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

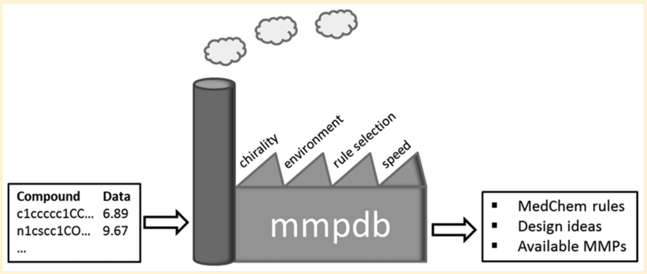
Andrew Dalke,*,† Jérôme Hert,‡ and Christian Kramer*,‡

†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.



INTRODUCTION

Matched molecular pair analysis (MMPA) has emerged as an attractive approach to guide compound optimization in hit-to-lead campaigns.[1−6] In short, a matched molecular pair (MMP) is formed by two molecules that differ by a defined structural transformation. For a given transformation, a MMP rule can be derived from the ensemble of corresponding MMPs and their associated property changes (activity readout, physicochemical property, etc.). A rule consists of a transformation and its associated statistics (average property difference, standard deviation, etc.). MMPA involves compiling all of the rules for one or several compound/property data sets and using them, e.g., to enumerate compounds with a specific profile or to estimate the impact of a compound modification on single or multiple properties. MMPA is based on the assumed additivity of chemical properties, including on-target activity. It automatically and systematically captures the implicit knowledge contained in data sets. In effect, MMP rules are practical tidbits of information that are directly usable to rationally modulate a compound's properties. MMP rules are of interest to medicinal chemists because they directly and intuitively reflect medicinal chemistry knowledge. There is already a significant body of literature that describes the approach[2,7−14] and reports its usage.[1,3−6,15−17]

The core concept of MMP is relatively straightforward to understand. Two different implementation flavors have been described in the literature: a maximum common substructure-based approach for example implemented in WizePairs[12] and a decomposition approach based on compound fragmentation and fragment indexing[8] (hereafter called fragment-and-index). Hussain and Rea's fragment-and-index approach is available from the RDKit repository as the open-source "mmpa" package, but it has some limitations related to canonicalization and handling of stereochemistry and does not enable consideration of the environment around attachment point(s).

Here we present mmpdb, an open-source MMP platform with a fragment-and-index engine. The platform enables the processing and handling of large data sets and implements a number of novel features. Rooted fingerprints are used to capture the environment around transformations and permit rule stratification. Novel canonicalization and indexing strategies have been devised to avoid indexing and chirality problems. Heuristics are employed to improve processing and searching execution times. The code has been released under a permissive three-clause BSD license and is available from https://github.com/rdkit/mmpdb.

MMP NOMENCLATURE

We use the nomenclature in Figure 1 to describe and discuss MMPs, MMP rules, and MMP analysis. Two compounds that differ by a single defined structural transformation form a matched pair. The part of those molecules that is identical is called constant, and the part that changes is called variable. The change from one variable part to another is the MMP transformation. MMP rules emerge from the combination of transformations plus environments, along with the statistics calculated from the effects of the observed transformation on a given property.

MMPA GENERAL PROCESS OVERVIEW

MMPA is applicable to both on-target SAR (e.g., biochemical activity measured against a single given protein) and ADMET and physicochemical properties (e.g., log D, kinetic solubility).

Received: March 22, 2018
Published: May 17, 2018

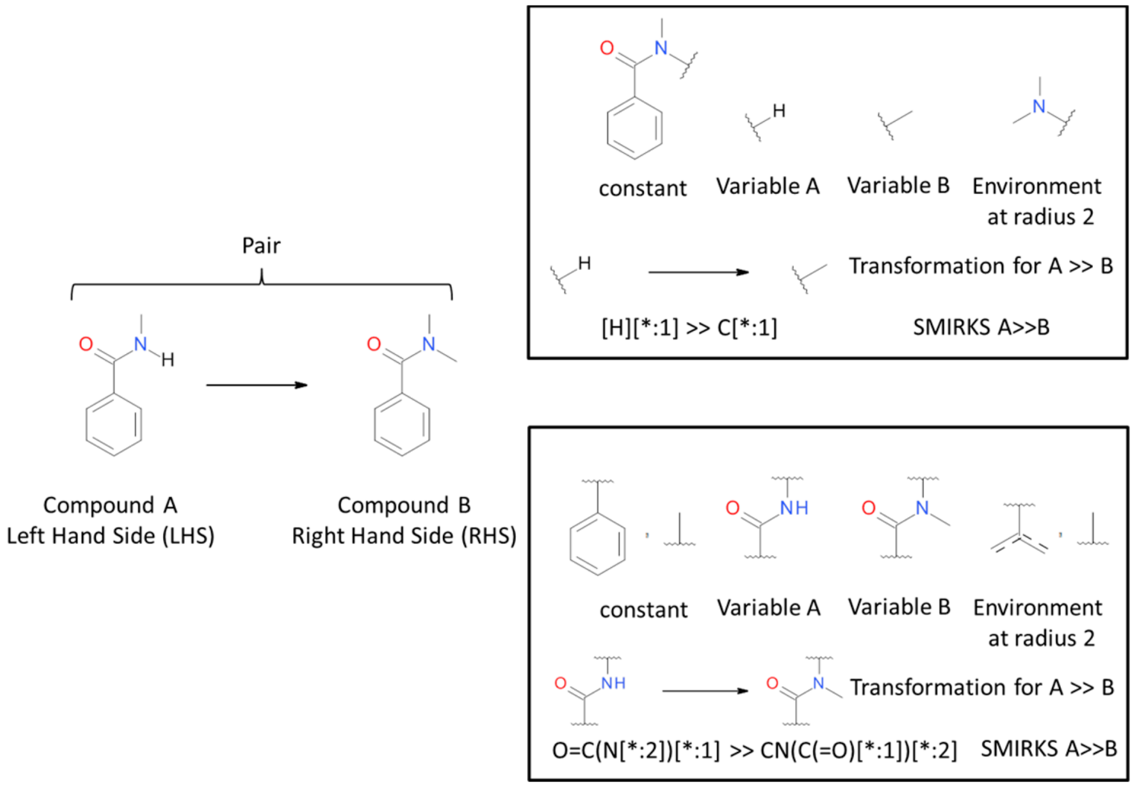
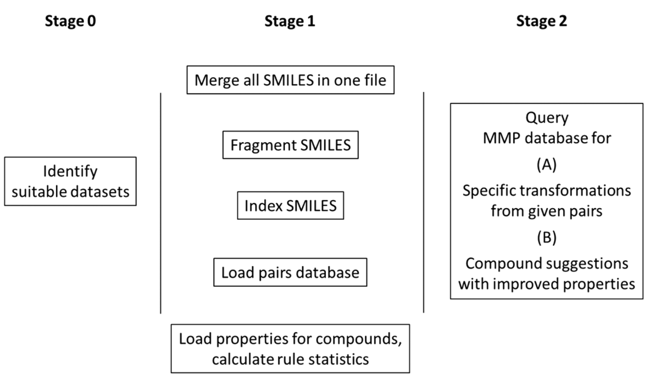


Figure 1. Nomenclature and denomination used for MMP: (left) a pair of matched compounds; (right) two of the possible transformations for the pair on the left.

