mmpdb 3.0

Andrew Dalke <dalke@dalkescientific.com>

RDKit User Group Meeting

13 October 2022

Summary

Two molecules form a matched molecular pair (MMP) if they only differ by a single set of connected atoms.

The connected atoms are the "variable" parts. The rest are the "constant".

`variable1>>variable2` defines a transformation.

When chemistry is additive, transformations can be used to predict property changes.

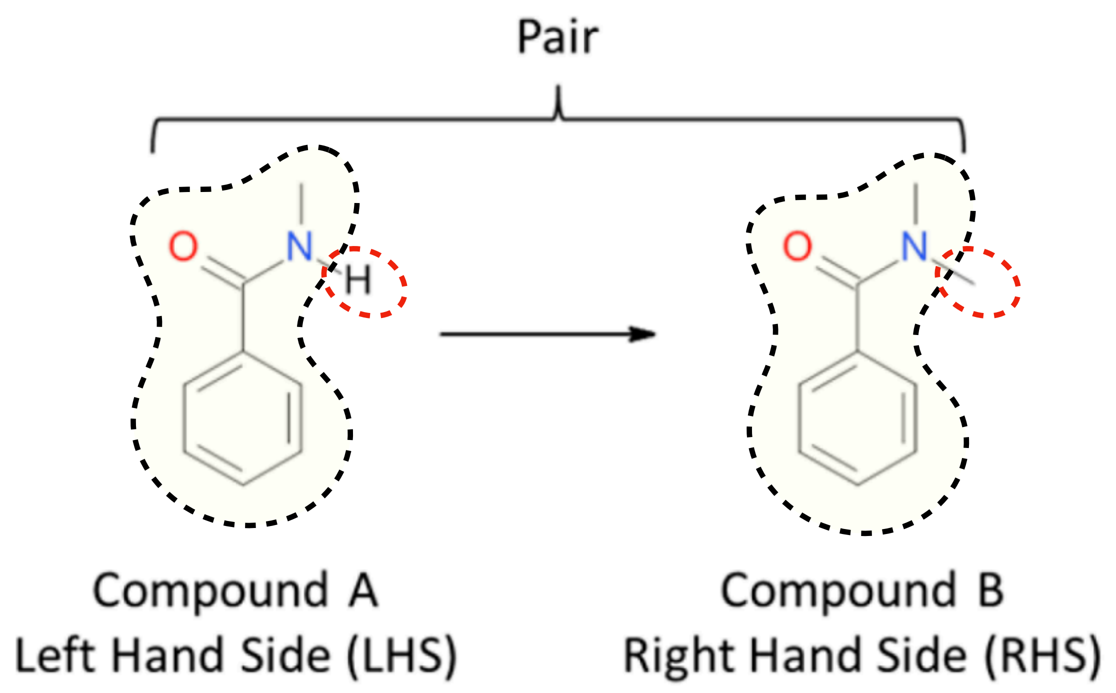
Including the local environment of the constant can help improve linearity.

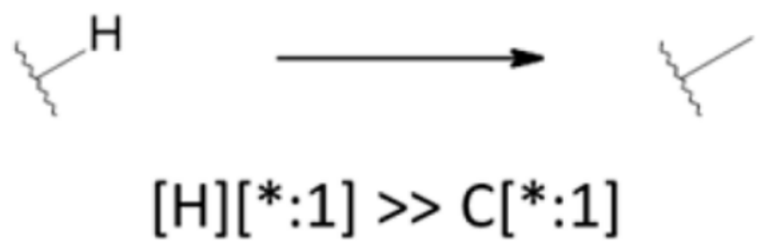
In mmpdb 3.0:

- better scaling for large databases
- can use MMPs to generate new structures based on known chemical space (no AI here!)
- environment information based on Morgan SMARTS

Uses the "fragment-and-index" approach of Hussain and Rea

1-cut pair (mmpdb has special support for a single hydrogen)



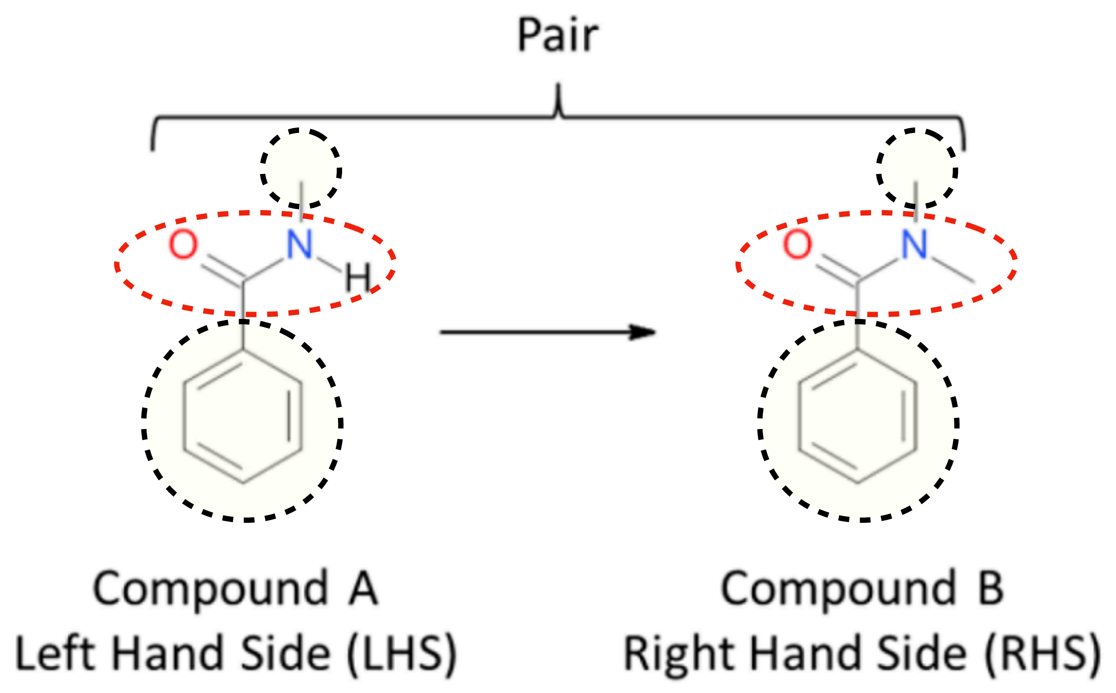


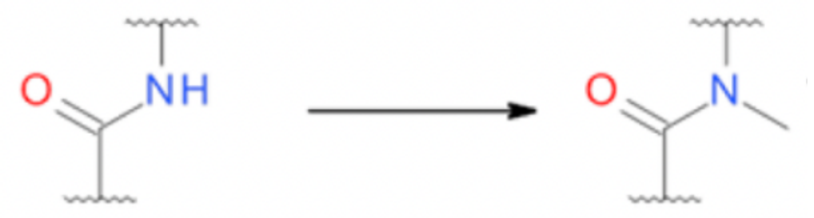
Canonicalize the constant then find matching constants.

	Constant	Variable
Compound A	[*:1]N(C)C(=O)c1ccccc1	[*:1][H]
Compound B	[*:1]N(C)C(=O)c1ccccc1	[*:1]C

Images from "mmpdb: An Open-Source Matched Molecular Pair Platform for Large Multiproperty Data Sets", J. Chem. Inf. Model. 2018, 58, 902–910

2-cut fragmentation





O=C(N[*:2])[*:1] >> CN(C(=O)[*:1])[*:2]

	Constant	Variable
Compound A	C[*:2].c1ccc([*:1])cc1	O=C(N[*:2])[*:1]
Compound B	C[*:2].c1ccc([*:1])cc1	CN(C(=O)[*:1])[*:2]

Two components in the constant.

Images from "mmpdb: An Open-Source Matched Molecular Pair Platform for Large Multiproperty Data Sets", J. Chem. Inf. Model. 2018, 58, 902–910

Complications

- Can't store the attachment points as labeled wildcards ("[*:1]", "[*:2]", "[*:3]") because the value affects canonicalization order.
- Attachment points may be in the same symmetry class.
- Fragmentation can create symmetry causing loss of chiral information. ("F[C@](Cl)(Br)O" -> "F[C@](*)(*)*")
  - Which you may or may not want when indexing.
- The attachment point order in the constants may be different than the attachment point order in the variables.

