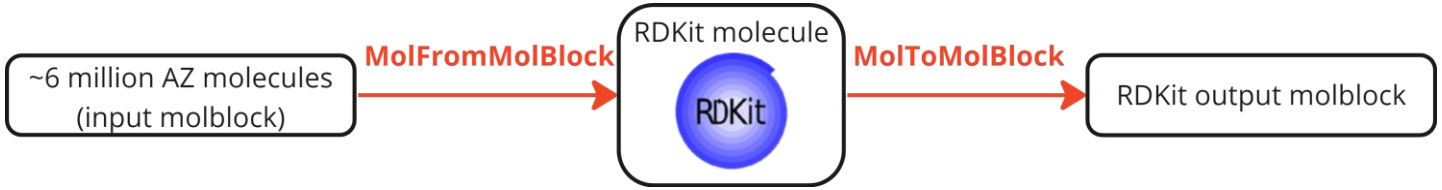



- <0.01% warning/error from reading input but molblock written:
  - Majority (90%) are to do with “Skipping unrecognised collection type… XMDL/SELECTION”
  - 10% mainly due to conflicting stereochemistry warnings.

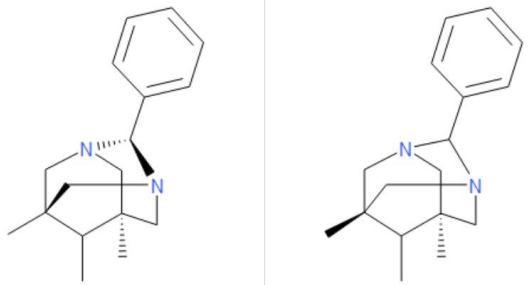
| Warning/error from MolFromMolBlock | Molblock written from MolToMolBlock | % |
|---|---|---|
| x | ✓ | >99.9 |
| ✓ | ✓ | <0.01 |
| | | |



1. Chen, William L., et al. "Self-contained sequence representation: bridging the gap between bioinformatics and cheminformatics." JCIM 51.9 (2011): 2186-2208.



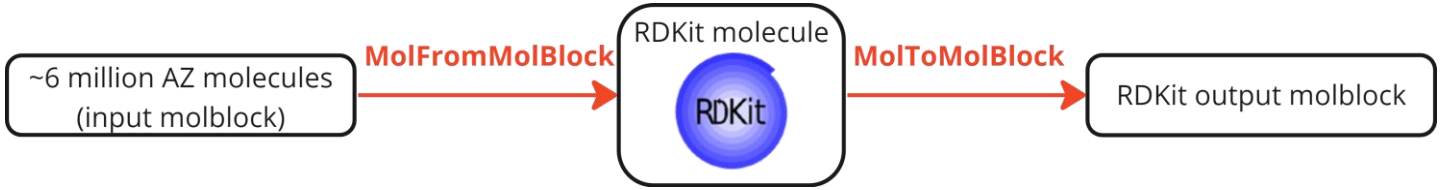

- <0.01% warning/error from reading input but molblock written:
  - Majority (90%) are to do with “Skipping unrecognised collection type… XMDL/SELECTION”
  - 10% mainly due to conflicting stereochemistry warnings.
    - Inspection: RDKit appears to correctly remove unnecessary wedges.



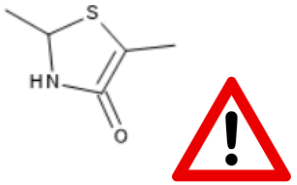

| Warning/error from MolFromMolBlock | Molblock written from MolToMolBlock | % |
|---|---|---|
| x | ✓ | >99.9 |
| ✓ | ✓ | <0.01 |
| | | |



1. Chen, William L., et al. "Self-contained sequence representation: bridging the gap between bioinformatics and cheminformatics." JCIM 51.9 (2011): 2186-2208.




- 0.05% that failed:
  - Majority (85%) have no rdkit error/warning recorded => all look to be SCSR.[1]
  - 15% are explicit valence errors i.e. chemists drawing 5 valent carbons!
  - < 1% are other errors.



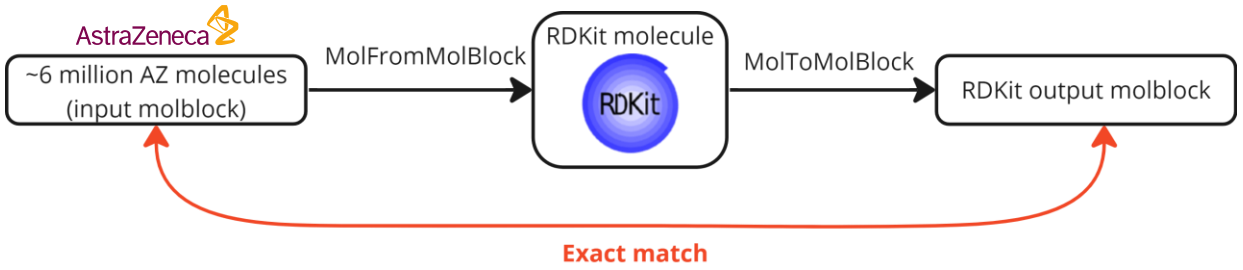

| Warning/error from MolFromMolBlock | Molblock written from MolToMolBlock | % |
|---|---|---|
| x | ✓ | >99.9 |
| ✓ | ✓ | <0.01 |
| x/✓ | x | 0.05 |



1. Chen, William L., et al. "Self-contained sequence representation: bridging the gap between bioinformatics and cheminformatics." JCIM 51.9 (2011): 2186-2208.

# Study 1.2: Import/Export

Are any features lost?



# Exact match results

0.03% (1.8k) did not match by exact match.

- 88% (1.6k) were matched with relaxed stereo match e.g. difference is related to stereo.
- 12% (225) were matched with relaxed tautomer match e.g. difference is related to tautomers.
- 11 were found to be in both categories i.e. the above two are not mutually exclusive.
- No others fall into other categories.

