Dictionary of disease ontologies (DODO): a graph database to facilitate access and interaction with disease and phenotype ontologies

Liesbeth François[1], **Jonathan van Eyll**[2], **and Patrice Godard**[2]

[1]**Corresponding author. UCB Pharma, Braine-l'Alleud, 1420, Belgium. liesbeth.francois@ucb.com**
[2]**UCB Pharma, Braine-l'Alleud, 1420, Belgium**

Abstract The formal, hierarchical classification of diseases and phenotypes in ontologies facilitates the connection to various biomedical databases (drugs, drug targets, genetic variant, literature information, ...). Connecting these resources is complicated by the use of heterogeneous disease definitions, differences in granularity and structure. Despite ongoing efforts on integration, two challenges remain: (1) no resource provides a complete mapping across the multitude of disease ontologies and (2) there is no software available to comprehensively explore and interact with disease ontologies. In this paper, the DODO (Dictionary of Disease Ontology) database and R package are presented. DODO aims to deal with these two challenges by constructing a meta-database incorporating information of different publicly available disease ontologies. Thanks to the graph implementation, DODO allows the identification of indirect cross-references by allowing some relationships to be transitive. The R package provides several functions to build and interact with disease networks or convert identifiers between ontologies. They specifically aim to facilitate the integration of information from life science databases without the need to harmonize these upfront. The workflow for local adaptation and extension of the DODO database and a docker image with a DODO database instance are available.

Keywords

disease ontologies, diseases, phenotypes, database, identifiers

R version: R version 3.6.0 (2019-04-26)

Bioconductor version: 3.10

Package: 1.0.0

Introduction

Ontologies have been developed to structure, classify, and describe diseases [1, 2, 3]. As ontologies were independently created to support different biomedical databases dealing with genetics, treatments, and demographics, they differ in granularity and organization and define diseases in different ways [2, 4, 3, 5, 6, 7, 8]. While this stimulated the construction of integrated biological knowledgebases, the use of independent, ontology-specific identifiers, heterogeneous decisions on disease definitions, and the inherent presence of errors complicates integrating disease ontologies [6, 8]. In addition, the navigation of these large integrated knowledgebases with often an inherently complicated data model, is difficult for most, non-expert users [4, 9, 6].

Efforts have been made to establish new integrative disease ontologies [10, 11, 8]. Using semantic similarity, the Monarch Disease Ontology (MonDO) aggregates different sources including OMIM, Orphanet, NCiT, GARD, DO, and MF [10, 11]. The Disease Ontology (DO) resource aims to standardize disease descriptions and classifications from a clinical perspective using equivalence mappings [12, 13, 14]. Finally, the Experimental Factor Ontology (EFO) also establishes a unified ontology (not limited to diseases) by re-using several reference ontologies that lie within its scope. It subsequently enriches these classes with additional axioms when needed (Malone et al. 2010). Currently, it combines information from OMIM, Orphanet, ICD9/10 and SNOMEDCT, HPO, UBERON, and MonDO [15].

Despite these ongoing efforts, two bottlenecks hamper the connection of information around disease ontologies efficiently: (1) no resource provides a complete mapping across the multitude of ontologies and (2) there is no software available to comprehensively explore and interact with disease ontologies. The Dictionary of Disease Ontologies (DODO) was developed to address these two challenges. (1) While efforts such as MonDO and EFO try to integrate different disease ontologies through semantic learning and manual curation, these resources, like the different disease ontologies themselves, are currently not providing a complete mapping across all diseases ontologies [9, 8]. In addition, the existing efforts for integration are not flexible to extend easily to proprietary disease ontologies. DODO combines the information provided by the different ontologies and allows connecting ontologies to one another, even if they don't have direct cross-references. This is achieved by transitive mapping and carefully considering some indirect cross-reference relationships. (2) The second challenge addressed by DODO is the lack of an efficient and straightforward manner to access disease information through well-established bioinformatics platforms (such as R or python) [8, 16]. Such access would facilitate a more flexible connection to the different life science resources and create a more complete biomedical knowledge landscape. Currently, the programmatic access provided by many ontologies often requires expertise in creating SPARQL queries and a high level of understanding of the underlying databases or data model to be able to generate more complex queries [4, 9, 8]. The DODO R package allows easier access, exploration, and definition of disease concepts of interest. It can work as an intermediate layer to facilitate access and ensure exhaustive extraction of information from life science databases without the need to harmonize upfront. Here, the DODO graph database and R package are introduced, and we exemplify their added value by going through some use cases.

Methods

Implementation

In this section, an overview of the DODO database and the R package is presented.

Data model

The data model underlying DODO aims to capture the relationship between disease and phenotypes as described across different databases (Figure 1). *Disease* and *Phenotype* are two different kinds of *Concepts* sharing the same properties and they are related through a *has_pheno* relationship. *Concepts* are identified by name, the concatenation of the short identifier (*shortID*) and *database*. Additional properties of a *Concept* are (when available) a unique canonical *label*, disease *definition*, several *Synonyms*, and node *type*. Each *Concept* *is_in* one or multiple *Databases* with their URL encoded in the *idURL* property. Therefore, the *Database name* can differ from the *Concept* database as some ontologies re-use Concept names like EFO [7]. If the concepts are hierarchically organized, the *level* captures the highest level a disease has in the ontological tree (the most general term being level 0). This information is encoded as a property of a *Concept* where it captures the *level* of the term in the original ontology. In addition, as *Concepts* can be re-used, the ontology-specific *level* is also encoded as a property of the *is_in* relationship. Hierarchical information is also encoded by identifying

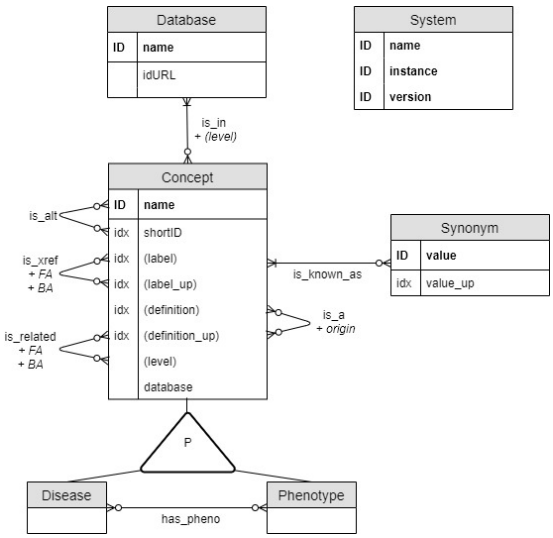

Figure 1. The DODO data model. It is shown as an Entity/Relationship (ER) diagram: entities (concept, disease, phenotype, database) correspond to graph nodes and relationships to graph edges. *ID* and *idx* refer to a unique or indexed entity respectively. Some properties are duplicated in upper case (_up) to improve the performance of case-insensitive searches. The system node is a technical node used to document the DODO instance."

Table 1. Different disease ontologies included into DODO database and link to GitHub repository.

Disease ontology	GitHub
Monarch Disease Ontology (MonDO)	https://github.com/Elysheba/Monarch
Experimental Factor Ontology (EFO)	https://github.com/Elysheba/EFO
Orphanet	https://github.com/Elysheba/Orphanet
MedGen	https://github.com/Elysheba/MedGen
Medical Subject Headings (MeSH)	https://github.com/Elysheba/MeSH
Human Phenotype Ontology (HPO)	https://github.com/patzaw/HPO
ClinVar	https://github.com/patzaw/ClinVar
Disease Ontology (DO)	https://github.com/Elysheba/DO
International Classification of Diseases (ICD11)	https://github.com/Elysheba/ICD11

a parent *Concept* through an *is_a* relationship. The property *origin* is assigned to this edge and corresponds to the ontology from which the relationship is derived (this can be useful when Concept names are re-used). Former and alternative identifiers of a *Concept* are documented using *is_alt* relationship.

Cross-references between *Concepts* are managed through two types of relationships depending of the confidence put on them (more detailed explanation below). The *is_xref* relationship is considered for more trusted relationships compared to *is_related* relationship. Each cross-reference (*is_xref* and *is_related*) between two *Concept* nodes is recorded as two edges, each in one direction. A *Concept* can have multiple cross-reference relationships to nodes of the same database. This ambiguity (one-to-many) is captured by the property *FA* (forward ambiguity) on a cross-reference edge and captures the number of cross-references to the same *Concept* database. The *BA* (backward ambiguity) property of the edge is defined symmetrically as the *FA* property of the edge going in the opposite direction. Both types of cross-reference edges and forward and backward ambiguities are used to define the relations used for transitivity mapping as explained below.

Feeding the database

A DODO instance is built on data from external resources that should be pre-processed to organize the information. This work was done for several public ontologies and the scripts can be found in the corresponding GitHub repositories (Table 1).

A R markdown document showing how to construct a new instance is provided alongside with a set of scripts to load and feed a Neo4j instance. These are not exported to avoid confusing the user when querying the database. The different steps to construct a new DODO Neo4j instance are briefly described below:

1. Structuring and harmonize the information derived from each ontology

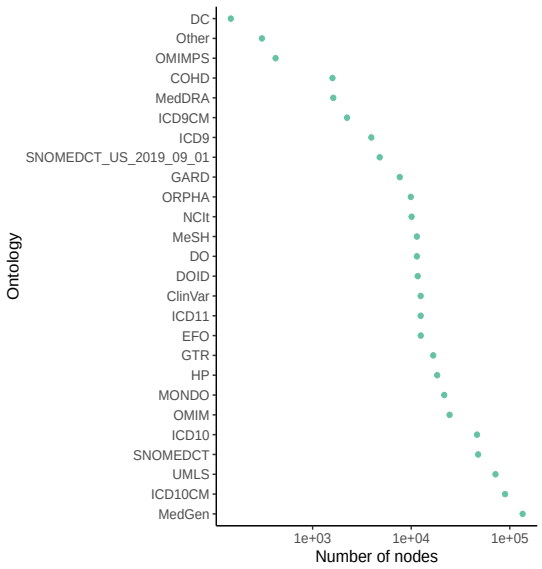

Figure 2. Overview of the number of nodes present for each disease ontology in DODO. 30 ontologies have less than 100 entries in DODO are summarized as 'other'.

Table 2. The number of edges for each relationship type present in DODO graph database

Relation	Number of edges
is_xref	280,691 (bidirectionally implemented)
is_related	225,021 (bidirectionally implemented)
has_pheno	363,589
is_a	221,360
is_alt	5,057

2. Combine the information obtained from the different ontologies and identify any duplicate or missing data

3. Start a new Neo4j instance and load data model

4. Information is imported into Neo4j by type. First, information around database nodes and concept nodes is imported. Next, information on the different relationships are loaded into the instance (cross-references, parent/child, alternative identifiers, phenotype mappings). For cross-reference identifiers, the type of edge is defined based on Supplementary Table 1 and subsequently imported into Neo4j. After import, the forward and backward ambiguity is calculated on each edge and assigned as property values to that edge.

