Package 'ontologics'

July 22, 2025

Title Code-Logics to Handle Ontologies

Version 0.7.4

Description Provides tools to build and work with an ontology of linked (open) data in a tidy workflow. It is inspired by the Food and Agrilculture Organizations (FAO) caliper platform <https://www.fao.org/statistics/caliper/web/> and makes use of the Simple Knowledge Organisation System (SKOS).

URL https://github.com/luckinet/ontologics

BugReports https://github.com/luckinet/ontologics/issues

Depends R (>= 3.5.0)

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports checkmate, rlang, tibble, dplyr, tidyr, tidyselect, magrittr, purrr, stringr, readr, httr, methods, rdflib, fuzzyjoin, utils, beepr

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Steffen Ehrmann [aut, cre] (ORCID: <https://orcid.org/0000-0002-2958-0796>), Arne Rümmler [aut, ctb] (ORCID: <https://orcid.org/0000-0001-8637-9071>), Carsten Meyer [ctb] (ORCID: <https://orcid.org/0000-0003-3927-5856>)

Maintainer Steffen Ehrmann <steffen.ehrmann@posteo.de>

Repository CRAN

Date/Publication 2025-01-17 16:50:02 UTC

Contents

`edit_matches` *Edit matches manually in a csv-table*

Description

Allows the user to match concepts with an already existing ontology, without actually writing into the ontology, but instead storing the resulting matching table as csv.

Usage

```
edit_matches(
  new,
  topLevel,
  source = NULL,
  ontology = NULL,
  matchDir = NULL,
  stringdist = TRUE,
  verbose = TRUE,
  beep = NULL
)
```

Arguments

`new`	`data.frame(.)` the new concepts that shall be manually matched, includes "label", "class" and "has_broader" columns.
`topLevel`	`logical(1)` whether or not the new concepts are at the highest level only, i.e., have to be matched without context, or whether they are contain columns that must be matched within parent columns.

source — character(1)
any character uniquely identifying the source dataset of the new concepts.

ontology — ontology(1)
either a path where the ontology is stored, or an already loaded ontology.

matchDir — character(1)
the directory where to store source-specific matching tables.

stringdist — logical(1)
whether or not to use string distance to find matches (should not be used for large datasets/when a memory error is shown).

verbose — logical(1)
whether or not to give detailed information on the process of this function.

beep — integerish(1)
Number specifying what sound to be played to signal the user that a point of interaction is reached by the program, see beep.

Details

In order to match new concepts into an already existing ontology, it may become necessary to carry out manual matches of the new concepts with already harmonised concepts, for example, when the new concepts are described with terms that are not yet in the ontology. This function puts together a table, in which the user would edit matches by hand. Whith the argument `verbose = TRUE`, detailed information about the edit process are shown to the user. After defining matches, and even if not all necessary matches are finished, the function stores a specific "matching table" with the name *match_SOURCE.csv* in the respective directory (`matchDir`), from where work can be picked up and continued at another time.

Fuzzy matching is carried out and matches with 0, 1 or 2 differing charcters are presented in a respective column.

Value

A table that contains all new matches, or if none of the new concepts weren't already in the ontology, a table of the already sucessful matches.

---

export_as_rdf *Export an ontology as RDF*

Description

Export an ontology as RDF

Usage

```
export_as_rdf(ontology, filename)
```

Arguments

ontology	ontology(1) an already loaded or created ontology object.
filename	character(1) the filename of the exported ontology. The format of the exported ontology is guessed by the extension of the filename. The guessing is performed by the rdflib package. Valid extensions are ".rdf" for "rdfxml", ".nt" for "ntriples", ".ttl" for "turtle" or ".json" for "jsonld".

Value

No return value, called for the side effect of exporting an ontology.

Examples

```
ontoDir <- system.file("extdata", "crops.rds", package = "ontologics")
onto <- load_ontology(path = ontoDir)

## Not run:

  export_as_rdf(ontology = onto, filename = "onto.ttl")

## End(Not run)
```

---

get_class *Get class(es) in an ontology*

Description

Get class(es) in an ontology

Usage

```
get_class(..., regex = FALSE, external = FALSE, ontology = NULL)
```

Arguments

...	combination of column name and value to filter that column by. The value to filter by can be provided as regular expression, if `regex = TRUE`.
regex	logical(1) whether or not the value in ... shall be matched in full, or whether any partial match should be returned.
external	logical(1) whether or not the external classes (TRUE), or the harmonized classes should be returned (FALSE, default).
ontology	ontology(1) either a path where the ontology is stored, or an already loaded ontology.

