UPPSALA UNIVERSITET

Metabolomics database resolver

Rajmund Csombordi
Supervisor: Sara Younes
Subject Reader: Johan Viklund

Abstract

This paper documents arising issues revolving categorizing metabolome compounds and a possible solution in the form of an R package that is capable of matching up different database identifiers with each other. Then, using this package we reflect on the average coverage of external reference between metabolome databases to highlight the lack of a universal compound primary identifier.

Metafetcher - accessing metabolomics data in a simplified way

Thesis documentation

Rajmund Csombordi

Metabolomics is the scientific study of small compounds taking part in metabolic processes, and it holds the key towards treating metabolic disorders (such as diabetes). Often these disorders relate to imbalance of metabolome compounds, and in general it is in our interest to understand these compounds to get a fuller picture of the pathways involved.

For this reason scientists – similarly to other fields of bioinformatics – store generalized models of these compounds in various databases that usually support searching and labelling items. Contrary to gene and protein databases, metabolome databases are neither as consistent, nor as interconnected. Working with metabolomics databases is cumbersome, as collecting compound data proposes several obstacles that needs resolving before a scientist could move on from the data collection phase in a research project.

To account these problems we have designed and developed an R package to ease up the work with databases. Our package focuses primarily on associating different metabolome database identifiers with each other, easing up the discovery of additional data and hopefully providing the possibility to reference metabolites in a research project with one database's identifier.

Degree project in bioinformatics, 2020
Examensarbete i bioinformatik 30 hp till masterexamen, 2020
Biology Education Centre and Biomedical Center
Supervisor: Sara Younes

Table of Contents

Abbreviations and terminology

ChEBI	Chemical Entities of Biological Interest
HMDB	Human Metabolome Database
ID	Primary identifier (of a database)
InChI	International Chemical Identifier
KEGG	Kyoto Encyclopedia of Genes and Genomes
LM	Lipidmaps
SMILES	Simplified molecular-input line-entry system

Discovery Algorithm **Resolve Algorithm**	This is the key algorithm that was developed as part of this thesis. It discovers different metabolome database records and links their primary keys together.
Metabolome database	A bioinformatics database containing metabolome compounds, their structure, chemical properties, onthology, pathways, and other related information
Local database **Local copy**	This refers to the database running on the user's computer.
External reference	This refers to database identifiers that refer to another database's primary key. Since these databases are technologically independent from each other, we can't call them foreign keys
Foreign key	This is used usually in the context of the local database. Since different metabolome records are kept in the same local database, their external references are treated as actual foreign keys.
Common dataframe interface	A generalized dataframe containing metabolome information, used by the discovery algorithm.

Please note that these abbreviations and terminology exist only within the context of this thesis and do not give an absolute definition within bioinformatics.

1. Introduction and Background

1.1. Metabolomics

Metabolomics is the study of small molecule substrates and compounds that take part in metabolic processes. Many leading causes of death can be traced to metabolic disorders, making it an especially important field of study. Metabolites are the substances occurring in metabolic pathways – either immediate products or the end results. As metabolites take the core role in metabolomics, it is in the interest of the scientific community to identify, label and reliably store these substances and compounds in metabolome databases (Johnson and Lange 2015).

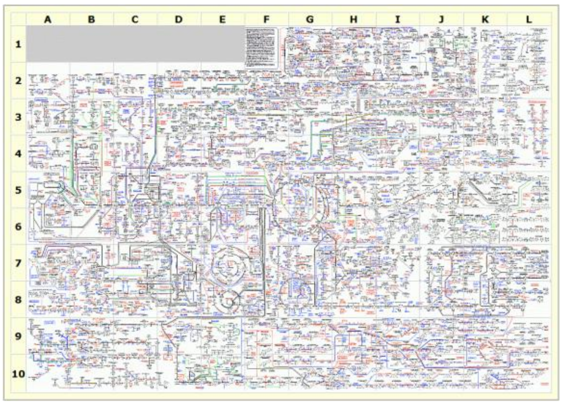


Figure 1: Human Metabolome Map. Pathway 1b, illustrating the complexity of metabolic pathways. Source: Credit: Victoria University, Au

The ones incorporated in this thesis were the following:

Human Metabolome Database (HMDB): a Canadian database focusing on human metabolism. (Wishart et al. 2018)

Chemical Entities of Biological Interest (ChEBI): EMBL-EBI's database consisting 'small molecular entities' [REF: user manual] that are involved in the processes of living organisms, the definition includes metabolome compounds as well.
(Hastings et al. 2016)

PubChem: a massive database from NCBI of approximately 103M chemical compounds. (Kim et al. 2019)

Kyoto Encyclopedia of Genes and Genomes (KEGG): stores many types of data as well, including compounds as well.
(Kanehisa et al. 2019)

Lipidomics Gateway (LipidMaps): is a database containing lipids sponsored by the Wellcome Trust.
(Fahy et al. 2009)

1.2. Problems with database referencing

Metabolome databases are relatively novel, and therefore not as mature as the ones that focus on storing genetic data, consequently, these databases propose several problems when it comes to accessing them for scientific research.

