

pyBiodatafuse: extending interoperability of data using modular queries across biomedical resources

Yojana Gadiya^{1,†,*,}, Javier Millán Acosta^{2,†,}, Ammar Ammar^{2,}, Alejandro Adriaque Lozano³, Delano Wetstede⁴, Dominik Martinát^{5,}, Ana Claudia Sima^{6,}, Hailiang Mei^{7,}, Egon Willighagen^{2,}, Tooba Abbassi-Daloi^{2,8,*,†,}

¹Enveda, Boulder, Colorado, 80301, United States

²Department of Translational Genomics, NUTRIM Institute of Nutrition and Translational Research in Metabolism, Maastricht University, Maastricht, 6229, The Netherlands

³Maastricht Centre for Systems Biology (MaCSBio), Maastricht University, Maastricht, 6229, The Netherlands

⁴Faculty of Life Science & Technology, Leiden University of Applied Sciences, Leiden, 2333, The Netherlands

⁵Department of Physical Chemistry, Palacky University, Olomouc, 779 00, Czech Republic

⁶SIB Swiss Institute of Bioinformatics, Lausanne, 1015, Switzerland

⁷Sequencing Analysis Support Core, Leiden University Medical Center (LUMC), Leiden, 2333, The Netherlands

⁸Department of Medical Biology, University of Amsterdam, Amsterdam UMC, Amsterdam, North Holland, 1105, The Netherlands

*Corresponding authors. Tooba Abbassi-Daloi, Department of Medical Biology, University of Amsterdam, Amsterdam UMC, Amsterdam.

E-mail: t.abbassidaloi@amsterdamumc.nl; Yojana Gadiya, Enveda, 5700 Flatiron Parkway, Boulder. E-mail: yojanagadiya@gmail.com

†Shared co-authorship.

Associate Editor: Jonathan Wren

Abstract

Motivation: Integrating omics data analysis with publicly available databases is crucial for unravelling complex biological mechanisms. However, this integration process is often intricate and time-consuming due to the diversity and complexity of the data involved. Achieving consistent harmonization across data types is challenging when managing disparate formats and sources. To address these issues, we introduce *pyBiodatafuse*, a query-based Python tool designed to integrate biomedical databases. This tool establishes a modular framework that simplifies data wrangling, enabling the creation of context-specific knowledge graphs (KGs) while supporting graph-based analyses.

Results: We developed a pipeline for generating context-specific knowledge graphs dynamically, allowing users to create KGs on the fly from a set of gene or metabolite identifiers. *pyBiodatafuse* features a user-friendly interface that streamlines this process, making it accessible even to researchers without extensive computational expertise. Additionally, the tool offers plugins for widely used platforms such as Cytoscape, Neo4j, and GraphDB, enabling local hosting of resulting property and RDF graphs. This versatility ensures that generated KGs can be efficiently utilized within diverse research workflows. To demonstrate its potential, we used *pyBiodatafuse* to create a graph for post-COVID syndrome using differential gene expression data, showcasing its ability to build adaptable and context-specific knowledge representations. Thus, *pyBiodatafuse* sets the stage for streamlined data integration, empowering researchers to focus on discovery and analysis without being hindered by data management complexities.

Availability and implementation: *pyBiodatafuse* is open-source, with its source code and PyPi package available at <https://github.com/BioDataFuse/pyBiodatafuse> and <https://pypi.org/project/pyBiodatafuse/>. The user interface can be accessed at <https://biodatafuse.org/>. Additionally, a release has been made on Zenodo at <https://doi.org/10.5281/zenodo.18468942>.

1 Introduction

There has been a growing interest in the generation and utility of biomedical knowledge graphs (KG), driven by the remarkable increase in available data across diverse biomedical resources (Mazein *et al.* 2024). This surge has led to the development of

numerous tools and databases dedicated to constructing and refining KGs within the biomedical domain (Belleau *et al.* 2008, Chen *et al.* 2010, Bauer 2011, Williams *et al.* 2012). However, the landscape of these tools and databases is somewhat fragmented, with crucial information often spread across multiple disconnected sources. The issue of data fragmentation is further

intensified by the presence of data silos, which arise due to the diverse but specific research focuses on groups across different regions of the world. These silos indirectly either limit the selection of specific tools and databases for answering questions or lead to the generation of isolated datasets or databases. This separation, in turn, limits the data accessibility and applicability, thus hindering collaborative efforts for the development of comprehensive solutions. Addressing this challenge requires efforts toward standardization, interoperability, and the breaking down of these silos to foster more effective global collaboration.

In biomedical research, data are distinguished into two types based on their source: experimental and literature-based aggregated data. Experimental data originates directly from laboratory experiments, clinical trials, or other empirical studies, yielding raw data specific to a particular context or condition. On the other hand, literature-based aggregated data is derived from published scientific literature, where researchers curate, summarize, and annotate findings from numerous studies, providing a broader understanding of various biological conditions. Despite their complementary nature, integrating experimentally generated data with literature-based data presents challenges due to differences in data formats, experimental protocols, annotation standards, data availability, and more. Bridging this divide requires sophisticated data integration techniques and robust frameworks to harmonize and contextualize disparate datasets, enabling researchers to gain deeper insights into complex biological phenomena. Consequently, initiatives and tools have emerged to address this challenge and provide resources for the community. A well-known resource in this context is StringDB (<https://string-db.org/>), a protein-protein interaction database (Mering 2003).

Another early example of a tool integrating biomedical resources is BioQuery (Brundage and Dubay 2003). Equipped with a standalone Java-based interface, researchers can utilize this tool to construct systematic queries that persist across database updates. A notable feature of this application was the “Query Object” framework, enabling users to formulate queries via a graphical user interface, while the application managed the intricacies of database communication. However, a limitation of the tool is its restricted access to data resources, particularly those supported by NCBI. In contrast, consortiums like Bio2RDF (Belleau *et al.* 2008), RDF Portal (Kawashima *et al.* 2018), OpenPHACTS (Williams *et al.* 2012), OpenTargets (Ochoa *et al.* 2021), and Linked Life Data (<http://linkedlifedata.com/>), among others, have facilitated the comprehensive collection, organization, and generation of biomedical data for community use.