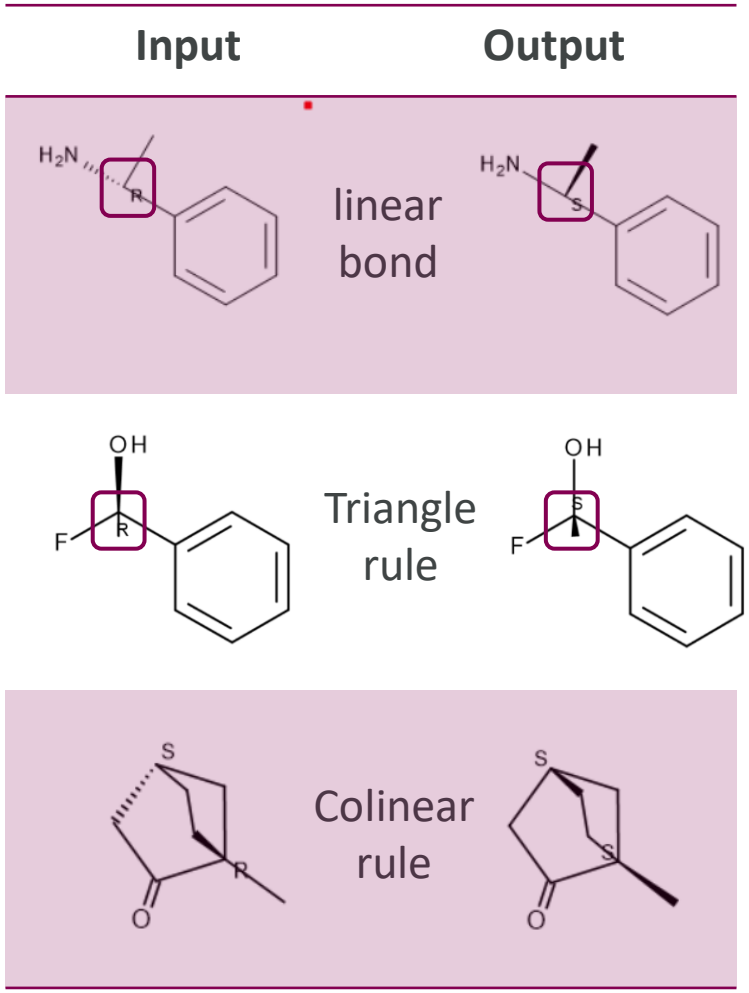
# Exact match results

0.03% (1.8k) did not match by exact match.

- **88% (1.6k) were matched with relaxed stereo match e.g. difference is related to stereo.**
- 12% (225) were matched with relaxed tautomer match e.g. difference is related to tautomers.
- 11 were found to be in both categories i.e. the above two are not mutually exclusive.
- No others fall into other categories.

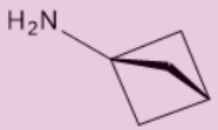
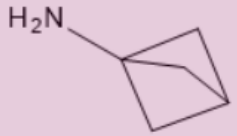
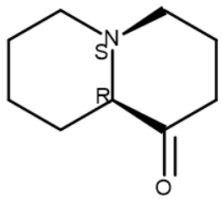
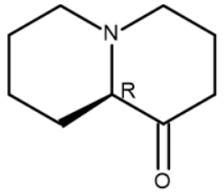
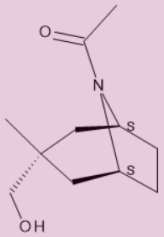
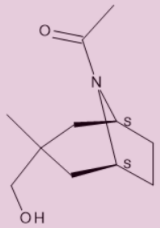
# Removing and canonicalizing wedges can cause issues

- RDKit appears to canonicalize the wedges, sometimes this puts the wedge on a different bond.
  - This is an issue if there is a linear bond. Problem seen also in Fischer projections and cyclic peptides (macrocycles).
  - This might also be an issue if there violation of the triangle rule.
  - This might also be an issue if one of the bonds violates the colinear rule.



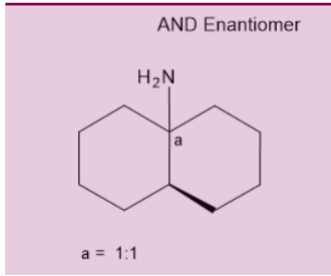
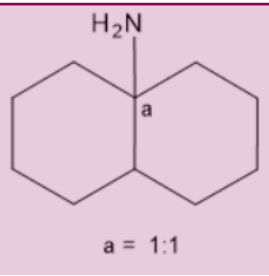
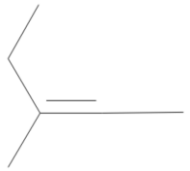
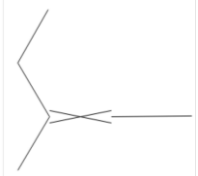
# Removing and canonicalizing wedges can cause issues

- RDKit appears to remove unnecessary wedge bonds – okay in simple cases.
- RDKit removes stereo from N at bridged position in bridged bicycle.
- Cis/trans relationship lost from perspective drawing.

| Input | | Output |
|---|---|---|
|  | Unnecessary wedges |  |
|  | Stereo on N at bridged position |  |
|  | Perspective drawing |  |

# Removing and canonicalizing wedges can cause issues

- Examples of poor/ambiguous stereo drawings? **Our problem?**
- It does not work if there is a label and no wedge on one of the bonds…
- Linear double to denote unknown E/Z geometry – RDKit sanitizes this to unknown double bond geometry.

| Input | Output |
|---|---|
|   |   |
|  |  |

# Exact match results

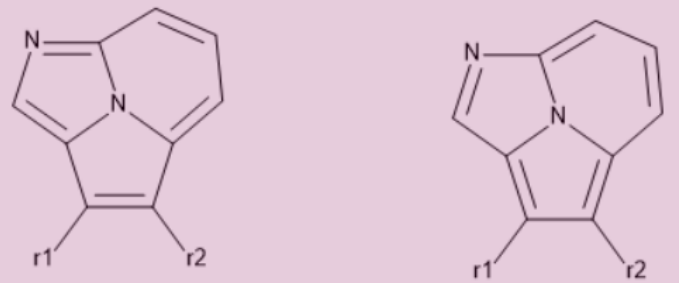
0.03% (1.8k) did not match by exact match.

- 88% (1.6k) were matched with relaxed stereo match e.g. difference is related to stereo.
- **12% (225) were matched with relaxed tautomer match e.g. difference is related to tautomers.**
- 11 were found to be in both categories i.e. the above two are not mutually exclusive.
- No others fall into other categories.

