pyCOMPASS Documentation

Release 0.5

Marco Moretto

May 05, 2022

Table of Contents

pyCOMPASS is a Python interface to COMPASS, the gene expression compendia GraphQL endpoint.

CHAPTER 1

Install pyCOMPASS

To install pyCOMPASS using pip just type:

```
pip install pycompass
```

CHAPTER 2

Examples

Here you can find a collection of Colab Notebook to show some use cases using the pyCOMPASS package

2.1 VESPUCCI

VESPUCCI is a COMMAND>_ technology-based database for exploring and analyzing a comprehensive *Vitis vinifera* specific cross-platform expression compendium. You can find use cases in the ReadTheDocs Use Cases page.

CHAPTER 3

API

3.1 Connect Class

class pycompass.connect.**Connect**(*url*)

Connect class is used to get a connection to a valid COMPASS GraphQL endpoint.

describe_compendia()

Get all available compendia

Returns dict of compendia structure

get_compass_version()

Get current backend version

Returns String

get_compendium(*name*, *version=None*, *database=None*, *normalization=None*)

Get a compendium by a given name, None otherwise

Parameters

- **name** – the compendium name
- **version** – the compendium version (use default if None)
- **database** – the compendium database (use default if None)
- **normalization** – the compendium normalization (use default if None)

Returns Compendium object

class pycompass.connect.**Connect**(*url*)

Connect class is used to get a connection to a valid COMPASS GraphQL endpoint.

describe_compendia()

Get all available compendia

Returns dict of compendia structure

get_compass_version()
Get current backend version

Returns String

get_compendium(*name*, *version=None*, *database=None*, *normalization=None*)
Get a compendium by a given name, None otherwise

Parameters

- **name** – the compendium name
- **version** – the compendium version (use default if None)
- **database** – the compendium database (use default if None)
- **normalization** – the compendium normalization (use default if None)

Returns Compendium object

3.2 Compendium Class

class pycompass.compendium.**Compendium**(**args*, ***kwargs*)
A Compendium object holds all the necessary information used to retrieve Experiments, Samples, BiologicalFeature objects, such as the compendium name, the version to be used as well as the Connection object

get_data_sources(*filter=None*, *fields=None*)
Get the experiments data sources both local and public

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of dict

get_platform_types(*filter=None*, *fields=None*)
Get the platform types

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of dict

get_score_rank_methods()
Get all the available ranking methods for biological features and sample sets

Returns

rank_biological_features(*module*, *rank_method=None*, *cutoff=None*)
Rank all biological features on the module's sample set using rank_method

Parameters

- **rank_method** –
- **cutoff** –

Returns

rank_sample_sets(*module*, *rank_method=None*, *cutoff=None*)
Rank all sample sets on the module's biological features using rank_method

Parameters

- **rank_method** –
- **cutoff** –

Returns

class pycompass.compendium.**Compendium**(**args*, ***kwargs*)
A Compendium object holds all the necessary information used to retrieve Experiments, Samples, BiologicalFeature objects, such as the compendium name, the version to be used as well as the Connection object

get_data_sources(*filter=None*, *fields=None*)
Get the experiments data sources both local and public

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of dict

get_platform_types(*filter=None*, *fields=None*)
Get the platform types

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of dict

get_score_rank_methods()
Get all the available ranking methods for biological features and sample sets

Returns

rank_biological_features(*module*, *rank_method=None*, *cutoff=None*)
Rank all biological features on the module's sample set using rank_method

Parameters

- **rank_method** –
- **cutoff** –

Returns

rank_sample_sets(*module*, *rank_method=None*, *cutoff=None*)
Rank all sample sets on the module's biological features using rank_method

Parameters

- **rank_method** –
- **cutoff** –

Returns

3.3 Module Class

class pycompass.module.**Module**(**args*, ***kwargs*)

A module is a subset of the entire compendium 2D matrix that holds the quantitative values. Rows are BiologicalFeatures and columns are SampleSets

add_biological_features(*biological_features=[]*)