Took a lot of work to handle these correctly!

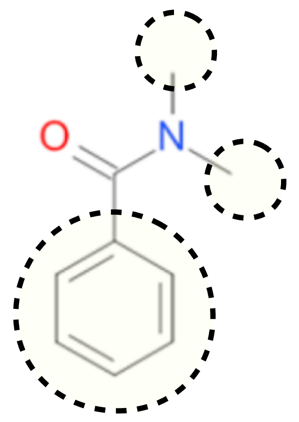
"mmpdb smifrag"

```
% mmpdb smifrag 'CN(C)C(=O)c1ccccc1'
                |----------------  variable  ----------------|         |-------------------------  constant  ------------------------
#cuts | enum.label | #heavies | symm.class | smiles               | order | #heavies | symm.class | smiles                | with-H
------+------------+----------+------------+----------------------+-------+----------+------------+-----------------------+----------------
   1  |      N     |    10    |      1     | *N(C)C(=O)c1ccccc1   |    0  |     1    |      1     | *C                    | C
   1  |      N     |     1    |      1     | *C                   |    0  |    10    |      1     | *N(C)C(=O)c1ccccc1    | CNC(=O)c1ccccc1
   2  |      N     |     9    |     11     | *N(*)C(=O)c1ccccc1   |   01  |     2    |     11     | *C.*C                 | -
→  3  |      N     |     3    |    122     | *C(=O)N(*)*          |  201  |     8    |    112     | *C.*C.*c1ccccc1       | -
   2  |      N     |     4    |     12     | *C(=O)N(*)C          |   10  |     7    |     12     | *C.*c1ccccc1          | -
   1  |      N     |     6    |      1     | *c1ccccc1            |    0  |     5    |      1     | *C(=O)N(C)C           | CN(C)C=O
   1  |      N     |     5    |      1     | *C(=O)N(C)C          |    0  |     6    |      1     | *c1ccccc1             | c1ccccc1
```

Variable

`*C(=O)N(*)*`

Symmetry class "122".
Last two "*" are in the same class.



Constant

`*C.*C.*c1ccccc1`

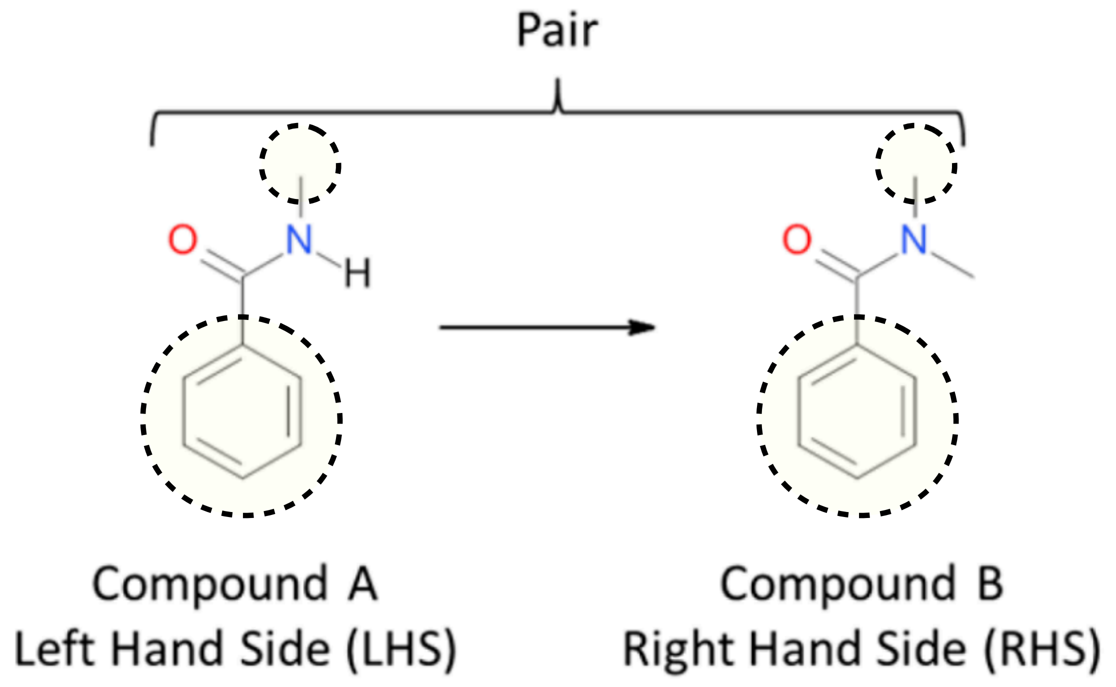
Symmetry class "122".
First two "*" are in the same class.

Order "201" describes how to re-connect the constant and variable.

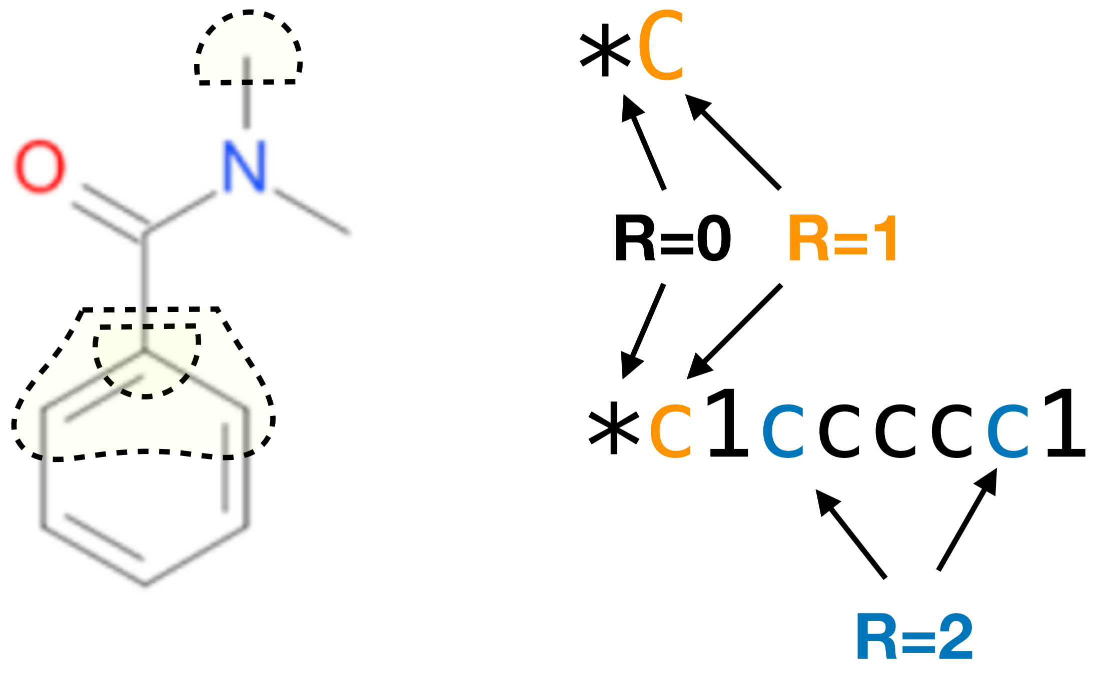
```
Wildcard attachments: [*:2]C(=O)N([*:0])[*:1].[*:0]C.[*:1]C.[*:2]c1ccccc1
         As closures: C%92(=O)N%90%91.C%90.C%91.c%921ccccc1
            "Welded": CN(C)C(=O)c1ccccc1
```

Environment

"MMP rules are highly dependent on the local environment around transformations. A transformation that substitutes a hydrogen atom in a carboxylic acid with a methyl group, for example, results in different molecular property changes than the same substitution in an aliphatic chain."

