The XL-mHG test for gene set enrichment

Florian Wagner[1,2,*]

[1]Graduate Program in Computational Biology and Bioinformatics, Duke University, Durham, NC, USA
[2]Center for Genomic and Computational Biology, Duke University, Durham, NC, USA
[*]Email: florian.wagner@duke.edu

ABSTRACT

The nonparametric *minimum hypergeometric* (mHG) test is a popular alternative to Kolmogorov-Smirnov (KS)-type tests for determining gene set enrichment. However, these approaches have not been compared to each other in a quantitative manner. Here, I first perform a simulation study to show that the mHG test is significantly more powerful than the one-sided KS test for detecting gene set enrichment. I then illustrate a shortcoming of the mHG test, which has motivated a semiparametric generalization of the test, termed the *XL-mHG* test. I describe an improved quadratic-time algorithm for the efficient calculation of exact XL-mHG p-values, as well as a linear-time algorithm for calculating a tighter upper bound for the p-value. Finally, I demonstrate that the XL-mHG test outperforms the one-sided KS test when applied to a reference gene expression study, and discuss general principles for analyzing gene set enrichment using the XL-mHG test. An efficient open-source Python/Cython implementation of the XL-mHG test is provided in the `xlmhg` package, available from PyPI and GitHub (https://github.com/flo-compbio/xlmhg) under an OSI-approved license.

Keywords: gene set enrichment, nonparametric statistics, algorithms, hypothesis testing

INTRODUCTION

Gene set enrichment (Mootha et al. 2003) can be thought of as a general framework for utilizing *prior knowledge* in the analysis of transcriptomic data. It is based on the observation that functionally related genes tend to be co-expressed, and that it is therefore possible to *borrow strength* by jointly analyzing the expression patterns of functionally related genes. GSEA (Subramanian et al. 2005), the most popular incarnation of this framework, has been cited more than 10,000 times, according to Google Scholar (as of 2/2017).

The enormous popularity of GSEA notwithstanding, an impressive number of alternative gene set enrichment methods have been described in the literature. Most approaches, including GSEA, comprise a stereotypical sequence of steps (Ackermann and Strimmer 2009):

- Step 1: Each gene is assigned a score. The way this score is calculated is application-specific: In supervised settings, this is typically a test statistic that quantifies differential expression on a gene-by-gene basis, as in Subramanian et al. (2005) and Mootha et al. (2003).
- Step 2: Based on the gene-level scores, a "global" test statistic is used to quantify the *enrichment* of individual gene sets. This can involve a transformation of the gene-level scores, such as a rank-transformation.
- Step 3: The statistical significance of each of the global test statistics obtained is established. This often involves one or more permutation tests, sometimes in combination with an FDR criterion.

Unlike the choices involved in Steps 1 and 3, which are largely based on theoretical considerations, the choice of an enrichment test statistic should first and foremost capture the biologist's intuition for what constitutes "enrichment". While the precise notion of enrichment can again vary among applications, the general idea referred to by Mootha et al. (2003) and Subramanian et al. (2005) is that a *subset* of genes in a gene set is overrepresented "at the top of the [ranked] list". No assumption is made about the behavior of the remaining genes in the gene set. This intuition can be justified by three observations: First, curated gene sets often reflect incomplete knowledge of the true set of genes involved a specific cellular process. Therefore, such gene sets can contain false positives. Second, even if the involvement of a gene in a specific cellular process is well-established, the same gene can also be involved in a number of other processes (gene sets are not mutually exclusive), which can impact its expression pattern in unexpected ways. Third, the expression of a gene is usually governed by a complex system of regulatory mechanisms. As a result, genes regularly exhibit unforeseen transcriptional responses. In other words, from a biologist's point of view, it is *expected* that only a subset of genes in an enriched gene set exhibit correlated expression patterns, while other gene members behave in some unpredictable fashion. To make this idea more explicit, this article will occasionally refer to this concept as "subset enrichment", although the author deems it generally synonymous with "enrichment".

Surprisingly, most of the test statistics proposed for quantifying enrichment, such as the simple mean (Irizarry et al. 2009), the GSEA "ES" score (Subramanian et al. 2005) and even the nonparametric wilcoxon rank-sum test statistic (Barry, Nobel, and Wright 2005) do not strictly reflect the aforementioned notion of subset enrichment. Specifically, for all the examples listed, the value of the statistic always depends on the precise scores or ranks of *all* genes in the gene set, never on just a subset of them. The value of the "maxmean" statistic proposed by Efron and Tibshirani (2007) depends only on genes with positive scores, or only on those with negative scores, depending on which mean is greater in absolute value. However, it cannot focus on only a subset of the genes whose scores have the same sign.

To the author's knowledge, among all the test statistics proposed for quantifying enrichment, the only two that directly embrace the notion of subset enrichment are the one-sided KS statistic (a slightly modified version of which was proposed by Mootha et al. (2003)), as well as the minimum hypergeometric (mHG) statistic (Eden, Lipson, et al. 2007; Eden, Navon, et al. 2009). These two statistics also have the added advantage that they allow for a direct calculation of an associated p-value, which greatly facilitates their interpretation, and obviates the need for performing gene-level permutations in order to "restandardize" the enrichment scores (Efron and Tibshirani 2007). While the properties of the KS test are well-understood, the mHG test has not received much attention by authors surveying the statistical merits of different approaches to quantifying gene set enrichment. For example, neither Ackermann and Strimmer (2009) nor Maciejewski (2014) included the mHG test in their respective studies. In fact, the study by Ackermann and Strimmer was published back-to-back with the paper by Eden, Navon, et al. (2009). This paper proposed the application of the mHG test for quantifying gene set enrichment, and described a web application named GOrilla designed for this purpose. Since then, GOrilla has become a popular tool for enrichment analysis, as judged by its over 1,000 citations (Google Scholar, as of 2/2017), and the statical properties of the mHG test therefore warrant a closer examination.

Like the KS test, the mHG test is both rank-based and completely nonparametric (Eden, Lipson, et al. 2007). Unlike the KS test, however, it is based on the observation that, given a cutoff that defines "the top of the list", enrichment can easily be quantified using a hypergeometric test (equivalent to Fisher's exact test). However, in most applications, there is no way of knowing an optimal cutoff *a priori*. Therefore, instead of working with a fixed cutoff, the mHG test goes over *all possible cutoffs* and calculates a hypergeometric p-value for each of them. The test statistic is then defined as the smallest of these p-values. By not relying on a fixed cutoff to define "the top", the mHG test can detect both an usual accumulation of 1's among the first few elements, as well as a moderate enrichment within, say, the entire first half of the list.

The XL-mHG test (Wagner 2015a) generalizes the mHG test by introducing two parameters, X and L. These parameters specify the *minimum number of 1's required for enrichment*, and the *lowest cutoff to be examined*, respectively. (It should be noted that the L parameter was already suggested by Eden, Lipson, et al. (2007), under the name $n_{\max}$.) Together, these parameters provide a certain level of control over the kind of enrichment that is being tested for, as well as a flexible trade-off between the sensitivity and robustness of the test. For $X = 1$ and $L = N$, the XL-mHG test reduces to the mHG test.

This manuscript describes multiple results concerning the mHG and XL-mHG tests: First, a simulation study is performed to compare the mHG test and the KS test in terms of their statistical power to detect different types of enrichment. Second, the differences between the KS and mHG tests are highlighted on real expression data, motivating the use of the XL-mHG test. Third, a new algorithm for calculating XL-mHG p-values is described, and its advantages over the algorithm described by Eden, Lipson, et al. (2007) are demonstrated. Finally, a general procedure for gene set enrichment analysis using the XL-mHG test is proposed, and results on real expression data are shown.

Notation and definitions

We represent a ranked list with boolean entries as a column vector $\boldsymbol{v}$ of length N, with all elements being either 0 or 1:

$$\boldsymbol{v} = (v_1, v_2, \ldots, v_N)^T,\ v_i \in \{0, 1\}$$

We therefore also refer to list entries as "elements". We refer to the set of all elements for which $v_i = 0$ as "the 0's", and to the set of all other elements as "the 1's". We also say that v_1 represents the "topmost" element, and v_N the "bottommost" element of the list. We further let K and W denote the total number of 1's and 0's in the list, respectively ($K + W = N$). Throughout this article, we assume that N and K (and therefore W) are fixed, unless stated otherwise. We next define $\mathcal{V}^{(N,K)}$ to be the set of all lists of length N that contain exactly K 1's (there are $\binom{N}{K}$ distinct lists in $\mathcal{V}^{(N,K)}$).

Let $f(k;\ N, K, n)$ represent the probability mass function of the hypergeometric distribution:

$$f(k;\ N, K, n) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} \tag{Hypergeometric PMF}$$

Then, let $p^{\text{HG}}(k;\ N, K, n)$ represent the hypergeometric p-value:

$$p^{\text{HG}}(k;\ N, K, n) = \sum_{j=k}^{\min(n,K)} f(j;\ N, K, n) \tag{Hypergeometric p-value}$$

For any $\boldsymbol{v} \in \mathcal{V}^{(N,K)}$ and $n \in \{1, 2, ..., N\}$, let $k_n(\boldsymbol{v})$ represent the number of 1's among the first n elements of $\boldsymbol{v}$:

$$k_n(\boldsymbol{v}) = \sum_{i=1}^{n} v_i$$

Then, let $p_n^{\text{HG}}(\boldsymbol{v})$ represent the hypergeometric p-value for $\boldsymbol{v}$ using n as the "cutoff":

$$p_n^{\text{HG}}(\boldsymbol{v}) = p^{\text{HG}}(k_n(\boldsymbol{v});\ N, K, n)$$

The mHG test statistic $s^{\text{mHG}}(\boldsymbol{v})$ is then defined as follows (Eden, Lipson, et al. 2007):

$$s^{\text{mHG}}(\boldsymbol{v}) := \min p_n^{\text{HG}}(\boldsymbol{v}) \tag{mHG test statistic}$$

Let V^0 be a random variable representing a list drawn uniformly at random from $\mathcal{V}^{(N,K)}$. Let $S^{\text{mHG},0}$ be the mHG test statistic of V^0. Then the mHG p-value $p^{\text{mHG}}(\boldsymbol{v})$ is defined as follows (Eden, Lipson, et al. 2007):

$$p^{\text{mHG}}(\boldsymbol{v}) := \Pr(S^{\text{mHG},0} \leq s^{\text{mHG}}(\boldsymbol{v})) \tag{mHG p-value}$$

Given parameters X and L, both $\in \{1, 2, ..., N\}$, the XL-mHG test statistic $s_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v})$ is defined as follows (Wagner 2015b):

$$s_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v}) := \begin{cases} \min\limits_{\substack{k_n(\boldsymbol{v}) \geq X \\ n \leq L}} p_n^{\text{HG}}(\boldsymbol{v}) & \text{if } k_L(\boldsymbol{v}) \geq X, \\ 1 & \text{otherwise} \end{cases} \tag{XL-mHG test statistic}$$

The XL-mHG p-value $p_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v})$ is defined analogous to $p^{\text{mHG}}(\boldsymbol{v})$ (Wagner 2015b). Let $S_{\text{X,L}}^{\text{XL-mHG},0}$ be the XL-mHG test statistic of V^0. Then:

$$p_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v}) := \Pr(S_{\text{X,L}}^{\text{XL-mHG},0} \leq s_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v})) \tag{XL-mHG p-value}$$

RESULTS

The mHG test is much more powerful than the Kolmogorov-Smirnov (KS) test in detecting certain types of enrichment

To compare the mHG test and the KS test in terms of their power to detect various types of gene set enrichment, I designed a series of simple experiments in which I simulated lists of length N=10,000, roughly corresponding to the number of genes expressed in a given cell type or tissue at or above a threshold of 1 RPKM (Ramsköld et al. 2009). In each experiment, I simulated varying levels of enrichment, corresponding to an overrepresentation of 1's among the first n elements of the list. For each simulated list, I applied both tests and asked whether it was significant at a stringent significance level of $\alpha = 10^{-6}$, which corresponds to a significance level of 0.05 after Bonferroni correction for testing 25,000 gene sets for enrichment among both the most up-regulated and down-regulated genes (for a total of 50,000 tests). The experiments differed by the choice of n parameter, as well as the total number of 1's in the list (K).

