bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

An algorithm for fast preranked gene set enrichment analysis using cumulative statistic calculation

Alexey A. Sergushichev

Computer Technologies Department, ITMO University, Saint Petersburg, 197101, Russia
alserg@rain.ifmo.ru

Abstract. Gene set enrichment analysis is a widely used tool for analyzing gene expression data. However, current implementations are slow due to a large number of required samples for the analysis to have a good statistical power. In this paper we present a novel algorithm, that efficiently reuses one sample multiple times and thus speeds up the analysis. We show that it is possible to make hundreds of thousands permutations in a few minutes, which leads to very accurate p-values. This, in turn, allows applying standard FDR correction procedures, which are more accurate than the ones currently used. The method is implemented in a form of an R package and is freely available at `https://github.com/ctlab/fgsea`.

Keywords: gene set enrichment, GSEA, sampling, R package

1 Introduction

Gene set enrichment analysis is a very widely used method for analyzing gene expression data. It allows to select from an *a priori* defined list of gene sets those which have non-random behavior in a considered experiment. Compared to a similar method of calculating Fisher p-values of overlap statistic it does not require an arbitrary thresholding. This also allows the method to identify pathways that contain many co-regulated genes but with small individual effects.

The method has a major drawback of being relatively slow. Because analytical from of the null distribution for the used gene set enrichment statistic is not known, empirical null distribution has to be calculated. That can be done in a straightforward manner by sampling random gene sets. However, a big number of gene sets are usually tested simultaneously. This leads to a requirement of a large number of samples for the test to have a good statistical power after a correction for multiple testing.

In the original paper [3] Subramanian et al. developed an ad-hoc method for multiple testing correction. However, the developed method is approximate for the commonly used parameters and it is unclear how accurate it is.

Here we present a fast gene set enrichment analysis (FGSEA) method which is much faster than the original method [3] in finding nominal p-values. The

bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

method is based on an algorithm to calculate cumulative gene set enrichment statistic values, which allows to rapidly calculate multiple sample statistic values from a single sample. Ability to get accurate nominal p-values achieved by the method in a reasonable time leads to using well-developed general methods for multiple testing correction such as Bonferroni or Benjamini-Hochberg.

The rest of the paper is structured as follows. First, in section 2 we formally define gene set enrichment statistic and introduce related definitions. In section 3 we explain the idea of the algorithm on a simple mean statistic. Then, in section 4 we show how GSEA statistic can be interpreted geometrically and present the algorithm for fast computation of cumulative values that follows from such interpretation. Finally, in section 5 we show how the algorithm works in practice and how it is compared to the reference implementation.

2 Definitions

The preranked gene set enrichment analysis takes as input two objects: an array of gene statistic values S and a list of query gene sets P. The goal of the analysis is to determine which of the gene sets from P has a non-random behavior.

The gene statistic array S of the size $|S| = N$ for each gene i, $1 \leq i \leq N$, contains a value $S_i \in \mathbb{R}$ that characterises the gene behavior in a considered process. Commonly, if $S_i > 0$ the expression of gene i goes up in a treatment compared to control and $S_i < 0$ means that the expression goes down. Absolute value $|S_i|$ represents a magnitude of the change. Array S is sorted in a decreasing order: $S_i > S_j$ for $i < j$. The value of N in practice is about 10000–20000.

The list of gene sets P of length $|P| = m$ usually contains groups of genes that are commonly regulated in some biological process. In this paper we assume that all gene sets $p \in P$ have a size upper bound of $K \approx 500$ genes: $|p| \leq K$, $\forall p \in P$.

To quantify a co-regulation of genes in a gene set p Subramanian et al. introduced a gene set enrichment score function $s_r(p)$ that uses gene rankings (values of S). The more positive is the value of $s_r(p)$ the more enriched the gene set is in positively-regulated genes g with $S_g > 0$, accordingly, negative $s_r(p)$ corresponds to enrichment of negatively regulated genes.