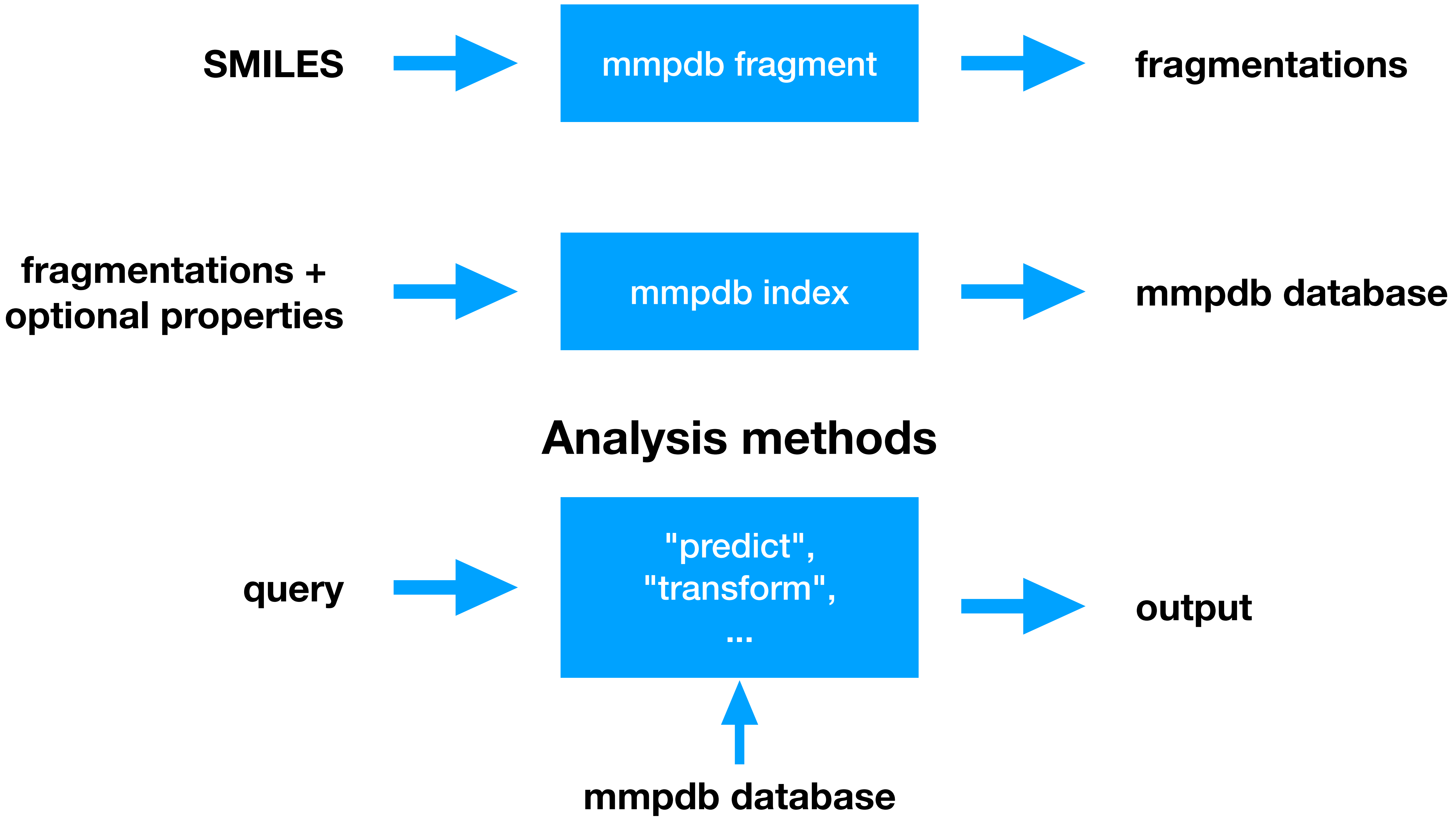


Constant environment with radius 2



From "mmpdb: An Open-Source Matched Molecular Pair Platform for Large Multiproperty Data Sets", J. Chem. Inf. Model. 2018, 58, 902−910

Basic mmpdb Data Flow



"predict"

Predict the effect of substituting a sulfur in diphenyl ether if the known melting point is 12C.

```
% mmpdb predict csd.mmpdb --smiles c1ccccc1Sc1ccccc1 \
      --reference c1ccccc1Oc1ccccc1 --property MP --value 12.0
predicted delta: +4.91667 predicted value: 16.9167 +/- 18.0477
```

Can also save details about the transformations and associated pairs to files.

"transform"

Generate all of the products of diphenyl ether using the MMP transforms where there are at least 30 pairs. Also include the predicted effects on the 'MP' property.

```
% mmpdb transform csd.mmpdb --smiles 'c1ccccc1Oc1ccccc1' --min-pairs 30 -p MP
   ID                 SMILES MP_from_smiles       MP_to_smiles  MP_radius  \
    1  Brc1ccc(Oc2ccccc2)cc1  [*:1]c1ccccc1  [*:1]c1ccc(Br)cc1          0
    2  COc1ccc(Oc2ccccc2)cc1  [*:1]c1ccccc1  [*:1]c1ccc(OC)cc1          0
    3            COc1ccccc1  [*:1]c1ccccc1             [*:1]C          0

                             MP_fingerprint  MP_rule_environment_id  \
   59SlQURkWt98BOD1VlKTGRkiqFDbG6JVkeTJ3ex3bOA                     947
   59SlQURkWt98BOD1VlKTGRkiqFDbG6JVkeTJ3ex3bOA                    4560
   59SlQURkWt98BOD1VlKTGRkiqFDbG6JVkeTJ3ex3bOA                      90

   MP_count   MP_avg  MP_std  MP_kurtosis  MP_skewness  MP_min  MP_q1  \
         34  14.5290  30.990    -0.267780      0.32663     -66   -7.0
         56   8.7143  38.945     7.013600      1.81870    -172  -10.0
        106 -23.4430  36.987     1.563800      0.65077    -159  -44.0

   MP_median  MP_q3  MP_max  MP_paired_t    MP_p_value
        15.5   37.0      67      -2.7338  9.987200e-03
        10.5   32.5      79      -1.6745  9.971500e-02
       -20.0   -3.0      49       6.5256  2.447100e-09
```

More on this fingerprint coming up.

mmpdb 3.0

fragmentation format

Previously the fragmentations were in JSON-Lines format.

- Large text file (compresses well).
- Needed 3rd party JSON parser for fast parsing.
- Hard to re-use the fragmentations for other purposes.

Switched to SQLite

```
% sqlite3 ChEMBL_CYP3A4_hERG.fragdb
SQLite version 3.38.5 2022-05-06 15:25:27
Enter ".help" for usage hints.
sqlite> .mode line
sqlite> SELECT id FROM record WHERE normalized_smiles = 'COc1ccc(C)cc1N';
   id = 12302
sqlite> SELECT attachment_order, variable_smiles, constant_smiles
   ...>   FROM fragmentation
   ...>   WHERE record_id = 12302 AND num_cuts = 3;
attachment_order = 012
 variable_smiles = *Oc1ccc(*)cc1*
 constant_smiles = *C.*C.*N

attachment_order = 021
 variable_smiles = *c1ccc(*)c(*)c1
 constant_smiles = *C.*N.*OC
```

With a bit of SQL, can generate a Free-Wilson table. (Still tricky to handle symmetry and chirality. And hydrogens.)