While comprehensive KGs have been instrumental in consolidating diverse biomedical data sources, their construction often necessitates extensive local data caching, demanding significant storage resources that may not be universally accessible. Among these tools, InterMine (<http://intermine.org/>) (Smith *et al.* 2012) has stood out as particularly successful, owing to its ability to integrate data from multiple resources (and organisms) in a data warehousing approach. Building upon this concept, TargetMine (<https://targetmine.mizuguchilab.org/>) was developed and tailored to assist drug discovery with a special focus on target prioritization (Chen *et al.* 2011). However, despite their effectiveness, a persistent challenge lies in keeping

these local data caches updated with the latest versions of the resources.

Although there is a wealth of data available in biomedical resources and an extensive storage requirement in classical KG-based solutions, not all data are directly relevant to a specific research question. This supply and demand mismatch underscores the need to carefully select and integrate data to construct a more question-specific or context-specific KG. To achieve efficient data integration, a combination of federated and modular queries can be utilized to extract relevant data subsets. Federated queries allow accessing, querying, and integrating diverse data sources without requiring data caching. However, they can be complex and data model-dependent, making maintenance challenging over time. Modular queries help mitigate this issue by breaking down the querying process into smaller, more manageable components. Together, federated and modular queries streamline data retrieval and reduce the burdens associated with traditional data storage and maintenance methods. Several other tools and data resources have since emerged, adopting concepts and frameworks inspired by the combination of federated and modular queries (Bizon *et al.* 2019, Queralt-Rosinach *et al.* 2020, Doğan *et al.* 2021, Morris *et al.* 2023). However, despite these advancements, the focus has primarily been on constructing the most comprehensive KG, historically leading to information bias based on community interests, as highlighted by Bonner *et al.* (2022). Furthermore, little attention has been given to the downstream utility of this comprehensive graph for various research domains.

To address the limitations in generating KGs, the BioCypher (<https://biocypher.org/>) initiative was launched (Lobentanzer *et al.* 2023). Its main mission is to democratize KG creation, enabling individuals to construct their own graphs more easily. Driven from an ontological harmonization perspective, the tool enabled the FAIRification of resources. However, the generation of graph schemas and yaml-based configuration files for data extraction is still required in the background. To further streamline and stimulate the use of KGs in the broader research community, we launched the BioDataFuse project during the ELIXIR BioHackathon Europe 2023 (Gadiya 2023). A fundamental part of this project was the *pyBiodatafuse* tool, which aimed to achieve two primary objectives: firstly, to integrate data from various biomedical resources in real-time, establishing a modular framework for data wrangling, and secondly, to provide researchers without extensive in-house data science expertise an easy-to-use pipeline to build and visualize context-specific graphs. *pyBiodatafuse* enables researchers to customize graph construction based on strategic decisions regarding their preferred data sources, thereby allowing the opportunity to reduce bias. Through *pyBiodatafuse*, we plan to assist researchers in leveraging publicly available data and integrating it with their local data in a structural, systematic, and context-specific manner.

The tool contributes to the community by addressing key aspects of integrated data analysis:

- **Extensive data collection:** The tool can integrate diverse and valuable biomedical data sources, aiming to gather comprehensive information about specific contexts.
- **Generation of KGs:** We devise a robust mechanism for structuring data presented in numerous resources into property

and RDF graphs, eliminating the need for researchers to perform this task manually.

- **Data summarization:** We leverage graphical representations of data to provide visual summaries that offer quick insights into the underlying graph, particularly useful for non-graph experts. These summaries include graph-oriented statistics, such as the number and types of nodes and edges added by querying each data source, as well as research-oriented summaries, like the number of publications and patents related to the context.
- **Use case-tailored pipeline:** We have developed an end-to-end pipeline that allows researchers to upload a list of genes, compounds, or differential gene expression analysis results and generate a context-specific KG tailored to their specific needs.

Ultimately, *pyBiodatafuse* aims to advance into a FAIR-compliant platform empowering researchers to explore, interpret, and visualize data spread across diverse biomedical resources alongside internally generated data. Moreover, it seeks to integrate graph-based algorithms, encompassing both classic and machine learning approaches, enabling researchers to make novel predictions on the contextual graph. Notably, its on-the-fly query capability distinguishes it from existing approaches, eliminating the need for any local data storage.

2 Materials and methods

2.1 Design architecture and framework

We built the *pyBiodatafuse* on four main components: (i) data harmonizer, (ii) data annotators, (iii) graph generator and visualizer, and (iv) graph analyzer (Fig. 1). In the following subsections, we elaborate on each of these components individually.

2.2 Data harmonizer

The data harmonizer component includes two primary elements designed to support different biological entities (genes, metabolites, or compounds) as input and an identifier mapper service. Together, these elements transform heterogeneous user-submitted data into a unified format that can be readily used by downstream “data annotators” modules. Currently, *pyBiodatafuse* supports three distinct input types: lists of genes or proteins, lists of metabolites or compounds, and the output table of a differential gene expression analysis (DEA) result table. While DEA results also contain gene identifiers, they are handled separately because they include additional metadata, such as fold changes and statistical significance, that are preserved and incorporated into the resulting KG for contextual enrichment.

Aligned with FAIR principles, our tool emphasizes the importance of unique persistent identifiers for diverse biological entities. To achieve this, the ID mapper service systematically maps biological entities from input data or the queried biomedical resources to preselected ontologies, controlled vocabularies, and terminologies. This model leverages the BridgeDb framework for database, controlled vocabulary, and ontology harmonization and mapping, enabling the *pyBiodatafuse* to link

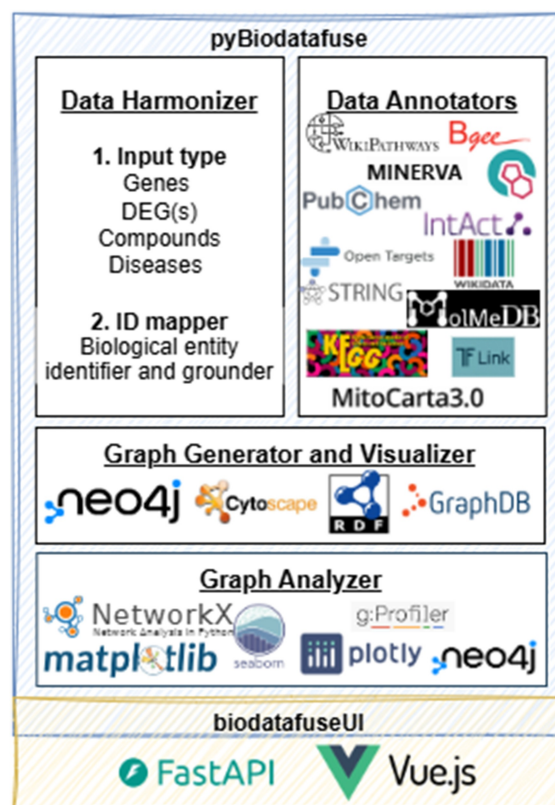


Figure 1 The *pyBiodatafuse* architecture and its connection with *biodatafuseUI*.

identical biological concepts across different data sources (Van Iersel et al. 2010).