The latter are especially well suited for MMPA because experimental data are abundant (at least in industry) from cross-project compound profiling and because ADME properties can be more "global", meaning that additivity[17] is more likely applicable—these properties are less likely to be dependent on the spatial orientation within a binding pocket.

mmpdb requires three stages to obtain useful MMP analyses and predictions, as illustrated in Figure 2.



Stage 0
Stage 1
Stage 2
Identify suitable datasets
Merge all SMILES in one file
Fragment SMILES
Index SMILES
Load pairs database
Load properties for compounds, calculate rule statistics
Query MMP database for
(A)
Specific transformations from given pairs
(B)
Compound suggestions with improved properties

Figure 2. Overview of the three stages required to build an MMP database and conduct MMPA.

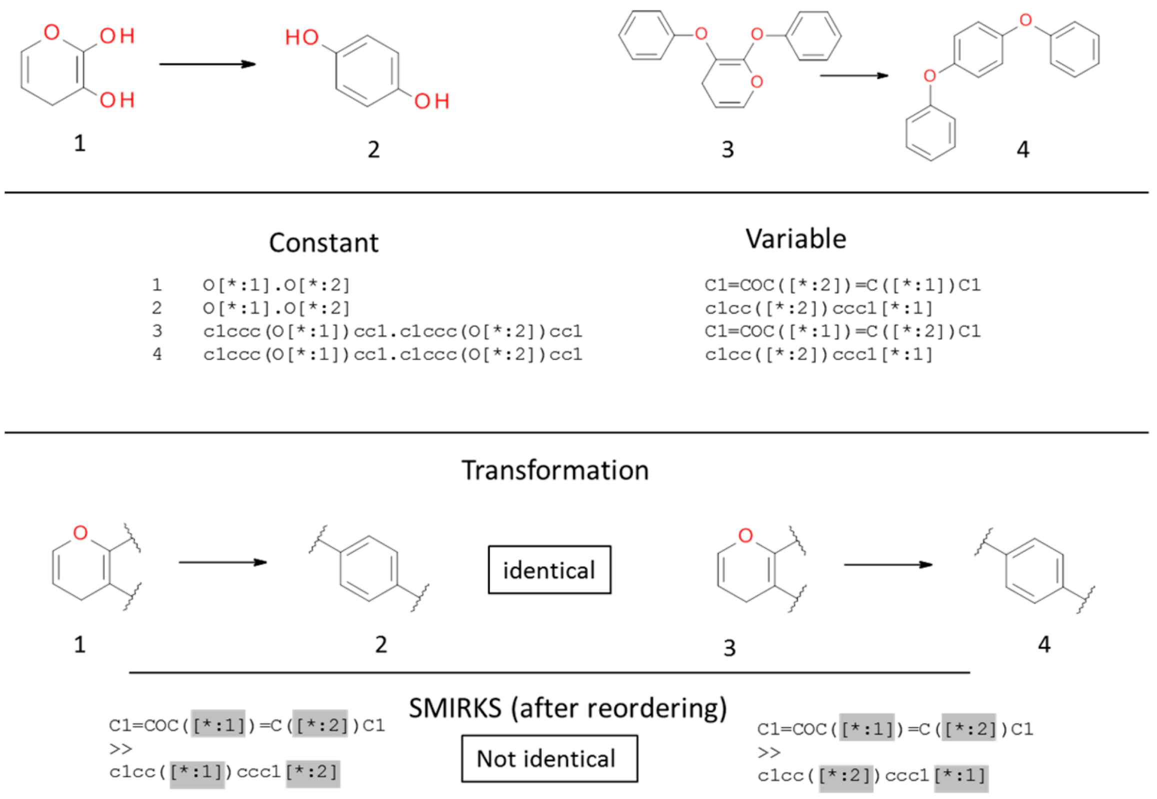
Stage 0 involves identifying and preparing suitable data sets for MMPA. This stage is not to be underestimated in practice, as the quality of the MMP rules and analysis depends on the assay quality and the number of pairs in the input data. Both the number of different transformations found and the number of pairs per transformation generally grow with the size of the data set. From our experience, we empirically recommend at least a few hundred compounds. Preparation of the data set should follow the same practices as those established for QSAR/QSPR.[18] Molecules should be cleaned up and standardized even though mmpdb itself can perform basic salt stripping and sanitizing. Since the MMP analysis works on desalted molecules, properties that depend on the salt form cannot be expected to be mapped well by mmpdb. Chiral bonds that are defined should be appropriately tagged, as mmpdb is capable of handling chirality for fragments and rules. However, mmpdb is based on SMILES input, so relative stereochemistry cannot be taken into account. Lastly, mmpdb expects at most one value per compound per property; multiple measurements need to be aggregated.

Stage 1 creates the MMP rule database. Molecules are first fragmented according to a user-defined cutting pattern. Fragments are then indexed and loaded into the database along with associated property information. Transformations are derived from the collection of fragments, and the aggregate statistics for each of their corresponding properties are calculated to form a rule. The details of how this stage has been automated in mmpdb are outlined in the sections below.

Stage 2 is the analysis stage, which uses the rules in the database to transform a query structure or predict property changes. Details of the two analysis features in mmpdb are provided below.

FRAGMENTATION AND MATCHED PAIR IDENTIFICATION

mmpdb is based on the fragment-and-index MMP algorithm as originally published by Hussain and Rea[8] and contributed to RDKit as the "mmpa package". Compounds are first fragmented to derive all possible constant and variable parts.