fragdb_list

Summarize the contents of a fragdb file.

```
% mmpdb fragdb_list ChEMBL_CYP3A4_hERG.*.fragdb
             Name              #recs #errs #frags #consts #vars  max.#pairs
ChEMBL_CYP3A4_hERG.0000.fragdb  1929    97 109087   48895  66808   13267833
ChEMBL_CYP3A4_hERG.0001.fragdb  1780   246 212777   70915 146526  116930089
ChEMBL_CYP3A4_hERG.0002.fragdb  1999    27 100594   52370  57399    9958159
ChEMBL_CYP3A4_hERG.0003.fragdb  1690   336 103350   49021  67210    8544224
ChEMBL_CYP3A4_hERG.0004.fragdb  1919   107 112930   52318  68386    8580341
ChEMBL_CYP3A4_hERG.0005.fragdb  1783   243 123463   55977  72733   15043504
ChEMBL_CYP3A4_hERG.0006.fragdb  1869   157 164259   64083 106427   52277090
ChEMBL_CYP3A4_hERG.0007.fragdb  1862   164 114113   51605  61439   15083149
ChEMBL_CYP3A4_hERG.0008.fragdb  1989    37  80613   44149  41764    4614913
ChEMBL_CYP3A4_hERG.0009.fragdb  1939    87  69029   35889  36577    2919084
```

fragdb_constants

Show counts for all (or selected) constants.

```
% mmpdb fragdb_constants test.fragdb --min-heavies-per-const-frag 5 --limit 10
constant	N
*c1ccccc1	304
*N1CCC[C@H]1C	90
*c1ccccc1.*c1ccccc1	90
*CCN1CCC[C@H]1C	86
*Cc1ccccc1	85
*c1ccc2cc(CCN3CCC[C@H]3C)ccc2n1	67
*c1cccnc1	53
*c1ccc(Br)cc1	44
*c1ccc([N+](=O)[O-])cc1	40
*c1ccncc1	36
```

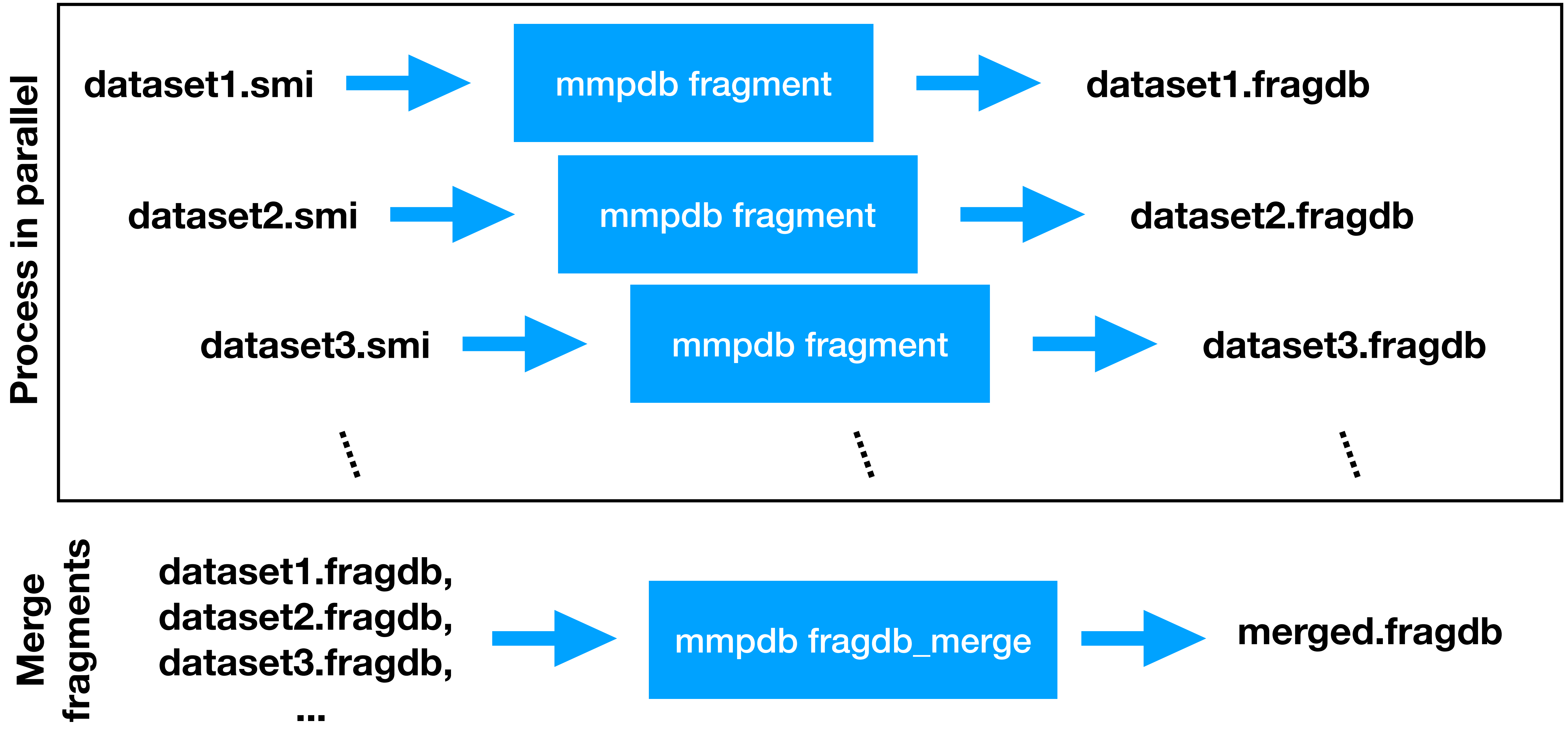
"From this analysis, we created a list of constant components that occurred 1000 times or more in either the RocheDB or SureChEMBL databases and discarded them from further consideration. This step is important to discard spurious MMP's that otherwise exponentially increase not only the transformation database size but also the computation power and memory requirement."

**** The Playbooks of Medicinal Chemistry Design Moves***
Mahendra Awale, Jérôme Hert, Laura Guasch, Sereina Riniker, and Christian Kramer
https://pubs.acs.org/doi/10.1021/acs.jcim.0c01143

Merge fragmentations

Can be used to fragment in parallel.

Only use for single-threaded indexing, or if you want a merged fragment db.



Parallel indexing

Approach developed by Mahendra Awale while at Roche.*

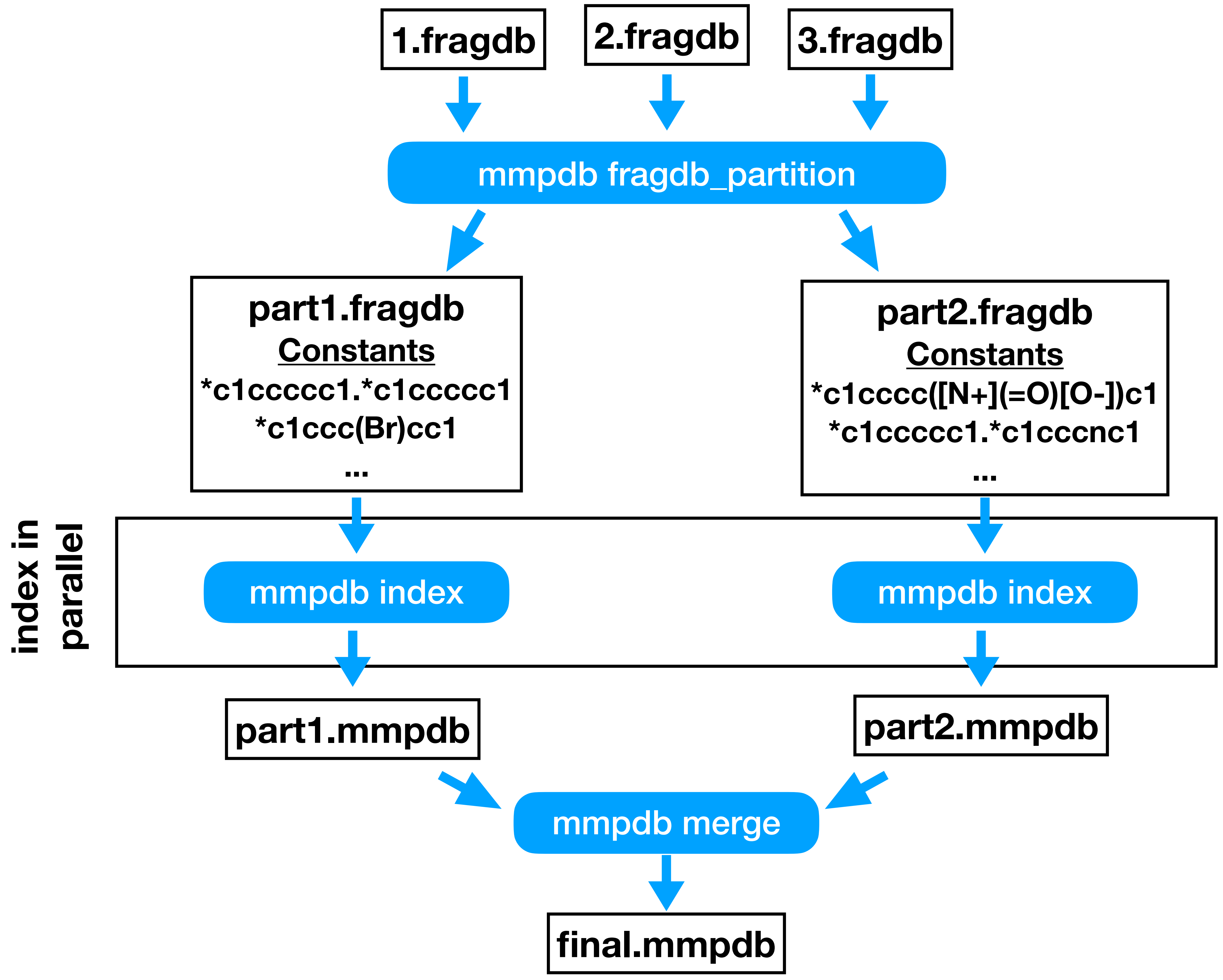
Fragment database merging is single-threaded.
Indexing is also single-threaded ...
... *for a given constant!*

