RssGsc: Rank Sum Statistics for Gene Set Collections

Therotical background

Pablo Cingolani

1. INTRODUCTION

Usually, as a result of a high-throughput experiment, we want interpret the biological meaning of large amounts of data, e.g. in the previous chapters we had methylation information for all the probes in a microarray from an MeDIP experiment. In order to understand the meaning of these large genomic data sets, there are several efforts that map genes into "biologically meaningful information". The most well known are: *Gene ontology* (GO)[16] that provides a controlled vocabulary to describe gene and gene product attributes in many organisms, *Kyoto Encyclopedia of Genes and Genomes* (KEGG) [10] a collection enzymatic pathways and *Molecular Signatures Database* (MSigDB), a collection of gene sets for use with GSEA software [15].

In the **Gene Ontology** project, genes are mapped to one or more nodes in a graph, called *GO-terms*. These GO-terms are part of a directed acyclic graph (DAG), a *tree-like* hierarchical structure (see Figure 1). In a DAG, parent nodes contain all child nodes but, unlike in a tree, a node in a DAG can have more than one parent. Furthermore, one gene can have many GO-terms associated. There are three DAGs, called ontologies, where GO-terms are annotated using a controled vocabulary that defines the function of the genes it contains. The onlogies are *molecular function*, *biological processes* and *cellular components*. In our methods, we do not make explicit usage of this DAG structure, so the methodologies can be applied to other information sources (not just GO).

The problem is to identify biologically meaningful gene sets from our experimental data. In the following section we explain how this can be done and discuss the problems of current methodologies.

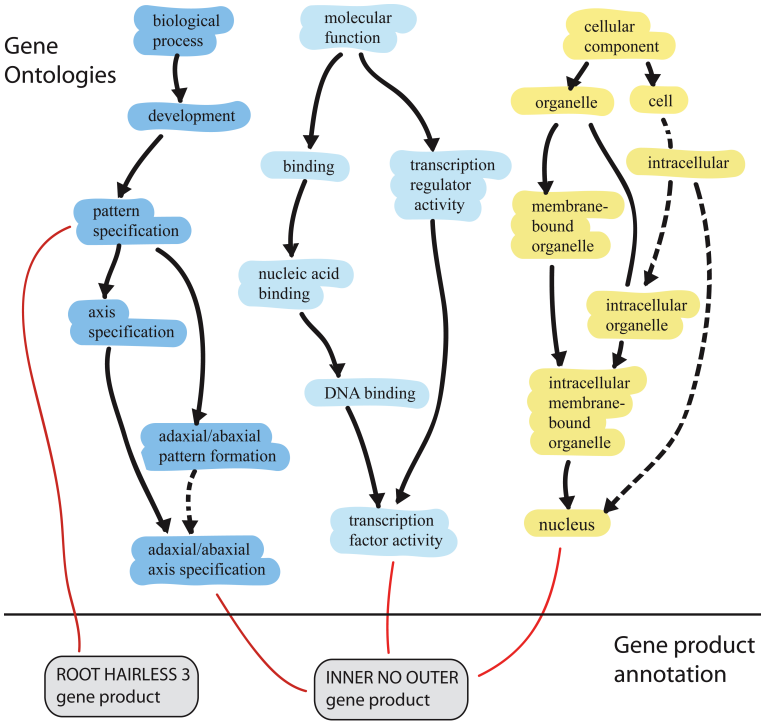

Figure 1: GO structure: A directed acyclic graph (DAG) for each ontology. Figure from [16]

2. PREVIOUS WORK

Biologically meaningful gene sets (for example from GO or KEGG) are usually identified using the following steps [14, 4, 6, 8, 13, 9, 11, 5, 12]: **i)** genes are ordered using experimental data, **ii)** a threshold is defined and genes above that threshold are marked as *"interesting"*, **iii)** a significance of the overlap between each gene set and the set of interesting genes is calculated, **iv)** multiple testing correction is applied [2] and **v)** gene sets are ranked by the corrected significance. The algorithm is shown in table 1.

Table 1: Algorithm for ranking gene sets using p-values

Problem:	Extract biologically meaningfull information from results of a high-throughput experiment.
Input:	$\mathcal{G} = \{g_1, g_2, ..., g_N\}$: A gene set $\mathcal{V} = \{v_1, v_2, ..., v_N\}$: Experimental results for each gene. t_h : A threshold $\Theta = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_H\}$: A collection of gene sets (e.g. GO-terms)
Output:	Gene sets ranked by p-value
Assumptions:	Very low p-values indicate a gene set is "meaningful", so biological interpretation of low p-value gene sets should "explain the data"
Algorithm:	
	Calculate a set of "interesting genes" $\mathcal{I} \subset \mathcal{G}$
	$g_i \in \mathcal{I} \Leftrightarrow v_i >= t_h$
	For each $\mathcal{T} \in \Theta$
	Calculate p-value
	$N = \lvert\mathcal{G}\rvert$: Number of genes in the experiment
	$n = \lvert\mathcal{I}\rvert$: Number of "interesting" genes
	$N_T = \lvert\mathcal{T}\rvert$: Number of genes in $\mathcal{T}$
	$n_T = \lvert\mathcal{I} \cap \mathcal{T}\rvert$: Number of "interesting" genes in $\mathcal{T}$
	p-value $= F(n_T; N, n, N_T) = \sum_{k=n_T}^{n} f(k; N, n, N_T) = \sum_{k=n_T}^{n} \frac{\binom{n}{k}\binom{N-n}{N_T-k}}{\binom{N}{N_T}}$
	Perform multiple testing correction
	Rank gene sets in Θ by p-value