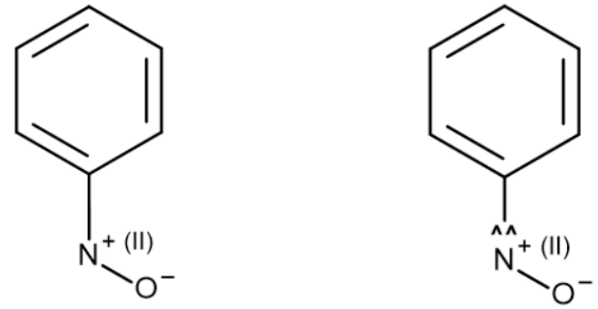
# Relaxed tautomer matching includes others….

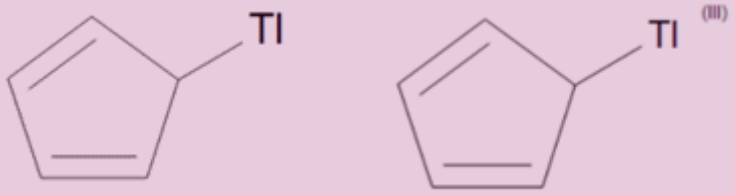
- Relaxing tautomer rules also includes examples where RDKit appears to sanitize and adds radical values to atoms
- Others due to sanitization differences:
  - perchlorate
  - porphyrin rings
  - metals (adds VAL values to atoms)

**Input** **Output**



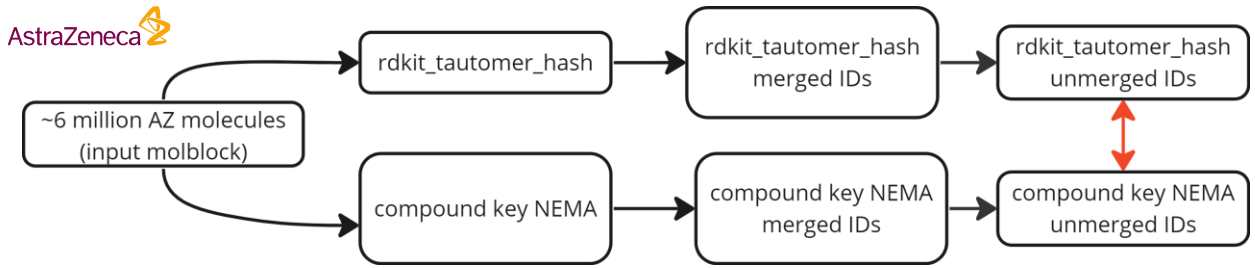

Only occurs with specific r1 and r2



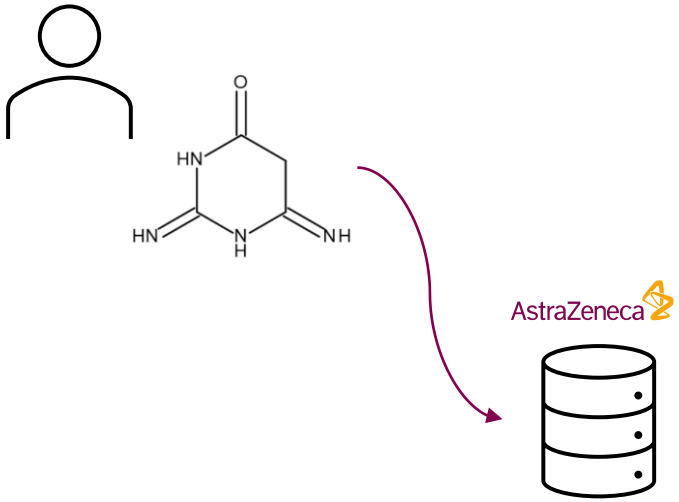



# Study 2: Molecular uniqueness

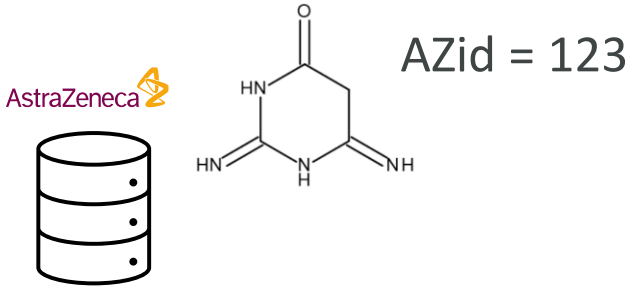
Does RDKit tautomer insensitive hash agree with our system on molecular uniqueness? What are the main differences?



