data_tools Documentation

Release 0.0.2

Nicolas Palacio

May 28, 2018

CONTENTS

1 Dependencies **3**

2 Installation **5**

3 Reference **7**

3.1 data_tools.Lasso 7

3.2 data_tools.plots 9

3.3 data_tools.sets 11

3.4 data_tools.strings 13

Python Module Index **15**

Index **17**

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

CHAPTER ONE

DEPENDENCIES

In module **data_tools.plots**:

- NumPy
- Matplotlib
- Pandas

In module **data_tools.Lasso**:

- 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.Lasso

Class for logistic regression models with L1 regularization.

class data_tools.Lasso.**Lasso**(*Cs=array([0.01, 0.01009272, 0.01018629, 0.01028073, 0.01037605, 0.01047225, 0.01056935, 0.01066734, 0.01076624, 0.01086606, 0.01096681, 0.01106848, 0.01117111, 0.01127468, 0.01137921, 0.01148472, 0.0115912, 0.01169866, 0.01180713, 0.0119166, 0.01202708, 0.01213859, 0.01225114, 0.01236472, 0.01247936, 0.01259506, 0.01271184, 0.0128297, 0.01294865, 0.0130687, 0.01318987, 0.01331216, 0.01343558, 0.01356015, 0.01368587, 0.01381276, 0.01394083, 0.01407008, 0.01420053, 0.01433219, 0.01446507, 0.01459919, 0.01473454, 0.01487115, 0.01500903, 0.01514819, 0.01528864, 0.01543038, 0.01557345, 0.01571784, 0.01586357, 0.01601064, 0.01615909, 0.01630891, 0.01646012, 0.01661273, 0.01676675, 0.0169222, 0.0170791, 0.01723745, 0.01739726, 0.01755856, 0.01772136, 0.01788566, 0.01805149, 0.01821885, 0.01838777, 0.01855825, 0.01873032, 0.01890397, 0.01907924, 0.01925614, 0.01943467, 0.01961486, 0.01979672, 0.01998026, 0.02016551, 0.02035248, 0.02054117, 0.02073162, 0.02092383, 0.02111783, 0.02131362, 0.02151123, 0.02171068, 0.02191197, 0.02211512, 0.02232016, 0.02252711, 0.02273597, 0.02294676, 0.02315951, 0.02337424, 0.02359095, 0.02380968, 0.02403043, 0.02425323, 0.02447809, 0.02470504, 0.02493409, 0.02516527, 0.02539859, 0.02563407, 0.02587174, 0.02611161, 0.02635371, 0.02659804, 0.02684465, 0.02709354, 0.02734474, 0.02759826, 0.02785414, 0.02811239, 0.02837304, 0.0286361, 0.0289016, 0.02916956, 0.02944, 0.02971296, 0.02998844, 0.03026648, 0.0305471, 0.03083031, 0.03111616, 0.03140465, 0.03169582, 0.03198969, 0.03228628, 0.03258562, 0.03288774, 0.03319266, 0.03350041, 0.03381101, 0.03412449, 0.03444087, 0.03476019, 0.03508247, 0.03540774, 0.03573602, 0.03606735, 0.03640175, 0.03673925, 0.03707988, 0.03742366, 0.03777064, 0.03812083, 0.03847427, 0.03883098, 0.039191, 0.03955436, 0.03992109, 0.04029122, 0.04066478, 0.04104181, 0.04142232, 0.04180637, 0.04219398, 0.04258518, 0.04298001, 0.0433785, 0.04378069, 0.0441866, 0.04459628, 0.04500975, 0.04542706, 0.04584824, 0.04627332, 0.04670234, 0.04713535, 0.04757236, 0.04801343, 0.04845859, 0.04890787, 0.04936132, 0.04981898, 0.05028087, 0.05074705, 0.05121755, 0.05169242, 0.05217169, 0.0526554, 0.05314359, 0.05363631, 0.0541336, 0.0546355, 0.05514206, 0.05565331, 0.0561693, 0.05669007, 0.05721568, 0.05774615, 0.05828155, 0.0588219, 0.05936727, 0.0599177, 0.06047322, 0.0610339, 0.06159978, 0.0621709, 0.06274732, 0.06332908, 0.06391624, 0.06450884, 0.06510694, 0.06571058, 0.06631981, 0.0669347, 0.06755528, 0.06818162, 0.06881377, 0.06945178, 0.0700957, 0.07074559, 0.07140151, 0.07206351, 0.07273165, 0.07340598, 0.07408657, 0.07477346, 0.07546673, 0.07616642, 0.0768726, 0.07758532, 0.07830465, 0.07903066, 0.07976339, 0.08050292, 0.0812493, 0.08200261, 0.08276289, 0.08353023, 0.08430468, 0.08508632, 0.08587519, 0.08667139, 0.08747496, 0.08828599, 0.08910453, 0.08993067, 0.09076446, 0.09160598, 0.09245531, 0.09331251, 0.09417766, 0.09505083, 0.09593209, 0.09682153, 0.09771921, 0.09862522, 0.09953962, 0.10046251, 0.10139395, 0.10233402, 0.10328281, 0.1042404, 0.10520687, 0.10618229, 0.10716676, 0.10816036, 0.10916317, 0.11017528, 0.11119677, 0.11222774, 0.11326826, 0.11431843, 0.11537833, 0.11644806, 0.11752771, 0.11861737, 0.11971713, 0.12082709, 0.12194734, 0.12307798, 0.1242191, 0.1253708, 0.12653318, 0.12770633, 0.12889036, 0.13008537, 0.13129146, 0.13250873, 0.13373729, 0.13497723, 0.13622868, 0.13749172, 0.13876648, 0.14005305, 0.14135156, 0.1426621, 0.1439848, 0.14531975, 0.14666709, 0.14802691, 0.14939935, 0.1507845, 0.15218251, 0.15359347, 0.15501751, 0.15645476,* Statistical model developer. Uses LASSO Logistic regression with cross validation (CV) for penalty parameter "C". Inherits from sklearn.linear_model.LogisticRegressionCV.