Value of $s_r(p)$ can be calculated as follows. Let $k = |p|$, $\mathrm{NS} = \Sigma_{i \in p}|S_i|$. Let also ES be an array specified by the following formula:

$$\mathrm{ES}_i = \begin{cases} 0 & \text{if } i = 0, \\ \mathrm{ES}_{i-1} + \frac{1}{\mathrm{NS}}|S_i| & \text{if } 1 \leq i \leq N \text{ and } i \in p, \\ \mathrm{ES}_{i-1} - \frac{1}{N-k} & \text{if } 1 \leq i \leq N \text{ and } i \notin p. \end{cases}$$

The value of $s_r(p)$ corresponds to the largest by absolute value entry of ES:

$$s_r(p) = \mathrm{ES}_{i^*}, \text{ where } i^* = \arg\max_i |\mathrm{ES}_i|.$$

For each $p \in P$ we need to find the enrichment statistic value and to calculate a p-value of this not to be random. To calculate a p-value for gene set p we can

bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

obtain an empirical null distribution by sampling n random gene sets of the same size as p.

Such straightforward implementation takes $\Theta(mnK \log K)$ time. For each of m gene sets we need to perform n permutations for all of which we need to calculate enrichment statistic value, that can be done in $\Theta(K \log K)$ time.

3 Cumulative statistic calculation for mean statistic

Let first describe the idea of the proposed algorithm on a simple statistic of mean rank s_m:

$$s_m(p) = \frac{1}{|p|} \Sigma_{i \in p} S_i.$$

The idea of the algorithm is to reuse sampling for different query gene sets. This can be done due to the fact that for an estimation of null distributions samples have to be independent only for a specific gene set size. Samples can be dependent between different sizes.

Instead of generating nm independent random gene set for each permutation and each gene set we will generate only n radom gene sets of size K. Let π_i be an i-th random gene set of size K. From that gene set we can generate gene sets for a all the query pathways P_j by using its prefix: $\pi_{i,j} = \pi_i[1..|P_j|]$.

The next step is to calculate enrichment scores for all generated gene sets $\pi_{i,j}$. Instead of calculating enrichment scores separately for each gene set we will calculate simultaneously scores for all $\pi_{i,j}$ for a fixed i. Using a simple procedure it can be done in $\Theta(K)$ time.

Let us find enrichment scores for all prefixes of π_i. This can be done by element-wise dividing of cumulative sums array by the length of the corresponding prefix:

$$s_m(\pi_i[1..k]) = \frac{1}{k} \Sigma_{i \in \pi_i[1..k]} S_i.$$

Selecting only the required prefixes takes an additional $\Theta(m)$ time.

The described procedure allows to find p-values for all query gene sets in $\Theta(n(K+m))$ time. This is about $\min(K, m)$ times faster than the straightforward procedure.

4 Cumulative statistic calculation for GSEA statistic

For the GSEA statistic we use the similar idea: we will also be sampling only gene sets of size K and from that sample will calculate statistic values for all the other sizes. However, calculation of the cumulative statistic values for the subsamples is more complex in this case.

In this section we only be considering positive mode of enrichment statistic s_r^+. It can be defined as follows:

$$s_r^+(p) = \mathrm{ES}_{i^+}, \text{ where } i^+ = \arg\max_i \mathrm{ES}_i.$$

bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

Calculation of $s_r^-(p) = \mathrm{ES}_{i^-}$ for $i^- = \arg\min_i \mathrm{ES}_i$ is very similar. From these two values it easy to find value of $s_r(p)$, which is equal to $s_r^+(p)$ if $|s_r^+(p)| > |s_r^-(p)|$ or $s_r^-(p)$ otherwise.