As shown in Figure 1, the mHG outperformed the KS test in all four experiments, with the differences being greatest in the first case, where n and K were very small. In that experiment, the mHG test achieved 100% power for 300-fold enrichment (corresponding to three out of five 1's being present among the first 20 elements of the list), whereas the KS test only achieved the same power for 500-fold enrichment (i.e., when all five 1's were present among the first 20 elements). In contrast, for large n, the difference was much smaller in terms of the absolute fold enrichment: The mHG test achieved 100% power for 2.4-fold enrichment, and the KS test for 3-fold enrichment.

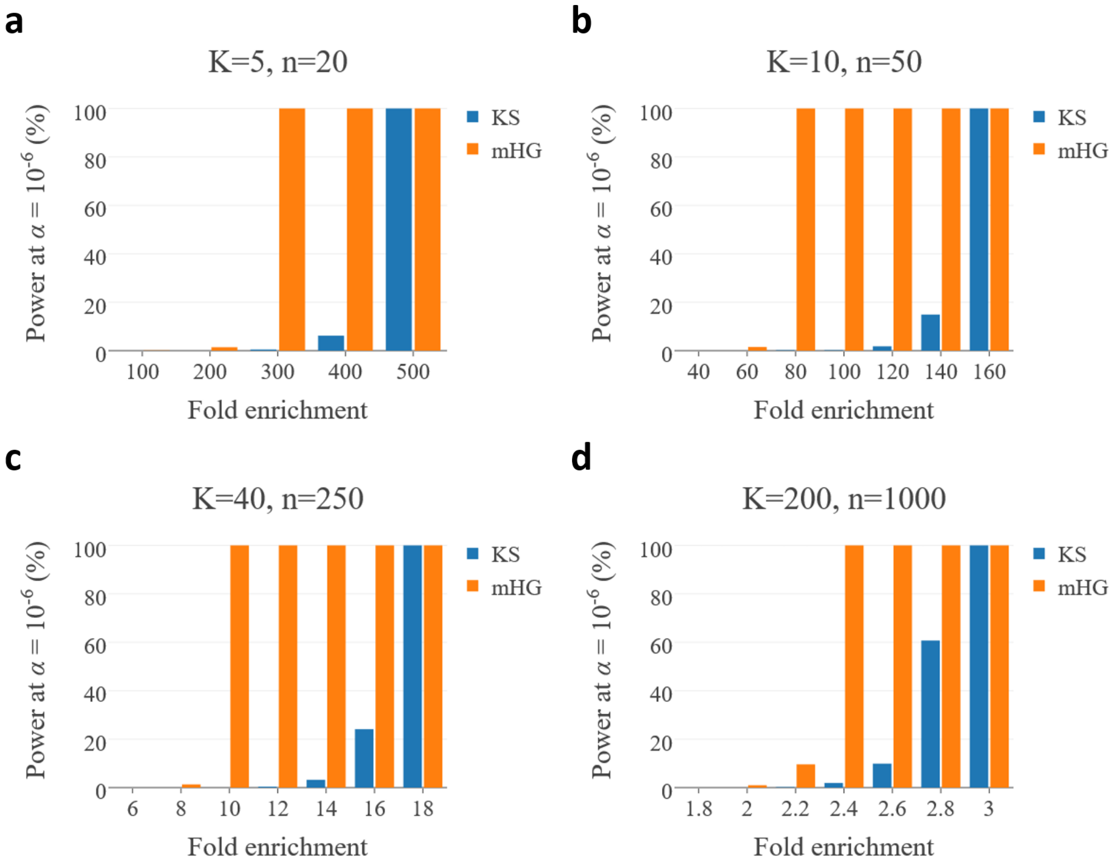

Figure 1. Power comparison between the minimum hypergeometric (mHG) test and the one-sided Kolmogorov-Smirnov (KS) test for detecting enrichment. Lists containing varying levels of fold enrichment within the "top of the list" (specified by the n parameter) were simulated. For each list, it was assessed whether the tests were significant at the level $\alpha = 10^{-6}$. Plots show the estimated power (fraction of significant tests), calculated based on 1,000 simulations for each fold enrichment value. **a**-**d** show the results of three experiments for different choices of K and n, as indicated above each panel.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1962v3 | CC BY 4.0 Open Access | rec: 12 Feb 2017, publ: 12 Feb 2017

The mHG and KS tests exhibit strong differences when applied to real expression data

To test how the differences between the mHG and KS test statistics affect the quantification of gene set enrichment in practice, I applied both tests to individual gene sets to the study by Subramanian et al. (2005) of 50 cell lines from the NCI-60 collection with and without mutations of the tumor protein p53, encoded by the *TP53* gene (this study is henceforth referred to as `p53`). p53 is important in regulating a cell's response to a variety of stresses, including DNA damage, and acts as a tumor suppressor in many cancers.

For the "p53Pathway" gene set, which Subramanian et al. reported as enriched among genes more highly expressed in wild-type cell lines, KS test p-value was 0.051, whereas the mHG test p-value was $4.0 * 10^{-8}$. Another pathway, "DNA_DAMAGE_SIGNALING" was not reported as enriched by the authors, had a KS test p-value of 0.29. However, its mHG test p-value was $3.1 * 10^{-7}$. These examples show that there can be dramatic differences between the KS test and the mHG test in terms of which gene sets are considered enriched. To better understand the basis of these differences, I visualized each of the four tests using a GSEA-style enrichment plot. For the "p53Pathway" gene set, the KS test statistic was based on the occurrence of 5/16 genes from the gene set among the first 191 genes in the ranked list (see Figure 2**a**). In contrast, the mHG test statistic was based on the occurrence of 3/16 genes from the gene set at the very top of the list (see Figure 2**b**). In other words, the first three genes in the ranked list were all contained in the gene set. Given a ranked list of over 10,000 genes, this is very unlikely to occur by chance for a set of 16 genes, which explains the highly significant mHG p-value. The situation for the "DNA_DAMAGE_SIGNALING" gene set was generally similar, but with the important difference that the gene set comprised 90 instead of 16 genes (see Figure 2**c**,**d**), and that a much smaller fraction of them appeared located at the top. For example, only 8/90 genes were among the first 200 genes, representing less than 10%. The KS test was not clearly not significant in this situation ($p = 0.29$), whereas the mHG test was highly significant ($p = 3.1 * 10^{-7}$), based on the occurrence of 7/90 genes among the first 31 genes.

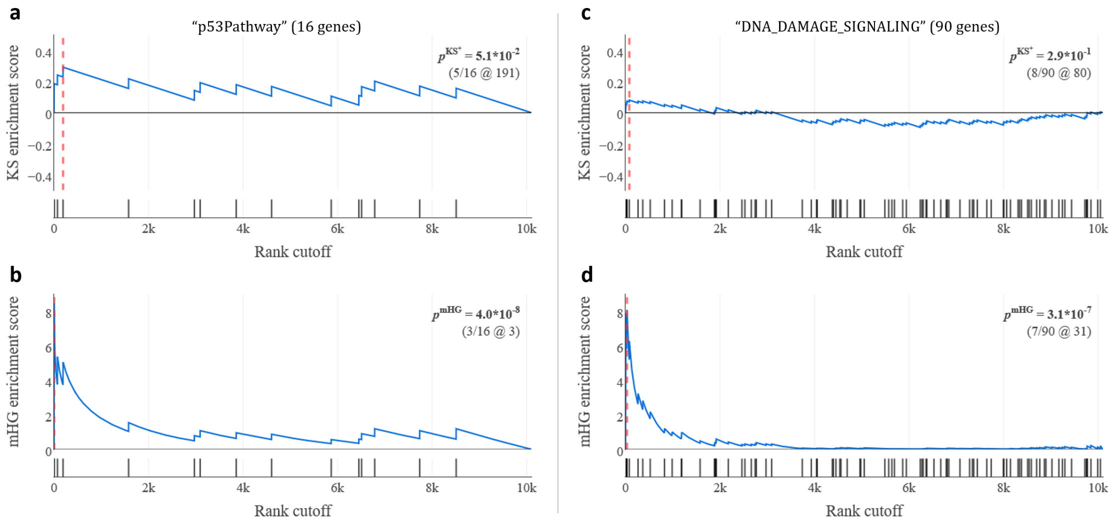

Figure 2. Enrichment of two example gene sets in the `p53` study by Subramanian et al. (2005), quantified using the KS test and the XL-mHG test. The behavior of the two different tests is shown using GSEA-style plots of the running enrichment scores that underlie the calculation of the respective test statistics. In each case, the cutoff that gives rise to the value of the test statistic is indicated by a dashed red line. The test p-values are shown in the top right corner of each plot, along with additional information about the number of gene set genes observed at the cutoff. **a**, **b** Enrichment of the "p53_pathway" gene set, quantified using the KS test (**a**) and the mHG test (**b**). **c**, **d** Enrichment of the "DNA_DAMAGE_SIGNALING" gene set, quantified using the KS test (**c**) and the mHG test (**d**).

The foregoing analysis demonstrated both the power of the mHG test, as well as a potential pitfall associated with it. In agreement with the simulation results, the mHG was much more sensitive than the KS test in detecting the enrichment of the "p53Pathway" gene set. However, this extreme sensitivity meant that the mHG test detected enrichment even there was only a very small fraction of genes located the very top of the list. This behavior is difficult to justify from a biological point of view: When seven out of 90 genes in the "DNA_DAMAGE_SIGNALING" gene set are among the first 30 genes, should we really conclude that the cells with wild-type p53 engage in a DNA damage signaling response (or that the p53 mutant cells repress this response)? If so, why are the vast majority of the genes in this gene set not up-regulated in the same fashion? At a certain point, i.e., when the size of the subset gets too small, the notion of "subset enrichment" clearly reaches its limits.

