

Metabolomics database resolver

Rajmund Csombordi

Supervisor: Sara Yones

Subject Reader: Johan Viklund

# Abstract

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. Metabolites are studied and labelled, and can be accessed by various means of searching. 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.

For this reason 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 with one database’s identifier.

Degree project in bioinformatics, 2016  
Examensarbete i bioinformatik 30 hp till masterexamen, 2016   
Biology Education Centre and Biomedical Center  
Supervisor: Sara Yones

Table of Contents

[Abstract 4](#_Toc40512996)

[Terminology and abbreviations 8](#_Toc40512997)

[Introduction and Background 9](#_Toc40512998)

[Metabolomics 9](#_Toc40512999)

[Problems with databases 10](#_Toc40513000)

[The package 13](#_Toc40513001)

[Methods 14](#_Toc40513002)

[Internal Database 15](#_Toc40513003)

[Data sources 15](#_Toc40513004)

[Bulk insertions 16](#_Toc40513005)

[Theory 17](#_Toc40513006)

[The discovery algorithm 17](#_Toc40513007)

[Results 19](#_Toc40513008)

[Installation 19](#_Toc40513009)

[Usage 19](#_Toc40513010)

[Resolving without an input dataframe: 20](#_Toc40513011)

[Coverage tests 20](#_Toc40513012)

[About the collected data **Error! Bookmark not defined.**](#_Toc40513013)

[Issues with databases 21](#_Toc40513014)

[1.1.1 Materials and Methods 24](#_Toc40513015)

[1.1.2 Results and Discussion 24](#_Toc40513016)

[1.1.3 Acknowledgement 24](#_Toc40513017)

[1.1.4 References in the text 24](#_Toc40513018)

[1.1.5 Tables and Figures 25](#_Toc40513019)

[1.2 Other parts outside the main text 25](#_Toc40513020)

[1.3 Layout 25](#_Toc40513021)

[2 Concluding part: References and appendix **Error! Bookmark not defined.**](#_Toc40513022)

[2.1 References **Error! Bookmark not defined.**](#_Toc40513023)

[2.2 Appendix **Error! Bookmark not defined.**](#_Toc40513024)

[3 Concerning writing **Error! Bookmark not defined.**](#_Toc40513025)

[4 What is required for an approved final report? **Error! Bookmark not defined.**](#_Toc40513026)

[X-heading 1 introductory part: Arial normal style 18 points, line spacing 1.15, 36 points before and 12 after the paragraph; not numbered in body **Error! Bookmark not defined.**](#_Toc40513027)

[X-heading 2: Arial normal style 14 points; line spacing 1.15; 18 points before and 12 after the paragraph; numbered **Error! Bookmark not defined.**](#_Toc40513028)

[X-heading 3: Arial bold style 12 points; line spacing 1.15; 10 points before and 0 after the paragraph; numbered **Error! Bookmark not defined.**](#_Toc40513029)

# 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.*

# Introduction and Background

## Metabolomics

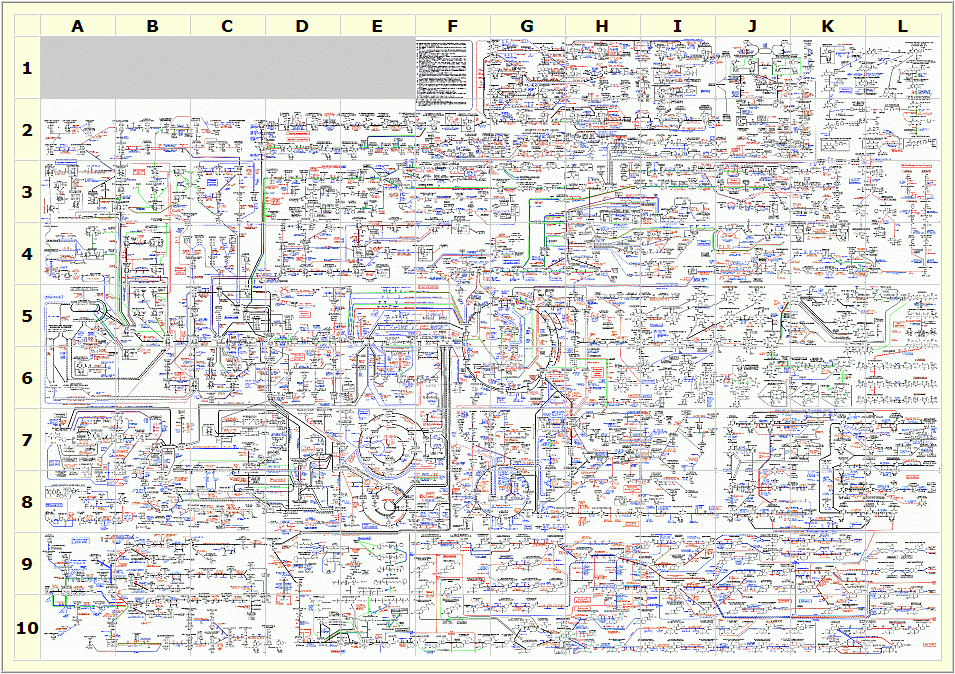
Metabolomics is the study of small molecule substrates and compounds that take part in metabolic processes. Many leading causes of death can be traced to metabolic disorders, making it an especially important field of study. Metabolites are the substances occurring in metabolic pathways – either immediate 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.

