pybiomart Documentation

Release 0.1+0.g2024323.dirty

Julian de Ruiter

March 28, 2016

Contents

Contents:

CHAPTER 1

Introduction

A simple and pythonic biomart interface for Python.

The intent of pybiomart is to provide a simple interface to biomart, which can be used to easily query biomart databases from Python. In this sense, pybiomart aims to provide functionality similar to packages such as biomaRt (which provides access to biomart from R).

1.1 Datasets

There main interface of pybiomart is provided by the *Dataset* class. A *Dataset* instance can be constructed directly if the name of the dataset and the url of the host are known:

```
>>> dataset = Dataset(name='hsapiens_gene_ensembl',
>>>                   host='http://www.ensembl.org')
```

1.1.1 Querying

Dataset instances can be used to query the biomart server using their *query* method. This method takes an optional argument *attributes* which specifies the attributes to be retrieved:

```
>>> dataset.query(attributes=['ensembl_gene_id', 'external_gene_name']})
```

The *query* method returns a pandas DataFrame instance, which contains a DataFrame representation of the requested attributes. If no attributes are given, the default attributes of the dataset are used. These default attributes can be identified using the *default_attributes* property of the dataset. A list of all available attributes can be obtained from the *attributes* property. Alternatively, a more convenient overview of all attributes can be obtained in DataFrame format using the *list_attributes* method.

1.1.2 Filtering

Dataset queries can be filtered to avoid fetching unneeded data from the server, thereby reducing the size of the result (and the required bandwidth):

```
>>> dataset.query(attributes=['ensembl_gene_id', 'external_gene_name'],
>>>               filters={'chromosome_name': ['1','2']})
```

The available filters depend on the dataset. All available filters can be accessed using the *filters* property or the *list_filters* method, the latter of which returns an overview of available filters in a DataFrame format. The type of a

filter describes what kind of values can be provided for a filter. For example, boolean filters require a boolean value, string filters require a string value, whilst list filters can take a list of values.

1.2 Servers and Marts

If the exact dataset not known, the *Server* and *Mart* classes can be used to explore the available marts and datasets on a biomart server. A server instance can be constructed using an optional host url (the url http://www.biomart.org is used by default). This instance can then be used to identify all available marts, either via the *marts* property or the *list_marts* method:

```
>>> server = Server(host='http://www.ensembl.org')
>>> server.list_marts()
```

Marts can be accessed by using the mart name as an index for the marts property, or directly as an index on the server instance. This mart instance can then similarly be used to identify datasets available in the mart, using the marts *datasets* property or its *list_datasets* method:

```
>>> mart = server['ENSEMBL_MART_ENSEMBL']
>>> mart.list_datasets()
```

Datasets can be retrieved from a mart instance by using the dataset name as an index on the mart object, or alternatively as an index for its *datasets* property.

```
>>> dataset = mart['hsapiens_gene_ensembl']
```

CHAPTER 2

API

2.1 pybiomart.Dataset

class pybiomart.**Dataset**(*name*, *display_name=''*, *host=None*, *path=None*, *port=None*, *use_cache=True*, *virtual_schema='default'*)

Class representing a biomart dataset.

This class is responsible for handling queries to biomart datasets. Queries can select a subset of attributes and can be filtered using any available filters. A list of valid attributes is available in the attributes property. If no attributes are given, a set of default attributes is used. A list of valid filters is available in the filters property. The type of value that can be specified for a given filter depends on the filter as some filters accept single values, whilst others can take lists of values.

Parameters

- **name** (*str*) – Id of the dataset.
- **display_name** (*str*) – Display name of the dataset.
- **host** (*str*) – Url of host to connect to.
- **path** (*str*) – Path on the host to access to the biomart service.
- **port** (*int*) – Port to use for the connection.
- **use_cache** (*bool*) – Whether to cache requests.
- **virtual_schema** (*str*) – The virtual schema of the dataset.

