Phenol Documentation

Release 1.4.3

Peter N. Robinson, Sebastian Koehler, Sebastian Bauer, Julius OB

Jun 07, 2023

Contents:

Phenol is a modern Java (version 8 and above) library for working with phenotype and other attribute ontologies (including Gene Ontology). Phenol provides full support for working with the Human Phenotype Ontology and HPO-based disease annotations.

CHAPTER 1

Authors

- Sebastian Bauer
- Peter N. Robinson
- Sebastian Koehler
- Manuel Holtgrewe
- HyeongSik Kim
- Michael Gargano
- Jules Jacobsen

CHAPTER 2

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

2.1 Types of Contributions

2.1.1 Report Bugs

Report bugs at https://github.com/monarch-initiative/phenol/issues

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

2.1.2 Fix Bugs

Look through the Github issues for bugs. If you want to start working on a bug then please write short message on the issue tracker to prevent duplicate work.

2.1.3 Implement Features

Look through the Github issues for features. If you want to start working on an issue then please write short message on the issue tracker to prevent duplicate work.

2.1.4 Write Documentation

Phenol could always use more documentation, whether as part of the official vcfpy docs, in docstrings, or even on the web in blog posts, articles, and such.

Phenol uses Sphinx for the user manual (that you are currently reading). See *doc_guidelines* on how the documentation reStructuredText is used. See *doc_setup* on creating a local setup for building the documentation.

2.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/monarch-initiative/phenol/issues

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

2.2 Documentation Guidelines

For the documentation, please adhere to the following guidelines:

- Put each sentence on its own line, this makes tracking changes through Git SCM easier.
- Provide hyperlink targets, at least for the first two section levels.
- Use the section structure from below.

```
.. heading_1:

=========
Heading 1
=========


.. heading_2:

---------
Heading 2
---------


.. heading_3:

Heading 3
=========


.. heading_4:

Heading 4
---------


.. heading_5:
```

(continues on next page)

(continued from previous page)

```
Heading 5
~~~~~~~~~


.. heading_6:

Heading 6
:::::::::
```

2.3 Documentation Setup

For building the documentation, you have to install the Python program Sphinx. This is best done in a virtual environment. The following assumes you have a working Python 3 setup.

Use the following steps for installing Sphinx and the dependencies for building the phenol documentation:

```
$ cd phenol/manual
$ virtualenv -p python3 .venv
$ source .venv/bin/activate
$ pip install --upgrade -r requirements.txt
```

Use the following for building the documentation. The first two lines is only required for loading the virtualenv. Afterwards, you can always use `make html` for building.

```
$ cd phenol/manual
$ source .venv/bin/activate
$ make html  # rebuild for changed files only
$ make clean && make html  # force rebuild
```

2.4 Get Started!

Ready to contribute? First, create your Java/Documentation development setup as described in *install_from_source/doc_setup*.

1. Fork the *phenol* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/phenol.git
   ```

3. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making your changes, make sure that the build runs through. For Java:

   $ mvn package

   For documentation:

   $ make clean && make html

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

2.5 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated.
3. Describe your changes in the `CHANGELOG.md` file.

CHAPTER 3

Changelog

3.1 latest

3.2 v2.0.1

3.2.1 Minor changes

- Use Jackson annotations to configure serialization of *TermId* and *Identified*
- update dependencies

3.3 v2.0.0

- Upgrade to Java 11+, add `module-info` files
- Support for GO GAF 2.2 files
- Speed up build by adding a new build profile
- `phenol-core` - `MinimalOntology` has a version - do not use non-propagating relationships during ontology traversals
- `phenol-io` - Dropping support for reading OBO/OWL ontologies - drop non-modular *curie-util* dependency
- `phenol-annotations` - Remodel `HpoDisease`, `HpoAnnotation`, `HpoAssociationData`, `GeneIdentifier``s, etc.. - ``HpoDiseases` has a version - Model temporal elements - Implement `HGNCGeneIdentifierLoader` for reading `GeneIdentifiers` from HGNC complete set archive. - Add new `HpoOnset` terms. - Consolidate hardcoded HPO constants (`TermId``s) into ``org.monarchinitiative.phenol.annotations.constants.hpo` package - Standardize HPO annotations header, ensure the parsers can read the older releases. - Deprecate the code for parsing small files and move to hpoannotQC

3.4 v1.6.3

- added class for efficient precalculation of Resnik scores for HPO
- various bug fixes

3.5 v1.6.1

- Fixed issue with parsing Orphanet en_product4.xml

3.6 v1.6.0

- Simplified interface to GO overrepresentation analysis classes.
- Parent-Child Gene Ontology overrepresentation analysis with unit tests
- MGSA bugfix

