Gene Set Scoring on the Nearest Neighbor Graph (gssnng) for Single Cell RNA-seq (scRNA-seq).

David L Gibbs[1],*, Michael K Strasser[2] and Sui Huang[2]

[1]Shmulevich Lab, Institute for Systems Biology, Seattle WA, [2]Huang Lab, Institute for Systems Biology, Seattle WA

**To whom correspondence should be addressed.*

Abstract

Motivation: Gene set scoring is a common task in bioinformatics, but the sparsity of data points in single cell gene expression creates difficulties.

Results: By applying matrix smoothing to the nearest neighbor graph of cells, high quality gene set scores can be produced on a per-cell level which is useful for visualization and statistical analysis. Importantly, within UMAP visualizations, the method preserves score-gradients across clusters of cells.

Contact: david.gibbs@isbscience.org

Software: gssnng is available on the python package index (PyPI). It can be installed using 'pip install gssnng'.

More information and demo notebook: See https://github.com/gibbsdavidl/gssnng

1. Introduction

Genes are not independent entities, rather they operate as sets, pathways, and in modules, making gene set analysis intuitive in bioinformatics. Such "gene set enrichment" determines whether a previously defined set of genes shows associated patterns of expression given a sample grouping (1–3). Among the many algorithms, single sample methods compute a score independently for each sample, which is then used in visualizations or statistical analysis(4). Since each sample has a score, or a set of scores from multiple gene sets, the analyst has flexibility in their choice of downstream techniques.

However, many methods are based on bulk expression data, where millions of cells are batch processed, providing a measure on nearly all genes. In single cell transcriptomics, the data is noisy and sparse; for a given cell only a small percentage of total known genes are measured and commonly have a small number of counts(5). This causes difficulties in gene set analysis. For example, some methods are rank based, and given the high proportion of zeros or integer collisions – how does one rank genes with so many identical measures?

An additional problem: in single cell analysis, visualization methods UMAP and tSNE represent transcriptomic variation across the clusters in UMAP or tSNE space (6–8). For example, gradations of gene expression across a cluster likely indicate cell states or subtypes(9-11). Ideally a gene set scoring method should preserve this variation as it may be related to the biology(12-15).

The sparse and noisy nature of single-cell transcriptomic data also makes computing differential expression difficult(5). However, recent work has shown that these biases can be avoided by using 'pseudobulk' profiles, which are gene expression profiles created simply by summing across cells(10). The summation creates higher abundances, lower noise floors, breaks many of the ties in expression counts, and allows for better overlap with gene sets since more genes have values.

To address these issues, we have developed software that produces a gene set score for each individual cell, addressing problems of low read counts and the many zeros and retains gradations that remain visible in UMAP plots. The method works by using a nearest neighbor graph in gene expression space to smooth the count matrix. The smoothed expression profiles are then used in single sample gene set scoring calculations.

Our software, gssnng, is written in python and works with AnnData objects, the standard format for scRNAseq data in Scanpy(16). The gssnng package contains a collection of scoring methods that can be selected according to the particular context.

2. Methods

2.1 Overview

Using gssnng, large collections of cells can be scored quickly even on a modest desktop. The method uses the nearest neighbor graph (kNN) of cells to smooth the gene expression count matrix which decreases sparsity and improves geneset scoring.

Algorithm in brief

I. Group cells by attributes like cell type, diagnosis, or condition (optional)

II. Recompute kNN graph within groups (optional)

III. Smooth the gene expression counts matrix

IV. In parallel, by group, score cells for each gene set

2.2 Grouping cells

When combining or integrating different datasets (e.g. different tumor subtypes, different patients) into a single dataset, it may be beneficial to first group cells into phenotypically similar subsets before building the kNN graph. This leads to a smoothed count matrix that is specific to a selected phenotype. The default is to group cells by their cluster label, so smoothing only involves transcriptionally similar cells.

