XL-mHG Documentation

Release 2.4.9

Florian Wagner

Mar 04, 2017

Contents

This is the official documentation for the xlmhg Python package.

CHAPTER 1

Contents

Statistical Background

The XL-mHG test is a powerful semiparametric test to assess *enrichment* in ranked lists. It is based on the nonparametric **mHG test**, developed by Dr. Zohar Yakhini and colleagues (Eden et al., 2007), who also proposed a dynamic programming algorithm that enables the efficient calculation of exact p-values for this test.

The input to the test is a ranked list of items, some of which are known to have some "interesting property". The test asks whether there exists an unusual accumulation of a subset of those "interesting items" at the "top of the list", without requiring the user to specify what part of the list should be considered "the top". Computationally, the ranked list can be represented as a column vector containing only 0's and 1's, with 1's representing the interesting items. For example, the following list of 20 items exhibits an accumulation of 1's "at the top" that is considered statistically significant ($p < 0.05$) by the mHG test:

$v = (1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)^T$

To better understand how enrichment is defined for the purposes of the mHG test, it is helpful to take a close look at the definition of its test statistic: For a given ranked list of length `N`, it is defined as the *minimum hypergeometric p-value* over all N possible cutoffs. This means that users do not have to specify a fixed cutoff that defines "the top of the list". This nonparametric approach makes the mHG test very flexible, meaning that it can detect enrichment when there are only a few "interesting" items that are extremely concentrated at the very top of the list (representing one extreme), as well as when there is a slight overabundance of interesting items within, say, the entire top half of the list.

However, for some applications, the mHG test is a little "too flexible", meaning that it would be beneficial to be able to somewhat restrict the type of enrichment that is being detected by the test. To this end, the **XL-mHG test** extends the mHG test, by introducing two parameters (`X` and `L`) that essentially allow certain cutoffs to be ignored in the calculation of the test statistic. The xlmhg package implements a dynamic programming algorithm to efficiently calculate XL-mHG p-values. This algorithm is based on the algorithm proposed by Eden et al., but has been modified to calculate exact p-values for the new test statistic, (Wagner, 2015), and improved to provide better numerical accuracy and performance (Wagner, 2016).

In biology, specifically in *GO enrichment analysis*, there are many situations in which the "best" cutoff is not known *a priori*. In those cases, the mHG and XL-mHG tests are excellent choices for detecting enrichment, and have been successfully applied for detecting GO enrichment in both supervised and unsupervised settings (Eden, Navon, et al., 2007; Wagner, 2015).

What do the `X` and `L` parameters mean?

- `X` refers to the minimum number of "1's" that have to be seen before anything can be called "enrichment".
- `L` is the lowest cutoff (i.e., the largest `n`) that is being tested for enrichment.

A more direct way to understand `X` and `L` is through the definition of the XL-mHG test statistic. It is defined as the minimum hypergeometric p-value over all cutoffs at which at least `X` "1's" have already been seen, and excluding any cutoffs larger than `L`. For `X=1` and `L=N`, the XL-mHG test reduces to the mHG test.

Further reading

For detailed discussions of the XL-mHG test and the algorithms implemented in the xlmhg package to efficiently calculate XL-mHG test statistics and p-values, please see the Technical Report on arXiv (Wagner, 2015), as well as the XL-mHG PeerJ Preprint article (Wagner, 2016).

Installation

Installing the xlmhg package should be straightforward on Linux, Windows, and Mac OS X. It only requires Python 2.7.x or 3.5.x to be installed. If you have a different version of Python, you have to *install the package from source*.

Note: To see which version of Python you're running, you can always run `python -V` in a terminal / command prompt window. Alternatively, you can run the following in Python:

```
import sys
print(sys.version)
```

The XL-mHG software is hosted on PyPI, the central repository for Python packages. The recommended installation procedure on Linux, Windows, and Mac OS X is using pip, the main tool for installing Python packages.

Installing the latest version

To install the latest version of the xlmhg Python package, run:

```
pip install xlmhg
```

Installing a specific version

To install a specific version of the xlmhg Python package, e.g., "2.3.1", run:

```
pip install xlmhg==2.3.1
```

Specifying a version range

XL-mHG follows semantic versioning, so changes in the major release number (e.g., 1.x.x vs. 2.x.x) indicate a backwards-incompatible API change. To install the latest version of a specific major release number (e.g., "2.x.x"), run:

```
pip install "xlmhg>=2,<3"
```

Installation from source

This installation method is only required for Python versions other than 2.7 or 3.5. The installation command is the same (`pip install xlmhg`), but the installation involves the compilation of C source code using a C compiler. The procedure for installing an appropriate compiler is different for different operating systems.