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)** [REF]**:** a Canadian database focusing on human metabolism.

**Chemical Entities of Biological Interest (ChEBI)** [REF]**: 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.

**PubChem** [REF]is a massive database from NCBI of approximately 103M chemical compounds.

**Kyoto Encyclopedia of Genes and Genomes** (KEGG) [REF] stores many types of data as well, including compounds as well.

**Lipidomics Gateway (LipidMaps)** [REF] is a database containing lipids sponsored by the Wellcome Trust.

## Problems with database referencing

As they are not as aged and mature as the ones that for example revolve around storing genetic data, metabolome databases propose several problems when it comes to using them for scientific research. While these databases support external references between each other, they 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** | | |  | **ChEBI** | | |
| hmdb\_id | metabolite | chebi\_id |  | chebi\_id | metabolite | hmdb\_id |
| HMDB0015405 | fenoterol | **149226** | 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** | | |  | **ChEBI** | | |
| hmdb\_id | metabolite | chebi\_id |  | chebi\_id | metabolite | hmdb\_id |
| ?????? | ???? | *NULL* | - | - | - |

In this example, ChEBI does contain information about ???.

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** | | |  | **ChEBI** | | |
| hmdb\_id | metabolite | chebi\_id |  | chebi\_id | metabolite | hmdb\_id |
| HMDB0002656 | prostaglandin A1 | **15545** | 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** | | |  | **ChEBI** | | |
| hmdb\_id | metabolite | chebi\_id |  | chebi\_id | metabolite | hmdb\_id |
| HMDB0000142 | Formic acid | **30751** | 30751 | Formic acid | **HMDB0000142** |
|  |  |  |  | 15740 | Formate | **HMDB0000142** |

Formic acid is the simplest carboxylic acid with a formula of CH2O2. Anions derived from formic acids are called formates with the formula CHO2. 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):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **HMDB** | | |  | **ChEBI** | | |
| hmdb\_id | metabolite | chebi\_id |  | chebi\_id | metabolite | hmdb\_id |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**TODO: BLA BLA BLA**

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 chose store (e.g. LipidMaps store lipids while HMDB stores Human Metabolites).

## 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, the more cumbersome the start stage is, in which we develop scripts for data parsing and storage. This type work shouldn’t be the responsibility of a scientist.

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 as secondary IDs in this document.

## Structure formats

Representing genetic data in a digital format is a fairly trivial task; nucleotides or amino acids in a sequential nature fits well with many types of databases and file formats. Molecular structure, on the other hand comes with various challenges [REF ?]. Amine groups propose a danger of reading the same atoms redundantly, or even pushing the parser algorithm into an endless cycle. When designing chemical formats, we have to decide which of the manifold chemical properties we chose to store and how they relate to atoms. Such properties include, but are not limited to atoms, bonds types, rings and aromaticity, stereochemistry, isotopes.

SMILES [REF] and InChI [REF] 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 dabatases:

**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]3CC[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 4 databases the InChI value is the same:

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 way scientist could trust that every metabolome database has a link to entries this primary database. When it comes to proteins, PDB identifiers [REF pdb] [REF pdb id] 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 and discourages progress.