Different records - that refer to the same metabolite - in different databases support linking each other (we refer to this as external references or external identifiers in this thesis). However, the databases are not as interconnected with each other as ‘classical’ bioinformatics databases. Below are some illustrative examples for typical issues that may arise when handling more than one database. Please note that the primary identifiers for a database are marked with underscore.

i) The default case is when a metabolite record exists in both databases and this association is properly linked by external reference keys:

HMDB

hmdb_id	metabolite	chebi_id
HMDB0015405	fenoterol	**149226**

ChEBI

chebi_id	metabolite	hmdb_id
149226	fenoterol	**HMDB0015405**

Fenoterol is a commonly occurring substance, and therefore it is found and properly linked in both databases.

ii) In many cases metabolome records do not refer to other, foreign records, because simply such entries do not exist:

HMDB

hmdb_id	metabolite	chebi_id
HMDB0029572	Diethyl disulfide	*NULL*

ChEBI

chebi_id	metabolite	hmdb_id
-	-	-

Diethyl disulphide is an expected metabolite by HMDB, which indicates that it has not yet been detected and quantified. For this reason this metabolite record only exists in HMDB.

iii) In other cases the same metabolite does exist in multiple databases, but this relationship is not stored due to a lack of awareness:

HMDB

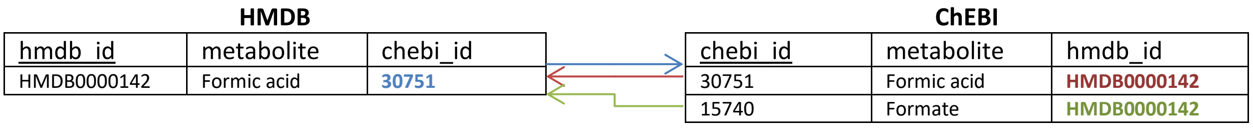
hmdb_id	metabolite	chebi_id
HMDB0002656	prostaglandin A1	**15545**

ChEBI

chebi_id	metabolite	hmdb_id
15545	prostaglandin A1	*NULL*

In this example, if we only had access to the ChEBI record then we are left unaware of the HMDB record’s existence.

iv) Sometimes external references do exist both ways, but they are incorrect and refer to the wrong metabolite, for the reason that different databases have divergent definitions of what a metabolite is:



HMDB

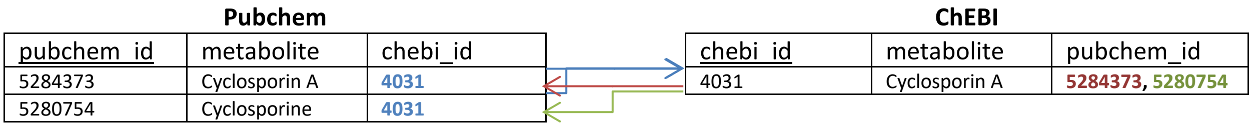
hmdb_id	metabolite	chebi_id
HMDB0000142	Formic acid	**30751**

ChEBI

chebi_id	metabolite	hmdb_id
30751	Formic acid	**HMDB0000142**
15740	Formate	**HMDB0000142**

Formic acid is the simplest carboxylic acid with a formula of CH_2O_2. Anions derived from formic acids are called formates with the formula CHO_2. While both ChEBI and HMDB store several formate types, the record titled 'formate' is only found in ChEBI and references HMDB's formic acid. While this association makes sense, it is incorrect as it assumes the two compounds to be chemically equivalent.

v) In a similar manner to *iv)* one entry in a database can also refer to multiple entries in another one (we call this a one to many relationship):



Pubchem

pubchem_id	metabolite	chebi_id
5284373	Cyclosporin A	**4031**
5280754	Cyclosporine	**4031**

ChEBI

chebi_id	metabolite	pubchem_id
4031	Cyclosporin A	**5284373**, **5280754**

In this case Cyclosporin A in ChEBI additionally references Cyclosporin A and Cyclosporine in Pubchem. Often such relationships simply refer to a secondary ID to the same metabolite, but in this particular case this is a true one to many relationship.

All of these scenarios may also happen for various reasons. For instance, different databases may have very little overlap of metabolites, because they differ in what metabolites they choose to store (e.g. LipidMaps store lipids while HMDB stores Human Metabolites, but both definition prescribe chemical compounds).

1.3. Other problems with databases

To make things worse, each database uses its own way of providing search, access and download features for researchers. Generally speaking the more databases we use in a research project, the more cumbersome the initial phase becomes, in which we merely develop scripts to access and parse metabolomics data.

Some databases - like PubChem - have a great overlap with other databases, because they store a vast amount of compounds relative to other databases. Databases like HMDB and ChEBI contain alternative primary identifiers for the same metabolite record – these are referred to as secondary IDs in this document.

