GeneFunnel: a mean absolute deviation-based, dispersion-adjusted gene set scoring method

Emir Turkes[1], Karen E. Duff[1,2]

[1]UK Dementia Research Institute at University College London, London, UK
[2]Taub Institute, Columbia University Medical Center, New York, NY, USA

***Corresponding authors:**
Emir Turkes; emir.turkes.19@ucl.ac.uk, Karen E. Duff; k.duff@ucl.ac.uk

Keywords:
Gene Set Scoring, Gene Set Enrichment, Pathway Analysis, Bioinformatics

Abstract:
Background: Gene set enrichment is central to the interpretation of transcriptomic and proteomic data. Functional class scoring methods such as GSVA and ssGSEA provide per-sample pathway activity but can introduce inter-sample dependence, handle zeros and missing values inconsistently, and vary widely in computational efficiency and stability. Over-representation approaches, for example g:Profiler, test predefined hit lists against gene set catalogues, yet they depend on arbitrary thresholds for differential expression, are sensitive to list size and background choice, and ignore the magnitude and evenness of expression across all features.
Results: We present GeneFunnel, which for each gene set in each sample computes a dispersion-adjusted sum using a size-aware mean absolute deviation-derived penalty, discouraging scores driven by a few outlier genes and favouring more even contributions. The resulting scores lie on an absolute, sample-independent scale with a baseline at zero, simplifying cross-dataset comparison and use with methods that assume non-negative inputs. In simulations probing partial activation, variance-only changes, and subtle coordinated shifts, GeneFunnel shows high sensitivity with low false-positive rates when paired with standard statistical testing using limma. In real single-cell RNA-seq data from Alzheimer's Disease post-mortem brain tissue, GeneFunnel highlights pathology-relevant processes and down-weights small sets driven by only a few genes. An Rcpp implementation delivers leading runtime and memory use when benchmarked against comparable methods.
Conclusions: GeneFunnel provides fast, interpretable and dispersion-adjusted per-sample pathway scores that integrate cleanly with common statistical pipelines and are practical for bulk and single-cell studies. The software is released as an R package with source code on GitHub (https://github.com/eturkes/genefunnel), alongside a companion web application for data upload, scoring and results export (https://data.duff-lab.org/app/genefunnel-shiny-app).

Introduction:

Gene set enrichment is widely used to interpret high-throughput expression data by summarising gene-level signals into pathway-level readouts, and has been reviewed extensively in [1–8]. Approaches fall into three broad families. Over-representation analysis evaluates predefined hit lists against gene-set catalogues and is simple to apply, but depends on arbitrary thresholds, is sensitive to list size and background choice, and discards information about magnitude and evenness of expression. Rank-based methods popularised by GSEA [9] assess whether set members concentrate at the extremes of a ranked list and can detect subtle coordinated shifts, yet they operate at the group level and are not inherently per-sample. Functional class scoring (FCS) methods, including GSVA [10] and ssGSEA [11], deliver per-sample scores but often borrow information across samples through ranking or normalisation, which can alter scores when the sample set changes and can complicate comparisons across datasets.

In practice, analysts face several recurring problems. First, many procedures entangle samples during scoring, which weakens the appeal of per-sample interpretation and undermines reusability of scores across studies. Second, zeros and missing values are handled inconsistently, despite being common in RNAseq and proteomics. Zeros in particular are often informative and should not be silently discarded. Third, very small gene sets can dominate results when a single highly expressed gene carries most of the signal, whereas large sets can be favoured simply by size when penalties are not calibrated. Finally, runtime and memory demands vary greatly between methods, which can limit routine use on single-cell data.

We introduce GeneFunnel, a dispersion-adjusted gene-set scoring method that retains strict sample independence and produces scores on an absolute scale with a natural zero baseline. For each gene set in each sample, GeneFunnel computes the set sum and subtracts a size-aware mean absolute deviation (MAD) penalty. The score rises with total activity but is reduced when contributions are uneven, so sets with broadly shared signal score higher than those dominated by a few genes. Zero counts are treated as measurements and retained, while missing values are not permitted at scoring; when a gene set includes features absent from the matrix, the set is restricted to the features present.

GeneFunnel is designed to be intuitive to reason with. Scores increase with total activity, but only to the extent that contributions are broadly shared within the set. The size-aware penalty is stronger for smaller sets, which controls spurious hits from two or three gene terms, and approaches a constant for large sets. The scoring rule yields non-negative outputs with zero as the limit under maximal unevenness, which facilitates downstream transformations and distance measures that assume non-negative inputs. A performant Rcpp implementation enables routine use on bulk and single-cell matrices in RNAseq, proteomics, or other assays with comparable data types.

Implementation:

The functional class scoring algorithm we propose, GeneFunnel, is straightforward: it is similar to subtracting the mean absolute deviation from the sum of values in a gene set. GeneFunnel iterates through each gene set for each sample, introducing no dependency between samples or between gene sets. For a given gene set in a given sample, we first subset the sample's features to those present in the set and compute the set sum and the set mean. We then take, for each feature in the set, the absolute difference between its expression and the set mean, and sum these absolute deviations. A size-aware scaling factor is applied to this deviation term, defined as the size of the gene set divided by twice the residual gene set size (1 minus the gene set size). Finally, the scaled deviation is subtracted from the set sum to yield the score. This procedure is repeated for all gene sets within the sample and then across all samples, producing a gene-set-by-sample score matrix that mirrors the shape of the original data (Figure 1A). The algorithm is expressed in mathematical notation in Figure 1B, and an excerpt of the Rcpp (C++ interface for R) implementation is shown in Figure 1C.

At the core of GeneFunnel's scoring method is its use of both the sum and deviation of feature expression levels within a gene set. By first summing expression values, the method captures the overall activity level of a pathway, akin to approaches that rely on simple averaging or summation. However, instead of assuming that all features contribute equally, GeneFunnel then computes deviance scores for each feature, measuring how much each feature's expression deviates from the mean expression of the set. This deviation-aware component ensures that pathways with highly variable expression across member features are penalised, preventing scenarios where a small number of highly expressed features dominate the enrichment score.

Proof of Non-negative Scores:

A fundamental requirement for GeneFunnel is that its scores remain non-negative, ensuring compatibility with common downstream analyses in functional genomics such as dimensionality reduction, normalisation, and differential expression analysis. The method is designed such that the minimum possible score is zero, which occurs in two biologically interpretable cases: when all features in the set have zero expression or when the set exhibits maximal internal deviation, meaning that the expression values are so dispersed that the deviation term fully offsets the total summed expression (i.e. the case when a single value is non-zero). Proving that GeneFunnel always produces non-negative scores formally validates that it is a proper transformation of gene expression data, ensuring interpretability and compatibility with standard computational workflows.

Theorem: GeneFunnel Scores Cannot be Negative

Let $X_{i,j}$ be the expression level of feature i in sample j, and let G_k be a predefined gene set containing $|G_k|$ features. The GeneFunnel score for gene set G_k in sample j is given by:

$$\text{score}_{k,j} = \sum_{i \in G_k} X_{i,j} - \left(\frac{|G_k|}{2(|G_k|-1)} \sum_{i \in G_k} \left| X_{i,j} - \bar{X}_{G_k,j} \right| \right)$$

Then, for all k and j:

$\text{score}_{k,j} \geq 0.$

Proof:

We begin by expanding the sum of values in the gene set (found left-hand-side or LHS of the parenthesis), where in a general case, the sum of values is equal to the mean of values times the number of values:

$$\sum_{i \in G_k} X_{i,j} = |G_k| \bar{X}_{G_k,j}$$

Substituting this into the scoring equation, located LHS of the parenthesis, we obtain:

$$\text{score}_{k,j} = |G_k| \bar{X}_{G_k,j} - \left(\frac{|G_k|}{2(|G_k|-1)} \sum_{i \in G_k} \left| X_{i,j} - \bar{X}_{G_k,j} \right| \right)$$

Factoring out $|G_k|$ from that substitution, and from within the parenthesis, simplifies the score to:

$$\text{score}_{k,j} = |G_k| \left(\bar{X}_{G_k,j} - \frac{1}{2(|G_k|-1)} \sum_{i \in G_k} \left| X_{i,j} - \bar{X}_{G_k,j} \right| \right)$$

Looking within the parenthesis, we form the following inequality, stating that the mean of values is always greater than the sum of absolute deviances from the mean multiplied by the scaling factor:

$$\bar{X}_{G_k,j} \geq \frac{1}{2(|G_k|-1)} \sum_{i \in G_k} \left| X_{i,j} - \bar{X}_{G_k,j} \right|$$

Note that omitting the scaling factor from the RHS yields the equation for Mean Absolute Deviation (MAD):

$$\frac{1}{|G_k|} \sum_{i \in G_k} \left| X_{i,j} - \bar{X}_{G_k,j} \right|$$

As shown in (Aghili-Ashtiani, 2021):

$$\text{MAD} = \frac{1}{n} \sum_{i=1}^{n} |x_i - \bar{x}|$$

Where $x_1, x_2, \ldots, x_n \in \mathbb{R}$ is a set of real numbers.

We therefore reformulate the problem as follows, noting however that the inequality that the mean of values as always being greater than or equal to the MAD does not hold:

$$\bar{x} \not\geq \frac{1}{n} \sum_{i=1}^{n} |x_i - \bar{x}|$$

In fact, in maximally deviating sets, where there is only a single non-zero value, the ratio of the MAD to the mean approaches 2 with increasing set size.

Let’s assume a vector $x = [a, 0, 0, \ldots, 0]$ of length n, where only the first value is non-zero. Then:

$$\bar{x} = \frac{a}{n}$$

and:

$$\mathrm{MAD} = \frac{2a(n-1)}{n^2}$$

The ratio between the MAD and mean can then be written as:

$$\frac{\mathrm{MAD}}{\bar{x}} = \frac{2a(n-1)/n^2}{a/n} = \frac{2(n-1)}{n}$$

Finally, as set size approaches infinity:

$$\lim_{n\to\infty} \frac{2(n-1)}{n} = \lim_{n\to\infty} \left(2 - \frac{2}{n}\right) = 2$$

Therefore:

$$\bar{x} \ngeq \frac{1}{n}\sum_{i=1}^{n} |x_i - \bar{x}|$$

These findings helped influence discovery of the appropriate GeneFunnel scaling factor. With the scaling factor applied, the above evaluates as follows:

$$\mathrm{MAD}_{\text{scaled}} = \frac{1}{2(n-1)}\sum_{i=1}^{n} |x_i - \bar{x}|$$

$$\mathrm{MAD}_{\text{scaled}} = \frac{1}{2(n-1)} \cdot \frac{2a(n-1)}{n} = \frac{a}{n}$$

$$\frac{\mathrm{MAD}_{\text{scaled}}}{\bar{x}} = \frac{a/n}{a/n} = 1$$

Therefore:

$$\bar{x} \geq \frac{1}{2(n-1)}\sum_{i=1}^{n} |x_i - \bar{x}|$$

Substituting in the definitions for GeneFunnel:

$$\bar{X}_{G_k,j} \geq \frac{1}{2(|G_k|-1)}\sum_{i\in G_k} \left|X_{i,j} - \bar{X}_{G_k,j}\right|$$

We show that the inequality is satisfied for GeneFunnel scores, and conclude that the scaling factor combined with MAD is necessary to ensure that for all k and j:

$\mathrm{score}_{k,j} \geq 0.$

Thus GeneFunnel always produces non-negative scores.

