mhg: Non-parametric Enrichment Test for a Ranked Binary List

Kamil Slowikowski

2015-05-26

Contents

Abstract

The `mhg` package implements an enrichment test for a ranked binary list. Given a ranked binary list of ones and zeros, this package enables you to test if the ones are enriched at the beginning of the list. The method uses a dynamic programming algorithm to obtain an exact permutation p-value.

Simulation

Suppose we have a set of $N = 5000$ genes and $K = 100$ of them are annotated with a Gene Ontology (GO) term. Further, suppose that we find some subset of these genes to be significantly differentially expressed (DE) between two conditions. Within the DE genes, we notice that $k = 15$ of the DE genes are annotated with the Gene Ontology term. At this point, we would like to know if the GO term is enriched for DE genes.

A common strategy is to use the hypergeometric distribution to compute a probability that we would observe a given number of DE genes annotated with a GO term. Suppose we only look at the top n genes with the greatest fold-change. Then, the probability to observe exactly k of the n genes to be annotated with a GO term is:

$$\text{Prob}(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

The intuition is commonly described in terms of an urn with marbles. Suppose there are N marbles, some of which are green and the rest are red. We know that K of them are green and $N - K$ are red. We draw n marbles from the urn at random and observe that k of them are green and $n - k$ are red. The probability to observe exactly k green marbles when we draw n marbles is the hypergeometric probability. This is also equivalent to the one-tailed version of Fisher's exact test (Rivals et al. 2007).

Algorithm

The minimum hypergeometric (mHG) algorithm consists of three steps:

1. Compute a hypergeometric probability at each rank in the list.
2. Choose the minimum hypergeometric probability (mHG) as the test statistic.
3. Use dynamic programming to compute the exact permutation p-value for observing a test statistic at least as extreme by chance.

Eden et al. described the original mHG algorithm in the context of transcription factor binding motif enrichments (Eden et al. 2007). The interested reader should go to the methods section for more details and mathematical proofs.

Wagner extended the algorithm to be more robust by using two extra parameters X and L that limit the number of tests (Wagner 2015). The X paramter requires at least X ones to be present in the beginning of the list. The L parameter limits testing to the first L ranks ($L < N$). The interested reader should go to the methods section for more details.

This R package implements the extended algorithm, which Wagner calls "XL-mHG". A Cython implementation of the extended algorithm is described in (Wagner 2015) and is available here.

Example

Run the enrichment test like this:

```
library(mhg)

# Size of the population.
N <- 5000
# Successes in the population.
K <- 100
# Only consider enrichments in the first L observations.
L <- N / 4
# Require at least X successes in the first L observations.
X <- 5
# Define 15 items in the population as successes.
set.seed(42)
x <- rep(0, N)
x[sample(100, 5)] <- 1
x[sample(200, 10)] <- 1

# Test for enrichment.
res <- mhg_test(x, N, K, L, X)
```

The results are stored in a list called `res` with three elements:

```
names(res)
```

```
## [1] "threshold" "mhg"       "pvalue"
```

```
res$threshold
```

```
## [1] 147
```

```
res$mhg[1:30]
```

```
##  [1] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##  [8] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [15] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [22] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.4212652 0.4329027
## [29] 0.1135398 0.1201272
```

