omnipath

Release master (1.0.7)

Michal Klein, Dénes Türei

Jun 01, 2023

GENERAL

This package is a Python equivalent of an R package OmnipathR for accessing web service of *OmniPath* database developed by Saezlab.

CHAPTER

ONE

INSTALLATION

Omnipath requires Python version >= 3.7 to run.

1.1 PyPI

Omnipath is also available on PyPI:

```
pip install omnipath
```

Additionally, `omnipath` may sometimes require `networkx` to create an interaction graph. This dependency can be installed as:

```
pip install omnipath[graph]
```

1.2 Development Version

To stay up-to-date with the newest version, run:

```
pip install git+https://github.com/saezlab/omnipath
```

CHAPTER

TWO

API

Import Omnipath as:

```
import omnipath as op
```

2.1 Requests

requests.Annotations()	Request annotations from [OmniPath].
requests.Complexes()	Request information about protein complexes from [OmniPath].
requests.Enzsub()	Request enzyme-substrate relationships from [OmniPath].
requests.Intercell()	Request *intercell* annotations from [OmniPath].
requests.SignedPTMs()	Request enzyme-substrate relationships and interactions from [OmniPath].

2.1.1 omnipath.requests.Annotations

class omnipath.requests.**Annotations**

Request annotations from [OmniPath].

Methods

get([proteins, resources, ...])	Import annotations from [OmniPath].
params()	Return the available values for each parameter, if available.
pivot_annotations(df)	Annotations from narrow to wide format
resources(**kwargs)	Return the available resources for this query.

omnipath.requests.Annotations.get

classmethod `Annotations.get`(*proteins=None*, *resources=None*, *force_full_download=False*, *wide=False*, ***kwargs*)

Import annotations from [OmniPath].

Retrieves protein annotations about function, localization, expression, structure and other properties of proteins from OmniPath.

Parameters

- **proteins** (`Union`[`str`, `Iterable`[`str`], `None`]) – Genes or proteins for which annotations will be retrieved (UniProt IDs, HGNC Gene Symbols or miRBase IDs).

  In order to download annotations for proteins complexes, write **'COMPLEX:'** before the gene symbols of the genes integrating the complex.

  If *None*, fetch annotations for all available genes or proteins.

- **resources** (`Union`[`str`, `Iterable`[`str`], `None`]) – Load the annotations only from these databases. See *`resources()`* for available options. If *None*, use all available resources.
- **force_full_download** (`bool`) – Force the download of the entire annotations dataset. The full size of the data is ~1GB. We recommend to retrieve the annotations for a set of proteins or only from a few resources, depending on your interest.
- **wide** (`bool`) – Pivot the annotations from a long to a wide dataframe format, reconstituting the format of the original resource.
- **kwargs** – Additional query parameters.

Returns

A dataframe containing different molecule (protein, complex, gene, miRNA, small molecule) annotations. If *wide* is *True* and the result contains more than one resource, a *dict* of dataframes will be returned, one for each resource.

Return type

`pandas.DataFrame`

Notes

There might be also a few miRNAs and small molecules annotated. A vast majority of protein complex annotations are inferred from the annotations of the members: if all members carry the same annotation the complex inherits.

omnipath.requests.Annotations.params

classmethod `Annotations.params`()

Return the available values for each parameter, if available.

Return type

`Dict`[`str`, `Any`]

omnipath.requests.Annotations.pivot_annotations

classmethod Annotations.**pivot_annotations**(*df*)

Annotations from narrow to wide format

Converts the annotations from a long to a wide dataframe format, reconstituting the format of the original resource.

Parameters
df (DataFrame) – An annotation dataframe.

Returns
A dataframe of various molecule (protein, complex, gene, miRNA, small molecule) annotations. If the data contains more than one resource, a *dict* of dataframes will be returned, one for each resource.

Return type
pandas.DataFrame or *dict*

omnipath.requests.Annotations.resources

classmethod Annotations.**resources**(***kwargs*)

Return the available resources for this query.

Return type
Tuple[str]

2.1.2 omnipath.requests.Complexes

class omnipath.requests.**Complexes**

Request information about protein complexes from [OmniPath].

Methods

complex_genes(genes[, complexes, total_match])	Get all the molecular complexes for a given `genes`.
get(**kwargs)	Perform a request to the [OmniPath] web service.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.requests.Complexes.complex_genes

classmethod Complexes.**complex_genes**(*genes*, *complexes=None*, *total_match=False*)

Get all the molecular complexes for a given `genes`.