In our experimental data we have a set of N genes, n of them are "interesting genes". We want to analyze a GO-term $\mathcal{T}$ containing N_T genes and n_T of them are "interesting" (see Figure 2). Intuitively we can explain how to calculate p-values by using the following analogy: assume we have an urn with 100 marbles (N genes), 30 of them are red (n interesting genes) and 70 are white. We draw 10 marbles (GO-term $\mathcal{T}$ containing N_T genes), the probability of having drawn 6 red marbles (n_T) is calculated using a hypergeometric distribution [7] $f(n_T; N, n, N_T) = \frac{\binom{n}{n_T}\binom{N-n}{N_T-n_T}}{\binom{N}{N_T}}$. The probability of having drawn 6 or more marbles is calculated using Fisher's exact tests, that is a sum of hypergeometric probabilities [7]:

$$F(n_T; N, n, N_T) = \sum_{k=n_T}^{n} f(k; N, n, N_T) = \sum_{k=n_T}^{n} \frac{\binom{n}{k}\binom{N-n}{N_T-k}}{\binom{N}{N_T}}$$

As Fisher's exact tests is hard to compute, a Chi-square approximation or a z-score is often used [6], [13].

There are some difficulties with this approach (the algorithm in Table 1): *i)* the number of significant gene sets might be too large to manually interpret, and *ii)* gene set significance depends on an arbitrary threshold.

Motivation: We mentioned that what most current algorithms do is to calculate p-values for every gene set and produce p-value ordered collection of all gene sets. We would like to extract the most meaningful gene sets. We attack problem i by comparing collections of gene sets. Given two collections of gene sets $\Theta_1 = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$ and $\Theta_2 = \{\mathcal{T}_4, \mathcal{T}_5, \mathcal{T}_6, \mathcal{T}_7\}$ we need a way to decide if Θ_1 is more informative than Θ_2. We propose a solution using mutual information [1].

3. RANKED LIST

Usually we have set of genes $\mathcal{G} = \{g_1, ..., g_N\}$, a set $\mathcal{V} = \{v_1, ..., v_N\}$ of experimental results for each gene and a subset of "interesting" genes $\mathcal{I} \subset \mathcal{G}$. To select these interesting genes, a threshold t_h is defined and all genes

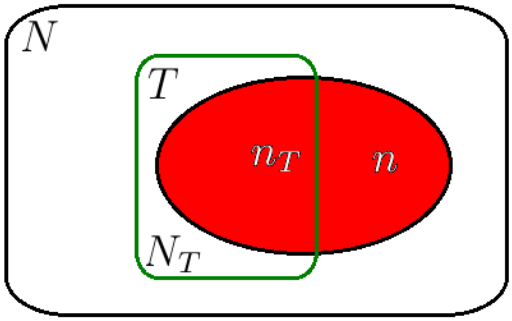

Figure 2: In an experiment analyzing N genes, n are "interesting" (red). Gene set $\mathcal{T}$ (show in green) has N_T genes, $n_T i$ are interesting.

g_i having values $v_i \geq t_h$ are considered "interesting". There is no exact way to define the threshold, this was mentioned in section 2 as problem *ii* and the methods we mentioned so far also have this problem.

In order to avoid this problem, we will use rank statistics. There are algorithms that use some type of rank statistics, the most well know is GSEA [15], [3], that performs brownian bridge statistics. In this section we will use rank sum statistics, will assume genes are ranked by their experimental values, i.e. we sort genes by V and we assign a rank $r_i \in \{1, ..., N\}$. So all genes are ordered and assigned a rank in the list:

Rank	*Gene*
1	g_{i_1}
2	g_{i_2}
3	g_{i_3}
$\vdots$	$\vdots$
N	g_{i_N}

Then we define the rank sum as $R = \sum_{i=1}^{k} r_i$. There is a relatioship between the ranked list and the set of "interesting" genes $\mathcal{I}$. Genes are "interesting" if they are high in the ranking (i.e. above a threshold). This means that the mean ranking of genes in $\mathcal{I}$ is less than r_{mean} (a rank mean value threshold) or the rank sum of genes in $\mathcal{I}$ is less than r_{sum} (a rank sum value threshold). For instance if there are $N = 1000$ genes in the ranked list and we say that the first 100 are the "interesting" genes, this is equivalent to saying that the average rank $r_{mean} \leq 50.5$, or that the rank sum is $r_{sum} \leq 101 * 50 = 5050$. So for a set of genes $\mathcal{T}$, the probability of a gene being interesting $P(G \in \mathcal{I} | G \in \mathcal{T})$ is analogous to the rank sum cumulative probability for the genes in $\mathcal{T}$, $P(R \leq r_{sum} | \mathcal{T})$.

Probaility density function for a random variable that represents a rank sum is derived in appendix B. As calculating this probabilies is computationally intensive, we also derive formulas to approximate these functions.

3.1. Simulations and results

We performed simulations comparing two simple algorithms, one using p-values (select gene set with lower p-value) and other that creates sets using greedy strategy (at each iteration we keep the set that, when incorporated to the current set, has the lowest p-value). Table 2 shows the criteria used for gene selection, and table 3 show mean recovery rate for both algorithms. Table 4 shows a comparisson with GSEA, in our simulations, our algorithm performed better.

