OXFORD

Databases and ontologies

KG-Hub—building and exchanging biological knowledge graphs

J. Harry Caufield [1], Tim Putman[2], Kevin Schaper[2], Deepak R. Unni[3], Harshad Hegde[1],
Tiffany J. Callahan[4], Luca Cappelletti[5], Sierra A.T. Moxon[1], Vida Ravanmehr[6], Seth Carbon[1],
Lauren E. Chan[7], Katherina Cortes [2], Kent A. Shefchek[2], Glass Elsarboukh[2], Jim Balhoff[8],
Tommaso Fontana[9], Nicolas Matentzoglu[10], Richard M. Bruskiewich[11], Anne E. Thessen [2],
Nomi L. Harris [1], Monica C. Munoz-Torres[2], Melissa A. Haendel [2], Peter N. Robinson [12],
Marcin P. Joachimiak[1], Christopher J. Mungall [1], Justin T. Reese [1,*]

[1]Division of Environmental Genomics and Systems Biology,
Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States
[2]Anschutz Medical Campus, University of Colorado, Aurora, CO 80045, United States
[3]SIB Swiss Institute of Bioinformatics, Basel 1015, Switzerland
[4]Department of Biomedical Informatics, Columbia University Irving Medical Center, New York, NY 10032, United States
[5]Department of Computer Science, University of Milano, Milan 20126, Italy
[6]Department of Lymphoma-Myeloma, MD Anderson Cancer Center, Houston, TX 77030, United States
[7]College of Public Health and Human Sciences, Oregon State University, Corvallis, OR 97331, United States
[8]Renaissance Computing Institute, University of North Carolina, Chapel Hill, NC 27517, United States
[9]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan 20133, Italy
[10]Semantically, Athens, Greece
[11]STAR Informatics, Delphinai Corporation, Sooke, BC V9Z 0M3, Canada
[12]The Jackson Laboratory for Genomic Medicine, Farmington, CT 06032, United States

*Corresponding author. Division of Environmental Genomics and Systems Biology, Lawrence Berkeley National Laboratory, 717 Potter Street, Berkeley, CA 94720, United States. E-mail: justinreese@lbl.gov (J.T.R.)

Associate Editor: Christina Kendziorski

Abstract

Motivation: Knowledge graphs (KGs) are a powerful approach for integrating heterogeneous data and making inferences in biology and many other domains, but a coherent solution for constructing, exchanging, and facilitating the downstream use of KGs is lacking.

Results: Here we present KG-Hub, a platform that enables standardized construction, exchange, and reuse of KGs. Features include a simple, modular extract–transform–load pattern for producing graphs compliant with Biolink Model (a high-level data model for standardizing biological data), easy integration of any OBO (Open Biological and Biomedical Ontologies) ontology, cached downloads of upstream data sources, versioned and automatically updated builds with stable URLs, web-browsable storage of KG artifacts on cloud infrastructure, and easy reuse of transformed subgraphs across projects. Current KG-Hub projects span use cases including COVID-19 research, drug repurposing, microbial–environmental interactions, and rare disease research. KG-Hub is equipped with tooling to easily analyze and manipulate KGs. KG-Hub is also tightly integrated with graph machine learning (ML) tools which allow automated graph ML, including node embeddings and training of models for link prediction and node classification.

Availability and implementation: https://kghub.org.

1 Introduction
1.1 Knowledge graphs: successes and challenges

Addressing scientific challenges such as climate change and treatment of complex or rare diseases requires integration of heterogeneous data from multiple disciplines. These datasets differ in terminology, units, granularity, and perspective, among other factors, and are difficult to combine using traditional relational databases. Knowledge graphs (KGs), which offer a more flexible and powerful way to link together heterogeneous datasets, are increasingly used to integrate data in various domains including biology, ecology, biomedicine, and personalized health (Poelen *et al.* 2014, Nickel *et al.* 2015, Su

et al. 2020). KGs represent entities (e.g. genes, diseases, phenotypes) as nodes in a graph, and relationships between these entities (e.g. gene to disease relationships) as edges between nodes. This enables the application of new types of analyses on biological data, such as network analysis and machine learning (ML). The structure of the KG emphasizes the relationships between entities, which in biology is important for understanding complex systems and is often lost in more traditional data formats.

Despite the demonstrated usefulness of KGs, barriers exist that limit their effectiveness and reusability. KGs are often represented in proprietary or nonstandard formats, and data

models frequently differ between KGs, which greatly limits their interoperability. Biological KGs typically lack standard procedures for ID normalization (Badal *et al.* 2019), graph representation (multiple competing formats and knowledge representation paradigms exist) (Chaves-Fraga *et al.* 2019), data source and transformation provenance, and change tracking between KG versions (Issa *et al.* 2021). KG-based research also imposes computational challenges, since analyzing a typical graph with a million+ edges can require substantial memory and CPU/GPU resources (Zeng *et al.* 2022).

Here, we present a solution, KG-Hub, built to address these challenges and accelerate the construction and reuse of KGs by developing tooling and design patterns to encourage FAIR-ness, and integrating with tooling such as GRAPE to make analysis of large KGs computationally feasible (Cappelletti *et al.* 2021).

1.2 Biological and biomedical KGs

While relatively new in biology, KGs have been used in other disciplines for decades, e.g. WordNet [1985 (Miller *et al.* 1990)], Geonames (https://www.geonames.org/), DBpedia [2007 (Auer *et al.* 2007)] and the Google KG [2012 (Singhal 2012)]. KGs are well-suited for biomedical and life science applications (Li *et al.* 2022), since these fields involve highly varied types and sources of data with complex interrelationships. Graph representation of these heterogeneous relationships can generate standardized concepts and entities from multiple data sources. Previously developed biological and biomedical KG platforms address parts of this data integration process, but not the full extent of KG construction from data ingestion to reusable graph data resources.