	Constant	Variable
1	O[*:1].O[*:2]	C1=COC([*:2])=C([*:1])C1
2	O[*:1].O[*:2]	c1cc([*:2])ccc1[*:1]
3	c1ccc(O[*:1])cc1.c1ccc(O[*:2])cc1	C1=COC([*:1])=C([*:2])C1
4	c1ccc(O[*:1])cc1.c1ccc(O[*:2])cc1	c1cc([*:2])ccc1[*:1]

```
C1=COC([*:1])=C([*:2])C1
>>
c1cc([*:1])ccc1[*:2]
```

```
C1=COC([*:1])=C([*:2])C1
>>
c1cc([*:2])ccc1[*:1]
```

Figure 3. Indexing problem that has been solved in mmpdb. Compounds (1, 2) and (3, 4) are linked by the same transformation, as drawn. The canonical SMILES from RDKit for the variable parts are different for compounds 1 and 3; the canonical SMILES produced depend on the attachment point labels. The resulting transformation SMIRKS for 1 ≫ 2 and 3 ≫ 4 do not match, although they should.

The bonds to be cut are specified by a SMARTS pattern. By default, mmpdb cuts acyclic single bonds (with some restrictions) and produces fragments resulting from single, double, and triple cuts. Alternatively, five other predefined cut-SMARTS patterns are defined or a user can specify a custom SMARTS pattern. Next, matching constant fragments are identified by simple string comparison of the canonical SMILES of the constants. This is very efficient and one of the major advantages of this algorithm. The associated compounds form an MMP. The combination of the variable parts of a given pair forms the MMP transformation. All of the transformations are passed to the indexing step.

This algorithm handles the simplest cases. Transformations involving hydrogens, multiple cuts, or chiral centers require specific treatment. Details of how these cases are handled in mmpdb are outlined below.

Transformations Involving Hydrogens. Systematically generating all of the fragments where one or several hydrogen atoms are cut off would result in a dramatic increase in the number of transformations, most of which have limited value. In order to take into account only relevant hydrogen transformations, mmpdb implements the same strategy as the original mmpa code. Only single-cut hydrogen transformations are supported. For every single cut of molecule A, a hydrogen is substituted on the attachment point of the constant fragment, and the resulting structure is canonicalized. If it matches the canonical SMILES of one of the input structures, call it molecule B, then the transformation from A to B is recorded as the substitution of A's variable part with a hydrogen, or vice versa for the transformation of B to A.

Transformations with Multiple Cuts. In the case of multiple cuts, the match between the attachment points on the variable and constant parts needs to be tracked appropriately. A simple label such as "[*:1]", "[*:2]", or "[*:3]" affects the RDKit[19] canonicalization procedure. Figure 3 shows one example of double-cut fragmentation where the mmpa code does not match two identical transformations because the asymmetry of the arbitrary labels results in inconsistent canonicalization. It should be noted that some but not all of the issues around this canonicalization were fixed in RDKit versions from 2017.9 onward.

To address this problem, mmpdb uses unlabeled attachment points in association with an "attachment order" term describing how the attachment points in the variable and constant parts are connected. The unlabeled points do not induce an artificial asymmetry in the constant part, so the canonicalization of the fragment is consistent, but the labels are useful when figuring out which attachment points in the variable and constant parts are connected. The new approach makes use of an array returned by the RDKit canonicalization algorithm that maps the rank order of the atom in the canonical SMILES with its original index in the molecule. This information together with the attachment order enables labeling of the attachment points after canonicalization.

Internal symmetries in the constant or variable part may further complicate the problem, as multiple attachment orders are possible. mmpdb defines the canonical attachment order as the numerically smallest one and computes it using a lookup table based on the attachment order and the two symmetry class descriptions.

Transformations Involving Chirality—Dealing with Existing Chiral Atoms. In RDKit, chirality depends on the bond ordering of the connection table. Adding, removing, or breaking bonds can modify this order and can result in the inversion of stereochemistry or double bonds. mmpdb cuts bonds using RDKit's FragmentOnBonds function, which preserves chirality when cutting a bond, plus code to also preserve bond directionality and *E*/*Z* geometric isomerism.

The fragmentation uses the "[*]" wildcard atom as a placeholder to indicate a connection point. If there are multiple connection points, then the variable part may gain a symmetry that was not present in the original structure, causing a loss of one or more chiral indicators. To recover the chirality, we first developed a "welding" technique. The SMILES of the constant and variable parts are modified at the syntax level to produce a valid SMILES for the reconnected structure. The welder rewrites the attachment points as bond closures with a high closure number and concatenates the resulting two SMILES strings with a dot disconnect. For example, if the constant and variable parts are "[*]O.[*]P" and "[*]CN[*]" and the attachment order is "21", then the welded SMILES is "O%91.P%92.C%92N%91". Welding was used to validate the fragmentation algorithm because the canonicalization of the welded terms ("ONCP" for this example) should match the original canonical structure. The syntax manipulation was made simpler because the RDKit canonical SMILES writer always places the wildcard atom either immediately before or immediately after the base atom or previous wildcard atom. If a wildcard atom is before a chiral base atom with an implicit chiral hydrogen, then the atom chirality must be inverted. Care must also be taken to preserve and possibly invert the "/" or "\" direction bond.

The welding technique is used to recover lost chirality. If the input structure is chiral and the canonicalization of the welded fragments leads to fewer chiral atoms, then the *n* atoms that have lost their chiral assignments are identified. The algorithm enumerates the 2^n possible ways to assign an "@" or "@@" tag to each such atom in the variable part at the SMILES syntax level, rewelds the original constant with the new variable part, and recanonicalizes. The enumeration process stops once the result matches the original structure, as the SMILES for the modified variable part is the chirality-preserving fragment. This method starts from a canonical representation, and the enumeration process is canonical-preserving, so the result is also canonical.

The chirality tags are easy to insert if the corresponding atom term in the SMILES string uses the bracket notation, e.g. "[C]". As a complication, if a chiral atom with an implicit hydrogen (e.g., "[C@@H]") lost its chirality tag, then the resulting SMILES term might use the organic subset short-hand notation for the atom (e.g., "C"). A preparation stage identifies all of these atoms and rewrites their SMILES terms in bracket form along with the correct hydrogen count. There will be at most one hydrogen because these atoms were originally chiral.