If we have a set of genes ranked from 1 to N, we randomly select N_T genes and add the ranks, we obtain a "rank sum" (equation 1). In this chapter we will explain the mathematical details calculating the probability dentity function of rank sum.

$$r_{sum} = \sum_{g_i \in T} r_i \tag{1}$$

Table 2: Methodology for selecting gene sets and genes for our simulation

Select gene sets:

Initialize:

Number of gene sets to select: s
Symbol (gene) selection probability: p
Noise to signal ratio: ns
A set of genes (empty): $\mathcal{G} = \emptyset$
Experimental values (empty): $\mathcal{V} = \emptyset$

Select gene sets:

Randomly select s gene sets: $\Theta_{ori} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_s\}$

Select genes:

For each gene set $\mathcal{T} \in \Theta_{ori}$

Randomly select genes $g \in \mathcal{T}$ with a probability p
Add each selected gene g to set $\mathcal{G} = \mathcal{G} \cup \{g\}$
Assign an exprimental value $v \sim \beta(\beta_1, \beta_2)$ using beta distribution
Add corresponding values v to set $\mathcal{V} = \mathcal{V} \cup \{v\}$

Add 'noise':

Calculate the number of "noise" genes to add (number of genes in $\mathcal{I}$ times noise to signal ratio)
$ng = ns\ |\mathcal{G}|$
Randomly select ng genes that do not belong to $\mathcal{G}$, add those genes to $\mathcal{G}$.

Apply algorithms:

Assign exprimental values $v \sim U(0, 1)$ using uniform distribution to genes recently added to $\mathcal{G}$.

Rank genes:

Sort genes by experimental value v and assign ranks

Apply algorithms:

For each algorithm { *"p-value"*, *"p-value Greedy"*, *"GSEA"*}

$\hat{\Theta}$ = Apply algorithm selecting best s gene sets
Recovery rate $rr_{algorithm} = |\hat{\Theta} \cap \Theta_{ori}|/|\Theta_{ori}|$

Table 3: Algorithm comparison: Mean recovery rate (and standard deviation) for different number of gene sets "s" (based on 38000 simulations, gene selection probability $p \in [0.1, ..., 0.9]$, noise to signal ratio $ns \in [100\%, ..., 2000\%]$

Number of gene sets: S	**"RankSum"**	**"Greedy RankSum"**
1	0.30 ±0.46	0.32 ±0.46
4	1.06 ±0.61	1.60 ±1.12
7	1.64 ±0.80	3.07 ±1.59
10	2.22 ±1.02	4.65 ±1.98
13	2.78 ±1.20	6.26 ±2.39
16	3.31 ±1.33	7.93 ±2.71
19	3.81 ±1.45	9.47 ±3.04
22	4.36 ±1.57	11.06 ±3.25
25	4.89 ±1.67	12.73 ±3.64
28	5.37 ±1.80	14.09 ±3.91

Table 4: Algorithm comparison with GSEA. Mean recovery rate and standard deviation, based on 300 simulations, p-value is less than 2.2×10^{-16}.

Number of gene sets: S	***"GSEA"***	***"Greedy RankSum"***
10	17.4% ±9.9%	39.7% ±18.3%
Selection Probability: p		
30.0 %	16.8% ±10%	36.6% ±16%
50.0 %	13.2% ±8%	29.1% ±15%
70.0 %	22.4% ±9%	53.5% ±14%
Noise to signal: ns		
100.0 %	13.2% ±8%	29.1% ±15%
500.0 %	19.6% ±10%	45.1% ±17%

It is usefull to keep in mind th following analogy: We have an urn filled with balls with numbers from 1 to N, we randomly draw N_T balls and add the numbers. There are two different cases: *i)* each time we draw a ball, we replace it back in the urn or *ii)* we do not replace it. The probability density function for the first case is easier to deduct, so we will assuming no replacement. Then we will address tha case when there is no replacement, as the deduction is similar.

4. RANKED SUM WITH REPLACEMENT

In these sections we will calculate the probability density functions of a rank sum. We'll assume that genes are ranked and that ranks can be repeated (e.g. gene g_i is ranked r_i and gene g_j can also be ranked r_i), this is equivalent to say that ranks are drawn with replacement. As an intuitive way of looking at this, assume we have all the ranks written in small pieces of paper in a bag, we get one piece of paper, read it and place it back in the bag. It's also important to note that we assume that all ranking positions are used, which means that all values from 1 to N are assigned to at least one gene.

4.1. Approximation by normal distribution

If we are analyzing a set of genes T which has N_T genes, we calculate the rank sum as

where r_i is gene's g_i rank. This is a sum of random variables that converges to a normal distribution when the number of genes in T tends to infinity. The expectation for r_i is $(N+1)/2$ and the variance is $(N^2-1)/12$ (these are the mean and varance of uniform a distribution between 1 and N). Then the mean for r_{mean} is

$$E[r_{sum}] = E\left[\sum_{gi \in T} r_i\right] = \sum_{gi \in T} E\left[r_i\right] = N_T \; E\left[r_i\right] = N_T \frac{(N+1)}{2}$$

and the variance is

$$Var(r_{sum}) = Var\left[\sum_{gi \in T} r_i\right] = \sum_{gi \in T} Var\left[r_i\right] = N_T \frac{N^2-1}{12}$$

so we can approximate r_{mean} by a normal distribution

$$\mathcal{N}\left\{\mu = N_T \frac{N+1}{2}, \sigma^2 = N_T \frac{N^2-1}{12}\right\}$$

Now, we can calculate $P(R \leq r_{sum})$ by using the normal cumulative distribution. However, there are several probelms with this approximation, specially when the number of terms in the set that we want to analyze is small (e.g. less than 20). In the following section we show how to calculate the exact value of this probability density function.