Various strategies have been developed for constructing and applying KGs (Nicholson and Greene 2020). In some cases, such as protein–protein interactions, the association between entities has a clear analog in a graph model: each protein is a node, while interactions between them are edges. In other contexts, the translation from data to graph requires data modeling rules. In a graph of scientific publications, e.g. authorship may be represented as a connection between "author" and "paper" nodes or author names may be stored as properties of "paper" nodes, with edges representing citations. Some KGs, such as SPOKE (Nelson *et al.* 2019) (https://spoke.ucsf.edu/), Harmonizome (Rouillard *et al.* 2016) (https://maayanlab.cloud/Harmonizome/), CROssBAR (Doğan *et al.* 2021), and RTX-KG2 (Wood *et al.* 2021) unify and de-silo isolated data resources ranging from biomolecular interactions to disease risk factors and phenotypes. Other KGs are built on relationships extracted from unstructured text or from computational inference. For example, EMMAA (Ecosystem of Machine-maintained Models with Automated Analysis) assembles disease-specific graphs from published statements describing drug, gene, protein, and disease associations (INDRALAB, https://emmaa.indra.bio/), while BioKDE (Pang *et al.* 2018) supports literature search by connecting related concepts. Edges may also be extracted directly from biomedical literature with rule-based or natural language processing (NLP) approaches.

1.3 Incorporating ontologies into KGs

Ontologies provide a convenient and standardized way to add domain knowledge to a KG. There are ontologies that represent knowledge in many biological and biomedical domains; many are freely available through projects such as the OBO (Open Biological and Biomedical Ontologies) Foundry (Jackson *et al.* 2021) and BioPortal (Whetzel *et al.* 2011). Adding ontologies provides valuable context to nodes and edges represented in KGs. For example, in a KG concerning dietary habits, the FOODON food ontology (Dooley *et al.* 2018) can provide hierarchical relationships defining both raspberry jam (FOODON:03305865) and bitter orange marmalade (FOODON:03306375) as "fruit preserve or jam food product" (FOODON:00001226). The Human Phenotype Ontology (Köhler *et al.* 2021) can be used to link individuals with a consistent set of phenotypes, e.g. abdominal pain (HP:0002027). KGs complemented with ontologies capture more of an entity's features and meaning, support complex queries (e.g. "which individuals represented in the KG ate any kind of fruit product *and* experienced abdominal pain?") and can reveal structural patterns. Including ontology data supports approaches to detect "emergent" and otherwise nonobvious associations between nodes. In the above example, incorporating diseases from the Mondo disease ontology (Vasilevsky *et al.* 2020) into the KG could reveal connections between digestive system disorders (MONDO:0004335) and diet.

Ontologies can be difficult to combine with each other and with instance data within KGs. Two or more resources may define concepts in conflicting ways, such as how the ChEBI ontology (Hastings *et al.* 2016) defines penicillin (CHEBI:17334) primarily in terms of its chemical identity, while DrugCentral (Ursu *et al.* 2017) has no specific entry for "penicillin", instead defining entries for benzylpenicillin (DrugCentral:2082) and derivatives such as ampicillin (DrugCentral:198). The potential for semantic conflict arises from merging ontology classes with instance data, e.g. the concept of "lung cancer" versus a specific lung cancer of an individual patient. Ontologies generally define a broader representation of a domain than a single set of observations is likely to cover, yet no single ontology captures the entirety of biology. A patient may experience all the symptoms consistent with a disease but not have the formal diagnosis, or a particular mutation in a given gene may not produce an identical phenotype in each case. Furthermore, our biological knowledge remains incomplete: just a subset of the phenotypes of a particular gene variant may have been observed for any individual due to multitudinous factors, such as age, genetic background, or environmental exposures. These variations present obstacles to harmonization.

1.4 Frameworks and registries for biomedical KGs

Several frameworks exist for constructing and disseminating biological and biomedical KGs. PheKnowLator serves as a framework for standardized KG assembly with consistent formats (Callahan *et al.* 2020), though with emphasis on adhering to Semantic Web conventions rather than a specific data model. RTX-KG2 (Wood *et al.* 2021) and the Knowledge Graph Exchange (KGE) Registry, both part of the NCATS Biomedical Translator project, permit researchers to assemble graphs whose contents remain intelligible and interoperable with others. Currently, creation and analysis of these graphs in NCATS Translator is closely tied to the Translator ecosystem. The Network Data Exchange (NDEx) project (Pillich *et al.* 2017) brings together a standard and platform for exchanging graph data, including support for Cytoscape graph visualization software. The NDEx supports KG dissemination but is not generally aligned with specific data models, an element we see as crucial to reproducible graph integration and analysis.

1.5 Graph machine learning on KGs

New knowledge can be derived from KGs using graph machine learning (graph ML). Drug repurposing is a popular use case for graph ML on KGs, as observations about drugs in therapeutic use can be integrated with basic research data from many other sources (disease to phenotype associations, protein–protein interaction data, gene functions) that provide more context for ML models. There have been notable successes in predicting new protein targets for existing drugs using graph ML over the past several years (Sosa *et al.* 2020, Zeng *et al.* 2020, Smith *et al.* 2021). Similarly, there have been successes in using graph ML to predict associations between genes and diseases (Hu *et al.* 2021), genes and phenotypes (Mungall *et al.* 2017), or genes and functions. These efforts are supported by resources such as the Gene Ontology (GO); the inclusion of GO terms in a graph can enable function prediction using supervised graph ML (Glass and Girvan 2015). Graph ML on KGs has also been applied to more complex clinical challenges, such as inferring disease status given a set of symptoms and patient characteristics from electronic health record (EHR) data (Choi *et al.* 2017, Rotmensch *et al.* 2017, Ma *et al.* 2018, Shang *et al.* 2021) or prioritizing genes by their relevance to disease phenotypes (Peng *et al.* 2021, Yamaguchi *et al.* 2022). Recent applications have explored novel treatments for SARS-CoV-2 (Domingo-Fernández *et al.* 2021, Reese *et al.* 2021, Zhang *et al.* 2021), predicted multiple sclerosis diagnoses (Nelson *et al.* 2022), and provided much-needed context for protein biomarkers of fatty liver disease (Santos *et al.* 2022).

1.6 KG-Hub

KG-Hub is a collection of tools and libraries for building and reusing KGs (Supplementary Table S1). It includes software for building interoperable KGs (e.g. KGX, a toolkit for manipulating graph data) as well as a mechanism for sharing them (summarized below in "KG-Hub design patterns" and "Integrated downstream tooling"). KG-Hub provides a template and documentation for creating new KG projects that follow KG-Hub design patterns and a standardized data model, Biolink Model (Unni *et al.* 2022). These design patterns can also be used in other KG ecosystems to preserve data provenance, provide versioned builds, adhere to data models, and incorporate ontologies.