Transformation Involving Chirality—Dealing with New Chiral Atoms. Upon fragmentation, chirality can be created or removed, with the consequence that valid matched pairs of compounds may not be identified. Figure 4 depicts such a case.

mmpdb implements a routine that detects when new chirality is formed upon bond breaking and uses an enumeration strategy to correctly identify matched pairs; we call this "up-enumeration". Specifically, if a new chiral center is present in the constant part of a fragmented molecule, the algorithm produces three fragments: one with no stereoconfiguration specified and each of the other two having one of the possible stereoconfigurations (clockwise and counterclockwise, denoted with the "@" and "@@" tags in the SMILES string). The latter two fragments are also tagged as up-enumerated. If there are *n* new chiral centers, then the up-enumeration produces 3^{n-1} fragmentations. During indexing, those tagged constant parts can only be matched to a non-tagged partner, i.e., one that was not up-enumerated. This procedure identifies all valid pairs with a different number of defined stereocenters. It should be noted that we do not label or enumerate the stereochemistry of chiral atoms with an undefined configuration; a compound with an undefined chiral center will only be matched with other compounds with the same undefined chiral center.

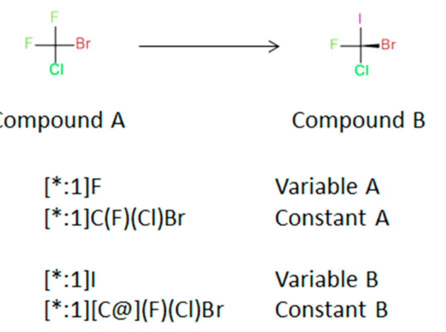


Figure 4. Example of a valid matched pair with chirality created in the transformation. With a standard fragmentation procedure, the pair A ≫ B is not found because the constant part SMILES differ by the chirality tag "@". mmpdb uses up-enumeration to match those compounds.

Environment Representation. MMP rules are highly dependent on the local environment around transformations.[10,17] 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. In our experience, stratifying matched pairs by the local environment of the attachment points leads to more meaningful and useful rules. In mmpdb, the local environment is encoded by a SHA256 hash of the concatenation of the circular fingerprints rooted at each attachment atom, ordered canonically. Six environment hashes are computed for each matched pair transformation, one for each radius from 0 to 5 bonds away. The hashed environment fingerprints do not reveal direct information about the chemical structure, but they do provide an easy and effective means of stratifying the transformation data by testing for equality. The environment fingerprint is not canonical for a given environment because it depends on the ordering of the fragments in the variable term. Instead of creating a new canonicalization method for the circular environments, we enumerate all one, two, or six environments for one-, two-, or three-cut fragmentations, respectively, and check for any exact match.

Recorded Information. MMP fragments are recorded in JSON Lines[20] format. The first few lines contain global configuration information, such as the fragmentation parameters used, followed by the compound records. Each compound record contains the compound identifier, the input and canonical SMILES strings, the number of heavy atoms, and a list of fragmentations. Each fragmentation record is a 10-element list. Figure 5 and Table 1 illustrate the fragmentation

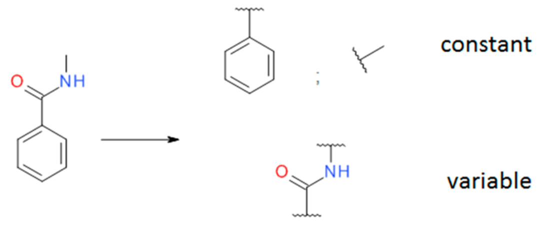


Figure 5. Example of fragments resulting from a two-cut fragmentation of *N*-methylbenzamide.

Table 1. Example of a Fragment Record Generated for One Fragment of *N*-Methylbenzamide

example value	meaning
2	number of cuts in the fragmentation
"N"	chiral enumeration flag:
	"N" = no enumeration done
	"C" = constant up-enumeration
	"V" = variable up-enumeration
3	number of heavy atoms in the variable part
"12"	symmetry of the attachment points in the variable part
"[*]NC([*])=O"	the variable part as a canonical SMILES
"01"	mapping from the variable part's attachment points to the constant
7	number of heavy atoms in the constant part
"12"	symmetry of the attachment points in the constant part
"[*]C.[*]c1ccccc1"	the constant part as a canonical SMILES
null	for single-cut structures, the canonical SMILES for the variable part attached to [H]; otherwise null.

and a corresponding fragment record for *N*-methylbenzamide. If an input structure could not be parsed or the structure did not pass one of several simple filters (e.g., too many rotatable bonds or too many heavy atoms), then the identifier, input SMILES, and a message describing why it was excluded may be recorded but are otherwise ignored.

Depending on the data set size, fragmentation can be a time-consuming step. mmpdb implements a number of ways to reduce the overall clock time. First, mmpdb can take advantage of multiple processors to carry out the fragmentation in parallel. A number of thresholds can be set to limit the size of fragments in the variable part by defining the maximum number of heavy atoms and/or the maximum number of rotatable bonds. Finally, for structures that have not changed, the fragmentation procedure can reuse fragmentations from a previous run.

Indexing. The indexing step is where matched pairs are identified and written to the database. What constitutes a valid pair is highly customizable to control the number of pairs and the amount of information encoded from the source compounds. Available options include the maximum number of atoms in a variable part, the minimum and maximum ratios of heavy atoms in the variable parts compared with the entire molecule both with and without the atoms from the constant part encoded in the environment, and the maximum difference of heavy atoms in the variable parts of a matched pair.

mmpdb Database Schema. The mmpdb database stores all of the information about the compounds and their properties, matched pairs, and MMPs. We decided to implement mmpdb with SQLite binding because SQLite is open-source and the update mechanism with SQLite is very easy: each update results in a single file that can be built and tested without creating a special database on the database server. Once the update is ready, it can be deployed by linking to the new database file and restarting the mmpdb analysis server.

The mmpdb database contains 11 tables, 10 of which are interlinked. The database schema is shown in Figure 6.

Load Properties. A matched molecular pair transformation is stored as a "rule" in the database, and its circular environments are stored as "rule environment" entries. One or more properties can be loaded into the database, in which case summary statistics are computed for each combination of environment radius and property. These include the number of pairs underlying a given transformation, radius, and property and the average change in value as well as its standard deviation, kurtosis, skewness, minimum, Q1, median, Q3, maximum, paired_t and p_value. The p_value denotes the probability for the given mean and standard deviation to be observed if the true average effect (e.g., from an infinite number of pairs) of the transformation within the given environment is zero. The p_value hence provides a measure of significance for the rules. It should be noted that significance indicates that the true average effect of a transformation, estimated from the pairs available, is likely to be different from zero. The user will always have to decide whether the expected change in a given property is sufficient to likely solve the problems he or she wants to solve.