1.4. Structure formats

Representing genetic data in a digital format is a relatively trivial task; nucleotides or amino acids in a sequential nature fit into many types of databases and file formats. Molecular structure, on the other hand comes with various challenges. If not handled correctly, a parser - an algorithm that parses molecule data into a digital format - may run into an endless cycle when reading ring groups, as these can not be read in a sequential way trivially. When designing chemical formats, we have to decide which of the manifold chemical properties we choose to acknowledge and how they relate to the structure of atoms. Such properties include, but are not limited to atoms, bond types, rings and aromaticity, stereochemistry, isotopes.

SMILES (Weininger 1988) and InChI (Dashti et al. 2017) (S. R. Heller et al. 2015) are two data formats that describe chemical structures with ASCII characters. It'd be intuitive to assume that the same metabolite always gets the same SMILES or InChI string, but this is not the case.

For example the Betulinic acid has these SMILES values in different databases:

Chebi (3087):
[H][C@]12CC[C@]3([H])[C@@]4(C)CC[C@H](O)C(C)(C)[C@]4([H])CC[C@@]3(C)[C@]1(C)CC[C@]1(CC[C@@H](C(C)
=C)[C@]21[H])C(O)=O

HMDB (HMDB0030094):
[H][C@]12[C@@H](CC[C@@]1(CC[C@]1(C)[C@]2([H])CC[C@]2([H])[C@@]3(C)CC[C@H](O)C(C)(C)[C@]3([H])CC[C
@@]12C)C(O)=O)C(C)=C

LipidMaps (LMPR0106140004):
C1[C@@]2(C)[C@@]([H])(CC[C@]3(C)[C@]2([H])CC[C@@]2([C@@]4([C@](CC[C@@]32C)(C(O)=O)CC[C@H]4C(C)=
C)[H])[H])C(C)(C)[C@@H](O)C1

PubChem (64971):
CC(=C)C1CCC2(C1C3CCC4C5(CCC(C(C5CCC4(C3(CC2)C)C)(C)C)O)C)C(=O)O,CC(=C)[C@@H]1CC[C@]2([C@H]1[C@H]3
CC[C@@H]4[C@]5(CC[C@@H](C([C@@H]5CC[C@]4([C@@]3(CC2)C)C)(C)C)O)C)C(=O)O

This is because SMILES is non-unique – meaning that the same compound can be represented with various strings. In comparison, for all four databases the InChI value is the same for Betulinic acid:

1S/C30H48O3/c1-18(2)19-10-15-30(25(32)33)17-16-28(6)20(24(19)30)8-9-22-27(5)13-12-23(31)26(3,4)21(27)11-
14-29(22,28)7/h19-24,31H,1,8-17H2,2-7H3,(H,32,33)/t19-,20+,21-,22+,23-,24+,27-,28+,29+,30-/m0/s1

However, InChI can also incorporate differences in isotopes, charges, stereochemical layer and other chemical properties as well. This makes InChI strings belonging to the same compound differ, such as in the case of Neohesperidin:

Chebi (59016):
1S/C28H34O15/c1-10-21(33)23(35)25(37)27(39-10)43-26-24(36)22(34)19(9-29)42-28(26)40-12-6-14(31)20-15(32)8-17(41-18(20)7-12)11-3-4-16(38-2)13(30)5-11**/h3-7,10,17,19,21-31,33-37H,8-9H2,1-2H3/t10-,17-,19+,21-,22+,23+,24-,25+,26+,27-,28+/m0/s1**

HMDB (HMDB0030748):
1S/C28H34O15/c1-10-21(33)23(35)25(37)27(39-10)43-26-24(36)22(34)19(9-29)42-28(26)40-12-6-14(31)20-15(32)8-17(41-18(20)7-12)11-3-4-16(38-2)13(30)5-11**/h3-7,10,17,19,21-31,33-37H,8-9H2,1-2H3**

LipidMaps (LMPK12140452):
1S/C28H34O15/c1-10-21(33)23(35)25(37)27(39-10)43-26-24(36)22(34)19(9-29)42-28(26)40-12-6-14(31)20-15(32)8-17(41-18(20)7-12)11-3-4-16(38-2)13(30)5-11**/h3-7,10,17,19,21-31,33-37H,8-9H2,1-2H3/t10?,17-,19?,21?,22?,23?,24?,25?,26?,27?,28?/m0/s1**

PubChem (442439):
1S/C28H34O15/c1-10-21(33)23(35)25(37)27(39-10)43-26-24(36)22(34)19(9-29)42-28(26)40-12-6-14(31)20-15(32)8-17(41-18(20)7-12)11-3-4-16(38-2)13(30)5-11**/h3-7,10,17,19,21-31,33-37H,8-9H2,1-2H3/t10-,17-,19+,21-,22+,23+,24-,25+,26+,27-,28+/m0/s1**

While these string formats are great tools for smooth structural representation and provide the possibility to execute structure-based searches in databases, they fail to label metabolites (or any chemical compound for that matter) in an unique and unambiguous manner.