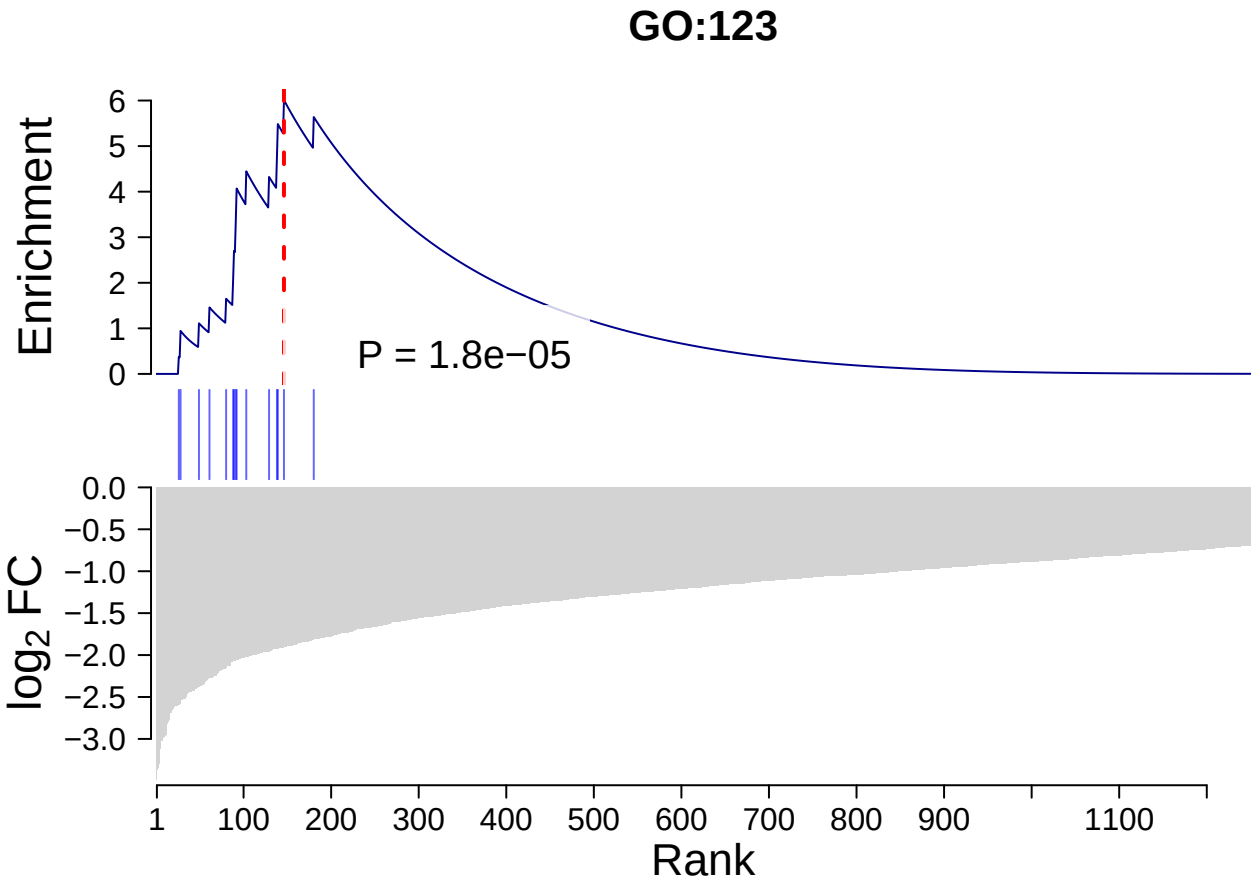
```
res$pvalue
```

```
## [1] 1.810658e-05
```

Plot the results like this:

```
# Simulate a fold-change for the plot.
fc <- sort(rnorm(N, 0, 1))

# This is how you can plot the results.
plot_mhg(
  values = fc,
  x = x,
  res = res,
  n = L,
  main = "GO:123",
  value = bquote("log"[2] ~ "FC")
)
```



The top panel shows the enrichment score. The y-value is the negative log10 of the hypergeometric probability to observe k successes after n trials. n is increased by one each time we step along the x-axis and k is increased when we encounter a new success. The red dotted line shows where we find the minimum hypergeometric probability (mHG).

The middle panel shows which of the ranked items in the tested list are successes (blue) or failures (white). In this case, a gene is a success if it is annotated with a GO term and significantly differentially expressed.

The bottom panel shows the values used to rank the items in the tested list in decreasing order. In this case, we found that some of the most down-regulated genes are enriched for the GO term. If we wish to test the most up-regulated genes instead, then we can reverse the list of genes by sorting in decreasing order, and then repeat the test. If we wish to test genes with large fold-change in either direction, then we can take the absolute value of log fold-change, rank the genes in decreasing order, and then repeat the test.

References

Eden, Eran, Doron Lipson, Sivan Yogev, and Zohar Yakhini. 2007. "Discovering Motifs in Ranked Lists of DNA Sequences." *PLoS Comput. Biol.* 3 (3): e39. doi:10.1371/journal.pcbi.0030039.

Rivals, Isabelle, Léon Personnaz, Lieng Taing, and Marie-Claude Potier. 2007. "Enrichment or Depletion of a GO Category Within a Class of Genes: Which Test?" *Bioinformatics* 23 (4): 401–7. doi:10.1093/bioinformatics/btl633.

Wagner, Florian. 2015. "GO-PCA: An Unsupervised Method to Explore Biological Heterogeneity Based on Gene Expression and Prior Knowledge." *BioRxiv*, 30~apr. doi:10.1101/018705.