4.2. Exact calculation

Now we calculate the exact value of a rank sum probability. In order to do this, we will assume that there can be repeated rank values (e.g. two genes g_i and g_j can share rank $r_i = r_j$ in a list).

We start considering the trivial case when the rank sum has only one term (i.e. $N_T = 1$ and $T = \{r_1\}$). In this case, the probability is

$$P_{N,1}(R = r_{sum}) = \begin{cases} 1/N & if\ (1 \leq R \leq N)\ and\ (N > 0) \\ 0 & otherwise \end{cases}$$

because the rank distribution is uniform between 1 and N (only for valid values of R and N, of course).

Now let's consider the case when the rank sum is composed of two or more terms (i.e. $N_T \geq 2$ and $T = \{r_1, r_2, r_3, ..., r_{N_T}\}$). To calculate the probability that the two or more ranks add to R, we have to add all possible combinations such that $r_1 + r_2 + ... + r_{N_T} = R$. This is the same as the probability of $r_1 = r$ and $r_2 + r_3 + ... + r_{N_T} = R - r$ for all possible values of r, the formula is:

$$P_{N,N_T}(R = r_{sum}) = \sum_{r=1}^{N} P_{N,1}(r)\ P_{N,N_T-1}(R - r) \tag{2}$$

This formula can be optimized by noting that the maximum value for r in this sum is either N or $R - N_T + 1$. This is because $P_{N,1}(r)$ is non-zero for values of $r \in [1, N]$. Likewise, $P_{N,N_T-1}(r)$ is zero when $r < N_T$ (i.e. there is no way to add N_T ranks to be less than N_T because the minumum possible rank is 1), then

$$P_{N,N_T-1}(r < N_t) = 0$$

$$P_{N,N_T-1}(r \leq N_t - 1) = 0$$

$$P_{N,N_T-1}(-r \geq -N_t + 1) = 0$$

$$P_{N,N_T-1}(R - r \geq R - N_t + 1) = 0$$

so $P_{N,N_T-1}(R - r)$ is zero when $R - r \geq R - N_t + 1$, which means we can rewrite the limits of the sum in equation 2 as

$$P_{N,N_T}(R = r_{sum}, N) = \sum_{r=1}^{r_{max}} P_{N,1}(r, N)\ P_{N_T-1}(R - r, N) \tag{3}$$

where $r_{max} = min(R - N_T + 1, N)$.

Figure 4 shows some calculated, simulated and normally approximated probability density functions for different N and N_T. Root mean squared error (RMS) between exact calculation and normal aproximation is shown in Figure 5, as we can see the error is very small when either N or N_T are over 20, so this is the criteria we use to decide when to use normal approximation.

4.3. Fast algorithm

Looking at equation 2 we can see that probabilities are zero outside the limits of sumation, so we can extend those limits to $\pm\infty$

$$P_{N,N_T}(R = r_{sum}) = \sum_{r=1}^{N} P_{N,1}(r)\ P_{N,N_T-1}(R - r) = \sum_{r=-\infty}^{+\infty} P_{N,1}(r)\ P_{N,N_T-1}(R - r)$$

$$= [P_{N,1}\ *\ P_{N,N_T-1}]_{(R)}$$

so we can express this probability as the convolution of the two probabilities, evaluated at R, expanding the term P_{N,N_T-1}

$$P_{N,N_T}(R = r_{sum}) = [P_{N,1}\ *\ P_{N,1}\ *\ P_{N,N_T-2}]_{(R)} = [P_{N,1}\ *\ P_{N,1}\ *\ \cdots\ * P_{N,N_T-(N_T-1)}]_{(R)}$$

this is a convolution of N_T functions. If we apply the Fourier transform on both sides $\mathcal{F}[P_{N,N_T}] = \mathcal{F}[P_{N,1}]^{N_T}$ $\Rightarrow P_{N,N_T}(R = r_{sum}) = \mathcal{F}^{-1}\{\mathcal{F}[P_{N,1}]^{N_T}\}_{(R)}$. If we calculate the probability density function this way, reduce the complexity of the computation to $O[N_T\ Nlog(N_T\ N)]$, which is the complexity of the Fast Fourier transform.

5. RANKED SUM WITHOUT REPLACEMENT

In these sections we will calculate the probability density functions of a rank sum when there is no placement (i.e. once a gene is assigned a rank, no other gene can have the same rank). As an intuitive way of looking at this, assume we have all the ranks written in small pieces of paper in a bag, we get one piece of paper, read it and throw it away (we don not place it back in the bag), adding all the numbers we've read we get the rank sum. As before, we assume that all ranking positions are used, which means that each value from 1 to N is assigned to one gene.

