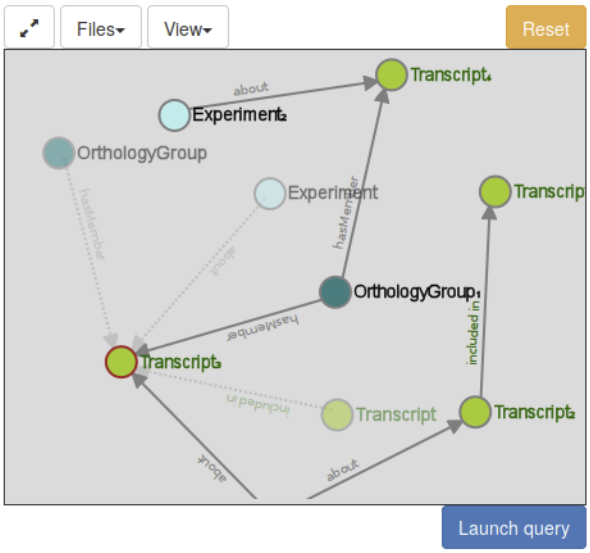
Python

Jan 16, 2020

Contents:

AskOmics is a visual SPARQL query interface supporting both intuitive data integration and querying while shielding the user from most of the technical difficulties underlying RDF and SPARQL


Files
View
Reset
Transcript4
about
Experiment2
OrthologyGroup
Experiment
Transcrip
hasMember
about
OrthologyGroup1
hasMember
included in
Transcript3
included in
Transcript
Transcript2
about
about
Launch query


Transcript3
Transcript
end
=
start
=
strand
minus

CHAPTER 1

Deployment

1.1 User

1.1.1 Dependencies

AskOmics need the Virtuoso triplestore to work.

Compile virtuoso

or install via docker

```
docker pull askomics/virtuoso

docker run --name my-virtuoso \
    -p 8890:8890 -p 1111:1111 \
    -e SPARQL_UPDATE=true \
    -v /tmp/virtuoso_data:/data \
    -d askomics/virtuoso
```

replace `/tmp/virtuoso` with a directory of your choice.

Your virtuoso is available at localhost:8890.

1.1.2 Manual installation

Dependencies

Installation needs some dependencies,

Ubuntu 18.04

```
sudo apt update
sudo apt install -y git python3 python3-venv python3-dev zlib1g-dev libsasl2-dev␣
↪libldap2-dev npm
```

Fedora 28/29

```
sudo dnf install -y git gcc gcc-c++ redhat-rpm-config zlib-devel bzip2 python3-devel␣
↪openldap-devel npm
```

Installation

Clone the AskOmics repository, and checkout the latest version

```
git clone https://github.com/askomics/askomics.git
cd askomics
# chekout the latest version
git checkout $(git describe --abbrev=0 --tags)
```

If you have installed virtuoso via docker, you have to inform AskOmics that the load url is not localhost:6543, but another ip address (dockers can't access host by http://localhost)

Run

```
docker exec my-virtuoso netstat -nr | grep '^0\.0\.0\.0' | awk '{print $2}'
```

and add

```
askomics.load_url=http://xxx.xx.x.x:6543
```

into `configs/production.virtuoso.ini` and `configs/development.virtuoso.ini` (replace `xxx.xx.x.x` with the ip obtained)

Install and run

```
./startAskomics.sh -d prod -t virtuoso
```

AskOmics is available at localhost:6543

Upgrade

Checkout the latest version is the AskOmics git directory.

```
git checkout $(git describe --abbrev=0 --tags)
```

1.1.3 Installation with docker

Pull the latest stable version of AskOmics

```
docker pull askomics/askomics
```

Run

```
docker run -p 6543:6543 askomics/askomics
```

AskOmics is available at localhost:6543

Upgrade with

```
docker pull askomics/askomics
```

1.1.4 Installation with docker-compose

Clone the askomics-docker-compose repository

```
git clone https://github.com/askomics/askomics-docker-compose
```

Choose which services you need and run with the docker-compose command. for example, if you need askomics+virtuoso :

```
cd askomics-docker-compose/virtuoso
docker-compose up -d
```

AskOmics is available at localhost/askomics

Upgrade with

```
# Stop dockers
docker-compose down
# upgrade the repo
git pull
# upgrade dockers
docker-compose pull
# start AskOmics
docker-compose up -d
```

1.2 Developer

Fork the AskOmics repository

then, clone your fork

```
git clone https://github.com/USERNAME/askomics.git # replace USERNAME with your␣
↪github username
```