This function returns all the molecular complexes where an input set of genes participate. User can choose to retrieve every complex where any of the input genes participate or just retrieve these complexes where all the genes in input set participate together.

Parameters

- **genes** (Union[str, Iterable[str]]) – The genes for which complexes will be retrieved (hgnc format).

- **complexes** (Optional[DataFrame]) – Complex data from *get()*. If *None*, new request will be made.
- **total_match** (bool) – If *True*, get only complexes where all the genes participate together, otherwise get complexes where any of the genes participate.

Returns
The filtered `complexes`.

Return type
pandas.DataFrame

omnipath.requests.Complexes.get

classmethod Complexes.**get**(***kwargs*)

Perform a request to the [OmniPath] web service.

Parameters
kwargs – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
pandas.DataFrame

omnipath.requests.Complexes.params

classmethod Complexes.**params**()

Return the available values for each parameter, if available.

Return type
Dict[str, Any]

omnipath.requests.Complexes.resources

classmethod Complexes.**resources**(***kwargs*)

Return the available resources for this query.

Return type
Tuple[str]

2.1.3 omnipath.requests.Enzsub

class omnipath.requests.**Enzsub**

Request enzyme-substrate relationships from [OmniPath].

Imports the enzyme-substrate (more exactly, enzyme-PTM) relationships database.

Methods

get(**kwargs)	Perform a request to the [OmniPath] web service.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.requests.Enzsub.get

`classmethod Enzsub.get(**kwargs)`

Perform a request to the [OmniPath] web service.

Parameters
`kwargs` – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.requests.Enzsub.params

`classmethod Enzsub.params()`

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.requests.Enzsub.resources

`classmethod Enzsub.resources(**kwargs)`

Return the available resources for this query.

Return type
`Tuple[str]`

2.1.4 omnipath.requests.Intercell

`class omnipath.requests.Intercell`

Request *intercell* annotations from [OmniPath].

Imports the [OmniPath] inter-cellular communication role annotation database.

It provides information on the roles in inter-cellular signaling, e.g. if a protein is a ligand, a receptor, an extra-cellular matrix (ECM) component, etc.

Methods

categories()	Return categories from the *intercell* database.
generic_categories()	Return generic categories from the *intercell* database.
get(**kwargs)	Perform a request to the [OmniPath] web service.
params()	Return the available values for each parameter, if available.
resources([generic_categories])	Return the resources falling into the specified generic categories.

omnipath.requests.Intercell.categories

classmethod `Intercell.`**categories**()

Return categories from the *intercell* database.

Return type
`Tuple[str]`

omnipath.requests.Intercell.generic_categories

classmethod `Intercell.`**generic_categories**()

Return generic categories from the *intercell* database.

Return type
`Tuple[str]`

omnipath.requests.Intercell.get

classmethod `Intercell.`**get**(***kwargs*)

Perform a request to the [OmniPath] web service.

Parameters
kwargs – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.requests.Intercell.params

classmethod `Intercell.`**params**()

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.requests.Intercell.resources

classmethod `Intercell.resources`(*generic_categories=None*)

Return the resources falling into the specified generic categories.

Parameters

generic_categories (`Union[str, Sequence[str], None]`) – For valid options, see *generic_categories*.

Returns

The filtered resources according to `generic_categories`.

Return type

tuple

2.1.5 omnipath.requests.SignedPTMs

class `omnipath.requests.SignedPTMs`

Request enzyme-substrate relationships and interactions from [OmniPath].

PTM data does not contain sign (activation/inhibition), we generate this information based on the interaction network.

Methods

get([ptms, interactions])	Get signs for enzyme-PTM interactions.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.requests.SignedPTMs.get

classmethod `SignedPTMs.get`(*ptms=None*, *interactions=None*)

Get signs for enzyme-PTM interactions.

Parameters

- **ptms** (`Optional[DataFrame]`) – Data generated by *omnipath.requests.Enzsub.get()*. If *None*, a new request will be performed.
- **interactions** (`Optional[DataFrame]`) – Data generated by *omnipath.interactions.OmniPath.get()*. If *None*, a new request will be performed.

Returns

The signed PTMs with columns **'is_inhibition'** and **'is_stimulation'**.

Return type

`pandas.DataFrame`

omnipath.requests.SignedPTMs.graph

classmethod SignedPTMs.**graph**(*data=None*, ***kwargs*)

Create a graph.