Results:

Exploration of GeneFunnel Properties:

To thoroughly evaluate the behaviour of GeneFunnel and understand its scoring properties, we conducted an exploration of its theoretical and practical characteristics before benchmarking it against existing methods. To facilitate this process, we developed a Shiny web application (https://data.duff-lab.org/app/genefunnel-benchmarks-viewer), which provides an interactive interface for investigating how GeneFunnel responds to different input scenarios.

A key component of this exploration involved constructing a hypothetical gene by sample matrix to simulate different patterns of gene expression (Figure 2). This synthetic dataset allowed for precise control over the relationships between genes, enabling a systematic examination of how GeneFunnel assigns scores under various conditions. Within the web app, users can interactively modify values within this matrix, effectively simulating different gene expression profiles. Each change is processed in real time, with GeneFunnel recomputing scores for all gene sets dynamically. The results are displayed as a heatmap of the gene set by sample matrix, providing immediate visual feedback on how alterations in individual genes affect pathway-level enrichment scores. This interactive approach not only aids in validating theoretical expectations, such as the behaviour of GeneFunnel under extreme cases, but also helps intuitively illustrate how the method differs from traditional functional class scoring approaches.

The selection of values for the originating gene by sample matrix was very deliberate, to try to cover the broad range of situations GeneFunnel was designed to excel in, within a minimal example. Starting with the first column of Figures 2A and B, with Sample 1, it can be seen that all of the values hover around the arbitrary expression value of 50. These values also include a small degree of noise or jitter. Upon examination of Figure 2C, the metrics below the Sample 1 cell confirm these properties. The mean is precisely 50, and with 10 values, this also results in a sum of 500. The small amount of deviance between features is also captured, which when subtracted from the mean results in a final value of 483.

The values in Sample 2 were specifically selected to contrast with Sample 1. Examining the original values, it is clear that this column contains much more feature deviance, with values above 100 and several values recorded as 0. With the inclusion of zero values, Sample 2 was designed to have the same sum and mean as Sample 1. However, the large total deviance of 298 brings a significant penalty to the final score, dropping it from 500 to just 202. This is in contrast to Sample 1, which has a final score of 483. Sample 3 meanwhile confirms that when all features are equal in value, the total deviance is zero. As the values in this sample now centre around 100 rather than 50, the mean is now 50 while the sum is 1,000. With a lack of feature deviance, this results in simply an enrichment score equal to the sum.

Sample 4 differs substantially from the others in that there are NA values in the original matrix. As a result, the gene set in Sample 4 is actually treated as a gene set with a size of 5 rather than 10. This changes the scaling factor. Whereas the other samples have a scaling factor of $\frac{10}{18}$, the scaling factor for Sample 4 changes to $\frac{5}{8}$. This would normally have an effect on the deviance score, though in this example, there is no deviance between features to begin with, so it remains zero. However, what is noteworthy is that the mean of Sample 4 is equal to Sample 3, but the final enrichment score and sum is 500 rather than 1,000. This indicates GeneFunnel's preference for scoring larger gene sets higher, with the argument being that an enriched larger gene set is more likely to be of biological interest than smaller ones.

The final column, Sample 5, showcases a situation where all features exhibit maximal deviance, producing a score of zero as supported by the proof in preceding sections. Containing a single non-zero value, the sum is fully cancelled out by an equivalent deviance penalty. This would be the case in all gene set sizes containing a single non-zero value. As the proportion of non-zero values increase, the enrichment score gradually increases until an equilibrium where half of values are non-zero. Assuming no additional deviance, the final enrichment score in such case would be half of the sum.

Comparing Properties With Other Functional Class Scoring Methods:
We next aimed to build off the exploratory approach in the preceding section and apply it to several other functional class scoring methods. We elected to compare GeneFunnel with other functional class scoring methods that transform expression data into a gene-set-by-sample matrix: GSVA (testing both Poisson and Gaussian kernels) [10], ssGSEA [11], PLAGE [12], and the Z-score method available from the GSVA R package. Like the last analysis, the results are wholly contained in the web app in the next tab section. The first series of explorations again focus on a hypothetical gene by sample matrix, constructed similarly as the first with slight modifications (Figure 3A). After running each of the models, the results are condensed into enrichment heatmaps (Figure 3B).

All methods were run as recommended by their authors for all benchmarking. Importantly, all data input into GSVA Gaussian, ssGSEA, PLAGE, and Z-score underwent a log2 + 1 transformation, with GSVA Poisson (and GeneFunnel) being the only methods taking the raw data. The minimum set size was also set to 2 for all methods. Finally, the normalisation step in ssGSEA was turned off, as the method is no longer a single-sample method with it applied. All methods were ran with parallel processing through BiocParallel.

Beginning with Sample 1, as the most basic test, both Gene Set X and Y should be more-or-less similar, as the data contained in each are nearly identical. GeneFunnel produces scores that reflect this, with 237.75 and 244.50 for Gene Set X and Y, respectively. All other methods show noticeable and generally large differences between them, especially with GSVA. GSVA in particular attempts to distribute its output along a range of -1 and 1, similar to a Z-score, making it the most inappropriate for assessing just a few gene sets. While this dataset is indeed a contrived example, it is not inconceivable to be interested in only scoring a few select gene sets in a real-world situation. GSVA was however, the only method other than GeneFunnel to attribute the highest score to Gene Set Z, the largest gene set encompassing all features in the test dataset.

In Sample 2, the expectation was to see generally lower scores than in Sample 1, while following the same pattern of Gene Set X and Y being comparable, and Gene Set Z having the largest scores. GeneFunnel fulfilled this criteria, while all others failed. Most of the methods showed similar patterning as in Sample 1, while Z-score appeared to similarly score Gene Set Y and Gene Set Z (the largest gene set) this time, which could not be explained.

Sample 3 is the most straightforward of the samples. With no deviance between features at all, Gene Set X and Y should be identical, with Gene Set Z at least being identical or larger. This time, there were two methods that could be considered comparable to the output seen in GeneFunnel. PLAGE showed very sensible results in that all gene sets of Sample 3 were the largest scoring sets in the whole dataset. Furthermore, the scores for Gene Set X and Y are quite similar (1.643413 vs. 1.594036), though not precisely identical like GeneFunnel. While Sample 3 Gene Set Z is indeed the highest score in the entire dataset for GeneFunnel as well, Gene Sets X and Y are more similar to Gene Set Z of Sample 1. The other method that demonstrated sensible performance in Sample 3 was GSVA Gaussian, as Gene Set X and Y are more similar to one another compared to those sets in other samples. Gene Set Z also received the highest score, though the methodology of GSVA makes it so that there is no difference in the score of Gene Set Z between samples; it converges towards 1 in all cases.

Finally, it was expected for Sample 4 to produce scores that increase in value slightly from Gene Set X, to Gene Set Y, to Gene Set Z, as the proportion of zeros to non-zero values decrease. GeneFunnel reflected this, although the difference between Gene Set Y and Z were small and hard to see on the heatmap (50 vs. 66.66). The only other method that had the correct trend was GSVA Gaussian, however, like other samples, the gap between Gene Set Z compared to the other gene sets is extreme. While PLAGE didn't show the complete expected pattern (Gene Set Z was the lowest scoring), it did correctly show Gene Set X as less enriched than Gene Set Y, which is an undebatable expectation. Furthermore, as a whole, the values in Sample 4 are the lowest in the entire dataset, which should also be expected.

In conclusion, at least in this contrived scenario, aside from GeneFunnel, all of the tested methods performed in ways that were hard to reason with. While it appears to be the case that none of these methods were constructed to work with such small test cases, it is still a significant drawback. After all, a very useful approach for exploratory work into understanding how a method works and interacts with changing parameters are through small, controlled experiments like these. Furthermore, some real-world datasets, particularly in proteomics, can be of small size, and it is unclear of what size dataset is Incomparability with such scenarios bring about major limitations to the adoption of these methods. Outside of this, not every real-world experiment is high-throughput especially when working with emerging technology such as spatial omics. It is important for bioinformatic methods to be robust to a range of dataset sizes and it can be demonstrated here that at least within small datasets, GeneFunnel performs sensibly.

Benchmarking Against Other Methods in Synthetic Data:
To test whether the small-panel results were an artifact of the setup rather than the methods, another synthetic benchmark was built comparing two groups on a large gene catalog and with formal statistical testing alongside a broader mix of approaches, including approaches that cover the main families of gene-set inference: ORA, camera, fgsea, and GSVA/ssGSEA. ORA (over-representation analysis) takes the final list of differentially expressed genes and asks, via an enrichment test against the background gene catalogue, whether each set contains more hits than expected by chance; this is the generalised approach taken by the popular g:Profiler [13], but this implementation permits an arbitrary set catalogue and background, which is necessary for synthetic benchmarks. camera, from limma [14, 15], is a competitive test that fits a linear model per gene and then evaluates whether genes in a set show stronger differential expression than genes outside the set. fgsea is an R implementation of GSEA [9], which operates on a ranked list of genes and computes an enrichment score that reflects whether set members concentrate near the top or bottom of the ranking [16]. GSVA and ssGSEA are among the functional class scoring methods used in the prior benchmark, computing a per-sample score for each set without using group labels, similar to GeneFunnel. For these, as well as GeneFunnel, a stock limma-trend pipeline was applied to test for differential enrichment between groups. This mixture of methods allow for a comparison of hit-list enrichment (ORA), model-based competitive testing (camera), rank-based enrichment

(fgsea), functional class scoring (GSVA and ssGSEA), and the proposed functional class scoring method (GeneFunnel) under one evaluation protocol.

