How To Use DO.db

Jiang Li

Harbin Medical University, Harbin, China

October 20, 2011

Contents

1 Overview

This vignette demonstrates how to easily use the DO.db package. The Disease Ontology (DO) is a manually inspected subset of Unified Medical Language System (UMLS) and includes concepts from outside the UMLS disease/disorder semantic network including various cancers, congenital abnormalities, deformities and metal disorders. Terms in DO are organized in Directed Acyclic Graph (DAG). DO.db is used to represent the content of Disease Ontology (for example, the child-parent relationship in DO) through Bimap object.

To start with DO.db package, type following code below:

```
> library(DO.db)
> help(DO.db)
```

2 Fetch whole DO terms

In DO.db, *DOTermsAnnDbBimap* object(a sub-class of Bimap) *DOTERM* represet the whole DO terms. Meanwhile a class named *DOTerms* represents Disease Ontology nodes with four slots. Four S4 methods *DOID,Term,Synonym* and *Secondary* are defined in DO.db to fetch the corresponding attribute for a DO term node, they can work on *DOTermsAnnDbBimap,DOTerms* and *character* classes.

Here is an example of how to use these objects and functions, it fetchs the first 10 records and turn them to list object, then use the four S4 methods to fetch the corresponding attribute. Here is the code and result:

```
> FirstTenDOBimap <- DOTERM[1:10] ##grab thet ten
> class(FirstTenDOBimap)

[1] "DOTermsAnnDbBimap"
attr(,"package")
[1] "DO.db"

> xx <- as.list(FirstTenDOBimap)
> DOID(xx[[2]])

[1] "DOID:0000405"

>         Term(xx[[2]])

[1] "vascular tissue disease"

>         Synonym(xx[[2]])

character(0)

>         Secondary(xx[[2]])

character(0)
```

3 Fetch the relationship between DO terms

In this section, we will introduce four Bimap objects which represent relationship between DO terms. There are *DOANCESTOR,DOPARENTS,DOOFFSPRING* and *DOCHILDREN*. we will introduce them in the following sub-sections.

3.1 DOANCESTOR

This data set describes associations between DO terms and their ancestor terms, based on the directed acyclic graph (DAG) defined by the Disease Ontology Consortium. The format is an R object mapping the DO terms to all ancestor terms, where an ancestor term is a more general DO term that precedes the given DO term in the DAG (in other words, the parents, and all their parents, etc.).

For example, to fetch all the *DOANCESTOR* in a list object and display the first 4 DOID's ancestors, here is the code:

```
> xx <- as.list(DOANCESTOR)
> # Remove DO IDs that do not have any ancestor
> xx <- xx[!is.na(xx)]
> # Display the first 4 DOID's ancestor
> xx[1:4]

$`DOID:0000000`
[1] "DOID:4"  "DOID:77" "DOID:7"

$`DOID:0000405`
[1] "DOID:4"    "DOID:7"    "DOID:1287" "DOID:178"

$`DOID:0001816`
[1] "DOID:4"       "DOID:0050687" "DOID:14566"   "DOID:1115"    "DOID:162"

$`DOID:0002116`
[1] "DOID:863"     "DOID:4"       "DOID:1242"    "DOID:10124"   "DOID:5614"
[6] "DOID:7"       "DOID:0050155" "DOID:1492"
```

3.2 DOPARENTS

This data set describes associations between DO terms and their direct parent terms, based on the directed acyclic graph (DAG) defined by the Disease Ontology Consortium. The format is an R object mapping the DO terms to all direct parent terms, where a direct parent term is a more general DO term that immediately precedes the given DO term in the DAG.

For example, to fetch all the *DOPARENTS* in a list object and display the first 4 DOID's parents, here is the code:

```
> xx <- as.list(DOPARENTS)
> # Remove DO IDs that do not have any ancestor
> xx <- xx[!is.na(xx)]
> # Display the first 4 DOID's ancestor
> xx[1:4]
```

```
$`DOID:0000000`
[1] "DOID:77"

$`DOID:0000405`
[1] "DOID:178"

$`DOID:0001816`
[1] "DOID:1115"

$`DOID:0002116`
[1] "DOID:10124"
```

3.3 DOOFFSPRING

This data set describes associations between DO terms and their offspring terms, based on the directed acyclic graph (DAG) defined by the Disease Ontology Consortium. The format is an R object mapping the DO terms to all offspring terms, where an ancestor term is a more specific DO term that is preceded by the given DO term in the DAG (in other words, the children and all their children, etc.).