4.1 Geometric interpretation of GSEA statistic

It is helpful to look at GSEA statistic from a geometric point of view. Let us consider $N+1$ points (Fig. 1) with coordinates (x_i, y_i) for $0 \le i \le N$ such that:

$$(x_0, y_0) = (0, 0), \tag{1}$$

$$x_i = x_{i-1} + [i \notin p], \qquad \forall i \in 1..N, \tag{2}$$

$$y_i = y_{i-1} + [i \in P] \cdot |S_i| \qquad \forall i \in 1..N. \tag{3}$$

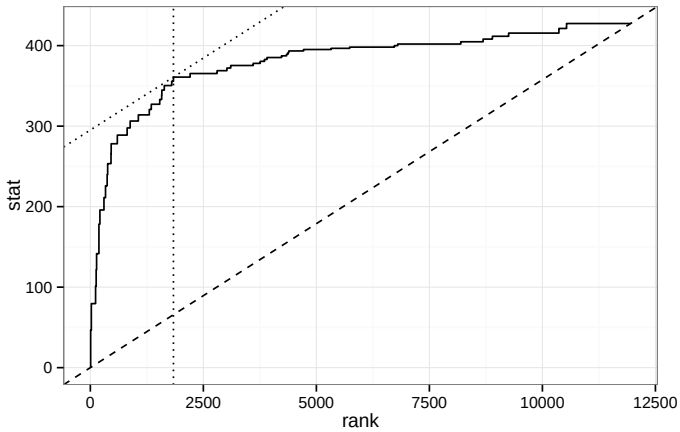

Fig. 1. A graph that corresponds to a calculation of GSEA statistic. Each breakpoint on a graph corresponds to a gene present in the pathway. Dotted line cross at a point which is the farthest up from a diagonal (dashed line). This point correspond to gene i^+, where the maximal value of ES_i is reached

The calculation of s_r^+ corresponds to finding the point farthest up from a diagonal $((x_0, y_0), (x_N, y_N))$. Indeed, it is easy to see that $x_N = N - |p| = N - k$ and $y_N = \Sigma_{j \in p} |S_j| = \mathrm{NS}$, while the individual enrichment scores ES_i can be calculated as $\mathrm{ES_i} = \frac{1}{NS} y_i - \frac{1}{N-k} x_i$. Value of $\mathrm{ES_i}$ is proportional to the directed distance from the line going through (x_0, y_0) and (x_N, y_N) to the point (x_i, y_i).

Let us fix a sample π of length K. To efficiently calculate cumulative values $s_r^+(\pi[1..k])$ for $k \le K$ we need a fast method of updating the farthest point when a new gene is added. In that case we can add genes from π one by one and calculate values $s_r^+(\pi[1..k])$ from the corresponding maximal distances.

bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

Because we are calculating values for $\pi[1..k]$ for $k \leq K$ we know in advance which K genes will be added. This allows us to consider $K+1$ points instead of $N+1$ for each iteration k. Let array o contains order of genes in π, that is π_{o_1} is minimal among π, π_{o_2} is second minimal and so on. The coordinates can be calclated as follows:

$$(x_0^k, y_0^k) = (0, 0), \tag{4}$$

$$x_i^k = x_{i-1}^k + \pi_{o_i} - \pi_{o_{i-1}} - [o_i \leq k]\,, \qquad \forall i \in 1..K, \tag{5}$$

$$y_i^k = y_{i-1}^k + [o_i \leq k] \cdot |S_i|, \qquad \forall i \in 1..K, \tag{6}$$

where we set π_{o_0} to be zero.

It can be shown that finding the farthest up point among (4)–(6) is equivalent to finding the farthest up point among (1)–(3) with (x_i^k, y_i^k) being equal to $(x_{\pi_{o_i}}, y_{\pi_{o_i}})$ calculated for $p = \pi[1..k]$.

Consider $x_{\pi_{o_i}} - x_{\pi_{o_{i-1}}}$. By the definition of x it is equal to:

$$x_{\pi_{o_i}} - x_{\pi_{o_{i-1}}} = \sum_{i=1}^{\pi_{o_i}} [i \notin \pi[1..k]] - \sum_{i=1}^{\pi_{o_{i-1}}} [i \notin \pi[1..k]] = \sum_{i=\pi_{o_{i-1}}+1}^{\pi_{o_i}} [i \notin \pi[1..k]] =$$

$$\pi_{o_i} - \pi_{o_{i-1}} - \sum_{i=\pi_{o_{i-1}}+1}^{\pi_{o_i}} [i \in \pi[1..k]].$$

By the definition of o, in the interval $[\pi_{o_{i-1}} + 1, \pi_{o_i} - 1]$ there are no genes from π and, thus, from $\pi[1..k]$. Thus we can replace sum with its last member:

$$x_{\pi_{o_i}} - x_{\pi_{o_{i-1}}} = \pi_{o_i} - \pi_{o_{i-1}} - [\pi_{o_i} \in \pi[1..k]] = \pi_{o_i} - \pi_{o_{i-1}} - [o_i \leq k].$$

We got the same difference as in (5).