Parameters

- **data** (Optional[DataFrame]) – The interaction data. If *None*, create a new request.
- **kwargs** – Keyword arguments for *get()* if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.requests.SignedPTMs.params

classmethod SignedPTMs.**params**()

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.requests.SignedPTMs.resources

classmethod SignedPTMs.**resources**(***kwargs*)

Return the available resources for this query.

Return type
`Tuple[str]`

2.2 Interactions

interactions.AllInteractions([include, exclude])	Request all [OmniPath] interaction datasets.
interactions.Dorothea()	Request interactions from the *dorothea* dataset.
interactions.KinaseExtra()	Request interactions from the *kinase extra* dataset.
interactions.LigRecExtra()	Request interactions from the *ligrec extra* dataset.
interactions.OmniPath()	Request interactions from the *omnipath* dataset.
interactions.PathwayExtra()	Request interactions from the *pathway extra* dataset.
interactions.PostTranslational([exclude])	Request all post-translational interactions from [OmniPath] .
interactions.TFmiRNA()	Request interactions from the *TF-miRNA* dataset.
interactions.TFtarget()	Request interactions from the *TF-target* dataset.
interactions.Transcriptional()	Request all *TF-target* interactions from [OmniPath].
interactions.lncRNAmRNA()	Request interactions from the *lncRNA-mRNA* dataset.
interactions.miRNA()	Request interactions from the *miRNA-target* dataset.
interactions.import_intercell_network([...])	Import intercellular network combining intercellular annotations and protein interactions.

2.2.1 omnipath.interactions.AllInteractions

`class omnipath.interactions.AllInteractions`(*include=None*, *exclude=None*)

Request all [OmniPath] interaction datasets.

The available interaction datasets are *`omnipath.constants.InteractionDataset`*.

Parameters

- **`include`** (`Union`[`str`, *`InteractionDataset`*, `Sequence`[`str`], `Sequence`[*`InteractionDataset`*], `None`]) – Interactions datasets to include from the [OmniPath] database. If *None*, include everything.
- **`exclude`** (`Union`[`str`, *`InteractionDataset`*, `Sequence`[`str`], `Sequence`[*`InteractionDataset`*], `None`]) – Interaction datasets to exclude from the [OmniPath] database. If *None*, don't exclude anything.

Methods

`get`([include, exclude])	Perform a request to the [OmniPath] web service.
`graph`([data])	Create a graph.
`params`()	Return the available values for each parameter, if available.
`resources`(**kwargs)	Return the available resources for this query.

omnipath.interactions.AllInteractions.get

`classmethod AllInteractions.get`(*include=None*, *exclude=None*, ***kwargs*)

Perform a request to the [OmniPath] web service.

The available interaction datasets are *`omnipath.constants.InteractionDataset`*.

Parameters

- **`include`** (`Union`[`str`, *`InteractionDataset`*, `Sequence`[`str`], `Sequence`[*`InteractionDataset`*], `None`]) – Interactions datasets to include from the [OmniPath] database. If *None*, include everything.
- **`exclude`** (`Union`[`str`, *`InteractionDataset`*, `Sequence`[`str`], `Sequence`[*`InteractionDataset`*], `None`]) – Interaction datasets to exclude from the [OmniPath] database. If *None*, don't exclude anything.
- **`kwargs`** – Additional query parameters.

Returns

The result which depends the type of the request and the supplied parameters.

Return type

`pandas.DataFrame`

omnipath.interactions.AllInteractions.graph

`classmethod AllInteractions.graph(data=None, **kwargs)`

Create a graph.

Parameters

- **data** (Optional[DataFrame]) – The interaction data. If *None*, create a new request.
- **kwargs** – Keyword arguments for `get()` if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.interactions.AllInteractions.params

`classmethod AllInteractions.params()`

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.interactions.AllInteractions.resources

`classmethod AllInteractions.resources(**kwargs)`

Return the available resources for this query.

Return type
`Tuple[str]`

2.2.2 omnipath.interactions.Dorothea

`class omnipath.interactions.Dorothea`

Request interactions from the *dorothea* dataset.

Imports the dataset which contains transcription factor (TF)-target interactions from DoRothEA.

Methods

get(**kwargs)	Perform a request to the [OmniPath] web service.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.interactions.Dorothea.get

classmethod `Dorothea.get`(***kwargs*)

Perform a request to the [OmniPath] web service.

Parameters
`kwargs` – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.Dorothea.graph

classmethod `Dorothea.graph`(*data=None*, ***kwargs*)

Create a graph.

Parameters

- `data` (`Optional`[`DataFrame`]) – The interaction data. If *None*, create a new request.
- `kwargs` – Keyword arguments for `get()` if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.interactions.Dorothea.params

classmethod `Dorothea.params`()

Return the available values for each parameter, if available.