Ubuntu Linux

Do the following to install the gcc compiler (requires root privileges):

```
$ apt-get install build-essential
```

Windows

For Python 2.6-3.2, use the Microsoft Visual Studio 2008 compiler (32-bit / 64-bit). For Python 3.3 and 3.4, use the Visual Studio 2010 compiler (32-bit / 64-bit).

Examples

The following examples illustrate how to conduct XL-mHG tests and visualize the results using the Python API. For details on each method, including all optional parameters, see the *API reference*.

Conducting a test using the simple test function

This example demonstrates the use of the simple test function, *`xlmhg_test()`*, for conducting an XL-mHG test.

Script:

```
import numpy as np
import xlmhg

v = np.uint8([1,0,1,1,0,1] + [0]*12 + [1,0])
X = 3
L = 10
stat, cutoff, pval = xlmhg.xlmhg_test(v, X=X, L=L)

print('Test statistic: %.3f' % stat)
print('Cutoff: %d' % cutoff)
print('P-value: %.3f' % pval)
```

Output:

```
Test statistic: 0.014
Cutoff: 6
P-value: 0.024
```

Conducting a test using the advanced test function

This example demonstrates the use of the advanced test function, *get_xlmhg_test_result()*, for conducting an XL-mHG test.

Script:

```
import numpy as np
import xlmhg

v = np.uint8([1,0,1,1,0,1] + [0]*12 + [1,0])
X = 3
L = 10

N = v.size
indices = np.uint16(np.nonzero(v)[0])

result = xlmhg.get_xlmhg_test_result(N, indices, X=X, L=L)

print('Result:', str(result))
print('Test statistic: %.3f' % result.stat)
print('Cutoff: %d' % result.cutoff)
print('P-value: %.3f' % result.pval)
```

Output:

```
Result: <mHGResult object (N=20, K=5, X=3, L=10, pval=2.4e-02)>
Test statistic: 0.014
Cutoff: 6
P-value: 0.024
```

Visualizing a test result

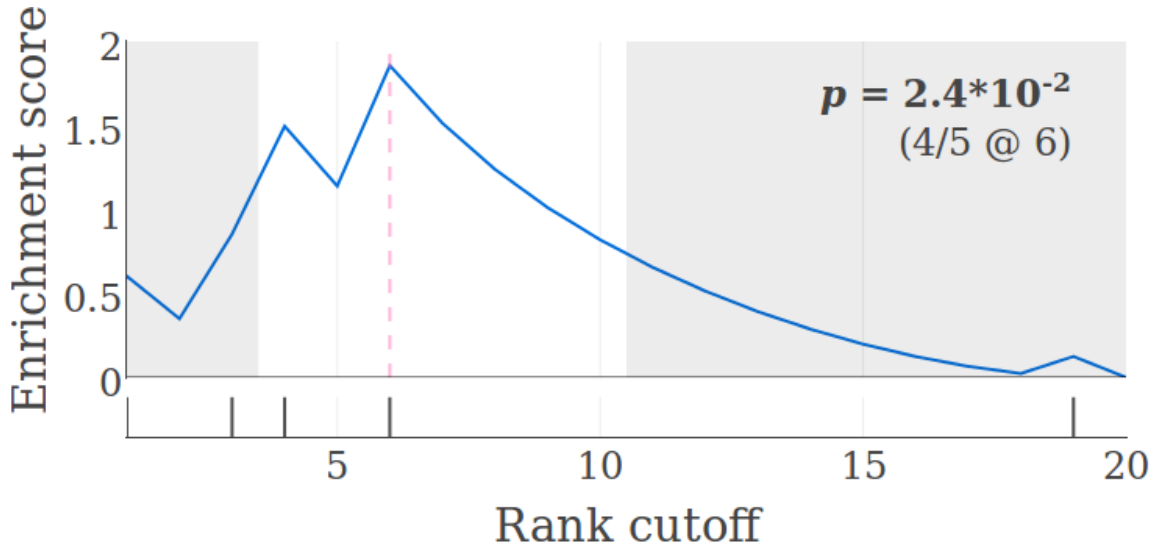
This example demonstrates how to visualize an XL-mHG test result using the *get_result_figure()* function and plotly.