2.3 Data annotators

The data annotator component enables the execution of modular queries across multiple resources. The choice of resources for integration in the tool is based on their integration with ELIXIR platforms and their popularity within research communities (Drysdale et al. 2020). Additional factors, such as the frequency of updates and the curator community of the resource, are also considered. Rather than aggregating a large number of heterogeneous databases, *pyBiodatafuse* focuses on a curated subset of resources that are both well-maintained and broadly relevant for typical biological data analysis tasks. Currently, the tool allows users to select and query from 15 data resources (Table 1).

Each data resource utilizes its own unique storage and representation format, often requiring a specific query language for data extraction. Of these, Structured Query Language (SQL), Graph Query Language (GraphQL), Application Programming Interface (API), and SPARQL Protocol And RDF Query Language (SPARQL) are among the most widely used access mechanisms (Date 1989, Angles et al. 2018, Burkhardt 2020). To address this heterogeneity, *pyBiodatafuse* implements a unified querying layer that translates user queries into the appropriate format for each resource. This design enables the tool to interact with multiple resources in parallel, regardless of their underlying query

Table 1 List of biomedical data annotators in *pyBiodatafuse*.

Data resource	Query language	Supported biological entities (interaction types)
AOP WIKI RDF (Martens et al. 2022)	SPARQL	Genes, compounds, AOP components, GO annotations
Bgee (Bastian et al. 2021)	SPARQL	Genes, anatomical entities
DisGeNET (Piñero et al. 2020)	API	Genes, diseases
g:Profiler (Kolberg et al. 2023)	API	Gene, pathways
IntAct (Del Toro et al. 2022)	API	Genes, compounds
KEGG (Kanehisa et al. 2017)	API	Compounds, genes, pathways
MINERVA (Gawron et al. 2016)	API	Compounds, genes, pathways
MitoCarta (Rath et al. 2021)	–	Mitochondrial proteins, pathways
MolMeDB (Juračka et al. 2019)	SPARQL	Compounds, genes
OpenTargets (Koscielny et al. 2017)	GraphQL	Compounds, genes, pathways, biological processes, molecular functions, cellular components, diseases
PubChem (Kim et al. 2023)	SPARQL	Compounds, genes
STRING (Mering et al. 2003)	API	Genes
TFlink (Liska et al. 2022)	–	Transcription factors, genes
Wikidata (Vrandečić 2012, Waagmeester et al. 2020)	SPARQL	Genes
WikiPathways (Agrawal et al. 2024)	SPARQL	Genes, pathways

language or data model. We ensure the use of consistent ontologies and terminologies for querying the resources, providing us with two main benefits: first, it enables the harmonization of biological entities within the resulting graph, thus avoiding the presence of duplicates; second, it allows us to extract data and metadata relevant to these entities, which may be distributed across multiple independent resources.

During the querying process, we store metadata about the data source and the query history itself, such as the data source version, the query size or result set size, and the query duration. This metadata provides valuable insights into the query performance and ensures reproducibility by precisely documenting the conditions and parameters of each executed query. In addition to metadata storage, *pyBiodatafuse* implements automated tests for each annotator, which are executed at regular intervals with every commit to the code base. These tests help detect breaking version updates, such as schema changes in upstream databases, ensuring that the information extracted by the tool remains accurate and up-to-date.

2.4 Graph generator

The graph generator component serves as the basis for generating context-enriched KGs from data captured by annotators, thus allowing for the visualization of complex connections within the dataset. The graph is primarily built using the NetworkX library (Hagberg 2020) and can be exported directly to user-desired databases and platforms such as Cytoscape (Shannon et al. 2003), Neo4j (Webber 2012), and GraphDB. This allows for enhanced visualization, semantic querying of complex relationships within the dataset, and interoperability with existing biomedical knowledge graphs available in RDF and SPARQL endpoints.

Our current KG supports biological entities such as genes or proteins, diseases, and metabolites or compounds. Each of these entities is represented using established ontologies and vocabularies. Entrez identifiers denote genes, UMLS identifiers identify diseases, and PubChem identifiers identify compounds.

Protein-related information is currently inferred through gene identifiers and not represented as distinct entities. Furthermore, in addition to these fundamental biological entities, we incorporate location, pathways, side effects, and anatomical entities, among others, into the KG. This inclusion allows researchers to visualize subgraphs within specific contexts, such as those related to specific tissues like the heart or kidney.

2.5 Graph analyzer

The final component of *pyBiodatafuse* is the graph analyzer, which is designed to provide concise visual summaries of the KG. Leveraging popular Python libraries such as Matplotlib (Tosi 2009), Seaborn (Bisong 2019), and Plotly (Sievert 2020), the graph analyzer supports the creation of basic plots as well as network-specific statistical summaries. These capabilities enable users to gain a deeper understanding of the data structure and the intricate interconnections within the KG.

In addition to visualization, this module allows programmers to interact with the network using pre-defined query functions. These functions streamline the exploration and analysis process, making it easier to extract meaningful insights. The query functions include graph-centric features, including the generation of subgraphs and the preparation of training datasets for integration into existing graph algorithms. These functionalities are particularly valuable for extending the utility of the KG in machine learning and plugging it into advanced analytics workflows.

2.6 User interface (UI)

The complete set of functionalities offered by the *pyBiodatafuse* package is encapsulated within our web-based user interface, *BiodatafuseUI*, which is publicly accessible at <https://biodatafuse.org>. Designed specifically for non-technical experts, this UI enables efficient utilization of the Python package in a friendly format for non-technical users. Built using FastAPI for the back-end and Vue.js for the frontend, the interface provides a user-

friendly and intuitive experience for interacting with pyBiodatafuse functionalities.

2.7 Development guideline

To package *pyBiodatafuse*, we utilized the Cookiecutter snekpack package (<https://github.com/cthoyt/cookiecutter-snekpack>), ensuring efficient and standardized packaging practices. Additionally, emphasis on documentation accompanying each function was made to enhance code reproducibility and readability. This documentation is readily accessible on Read the Docs (<https://pybiodatafuse.readthedocs.io/en/latest/index.html>).