2.3 The nearest neighbor graph

Within each group of cells, the analyst provides the desired parameters in building the kNN, most importantly the number of neighbors, k. In practice, setting k to between 32 and 128 usually works well. However, k should be determined through experimentation as it might depend on the particularities of the data. The graph is produced using the "scanpy.pp.neighbors" function found in Scanpy.

2.4 Matrix smoothing

As single cell expression data is often sparse, for a given cell and gene set, only a fraction of genes in the set have values. To address this issue, we apply nearest neighbor smoothing to produce a smoothed gene expression profile for each cell based on its neighbors. This assumes that gene expression varies smoothly along the data manifold (approximated by the nearest neighbor graph) and hence we can use information from neighboring cells to denoise the expression profiles of cells corrupted by technical noise (similar to Gaussian smoothing in images, which assumes that pixel intensities vary smoothly in space)(17,22). (see Suppl.)

In order to retain variability in cell-level scores across the cluster of cells, neighborhood sub-sampling is applied. Two nearby cells can have different samples producing scores that retain variability in the UMAP score gradients that can be carried into statistical tests downstream.

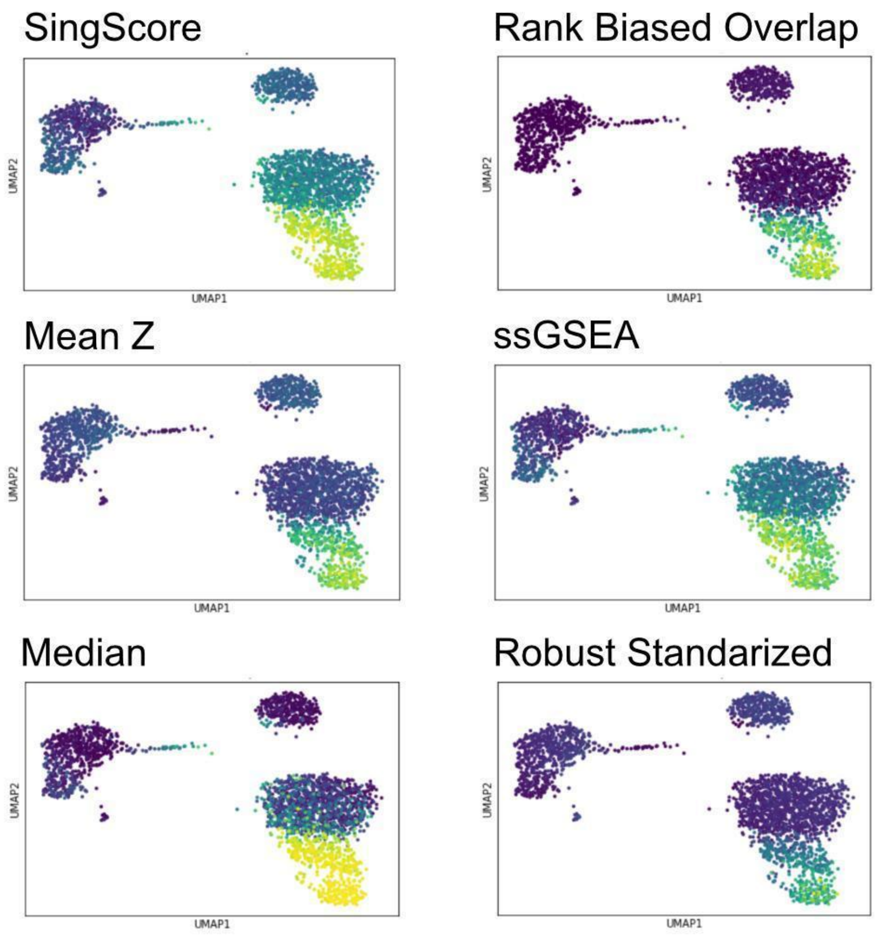

Figure 1. Gradients of gene set scores are preserved and useful for visualization in UMAPs and TSNEs. This example shows scores for a CD8 T cell specific signature using different scoring methods. Example data from 10x genomics "3k PBMCs from a Healthy Donor" (Data)

2.5 Scoring Functions

Once the smoothed matrix is available, several methods are available for gene set scoring. These functions operate on the smoothed values and some functions can be given parameters. Methods for scoring include SingScore(18) ssGSEA(4,19), Rank Biased Overlap(20), Mean Z score, Average Score, Median Score, Summed Up(21). For each gene set, the scores are saved in the AnnData.obs table, with one column per gene set, facilitating visualization. The granular structure of the data allows for parallel processing; python's multiprocessing starmap asynchronously processes each group of cells given a list of gene sets.