This latter issue could be solved by taking one database's primary identifier as a starting point for reference. This identifier could then be treated as a de-facto standard for labelling novel and known metabolites, making it almost if not mandatory to store the identifier in all databases. When it comes to proteins, PDB identifiers (Berman et al. 2002) are treated in this manner to such an extent that PDB IDs can be reliably used in white papers or everyday conversations and has a general support in bioinformatics applications as well. To the best of our knowledge, such ID system for metabolites is yet to exist, the next best thing would be pubchem's primary IDs. All of these issues are making data preparation more difficult, discouraging progress.

2. Methods

2.1. Pre-parsing the data

To get a more complete understanding of the underlying data for each database, various scripts were created that parse through the downloaded database formats, exploring the attributes and their values. Since these databases store similar data, the same metabolomic and chemical attributes can be found in them. Therefore, we defined which commonly occurring attributes are queried and stored by our package. These are chemical name (referred to as names or synonyms), chemical formula, SMILES and INCHI strings, other database IDs, average mass of the compound. The full list is the following:

Database identifiers:
- chebi_id
- hmdb_id
- lipidmaps_id
- kegg_id
- pubchem_id

Other attributes:
- inchi
- inchikey
- smiles
- names
- formula
- mass
- monoisotopic_mass

These scripts also measured cardinality of attributes. Cardinality means the maximum number of occurrence of unique values per attribute. For example cardinality of 1 means that every database record is a scalar value, this is the ideal case. The table below shows the number of records that had cardinality larger than 1, for each attribute:

	hmdb	**chebi**	**pubchem**	**kegg**	**lipidmap**
synonyms	-	34346	795	1900	-
formula	0	668	0	0	0
smiles	0	0	800	-	0
inchi/key	0	1	0	-	0
Hmdb id	0	8	0	-	0
Chebi id	0	0	4	400	0
Pubchem id	0	2	0	0	0
Kegg id	0	109	0	0	0
Lipidmap id	0	13	-	23	0

Note: Since I do not have direct access to the entire database of Kegg and Pubchem, I took a random sample of 4000 records and based the statistics on them

This table is important because attributes with multiple possible values should be regarded as arrays. However, attributes where most records have a cardinality over 1 can be represented as scalar values in the database, while arrays can be stored in an optional extra column.

2.2. Internal Database

The main component of the package is the local database. The package gathers information from external metabolome databases, then stores it in the local one. For records that are not present in the database, an HTTP call is issued towards the API endpoint of the external databases. For this reason, the database also acts as a cache for api calls, serving as a way to reduce spamming external databases. The algorithm will not query the api if the underlying record is already found in the local database.

Each external database is stored locally in a single table. As the information stored varies greatly between different databases, we created R handler classes, whose responsibility is to convert the specific table format to a common dataframe interface. Dataframes are complex data structures in R, that contain tabular data.

The handler classes were designed to be modular and thus easily extensible. The discovery algorithm does not deal with implementation details of the individual databases, and therefore adding support for a new metabolomics database comes with relatively low development overhead as long as the database supports bulk downloading or a web api.

