



## Article

# MetaFetchR: An R Package for Complete Mapping of Small-Compound Data

Sara A. Yones<sup>1,\*</sup> , Rajmund Csombordi<sup>1</sup>, Jan Komorowski<sup>1,2,3,4</sup> and Klev Diamanti<sup>1,5,\*</sup> 

<sup>1</sup> Department of Cellular and Molecular Biology, Uppsala University, 751 24 Uppsala, Sweden; rajmund.csombordi@gmail.com (R.C.); jan.komorowski@icm.uu.se (J.K.)

<sup>2</sup> Institute of Computer Science, Polish Academy of Sciences, 01-248 Warsaw, Poland

<sup>3</sup> Washington National Primate Research Center, Seattle, WA 98121, USA

<sup>4</sup> Swedish Collegium for Advanced Study, 752 38 Uppsala, Sweden

<sup>5</sup> Department of Immunology, Genetics and Pathology, Uppsala University, 751 85 Uppsala, Sweden

\* Correspondence: sara.younes@icm.uu.se (S.A.Y.); klev.diamanti@igp.uu.se (K.D.);

Tel.: +46-76-592-2512 (S.A.Y.); +46-73-926-7648 (K.D.)

**Abstract:** Small-compound databases contain a large amount of information for metabolites and metabolic pathways. However, the plethora of such databases and the redundancy of their information lead to major issues with analysis and standardization. A lack of preventive establishment of means of data access at the infant stages of a project might lead to mislabelled compounds, reduced statistical power, and large delays in delivery of results. We developed MetaFetchR, an open-source R package that links metabolite data from several small-compound databases, resolves inconsistencies, and covers a variety of use-cases of data fetching. We showed that the performance of MetaFetchR was superior to existing approaches and databases by benchmarking the performance of the algorithm in three independent case studies based on two published datasets.

**Keywords:** small-compound databases; metabolomics; metabolites; queue-based algorithm



**Citation:** Yones, S.A.; Csombordi, R.; Komorowski, J.; Diamanti, K. MetaFetchR: An R Package for Complete Mapping of Small-Compound Data. *Metabolites* **2021**, *11*, 743. <https://doi.org/10.3390/metabo11110743>

Academic Editor: Hunter N. B. Moseley

Received: 30 August 2021

Accepted: 27 October 2021

Published: 28 October 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Metabolomics allows the study of small-molecule substrates and compounds that are involved in metabolic processes. A small compound (<1500 Da) is a low-molecular-weight organic compound that is involved in or may regulate biological processes. Examples of small compounds include various sugars, lipids, and amino acids. Various complex diseases have been strongly linked to metabolic disorders, such as type 2 diabetes and cancer, making metabolomics a highly relevant field for single- and multi-omics studies [1–3]. Pathway enrichment analysis is a widespread analysis approach for metabolomics that requires metabolites to map a predefined set of unique identifiers [4]. In this setup there are several issues that arise when accessing, pre-processing, and analysing metabolite data. For instance, the overlapping and non-overlapping information for metabolites is scattered across several small-compound databases, leading to major analysis and standardization issues [5–7]. Additional challenges occur with databases that deliver data, which contain multiple entries for one metabolite or incomplete data. Finally, foreign reference identifiers may be missing, making it difficult, sometimes impossible, to find the link between two records of the same metabolite in different databases, while in other cases, the small fraction of reference identifiers that are present might lead to incorrect compounds. The aforementioned issues delay the delivery of results and more importantly, might lead to inconsistent or biased results.

Xia and colleagues developed MetaboAnalyst, which is a versatile computational tool for metabolomics. This tool contains a module aimed at mapping names to identifiers of metabolites from the human metabolome database (HMDB), the chemical entities of biological interest (ChEBI), the Kyoto encyclopedia of genes and genomes (KEGG), PubChem, and METLIN [5,7–11]. However, the lack of a shared nomenclature for metabolite

names commonly leads to numerous mismatches or no-matches. Additionally, Wishart and colleagues mapped compounds of HMDB to identifiers of other databases that suffer from inconsistent matches [5]. Moreover, the aforementioned tools map metabolite names to entries in HMDB that may lead to loss of information in case of a mismatch or absence of the metabolite from this specific database.

MetaFetcheR is a unified package targeted towards the metabolomics community, and it is able to resolve multiple inconsistencies and incompleteness in data fetching. The algorithm exhaustively resolves such inconsistent cases and leads to improved mapping of small compounds to identifiers. This is showcased in two case studies using two published datasets by Diamanti et al. and Priolo et al. [1,12] and three existing mappers MS\_targeted, MetaboAnalystR along with the webtool MetaboAnalyst 5.0, and Chemical Translation Service (CTS) [1,13–15].