To ensure streamlined functionality and detect any potential issues, we have implemented comprehensive unit tests for each annotator. These test scripts serve to maintain a stable connection between resources during querying processes. Furthermore, we have integrated Tox-based testing into our GitHub Actions, automating style, dynamic typing, and documentation tests on the codebase. This automated approach significantly streamlines the process of identifying and addressing any potential integration issues and enables community development.

2.8 Implementation details

pyBiodatafuse is developed using the Git version-control system, is written in Python, and can be found at PyPi (<https://pypi.org/project/pyBiodatafuse/>) and on GitHub (<https://github.com/BioDataFuse/pyBiodatafuse>). The current stable version of the package is 1.2.0. Additionally, the scripts associated with the UI can be assessed on GitHub (<https://github.com/BioDataFuse/biodatafuseUI>), and the UI is publicly accessible at <https://biodatafuse.org/>.

3 Results

The BioDataFuse project is a community-driven initiative aimed at supporting the construction of context-specific KGs and facilitating

hypothesis generation based on the KG. This project involves the development of a Python package called *pyBiodatafuse* to allow for cross-talk across different biological resources and a FastAPI-based UI, *BiodatafuseUI*, to increase accessibility to domain and non-domain experts. We have crafted our data collection approach such that the generated graphs are not only rich in data sourced from various biomedical resources but also enriched with underlying metadata captured across these resources independently.

In the following sub-section, we discuss the intricacies of the data model underpinning our biomedical KG. We then illuminate the significance of the metadata-enriched KG that serves as the foundation of our primary biomedical KG and demonstrate an example case by generating the first context-specific subgraph for post-COVID-19 syndrome. Finally, we conclude the section by listing the limitations of the *pyBiodatafuse*, showcasing its potential for future growth and development in the Biomedical Knowledge Graph community.

3.1 Graph data model

We have developed a robust property graph data model connecting various biological entities in a structured manner. As shown in Fig. 2, our data model currently covers 18 distinct node types: gene, disease, compound, anatomical entity, pathway, molecular function, biological process, cellular component, side effect, homolog, phenotype, miRNA, transcription factor, mitochondrial pathway, key event, molecular initiating event, adverse outcome pathway, and adverse outcome. Furthermore, each node within the KG is enriched with attributes, also known as metadata, offering insights into its origin and context distributed across resources. For instance, disease nodes include information about the namespace and cross-references to unique identifiers in different databases and ontologies, as well as pertinent attributes like disease type. Similarly, anatomical entity nodes include details on expression levels and developmental stages, providing a comprehensive view of their biological context.

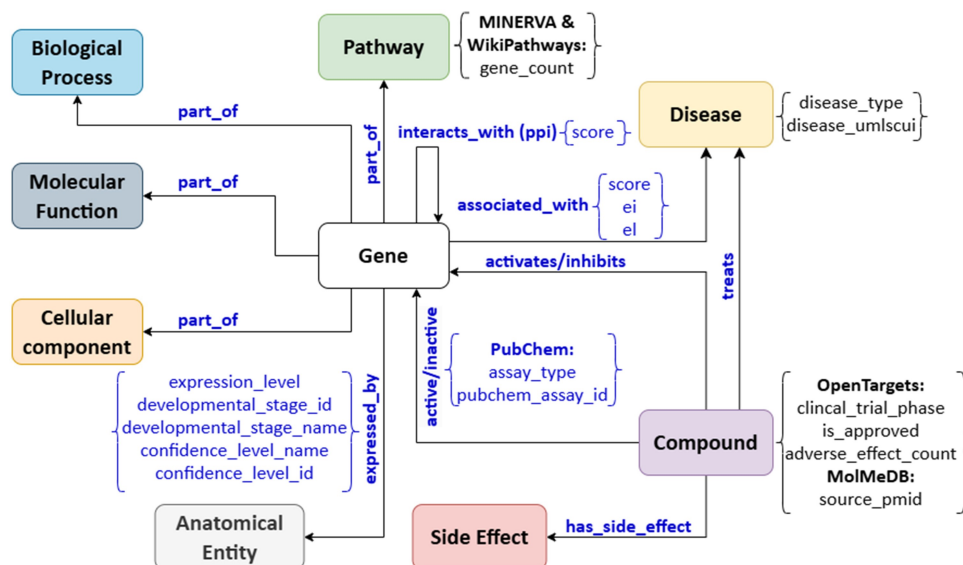


Figure 2 The property graph data model of the biomedical knowledge graph was generated by *pyBiodatafuse*.

All the nodes are connected through directed edges categorized into ten types (interacts with, part of, associated with, expressed by, activates, inhibits, treats, has side effect, upstream of, and downstream of). This clear and structured representation of our KG reveals patterns, clusters, and trends embedded across the data resources. Such insights are valuable for researchers, analysts, and decision-makers, empowering them to make informed decisions. For example, the KG can provide a quick overview of the diseases, drugs, pathways, and cellular component context around a gene rather than searching for this information by accessing multiple resources. Additionally, each edge in the KG tracks provenance information, that is, the data resource it was extracted from, allowing researchers to look into the original data resource for more detailed information.

Simultaneously, the graph acts as an intermediary stage, ready for serialization into three distinct formats. The first format follows Cytoscape's specifications in JSON, enabling integration with the tool. The second format, GraphML, aligns with Neo4j's requirements and can be loaded using Neo4j's APOC extension. The last format is its serialization to linked data, allowing for more straightforward integration with other resources while explicitly representing the relationships and classes in the graph using established ontologies in a computer-actionable format. The linked data version of the graph (Fig. 3) has been modeled in the RDF using well-known classes and predicates from ontologies and controlled vocabularies such as the Semanticscience Integrated Ontology (Dumontier, 2014) and community staples collected in the Open Biomedical Ontologies (Jackson *et al.* 2021). We facilitate the setup of RDF triplestores with a wrapper for the GraphDB API, GraphDBManager, in order to more efficiently generate GraphDB repositories, and upload, manage, and query their RDF data.

Moreover, the Shape Expressions (ShEx) or Shapes Constraint Language graph (SHACL) for the generated RDF dataset can also be automatically retrieved based on the classes used to generate the RDF version of the KG.