3. Conclusion

This package, gssnng, provides analysts a convenient yet powerful tool for gene set and signature scoring on the single cell level. To provide flexibility, a range of function types are available that can be selected according to the data and project goals.

Funding

This work was supported by a Cancer Research UK Grand Challenge (CRUK grant 29068). Additional support from the Tlsty Lab at UCSF (PI Tlsty, CRUK grant 27145), McGill University Thoracic and Upper GI Cancer Research Laboratories (PI Ferri, CRUK grant 29071) and the Advanced Genomic Technologies Laboratory (PI Ragoussis, CRUK grant 29078) is greatly appreciated.

References

1. Hung JH, Yang TH, Hu Z, Weng Z, DeLisi C. Gene set enrichment analysis: performance evaluation and usage guidelines. Brief Bioinform. 2012 May;13(3):281–91.

2. Maciejewski H. Gene set analysis methods: statistical models and methodological differences. Brief Bioinform. 2014 Jul;15(4):504–18.

3. Maleki F, Ovens K, Hogan DJ, Kusalik AJ. Gene Set Analysis: Challenges, Opportunities, and Future Research. Front Genet. 2020;11.

4. 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 Nov;462(7269):108–12.

5. Kim TH, Zhou X, Chen M. Demystifying "drop-outs" in single-cell UMI data. Genome Biol. 2020 Aug 6;21(1):196.

6. Becht E, McInnes L, Healy J, Dutertre CA, Kwok IW, Ng LG, et al. Dimensionality reduction for visualizing single-cell data using UMAP. Nat Biotechnol. 2019;37(1):38–44.

7. Kobak D, Berens P. The art of using t-SNE for single-cell transcriptomics. Nat Commun. 2019;10(1):1–14.

8. Kobak D, Linderman GC. Initialization is critical for preserving global data structure in both t-SNE and UMAP. Nat Biotechnol. 2021;39(2):156–7.

9. Dorrity MW, Saunders LM, Queitsch C, Fields S, Trapnell C. Dimensionality reduction by UMAP to visualize physical and genetic interactions. Nat Commun. 2020 Mar 24;11(1):1537.

10. Squair JW, Gautier M, Kathe C, Anderson MA, James ND, Hutson TH, et al. Confronting false discoveries in single-cell differential expression. Nat Commun. 2021 Sep 28;12(1):5692.

11. Dijk D van, Sharma R, Nainys J, Yim K, Kathail P, Carr AJ, et al. Recovering Gene Interactions from Single-Cell Data Using Data Diffusion. Cell. 2018 Jul 26;174(3):716-729.e27.

12. Bargaje R, Trachana K, Shelton MN, McGinnis CS, Zhou JX, Chadick C, et al. Cell population structure prior to bifurcation predicts efficiency of directed differentiation in human induced pluripotent cells. Proc Natl Acad Sci U S A. 2017 Feb 28;114(9):2271–6.

13. Kalmar T, Lim C, Hayward P, Muñoz-Descalzo S, Nichols J, Garcia-Ojalvo J, et al. Regulated fluctuations in nanog expression mediate cell fate decisions in embryonic stem cells. PLoS Biol. 2009 Jul;7(7):e1000149.

14. Canham MA, Sharov AA, Ko MSH, Brickman JM. Functional heterogeneity of embryonic stem cells revealed the rough translational amplification of an early endodermal transcript. PLoS Biol. 2010 May 25;8(5):e1000379.