## 2. Results

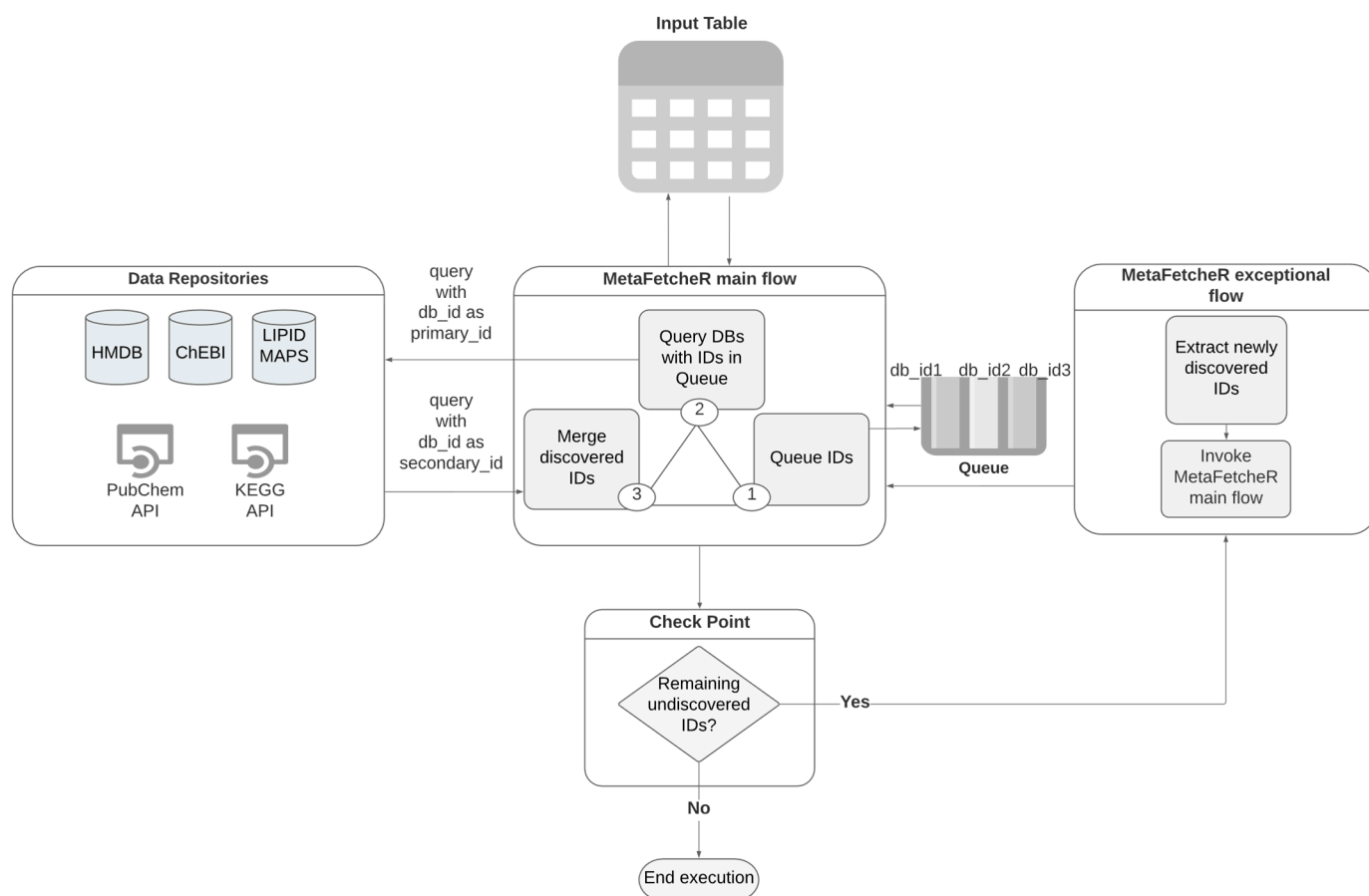
MetaFetcheR is an R package that uses the sparse input of primary database identifiers as a reference point to retrieve identifiers from other databases. The output of MetaFetcheR can be directly incorporated in analysis pipelines. The package unifies data from five open-access and widely used small-compound databases that include HMDB, ChEBI, PubChem, KEGG, and Lipidomics gateway (LIPID MAPS) [16]. Each database has a standardized representation of the identifiers of compounds. The two most widely used representations that are also supported by MetaFetcheR include the simplified molecular input line entry system (SMILES) [17] and the IUPAC international chemical identifier (InChI) [18,19] that describe chemical structures using ASCII characters.

The foundation of the underlying algorithm relies on constructing a local PostgreSQL database that acts as a cache memory of information. Initially, database dump files provided by HMDB, ChEBI, and LIPID MAPS are downloaded. Subsequently, the bulk insertion function of MetaFetcheR is invoked to construct a local database (Supplementary Materials Figure S1). In the interest of storage space and time, we chose MetaFetcheR to cache data through HTTP calls on the fly from KEGG and PubChem. Cached instances from KEGG and PubChem are stored in the PostgreSQL database for later use in order to avoid unnecessary HTTP calls and timeouts due to excessive calls. A schematic of the local database is illustrated in Figure 1 under the label “Data Repositories”.

