data_tools Documentation

Release 0.0.3

Nicolas Palacio

Jul 04, 2018

CONTENTS

Collection of Python functions and classes designed to make a Computational Biologist's life easier.

Contact: nicolaspalacio91@gmail.com

CHAPTER

ONE

DEPENDENCIES

In module **data_tools.plots**:

- NumPy
- Matplotlib
- Pandas
- SciPy

In module **data_tools.models**:

- NumPy
- Matplotlib
- Pandas
- Scikit-learn

CHAPTER

TWO

INSTALLATION

First download/clone `data_tools` from the GitHub repository. From the terminal:

```
git clone https://github.com/Nic-Nic/data_tools.git
cd data_tools
```

Then you can install it by running `setup.py` as follows:

```
python setup.py sdist
```

Or using `pip`:

```
pip install .
```

CHAPTER

THREE

REFERENCE

3.1 data_tools.models

Model classes module.

class data_tools.models.**Lasso**(*Cs=500*, *cv=10*, *sampler='skf'*, *solver='liblinear'*, ***kwargs*)

Wrapper class inheriting from `sklearn.linear_model.LogisticRegressionCV` with L1 regularization.

- **Arguments:**
  - *Cs* [int]: Optional, `500` by default. Integer or list of float values of regularization parameters to test. If an integer is passed, it will determine the number of values taken from a logarithmic scale between `1e-4` and `1e4`. Note that the value of the parameter is defined as the inverse of the regularization strength.
  - *cv* [int]: Optional, `10` by default. Denotes the number of cross validation (CV) folds.
  - *sampler* [str]: Optional, `'skf'` by default. Determines which sampling method is used to generate the test and training sets for CV. Methods available are K-Fold (`'kf'`), Shuffle Split (`'ss'`) and their stratified variants (`'skf'` and `'sss'` respectively).
  - *solver* [str]: Optional, `'liblinear'` by default. Determines which solver algorithm to use. Note that L1 regularization can only be handled by `'liblinear'` and `'saga'`. Additionally if the classification is multinomial, only the latter option is available.
  - ***kwargs*: Optional. Any other keyword argument accepted by the `sklearn.linear_model.LogisticRegressionCV` class.

  Other keyword arguments and functions available from the parent class `LogisticRegressionCV` can be fount in Scikit-Learn's reference.

fit_data(*x*, *y*, *silent=False*)

Fits the data to the logistic model.

- **Arguments:**
  - *x* [pandas.DataFrame]: Contains the values/measurements [float] of the features (columns) for each sample/replicate (rows).
  - *y* [pandas.Series]: List or any iterable containing the observed class of each sample (must have the same order as in *x*).
  - *silent* [bool]: Optional, `False` by default. Determines whether messages are printed or not.

plot_coef(*filename=None*, *figsize=None*)

Plots the non-zero coefficients for the fitted predictor features.

- **Arguments:**

- *filename* [str]: Optional, `None` by default. If passed, indicates the file name or path where to store the figure. Format must be specified (e.g.: .png, .pdf, etc)
- *figsize* [tuple]: Optional, `None` by default (default matplotlib size). Any iterable containing two values denoting the figure size (in inches) as [width, height].

- **Returns:**
  - [matplotlib.figure.Figure]: Figure object containing the bar plot of the non-zero coefficients.

plot_score(*filename=None*, *figsize=None*)

Plots the mean score across all folds obtained during CV. The optimum C parameter chosen and its score are highlighted.

- **Arguments:**
  - *filename* [str]: Optional, `None` by default. If passed, indicates the file name or path where to store the figure. Format must be specified (e.g.: .png, .pdf, etc)
  - *figsize* [tuple]: Optional, `None` by default (default matplotlib size). Any iterable containing two values denoting the figure size (in inches) as [width, height].
- **Returns:**
  - [matplotlib.figure.Figure]: Figure object containing the score plot.