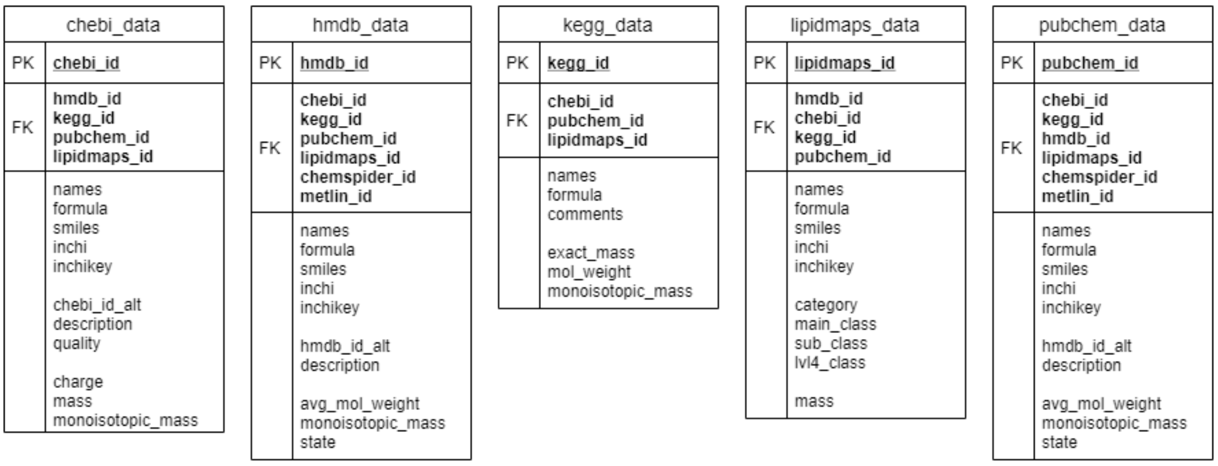

chebi_data
PK chebi_id
FK hmdb_id kegg_id pubchem_id lipidmaps_id
names formula smiles inchi inchikey
chebi_id_alt description quality
charge mass monoisotopic_mass
hmdb_data
PK hmdb_id
FK chebi_id kegg_id pubchem_id lipidmaps_id chemspider_id metlin_id
names formula smiles inchi inchikey
hmdb_id_alt description
avg_mol_weight monoisotopic_mass state
kegg_data
PK kegg_id
FK chebi_id pubchem_id lipidmaps_id
names formula comments
exact_mass mol_weight monoisotopic_mass
lipidmaps_data
PK lipidmaps_id
FK hmdb_id chebi_id kegg_id pubchem_id
names formula smiles inchi inchikey
category main_class sub_class lvl4_class
mass
pubchem_data
PK pubchem_id
FK chebi_id kegg_id hmdb_id lipidmaps_id chemspider_id metlin_id
names formula smiles inchi inchikey
hmdb_id_alt description
avg_mol_weight monoisotopic_mass state

Figure 2. ER diagram of the local database. Each table represents an external metabolome database and they keep links of each other as foreign keys.

2.3. Bulk insertions

The underlying algorithm used in the package relies on API fetches and a local database acting as a cache. However, fetching previously undiscovered records in bulk slows down the execution time noticeably. To account for this problem the package provides a possibility to download all records from a remote database into the local cache.

HMDB, ChEBI and Lipidmaps provide a possibility to download their entire database consisting 114k, 103k and 44k records, approximately as of today. The user starts using the package by downloading these three databases entirely to the local cache. Each database has its own bulk output format which had to be accounted for when this feature was written. For example HMDB's file format is a large file that contains XML records separated with a newline, while ChEBI and Lipidmaps store their export files in SDF (SDF Toolkit n.d.).

KEGG does not support bulk database download for non-subscribing users. Pubchem does provide a possibility to download in bulk, however there are 103 million records in its database. Requiring users of the package to have disk space for such amount of records is doubtful. Therefore, when resolving Pubchem and KEGG records, the database entirely relies on their public APIs.

Running the bulk insertions is a required step for users, as the current version of the package does not support API fetching for databases that have bulk insertion option.

2.4. The discovery algorithm

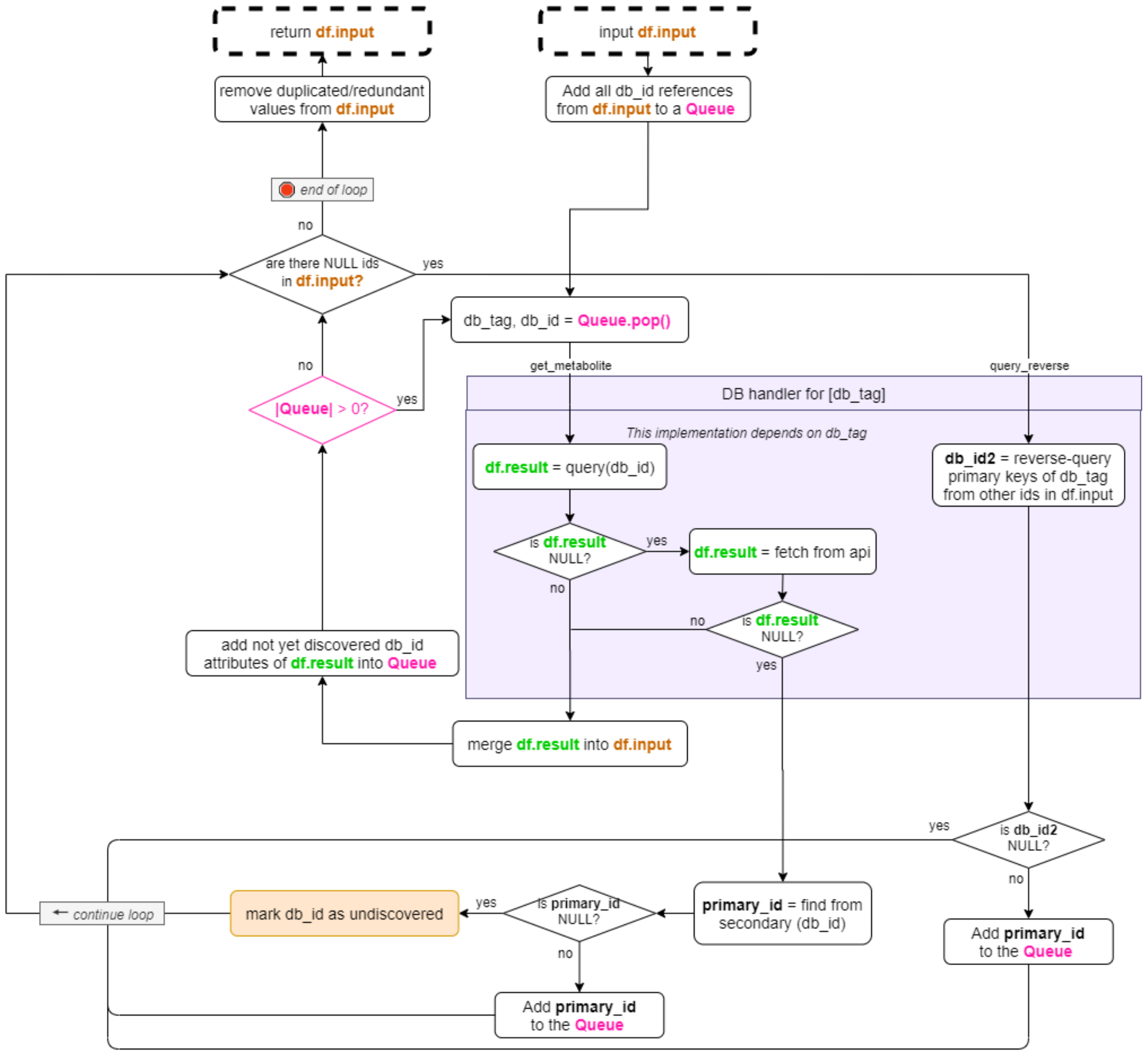
The algorithm resolves IDs by fetching the appropriate record and scanning it for additional database IDs. This is orchestrated via a queue-based algorithm that puts discovered IDs one by one and fetches the record on the top of the queue. The algorithm guarantees finding all relevant IDs by keeping track of already discovered records and not putting them into the queue twice.