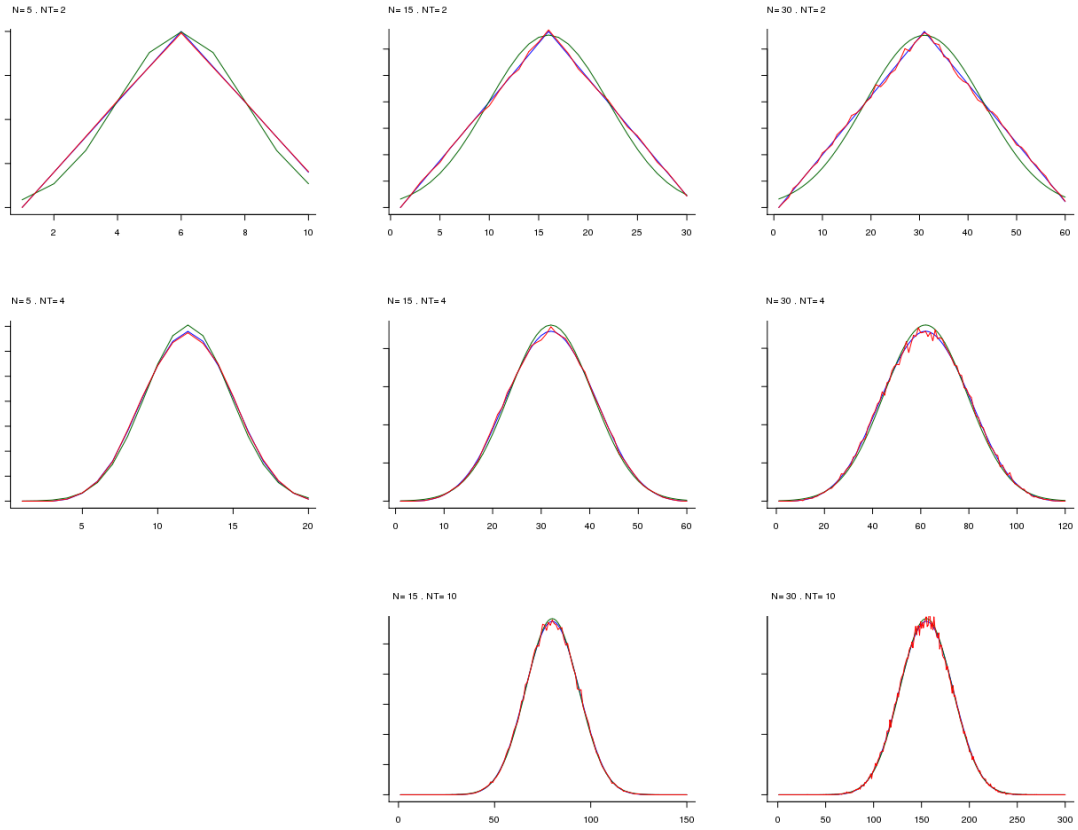

Figure 3: Probability density function $P_{N,N_T}(R)$ shown in blue, simulated values (red) and normal approximation (green).

5.1. Min / Max values

Before calculating the probability density function, we need to know what are the minimum and maximum possible values for a rank sum without replacement. Let's calculate the minimum value for a rank of N terms if we select N_T terms. Clearly the minimum possible value is

$$R_{min}(N, N_T) = \sum_{i=1}^{N_T} i = (N_T + 1)\frac{N_T}{2}$$

We'd like to calculate the minimum possible rank sum when selecting N_T items that have been ranked from r_{min} to N (note that the minimum rank is not 1):

$$R_{min}(N, N_T, r_{min}) = \sum_{i=r_{min}}^{r_{min}+N_T-1} i = \sum_{j=1}^{N_T} (j + r_{min} - 1) = N_T(r_{min} - 1) + \sum_{j=1}^{N_T} j$$

the last term is $R_{min}(N, N_T) = R_{min}(N, N_T, 1)$, so

$$R_{min}(N, N_T, r_{min}) = N_T(r_{min} - 1) + R_{min}(N, N_T) \tag{4}$$

Now the maximum possible when we select N_T items ranks form 1 to N:

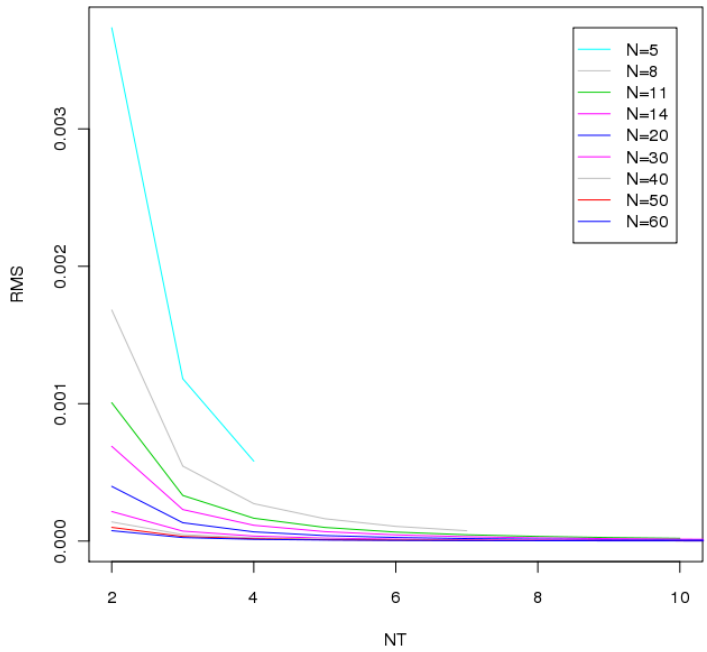

Figure 4: Normal approximation's RMS error for different N and N_T values.