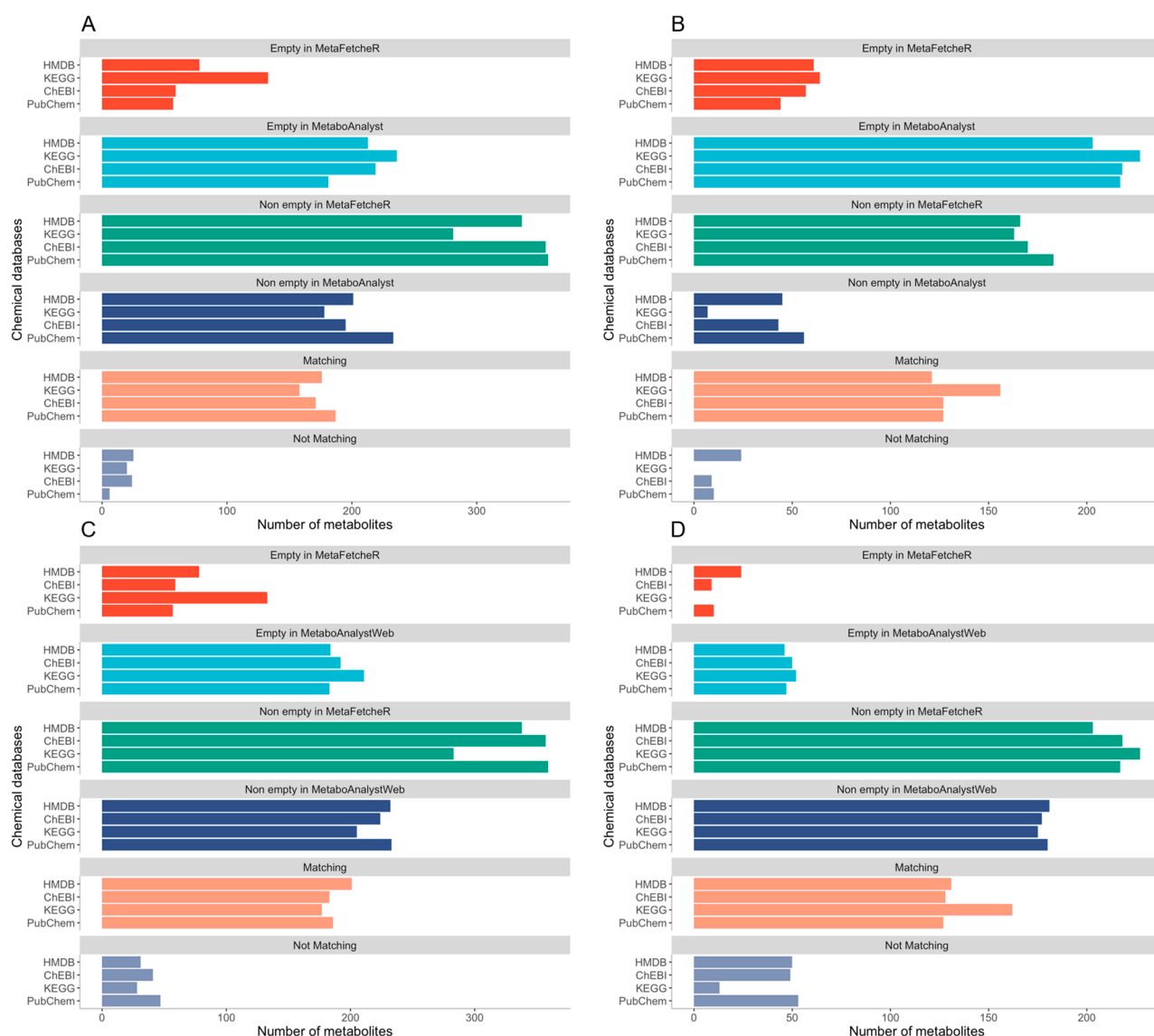
The algorithm takes as input a sparse table of known identifiers of a collection of small compounds from the five databases (HMDB, ChEBI, KEGG, LIPID MAPS, PubChem) [5,7,9,16,20] and works on mapping them to identifiers of other databases by filling in the empty fields. This is orchestrated via a queue-based algorithm. Database identifiers are stored in the columns while rows represent metabolites of interest whose identifiers require mapping (Supplementary Materials Table S11) (Figure 1-Input table). For each row in the input table, the algorithm appends the known identifiers one by one to a queue (Figure 1-MetaFetcheR main Flow-step 1). The algorithm fetches the db\_id, where db is the database and id the identifier, from the top of the queue and queries the respective database for the record with this db\_id as a primary identifier (Figure 1-MetaFetcheR main flow-step 2). The algorithm then fills all remaining identifiers for this metabolite with the returned results of the query (Figure 1-MetaFetcheR main flow-step 3). The query return might be empty, in which case, the algorithm issues a new mapping attempt and queries the database again with the db\_id as a secondary identifier, which several databases have (Figure 1-MetaFetcheR main flow-step 2). When the queue is empty and the algorithm has filled in most of the empty fields with the newly discovered identifiers (Figure 1-MetaFetcheR main flow-step 3), it reiterates to check whether there are any remaining empty fields (Figure 1-Check Point). In the case of an empty field, a reverse query is issued with one of the identifiers that were resolved during the first pass (Figure 1-MetaFetcheR exceptional flow). The reverse query uses the discovered identifiers to query the respective database in an attempt to fill in the missing field (Figure 1-MetaFetcheR exceptional flow). The algorithm reiterates until all identifiers have been filled in or cannot be further resolved

(Figure 2-MetaFetcheR main flow). At the same time, it tracks already discovered records to avoid re-adding them to the queue. Linked identifiers are updated in the local database to avoid future queries and remapping. Moreover, the algorithm stores multiple mapped identifiers of the same compound from the same database that marks ambiguous situations. This allows the user to choose among the discovered identifiers those that are the most fit for downstream analysis. Additionally, it tracks all the identifiers that were used for mapping but returned no result along with the identifiers that were used as secondary\_ids and returns the result to the user (Supplementary Materials Figure S2).

The ability of the algorithm to exhaustively reiterate to resolve cases along with storing multiple discovered database identifiers for the same compound is one of the traits that allows MetaFetcheR to stand out. The algorithm optimizes speed performance by keeping track of formerly discovered records to avoid unnecessary iterations. The size of the database after installation of HMDB, ChEBI, and LIPID MAPS was 189 MB, and after retrieving records for 100 metabolites from PubChem and KEGG, the total size increased by 0.94 MB for the former and 0.20 MB for the latter.



**Figure 1.** A simplified graphic illustration of the MetaFetcheR algorithm. Data repositories represent the local database built during installation of the package and the http calls to the application programming interfaces (APIs) for PubChem and KEGG. MetaFetcheR main flow represents the main flow of the algorithm, which constitutes three main steps. Queue represents the working queue data structure the algorithm utilizes to add all the primary and secondary known IDs from the input table and consequently perform the search queries with the IDs present in the queue. Check point is the step when the algorithm has emptied the queue after a round of search to check if the input table still has empty fields, in which case, it utilizes the steps in MetaFetcheR exceptional flow. A detailed version is available in Supplementary Materials Figure S2.