Partition the fragmentations by constant to generate new fragment databases.

- **Indexing cost is ~$O(N^2)$ in the constant's number of fragmentations.**
- **Round-robin to *N* output files, or use max cost per file.**

**** The Playbooks of Medicinal Chemistry Design Moves***
Mahendra Awale, Jérôme Hert, Laura Guasch, Sereina Riniker, and Christian Kramer
https://pubs.acs.org/doi/10.1021/acs.jcim.0c01143

Parallel data flow



Attach SQLite databases

"merge" is still single-threaded.
Needs to normalize a lot of tables and fields.

Current system uses SQLite's ability to "attach" a database.
All connected through an SQLite :memory: database.

Store mappings in an in-memory database.

```
working_db = sqlite3.connect(":memory:")
schema.create_schema_for_sqlite(working_db)
c = working_db.cursor()
c.execute("ATTACH DATABASE ? AS new", (output_filename,))
...
process_compound_tables(c, filenames, reporter)
process_rule_smiles_tables(c, filenames, reporter)
process_rule_tables(c, filenames, reporter)
process_environment_fingerprint_tables(c, filenames, reporter)
process_rule_environment_tables(c, filenames, reporter)
process_pair_tables(c, filenames, reporter)
...
```

Attach the output database.

Attach then process the input database.

Each "`process_*()`" uses:

```
c.execute("ATTACH DATABASE ? AS old", (filename,))
 ...
c.execute("DETACH DATABASE old")
```

Example merge SQL

```
def process_pair_table(c, db_id):
    c.execute("""
INSERT INTO new.pair (rule_environment_id, compound1_id, compound2_id, constant_id)
 SELECT rule_environment_map.new_rule_environment_id,
        compound1_map.new_compound_id,
        compound2_map.new_compound_id,
        new_constant_smiles.id

   FROM old.pair AS old_pair,
        rule_environment_map_{db_id} AS rule_environment_map,
        compound_map_{db_id} AS compound1_map,
        compound_map_{db_id} AS compound2_map,
        old.constant_smiles as old_constant_smiles,
        constant_smiles as new_constant_smiles
  WHERE old_pair.rule_environment_id = rule_environment_map.old_rule_environment_id
    AND old_pair.compound1_id = compound1_map.old_compound_id
    AND old_pair.compound2_id = compound2_map.old_compound_id
    AND old_pair.constant_id = old_constant_smiles.id
    AND old_constant_smiles.smiles = new_constant_smiles.smiles
    ;
    """.format(db_id=db_id))
```

**Advice: Don't treat SQLite as simple table storage.
Do more data processing in SQL instead of Python.**

Timing info

ChEMBL 25 @ ~1.6M compounds

smi_split: ~30s (into 250 files)
fragment: ~2300s+/-800s (per task, 250 files in parallel)
frag_constants + frag_partition dry run: 125s
frag_partition: ~88s+/-7s (per task, 31 files in parallel)
index: ~209s+/-275s (per task, 31 files in parallel;
last 2 tasks took 800s and 1500s)
merge: ~2000s
resulting file: 25GB

~1 CPU week (serial)

~ 2 hours (parallel)

SureChEMBL (Oct. 2019) @ ~13.5M compounds

smi_split: 140s
fragment: ~7000s+/-1700s (per task, 250 files in parallel)
frag_constants + frag_partition dry run: 1124s
frag_partition: ~720s+/-220s (per task, 56 files in parallel)
index: ~1377s+/-1952s (per task, 56 files in parallel;
last 2 tasks took 9200s and 11500s)
merge: ~24 hours (requires a lot of memory! 512GB node)
resulting file: ~200GB

~3 CPU week (serial)

~ 30 hours (parallel)

New "generate" method

The MMP database can be seen as a medchem "playbook".*
Convert implicit database information into explicit design knowledge.
Apply those rules to a structure to generate new structures.

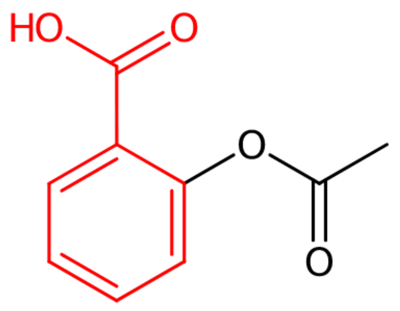
Similar to "transform" but with more control over which part to substitute.
Only works for 1-cut fragmentations.

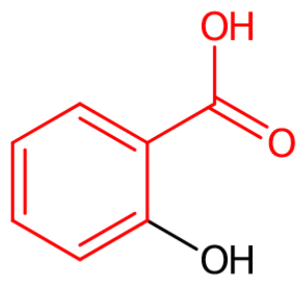
Specify only a SMILES	Specify constant and query	Specify SMILES and constant or query
Generate all transforms using all fragmentations.	Generate transforms only for that query.	mmpdb will figure out the other component, then generate transforms for that query.

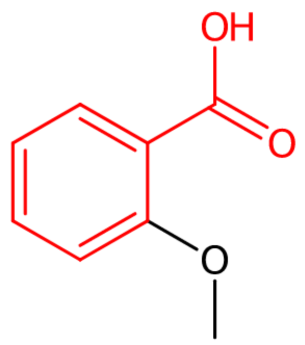
** The Playbooks of Medicinal Chemistry Design Moves*
Mahendra Awale, Jérôme Hert, Laura Guasch, Sereina Riniker, and Christian Kramer
https://pubs.acs.org/doi/10.1021/acs.jcim.0c01143

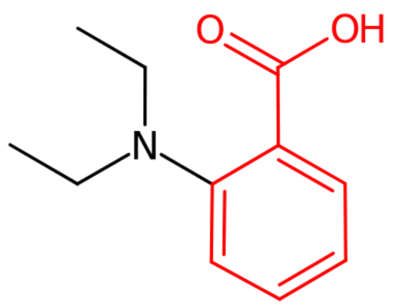
Generate from aspirin

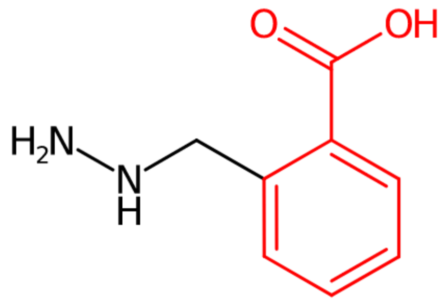
```
% mmpdb generate --smiles 'O=C(C)Oc1ccccc1C(=O)O' --query '*OC(C)=O'\
     --radius 1 merged.mmpdb --columns #pairs,final
#pairs  final
4   O=C(O)c1ccccc1O
1   COc1ccccc1C(=O)O
1   CCN(CC)c1ccccc1C(=O)O
1   NNCc1ccccc1C(=O)O
1   O=C(O)c1ccccc1Nc1ccccc1
1   O=C(Oc1ccccc1C(=O)O)c1cccs1
```