An improved algorithm for calculating the XL-mHG p-value

The overly sensitive behavior of the mHG test illustrated above is a direct result of the fact that it considers all possible cutoffs in the calculation of its test statistic. Therefore, I have argued that the test can be made more robust and specific by the introduction of two parameters, X and L, which restrict the set of cutoffs considered (Wagner 2015b). The X parameter dictates that all cutoffs at the beginning of the list for which fewer than X genes from the gene set have been encountered are not to be considered. This addresses cases like the one discussed above, since X can for example be chosen to equal at least 25% of the number of genes. In contrast, the L parameter dictates that cutoffs at the end of the list, beyond rank L, are not to be considered. I have termed the resulting test the "XL-mHG" test, and proposed a modification to the dynamic programming algorithm proposed by Eden, Lipson, et al. (2007) that allows an efficient calculation of exact p-values for the XL-mHG test (Wagner 2015b). I will henceforth refer to this modified algorithm as `PVAL1`.

Briefly, for given N, K, X, L, and $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, `PVAL1` determines the fraction of lists in $\mathcal{V}^{(N,K)}$ with an XL-mHG test statistic at least as good as (i.e., equal to or smaller than) $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$. By definition, this is the XL-mHG p-value $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$. The first key insight behind the approach developed by Eden, Lipson, et al. (2007) is that even though the number of lists in $\mathcal{V}^{(N,K)}$ grows extremely quickly with N (e.g., $|\mathcal{V}^{(100,20)}| \approx 5.4 \times 10^{20}$), there exist only $(K+1)*(W+1)$ unique "hypergeometric configurations" $\mu_{(n,k)} \in \mathcal{M}^{(N,K)}$ (with $W = N - K$), each associated with a hypergeometric p-value $p_{(n,k)}$. Any list $\boldsymbol{v} \in \mathcal{V}^{(N,K)}$ has a unique representation as a sequence of hypergeometric configurations $(\mu_1, \mu_2, ..., \mu_N)$, corresponding to all possible cutoffs $(1, ..., N)$. Eden, Lipson, et al. (2007) refer to this sequence of configurations as a *path* (through $\mathcal{M}^{(N,K)}$; see Figure 3). Let $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ be the set of all configurations with $p_{(n,k)} \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$. Then, each list whose path "enters" $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ has a mHG test statistic of $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ or smaller. In this scheme, $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ therefore equals the fraction of lists whose paths enter $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$. For the mHG test, Eden, Lipson, et al. (2007) showed that this problem exhibits *optimal substructure*, making it amenable to dynamic programming. First, the authors observed that each path that contains a configuration $\mu_{(n,k)}$ either also contains the configuration $\mu_{(n-1,k)}$ or $\mu_{(n-1,k-1)}$. In the grid representation of $\mathcal{M}^{(N,K)}$ shown in Figure 3, this means that a configuration (dot) is reached "from the left" or "from below", respectively. Furthermore, they proposed to calculate the fraction of paths $\pi(\boldsymbol{v})$ that do *not* enter $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ (so that $p^{\text{mHG}}(\boldsymbol{v}) = 1 - \pi(\boldsymbol{v})$). The algorithm relies on the following recurrence relation for calculating the fraction of all paths (i.e., all $\boldsymbol{v} \in \mathcal{V}^{(N,K)}$) that do not enter $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ before arriving at a given configuration $\mu_{(n,k)}$:

$$\pi_{(n,k)}(\boldsymbol{v}) = \begin{cases} 0, & \text{if } \mu_{(n,k)} \in \mathcal{R}_{\text{X,L}}(\boldsymbol{v}), \\ \pi_{(n-1,k)}(\boldsymbol{v})\frac{W-w+1}{N-n+1} + \pi_{(n-1,k-1)}(\boldsymbol{v})\frac{K-k+1}{N-n+1} & \text{otherwise} \end{cases}$$

(Recurrence relation for `PVAL1`)

Obviously, if $\mu_{(n,k)} \in \mathcal{R}_{\text{X,L}}(\boldsymbol{v})$, all paths arriving at $\mu_{(n,k)}$ have now entered $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$, and $\pi_{(n,k)}(\boldsymbol{v}) = 0$. The coefficients in the other case represent the fraction of lists with configuration $\mu_{(n-1,k)}$ that have a 0 in position n, and the proportion of lists with configuartion $\mu_{(n-1,k-1)}$ that have a 1 in position n, respectively. If $w = 0$, or if $k = 0$, the first or second term of the recurrence relation is omitted, respectively, for the case $\mu_{(n,k)} \notin \mathcal{R}_{\text{X,L}}(\boldsymbol{v})$. Together with the initial value $\pi_{(0,0)} = 1.0$ — at the beginning, none of the paths have entered $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ —, and an efficient algorithm for determining whether $\mu_{(n,k)} \in \mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ for all $\mu_{(n,k)}$, this allows the calculation of $\pi(\boldsymbol{v}) = \pi_{(N,K)}$ in $\mathcal{O}(N^2)$; see Wagner (2015b) for a more detailed discussion.

`PVAL1`, while mathematically accurate and computationally efficient, still has some drawbacks in practice. First, it always requires the calculation of *all* $\pi_{(n,k)}(\boldsymbol{v})$, even though in many cases, only a small fraction of configurations are in $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$. For example, when $L = N/10$, approx. 90% of all $\mu_{(n,k)}$ are excluded from $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ by definition. Moreover, since $s^{\text{mHG}}(\boldsymbol{v})$ serves as a lower bound for $p^{\text{mHG}}(\boldsymbol{v})$,

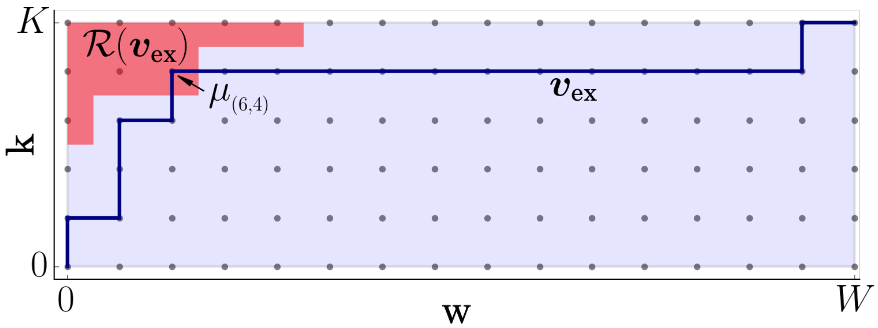
K
k
0
0
w
W
$\mathcal{R}(\boldsymbol{v}_{\text{ex}})$
$\mu_{(6,4)}$
$\boldsymbol{v}_{\text{ex}}$

Figure 3. Representation of lists $\boldsymbol{v} \in \mathcal{V}^{(N,K)}$ as *paths* through $\mathcal{M}^{(N,K)}$ (Eden, Lipson, et al. 2007). Each gray dot represents a hypergeometric configuration $\mu_{(n,k)}$ (with $n = w + k$), and collectively, the dots in the $(K+1) \times (W+1)$ grid represent the set of all configurations in $\mathcal{M}^{(N,K)}$. In this example, $N = 20$ and $K = 5$. The path of the list $\boldsymbol{v}_{\text{ex}} = (1,0,1,1,0,1,0,...,0,1,0)^T$ is shown in navy blue. The mHG test statistic $s^{\text{mHG}}(\boldsymbol{v}_{\text{ex}})$ of this list is attained at the cutoff $n = 6$ (see arrow), for which $\boldsymbol{v}_{\text{ex}}$ has the configuration $\mu_{(6,4)}$. Shown in red is the space of all configurations in $\mathcal{R}(\boldsymbol{v}_{\text{ex}})$. These configurations are associated with an mHG test statistic equal to or smaller than $s^{\text{mHG}}(\boldsymbol{v}_{\text{ex}})$. The mHG p-value for $s^{\text{mHG}}(\boldsymbol{v}_{\text{ex}})$ is equal to the fraction of lists in $\mathcal{V}^{(20,5)}$ whose paths enter $\mathcal{R}(\boldsymbol{v}_{\text{ex}})$.

calculating the mHG p-value is mostly of interest when $s^{\text{mHG}}(\boldsymbol{v})$ is below a specific significance threshold α (e.g., $\alpha = 10^{-6}$). In these cases the number of configurations in $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ can be expected to be very small as well. A second drawback arises from the fact that for technical reasons, computers typically do not represent decimal numbers as a string of (significant) digits. Instead, they use a *floating point* system which can only represent certain numbers from the real line. This can lead to inaccuracies when very small numbers are involved in addition or substraction. For example, in most computer programs, the expression $1.0 - 10^{-20}$ will surprisingly evaluate to (exactly) 1.0, because 1.0 is the closest representable number to $1.0 - 10^{-20}$ (see footnote[1]). For `PVAL1`, this means that when the true p-value is very small — say, smaller than 10^{-15} — , numerical inaccuracies start to occur in filling in the dynamic programming table (which relies on addition) and in the calculation of $p^{\text{mHG}}(\boldsymbol{v}) = 1 - \pi(\boldsymbol{v})$, resulting in an inaccurate p-value. More concretely, due to the lack of representable numbers between 1.0 and $1.0 - 10^{-16}$, the smallest non-zero p-value that can be obtained from `PVAL1` is $\approx 10^{-16}$ (see Figure 4**a**). When using an 80-bit "extended precision" data type, the smallest possible p-value is $\approx 10^{-19}$ (see Figure 4**b**).

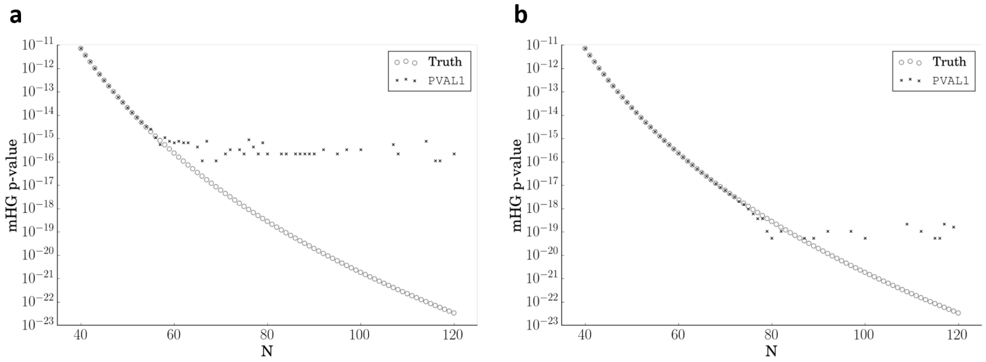

Figure 4. Numerical accuracy of `PVAL1`. Lists of varying length N ($N \in \{40, 41, ..., 120\}$), each consisting of exactly 20 1's followed by only 0's, were generated, and the mHG p-value for each list was calculated using `PVAL1`. Missing values correspond to cases where `PVAL1` returned a value of 0 or lower due to limited floating point accuracy. **a** Python implementation using the 64-bit "double-precision" data type. **b** Cython implementation using the 80-bit "extended precision" data type.

[1] In the commonly used IEEE-754 *binary64* ("double-precision") system, the first representable number below 1.0 is approximately 0.9999999999999999 or 1.0 - 10^{-16}.