15.Chang HH, Hemberg M, Barahona M, Ingber DE, Huang S. Transcriptome-wide noise controls lineage choice in mammalian progenitor cells. Nature. 2008 May 22;453(7194):544–7.

16. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. 2018 Feb;19(1):15.

17. Shapiro LG, Stockman GC. Computer vision. Vol. 3. Prentice Hall New Jersey; 2001.

18. Foroutan M, Bhuva DD, Lyu R, Horan K, Cursons J, Davis MJ. Single sample scoring of molecular phenotypes. BMC Bioinformatics. 2018 Nov 6;19(1):404.

19. Abazeed ME, Adams DJ, Hurov KE, Tamayo P, Creighton CJ, Sonkin D, et al. Integrative radiogenomic profiling of squamous cell lung cancer. Cancer Res. 2013 Oct 15;73(20):6289–98.

20. Webber W, Moffat A, Zobel J. A similarity measure for indefinite rankings. ACM Trans Inf Syst. 2010 Nov 23;28(4):20:1-20:38.

21. Pont F, Tosolini M, Fournié JJ. Single-Cell Signature Explorer for comprehensive visualization of single cell signatures across scRNA-seq datasets. Nucleic Acids Res. 2019 Dec 2;47(21):e133.

22. Ronen J, Akalin A. netSmooth: Network-smoothing based imputation for single cell RNA-seq. F1000Res. 2018 Jul 10;7:8.

23. Sarkar A, Stephens M. Separating measurement and expression models clarifies confusion in single-cell RNA sequencing analysis. Nat Genet. 2021 Jun;53(6):770–7.

24. Liberzon A, Birger C, Thorvaldsdóttir H, Ghandi M, Mesirov JP, Tamayo P. The Molecular Signatures Database (MSigDB) hallmark gene set collection. Cell Syst. 2015 Dec 23;1(6):417–25.

25. 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. Proc Natl Acad Sci U S A. 2005 Oct 25;102(43):15545–50.

Data: https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/pbmc3k?

SUPPLEMENTARY MATERIALS

Sampling and smoothing

The first step involves generating the nearest neighbor graph of cells prior to matrix smoothing. The nearest neighbor graph can be defined using a binary adjacency matrix, each entry of the matrix stating whether two cells are neighbors or not. Also a weighted graph is possible where neighbors are more strongly weighted if they are closer (in transcriptional space). The two options are selected using the 'smooth_mode' parameter. Our implementation uses the scipy sparse matrix library as an effort to be mindful of memory use (6).

With either the binary or weighted adjacency matrix, when smoothing gene expression counts, we sample a set of cells from the cell's neighborhood by randomly dismissing cells through setting some adjacencies or weights to zero. To smoothed expression matrix $M \in \mathbb{R}^{n \times p}$, ($n$ cells, p genes) is calculated via matrix multiplication:

$$M = SX$$

where $S \in \mathbb{R}^{n \times n}$ is the normalized and subsampled adjacency matrix for the nearest neighbor graph, and $X \in \mathbb{R}^{n \times p}$ is the cell by gene matrix of gene expression counts.

The collection of thousands of gene sets at MSigDB often contain on the order of several hundred genes (24,25). Commonly, the number of genes expressed by a single cell is sparse, resulting in weak overlaps between a given cell and gene set. Through application of the nearest neighbor smoothing, the quantity of genes overlapping is greatly increased (Figure S1). Using the "3k PBMCs from 10x Genomics", gene overlap was assessed for each of 2700 cells using the "HALLMARK IL6 JAK STAT3 SIGNALING" gene set which contains 87 genes. With no smoothing, the average was 6.5 genes per cell (out of 87), making reliable estimation of a score difficult Smoothing with only 20 neighbors increased the overlap about five-fold to 33.6 genes.