3.2 Metadata-enriched graphs

In the previous section, we explored the data model of *pyBiodatafuse* KG, which enables systematic connections between entities. While some metadata is directly incorporated into the graph data model (refer to Fig. 2), others, although not explicitly part of the schema, wield equal significance in hypothesis generation. The gene-anatomical entity edge ("expressed_in") is peculiar in our data model, showcasing the gene expression levels (either over- or under-expressed) within specific tissues or organs. This feature contributes to a better understanding of gene functionality in a single tissue and facilitates cross-species comparisons, such as in mice. Similarly, the gene-cellular component edge ("part_of") pinpoints cellular locations where the gene is situated, thereby helping to mitigate potential cellular toxicity effects for prospective drugs.

With the wave of open science, there is an increasing challenge for researchers to stay abreast of recent developments and publications. Even generative AI technologies like ChatGPT struggle to keep pace, occasionally resulting in hallucinations (Emsley 2023). Consequently, there is a natural inclination to revert to non-generative AI approaches to handle this. In *pyBiodatafuse*, we leverage the collaborative data curation effort in Wikidata to guide researchers toward comprehensive overviews of genes and compounds in publications. This way, the researchers can constantly monitor community interests and gauge their significance effectively.

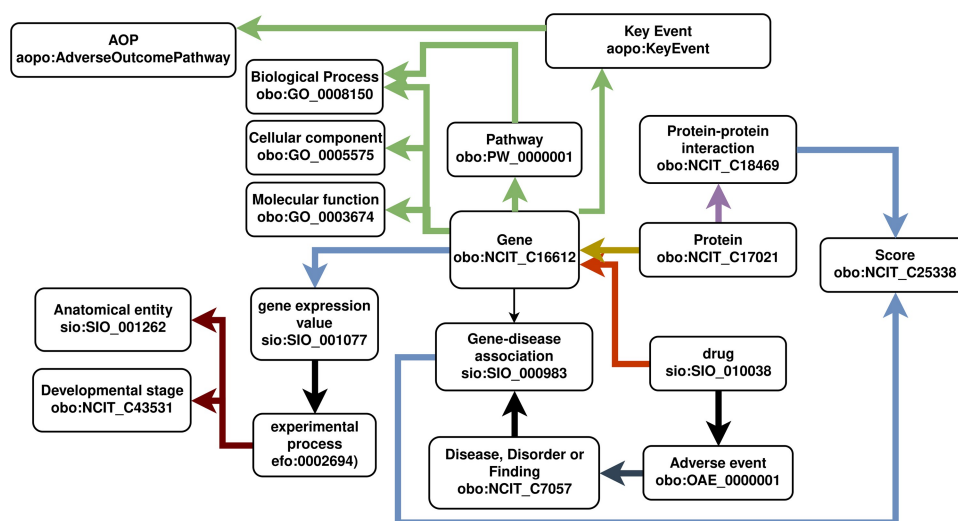


Figure 3 Simplified RDF schema of the biomedical knowledge graph generated through *pyBiodatafuse*. Color legend—Green: sio: has_part, blue: sio: has_measurement_value, yellow: so: translation_of, red: Relation Ontology (RO) predicates, subclasses of *Regulates activity of* (obo: RO_0011002). The detailed RDF schema for BioDataFuse RDF graphs is available as Shape Constraints Language in https://raw.githubusercontent.com/BioDataFuse/pyBiodatafuse/refs/heads/main/examples/bdf_example.shacl and as a SVG diagram under https://raw.githubusercontent.com/BioDataFuse/pyBiodatafuse/refs/heads/fixes_compoundwiki/examples/BDF_example_SHACL.svg.

3.3 Context-specific graph: first long COVID KG

We initiated the graph generation through the collection of differentially expressed genes from a transcriptomic dataset by [Aschman et al. \(2023\)](#). This dataset compared the gene expression profiles of the PCS cohort and the historical control cohort, 2023 differentially expressed genes (P -value < 0.05). These genes served as the starting point for the *pyBiodatafuse* tool, of which only 1667 genes were found to be consistently mapped to established ontologies. Next, we selected four data sources (DisGeNet, OpenTargets, MINERVA, and StringDB) to generate the PCS KG, resulting in five different edge types (gene-disease, gene-compound, disease-compound, gene-pathway, and gene-gene). We also included an additional disease node labelled “Long COVID-19” (UMLS:C5433293), with literature-based data aiding in connecting some input genes to the disease, reflecting insights from Das and Kumar’s study (2023). Alongside these, we incorporated gene-specific annotations from GO. The resultant graph comprises 19 345 nodes and 92 483 edges. The PCS KG has been made available in different graph formats for community use via <https://github.com/BioDataFuse/pyBiodatafuse/tree/main/examples/usecases/PCS>. The RDF version can be easily uploaded to any RDF graph database or endpoint. [Figure 4](#) provides a sample visualization of part of the graph showing which drugs could potentially regulate genes involved with PCS.

3.4 Limitations

The *pyBiodatfuse* tool, while powerful, comes with several limitations that should be addressed for optimal use and future improvements. Firstly, the framework relies heavily on requesting data from upstream services via mechanisms such as SPARQL, APIs, GraphQL, or similar interfaces to build the KG. If an endpoint becomes temporarily unavailable, is deprecated, or