Script:

```
import numpy as np
import xlmhg
from plotly.offline import plot

v = np.uint8([1,0,1,1,0,1] + [0]*12 + [1,0])
X = 3
L = 10

N = v.size
indices = np.uint16(np.nonzero(v)[0])

result = xlmhg.get_xlmhg_test_result(N, indices, X=X, L=L)

fig = xlmhg.get_result_figure(result)

plot(fig, filename='test_figure.html')
```

This produces an html file (`test_figure.html`) that contains an interactive figure. Open the file in a browser (if it doesn't open automatically) and click on the camera symbol (the left-most symbol on top of the figure) to download

it as a PNG image. The image looks as follows:



API Reference

The xlmhg Python API includes two alternative functions to **conduct an XL-mHG test**:

- The *simple test function*, `xlmhg_test()`, accepts a ranked list in the form of a vector, and (optionally) the `X` and `L` parameters, and returns a 3-tuple containing the test statistic, cutoff, and p-value.
- The *advanced test function*, `get_xlmhg_test_result()`, accepts a more compact representation of a list (consisting of its length `N` and a vector specifying the `indices` of the 1's in the ranked list), as well as several additional arguments that can improve the performance of the test. Instead of a simple tuple, this API returns the test result as an `mHGResult` object, which includes additional information such as the test parameters, and methods to calculate additional quantities like E-Scores.

Additionally, the API includes a function, `get_result_figure()`, for **visualizing a test result** in a Plotly figure. See *Examples* for concrete examples of how to use these functions.

Simple test function - `xlmhg_test()`

`xlmhg.`**`xlmhg_test`**(*v*, *X=None*, *L=None*, *table=None*)

Perform an XL-mHG test (simplified interface).

This function accepts a vector containing zeros and ones, and returns a 3-tuple with the XL-mHG test statistic, cutoff, and p-value.

Parameters

- **v** (1-dim `numpy.ndarray` of integers) – The ranked list. All non-zero elements are considered "1"s. (Let N denote the length of the list.)
- **X** (*int, optional*) – The `X` parameter. [1]
- **L** (*int, optional*) – The `L` parameter. [N]

- **table** (np.ndarray with `ndim=2` and `dtype=numpy.longdouble`, optional) – The dynamic programming table. Size has to be at least (K+1) x (W+1), with W = N-K. Providing this array avoids memory reallocation when conducting multiple tests. [None]

Returns

- **stat** (*float*) – The XL-mHG test statistic.
- **cutoff** (*int*) – The (first) cutoff at which stat was attained. (0 if no cutoff was tested.)
- **pval** (*float*) – The XL-mHG p-value (either exact or an upper bound).

Advanced test function - `get_xlmhg_test_result()`

`xlmhg.`**`get_xlmhg_test_result`**(*N*, *indices*, *X=None*, *L=None*, *exact_pval='always'*, *pval_thresh=None*, *escore_pval_thresh=None*, *table=None*, *use_alg1=False*, *tol=1e-12*)

Perform an XL-mHG test.

This function accepts a list in the form of a numpy `indices` array containing the indices of the non-zero elements (sorted), along with the length `N` of the list. It returns an *`mHGResult`* object.

Parameters

- **int** (`N,`) – The length of the list.
- **indices** (1-dim `numpy.ndarray` with `dtype` = numpy.uint16) – Sorted list of indices corresponding to the "1"s in the ranked list.
- **X** (*`int, optional`*) – The `X` parameter. Should be between 0 and K (inclusive), where K is the length of `indices`. [0]
- **L** (*`int, optional`*) – The `L` parameter. Should be between 0 and `N` (inclusive). If `None`, this parameter will be set to `N` [None]
- **exact_pval** (*`str, enumerated`*) – Valid values are: 'always', 'if_significant', and 'if_necessary'. Determines in which cases exact p-values should be calculated. This option helps users avoid the time-consuming calculation of an exact p-value in cases where they do not require it, which can lead to significant performance gains. ['always']

  Specifically, this setting (in conjunction with `pval_thresh`) determines in which cases the PVAL-THRESH algorithm is invoked to efficiently determine whether the test is significant. This algorithm first tries to make this determination by calculating O(1)- and O(N)-bounds of the XL-mHG p-value. Only if this fails to give a conclusive answer, an O(N^2)-algorithm is used to calculate the exact p-value.

  Note that whenever 'if_necessary' or 'if_significant' is specified, a significance level (p-value threshold; argument `pval_thresh`) must be specified as well.
- **pval_thresh** (*`float, optional`*) – The significance threshold, i.e., the p-value below which the test should be considered statistically significant. Note that this argument must be given whenever the `escore_pval_thresh` argument is given. [None]
- **escore_pval_thresh** (*`float, optional`*) – The significance threshold to be used in the calculation of an E-score. The E-score is a measure of the strength of enrichment that is similar to "fold enrichment". [None]
- **table** (`numpy.ndarray` with `ndim=2` and `dtype=numpy.longdouble`, optional) – The dynamic programming table. Size has to be at least (K+1) x (W+1). Providing this array avoids memory reallocation when conducting multiple tests. [None]

- **use_alg1** (*bool, optional*) – Whether to use PVAL1 (instead of PVAL2) for calculating the p-value. [False]
- **tol** (*float, optional*) – The tolerance used for comparing floats. [1e-12]

Returns The test result.

