omicsdi Documentation

Release latest

Mar 19, 2020

Contents

Welcome to the OmicsDI Developer documentation. Here we provide documentation about different OmicsDI tools, libraries and the Restful API

CHAPTER 1

Contents

1.1 Introduction

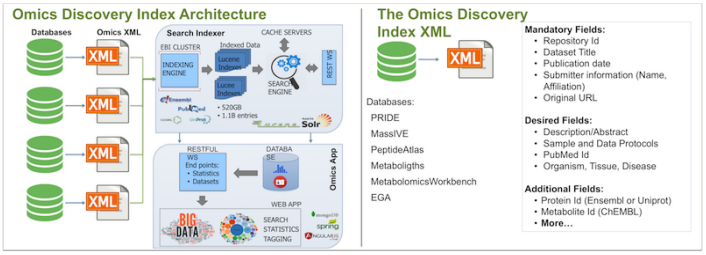
Individual libraries documentation

It can make your life easier if you want to explore individual libraries:

- *OmicsDI Restful Documentation*
- *ddipy: Python package*
- *ddiR: R package*

Omics Discovery Index is an integrated and open source platform facilitating the access and dissemination of omics datasets. It provides a unique infrastructure to integrate datasets coming from multiple omics studies, including at present proteomics, genomics, transcriptomics and metabolomics.

OmicsDI stores metadata coming from the public datasets from every resource using an efficient indexing system, which is able to integrate different biological entities including genes, proteins and metabolites with the relevant life science literature. OmicsDI is updated daily, as new datasets get publicly available in the contributing repositories.



After the data is submitted to a formal Archive, Knowledge Base Databases (BDs) reuse part of the public data to respond to specific questions (e.g. Gene Expression Profiles - ExpressionAtlas). The number of these DBs has growth in recent years (https://www.omicsdi.org/database).

Note: You can read a more about the topic here: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5831141/

1.2 OmicsDI Restful Documentation

Most data in the Datatsets Discovery Index can be accessed programmatically using a RESTful API. The API implementation is based on the Spring Rest Framework.

1.2.1 Web-browsable API

The OmicsDI API is web browsable, which means that:

- The query results returned by the API are available in JSONformat and also XML. This ensures that they can be viewed by human and accessed programmatically by computer.
- The main RESTful API page provides a simple web-based user interface, which allows developers to familiarize themselves with the API and get a better sense of the OmicsDI data before writing a single line of code.

many resources are hyperlinked so that it's possible to navigate the API in the browser.

As a result, developers can familiarize themselves with the API and get a better sense of the OmicsDI data.

1.2.2 API documentation

Responses containing multiple entries have the following fields:

- the count is the number of entries in the matching set.
- dataset is an array of datasets.
- facet is an array of facets.

Example

```
    http://www.omicsdi.org/ws/dataset/search?query=human
       {
          "count": 733,
          "datasets": [
               {
                 "id": "PXD000456",
                 "source": "pride",
                 "title": "Human glomerular extracellular matrix analysed by LC-MSMS",
                 "description": "Extracellular matrix proteins were isolated from␣
↪human glomeruli and analysed by LC-MSMS",
                 "keywords": [
                     "Human",
                     "kidney",
                     "glomerulus",
                     "extracellular matrix"
                 ],
                 "organisms": [
                    {
                      "acc": "9606",
                      "name": "Homo sapiens"
                    }
```

(continues on next page)

(continued from previous page)

```
                ],
                "publicationDate": "20140122"
            },
          // 19 more datasets
        ],
        "facets": [
          {
              "id": "modification",
              "label": "Modification",
              "total": 181,
              "facetValues": [
             {
                "label": "Unknown modification",
                "value": "unknown modification",
                "count": "5"
          },
          //other facet values
        ],
        },
      //other facets
      ]
    }
```