The simulation uses a 20,000 gene matrix partitioned into 1,000 non-overlapping gene sets (20 genes per set) and 10 samples (5 in group A, 5 in group B). In each experiment, 50 sets are designated signal and the remaining 950 null. Counts are drawn from a negative-binomial model with realistic library-size variation, then normalised with edgeR TMM before set-level scoring and testing across the gene matrix. Signals were injected under three patterns that isolate different behaviours of gene set enrichment methods and expose different dynamic ranges of gene set activity. In “spike”, only half of genes in a signal set is perturbed, but strongly, while the remainder is left untouched, which probes a method’s tolerance to partial activation and within-set heterogeneity (Figure 4A). In “variance”, the set mean of the signal set is preserved while the within-set dispersion is deliberately reduced in one group, testing the ability of the methods in assessing within-set consistency (Figure 4B). In “coordinated”, a small but consistent log fold change is applied to all genes in a signal set in one group, producing the most classic example of gene set enrichment but within a small dynamic range so as to stress the sensitivity of each method (Figure 4C). Each setting was then evaluated at FDR 0.05, using statistical testing intrinsic to each method or limma-trend otherwise, recording sensitivity, specificity, precision and other common benchmarks. In contrast with the last simulation, which functions as an exploration of functional class scoring properties under precisely defined but ultimately unrealistic scenarios, this simulation study intends to more comprehensively cover the various approaches to gene set enrichment in a dataset with realistic properties and signal structures. It furthermore provides a clearer picture of where GeneFunnel’s design, which rewards both signal magnitude and within-set consistency, confers advantages or exposes limitations in practical use.

The tables in Figure 5 summarise method performance at the threshold of FDR (BH adjusted p-value) 0.05 for each perturbation paradigm. With 50 signal sets and 950 null sets per experiment, TP (true positive) counts signal sets correctly detected, FN (false negative) the missed signal sets, TN (true negative) the correctly rejected null sets, and FP (false positive) the null sets falsely called. From these we report sensitivity (TP/P), specificity (TN/N), precision (TP/(TP+FP)), accuracy ((TP+TN)/(P+N)), and F1 (the harmonic mean of precision and sensitivity). We also report the average FDR for the signal sets, defined as the mean BH adjusted p-value across all 50 signal sets for each paradigm. Higher is better for all rates except average FDR, where lower is better.

In the “spike” paradigm, where only half of the genes in a signal set are perturbed, though robustly, all methods perform well except ORA. GeneFunnel and camera achieve perfect detection (50/50 true positives with no false positives). The rank-based and unsupervised scoring approaches are close to this ceiling; fgsea detects all 50 signal sets with two false positives, GSVA recovers 45 of 50 with one false positive, and ssGSEA detects all 50 with three false positives. ORA remains highly specific but has low sensitivity (15/50), likely because partial activation leaves too few genes surpassing the differential expression threshold to trigger over-representation at the set level. Although the 50 signal sets do not overlap any other sets, several methods are sensitive to the overall distribution of gene-level statistics or ranks. The robust perturbation of the signal sets may have shifted this background slightly, resulting in false positives for those methods. GeneFunnel is only susceptible to this issue at the statistical testing stage, i.e. limma, as the scoring mechanism itself operates on each gene set and sample in isolation.

In the “variance” paradigm, where the mean is preserved and only within-set dispersion is altered, procedures that target location differences lose power. GeneFunnel retains the highest sensitivity because its scoring emphasises within-set consistency as well as magnitude, allowing reduced variability to register as a stronger, more coherent pattern despite the lack of mean change. Camera and ORA identify a smaller fraction of signal sets, and the rank-based and unsupervised scoring methods detect none at the chosen threshold. Though no other method claims to measure inter-gene variance, individual changes to gene counts to reduce inter-gene variance pushes some genes past the significance threshold for regular differential expression testing. It is likely that when several such genes occur in the same set, methods that aggregate gene-level evidence, such as camera or ORA, can incidentally report enrichment despite not explicitly including criteria for within-set consistency. Across methods, average FDRs are higher than in “spike”, reflecting weaker evidence when the signal resides in dispersion rather than in the mean.

In the “coordinated” paradigm, where a very small (0.25 logFC with 0.1 standard deviation), but consistent log-fold change is applied to all members of each signal set, GeneFunnel again achieves the best combination of sensitivity and F1. GSVA is second, in line with its design to capture coordinated per-sample shifts, and camera detects fewer sets at this subtle effect size. fgsea and ORA do not register signal sets at all here, indicating that the per-gene effects are too small to accumulate sufficient ranked-list or hit-list evidence at an FDR of 0.05. This paradigm is the most standard formulation of gene set enrichment and serves as a direct test of method sensitivity to small but coherent shifts. With the current effect size and 5 vs 5 samples the signal is intentionally challenging, so power concentrates in methods that aggregate weak, consistent changes across all genes in a set.

Across these simulations GeneFunnel shows the most consistent power compared to alternative methods. It reaches the ceiling in the spike setting, retains the highest sensitivity when the signal is variance only, and remains competitive for small coordinated shifts. This matches the design goal of the method, which produces per-sample set scores that reward both effect magnitude and within-set consistency, so partial activation, tighter dispersion and subtle coordinated changes can each yield a detectable set-level signal. GeneFunnel works within a standard limma-trend workflow, gives interpretable profiles at the sample level, and maintains low false-positive rates at the stated FDR.

A few caveats remain. The current simulation uses 5 vs. 5 samples, non-overlapping sets of size 20 and a single catalogue of genes. Performance could change with larger or smaller cohorts, different set sizes, heavy set overlap or highly redundant catalogues, and stronger gene-gene correlation. In this experiment, GeneFunnel and other functional class scoring methods relied on limma-trend for statistical testing, and ensuring its proper calibration is non-trivial. Furthermore, there are other downstream testing frameworks that can significantly affect the performance of these methods. Finally, the evidence here remains fully synthetic, and while the proceeding section covers usage in real-world data, testing in biological “ground truth” data, such as those utilising RNA spike-ins may be of value, though such datasets still contain non-trivialities in generation and interpretation.

Benchmarking Against Other Methods in Real Data:
Having run two synthetic experiments, one exploratory within the functional class scoring family and one spanning method families with formal testing, we then compared GeneFunnel to other methods in real data. Because ground truth is unknown in this setting, we restrict the comparison to functional class scoring methods to create a more like-for-like testing framework, which makes qualitative comparisons more interpretable. The dataset of choice was a single-cell RNA sequencing dataset in post-mortem Alzheimer's Disease brain tissue, where neurons containing neurofibrillary tangles where compared to adjacent neurons without tangles. Full details of the dataset can be found in [17]. Details of its pre-processing pipeline can be found in Supplementary Methods. For these benchmarks, we used the annotated Seurat object provided by the authors. This object is a single-cell-by-gene matrix which we then psuedobulked into a sample-by-gene matrix using the *aggregateAcrossCells* function from [18]. The parameters for psueduobulking were set to produce a simple two column output, aggregating cells into either a tangle-bearing or non-tangle-bearing group while ignoring donor label information. In the following section we describe a number of controlled transformations of these objects and test the ability of each functional class scoring methods to capture these transformations.

The chosen transformation was to arbitrarily select a single column, in this case the tangle-bearing neurons, and alter the gene expression of genes corresponding to specific gene sets. To do so, we choose two particular gene sets: NELF Complex and Trace-amine Receptor Activity. These gene sets were chosen because they have no gene overlap with other gene sets in the testing set, therefore, any changes detected should only be in these two sets (Figure 6A). NELF Complex was modified to reduce variability, that is, all genes of the set were transformed into the sum of the gene counts divided by the total number of genes in the set. Trace-amine Receptor Activity was simply modified to have increased counts; all genes in the set had 100 counts added to them. A column containing these modifications was added to the original object, while leaving the original column unmodified. We then ran each functional class scoring method on the modified and unmodified columns. The result of each functional class scoring output is shown in Figure 6B. We subset to the top two gene sets sorted by greatest absolute difference between the modified and unmodified columns. If a method successful captured the induced modifications, then the altered gene sets should be the ones present in the sorted data.

As can be seen in Figure 6B, all methods successfully captured the change in Trace-amine Receptor Activity, which simply had counts of associated genes increased by 100 counts. This demonstrates that all methods have the capacity to capture simple linear changes in expression level. However, only three methods also showed NELF as being among the top two hits: ssGSEA, PLAGE, and GeneFunnel. This shows that these methods are sensitive, at least to some extent and whether incidental or not, to changes in the variability between features, even without changes in overall expression levels.

In our next test, we examined the ability of each method to detect changes in the condition-level pseudobulked dataset without modifications, in other words, a comparison of the gene set composition of tangle-bearing vs. non-tangle-bearing neurons. In order to make this comparison as straightforward as possible, all the donors were pooled together and no statistical testing is performed. The hypothesis is that when sorting gene sets by the absolute difference between the two conditions, as done in the prior tests, gene sets relevant to Alzheimer's Disease should rise to the top. If not, then manual inspection of the top gene sets should at least reveal that they are reasonable and reflect likely real changes. The results of gene set enrichment for this experiment is shown in Figure 7A, along with inspection of the genes within some of the gene sets in Figure 7B.

This last benchmark appeared to to produce a large divergence between GeneFunnel and the other methods. Comparisons with other methods aside, GeneFunnel does appear to highlight gene sets of immediate relevance to AD: containing terms such as Tau Protein Binding and Positive Regulation of Tau-protein Kinase Activity. Neither of these terms are shown among the top five for the other methods. In regard to term overlap between GeneFunnel and other methods, there is a term related to dendrites in both GeneFunnel and GSVA Poisson and a term related to synapses in both GeneFunnel and the Z-score method, with neither overlaps being exact matches.

Aside from the Z-score method and GeneFunnel, the other methods do appear to have a noticeable level of alignment with one another. In particular, they all seem to focus highly on processes related to CHOP, a factor known to interact with the C/EBP family of transcription factors [19]. In AD, CHOP is implicated to protect neurons from ER stress [20], so this finding may indeed warrant further inspection. However, examination of the actual gene set raises suspicion for the reasons behind its prioritisation by various methods. As can be seen in Figure 7B, this is a very small gene set, composed of just two genes. One gene, *ATF4*, is highly expressed, and is likely solely dependent for driving the large difference in enrichment between the NFT and CTRL conditions. As described in prior sections, GeneFunnel is designed to balance gene set size, increasing the weight of deviance penalty for small gene sets, with gene sets specifically of size 2 carrying the most weight. Indeed, there is a large difference between *ATF4* and the only other gene *DDIT3* and GeneFunnel uses this difference to penalise the gene set highly. This allows for the higher prioritisation of gene sets like Tau Protein Binding, where the magnitude of no singular gene change is comparable to ATF4, but across the 25 genes comprising the gene set, many are increased by some degree in NFT vs. CTRL.