Return type
`Dict`[`str`, `Any`]

omnipath.interactions.Dorothea.resources

classmethod `Dorothea.resources`(***kwargs*)

Return the available resources for this query.

Return type
`Tuple`[`str`]

2.2.3 omnipath.interactions.KinaseExtra

`class omnipath.interactions.KinaseExtra`

Request interactions from the *kinase extra* dataset.

Imports the dataset which contains enzyme-substrate interactions without literature reference.

The enzyme-substrate interactions supported by literature references are part of the `omnipath.requests.AllInteractions`.

Methods

get(**kwargs)	Perform a request to the [OmniPath] web service.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.interactions.KinaseExtra.get

`classmethod KinaseExtra.get(**kwargs)`

Perform a request to the [OmniPath] web service.

Parameters
`kwargs` – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.KinaseExtra.graph

`classmethod KinaseExtra.graph(data=None, **kwargs)`

Create a graph.

Parameters

- `data` (`Optional[DataFrame]`) – The interaction data. If *None*, create a new request.
- `kwargs` – Keyword arguments for *get()* if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.interactions.KinaseExtra.params

classmethod KinaseExtra.**params**()

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.interactions.KinaseExtra.resources

classmethod KinaseExtra.**resources**(***kwargs*)

Return the available resources for this query.

Return type
`Tuple[str]`

2.2.4 omnipath.interactions.LigRecExtra

class omnipath.interactions.**LigRecExtra**

Request interactions from the *ligrec extra* dataset.

Imports the dataset which contains ligand-receptor interactions without literature reference.

Methods

get(**kwargs)	Perform a request to the [OmniPath] web service.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.interactions.LigRecExtra.get

classmethod LigRecExtra.**get**(***kwargs*)

Perform a request to the [OmniPath] web service.

Parameters
kwargs – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.LigRecExtra.graph

`classmethod LigRecExtra.graph(data=None, **kwargs)`

Create a graph.

Parameters

- **data** (Optional[DataFrame]) – The interaction data. If *None*, create a new request.
- **kwargs** – Keyword arguments for `get()` if `data = None`.

Returns

The interaction graph.

Return type

`networkx.DiGraph`

omnipath.interactions.LigRecExtra.params

`classmethod LigRecExtra.params()`

Return the available values for each parameter, if available.

Return type

`Dict[str, Any]`

omnipath.interactions.LigRecExtra.resources

`classmethod LigRecExtra.resources(**kwargs)`

Return the available resources for this query.

Return type

`Tuple[str]`

2.2.5 omnipath.interactions.OmniPath

`class omnipath.interactions.OmniPath`

Request interactions from the *omnipath* dataset.

Imports the database, which contains only interactions supported by literature references.

This part of the interaction database was compiled in a similar way as it has been presented in [OmniPath16].

Methods

get(**kwargs)	Perform a request to the [OmniPath] web service.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.interactions.OmniPath.get

`classmethod OmniPath.get(**kwargs)`

Perform a request to the [OmniPath] web service.

Parameters
`kwargs` – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.OmniPath.graph

`classmethod OmniPath.graph(data=None, **kwargs)`

Create a graph.

Parameters

- `data` (`Optional[DataFrame]`) – The interaction data. If *None*, create a new request.
- `kwargs` – Keyword arguments for `get()` if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.interactions.OmniPath.params

`classmethod OmniPath.params()`

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.interactions.OmniPath.resources

`classmethod OmniPath.resources(**kwargs)`

Return the available resources for this query.

Return type
`Tuple[str]`

2.2.6 omnipath.interactions.PathwayExtra

`class omnipath.interactions.`**`PathwayExtra`**

Request interactions from the *pathway extra* dataset.

Imports the dataset which contains activity flow interactions without literature reference.

Methods

`get`(**kwargs)	Perform a request to the [OmniPath] web service.
`graph`([data])	Create a graph.
`params`()	Return the available values for each parameter, if available.
`resources`(**kwargs)	Return the available resources for this query.

omnipath.interactions.PathwayExtra.get

`classmethod PathwayExtra.`**`get`**(***kwargs*)

Perform a request to the [OmniPath] web service.

Parameters

`kwargs` – Additional query parameters.

Returns

The result which depends the type of the request and the supplied parameters.

Return type

`pandas.DataFrame`

omnipath.interactions.PathwayExtra.graph

`classmethod PathwayExtra.`**`graph`**(*data=None*, ***kwargs*)

Create a graph.