2 Materials and methods

2.1 Harmonizing data sources

A wide variety of sources are used to construct knowledge graphs in KG-Hub. To unify the representation of overlapping concepts, terms, and data structures in these often disparate sources, KG-Hub uses a data model, Biolink Model, to allow cross-source interoperation. Biolink Model is an open-source data model that provides a set of hierarchical, interconnected classes (or categories), and associations that guide how entities should relate to one another. Biolink represents a wide array of biomedical entities such as gene, disease, chemical, anatomical structure, and phenotype, and establishes mappings to existing biomedical ontologies and reuses existing ontology term definition in its structure (Unni *et al.* 2022). In addition, Biolink is an actively maintained standard that evolves quickly with the use cases generated for it. However, because Biolink is developed using the LinkML framework, LinkML tooling can aid in the process of subsetting or extending Biolink Model. For example, in

the BioCypher project (https://biocypher.org/tutorial.html), Biolink Model is used as an ontological backbone for all extended classes that may express more narrowly focused or detailed descriptions of the Biolink class hierarchy. While Biolink maintainers welcome additions and encourage extensions to make their way back into the main model, the extendable LinkML framework helps maintain an active community that is responsive to changing biology and adaptable to many disparate use cases. A more detailed description of how Biolink Model is used in KG-Hub is contained in the supplement (Supplementary Fig. S1).

2.2 Data downloading, transformation, and graph assembly

KGs are constructed with an extract, transform, and load (ETL) process driven by the KGX and Koza toolkits and the kghub-downloader module (https://github.com/monarch-initiative/kghub-downloader). KGX (Knowledge Graph Exchange) is a KG serialization standard supported by a Python library and command line utilities that help transform data into Biolink Model compliant graphs. Koza (https://github.com/monarch-initiative/koza) is a declarative transformation framework that uses user-defined templates to both document and define the parsing and transformation of data into KGX format. KG-Hub uses these tools to transform data sources into standalone Biolink Model compliant graphs. KG-Hub orchestrates the transformation using a *download.yaml* configuration file and a declarative transformation configuration (enabled by Koza), and a Python control script. The kghub-downloader module uses the *download.yaml* file to document and manage retrieval of the data from a variety of possible sources including a local file, a URL or via queries to an Elasticsearch API (e.g. ChEMBL API). The declarative transformation configuration (as documented in the Koza framework here: https://github.com/monarch-initiative/koza) and accompanying transformation script, process raw input data into KGX TSV format. KGX TSV format is a flattened serialization of the nodes and edges in a graph as tab-separated values. In addition, the declarative nature of the configuration file also serves as documentation and data instance validation for the transform (e.g. it reports an error if HP:0002362 *Shuffling gait*, a phenotype, were to be typed as a Biolink:Gene). An example of this process may be found at https://github.com/Knowledge-Graph-Hub/kg-example and in Supplementary Table S2. Koza transformations are modular, documented, reusable, and may cover multiple sources (e.g. the *ontology* transform in the *kg-example* example process is a single Python script capable of transforming any OWL ontology from the OBO Foundry). The final step in each graph's ETL process is to merge the individual transform products into one final graph. This step is handled by the KGX *merge* function and defined by a configuration file (*merge.yaml*). Once transformed, each subgraph and the merged graph are available from a centralized open data repository (in Amazon Web Services, AWS), allowing users to reuse, mix, and match subgraphs, as well as share the final merged graph.

By leveraging Biolink Model, KGX, and Koza, KG-Hub graphs produced by this framework are interoperable, consistently formatted, and use the same data structures to communicate knowledge.

2.3 Storage of graph and other data

The design of KG-Hub promotes reuse and reproduction of graph resources. Amazon Web Services (AWS) S3 is used to host files on KG-Hub, providing consistent uptime for KG-Hub. The data repository is available at https://kghub.io, and a description of the KG projects in KG-Hub is provided at https://kghub.org.

KG-Hub provides a default template to generate modular KGs, the kg-cookiecutter (https://github.com/Knowledge-Graph-Hub/kg-cookiecutter). The kg-cookiecutter guides users through a series of prompts to customize their KG project code repository, while relying on KG-Hub to specify best practices, provide basic configuration and setup, and wiring to deploy new KGs consistently. The kg-cookiecutter uses the more generic, cookiecutter framework to make its templates (https://cookiecutter.readthedocs.io/en/stable/). The kg-cookiecutter also provides example download and merge configuration files, as well as an example transformation script that specifies the preferred location for the resulting graph and data. It stores all project materials according to KG-Hub design patterns, ensuring that the directory for each project contains the KG itself and all necessary code, data, and configurations to rebuild the graph. Users are able to customize the rebuild frequency of their KG-Hub graph using a custom Jenkins configuration. Jenkins is a continuous integration framework that helps coordinate and build systems (https://www.jenkins.io/), where systems are broadly defined as containers for code coupled with data.

2.4 Querying and accessing KG-Hub resources

The availability of biomedical KGs in standard formats supports a variety of downstream use cases. KGs can be loaded into a Neo4j graph database (https://neo4j.com/) during the merge stage of the ETL process, or the KGX TSVs can be loaded directly into Neo4j using the KGX library. These Neo4j databases can then be queried using the Cypher query language (Francis *et al.* 2018). Some projects (e.g. KG-COVID-19) provide RDF serialization as well as Blazegraph journal files which may be queried using SPARQL query language (https://www.w3.org/TR/rdf-sparql-query/), a popular language for querying graph data in RDF format. KGX TSV files can also be loaded into a tool like Cytoscape for visualization, querying, and browsing.

3 Results

3.1 KG-Hub design patterns

Below we describe the design patterns (reusable solutions) we developed to address commonly occurring problems in the construction and use of KGs. These design patterns can be repurposed for other KG efforts.