Database instance availability

The DODO instance build using the workflow described above is provided as a Docker image [17]: `https://hub.docker.com/repository/docker/elysheba/public-dodo` (tag: 02.04.2020). This instance is build on information from the following disease ontologies listed in (Table 1).

DODO contains information on 54 different disease ontologies (Figure 2). There are 418,881 disease nodes and 18,354 phenotype nodes present in the database. 92,300 disease nodes have no recorded is_xref or is_related relationships across ontologies. The number of edges per relationship type is listed in Table 2.

In addition, a *setDisNet* S3 object is also available which build as a list of *disNet* objects. Figure @ref{fig:disNet} shows an example disNet object for epilepsy.

Operation

The database is implemented in Neo4j which uses the Cypher query language [18]. A DODO R package was developed to interact with and provide higher level functions to query the Neo4j graph database based on the data model described above [19].

The minimal system requirements are:

- R $\geq$ 3.6
- Operating system: Linux, macOS, Windows
- Memory $\geq$ 4GB RAM

The graph database has been implemented with Neo4j 3.5.14 [18] with the apoc.path.expand procedure 3.5.0.11. The DODO R package uses the following packages:

- *dplyr* [20]
- *tibble* [21]
- *neo2R* [22]
- *rlist* [23]
- *stringr* [24]
- *readr* [25]
- *visNetwork* [26]
- *shinythemes* [27]
- *DT* [28]
- *igraph* [29]
- *shiny* [30]

Querying the database

The DODO R package combines several functions to construct, interact, and explore the relationships between disease and phenotype identifiers. These can be divided in five different scopes (see Supplementary table 2):

- **Disease network functions:** these functions allow the user to build a disease network based on their relationships in the graph database: cross-reference, hierarchical information, and phenotypes. These functions include ways to extend, filter, split or combine disease networks. The information on diseases and phenotypes and their relationships is structured as a S3 disease network (disNet) object. It captures all information (disease node information, hierarchical information, phenotype information, alternative identifiers, and cross-reference information) around a disease.
- **Visualization and exploration:** these functions allow the exploration and visualization of relationships between disease and phenotype identifiers by querying them or providing a disease network.
- **Conversion:** this category of functions converts a list of disease and phenotype identifiers across ontologies or concepts based on the data model and provided parameters. Conversion can also include indirect relationships across ontologies using transitive mappings.
- **Connection and low level interactions:** these functions are technical helpers for establishing and managing connection with a DODO graph database. These functions also return information on the content of the current database and allow to directly send cypher queries to the database.
- **Data information:** these functions give access to content information such as the list of original databases, concept description and reference URL and allow ontology dump.

Transitivity mapping

Ambiguity of cross-reference relationships Disease identifiers within the different ontologies are annotated with cross-reference relationships to connect to independent biomedical databases. However, no ontology provides a complete mapping across all existing ontologies [9, 8]. Efforts to create an integrated ontology such as MonDO and EFO are continuously expanding to enrich their mappings but are currently not exhaustive and will lack mappings to proprietary ontologies. Here we propose the use of transitive mappings to extend the cross-reference information and connect ontologies (and biomedical resources) that lack direct relationship(s). This is exemplified in Figure 3, where transitivity mappings is needed to connect the initial MONDO identifier to a MeSH identifier using the indirect cross-reference relationship to the EFO and ORPHA node.

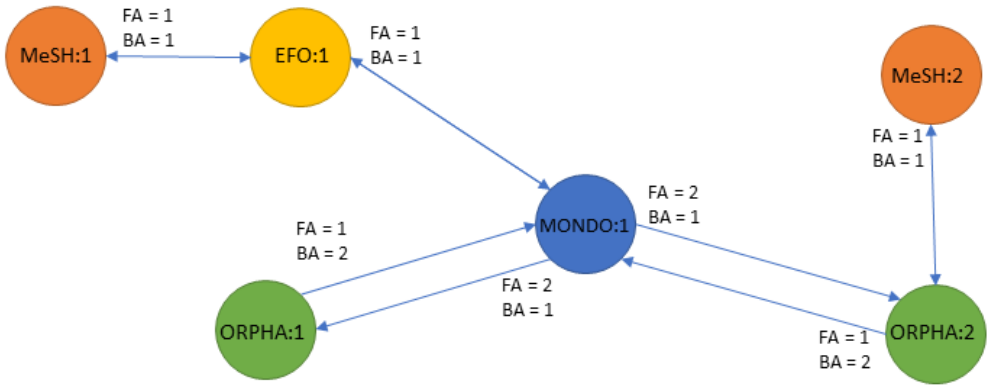

Figure 3. Example of transitivity mapping to infer an indirect relation with information on the ambiguity on the cross-reference edges.

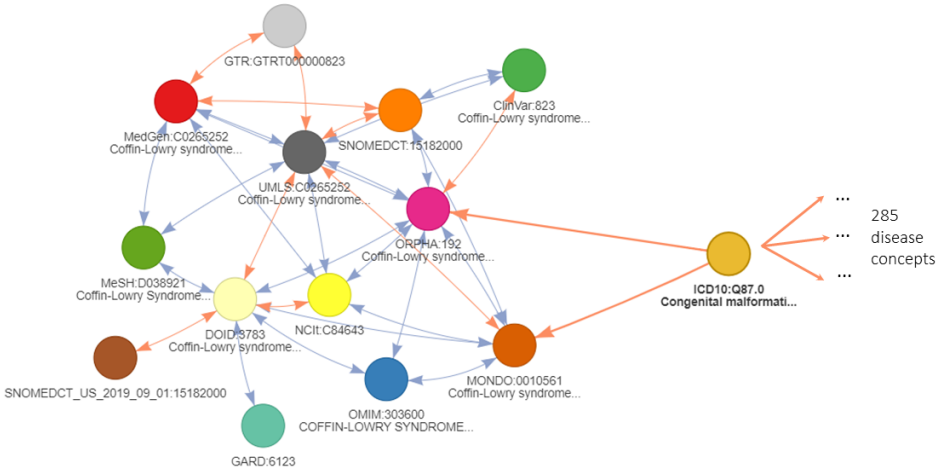

Figure 4. Subset of a disease network around Coffin-Lowry syndrome (ORPHA:192). The network shows equivalent cross-reference relations and ambiguous cross-reference relation to ICD10:Q87.0. This ICD10 disease identifier connects to 283 additional disease identifiers. The colors refer to different databases. *Is_xref* and *is_related* edges are indicated in blue and orange respectively. The arrows on the edge refer to the backward ambiguity: the identifier at the arrow destination has only one cross-reference in the database of the identifier at the arrow start. A double arrow indicates an unambiguous mapping between the 2 databases for the 2 identifiers in both directions. An edge should be considered transitive only in the direction of the arrow otherwise it can only be the final edge of a conversion path.

Most mappings are unambiguous (one concept in an ontology is related to only one concept in another ontology); however, some concepts map to many similar concepts within the same ontology. This is conceptually visualized in Figure 3 where the MONDO identifier maps to 2 ORPHA identifiers.

A real example is provided in Figure 4 with focus on the "Coffin-Lowry syndrome" (ORPHA: 192), a rare syndrome affecting brain and skeleton development. It relates to many disease identifiers, that often share similar disease definitions in an unambiguous or one-on-one manner. These relationships are valuable for transitive mapping as they extend the initial node to very similar nodes in other ontologies. However, the cross-reference relationship to ICD10 deals with a very broad term of "Congenital malformation syndromes predominantly affecting facial appearance" (ICD:Q87.0). This identifier is highly ambiguous as it has 285 additional direct cross-reference relationships to other disease identifiers. The use of this type of indirect relationships needs to be carefully considered. This ambiguity is a consequence of the inter-ontology differences in concept definitions and precision where some cross-reference edges connect identifiers that are not equivalent. Ontologies such as MonDO or EFO use a greater granularity in disease definitions than others like ICD10 or ICD9. If cross-reference edges are all considered equal without taking this distinction into account, it would be detrimental to the relevance of the conversion as it will return numerous more distantly related concepts when traversing these edges. Therefore, such relationships need to be avoided for transitive mappings.

To prevent these inaccurate conversions, we propose the use of backward ambiguity filtering. First, the forward ambiguity of a cross-reference edge needs to be quantified and it is the number of relationships where a node maps to several similar concepts within the same ontology. The value of the forward ambiguity is shown for the example in Figure 4 on each cross-reference edge between the nodes. A forward ambiguity greater than one indicates that the original concept is likely more general than the concepts it maps to. By following the relationship in the other direction, this ambiguity value is considered as the backward ambiguity. Similarly, a backward ambiguity greater than one, indicates that the original concept is probably more precise than the concepts it is mapped to (Figure 4. Applied to our example, the filtering on ambiguity will prevent traversing through the ambiguous ICD10 when using the transitivity mechanisms and return only cross-reference identifiers close to the original node.

Defining subtypes of cross-reference edges The filter on backward ambiguity improves greatly the accuracy of conversions. However, some inaccurate mappings may still be present due to higher general ambiguity between specific ontologies. The maximum total ambiguity of all relationships between two ontologies (the maximum value of the sum of forward and backward ambiguities of all cross-reference edges) quantifies the symmetry of their cross-reference relationships. The heatmap in Figure 5 shows this value in a log_{10} scale. While many ontologies are using concepts of similar level as identified by low maximum total ambiguity, a few can be identified that are more ambiguous in their mappings. Therefore, the general trust assigned to cross-reference relationships between ontologies is captured by defining two types of cross-reference edges: (1) the *is_xref* edge is used for equal cross-reference relationships where the concepts relate more directly to each other (similar concept definitions); (2) the *is_related* edge is used for all other cross-reference edges.

To assign relations to either type, a threshold is applied on the maximum total ambiguity between two ontologies. Different thresholds between 2 and 20 have been explored taking 14 different ontologies into account. For each step the cross-reference relationships between ontologies with the maximum total ambiguity falling below the threshold are defined as *is_xref* and otherwise as *is_related*. Next, all the identifiers from an ontology are converted to identifiers from any other ontology by applying transitive mappings only on *is_xref* relationships with a backward ambiguity of 1 (step = NULL, intransitive_ambiguity = 1). Since the number of *is_xref* relationships increases with the threshold applied, the number of conversions achieved from each identifier increases as well (see Figure 6 for an example focused on the MONDO ontology).