Parameters

- **`data`** (`Optional`[`DataFrame`]) – The interaction data. If *None*, create a new request.
- **`kwargs`** – Keyword arguments for *`get()`* if `data = None`.

Returns

The interaction graph.

Return type

`networkx.DiGraph`

omnipath.interactions.PathwayExtra.params

`classmethod PathwayExtra.params()`

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.interactions.PathwayExtra.resources

`classmethod PathwayExtra.resources(**kwargs)`

Return the available resources for this query.

Return type
`Tuple[str]`

2.2.7 omnipath.interactions.PostTranslational

`class omnipath.interactions.PostTranslational(exclude=None)`

Request all post-translational interactions from [OmniPath] .

Imports the dataset which contains post-transcriptional (i.e. protein-protein) interactions. This query requests the interactions from the following datasets:

- *omnipath.constants.InteractionDataset.OMNIPATH*
- *omnipath.constants.InteractionDataset.PATHWAY_EXTRA*
- *omnipath.constants.InteractionDataset.KINASE_EXTRA*
- *omnipath.constants.InteractionDataset.LIGREC_EXTRA*

Parameters
exclude (`Union[str, InteractionDataset, Sequence[str], Sequence[InteractionDataset], None]`) – Post-translational interaction datasets to exclude. If *None*, don't exclude anything.

Methods

get([exclude])	Perform a request to the [OmniPath] web service.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.interactions.PostTranslational.get

classmethod `PostTranslational.get`(*exclude=None*, ***kwargs*)

Perform a request to the [OmniPath] web service.

The available interaction datasets are *omnipath.constants.InteractionDataset*.

Parameters

- **exclude** (Union[str, *InteractionDataset*, Sequence[str], Sequence[*InteractionDataset*], None]) – Post-translational interaction datasets to exclude. If *None*, don't exclude anything.
- **kwargs** – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
pandas.DataFrame

omnipath.interactions.PostTranslational.graph

classmethod `PostTranslational.graph`(*data=None*, ***kwargs*)

Create a graph.

Parameters

- **data** (Optional[DataFrame]) – The interaction data. If *None*, create a new request.
- **kwargs** – Keyword arguments for *get()* if `data = None`.

Returns
The interaction graph.

Return type
networkx.DiGraph

omnipath.interactions.PostTranslational.params

classmethod `PostTranslational.params`()

Return the available values for each parameter, if available.

Return type
Dict[str, Any]

omnipath.interactions.PostTranslational.resources

classmethod `PostTranslational.resources`(***kwargs*)

Return the available resources for this query.

Return type
Tuple[str]

2.2.8 omnipath.interactions.TFmiRNA

`class omnipath.interactions.`**`TFmiRNA`**

Request interactions from the *TF-miRNA* dataset.

Imports the dataset which contains transcription factor-miRNA gene interactions.

Methods

`get`(**kwargs)	Perform a request to the [OmniPath] web service.
`graph`([data])	Create a graph.
`params`()	Return the available values for each parameter, if available.
`resources`(**kwargs)	Return the available resources for this query.

omnipath.interactions.TFmiRNA.get

`classmethod TFmiRNA.`**`get`**(***kwargs*)

Perform a request to the [OmniPath] web service.

Parameters
`kwargs` – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.TFmiRNA.graph

`classmethod TFmiRNA.`**`graph`**(*data=None*, ***kwargs*)

Create a graph.

Parameters

- **`data`** (`Optional`[`DataFrame`]) – The interaction data. If *None*, create a new request.
- **`kwargs`** – Keyword arguments for *`get()`* if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.interactions.TFmiRNA.params

classmethod `TFmiRNA.params()`

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.interactions.TFmiRNA.resources

classmethod `TFmiRNA.resources(**kwargs)`

Return the available resources for this query.

Return type
`Tuple[str]`

2.2.9 omnipath.interactions.TFtarget

class `omnipath.interactions.TFtarget`

Request interactions from the *TF-target* dataset.

Imports the dataset which contains transcription factor-target protein coding gene interactions.

Other TF-target datasets in `omnipath` are *`omnipath.interactions.Dorothea`* and *`omnipath.interactions.TFmiRNA`* which provides TF-miRNA gene interactions.

Methods

get(**kwargs)	Perform a request to the [OmniPath] web service.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.interactions.TFtarget.get

classmethod `TFtarget.get(**kwargs)`

Perform a request to the [OmniPath] web service.

Parameters
kwargs – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.TFtarget.graph

classmethod `TFtarget.`**`graph`**(*data=None*, ***kwargs*)

Create a graph.