Examples

Directly connecting to a dataset:

```
>>> dataset = Dataset(name='hsapiens_gene_ensembl',
>>>                   host='http://www.ensembl.org')
```

Querying the dataset:

```
>>> dataset.query(attributes=['ensembl_gene_id',
>>>                           'external_gene_name'],
>>>               filters={'chromosome_name': ['1','2']})
```

Listing available attributes:

```
>>> dataset.attributes
>>> dataset.list_attributes()
```

Listing available filters:

```
>>> dataset.filters
>>> dataset.list_filters()
```

attributes
List of attributes available for the dataset (cached).

default_attributes
List of default attributes for the dataset.

display_name
Display name of the dataset.

filters
List of filters available for the dataset.

list_attributes()
Lists available attributes in a readable DataFrame format.

Returns Frame listing available attributes.

Return type pd.DataFrame

list_filters()
Lists available filters in a readable DataFrame format.

Returns Frame listing available filters.

Return type pd.DataFrame

name
Name of the dataset (used as dataset id).

query(*attributes=None*, *filters=None*, *only_unique=True*, *use_attr_names=False*)
Queries the dataset to retrieve the contained data.

Parameters

- **attributes** (*list[str]*) – Names of attributes to fetch in query. Attribute names must correspond to valid attributes. See the attributes property for a list of valid attributes.
- **filters** (*dict[str,any]*) – Dictionary of filters –> values to filter the dataset by. Filter names and values must correspond to valid filters and filter values. See the filters property for a list of valid filters.
- **only_unique** (*bool*) – Whether to return only rows containing unique values (True) or to include duplicate rows (False).
- **use_attr_names** (*bool*) – Whether to use the attribute names as column names in the result (True) or the attribute display names (False).

Returns DataFrame containing the query results.

Return type pandas.DataFrame

2.2 pybiomart.Server

class pybiomart.**Server**(*host=None*, *path=None*, *port=None*, *use_cache=True*)

Class representing a biomart server.

Typically used as main entry point to the biomart server. Provides functionality for listing and loading the marts that are available on the server.

Parameters

- **host** (*str*) – Url of host to connect to.
- **path** (*str*) – Path on the host to access to the biomart service.
- **port** (*int*) – Port to use for the connection.
- **use_cache** (*bool*) – Whether to cache requests.

Examples

Connecting to a server and listing available marts:

```
>>> server = Server(host='http://www.ensembl.org')
>>> server.list_marts()
```

Retrieving a mart:

```
>>> mart = server['ENSEMBL_MART_ENSEMBL']
```

list_marts()

Lists available marts in a readable DataFrame format.

Returns Frame listing available marts.

Return type pd.DataFrame

marts

List of available marts.

2.3 pybiomart.Mart

class pybiomart.**Mart**(*name*, *database_name*, *display_name*, *host=None*, *path=None*, *port=None*, *use_cache=True*, *virtual_schema='default'*, *extra_params=None*)

Class representing a biomart mart.

Used to represent specific mart instances on the server. Provides functionality for listing and loading the datasets that are available in the corresponding mart.

Parameters

- **name** (*str*) – Name of the mart.
- **database_name** (*str*) – ID of the mart on the host.
- **display_name** (*str*) – Display name of the mart.
- **host** (*str*) – Url of host to connect to.
- **path** (*str*) – Path on the host to access to the biomart service.

- **port** (*int*) – Port to use for the connection.
- **use_cache** (*bool*) – Whether to cache requests.
- **virtual_schema** (*str*) – The virtual schema of the dataset.

Examples

Listing datasets:

```
>>> server = Server(host='http://www.ensembl.org')
>>> mart = server.['ENSEMBL_MART_ENSEMBL']
>>> mart.list_datasets()
```

Selecting a dataset:

```
>>> dataset = mart['hsapiens_gene_ensembl']
```

database_name
Database name of the mart on the host.

datasets
List of datasets in this mart.

display_name
Display name of the mart.

list_datasets()
Lists available datasets in a readable DataFrame format.

Returns Frame listing available datasets.

Return type pd.DataFrame

name
Name of the mart (used as id).

CHAPTER 3

Development

The source code is currently hosted on GitHub at: https://github.com/jrderuiter/pybiomart. Any issues/requests should be reported there.

3.1 Installation

The package can be installed from pypi via pip:

```
pip install pybiomart
```

The development version can be installed from GitHub:

```
pip install git+https://github.com/jrderuiter/pybiomart.git#egg=pybiomart
```

3.2 Dependencies

- Python 3.3+, Python 2.7
- future, pandas, requests, requests-cache

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

A

D

F

L

M

N

Q

S