■ MMP ANALYSIS CASES

We have implemented two specific analysis cases for ADMET and physicochemical MMPA: "transform" and "predict".

Transform. Our "transform" case implements the following question: Which new (matched-pair derived) molecules would improve the given properties? Using an input molecule as starting point, it identifies relevant MMP rules by fragmenting the input, finding transformations that contain the variable part, and applying those transformations to generate a set of novel virtual molecules for which properties can be anticipated. That set of compounds is often large (several hundreds or thousands of compounds) but can be reduced to a more practical size using the available filters. The output molecules may be restricted to those that match a particular substructure, e.g., to help focus on the transformations that occur on a specific part of the molecule. A number of thresholds are available to filter undesired molecules or transformations, including the numbers of heavy atoms in the constant and variable parts, the number of pairs underlying a rule, minimum environment radius, and direction of the property change. Filter expressions that involve several properties can also be written.

There are typically several rule environments that can lead to a given output compound. Thus, an assessment to determine the most representative rule is required. The mmpdb heuristic selects the candidate rule with the smallest standard deviation and a sufficient number of representative pairs.[21] Preference is first given to rules based on more than 10 matched pairs. If multiple rules fulfill these criteria, the preferred rule is the one with the smallest associated standard deviation. Any further ties are broken in preference for the largest radius, then the largest number of heavy atoms in the product side of the transformation, and finally the alphabetically first product SMILES string. If none of the rules are based on more than 10 pairs, the same selection and tie-breaking process is applied using a threshold of five pairs and then at least one pair. If several properties are queried at the same time, different trans-