TODO ITT

* Introduction ✓
* Background ✓
* Materials and methods ✓
  + Algorithm & stuff
  + Installation? Or not?
* Theory
  + alorithm
* >Results
  + >Coverage test
* Discussion & Conclusion
  + Dissing databases
* Acknowledgement

# Methods

## Preparations

Before we began work on the package, we discussed the problems regarding the databases and proposed various strategies to overcome them. An early prototype of the algorithm was made in python in order to get a view of the workflow and adjust changes quickly. This prototype would then serve as a guide for writing the R package. Additional python scripts were made to parse the bulk inserts and explore the data provided by the various databases.

## Pre-parsing the data

To get a more complete understanding of the underlying data for each database, I've created scripts to measure the cardinality of the attributes and external references of metabolite records.

The below table shows how many records in each database has a cardinality larger than 1. 0 means a cardinality of 1.

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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **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 |

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.

## Internal Database

The underlying component behind the package is the local database. The package gathers information from external metabolome databases, then store it in the local one. It 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 so called ‘common dataframe interface’. This dataframe is returned to the discovery algorithm and contains the same columns as the input dataframe (to understand this in detail, please refer to the Theory chapter).

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.

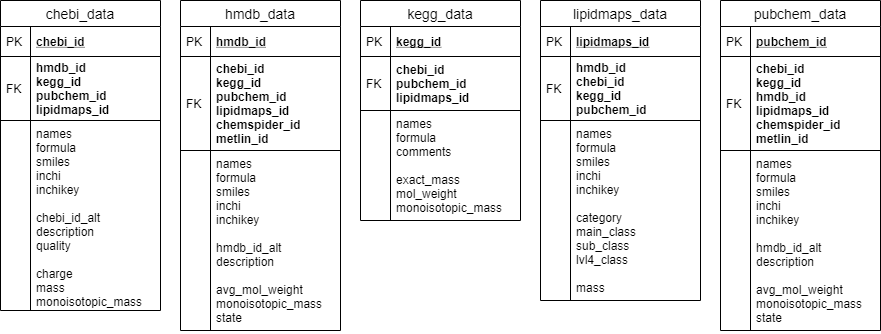


Figure 2. ER diagram of the local database. Each table represents an external metabolome database and they keep links of each other as foreign keys.

## Data sources

**Human Metabolome Database (HMDB)** stores approximately 114k metabolites found in the human body. They provide XML format for their web service and bulk download format.

**Chemical Entities of Biological Interest (ChEBI)** is EMBL-EBI’s database for small molecular entities that are involved in the processes of living organisms. They provide various formats for both web service and bulk downloading.

**PubChem** is a massive database of approximately 103M chemical compounds. Due to its size we rely entirely on its web service for accessing compounds.

**LipidMaps** is a database containing ~44k lipids. They provide bulk download possibility in SDF format.

**Kyoto Encyclopedia of Genes and Genomes** offers a compound database as well. KEGG does not support bulk download for free, so we rely on its web service access solely.

## Bulk insertions

The resolving algorithm relies on API fetches and a local database acting as a cache. However, fetching new records in bulks 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. The databases have to support mass downloading for this, therefore, this option is only available HMDB, ChEBI and Lipidmaps. Each database have its own bulk output format which had to be accounted for when this feature was written.

Running the bulk insertions scripts on my local computer took around 1 hour in total for all three databases. This step is required for users, as the current version of the package does not support API fetching for databases that have bulk insertion option.

# Theory

In this chapter we will elaborate on the details and demonstrate a proof of the correctness of the queue-based discovery algorithm.

## 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. After emptying the queue, the algorithm checks if there are external IDs that are null. Reverse-queries are then issued for each missing ID. A reverse-query means that the database is queried based on its foreign keys, for example:

SELECT chebi\_id FROM chebi\_data WHERE hmdb\_id = ’…’

The algorithm also supports resolving secondary IDs – IDs that do not have a record in the database, but instead these point to another ID. These can occur when a record is merged into another one, linking its primary ID to that of the other record’s.

Below is an overview flow chart that depicts the complete logic of the discovery algorithm.