The algorithm also supports 'reverse-querying' records. This means that instead of querying tables by their primary identifiers (e.g. hmdb_id for HMDB's local table), all the other foreign keys are used:

```
SELECT chebi_id FROM chebi_data WHERE hmdb_id = '…'
```

Reverse-queries are only run at the end of the discovery algorithm in the case of missing database identifiers.

The algorithm also supports resolving secondary IDs – these identifiers are redundant, and point to another primary identifier in the database. These can occur when a record is merged into another one, linking one record's primary ID to that of the other record's. This is implemented using an extra table that keeps track of secondary ID -> primary ID relationships.



return df.input
input df.input
remove duplicated/redundant values from df.input
Add all db_id references from df.input to a Queue
end of loop
no
are there NULL ids in df.input?
yes
db_tag, db_id = Queue.pop()
no
yes
|Queue| > 0?
get_metabolite
query_reverse
DB handler for [db_tag]
This implementation depends on db_tag
df.result = query(db_id)
db_id2 = reverse-query primary keys of db_tag from other ids in df.input
is df.result NULL?
yes
df.result = fetch from api
no
no
is df.result NULL?
yes
add not yet discovered db_id attributes of df.result into Queue
merge df.result into df.input
yes
is db_id2 NULL?
no
continue loop
mark db_id as undiscovered
yes
is primary_id NULL?
primary_id = find from secondary (db_id)
Add primary_id to the Queue
no
Add primary_id to the Queue

Figure 3. flow diagram of the discovery algorithm. The grey box

The grey box in Figure 3 represents a generalized implementation of how the handler classes deal with the local metabolome tables. The initial dataframe provided by the user (**df.input**) is continuously updated during the execuation of the algorithm. The area in grey marks the individual database handler classes which deal with the implementation of querying and reverse-querying from the local database, and fetching from the database's api. The dataframe returned by the handler class (**df.result**) is merged into the main dataframe (**df.input**) and the loop continue execution until the queue is empty. At the very end of the queue, the reverse queries execute. Once the algorithm stops, the initial dataframe is returned, updated with its missing values.

2.5. Usage

Below is an example use-case of the package to highlight its benefits due to ease of data access. For example, the user can rely on a CSV to resolve missing IDs the following way:

```
# discovery.csv:
hmdb_id,chebi_id,inchi,mass
,8337,,
HMDB0001008,,,
```

By loading this csv into a dataframe, the user then calls the *resolve_metabolites* function with the dataframe as the input parameter:

```
df.res <- read.csv("discovery.csv", stringsAsFactors=FALSE)
resp <- resolve_metabolites(df.res)
```

The output will be a list containing the filled dataframe, and sets of unresolved and ambigous cases. For more information, please check the manual of the package.

```
# resulting dataframe:
df.out <- revert_df(resp$df)
```

Resolving without an input dataframe:
There's an additional, simplified interface for the package. By calling "resolve_single_id" the user only has to provide one database ID to start the discovery algorithm from. This interface is encouraged for simpler use-cases. The function's output is the same.

```
# simplified interface:
resp <- resolve_single_id('hmdb_id', 'HMDB0035495')

# resulting dataframe:
df.out <- revert_df(resp$df)
```

3. Results

To get an idea of how the algorithm performs over the collected data, we conducted a coverage test. This test begins by randomly sampling 7500 identifiers from the local database (HMDB, ChEBI and Lipidmaps). 1500 HMDB, 500 pubchem and 500 kegg identifiers are collected from the local HMDB database, and the other two databases are sampled in a similar way. These identifiers are sequential, but at each run they start at a random location within the database. This is done in order to maximize a fair representation of each database's identifiers and also to see how consistent the test is with the randomized input.