Parameters

- **data** (`Optional`[`DataFrame`]) – The interaction data. If *None*, create a new request.
- **kwargs** – Keyword arguments for *`get()`* if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.interactions.TFtarget.params

classmethod `TFtarget.`**`params`**()

Return the available values for each parameter, if available.

Return type
`Dict`[`str`, `Any`]

omnipath.interactions.TFtarget.resources

classmethod `TFtarget.`**`resources`**(***kwargs*)

Return the available resources for this query.

Return type
`Tuple`[`str`]

2.2.10 omnipath.interactions.Transcriptional

class `omnipath.interactions.`**`Transcriptional`**

Request all *TF-target* interactions from [OmniPath].

Imports the dataset which contains transcription factor-target protein coding gene interactions.

Methods

get(**kwargs)	Perform a request to the [OmniPath] web service.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.interactions.Transcriptional.get

`classmethod Transcriptional.get(**kwargs)`

Perform a request to the [OmniPath] web service.

Parameters
`kwargs` – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.Transcriptional.graph

`classmethod Transcriptional.graph(data=None, **kwargs)`

Create a graph.

Parameters

- `data` (`Optional[DataFrame]`) – The interaction data. If *None*, create a new request.
- `kwargs` – Keyword arguments for `get()` if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.interactions.Transcriptional.params

`classmethod Transcriptional.params()`

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.interactions.Transcriptional.resources

`classmethod Transcriptional.resources(**kwargs)`

Return the available resources for this query.

Return type
`Tuple[str]`

2.2.11 omnipath.interactions.lncRNAmRNA

`class omnipath.interactions.`**`lncRNAmRNA`**

Request interactions from the *lncRNA-mRNA* dataset.

Imports the dataset which contains lncRNA-mRNA interactions.

Methods

`get`(**kwargs)	Perform a request to the [OmniPath] web service.
`graph`([data])	Create a graph.
`params`()	Return the available values for each parameter, if available.
`resources`(**kwargs)	Return the available resources for this query.

omnipath.interactions.lncRNAmRNA.get

`classmethod lncRNAmRNA.`**`get`**(***kwargs*)

Perform a request to the [OmniPath] web service.

Parameters
`kwargs` – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.lncRNAmRNA.graph

`classmethod lncRNAmRNA.`**`graph`**(*data=None*, ***kwargs*)

Create a graph.

Parameters

- **`data`** (`Optional`[`DataFrame`]) – The interaction data. If *None*, create a new request.
- **`kwargs`** – Keyword arguments for *`get()`* if `data = None`.

Returns
The interaction graph.

Return type
`networkx.DiGraph`

omnipath.interactions.lncRNAmRNA.params

`classmethod lncRNAmRNA.params()`

Return the available values for each parameter, if available.

Return type
`Dict[str, Any]`

omnipath.interactions.lncRNAmRNA.resources

`classmethod lncRNAmRNA.resources(**kwargs)`

Return the available resources for this query.

Return type
`Tuple[str]`

2.2.12 omnipath.interactions.miRNA

`class omnipath.interactions.miRNA`

Request interactions from the *miRNA-target* dataset.

Imports the dataset. which contains miRNA-mRNA interactions.

Methods

get(**kwargs)	Perform a request to the [OmniPath] web service.
graph([data])	Create a graph.
params()	Return the available values for each parameter, if available.
resources(**kwargs)	Return the available resources for this query.

omnipath.interactions.miRNA.get

`classmethod miRNA.get(**kwargs)`

Perform a request to the [OmniPath] web service.

Parameters
kwargs – Additional query parameters.

Returns
The result which depends the type of the request and the supplied parameters.

Return type
`pandas.DataFrame`

omnipath.interactions.miRNA.graph

classmethod `miRNA.`**`graph`**(*data=None*, ***kwargs*)

Create a graph.

Parameters

- **`data`** (`Optional`[`DataFrame`]) – The interaction data. If *None*, create a new request.
- **`kwargs`** – Keyword arguments for *`get()`* if `data = None`.

Returns

The interaction graph.

Return type

`networkx.DiGraph`

omnipath.interactions.miRNA.params

classmethod `miRNA.`**`params`**()

Return the available values for each parameter, if available.

Return type

`Dict`[`str`, `Any`]

omnipath.interactions.miRNA.resources

classmethod `miRNA.`**`resources`**(***kwargs*)

Return the available resources for this query.