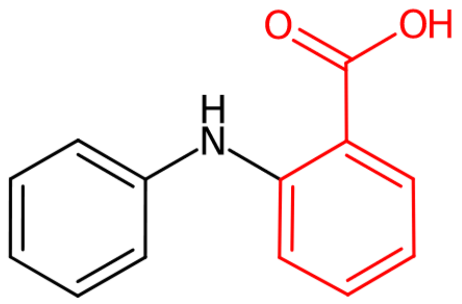

aspirin

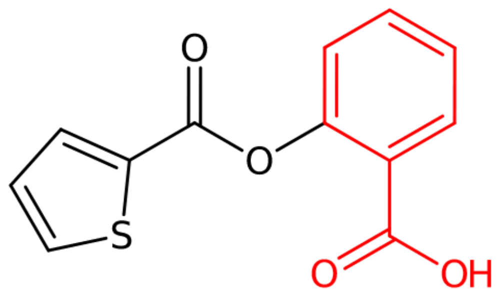

#1


#2


#3


#4


#5


#6

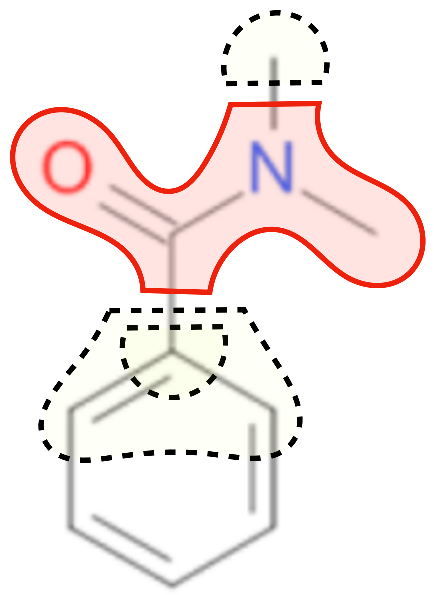
Additionally, can use `--subqueries` to generate subfragments of the query fragments and use those as additional query fragments.

Depictions from https://www.simolecule.com/cdkdepict/depict.html - Built with the Chemistry Development Kit. Depict v1.9.3, CDK v2.8.

Environment Fingerprint

Require a transform rule like [*:2]NC([*:1])=O>>[*:1]C(=O)N([*:2])C to also match some of the circular environment around the attachment points.

Constant environment with radius 2



<u>Old method</u>

1) **Compute Morgan fingerprints centered at each attachment point.**
2) **hashlib.sha256(fp.ToBinary()).digest()**
3) **Concatenate the sha256s.**
4) **SHA256 the concatenation.**
5) **Base64-encode the result.**

59SlQURkWt98BOD1VlKTGRkiqFDbG6JVkeTJ3ex3bOA

Only useful for identify match. Fixed-width size. Uninterpretable.

Morgan Atom SMARTS

The Morgan atom connectivity invariants are:

```
components.push_back(atom->getAtomicNum());
components.push_back(atom->getTotalDegree());
components.push_back(atom->getTotalNumHs());
components.push_back(atom->getFormalCharge());
```

.. more about mass, but mmpdb removes isotopes on input ...

```
if (includeRingMembership &&
    atom->getOwningMol().getRingInfo()
           ->numAtomRings(atom->getIdx())) {
 components.push_back(1);
}
```

These can be represented directly in SMARTS:

```
[#7,X3,H2,+1,R]
```

Morgan Bond SMARTS

The Morgan bond connectivity invariant is:

```
bondInvariant = static_cast<int32_t>(bond->getBondType());
```

The mapping to SMARTS is:

```
_bond_smarts_symbols = {
    Chem.BondType.SINGLE: "-",
    Chem.BondType.DOUBLE: "=",
    Chem.BondType.TRIPLE: "#",
    Chem.BondType.AROMATIC: "~",
   }

...   _bond_smarts_symbols[bond.GetBondType()] ...
```

Fragment SMARTS

MolFragmentToSmiles lets you specify the atom and bond symbols. Use them to generate a SMARTS.

```
smarts = Chem.MolFragmentToSmiles(
        mol,
        atomsToUse = list(atom_ids),
        bondsToUse = list(bond_ids),
        atomSymbols = atom_symbols,
        bondSymbols = bond_symbols,
        isomericSmiles = False,
        )
```

Can't specify the atom&bond invariants.

The output will likely not be canonical ...
... unless the atom and bond symbols match the internal invariants!

Disable isomeric SMILES because Morgan fingerprints don't use them.

Canonical SMARTS for a Rooted Morgan Fingerprint

1) **Get the atom and bond symbols.**
2) **Set aromatic atoms to aliphatic.**
   (To match the Morgan invariants)
3) **Chem.MolFragmentToSmiles()**
4) **Reorder SMILES components;**
   First [*:1] then [*:2] and [*:3]
5) **Restore aromatic atoms.**

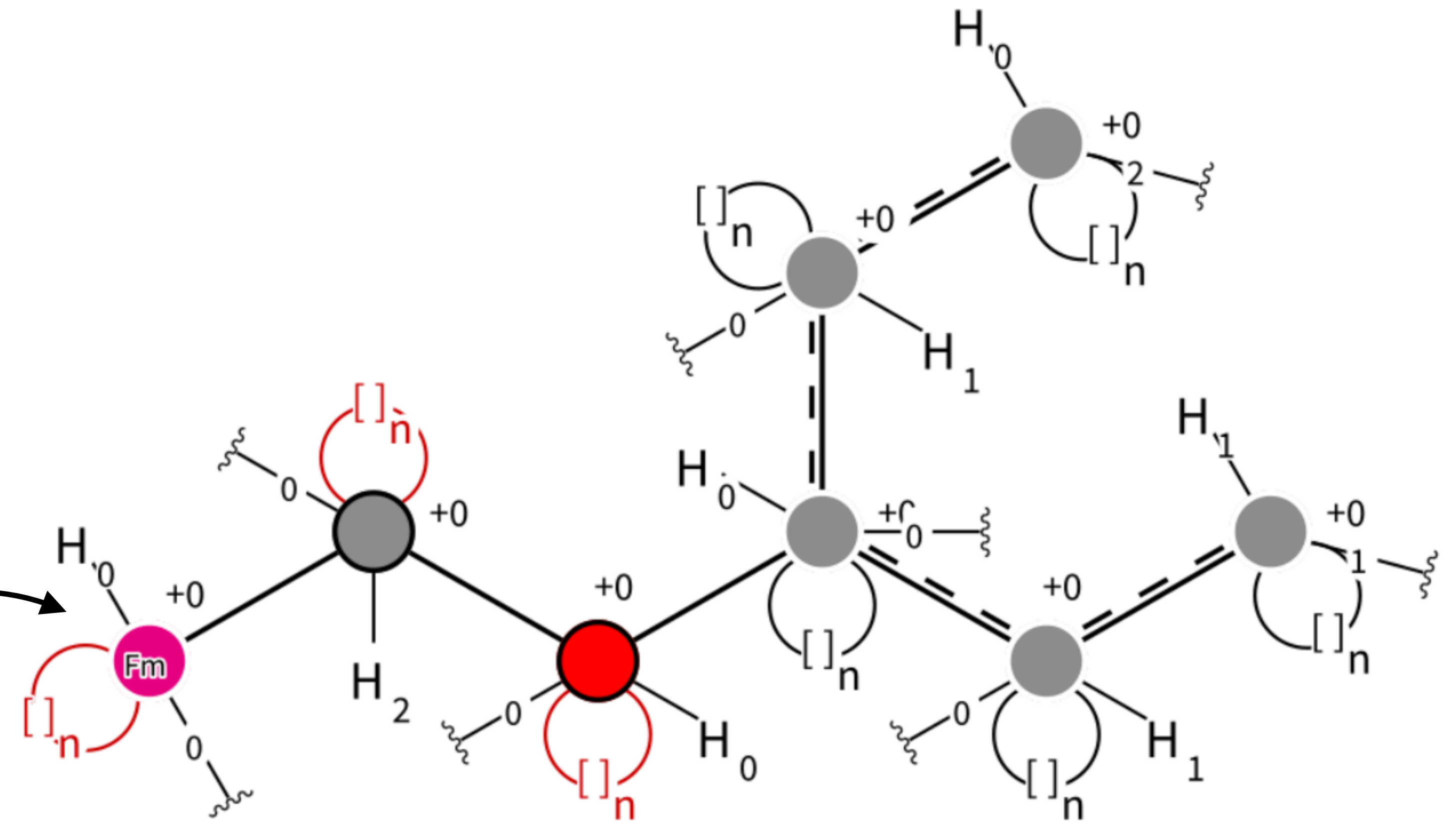
Only works because the SMARTS is rooted at the attachment point.