Install AskOmics

Run it with dev mod

```
./startAskomics.sh -d dev -t virtuoso
```

AskOmics is available at localhost:6543

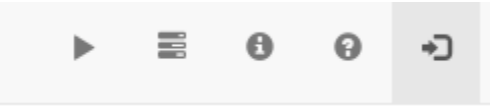
CHAPTER 2

AskOmics tutorials

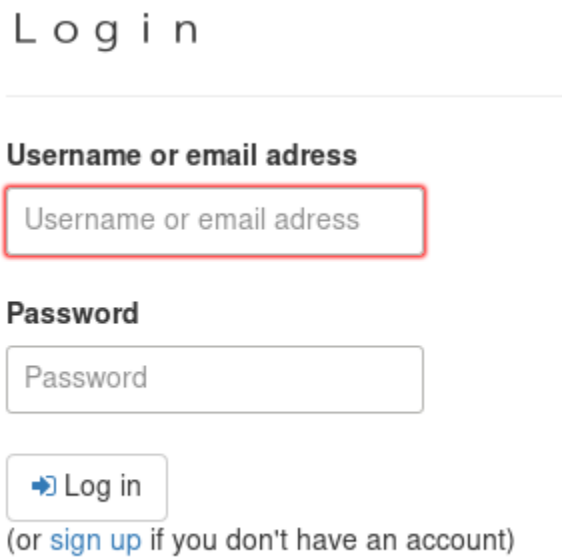
2.1 User account

2.1.1 Account creation

To use AskOmics, you will need an account. Go to the sign-up page by clicking on the login icon.



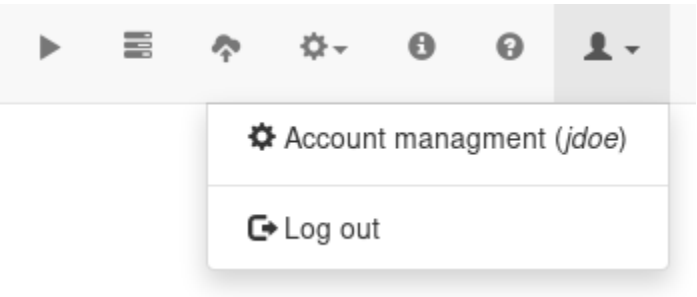
Then, click on the “sign up” link:



Fill the form with the requested information.

2.1.2 Account management

To manage your account, use the account management icon.



Update information

This section allows you to change your email address and your password.

API key

Your API key allows third-party applications (like Galaxy) to access AskOmics programmatically without revealing your personal password.

When updating your API key, old ones will no longer work.

Galaxy account

Link a Galaxy account to load Galaxy datasets into AskOmics.

Account deletion

The account deletion is permanent, all your information, as well as all your data will be deleted. There is no way back.

2.2 Use case 1: Gene expression

All files needed for the tutorial are available here

3 files are provided:

- gene.tsv: Genes locations on a genome
- orthogroup.tsv: Groups of ortholog genes
- differential_expression.tsv: Results of differential expression analysis

2.2.1 Files organization

AskOmics takes as inputs CSV (Comma-Separated Values) files. But these files have to respect a certain structure.

A CSV file describes an **entity**. The entity name is the header of the first column of the CSV file (e.g. the entity name of the file `gene.tsv` is *Gene*).

Other column headers describe the entity **attributes** and **relations**:

- An attribute is a simple column in the CSV file. For example, *Gene* have 5 attributes: *organism*, *chromosome*, *strand*, *start* and *end*.
- A relation allows to create a link between an entity and another one. It is described by a header like *relation_name@entity*. On the `orthogroup.tsv` file, *Orthogroup* entity have a *concerns* relation. This relation targets the *Gene* entity.

2.2.2 Uploading files

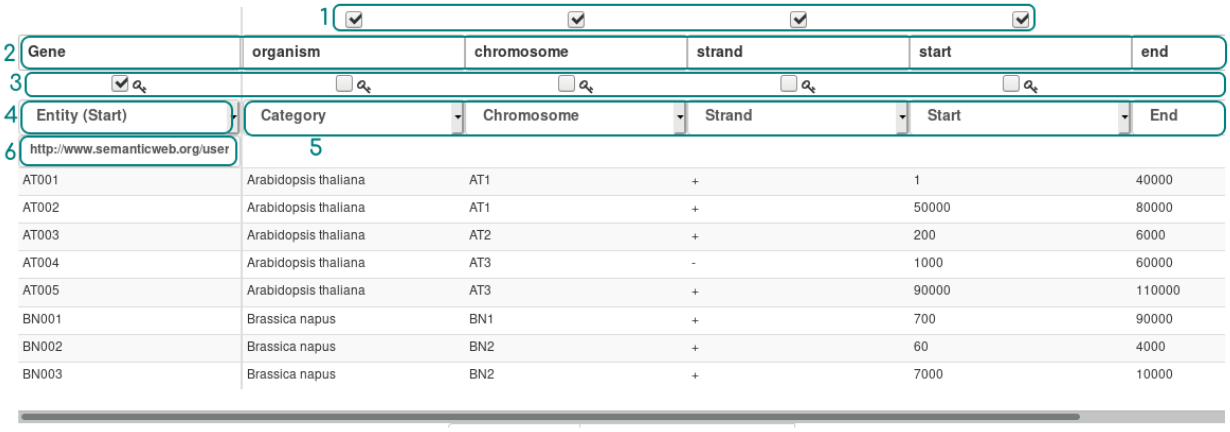
The first step is to upload your CSV files into AskOmics. Click on the *upload* icon to go to the upload page.