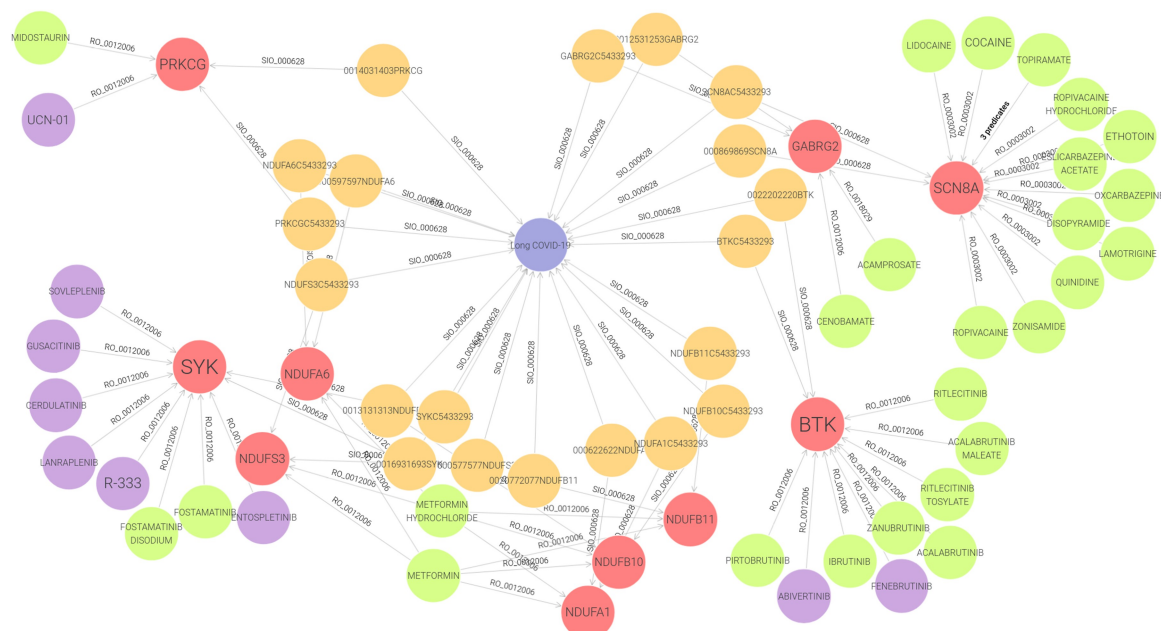


Figure 4 Visualization of an RDF subgraph in GraphDB. The subgraph centers on the “Long COVID” node and includes its associated gene–disease association nodes, the related genes, and drugs that inhibit, block, or act as inverse agonists of those genes. Both approved and non-approved drugs are represented. The SPARQL query used to construct the subgraph in GraphDB is available at: https://github.com/BioDataFuse/pyBiodatafuse/tree/main/examples/usecases/PCS/query_genes_drugs.rq.

is relocated, *pyBiodatafuse* cannot fetch the required data. There are currently no standardized practices for reporting new database releases, nor are such updates systematically collected by tool registries such as bio.tools. This highlights a broader community-wide gap in software dissemination and curation practices, which, while beyond the scope of this paper, warrants greater attention. This dependency can disrupt the KG generation process downstream and introduce bottlenecks; however, these incidents are caught and reported to the user in both the python package and UI versions of BioDataFuse. Secondly, each “data annotator” within the tool is highly customized to extract information from specific resources and databases. This customization necessitates a deep understanding of the underlying data models of these resources. Users without this expertise may face challenges in adapting or extending the framework to accommodate additional data sources. Thirdly, despite taking experimental data (e.g. differentially expressed genes) as input, the tool does not handle any raw data processing and expects users to preprocess their raw data presented in proprietary or non-proprietary formats. Lastly, the tool is tightly coupled with the data models of the resources it integrates. Any changes to these data models, such as schema updates, attribute renaming, or structural modifications, can significantly impact the tool’s functionality. This sensitivity requires continuous maintenance to ensure compatibility with evolving data sources. These limitations highlight the need for strategies to improve the robustness and scalability of *pyBiodatafuse*, such as incorporating redundancy in endpoint access, developing more adaptable data annotators, and implementing automated tools to detect and adapt to changes in data models.

4 Discussions

KGs are comprehensive representations of data and enable the capturing and harmonising of heterogeneous, multi-modal data with explicit semantics. Previously, generating such KGs required informaticians and engineering experts, thus restricting the ability to build graphs in an experimental lab setting. Frameworks like BioCypher were developed to decentralize KG generation, but these tools still require sufficient technical expertise for utility. The BioDataFuse distinguishes itself from other tools by providing a consistent modular schema, completely abstracting the knowledge engineering task from the process of generating knowledge graphs to ensure continuous integration. Since the graph is populated with the requested node types only, the resulting schema utilizes the relevant set of nodes and predicates for the customized graph.

In this work, we proposed the *pyBiodatafuse* tool that allows users to generate context-specific graphs irrespective of their technical expertise. This tool has significantly advanced the capability for on-the-fly graph building by enabling modular querying across endpoint-independent resources and databases. By eliminating dependency on specific endpoints, the tool ensures greater flexibility and adaptability in integrating diverse datasets. With its community-driven model, we aim to onboard experts and data owners to continually expand and diversify the range of data sources integrated into the tool. This collaborative approach not only enhances the breadth of data but also fosters

innovation through shared expertise. In this context, we are now looking at the prospect of integration of the tool into the Biocypher community ecosystem (<https://biocypher.org/>).

Looking ahead, we aspire to add prediction models, such as DreamWalk (Bang *et al.* 2023), directly into our resources. This initiative will provide two key advantages: (i) wider applicability and usability of the models through Python packaging and (ii) direct plugins to model input data (i.e., graphs), allowing for completing the loop from data to predictions. Beyond prediction modeling, we plan to enrich the tool by supporting additional input types, such as outputs from gene co-expression analysis or mass spectrometry data, to broaden its analytical capabilities. Another critical enhancement will be enabling efficient cross-species data comparisons, such as between humans and model organisms like mice. This feature will significantly extend the tool’s utility across diverse biological contexts, paving the way for innovative research and discovery. An additional improvement point revealed by the addition of several different annotators, which had to extend support for a variety of controlled vocabularies, is the lack of mapping efforts between tightly connected ontologies and vocabularies, like matching the Key Events from the AOP framework to the adverse events in the Ontology of Adverse Events (He *et al.* 2014) or Gene Ontology processes. Finally, to speed up data annotator building, we envision connecting schema extraction tools such as VoID generation, RDF-Config, SheXer, and others, allowing for the generation of an annotator template in a semi-automated fashion (Millán Acosta *et al.* 2025). These planned developments position *pyBiodatafuse* as a comprehensive and versatile biological data integration and analysis framework.

Acknowledgements

We would like to thank Denise Slenter (ORCID: 0000-0001-8449-1318) and the members of the BioDataFuse project for their scientific advice and discussions in the generation of the workflow. Additionally, we would like to thank the ELIXIR BioHackathon Europe 2023 and 2024 communities for selecting our project and for providing us with the platform to network and build a community around the tool. Part of the work was conducted by the Advanced Compute and Data Core at the Amsterdam UMC, sponsored by NL-Biolmaging: NWO 184.036.012. The integration of MolMeDB and PubChem was carried out with the support of ELIXIR CZ Research Infrastructure (ID LM2023055, MEYS CR) and Palacký University (IGA_PrF_2026_002). We also extend our thanks to Amin Aghabeiki, Saeed Masteri Farahani, and Youp Hendriks for their assistance in upgrading the user interface.