When choosing a conservative and low threshold, the conversions may be more precise but will hamper the ability of the transitive mappings to identify the most relevant mappings. It may therefore not return all cross-reference identifiers whereas a too high threshold impacts strongly the number of converted nodes at the cost of some inaccurate conversions. Especially the effect of on a limited set of identifiers is strong with the return of many, more distantly related and less precise cross-reference identifiers as can be seen by looking at the tail of the boxplot shown on Figure 6.

After having compared the results of the different ontologies, the threshold on the maximum total ambiguity value has been arbitrarily set to 4: cross-references between ontology with a maximum total ambiguity below 4 were considered as *is_xref* and others as *is_related* edges. However, two exceptions are ICD9/ICD10 and OMIM/Orphanet. Both ICD9 and ICD10 use higher level disease definitions, and therefore, these will never be connected through an *is_xref* edge except between themselves. OMIM and Orphanet on the other hand, use very narrow subtypes of diseases or mention specific variants. This results in a higher ambiguity as a general disease may have many relations to subtypes but as these ontologies use a higher granularity their relationships are still encoded as an *is_xref* edge. Supplementary table 2 shows the ontologies among which any cross-reference relationship will be encoded as *is_xref*.

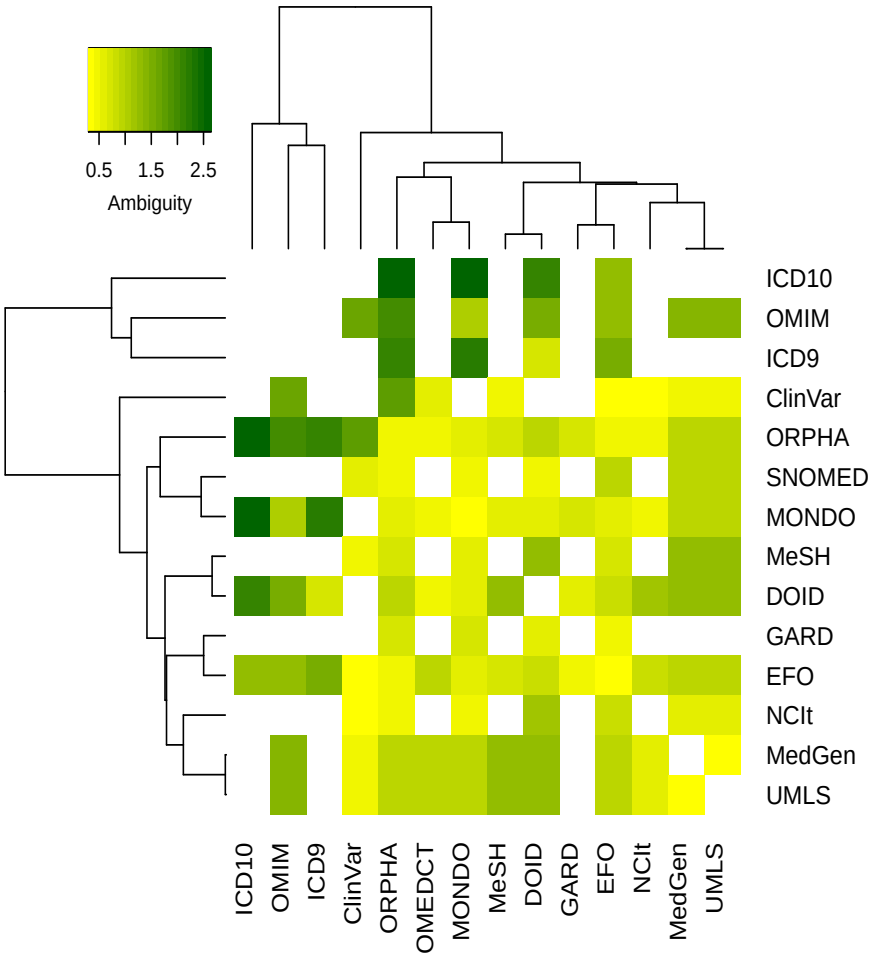

Figure 5. The heatmap shows the maximum value of total ambiguity between ontologies using a log_{10} transformation

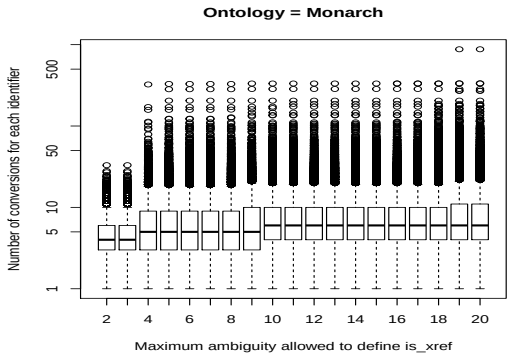

Figure 6. Number of conversions from each MONDO identifiers. This number depends on the cut-offs in the (total) ambiguity to define *is_xref* and *is_related* cross-reference edges on an ontology scale shown in x axis

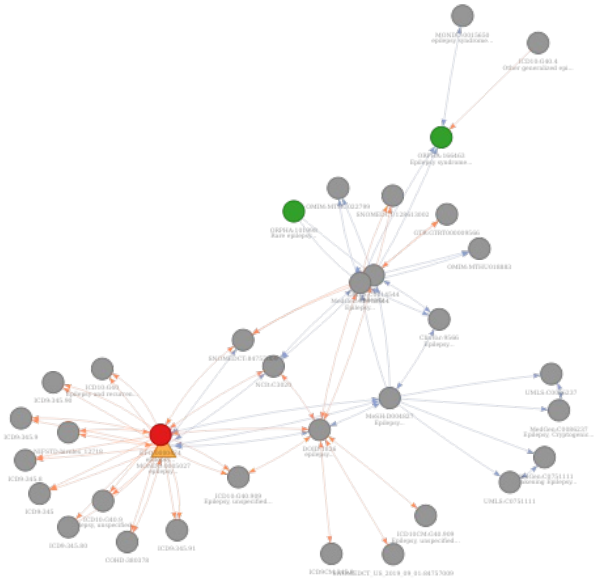

Figure 7. The network of identifiers around the MonDO identifier for 'Epilepsy' ('MONDO:0005027'). The seed node, MONDO:0005027, is depicted as a yellow triangle. Subsequently, the results of the different use cases are highlighted. The results of use case 1 (direct conversion) is indicated in red. The results of use case 2 (strict indirect conversion) and 3 (extended indirect conversion) to obtain corresponding Orphanet identifiers using transitivity are indicated in blue and green, respectively. *Is_xref* and *is_related* edges are indicated in blue and orange respectively. The arrows on the edge indicate the direction mapping can occur taking into account a backward ambiguity of one. Absence of arrow on an edge indicate these relations can only be exploited when no filtering is applied (generally final step of conversion).

Use cases

Conversion of concept identifiers

One of the basic functionalities of DODO is the ability to convert disease and phenotype identifiers between ontologies. The conversion of identifiers is generally performed using a two-step process based on the type of cross-reference edge to traverse and ambiguity values to filter on. The first step uses transitivity on is_xref edges only to identify the high confidence identifiers mapped to the initial identifier of interest. This makes the core of cross-referenced identifiers. It is strongly recommended to limit the backward ambiguity in this transitive mapping to one (transitive_ambiguity = 1 (default)) to avoid unwilled conversion to less accurate concepts. Once the core of identifiers is established, the second step expands it by a single step using both types of cross-reference edges (is_xref and is_related) with specification on the backward ambiguity filtering ("intransitive_ambiguity" parameter).

Six separate use cases can be identified for converting disease or phenotype identifiers to other ontologies or concepts:

- Direct conversion: only return the direct cross-references without any transitive mapping
- Strict indirect conversion: uses transitive mapping and applies a filter on the backward ambiguity of the last step to only return equivalent concepts. Can be used to convert between ontologies that use similar granularity to define concepts.
- Extended indirect conversion (default): uses transitive mapping without any filter on the backward ambiguity of the last step to return both equivalent and more broader terms. This way of conversion returns ambiguous relations, but these relations are not used for transitive mapping steps. In general, when the aim is to reach a broader concept related to the original identifiers but not move through it, it is recommended to put no filter on the backward ambiguity filtering of the final step.

- Loosened indirect conversion: a specific conversion procedure is created for ontologies that are less connected through is_xref edges such as ontologies like WHO's ICD10 or ICD9. These ontologies have highly ambiguous mappings and use very broad disease definitions. The absence of is_xref cross-reference relationships restrict the utility of the transitivity mechanism when applying the standard conversion. Therefore, we recommend an additional step to the standard conversion implemented in the get_related function.
- Conversion between concepts: convert between concepts types, i.e. from disease identifier to phenotype identifiers or vice versa.
- Return deprecated identifiers: return previous version (deprecated) identifiers

The use cases will be illustrated using the Mondo identifier for epilepsy (MONDO:0005027). The conversion between concept types (phenotype to disease or vice versa) is exemplified here for one identifier; however, the conversion procedure can take multiple identifiers at once as input. Finally, a comparison of the different conversion possibilities is performed using the entire MonDO ontology.

Use case 1: Direct conversion

A first basic use case is conversion of the identifier of interest to direct cross-references (without any transitive mapping - parameter "step = 1"). As an example, MonDO identifier for epilepsy (MONDO:0005027) is mapped to its corresponding EFO identifier using *convert_concept* function.