3.1.1 Simple, modular ETL

The KG-Hub framework includes ETL Python code for transforming upstream data into KGs (https://github.com/Knowledge-Graph-Hub), with reusable, modular software for downloading data, transforming data into subgraphs, and merging subgraphs into KGs (Fig. 1). See Section 2 for further details. In brief, the download step retrieves and saves the upstream source data, the transform step ingests and converts each upstream source data into KGX TSV format (https://github.com/biolink/kgx/blob/master/specification/kgx-format.md#kgx-format-as-tsv), and the merge step combines the
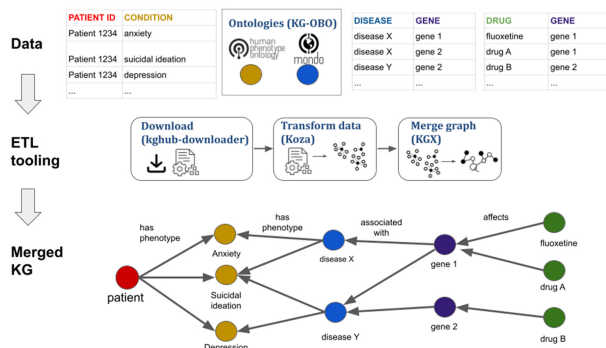

Figure 1. Integration of instance data and ontologies into knowledge graphs using KG-Hub ETL (extract, transform, and load) tooling to create new, emergent knowledge that is not present in any one data source. KG-Hub tooling comprises download (kghub-downloader), transform (Koza), and merge (KGX) components. When combined, data can provide new knowledge such as indirect relationships between patient phenotypes and drugs.

subgraphs from each upstream source into a single, merged KGX TSV. The transform step in some KG projects (e.g. KG-IDG) relies on Koza (https://github.com/monarch-initiative/koza), a Python package that facilitates the ingestion of data into KGX TSV format. The merge step uses KGX (https://github.com/biolink/kgx) to perform ID normalization and combine the subgraphs.

3.1.2 Graph representations within KG-Hub

All graphs in KG-Hub are represented as directed, heterogeneous property graphs. Edges and nodes are typed according to a data model, edges have direction (e.g. A "affects risk for" B is distinct from B "affects risk for" A), and both nodes and edges may have one or more properties (e.g. a node may have a name or a textual description, and an edge may have a reference to a paper that provides provenance). A property graph model offers the features needed for a variety of downstream applications, such as storage in Neo4j, while also remaining sufficiently flexible to transform products to other formats (e.g. n-triples or RDF/XML).

3.1.3 Biolink data model compliance

One of the identifying features of a KG built with KG-Hub is data harmonized according to Biolink Model. Domain knowledge in a KG that conforms to Biolink is represented using associations. An association minimally includes a subject and an object related by a Biolink Model predicate, together comprising its core triple (statement or primary assertion). A key step in KG development is identifying the concepts and relationships in Biolink Model that map to the data source being transformed. KG-Hub provides examples and guidance in selecting Biolink categories for any KG. In addition, KGX automatically assigns Biolink categories to data based on namespaces and mappings to external resources curated in Biolink Model. During data loading, nodes and edges are typed with Biolink categories and association types. With a single model that spans data sources and transformed KGs, KG-Hub facilitates analysis of KG contents in a clearly defined way, and aids interoperability of data between KG projects. For example, a KG of biolink:chemicals, biolink:proteins, and biolink:diseases can be filtered to just interactions between chemicals and proteins while retaining all subcategories. For use cases not covered by Biolink Model, users may extend the model or

even create a new LinkML data model (https://linkml.io), then use the new model in place of Biolink in KG-Hub's graph assembly pipeline.

3.1.4 Automated, self-updating, versioned builds with data provenance

In designing KG-Hub, we kept in mind that biomedical data sources are often volatile and may undergo massive updates at any time. Downloading upstream data is a frequent point of failure in bioinformatics ETL pipelines, due to issues such as changes in URLs and network instability. KG-Hub caches the most recent version of upstream data that was successfully downloaded for each data source to ensure that the ingestion can proceed even when these issues arise. KG-Hub captures data source versions and which version of its own software is used in each build and adds provenance to the data as it moves through the build process. Provenance tracking enables users and systems downstream to trace the transformation of data from its source to the resulting KG. KG-Hub also produces a permanent URL (PURL) for all artifacts in the build ensuring that downstream consumers can reproduce the results of each build even if the upstream sources have changed. Each versioned KG-Hub build, or pipeline run, is scheduled to run on a monthly basis for each KG project, using continuous integration/continuous delivery software (https://www.jenkins.io/). In addition, a new build is triggered with each update to the ETL software used in the pipeline. The build system provides error messages and guidance when volatile upstream data breaks the ETL code or fails validation.

3.1.5 Easy reuse of transformed subgraphs across projects

KG-Hub is designed to allow and encourage reuse of transformed data across different projects. Each KG project produces a subgraph representing the data from each of the upstream sources that it ingests and transforms. These subgraphs are stored separately in a subdirectory (*transformed*) in the build directory for each project, with PURLs. This design allows projects to easily ingest the transformed subgraph from an upstream data source that was transformed by a different KG-Hub project, eliminating duplication of effort and encouraging alignment of data across KG projects.

For example, a new KG project that wishes to incorporate STRING protein–protein interactions in the KG could simply reuse the transformed version of this ingest from KG-COVID-19 located here: https://kghub.io/kg-covid-19/current/transformed/STRING/ or pin to the November 2, 2022 version of the STRING ingest from KG-COVID-19 by reusing: https://kghub.io/kg-covid-19/20221102/transformed/STRING/.

3.1.6 Reuse of OBO ontologies

Ontologies provide a convenient means to incorporate knowledge from domains of interest into KGs in order to contextualize instance-level data. For example, to harmonize knowledge about human diseases and disease phenotypes, one might incorporate the Mondo disease ontology (Vasilevsky *et al.* 2020) and the Human Phenotype Ontology (HPO) (Köhler *et al.* 2021) into a KG. KG-OBO, a KG-Hub project that ingests and produces versioned builds of all ontologies in the OBO Foundry (Jackson *et al.* 2021), eases the incorporation and reuse of OBO ontologies. These versioned builds can be easily incorporated into other KG-Hub projects. Individual projects can utilize the freely available ROBOT

tool (http://robot.obolibrary.org/) as an interlocutor between OBO ontologies and their graph representation.

3.2 KG-Hub projects

KG-Hub currently includes seven biomedical KG projects that integrate data pertaining to COVID-19 biology (Reese *et al.* 2021), drugs and drug targets, microbial phenotypes (Joachimiak *et al.* 2021), and more. Each KG has a distinct set of sources, use cases, and domains. These projects exemplify KG-Hub's value as an open, general-purpose platform for exchanging biological and biomedical KGs. Figure 2 provides a summary of the types of data integrated in each project, and Table 1 provides a description of each of these projects. For each project, summary-level data (https://kghub.org/), a browsable interface with graph data and other artifacts (https://kghub.io/), and a GitHub organization for source code (https://github.com/knowledge-Graph-Hub/) are also available.