Author contributions

Yojana Gadiya (Conceptualization [lead], Data curation [lead], Formal analysis [lead], Funding acquisition [lead], Investigation [lead], Methodology [lead], Project administration [lead], Resources [lead], Software [lead], Supervision [supporting], Validation [lead], Visualization [lead], Writing—original draft [lead], Writing—review & editing [lead]), Javier Millán Acosta (Data curation [supporting], Formal analysis [supporting], Software [supporting], Validation [supporting], Visualization

[supporting], Writing—original draft [equal], Writing—review & editing [equal]), Ammar Ammar (Data curation [supporting], Methodology [supporting], Software [supporting], Validation [supporting], Visualization [supporting], Writing—review & editing [supporting]), Alejandro Adriaque Lozano (Formal analysis [equal], Software [equal], Writing—review & editing [equal]), Dominik Martinát (Software [equal], Writing—review & editing [equal]), Ana Claudia Sima (Software [equal], Writing—review & editing [equal]), Leon Mei (Writing—review & editing [equal]), Egon Willighagen (Software [equal], Writing—review & editing [equal]), and Tooba Abbassi-Daloi (Conceptualization [lead], Data curation [lead], Formal analysis [lead], Funding acquisition [lead], Investigation [lead], Methodology [lead], Project administration [lead], Resources [lead], Software [lead], Supervision [lead], Validation [lead], Visualization [lead], Writing—original draft [lead], Writing—review & editing [lead])

Conflict of interests

None declared.

Funding

None declared.

Data availability

The data and code underlying this study are available at <https://github.com/BioDataFuse/pyBiodatafuse>. A permanent archived version is available at Zenodo (doi: <https://doi.org/10.5281/zenodo.18468942>).

References

- Agrawal A, Balci H, Hanspers K *et al*. WikiPathways 2024: next generation pathway database. *Nucleic Acids Res* 2024;**52**: D679–89. <https://doi.org/10.1093/nar/gkad960>
- Angles R, Arenas M, Barceló P *et al*. Foundations of modern query languages for graph databases. *ACM Comput Surv* 2018; **50**:1–40. <https://doi.org/10.1145/3104031>
- Aschman T, Wyler E, Baum O *et al*. Post-COVID exercise intolerance is associated with capillary alterations and immune dysregulations in skeletal muscles. *Acta Neuropathol Commun* 2023;**11**:193. <https://doi.org/10.1186/s40478-023-01662-2>
- Bang D, Lim S, Lee S *et al*. Biomedical knowledge graph learning for drug repurposing by extending guilt-by-association to multiple layers. *Nat Commun* 2023;**14**:3570. <https://doi.org/10.1038/s41467-023-39301-y>
- Bastian FB, Roux J, Niknejad A *et al*. The bgee suite: integrated curated expression atlas and comparative transcriptomics in animals. *Nucleic Acids Res* 2021;**49**:D831–47. <https://doi.org/10.1093/nar/gkaa793>
- Bauer F. Linked open data: the essentials. Vienna: Edition Mono/ Monochrom, Vol. 710, 2011.
- Belleau F, Nolin M-A, Tourigny N *et al*. Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *J Biomed Inform* 2008;**41**:706–16. <https://doi.org/10.1016/j.jbi.2008.03.004>
- Bisong E. Matplotlib and seaborn. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress 2019, 151–65. https://doi.org/10.1007/978-1-4842-4470-8_12
- Bizon C, Cox S, Balhoff J *et al*. ROBOKOP KG and KGB: integrated knowledge graphs from federated sources. *J Chem Inf Model* 2019;**59**:4968–73. <https://doi.org/10.1021/acs.jcim.9b00683>
- Bonner S, Barrett IP, Ye C *et al*. A review of biomedical datasets relating to drug discovery: a knowledge graph perspective. *Brief Bioinform* 2022;**23**:404. <https://doi.org/10.1093/bib/bbac404>
- Brundege JM, Dubay C. BioQuery: an object framework for building queries to biomedical databases. *Bioinformatics* 2003;**19**: 901–2. <https://doi.org/10.1093/bioinformatics/btg094>
- Burkhardt M. The evolution of Facebook's graph API. In: *AoIR Selected Papers of Internet Research*. Journals@UIC, 2020, 1–4. <https://doi.org/10.5210/spir.v2020i0.11185>
- Chen Y-A, Tripathi LP, Mizuguchi K. TargetMine, an integrated data warehouse for candidate gene prioritisation and target discovery. *PLoS One* 2011;**6**:e17844. <https://doi.org/10.1371/journal.pone.0017844>
- Chen B, Dong X, Jiao D *et al*. Chem2Bio2RDF: a semantic framework for linking and data mining chemogenomic and systems chemical biology data. *BMC Bioinformatics* 2010;**11**:255–13. <https://doi.org/10.1186/1471-2105-11-255>
- Date CJ. *A Guide to the SQL Standard*. Reading Massachusetts: Addison-Wesley Longman Publishing Co., Inc, 1989.
- Del Toro N. The IntAct database: efficient access to fine-grained molecular interaction data. *Nucleic Acids Res* 2022;**50**:648–53.
- Doğan T, Atas H, Joshi V *et al*. CROssBAR: comprehensive resource of biomedical relations with knowledge graph representations. *Nucleic Acids Res* 2021;**49**:e96. <https://doi.org/10.1093/nar/gkab543>
- Drysdale R, Cook CE, Petryszak R *et al*. The ELIXIR core data resources: fundamental infrastructure for the life sciences. *Bioinformatics* 2020;**36**:2636–42. <https://doi.org/10.1093/bioinformatics/btz959>
- Dumontier M. The semanticscience integrated ontology (SIO) FOR biomedical research and knowledge discovery. *Journal of Biomedical Semantics* 2014;**5**:14. <https://doi.org/10.1186/2041-1480-5-14>
- Emsley R. ChatGPT: these are not hallucinations—they're fabrications and falsifications. *Schizophrenia (Heidelb)* 2023;**9**:52. <https://doi.org/10.1038/s41537-023-00379-4>
- Gadiya Y. BioHackEU23 report: extending interoperability of experimental data using modular queries across biomedical resources. BioHackrXiv, 2023, preprint: not peer reviewed.
- Gawron P, Ostaszewski M, Satagopam V *et al*. MINERVA—a platform for visualization and curation of molecular interaction networks. *NPJ Syst Biol Appl* 2016;**2**:16020–6. <https://doi.org/10.1038/npijsba.2016.20>
- Hagberg A. *Networkx: Network Analysis With Python*. 2020. <https://networkx.github.io>.
- He Y, Sarntivijai S, Lin Y *et al*. OAE: the ontology of adverse events. *J Biomed Semantics* 2014;**5**:29. <https://doi.org/10.1186/2041-1480-5-29>
- Jackson R, Matentzoglou N, Overton JA *et al*. OBO foundry in 2021: operationalizing open data principles to evaluate ontologies. *Database* 2021;**2021**:baab069. <https://doi.org/10.1093/database/baab069>