compute_second_line()
: Method that computes second-line predictors. This is, initially selected predictors' data is dropped out, and another model is trained with the remaining data.

fit_data(*x*, *y*, *silent=False*)
: Fits the passed data, assumed to follow pandas data structures [pandas.DataFrame] or [pandas.Series]. Where x is the normalized MS intensity [float] and y the disease status category [int].

plot_coef()
: Plots the non-zero coefficients for the fitted predictors.

plot_sample_probs(*x=None*, *y=None*)

plot_score()
: Plots the mean score across all folds obtained during CV. Highlights the optimum C parameter chosen (NOTE: Since C represents the inverse regularization parameter, it's chosen so that maximizes the score).

3.2 data_tools.plots

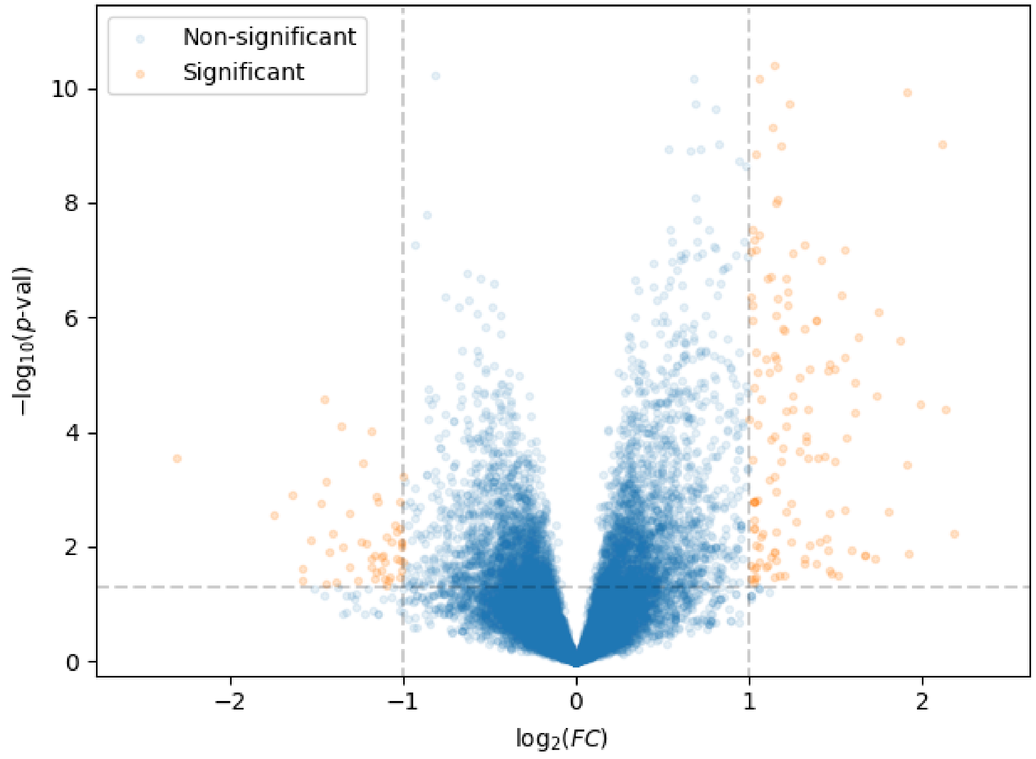
Plotting functions module.