Benchmarking of Computational Efficiency:
Even if a method has high analytical performance compared to others, that method may not be feasible to use if runtime or memory usage is excessive. For this reason, GeneFunnel is implemented in Rcpp (a C++ interface to R) [21, 22] with optimised RcppArmadillo linear algebra libraries [23]. In order to compare computational efficiency across methods, we took the original FACS ssRNAseq data and replicated samples or cells to different sizes and then passed each method through the function mark from the R package bench. All tests were performed with 5 iterations to ensure robustness. Furthermore, when comparing serial and parallel processing, the same framework was used in all methods: BiocParallel.

Three variations of this approach were recorded. For the first, we used a pseudobulked version of the FACS ssRNAseq dataset with 6 total samples and ran all methods using serial processing. The output is summarised in Figure 8A. Next, we took this same data and reran the methods with 60 samples using parallel processing. This output is summarised in Figure 8B. Finally, for the last test, we went back to the original single-cell data without pseudobulking. Using parallel processing, we tested each method on a maximum of 600 cells alongside various subsets of the data. Using six subsets, at each subset the number of cells was halved. For example, while the sixth subset contained 600 cells, the fifth subset contained 300, and so-on. The results of this experiment is captured in Figure 8C.

Inspection of the figures shows that GeneFunnel is the leader in computational efficiency in both runtime and memory usage in all scenarios, although PLAGE is comparable when comparing runtime in single-cell data (Figure 8C). When using serial processing, the median runtime and memory usage of GeneFunnel is 379.54ms and 2.24MB, respectively. The next most performant methods, PLAGE and Z-score have runtimes measured in the seconds and several tens of megabytes of memory usage. ssGSEA was notably unoptimised, taking almost 40 seconds and consuming 10GB of memory. The GSVA methods, while reasonably quick (~2 seconds), also consumed about 8GB memory. When using parallel processing and increasing the number of samples by a factor of 10 (6 to 60 samples), the same efficiency rankings hold true (Figure 8B). GeneFunnel is the quickest by far, taking a median of 12.58s, with PLAGE being the next quickest at 1.37m and Z-score and GSVA Gaussian tied at 2.13m. Similarly to the first experiment, ssGSEA took an excessively long time, at a median of 17.26m to the time of completion.

Discussion:

GeneFunnel introduces a novel approach to functional class scoring that directly addresses limitations in existing methods by incorporating deviation-aware scoring while maintaining sample independence. One of its most significant advantages is that it ensures pathway-level enrichment scores reflect coordinated gene expression rather than being driven by a few highly expressed genes. Many existing methods, such as GSVA and ssGSEA, operate on the assumption that total expression within a gene set is a sufficient proxy for pathway activity. However, this can lead to inflated scores for gene sets where only a subset of genes are highly expressed while others are inactive, producing misleading conclusions about pathway activation. GeneFunnel overcomes this by introducing an internal deviation penalty, which ensures that gene sets exhibiting extreme dispersion do not receive high scores. This property makes it particularly well-suited for cases where internal consistency within a pathway is biologically relevant, such as distinguishing truly co-regulated gene sets from those that are only partially activated.

The benchmarking performed in this work supports the predictability of GeneFunnel in the controlled scenarios, a major advantage over other methods that often use more complicated algorithms, making intuitive reasoning and troubleshooting difficult. In all of the test cases, GeneFunnel outperformed others significantly in capturing differentially enriched gene sets. In addition, it always maintains independence between samples and gene sets. This is important, particularly as datasets are re-analysed, expanded, or meta-analysed with other datasets. Furthermore, while no method can be absolutely quantifiable when working with data that is intrinsically relative, GeneFunnel retains the original range of genes composing gene sets, i.e. a gene set composed of relatively lowly expressed genes will receive a low expression score. This in contrast to methods that solely focus on differential expression like GSVA. Finally, GeneFunnel carries out its function in an efficient manner, ranking above far above peers in terms of runtime and memory usage.

Despite its advantages, GeneFunnel comes with certain theoretical and practical limitations that warrant further consideration. One of the most significant concerns is whether variability within a gene set is truly biologically meaningful. While GeneFunnel penalises pathway scores when gene expression is highly inconsistent within a set, it is important to recognize that gene expression levels exist within intrinsic biological ranges. A gene expressed at low levels relative to others in a pathway is not necessarily inactive, its expression may be at the upper limit of its normal dynamic range, even if its absolute expression is much lower than other genes in the set [24]. Like all gene set enrichment methods, GeneFunnel's accuracy is inherently dependent on the biases and completeness of the gene set database being used. The gene sets in this study were exclusively derived from Gene Ontology (GO), meaning that the benchmarks primarily reflect GO-specific enrichment performance. Since pathway definitions vary across different gene set collections, it remains unclear how well GeneFunnel generalizes beyond GO.

Conclusion:
GeneFunnel provides a simple and effective way to obtain per-sample pathway activity by combining total set activity with a size-aware mean absolute deviation penalty. The statistic remains sample-independent and on an absolute, non-negative scale, which makes scores easy to interpret, straightforward to compare across datasets, and convenient to use with standard pipelines. Across synthetic settings that probe partial activation, variance-only changes and subtle coordinated shifts, and in real single-cell data from Alzheimer's disease brain, the method prioritises biologically plausible pathways while reducing the influence of small, outlier-driven sets. The implementation is fast and memory-efficient, supporting routine use in bulk and single-cell analyses.

Figure Legends:

Figure 1:
(A) High-level schematic of the intent of GeneFunnel. From an initial matrix of genes (or proteins) by samples, and with the provision of an object containing gene sets, the input matrix is transformed into a gene set by sample matrix. (B) Mathematical description of the GeneFunnel algorithm. (C) Rcpp implementation of the singular GeneFunnel function, *calculateScores*.

Figure 2:
(A) Heatmap of a hypothetical gene-by-sample-matrix to simulate different patterns of gene expression. Gray cells indicate NA values. (B) The gene count values underlying the heatmap. The table uses the shinyMatrix R library to allow users to edit values within the web app and update GeneFunnel output in real-time. (C) Heatmap coloured by GeneFunnel scores from the hypothetical data. The entire set of genes (rows) were considered to be a single gene set. This results in a collapse of the original 10 row by 5 column to the 1 row by 5 column matrix seen here. The heatmap contains information below the cells corresponding to the GeneFunnel algorithm: the sum of the gene set, the mean, the deviance penalty (including scaling factor), and the final score.

Figure 3:
(A) Heatmap of a hypothetical gene-by-sample-matrix for use with comparing various functional class scoring methods with GeneFunnel. It is identical to the one in Figure 2 aside from three key points. 1) The sample containing NA values is removed, as all the tested methods fail to run when the input matrix contains NA or missing values. 2) Any gene sets that would have all zeros are modified to have at least one non-zero value (Sample 4), as the tested methods discard such gene sets. 3) During testing, the first and second half of the genes are evaluated as separate gene sets (designated as Gene Set X and Y in the right-side annotations), as well as a gene set encompassing all genes (Gene Set Z). (B) Output of various functional class scoring methods, including GeneFunnel, on the hypothetical matrix. Gene sets correspond to the groupings shown in the right-side annotation of Figure 103. Gray cells indicate NA values produced as output.

Figure 4:
An example of the perturbations for one randomly selected signal set in each of the three signal paradigms. In each heatmap, columns are split by group, rows are clustered within the set, and the colour bar shows centred log2+1 expression. (A) In the “spike” paradigm, half of genes of the signal set in one group have 200 counts added to their signal while the rest remain unchanged. (B) In the “variance” paradigm, set means are preserved but intergene variance of the genes in the signal sets of one group is reduced to 25% of its

original value. (C) In the “coordinated” paradigm, all genes in the signal set shift by a small (0.25 logFC with 0.1 standard deviation) same-direction amount in one group.

Figure 5:
Per-method performance for the each of the three signal paradigms at the FDR threshold of 0.05. Rows list methods, columns report detection counts (TP, FN, TN, FP) and the derived rates (sensitivity, specificity, precision, F1, accuracy, average FDR). Colours correspond to magnitude while bold font marks the highest values within a column.

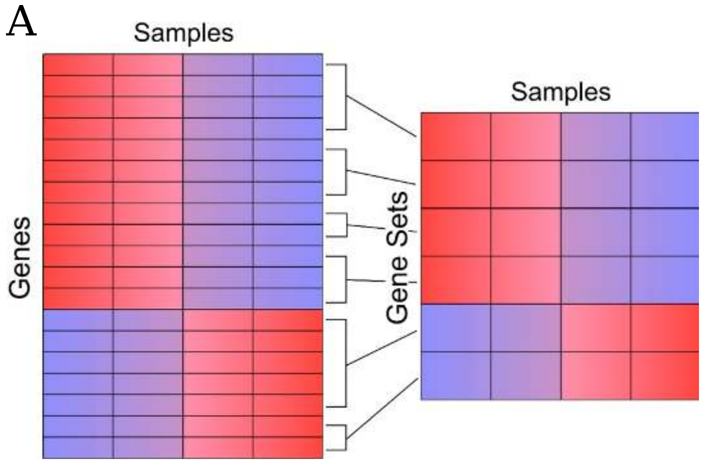
Figure 6:
(A) Heatmaps showing controlled modifications of specific gene sets in pseudobulked data from the Alzheimer's Disease derived real dataset [17]. In each, I introduce a Modified column where the counts for genes in NELF Complex were altered to reduce variability, and the counts for genes in Trace-amine Receptor Activity were increased, as described in the text. The data is log2+1 transformed before plotting. (B) Comparison of the six functional class scoring methods in the reflecting the controlled modifications on the real dataset. Shown are the top two gene sets for each method after sorting by greatest absolute difference between the Modified and Original columns.

Figure 7:
(A) Gene set enrichment results across the six functional class scoring methods when comparing tangle-bearing and non-tangle-bearing neurons in pseudobulked data from the Alzheimer's Disease derived real dataset [17] without modifications. The top five gene sets sorted by absolute difference between NFT and CTRL columns is shown for each method. (B) The expression of genes in selected gene sets that were highlighted by the functional class scoring methods.

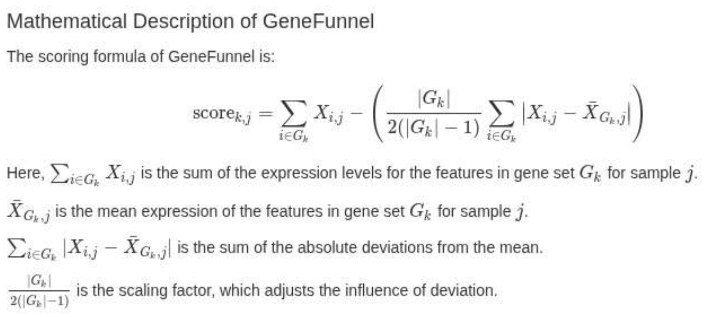
Figure 8:
(A) Runtime and memory usage across the six functional class scoring methods when using serial processing on 6 pseudobulked samples from the Alzheimer's Disease derived real dataset [17]. (B) Runtime using the same data but with parallel processing and 60 rather than 6 samples. Note that when using parallel processing, memory usage cannot be captured using the framework provided by the R package *bench*. (C) Runtime benchmarking on various subsets of samples from the same data but without pseudobulking of the single-cells. At subset 6, the largest subset, 600 cells are used. At each prior subset, the number of cells is halved; 300 cells at subset 5, 150 cells at subset 4, etc.