- Juračka J, Šrejber M, Melíková M *et al.* MolMeDB: molecules on membranes database. *Database* 2019;**2019**:baz078. <https://doi.org/10.1093/database/baz078>
- Kanehisa M, Furumichi M, Tanabe M *et al.* KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Res* 2017;**45**:D353–61. <https://doi.org/10.1093/nar/gkw1092>
- Kawashima S, Katayama T, Hatanaka H *et al.* NBDC RDF portal: a comprehensive repository for semantic data in life sciences. *Database* 2018;**2018**:1–11. <https://doi.org/10.1093/database/bay123>
- Kim S, Chen J, Cheng T *et al.* PubChem 2023 update. *Nucleic Acids Res* 2023;**51**:D1373–80. <https://doi.org/10.1093/nar/gkac956>
- Kolberg L, Raudvere U, Kuzmin I *et al.* Profiler—interoperable web service for functional enrichment analysis and gene identifier mapping (2023 update). *Nucleic Acids Res* 2023;**51**:W207–12. <https://doi.org/10.1093/nar/gkad347>
- Koscielny G, An P, Carvalho-Silva D *et al.* Open targets: a platform for therapeutic target identification and validation. *Nucleic Acids Res* 2017;**45**:D985–94. <https://doi.org/10.1093/nar/gkw1055>
- Liska O, Bohár B, Hidas A *et al.* TFLink: an integrated gateway to access transcription factor–target gene interactions for multiple species. *Database* 2022;**2022**:083. <https://doi.org/10.1093/database/baac083>
- Lobentanzer S, Aloy P, Baumbach J *et al.* Democratizing knowledge representation with BioCypher. *Nat Biotechnol* 2023;**41**:1056–9. <https://doi.org/10.1038/s41587-023-01848-y>
- Lopez-Leon S, Wegman-Ostrosky T, Perelman C *et al.* More than 50 long-term effects of COVID-19: a systematic review and meta-analysis. *Sci Rep* 2021;**11**:16144–12. <https://doi.org/10.1038/s41598-021-95565-8>
- Martens M, Evelo CT, Willighagen EL. Providing adverse outcome pathways from the AOP-Wiki in a semantic web format to increase usability and accessibility of the content. *Appl In Vitro Toxicol* 2022;**8**:2–13. <https://doi.org/10.1089/aivt.2021.0010>
- Mazein I, Rougny A, Mazein A *et al.* Graph databases in systems biology: a systematic review. *Brief Bioinform* 2024;**25**:bbae561. <https://doi.org/10.1093/bib/bbae561>
- Mering C. STRING: a database of predicted functional associations between proteins. *Nucleic Acids Research* 2003;**31**:258–61. <https://doi.org/10.1093/nar/gkg034>
- Millán Acosta J, Kawashima S, Katayama T *et al.* BioHackEU24 report: expanding FAIR database integration through elucidation and transformation of underlying graph schemas. *BioHackRxiv*, https://doi.org/10.37044/osf.io/ptmg5_v1, 2025, preprint: not peer reviewed.
- Morris JH, Soman K, Akbas RE *et al.* The scalable precision medicine open knowledge engine (SPOKE): a massive knowledge graph of biomedical information. *Bioinformatics* 2023;**39**:080. <https://doi.org/10.1093/bioinformatics/btad080>
- Ochoa D, Hercules A, Carmona M *et al.* Open targets platform: supporting systematic drug–target identification and prioritisation. *Nucleic Acids Res* 2021;**49**:D1302–10. <https://doi.org/10.1093/nar/gkaa1027>
- Piñero J, Ramírez-Anguita JM, Saüch-Pitarch J *et al.* The DisGeNET knowledge platform for disease genomics: 2019 update. *Nucleic Acids Res* 2020;**48**:D845–55. <https://doi.org/10.1093/nar/gkz1021>
- Queralt-Rosinach N, Stupp GS, Li TS *et al.* Structured reviews for data and knowledge-driven research. *Database* 2020;**2020**:1–12. <https://doi.org/10.1093/database/baaa015>
- Rath S, Sharma R, Gupta R *et al.* MitoCarta3. 0: an updated mitochondrial proteome now with Sub-organelle localization and pathway annotations. *Nucleic Acids Res* 2021;**49**:D1541–47. <https://doi.org/10.1093/nar/gkaa1011>
- Shannon P, Markiel A, Ozier O *et al.* Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res* 2003;**13**:2498–504. <https://doi.org/10.1101/gr.1239303>
- Sievert C. *Interactive Web-Based Data Visualization With R, Plotly, and Shiny*. Chapman and Hall/CRC Press; 2020.
- Smith RN, Aleksic J, Butano D *et al.* InterMine: a flexible data warehouse system for the integration and analysis of heterogeneous biological data. *Bioinformatics* 2012;**28**:3163–5. <https://doi.org/10.1093/bioinformatics/bts577>
- Tosi S. *Matplotlib for Python developers*. Vol. 307. Birmingham, UK: Packt Publishing Ltd; 2009.
- Van Iersel MP, Pico AR, Kelder T *et al.* The BridgeDb framework: standardized access to gene, protein and metabolite identifier mapping services. *BMC Bioinformatics* 2010;**11**:5. <https://doi.org/10.1186/1471-2105-11-5>
- Vrandečić D. 'Wikidata: A new platform for collaborative data collection'. In: *Proceedings of the 21st International Conference on World Wide Web*. New York, NY: Association for Computing Machinery, 2012, 1063–1064. <https://doi.org/10.1145/2187980.2188242>
- Waagmeester A, Stupp G, Burgstaller-Muehlbacher S *et al.* Wikidata as a knowledge graph for the life sciences. *Elife* 2020;**9**:52614. <https://doi.org/10.7554/eLife.52614>
- Webber J. A programmatic introduction to Neo4j. In: *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*. New York, NY: Association for Computing Machinery, 2012, 217–218. <https://doi.org/10.1145/2384716.2384777>
- Williams AJ, Harland L, Groth P *et al.* Open PHACTS: semantic interoperability for drug discovery. *Drug Discov Today* 2012;**17**:1188–98. <https://doi.org/10.1016/j.drudis.2012.05.016>