$$R_{max}(N, N_T) = \sum_{i=N-N_T+1}^{N} i$$

Changing variables $i = N - (N_T - j)$:

$$R_{max}(N, N_T) = \sum_{j=1}^{N_T} N - (N_T - j) = \sum_{j=1}^{N_T} N - N_T + \sum_{j=1}^{N_T} j = \sum_{j=1}^{N_T} N - N_T + R_{min}(N, N_T)$$
$$= N_T(N - N_T) + R_{min}$$
$$R_{max}(N, N_T) = N_T(N - N_T) + R_{min} \tag{5}$$

5.2. Exact calculation

In this section we'll find the probability density function for a ranked sum when there are no repeated ranks (i.e. no replacement). The only difference between this section and section 4.2 is that we assume that for any two different genes g_i and g_j the ranks are different (i.e. $r_i \neq r_j \ \forall g_i \neq g_j$). Although the formulas and the distribution shapes are different, the methodology to derive the formulas almost the same.

We start considering the case when the rank sum has only one term (i.e. $N_T = 1$ and $T = \{r_1\}$). We'll add two parameters: N_{out} to specify how many ranks have already been drawn (i.e. we cannot use those ranks) and r_{min} to specify the minimum value to consider in the rank sum (the use of this variable will became evident later). In this case, the probability is uniform, we can only select 1 out of $N - N_{out}$ ranks:

$$P_{N,1}(R = r_{sum}|N_{out}) = \begin{cases} 1/(N - N_{out}) & if\ (N > 0)\ and\ (1 \leq R \leq N) \\ 0 & otherwise \end{cases}$$

Now we define

$$Q_{N,N_T}(R|N_{out}, r_{min}) = P_{N,N_T}(R|N_{out})\ \delta_{R<r_{min}}$$

where

$$\delta_{R<r_{min}} = \begin{cases} 1 & if\ R < r_{min} \\ 0 & otherwise \end{cases}$$

When the rank sum is composed of two or more terms (i.e. $N_T \geq 2$ and $T = \{r_1, ..., r_{N_T}\}$), as we did before, we add all possible combinations such that $r_1 + ... + r_{N_T} = R$. Which is the same as the probability of $r_1 = r$ and $r_2 + ... + r_{N_T} = R - r$ for all possible values of r. This is the same as doing the calculation for $r_1 < (r_2 + ... + r_{N_T})$ and then multipliying by all posible combinations. We use the parameter r_{min} to indicate that $r_2 + ... + r_{N_T}$ cannot be less or equal to r_1. So the formula is:

$$Q_{N,N_T}(R = r_{sum}|N_{out}, r_{min}) = N_T \sum_{r=r_{min}}^{N} Q_{N,1}(r|N_{out}, r)\ Q_{N,N_T-1}(R - r|N_{out} + 1, r + 1)$$

This is recursive formula that depends on five different parameters, so it's important to narrow down the recursion as much as possible. In order to reduce our search space, we can use equations 4 and 5 which tell us the minimum and macimum possible values for R. The final equation becomes

$$Q_{N,N_T}(R = r_{sum}|N_{out}, r_{min}) = \begin{cases} N_T \sum_{r=1}^{N} Q_{N,1}(r|N_{out}, r)\ Q_{N,N_T-1}(R - r|N_{out} + 1, r + 1) \\ \qquad If\ R_{min}(N, N_T, R_{min}) \leq R \leq R_{max}(N, N_T) \\ 0 \qquad\qquad Otherwise \end{cases}$$

So if we want to calculate the probability of having an $R = r_{sum}$ when we draw N_T numbers from a ranked list of N numbers, we just need to calculate

$$P_{N,N_T}(R = r_{sum}) = Q_{N,N_T}(R = r_{sum}|N_{out} = 0, r_{min} = 1)$$

5.3. Normal approximation

In order to approximate this function using a Gaussian probability density function, we need to calculate the mean and variance.

Mean: Now we want to calculate the mean of a rank sum when there is no replacement. We'll start by writing the definition of a rank (see equation 1) sum as

$$r_{sum} = \sum_{gi \in T} r_i = \sum_{r=1}^{N} r\ I(r) \tag{6}$$

where $I(r)$ is the indicator function that is 1 when a gene that has rank r belongs to set T. The mean value is

$$E[r_{rum}] = \sum_{r=1}^{N} r\ E[I(r)]$$

We know that there are N_T items in T, so the expected value of the indicator function is $E[I(r)] = N_T/N$ (i.e. the set T consist of N_T items choosen from N possible ones), then:

$$E[r_{rum}] = \frac{N_T}{N} \sum_{r=1}^{N} r = \frac{N_T}{N}\ \frac{(N+1)N}{2} = N_T \frac{(N+1)}{2} \tag{7}$$

This is the same value than mean rank sum *with* replacement (see section 4.1).