On the upload page, use the *Upload* button, and add the 3 files into the upload queue. Then, start uploading the files.

The CSV files are now uploaded on AskOmics.

2.2.3 Integrating files

On the upload page, select the Gene file to integrate, and click to the *Integrate* button. AskOmics shows an overview of the file.



1. Columns disabler: uncheck columns to ignore them (their content will not be loaded at all)
2. Header updater: optionally update entity or attribute names
3. Key columns: check several columns to create a new one by concatenate the columns checked
4. Entity type: choose between simple `entity` or `entity start` (default). An `entity start` will be displayed on the startpoint page.
5. Attributes types: select the attributes types (see below)

6. Custom URI: update the attributes URI (advanced feature)

Attributes can be one of the following types:

- Attributes
  - Numeric
  - Text
  - Category
  - Date/time
- Positionable attributes
  - Taxon
  - Chromosome
  - Strand
  - Start
  - End
- Relation
  - General relation to entity
  - Symmetric relation to entity

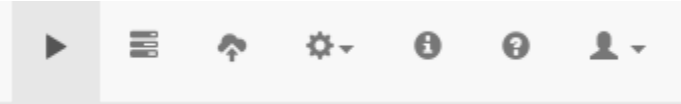
Types are automatically detected by AskOmics, but you can override them if needed. Depending on the type you choose, different options will be available in the query builder.

You can then integrate the 2 remaining files.

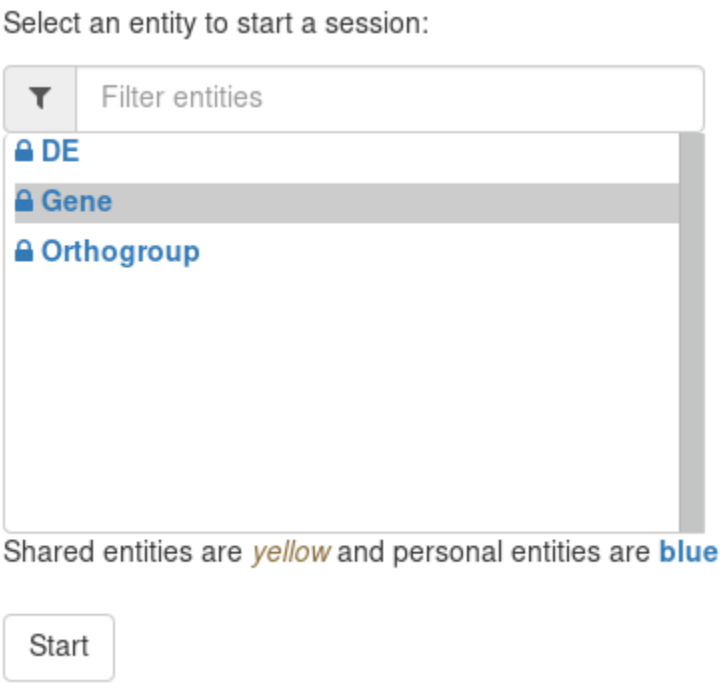
2.2.4 Interrogating datasets

Once you have integrated all the datasets, it's time to query them.

Click on the *Ask* icon



The page show you the starting points of you query. Select The *Gene* entity and start a query.



The *query builder* is composed of two panels: the *left panel*, representing entities and their relations, and the *right panel*, representing attributes of the selected entity.

On the left panel, the *Gene* entity is selected. We see two transparent node: *Orthogroup* and *DE*. These two nodes are proposed, but not instantiated.

On the right panel, attributes of *Gene* are displayed on **attributes cells**.

Simple query

Click on the *Launch query* button to perform a query. It leads to the job page, query section. Click on the query to display a preview of the results.

Results show all the *gene* URI present on the triplestore.

Gene1
Gene1
http://www.semanticweb.org/user/ontologies/2018/1#AT001
http://www.semanticweb.org/user/ontologies/2018/1#AT002
http://www.semanticweb.org/user/ontologies/2018/1#AT003
http://www.semanticweb.org/user/ontologies/2018/1#AT004
http://www.semanticweb.org/user/ontologies/2018/1#AT005
http://www.semanticweb.org/user/ontologies/2018/1#BN001
http://www.semanticweb.org/user/ontologies/2018/1#BN002
http://www.semanticweb.org/user/ontologies/2018/1#BN003

2.2.5 Display attributes

Return to the query builder (*Ask* tab). Now, we want to display some attributes of the genes.

On the right view, all attributes have button. Click on the *eye* button to display attributes.

The eye has 3 states:

- closed eye: the attribute won't appear in the results
- open eye: the attribute will appear in the results
- question mark: show the attribute, even if there is no value

Show the *organism*, *start* and *end* and launch the query.

Results show all the genes with their *organism*, *start* and *end*.