# Background

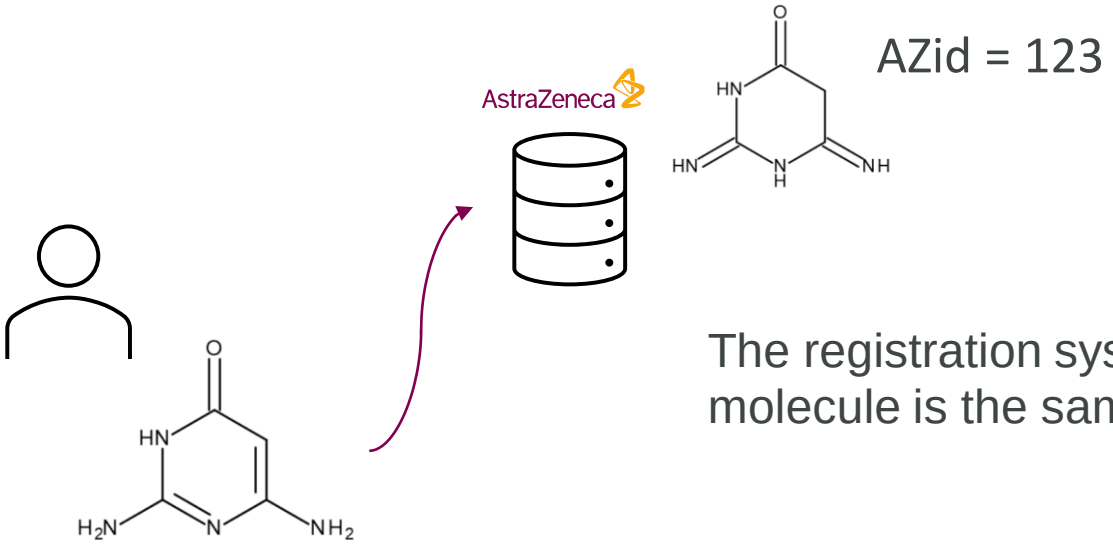


# Background




AZid = 123

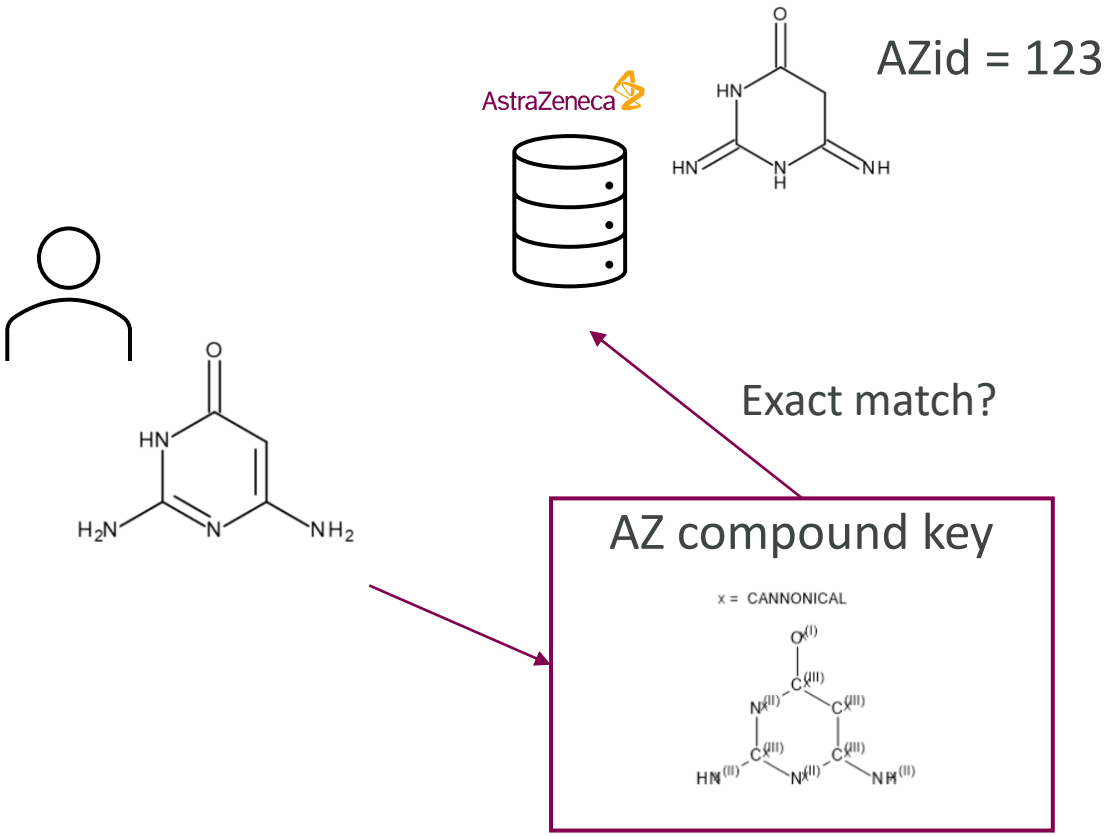
# Background




AZid = 123

The registration system should be able to identify this molecule is the same and not give it a new AZid

# Background




- Our current strategy to determine uniqueness is to generate a **compound key representation**, followed by an exact match against the registration database.
- Alternatively, RDKit's molecular hash can be used.

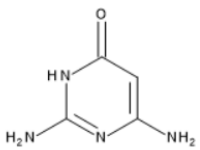
Alternatively…. the RDKit hash?

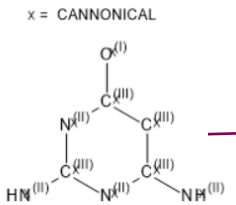
# AZ compound key vs RDKit tautomer insensitive hash

## AZ Compound Key (ckey)

- Tautomer-skeleton-like structure used to identify tautomer relationships

Input AZ ckey ckey NEMA Key






UFNNQHP53KZXHUKFQ3SCTDVE26QCBE

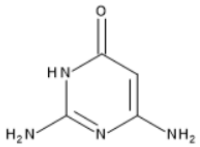
- In our studies we use the NEMA key of the compound key (ckey). This is a surrogate to an exact match against the database.

# AZ compound key vs RDKit tautomer insensitive hash

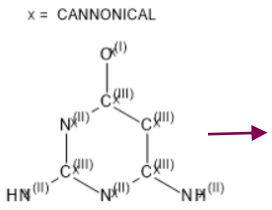
## AZ Compound Key (ckey)

- Tautomer-skeleton-like structure used to identify tautomer relationships

Input



AZ ckey

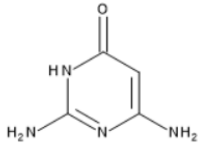



ckey NEMA Key

UFNNQHP53KZXHUKFQ3SCTDVE26QCBE

## RDKit tautomer insensitive hash

Input



| | |
|---|---|
| **HashLayer.CANONICAL_SMILES** | **Nc1cc(=O)[nH]c(N)n1** |
| **HashLayer.ESCAPE** | |
| **HashLayer.FORMULA** | C4H6N4O |
| **HashLayer.NO_STEREO_SMILES** | Nc1cc(=O)[nH]c(N)n1 |
| **HashLayer.NO_STEREO_TAUTOMER_HASH** | [N][C]1[CH][C]([O])[N][C]([N])[N]1_5_0 |
| **HashLayer.SGROUP_DATA** | [] |
| **HashLayer.TAUTOMER_HASH** | [N][C]1[CH][C]([O])[N][C]([N])[N]1_5_0 |
| **RegistrationHash (all layers)** | db02c0e77f0672cc00e067ca69272c5fb04fd2d3 |
| **RegistrationHash (stereo insensitive layers)** | 75103411a7d895949f0e6de3b992d8a89417e912 |
| **RegistrationHash (tautomer insensitive layers)** | 87fdeaca6516b08a5895448eba0e8eaa96420e0a |

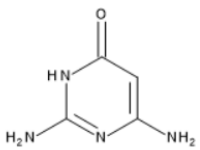
- In our studies we use the NEMA key of the compound key (ckey). This is a surrogate to an exact match against the database.
- To compare against this we use RDKit tautomer insensitive hash, aka RDKit tautomer hash.