Motivated by these limitations, I sought to design an algorithm for calculating the XL-mHG p-value $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ that would not require filling in the entire dynamic programming table, and avoid numerical inaccuracies in cases where the true p-value is very small. I realized that both of these limitations result from the fact that PVAL1 requires the computation of $\pi(\boldsymbol{v})$. If we could directly count the fraction of paths entering $\mathcal{R}(\boldsymbol{v})$ (instead of calculating the opposite, and then substracting that number from 1), this would allow us to stop the algorithm once we are confident that we have discovered all configurations in $\mathcal{R}(\boldsymbol{v})$, and it would avoid substracting a very small number from 1.0 for highly significant tests (instead, we would add several small numbers that are close to 0, where the density of representable numbers is much higher). I first made the following observation: In the visual representation of $\mathcal{M}^{(N,K)}$ as a $(K+1)\times(W+1)$ grid (see Figure 3), paths can only enter $\mathcal{R}(\boldsymbol{v})$ "from below". To see this, we first introduce the following lemma:

Lemma 1 (Monotonicity property of the hypergeometric p-value)**.** *For all $n < N$ and $k \leq \min(\{n, K\})$, $p^{HG}(k;\, N, K, n) < p^{HG}(k;\, N, K, n+1)$.*

Proof. $p^{\text{HG}}(k;\, N, K, n+1)$ is the probability of having k or more successes among $n+1$ draws. We can represent "k or more successes among $n+1$ draws" as the union of two mutually exclusive events A and B, so that $p^{\text{HG}}(k;\, N, K, n+1) = \Pr(\text{A}\cup\text{B}) = \Pr(\text{A}) + \Pr(\text{B})$. Event A: "$k$ or more successes among n draws". Event B: "a successful draw, conditional on exactly $k-1$ successes among n draws". We then have $\Pr(\text{A}) = p^{\text{HG}}(k;\, N, K, n)$, and $\Pr(\text{B}) > 0$. Therefore, $p^{\text{HG}}(k;\, N, K, n+1) > p^{\text{HG}}(k;\, N, K, n)$. □

Since $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ is defined as the set of all configurations whose hypergeometric p-value is equal to or smaller than fixed value (namely, $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$), we know from Lemma 1 that when a configuration $\mu_{(n,k)}$ is in $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$, then so is $\mu_{(n-1,k)}$, its "left neighbor" in the grid representation. Therefore, the only way for a path to *enter* $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ is "from below". In this case, $\mu_{(n,k)} \in \mathcal{R}_{\text{X,L}}(\boldsymbol{v})$, but $\mu_{(n-1,k-1)} \notin \mathcal{R}_{\text{X,L}}(\boldsymbol{v})$. We can refer to configurations for which this is true as "entry points" into $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ (see Figure 5). The basis of our new algorithm is then to calculate what fraction of paths enter $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ from below at all entry points, and then report the sum of all these fractions as the (XL-)mHG pvalue. However, since paths can exit and re-enter $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$, we need to ensure that we only count each path once, when it enters $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ for the first time. In other words, we must only consider paths that have never entered $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ before. Coincidentally, this is the exact same quantity that PVAL1 uses in order to calculate $\pi(\boldsymbol{v})$ (see above).

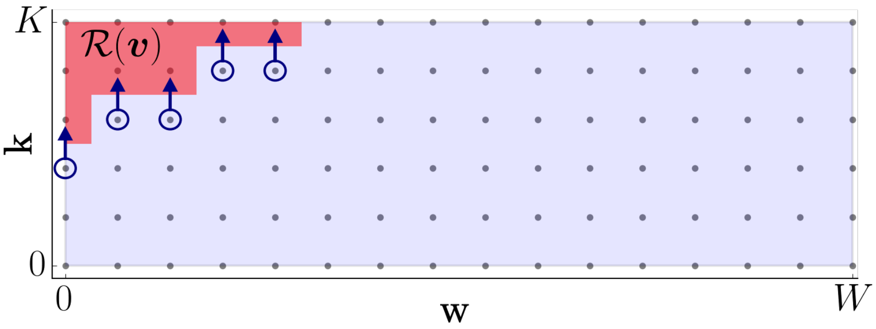

Figure 5. Idea behind PVAL2, illustrated using the example from Figure 3. At each "entry point" into $\mathcal{R}(\boldsymbol{v})$ (arrow tips), we calculate the fraction of paths entering from the configuration below (circles). However, in order to avoid counting paths more than once (some may exit and then re-enter $\mathcal{R}(\boldsymbol{v})$), we must base our calculation on only those paths that have not previously entered $\mathcal{R}(\boldsymbol{v})$. This is the exact same quantity used by PVAL1 to calculate $\pi(\boldsymbol{v})$. The (XL-)mHG p-value corresponds to total fraction of entering paths.

I refer to this new algorithm as PVAL2. Due to its reliance on the same recurrence relation as PVAL1, it requires only surprisingly small modifications to PVAL1. These are illustrated on a simplified version of PVAL2, which relies on a separate routine to determine $\mathcal{R}(\boldsymbol{v})$ (see pseudocode below). The full algorithm is provided in Appendix A.

To test whether PVAL2 exhibits better numerical stability than PVAL1, I repeated the experiment shown in Figure 4 for PVAL2. As can be seen in Figure 6, the new algorithm is able to calculate p-values much smaller than 10^{-16}, and numerical errors are no longer apparent.

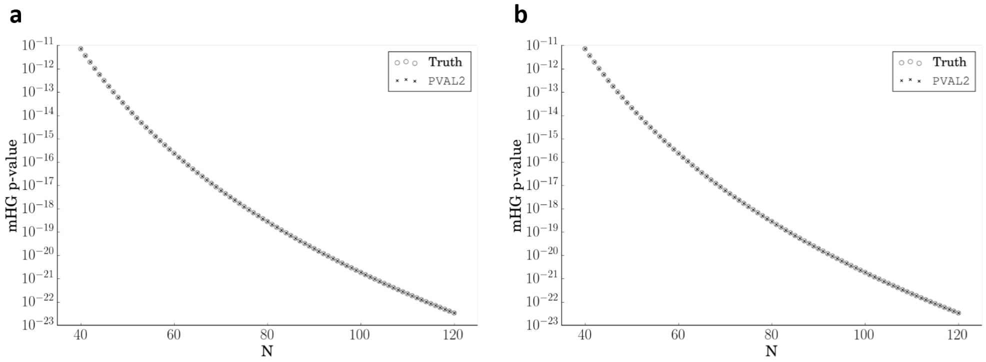

Figure 6. Numerical accuracy of `PVAL2`. Shown are results of an experiment as described in Figure 4, but conducted using `PVAL2`. **a** Python implementation using the 64-bit "double-precision" data type. **b** Cython implementation using the 80-bit "extended precision" data type.

To determine how the modifications introduced in `PVAL2` affect the runtime of the algorithm, I performed several benchmarks. As discussed above, I expected `PVAL2` to run significantly faster for lists containing significant enrichment, and for $L < N$. The benchmark results confirm this expectation, and show that in lists without enrichment and $L = N$, `PVAL2` runs only marginally faster than `PVAL1` (see Figure 7).

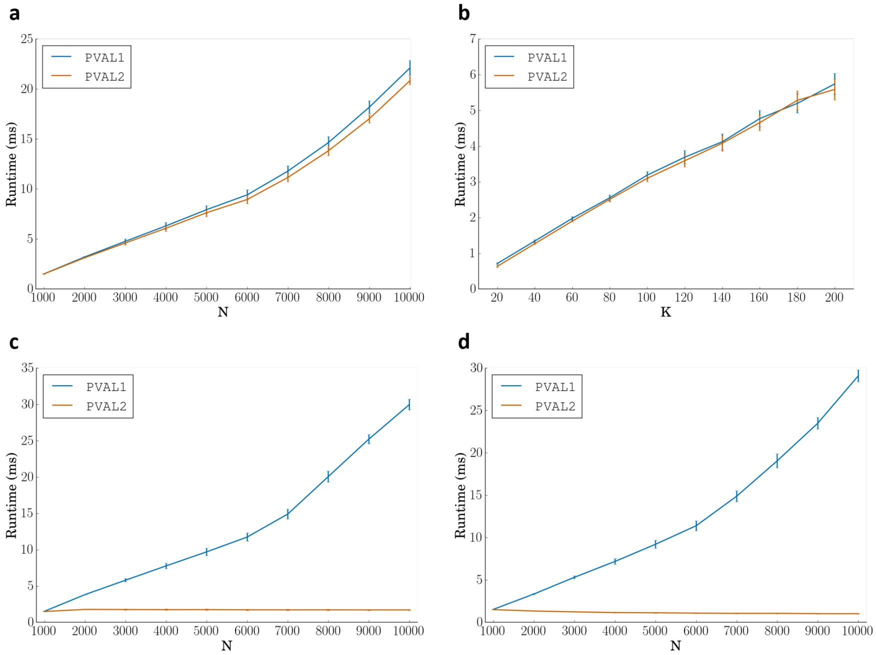

Figure 7. Comparison of runtimes of `PVAL1` and `PVAL2`. For each benchmark and each set of parameters, 100 lists were generated independently, and both algorithms were used to calculate the (XL)-mHG p-value for those lists. Shown are the means and standard deviations (error bars) over the 100 runs. All benchmarks were conducted using randomly generated lists where the positions of the 1's were sampled uniformly from all positions (except for **c**). **a** Benchmark using fixed K=100, for variable N (X=1; L=N). **b** Benchmark using fixed N=2,000, for variable K (X=1;L=N). **c** Benchmark for lists with enrichment, using fixed K=100 and variable N (X=1; L=N). The positions of the 1's were sampled uniformly from only the top 1,000 positions. **d** Benchmark using fixed K=100 and L=1,000, for variable N (X=1).

Algorithm 1: `PVAL2-SIMPLE`, an improved algorithm to calculate $p_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v})$ in $\mathcal{O}(N^2)$. This simplified version of `PVAL2` uses a separate routine to determine $\mathcal{R}(\boldsymbol{v})$, and does not handle comparisons of floating point variables properly. See Algorithm 5 in Appendix A for `PVAL2`.

```
Input: stat=$s_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v})$, N, K, X, L
Output: pval=$p_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v})$
R ← Algorithm 2 (stat, N, K, X, L) from Wagner (2015b)
pval ← 0.0
table ← empty $(K+1) \times (W+1)$ array of floats
table[0, 0] ← 1.0
W ← N-K
for n = 1 to L do
    k ← min(n,K)
    w = n-k
    // check whether we have seen all of $\mathcal{R}(\boldsymbol{v})$
    if k = K and R[k, w] = 0 then
        break
    end if
    while k $\geq$ 0 and w $\leq$ W do
        if R[k, w] = 1 then
            table[k, w] ← 0.0
            // check if this is an entry point into $\mathcal{R}(\boldsymbol{v})$ (entering is only possible “from below”)
            if k > 0 and R[k-1, w] = 0 then
                pval ← pval + (table[k-1, w] * (K-k+1)/(N-n+1))
            end if
        else if w > 0 and k > 0 then
            table[k, w] ← table[k, w-1] * (W-w+1)/(N-n+1) +
                    table[k-1, w] * (K-k+1)/(N-n+1)
        else if w > 0 then
            table[k, w] ← table[k, w-1] * (W-w+1)/(N-n+1)
        else if k > 0 then
            table[k,w] ← table[k-1, w] * (K-k+1)/(N-n+1)
        end if
        w ← w + 1
        k ← k - 1
    end while
end for
return pval
```

Bounds for the XL-mHG p-value

Eden, Lipson, et al. (2007) described one lower and two upper bounds for the mHG p-value, all of which I review in Appendix B. The mHG test statistic $s^{\text{mHG}}(\boldsymbol{v})$ itself serves as a lower bound for $p^{\text{mHG}}(\boldsymbol{v})$ (see Theorem 1). I found that the lower bound applies unchanged to the XL-mHG p-value (see Theorem 4 in Appendix C).