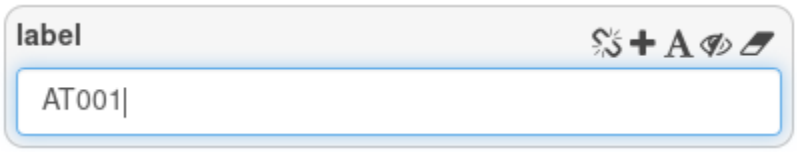
Gene1			
Gene1	**position_end1**	**position_start1**	**position_taxon1**
http://www.semanticweb.org/user/ontologies/2018/1#AT002	80000	50000	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#AT004	60000	1000	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#AT001	40000	1	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#AT005	110000	90000	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#AT003	6000	200	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#BN002	4000	60	Brassica napus
http://www.semanticweb.org/user/ontologies/2018/1#BN003	10000	7000	Brassica napus
http://www.semanticweb.org/user/ontologies/2018/1#BN001	90000	700	Brassica napus

2.2.6 Filter on attributes

Attributes can be filtered in different ways depending on their type (numeric, categorical or text).

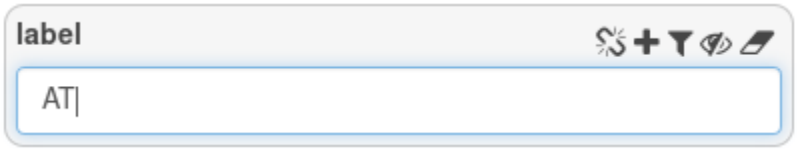
Text

Go back to the query builder. To filter on a text attributes, enter some test in the field.



Here, we ask for all entities that match exactly the string `AT001`. This query will return one result.

You can also use a regular expression filter by clicking on the A icon (this will change the icon into a funnel).

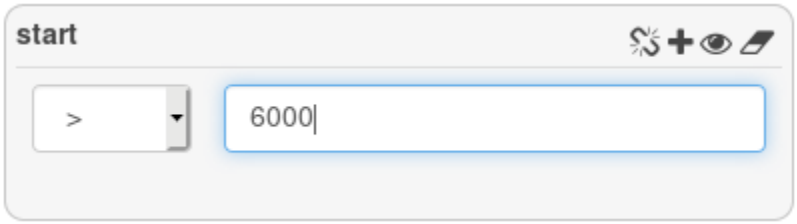


We ask for all genes whose label contains the `AT` string. This will return 5 results.

Numeric

Go back to the query builder and reset the label filter by clicking to the rubber icon.

Filter the start attribute to get all genes with a start position greater than 6000.

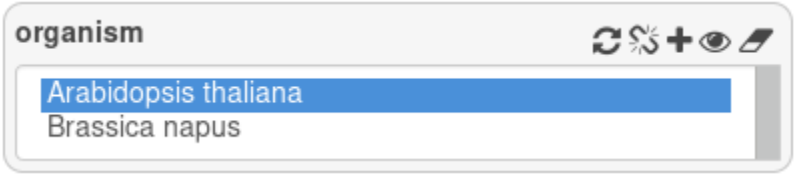


3 genes are returned.

Category

Attributes of type Category have a limited number of text value. Here, *strand* , *chromosome* and *taxon* are categories.

On the query builder, filter the organism to get all *Arabidopsis thaliana* gene.



5 genes are returned.

Other filtering features

Some other filtering functionalities are common to all the attributes:

- Negation: the + icon (e.g. if you want to find attributes with a value different to the one you entered)
- Cancel filter: use the rubber icon to reset the attribute filtering
- Link: the chain link link an attributes to the same attributes on another node