3.2 data_tools.plots

Plotting functions module.

data_tools.plots.**density**(*df*, *cvf=0.25*, *title=None*, *filename=None*, *figsize=None*)

Generates a density plot of the values on a data frame (row-wise).

- **Arguments:**
  - *df* [pandas.DataFrame]: Contains the values to generate the plot. Each row is considered as an individual sample while each column contains a measured value.
  - *cvf* [float]: Optional, `0.25` by default. Co-variance factor used in the gaussian kernel estimation. A higher value increases the smoothness.
  - *title* [str]: Optional, `None` by default. Defines the plot title.
  - *filename* [str]: Optional, `None` by default. If passed, indicates the file name or path where to store the figure. Format must be specified (e.g.: .png, .pdf, etc)
  - *figsize* [tuple]: Optional, `None` by default (default matplotlib size). Any iterable containing two values denoting the figure size (in inches) as [width, height].
- **Returns:**
  - [*matplotlib.figure.Figure*]: the figure object containing the density plot.

data_tools.plots.**piano_consensus**(*df*, *nchar=40*, *boxes=True*, *title=None*, *filename=None*, *figsize=None*)

Generates a GSEA consensus score plot like R package `piano`'s `consensusScores` function, but prettier. The main input is assumed to be a `pandas.DataFrame` whose data is the same as the `rankMat` from the result of `consensusScores`.