Responses containing just a single dataset have some extra navigation fields, and without the facets

```
 http://www.omicsdi.org/ws/dataset/get?acc=PXD001848&database=PRIDE
    {
        "id": "PXD001848",
        "name": "Global Analysis of Protein Folding Thermodynamics for Disease State␣
↪Characterization, MCF7 vs MDAMB231",
        "description": "Protein biomarkers can be used to characterize and diagnose␣
↪disease states such as cancer. ....",
        "keywords": null,
        "publicationDate": "20150410",
        "publications": [
            {
                "id": "25825992",
                "publicationDate": "2015-04-09",
                "title": "Global analysis of protein folding thermodynamics for␣
↪disease state characterization.",
                "pubabstract": "Current methods for the large-scale characterization␣
↪of disease states ....",
                "cycle": "testcyclehere"
            }
        ],
        "related_datasets": null,
        "data_protocol": "Peak lists were extracted from the raw LC-MS/MS data files␣
↪and the data were searched against t...."
    }
```

1.2.3 Pagination

Responses containing multiple datasets are paginated to prevent accidental downloads of large amounts of data and to speed up the `API`. The `page size` is controlled by the size parameter. Its default value is 20 datasets per page, and the maximum number of datasets per page is 100.

Another parameter is start which indicates the numeric order (starting from 0, not 1) of the first dataset in this page. Its default value is 0.

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human&start=0&size=50
- http://www.omicsdi.org/ws/dataset/search?query=human&start=0&size=20

1.2.4 Sort

The result datasets can be sorted using the title, description, publication date, accession id and the relevance of the query term.

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human&sort_field=id
- http://www.omicsdi.org/ws/dataset/search?query=human&sort_field=publication_date

1.2.5 Filtering

The API supports several filtering operations that complement the main `OmicsDI` search functionality.

Filtering by search term, there is 1 URL parameter: query

Examples

- http://www.omicsdi.org/ws/dataset/search?query=human
- http://www.omicsdi.org/ws/dataset/search?query=cancer

1.2.6 Filtering by omics type:

The omics type can be specified by adding terms in the query url parameter with key: omics_type (possible values: Proteomics, Metabolomics, Genomics, Transcriptomics).

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND omics_type:"Proteomics"

Filtering by database

The database can be specified by adding terms in the query URL parameter with key: repository (possible values: MassIVE, Metabolights, PeptideAtlas, PRIDE, GPMDB, EGA, Metabolights, Metabolomics Workbench, MetabolomeExpress, GNPS, ArrayExpress, ExpressionAtlas).

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND repository:"Metabolights"

Filtering by Organism

The organism can be specified by adding terms in the query URL parameter with key: TAXONOMY (possible values must be the TAXONOMY id: 9606, 10090...).

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND TAXONOMY:"9606"]

Filtering by Tissue

The tissue can be specified by adding terms in the query URL parameter with key: tissue (possible values: Liver, Cell culture, Brain, Lung...).

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND tissue:"Brain"

Filtering by Disease

The disease can be specified by adding terms in the query URL parameter with key: disease (possible values: Breast cancer, Lymphoma, Carcinoma, prostate adenocarcinoma...).

Examples

- http://www.omicsdi.org/ws/dataset/search?query=human AND tissue:"Breast cancer"

Filtering by Modification (in proteomics)

The Modifications (in proteomics) can be specified by adding terms in the query URL parameter with key: disease (possible values: Deamidated residue, Deamidated, Monohydroxylated residue, Iodoacetamide derivatized residue...).

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND modification:"iodoacetamide derivatized residue"

Filtering by Instruments & Platforms

The Instruments & Platforms can be specified by adding terms in the query URL parameter with key: instrument_platform (possible values: QSTAR, LTQ Orbitrap, Q Exactive, LTQ...).

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND instrument_platform:"Q Exactive"

Filtering by Publication Date

The Publication Date can be specified by adding terms in the query URL parameter with key: "publication_date" (possible values: 2015, 2014, 2013, 2014...).

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND publication_date:"2015"

Filtering by Technology Type

The Technology Type can be specified by adding terms in the query URL parameter with key: "technology_type" (possible values: Mass Spectrometry, Bottom-up proteomics, Gel-based experiment, Shotgun proteomics...).

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND technology_type:"Mass Spectrometry"

Combined filters

Any filters can be combined to narrow down the query using the AND operator. More logical operators will be supported in the future.

Examples:

- http://www.omicsdi.org/ws/dataset/search?query=human AND technology_type:"Shotgun proteomics" and AND modification:"monohydroxylated residue"

1.3 ddipy: Python package

An Python package to obtain data from the Omics Discovery Index. It uses the RESTful Web Services at OmicsDI WS for that purpose.