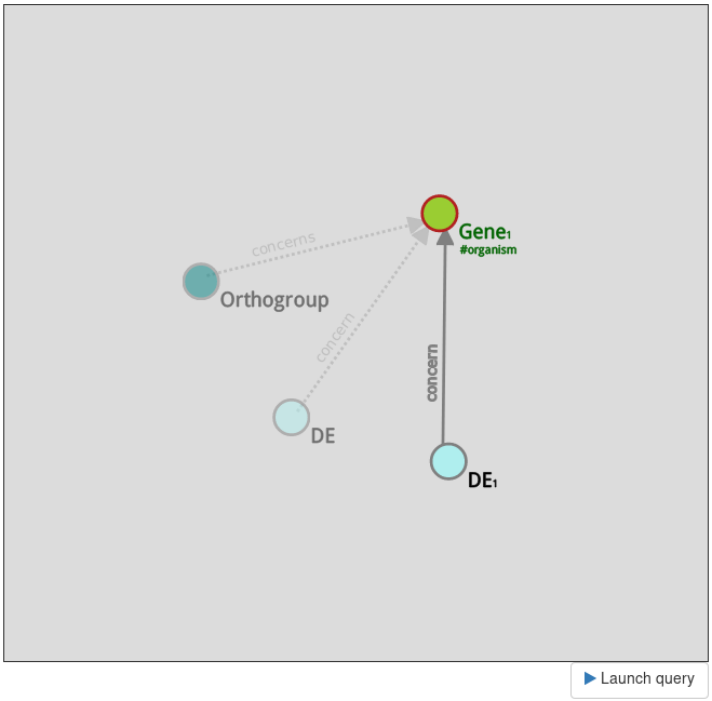
2.2.7 Link data

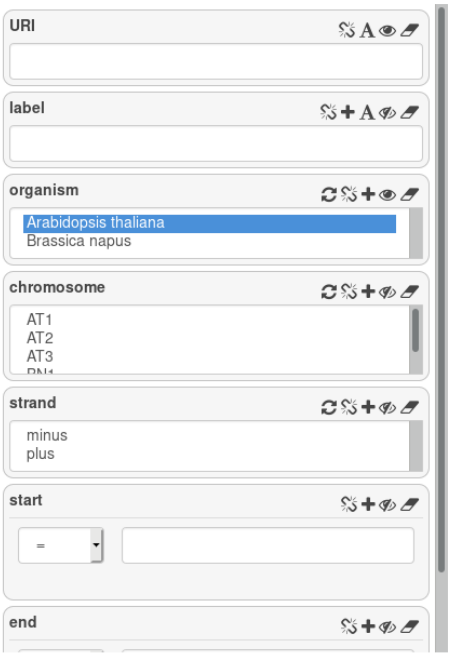
Back on the query builder, we will now cross *Gene* with *DE* and *Orthogroup*

Start to design a new query from scratch by clicking on the *Reset* button.

Start a new query with *DE*. This datasets contain results of gene differential expression analysis. Display Dpi (day post infection) and trend by clicking the eye on the attributes cells.

Then, instantiate the *Gene* node by clicking on it. Display *organism* and filter only the *Arabidopsis thaliana genes*.





This query gives you all differential expression measures that concern *Arabisopsis thaliana* species.

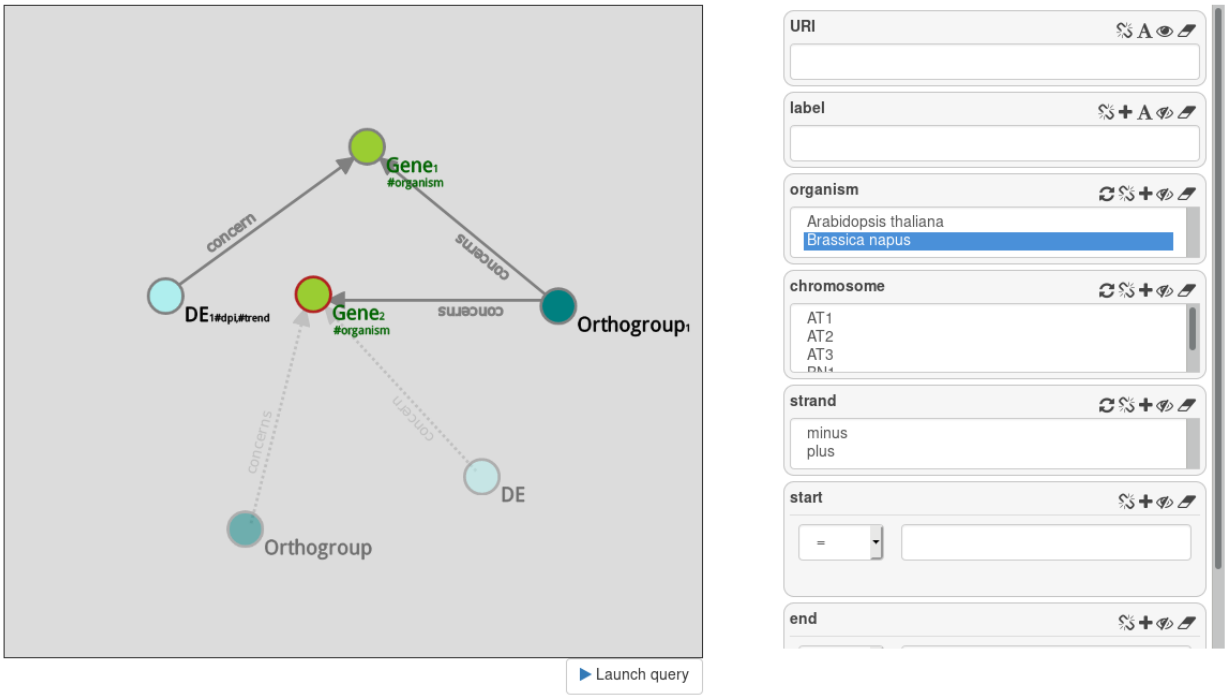
Go back to the *DE* node and filter attributes to get only genes that are overexpressed at day 7.

This query returns 5 results.

