

# *ontocat*: package for basic operations with ontologies

Natalja Kurbatova\*, Tomasz Adamusiak, Pavel Kurnosov, Morris Swertz and Misha Kapushesky

15 March, 2011

Version 1.2.1. released 6 June, 2011.

## 1 Introduction

The *ontocat* package:

- gives unified, format-independent access to ontology terms and the ontology hierarchy represented in OWL and OBO formats;
- provides basic methods for ontology traversal, such as searching for terms, listing a specific term's relations, showing paths to the term from the root element of the ontology, showing flattened-tree representations of the ontology hierarchy;
- supports working with groups of ontologies and with major public ontology repositories: searching for terms across ontologies, listing available ontologies and loading ontologies for further analysis as necessary.

In *ontocat* the subsumption “subclass/superclass” is supported in a user friendly form of “child – parent” relationship. No distinction is made between universals (classes) and particulars (instances) as they are both treated as ontology terms.

The package is based primarily on the Ontology Common API Tasks Java library, on the OWL API and depends on rJava R package. HermiT reasoner is used to support relationships. OBO ontologies are translated by OWL API into valid OWL format that can be reasoned over.

We provide two versions of *ontocat*:

- Light-weight *ontocat* package version is available in Bioconductor starting from release 2.7, and includes all single-ontology functionality except for methods to work with multiple ontologies and search in OLS and BioPortal.
- Full version includes batch methods and due to package size limitations is available only from the project website <http://www.ontocat.org/wiki/r>.

## 2 Basic operations with ontology

The *ontocat* package can load an ontology from a local file or on-the-fly from a URI. An inferred ontology view is created internally, using **Pellet** to classify the ontology. Ontologies described in OWL or OBO format are supported.

---

\*<mailto:natalja@ebi.ac.uk>

## 2.1 Create Ontology object

To create an object of *Ontology* you can use one of the three methods:

- `getEFO()` loads the latest EFO version on the fly from the EFO SVN repository and creates `Ontology` object.
- `getOntology("pathToOntology")` loads the ontology described in OWL or OBO format from a local file or a UR and creates `Ontology` object. Reasoning over ontologies and extracting relationships is supported by using Hermit reasoner.
- `getOntologyNoReasoning("pathToOntology")` loads the ontology described in OWL or OBO format from a local file or a UR and creates `Ontology` object without reasoning over loaded ontology.

```
> library(ontoCAT)
> efo <- getEFO()
> biotop <- getOntology("http://purl.org/biotop/biotop.owl")
```

## 2.2 Ontology traversal and search

- 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_efo <- 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.

```
> getAllTermParentsById(efo, "EFO_0000322")
> getAllTermChildrenById(biotop, "DeadBody")
> getAllTermParents(efo, term_efo)
> getAllTermChildren(biotop, term_biotop)
```

Arguments: appropriate `Ontology` object, the term's accession or `OntologyTerm` object.

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

```
> getTermParentsById(efo, "EFO_0000322")
> getTermChildrenById(efo, "EFO_0000322")
> getTermParents(efo, term_efo)
> getTermChildren(biotop, term_biotop)
```

Arguments: appropriate `Ontology` object, the term's accession or `OntologyTerm` object.

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

```
> getTermAndAllChildren(efo, term_efo)
> getTermAndAllChildrenById(biotop, "DeadBody")
```

Arguments: appropriate `Ontology` object, the term's accession or `OntologyTerm` object.

- To create a flat subtree representation of the ontology "opened" down to the specified term function `showHierarchyDownToTerm` can be used. Hierarchy of ontology will be shown down to requested term.

```
> showHierarchyDownToTermById(efo, "EFO_0000322")
> showHierarchyDownToTerm(efo, term_efo)
```

Arguments: appropriate `Ontology` object, the term's accession or `OntologyTerm` object.

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

```
> getTermDefinitionsById(efo, "EFO_0000322")
> getTermSynonymsById(efo, "EFO_0000322")
> getTermDefinitions(efo, term_efo)
> getTermSynonyms(efo, term_efo)
```

Arguments: appropriate `Ontology` object, the term's accession or `OntologyTerm` object.

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

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

Arguments: appropriate `Ontology`.

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

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

Arguments: appropriate `Ontology` and the term accession.

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

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

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

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

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

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

## 2.3 Relations

- To list all supported relations in ontology the following function can be used:

```
> getOntologyRelationNames(efo)
```

- To list relations that has particular term the following function can be used:

```
> getTermRelationNamesById(efo, "EFO_0000322")
> getTermRelationNames(efo, term_efo)
```

Arguments: appropriate *Ontology* object, the term's accession or *OntologyTerm* object.

- Please use the following functions to find out terms that are in some relation with the term of interest:

```
> getTermRelationsById(efo, "EFO_0000322", "has_part")
> getTermRelations(efo, term_efo, "has_part")
```

Arguments: appropriate *Ontology* object, the term's accession or *OntologyTerm* object, relation name.

## 2.4 Functions specific for EFO ontology

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

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

## 2.5 Ontology term object

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
```

## 3 Methods to Work Across Multiple Ontologies

The *ontoCAT* package provides methods to work with groups or “batches” of ontologies, local or web-based. Users can search for the terms across such resources and load specified individual ontologies by accession number. An Internet connection is required to load remote ontologies.

### 3.1 Create Ontology Batch

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

- `getEFOBatch()` without any arguments loads the EFO ontology. Ontologies can be added to an existing batch as needed via the `addOntology()` method.
- `getOntologyBatch("pathToDir")` creates a local batch of ontologies, taking a single argument: the path to the local directory containing ontology files.

```
> library(ontoCAT)
> batch <- getEFOBatch()
```

### 3.2 Term Searching

After a batch of ontologies is created, various methods for term searching become available.

- To search term in all ontologies included into the batch:  

```
> searchTermInBatch(batch, "thymus")
```
- Searches in two public ontology repositories are supported. The NCBO BioPortal is a web-based application for accessing and sharing biomedical ontologies, currently hosting **241** ontologies. The Ontology Lookup Service (OLS) is a web services SOAP API for querying multiple ontologies, hosting **81** ontologies.  

```
> searchTermInBioportal(batch, "thymus")
> searchTermInOLS(batch, "thymus")
```
- Search by using all available sources will search for the term in local batch and additionally in OLS and BioPortal  

```
> searchTermInAll(batch, "thymus")
```

### 3.3 Other Batch Methods

- To add ontology into the batch:  

```
> addOntology(batch, "http://purl.org/biotop/biotop.owl")
> addEFO(batch)
```
- To list all available ontologies in the local batch:  

```
> listLoadedOntologies(batch)
```
- To get ontology parser in order to work with single ontology parsing and querying methods:  

```
> ontology <- getOntologyFromBatch(batch, "EFO")
```

The second argument is the ontology accession.

## 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. OntoCAT website <http://www.ontocat.org/wiki/r>
6. Java sources and javadocs: <http://sourceforge.net/projects/ontocat/files/>