Add biological feature to the module

Parameters **biological_features** – list of BioFeatures objects

Returns None

add_sample_sets(*sample_sets=[]*)

Add sample sets to the module

Parameters **sample_sets** – list of SampleSet objects

Returns None

create(*biofeatures=None*, *samplesets=None*, *rank=None*, *cutoff=None*)

Create a new module

Parameters

- **biofeatures** – the biofeatures list for the module (inferred if None)
- **samplesets** – the samplesets list for the module (inferred if None)
- **rank** – the rank method to be used for the inference
- **cutoff** – the cutoff to be used for the inference
- **normalization** – the normalization to be used for the inference

Returns a Module object

static difference(*first*, *second*, *biological_features=True*, *sample_sets=True*)

Difference between two modules

Parameters

- **first** – first module
- **second** – second module

Returns a new Module

get_description()

Get module brief sample description using ontology annotation terms

Returns ModuleDescription

get_enrichment(*bf_p_value=0.05*, *ss_p_value=0.05*)

Get module ontology annotation terms enrichment

Returns ModuleEnrichment

static intersection(*first*, *second*, *biological_features=True*, *sample_sets=True*)

Intersection of two modules

Parameters

- **first** – first module
- **second** – second module

Returns a new Module

static read_from_file(*filename*)
Read module data from a local file

Parameters **filename** –

Returns

remove_biological_features(*biological_features=[]*)
Remove biological feature from the module

Parameters **biological_features** – list of BioFeatures objects

Returns None

remove_sample_sets(*sample_sets=[]*)
Remove sample sets from the module

Parameters **sample_sets** – list of SampleSet objects

Returns None

split_module_by_biological_features()
Split the current module in different modules dividing the module in distinct groups of coexpressed biological features

Returns list of Modules

split_module_by_sample_sets()
Split the current module in different modules dividing the module in distinct groups of sample_sets showing similar values.

Returns list of Modules

static union(*first*, *second*, *biological_features=True*, *sample_sets=True*)
Union of two modules

Parameters

- **first** – first module
- **second** – second module

Returns a new Module

values
Get module values

Returns np.array

write_to_file(*filename*)
Dump a module into a local file

Parameters **filename** –

Returns

class pycompass.module.**Module**(**args*, ***kwargs*)
A module is a subset of the entire compendium 2D matrix that holds the quantitative values. Rows are BiologicalFeatures and columns are SampleSets

add_biological_features(*biological_features=[]*)
Add biological feature to the module

Parameters **biological_features** – list of BioFeatures objects

Returns None

add_sample_sets(*sample_sets=[]*)
Add sample sets to the module

Parameters **sample_sets** – list of SampleSet objects

Returns None

create(*biofeatures=None*, *samplesets=None*, *rank=None*, *cutoff=None*)
Create a new module

Parameters

- **biofeatures** – the biofeatures list for the module (inferred if None)
- **samplesets** – the samplesets list for the module (inferred if None)
- **rank** – the rank method to be used for the inference
- **cutoff** – the cutoff to be used for the inference
- **normalization** – the normalization to be used for the inference

Returns a Module object

static difference(*first*, *second*, *biological_features=True*, *sample_sets=True*)
Difference between two modules

Parameters

- **first** – first module
- **second** – second module

Returns a new Module

get_description()
Get module brief sample description using ontology annotation terms

Returns ModuleDescription

get_enrichment(*bf_p_value=0.05*, *ss_p_value=0.05*)
Get module ontology annotation terms enrichment

Returns ModuleEnrichment

static intersection(*first*, *second*, *biological_features=True*, *sample_sets=True*)
Intersection of two modules

Parameters

- **first** – first module
- **second** – second module

Returns a new Module

static read_from_file(*filename*)
Read module data from a local file

Parameters **filename** –

Returns

remove_biological_features(*biological_features=[]*)
Remove biological feature from the module

Parameters **biological_features** – list of BioFeatures objects

Returns None

remove_sample_sets(*sample_sets=[]*)
Remove sample sets from the module

Parameters **sample_sets** – list of SampleSet objects

Returns None

split_module_by_biological_features()
Split the current module in different modules dividing the module in distinct groups of coexpressed biological features

Returns list of Modules

split_module_by_sample_sets()
Split the current module in different modules dividing the module in distinct groups of sample_sets showing similar values.