3.3 Integrated downstream tooling

KG-Hub integrates a variety of tools to facilitate functions that are frequently required downstream, such as storage, analysis and display of KG contents, conversion between different graph formats, cloud computing, and ML (Fig. 3). A detailed summary of all internal and external tooling that is integrated in KG-Hub is shown in Supplementary Table S1. Each module is described below.

3.3.1 Visualization and querying of KG contents

A surprisingly frequent problem in previous projects was the inability to easily assess KG contents. Since KG-Hub projects use Biolink during the transform step to type nodes and edges, KG contents can be easily inventoried and displayed. We implemented a dashboard in JavaScript to summarize the contents of KG projects, including node and edge types by source, as well as a Sankey plot showing the frequency of different pairs of node categories by source. An example of this dashboard is here: https://kghub.org/kg-hub-dashboard/.

KG-Hub projects typically emit the representation of the graph in several formats, including minimally KGX TSV format, but also RDF/ntriples and Blazegraph journal file formats. The Blazegraph journal file provides the ability to load the data into a Blazegraph instance, as well as to analyze and compare different builds of various KG projects using a tool
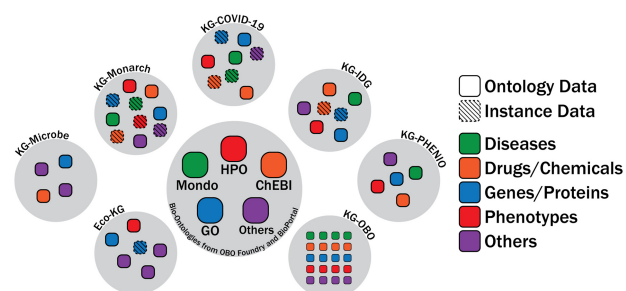

Figure 2. KG projects currently included in KG-Hub. KG-Hub currently hosts seven KG projects, which integrate disease, drug/chemical, gene/protein, phenotype, and other data. Graph projects may contain both ontology and instance data. Many KG-Hub projects are constructed around a core set of ontologies related to biomedicine: GO (gene ontology/gene function), Mondo (human diseases), HPO (human disease phenotypes), and ChEBI (drugs/chemicals).

Table 1. KG-Hub projects.[a]

Project name	Description	Size (nodes/edges), in thousands	Code repository URL
KG-COVID-19	Knowledge concerning SARS-CoV-2, SARS-CoV, and MERS-CoV, including viral interactions with human proteins (Reese *et al.* 2021). Sourced from 10 different data sources and 4 OBO ontologies, this KG was incorporated into the N3C Enclave (Bennett *et al.* 2021), was used in the NVBL (https://science.osti.gov/nvbl) project to provide integrated publicly available data relevant to COVID-19, and has been used to identify drugs that may affect COVID-19 outcome (Chan *et al.* 2022, Reese *et al.* 2022).	574/24 145	https://github.com/Knowledge-Graph-Hub/kg-covid-19
KG-Microbe	Data about microbial traits, environment types, carbon substrates, and taxonomy. Its contents unite bacterial and archeal phenotypes across a broad range of species, supporting identification of common metabolic and environmental patterns.	276/535	https://github.com/Knowledge-Graph-Hub/kg-microbe
KG-IDG	A graph assembled to support the Illuminating the Druggable Genome (IDG) project (https://druggablegenome.net/), with the objective of characterizing poorly-understood members of protein families that are frequently targeted by approved drugs. KG-IDG unifies structured data from 14 different sources concerning drugs, proteins, and diseases.	560/4431	https://github.com/Knowledge-Graph-Hub/kg-idg.
KG-OBO	A collection of OBO Foundry (https://obofoundry.org/) ontologies transformed into obograph JSON and graph-compatible KGX formats. 201 ontologies are currently included, many with multiple versions.	N/A	https://github.com/Knowledge-Graph-Hub/kg-obo.
ecoKG	Plant genes and traits, spanning 46 different species, with the objective of exploring gene, phenotype, and environment interactions.	400/5000	https://github.com/Knowledge-Graph-Hub/eco-kg.
KG-Monarch	A project to integrate data relevant to human diseases, especially rare diseases (Shefchek *et al.* 2020) (https://monarchinitiative.org). This includes 12 biomedical ontologies such as HPO, Mondo, and GO, data regarding human genes, diseases, phenotypes versus gene expression associations, as well as a range of data from many model organism databases.	794/6970	https://github.com/monarch-initiative/monarch-ingest
KG-Phenio	A KG representation of the Phenomics Integrated Ontology (PHENIO) (https://github.com/monarch-initiative/phenio), a resource combining more than 20 ontologies relevant to phenotype-driven biomedical research.	275/1183	https://github.com/Knowledge-Graph-Hub/kg-phenio

[a] For each of the seven KG-Hub projects, a description, size, and link to the project source code is provided.

such as Blazegraph runner (https://github.com/balhoff/blazegraph-runner).

3.3.2 Integration with KGX graph utility

KGX is a tool for working with graph data that provides utilities for converting between common graph formats, extracting subgraphs, merging graphs, and loading data into graph databases (https://github.com/biolink/kgx). KG-Hub projects store graph data in KGX TSV format (https://github.com/biolink/kgx/blob/master/specification/kgx-format.md) and are therefore natively compatible with KGX. In addition to using KGX in the ETL process (in the merge step to combine subgraphs from different upstream sources), it can be used downstream to convert to other data formats, and also to load a Neo4j graph database.