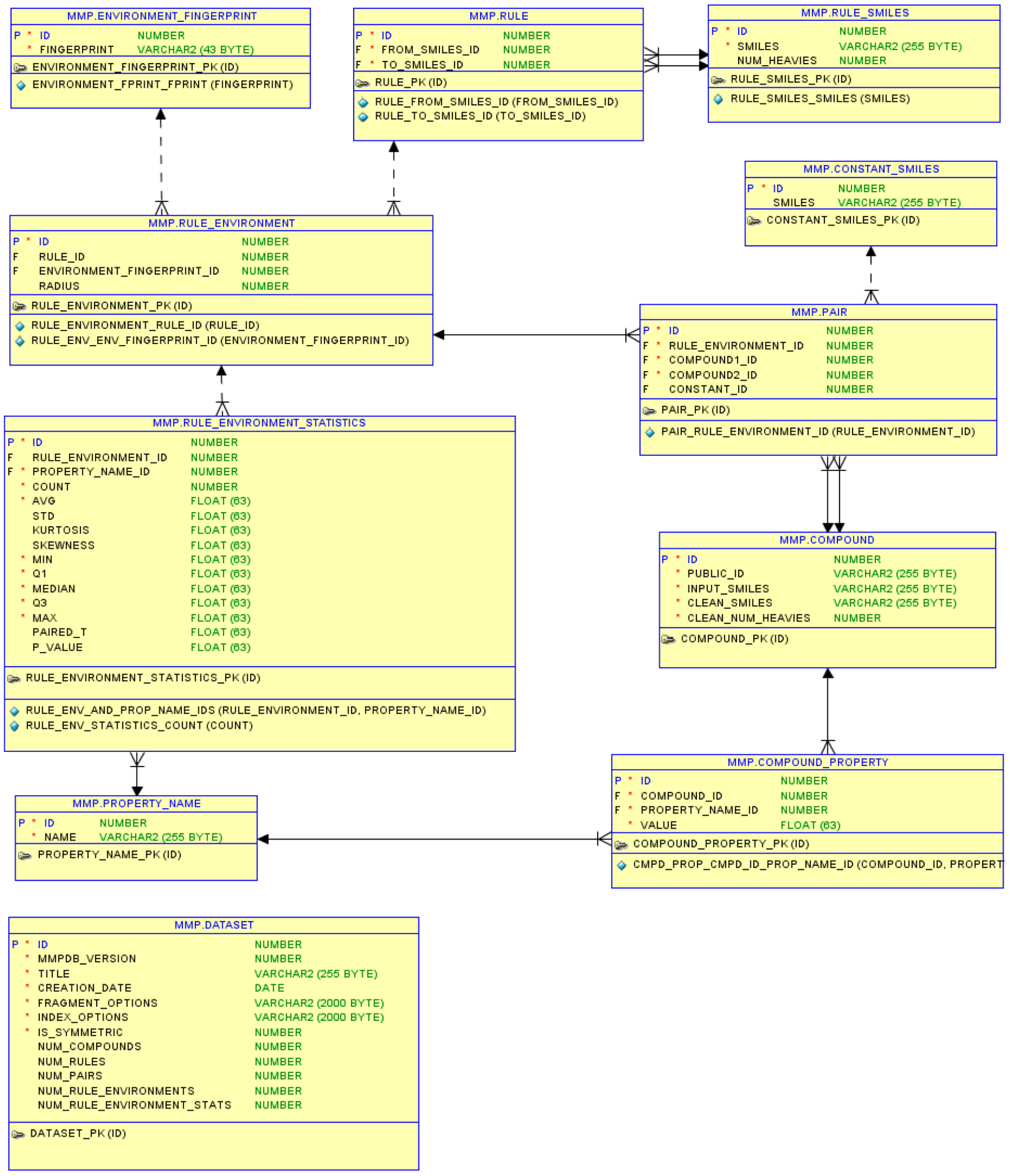

MMP.ENVIRONMENT_FINGERPRINT
P * ID NUMBER
* FINGERPRINT VARCHAR2 (43 BYTE)
ENVIRONMENT_FINGERPRINT_PK (ID)
ENVIRONMENT_FPRINT_FPRINT (FINGERPRINT)
MMP.RULE
P * ID NUMBER
F * FROM_SMILES_ID NUMBER
F * TO_SMILES_ID NUMBER
RULE_PK (ID)
RULE_FROM_SMILES_ID (FROM_SMILES_ID)
RULE_TO_SMILES_ID (TO_SMILES_ID)
MMP.RULE_SMILES
P * ID NUMBER
* SMILES VARCHAR2 (255 BYTE)
NUM_HEAVIES NUMBER
RULE_SMILES_PK (ID)
RULE_SMILES_SMILES (SMILES)
MMP.CONSTANT_SMILES
P * ID NUMBER
SMILES VARCHAR2 (255 BYTE)
CONSTANT_SMILES_PK (ID)
MMP.RULE_ENVIRONMENT
P * ID NUMBER
F RULE_ID NUMBER
F ENVIRONMENT_FINGERPRINT_ID NUMBER
RADIUS NUMBER
RULE_ENVIRONMENT_PK (ID)
RULE_ENVIRONMENT_RULE_ID (RULE_ID)
RULE_ENV_ENV_FINGERPRINT_ID (ENVIRONMENT_FINGERPRINT_ID)
MMP.PAIR
P * ID NUMBER
F * RULE_ENVIRONMENT_ID NUMBER
F * COMPOUND1_ID NUMBER
F * COMPOUND2_ID NUMBER
F CONSTANT_ID NUMBER
PAIR_PK (ID)
PAIR_RULE_ENVIRONMENT_ID (RULE_ENVIRONMENT_ID)
MMP.RULE_ENVIRONMENT_STATISTICS
P * ID NUMBER
F RULE_ENVIRONMENT_ID NUMBER
F * PROPERTY_NAME_ID NUMBER
* COUNT NUMBER
* AVG FLOAT (63)
STD FLOAT (63)
KURTOSIS FLOAT (63)
SKEWNESS FLOAT (63)
* MIN FLOAT (63)
* Q1 FLOAT (63)
* MEDIAN FLOAT (63)
* Q3 FLOAT (63)
* MAX FLOAT (63)
PAIRED_T FLOAT (63)
P_VALUE FLOAT (63)
RULE_ENVIRONMENT_STATISTICS_PK (ID)
RULE_ENV_AND_PROP_NAME_IDS (RULE_ENVIRONMENT_ID, PROPERTY_NAME_ID)
RULE_ENV_STATISTICS_COUNT (COUNT)
MMP.COMPOUND
P * ID NUMBER
* PUBLIC_ID VARCHAR2 (255 BYTE)
* INPUT_SMILES VARCHAR2 (255 BYTE)
* CLEAN_SMILES VARCHAR2 (255 BYTE)
* CLEAN_NUM_HEAVIES NUMBER
COMPOUND_PK (ID)
MMP.PROPERTY_NAME
P * ID NUMBER
* NAME VARCHAR2 (255 BYTE)
PROPERTY_NAME_PK (ID)
MMP.COMPOUND_PROPERTY
P * ID NUMBER
F * COMPOUND_ID NUMBER
F * PROPERTY_NAME_ID NUMBER
* VALUE FLOAT (63)
COMPOUND_PROPERTY_PK (ID)
CMPD_PROP_CMPD_ID_PROP_NAME_ID (COMPOUND_ID, PROPERT
MMP.DATASET
P * ID NUMBER
* MMPDB_VERSION NUMBER
* TITLE VARCHAR2 (255 BYTE)
* CREATION_DATE DATE
* FRAGMENT_OPTIONS VARCHAR2 (2000 BYTE)
* INDEX_OPTIONS VARCHAR2 (2000 BYTE)
* IS_SYMMETRIC NUMBER
NUM_COMPOUNDS NUMBER
NUM_RULES NUMBER
NUM_PAIRS NUMBER
NUM_RULE_ENVIRONMENTS NUMBER
NUM_RULE_ENVIRONMENT_STATS NUMBER
DATASET_PK (ID)

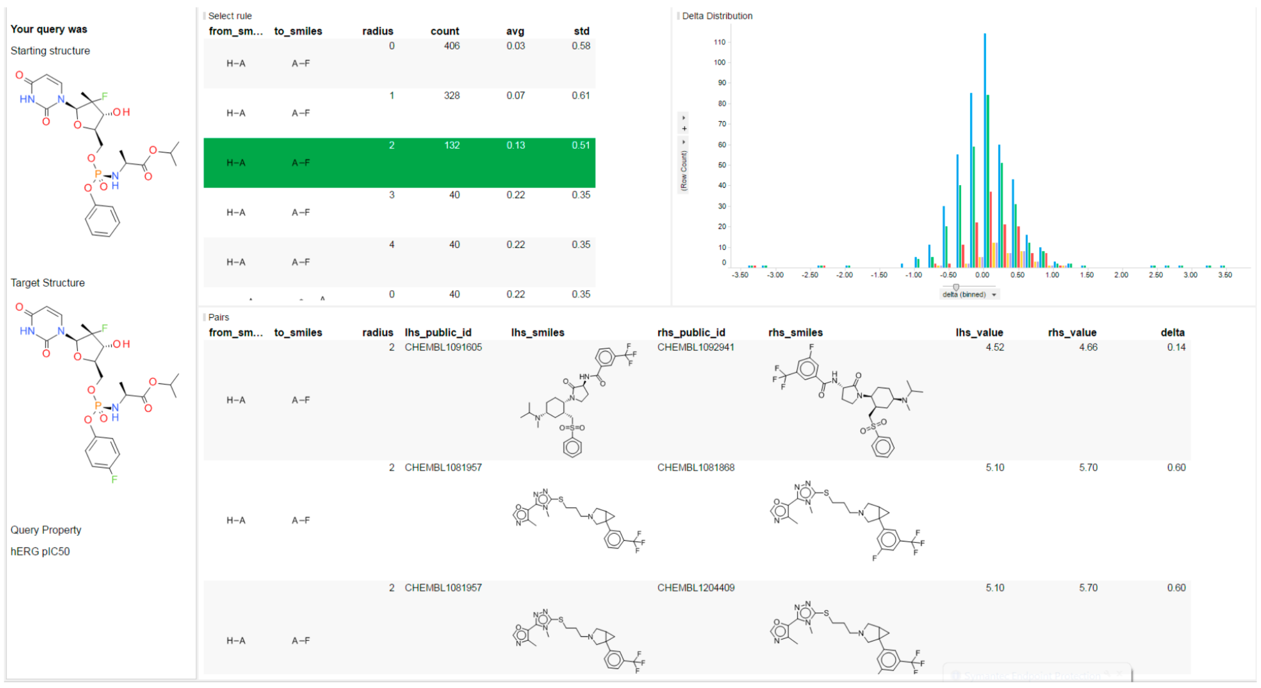
Figure 6. Mmpdb database schema.

formations may be selected for different properties, although both lead to the same compound. This is reflected in the output that shows the transformation and radius used for each individual property.