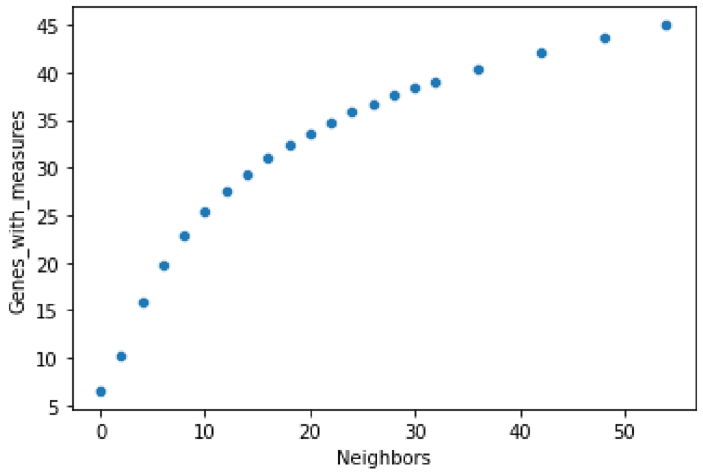

Figure S1. The average number of genes with measures

Function definitions

For a given cell i, X_i is the vector of values which are either gene expression counts or ranked values.Then our gene set, G, is a set of indices over X_c, written $j \in G$.

Our goal is producing a score per cell, S_{iG}, where i indexes the cell and G is the gene set of interest.

1. Summed Up:

Here we sum up the ranks or counts. This is similar to the SCSE(21) method without dividing by total UMIs.

$$S_{iG} = \sum_{j \in G} X_{ij}$$

2. Average Score.

The average score takes the mean of counts or ranks given the gene set.

$$S_{iG} = \frac{1}{|G|} \sum_{j \in G} X_{ij}$$

3. Median Score.

Similarly, the median score takes the median counts or rank for the gene set.

$$S_{iG} = med(X_{ij}) \text{ for } j \in G$$

4. Mean Z.

The score is based on computing a Z score for each gene in the set, and then taking the mean over the gene set. Here it's recommended to work with unranked data.

$$S_{iG} = \frac{1}{|G|} \sum_{j \in G} (X_{ij} - \mu_i)/\sigma_i$$

Where μ_i and σ_i are the mean and standard deviation for cell i from X_i.

5. Rank Biased Overlap (20). Scores derived using rank biased overlap (RBO(14)) are generated by considering the overlap of a gene set (or other set of items) with the top k ranked genes in the list. The overlap is summed as k ranges from a lower limit (default 1) to a parameter value. However, as we descend in the ranking, a weighting of 1/k is multiplied to the overlap, so the value produced is reduced with each step down the ranks. It could be described as the weighted average of agreement between sorted ranks and gene set.

$$S_{iG} = \sum_{1..k} |H_k \cap G|$$

where H_k is the set of the first k genes after rank sorting.

7. ssGSEA (4,19): This method is born from GSEA, where empirical distributions of genes, both within and outside of the gene set are compared.

$$S_{iG} = \sum_{n=1..N} P_G(G, r_i, n) - P_{NG}(G, r_i, n)$$

where N is the number of genes, r_i is rank ordered genes, and m is the index to genes, and $r_i m$ is the rank of gene m.

$$P_G(G, X_i, n) = \frac{\sum_{m \in G, m<n} r_{im}^{\omega}}{\sum_{m \in G} r_{im}^{\omega}}$$

and

$$P_G(G, X_i, n) = \frac{\sum_{m \notin G, m<n} 1}{N - |G|}$$

8. SingScore (18). SingScore is a single sample gene set scoring software that was developed for use with bulk RNA-seq data. It represents an improved version of ssGSEA. Generally, it can be described as a normalized mean of median centered ranks.

$$S_{iG} = \frac{\sum r_{im}}{|G|}$$

The reported SingScore is a normalized SiG which is either done using a theoretical normalization or a simplification which divides the score by the total number of genes.