`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)
```



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

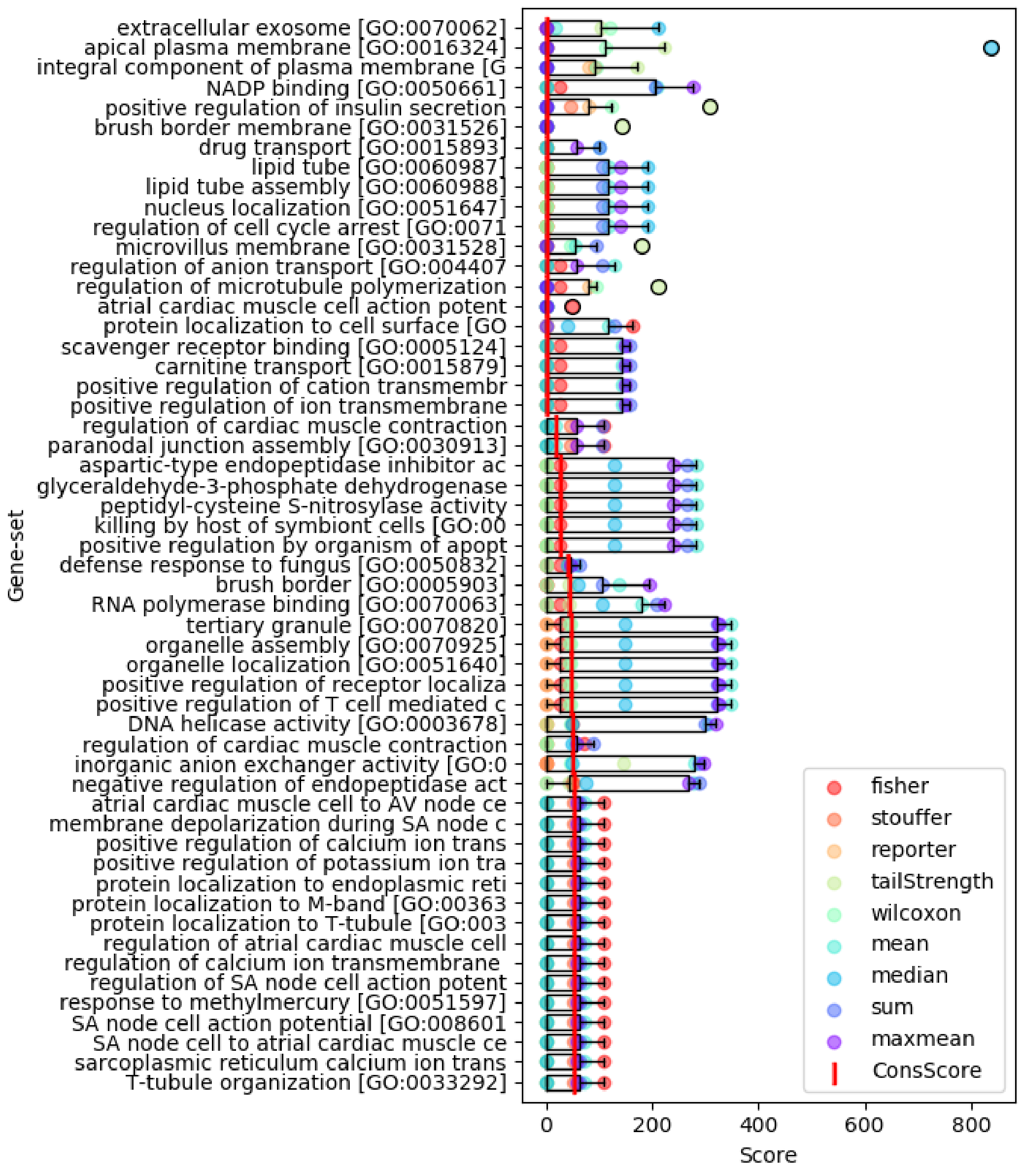
Generates a GSEA consensus score plot like R package `piano`'s `consensusScores` function, but prettier. The main input is assumed to be a `pandas.DataFrame` whose data is the same as the `rankMat` from the result of `consensusScores`.

- **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])
```



3.3 data_tools.sets

Set operations module.

`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.`**`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.`**`multi_union`**(*N*)

Returns the union set of all sets contained in a list *N*.

- **Arguments:**
    - *N* [list]: Or any iterable type containing [set] objects.
- **Returns:**
    - [set]: The union of all sets contained in *N*.
- **Examples:**

```
>>> A = {1, 3, 5}
>>> B = {0, 1, 2}
>>> C = {0, 2, 5}
>>> multi_union([A, B, C])
set([0, 1, 2, 3, 5])
```

`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)])
```

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

data_tools.Lasso, 7
data_tools.plots, 9
data_tools.sets, 11
data_tools.strings, 13

INDEX

B

bit_or() (in module data_tools.sets), 12

C

compute_second_line() (data_tools.Lasso.Lasso method), 9

D

data_tools.Lasso (module), 7
data_tools.plots (module), 9
data_tools.sets (module), 11
data_tools.strings (module), 13

F

find_min() (in module data_tools.sets), 12
fit_data() (data_tools.Lasso.Lasso method), 9

I

in_all() (in module data_tools.sets), 11
is_numeric() (in module data_tools.strings), 13

J

join_str_lists() (in module data_tools.strings), 13

L

Lasso (class in data_tools.Lasso), 7

M

multi_union() (in module data_tools.sets), 12

P

piano_consensus() (in module data_tools.plots), 10
plot_coef() (data_tools.Lasso.Lasso method), 9
plot_sample_probs() (data_tools.Lasso.Lasso method), 9
plot_score() (data_tools.Lasso.Lasso method), 9

V

volcano() (in module data_tools.plots), 9