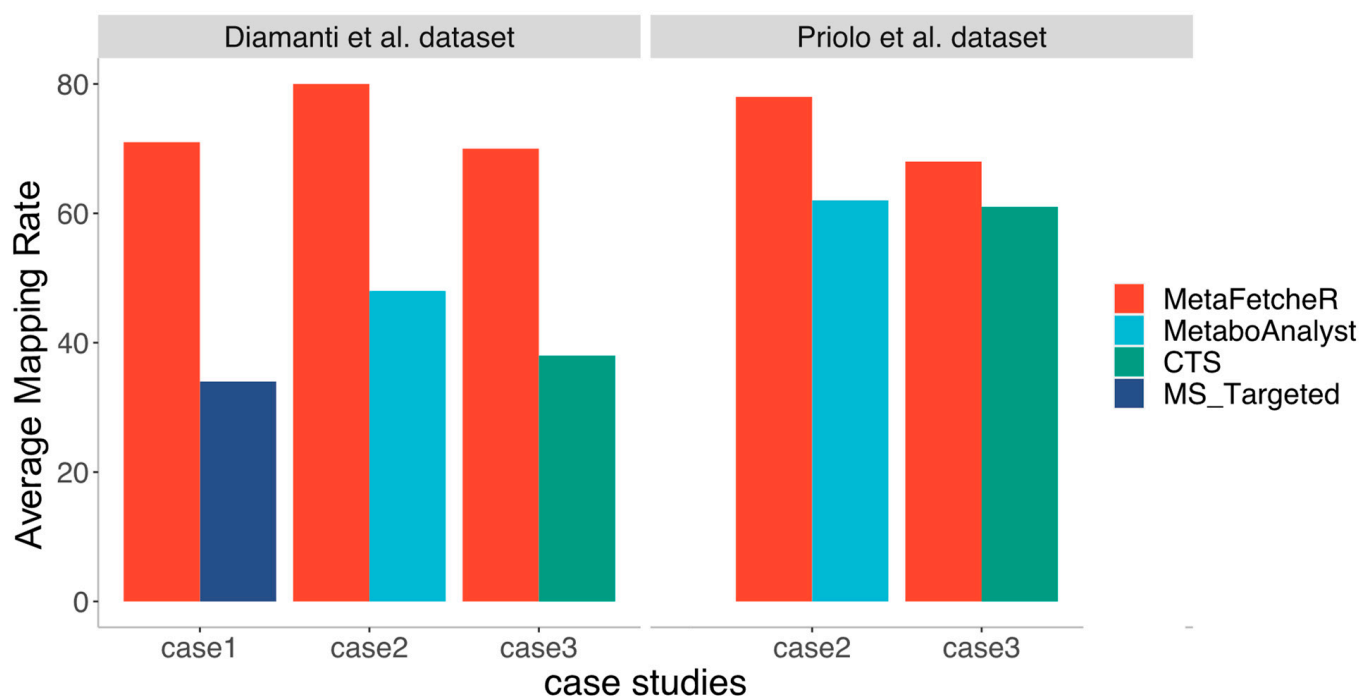


**Figure 2.** Comparison of the mapping performance of MetaFetcher to MetaboAnalystR and MetaboAnalyst 5.0 webtool. (A) Comparison of the mapping performance for MetaFetcher and MetaboAnalystR on Diamanti et al.'s [1] dataset, (B) Comparison of the mapping performance for MetaFetcher and MetaboAnalystR on Priolo et al.'s [12] dataset, (C) Comparison of the mapping performance for MetaFetcher and MetaboAnalyst 5.0 webtool on Diamanti et al.'s [1] dataset, (D) Comparison of the mapping performance for MetaFetcher and MetaboAnalyst 5.0 webtool on Priolo et al.'s [12] dataset. Empty in the MetaFetcher and MetaboAnalyst or MetaboAnalystWeb panels illustrates the number of identifiers that could not be mapped using the respective tool. Non-empty in the MetaFetcher and MetaboAnalyst or MetaboAnalystWeb panels presents the number of identifiers that were successfully mapped using the respective tool. Matching panel shows the number of mapped identifiers that agreed between tools. Non-matching panel shows the number of mapped identifiers that were not in agreement between tools. The number of identifiers is shown on the x-axis.

### Usage Scenarios and Benchmarking

MetaFetcher resolves problematic situations that arise when mapping identifiers of metabolites (Supplementary Materials Figure S3). We benchmarked the matching performance of MetaFetcher against three existing mappers in three case studies. We compared the performance of MetaFetcher to the MetaboAnalystR R package version 3.0.3 using two datasets from Diamanti et al. and Priolo et al. [1,12]. The average mapping rate of MetaFetcher was ~81% (non-empty fields), while MetaboAnalystR achieved an ~48% average mapping rate on Diamanti et al.'s [1] dataset (Figure 2A-Supplementary Materials Table S1). For the dataset

from Priolo et al. [12], MetaFetcheR achieved ~95% average non-empty fields rate, while MetaboAnalystR resulted in an ~73% average mapping rate (Figure 2B-Supplementary Materials Table S2). Furthermore, we compared the mapping performance of MetaFetcheR to the one of MetaboAnalyst 5.0 web tool and for both datasets, MetaFetcheR performed better (Figure 2B,C, Supplementary Materials Tables S3 and S4). Specifically, the MetaboAnalyst web tool achieved average ~54% mapping rate on Diamanti et al.'s [1] dataset (Figure 2C, Supplementary Materials Table S3) and ~78% average mapping rate for the dataset from Priolo et al. [12] (Figure 2D, Supplementary Materials Table S4). A similar performance to MetaboAnalyst was observed in the test utilizing CTS and MS\_targeted (Figure 3); however, the setting of the experiment was different than that with MetaboAnalystR since the latter can only perform mapping using metabolite names as an input but not metabolite identifiers, while CTS can map using multiple types of metabolite identifiers as input. The mapping rate of MetaFetcheR was on average ~70%, 68% (non-empty fields), while CTS achieved ~38%, 61% average mapping rate on Diamanti et al.'s [1] and Priolo et al.'s [12] datasets, respectively (Supplementary Materials Figures S4 and S5). For MS\_targeted, the average mapping rate was ~34% compared to MetaFetcheR's ~71% on Diamanti et al.'s [1] dataset (Supplementary Materials Figure S6). More details about the mapping rate of the three tools compared to MetaFetcheR can be found in Supplementary Note—Results of benchmarking mapping performance of MetaFetcheR. In terms of the running time MetaFetcheR outperformed CTS on both datasets. However, MetaboAnalystR had slightly better running times (Table 1).