Now consider $y_{\pi_{o_i}} - y_{\pi_{o_{i-1}}}$. By the definition of y it is equal to:

$$y_{\pi_{o_i}} - y_{\pi_{o_{i-1}}} = \sum_{i=1}^{\pi_{o_i}} [i \in \pi[1..k]] \cdot |S_i| - \sum_{i=1}^{\pi_{o_{i-1}}} [i \in \pi[1..k]] \cdot |S_i| = \sum_{i=\pi_{o_{i-1}}+1}^{\pi_{o_i}} [i \in \pi[1..k]] \cdot |S_i|.$$

Again, in the interval $[\pi_{o_{i-1}} + 1..\pi_{o_i} - 1]$ there are no genes from $\pi[1..k]$. Thus we can replace the sum with only the last member:

$$y_{\pi_{o_i}} - y_{\pi_{o_{i-1}}} = [\pi_{o_i} \in \pi[1..k]] \cdot |S_i| = [o_i \leq k] \cdot |S_i|.$$

We got the same difference as in (6).

We do not need to consider other points, because points $o_{i-1}..o_i - 1$ have the same y coordinate and o_{i-1} is the leftmost of them. Thus, when at least one gene is added the diagonal $((x_0, y_0), (x_N, y_N))$ is not horizontal and o_{i-1} is the farthest point among $o_{i-1}..o_i - 1$.

bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

4.2 Square root decomposition

Let consider what happens when gene π_k is added to query set $\pi[1..k-1]$ (Fig. 2). Let r_k be a rank of gene π_k among genes π, then coordinate of points (x_i, y_i) for $i < r_k$ do not change, while all (x_i, y_i) for $i \geq r_k$ are changed on $(\Delta_x, \Delta_y) = (-1, |S_{\pi_k}|)$.

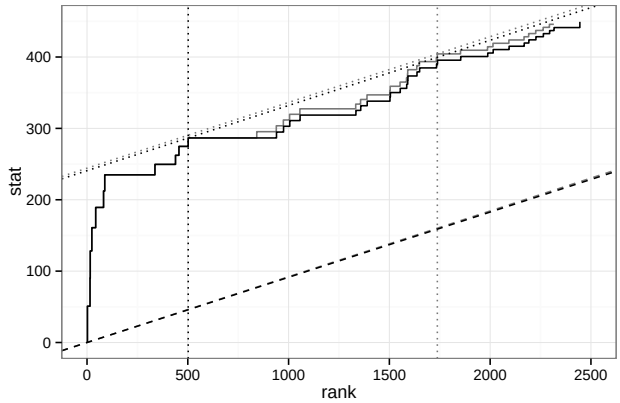

Fig. 2. Update of a GSEA graph when gene $\pi_k \approx 800$ is added. Only a fragment is shown. Black graph corresponds to the GSEA graph for gene set $\pi[1..k-1]$, gray graph corresponds to $\pi[1..k]$. A part of the graph to left of $x = x_{r_k}$ does not change and the other part is shifted to the top-left corner. The diagonal $((x_0, y_0), (x_N, y_N))$ is rotated counterclockwise.

To make fast incremental updates we will decompose the problem into multiple smaller ones. For simplicity we assume that $K + 1$ is an exact square of an integer b. Let split $K + 1$ points into b consecutive blocks of the size b: $\{(x_0^k, y_0^k), ..., (x_{b-1}^k, y_{b-1}^k)\}$, $\{(x_b^k, y_b^k), ..., (x_{2b-1}^k, y_{2b-1}^k)\}$ and so on.