In the construction of their proof for the upper bound, Eden, Lipson, et al. (2007) introduced the notion of special cutoffs n_k, for $k \in \{1, ..., K\}$, that correspond to the lowest cutoffs so that $p^{\text{HG}}(k;\, N, K, n_k) \leq s^{\text{mHG}}(\boldsymbol{v})$. This allowed the authors to represent the mHG p-value as a union of K events, which correspond to observing a hypergeometric p-value equal to or smaller than $s^{\text{mHG}}(\boldsymbol{v})$ at the respective n_k. By applying a union bound, Eden, Lipson, et al. (2007) found that an upper bound for $p_n^{\text{HG}}(\boldsymbol{v})$ is given by $K * s^{\text{mHG}}(\boldsymbol{v})$ (see Theorem 3). Depending on the choice of the parameters X and L, not all k need to be considered in the corresponding expression for XL-mHG p-value, since it is required that $k \geq X$ and $k \leq \min\{K, L\}$. Therefore, some of the events in are by definition excluded from the union, which results in a tighter bound of $((\min\{K, L\} - X + 1)s_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v})$ (see Theorem 5 in Appendix C).

A closer examination of the proof for Theorem 5 suggests that depending on X, L, and $s_{\text{X,L}}^{\text{XL-mHG}}(\boldsymbol{v})$, the actual number of events in the union of Equation (6) can be smaller than $(\min\{K, L\} - X + 1)$. This statement can be made more precise using the following two definitions:

$$k^{\min}_{\text{X,L}}(\boldsymbol{v}) := \min\{k : k \geq X, p^{\text{HG}}(k;\, N, K, k) \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})\}$$

$$k^{\max}_{\text{X,L}}(\boldsymbol{v}) := \begin{cases} \min\{k : n_k \geq L\}, & \text{if } n_K \geq L \\ K & \text{otherwise} \end{cases}$$

The number of unique events in Equation (6) is exactly $(k^{\max}_{\text{X,L}}(\boldsymbol{v}) - k^{\min}_{\text{X,L}}(\boldsymbol{v}) + 1)$, resulting in the following bound:

$$p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) \leq (k^{\max}_{\text{X,L}}(\boldsymbol{v}) - k^{\min}_{\text{X,L}}(\boldsymbol{v}) + 1) s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) \qquad (\mathcal{O}(N) \text{ upper bound for the XL-mHG p-value})$$

Let $b^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) := (k^{\max}_{\text{X,L}}(\boldsymbol{v}) - k^{\min}_{\text{X,L}}(\boldsymbol{v}) + 1) s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$. It turns out that $k^{\min}_{\text{X,L}}(\boldsymbol{v})$ and $k^{\max}_{\text{X,L}}(\boldsymbol{v})$, and therefore $b^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, can be obtained in $\mathcal{O}(N)$. To do so, I designed the algorithm `PVAL-BOUND` (see Algorithm 6 in Appendix A). Therefore, in cases where we need to determine whether $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ is equal to or smaller than a pre-specified significance threshold α, we can first calculate the original upper bound in $\mathcal{O}(1)$. If this bound is larger than $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, we can invoke `PVAL-BOUND` to calculate a potentially tighter upper bound in $\mathcal{O}(N)$. Only if this value is still larger than $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ do we need to calculate the exact value of $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ in $\mathcal{O}(N^2)$ (using `PVAL2`). This procedure is summed up in `PVAL-THRESH` (see Algorithm 2).

Algorithm 2: `PVAL-THRESH`— Efficiently determine whether $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) \leq \alpha$.

Input: thresh=α, stat=$s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, N, K, X, L
Output: `TRUE` if $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) \leq$ thresh, `FALSE` otherwise

```
if stat $>$ $\alpha$ then
    // using lower bound
    return FALSE
else if (MIN(K,L) - X + 1) * $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ $<$ thresh then
    // using upper bound
    return TRUE
end if
// calculate tighter bound in $\mathcal{O}(N)$
bound $\leftarrow$ PVAL-BOUND(stat, N, K, X, L)
if bound $\leq$ thresh then
    return TRUE
end if
// calculate exact p-value in $\mathcal{O}(N^2)$
pval $\leftarrow$ PVAL2(stat, N, K, X, L)
if pval $\leq$ thresh then
    return TRUE
end if
return FALSE
```

The XL-mHG test provides a more powerful alternative to the KS test for quantifying gene set enrichment

To assess the ability of the XL-mHG test to detect enrichment in real expression studies, I decided to compare the performance of the XL-mHG to that of the mHG test for all gene sets analyzed by Subramanian et al. (2005) in their `p53` study. I was particular interested to see if the XL-mHG test would be able to produce more significant results than the KS test for the gene sets reported as enriched by the authors. I specified the XL-mHG L parameter to the number of genes with positive scores, thereby making sure that cutoffs corresponding to genes that did not have higher expression in the wild-type compared to the mutant cell lines were not tested for enrichment. I set the XL-mHG X parameter, in a gene set-dependent fashion, to 25% of the number of genes in the gene set, or to 5, whichever was larger.

In supervised gene set enrichment analysis, it is considered best practice to perform a sample label permutation test (Chen et al. 2007), in order to avoid reporting artificially low p-values that can result when genes are assumed to be independent. I therefore decided to combine both the KS and XL-mHG tests with a sample permutation test. I will henceforth refer to the p-values associated with the KS and XL-mHG test statistics as “nomimal p-values”, and to the p-values obtained from the subsequent permutation test as “permutation p-values”. I will refer to the combined test procedures as the “KS/permutation test” and the “XL-mHG/permutation test”, respectively. The results of applying both tests to the `p53` study are shown in Figure 8**a**.

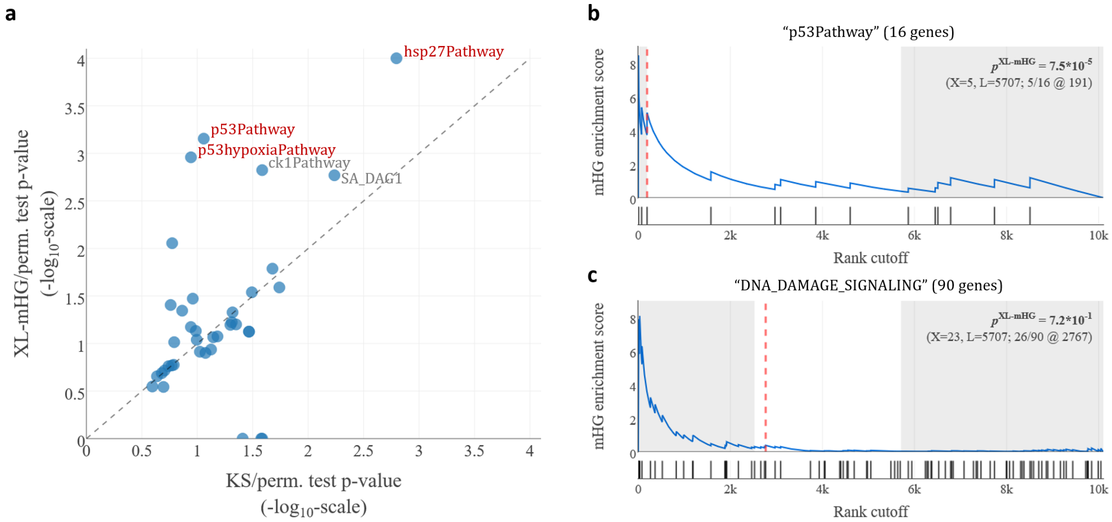

Figure 8. Application of the XL-mHG test for gene set enrichment to the `p53` study by Subramanian et al. (2005). a Comparison of permutation-based p-values obtained using the KS test (x-axis) and the XL-mHG test (y-axis). Only gene sets that had nominal p-values of 0.05 or lower for at least one of the two tests are shown. Gene sets annotated in red correspond to the three gene sets reported as enriched (FDR ¡= 0.05) in wild-type vs. mutated cell lines by Subramanian et al. **b, c** GSEA-style enrichment plots showing the application of the XL-mHG test to the “p53Pathway” and “DNA_DAMAGE_SIGNALING” gene sets, respectively (cf. Figure 2). Nominal p-values are shown in the top right corner of each plot. See text for details of how the X and L parameters were chosen.

For all three gene sets reported as enriched by Subramanian et al. (2005) (“p53Pathway”, “p53hypoxiaPathway”, and “hsp27Pathway”[2], I observed that the permutation p-values obtained using the XL-mHG test statistic were at least one order of magnitude lower (better) than when using the KS test statistic. Furthermore, for all gene sets that obtained a p-value of 0.01 or smaller using eihter test, the XL-mHG/permutation test yielded a smaller p-value. These results suggested that the XL-mHG was significantly more sensitive in detecting gene set enrichment.

To illustrate the specificity of the XL-mHG test, I visualized the XL-mHG test results for the “p53Pathway” and the “DNA_DAMAGE_SIGNALING” gene sets discussed earlier. The “p53Pathway” gene set was reported as enriched by Subramanian et al., and the XL-mHG test assigned it a nomimal p-value of $7.5 * 10^{-5}$ (see Figure 8**b**). This was not as good as the mHG test p-value of $4.0 * 10^{-8}$, but still much more significant than the KS test p-value of 0.051 (cf. Figure 2**a**,**b**). The “DNA_DAMAGE_SIGNALING” gene set was not reported as enriched by Subramanian et al. The XL-mHG test assigned that gene set a p-value of 0.72. This in sharp contrast to its mHG test p-value of $3.1 * 10{-}7$. These examples highlighted the fact that the XL-mHG test generally maintains the sensitivity of the mHG test, but can be made more specific to avoid detecting cases in which only a small fraction of genes in the gene set is located at the top of the list.