DE1			Gene1	
DE1	dpi1	trend1	Gene1	position_taxon1
http://www.semanticweb.org/user/ontologies/2018/1#7dpi_AT005	7dpi	down	http://www.semanticweb.org/user/ontologies/2018/1#AT005	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#7dpi_AT003	7dpi	down	http://www.semanticweb.org/user/ontologies/2018/1#AT003	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#7dpi_AT004	7dpi	down	http://www.semanticweb.org/user/ontologies/2018/1#AT004	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#7dpi_AT001	7dpi	down	http://www.semanticweb.org/user/ontologies/2018/1#AT001	Arabidopsis thaliana
http://www.semanticweb.org/user/ontologies/2018/1#7dpi_AT002	7dpi	down	http://www.semanticweb.org/user/ontologies/2018/1#AT002	Arabidopsis thaliana

Now, we want genes of *Brassica napus* that are ortholog to the *Arabidopsis thaliana* genes that are overexpressed at day 7.

Instanciate a *Orthogroup* node from the *Gene*. From this *Orthogroup* node, instanciate another *Gene* node, and filter it with *Brassica napus*.



We have 2 genes returned

DE1			Gene1		Orthogroup1	Gene2
DE1	dpi1	trend1	Gene1	position_taxon1	Orthogroup1	Gene2
http://www.semanticweb.org/user/ontologies/2018/1#7dpi_AT001	7dpi	down	http://www.semanticweb.org/user/ontologies/2018/1#AT001	Arabidopsis thaliana	http://www.semanticweb.org/user/ontologies/2018/1#OG001	http://www.semanticweb.org/user/ontologies/2018/1#BN001
http://www.semanticweb.org/user/ontologies/2018/1#7dpi_AT004	7dpi	down	http://www.semanticweb.org/user/ontologies/2018/1#AT004	Arabidopsis thaliana	http://www.semanticweb.org/user/ontologies/2018/1#OG002	http://www.semanticweb.org/user/ontologies/2018/1#BN003

Well done, you have completed the AskOmics tutorial! Now try with your own data.

2.2.8 Saving a query state

When you are proud of one of your query, you can save it for future reuse. On the query builder page, use the *Files* > *Save Query* to save the query state into your computer. This file represents the state of the query.

Later, on the ask page, you can upload this query file to work on your query again.

2.2.9 Download the results

The job page only shows you a preview of the results. To download the full results, click on *Save* to download the complete CSV file.

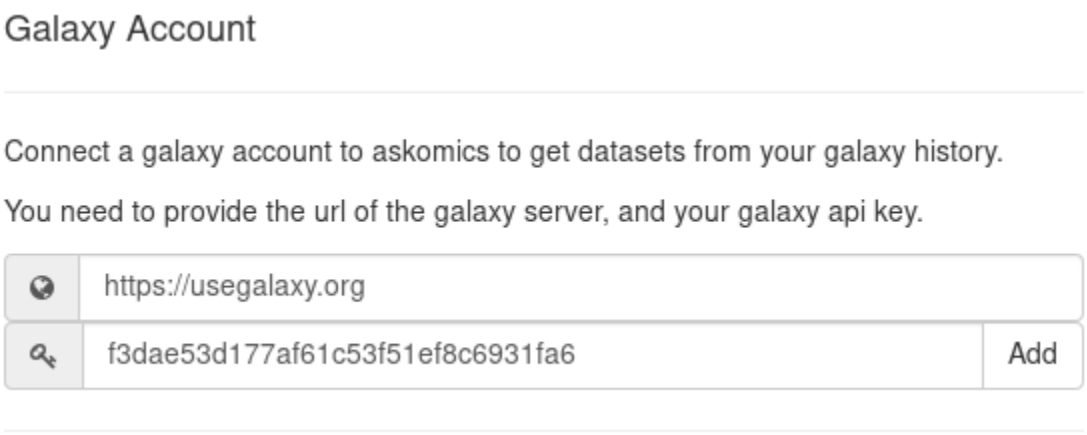
2.3 Use AskOmics with Galaxy

Galaxy is an open source, web-based platform for data intensive biomedical research. You can integrate Galaxy datasets into AskOmics by linking a Galaxy account into AskOmics.