In this first use case, only the direct relations are used to return the corresponding EFO identifiers (highlighted in red on Figure 7. Using transitive mappings is also not required for this example, as there would be no additional EFO identifiers returned by moving through indirect relationships.

```
## Use case 1
conv <- convert_concept(from = "MONDO:0005027",
                        to = "EFO",
                        from.concept = "Disease",
                        to.concept = "Disease",
                        step = 1,
                        intransitive_ambiguity = 1)
conv
```

```
## # A tibble: 1 x 3
##   from          to          deprecated
##   <chr>         <chr>       <lgl>
## 1 MONDO:0005027 EFO:0000474 NA
```

Use case 2: Strict indirect conversion

A second use case deals with the conversion using equivalent indirect relations. The same seed MonDO identifier for epilepsy will be mapped to return the corresponding Orphanet ontology identifier. By specifying "step = NULL", transitive mappings will be used to convert the identifier(s). To return only equivalent concepts, thereby avoiding the mapping to less precise disease concepts, backward ambiguity filtering is applied on the final step of conversion ("intransitive_ambiguity = 1"). This can be a good practice when converting between ontologies that define concept with similar granularity. As there is no direct mapping provided by the resources between MonDO and Orphanet, transitive mapping provided by DODO needs to be applied. This conversion of the seed identifier MONDO:0005027 returns one corresponding Orphanet identifier (ORPHA:166463) as shown in Figure 7 (blue node).

```
## Use case 2
conv <- convert_concept(from = "MONDO:0005027",
                        to = "ORPHA",
                        from.concept = "Disease",
                        to.concept = "Disease",
                        step = NULL,
                        intransitive_ambiguity = 1)
conv
```

```
## # A tibble: 1 x 3
##   from          to           deprecated
##   <chr>         <chr>        <lgl>
## 1 MONDO:0005027 ORPHA:166463 NA
```

Use case 3: Extended indirect conversion (default)

In some instances, you may want to define a list of disease identifiers more largely related to a disease of interest. The third use case addresses this by extending the transitivity mapping with no filtering on the final step of conversion ("intransitive_ambiguity = NULL" and "step = NULL"). The conversion of the seed identifier MONDO:0005027 to Orphanet with this setup, returns an additional Orphanet identifier (ORPHA:101993) indicated in green on Figure 7. The initial transitive mapping steps apply filtering with ambiguity equal to one via the is_xref edges (blue). The final mapping step is intransitive and will therefore return all nodes related through either an *is_related* or *is_xref* edge with no filtering on backward ambiguity. It does return ambiguous relations at the last step, but these relations are not used for transitive mapping steps. This conversion can be used to get all identifiers around a disease concept both equivalent and more broader terms or when converting from a narrower concept to a broader concept. In general, when the aim is to reach a broader concept related to the original identifiers but not move through it, it is recommended to put no filter on the "intransitive_ambiguity".

```
## Use case 3
conv <- convert_concept(from = "MONDO:0005027",
                        to = "ORPHA",
                        from.concept = "Disease",
                        to.concept = "Disease",
                        step = NULL,
                        intransitive_ambiguity = NULL)
conv
```

```
## # A tibble: 2 x 3
##   from          to           deprecated
##   <chr>         <chr>        <lgl>
## 1 MONDO:0005027 ORPHA:101998 NA
## 2 MONDO:0005027 ORPHA:166463 NA
```

Use case 4: Loosened indirect conversion

The fourth use case deals with the specific conversion procedure that is recommended for the ontologies which are less connected through *is_xref* edges to the "core" ontologies in DODO (e.g. MonDO, EFO, MedGen, etc.) such as ontologies like WHO's ICD10 or ICD9. These ontologies have highly ambiguous mappings and use very broad disease definitions. When starting the mapping from such ambiguous identifiers, the convert_concept function will not return cross-reference relationships to most other databases. It is restricted by the rules defined by transitivity mechanism. Therefore, we recommend an additional step to the standard conversion implemented in the get_related function. It performs an additional expansion step through *is_related* and *is_xref* edges before the standard conversion procedure. The ambiguity on this additional step is the same for the final step in the standard conversion procedure (use case 2 – strict indirect conversion - modified by the intransitive_ambiguity parameter equal to NULL). To illustrate the difference, we will show the conversion of the general ICD10 identifier for Epilepsy (ICD10:G40.9) to corresponding identifiers in DO. When using the standard *convert_concept* function, it is not possible to convert the ICD10 identifier to any DO identifier. As ICD10 is only related to other nodes via *is_related* edges, transitive relationships cannot be used to identify cross-references using the standard conversion. Using the *get_related* function, recommended for ontologies such as ICD10, does convert ICD10:G40.9 to corresponding DO identifiers through transitive relations and returns DOID:1826 (Epilepsy).

```
## Use case 4
## convert_concept()
conversion <- convert_concept(from = "ICD10:G40.9",
                              to = "DOID",
                              from.concept = "Disease",
                              to.concept = "Disease")
conversion
```

```
## # A tibble: 0 x 3
## # ... with 3 variables: from <chr>, to <chr>, deprecated <lgl>
```

```
## get_related()
related <- get_related(from = "ICD10:G40.9",
                       to = "DOID",
                       from.concept = "Disease",
                       to.concept = "Disease")
related
```

Figure 8. The entire network of disease identifiers around the ICD10 identifier for 'Epilepsy' (ICD10:G40.9 – green triangle). *Is_xref* and *is_related* edges are indicated in blue and orange respectively. The arrows on the edge indicate the direction mapping can occur taking into account the BA = 1 rule. Absence of arrow on an edge indicate these relations can only be exploited when no filtering is applied. Using the *get_related* functionality adds an additional step of conversion before applying the standard conversion procedure. The nodes indicated in green are those that would be returned using the standard *convert_concept* function. The grey nodes are those reachable by *get_related* function that relaxes the initial step of transitivity mappings through *is_related* edges. The retrieval of corresponding DO identifiers for ICD10:G40.9 returns DOID:1826 (indicated in orange).

```
## # A tibble: 1 x 3
##   from        to         deprecated
##   <chr>       <chr>      <lgl>
## 1 ICD10:G40.9 DOID:1826 FALSE
```

The underlying reason for this behavior and difference between the two functions is depicted in Figure 8. As there are no relationships from the ICD10 seed node that meet the criteria defined for transitive mapping (*intranstivity_ambiguity* = 1 and cross-reference relationships of *is_xref* type), the seed node cannot be mapped to the DO node of interest using the *convert_concept* function. The *get_related* function relaxes these transitive mapping criteria in the first step and allow to reach the directly related nodes connected by either type of relationships and with no constraint on backward ambiguity (green round circles in Figure 8). After this initial step, transitivity rules are applied to map the direct cross-reference identifiers to the identifier in DO (orange node in Figure 8).

Currently, the default conversion procedure refers to the third use case (extended indirect conversion) when transitivity mappings are used to extend through cross-reference relationships and no filtering is applied to the final step of extension through the network (default parameters: step = NULL, intransitive_ambiguity = 1). This conversion can be used to get all identifiers around a disease concept and it is the approach to use when converting from a narrower concept to a broader concept. Additional filtering can then subsequently be put in place to allow adaptation for specific use cases. However, for the first step using transitivity mapping on *is_xref* edges it is strongly recommended to use the default filtering on ambiguity by limiting (backward) ambiguity to one.

Use case 5: conversion between concepts

Conversion can also be used to convert between concepts types, i.e. from disease identifier to phenotype identifiers or vice versa. This conversion is achieved by the same convert_concept function that will leverage

has_pheno relationships. Practically, it is handled in two phases. The first phase uses the transitivity mechanism as detailed above to connect disease identifiers to phenotype identifiers even when no direct *has_pheno* relation is available. This initial step takes the same options as listed above to convert identifiers within the same concept (this step can be avoided by using parameter "step = NA"). The second phase converts identifiers between concepts by returning phenotype or disease nodes related to the original identifiers (including the converted identifiers obtained in the first phase) with the possibility to return direct and/or indirect relations with the parameter "step".

```
## From disease to phenotype
toPhenotype <- convert_concept(from = "MONDO:0012391",
                               to = "HP",
                               from.concept = "Disease",
                               to.concept = "Phenotype")
toPhenotype <- toPhenotype %>%
  mutate(diseaseLabel = describe_concept(from)$label,
         phenotypeLabel = describe_concept(to)$label)
toPhenotype
```

```
## # A tibble: 18 x 5
##    from    to      deprecated diseaseLabel           phenotypeLabel
##    <chr>   <chr>   <lgl>      <chr>                  <chr>
##  1 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Cerebellar atrophy
##  2 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Cerebral atrophy
##  3 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Increased neuronal autoflu~
##  4 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Clumsiness
##  5 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Progressive visual loss
##  6 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Mental deterioration
##  7 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Irritability
##  8 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Generalized tonic-clonic s~
##  9 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ EEG abnormality
## 10 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Focal impaired awareness s~
## 11 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Intellectual disability
## 12 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Developmental regression
## 13 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Curvilinear intracellular ~
## 14 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Restlessness
## 15 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Psychosis
## 16 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Behavioral abnormality
## 17 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Slow progression
## 18 MONDO:~ HP:000~ FALSE      neuronal ceroid lipof~ Autosomal recessive inheri~
```

```
## From phenotype to disease
toDisease <- convert_concept(from = "HP:0002384",
                               from.concept = "Phenotype",
                               to.concept = "Disease")
toDisease <- toDisease %>%
  mutate(phenotypeLabel = describe_concept(from)$label,
         diseaseLabel = describe_concept(to)$label)
toDisease
```

```
## # A tibble: 72 x 5
##    from    to       deprecated phenotypeLabel       diseaseLabel
##    <chr>   <chr>    <lgl>      <chr>                <chr>
##  1 HP:000~ HP:0002~ FALSE      Focal impaired awar~ neuronal ceroid lipofuscino~
##  2 HP:000~ ClinVar~ FALSE      Focal impaired awar~ spinocerebellar ataxia type~
##  3 HP:000~ UMLS:C0~ FALSE      Focal impaired awar~ hyperekplexia-epilepsy synd~
##  4 HP:000~ MONDO:0~ FALSE      Focal impaired awar~ bilateral parasagittal pari~
##  5 HP:000~ MONDO:0~ FALSE      Focal impaired awar~ benign familial neonatal-in~
##  6 HP:000~ MONDO:0~ FALSE      Focal impaired awar~ unilateral polymicrogyria
##  7 HP:000~ MONDO:0~ FALSE      Focal impaired awar~ Spinocerebellar ataxia type~
##  8 HP:000~ MONDO:0~ FALSE      Focal impaired awar~ Progressive epilepsy - inte~
##  9 HP:000~ MONDO:0~ FALSE      Focal impaired awar~ Lafora disease
## 10 HP:000~ MONDO:0~ FALSE      Focal impaired awar~ isolated focal cortical dys~
## # ... with 62 more rows
```