DISCUSSION

The results presented here extend the work of Eden, Lipson, et al. (2007) and Eden, Navon, et al. (2009), who introduced the nonparametric mHG test statistic, developed the dynamic programming approach for calculating its p-value, described both upper and lower bounds for the p-value, and developed a web

[2] The exact results from their study can be found at `http://software.broadinstitute.org/gsea/resources/gsea_pnas_results/p53_C2.Gsea/gsea_report_for_WT_1130958999391.html`

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1962v3 | CC BY 4.0 Open Access | rec: 12 Feb 2017, publ: 12 Feb 2017

interface[3] for analyzing gene set enrichment using the mHG test. Using simulated data, I have also shown that the mHG test is more powerful than the KS test for detecting enrichment, especially when a small number of genes are located at the very top of the list. However, using an example from the `p53` study by Subramanian et al. (2005), I have demonstrated that this extreme sensitivity can sometimes lead to positive test results, even when only a small fraction of genes in a gene set exhibit an expression response. To overcome this limitation, I have proposed to quantify gene set enrichment using the XL-mHG test (Wagner 2015b), which represents a semiparametric generalization of the mHG test that provides users with some control over which cutoffs are considered in the calculation of the test statistic. I have proposed an alternative algorithm for calculating mHG and XL-mHG p-values, which results in better numerical stability, and leads to significant speed-ups when enrichment is present, or when $L < N$. Furthermore, I have described lower and upper bounds for the XL-mHG p-value, and proposed an additional $\mathcal{O}(N)$-bound that is tighter than the $\mathcal{O}(1)$-bound. Finally, I have shown that when conducting a full analysis of all gene sets considered in the study by Subramanian et al., the XL-mHG test resulted in much better p-values than the KS test for the gene sets reported as enriched in that study. Importantly, my analysis was based on a sample permutation test, and therefore accounted for the dependency structure among genes.

Beyond the KS test, this work does not include a comparison of the XL-mHG test to other tests and test statistics that have been proposed for quantifying gene set enrichment. In particular, the XL-mHG was not compared to the popular “ES” test statistic employed by GSEA[4]. However, there are a number of concerns associated with the use of GSEA’s ES test statistic: First, it is not purely rank-based. Instead, it takes into account the (absolute) score associated with each gene. This means that the choice of differential expression metric can have a strong impact on whether a gene set is considered enriched or not. As there are many different metrics available for quantifying differential expression, this means that a largely subjective choice can strongly affect the conclusions of the analysis, and that users may be tempted to try different metrics and choose the most favorable result. Differential expression metrics that have been used in the literature include the t statistic, a moderated t statistic, signal-to-noise ratio, etc. According to the GSEA Manual, GSEA allows users to choose among five different metrics for categorical phenotypes. Although this effect was not demonstrated here, the impact of the specific differential expression metric used can be reduced by relying on a purely rank-based method for quantifying gene set enrichment. Second, the ES test statistic does not exhibit the “subset enrichment” characteristic, meaning that it cannot effectively ignore the exact rank or score of some genes in the gene set. Instead, the more negative the score of a gene in the gene set is, the more it will reduce the value of the test statistic and therefore significance of the enrichment. Third, the ES test statistic does not allow the direct calculation of p-values, which makes it more difficult to relate the test statistics obtained for different gene sets to one another. (The same is true for the maxmean statistic proposed by Efron and Tibshirani (2007).) The ES test statistic was mainly motivated by the lack of power of the KS test (see paragraph “Benefits of Weighting by Gene Correlation.” in the Supporting Text of Subramanian et al. (2005)). The XL-mHG test addresses this concern, while also addressing the lack of control over the type of enrichment tested for that is inherent to the mHG test. I therefore believe that the XL-mHG test should be considered an attractive alternative to the GSEA test in most supervised settings.

The analysis presented here exemplified a general strategy for choosing the X and L parameters of the XL-mHG test, as well as for combining the XL-mHG test with a sample label permutation test. L can be chosen globally so that only cutoffs that are associated with positive differential expression scores are considered in the calculation of the test statistic. X can be chosen in a gene set-specific manner, to ensure that enrichment is based on a minimum fraction of gene in the gene set (e.g., 25%), but no fewer than a certain absolute number of genes (e.g., 5). In, GO-PCA (Wagner 2015a), I have referred to these parameters as X_{frac} and X_{abs}, respectively, so that for a gene set g containing K_g genes, $X_g = \max(\{\lceil X_{frac} * K_g \rceil, X_{abs}\})$. Finally, the sample label permutation test is straightforward to implement based on the individual gene set p-values, and conducting 10,000 permutations is computationally feasible given the algorithmic efficiency of the test and the performance of modern CPUs. For the permutations, the same L parameter and gene set-specific X parameters should be used as for the unpermuted data.

The motivation for this work was to encourage the more widespread adoption of the XL-mHG test for quantifying gene set enrichment. To this end, I have provided a rigorous and transparent treatment of the statistical and algorithmic aspects of the XL-mHG test, and developed an efficient,

[3] See http://cbl-gorilla.cs.technion.ac.il/

[4] Note that GSEA does provide an option to use a standard rank-based KS test statistic instead of the score-based default statistic.

tested, free and open-source implementation in the form of the xlmhg Python/Cython package (see https://github.com/flo-compbio/xlmhg). There are multiple features that can potentially make the XL-mHG test an attractive choice in a wide range of applications: The semiparametric nature of the test, i.e., the nonparametric approach of the mHG test in combination with the X and L parameters, provide an efficient way to tailor the test to the kind of enrichment that is of interest in a particular application. In certain scenarios, the XL-mHG test is much more sensitive than the KS test, but the X parameter provides a means for trading off some of the sensitivity for increased robustness. Through its reliance on the hypergeometric distribution, the XL-mHG test also has the property that the exact distribution of 1's below n^*, the cutoff giving rise to the value of the test statistic, is not important. In other words, the test is robust to outliers, which is especially desirable when some of the 1's are expected to represent "false positives". Finally, efficient algorithms and implementations allow an individual test to be performed in only a few milliseconds, even for large values of N.

METHODS

Implementation of PVAL1 and PVAL2

The PVAL1 and PVAL2 algorithms were implemented twice, once in Python and once in Cython. The Cython programming language is a superset of Python that compiles to C code. When type declarations are added, the generated C code can avoid (slow) calls to the Python C-API, resulting in speeds comparable to that of native C programs. At the same time, results (in this case, XL-mHG p-values) can easily be passed back into Python code. The Cython implementation uses the long double variable type for all floating point operations. Most compilers implement this type using 80-bit "extended precision", with the notable exception of the Microsoft Visual C++ compiler[5]. Therefore, the Cython implementation is much faster and more accurate compared to the Python implementation. However, the Python implementation does not require compilation. By default, all implementations use a relative tolerance of 10^{-12} (see Algorithm 3), which was found to give accurate results.

Testing PVAL2 and PVAL-BOUND for correctness

Since the algorithms proposed here are not entirely trivial, it can be difficult to establish their correctness. I therefore implemented test procedures for the Cython implementations of PVAL2 and PVAL-BOUND that rely on alternative algorithms for calculating the XL-mHG p-value and the $\mathcal{O}(N)$-bound, respectively. I then compared the results of those alternative algorithms to those obtained with PVAL2 and PVAL-BOUND. I found that the results were identical for all cases tested, which led me to conclude that both algorithms are in fact correct. The tests were implemented and executed as *unit tests* within the framework provided by the pytest Python package (version 2.8.5), and are included in the xlmhg Python/Cython package, under tests/test_correct_pval.py and tests/test_correct_bound.py (see https://github.com/flo-compbio/xlmhg).

More specifically, to test the correctness of PVAL2, I chose $N = 50$ and $K = 10$, and generated a reference table of hypergeometric p-values $p_{(n,k)}$, for all possible hypergeometric configurations (i.e., for all possible n and k), using the scipy.stats.hypergeom.sf function from the scipy Python package (version 0.17.0). Then, for each possible combination of X and L ($X, L \in \{1, ..., N\}$), I used the reference table to obtain all possible values of the XL-mHG test statistic $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ (by setting $p_{(n,k)} = 1$ for all $k < X$ and $n > L$). For each value of the test statistic, I then calculated the XL-mHG p-value $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ using both PVAL1 and PVAL2, and tested whether the output of both algorithms was identical, within a margin of error due to the numerical errors discussed in the results section. Specifically, I used the IS_EQUAL algorithm with a relative tolerance of 10^{-8} to determine if the two results were identical. In total, $56,400$ such comparisons were conducted, and the p-values were found to be identical in all cases.

To test the correctness of PVAL-BOUND, I implemented another testing procedure, again choosing $N = 50$ and $K = 10$. To obtain an alternative algorithm for calculating $b^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, I designed a simpler version of PVAL-BOUND that assumes that all $p_{(n,k)}$ are already known. In addition to testing whether both algorithms returned identical values for $b^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, I also tested whether those values were in fact equal to or larger than $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, and whether in all cases $b^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ was equal to or smaller than the $\mathcal{O}(1)$-bound (i.e., $(\min\{K, L\} - X + 1)s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$). Again, a total of $56,400$ tests were conducted, and

[5] see https://en.wikipedia.org/wiki/Long_double

all tests passed. Furthermore, in $34,858$ out of the $56,400$ cases, $b^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ was found to be strictly smaller than $(\min\{K,L\} - X + 1)s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, indicating that the $\mathcal{O}(N)$-bound is indeed tighter than the $\mathcal{O}(1)$-bound.

Assessing the numerical stability of PVAL1 and PVAL2

All lists tested in Figures 4 and 6 consisted of 20 1's, followed by a varying number of 0's. Obviously, we have $n^* = 20$ for all those lists. In other words, the best cutoff for all those lists is 20, so that the "top of the list" contains all 1's and no 0's, and the mHG test statistic is the hypergeometric p-value at that cutoff. Due to the special structure of those lists, calculation of the true mHG p-value $p^{\text{mHG}}(\boldsymbol{v})$ is trivial as well. Since for given N and K, no other list exhibits an equally good minimum hypergeomtric p-value, $p^{\text{mHG}}(\boldsymbol{v})$ corresponds to $1/|\mathcal{V}^{(N,K)}| = s^{\text{mHG}}(\boldsymbol{v})$.

Benchmarks of PVAL1 and PVAL2

The benchmarks of PVAL1 and PVAL2 were carried out using the repeat function from Python's timeit module. For each randomly generated list, $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ was pre-calculated, and then the runtime of the functions get_xlmhg_pval1 and get_xlmhg_pval2 from the Cython module (xlmhg.mhg_cython) were measured. The measurements were taken for 10 identical calls of the function (number=10), and the minimum runtime over three tests (repeat=3) was recorded. To obtain the final runtime, this minimum was divided by the number of calls (10).

Power comparison between the mHG test and the KS test

For each experiment (i.e., each choice of K and n), and each fold change value f, I generated random lists as follows: First, I calculated the number of 1's within the "top of the list" (i.e., above the n'th cutoff) as $k = f * (n/N)$ (the fold enrichment values were chosen in a way that would result in integer numbers). I then used the numpy.random.choice function from the numpy Python package (version 1.10.4) to sample k ranks from $\{1, ..., n\}$ without replacement. I then set the elements at those ranks to 1. I then used the same function to sample $K - k$ ranks from $\{n+1, ..., N\}$ without replacement, and set the elements of at those ranks to 1. All elements were set to 0. I repeated this procedure 1,000 times, to generate 1,000 random lists. I then applied both the mHG test and the KS test to each list, and tested whether the p-values were equal to or smaller than 10^{-6}. For the KS test, I provided the list of cutoffs corresponding to the 1's (0-based indices, with an added continuity correction of 0.5) to the scipy.stats.kstest function from the scipy Python package (version 0.17.0), and also specified the following arguments: cdf='uniform', alternative='greater', mode='approx'.https://github.com/flo-compbio/xlmhg-paper.