# AZ compound key vs RDKit tautomer insensitive hash

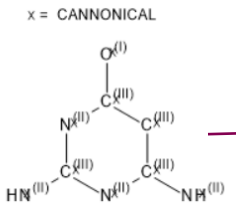
## AZ Compound Key (ckey)

- Tautomer-skeleton-like structure used to identify tautomer relationships

Input



AZ ckey

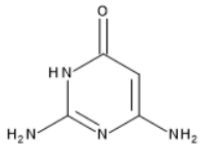



ckey NEMA Key

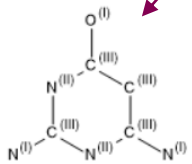
UFNNQHP53KZXHUKFQ3SCTDVE26QCBE

## RDKit tautomer insensitive hash

Input



| | |
|---|---|
| **HashLayer.CANONICAL_SMILES** | **Nc1cc(=O)[nH]c(N)n1** |
| **HashLayer.ESCAPE** | |
| **HashLayer.FORMULA** | C4H6N4O |
| **HashLayer.NO_STEREO_SMILES** | Nc1cc(=O)[nH]c(N)n1 |
| **HashLayer.NO_STEREO_TAUTOMER_HASH** | [N][C]1[CH][C]([O])[N][C]([N])[N]1_5_0 |
| **HashLayer.SGROUP_DATA** | [] |
| **HashLayer.TAUTOMER_HASH** | [N][C]1[CH][C]([O])[N][C]([N])[N]1_5_0 |
| **RegistrationHash (all layers)** | db02c0e77f0672cc00e067ca69272c5fb04fd2d3 |
| **RegistrationHash (stereo insensitive layers)** | 75103411a7d895949f0e6de3b992d8a89417e912 |
| **RegistrationHash (tautomer insensitive layers)** | 87fdeaca6516b08a5895448eba0e8eaa96420e0a |



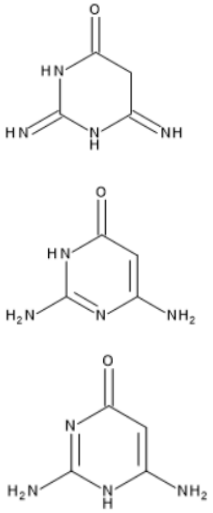
- In our studies we use the NEMA key of the compound key (ckey). This is a surrogate to an exact match against the database.
- To compare against this we use RDKit tautomer insensitive hash, aka RDKit tautomer hash.

# AZ compound key vs RDKit tautomer insensitive hash

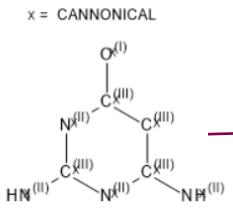
## AZ Compound Key (ckey)

- Tautomer-skeleton-like structure used to identify tautomer relationships

Input



AZ ckey

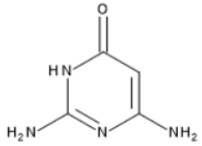



ckey NEMA Key

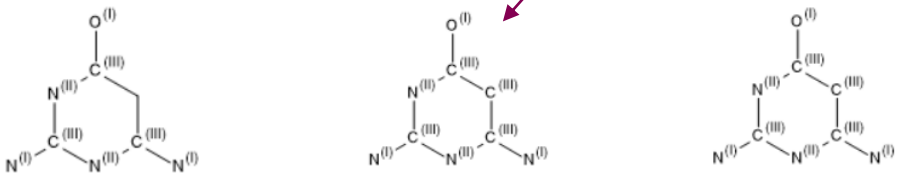
UFNNQHP53KZXHUKFQ3SCTDVE26QCBE

## RDKit tautomer insensitive hash

Input

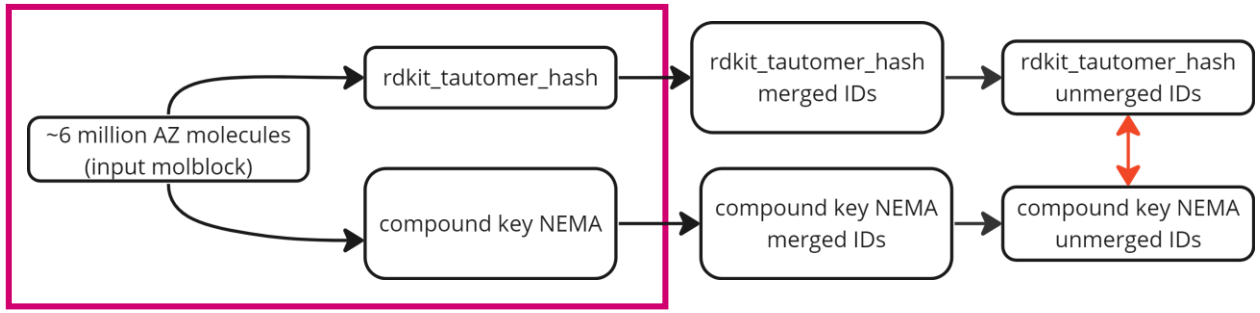


| | |
|---|---|
| **HashLayer.CANONICAL_SMILES** | **Nc1cc(=O)[nH]c(N)n1** |
| **HashLayer.ESCAPE** | |
| **HashLayer.FORMULA** | C4H6N4O |
| **HashLayer.NO_STEREO_SMILES** | Nc1cc(=O)[nH]c(N)n1 |
| **HashLayer.NO_STEREO_TAUTOMER_HASH** | [N][C]1[CH][C]([O])[N][C]([N])[N]1_5_0 |
| **HashLayer.SGROUP_DATA** | [] |
| **HashLayer.TAUTOMER_HASH** | [N][C]1[CH][C]([O])[N][C]([N])[N]1_5_0 |
| **RegistrationHash (all layers)** | db02c0e77f0672cc00e067ca69272c5fb04fd2d3 |
| **RegistrationHash (stereo insensitive layers)** | 75103411a7d895949f0e6de3b992d8a89417e912 |
| **RegistrationHash (tautomer insensitive layers)** | 87fdeaca6516b08a5895448eba0e8eaa96420e0a |



- In our studies we use the NEMA key of the compound key (ckey). This is a surrogate to an exact match against the database.
- To compare against this we use RDKit tautomer insensitive hash, aka RDKit tautomer hash.

# Methods

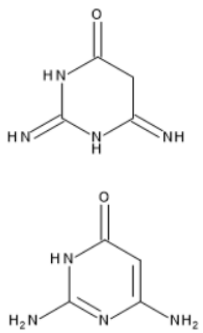


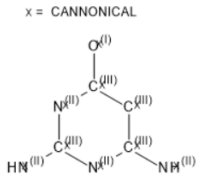

- ~6 million unique mol_ids.
- Apply some standardizations and generate AZ compound key.
  - Some failed sanitizations (1.5k). Some failed compound key generation (1.5k).
  - Some (7.7k) failed NEMA key generation.
  - Exclude those with mixed stereo (113k) i.e. a combination of ABS, AND, OR stereo, as NEMA key does not handle.
- Use RDKit to generate tautomer insensitive hash.
  - Some molecules failed hash generation (131) – all had HELM strings stored in molfile.