1.3.1 Installation

we need to install *ddipy*:

```
pip install ddipy
```

Table 1: Client Documents

Client	Method	Result Structure	Description
DatasetClient	search	*DataSetResult*	Search for datasets in the resource
	get_dataset_details	*DatasetSummary*	Retrieve an Specific Dataset
	get_dataset_files	array[string]	Retrieve the list of dataset's file using positions
	batch	*BatchDataset*	Retrieve a batch of datasets
	latest	*DataSetResult*	Retrieve the latest datasets in the repository
	most_accessed	*DataSetResult*	Retrieve an Specific Dataset
	get_file_links	array[string]	Retrieve all file links for a given dataset
	get_similar	*DataSetResult*	Retrieve the related datasets to one Dataset
	get_similar_by_pubmed	array[*DatasetSummary*]	Retrieve all similar dataset based on pubmed id
DatabaseClient	get_database_all	array[*DatabaseDetail*]	Get details of all databases
SeoClient	get_seo_home	*StructuredDataGraph*	Retrieve JSON+LD for home page
	get_seo_search	*StructuredData*	Retrieve JSON+LD for browse page
	get_seo_api	*StructuredData*	Retrieve JSON+LD for api page
	get_seo_database	*StructuredData*	Retrieve JSON+LD for databases page
	get_seo_dataset	*StructuredData*	Retrieve JSON+LD for dataset page
	get_seo_about	*StructuredData*	Retrieve JSON+LD for about page
TermClient	get_term_by_pattern	*DictWord*	Search dictionary Terms
	get_term_frequently_term_list	*Term*	Retrieve frequently terms from the Repo
StatisticsClient	get_statistics_organisms	array[*StatRecord*]	Return statistics about the number of datasets per Organisms
	get_statistics_tissues	array[*StatRecord*]	Return statistics about the number of datasets per Tissue
	get_statistics_omics	array[*StatRecord*]	Return statistics about the number of datasets per Omics Type
	get_statistics_diseases	array[*StatRecord*]	Return statistics about the number of datasets per dieases
	get_statistics_domains	array[*DomainStats*]	Return statistics about the number of datasets per Repository
	get_statistics_omics_by_year	array[*StatOmicsRecord*]	Return statistics about the number of datasets By Omics type on recent 5 years

1.3.2 Examples

DatasetClient

This example shows how retrieve details of one dataset by using the Python package ddipy.

```
from ddipy.dataset_client import DatasetClient

if __name__ == '__main__':
    client = DatasetClient()
    res = client.get_dataset_details("pride", "PXD000210", False)
```

This example shows a search for 20 the datasets for cancer human.

```
from ddipy.dataset_client import DatasetClient

if __name__ == '__main__':
   client = DatasetClient()
   res = client.search("cancer human", "publication_date", "ascending")
```

This example shows a search for 30 the datasets for cancer human and skip first 1200 datasets

```
from ddipy.dataset_client import DatasetClient

if __name__ == '__main__':
   client = DatasetClient()
   res = client.search("cancer human", "publication_date", "ascending", 1200, 30, 20)
```

This example is a query to retrieve all the datasets that reported the UniProt protein P21399 as identified.

```
from ddipy.dataset_client import DatasetClient

if __name__ == '__main__':
    client = DatasetClient()
    res = client.search("UNIPROT:P21399")
```

This example is a query to find all the datasets where the gene ENSG00000147251 is reported as differentially expressed.

```
from ddipy.dataset_client import DatasetClient

if __name__ == '__main__':
    client = DatasetClient()
    res = client.search("ENSEMBL:ENSG00000147251")
```

DatabaseClient

This example is a query to retrieve all databases recorded in OmicsDI

```
from ddipy.dataset_client import DatabaseClient

if __name__ == '__main__':
    client = DatabaseClient()
    res = client.get_database_all()
```

SeoClient

This example is retriveing JSON+LD for dataset page

```
from ddipy.dataset_client import SeoClient

if __name__ == '__main__':
     client = SeoClient()
     res = client.get_seo_dataset("pride", "PXD000210")
```