Returns list of Modules

static union(*first*, *second*, *biological_features=True*, *sample_sets=True*)
Union of two modules

Parameters

- **first** – first module
- **second** – second module

Returns a new Module

values
Get module values

Returns np.array

write_to_file(*filename*)
Dump a module into a local file

Parameters **filename** –

Returns

3.4 Annotation Class

class pycompass.annotation.**Annotation**(*obj*)
The Annotation class wraps a BiologicalFeature or a Sample object to return its annoation

get_triples()
Return the annotation as a list of RDF triples

Returns list

plot_network(*output_format='html'*)
Return the Cytoscape JS representation of RDF graph used to annotate the BiologialFeature or Sample passed to the Annotation object

Parameters **output_format** – html or json

Returns

class pycompass.annotation.**Annotation**(*obj*)
The Annotation class wraps a BiologicalFeature or a Sample object to return its annoation

get_triples()
Return the annotation as a list of RDF triples

Returns list

plot_network(*output_format='html'*)
Return the Cytoscape JS representation of RDF graph used to annotate the BiologialFeature or Sample passed to the Annotation object

Parameters **output_format** – html or json

Returns

3.5 BiologicalFeature Class

class pycompass.biological_feature.**BiologicalFeature**(**args*, ***kwargs*)
A BiologicalFeature object represent the measured biological entity (tipically gene expression)

by(**args*, ***kwargs*)
Get BiolgicalFeature list from other high level objects

Parameters

- **args** –
- **kwargs** – sparql="SELECT ?s ?p ?o ..."

Returns

get(*filter=None*, *fields=None*)
Get biological feature

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of BiologicalFeature objects

class pycompass.biological_feature.**BiologicalFeature**(**args*, ***kwargs*)
A BiologicalFeature object represent the measured biological entity (tipically gene expression)

by(**args*, ***kwargs*)
Get BiolgicalFeature list from other high level objects

Parameters

- **args** –
- **kwargs** – sparql="SELECT ?s ?p ?o ..."

Returns

get(*filter=None*, *fields=None*)
Get biological feature

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of BiologicalFeature objects

3.6 Experiment Class

class pycompass.experiment.**Experiment**(**args*, ***kwargs*)
The Experiment class

get(*filter=None*, *fields=None*)
Get compendium experiments

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of Experiment objects

class pycompass.experiment.**Experiment**(**args*, ***kwargs*)
The Experiment class

get(*filter=None*, *fields=None*)
Get compendium experiments

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of Experiment objects

3.7 Ontology Class

class pycompass.ontology.**Ontology**(**args*, ***kwargs*)
Ontology class represent the different ontologies used to annotate BiologicalFeature and Sample objects

get(*filter=None*, *fields=None*)
Get the ontology used to annotate samples and biological features

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of Ontology objects

structure
Get the whole ontology hierarchy structure

:param :return: ontology structure in node-link format

class pycompass.ontology.**Ontology**(**args*, ***kwargs*)
Ontology class represent the different ontologies used to annotate BiologicalFeature and Sample objects

get(*filter=None*, *fields=None*)
Get the ontology used to annotate samples and biological features

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of Ontology objects

structure
Get the whole ontology hierarchy structure

:param :return: ontology structure in node-link format

3.8 Platform Class

class pycompass.platform.**Platform**(**args*, ***kwargs*)
The technological platform used to measure the BiologicalFeature in specific Sample

get(*filter=None*, *fields=None*)
Get the technological platforms used in the experiments

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of dict

class pycompass.platform.**Platform**(**args*, ***kwargs*)
The technological platform used to measure the BiologicalFeature in specific Sample

get(*filter=None*, *fields=None*)
Get the technological platforms used in the experiments

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of dict

3.9 Plot Class

class pycompass.plot.**Plot**(*module*)
The Plot class wraps a module and provides different methods to plot it

plot_distribution(*plot_type*, *output_format='html'*, *get_rank=False*, **args*, ***kwargs*)
Get the HTML or JSON code that plot module distributions