# n(ckey_nema_merged_mol_id) > n(tautomer_hash_merged_mol_id)

| Mol_id | NEMA Key | Tautomer hash | ckey_nema merged_mol_id | tautomer_hash merged_mol_id |
|---|---|---|---|---|
| S123 | 7YFUNB2ZXJ1T<br>2DZMWMSZE<br>MZP78WV33 | 87fdeaca6516b08<br>a5895448eba0e8<br>eaa96420e0a | S123<br>S124<br>S125<br>S126<br>S127 | S123<br>S124<br>S125 |
| S126 | 7YFUNB2ZXJ1T<br>2DZMWMSZE<br>MZP78WV33 | 254c4c2010e3bcd<br>f966c276006e82b<br>2201eec8ca | S123<br>S124<br>S125<br>S126<br>S127 | S126<br>S127 |

-> ckey_nema groups more mol_ids together, e.g.



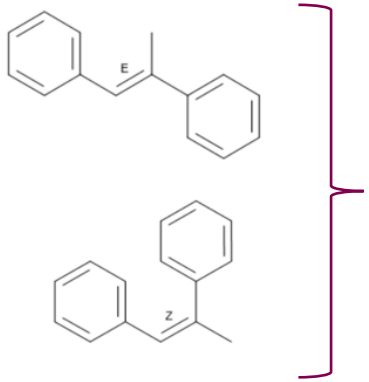


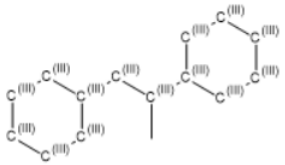

Same ckey NEMA key

# n(ckey_nema_merged_mol_id) < n(tautomer_hash_merged_mol_id)

| Mol_id | NEMA Key | Tautomer hash | ckey_nema merged_mol_id | tautomer_hash merged_mol_id |
|---|---|---|---|---|
| S123 | A5EQKDFCS3KH<br>T9GNEA5WG6<br>WBR4RD8K | d8130ae90143f88<br>25d626fcaf293a5<br>82d0c2966a | S123<br>S124<br>S125 | S123<br>S124<br>S125<br>S126<br>S127 |
| S126 | FPNGG2MCHBH<br>2BGKN5UMDU3<br>E5UM4T3F | d8130ae90143f88<br>25d626fcaf293a5<br>82d0c2966a | S126<br>S127 | S123<br>S124<br>S125<br>S126<br>S127 |

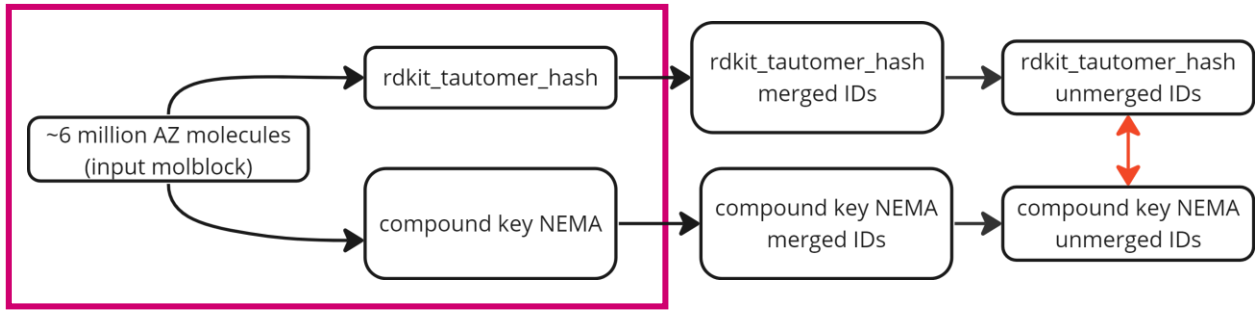
-> tautomer_hash groups more mol_ids together, e.g.





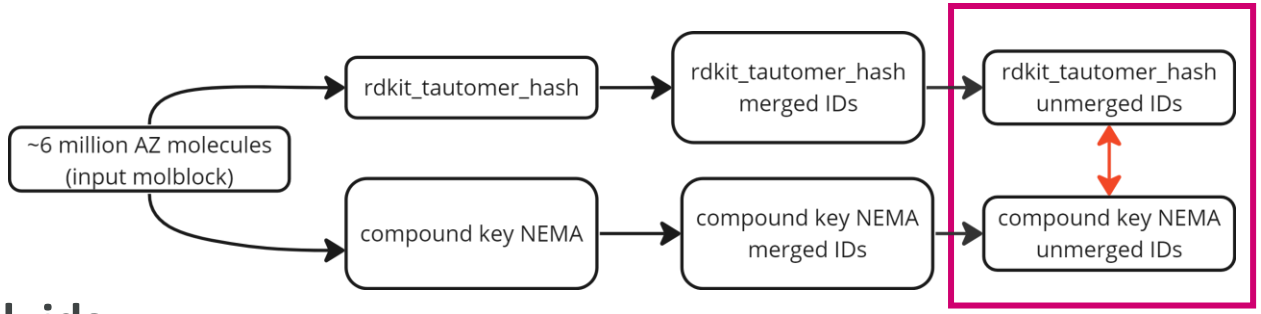
Same RDKit tautomer hash

# Methods




- ~6 million unique mol_ids.
- Apply some standardizations and generate AZ compound key.
  - Some failed sanitizations (1.5k). Some failed compound key generation (~1.5k).
  - Some (7.7k) failed NEMA key generation.
  - Exclude those with mixed stereo (113k) i.e. a combination of ABS, AND, OR stereo, as NEMA key does not handle.
- Use RDKit to generate tautomer insensitive hash.
  - Some molecules failed hash generation (131) – all had HELM strings stored in molfile.
- After all processing, 5,937,998 unique mol_ids left.
  - 3,945,148 unique ckey NEMA
  - 3,932,623 unique RDKit hash
  - 3,932,139 unique AZids

# Results

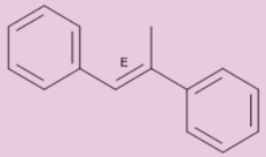
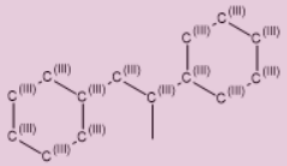
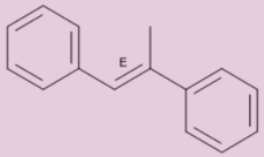
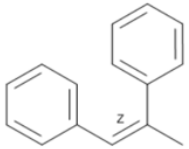
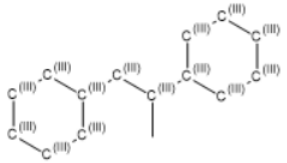
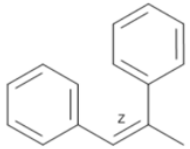



- **70k mol_ids had different merged mol_ids**
  - 28k unique ckey NEMA = **0.70% disagree by ckey NEMA key**.
  - 15k unique rdkit hash = 0.38% disagree by RDKit hash. => fewer unique rdkit hash => rdkit recognises fewer unique parents? Lose of information?
  - 27k unique AZid – 0.70% disagree by AZid (the truth?)