Value

A table of the class(es) in the ontology according to the values in ...

Examples

```
ontoDir <- system.file("extdata", "crops.rds", package = "ontologics")
onto <- load_ontology(path = ontoDir)

# exact classes from a loaded ontology ...
get_class(label = "class", ontology = onto)

# ... or one stored on the harddisc
get_class(id = ".xx.xx", ontology = ontoDir)

# use regular expressions ...
get_class(label = "ro", regex = TRUE, ontology = onto)

# get all sources
get_class(ontology = onto)
```

---

get_concept *Get a concept in an ontology*

Description

Get a concept in an ontology

Usage

```
get_concept(..., external = FALSE, matches = FALSE, ontology = NULL)
```

Arguments

...	combination of column name and value to filter that column by.
external	logical(1) whether or not to return merely the table of external concepts.
matches	logical(1) whether or not to include external concepts as label instead of id in the match columns of the harmonised concepts; this allows querying the external concepts in the harmonised concepts (only if external = FALSE).
ontology	ontology(1) either a path where the ontology is stored, or an already loaded ontology.

Value

A table of a subset of the ontology according to the values in ...

Examples

```
ontoDir <- system.file("extdata", "crops.rds", package = "ontologics")
onto <- load_ontology(path = ontoDir)

# exact matches from a loaded ontology ...
get_concept(label = "FODDER CROPS", ontology = onto)

# ... or a path
get_concept(label = c("FODDER CROPS", "CEREALS"), ontology = ontoDir)

# ignore querries that would not be valid in filter()
get_concept(label != 'Bioenergy woody' & has_broader == '.01', ontology = onto)

# extract concepts based on regular expressions
library(stringr)
get_concept(str_detect(label, "crop") & str_detect(id, ".03$"), ontology = ontoDir)
```

---

get_source *Get source(e) in an ontology*

Description

Get source(e) in an ontology

Usage

```
get_source(..., regex = FALSE, ontology = NULL)
```

Arguments

...	combination of column name and value to filter that column by. The value to filter by can be provided as regular expression, if `regex = TRUE`.
regex	`logical(1)` whether or not the value in ... shall be matched in full, or whether any partial match should be returned.
ontology	`ontology(1)` either a path where the ontology is stored, or an already loaded ontology.

Value

A table of the source(s) in the ontology according to the values in ...

Examples

```
ontoDir <- system.file("extdata", "crops.rds", package = "ontologics")
onto <- load_ontology(path = ontoDir)

# exact sources from a loaded ontology ...
get_source(label = "harmonised", ontology = onto)

# ... or one stored on the harddisc
get_source(version = "0.0.1", ontology = ontoDir)

# get all sources
get_source(ontology = onto)
```

`load_ontology` *Load an ontology*

Description

Load an ontology

Usage

```
load_ontology(path = NULL)
```

Arguments

`path`	`character(1)` the path where the ontology to load is stored.

Value

A table of the full ontology (i.e., where attribute and mapping tables are joined).

Examples

```
# load an already existing ontology
load_ontology(path = system.file("extdata", "crops.rds", package = "ontologics"))
```

make_tree *Make a tree of an ontology*

Description

Make a tree of an ontology

Usage

```
make_tree(..., reverse = FALSE, ontology = NULL)
```

Arguments

...	character(1) the concepts that shall be the target, combination of *'column name = value'*.
reverse	logical(1) whether or not to make a tree that gives the parents, instead of the children, of target concepts.
ontology	ontology(1) either a path where the ontology is stored, or an already loaded ontology.

new_class *Add a new valid class to an ontology*

Description

Add a new valid class to an ontology

Usage

```
new_class(new, target, description = NULL, ontology = NULL)
```

Arguments

new	character(1) the new class label.
target	character(1) the class into which the new class shall be nested.
description	character(1) a verbatim description of the new class.
ontology	ontology(1) either a path where the ontology is stored, or an already loaded ontology.