2.3.1 Link Galaxy account into AskOmics

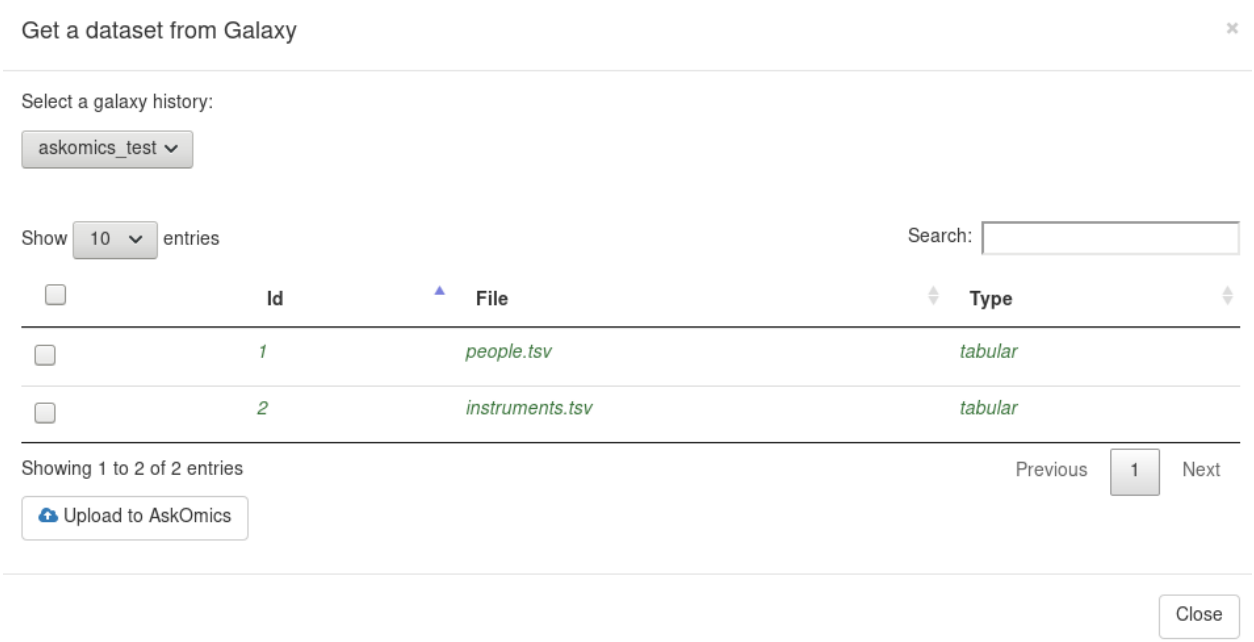
In you galaxy account, copy your Galaxy API key (User > Preferences > Manage API key).

Back in AskOmics, go to Account Management and add the Galaxy server URL and Galaxy API key



2.3.2 Upload a Galaxy datasets into AskOmics

On the upload page, you can now upload a Galaxy datasets with the button *Get from Galaxy*.

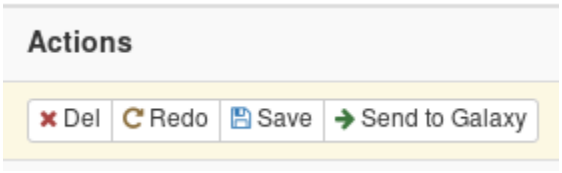


2.3.3 Save a query into Galaxy history

On the query builder page, you can save a query state into a galaxy history. You can also start a query with a saved state from galaxy on the ask page.

2.3.4 Save query results into Galaxy history

Result can be sent into galaxy on the job page. Use the *Send to Galaxy* button.



CHAPTER 3

Abstraction

3.1 Definition

What we called abstraction is the askomics ontology, this is what describe the data. It is quite small and defines what is a bubble and what is a link the the graphical interface. Its prefix is "askomics:".

- entity : what will be bubble, usually a owl:Class
- startPoint : an entity that could start an askomics query. What will be displayed in the first query page.
- attribute : what will be links between bubbles or bubble and value.
- category : what will be choice list, used in some attribute value.

3.2 Turtle Example

Here i show you the minimal information to provide as an abstraction.

3.2.1 prefixes

```
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>
@prefix owl:      <http://www.w3.org/2002/07/owl#> .
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
@prefix askomics: <askomics_is_good#> .
@base <scrap#> .
```

3.2.2 entity

```
# entity (startpoint to have a start, can be avoid in standard entity)
<People>
            askomics:entity "true"^^xsd:boolean ;
            rdfs:label "People"^^xsd:string ;
            askomics:startPoint "true"^^xsd:boolean ;
.
```

3.2.3 <entity> –relation–> value

```
# attribute DatatypeProperty
<First_name>
            askomics:attribute "true"^^xsd:boolean ;
            rdf:type    owl:DatatypeProperty ;
            rdfs:label  "First_name"^^xsd:string ;
            rdfs:domain <People> ;
            rdfs:range  xsd:string ;
.
```

3.2.4 <entity> –relation–> category=short list

```
# attribute DatatypeProperty
<Sex>
        askomics:attribute "true"^^xsd:boolean ;
        rdf:type    owl:DatatypeProperty ;
        rdfs:label  "Sex"^^xsd:string ;
        rdfs:domain <People> ;
        rdfs:range  <SexCategory> ;
.
<SexCategory>
        askomics:category <M>, <F> ;
.
<M>
    rdfs:label "M"^^xsd:string ;
.
<F>
    rdfs:label "F"^^xsd:string ;
.
```

3.2.5 <entity> –relation–> <entity>

```
# attribute ObjectProperty
<PlayWith>
        askomics:attribute "true"^^xsd:boolean ;
        rdf:type    owl:ObjectProperty ;
        rdfs:label  "play with"^^xsd:string ;
        rdfs:domain <People> ;
        rdfs:range  <People> ;
.
```

full file in people_mini.abstract.ttl

3.3 Python Management Code

As seen above, we have 2 kind of classes, “entity” and “attribute”/relation. To manage them (~get turtle strings), we use the 2 classes `AbstractedEntity__` and `AbstractedRelation__` in libaskomics/integration.

cf **python doc** to have details.