3.3.3 Integration with graph ML tooling

A frequent downstream use case for KGs is the application of graph ML to derive new insights. To support downstream ML use cases, we have tightly integrated with GRAPE (Cappelletti *et al.* 2021), a performant graph ML package. GRAPE can import KG-Hub graphs programmatically through the API. For example, the most current KG-COVID-19 graph can be imported as follows:

```
from grape.datasets.kghub import KGCOVID19
graph = KGCOVID19()
```

This API can also retrieve a specific build for a given KG-Hub project:
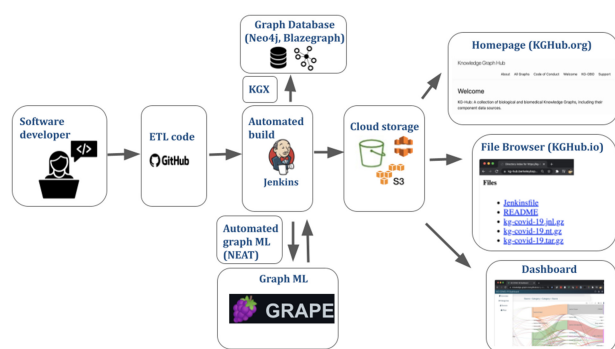
```
graph = KGCOVID19(version="20210727")
```

Figure 3. Schematic of tooling integrated into KG-Hub. Software developers store ETL code on GitHub. Automated builds are orchestrated on a KG-Hub server using Jenkins. Optionally, graph ML tasks can be specified for each build using NEAT yaml, and are executed using GRAPE. KGs can be directly loaded into graph databases (Neo4j, Blazegraph) using KGX, a Python library for working with graphs. Graph builds, graph ML output, provenance, and other artifacts are stored on the cloud (S3). Project summary data can be browsed on KGHub.org, and graphs and other artifacts can be browsed and downloaded on KGHub.io. A dashboard (https://kghub.org/kg-hub-dashboard/) displays detailed graph statistics for KG projects.

and a list of all KG-Hub projects that are available through the API:

```
from grape.datasets import
    get_available_graphs_from_repository
get_available_graphs_from_repository('kghub')
```

Tight integration with GRAPE allows KG-Hub graphs to be easily downloaded, queried, and analyzed, and also allows procedures such as node embedding and graph neural networks to be applied to these KGs. In addition to reducing the difficulty in finding and accessing KG data, integration of KG-Hub with GRAPE addresses the computational challenges associated with analyzing large KGs, as GRAPE is able to efficiently load and train graph ML models on very large KGs (Cappelletti *et al.* 2021).

NEAT (https://github.com/Knowledge-Graph-Hub/neat-ml) is a Python package that allows graph ML algorithms to be applied to KG-Hub projects in a simple, declarative, YAML-driven way. This YAML file serves to describe the exact task, and after completion serves as provenance and documentation. An example YAML file used in the KG-IDG project is shown in Supplementary Fig. S2. NEAT in turn uses GRAPE to execute the graph ML task.

3.3.4 Integration with cloud computing services

All KG-Hub artifacts are stored on cloud storage (AWS S3). During the build process, index.html files are constructed to make these artifacts browsable, and these are made public using a content delivery solution (AWS CloudFront). To run ML tasks, we have integrated with a cloud computing platform, Google Cloud Platform (GCP). Within our continuous integration software configuration (Jenkins), we create new GCP instances to run the ML task, after which ML artifacts are uploaded to our browsable S3 cloud storage An automated scheduler script runs each pipeline (NEAT, etc.) on GCP and returns results to the project directory on KG-Hub.

3.3.5 KG construction template

We have created a template to facilitate new KG projects (https://github.com/Knowledge-Graph-Hub/kg-cookiecutter). In addition to providing consistent project structure, the template includes example Python code for ETL functions, as well as tests and automation configurations.

3.4 Minimum requirements for new projects

The minimal necessary requirements for a new, KG-Hub-compatible project are modest by design: the project must be a biological KG and must make at least one KG build available in KGX format.

Additional guidelines serve to enhance a project's integration with the broader KG-Hub system. A project should:

- Have its own code repository within the Knowledge-Graph-Hub GitHub organization or plan to move an existing code repository there.
- Include ETL code and configurations that produce a KG.
- Provide a KG for public download, following semantic versioning guidelines.
- Provide the KG in additional serialization formats as needed (e.g. n-triples or obograph JSON (https://zenodo.org/record/5070191)

Where appropriate, each KG should model data using Biolink Model and utilize ontologies from the OBO Foundry. Each project's creator is responsible for confirming the accuracy of the datasets composing their KG and keeping track of evidence and provenance for assertions within their KG. We also strongly encourage each new project to include documentation describing the KG's intended applications, its contributors, contribution guidelines, a code of conduct, and an open license agreement.

No KG project is required to implement all of these patterns, but this enables integration with KG-Hub features. For example, automated indexing of graph components and statistics is only possible when graph components are in expected formats and predetermined locations. Data sharing limitations may also preclude following KG-Hub guidelines. In some graph assembly cases, development time may be limited, or researchers may elect to use previously defined strategies for obtaining data, preprocessing it, or producing a merged graph. KG-Hub's modular nature fully supports these design choices as long as graph data is transformed to KGX TSV format. For example, both the ecoKG and Monarch graphs (see Table 1) incorporate specific data processing procedures to adapt KG-Hub to their exact requirements.

3.5 Process for creating a new KG-Hub project

Assembling a KG-Hub project entails configuration, building, and analysis. Starting with the project template, a new project must be set up to retrieve data sources, transform them as needed, and merge the resulting subgraph products. The additional code necessary to perform a given transformation may be minimized by using the existing subgraphs provided on KG-Hub (e.g. DrugCentral drug–target interactions from KG-COVID-19; ontologies such as HPO, GO, or ChEBI from KG-OBO, or even entire other KG projects). After merging transformed sources with KGX, the resulting graph may then

be examined and analyzed further with GRAPE and NEAT, as described above.

As all KG-Hub components are open-source, they may be replicated in other software environments and infrastructure. If an organization would like to set up their own, private KG-Hub, they may do so by setting up each graph project as a private GitHub repository based on the kg-cookiecutter template. They may then set up their own AWS S3 cloud storage as a data repository, or elect to use other cloud storage providers supporting static web page hosting, such as Google Cloud Storage or Microsoft's Azure Storage. Automated KG updating may be performed through a combination of GitHub Actions and an instance of the Jenkins automation server, as we have done, but may alternatively be accomplished through other continuous integration systems and data orchestration platforms such as Dagster or Apache Airflow.

4 Discussion

KG-Hub addresses many hurdles involved in assembling, sharing, and using KGs. While KGs provide an elegant method to integrate biological data, work involving KGs remains challenging for a variety of technical and sociological reasons. Analyses of KGs are difficult to reproduce without versioned builds of KGs (as provided by KG-Hub), and interpretation of these analyses are often confounded by lack of provenance and lack of access to upstream data. Integrating data across multiple KGs is challenging due to incompatibilities in format, schema, and data representation. KG-Hub helps to lower these barriers.