Morgan SMARTS

Can be quite large!

(VARCHAR(1024) instead of 32)

[#0;X1;H0;+0;!R:1]-[C;X4;H2;+0;!R]-[O;X2;H0;+0;!R]-[#6;X3;H0;+0;R](:[#6;X3;H1;+0;R]:[#6;X3;H0;+0;R]):[#6;X3;H1;+0;R]:[#6;X3;H1;+0;R]



SMARTSViewer doesn't accept #0 so I used #100

Picture created by the SMARTSviewer [https://smarts.plus/].
Copyright: ZBH - Center for Bioinformatics Hamburg.

pseudo-SMILES

Most people can't easily read SMARTS.
Few tools can visualize a complex SMARTS.

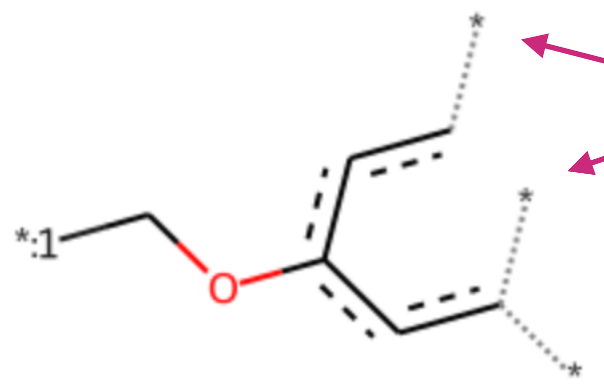
Convert the SMARTS into a SMILES
that RDKit will parse with sanitize=False.

```
[*:1]-[CH2]-[O]-[c](:[cH]:[c](~*)(~*)):[cH]:[cH](~*)
```

[O;X2;H0;+0;!R]

:[#6;X3;H0;+0;R]

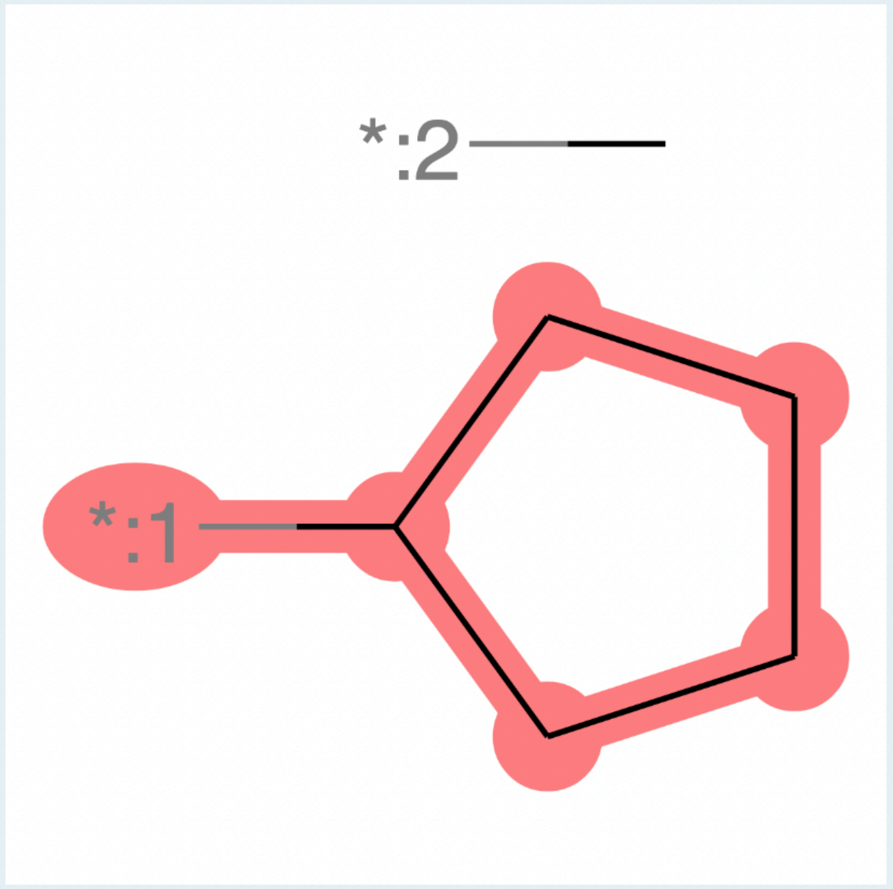
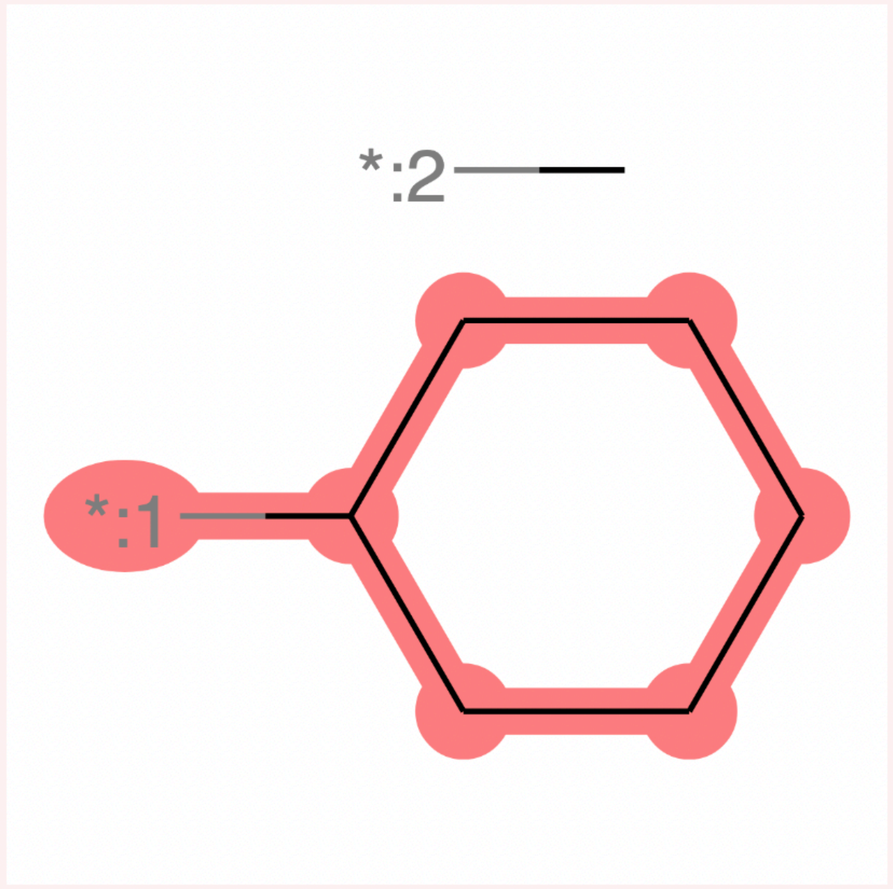
Needed for the X3.



Watch out! Two distinct "*" atoms in the pseudoSMILES *might* refer to the same atom in the original molecule!