Value

the updated ontology that contains the new class(es) defined here.

Examples

```
ontoDir <- system.file("extdata", "crops.rds", package = "ontologics")
onto <- load_ontology(path = ontoDir)

onto <- new_class(new = "use type", target = "class", description = "something",
                  ontology = onto)
```

`new_concept` *Add a new concept to an ontology*

Description

This adds a new concept to an existing ontology to semantically integrate and thus harmonise it with the already existing ontology.

Usage

```
new_concept(
  new,
  broader = NULL,
  description = NULL,
  class = NULL,
  ontology = NULL
)
```

Arguments

`new`	`character(.)` the english label(s) of new concepts that shall be included in the ontology.
`broader`	`data.frame(.)` the english label(s) of already harmonised concepts to which the new concept shall be semantically linked via a skos:broader relation, see Details.
`description`	`character(.)` a verbatim description of the new concept(s).
`class`	`character(.)` the class(es) of the new labels.
`ontology`	`ontology(1)` either a path where the ontology is stored, or an already loaded ontology.

Value

returns invisibly a table of the new harmonised concepts that were added to the ontology, or a message that nothing new was added.

Examples

```
ontoDir <- system.file("extdata", "crops.rds", package = "ontologics")
onto <- load_ontology(path = ontoDir)

# add fully known concepts
concepts <- data.frame(
  old = c("Bioenergy woody", "Bioenergy herbaceous"),
  new = c("acacia", "miscanthus")
)

onto <- new_source(
  version = "0.0.1",
  name = "externalDataset",
  description = "a vocabulary",
  homepage = "https://www.something.net",
  license = "CC-BY-0",
  ontology = onto
)

onto <- new_concept(
  new = concepts$new,
  broader = get_concept(label = concepts$old, ontology = onto),
  class = "crop",
  ontology = onto
)

# add concepts where the nesting is clear, but not the new class
concepts <- data.frame(
  old = c("Barley", "Barley"),
  new = c("food", "bio-energy")
)

onto <- new_concept(
  new = concepts$new,
  broader = get_concept(label = concepts$old, ontology = onto),
  ontology = onto
)

# define that class ...
onto <- new_class(
  new = "use type", target = "class",
  description = "the way a crop is used", ontology = onto
)

# ... and set the concepts again
onto <- new_concept(
  new = concepts$new,
```

```
  broader = get_concept(label = concepts$old, ontology = onto),
  class = "use type",
  ontology = onto
)
```

`new_mapping` *Add a new mapping to an ontology*

Description

Extend an ontology by creating mappings between classes and concepts of external vocabularies and the harmonised classes and concepts.

Usage

```
new_mapping(
  new = NULL,
  target,
  source = NULL,
  lut = NULL,
  match = NULL,
  certainty = NULL,
  type = "concept",
  ontology = NULL,
  verbose = FALSE,
  beep = NULL
)
```

Arguments

`new`	`character(.)` the english external label(s) that shall be mapped to labels that do already exist in the ontology.
`target`	`data.frame(.)` the already harmonised English label(s) to which the external labels shall be mapped; derive with get_concept().
`source`	`character(1)` any character uniquely identifying the source dataset of the new label.
`lut`	`character(.)` in case the terms used for mapping are from a look up table (i.e. a standardised set of terms with a description), provide this table with column names 'label' and 'description' here.
`match`	`character(1)` the skos mapping property used to describe the link, possible values are `"close"`, `"exact"`, `"broad"` and `"narrow"`.

certainty — integerish(1)
the certainty of the match. Possible values are between 1 and 4, with meaning

- 1 = probably unreliable
- 2 = unclear, assigned according to a given definition
- 3 = clear, assigned according to a given definition
- 4 = original, harmonised term (can't be assigned by a user)

.

type — character(1)
whether the new labels are mapped to a "concept", or to a "class".

ontology — ontology(1)
either a path where the ontology is stored, or an already loaded ontology.

verbose — logical(1)
whether or not to give detailed information on the process of this function.

beep — integerish(1)
Number specifying what sound to be played to signal the user that a point of interaction is reached by the program, see beep.

Value

No return value, called for the side effect of adding new mappings to an ontology.