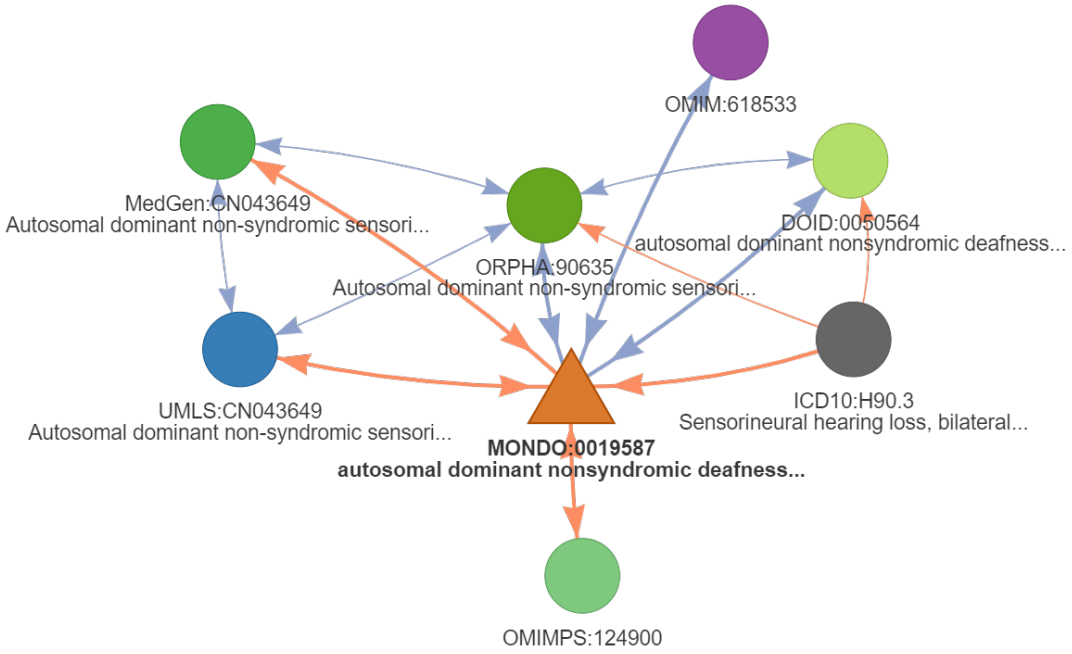

Figure 9. Direct cross-reference of MonDO identifier for 'autosomal dominant non-syndromic deafness' (MONDO:0019587)

Use case 6: return deprecated identifiers

Finally, conversion can also be used to return previous version (deprecated) identifiers when these are available.

```
deprecated <- convert_concept(from = "HP:0009638",
                              deprecated = TRUE,
                              from.concept = "Phenotype",
                              to.concept = "Phenotype")
deprecated
```

```
## # A tibble: 2 x 3
##   from       to         deprecated
##   <chr>      <chr>      <lgl>
## 1 HP:0009638 HP:0004079 TRUE
## 2 HP:0009638 HP:0006073 TRUE
```

Efficiency of conversion strategies

The different use cases were applied to only one identifier to show in more detail the conversion procedure. However, the conversion procedure is designed to take multiple identifiers as input. Here, the first three different approaches outlined in the previous section are applied to convert all 21,653 identifiers in the MonDO ontology to their corresponding identifiers in EFO, MesH, and DO. The mapping outcomes are summarized in Table 3.

```
mondo <- get_ontology("MONDO")

## option 1
conv1 <- convert_concept(from = mondo$nodes$id,
                         to = "EFO",
                         from.concept = "Disease",
                         to.concept = "Disease",
                         step = 1)
```

Table 3. Comparison of different conversion strategies using the MonDO ontology

	#MonDO identifiers[1]	#Target ontology identifiers[2]	Distribution of equivalent mapping[3]			#MonDO with ambiguous conversion[4]
			Median	Mean	Max	
EFO						
Direct conversion[5]	2.989	2.999	1	1006	2	19.000
Indirect strict conversion[6]	3.621	3.150	1	1145	9	378.000
Indirect extended conversion[7]	3.944	3.152	1	1158	9	444.000
MeSH						
Direct conversion[5]	7.798	7.835	1	1005	2	38.000
Indirect strict conversion[6]	9.574	9.207	1	1361	90	1.652
Indirect extended conversion[7]	10.318	9.315	1	1377	91	1.895
DO						
Direct conversion[5]	21.653	30.644	1	1416	4	8.933
Indirect strict conversion[6]	21.653	31.735	2	2070	294	11.314
Indirect extended conversion[7]	21.653	31.736	2	2133	294	12.269

[1] Number of unique MonDO identifiers with a conversion
[2] Number of unique converted identifiers in the targeted ontology
[3] Distribution of the number of conversions returned per MonDO identifier.
[4] Number of MonDO identifiers with ambiguous conversions
[5] Conversion as performed by use case 1: direct conversion (see above), the parameter is step = 1
[6] Conversion as performed by use case 2: strict indirect conversion (see above), the parameters are step = NULL and intransitive_ambiguity = 1
[7] Conversion as performed by use case 3: extended indirect conversion (see above), the parameters are step = NULL and intransitive_ambiguity = NULL)

```
summary(conv1 %>% count(from) %>% pull(n))

conv2 <- convert_concept(from = mondo$nodes$id,
                         to = "EFO",
                         from.concept = "Disease",
                         to.concept = "Disease",
                         step = NULL,
                         intransitive_ambiguity = 1)
summary(conv2 %>% count(from) %>% pull(n))


conv3 <- convert_concept(from = mondo$nodes$id,
                         to = "EFO",
                         from.concept = "Disease",
                         to.concept = "Disease",
                         step = NULL,
                         intransitive_ambiguity = NULL)
summary(conv3 %>% count(from) %>% pull(n))
```

Except for Disease Ontology (DO) (which is included by MonDO while constructing their ontology based on semantic similarity), the majority of MonDO identifiers cannot be converted to EFO or MeSH identifiers. However, compared to direct mapping of encoded relationships (use case 1 – direct conversion), the use of transitive mapping allows the conversion of 20% additional MonDO identifiers (use case 2 – strict indirect conversion). Enabling the extension to broader disease concepts (use case 3 – extended indirect conversion), still increases, by design and as expected, the number of mappings between two ontologies.

In parallel, the average ambiguity in mappings also increases with the different use cases (Table 3 - column 3). While it varies strongly across the different ontologies, the ambiguity is mostly minor for most identifiers (based on median and mean). Still, a limited set of mappings is strongly affected by ambiguity with as many as 294 corresponding DO identifiers and 91 corresponding MeSH identifiers for “autosomal dominant non-syndromic deafness” (MONDO:0019587). This identifier could not be mapped to EFO. Looking into this identifier in more detail shows that it only has six direct cross-references among which one DO identifier (DOID:0050564) (Figure 9). However, these direct cross-reference identifiers themselves report a multitude of cross-reference identifiers encoding various subtypes of the condition. As mentioned before, the mappings are derived directly from the original resources. The observed ambiguity after transitive mapping highlights disease areas that are heterogeneously defined across ontologies. This ambiguity is naturally more prevalent for those indications with a lot of reported subtypes reported in different ontology and/or the ontology using a higher level of granularity used to define diseases.

Build and explore a network of diseases

Building a disease network

While conversion facilitates connecting biomedical resources directly, another possibility provided by DODO is the exploration of diseases and their relationships as a disease network. Contrary to conversion, a network

retains all relationships as they are encoded in the DODO graph database. In this use case, we will show how to construct such a disease network and which functionalities are available to interact with a network.

```
disNet <- build_disNet(term = "amyotrophic lateral sclerosis",
                       fields = c("label", "synonym"))
disNet
```

```
## # A tibble: 251 x 7
##    id      label             definition           shortID level type    database
##    <chr>   <chr>             <chr>                <chr>   <int> <chr>   <chr>
##  1 ORPHA:~ Juvenile amyotro~ Juvenile amyotrophi~ 300605      6 Concep~ ORPHA
##  2 UMLS:C~ Amyotrophic late~ <NA>                 C18629~    NA Concep~ UMLS
##  3 ClinVa~ Amyotrophic late~ Amyotrophic lateral~ 16012      NA Concep~ ClinVar
##  4 OMIM:6~ AMYOTROPHIC LATE~ AMYOTROPHIC LATERAL~ 606640     NA Concep~ OMIM
##  5 MedGen~ Amyotrophic late~ Amyotrophic lateral~ C26754~    NA Concep~ MedGen
##  6 UMLS:C~ <NA>              <NA>                 C18629~    NA Concep~ UMLS
##  7 UMLS:C~ Amyotrophic late~ <NA>                 C18654~    NA Concep~ UMLS
##  8 MedGen~ Amyotrophic late~ Amyotrophic lateral~ C32805~    NA Concep~ MedGen
##  9 OMIM:6~ AMYOTROPHIC LATE~ AMYOTROPHIC LATERAL~ 613954     NA Concep~ OMIM
## 10 MedGen~ Amyotrophic late~ Amyotrophic lateral~ C31514~    NA Concep~ MedGen
## # ... with 241 more rows
##
## The disNet contains:
##  -  250 disease nodes from 11 ontologies and 1 phenotype nodes from 1 ontologies
##  -  1126 synonyms of the disease nodes
##  -  62 parent/child edges
##  -  470 crossreference edges
##  -  0 alternative edges
##  -  66 phenotype edges
##  -  The disNet was build based on 251 seeds
```

Extension through different relationships

After the construction of a disease network, it is likely that it doesn't contain the complete information on that particular disease of interest. The *extend_disNet* function enriches the *disNet* and extends it to cross-reference identifiers, child/parent terms, annotated phenotypes/disease, and/or alternative identifiers when available. In concordance with the conversion procedure, extension follows the same two-step approach using transitive mapping on is_xref edges followed by a final one-step extension on any cross-reference relationship taking filtering on backward ambiguity into account. To perform this extension the same parameters can be supplied with similar aims as for the conversion procedure (see above).

```
disNet <- build_disNet(term = "amyotrophic lateral sclerosis",
                       fields = c("label", "synonym"))
extendedDisNet <- extend_disNet(disNet,
                                relations = c("xref", "child"),
                                intransitive.ambiguity = 1)
extendedDisNet
```

```
## # A tibble: 494 x 7
##    id      label             definition            shortID level type   database
##    <chr>   <chr>             <chr>                 <chr>   <int> <chr>  <chr>
##  1 ORPHA:~ Juvenile amyotro~ Juvenile amyotrophic~ 300605      6 Conce~ ORPHA
##  2 UMLS:C~ Amyotrophic late~ <NA>                  C18629~    NA Conce~ UMLS
##  3 GTR:GT~ <NA>              <NA>                  GTRT00~    NA Conce~ GTR
##  4 MedGen~ Inclusion body m~ Inclusion body myopa~ C38094~    NA Conce~ MedGen
##  5 GARD:0~ <NA>              <NA>                  0010499    NA Conce~ GARD
##  6 ClinVa~ Amyotrophic late~ Amyotrophic lateral ~ 16012      NA Conce~ ClinVar
##  7 MeSH:C~ <NA>              <NA>                  C566550    NA Conce~ MeSH
##  8 OMIM:6~ AMYOTROPHIC LATE~ AMYOTROPHIC LATERAL ~ 606640     NA Conce~ OMIM
##  9 OMIM:1~ <NA>              <NA>                  164015     NA Conce~ OMIM
## 10 OMIM:6~ <NA>              AMYOTROPHIC LATERAL ~ 602572~    NA Conce~ OMIM
## # ... with 484 more rows
##
## The disNet contains:
```