For the theoretical normalization,

$$\bar{S}_{iG} = \frac{S_{iG} - S_{min}}{S_{max} - S_{min}}$$

where

$$S_{min} = (|G| + 1)/2$$

and $S_{max} = (2|X_i| - |G| + 1)/2$

where $|X_i|$ is the total number of genes, and $|G|$ is the size of the gene set.

Application Details

Parallel processing of data

The granular structure of the data is a natural fit for using parallel processing; here we used python's multiprocessing.starmap() to asynchronously process each groupby-group with a list of gene sets.

The "groupby" parameter, which maps to a set of categorical variable names in the AnnData.obs table, breaks up the cells into chunks that can be processed in parallel. Since the smoothing function depends on having the nearest neighbor graph already calculated, the 'groupby' and 'recompute neighbors' parameters are related. By selecting a groupby parameter and a recompute neighbors parameter greater than the desired sample size, then the kNN graph is constructed within each group, which should be (nearly) guaranteed to have the expected number of neighbors for each cell. Without recomputing neighbors within each group, we have little control over the number of neighbors since neighboring cells are removed that are not part of the groupby.

Through experiments we have found the runtime to be approximately linear with the number of cells (Figure S2).

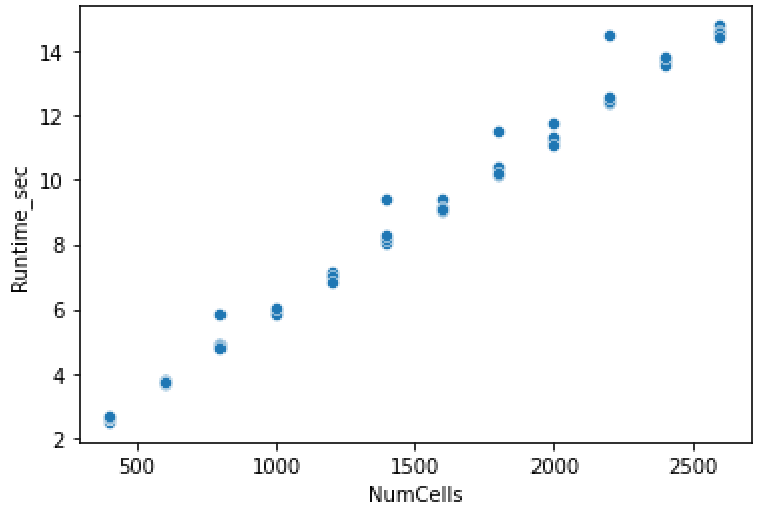

Figure S2. Plot shows the runtime in seconds versus the number of cells over five runs. The runtime is approximately linear in the number of cells and is generally consistent from run to run. At 2600 cells, in eight groups, the runtime is approximately 15s using a Google Colab notebook instance with 4 threads.

Gene sets formatting and naming

The implementation requires gene sets to be provided using the .gmt file format, which puts one gene set per line. Each line is tab delimited, starting with a name, description, and then list of genes. Following convention in single cell analysis, the genes should be encoded using HUGO symbols.

Gene sets have at least four ways of being scored according to whether one expects (is interested in) genes being highly expressed, highly reduced, extreme in general, or organized into two sets, one high one low. Genes in an extreme set would be found both high and low expressed. In practice, the three modes of expectation can be accessed by including an '_UP', '_DN', or '_BOTH' onto the suffix of the gene set name. This follows the MSigDB style of gene set naming. Often, one finds that where genes are expected to be both high and low expressed, they are split into two gene sets, and processed separately.

Gene sets that are paired (up + down) can appear in any order in the .gmt file, but the prefixes must be identical, the only difference is '_up' and '_dn'. By default, if a gene set name doesn't have a up/down suffix, it's taken as an upregulated gene set.

One way to score lowly expressed genes, is to rank them in the opposite order, and use the same function that scores high expressed genes.

The package can be found at the PyPi index and can be easily installed using pip. The repository contains both example data and gene sets to get started.

Please see: https://pypi.org/project/gssnng/ and https://github.com/Gibbsdavidl/gssnng