3.7 v1.5.0

- improved functions for display of upper level HPO categories

3.8 v1.4.3

- fixing bug in parsing of MGI Genetic Marker file
- fixing bug in parsing of orphanet genes file, en_product6.xml

3.9 v1.4.2

- Prototype ingest of JSON ontology files
- flexible handling of relation types
- bug fix of previously incorrect handling of tRNA genes

3.10 v1.4.1

- Added workaround for duplicated lines in Homo_sapiens_gene_info file
- Added phenotype to gene extraction

3.11 v1.4.0

- Added Orphanet inheritance parser
- Aded several demonstration programs
- refactored files for constructing the phenotype.hpoa file

3.12 v1.3.3

- Various bug fixes
- Orphanet inheritance XML file ingest
- Adding additional demo app to show how to access Term information (hpdemo).

3.13 v1.3.0

- refactored TermAnnotation interface to use TermId instead of String to identify objects being annotation
- refactored GoGaf21Annotation class to use TermId internally instead of Strings for db and dbObjectId
- refactored to use junit 5 (allowing legacy use of junit 4, will migrate completely in coming releases)

3.14 v1.2.5-SNAPSHOT

- moving to SNAPSHOT version names to conform with maven standards
- fixed bug in initialized association lists for Gene Ontology analysis.

3.15 v1.1.4

- Adding parsing of onset, modifier, PMID/source to HpoAnnotation class
- Adding all relation types relevant to MONDO

3.16 v1.1.3

- Adding parsing of relations other than IS_A for Gene Ontology
- Fixing calculation of frequency (double) from frequency category
- allowing any valid curie as cross-ref

3.17 v1.1.2

- Adding MP annotation parser for MGI_GenePheno.rst and MGI_Pheno_Sex.rst

3.18 v1.1.1

- HPO Annotation parser now indexes diseases as a TermId representing the disease CURIE, e.g., MONDO:0000042.
- HPO Annotation parser now uses new 'big-file' format (with updated treatment of biocuration field)

3.19 v1.0.3

- refactored MP and GO parsing to use new OWLAPI-based parser
- adding support for adding artificial root to ontologies such as GO with multiple root terms.
- upgraded to obographs v0.1.1

3.20 v1.0.2

- refactored TermId to remove superfluous interface and renamed ImmutableTermId to TermId
- refactored TermSynonym to remove superfluous interface
- adding support for alt term ids to Owl2OboTermFactory (class renamed from GenericOwlFactory)
- adding support for database_cross_reference (usually PMID, ISBM, HPO, or MGI–added to term definitions)

3.21 v1.0.0

- completed refactoring to use single Term/Relationship. The API is not backwards compatible with versions prior to v0.1.9.

3.22 v0.1.9

- refactored to use just a single Term and Relationship instead of having separate types for each ontology. Simplified

classes that were templated to allow e.g., MpoTerm, MpoRelationship by hardcoding Term,Relationship and removing template.

3.23 v0.1.8

- refactored HpoAnnotation from HpoTermId

3.24 v0.1.7

- refactored phenol to use JGraphT library
- Adding OWLAPI based parser

- Refactoring HPO Disease annotation parser

3.25 v0.1.6

- refactored HPO disease annotation parser (changed API)

3.26 v0.1.5

- changed package and project name to phenol - Phenotype Ontology Library

3.27 v0.1.4

- fix to GOA parser
- added HPODiseaseWithMetaData parser
- added functions to calculate Term relationships (sibling, subclass, related, not-related)

3.28 v0.1.2

- refactored HpoFreqeuncy class to return frequencies (i.e., a number in [0,1]) rather than percentage
- Added HpoOnset classes
- Added HpoDiseaseWithMetadata class to encompass frequency and onset data

3.29 v0.4/v0.1.1

- forked from ontolib
- fixed mp.obo parse error
- fixed subontology creation error (TermMap, TermRelation)
- Adding Adding class `OntologyAlgorithm` with test class `OntologyAlgorithmTest`.

Implements functions to get children, parents, descendents and ancestors.

3.30 v0.3

- `xref` tags are now parsed and their content is available in `Term`. Added appropriate classes for representation.
- Added `Ontology.getParent()`.
- Removed `JaccardIcWeightedSimilarity`, `JiangSimilarity`, `LinSimilarity`, supporting code and tests.
- Refactoring the code for object score I/O into `ontolib-io` package.
- Adding support for score distribution reading and writing to H2 database files.

- `Ontology.getAncestorTermIds()` now also resolves alternative term IDs.
- Fixing dependency on slf4j components in `ontolib-core` and `ontolib-io`.
- Adding `getPrimaryTermId()` in `Ontology`.

3.31 v0.2

- Making date parser for HPO annotation files more robust. It works now for positive and negative associations.
- Small bug fix in HPO OBO parser.
- Adding `ontolib-cli` package that allows score distribution precomputation from the command line.
- Removed some dead code.
- Added various tests, minor internal refactoring.
- Moved `OntologyTerms` into `ontology.algo` package.