Variance: Now we want to calculate the variance of a rank sum when there is no replacement. We will start by using equation 6

$$Var(r_{sum}) = E[r_{sum}^2] - E[r_{sum}]^2 = E\left[\left(\sum_{gi \in T} r_i\right)^2\right] - E[r_{sum}]^2 = E\left[\left(\sum_{r=1}^{N} r\ I(r)\right)^2\right] - E[r_{sum}]^2$$

Using equation 7 we know that $\mu = E[r_{sum}] = N_T \frac{(N+1)}{2}$

$$Var[r_{sum}] = E\left[\left(\sum_{r=1}^{N} r\ I(r)\right)^2\right] - \mu^2 = \sum_{r=1}^{N}\sum_{r'=1}^{N} r\ r'\ E[I(r)\ I(r')] - \mu^2 \tag{8}$$

Now we need to calculate $E[I(r)\ I(r')]$. When $r = r'$ then this is just the probability of picking gene ranked r when choosing N_T out of N genes, this is

$$E[I(r)\ I(r)] = E[I(r)] = \frac{N_T}{N} \tag{9}$$

When $r \neq r'$ then we can calculate $E[I(r)\ I(r')]$ as the probability of choosing N_T out of N values where two of them are 'fixed' (one is r and the other is r'), that is

$$E[I(r)\ I(r')] = \frac{\binom{N-2}{N_T-2}}{\binom{N}{N_T}} = \frac{N_T(N_T-1)}{N(N-1)} \tag{10}$$

Using equations 9 and 10 in equation 8

$$Var(r_{sum}) = \sum_{r=1}^{N}\sum_{r'=1}^{N} r\ r'\ E[I(r)\ I(r')] - \mu^2$$

$$= \sum_{r=1}^{N}\sum_{r' \neq r} r\ r'\ E[I(r)\ I(r')] + \sum_{r=1}^{N}\sum_{r'=r} r\ r'\ E[I(r)\ I(r')] - \mu^2$$

$$= \frac{N_T(N_T-1)}{N(N-1)}\sum_{r=1}^{N}\sum_{r' \neq r} r\ r' + \frac{N_T}{N}\sum_{r=1}^{N} r^2 - \mu^2 \tag{11}$$

It is known that $\sum_{r=1}^{N} r^2 = (N+1)(2N+1)N/6$, we'll call this K_r, so

$$K_r = \sum_{r=1}^{N} r^2 = (N+1)(2N+1)N/6 \tag{12}$$

replacing 12 in 11

$$Var(r_{sum}) = \frac{N_T(N_T-1)}{N(N-1)}\sum_{r=1}^{N}\sum_{r' \neq r} r\ r' + \frac{N_T}{N}K_r - \mu^2$$

Now we need to calculate $\sum_{r=1}^{N}\sum_{r' \neq r} r\ r'$, we'll call that term $K_{rr'}$

$$K_{rr'} = \sum_{r=1}^{N}\sum_{r' \neq r} r\ r' = \sum_{r=1}^{N}\sum_{r'=1}^{N} r\ r' - \sum_{r=1}^{N}\sum_{r'=r} r\ r'$$

$$= \sum_{r=1}^{N}\left[r\sum_{r'=1}^{N} r'\right] - \sum_{r=1}^{N} r^2$$

$$= \sum_{r=1}^{N} \left[r \sum_{r'=1}^{N} r' \right] - K_r = \left[N \frac{(N+1)}{2} \right]^2 - K_r \tag{13}$$

replacing this latest result into 11 we get

$$Var(r_{sum}) = \frac{N_T(N_T-1)}{N(N-1)} K_{rr'} + \frac{N_T}{N} K_r - \mu^2 \tag{14}$$

where μ, K_r and $K_{rr'}$ are defined in equations 7 12 and 13 respectively (these terms depend only on N and N_T).

5.4. Approximation

Calculating the probability of a rank sum without replacement means calculating the recursive formula 6 which is computationally expensive even for small N. As we can see from Figure 6 the shape of the distribution becomes similar to a Gaussian as N increases, but even for large N a Gaussian approximation cannot be used when $N_T \in \{1, 2, N-2, N-1\}$ because in these cases the distributions are always uniform ($N_T = 1$ and $N_T = N-1$) or triangular ($N_T = 2$ and $N_T = N-2$). A summary of how to approximate the rank sum distribution is shown in table 5.

Table 5: Rank sum approximation

If $N_T \in \{1, N-1\}$
$R_{min}(N, N_T, r_{min}) = N_T(r_{min} - 1) + R_{min}(N, N_T)$
$R_{max}(N, N_T) = N_T(N - N_T) + R_{min}$
Uniform distribution $[R_{min}, R_{max}]$
If $N_T \in \{2, N-2\}$
$R_{min}(N, N_T, r_{min}) = N_T(r_{min} - 1) + R_{min}(N, N_T)$
$\mu = N_T \frac{(N+1)}{2}$
$R_{max}(N, N_T) = N_T(N - N_T) + R_{min}$
Triangular distribution $[R_{min}, \mu, R_{max}]$
If $N > 30$ and $N_T \notin \{1, 2, N-2, N-1\}$
$\mu = N_T \frac{(N+1)}{2}$
$K_{rr'} = \left[\frac{N}{2}(N+1)\right]^2 - K_r$
$K_r = (N+1)(2N+1)N/6$
$\sigma = \sqrt{\frac{N_T(N_T-1)}{N(N-1)} K_{rr'} + \frac{N_T}{N} K_r - \mu^2}$
Gaussian distribution $\mathcal{N}(\mu, \sigma)$
Otherwise: Should not approximate, use exact formula 6.