This example is retriveing JSON+LD for home page

```
from ddipy.dataset_client import SeoClient

if __name__ == '__main__':
     client = SeoClient()
     res = client.get_seo_home()
```

StatisticsClient

This example is a query for statistics about the number of datasets per Tissue

```
from ddipy.dataset_client import StatisticsClient

if __name__ == '__main__':
     client = StatisticsClient()
     res = client.get_statistics_tissues(20)
```

This example is a query for statistics about the number of datasets per dieases

```
from ddipy.dataset_client import StatisticsClient

if __name__ == '__main__':
     client = StatisticsClient()
     res = client.get_statistics_diseases(20)
```

TermClient

This example for searching dictionary terms

```
from ddipy.dataset_client import TermClient

if __name__ == '__main__':
     client = TermClient()
     res = client.get_term_by_pattern("hom", 10)
```

This example for retrieving frequently terms from the repo

```
from ddipy.dataset_client import TermClient

if __name__ == '__main__':
     client = TermClient()
     res = client.get_term_by_pattern("pride", "description", 20)
```

1.3.3 Structure

DataSetResult

Table 2: DataSetResult Structure

Name	Type
datasets	array[*DatasetSummary*]
facets	array[*Facet*]
count	integer

DatasetSummary

Table 3: DatasetSummary Structure

Name	Type
accession	string
database	string
title	string
description	string
dates	*Date*
scores	*Score*
keywords	array[string]
omics_type	array[string]
organisms	array[*Organism*]
cross_references	any
files	array[string]
additional	any

Date

Table 4: Date Structure

Name	Type
publication	string
submission	string
update	string

Score

Table 5: Score Structure

Name	Type
citationCount	integer
reanalysisCount	integer
searchCount	integer
viewCount	integer
connectionsCount	integer
downloadCount	integer

Organism

Table 6: Organism Structure

Name	Type
acc	string
name	string

Facet

Table 7: Facet Structure

Name	Type
facet_values	array[*FacetValue*]
label	string
total	integer
id	string

FacetValue

Table 8: FacetValue Structure

Name	Type
label	string
count	string
value	string

BatchDataset

Table 9: BatchDataset Structure

Name	Type
failure	array[*Failure*]
datasets	array[*DatasetSummary*]

Failure

Table 10: Failure Structure

Name	Type
database	string
accession	string
name	string
source_url	string

DatabaseDetail

Table 11: DatabaseDetail Structure

Name	Type
repository	string
orcid_name	string
url_template	string
accession_prefix	array[string]
title	string
img_alt	string
source_url	string
description	string
domain	string
image	array[byte]
icon	string
source	string
database_name	string

DictWord

Table 12: DictWord Structure

Name	Type
total_count	integer
items	array[*Item*]

Item

Table 13: Item Structure

Name	Type
name	string

Term

Table 14: Term Structure

Name	Type
frequent	string
label	string

StructuredDataGraph

Table 15: StructuredDataGraph Structure

Name	Type
graph	array[*StructuredData*]

StructuredData

Table 16: StructuredData Structure

Name	Type
logo	string
alternateName	string
potentialAction	*StructuredDataAction*
variableMeasured	string
sameAs	string
creator	array[*StructuredDataAuthor*]
citation	*StructuredDataCitation*
email	string
keywords	string
primaryImageOfPage	*StructuredDataImage*
description	string
image	string
name	string
context	string
type	string
url	string

StructuredDataAction

Table 17: StructuredDataAction Structure

Name	Type
query_input	string
type	string
target	string

StructuredDataAuthor

Table 18: StructuredDataAuthor Structure

Name	Type
name	string
type	string

StructuredDataCitation

Table 19: StructuredDataCitation Structure

Name	Type
author	*StructuredDataAuthor*
publisher	*StructuredDataAuthor*
name	string
type	string
url	string

StructuredDataImage

Table 20: StructuredDataImage Structure

Name	Type
author	string
contentUrl	string
contentLocation	string
type	string

StatRecord

Table 21: StatRecord Structure

Name	Type
label	string
name	string
value	string
id	string

DomainStats

Table 22: DomainStats Structure

Name	Type
domain	*StatRecord*
subdomains	array[*DomainStats*]

StatOmicsRecord

Table 23: StatOmicsRecord Structure

Name	Type
proteomics	string
transcriptomics	string
genomics	string
metabolomics	string
year	string

1.4 ddiR: R package

An R package to obtain data from the Omics Discovery Index OmicsDI. It uses its RESTful Web Services at OmicsDI WS for that purpose.