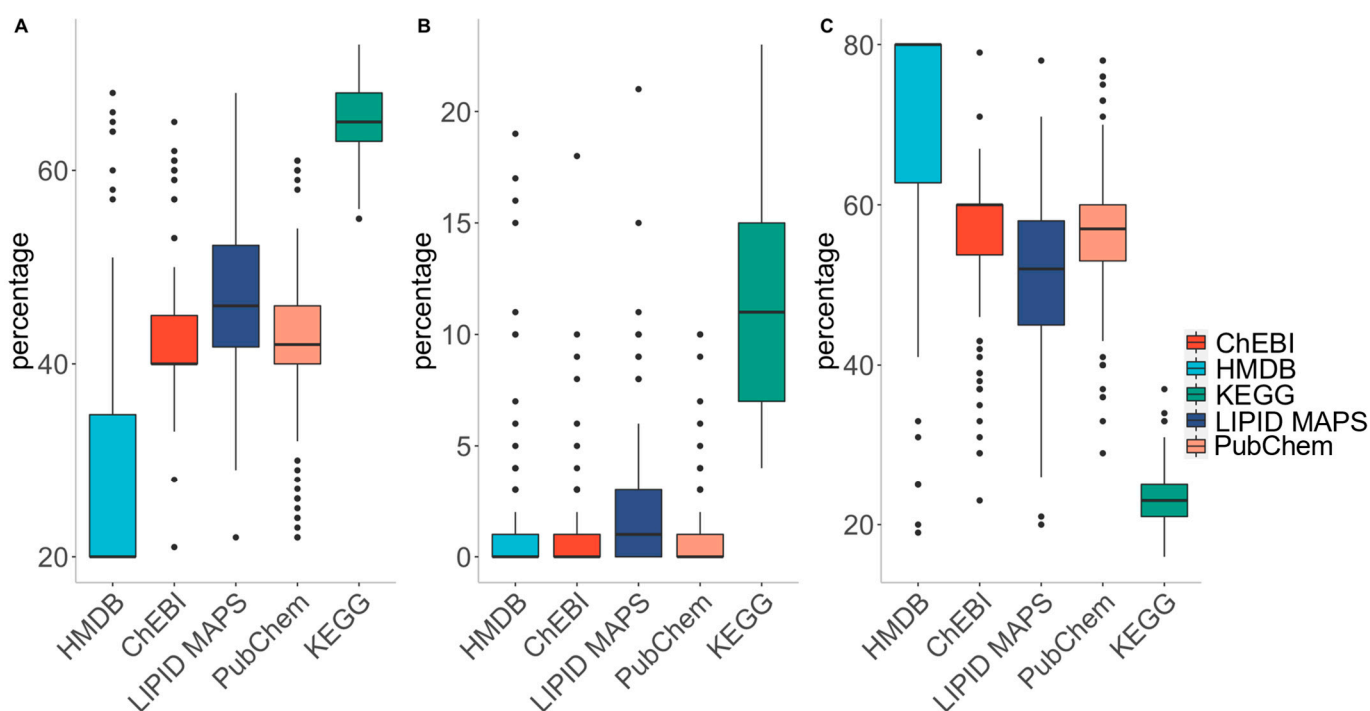
**Figure 3.** Comparison of the mapping performance for the three tools with the mapping performance of MetaFetcheR. The comparison utilizes the data from the three case studies performed on the two datasets from Diamanti et al. [1] and Priolo et al. [12].

In addition to its competitiveness in mapping the identifiers of metabolites, MetaFetcheR also provides insights into the quality of small-compound databases. A test was run by selecting 1000 random identifiers from one of the five databases as input to MetaFetcheR and then we investigated the quality of the collection of retrieved identifiers. The test was performed 100 times for each database (Supplementary Materials Tables S5–S9). The quality of the databases was assessed using three different metrics: (i) percentage of consistency, (ii) percentage of ambiguity, and (iii) percentage of unresolved cases. Consistency represents the percentage of one-to-one associated cases across all identifiers. Ambiguity is

the percentage of original metabolite identifiers linked to multiple identifiers from other databases. Unresolved cases represent the percentage of cases that the original metabolite identifiers failed to link or were absent in all other databases. KEGG showed the highest consistency percentage (~65%) and the lowest fraction of unresolved cases (~23%) compared to HMDB, which had highest fraction of unresolved cases (~71%) (Figure 4).

**Table 1.** Comparison of the running time for MetaFetcher, MetaboAnalystR, and CTS.

Tool	Number of Input Metabolites	Input Type	Dataset	Running Time
MetaboanalystR	434	Metabolites names	Diamanti et al. [1]	1 min
	228	Metabolites names	Priolo et al. [12]	30 s
Metafetcher	434	HMDB, ChEBI, LIPID MAPS, KEGG, PubChem identifiers	Diamanti et al. [1]	5 min
	228	HMDB, ChEBI, LIPID MAPS, KEGG, PubChem identifiers	Priolo et al. [12]	2 min
	328	HMDB identifiers	Diamanti et al. [1]	1 min
	219	KEGG identifiers	Diamanti et al. [1]	1 min
	68	LIPID MAPS identifiers	Diamanti et al. [1]	20 s
	228	KEGG identifiers	Priolo et al. [12]	2 min
CTS	328	HMDB identifiers	Diamanti et al. [1]	4 min
	219	KEGG identifiers	Diamanti et al. [1]	10 min
	68	LIPID MAPS identifiers	Diamanti et al. [1]	4 min
	228	KEGG identifiers	Priolo et al. [12]	18 min



**Figure 4.** The results for the test that was run on each database to investigate data quality. (A) Box plot for the fraction of successfully mapped identifiers that had one-to-one mappings to the rest of the database metabolite identifiers after running the test 100 times for each database. (B) Box plot for the fraction of successfully mapped identifiers that had at least one one-to-many mappings to the rest of the database metabolite identifiers after running the test 100 times for each database. (C) Box plot for the extent of unmapped identifiers to the rest of the database metabolite identifiers after running the test 100 times for each database.