Figure 1:

A



B



Mathematical Description of GeneFunnel

The scoring formula of GeneFunnel is:

$$\text{score}_{k,j} = \sum_{i \in G_k} X_{i,j} - \left(\frac{|G_k|}{2(|G_k| - 1)} \sum_{i \in G_k} |X_{i,j} - \bar{X}_{G_k,j}| \right)$$

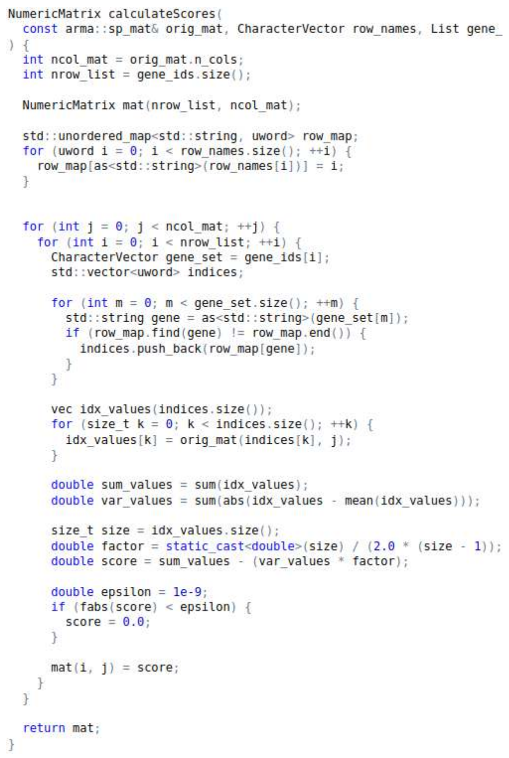
Here, $\sum_{i \in G_k} X_{i,j}$ is the sum of the expression levels for the features in gene set G_k for sample j.

$\bar{X}_{G_k,j}$ is the mean expression of the features in gene set G_k for sample j.

$\sum_{i \in G_k} |X_{i,j} - \bar{X}_{G_k,j}|$ is the sum of the absolute deviations from the mean.

$\frac{|G_k|}{2(|G_k|-1)}$ is the scaling factor, which adjusts the influence of deviation.

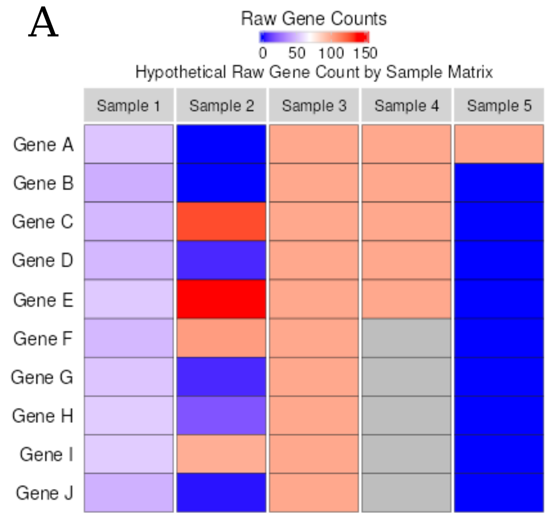
C



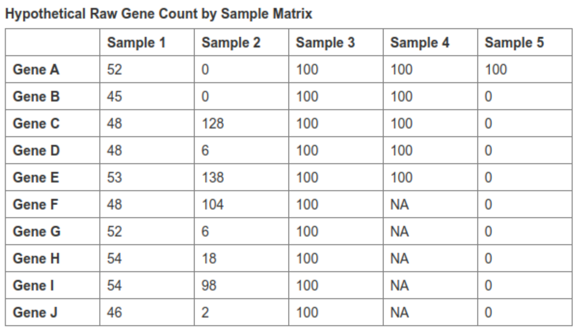
```
NumericMatrix calculateScores(
  const arma::sp_mat& orig_mat, CharacterVector row_names, List gene_
) {
  int ncol_mat = orig_mat.n_cols;
  int nrow_list = gene_ids.size();

  NumericMatrix mat(nrow_list, ncol_mat);

  std::unordered_map<std::string, uword> row_map;
  for (uword i = 0; i < row_names.size(); ++i) {
    row_map[as<std::string>(row_names[i])] = i;
  }


  for (int j = 0; j < ncol_mat; ++j) {
    for (int i = 0; i < nrow_list; ++i) {
      CharacterVector gene_set = gene_ids[i];
      std::vector<uword> indices;

      for (int m = 0; m < gene_set.size(); ++m) {
        std::string gene = as<std::string>(gene_set[m]);
        if (row_map.find(gene) != row_map.end()) {
          indices.push_back(row_map[gene]);
        }
      }

      vec idx_values(indices.size());
      for (size_t k = 0; k < indices.size(); ++k) {
        idx_values[k] = orig_mat(indices[k], j);
      }

      double sum_values = sum(idx_values);
      double var_values = sum(abs(idx_values - mean(idx_values)));

      size_t size = idx_values.size();
      double factor = static_cast<double>(size) / (2.0 * (size - 1));
      double score = sum_values - (var_values * factor);

      double epsilon = 1e-9;
      if (fabs(score) < epsilon) {
        score = 0.0;
      }

      mat(i, j) = score;
    }
  }

  return mat;
}
```

Figure 2:

A



B



Hypothetical Raw Gene Count by Sample Matrix

	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Gene A	52	0	100	100	100
Gene B	45	0	100	100	0
Gene C	48	128	100	100	0
Gene D	48	6	100	100	0
Gene E	53	138	100	100	0
Gene F	48	104	100	NA	0
Gene G	52	6	100	NA	0
Gene H	54	18	100	NA	0
Gene I	54	98	100	NA	0
Gene J	46	2	100	NA	0

C

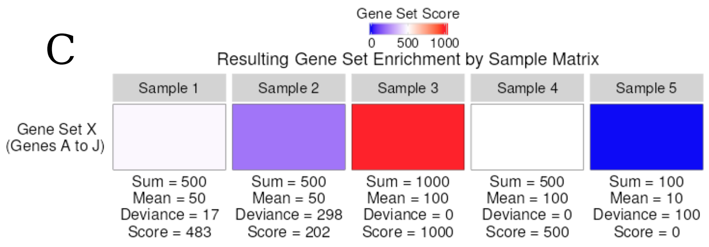
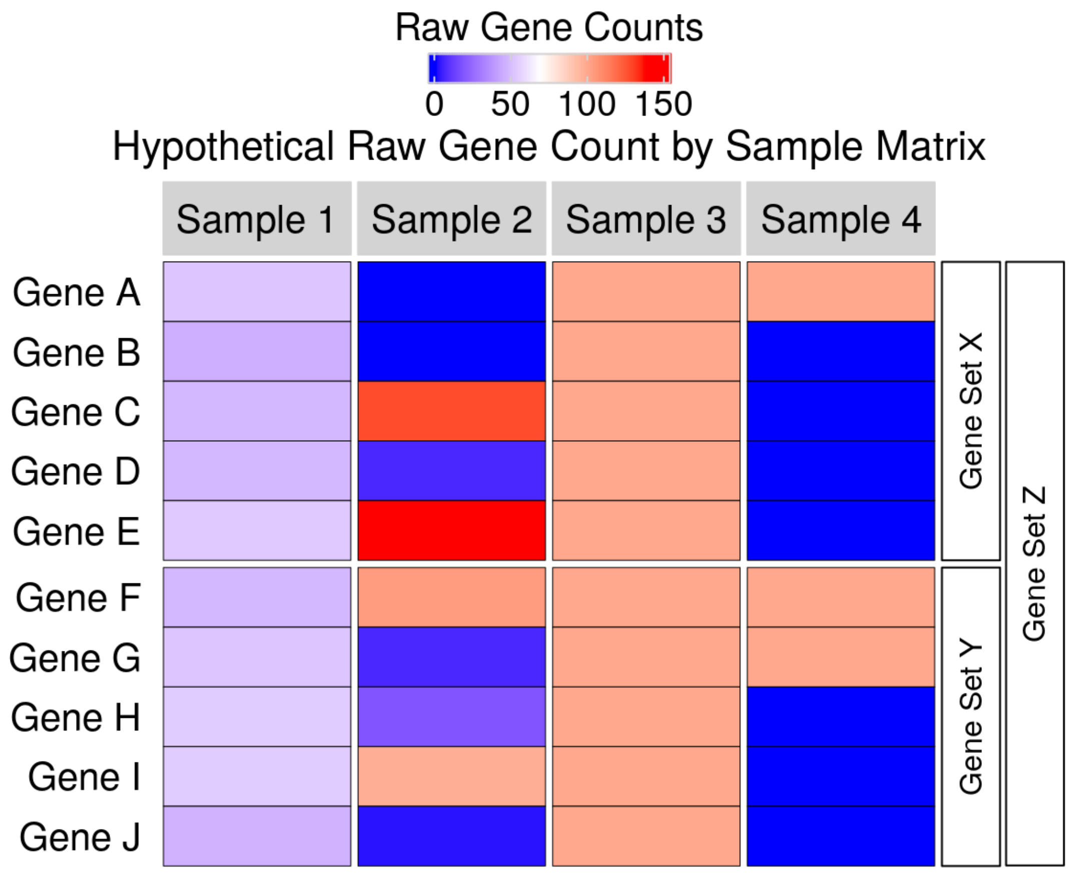