3.32 v0.1

- Everything is new.

CHAPTER 4

Working with HPO Annotations

Please see rstterm for information about how to access individual HPO terms. This tutorial intends to explain how to access HPO annotation data as contained in the phenotype.hpoa file.

4.1 Parsing Annotation Files

You can parse the phenotype-to-disease annotation files as follows.

```
import org.monarchinitiative.phenol.ontology.data.TermId;
import org.monarchinitiative.phenol.formats.hpo.HpoDisease;
import org.monarchinitiative.phenol.io.obo.hpo.HpoDiseaseAnnotationParser;
HpoDiseaseAnnotationParser annotationParser =
    new HpoDiseaseAnnotationParser(phenotypeAnnotationPath,ontology);
try {
  Map<TermId, HpoDisease> diseaseMap = annotationParser.parse();
  if (!annotationParser.validParse()) {
        int n = annotationParser.getErrors().size();
         logger.warn("Parse problems encountered with the annotation file at {}. Got
↪{} errors",
               this.phenotypeAnnotationPath,n);
  }
  return diseaseMap; // or do something else with the data
} catch (PhenolException e) {
     e.printStackTrace(); // or do something else
}
```

4.2 Parsing Annotation Files for Specific Sources

To limit the import to data representing diseases in the DECIPHER database, use the following code (the rest is identical). Currently, DECIPHER, OMIM, and ORPHA are available.

```
List<String> desiredDatabasePrefixes=ImmutableList.of("DECIPHER");
HpoDiseaseAnnotationParser annotationParser =
    new HpoDiseaseAnnotationParser(phenotypeAnnotationPath,ontology,
↪desiredDatabasePrefixes);
```

CHAPTER 5

Using phenol for Gene Ontology

phenol supports working with the GO in several following ways including some GO term enrichment analysis approaches.

5.1 Loading the data

To perform enrichment analysis, we require the GO ontology file, the annotation file, as well as a population set (e.g., all genes in a genome) and a study set (e.g., some set of genes determined to be differentially expressed).

```
Ontology gontology = OntologyLoader.loadOntology(new File(pathGoObo), "GO");
final GoGeneAnnotationParser annotparser = new GoGeneAnnotationParser(pathGoGaf);
List<TermAnnotation> goAnnots = annotparser.getTermAnnotations();
AssociationContainer associationContainer = new AssociationContainer(goAnnots);
Set<TermId> populationGenes = getPopulationSet(goAnnots);
StudySet populationSet = new StudySet(populationGenes,"population",
↪associationContainer,gontology);
Set<TermId> studyGenes = ... // get list of genes from study set
StudySet studySet = new StudySet(studyGenes,"study",associationContainer,gontology);
```

5.2 Perform testing for overrepresentation

In this example, we show how to use the exact Fisher test to assess term overrepresentation.

See the implementation in `GoEnrichmentDemo.java` for more details.

CHAPTER 6

Input

The phenol library is mainly intended to support working with the Human Phenotype Ontology, the Mammalian Phenotype Ontology, the Gene Ontology, MONDO, and ECTO, but has also been tested with the OBO version of NCIT.

6.1 Human Phenotype Ontology

To load the Human Phenotype Ontology (HPO), use the following code. The HPO is in the default curie map and only contains known relationships (is-a) and HP terms.

```
String hpoPath="/some/path/hp.obo";
Ontology hpoOntology = OntologyLoader.loadOntology(new File(hpoPath));
```

6.2 Mammalian Phenotype Ontology

The Mammalian Phenotype Ontology (MP) can be loaded using the same command.

```
String mpPath="/some/path/mp.obo";
Ontology hpoOntology = OntologyLoader.loadOntology(new File(mpPath));
```

6.3 Gene Ontology

The Gene Ontology (GO) is in the default curie map but also contains BFO and RO terms with unknown relationships we want to ignore these so here we specify the term prefixes we want to use. It has three possible root nodes (biological_process, cellular_component, biological_function) so an artificial GO:0000000 root is added.

```
String goPath="/some/path/go.obo";
Ontology goOntology = OntologyLoader.loadOntology(goPath, "GO")
```

6.4 Environmental conditions, treatments and exposures ontology

The Environmental conditions, treatments and exposures ontology (ECTO) contains multiple relationships so we're going to simplify this graph by only loading ECTO nodes (this ignores the true root term XCO:0000000) and other nodes from CHEBI. BFO and UBERON among others.

```
CurieUtil curieUtil = CurieUtilBuilder.withDefaultsAnd(ImmutableMap.of("ECTO", http://
↪purl.obolibrary.org/obo/ECTO_"));
Ontology ecto = OntologyLoader.loadOntology(ectoFile, curieUtil, "ECTO");
ecto.getRelationMap().values().forEach(relationship -> assertEquals(RelationshipType.
↪IS_A, relationship.getRelationshipType()));
// test if you like..
assertEquals(TermId.of("owl:Thing"), ecto.getRootTermId());
assertEquals(2272, ecto.countNonObsoleteTerms());
assertEquals(0, ecto.countObsoleteTerms());
```

CHAPTER 7

Installation

7.1 Use Maven Central Binaries

Note: This is the recommended way of installing for normal users.