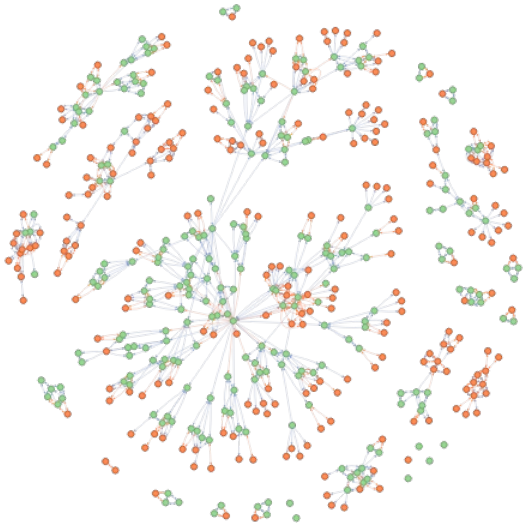

Figure 10. The disNet build on the term 'amyotrophic lateral sclerosis' querying both labels and synonyms provided in DODO (green nodes). This disNet is subsequently extended to return all cross-reference identifiers and child terms using the *extend_disNet* function (orange nodes) (parameters relations = c('xref', 'child') and intransitive_ambiguity = 1 to return only equivalent identifiers)

```
##  -  493 disease nodes from 24 ontologies and 1 phenotype nodes from 1 ontologies
##  -  1431 synonyms of the disease nodes
##  -  81 parent/child edges
##  -  888 crossreference edges
##  -  0 alternative edges
##  -  0 phenotype edges
##  -  The disNet was build based on 251 seeds
```

The disease network gathers 488 disease concepts across 25 ontologies; only 251 where identified directly matching the search term (Figure 10). The additional terms were obtained through extension of both cross-references and parent/child relationships. Of specific note is the extension to (or from) phenotype information. Within one extension all different parameters (xref, child, parent, alt, and disease/phenotype) can be supplied with the exception that it is not possible to extend to both disease and phenotype simultaneously. In contrast with the conversion procedure, it does not use the transitivity mechanisms but rather takes all the diseases within the network and returns any associated phenotypes that can be obtained through the has_pheno relationship.

```
disNet <- build_disNet(id = c("HP:0003394", "HP:0002180", "HP:0002878"))
disNet <- extend_disNet(disNet = disNet, relations = "disease")
disNet
```

```
## # A tibble: 397 x 7
##    id      label            definition            shortID level type    database
##    <chr>   <chr>            <chr>                 <chr>   <int> <chr>   <chr>
##  1 MONDO:~ glycogen storag~ "Phosphoglycerate ki~ 0010392    11 Concep~ MONDO
##  2 OMIM:6~ NEURODEGENERATI~ "NEURODEGENERATION W~ 610217     NA Concep~ OMIM
##  3 MONDO:~ adult-onset dis~ "Adult-onset distal ~ 0018006    10 Concep~ MONDO
##  4 MONDO:~ Charcot-Marie-T~ "Autosomal dominant ~ 0011675     9 Concep~ MONDO
##  5 OMIM:6~ MOTOR NEURON DI~ "MOTOR NEURON DISEAS~ 600333     NA Concep~ OMIM
##  6 MONDO:~ pure mitochondr~ "Pure mitochondrial ~ 0016807    11 Concep~ MONDO
##  7 ORPHA:~ Congenital musc~ "Congenital muscular~ 258         8 Concep~ ORPHA
##  8 OMIM:3~ MITOCHONDRIAL C~ "MITOCHONDRIAL COMPL~ 301021     NA Concep~ OMIM
##  9 ORPHA:~ Nocardiosis      "Nocardiosis is a lo~ 31204       4 Concep~ ORPHA
## 10 ORPHA:~ Mitochondrial e~ " mutation is charac~ 1194        9 Concep~ ORPHA
```

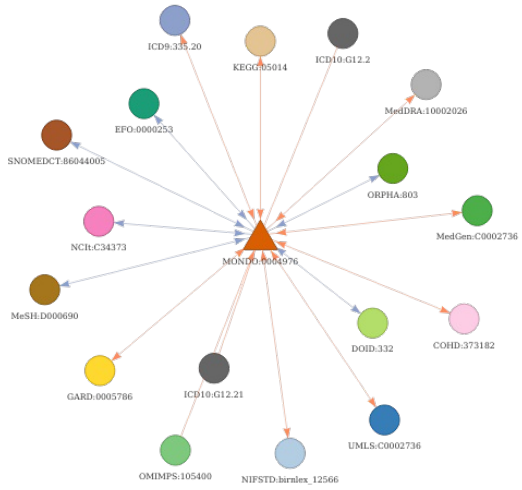

Figure 11. Plot of a small disease network constructed around 'Amyotrophic lateral sclerosis' (MONDO:0004976). The colors refer to different databases. *Is_xref* and *is_related* edges are indicated in blue and orange respectively. The arrows on the edge refer to the backward ambiguity and show how an edge can be traversed. When no arrow is present, it can only be reached through the final step of conversion when no filtering is present.

```
## # ... with 387 more rows
##
## The disNet contains:
##  -  394 disease nodes from 6 ontologies and 3 phenotype nodes from 1 ontologies
##  -  2206 synonyms of the disease nodes
##  -  0 parent/child edges
##  -  0 crossreference edges
##  -  0 alternative edges
##  -  398 phenotype edges
##  -  The disNet was build based on 3 seeds
```

Explore a network of diseases

DODO is built as a meta-database incorporating several disease ontologies and their listed relationships. As disease concepts and definitions are not a natural process but rather an artificial, human-biased effort, concepts might not always be clearly defined or related to each other in a straightforward manner. The different ontologies employ heterogeneous definitions, cross-reference axes are not always exact, and errors present in the original ontologies will impact DODO as well. The *explore_disNet* on a single *disNet* object returns a data table presenting information on the different identifiers present in the network. The plot function displays as a network how diseases are related to each other across the different ontologies (Figure 11).

It may be required to review the returned network of diseases after building and/or extending it to assess whether all nodes are relevant or of interest. This process can be simplified by considering clusters of cross-references (nodes dealing with similar concepts) using the *cluster_disNet* functionality.

```
disNet <- build_disNet(term = "amyotrophic lateral sclerosis",
                       fields = c("label", "synonym"))
clDisNet <- cluster_disNet(disNet = disNet,
                           clusterOn = "xref")
clDisNet
```

```
## The setDisNet contains 29 disNet clusters
```

Table 4. Annotation of the different cross-reference clusters of nodes identified for a disNet around 'amyotrophic lateral sclerosis'.

cluster	clusterSize	id	label
1	150	ICD10CM:G12.21	Amyotrophic lateral sclerosis
2	22	MONDO:0017161	frontotemporal dementia with motor neuron disease
3	2	MedGen:C1862940	Amyotrophic Lateral Sclerosis, Autosomal Recessive
4	9	ORPHA:357043	Amyotrophic lateral sclerosis type 4
5	3	MedGen:C3542025	Amyotrophic lateral sclerosis 1, autosomal recessive
6	8	MONDO:0005145	sporadic amyotrophic lateral sclerosis
7	6	MONDO:0014640	FTDALS3
8	5	MONDO:0008781	juvenile amyotrophic lateral sclerosis with dementia
9	6	MONDO:0011632	amyotrophic lateral sclerosis type 21
10	2	MedGen:C4302169	Amyotrophic lateral sclerosis plus syndrome
11	3	UMLS:C2750729	Amyotrophic lateral sclerosis 6, autosomal recessive
12	3	UMLS:CN239196	Amyotrophic Lateral Sclerosis, Recessive
13	2	MedGen:CN260033	Amyotrophic lateral sclerosis 10, with or without FTD
14	4	ORPHA:52430	Inclusion body myopathy with Paget disease of bone and frontotemporal dementia
15	3	UMLS:C2931441	Infantile-onset ascending hereditary spastic paralysis
16	2	MedGen:C2931786	Amyotrophic lateral sclerosis, type 6
17	3	ORPHA:98756	Spinocerebellar ataxia type 2
18	2	MedGen:C3662062	Restrictive lung disease due to amyotrophic lateral sclerosis
19	3	MedGen:CN239175	Amyotrophic Lateral Sclerosis, Dominant
20	2	MedGen:C4551993	Amyotrophic Lateral Sclerosis, Familial
21	3	UMLS:CN239211	Amyotrophic Lateral Sclerosis/Frontotemporal Dementia
22	1	ClinVar:10103	Amyotrophic lateral sclerosis, typical
23	1	ClinVar:10438	Amyotrophic lateral sclerosis, susceptibility to
24	1	ClinVar:10387	Amyotrophic lateral sclerosis 13
25	1	ClinVar:32676	Amyotrophic lateral sclerosis 22 with frontotemporal dementia
26	1	MONDO:0008178	inclusion body myopathy with Paget disease of bone and frontotemporal dementia type 1
27	1	ClinVar:10104	Amyotrophic lateral sclerosis-parkinsonism/dementia complex 1, susceptibility to
28	1	ClinVar:16925	Amyotrophic lateral sclerosis 14 without frontotemporal dementia
29	1	HP:0007354	Amyotrophic lateral sclerosis

Table 5. Using the disNet to connect to CHEMBL results identifies compounds available for different disease identifiers listed here.

Disease identifier	Number of compounds
EFO:0000253	73
EFO:0001356	1
MeSH:D000690	74
ORPHA:98756	2

```
explore_disNet(clDisNet)
```

Instead of reviewing each node, the different cross-reference clusters can be reviewed to identify those of interest while using the relationships between nodes to handle equivalent nodes simultaneously without the need to review them separately. A summary of the clusters can be visualized using the explore_disNet function and the output is shown in Table 4. For a list of disease networks created after clustering the *explore_disNet* functionality summarizes information on the different clusters, provides information on the size of each cluster and presents a tag identifier information. This tag identifier is identified as the node with the highest level in the ontology and a label available. If multiple identifiers have the same level, the tag one is picked on alphabetical order of the label. Summarizing disease networks using cluster of cross-reference edges also allows the revision of identifiers that have no label information attached.

Connecting to external resources

The aim of DODO is to facilitate the connection with external resources. As described above, DODO allows the creation of a structured disease network around diseases of interest and their relationships and information. The use of this disease network facilitates the connection and exploration of different biomedical knowledge resources as it also allows tracing their connections more easily.

It is exemplified below by connecting two different external resources: ClinVar and CHEMBL using "amyotrophic lateral sclerosis" (ALS) as an example. We start by building a disease network around the term "amyotrophic lateral sclerosis" and extending through both cross-reference and parent/child relationships as described in the previous section. For comparison, CHEMBL and ClinVar are queried directly using the same search term. Each of these resources uses a different ontology as reference. CHEMBL uses both EFO and MeSH to annotate compounds to indication. ClinVar uses a variety of ontologies to connect variants and diseases, such as SNOMEDCT, MedGen, Orphanet or OMIM.