### 3. Discussion

In this study, we presented MetaFetcher, an R package for mapping metabolite identifiers across five small-compound databases. Using two published datasets by Diamanti et al. and Priolo et al. [1,12], MetaFetcher was shown to outperform other existing tools, such as MS\_targeted, MetaboAnalyst (both R package and webtool v5.0), and CTS, that provide similar mapping functionalities. Additionally, MetaFetcher showed a reasonable running time compared to similar tools even given its exhaustive reiteration. For instance, MetaboAnalystR and MetaboAnalyst 5.0 web tool were slightly faster, which indicates that the better mapping performance of MetaFetcher compared to MetaboAnalystR and MetaboAnalyst 5.0 web tool was slightly compromised by the running time. MetaFetcher will be continuously updated to include additional databases. Features that include searching by metabolite name and automatic updates of the local database will be provided in future builds. A limitation of our current solution is that the user might be required to manually curate the output dataset in order to resolve the inconsistencies of the databases.

### 4. Materials and Methods

MetaFetcher package was developed using R version 3.5. All the database queries for installation and mapping tasks are performed using PostgreSQL version 12. PostgreSQL is a free and open-source database and is released under the PostgreSQL License, which is a liberal Open-Source license.

MetaFetcher is available at <https://github.com/komorowskilab/MetaFetcher/> (accessed on 17 October 2021). Details about installing the package, the prerequisites, and the main functions for running MetaFetcher are available at <https://komorowskilab.github.io/metafetcher/> (accessed on 14 October 2021).

The SQL dumps were downloaded to install the local database for testing on 7 May 2020.

In order to allow reproducibility of the results, we placed all the scripts used to generate the results and figures in [https://github.com/komorowskilab/MetaFetcher\\_Experiments](https://github.com/komorowskilab/MetaFetcher_Experiments) (accessed on 22 October 2021).

#### 4.1. Performance Measures

Mapping Rate<sub>db\_id</sub> represents the coverage of each type of database identifier after running tool X.

db\_id represents the type of identifier (HMDB, ChEBI, LIPID MAPS, KEGG, or PubChem). The number of non-empty fields<sub>db\_id</sub>(X) represents the number of non-empty fields for the db\_id identifier type after running tool X. The total number of records represents the total number of metabolites records in the input table:

$$\text{Mapping Rate}_{\text{db\_id}}(X) = \frac{\text{Number of non-empty fields}_{\text{db\_id}}(X)}{\text{Total number of records}} \quad (1)$$

$$\text{Average Mapping Rate}_{(i_1, i_2, i_3, \dots, i_k)}(X) = \frac{\text{Mapping Rate}_{i_1}(X) + \text{Mapping Rate}_{i_2}(X) + \dots + \text{Mapping Rate}_{i_k}(X)}{k} \quad (2)$$

The average mapping rate is the summation of all mapping rates of tool X on all identifier types ( $i_1, i_2, \dots, i_k$ ) divided by the number of identifier types (k):

$$\text{Percentage of unmapped}_{\text{db\_id}}(X) = \frac{\text{Number of empty fields}_{\text{db\_id}}(X)}{\text{Total number of records}} \quad (3)$$

where percentage of unmapped<sub>db\_id</sub>(X) represents the percentage of empty fields for each identifier type after running tool X. db\_id represents the type of identifier (HMDB, ChEBI, LIPID MAPS, KEGG or PubChem). The number of empty fields<sub>db\_id</sub>(X) represents the number of empty fields for the db\_id identifier type after running tool X. The total number of records represents the total number of metabolites records in the input table:

$$\text{Matching percentage}_{\text{db\_id}}(\text{MetaFetcher}, Y) = \frac{\text{Number of matching feilds}_{\text{db\_id}}(\text{MetaFetcher}, Y)}{\text{non-empty fields}_{\text{db\_id}}(Y)} \quad (4)$$

where matching percentage<sub>db\_id</sub> (MetaFetcheR, Y) represents the percentage of matching identifiers between both tools MetaFetcheR and Y after running them for the identifier type db\_id. db\_id represents the type of identifier (HMDB, ChEBI, LIPID MAPS, KEGG, or PubChem). The number of matching fields<sub>db\_id</sub> (MetaFetcheR, Y) represents the number of matching identifiers between both tools MetaFetcheR and Y after running them for the identifier type db\_id. Non-empty fields<sub>db\_id</sub> (Y) represents the number of non-empty fields for db\_id identifier type after running tool Y.

#### 4.2. Benchmarking Mapping Performance of MetaFetcheR

The performance of MetaFetcheR was benchmarked based on three case studies using two datasets and three existing tools, with the first dataset by Diamanti et al. [1] and the second one by Priolo et al. [12]. The three tools are MS\_targeted, MetaboAnalystR along with MetaboAnalyst 5.0 web tool, and Chemical Translation Service (CTS) [1,13–15].

##### 4.2.1. Case 1

We compared the performance of the algorithm for mapping metabolite identifiers to the identifiers mapped by MS\_targeted on Diamanti et al.'s [1] dataset. MS\_targeted is a command-line tool that was used earlier for mapping metabolites' identifiers. The comparison was based on the rate of mapped and unmapped metabolite identifiers from both tools. Subsequently, the results from MS\_targeted were manually curated and the concordance between MetaFetcheR and the manual curation of MS\_targeted results was assessed.

##### 4.2.2. Case 2

We compared the MetaFetcheR mapping performance to that of the compound ID conversion function of MetaboAnalystR [13] and MetaboAnalyst 5.0 webtool using data from [1,12]. MetaboAnalystR accept input in the form of metabolites' names, which is problematic since there is no commonly accepted nomenclature across small-compound databases. Both MetaboAnalystR and MetaboAnalyst5.0 cannot map to LIPID MAPS identifiers. Based on these limitations, the comparison of the mapping performance with MetaboAnalystR was restricted solely to the metabolites that were mappable and was limited for both tools to HMDB, ChEBI, KEGG, and PubChem identifiers. The list of metabolites that could not be mapped is shown in Supplementary Materials Table S10. The comparison of the mapping performance was based on the rate of mapped and unmapped metabolite identifiers when using metabolite names as input for both MetaboAnalyst tools. Unlike MetaboAnalystR, MetaboAnalyst 5.0 accepts metabolite identifiers as input. In addition to the previous comparison, we also compared the number of metabolite identifiers that MetaboAnalyst 5.0 webtool mapped when the input was the available HMDB, KEGG, and LIPID MAPS identifiers in Diamanti et al.'s [1] dataset, and the available KEGG identifiers in Priolo et al.'s [12] dataset.