Predict. The "predict" case answers the following question: What will be the change in one property value for a given pair of compounds? Those typically consist of an existing reference compound and a virtual query compound. mmpdb identifies the MMP rules that transform the query into the reference and outputs the rules, statistics, and associated matched pairs. By default the prediction algorithm applies the same heuristic as described above to select the most representative rule environment and predict the property difference. It can also be configured to export all of the relevant transformations and the pairs of database structures to flat files that can visualized and analyzed in other tools. Figure 7 shows what such an analysis can look like in SpotFire.

■ MMPDB—TIPS TO IMPROVE RUNTIME

The fragmentation and indexing steps are needed only when the underlying data change. For our data sets, they can be completed over the course of a day, or faster if the data have



Select rule

from_sm...	to_smiles	radius	count	avg	std
H–A	A–F	0	406	0.03	0.58
H–A	A–F	1	328	0.07	0.61
H–A	A–F	2	132	0.13	0.51
H–A	A–F	3	40	0.22	0.35
H–A	A–F	4	40	0.22	0.35
		0	40	0.22	0.35

Pairs

from_sm...	to_smiles	radius	lhs_public_id	lhs_smiles	rhs_public_id	rhs_smiles	lhs_value	rhs_value	delta
H–A	A–F	2	CHEMBL1091605		CHEMBL1092941		4.52	4.66	0.14
H–A	A–F	2	CHEMBL1081957		CHEMBL1081868		5.10	5.70	0.60
H–A	A–F	2	CHEMBL1081957		CHEMBL1204409		5.10	5.70	0.60

Figure 7. Output of the "predict" algorithm visualized in SpotFire.

not changed much and the previous fragmentation output is available. This time is not important from the perspective of the scientific user, as those steps are conducted ahead of an MMPA.

The two analysis routines, transform and predict, are directly executed by users, and their runtime is more important. Our analysis identified communication with the SQLite database as the rate-limiting step. Both routines cause the SQLite database to make a lot of random access seeks to the filesystem in order to retrieve data from the database file, which in our case was located on a hard disk, where the seek latency is measured in milliseconds. We originally sped up the process by loading the database into memory, using the ":in-memory:" option in APSW, a Python wrapper for the SQLite library. However, this procedure is incompatible with multiprocessing, since multiprocessing duplicates the database copies in memory. We found that a more effective way of accelerating database access is to load the database into memory with RAM drive, i.e., to use part of the computer RAM as a file system, store the database there, and employ the standard SQLite database access mechanism. With this workaround, we found that the runtime to return the output of a query can be utilized for a web service (i.e., <10 s) even with a large database (~10 GB). Another option would be to place the data on a solid-state drive.

■ TIMING OVERVIEW

We used all of the CYP3A4 (ChEMBL target ID ChEMBL340) and hERG (ChEMBL target ID ChEMBL340) data from ChEMBL23 to generate a reproducible timing benchmark. We merged all of the IC_{50} and K_i data for hERG and IC_{50}, AC_{50}, and K_i data for CYP3A4 with PCHEMBL values and removed undefined compounds and duplicates. The result was 14 377 compounds for CYP3A4 and 6192 compounds for hERG, with 302 compounds having a measured value for both hERG and CYP3A4, yielding a data set with 20 267 compounds overall. It should be noted that we employed this very coarse data cleaning and merging protocol for illustration purposes only. Additional care would need to be taken to assemble a hERG or CYP3A4 data set for actual MMPA and ensure compatibility of the assay, reproducibility of the data, etc. The data set used for this analysis is available in the Supporting Information.

Calculations were performed on a stand-alone desktop computer running RedHat 7 using 10 Intel Xeon CPUs with a processing speed of 2.3 GHz and 32 GB of RAM. mmpdb was executed with Python 3.6 and the default parameters. Overall, the fragmentation completed in about 13 min. Indexing and database upload completed in about 80 s. Loading the hERG and CYP3A4 properties to the database took an additional 62 s. This database consists of 26 427 different fragments, 1 811 988 pairs, 176 952 rules, 1 255 848 rule environments, and 1 239 206 statistics for the two properties and given rule environments.

A "transform" analysis of the benchmark database on the command line using the default parameters and sofosbuvir as both the query structure and substructure filter produced a set of 1620 novel compounds in 51 s. The same analysis run through a web service and using a RAM drive, which avoids all of the startup overhead and hard disk seek times, completed in 1.7 s.

A "predict" analysis with sofosbuvir and *p*-fluorophenyl sofosbuvir as a matched pair and hERG as query target property completed in 17 s on the command line and 1.4 s through a web service. It is important to note that the overall timing for the queries greatly depends on the size of the query molecules and the size of the database.

■ OUTLOOK

mmpdb provides what we believe is the essential and core functionality necessary to conduct MMPA. However, there are various opportunities to improve and expand the platform. In its current form, the mmpdb platform is an expert tool. The platform would greatly benefit from a graphical user interface, first to carry out "predict" and "transform" queries but also to build up and maintain MMP rule databases.

Currently, mmpdb handles only exact single values, and the entire analysis assumes normally distributed data. This does not map the outcome of typical ADMET and physicochemical assays, since many of the measurements are qualified, i.e., the measured value is above or below the assay thresholds and is only known as >*X* or <*X*. Qualified data are rather abundant and require other types of statistics. Extending mmpdb to include the ability to deal with qualitative data together with an implementation of the associated statistics would further improve its usefulness.

Another area of further interest concerns how to select a rule when multiple rules lead to the identical transformation. The current mmpdb heuristic is just one approach, and better ones may be developed. Database pruning represents another challenge for the platform. The number of rules generated increases with the number of compounds, though they can be either redundant or irrelevant. If those rules can be identified, it may be possible to significantly reduce the overall size of the database.

Lastly, the functionality could be improved by implementing additional analysis methods and techniques to mine the rules. One example could involve a query based on a reaction rather than on exact structures.

■ SUMMARY

mmpdb is an open-source MMP platform written in Python and based on the RDKit. It can be used for MMPA and is suitable for ADMET and physicochemical data sets of the size encountered in the pharmaceutical and agrochemical industry. mmpdb is based on the fragment-and-index algorithm pioneered by Hussain and Rea[8] and permits the calculation and storage of MMP rules including transformation statistics and the effects of attachment point environments. mmpdb includes a number of novel solutions to fragmentation and indexing problems with the aim of identifying all desired pairs and handling all queries correctly.

Two routines are available to make use of the MMP rules. mmpdb has been optimized to be efficient enough for utilization in design workflows. The source code is available from https://github.com/rdkit/mmpdb. Suggestions and contributions are welcome.