- Breakdown of 70k
  - 60k corrected by clearing cis/trans stereo from NEMA key => due to cis/trans?
    => **91% according to ckey NEMA** , 86% according to RDKit hash
  - 3.7k corrected by using parent NEMA instead of ckey NEMA => related to tautomers?
    => **4% according to ckey NEMA** , 7% according to RDKit hash
  - 5.2k are unaccounted for
    => **5% according to ckey NEMA** , 8% according to RDKit hash

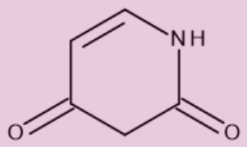
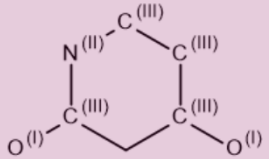
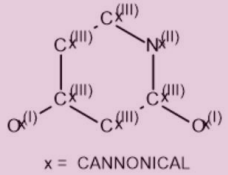
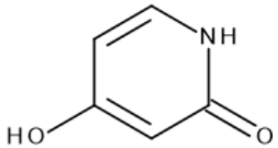
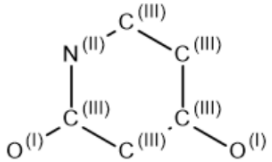
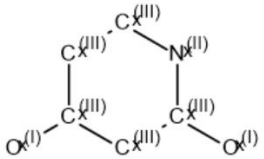
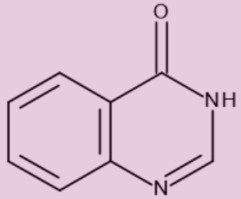
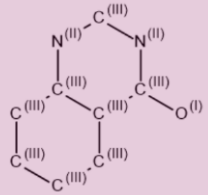
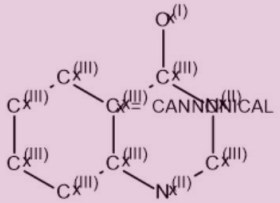
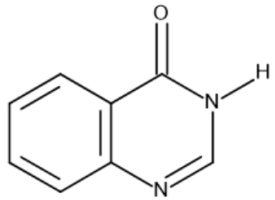
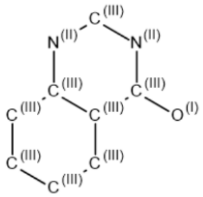
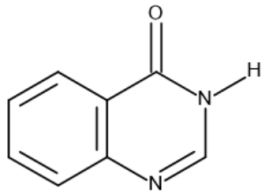
# 60k corrected by clearing cis/trans stereo from NEMA key

- 25k unique ckey nema = 91%
- 12k unique rdkit tautomer hash = 82%
- All 60k have n(ckey_nema_merged_mol_id) < n(tautomer_hash_merged_mol_id)
  - Agrees that these are cases that RDKit cannot distinguish cis/trans isomers.
  - All 100 inspected examples are due to conjugated cis/trans/unknown alkenes/imines
    - E.g. conjugated alkene

| Input | RDKit tautomer skeleton | AZ compound key |
|---|---|---|
|  |  |  |
|  |  |  |

# 3.7k corrected by using parent NEMA instead of ckey NEMA

- 1.0k unique ckey NEMA (4% of differences)
- 1.1k unique tautomer hash (7% of differences)
- 2.5k n(ckey_nema_merged_mol_id) > n(tautomer_hash_merged_mol_id)
- These look like examples of tautomer cases. Where RDKit cannot identify equivalent tautomers.
- 1.2k n(ckey_nema_merged_mol_id) < n(tautomer_hash_merged_mol_id).
  - These look like false negatives. Explicit hydrogens not stripped by our standardization pipeline.

| Input | RDKit tautomer skeleton | AZ compound key |
|---|---|---|
| differences in tautomer identification (RDKit does not recognise enol/keto) | | |
|  |  |  x = CANNONICAL |
|  |  |  x = CANNONICAL |
| Explicit hydrogen interferes with ckey generation – methodology | | |
|  |  |  x = CANNONICAL |
|  |  |  |

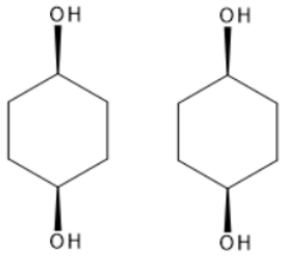
# 5.2k are unaccounted for

- 1.2k unique ckey NEMA = 2%
- 1.3k unique RDKit tautomer hash = 6%
- 3.0k n(ckey_nema_merged_mol_id) > n(tautomer_hash_merged_mol_id), including:
  - Examples of meso compounds that RDKit cannot handle.
  - Examples of ckey identifies tautomer but RDKit does not.
  - RDKit does not take account of sgroup superatoms e.g. two equivalent molecules but one has extra superatom.
- 2.1k n(ckey_nema_merged_mol_id) < n(tautomer_hash_merged_mol_id), including:
  - Examples of RDKit identifies tautomer but ckey does not.
  - Also when double bond stereo is different but part of system that RDKit identifies as part of tautomer skeleton.
  - Explicit hydrogen not stripped properly in ckey.
  - Parent is not charge normalized/neutral.