##### 4.2.3. Case 3

We compared the MetaFetcheR mapping performance to that of CTS using data from [1,12]. CTS accepts lists of metabolites' names or metabolite identifiers of the same kind as input. Additionally, CTS does not support PubChem identifiers. To achieve a fair comparison, we ran CTS and MetaFetcheR three times with the available HMDB, KEGG, and LIPID MAPS identifiers in Diamanti et al.'s [1] dataset and the available KEGG identifiers in Priolo et al.'s [12] dataset. For instance, the input table to MetaFetcheR in case of using the available HMDB identifiers in Diamanti et al.'s [1] dataset had a complete HMDB\_id column and the rest of the identifiers' columns were empty whereas the input to CTS was the same list of HMDB identifiers. ChEBI identifiers were discarded from the comparison using Diamanti et al. [1] since there were only two available entries.

Figure 1, Supplementary Materials Figures S1 and S3 were created using [www.lucidchart.com](http://www.lucidchart.com) (accessed on 14 October 2021) resources. Supplementary Materials Figure S2 was created using [www.cacoo.com](http://www.cacoo.com) (accessed on 14 October 2021) resources.



**Supplementary Materials:** The following are available online at <https://www.mdpi.com/article/10.3390/metabo11110743/s1>, Figure S1: Entity Relationship Diagram (ERD) of PostgreSQL database that MetaFetcheR builds locally, Figure S2: Detailed flow chart of the MetaFetcheR algorithm, Figure S3: A representation of possible scenarios for mapping inconsistencies of metabolite identifiers between HMDB and ChEBI, Figure S4: Mapping performance comparison of MetaFetcheR to CTS on the dataset from Diamanti et al. [1], Figure S5: Performance comparison based on mapping KEGG identifiers available in Priolo et al. [12] dataset to other identifiers, Figure S6: Results of comparing MetaFetcheR to MS\_targeted with manual curation using Diamanti et al. [1] dataset, Supplementary Table S1: Results of comparing the performance of MetaFetcheR to MetaboAnalystR using Diamanti et al. [1] dataset, Supplementary Table S2: Results of comparing the performance of MetaFetcheR to MetaboAnalystR for Priolo et al. [12] dataset, Supplementary Table S3: Results of comparing the performance of MetaFetcheR to MetaboAnalyst 5.0 webtool using Diamanti et al. [1] dataset, Supplementary Table S4: Results of comparing the performance of MetaFetcheR to MetaboAnalyst 5.0 webtool for Priolo et al. [12] dataset, Supplementary Table S5: Data quality test results for running 100 iterations on HMDB database, Supplementary Table S6: Data quality test results for running 100 iterations on KEGG database, Supplementary Table S7: Data quality test results for running 100 iterations on ChEBI database, Supplementary Table S8: Data quality test results for running 100 iterations on PubChem database, Supplementary Table S9: Data quality test results for running 100 iterations on LIPID MAPS database, Supplementary Table S10: The list of metabolites that were not mapped by MetaboAnalystR for Diamanti et al. [1] dataset, Supplementary Table S11: An example of an input matrix for MetaFetcheR, Supplementary Table S12: Results of comparing the performance of MetaFetcheR to MS\_targeted using Diamanti et al. [1] dataset, Supplementary Table S13: Data set from Diamanti et al. [1], Supplementary Table S14: Data set from Priolo et al. [12], Supplementary Table S15: Results of comparing the performance of MetaFetcheR to CTS using KEGG identifiers available in Diamanti et al. [1] dataset, Supplementary Table S16: Results of comparing the performance of MetaFetcheR to CTS using LIPID MAPS identifiers available in Diamanti et al. [1] dataset, Supplementary Table S17: Results of comparing the performance of MetaFetcheR to CTS using KEGG identifiers available in Priolo et al. [12] dataset, Supplementary Table S18: Results of comparing the performance of MetaFetcheR to CTS using KEGG identifiers available in Priolo et al. [12] dataset.

**Author Contributions:** S.A.Y. prepared the first draft of the manuscript, compiled scripts for the R package, designed benchmark strategies, generated results, and assisted in the software and database design. R.C. wrote scripts to implement the algorithm and the database. K.D. is the owner of the main idea, co-wrote the manuscript and supervised the study together with J.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** Open access funding provided by Uppsala University. This research was supported in part by Foundation for the National Institutes of Health (Grant No. 0925-0001), Uppsala University, Sweden, and the Institute of Computer Science, Polish Academy of Sciences, Poland and the eSENCE grant to S.A.Y., R.C., J.K. and K.D.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data is contained within the article or supplementary material. The data presented in this study are available in Diamanti et al. [1] and Priolo et al. [12].