For each of b blocks we will store and update the farthest up point from the diagonal. When we know for each block its farthest point we can find the globally farthest point by a simple pass in $O(b)$ time.

Next, we show how to update the farthest points in blocks in amortized time $O(b)$. This taken together with one $O(b)$ pass will get us an algorithm to update the globally farthest point in amortized $O(b)$ time.

Below we use $c = \lfloor r_k/b \rfloor$ as an index of a block where gene π_k belongs.

First, we describe the procedure to update points coordinates. We will store x_i coordinates using two vectors: B of size b and D of size $K + 1$, such that $x_i = B_{i/b} + D_i$. When gene π_k is added all x_i for $i \geq r_k$ are decremented by one. To reflect this we will decrement all B_j for $j > c$ and decrement all D_i for $r_k \leq i < cb$. It is easy to see that it takes $O(b)$ time. After this update

bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

procedure we can get value x_i in $O(1)$ time. The same procedure is applied for y coordinates.

Second, for each block we will maintain an upper part of its convex hull. Having convex hull is useful because the farthest point in block always lays on its convex hull. All blocks except c have the points either not changed or shifted simultaneously on the same value. That means that lists of points on the convex hulls for these blocks remain unchanged. For the block c we can reconstruct convex hull from scratch using Graham scan algorithm. Because the points are already sorted by x coordinate, this reconstruction takes $O(b)$ time. In total, it takes $O(b)$ time to update the convex hulls.

Third, the farthest points in blocks can be updated using the stored convex hulls. Consider a block where the convex hull was not changed (every block except, possibly, block c). Because diagonal always rotates in the same counter-clockwise direction, the farthest point in block on iteration k either stays the same or moves on the convex hull to the left of the farthest point on the $(k-1)$-th iteration. Thus, for each such block we can compare current farthest point with its left neighbor on the convex hull and update the point if necessary. It is repeated until the next neighbor is closer to the diagonal than the current farthest point. In the block c we just find the farthest point in a single pass by the points on the convex hull.

Using potential method we can show that the updating farthest points takes $O(b)$ amortized time. Let a potential after adding k-th gene Φ_k be a sum of relative indexes of the farthest points for all the blocks. As there are b blocks of size b the sum of relative indexes lies between 0 and b^2. Thus, $\Phi_k = O(b^2)$. For an update of all $b-1$ blocks except c we need to make $t_k = b-1+z$ operations of comparing two points, where z is the number of times the farthest points were updated. This can take up to $\Theta(b^2)$ time in the worst case. However, it can be noticed, that potential change $\Phi_k - \Phi_{k-1}$ is equal to $-z+O(b)$: the sum of indexes is decreased by a number of times the farthest points were updated plus $O(b)$ for the block c where the index can go from 0 to $b-1$. This gives an amortized cost of k-th iteration to be $a_k = t_k + \Phi_k - \Phi_{k-1} = b-1+z-z+O(b) = O(b)$. The total real cost of K iterations is $\sum_{k=1}^{K} a_k + \Phi_0 - \Phi_K = O(Kb) + O(b^2) = O(Kb)$, which means amortized cost of one iteration to be $O(b)$.

Taken together the algorithm allows to find all cumulative enrichment scores $s_r(\pi[1..k])$ in $O(Kb) = O(K\sqrt{K})$ time. The straightforward implementation of calculating cumulative values from scratch would take $O(KK\log K)$ time. Thus, we have improved the performance $O(K\frac{\log K}{\sqrt{K}})$ times.

4.3 Implementation details

We also implemented an optimization that does not build convex hull from scratch for a changed block c, but only updates the changed points. This does not influence on asymptotic performance, but decreases the constant factor.

First, we start updating convex hull from position r_k and not from 1. To be able to do this, we have an array `prev` that for each gene $g \in \pi$ stores a previous

bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

point on a convex hull if g were the last gene in the block. This actually is the same as the top of the stack in Graham algorithm and represent the algorithms state for any given point. As all points h to the left of g are not changed $\texttt{prev}_h$ also remains unchanged and need not to be recalculated.