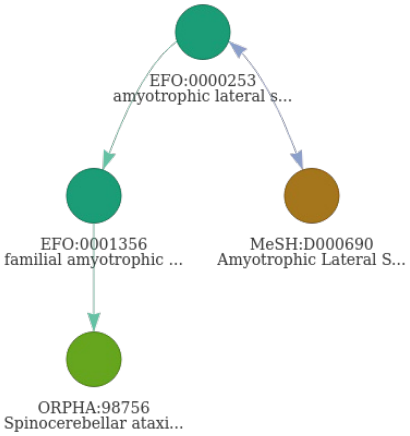

Figure 12. **The figure shows the relations between the different diseases with compounds available in CHEMBL resource. The term 'ORPHA:98756' was identified as a child term of 'EFO:0001356' through extension using *is_a* edge (green edges). Cross-reference edges are indicated in blue. The arrows on edges refer to the direction an edge can be traversed.**

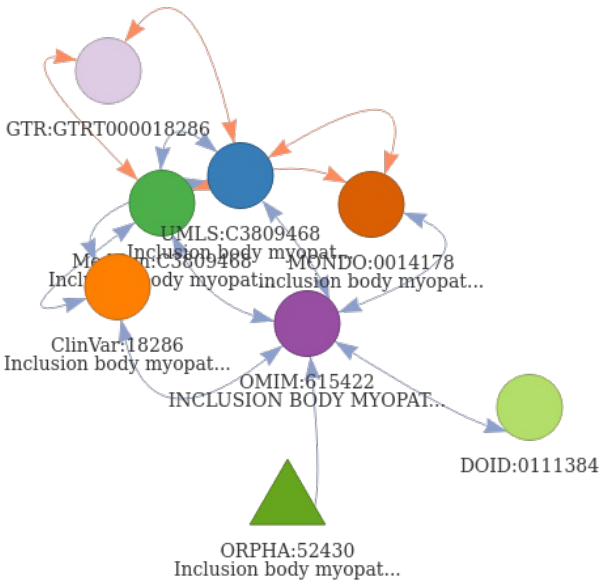

Figure 13. **The identifier 'ClinVar:18286' ('Inclusion body myopathy with early-onset Paget disease with or without frontotemporal dementia 2') connects to 'ORPHA:52430' through the use of transitivity on cross-reference edges (label: 'Inclusion body myopathy with Paget disease of bone and frontotemporal dementia' synonym: 'Pagetoid amyotrophic lateral sclerosis'). The color of the nodes refers to the ontology, the *is_xref* edges are indicated in blue and *is_related* edges indicated in orange. The arrows on the edge refer to the backward ambiguity and show how an edge can be traversed. When no arrow is present, it can only be reached through the final step of conversion when no filtering is present.**

Through the use a disease network, 96 unique compounds were identified in CHEMBL for ALS connected to four different disease identifiers (Table 5). The same set of compounds is identified by querying the resource directly, demonstrating the performance of DODO to properly map those different ontologies. One associated disease is missing in CHEMBL, namely the identifier "ORPHA:98756". This term was identified as a child term of 'EFO:0001356' with the extension of the *disNet* using DODO and providing more granularity (Figure 12). The disease can not be identified using a free-text query in CHEMBL directly as it's labelled 'Spinocerebellar ataxia type 2'. However, while different resources such as Monarch Initiative and EFO report ALS as a parent term, it is unclear whether this can be considered as ALS disease. OMIM does report a genetic overlap between spinocerebellar ataxia and amyotrophic lateral sclerosis type 13. The information integrated in the DODO graph database is based on the original information provided by ontologies such as EFO and MonDO without any additional curation. Errors in disease definitions and provided mappings will inherently be present in these ontologies and will therefore persist within DODO as well. While it was not within the initial scope of DODO, it may help to assess and identify underlying issues present in the ontologies.

For the ClinVar resource associating gene variants to diseases, all 105 unique Entrez gene variants returned querying ClinVar directly for "amyotrophic lateral sclerosis" are also identified using a network of diseases as a query start. However, an additional variant was reported for "Inclusion body myopathy with early-onset Paget disease with or without frontotemporal dementia 2" (ClinVar:18286). This identifier was found by extending through cross-references edges using transitivity mapping starting from "ORPHA:52430" ("Inclusion body myopathy with Paget disease of bone and frontotemporal dementia" and "Pagetoid amyotrophic lateral sclerosis" as a synonym (Figure 13)). This disease identifier is not an actual ALS disease but rather another neurodegenerative disorder related to frontotemporal dementia. This example highlights the necessity of reviewing the queried disease not only when using a network of diseases, but the same need remains when querying resources directly. Using DODO, it is possible to use underlying disease relationships to cluster diseases and identify groups of identifiers with similar or related concepts (Table 4) Particular clusters that are outside of the scope can be dropped and a more precise network of diseases returned to connect to external knowledgebases. The reviewed network of diseases no longer includes identifiers outside of the scope and identifies the same set of 105 unique Entrez gene identifiers compared to querying the ClinVar resource directly. For CHEMBL, the results remain the same. The ability to apply use and review easily disease networks should facilitate the integration of biomedical resources.

```
##########################@
## reviewing disNet
clDisNet <- cluster_disNet(disNet = extendedDisNet,
                           clusterOn = "xref")
explore_disNet(clDisNet)
clDisNet <- clDisNet[c(1:2, 4:6, 8:9, 13:20, 22, 24:28)]
fedisNet <- merge_disNet(list = clDisNet)
```

Tracing connections

Understanding the relation between disease identifiers obtained when querying a resource directly through a search term is not a trivial thing. The question remains whether these identifiers are dealing with the same disease concepts. An additional feature from connecting resources using a network of diseases, is the possibility to identify if and how diseases returned from each resource are connected to each other. This does not only allow a better understanding of disease, but also facilitates downstream analyses. Figure 14 shows the original ALS extended and reviewed network with disease identifiers matching in CHEMBL (orange) and those matching in ClinVar (blue). As both resources use different ontologies as references, there is a necessity to use cross-reference to understand their relationship to each other. Indirect relationships are used and recorded when extending and can facilitate the understanding and integration of different biological resources.

Conclusions

Disease ontologies have allowed a more formal classification of diseases. They facilitate the integration of biological databases thereby increasing disease information usage and supporting the development of novel treatments. However, efforts to integrate biological databases or the ontologies directly are complicated by ontology-specific identifiers, heterogeneous decisions on disease definitions, and the inherent presence of errors. Despite ongoing integration efforts, we identified two remaining challenges that prevent seamless integration of different databases based on disease ontologies:

- Currently no resource provides a flexible and complete mapping across the multitude of disease ontologies
- There is no software available to comprehensively explore and interact with disease ontologies

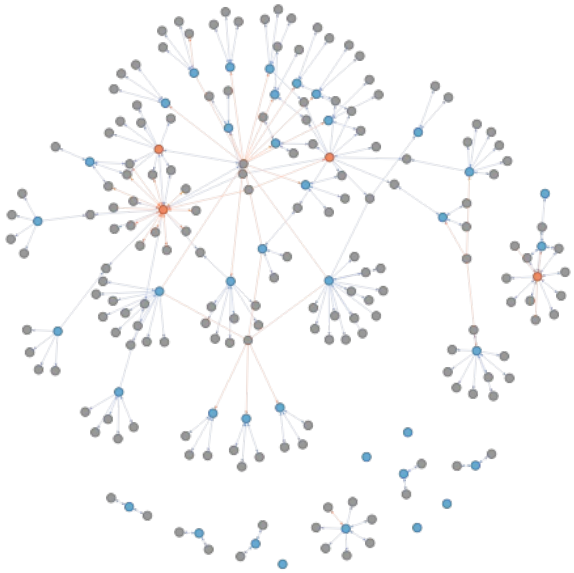

Figure 14. The figure shows the extended network of ALS constructed in DODO. The colour of the nodes refers to the disease identifiers that are also identified through a direct query in CHEMBL (orange) and ClinVar (blue). The edges between the nodes capture the *is_xref* and *is_related* relationship in blue and orange respectively.

DODO aims to tackle these two challenges by constructing a meta-database containing information on disease identifiers and their relationships across different ontologies. Through well-defined and controlled transitivity mechanisms, the combined information across resources can be used dynamically to identify indirect cross-references. The R package contains several functions to build and interact with disease networks or convert concept identifiers between ontologies. The workflow to construct a custom, local DODO database is provided with the intent to allow adaptation. A docker image with the presented ontologies is provided for convenience.

DODO helps clarifying and defining conditions of interest in addition to help in the understanding of relationships between disease concepts. It improves accessibility of disease ontologies for a standard user. In addition, connecting different biomedical knowledge resources through a disease network facilitates the integration of all this information. It also ensures these resources are queried transparently using equivalent identifiers of the disease of interest. In addition, it also allows visualizing the connection between these resources directly.

Through the aggregation of different ontologies and their mappings, DODO facilitates the generation of exhaustive descriptions of disease landscapes. The code to build and query DODO is provided under open source license to allow further improvement by other developers.

Software availability

The source code for parsing disease ontologies are available at:

- `https://github.com/Elysheba/Monarch`
- `https://github.com/Elysheba/EFO`
- `https://github.com/Elysheba/Orphanet`
- `https://github.com/Elysheba/MedGen`
- `https://github.com/Elysheba/MeSH`
- `https://github.com/patzaw/HPO`
- `https://github.com/patzaw/ClinVar`

Table 6. List of all functions available in DODO R package with description and scope details.