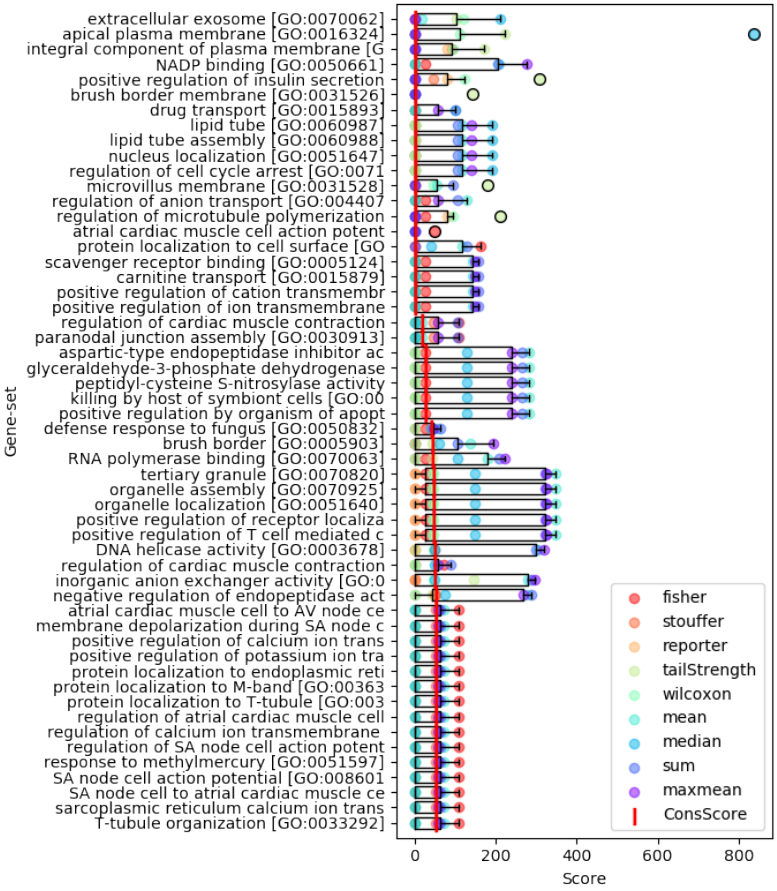
- **Arguments:**

- *df* [pandas.DataFrame]: Values contained correspond to the scores of the gene-sets (consensus and each individual methods). Index must contain the gene-set labels. Columns are assumed to be `ConsRank` (ignored), `ConsScore` followed by the individual methods (e.g.: `mean`, `median`, `sum`, etc).
- *nchar* [int]: Optional, `40` by default. Number of string characters of the gene-set labels of the plot.
- *boxes* [bool]: Optional, `True` by default. Determines whether to show the boxplots of the gene-sets or not.
- *title* [str]: Optional, `None` by default. Defines the plot title.
- *filename* [str]: Optional, `None` by default. If passed, indicates the file name or path where to store the figure. Format must be specified (e.g.: .png, .pdf, etc)
- *figsize* [tuple]: Optional, `None` by default (default matplotlib size). Any iterable containing two values denoting the figure size (in inches) as [width, height].

- **Returns:**
  - [*matplotlib.figure.Figure*]: the figure object containing a combination of box and scatter plots of the gene-set scores.

- **Examples:**

```
>>> piano_consensus(df, figsize=[7, 8])
```

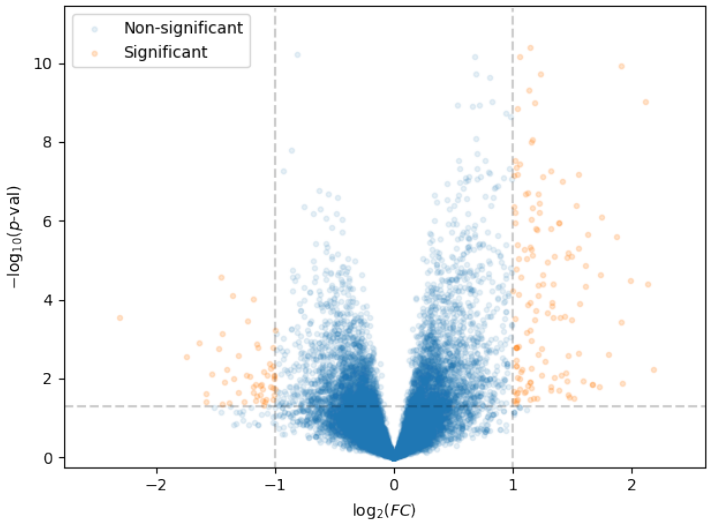


data_tools.plots.**volcano**(*logfc*, *logpval*, *thr_pval=0.05*, *thr_fc=2.0*, *c=('C0', 'C1')*, *legend=True*, *title=None*, *filename=None*, *figsize=None*)

Generates a volcano plot from the differential expression data provided.

- **Arguments:**
  - *logfc* [list]: Or any iterable type. Contains the log (usually base 2) fold-change values. Must have the same length as *logpval*.
  - *logpval* [list]: Or any iterable type. Contains the -log p-values (usually base 10). Must have the same length as *logfc*.
  - *thr_pval* [float]: Optional, `0.05` by default. Specifies the p-value (non log-transformed) threshold to consider a measurement as significantly differentially expressed.
  - *thr_fc* [float]: Optional, `2.` by default. Specifies the FC (non log-transformed) threshold to consider a measurement as significantly differentially expressed.
  - *c* [tuple]: Optional, `('C0', 'C1')` by default (matplotlib default colors). Any iterable containing two color arguments tolerated by matplotlib (e.g.: `['r', 'b']` for red and blue). First one is used for non-significant points, second for the significant ones.
  - *legend* [bool]: Optional, `True` by default. Indicates whether to show the plot legend or not.
  - *title* [str]: Optional, `None` by default. Defines the plot title.
  - *filename* [str]: Optional, `None` by default. If passed, indicates the file name or path where to store the figure. Format must be specified (e.g.: .png, .pdf, etc)
  - *figsize* [tuple]: Optional, `None` by default (default matplotlib size). Any iterable containing two values denoting the figure size (in inches) as [width, height].
- **Returns:**
  - [matplotlib.figure.Figure]: Figure object containing the volcano plot.
- **Examples:**

```
>>> volcano(my_log_fc, my_log_pval)
```



3.3 data_tools.sets

Set operations module.