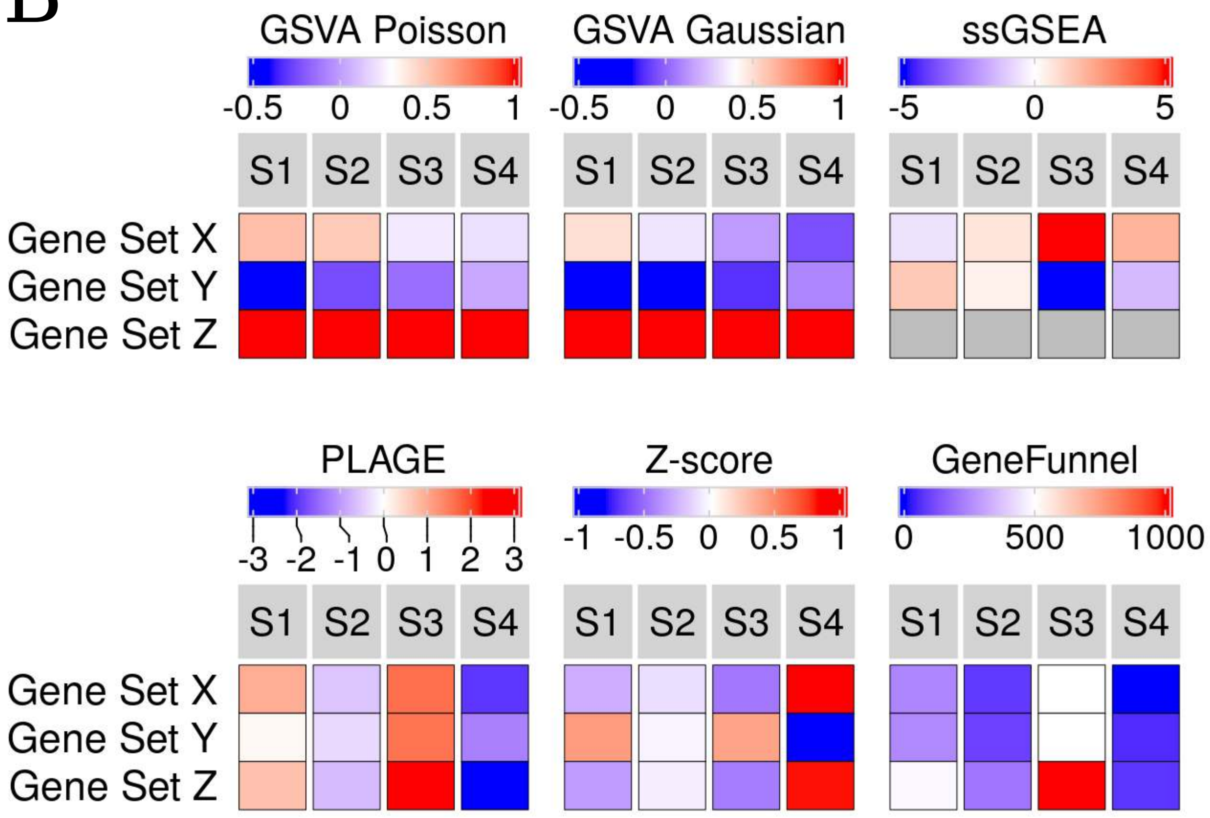
Figure 3:

A



Raw Gene Counts
0
50
100
150
Hypothetical Raw Gene Count by Sample Matrix
Sample 1
Sample 2
Sample 3
Sample 4
Gene A
Gene B
Gene C
Gene D
Gene E
Gene F
Gene G
Gene H
Gene I
Gene J
Gene Set X
Gene Set Y
Gene Set Z

B



GSVA Poisson
-0.5
0
0.5
1
GSVA Gaussian
-0.5
0
0.5
1
ssGSEA
-5
0
5
S1
S2
S3
S4
Gene Set X
Gene Set Y
Gene Set Z
PLAGE
-3
-2
-1
0
1
2
3
Z-score
-1
-0.5
0
0.5
1
GeneFunnel
0
500
1000

Figure 4:

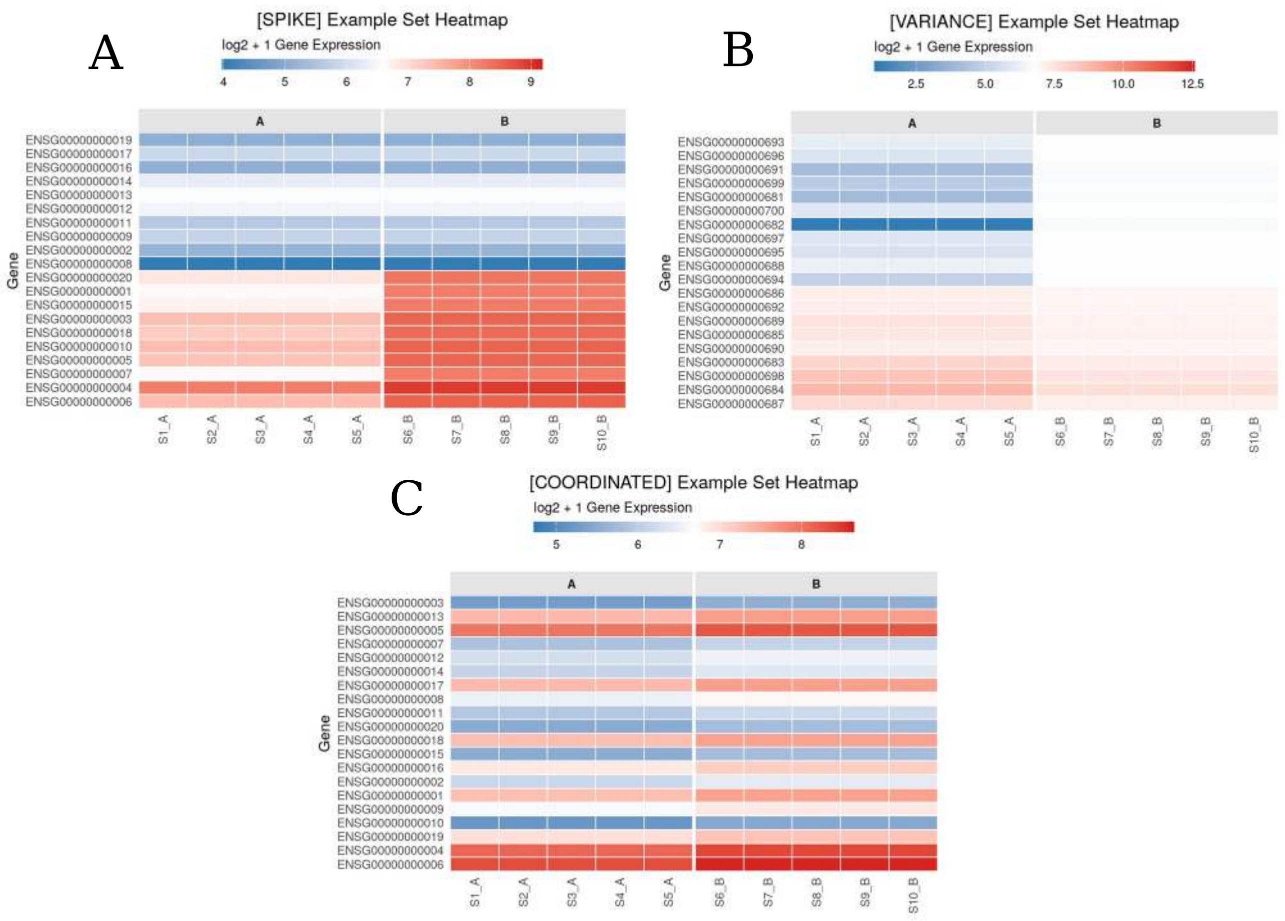
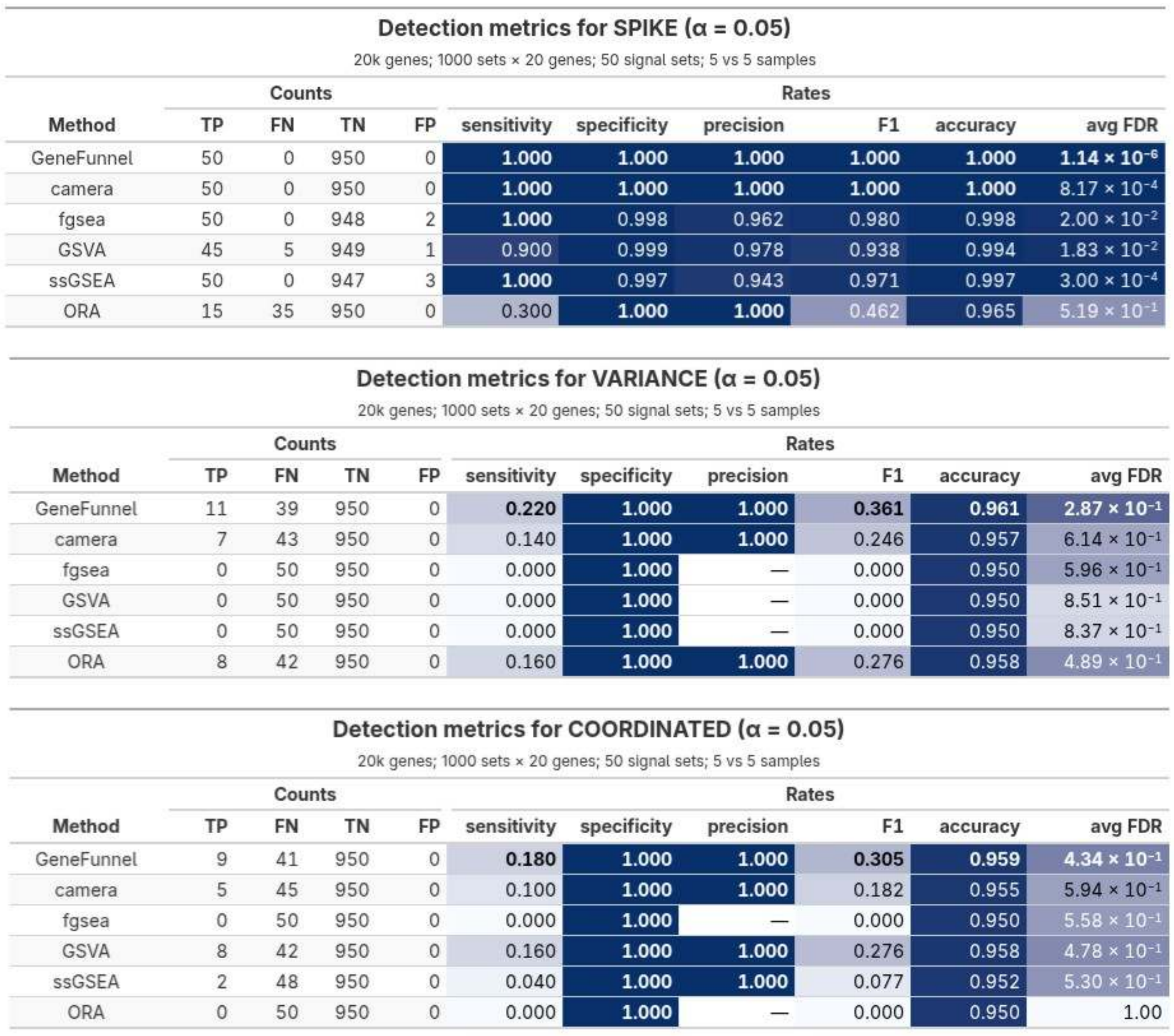