Data for the p53 study by Subramanian et al. (2005)

All data used were downloaded from the GSEA "Example Datasets" website, (http://software.broadinstitute.org/gsea/datasets.jsp, which I will henceforth refer to as "GSEA website".

The gene expression dataset used was contained in the file P53_collapsed.gct (to be found under "DATASET/p53" on the GSEA website). This dataset contains 10,100 genes ("collapsed" affymetrix probes), and 50 samples (cell lines). The sample class assignments (wild-type vs. mutant) were contained in the file P53.cls (found in the same section of the GSEA website). The "C2" collection of 522 gene sets used by Subramanian et al. in their analyses was found in the file c2.symbols, to be found under the "DATASET/Gene Sets" on the GSEA website.

Enrichment analysis for the p53 study by Subramanian et al.

To rank the genes by their differential expression (with genes most highly up-regulated in wild-type vs. control first), I used the "signal-to-noise" score, which is the default score used by GSEA (see the GSEA User Manual; http://software.broadinstitute.org/gsea/doc/GSEAUserGuide.pdf). I confirmed that I obtained values identical to those calculated in GSEA by comparing my results to those provided on the GSEA website (see the link in the "Description" field under "DATASETS/p53"; the tab-delimited file containing the ranking and scores for this particular analysis was found at http://software.broadinstitute.org/gsea/resources/gsea_pnas_results/p53_C2.Gsea/ranked_gene_list_WT_versus_MUT_1130958999391.xls).

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.1962v3 | CC BY 4.0 Open Access | rec: 12 Feb 2017, publ: 12 Feb 2017

To test for gene set enrichment, I first performed XL-mHG tests and KS tests as described above. I then performed 10,000 sample label permutations. For each permutation, I recalculated the gene scores (signal-to-noise ratios) using the permuted sample labels. Then, I ranked the genes based on the new scores, and performed the XL-mHG and KS tests using the new gene ranking. To increase the computational efficiency of the procedure, only gene sets that had a nominal p-value of 0.05 of lower in the unpermuted data using either test were tested in this manner. Since nominal p-values are expected to be anticonservative, this was not likely to result in the exclusion of any enriched gene sets. For the XL-mHG tests in the permuted data, I used the same X and L values that were used in the unpermuted tests. For each gene set, the permutation p-value is equal to the fraction of permutations for which the nominal p-value of the permuted test was equal to or lower to the nominal p-value of the unpermuted test.

The gene sets reported as enriched in Table 2 of Subramanian et al. (2005) were mapped to names of gene sets in the C2 collection (see above) using the detailed analysis results provided by the authors on the GSEA website (see the link in the “Description” field under “DATASETS/p53”; the tab-delimited file containing the ranking and scores for this particular analysis was found at http://software.broadinstitute.org/gsea/resources/gsea_pnas_results/p53_C2.Gsea/gsea_report_for_WT_1130958999391.xls). Specifically, “Hypoxia and p53 in the cardiovascular system” was mapped to the “p53hypoxiaPathway” gene set, “Stress induction of HSP regulation” was mapped to the “hsp27Pathway” gene set, and “p53 signaling pathway” was mapped to “p53Pathway”. These mappings were also validated using Google queries and data from the WikiPathways website (http://wikipathways.org).

ACKNOWLEDGMENTS

I would like to thank Dr. Sandeep Dave for his support.

REFERENCES

Ackermann, Marit and Korbinian Strimmer (2009). “A general modular framework for gene set enrichment analysis”. In: *BMC bioinformatics* 10, p. 47. DOI: 10.1186/1471-2105-10-47.

Barry, William T., Andrew B. Nobel, and Fred A. Wright (2005). “Significance analysis of functional categories in gene expression studies: a structured permutation approach”. In: *Bioinformatics (Oxford, England)* 21.9, pp. 1943–1949. DOI: 10.1093/bioinformatics/bti260.

Chen, James J. et al. (2007). “Significance analysis of groups of genes in expression profiling studies”. In: *Bioinformatics (Oxford, England)* 23.16, pp. 2104–2112. DOI: 10.1093/bioinformatics/btm310.

Eden, Eran, Doron Lipson, et al. (2007). “Discovering motifs in ranked lists of DNA sequences”. In: *PLoS Computational Biology* 3.3, e39. DOI: 10.1371/journal.pcbi.0030039.

Eden, Eran, Roy Navon, et al. (2009). “GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists”. In: *BMC Bioinformatics* 10, p. 48. DOI: 10.1186/1471-2105-10-48. URL: http://www.ncbi.nlm.nih.gov/pubmed/19192299 (visited on 12/28/2011).

Efron, Bradley and Robert Tibshirani (2007). “On testing the significance of sets of genes”. In: *The Annals of Applied Statistics* 1.1, pp. 107–129. DOI: 10.1214/07-AOAS101. URL: http://projecteuclid.org/euclid.aoas/1183143731 (visited on 02/02/2017).

Irizarry, Rafael A. et al. (2009). “Gene set enrichment analysis made simple”. In: *Statistical Methods in Medical Research* 18.6, pp. 565–575. DOI: 10.1177/0962280209351908.

Maciejewski, Henryk (2014). “Gene set analysis methods: statistical models and methodological differences”. In: *Briefings in Bioinformatics* 15.4, pp. 504–518. DOI: 10.1093/bib/bbt002.

Mootha, Vamsi K. et al. (2003). “PGC-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes”. In: *Nature Genetics* 34.3, pp. 267–273. DOI: 10.1038/ng1180.

Ramsköld, Daniel et al. (2009). “An abundance of ubiquitously expressed genes revealed by tissue transcriptome sequence data”. In: *PLoS computational biology* 5.12, e1000598. DOI: 10.1371/journal.pcbi.1000598.

Subramanian, Aravind et al. (2005). “Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles”. In: *Proceedings of the National Academy of Sciences of the United States of America* 102.43, pp. 15545–15550. DOI: 10.1073/pnas.0506580102.

Wagner, Florian (2015a). “GO-PCA: An Unsupervised Method to Explore Gene Expression Data Using Prior Knowledge”. In: *PloS One* 10.11, e0143196. DOI: 10.1371/journal.pone.0143196.

— (2015b). “The XL-mHG Test For Enrichment: A Technical Report”. In: *arXiv:1507.07905 [stat]*. arXiv: 1507.07905. URL: http://arxiv.org/abs/1507.07905 (visited on 08/04/2015).

Appendices

A ALGORITHMS

A.1 Pseudocode for PVAL2

We first describe two auxiliary algorithms, IS_EQUAL and HGP, and then describe PVAL2.

Algorithm 3: IS_EQUAL— Test whether two floating point numbers should be considered equal.

```
Input: a, b, tol (relative tolerance)
Output: TRUE or FALSE
if a = b or $|a-b| \leq$ tol * MAX($|a|$, $|b|$) then
    return TRUE
else
    return FALSE
end if
```

Algorithm 4: HGP— Calculate hypergeometric p-value $p_n^{\text{HG}}(\boldsymbol{v})$ when $f(k;\, N, K, n)$ is already known.

```
Input: N, K, n, k, p=$f(k;\, N, K, n)$
Output: pval=$p_n^{\text{HG}}(\boldsymbol{v})$
pval $\leftarrow$ p
while k $<$ MIN(K, n) do
    p $\leftarrow$ p * ((n-k)*(K-k)) / ((k+1)*(N-K-n+k+1))
    pval $\leftarrow$ pval + p
    k $\leftarrow$ k + 1
end while
return pval
```

Algorithm 5: `PVAL2`— Improved algorithm to calculate $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ in $\mathcal{O}(N^2)$.

Input: N, K, X, L (X, L $\in \{1, ..., N\}$), stat=$s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, tol (relative tolerance)
Output: pval=$p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$

```
pval ← 0
W ← N-K
table ← empty $(K+1) \times (W+1)$ array of floats
table[0, 0] ← 1
p_start ← 1
pval ← = 0
for n = 1 to L do
    if K $\geq$ n then
        k = n
        p_start = p_start * (K-n+1)/(N-n+1)
    else
        k = K
        p_start = p_start * n/(n-K)
    end if
    p = p_start
    hgp = p
    w = n-k
    if k = K and (hgp > stat and not IS_EQUAL(hgp, stat, tol)) then
        // we're not in $\mathcal{R}(\boldsymbol{v})$, even though k = K
        // this means we've seen all of $\mathcal{R}(\boldsymbol{v})$, so we're done
        break
    end if
    while k $\geq$ X and w $\leq$ W and (hgp < stat or IS_EQUAL(hgp, stat, tol)) do
        // we're in $\mathcal{R}(\boldsymbol{v})$, so $\pi_{(n,k)}(\boldsymbol{v}) = 0$
        table[k, w] ← 0
        // check if this is an entry point into $\mathcal{R}(\boldsymbol{v})$ (entering is only possible "from below")
        if table[k-1, w] > 0 then
            // calculate the fraction of paths entering (only those that have never entered $\mathcal{R}(\boldsymbol{v})$ before),
            // then add that number to pval
            pval ← pval + (table[l-1, w] * (K-k+1) / (N-n+1))
        end if
        p ← p * (k*(N-K-n+k)) / ((n-k+1)*(K-k+1))
        hgp ← hgp + p
        w ← w + 1
        k ← k - 1
    end while
    // we have left $\mathcal{R}(\boldsymbol{v})$, now calculate $\pi_{(n,k)}(\boldsymbol{v})$ for the remaining configurations for cutoff n
    while k $\geq$ 0 w $\leq$ W do
        if k = 0 then
            table[k, w] ← table[k, w-1] * (W-w+1)/(N-n+1)
        else if w = 0 then
            table[k, w] ← table[k-1, w] * (K-k+1)/(N-n+1)
        else
            table[k, w] ← table[k, w-1] * (W-w+1)/(N-n+1) +
                    table[k-1, w] * (K-k+1)/(N-n+1)
        end if
        w ← w + 1
        k ← k - 1
    end while
end for
return pval
```

A.2 Pseudocode for `PVAL-BOUND`

Algorithm 6: `PVAL-BOUND`— Calculate an upper bound for the XL-mHG p-value in $\mathcal{O}(N)$.

```
Input: N, K, X, L (X, L ∈ $\{1, ..., N\}$), stat=$s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, tol (relative tolerance)
Output: $b^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$
if stat = 1 then
    return 1
else if X > K or X > L then
    return 0
end if
min_KL ← MIN(K,L)
k_min ← 0
p ← 1.0
n = 1
while (n ≤ K or (p ≤ stat or IS_EQUAL(p, stat, tol)) and n ≤ L do
    if n ≤ K then
        k ← n
        p ← p * ((K-n+1) / (N-n+1))
        if k < X or (p > stat and not IS_EQUAL(p, stat, tol)) then
            k_min ← n
        end if
    else
        k ← K
        p ← p * (n / (n-K))
    end if
    n ← n + 1
end while
if k_min = min_KL then
    // $\mathcal{R}$ is empty (this never happens for valid $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$)
    return 0
end if
k_min ← k_min + 1
if n ≤ L or (n = L+1 and p ¿ stat and not IS_EQUAL(p, stat, tol)) then
    // we left $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ at or before reaching the L'th cutoff $\implies$ $k^{\max}_{\text{X,L}}(\boldsymbol{v})$ = K
    return MIN((K-k_min+1)*stat, 1)
end if
// we did not leave $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$ — "go down the diagonal" until we step out of $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$
n ← n - 1
k ← MIN(n, K)
hgp ← p
while hgp ≤ stat or IS_EQUAL(hgp, stat, tol) do
    p ← p * ((k*(N-K-n+k)) / ((n-k+1)*(K-k+1)))
    hgp ← hgp + p
    k ← k - 1
end while
// now we left $\mathcal{R}_{\text{X,L}}(\boldsymbol{v})$
k_max ← k+1
return MIN((k_max-k_min+1)*stat, 1)
```

B REVIEW OF BOUNDS FOR THE MHG P-VALUE

In this section, I will review the bounds for the mHG p-value that were first described by Eden, Lipson, et al. (2007).