`data_tools.sets.`**`bit_or`**(*a*, *b*)

Returns the bit operation OR between two bit-strings *a* and *b*. NOTE: *a* and *b* must have the same size.

- **Arguments:**
  - *a* [tuple]: Or any iterable type.
  - *b* [tuple]: Or any iterable type.
- **Returns:**
  - [tuple]: OR operation between *a* and *b* element-wise.
- **Examples:**

```
>>> a, b = (0, 0, 1), (1, 0, 1)
>>> bit_or(a, b)
(1, 0, 1)
```

`data_tools.sets.`**`find_min`**(*A*)

Finds and returns the subset of vectors whose sum is minimum from a given set *A*.

- **Arguments:**
  - *A* [set]: Set of vectors ([tuple] or any iterable).
- **Returns:**
  - [set]: Subset of vectors in *A* whose sum is minimum.
- **Examples:**

```
>>> A = {(0, 1, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)}
>>> find_min(A)
set([(0, 1, 0), (1, 0, 0)])
```

`data_tools.sets.`**`in_all`**(*x*, *N*)

Checks if a vector *x* is present in all sets contained in a list *N*.

- **Arguments:**
  - *x* [tuple]: Or any hashable type as long as is the same contained in the sets of *N*.
  - *N* [list]: Or any iterable type containing [set] objects.
- **Returns:**
  - [bool]: `True` if *x* is found in all sets of *N*, `False` otherwise.
- **Examples:**

```
>>> N = [{(0, 0), (0, 1)}, # <- set A
...      {(0, 0), (1, 1), (1, 0)}] # <- set B
>>> x = (0, 0)
>>> in_all(x, N)
True
>>> y = (0, 1)
>>> in_all(y, N)
False
```

data_tools.sets.**subsets**(*N*)

Function that computes all possible logical relations between all sets on a list *N* and returns all subsets. This is, the subsets that would represent each intersecting area on a Venn diagram.

- **Arguments:**
  - *N* [list]: Or any iterable type containing [set] objects.
- **Returns:**
  - [dict]: Collection of subsets according to the logical relations between the sets in *N*. The keys are binary codes that denote the logical relation (see examples below).
- **Examples:**

```
>>> N = [{0, 1, 2}, {2, 3, 4}]
>>> subsets(N)
{'11': set([2]), '10': set([0, 1]), '01': set([3, 4])}
>>> N = [{0, 1}, {2, 3}, {1, 3, 4}]
>>> subsets(N)
{'010': set([2]), '011': set([3]), '001': set([4]), '111': set([
]), '110': set([]), '100': set([0]), '101': set([1])}
```

3.4 data_tools.strings

String operations module.

data_tools.strings.**is_numeric**(*s*)

Determines if a string can be considered a numeric value. NaN is also considered, since it is float type.

- **Arguments:**
  - *s* [str]: String to be evaluated.
- **Returns:**
  - [bool]: `True`/`False` depending if the condition is satisfied.
- **Examples:**

```
>>> is_numeric('4')
True
>>> is_numeric('-3.2')
True
>>> is_numeric('number')
False
>>> is_numeric('NaN')
True
```

data_tools.strings.**join_str_lists**(*a*, *b*, *sep=''*)

Joins element-wise two lists (or any 1D iterable) of strings with a given separator (if provided). Length of the input lists must be equal.

- **Arguments:**
  - *a* [list]: Contains the first elements [str] of the joint strings.
  - *b* [list]: Contains the second elements [str] of the joint strings.
  - *sep* [str]: Optional `''` (non separated) by default. Determines the separator between the joint strings.

- **Returns:**
  - [list]: List of the joint strings.
- **Example:**

```
>>> a = ['a', 'b']
>>> b = ['1', '2']
>>> join_str_lists(a, b, sep='_')
['a_1', 'b_2']
```

PYTHON MODULE INDEX

d

INDEX