A
[SPIKE] Example Set Heatmap
log2 + 1 Gene Expression
B
[VARIANCE] Example Set Heatmap
log2 + 1 Gene Expression
C
[COORDINATED] Example Set Heatmap
log2 + 1 Gene Expression
Gene
A
B
S1_A
S2_A
S3_A
S4_A
S5_A
S6_B
S7_B
S8_B
S9_B
S10_B

Figure 5:



Detection metrics for SPIKE (α = 0.05)

20k genes; 1000 sets × 20 genes; 50 signal sets; 5 vs 5 samples

	Counts				Rates					
Method	TP	FN	TN	FP	sensitivity	specificity	precision	F1	accuracy	avg FDR
GeneFunnel	50	0	950	0	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.14×10^{-6}**
camera	50	0	950	0	1.000	1.000	1.000	1.000	1.000	8.17×10^{-4}
fgsea	50	0	948	2	1.000	0.998	0.962	0.980	0.998	2.00×10^{-2}
GSVA	45	5	949	1	0.900	0.999	0.978	0.938	0.994	1.83×10^{-2}
ssGSEA	50	0	947	3	1.000	0.997	0.943	0.971	0.997	3.00×10^{-4}
ORA	15	35	950	0	0.300	1.000	1.000	0.462	0.965	5.19×10^{-1}

Detection metrics for VARIANCE (α = 0.05)

20k genes; 1000 sets × 20 genes; 50 signal sets; 5 vs 5 samples

	Counts				Rates					
Method	TP	FN	TN	FP	sensitivity	specificity	precision	F1	accuracy	avg FDR
GeneFunnel	11	39	950	0	**0.220**	**1.000**	**1.000**	**0.361**	**0.961**	**2.87×10^{-1}**
camera	7	43	950	0	0.140	1.000	1.000	0.246	0.957	6.14×10^{-1}
fgsea	0	50	950	0	0.000	1.000	—	0.000	0.950	5.96×10^{-1}
GSVA	0	50	950	0	0.000	1.000	—	0.000	0.950	8.51×10^{-1}
ssGSEA	0	50	950	0	0.000	1.000	—	0.000	0.950	8.37×10^{-1}
ORA	8	42	950	0	0.160	1.000	1.000	0.276	0.958	4.89×10^{-1}

Detection metrics for COORDINATED (α = 0.05)

20k genes; 1000 sets × 20 genes; 50 signal sets; 5 vs 5 samples

	Counts				Rates					
Method	TP	FN	TN	FP	sensitivity	specificity	precision	F1	accuracy	avg FDR
GeneFunnel	9	41	950	0	**0.180**	**1.000**	**1.000**	**0.305**	**0.959**	**4.34×10^{-1}**
camera	5	45	950	0	0.100	1.000	1.000	0.182	0.955	5.94×10^{-1}
fgsea	0	50	950	0	0.000	1.000	—	0.000	0.950	5.58×10^{-1}
GSVA	8	42	950	0	0.160	1.000	1.000	0.276	0.958	4.78×10^{-1}
ssGSEA	2	48	950	0	0.040	1.000	1.000	0.077	0.952	5.30×10^{-1}
ORA	0	50	950	0	0.000	1.000	—	0.000	0.950	1.00

Figure 6:

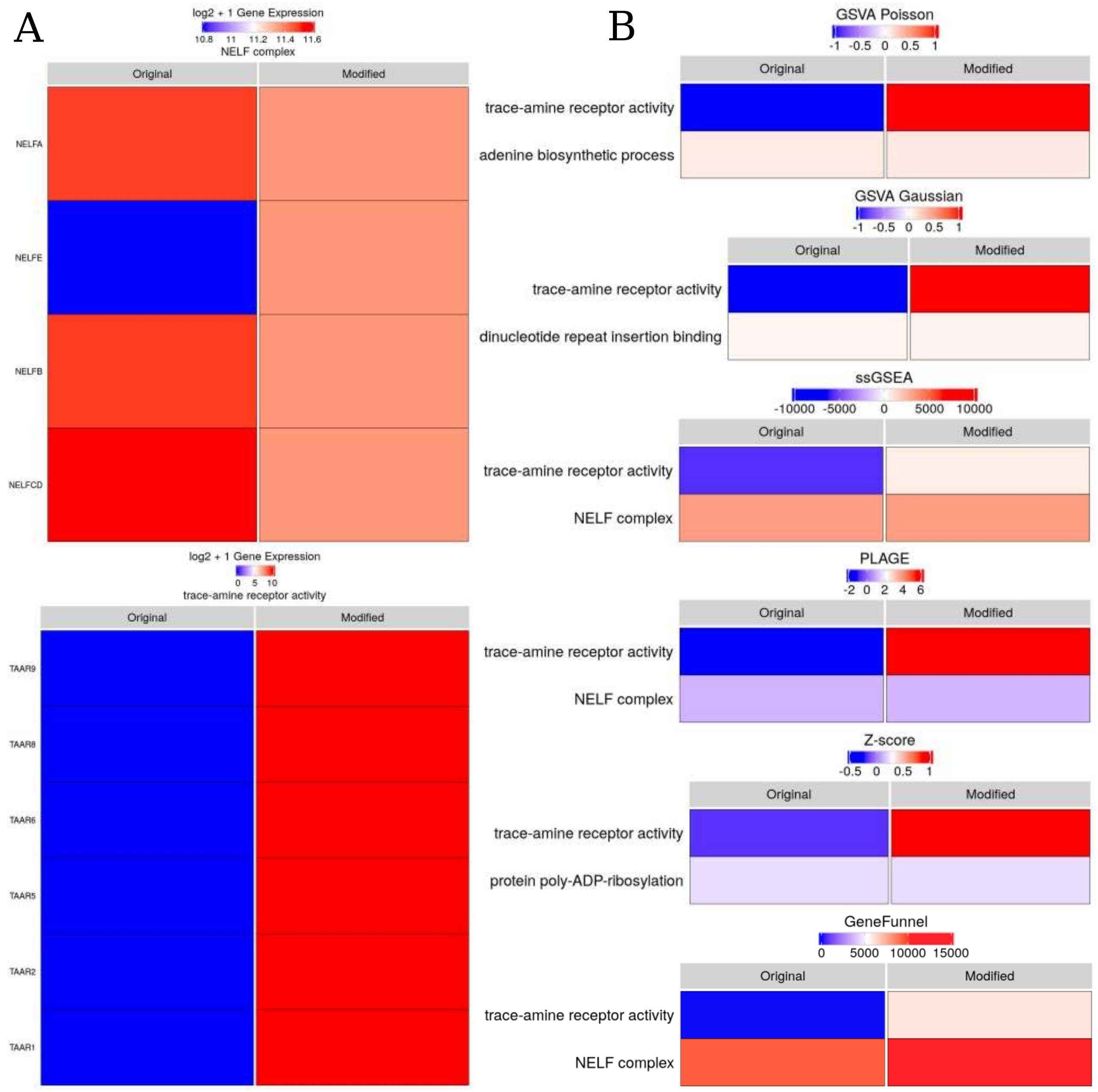

A
log2 + 1 Gene Expression
10.8 11 11.2 11.4 11.6
NELF complex
Original
Modified
NELFA
NELFE
NELFB
NELFCD
log2 + 1 Gene Expression
0 5 10
trace-amine receptor activity
Original
Modified
TAAR9
TAAR8
TAAR6
TAAR5
TAAR2
TAAR1
B
GSVA Poisson
-1 -0.5 0 0.5 1
Original
Modified
trace-amine receptor activity
adenine biosynthetic process
GSVA Gaussian
-1 -0.5 0 0.5 1
Original
Modified
trace-amine receptor activity
dinucleotide repeat insertion binding
ssGSEA
-10000 -5000 0 5000 10000
Original
Modified
trace-amine receptor activity
NELF complex
PLAGE
-2 0 2 4 6
Original
Modified
trace-amine receptor activity
NELF complex
Z-score
-0.5 0 0.5 1
Original
Modified
trace-amine receptor activity
protein poly-ADP-ribosylation
GeneFunnel
0 5000 10000 15000
Original
Modified
trace-amine receptor activity
NELF complex

Figure 7:

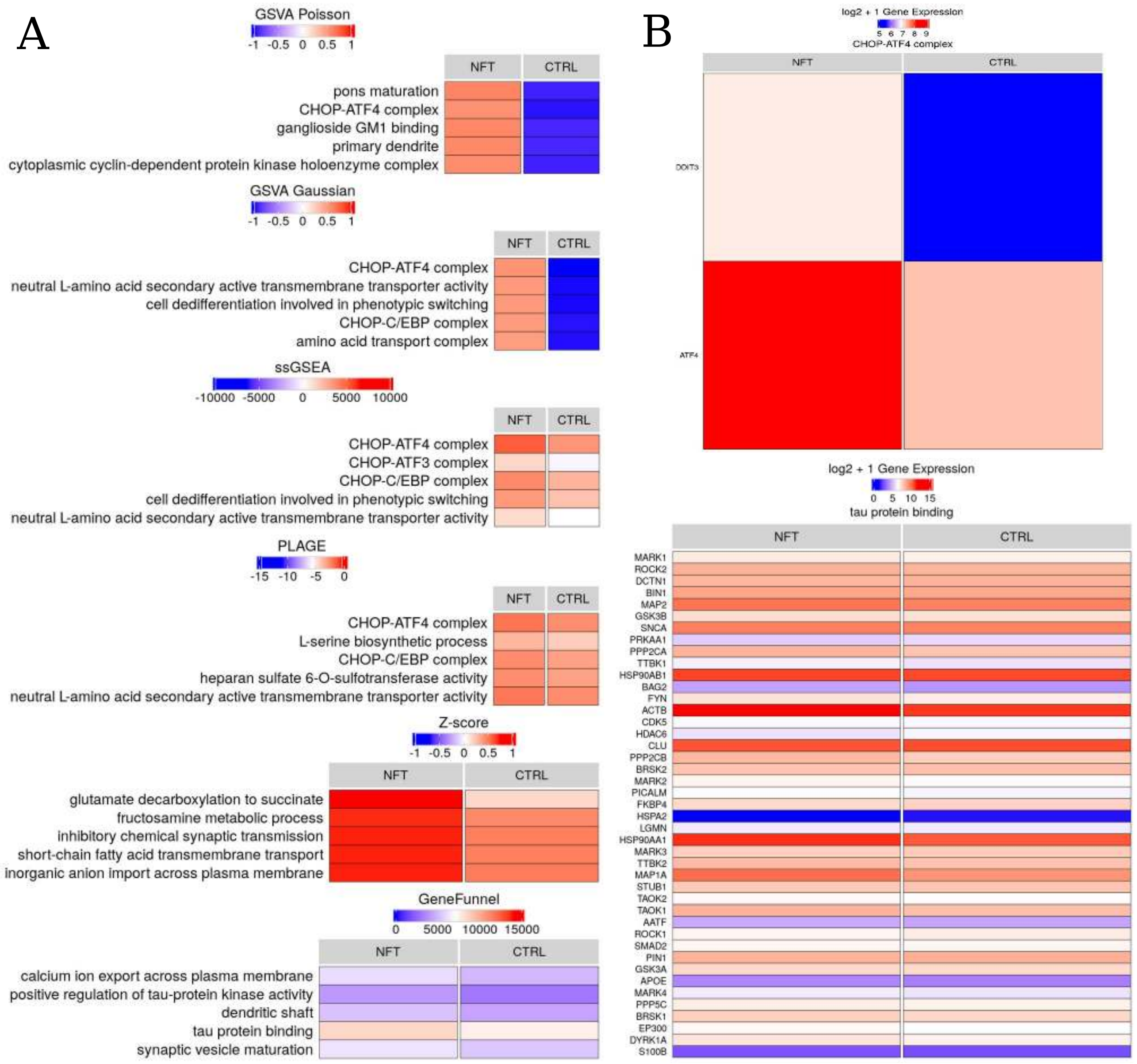

A
GSVA Poisson
NFT
CTRL
pons maturation
CHOP-ATF4 complex
ganglioside GM1 binding
primary dendrite
cytoplasmic cyclin-dependent protein kinase holoenzyme complex
GSVA Gaussian
CHOP-ATF4 complex
neutral L-amino acid secondary active transmembrane transporter activity
cell dedifferentiation involved in phenotypic switching
CHOP-C/EBP complex
amino acid transport complex
ssGSEA
CHOP-ATF4 complex
CHOP-ATF3 complex
CHOP-C/EBP complex
cell dedifferentiation involved in phenotypic switching
neutral L-amino acid secondary active transmembrane transporter activity
PLAGE
CHOP-ATF4 complex
L-serine biosynthetic process
CHOP-C/EBP complex
heparan sulfate 6-O-sulfotransferase activity
neutral L-amino acid secondary active transmembrane transporter activity
Z-score
glutamate decarboxylation to succinate
fructosamine metabolic process
inhibitory chemical synaptic transmission
short-chain fatty acid transmembrane transport
inorganic anion import across plasma membrane
GeneFunnel
calcium ion export across plasma membrane
positive regulation of tau-protein kinase activity
dendritic shaft
tau protein binding
synaptic vesicle maturation
B
log2 + 1 Gene Expression
CHOP-ATF4 complex
DDIT3
ATF4
log2 + 1 Gene Expression
tau protein binding
MARK1
ROCK2
DCTN1
BIN1
MAP2
GSK3B
SNCA
PRKAA1
PPP2CA
TTBK1
HSP90AB1
BAG2
FYN
ACTB
CDK5
HDAC6
CLU
PPP2CB
BRSK2
MARK2
PICALM
FKBP4
HSPA2
LGMN
HSP90AA1
MARK3
TTBK2
MAP1A
STUB1
TAOK2
TAOK1
AATF
ROCK1
SMAD2
PIN1
GSK3A
APOE
MARK4
PPP5C
BRSK1
EP300
DYRK1A
S100B

Figure 8:

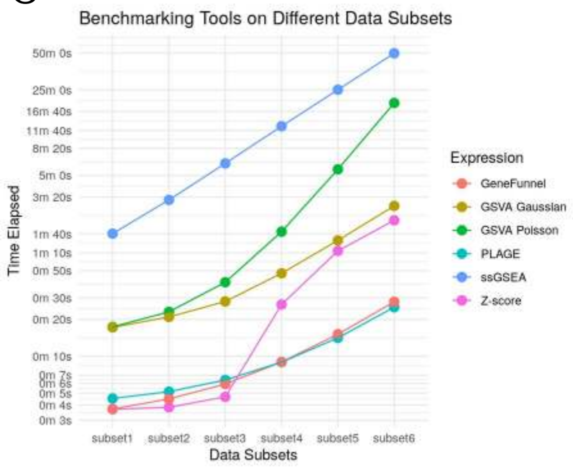
A

```
  expression         min    median `itr/sec` mem_alloc `gc/sec`
  <chr>         <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
1 GSVA Poisson     2.91s    2.94s     0.275    8.57GB    0.659
2 GSVA Gaussian     2.8s    2.85s     0.347    8.57GB    0.833
3 ssGSEA          39.06s   39.64s    0.0251   10.02GB    0.126
4 PLAGE            4.67s    4.72s     0.212   23.67MB    0.763
5 Z-score           1.7s    1.72s     0.578   76.97MB    1.16
6 GeneFunnel    375.91ms 379.54ms     2.57     2.24MB    0.514
```

B

```
  expression         min    median `itr/sec` mem_alloc `gc/sec`
  <chr>         <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
1 GSVA Poisson     5.35m    5.68m  0.00297          NA  0.00297
2 GSVA Gaussian    2.02m    2.13m  0.00789          NA  0.00789
3 ssGSEA          17.22m   17.26m  0.000964         NA  0
4 PLAGE            1.36m    1.37m  0.0122           NA  0.0244
5 Z-score          2.12m    2.13m  0.00782          NA  0
6 GeneFunnel      12.45s   12.58s  0.0794           NA  0
```

C



References:

1. Bayerlová M. Comparative study on gene set and pathway topology-based enrichment methods. BMC Bioinformatics. 2015;:15.

2. Bull C, Byrne RM, Fisher NC, Corry SM, Amirkhah R, Edwards J, et al. Evaluation of Gene Set Enrichment Analysis (GSEA) tools highlights the value of single sample approaches over pairwise for robust biological discovery. bioRxiv. 2024. https://doi.org/10.1101/2024.03.15.585228.

3. Candia J, Ferrucci L. Assessment of Gene Set Enrichment Analysis using curated RNA-seq-based benchmarks. PLoS ONE. 2024;19:e0302696. https://doi.org/10.1371/journal.pone.0302696.

4. Das S, McClain CJ, Rai SN. Fifteen Years of Gene Set Analysis for High-Throughput Genomic Data: A Review of Statistical Approaches and Future Challenges. Entropy. 2020;22:427. https://doi.org/10.3390/e22040427.

5. Geistlinger L, Csaba G, Santarelli M, Ramos M, Schiffer L, Turaga N, et al. Toward a gold standard for benchmarking gene set enrichment analysis. Briefings in Bioinformatics. 2020;:bbz158. https://doi.org/10.1093/bib/bbz158.

6. Khatri P, Sirota M, Butte AJ. Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges. PLoS Comput Biol. 2012;8:e1002375. https://doi.org/10.1371/journal.pcbi.1002375.

7. Maleki F, Ovens K, Hogan DJ, Kusalik AJ. Gene Set Analysis: Challenges, Opportunities, and Future Research. Front Genet. 2020;11:654. https://doi.org/10.3389/fgene.2020.00654.

8. Wijesooriya K, Jadaan SA, Perera KL, Kaur T, Ziemann M. Urgent need for consistent standards in functional enrichment analysis. PLoS Comput Biol. 2022;18:e1009935. https://doi.org/10.1371/journal.pcbi.1009935.

9. Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences. 2005;102:15545–50. https://doi.org/10.1073/pnas.0506580102.

10. Hänzelmann S, Castelo R, Guinney J. GSVA: gene set variation analysis for microarray and RNA-Seq data. BMC Bioinformatics. 2013;14:7. https://doi.org/10.1186/1471-2105-14-7.

11. Barbie DA, Tamayo P, Boehm JS, Kim SY, Moody SE, Dunn IF, et al. Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. Nature. 2009;462:108–12. https://doi.org/10.1038/nature08460.

12. Tomfohr J, Lu J, Kepler TB. Pathway level analysis of gene expression using singular value decomposition. BMC Bioinformatics. 2005;6:225. https://doi.org/10.1186/1471-2105-6-225.

13. Raudvere U, Kolberg L, Kuzmin I, Arak T, Adler P, Peterson H, et al. g:Profiler: a web server for functional enrichment analysis and conversions of

gene lists (2019 update). Nucleic Acids Research. 2019;:gkz369. https://doi.org/10.1093/nar/gkz369.

14. Phipson B, Lee S, Majewski IJ, Alexander WS, Smyth GK. Robust hyperparameter estimation protects against hypervariable genes and improves power to detect differential expression. Ann Appl Stat. 2016;10:946–63. https://doi.org/10.1214/16-AOAS920.

15. Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research. 2015;43:e47–e47. https://doi.org/10.1093/nar/gkv007.

16. Korotkevich G, Sukhov V, Budin N, Shpak B, Artyomov MN, Sergushichev A. Fast gene set enrichment analysis. bioRxiv. 2016. https://doi.org/10.1101/060012.

17. Otero-Garcia M, Mahajani SU, Wakhloo D, Tang W, Xue Y-Q, Morabito S, et al. Molecular signatures underlying neurofibrillary tangle susceptibility in Alzheimer’s disease. Neuron. 2022;110:2929-2948.e8. https://doi.org/10.1016/j.neuron.2022.06.021.

18. McCarthy DJ, Campbell KR, Lun ATL, Wills QF. Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. Bioinformatics. 2017;:btw777. https://doi.org/10.1093/bioinformatics/btw777.

19. Sok J, Wang X-Z, Batchvarova N, Kuroda M, Harding H, Ron D. CHOP-Dependent Stress-Inducible Expression of a Novel Form of Carbonic Anhydrase VI. Molecular and Cellular Biology. 1999;19:495–504. https://doi.org/10.1128/MCB.19.1.495.

20. Aceves M, Granados J, Leandro AC, Peralta J, Glahn DC, Williams-Blangero S, et al. Role of Neurocellular Endoplasmic Reticulum Stress Response in Alzheimer’s Disease and Related Dementias Risk. Genes. 2024;15:569. https://doi.org/10.3390/genes15050569.

21. Eddelbuettel D, Balamuta JJ. Extending R with C++: A Brief Introduction to Rcpp. The American Statistician. 2018;72:28–36. https://doi.org/10.1080/00031305.2017.1375990.

22. Eddelbuettel D, François R. Rcpp: Seamless R and C++ Integration. Journal of Statistical Software. 2011.

23. Eddelbuettel D, Sanderson C. RcppArmadillo: Accelerating R with high-performance C++ linear algebra. Computational Statistics & Data Analysis. 2014;71:1054–63. https://doi.org/10.1016/j.csda.2013.02.005.

24. Buccitelli C, Selbach M. mRNAs, proteins and the emerging principles of gene expression control. Nat Rev Genet. 2020;21:630–44. https://doi.org/10.1038/s41576-020-0258-4.