KG-Hub could be considered the analog of an OBO Foundry for KGs. Much of KG-Hub's philosophy echoes that of the OBO Foundry (Jackson *et al.* 2021), specifically the concept that data collections should follow a consistent format and be obtained from a persistent source. Beyond standardization of KGX format graph files and the Biolink data model, KG-Hub provides a blueprint and specific templates for data ingestion, transformation, and graph assembly. The unification of methods, formats, models, validation, and analysis strategies serves as an ecosystem for reproducible research that uses KGs.

Demonstrations of relevant use cases are provided in the KG-Hub tutorials (https://github.com/Knowledge-Graph-Hub/knowledge-graph-hub-support/tree/main/kg-hub-tutorials). These tutorials provide practice examples of how KG-Hub can be used, especially for downstream graph ML use cases such as node embedding and link prediction using GRAPE, and automated graph ML using NEAT.

KG-Hub is designed to be straightforward to use and comprehensible to both seasoned KG engineers and beginning researchers. Some understanding of graph theory, ontologies, and the idiosyncrasies of various graph representations (e.g. how relationships between entities are described within triples versus the property graph model that is used in KG-Hub) helps. For ontologies in particular, users must remain aware of the limitations of each set of axioms and terms and how they relate to a chosen domain. If chemicals are to be represented by ChEBI terms, e.g. all newly added chemical nodes should be linked to ChEBI identifiers. If the community identifies needs outside KG-Hub's current technical capabilities, we will explore further improvements. Our first point of contact for

technical requests is the KG-Hub Support code repository on GitHub (https://github.com/Knowledge-Graph-Hub/knowledge-graph-hub-support).

Future improvements to KG-Hub will largely depend on community needs and contributions. We will continue to include new KG projects and provide self-contained versions of other graph collections. We foresee creating further valuable assets for graph-based biological and biomedical data analysis. KG-Hub will continue to be an evolving resource for driving reproducible, standardized KG construction and reuse.

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest

None declared.

Data availability

Artifacts (KGs, provenance, statistics) from KG-Hub projects mentioned in this article can be obtained from the KG-Hub web site (https://kghub.io). A template project code repository is available (https://github.com/Knowledge-Graph-Hub/kg-cookiecutter). Source code for ongoing KG-Hub projects is available at the following GitHub code repositories: KG-COVID-19 (https://github.com/Knowledge-Graph-Hub/kg-covid-19), KG-Microbe (https://github.com/Knowledge-Graph-Hub/kg-microbe), KG-IDG (https://github.com/Knowledge-Graph-Hub/kg-idg), KG-OBO (https://github.com/Knowledge-Graph-Hub/kg-obo), Eco-KG (https://github.com/Knowledge-Graph-Hub/eco-kg), KG-Monarch (https://github.com/monarch-initiative/monarch-ingest), and KG-PHENIO (https://github.com/Knowledge-Graph-Hub/kg-phenio).

References

Auer S, Bizer C, Kobilarov G *et al.* DBpedia: a nucleus for a web of open data. In: *The Semantic Web*. Berlin Heidelberg: Springer, 2007, 722–35.

Badal VD, Wright D, Katsis Y *et al.* Challenges in the construction of knowledge bases for human microbiome–disease associations. *Microbiome* 2019;**7**:129.

Bennett TD, Moffitt RA, Hajagos JG *et al.* The national COVID cohort collaborative: clinical characterization and early severity prediction. medRxiv, 2021, preprint: not peer reviewed.

Callahan TJ, Tripodi IJ, Hunter LE *et al.* A framework for automated construction of heterogeneous Large-Scale biomedical knowledge graphs. bioRxiv, 2020, preprint: not peer reviewed.

Cappelletti L, Fontana T, Casiraghi E *et al.* GRAPE for fast and scalable graph processing and random-walk-based embedding *Nature Comp Sci* 2023;**3**:552–568 (2023). https://doi.org/10.1038/s43588-023-00465-8.

Chan LE, Casiraghi E, Laraway B *et al.*; N3C Consortium. Metformin is associated with reduced COVID-19 severity in patients with prediabetes. *Diabetes Res Clin Pract* 2022;**194**:110157.

Chaves-Fraga D, Endris KM, Iglesias E *et al.* What are the parameters that affect the construction of a knowledge graph? In: *On the Move to Meaningful Internet Systems: OTM 2019 Conferences, Rhodes, Greece, October 21–25, 2019*. Springer International Publishing, 2019, 695–713.

Choi E, Bahadori MT, Song L *et al.* GRAM: graph-based attention model for healthcare representation learning. *KDD* 2017;**2017**: 787–95.

Doğan T, Atas H, Joshi V *et al.* CROssBAR: comprehensive resource of biomedical relations with knowledge graph representations. *Nucleic Acids Res* 2021;**49**:e96.

Domingo-Fernández D, Baksi S, Schultz B *et al.* COVID-19 knowledge graph: a computable, multi-modal, cause-and-effect knowledge model of COVID-19 pathophysiology. *Bioinformatics* 2021;**37**: 1332–4.

Dooley DM, Griffiths EJ, Gosal GS *et al.* FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration. *NPJ Sci Food* 2018;**2**:23.

Francis N, Green A, Guagliardo P *et al.* Cypher: an evolving query language for property graphs. In: *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, 1433–45. New York: Association for Computing Machinery, 2018.

Glass K, Girvan M. Finding new order in biological functions from the network structure of gene annotations. *PLoS Comput Biol* 2015;**11**: e1004565.

Hastings J, Owen G, Dekker A *et al.* ChEBI in 2016: improved services and an expanding collection of metabolites. *Nucleic Acids Res* 2016; **44**:D1214–9.

Hu J, Lepore R, Dobson RJB *et al.* DGLinker: flexible knowledge-graph prediction of disease-gene associations. *Nucleic Acids Res* 2021;**49**: W153–61.

Issa S, Adekunle O, Hamdi F *et al.* Knowledge graph completeness: a systematic literature review. *IEEE Access* 2021;**9**:31322–39.

Jackson R, Matentzoglu N, Overton JA *et al.* OBO foundry in 2021: operationalizing open data principles to evaluate ontologies. *Database* 2021;**2021**:baab069.

Joachimiak MP, Hegde H, Duncan WD *et al.* KG-Microbe: a reference knowledge-graph and platform for harmonized microbial information. In: *Proceedings of the 12th International Conference on Biomedical Ontologies, Bolzano, Italy, 15-18 September 2021*. 2021.