Currently, the following domain entities are supported:

- Dataset as S4 objects, including methods to get them from OmicsDI by accession and *as.data.frame*
- Publication as S4 objects, including methods to get them from OmicsDI by accession and *as.data.frame*
- Term as S4 objects, including methods to get them from OmicsDI by term and *as.data.frame*

1.4.1 Installation

First, we need to install *devtools*:

install.packages("devtools") library(devtools)

Then we just call

install_github("enriquea/ddiR") library(ddiR)

1.4.2 Examples

This example retrives all dataset details given accession and database identifier

```
library(ddiR)

dataset = get.DatasetDetail(accession="PXD000210", database="pride")

# print dataset full name
get.dataset.name(dataset)

# print dataset omics type
get.dataset.omics(dataset)
```

Access to all datasets for NOTCH1 gene

```
library(ddiR)

datasets <- search.DatasetsSummary(query = "NOTCH1")

sink("outfile.txt")
for(datasetCount in seq(from = 0, to = datasets@count, by = 100)){

   datasets <- search.DatasetsSummary(query = "NOTCH1", start = datasetCount, size =␣
↪100)

   for(dataset in datasets@datasets){
         dataset = get.DatasetDetail(accession=dataset.id(dataset),␣
↪database=database(dataset))
         print(paste(dataset.id(dataset), get.dataset.omics(dataset), get.dataset.
↪link(dataset)))
        }
   }
}
sink()
```

Getting the dataset IDs and full link of 20 Genomics studies in Cancer

```
datasets <- search.DatasetsSummary(query = "Cancer AND Genomics")

for(dataset in datasets@datasets){
    dataset = get.DatasetDetail(accession=dataset.id(dataset),␣
↪database=database(dataset))
    print(paste(dataset.id(dataset), get.dataset.link(dataset), sep = ' '))
}
```

Print the dataset IDs and short description of 20 Proteomics studies for tumor supressor p53

```
datasets <- search.DatasetsSummary(query = "p53 AND Proteomics")

for(dataset in datasets@datasets){
    dataset = get.DatasetDetail(accession=dataset.id(dataset),␣
↪database=database(dataset))
    print(paste(dataset.id(dataset), get.dataset.name(dataset), sep = ' '))
}
```

Getting Proteomics studies in Heart tissue from PRIDE database

```
datasets <- search.DatasetsSummary(query = "Heart")

for(dataset in datasets@datasets){
    dataset = get.DatasetDetail(accession=dataset.id(dataset),␣
↪database=database(dataset))
    if(database(dataset)=='pride')
     print(paste(dataset.id(dataset), get.dataset.tissues(dataset), get.dataset.
↪omics(dataset), sep = ' '))
}
```

This example shows how retrieve all the metadata similarity scores by using the R-package ddiR.

```
library(ddiR)
datasets <- search.DatasetsSummary(query = "*:*")
i  = 0
sink("outfile.txt")
for(datasetCount in seq(from = 0, to = datasets@count, by = 100)){

    datasets <- search.DatasetsSummary(query = "*:*", start = datasetCount, size =␣
↪100)

    for(dataset in datasets@datasets){
          Similar = get.MetadataSimilars(accession = dataset@dataset.id, database =␣
↪dataset@database)
          rank = 0
          for(similarDataset in Similar@datasets){
             print(paste(dataset@dataset.id, similarDataset@dataset.id,␣
↪similarDataset@score, dataset@omics.type, rank))
             rank = rank + 1
          }
    }
  }
  sink()
```

1.5 About

The OmicsDI Analysis Toolkit is developed by the following people:

- Yasset Perez-Riverol (EMBL-EBI)
- Enrique Audain (Kiel University)
- Gaurhari Dass (EMBL-EBI)
- Pan Xu (Beijin Proteomics Center)
- Ariana Barbera-Betancourt (Cambridge University)

1.5.1 Support

You can ask support questions here: https://github.com/OmicsDI/specifications/issues or send an email to omicsdi-support@ebi.ac.uk