Once the 7500 identifiers are collected, the test is launches the discovery algorithm for each individual ID. Next, the results of the algorithm are evaluated into the following three categories:

1. **Consistent database ID or attribute:** the algorithm managed to query this attribute which contains one and only one value. In other words, every attribute that is stored as a scalar within the dataframe is labelled as *consistent.*

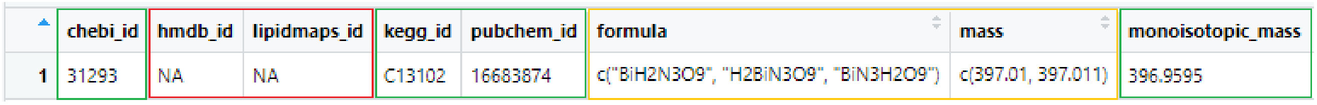
   This is the ideal outcome for all identifiers, as it proposes no additional task of sorting for the end user.

2. **Ambiguous database ID or attribute:** while the algorithm managed to query this attribute, more than one alternative values were found. This presents a problem for the users as it is up to them to identify the cause for having alternative values for the same attribute.

   In this scenario the end user has to manually figure out why certain identifiers contain multiple values. In many cases there are no actual ambiguity. For example, for the attributes of 'mass' and 'monoisotopic mass' the exact same mass can be interpreted as multiple values due to rounding errors.

3. **Missing database ID or attribute:** in some cases no values are found for an attribute.

   This scenario is problematic; the algorithm failed to match any database identifiers for the metabolite.



	chebi_id	hmdb_id	lipidmaps_id	kegg_id	pubchem_id	formula	mass	monoisotopic_mass
1	31293	NA	NA	C13102	16683874	c("BiH2N3O9", "H2BiN3O9", "BiN3H2O9")	c(397.01, 397.011)	396.9595

Figure 4. illustrating labels within the coverage test. Green, red and yellow cells indicate consistent, missing and ambiguous attributes, respectively

The coverage test results in a cumulative percentage over all there categories, for all the attributes. This test also generates a cumulative CSV file containing all resolved dataframes. The files generated are attached as a supplement.

We have conducted randomized 4 runs and they have shown similar results:

Run #1:

	hmdb id	chebi id	pubchem id	kegg id	lipidmaps id
consistent:	53.60%	50.47%	73.21%	23.77%	27.20%
ambiguous:	2.33%	4.79%	5.75%	4.04%	1.23%
missing:	44.07%	44.75%	21.04%	72.19%	71.57%

Run #2:

	hmdb id	chebi id	pubchem id	kegg id	lipidmaps id
consistent:	45.03%	51.11%	61.45%	21.31%	25.39%
ambiguous:	3.85%	4.19%	9.69%	3.72%	5.83%
missing:	51.12%	44.71%	28.85%	74.97%	68.79%

Run #3:

	hmdb id	chebi id	pubchem id	kegg id	lipidmaps id
consistent:	40.29%	56.65%	83.01%	26.35%	31.73%
ambiguous:	6.25%	5.97%	10.29%	5.43%	1.33%
missing:	53.45%	37.37%	6.69%	68.23%	66.93%

Run #4:

	hmdb id	chebi id	pubchem id	kegg id	lipidmaps id
consistent:	40.00%	57.01%	81.16%	24.76%	31.43%
ambiguous:	7.12%	6.13%	10.36%	6.11%	1.49%
missing:	52.88%	36.85%	8.48%	69.13%	67.08%

Similarly, here are the results for non-database identifiers:

Run #1

	name(s)	inchi	smiles	formula	mass	monoiso. mass
consistent:	20.47%	89.12%	21.15%	92.79%	26.71%	38.57%
ambiguous:	78.56%	9.39%	78.00%	6.63%	72.63%	60.72%
missing:	0.57%	1.49%	0.85%	0.59%	0.67%	0.71%

Run #2

	name(s)	inchi	smiles	formula	mass	monoiso. mass
consistent:	23.44%	72.40%	24.95%	75.24%	27.11%	34.44%
ambiguous:	65.17%	12.23%	63.56%	13.37%	61.44%	53.88%
missing:	11.39%	15.37%	11.49%	11.39%	11.45%	11.68%

Run #3

	name(s)	inchi	smiles	formula	mass	monoiso. mass
consistent:	6.20%	80.36%	21.55%	86.57%	12.17%	29.41%
ambiguous:	93.36%	17.20%	77.83%	12.97%	87.25%	69.79%
missing:	0.44%	2.44%	0.63%	0.45%	0.57%	0.80%

