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

Kamil Slowikowski

2015-05-22

Contents

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.

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 is:

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

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.

This algorithm was first described in Eden et al. 2007, and interested readers should go to the methods section for more details. A Cython implementation of the algorithm was described in Wagner 2015 and is available here.

Example

Run the enrichment test like this:

```r
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:

```r
names(res)
```

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

```r
res$threshold
```

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

```r
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
```

```r
res$pvalue
```

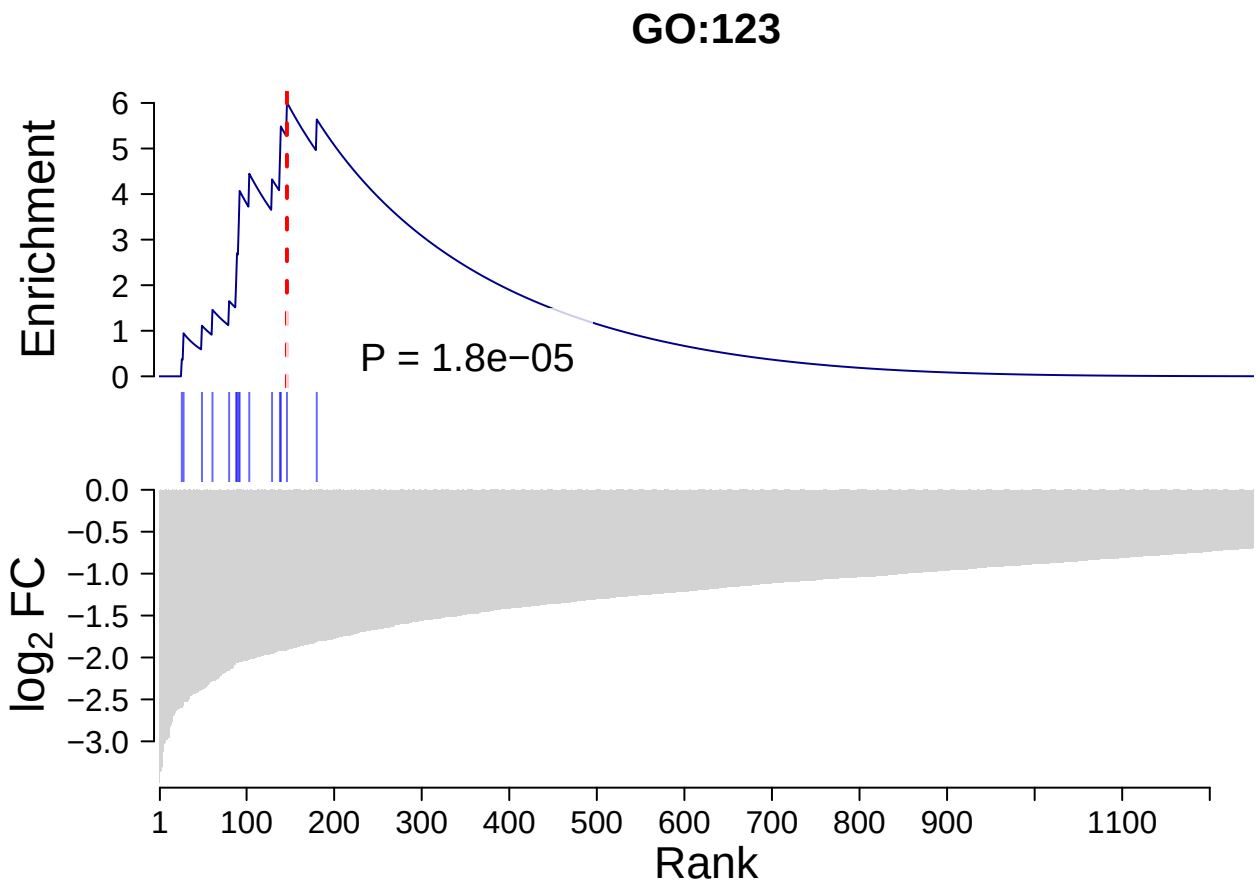
```
## [1] 1.810658e-05
```

Plot the results like this:

```r
# 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.