Theorem 1 (Lower bound for the mHG p-value)**.** *For any* $v \in \mathcal{V}^{(N,K)}$*,* $p^{\mathit{mHG}}(\boldsymbol{v}) \geq s^{\mathit{mHG}}(\boldsymbol{v})$*.*

Proof. Recall that V represents a list drawn uniformly at random from $\mathcal{V}^{(N,K)}$. Let $P_n^{\text{HG},0}$ be the hypergeometric p-value of V for the cutoff n. From the definition of the mHG test statistic, it follows that:

$$\begin{aligned} p^{\text{mHG}}(\boldsymbol{v}) &= \Pr\left(S^{\text{mHG},0} \leq s^{\text{mHG}}(\boldsymbol{v})\right) \\ &= \Pr\left(\bigcup_{n=1}^{N}\left(P_n^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})\right)\right) \end{aligned} \tag{1}$$

In other words, we know that $S^{\text{mHG},0} \leq s^{\text{mHG}}(\boldsymbol{v})$ whenever there exists at least one cutoff n for which $P_n^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})$. We also know that $s^{\text{mHG}}(\boldsymbol{v})$ is attained at some $n = n^*$. We therefore observe the following inequality:

$$\begin{aligned} p^{\text{mHG}}(\boldsymbol{v}) &= \Pr\left(\bigcup_{n=1}^{N}\left(P_n^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})\right)\right) \\ &\geq \Pr\left(P_{n^*}^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})\right) \end{aligned} \tag{2}$$

By definition of the hypergeometric p-value, $\Pr(P_{n^*}^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})) = s^{\text{mHG}}(\boldsymbol{v})$. The theorem therefore follows. □

Theorem 2 (Loose upper bound for the mHG p-value)**.** *For any* $v \in \mathcal{V}^{(N,K)}$*,* $p^{\mathit{mHG}}(\boldsymbol{v}) \leq N s^{\mathit{mHG}}(\boldsymbol{v})$*.*

Proof. When we apply a union bound to Equation (1), we have:

$$p^{\text{mHG}}(\boldsymbol{v}) \leq \sum_{n=1}^{N} \Pr(P_n^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})) \tag{3}$$

By definition of the hypergeometric p-value, $\Pr(P_n^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})) = s^{\text{mHG}}(\boldsymbol{v})$. The theorem then follows follows from Equation (3). □

For the proof of the next bound, we need the following monotonicity property of the mHG p-value.

Theorem 3 (Tighter upper bound for the mHG p-value; Lipson bound)**.** *For any* $\boldsymbol{v} \in \mathcal{V}^{(N,K)}$*,* $p^{\mathit{mHG}}(\boldsymbol{v}) \leq K s^{\mathit{mHG}}(\boldsymbol{v})$*.*

Proof. Given $s^{\text{mHG}}(\boldsymbol{v})$, let $\mathcal{K}^{\text{mHG}}(\boldsymbol{v})$ be the set of all k for which $p^{\text{HG}}(k;\, N, K, k) \leq s^{\text{mHG}}(\boldsymbol{v})$. We know that $\mathcal{K}^{\text{mHG}}(\boldsymbol{v})$ is not empty, since $s^{\text{mHG}}(\boldsymbol{v})$ was attained for some $k = k_{n^*}(\boldsymbol{v})$. Then, for each $k \in \mathcal{K}^{\text{mHG}}(\boldsymbol{v})$, let n_k be the largest value of n for which $p^{\text{HG}}(k;\, N, K, n) \leq s^{\text{mHG}}(\boldsymbol{v})$. This definition makes sense because of the aforementioned monotonicity property (Lemma 1). Let $P_{n_k}^{\text{HG},0}$ be the hypergeometric p-value of V for the cutoff n_k. Then we can represent $p^{\text{mHG}}(\boldsymbol{v})$ as follows:

$$\begin{aligned} p^{\text{mHG}}(\boldsymbol{v}) &= \Pr\left(S^{\text{mHG},0} \leq s^{\text{mHG}}(\boldsymbol{v})\right) \\ &= \Pr\left(\bigcup_{k \in \mathcal{K}^{\text{mHG}}(\boldsymbol{v})}\left(P_{n_k}^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})\right)\right) \end{aligned} \tag{4}$$

In other words, we have $S^{\text{mHG},0} \leq s^{\text{mHG}}(\boldsymbol{v})$ whenever the hypergeometric p-value for at least one of the n_k is equal to or smaller than $s^{\text{mHG}}(\boldsymbol{v})$. We can then apply another union bound to Equation (4):

$$p^{\text{mHG}}(\boldsymbol{v}) \leq \sum_{k \in \mathcal{K}^{\text{mHG}}(\boldsymbol{v})} \Pr(P_{n_k}^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})) \tag{5}$$

Again, by definition of the hypergeometric p-value, $\Pr(P_n^{\text{HG},0} \leq s^{\text{mHG}}(\boldsymbol{v})) = s^{\text{mHG}}(\boldsymbol{v})$. We have $|\mathcal{K}^{\text{mHG}}(\boldsymbol{v})| \leq K$. The theorem therefore follows from Equation (5).

□

C BOUNDS FOR THE XL-MHG P-VALUE

Theorem 4 (Lower bound for the XL-mHG p-value)**.** *For any $\boldsymbol{v} \in \mathcal{V}^{(N,K)}$, $p^{\textit{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) \geq s^{\textit{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$.*

Proof. In the trivial case $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) = 1$, we have $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) = 1$. In the remainder, we therefore treat the case $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) < 1$. Let $s_n(\boldsymbol{v};\, X, L)$ represent the value "contributed" by the n'th cutoff in the calculation of the XL-mHG test statistic:

$$s_n(\boldsymbol{v};\, X, L) = \begin{cases} p^{\text{HG}}_n(\boldsymbol{v}), & \text{if } k_n(\boldsymbol{v}) \geq X\text{and } n \leq L, \\ 1.0 & \text{otherwise} \end{cases}$$

Furthermore, let the random variable S^0_n represent the value of $s_n(\boldsymbol{v};\, X, L)$ for a list drawn uniformly at random from $\mathcal{V}^{(N,K)}$. We then have:

$$\begin{aligned} p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) &= \Pr\left(S^{\text{XL-mHG},0}_{\text{X,L}} \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})\right) \\ &= \Pr\left(\bigcup_{n=1}^{N} \left(S^0_n \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})\right)\right) \end{aligned}$$

Furthermore, we know that since $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) < 1$, the test statistic was attained at some n^*; i.e., $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) = s_{n^*}(\boldsymbol{v};\, X, L)$. Therefore, we have:

$$\Pr\left(\bigcup_{n=1}^{N} \left(S^0_n \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})\right)\right) \geq Pr(S^0_{n^*} \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}))$$

Since $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ was attained at n^*, we know that $n^* \leq L$. Furthermore, we know that $k_{n^*}(\boldsymbol{v}) \geq X$ and that $p^{\text{HG}}(k;\, N, K, n^*) > s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ for any $k < k_{n^*}(\boldsymbol{v})$ (hypergeometric p-values strictly increase with smaller k). Therefore, we have $\Pr(S^0_{n^*} \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})) = \Pr(P^{\text{HG},0}_{n^*} \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})) = s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$, and therefore $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) \geq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$.

□

Theorem 5 (Upper bound for the XL-mHG p-value)**.** *For any $\boldsymbol{v} \in \mathcal{V}^{(N,K)}$, $p^{\textit{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) \leq (\min\{K, L\} - X + 1) s^{\textit{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$.*

Proof. In the trivial case $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) = 1$, we have $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) = 1$. In the remainder, we therefore treat the case $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) < 1$. Let $\mathcal{K}^{\text{mHG}}_{\text{X,L}}(\boldsymbol{v})$ be defined as follows:

$$\mathcal{K}^{\text{mHG}}_{\text{X,L}}(\boldsymbol{v}) = \{k : p^{\text{HG}}(k;\, N, K, k) \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}), k \geq X, k \leq L\}$$

Since $s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) < 1$, the test statistic was attained at some cutoff n^*, for some $k = k_{n^*}(\boldsymbol{v})$:

$$p^{\text{HG}}(k_{n^*}(\boldsymbol{v});\, N, K, n^*) = s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$$

Since $k_{n^*}(\boldsymbol{v}) \leq n^*$, we can rely on Lemma 1 to infer that $k_{n^*}(\boldsymbol{v}) \in \mathcal{K}^{\text{mHG}}(\boldsymbol{v})$, so $\mathcal{K}^{\text{mHG}}_{\text{X,L}}(\boldsymbol{v})$ is not empty. We define n_k for all $k \in \mathcal{K}^{\text{mHG}}_{\text{X,L}}(\boldsymbol{v})$ as in the proof for Theorem 3 (see Appendix B), and then define and $n'_k = \min\{n_k, L\}$ for all n_k. We can then represent $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$ as:

$$\begin{aligned} p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) &= \Pr\left(S^{\text{XL-mHG},0}_{\text{X,L}} \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})\right) \\ &= \Pr\left(\bigcup_{k \in \mathcal{K}^{\text{mHG}}_{\text{X,L}}(\boldsymbol{v})} \left(P^{\text{HG},0}_{n'_k} \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})\right)\right) \end{aligned} \tag{6}$$

We apply a union bound to Equation (6) and observe, as in Theorem 3 (see Appendix B), that $\Pr(P^{\text{HG},0}_{n'_k} \leq s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})) = s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$. We have $|\mathcal{K}^{\text{mHG}}_{\text{X,L}}(\boldsymbol{v})| \leq \min\{K, L\} - X + 1$ events in the union, which means that $p^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v}) \leq (\min\{K, L\} - X + 1) s^{\text{XL-mHG}}_{\text{X,L}}(\boldsymbol{v})$.

□