Run #4

	name(s)	inchi	smiles	formula	mass	monoiso. mass
consistent:	8.95%	82.61%	12.07%	73.73%	15.88%	26.36%
ambiguous:	90.64%	15.08%	86.92%	25.84%	83.51%	71.23%
missing:	0.41%	2.31%	1.01%	0.43%	0.61%	2.41%

The cumulative result of the same test is shown here:

Run #1-4 identifiers:

	hmdb id	chebi id	pubchem id	kegg id	lipidmaps id
consistent:	44.73%	53.81%	74.71%	24.05%	28.94%
ambiguous:	4.89%	5.27%	9.02%	4.82%	2.47%
missing:	50.38%	40.92%	16.27%	71.13%	68.59%

Run #1-4 attributes:

	name(s)	inchi	smiles	formula	mass	monoiso. mass
consistent:	14.76%	81.12%	19.93%	82.08%	20.47%	32.20%
ambiguous:	81.93%	13.47%	76.58%	14.70%	76.21%	63.90%
missing:	3.20%	5.40%	3.50%	3.21%	3.33%	3.90%

4. Discussion

Metabolomics databases are inconsistent because they have different rules of categorizing metabolites and often fail to reference external records correctly. In this thesis we created a package that helps associating metabolite records with their correct counterparts in various databases, but it does not help the users resolve ambiguous cases or finding IDs that are missing from all relevant databases.

As demonstrated in the coverage test, database identifiers had varying results. Pubchem have a larger coverage of other databases due to its immense number of records – 3 folds more records than Chebi or HMDB, for example. Lipidmaps have a smaller coverage, because it focuses on a smaller subset of compounds.

Inchi’s coverage was also notable and it shows a great candidate for a universal compound or chemical identifier.

4.1. Future Work

We could improve the efficiency of our package greatly, but that wouldn’t solve the underlying issues in metabolome databases. Defining a universal metabolite identifier could bring a common reference point regardless of where we get our data from and how we label it. One way would be treating a structural data – such as InChI - format as the identifier itself. As we showed, algorithmically or manually identifying a structure as a string may still propose ambiguous edge cases. Another semi-automated way could be taking one database’s primary identifier as a de-facto metabolite ID, analogous to PDB’s ID as protein identifier. For this to work, the database must contain or at least reference all other major database’s entries as well. In our opinion PubChem is the next best candidate for such a universal identifier, however even with its marvellous 103 million entries, it is far from containing a complete set of metabolites.

We may visit the possibility of creating a new, derivative database whose sole purpose is to keep track of metabolite databases, their entries, and assigning an unambiguous ID to settle this obstacle.

4.2. Acknowledgement

I would like to thank the work during this thesis to Klev Diamanti, Sara Younes, Jan Komorowski and his laboratory. The underlying idea behind this package is credited to Klev, and the work was made possible thanks to Sara’s careful supervision and technical assistance.

References

Berman, Helen M. et al. 2002. “The Protein Data Bank.” *Acta Crystallographica Section D: Biological Crystallography* 58(6 I): 899–907.

Dashti, Hesam, William M. Westler, John L. Markley, and Hamid R. Eghbalnia. 2017. “Unique Identifiers for Small Molecules Enable Rigorous Labeling of Their Atoms.” *Scientific Data* 4(1): 1–9.

Fahy, Eoin et al. 2009. “Update of the LIPID MAPS Comprehensive Classifica-Tion System for Lipids.” *J. Lipid Res*. www.lipidmaps.org. (June 3, 2020).

Hastings, Janna et al. 2016. “ChEBI in 2016: Improved Services and an Expanding Collection of Metabolites.” *Nucleic Acids Research* 44(D1): D1214–19. http://www.ebi.ac.uk/chebi/ (June 3, 2020).

Heller, Stephen R. et al. 2015. “InChI, the IUPAC International Chemical Identifier.” *Journal of Cheminformatics* 7(1).

Johnson, Sean R., and Bernd Markus Lange. 2015. “Open-Access Metabolomics Databases for Natural Product Research: Present Capabilities and Future Potential.” *Frontiers in Bioengineering and Biotechnology* 3(MAR).

Kanehisa, Minoru et al. 2019. “New Approach for Understanding Genome Variations in KEGG.” *Nucleic Acids Research* 47. https://www.kegg.jp/ (June 3, 2020).

Kim, Sunghwan et al. 2019. “PubChem 2019 Update: Improved Access to Chemical Data.” *Nucleic Acids Research* 47(D1): D1102–9.

“SDF Toolkit.” https://cactus.nci.nih.gov/SDF_toolkit/ (June 3, 2020).

Weininger, David. 1988. “SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules.” *Journal of Chemical Information and Computer Sciences* 28(1): 31–36.

Wishart, David S et al. 2018. “HMDB 4.0: The Human Metabolome Database for 2018.” *Nucleic Acids Research* 46(D1): D608–17. www.nmrml.org (June 3, 2020).