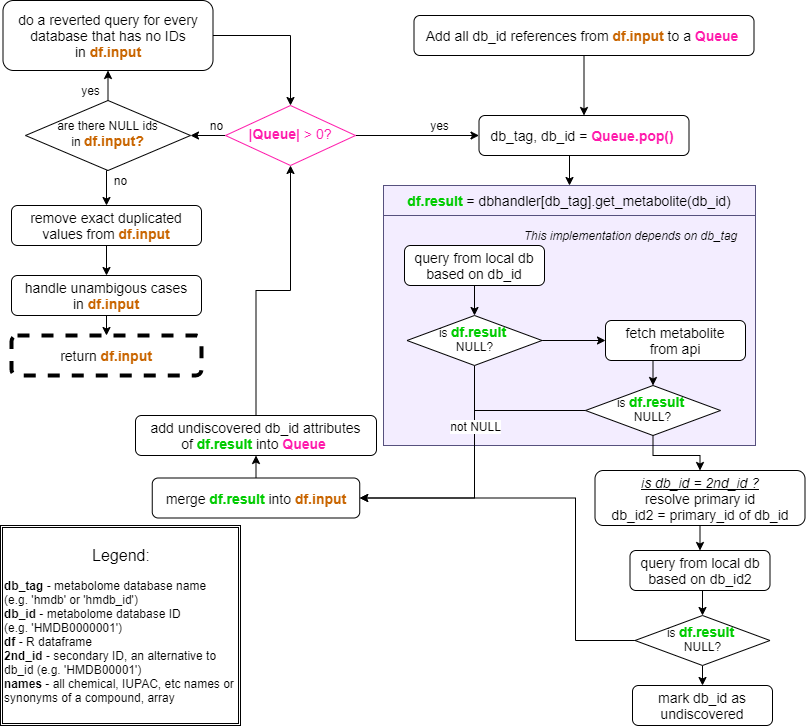


Figure 3. flow diagram of the discovery algorithm. The grey box

The grey box in Figure 3 represents a generalized implementation of how the handler classes deal with the local metabolome tables. Once a data is retrieved either by api or a local query it is converted into a common metabolome dataframe interface, depicted by **df.result** on the diagram.

The initial dataframe provided by the user is continuously updated with the latest query. It is marked with **df.input** on the diagram, as is eventually returned to the end user.

# Results

## Installation

**todo** @later

## Usage

The main feature of the package is resolving missing IDs in a user-prepared dataframe. The user can chose which columns to include in the dataframe, please keep in mind that only these columns will be used as the bases of the discovery algorithm. For example, if the user excludes ''pubchem\\_id'', then pubchem won't be queried during discovery. The possible column names are:

* chebi\_id
* hmdb\_id
* lipidmaps\_id
* kegg\_id
* pubchem\_id
* inchi
* inchikey
* smiles
* names
* formula
* mass
* monoisotopic\_mass

*Note: these attributes are also the columns for the common dataframe interface.*

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 the user can add it straight into the resolve function, given that only the listed columns are present.

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

resp <- resolve(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)

## Coverage tests

To get an idea of how the algorithm performs over the collected data, we conducted a coverage test. This test begins by randomly sampling a few thousand local IDs from HMDB, ChEBI and LipidMaps. Then the resolve algorithm is launched for each ID and the results are labelled into three categories:

* **Consistent database ID or attribute:** the algorithm managed to query this attribute and it contains one and only one value. In other words, every attribute that is stored as a scalar within the dataframe is labelled as *consistent.*
* **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.
* **Missing database ID or attribute:** in some cases no values are found for an attribute.

****

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

The end result of the test is calculating the fraction of all three labels (compared to the number of records in the test), for each attribute.

This test also generates a cumulative CSV file containing all resolved dataframes. The files generated during the thesis are attached as a supplement for this thesis.

**TODO: test results**

## Issues with databases

The issues with chemical structure formats

SMILES and InChI are two data formats that describe chemical structures with ASCII characters. It’d be intuitive to assume that the same metabolite gets the same SMILES or InChI, but this is not the case.

Perhaps visiting the possibility of using InChI strings (with the exclusion the optional format extensions after the “/” symbol) for chemical-based searches in metabolites would be beneficial both for this package and in the construction of a non-structure based metabolome identifier as well in the future.