Return type *mHGResult*

Test result objects - `mHGResult`

class `xlmhg.`**`mHGResult`**(*N*, *indices*, *X*, *L*, *stat*, *cutoff*, *pval*, *pval_thresh=None*, *escore_pval_thresh=None*, *escore_tol=None*)

The result of an XL-mHG test.

This class is used by the *get_xlmhg_test_result* function to represent the result of an XL-mHG test.

Parameters

- **N** (*int*) – See *N* attribute.
- **indices** – See *indices* attribute.
- **X** (*int*) – See *X* attribute.
- **L** (*int*) – See :attr:'L' attribute.
- **stat** (*float*) – See *stat* attribute.
- **cutoff** (*int*) – See *cutoff* attribute.
- **pval** (*float*) – See *pval* attribute.
- **pval_thresh** (*float, optional*) – See *pval_thresh* attribute.
- **escore_pval_thresh** (*float, optional*) – See *escore_pval_thresh* attribute.
- **escore_tol** (*float, optional*) – See *escore_tol* attribute.

N
int – The length of the ranked list (i.e., the number of elements in it).

indices
`numpy.ndarray` with `ndim=1` and `dtype=np.uint16`. – A sorted (!) list of indices of all the 1's in the ranked list.

X
int – The XL-mHG X parameter.

L
int – The XL-mHG L parameter.

stat
float – The XL-mHG test statistic.

cutoff
int – The XL-mHG cutoff.

pval
float – The XL-mHG p-value.

pval_thresh
float or None – The user-specified significance (p-value) threshold for this test.

escore_pval_thresh
float or None – The user-specified p-value threshold used in the E-score calculation.

escore_tol
float or None – The floating point tolerance used in the E-score calculation.

K
(property) Returns the number of 1's in the list.

escore
(property) Returns the E-score associated with the result.

fold_enrichment
(property) Returns the fold enrichment at the XL-mHG cutoff.

hash
(property) Returns a unique hash value for the result.

k
(property) Returns the number of 1's above the XL-mHG cutoff.

v
(property) Returns the list as a `numpy.ndarray` (with dtype `np.uint8`).

Visualizing test results - `get_result_figure()`

`xlmhg.`**`get_result_figure`**(*result*, *show_title=False*, *title=None*, *show_inset=True*, *plot_fold_enrichment=False*, *width=800*, *height=350*, *font_size=24*, *margin=None*, *font_family='Computer Modern Roman, serif'*, *score_color='rgb(0, 109, 219)'*, *enrichment_color='rgb(219, 109, 0)'*, *cutoff_color='rgba(255, 52, 52, 0.7)'*, *line_width=2.0*, *ymax=None*, *mHG_label=False*)

Visualize an XL-mHG test result.

Parameters

- **result** (*mHGResult*) – The test result.
- **show_title** (*bool, optional*) – Whether to include a title in the figure. If `title` is not `None`, this parameter is ignored. [False]
- **title** (*str or None, optional*) – Figure title. If not `None`, `show_title` is ignored. [None]
- **show_inset** (*bool, optional*) – Whether to show test parameters and p-value as an inset. [True]
- **plot_fold_enrichment** (*bool, optional*) – Whether to plot the fold enrichment on a second axis. [False]
- **width** (*int, optional*) – The width of the figure (in pixels). [800]
- **height** (*int, optional*) – The height of the figure (in pixels). [350]
- **font_size** (*int, optional*) – The font size to use. [20]
- **margin** (*dict, optional*) – A dictionary specifying the figure margins (in pixels). Valid keys are "l" (left), "r" (right), "t" (top), and "b" (bottom). Missing keys are replaced by Plotly default values. If `None`, will be set to a dictionary specifying a left margin of 100 px, and a top margin of 40 px. [None]

- **font_family** (*str, optional*) – The font family (name) to use. ["Computer Modern Roman, serif"]
- **score_color** (*str, optional*) – The color used for plotting the enrichment scores. ["rgb(0,109,219)"]
- **enrichment_color** (*str, optional*) – The color used for plotting the fold enrichment values (if enabled). ["rgb(219,109,0)"]
- **cutoff_color** (*str, optional*) – The color used for indicating the XL-mHG test cutoff. ["rgba(255, 109,182,0.5)"]
- **line_width** (*int or float, optional*) – The line width used for plotting. [2.0]
- **ymax** (*int or float or None, optional*) – The y-axis limit. If `None`, determined automatically. [None]
- **mHG_label** (*bool, optional*) – If `True`, label the p-value with "mHG" instead of "XL-mHG". [False]

Returns The Plotly figure.

Return type `plotly.graph_obs.Figure`

How to cite the XL-mHG test

If you use the XL-mHG test in your research, please cite Eden et al. (PLoS Comput Biol, 2007) and Wagner (PLoS One, 2015).

License

XL-mHG Documentation

XL-mHG

Index