Return type

`Tuple`[`str`]

2.2.13 omnipath.interactions.import_intercell_network

`omnipath.interactions.`**`import_intercell_network`**(*include=(<InteractionDataset.OMNIPATH>, <InteractionDataset.PATHWAY_EXTRA>, <InteractionDataset.KINASE_EXTRA>, <InteractionDataset.LIGREC_EXTRA>), interactions_params=None, transmitter_params=None*, *receiver_params=None*)

Import intercellular network combining intercellular annotations and protein interactions.

First, it imports a network of protein-protein interactions. Then, it retrieves annotations about the proteins intercellular communication roles, once for the transmitter (delivering information from the expressing cell) and second, the receiver (receiving signal and relaying it towards the expressing cell) side.

These 3 queries can be customized by providing parameters which will be passed to *`omnipath.interactions.OmniPath.get()`* for the network and *`omnipath.requests.Intercell()`* for the annotations.

Finally the 3 `pandas.DataFrame` are combined in a way that the source proteins in each interaction annotated by the transmitter, and the target proteins by the receiver categories. If undirected interactions present (these are disabled by default) they will be duplicated, i.e. both partners can be both receiver and transmitter.

Parameters

- **include** (Union[str, *InteractionDataset*, Sequence[str], Sequence[*InteractionDataset*]]) – Interaction datasets to include for *omnipath.interactions.AllInteractions.get()*.
- **interactions_params** (Optional[Mapping[str, Any]]) – Parameters for the *omnipath.interactions.AllInteractions.get()*.
- **transmitter_params** (Optional[Mapping[str, Any]]) – Parameters defining the transmitter side of intercellular connections. See *omnipath.interactions.AllInteractions.params()* for available values.
- **receiver_params** (Optional[Mapping[str, Any]]) – Parameters defining the receiver side of intercellular connections. See *omnipath.interactions.AllInteractions.params()* for available values.

Returns

A dataframe containing information about protein-protein interactions and the inter-cellular roles of the proteins involved in those interactions.

Return type

pandas.DataFrame

2.3 Other

2.3.1 Constants

constants.InteractionDataset(*args, **kw)	Available interaction datasets in [OmniPath].
constants.License(*args, **kw)	License types.
constants.Organism(*args, **kw)	Organism types.

omnipath.constants.InteractionDataset

class omnipath.constants.**InteractionDataset**(**args*, ***kw*)

Available interaction datasets in [OmniPath].

See `omnipath.interactions` for more information.

Attributes

COLLECTRI
DOROTHEA
KINASE_EXTRA
LIGREC_EXTRA
LNCRNA_MRNA
MIRNA_TARGET
OMNIPATH
PATHWAY_EXTRA
SMALL_MOLECULE
TF_MIRNA
TF_REGULONS
TF_TARGET

omnipath.constants.InteractionDataset.COLLECTRI

```
InteractionDataset.COLLECTRI = 'collectri'
```

omnipath.constants.InteractionDataset.DOROTHEA

```
InteractionDataset.DOROTHEA = 'dorothea'
```

omnipath.constants.InteractionDataset.KINASE_EXTRA

```
InteractionDataset.KINASE_EXTRA = 'kinaseextra'
```

omnipath.constants.InteractionDataset.LIGREC_EXTRA

```
InteractionDataset.LIGREC_EXTRA = 'ligrecextra'
```

omnipath.constants.InteractionDataset.LNCRNA_MRNA

```
InteractionDataset.LNCRNA_MRNA = 'lncrna_mrna'
```

omnipath.constants.InteractionDataset.MIRNA_TARGET

```
InteractionDataset.MIRNA_TARGET = 'mirnatarget'
```

omnipath.constants.InteractionDataset.OMNIPATH

```
InteractionDataset.OMNIPATH = 'omnipath'
```

omnipath.constants.InteractionDataset.PATHWAY_EXTRA

```
InteractionDataset.PATHWAY_EXTRA = 'pathwayextra'
```

omnipath.constants.InteractionDataset.SMALL_MOLECULE

```
InteractionDataset.SMALL_MOLECULE = 'small_molecule'
```

omnipath.constants.InteractionDataset.TF_MIRNA

```
InteractionDataset.TF_MIRNA = 'tf_mirna'
```

omnipath.constants.InteractionDataset.TF_REGULONS

```
InteractionDataset.TF_REGULONS = 'tfregulons'
```

omnipath.constants.InteractionDataset.TF_TARGET

```
InteractionDataset.TF_TARGET = 'tf_target'
```

omnipath.constants.License

`class omnipath.constants.License(*args, **kw)`

License types.