■ ASSOCIATED CONTENT

Ⓢ Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jcim.8b00173.

Two data sets with hERG and CYP3A4 data extracted from ChEMBL23 used to time the performance of mmpdb (ZIP)

■ AUTHOR INFORMATION

Corresponding Authors

*E-mail: dalke@dalkescientific.com.
*E-mail: Christian.Kramer@roche.com.

ORCID

Christian Kramer: 0000-0001-8663-5266

Notes

The authors declare no competing financial interest.

■ REFERENCES

(1) Wassermann, A. M.; Dimova, D.; Iyer, P.; Bajorath, J. Advances in Computational Medicinal Chemistry: Matched Molecular Pair Analysis. *Drug Dev. Res.* **2012**, *73*, 518−527.

(2) Dossetter, A. G.; Griffen, E. J.; Leach, A. G. Matched Molecular Pair Analysis in Drug Discovery. *Drug Discovery Today* **2013**, *18*, 724−731.

(3) Griffen, E.; Leach, A. G.; Robb, G. R.; Warner, D. J. Matched Molecular Pairs as a Medicinal Chemistry Tool. *J. Med. Chem.* **2011**, *54*, 7739−7750.

(4) Leach, A. G.; Jones, H. D.; Cosgrove, D. A.; Kenny, P. W.; Ruston, L.; MacFaul, P.; Wood, J. M.; Colclough, N.; Law, B. Matched Molecular Pairs as a Guide in the Optimization of Pharmaceutical Properties; a Study of Aqueous Solubility, Plasma Protein Binding and Oral Exposure. *J. Med. Chem.* **2006**, *49*, 6672−6682.

(5) Ritchie, T. J.; Macdonald, S. J. F.; Pickett, S. D. Insights into the Impact of N- and O-Methylation on Aqueous Solubility and Lipophilicity Using Matched Molecular Pair Analysis. *MedChemComm* **2015**, *6*, 1787−1797.

(6) Meanwell, N. A. Synopsis of Some Recent Tactical Application of Bioisosteres in Drug Design. *J. Med. Chem.* **2011**, *54*, 2529−2591.

(7) Griffen, E.; Leach, A. G.; Robb, G. R.; Warner, D. J. Matched Molecular Pairs as a Medicinal Chemistry Tool: Miniperspective. *J. Med. Chem.* **2011**, *54*, 7739−7750.

(8) Hussain, J.; Rea, C. Computationally Efficient Algorithm to Identify Matched Molecular Pairs (MMPs) in Large Data Sets. *J. Chem. Inf. Model.* **2010**, *50*, 339−348.

(9) Lukac, I.; Zarnecka, J.; Griffen, E. J.; Dossetter, A. G.; St-Gallay, S. A.; Enoch, S. J.; Madden, J. C.; Leach, A. G. Turbocharging Matched Molecular Pair Analysis: Optimizing the Identification and Analysis of Pairs. *J. Chem. Inf. Model.* **2017**, *57*, 2424−2436.

(10) Papadatos, G.; Alkarouri, M.; Gillet, V. J.; Willett, P.; Kadirkamanathan, V.; Luscombe, C. N.; Bravi, G.; Richmond, N. J.; Pickett, S. D.; Hussain, J.; Pritchard, J. M.; Cooper, A. W. J.; Macdonald, S. J. F. Lead Optimization Using Matched Molecular Pairs: Inclusion of Contextual Information for Enhanced Prediction of HERG Inhibition, Solubility, and Lipophilicity. *J. Chem. Inf. Model.* **2010**, *50*, 1872−1886.

(11) Tyrchan, C.; Evertsson, E. Matched Molecular Pair Analysis in Short: Algorithms, Applications and Limitations. *Comput. Struct. Biotechnol. J.* **2017**, *15*, 86−90.

(12) Warner, D. J.; Griffen, E. J.; St-Gallay, S. A. WizePairZ: A Novel Algorithm to Identify, Encode, and Exploit Matched Molecular Pairs with Unspecified Cores in Medicinal Chemistry. *J. Chem. Inf. Model.* **2010**, *50*, 1350−1357.

(13) Keefer, C. E.; Chang, G.; Kauffman, G. W. Extraction of Tacit Knowledge from Large ADME Data Sets via Pairwise Analysis. *Bioorg. Med. Chem.* **2011**, *19*, 3739−3749.

(14) Sheridan, R. P.; Hunt, P.; Culberson, J. C. Molecular Transformations as a Way of Finding and Exploiting Consistent Local QSAR. *J. Chem. Inf. Model.* **2006**, *46*, 180−192.

(15) Dossetter, A. G. A Matched Molecular Pair Analysis of in Vitro Human Microsomal Metabolic Stability Measurements for Methylene Substitution or Replacements — Identification of Those Transforms More Likely to Have Beneficial Effects. *MedChemComm* **2012**, *3*, 1518−1525.

(16) Hu, Y.; Bajorath, J. Hierarchical Analysis of Bioactive Matched Molecular Pairs, Encoded Chemical Transformations, and Associated Substructures. *Mol. Inf.* **2016**, *35*, 483−488.

(17) Kramer, C.; Ting, A.; Zheng, H.; Hert, J.; Schindler, T.; Stahl, M.; Robb, G.; Crawford, J. J.; Blaney, J.; Montague, S.; Leach, A. G.; Dossetter, A. G.; Griffen, E. J. Learning Medicinal Chemistry Absorption, Distribution, Metabolism, Excretion, and Toxicity (ADMET) Rules from Cross-Company Matched Molecular Pairs Analysis (MMPA): Miniperspective. *J. Med. Chem.* **2018**, *61*, 3277.

(18) Fourches, D.; Muratov, E.; Tropsha, A. Trust, But Verify: On the Importance of Chemical Structure Curation in Cheminformatics and QSAR Modeling Research. *J. Chem. Inf. Model.* **2010**, *50*, 1189−1204.

(19) Landrum, G. RDKit: Open-Source Cheminformatics Software. http://www.rdkit.org (accessed Feb 5, 2018).
(20) JSON Lines: Documentation for the JSON Lines text file format. http://jsonlines.org (accessed Feb 5, 2018).
(21) Kramer, C.; Fuchs, J. E.; Whitebread, S.; Gedeck, P.; Liedl, K. R. Matched Molecular Pair Analysis: Significance and the Impact of Experimental Uncertainty. *J. Med. Chem.* **2014**, *57*, 3786−3802.