3.3.1 basics uses

```
ttl  += AbstractedEntity__( uri, label, startpoint=True ).get_turtle()
ttl  += AbstractedRelation__( uri, rdf_type, domain, range_, label ).get_turtle()
```

CHAPTER 4

Contribute to AskOmics

4.1 Issues

If you have an idea for a feature to add or an approach for a bugfix, it is best to communicate with developers early. The most common venues for this are GitHub issues.

4.2 Pull requests

All changes to AskOmics should be made through pull requests to this repository.

For the askomics repository to your account. To keep your copy up to date, you need to frequently sync your fork:

```
git remote add upstream https://github.com/askomics/askomics
git fetch upstream
git checkout master
git merge upstream/master
```

Then, create a new branch for your new feature

```
git checkout -b my_new_feature
```

Commit and push your modification to your fork. If your changes modify code, please ensure that is conform to *AskOmics style*

Write tests for your changes, and make sure that they *passes*.

Open a pull request against the master branch of askomics. The message of your pull request should describe your modifications (why and how).

The pull request should pass all the continuous integration tests which are automatically run by Github using Travis CI. The coverage must be at least remain the same (but it's better if it increases)

4.3 Tests

AskOmics use `nosetests` for Python tests.

4.3.1 Dependencies

Tests needs some services to work.

- A virtuoso instance
- A galaxy instance
- A Ldap server with some entry

You can use some docker images

```
# Virtuoso
sudo docker run -d --name test_virtuoso -p 127.0.0.1:8890:8890 -p 127.0.0.1:1111:1111␣
↪ -e DBA_PASSWORD=dba -e SPARQL_UPDATE=true -e DEFAULT_GRAPH=http://localhost:8890/
↪DAV --net="host" -t tenforce/virtuoso
# Galaxy
sudo docker run -d --name galaxy -p 8080:80 -p 8021:21 -p 8022:22 bgruening/galaxy-
↪stable
#ldap
sudo docker run -d --name simple-ldap -p 9189:389 -e ORGANISATION_NAME="AskoTests" -e␣
↪SUFFIX="dc=askotest,dc=org" -e ROOT_USER="admin" -e ROOT_PW_CLEAR="askotest" -e␣
↪FIRST_USER="true" -e USER_UID="jwick" -e USER_GIVEN_NAME="John" -e USER_SURNAME=
↪"Wick" -e USER_EMAIL="jwick@askotest.org" -e USER_PW_CLEAR="iamjohnwick" xgaia/
↪simple-ldap
```

4.3.2 Run tests

Activate the Python virtual environment and run nosetests.

```
source venv/bin/activate
nosetests
```

To skip the Galaxy tests, run

```
nosetests -a '!galaxy'
```

To target a single file test

```
nosetests --tests askomics/test/askView_test.py
```

The testing configuration is set in the `askomics/config/test.virtuoso.ini` INI file. You can see that the Galaxy account API key is `admin`. The docker image `bgruening/galaxy-stable` have a default admin account with this API key. If you use another galaxy instance, change the url and API key.

4.4 Coding style guidelines

4.4.1 General

Ensure all user-enterable strings are unicode capable. Use only English language for everything (code, documentation, logs, comments, . . .)

4.4.2 Python

We follow PEP-8, with particular emphasis on the parts about knowing when to be inconsistent, and readability being the ultimate goal.

- Whitespace around operators and inside parentheses
- 4 spaces per indent, spaces, not tabs
- Include docstrings on your modules, class and methods
- Avoid from module import *. It can cause name collisions that are tedious to track down.
- Class should be in `CamelCase`, methods and variables in `lowercase_with_underscore`

4.4.3 Javascript

We follow W3 JavaScript Style Guide and Coding Conventions

4.5 Contribute to docs

all the documentation (including what you are reading) can be found here. Files are on the AskOmics repository.

To preview the docs, run

```
cd askomics
# source the askomics virtual env
source venv/bin/activate
cd docs
make html
```

html files are in `build` directory.

CHAPTER 5

askomics package

5.1 Subpackages

5.1.1 askomics.libaskomics package

Subpackages

askomics.libaskomics.integration package

Submodules

askomics.libaskomics.integration.AbstractedEntity module

askomics.libaskomics.integration.AbstractedRelation module

Module contents

askomics.libaskomics.rdfdb package

Submodules

askomics.libaskomics.rdfdb.FederationQueryLauncher module

askomics.libaskomics.rdfdb.MultipleQueryLauncher module

askomics.libaskomics.rdfdb.QueryLauncher module

askomics.libaskomics.utils module

Module contents

5.2 Submodules

5.3 askomics.ask_view module

5.4 askomics.upload module

5.5 askomics.views module

5.6 Module contents

CHAPTER 6

Indices and tables

- genindex
- modindex
- search