Attributes

ACADEMIC	Academic license.
COMMERCIAL	Commercial license.
NON_PROFIT	Non-profit license.
FOR_PROFIT	For-profit license.
IGNORE	Ignore the license type.

omnipath.constants.License.ACADEMIC

`License.ACADEMIC = 'academic'`

Academic license.

omnipath.constants.License.COMMERCIAL

`License.COMMERCIAL = 'commercial'`

Commercial license.

omnipath.constants.License.NON_PROFIT

`License.NON_PROFIT = 'non_profit'`

Non-profit license.

omnipath.constants.License.FOR_PROFIT

`License.FOR_PROFIT = 'for_profit'`

For-profit license.

omnipath.constants.License.IGNORE

`License.IGNORE = 'ignore'`

Ignore the license type.

omnipath.constants.Organism

class omnipath.constants.**Organism**(**args*, ***kw*)

Organism types.

Attributes

HUMAN	NCIB taxonomy id `9606`.
MOUSE	NCIB taxonomy id `10090`.
RAT	NCIB taxonomy id `10116`.
code	Return the code for this organism.

omnipath.constants.Organism.HUMAN

Organism.**HUMAN = 'human'**

NCIB taxonomy id `9606`.

omnipath.constants.Organism.MOUSE

Organism.**MOUSE = 'mouse'**

NCIB taxonomy id `10090`.

omnipath.constants.Organism.RAT

Organism.**RAT = 'rat'**

NCIB taxonomy id `10116`.

omnipath.constants.Organism.code

property Organism.**code:** int

Return the code for this organism.

2.3.2 Options

omnipath.clear_cache()	Remove all cached data from `omnipath.options.cache`.
omnipath.options	Class defining various `omnipath` options.

omnipath.clear_cache

`omnipath.clear_cache()`

Remove all cached data from `omnipath.options.cache`.

Return type

`None`

omnipath.options

`omnipath.options = Options(url='https://omnipathdb.org', license=None, cache=<FileCache[size=0, path='/home/docs/.cache/omnipathdb']>, autoload=True, convert_dtypes=True, num_retries=3, timeout=600.0, chunk_size=8196)`

Class defining various `omnipath` options.

Parameters

- **`url`** – URL of the web service.
- **`license`** – License to use when fetching the data.
- **`password`** – Password used when performing requests.
- **`cache`** – Type of a cache. Valid options are:
  - *None*: do not save anything into a cache.
  - *'memory'*: cache files into the memory.
  - `str`: persist files into a directory.
- **`autoload`** – Whether to contact the server at `url` during import to get the server version and the most up-to-date query parameters and their valid options.
- **`convert_dtypes`** – Whether to convert the data types of the resulting `pandas.DataFrame`.
- **`num_retries`** – Number of retries before giving up.
- **`timeout`** – Timeout in seconds when awaiting response.
- **`chunk_size`** – Size in bytes in which to read the data.
- **`progress_bar`** – Whether to show the progress bar when downloading data.

CHAPTER

THREE

RELEASE NOTES

3.1 Version 1.0

3.1.1 1.0.5 2021-16-08

- Setting `omnipath.options.cache` to `None` will now disable it (use `'memory'` instead)
- Fix writing empty values into cache
- Fix memory cache not copying data before storing it
- Fix various `pandas` warnings
- Remove redundant step from CI

3.1.2 1.0.4 2020-27-12

- Fix recursion error
- Remove duplicated `PostTranslational` class
- Add interactions tests

3.1.3 1.0.3 2020-08-12

- Add *`omnipath.interactions.PostTranslational`*
- Add possibility to download all *`omnipath.requests.Annotations`*

3.1.4 1.0.2 2020-29-11

- Fix small bug when converting boolean values
- Fix typos
- Add option to create interaction graphs

3.1.5 1.0.1 2020-29-11

- Fix bug of not correctly passing datasets in interactions
- Fix the way the progress bar is getting content size
- Add comparison tests with OmnipathR

3.1.6 1.0.0 2020-23-11

- Fix minor bugs
- Add options improvements
- Add tests

CHAPTER

FOUR

REFERENCES

BIBLIOGRAPHY

[OmniPath] Türei, D., Valdeolivas, A. *et al.* (2020), *Integrated intra- and intercellular signaling knowledge for multicellular omics analysis*, bioRxiv 2020.08.03.221242.

[OmniPath16] Türei, D., Korcsmáros, T. & Saez-Rodriguez, J. (2016), *OmniPath: guidelines and gateway for literature-curated signaling pathway resources.*, Nat Methods 13, 966–967.

INDEX