By comparing all records in HMDB, ChEBI and LipidMaps we find out that **XX**% of records do …

**Show: records that have the same pubchem\_chebi id if their SMILES isn’t**

**%-ban kifejezve!**

## Future work

**@Todo:**

* Store pathway information +
* Meta ID
* Web api for a centralized database

## Acknowledgement

**TODO**

References

iupac, inchi, smiles

IBG. 2016a. Degree project - Disciplinary Domain of Science and Technology - Uppsala University, Sweden. WWW document 26 October 2016: [http://teknat.uu.se/student-en/programmes-courses/degree-project/](https://teknat.uu.se/student-en/programmes-courses/degree-project/). Accessed 26 October 2016.

IBG. 2016b. Reference management - Biology Education Centre - Uppsala University, Sweden. WWW document 4 May 2016: <http://www.ibg.uu.se/student-en/support-service/reference-management/>. Accessed 26 October 2016.

Rydin H, Carlson K, Berglund A, Svensson BG. 2014. Presenting Science. Biology Education Centre

TEKNAT. 2012. Guidelines and instructions for the degree project course within the technical programs at Uppsala University.

Appendix 1.

In the appendix, you can account for results that are too extensive to be included in the main report, but motivated to show. The appendix always starts on a new page and is provided as Appendix A, Appendix B, etc. Each appendix begins on a new page.

### Materials and Methods

After introducing the project, you describe how it was carried out. You must account for all the experiments you made and the basic rule is that it should be possible to repeat the experiments by reading the description. There are different traditions on how to describe experiments in different disciplines. Read Presenting Science (Rydin *et al.* 2014) and consult your supervisor or subject reader if you are unsure on how to describe the methods in your report.

### Results and Discussion

In the results section you write what you did and which results you got. Structure the results section in a way that makes the text easy to follow and understand. In the discussion section, put the results in a context and return to the questions you presented in the introduction. How does your work relate to previous research and development?

Also note whether there are other aspects of the work that may be of interest, but that may not always be obvious to people in the relevant field of research. Are there for example ethical aspects on what you have done, or what your work is aimed at? What role does your work have in society at large? Is your work a contribution to a more sustainable world? Mention relevant aspects appropriately in your report and make it more interesting for a wider readership. See Presenting Science (Rydin *et al.* 2014) for further instructions on how to write the results and discussion section.

### Acknowledgement

Sometimes there may be reasons to thank someone who contributed to the work. This is done at the end of the main body, before the references.

### References in the text

You need to provide references for all data in the report that is not your own, or commonly known. You can read about how to reference on the IBG's website (IBG 2016b) and in Presenting Science (Rydin *et al.* 2014). For Zotero there is a ready template for IBG's reference system to download from the IBG website (<https://www.ibg.uu.se/student-en/support-service/reference-management/>).

### Tables and Figures

Often it is convenient to describe methods, results and conclusions in the form of a figure or a table. Use, as far as possible, self-produced images and tables. Create your illustrations with software that is designed for it. If you use previously published material, you must obtain permission from the copyright holder. State this, and the source, last in the figure caption: "Illustration used with permission from ...". How to design tables and figures and how you refer to them in the text are described in detail in Presenting Science (Rydin *et al.* 2014). Make sure to refer to all figures and tables in the text, number them in the order they appear in the text, and insert them into the document in order. That is, Figure 1 is referenced to in the text before Figure 2 and also placed before Figure 2 in the document. An easy way to get this right is to use cross-references.

When referencing the figure in the text you use insert-cross-reference and select figure, and “only label and number” as within this parenthesis (Figure 1). If a figure is inserted before another in the text, just mark the entire text after inserting the figure and figure text, and update the field. Then all figure numbers and cross-references in the text will be updated.

## Other parts outside the main text

In addition to tables and figures it might be appropriate to use other elements in the report that do not belong to the text, for example equations, chemical reactions or programming code. Make sure to then follow the practice of the subject area in which you have done your work. Consult your supervisor or subject reader if you want advise.

## Layout

A report with good layout gives a good impression. The basis is that you use this template. Other things to consider are:

* make page breaks at appropriate places to minimize empty spaces in the report,
* gather the body text - avoid single lines of running text above or below tables and figures,
* keep figures and tables within the margins used for the text,
* make figures and tables as uniform as possible with respect to font, font size and colors,
* avoid naked headings, that is, when there is no text between two headings at different levels.