Pathway Analysis Documentation

Release 0.1

KNBiBS

Nov 11, 2018

Contents:

CHAPTER 1

Command Line Interface

1.1 User guide

The command line interface has built in help. To display the help, please append *-h* to the program call, for example:

```
./patapy.py -h
```

The help option responds to arguments your provide, so you can get details about your method of choice with:

```
./patapy.py gsea -h
```

where gsea is the name of a method; likewise, you can display help for any of samples specification options (case/control/data), e.g.:

```
./patapy.py control -h
```

1.2 Predefined parsers

Parsers are defined in `command_line.main` module.

1.3 Creating custom arguments and parsers

Please see declarative_parser module documentation for instructions.

CHAPTER 2

Methods

2.1 Implemented methods

2.2 Adding a new Method

To implement a new method and integrate it with the Command Line Interface provided by this package, please inherit from `Method` class.

CHAPTER 3

Biological Objects

class Gene(*name*, *description=None*)

Stores gene's identifier and description (multiton).

At a time there can be only one gene with given identifier, i.e. after the first initialization, all subsequent attempts to initialize a gene with the same identifier will return exactly the same object. This is so called multiton pattern.

Example

```
>>> x = Gene('TP53')
>>> y = Gene('TP53')
>>> assert x is y   # passes, there is only one gene
```

class Sample(*name*, *data*)

Sample contains expression values for genes.

as_array()

Returns: one-dimensional labeled array with Gene objects as labels

classmethod from_array(*name*, *panda_series*, *descriptions=False*)

Create a sample from pd.Series or equivalent.

Parameters

- **name** – name of the sample
- **panda_series** (`Series`) – series object where columns represent values of genes and names are either gene identifiers of tuples: `(gene_identifier, description)`
- **descriptions** – are descriptions present in names of the series object?

classmethod from_names(*name*, *data*)

Create a sample from a gene_name: value mapping.

Parameters

- **name** – name of sample

- **data** (Mapping[str, float]) – mapping (e.g. dict) where keys represent gene names

class SampleCollection(*name*, *samples=None*)

A collection of samples of common origin or characteristic.

An example sample_collection can be: (Breast_cancer_sample_1, Breast_cancer_sample_2) named "Breast cancer".

The common origin/characteristics for "Breast cancer" sample_collection could be "a breast tumour", though samples had been collected from two donors.

Another example are controls: (Control_sample_1, Control_sample_2) named "Control".

The common characteristic for these samples is that both are controls.

as_array()

Returns: pandas.DataFrame object with data for all samples.

classmethod from_file(*name*, *file_object*, *columns_selector=None*, *samples=None*, *delimiter='t'*, *index_col=0*, *use_header=True*, *reverse_selection=False*, *prefix=None*, *header_line=0*, *description_column=None*)

Create a sample_collection (collection of samples) from csv/tsv file.

Parameters

- **name** – a name of the sample_collection (or group of samples) which will identify it (like "Tumour_1" or "Control_in_20_degrees")
- **file_object** – a file (containing gene expression) of the following structure:
  - names of samples separated by a tab in the first row,
  - gene symbol/name followed by gene expression values for every sample in remaining rows;

  an additional column "description" is allowed between genes column and sample columns, though it has to be explicitly declared with *description_column* argument.
- **columns_selector** (Optional[Callable[[Sequence[int]], Sequence[int]]]) – a function which will select (and return) a subset of provided column identifiers (do not use with *samples*)
- **samples** – a list of names of samples to extract from the file (do not use with *columns_selector*)
- **reverse_selection** – if you want to use all columns but the selected ones (or all samples but the selected) set this to True
- **delimiter** (str) – the delimiter of the columns
- **index_col** (int) – column to use as the gene names
- **use_header** – does the file have a header?
- **prefix** – prefix for custom samples naming schema
- **header_line** – number of non-empty line with sample names None - do not use, 0 - use first row
- **description_column** – is column with description of present in the file (on the second position, after gene identifiers)?

classmethod from_gct_file(*name*, *file_object*, ***kwargs*)

Parse file in Gene Cluster Text file format, as defined on:

software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats User is allowed to provide settings different from the standard.

genes

Return all genes present in the collection of samples.

CHAPTER 4

Statistics Utilities

ttest_ind_phenotype(*case*, *control*, *alternative='two-sided'*)

Two sided t-test of case sample(s) and mean expression values in base samples across all genes :type case: Union[*SampleCollection*, *Sample*] :param case: either Sample of SampleCollection object with case sample(s) :type control: Union[*SampleCollection*, *Sample*] :param control: either Sample of SampleCollection object with control sample(s) :param alternative: string with the alternative hypothesis, H1, has to be one of the following:

> 'two-sided': H1: difference in means not equal to value (default) 'larger' : H1: difference in means larger than value 'smaller' : H1: difference in means smaller than value

Returns: tstat [float or numpy array in case of multiple case samples - test statisic] pvalue : float or numpy array in case of multiple case samples - pvalue of the t-test df : int or float - degrees of freedom used in the t-test

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

s

Index