Second, we stop updating the hull, when we reach the point on the previous iteration convex hull. We can do this because every point to the left of g is rotated counterclockwise of any point to the right of g, which means that the first point on the convex hull right of g on $(k-1)$-th iteration remains being a convex hull point at k-th iteration.

5 Experimental results

To assess the algorithm performance we ran the algorithm on a T-cells differentiation dataset [4]. The ranking was obtained from differential gene expression analysis for Naive vs. Th1 states using *limma* [2]. From that results we selected 12000 genes with the highest mean expression levels.

As a pathway databases we used Reactome database [1]. There were 586 gene sets that had overlaps with the selected genes of the size 15 to 500 (common gene set size limits for preranked gene set enrichment analysis).

We compared the algorithm to the reference GSEA implementation [3] version 2.1.

Experiments were run on an Intel Core i3 2.10GHz processor. Both methods were run in one thread.

We ran reference GSEA with default parameters. The permutation number was set to 1000, which means that for each input gene set 1000 independent samples were generated. The run took 100 seconds and resulted in 79 gene sets with GSEA-adjusted FDR q-value of less than 10^{-2}. All significant gene sets were in a positive mode.

First, to get a similar nominal p-values accuracy we ran FGSEA algorithm on 1000 permutations. This took 2 seconds, but resulted in no significant hits due after multiple testing correction (with FRD $\leq 1\%$). The same effect required GSEA authors to develop a custom method approximate FDR correction.

Second, to get a similar number of significant hits FGSEA was run with 10000 permutations. It took 9 seconds and resulted in 78 gene sets with BH-adjusted p-value of less than 10^{-2}. Is is important to note, that, unlike for GSEA, 3 of these 78 gene sets were in a negative mode.

Last, we ran FGSEA with a similar total running time. Withing 90 seconds FGSEA was able to do 100000 permutations. This resulted in 77 gene sets with BH-adjusted p-value of less than 10^{-2} (3 sets were in the negative mode). The minimal nominal p-value was $1.23 \cdot 10^{-5}$.

6 Conclusion

Preranked gene set enrichment analysis is a widely used tool in analysis of gene expression data. However, current implementations are slow due to a lot of sam-

bioRxiv preprint doi: https://doi.org/10.1101/060012; this version posted June 20, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

pling. Here we present an algorithm that allows to decrease the number of required samples while keeping the same accuracy of nominal p-values. This allows to achieve more accurate p-values in a faster time. Consequently, gene sets can be ranked more precisely in the results and, which is even more important, standard multiple testing correction methods can be applied instead of approximate ones as in [3].

Availability

The FGSEA method implemented as a package for R and is available at `https://github.com/ctlab/fgsea` along with the example data and corresponding results.

Acknowledgements

This work was supported by Government of Russian Federation Grant 074-U01.

The author thanks Gennady Korotkevich for the idea of the algorithm.

References

1. Joshi-Tope, G., Gillespie, M., Vastrik, I., D'Eustachio, P., Schmidt, E., de Bono, B., Jassal, B., Gopinath, G.R., Wu, G.R., Matthews, L., Lewis, S., Birney, E., Stein, L.: Reactome: a knowledgebase of biological pathways. Nucleic acids research 33(Database issue), D428–32 (2005)
2. Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., Smyth, G.K.: limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research pp. gkv007– (2015)
3. Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., Mesirov, J.P.: Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences of the United States of America 102(43), 15545–50 (2005)
4. Wei, G., Wei, L., Zhu, J., Zang, C., Hu-Li, J., Yao, Z., Cui, K., Kanno, Y., Roh, T.Y., Watford, W.T., Schones, D.E., Peng, W., Sun, H.W., Paul, W.E., O'Shea, J.J., Zhao, K.: Global mapping of H3K4me3 and H3K27me3 reveals specificity and plasticity in lineage fate determination of differentiating CD4+ T cells. Immunity 30(1), 155–67 (2009)