Cross-checking

***Almost* perfect one-to-one mapping from SHA256-hashed fingerprint to Morgan SMARTS. Can have multiple SMARTS for the same SHA256 fp.**

fingerprint Tcu44H+ojDtSVcNYf9pQmSOIAOtxfmCKHzfQZGWEzAs	
env #1 (r = 4)	**env #2 (r = 4)**
context: C1CCC([*:1])C1.C[*:2]	**context:** C1CCC([*:1])CC1.C[*:2]
smarts: [#6&X4&H2&+0&R]1-[#6&X4&H2&+0&R]-[#6&X4&H2&+0&R]-[#6&X4&H1&+0&R](-[#6&X4&H2&+0&R]-1)-[#0&X1&H0&+0&!R:1]	**smarts:** [#6&X4&H2&+0&R]1-[#6&X4&H2&+0&R]-[#6&X4&H2&+0&R]-[#6&X4&H1&+0&R](-[#6&X4&H2&+0&R]-[#6&X4&H2&+0&R]-1)-[#0&X1&H0&+0&!R:1]
	

Other changes

Added support for Postgres for direct index database creation.
Does not handle the new parallelized indexing.
(Use one of the SQLite→Postgres converters instead?)

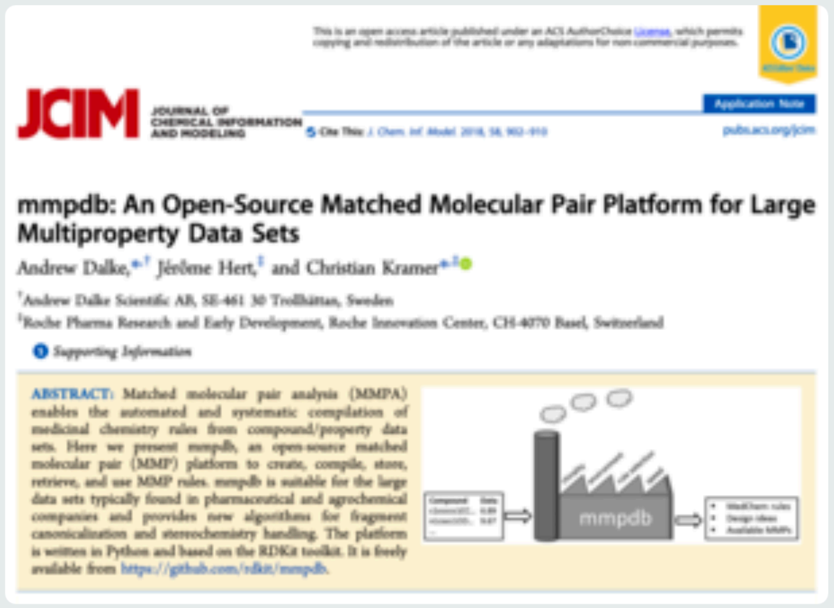
Also added a "csvd" output.
Save tables to a directory containing CSV files.

'mmpdb proprulecat' command to export the property rules in the database (transformation + statistics) in CSV form

Moved from argparse to click.

mmpdb crowdfunding project

How can we **raise money to fund open source software development in cheminformatics**? It’s a hard question. **Simple donations don’t work** – companies might not even have a mechanism to make donations. **Consultant-based funding doesn’t work** that well either, because the cost of developing a general-purpose tool is two or three times more expensive than developing a tool which only meets the specialized needs of one client, and few clients are willing to subsidize the rest of the field. Proprietary software development solves the problem by getting many people to pay for the same product. **Can we learn from the success of proprietary software** to get the funds which would certainly be useful in improving open source software?


This is an open access article published under an ACS AuthorChoice License, which permits copying and redistribution of the article or any adaptations for non-commercial purposes.

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

Application Note

Cite This: J. Chem. Inf. Model. 2018, 58, 902–910

pubs.acs.org/jcim

mmpdb: An Open-Source Matched Molecular Pair Platform for Large Multiproperty Data Sets

Andrew Dalke,*[†] Jérôme Hert,[‡] and Christian Kramer*[‡]

[†]Andrew Dalke Scientific AB, SE-461 30 Trollhättan, Sweden
[‡]Roche Pharma Research and Early Development, Roche Innovation Center, CH-4070 Basel, Switzerland

Supporting Information

ABSTRACT: Matched molecular pair analysis (MMPA) enables the automated and systematic compilation of medicinal chemistry rules from compound/property data sets. Here we present mmpdb, an open-source matched molecular pair (MMP) platform to create, compile, store, retrieve, and use MMP rules. mmpdb is suitable for the large data sets typically found in pharmaceutical and agrochemical companies and provides new algorithms for fragment canonicalization and stereochemistry handling. The platform is written in Python and based on the RDKit toolkit. It is freely available from https://github.com/rdkit/mmpdb.

J. Chem. Inf. Model. 2018, 58, 902–910.
ACS Editors’ Choice

Consortium model

- **My company is the supplier/organizer.**
  - **Easy to invoice.**
- **Anyone could pay to join. Suggested prices:**
  - **Academics - EUR 1 000 (no warranty, limited support)**
  - **Industry - EUR 5 000 (includes 9 months of support)**
- **Members get the source code under an open source license.**
  - **No promise the resulting work would be made public.**
- **Started the effort *after* I had some new features in place.**
  - **A company paid me (as a consultant) to add them.**
  - **I asked for permission to use that code as a seed.**
  - **Could promise those features to all consortium members.**
  - **Made them the first consortium member.**

Funding Levels and Goals

Companies are more willing to fund features.
Need to improve mmpdb's "infrastructure."

Raised
EUR 22 500
🙂
→

EUR 16 000 - Environment fragment SMILES

EUR 23 000 - Public release after 9 months

EUR 29 000 - Documentation

EUR 34 000 - mmpdb/GitHub user support

EUR 40 000 - Test suite development

EUR 50 000 - Immediate public release

Not enough funding for infrastructure improvements.
🙁

These prices are low compared to industry standards!

Plus overhead for marketing, web designer, and accounting.

Roche Funding

Roche funded nearly all of mmpdb development.

They had an in-house branch to handle large data sets, and they implemented the "generate" functionality.

Paid me to turn "research" code into "production" code. And they were consortium members.

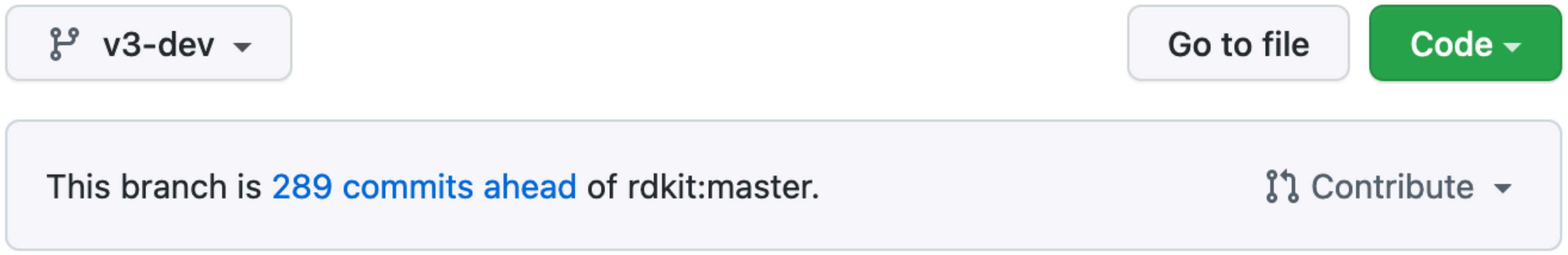
I merged the three code bases together, and decided to release the result to the public.

Current Status

mmpdb 3.0b1 is ready

https://github.com/adalke/mmpdb/tree/v3-dev

"v3-dev" branch



This branch is 289 commits ahead of rdkit:master.

Working on a speedup for "generate" for large DBs. Then will push to main branch under RDKit's account.

Future

No long-term development or support plans.

Christian Kramer @ Roche provides support in his spare time.

Still missing infrastructure (documentation and testing)
I think these are important for future development.

I don't know how to get stable long-term funding.

Thanks!

Roche

Mahendra Awale
Jérôme Hert
Christian Kramer

and to all the members of the mmpdb consortium!