ontoCAT: package for ontology parsing

Misha Kapushesky, Pavel Kurnosov, Natalja Kurbatova*

20 September, 2010

Version 0.0.1. released 20 September, 2010.

1 Introduction

The *ontoCAT* package provides a simple interface to the Experimental Factor Ontology (`http://www.ebi.ac.uk/efo`) and to any other ontology described in OWL or OBO format.

Package can load the ontology from a local file or on the fly from a URL and internally create the inferred ontology view. Experimental Factor Ontology (EFO) is the default ontology, loaded from: `http://efo.svn.sourceforge.net/viewvc/efo/trunk/src/efoinowl/InferredEFOOWLview/EFO_inferred.owl`. The package's methods allow to parse an ontology, search terms in it, find out term parents and children. The package is based on the Ontology Common API Tasks Java library (`http://www.ontocat.org`) as well as various other utilites methods and depends on *rJava* R package.

There are two clasess in the *ontoCAT* package:

- *OntologyParser* for parsing and representation of the ontology hierarchy.
- *OntologyTerm* for external view of an ontology term in the hierarchy.

2 Functions

The *ontoCAT*package provides a number of functions to work with ontologies in R.

2.1 Create Ontology Parser

To create an object of *OntologyParser* you can use one of the two methods:

- `getEFOParser()` loads the latest EFO version on the fly from the EFO SVN repository and creates `OntologyParser` object.
- `getOntologyParser("pathToOntology")` loads the ontology described in OWL or OBO format from a local file or a UR and creates `OntologyParser` object.

```
> library(ontoCAT)
> efo <- getEFOParser()
> biotop <- getOntologyParser("http://purl.org/biotop/biotop.owl#Disposition")
```

*mailto:natalja@ebi.ac.uk

2.2 Ontology Parsing

- To find all ontology terms two functions can be used: `getAllTerms` - returns a list of *OntologyTerm* objects. In turn, `getAllTermIds` - returns a list of term accessions.

```
> getAllTerms(biotop)
> getAllTermIds(efo)
```

- Function `getTermById` returns the accession number of the term. In turn, `getTermNameById` returns the name of the term.

```
> term <- getTermById(efo, "EFO_0000322")
> term_biotop <- getTermById(biotop, "DeadBody")
> getTermNameById(efo, "EFO_0000311")
> getTermNameById(biotop, "EmbryonicStructure")
```

- To find out all term parents or children the following functions can be used.

```
> getAllTermParents(efo, "EFO_0000322")
> getAllTermChildren(efo, "EFO_0000322")
> getAllTermParents(biotop, "DeadBody")
> getAllTermChildren(biotop, "DeadBody")
```

  Arguments: appropriate `OntologyParser` and the term accession.

- To find out only direct parents or children of the term functions `getTermParents` or `getTermChildren` can be used.

```
> getTermParents(efo, "EFO_0000322")
> getTermChildren(efo, "EFO_0000322")
> getTermParents(biotop, "DeadBody")
> getTermChildren(biotop, "DeadBody")
```

  Arguments: appropriate `OntologyParser` and the term accession.

- One more function to get term children together with the queried term accession is `getTermAndAllChildrenIds`.

```
> getTermAndAllChildrenIds(efo, "EFO_0000322")
> getTermAndAllChildrenIds(biotop, "DeadBody")
```

  Arguments: appropriate `OntologyParser` and the term accession.

- To create a flat subtree representation of the ontology "opened" down to the specified term function `getTreeDownTo` can be used. All possible paths from the root will be returned.

```
> getTreeDownTo(efo, "EFO_0000322")
```

  Arguments: appropriate `OntologyParser` and the term accession.

- A few simple functions allow to get term definition and synonyms:

```
> getDefinitions(efo, "EFO_0000322")
> getSynonyms(efo, "EFO_0000322")
```

  Arguments: appropriate `OntologyParser` and the term accession.

- A few simple functions allow to get some metadata about the used ontology:

```
> getOntologyAccession(efo)
> getOntologyDescription(efo)
```

Arguments: appropriate `OntologyParser`.

- To check if the term is present in the ontology function `hasTerm` can be used.

```
> hasTerm(efo, "CL000023")
> hasTerm(efo, "EFO_0000322")
```

Arguments: appropriate `OntologyParser` and the term accession.

- The following functions can be used to search terms in the ontology:

```
> searchTerm(efo, "thymus")
> searchTermPrefix(efo, "thym")
```

Arguments: appropriate `OntologyParser` and stringstring prefix to search for.

- The following functions can be used to investigate the ontology hierarchy:

```
> isRoot(efo, "EFO_0000322")
> getRoots(efo)
> getRootIds(efo)
```

For some ontologies these functions might fail when the ontology used was not design to have root classes.

2.3 Functions Specific for EFO

There are few functions specific for EFO class hierarchy to work with EFO branch roots.

```
> getEFOBranchRootIds(efo)
> isEFOBranchRoot(efo, "EFO_0000322")
```

2.4 Ontology Term

There are only three functions for the *OntologyTerm* class.

- `getLabel` to get term name;
- `getAccession` to get term accession;
- `show` to view the term.

```
> term <- getTermById(efo, "EFO_0000322")
> getLabel(term)
> getAccession(term)
> term
```

References

1. Adamusiak T, Burdett T, van der Velde K J, Abeygunawardena N, Antonakaki D, Parkinson H and Swertz M: OntoCAT – a simpler way to access ontology resources. *Available from Nature Precedings* `http://dx.doi.org/10.1038/npre.2010.4666.1` (2010)

2. Malone J, Holloway E, Adamusiak T, Kapushesky M, Zheng J, Kolesnikov N, Zhukova A, Brazma A, Parkinson H: Modeling Sample Variables with an Experimental Factor Ontology. *Bioinformatics* 2010, **26**(8):1112–1118

3. Experimental Factor Ontology `http://www.ebi.ac.uk/efo`

4. Ontology Common API Tasks java library `http://www.ontocat.org`

5. Java sources and javadocs: `http://sourceforge.net/projects/ontocat/files/`