**Acknowledgments:** The authors thank Linda Holmfeldt and Aaron Vogan for reviewing the manuscript.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Diamanti, K.; Cavalli, M.; Pan, G.; Pereira, M.J.; Kumar, C.; Skrtic, S.; Grabherr, M.; Risérus, U.; Eriksson, J.W.; Komorowski, J.; et al. Intra- and inter-individual metabolic profiling highlights carnitine and lysophosphatidylcholine pathways as key molecular defects in type 2 diabetes. *Sci. Rep.* **2019**, *9*, 9653. [[CrossRef](#)] [[PubMed](#)]
2. Diamanti, K.; Visvanathar, R.; Pereira, M.J.; Cavalli, M.; Pan, G.; Kumar, C.; Skrtic, S.; Risérus, U.; Eriksson, J.W.; Kullberg, J.; et al. Integration of whole-body [18 F]FDG PET/MRI with non-targeted metabolomics can provide new insights on tissue-specific insulin resistance in type 2 diabetes. *Sci. Rep.* **2020**, *10*, 1–9. [[CrossRef](#)]

3. Kaushik, A.K.; DeBerardinis, R.J. Applications of metabolomics to study cancer metabolism. *Biochim. Biophys. Acta (BBA)—Rev. Cancer* **2018**, *1870*, 2–14. [[CrossRef](#)]
4. Chong, J.; Soufan, O.; Li, C.; Caraus, I.; Li, S.; Bourque, G.; Wishart, D.S.; Xia, J. MetaboAnalyst 4.0: Towards more transparent and integrative metabolomics analysis. *Nucleic Acids Res.* **2018**, *46*, W486–W494. [[CrossRef](#)] [[PubMed](#)]
5. Wishart, D.S.; Feunang, Y.D.; Marcu, A.; Guo, A.C.; Liang, K.; Vázquez-Fresno, R.; Sajed, T.; Johnson, D.; Allison, P.; Karu, N.; et al. HMDB 4.0: The human metabolome database for 2018. *Nucleic Acids Res.* **2018**, *46*, D608–D617. [[CrossRef](#)] [[PubMed](#)]
6. Hastings, J.; De Matos, P.; Dekker, A.; Ennis, M.; Harsha, B.; Kale, N.; Muthukrishnan, V.; Owen, G.; Turner, S.; Williams, M.; et al. The ChEBI reference database and ontology for biologically relevant chemistry: Enhancements for 2013. *Nucleic Acids Res.* **2013**, *41*, D456–D463. [[CrossRef](#)] [[PubMed](#)]
7. Kim, S.; Chen, J.; Cheng, T.; Gindulyte, A.; He, J.; He, S.; Li, Q.; Shoemaker, B.A.; Thiessen, P.; Yu, B.; et al. PubChem 2019 update: Improved access to chemical data. *Nucleic Acids Res.* **2019**, *47*, D1102–D1109. [[CrossRef](#)] [[PubMed](#)]
8. Degtyarenko, K.; De Matos, P.; Ennis, M.; Hastings, J.; Zbinden, M.; McNaught, A.; Alcántara, R.; Darsow, M.; Guedj, M.; Ashburner, M. ChEBI: A database and ontology for chemical entities of biological interest. *Nucleic Acids Res.* **2008**, *36*, D344–D350. [[CrossRef](#)] [[PubMed](#)]
9. Hastings, J.; Owen, G.; Dekker, A.; Ennis, M.; Kale, N.; Muthukrishnan, V.; Turner, S.; Swainston, N.; Mendes, P.; Steinbeck, C. ChEBI in 2016: Improved services and an expanding collection of metabolites. *Nucleic Acids Res.* **2016**, *44*, D1214–D1219. [[CrossRef](#)] [[PubMed](#)]
10. Kanehisa, M.; Furumichi, M.; Tanabe, M.; Sato, Y.; Morishima, K. KEGG: New perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Res.* **2017**, *45*, D353–D361. [[CrossRef](#)] [[PubMed](#)]
11. Smith, C.A. METLIN: A metabolite mass spectral database. *Ther. Drug Monit.* **2005**, *27*, 747–751. [[CrossRef](#)] [[PubMed](#)]
12. Priolo, C.; Pyne, S.; Rose, J.; Regan, E.R.; Zadra, G.; Photopoulos, C.; Cacciatore, S.; Schultz, D.; Scaglia, N.; McDunn, J.; et al. AKT1 and MYC Induce Distinctive Metabolic Fingerprints in Human Prostate Cancer. *Cancer Res.* **2014**, *74*, 7198–7204. [[CrossRef](#)] [[PubMed](#)]
13. Pang, Z.; Chong, J.; Li, S.; Xia, J. MetaboAnalystR 3.0: Toward an Optimized Workflow for Global Metabolomics. *Metabolites* **2020**, *10*, 186. [[CrossRef](#)] [[PubMed](#)]
14. Pang, Z.; Chong, J.; Zhou, G.; de Lima Morais, D.A.; Chang, L.; Barrette, M.; Gauthier, C.; Jacques, P.; Li, S.; Xia, J. MetaboAnalyst 5.0: Narrowing the gap between raw spectra and functional insights. *Nucleic Acids Res.* **2021**, *49*, W388–W396. [[CrossRef](#)] [[PubMed](#)]
15. Wohlgemuth, G.; Haldiya, P.K.; Willighagen, E.; Kind, T.; Fiehn, O. The Chemical Translation Service—A web-based tool to improve standardization of metabolomic reports. *Bioinformatics* **2010**, *26*, 2647–2648. [[CrossRef](#)] [[PubMed](#)]
16. Sud, M.; Fahy, E.; Cotter, D.; Dennis, E.A.; Subramaniam, S. LIPID MAPS-Nature Lipidomics Gateway: An Online Resource for Students and Educators Interested in Lipids. *J. Chem. Educ.* **2012**, *89*, 291–292. [[CrossRef](#)] [[PubMed](#)]
17. Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36. [[CrossRef](#)]
18. Dashti, H.; Westler, W.M.; Markley, J.L.; Eghbalnia, H.R. Unique identifiers for small molecules enable rigorous labeling of their atoms. *Sci. Data* **2017**, *4*, 170073. [[CrossRef](#)] [[PubMed](#)]
19. Heller, S.R.; McNaught, A.; Pletnev, I.; Stein, S.; Tchekhovskoi, D. InChI, the IUPAC International Chemical Identifier. *J. Cheminform.* **2015**, *7*, 23. [[CrossRef](#)] [[PubMed](#)]
20. Kanehisa, M.; Goto, S. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.* **2000**, *28*, 27–30. [[CrossRef](#)] [[PubMed](#)]