For example, to fetch all the *DOOFFSPRING* in a list object and display the first DOID's offsprings, here is the code:

```
> xx <- as.list(DOOFFSPRING)
> # Remove DO IDs that do not have any ancestor
> xx <- xx[!is.na(xx)]
> # Display the first 4 DOID's ancestor
> xx[1]

$`DOID:0000000`
 [1] "DOID:4140"  "DOID:1949"  "DOID:10211" "DOID:9717"  "DOID:11151"
 [6] "DOID:9765"  "DOID:2828"  "DOID:9714"  "DOID:11755" "DOID:10254"
[11] "DOID:9766"
```

3.4 DOCHILDREN

This data set describes associations between DO terms and their direct children terms, based on the directed acyclic graph (DAG) defined by the Disease Ontology Consortium. The format is an R object mapping the DO terms to all direct children terms, where a direct child term is a more specific DO term that is immediately preceded by the given DO term in the DAG.

For example, to fetch all the *DOCHILDREN* in a list object and display the first DOID's children, here is the code:

```
> xx <- as.list(DOCHILDREN)
> # Remove DO IDs that do not have any ancestor
> xx <- xx[!is.na(xx)]
> # Display the first 4 DOID's ancestor
> xx[1]

$`DOID:0000000`
[1] "DOID:10211" "DOID:10254" "DOID:11151" "DOID:11755" "DOID:1949"
[6] "DOID:4140"  "DOID:9714"  "DOID:9717"
```

4 Others

In this section, we will introduce the schema of DO.db and number of mapped keys for the maps in DO.db.

To get the schema of DO.db, type following:

```
> DO_dbschema()

  --
  -- DO_DB schema
  -- ====================
  --
  CREATE TABLE do_term (
  _id INTEGER PRIMARY KEY,
  do_id VARCHAR(12) NOT NULL UNIQUE, -- DI ID
  term VARCHAR(255) NOT NULL -- textual label for the DO term
  );

  CREATE TABLE do_synonym (
  _id INTEGER NOT NULL, -- REFERENCES do_term
  synonym VARCHAR(255) NOT NULL, -- label or DO ID
  secondary VARCHAR(12) NULL, -- DO ID
  like_do_id SMALLINT, -- boolean (1 or 0)
  FOREIGN KEY (_id) REFERENCES do_term (_id)
  );


  CREATE TABLE do_parents (
  _id INTEGER NOT NULL, -- REFERENCES do_term
  _parent_id INTEGER NOT NULL, -- REFERENCES do_term
  relationship_type VARCHAR(7) NOT NULL, -- type of DO child-parent
    relationship
```

```
FOREIGN KEY (_id) REFERENCES do_term (_id),
FOREIGN KEY (_parent_id) REFERENCES do_term (_id)
);

CREATE TABLE do_offspring (
_id INTEGER NOT NULL, -- REFERENCES do_term
_offspring_id INTEGER NOT NULL, -- REFERENCES do_term
FOREIGN KEY (_id) REFERENCES do_term (_id),
FOREIGN KEY (_offspring_id) REFERENCES do_term (_id)
);


CREATE TABLE do_obsolete (
do_id VARCHAR(12) PRIMARY KEY, -- DO ID
term VARCHAR(255) NOT NULL -- textual label for the DO term
)
;

CREATE TABLE map_counts (
map_name VARCHAR(80) PRIMARY KEY,
count INTEGER NOT NULL
);

CREATE TABLE map_metadata (
map_name VARCHAR(80) NOT NULL,
source_name VARCHAR(80) NOT NULL,
source_url VARCHAR(255) NOT NULL,
source_date VARCHAR(20) NOT NULL
);

CREATE TABLE metadata (
name VARCHAR(80) PRIMARY KEY,
value VARCHAR(255)
);

-- Indexes
```

And to get the number of mapped keys for the maps in DO.db, type following:

```
> DOMAPCOUNTS

 DOANCESTOR  DOCHILDREN  DOOBSOLETE DOOFFSPRING   DOPARENTS      DOTERM
       6272        1790        2327        1790        6272        6273
```

>