Function	Description	scope
build_disNet	Building a network of disease identifiers	Disease network functions
extend_disNet	Extending a disNet by different edges	Disease network functions
filter_by_id	Filtering a disNet by id	Disease network functions
filter_by_database	Filtering a disNet by database	Disease network functions
focus_disNet	Focus on identifiers of interest and its neighbors	Disease network functions
cluster_disNet	Clustering a disNet, generates a setDisNet	Disease network functions
setdiff_disNet	Substract one disNet from another	Disease network functions
split_disNet	Split a disNet based on a list of identifiers into a setDisNet	Disease network functions
explore_disNet	Visualizes a datatable to explore a disNet	Visualization and exploration
show_relations	Visualizes cross-reference relationships for the provided identifier	Visualization and exploration
plot.disNet	Visualizes a disNet using visNetwork	Visualization and exploration
convert_concept	Convert the provided set of identifiers to another ontology or between concepts	Conversion
get_related	Convert function for ontologies separated by *is_related* edges mainly	Conversion
check_dodo_connection	Check connection with DODO graphical database	Connection and low-level interactions
connect_to_dodo	Establish connection with DODO graphical database	Connection and low-level interactions
forget_dodo_connection	Forget a saved connection to DODO	Connection and low-level interactions
list_dodo_connections	List all saved connections to DODO	Connection and low-level interactions
call_dodo	Calls a function on the DODO graphical database	Connection and low-level interactions
show_dodo_model	Return DODO data model	Connection and low-level interactions
get_version	Return DODO database version	Connection and low-level interactions
get_concept_url	Returns concept url	Connection and low-level interactions
list_database	Lists databases in DODO	Connection and low-level interactions
list_node_type	Lists node type in DODO	Data information
get_ontology	Returns whole ontology	Data information
describe_concept	Returns concept description	Data information

- `https://github.com/Elysheba/DO`
- `https://github.com/Elysheba/ICD11`

The source code for DODO is available at: `https://github.com/Elysheba/DODO`

A docker image of the DODO Neo4j instance is available at: `https://hub.docker.com/repository/docker/elysheba/dodo` (`tag:02.04.2020`) Software is available to use under a GPL-3 license.

Competing interests

L.F., J.v.E., and P.G. are employees of UCB Pharma. J.v.E. and P.G. own stocks and/or shares from UCB Pharma. The authors declare no other competing interests.

Grant information

This work was entirely supported by UCB Pharma. The authors declared that no grants were involved in supporting this work.

Supplementary tables

References

[1] Thomas R Gruber. A Translation Approach to Portable Ontology Specifications. *Knowl. Aquis.*, 5(2):199–220, 1993.

[2] Melissa A Haendel, Julie A Mcmurry, Rose Relevo, Christopher J Mungall, N Peter, and Christopher G Chute. A Census of Disease Ontologies. *Annu. Rev. Biomed. Data Sci.*, 1:305–331, 2018.

[3] Robert Hoehndorf, Michel Dumontier, and Georgios V. Gkoutos. Evaluation of research in biomedical ontologies. *Brief. Bioinform.*, 14(6):696–712, 2013.

[4] Ali Hasnain, Maulik R. Kamdar, Panagiotis Hasapis, Dimitris Zeginis, Claude N. Warren, Helena F. Deus, Dimitrios Ntalaperas, Konstantinos Tarabanis, Muntazir Mehdi, and Stefan Decker. Linked biomedical dataspace: Lessons learned integrating data for drug discovery. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 8796:114–130, 2014.

[5] Warren A. Kibbe, Cesar Arze, Victor Felix, Elvira Mitraka, Evan Bolton, Gang Fu, Christopher J. Mungall, Janos X. Binder, James Malone, Drashtti Vasant, Helen Parkinson, and Lynn M. Schriml. Disease Ontology 2015 update: An expanded and updated database of Human diseases for linking biomedical knowledge through disease data. *Nucleic Acids Res.*, 43(D1):D1071–D1078, 2015.

[6] Kevin M Livingston, Michael Bada, William A. Baumgartner, and Lawrence E Hunter. KaBOB: ontology-based semantic integration of biomedical databases. *BMC Bioinformatics*, 16(1):1–21, 2015.

Table 7. List of ontologies among which the cross-reference relations are encoded as is_xref

DB1	DB2
EFO	ClinVar
MedGen	ClinVar
OPRHA	ClinVar
ORPHA	Cortellis_condition
Cortellis_condition	Cortellis_indication
ORPHA	Cortellis_indication
ORPHA	DOID
ORPHA	EFO
DOID	GARD
EFO	GARD
ORPHA	GARD
ICD10	ICD9
Cortellis_indication	MEDDRA
ORPHA	MedGen
ClinVar	MeSH
Cortellis_indication	MeSH
MONDO	MeSH
ORPHA	MeSH
Cortellis_condition	MeSH
EFO	MeSH
DOID	MeSH
OMIM	MeSH
MedGen	MeSH
UMLS	MeSH
Cortellis_condition	MetaBase_disease
Cortellis_indication	MetaBase_disease
DOID	MONDO
EFO	MONDO
ORPHA	MONDO
ClinVar	NCIt
MedGen	NCIt
MONDO	NCIt
ORPHA	NCIt
UMLS	NCIt
ClinVar	OMIM
DOID	OMIM
EFO	OMIM
MedGen	OMIM
MONDO	OMIM
ORPHA	OMIM
UMLS	OMIM
ClinVar	SNOMEDCT
DOID	SNOMEDCT
MONDO	SNOMEDCT
ORPHA	SNOMEDCT
ClinVar	UMLS
MedGen	UMLS
ORPHA	UMLS

[7] James Malone, Ele Holloway, Tomasz Adamusiak, Misha Kapushesky, Jie Zheng, Nikolay Kolesnikov, Anna Zhukova, Alvis Brazma, and Helen Parkinson. Modeling sample variables with an Experimental Factor Ontology. *Bioinformatics*, 26(8):1112–1118, 2010.

[8] Noa Rappaport, Noam Nativ, Gil Stelzer, Michal Twik, Yaron Guan-Golan, Tsippi Iny Stein, Iris Bahir, Frida Belinky, C. Paul Morrey, Marilyn Safran, and Doron Lancet. MalaCards: An integrated compendium for diseases and their annotation. *Database*, 2013:1–14, 2013.

[9] Wei Hu, Honglei Qiu, Jiacheng Huang, and Michel Dumontier. BioSearch: a semantic search engine for Bio2RDF. *Database (Oxford).*, 2017:1–13, 2017.

[10] Christopher J. Mungall, Julie A. McMurry, Sebastian Kohler, James P. Balhoff, Charles Borromeo, Matthew Brush, Seth Carbon, Tom Conlin, Nathan Dunn, Mark Engelstad, Erin Foster, J. P. Gourdine, Julius O.B. Jacobsen, Dan Keith, Bryan Laraway, Suzanna E. Lewis, Jeremy Nguyen Xuan, Kent Shefchek, Nicole Vasilevsky, Zhou Yuan, Nicole Washington, Harry Hochheiser, Tudor Groza, Damian Smedley, Peter N. Robinson, and Melissa A. Haendel. The Monarch Initiative: An integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic Acids Res.*, 45 (D1):D712–D722, 2017.

[11] Kent A Shefchek, Nomi L Harris, Michael Gargano, Nicolas Matentzoglu, Deepak Unni, Matthew Brush, Daniel Keith, Tom Conlin, Nicole Vasilevsky, Aaron Zhang, James P Balhoff, Larry Babb, Susan M Bello, Hannah Blau, Yvonne Bradford, Seth Carbon, Leigh Carmody, Lauren E Chan, Valentina Cipriani, Alayne Cuzick, Maria D Rocca, Nathan Dunn, Shahim Essaid, Petra Fey, Chris Grove, Jean-phillipe Gourdine, Ada Hamosh, Midori Harris, Ingo Helbig, Maureen Hoatlin, Marcin Joachimiak, Simon Jupp, M Pendlington, Clare Pilgrim, B Lett, Suzanna E Lewis, Craig Mcnamara, Tim Putman, Vida Ravanmehr, Justin Reese, Erin Riggs, Sofia Robb, Paola Roncaglia, James Seager, Erik Segerdell, Morgan Similuk, Andrea L Storm, Courtney Thaxon, Anne Thessen, Julius O B Jacobsen, Julie A Mcmurry, Tudor Groza, K Sebastian, Damian Smedley, Peter N Robinson, J Mungall, Melissa A Haendel, Monica C Munoz-torres, and David Osumi-sutherland. The Monarch Initiative in 2019 : an integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic Acids Res.*, 1:1–12, 2019.

[12] Liang Cheng, Guohua Wang, Jie Li, Tianjiao Zhang, Peigang Xu, and Yadong Wang. SIDD: A Semantically Integrated Database towards a Global View of Human Disease. *PLoS One*, 8(10):1–9, 2013.

[13] Lynn M Schriml and Elvira Mitraka. The Disease Ontology : fostering interoperability between biological and clinical human disease-related data. *Mamm. Genome*, 26(9):584–589, 2015.

[14] Guangchuang Yu, Li Gen Wang, Guang Rong Yan, and Qing Yu He. DOSE: An R/Bioconductor package for disease ontology semantic and enrichment analysis. *Bioinformatics*, 31(4):608–609, 2015.

[15] David Ochoa. EFO3: A community-driven ontology to advance clinical discoveries, 2019. URL `https://blog.opentargets.org/2019/12/19/efo3-a-community-driven-ontology-to-advance-clinical-discoveries/`.

[16] Mansoor Saqi, Artem Lysenko, Yi-ke Guo, Tatsuhiko Tsunoda, and Charles Auffray. Navigating the disease landscape: knowledge representations for contextualizing molecular signatures. *Brief. Bioinform.*, (November 2017):1–15, 2018. ISSN 1467-.

[17] Docker Inc. Docker Community Edition, 2019.

[18] Neo4J Inc. Neo4j Community Edition, 2020.

[19] R Core Team. R: A Language and Environment for Statistical Computing, 2019. URL `https://www.r-project.org/`.

[20] Hadly Wickham, Romain François, Lionel Henry, and Kirill Müller. dplyr: a grammar of data manipulation, 2019. URL `https://cran.r-project.org/package=dplyr`.

[21] Kirill Müller and Hadly Wickham. tibble: simple data frames, 2019. URL `https://cran.r-project.org/package=tibble`.

[22] Patrice Godard and Jonathan van Eyll. BED: A Biological Entity Dictionary based on a graph data model [version 2; referees: 2 approved]. *F1000Research*, 7:1–32, 2018.

[23] Kun Ren. rlist: a toolbox from non-tabular data manipulation, 2016. URL `https://cran.r-project.org/package=rlist`.

[24] Hadly Wickham. stringr: simple, consistent wrappers for common string operations, 2019. URL `https://cran.r-project.org/package=stringr`.

[25] Hadly Wickham, Jim Hester, and Romain François. readr: Read Rectangular Text Data, 2018.

[26] Almende B.V., Benoit Thieurmel, and Titouan Robert. visNetwork: network visualization using vis.js library, 2019. URL `https://cran.r-project.org/package=visNetwork`.

[27] Winston Chang. shinythemes: themes for shiny, 2018. URL `https://cran.r-project.org/package=shinythemes`.

[28] Yihui Xie, Joe Cheng, and Xianying Tan. DT: a wrapper for the JavaScript Library "DataTables", 2019. URL `https://cran.r-project.org/package=DT`.

[29] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Compex Sys:1695, 2006. URL `http://igraph.org`.

[30] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. shiny: Web Application Framework for R, 2019. URL `https://cran.r-project.org/package=shiny`.