Parameters

- **plot_type** – the plot type
- **output_format** – html or json

Returns str

plot_heatmap(*plot_type=None*, *output_format='html'*, **args*, ***kwargs*)
Get the HTML or JSON code that plot module heatmaps

Parameters

- **plot_type** – the plot type
- **output_format** – html or json

Returns str

plot_network (*plot_type=None*, *output_format='html'*, **args*, ***kwargs*)
Get the HTML or JSON code that plot the module networks

Parameters

- **plot_type** – the plot type
- **output_format** – html or json

Returns str

class pycompass.plot.**Plot** (*module*)
The Plot class wraps a module and provides different methods to plot it

plot_distribution (*plot_type*, *output_format='html'*, *get_rank=False*, **args*, ***kwargs*)
Get the HTML or JSON code that plot module distributions

Parameters

- **plot_type** – the plot type
- **output_format** – html or json

Returns str

plot_heatmap (*plot_type=None*, *output_format='html'*, **args*, ***kwargs*)
Get the HTML or JSON code that plot module heatmaps

Parameters

- **plot_type** – the plot type
- **output_format** – html or json

Returns str

plot_network (*plot_type=None*, *output_format='html'*, **args*, ***kwargs*)
Get the HTML or JSON code that plot the module networks

Parameters

- **plot_type** – the plot type
- **output_format** – html or json

Returns str

3.10 Sample Class

class pycompass.sample.**Sample** (**args*, ***kwargs*)
The Sample class

by (**args*, ***kwargs*)
Get samples by using another object Example: Sample.using(compendium).by(platform=plt)

Returns list of Sample objects

get (*filter=None*, *fields=None*)
Get compendium samples

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of Sample objects

class pycompass.sample.**Sample**(**args*, ***kwargs*)
The Sample class

by(**args*, ***kwargs*)
Get samples by using another object Example: Sample.using(compendium).by(platform=plt)

Returns list of Sample objects

get(*filter=None*, *fields=None*)
Get compendium samples

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of Sample objects

3.11 SampleSet Class

class pycompass.sample_set.**SampleSet**(*s=()*, **args*, ***kwargs*)
A SampleSet class is a iterable class and is composed by a colletion of Sample objects. A SampleSet represent the measured condition.

by(**args*, ***kwargs*)
Get sample sets by using other objects Example:

Returns list of SampleSet objects

get(*filter=None*, *fields=None*)
Get the sample sets

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of SampleSet objects

class pycompass.sample_set.**SampleSet**(*s=()*, **args*, ***kwargs*)
A SampleSet class is a iterable class and is composed by a colletion of Sample objects. A SampleSet represent the measured condition.

by(**args*, ***kwargs*)
Get sample sets by using other objects Example:

Returns list of SampleSet objects

get(*filter=None*, *fields=None*)
Get the sample sets

Parameters

- **filter** – return results that match only filter values
- **fields** – return only specific fields

Returns list of SampleSet objects

3.12 RawData Class

class pycompass.raw_data.**RawData**(*sample*)
The RawData class wraps a Sample object to return its raw data

get_biofeature_reporters()
Get the reporter names (i.e. probes for microarray or gene locustag for RNA-seq)

Returns a numpy array of string

get_biofeatures()
Get the raw data BiologicalFeature objects

Returns a list of BiologicalFeature objects

get_value_types()
Get the raw data value type

Returns a numpy array of string

get_values()
Get raw data values

Returns a numpy array of float

class pycompass.raw_data.**RawData**(*sample*)
The RawData class wraps a Sample object to return its raw data

get_biofeature_reporters()
Get the reporter names (i.e. probes for microarray or gene locustag for RNA-seq)

Returns a numpy array of string

get_biofeatures()
Get the raw data BiologicalFeature objects

Returns a list of BiologicalFeature objects

get_value_types()
Get the raw data value type

Returns a numpy array of string

get_values()
Get raw data values

Returns a numpy array of float

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

R

S

U

V

W