Köhler S, Gargano M, Matentzoglu N *et al.* The human phenotype ontology in 2021. *Nucleic Acids Res* 2021;**49**:D1207–17.

Li MM, Huang K, Zitnik M *et al.* Graph representation learning in biomedicine and healthcare. *Nat Biomed Eng* 2022;**6**:1353–69.

Ma F, You Q, Xiao H *et al.* KAME: knowledge-based attention model for diagnosis prediction in healthcare. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, 743–52. New York: Association for Computing Machinery, 2018.

Miller GA, Beckwith R, Fellbaum C *et al.* Introduction to WordNet: an on-line lexical database. *Int J Lexicography* 1990;**3**:235–44.

Mungall CJ, McMurry JA, Köhler S *et al.* The monarch initiative: an integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic Acids Res* 2017;**45**:D712–22.

Nelson CA, Bove R, Butte AJ *et al.* Embedding electronic health records onto a knowledge network recognizes prodromal features of multiple sclerosis and predicts diagnosis. *J Am Med Inform Assoc* 2022; **29**:424–34.

Nelson CA, Butte AJ, Baranzini SE *et al.* Integrating biomedical research and electronic health records to create knowledge-based biologically meaningful machine-readable embeddings. *Nat Commun* 2019;**10**: 3045.

Nicholson DN, Greene CS. Constructing knowledge graphs and their biomedical applications. *Comput Struct Biotechnol J* 2020;**18**:1414–28.

Nickel M, Murphy K, Tresp V *et al.* A review of relational machine learning for knowledge graphs. Proceedings of the IEEE 104, no. 1 (2015): 11–33.

Pang X, Bou-Dargham MJ, Liu Y *et al.* Abstract 2247: accelerating cancer research using big data with BioKDE platform. *Cancer Res* 2018; **78**:2247.

Peng C, Dieck S, Schmid A *et al.* CADA: phenotype-driven gene prioritization based on a case-enriched knowledge graph. *NAR Genom Bioinform* 2021;**3**:lqab078.

Pillich RT, Chen J, Rynkov V *et al.* NDEx: a community resource for sharing and publishing of biological networks. In: Wu CH (ed.), *Protein Bioinformatics: From Protein Modifications and Networks to Proteomics*. New York: Springer, 2017, 271–301.

Poelen JH, Simons JD, Mungall CJ *et al.* Global biotic interactions: an open infrastructure to share and analyze species-interaction datasets. *Ecol Inform* 2014;**24**:148–59.

Reese JT, Unni D, Callahan TJ *et al.* KG-COVID-19: a framework to produce customized knowledge graphs for COVID-19 response. *Patterns (NY)* 2021;**2**:100155.

Reese JT, Coleman B, Chan L *et al.* NSAID use and clinical outcomes in COVID-19 patients: a 38-center retrospective cohort study. *Virol J* 2022;**19**:84.

Rotmensch M, Halpern Y, Tlimat A *et al.* Learning a health knowledge graph from electronic medical records. *Sci Rep* 2017;**7**:5994.

Rouillard AD, Gundersen GW, Fernandez NF *et al.* The harmonizome: a collection of processed datasets gathered to serve and mine knowledge about genes and proteins. *Database* 2016;**2016**:baw100.

Santos A, Colaço AR, Nielsen AB *et al.* A knowledge graph to interpret clinical proteomics data. *Nat Biotechnol* 2022;**40**:692–702.

Shang Y, Tian Y, Zhou M *et al.* EHR-Oriented knowledge graph system: toward efficient utilization of Non-Used information buried in routine clinical practice. *IEEE J Biomed Health Inform* 2021;**25**: 2463–75.

Shefchek KA, Harris NL, Gargano M *et al.* The monarch initiative in 2019: an integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic Acids Res* 2020;**48**: D704–15.

Singhal A. Introducing the knowledge graph: things, not strings. *Google*, May 16, 2012. https://blog.google/products/search/introducing-knowledge-graph-things-not/

Smith DP, Oechsle O, Rawling MJ *et al.* Expert-augmented computational drug repurposing identified baricitinib as a treatment for COVID-19. *Front Pharmacol* 2021;**12**:709856.

Sosa DN, Derry A, Guo M *et al.* A literature-based knowledge graph embedding method for identifying drug repurposing opportunities in rare diseases. *Pac Symp Biocomput* 2020;**25**:463–74.

Su C, Tong J, Zhu Y *et al.* Network embedding in biomedical data science. *Brief Bioinform* 2020;**21**:182–97.

Unni DR, Moxon SAT, Bada M *et al.*; Biomedical Data Translator Consortium. Biolink model: a universal schema for knowledge graphs in clinical, biomedical, and translational science. *Clin Transl Sci* 2022;**15**:1848–55.

Ursu O, Holmes J, Knockel J *et al.* DrugCentral: online drug compendium. *Nucleic Acids Res* 2017;**45**:D932–9.

Vasilevsky N, Essaida S, Matentzoglu N *et al.* Mondo disease ontology: harmonizing disease concepts across the world. In: Hastings J (ed.) *CEUR Workshop Proceedings, CEUR-WS.* informatik.uni-leipzig.de, Germany: Otto-von-Guericke University Magdeburg 2020.

Whetzel PL, Noy NF, Shah NH *et al.* BioPortal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Res* 2011;**39**:W541–5.

, Wood EC, Glen AK, Kvarfordt LG *et al.* RTX-KG2: a system for building a semantically standardized knowledge graph for translational biomedicine. *BMC Bioinformatics* 2022;**23**:400 https://doi.org/10.1186/s12859-022-04932-3.

Yamaguchi A, Shin J, Fujiwara T *et al.* Gene ranking based on paths from phenotypes to genes on knowledge graph. In: *Proceedings of the 10th International Joint Conference on Knowledge Graphs*, IJCKG '21, 131–134, New York: Association for Computing Machinery, 2022.

Zeng X, Song X, Ma T *et al.* Repurpose open data to discover therapeutics for COVID-19 using deep learning. *J Proteome Res* 2020;**19**:4624–36.

Zeng X, Tu X, Liu Y *et al.* Toward better drug discovery with knowledge graph. *Curr Opin Struct Biol* 2022;**72**:114–26.

Zhang R, Hristovski D, Schutte D *et al.* Drug repurposing for COVID-19 via knowledge graph completion. *J Biomed Inform* 2021;**115**:103696.