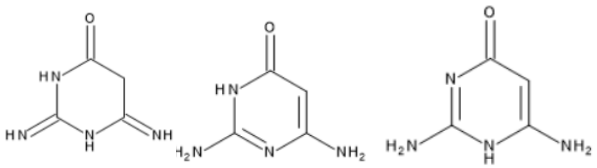
# 5.2k are unaccounted for

- 3.1k n(ckey_nema_merged_mol_id) > n(tautomer_hash_merged_mol_id), examples:

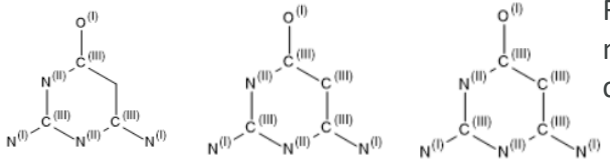
AND Enantiomer




**Meso compound**. Symmetrical but has enhanced stereo so two hashes are produced but only one ckey NEMA
C[C@H]1CC[C@@H]([O])CC1_1_0 |&1:1,4
C[C@H]1CC[C@@H]([O])CC1_1_0 |a:1,4||

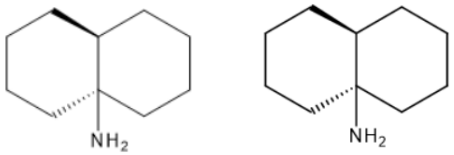


ckey identifies tautomer but RDKit does not



RDKit identifies 2 of the 3 tautomers (so this does not show up in results using parent NEMA instead of ckey NEMA)

AND Enantiomer



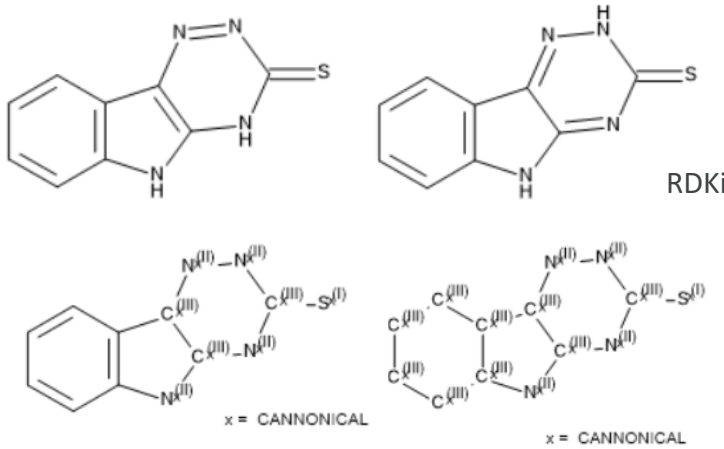

No chiral centre but RDKit gives different hashes according to enhanced stereo in the molfile
[N][C@]12CCCC[C@@H]1CCCC2_2_0 |&1:1,6|
[N][C@]12CCCC[C@@H]1CCCC2_2_0 |a:1,6|

Peptide example not shown

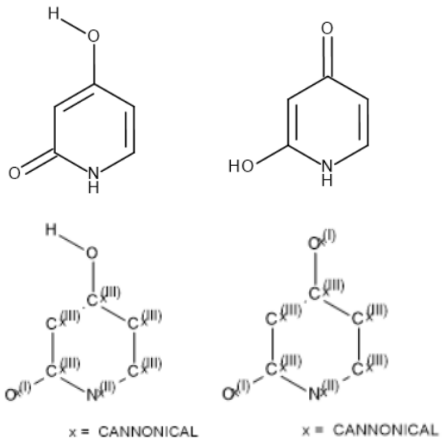
RDKit does not take account of **sgroup superatoms** e.g. two equivalent molecules but one has extra superatom?

# 5.2k are unaccounted for

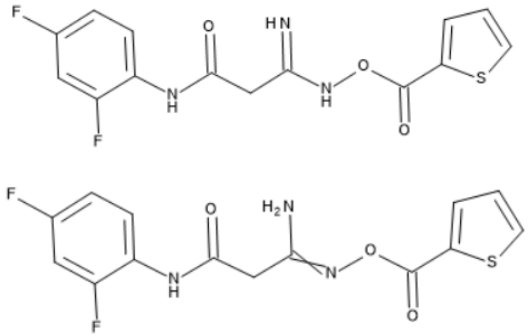
- 2.2k n(ckey_nema_merged_mol_id) < n(tautomer_hash_merged_mol_id), examples:



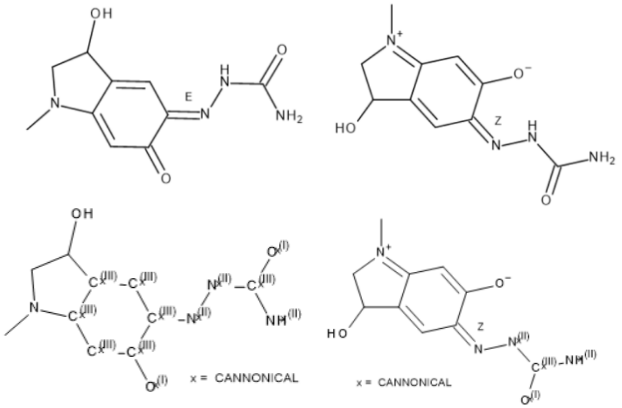

RDKit identifies tautomer but ckey does not.




Explicit hydrogen not stripped properly in ckey.



when double bond stereo is different but part of system that RDKit identifies as part of tautomer skeleton




Parent is not charge normalized/neutral in ckey. Problem with the standardisation methodology.

# Conclusions

# Conclusions

- We have started to assess the implications of using RDKit in our systems and for molecular uniqueness checking.
- Results from import/export:
  - Only 0.05% completely failed to write out molblock.
  - Some issues from canonicalization/removal of wedge bonds.
  - Some differences in standardisation, e.g. salts, metals, radicals, Sulfoxides
  - Some badly drawn structures.
- Results from RDKit hash
  - 0.70% out of total are incorrect by NEMA key.
  - Main sources of differences: RDKit does not differentiate between conjugated cis/trans bonds.
  - Some other differences in tautomer identification, e.g. ketol-enol, requirement for push/pull system.
  - Minority: stereochemistry, explicit hydrogens, sgroup data.
  - May need to consider what standardisations to apply before uniqueness checking.

  Future work:
  - Try it out on some production systems…..

# Acknowledgements


- AZ R&D IT
  - Nick Tomkinson
  - Aleksandr Savelev
  - Arthur Garon
  - Ioana Oprisiu
  - Lars Brive
  - Kevin Pinto Gil
  - Justin Morley
  - Frank Kilty
  - Prakash Rathi
  - Colin Blackmore
- RDKit
  - Greg Landrum
- Schrodinger
  - Christopher Von Bargen

# Thank you.