Simply use the following snippet for your `pom.xml` for using phenol modules in your Maven project.

```
<dependencies>
  <dependency>
    <groupId>org.monarchinitiative.phenol</groupId>
    <artifactId>phenol-core</artifactId>
    <version>${project.version}</version>
  </dependency>
  <dependency>
    <groupId>org.monarchinitiative.phenol</groupId>
    <artifactId>phenol-io</artifactId>
    <version>${project.version}</version>
  </dependency>
</dependencies>
```

7.2 Install from Source

Note: You only need to install from source if you want to develop Phenol in Java yourself.

7.2.1 Prerequisites

For building Phenol, you will need

1. Java JDK 8 for compiling phenol,
2. Maven 3 for building phenol, and
3. Git for getting the sources.

7.2.2 Git Checkout and maven build

The following code snippet downloads the phenol sources and builds them.

```
$ git clone https://github.com/monarch-initiative/phenol
$ cd phenol
$ mvn package
```

7.2.3 Maven Proxy Settings

If you are behind a proxy, you will get problems with Maven downloading dependencies. If you run into problems, make sure to also delete `~/.m2/repository`. Then, execute the following commands to fill `~/.m2/settings.xml`.

```
$ mkdir -p ~/.m2
$ test -f ~/.m2/settings.xml || cat >~/.m2/settings.xml <<END
<settings>
  <proxies>
   <proxy>
      <active>true</active>
      <protocol>http</protocol>
      <host>proxy.example.com</host>
      <port>8080</port>
      <nonProxyHosts>*.example.com</nonProxyHosts>
    </proxy>
  </proxies>
</settings>
END
```

CHAPTER 8

License

Phenol is licensed under the BSD Clear 3-Clause License.

CHAPTER 9

How-To: Release on Maven Central

This page describes the steps to release Phenol on Maven Central.

9.1 Read the following first

- http://java.dzone.com/articles/deploy-maven-central
- http://central.sonatype.org/pages/apache-maven.html

9.2 Update the `README.rst` file

Change the version in the `README.rst`.

9.3 Update the `CHANGELOG.rst` file

- Update the `CHANGELOG.rst` file to reflect the new version.
- Create a new commit with this version.
- Do not create a git tag as this will be done by Maven below.

9.4 Prepare the Release using Maven

```
mvn release:prepare
```

Answer with the default everywhere but use "vMAJOR.MINOR" for giving the tag name, e.g. "v0.15". Eventually, this will update the versions, create a tag for the version and also push the tag to Github.

9.5 Perform the Release

```
mvn release:perform
```

Create the release and push it to Maven central/Sonatype.

9.6 Releasing the Deployment

Read this:

- http://central.sonatype.org/pages/releasing-the-deployment.html

The publisher backend to Maven Central is here:

- https://oss.sonatype.org/

9.7 Update README CHANGELOG

Open README.md and CHANGELOG.md and adjust the files to include the header for the next `SNAPSHOT` version.

9.8 Maven comments

- `mvn versions:set` is useful for bumping versions

CHAPTER 10

Quick Example

The following snippet will load the `hp.obo` file (which can be downloaded from the HPO website) into an `Ontology` object. The HPO has multiple subontologies, and the following code extracts all of the terms of the Phenotypic Abnormality subontology.

```
String hpoOboFilePath=...; // initialize to path of hp.obo file
Ontology ontology =  OntologyLoader.loadOntology(new File(hpoOboFilePath));
final TermId PHENOTYPIC_ABNORMALITY = TermId.of("HP:0000118");
for (TermId tid : getDescendents(ontology, PHENOTYPIC_ABNORMALITY)) {
  Term hpoTerm = ontology.getTermMap().get(tid);
  // ... do something with the Term
}
```

CHAPTER 11

History

Phenolib was forked from Ontolib in April 2018, and since has been extensively refactored and extended to provide additional functionality for working with phenotype annotations. Several classes from the Ontologizer, which we initially programmed in Java 1.5, have been refactored to support Gene Ontology overrepresentation analysis (see Bauer et al., 2008).

CHAPTER 12

Feedback

The best place to leave feedback, ask questions, and report bugs is the Phenol Issue Tracker.