Examples

```
ontoDir <- system.file("extdata", "crops.rds", package = "ontologics")
onto <- load_ontology(path = ontoDir)

mapping <- data.frame(old = c("BIOENERGY CROPS", "Bioenergy woody",
                              "Other bioenergy crops"),
                      new = c("bioenergy plants", "Wood plantation for fuel",
                              "Algae for bioenergy"),
                      type = c("close", "broader", "broader"))

onto <- new_source(name = "externalDataset",
                   version = "0.0.1",
                   description = "a vocabulary",
                   homepage = "https://www.something.net",
                   license = "CC-BY-0",
                   ontology = onto)

onto <- get_concept(label = mapping$old, ontology = onto) %>%
  new_mapping(new = mapping$new,
              target = .,
              match = mapping$type,
              source = "externalDataset",
              certainty = 3,
              ontology = onto)
```

new_source *Add a new valid source to an ontology*

Description

Add a new valid source to an ontology

Usage

```
new_source(
  ontology = NULL,
  name = NULL,
  version = NULL,
  date = NULL,
  description = NULL,
  homepage = NULL,
  uri_prefix = NULL,
  license = NULL,
  notes = NULL
)
```

Arguments

ontology	ontology(1) either a path where the ontology is stored, or an already loaded ontology into which the new source should be included.
name	character(1) the name of the new source (must not contain empty spaces).
version	character(1) an optional version of the new source (any value is allowed, but should be a value that follows semantic versioning). Either version or date need to be given.
date	character(1) an optional date at which that version of an external vocabulary has been created. Should be a value of the form YYYY-MM-DD. Either version or date need to be given.
description	character(1) a verbatim description of the new source.
homepage	character(1) the homepage of the new source, typically the place where additional information or meta-data could be retrieved in a non-formalised way.
uri_prefix	character(1) the basic uniform resource locator (URL) all concepts of a new source have in common and which is thus the basis to construct the concept specific URI.
license	character(1) the licenses under which the new source is published.

notes character(1)
: any notes on the new source that don't fit into any of the other meta-data fields here.

Details

Fundamentally, there are two types of sources that can be defined with this function.

- *attribute collections*: where a collection of terms or concepts are associated as a descriptive attribute to the harmonised concepts, and
- *linked open data*: where the concepts that occur in another vocabulary or ontology and which are themselves part of linked datasets (and hence have a valid URI) are associated as related concepts to the harmonised concepts.

In the latter case, each mapped concept should be provided by its ID and the source needs to have a URL that allows in combination with the concept IDs to construct the URI under which the mapped concepts are stored in the semantic web.

Value

the updated ontology that contains the new source defined here.

Examples

```
ontoDir <- system.file("extdata", "crops.rds", package = "ontologics")
onto <- load_ontology(path = ontoDir)

onto <- new_source(name = "externalDataset",
                   version = "0.0.1",
                   description = "a vocabulary",
                   homepage = "https://www.something.net",
                   license = "CC-BY-0",
                   ontology = onto)
```

---

onto-class *Ontology class (S4) and methods*

Description

Ontology class (S4) and methods

Slots

sources data.frame(.)

classes data.frame(.)

concepts data.frame(.)

show,onto-method *Print onto in the console*

Description

Print onto in the console

Usage

```
## S4 method for signature 'onto'
show(object)
```

Arguments

object	object to show.

start_ontology *Start an ontology*

Description

Start an ontology

Usage

```
start_ontology(
  name = NULL,
  version = NULL,
  path = NULL,
  code = ".xx",
  description = NULL,
  homepage = NULL,
  uri_prefix = NULL,
  license = NULL,
  notes = NULL
)
```

Arguments

name	character(1) the path of the ontology.
version	character(1) the version of the ontology.
path	character(1) the path where the ontology shall be stored.

code — double(1)
format of a single code snippet that is concatenated for nested levels.

description — character(1)
a brief description of the new ontology.

homepage — character(1)
the URL to the homepage of the new ontology.

uri_prefix — character(1)
the basic URL to construct URIs for all concepts.

license — character(1)
any string describing the license under which this ontology can be (re)used.

notes — character(1)
any notes that might apply to this ontology.

Value

it returns the new, empty ontology and also stores that within the directory specified in path.

Examples

```
start_ontology(name = "crops", path = tempdir())
```

Index