6. REFERENCES

[1] Robert B. Ash. Information theory. *Dover*, 1990.

[2] Adrian Alexa et. al. Improved scoring of functional groups from gene expression data by decorrelating go graph structure. *Bioinformatics*, pages 1600–1607, 2006.

[3] Aravind Subramanian et. al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *PNAS*, pages 15545–15550, 2005.

[4] Barry R Zeeberg et. al. Gominer: a resource for biological interpretation of genomic and proteomic data. *Genome Biol.*, 2003.

[5] Barry R Zeeberg et. al. High-throughput gominer, an 'industrial-strength' integrative gene ontology tool for interpretation of multiple-microarray experiments, with application to studies of common variable immune deficiency (cvid). *BMC Bioinformatics*, page 168, 2005.

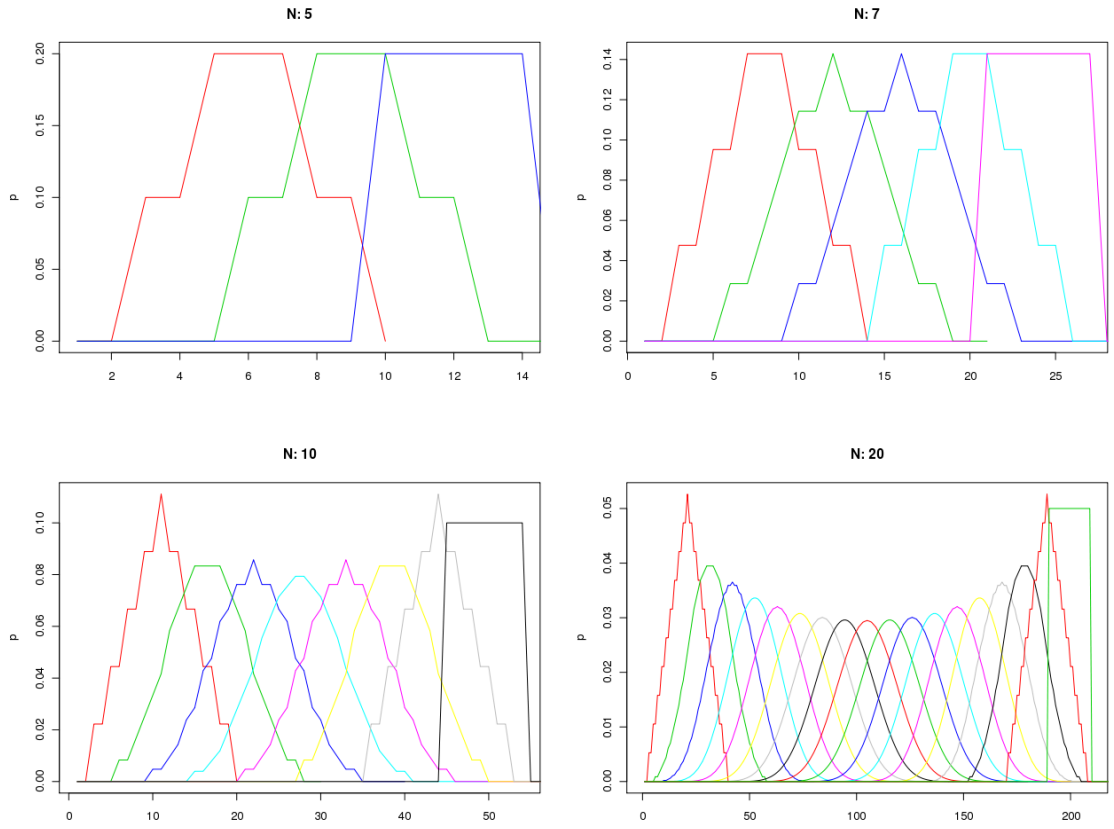

Figure 5: $P_{N,N_T}(R)$ for different values of N and N_T

[6] Bing Zhang et. al. Gotree machine (gotm): a web-based platform for interpreting sets of interesting genes using gene ontology hierarchies. *BMC Bioinformatics*, 2004.

[7] D. Wackerly et. al. Mathematical statistics with applications, sixth edition. *Duxbury*, 2002.

[8] Douglas A Hosack et. al. Identifying biological themes within lists of genes with ease. *Genome Biology*, 2003.

[9] Gil Alterovitz et. al. Go pad: the gene ontology partition database. *Proc Natl Acad Sci*, pages 15545–15550, 2005.

[10] M. Kanehisa et. al. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, pages 27–30, 2000.

[11] S. Falcon et. al. Using gostats to test gene lists for go term association. *Bioinformatics*, pages 257–258, 2006.

[12] S. Falcon et. al. Using gostats to test gene lists for go term association. *Bioinformatics*, 2007.

[13] Scott W Doniger et. al. Mappfinder: using gene ontology and genmapp to create a global gene-expression profile from microarray data. *Genome Biology*, 2003.

[14] Tim Beissbarth et. al. Gostat: Find statistically overrepresented gene ontologies within a group of genes. *Bioinformatics*, pages 1464–1465, 2004.

[15] Vamsi K Mooth et. al. Pgc-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. *Nature Genetics*, page 267, 2003.

[16] The Gene Ontology Consortium (2000) Nature Genet. 25: 25-29 PDF. Gene ontology: tool for the unification of biology. *Nature Genetics*, pages 25–29, 2000.