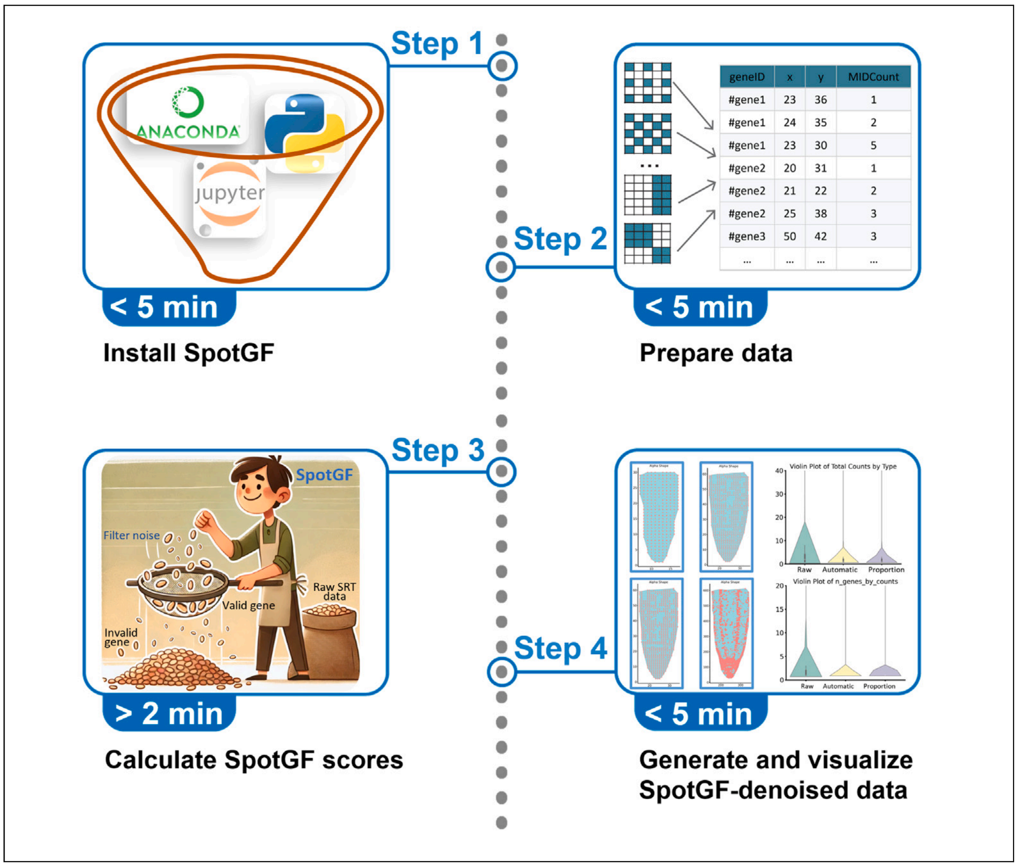
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol to denoise spatially resolved transcriptomics data utilizing optimal transport-based gene filtering algorithm



geneID	x	y	MIDCount
#gene1	23	36	1
#gene1	24	35	2
#gene1	23	30	5
#gene2	20	31	1
#gene2	21	22	2
#gene2	25	38	3
#gene3	50	42	3
...	...	...	...

Lin Du, Jingmin Kang, Jie Li, Hua Qin, Yong Hou, Hai-Xi Sun

sunhaixi@genomics.cn

Highlights

Steps to denoise SRT data without false positive signals and additional noise

Guidance on setting parameters for SpotGF to denoise a variety of SRT datasets

User-friendly and computationally efficient implementation of SpotGF

Spatially resolved transcriptomics (SRT) data contain intricate noise due to the diffusion of transcripts caused by tissue fixation, permeabilization, and cell lysis during the experiment. Here, we present a protocol for denoising SRT data using SpotGF, an optimal transport-based gene filtering algorithm, without modifying the raw gene expression. We describe steps for data preparation, SpotGF score calculation, filtering threshold determination, denoised data generation, and visualization. Our protocol enhances SRT quality and improves the performance of downstream analyses.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Du et al., STAR Protocols 6, 103625
March 21, 2025 © 2025 The Authors. Published by Elsevier Inc.
https://doi.org/10.1016/j.xpro.2025.103625

CellPress OPEN ACCESS

Protocol

Protocol to denoise spatially resolved transcriptomics data utilizing optimal transport-based gene filtering algorithm

Lin Du,[1,2,4,5] Jingmin Kang,[2,3,4] Jie Li,[2] Hua Qin,[2] Yong Hou,[1,3] and Hai-Xi Sun[1,2,3,6,*]

[1]College of Life Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

[2]BGI Research, Beijing 102601, China

[3]BGI Research, Shenzhen 518083, China

[4]These authors contributed equally

[5]Technical contact

[6]Lead contact

*Correspondence: sunhaixi@genomics.cn

https://doi.org/10.1016/j.xpro.2025.103625

SUMMARY

Spatially resolved transcriptomics (SRT) data contain intricate noise due to the diffusion of transcripts caused by tissue fixation, permeabilization, and cell lysis during the experiment. Here, we present a protocol for denoising SRT data using SpotGF, an optimal transport-based gene filtering algorithm, without modifying the raw gene expression. We describe steps for data preparation, SpotGF score calculation, filtering threshold determination, denoised data generation, and visualization. Our protocol enhances SRT quality and improves the performance of downstream analyses.

For complete details on the use and execution of this protocol, please refer to Du et al.[1]

BEFORE YOU BEGIN

Spatially resolved transcriptomics (SRT) technology combines gene expression profiles with the physical location of cells. However, factors such as cell damage, reagent exposure, and experimental procedures introduce complex spatial noise into the data, which interferes with downstream analysis. Previous denoising solutions, such as Sprod[2] and SpotClean,[3] rely on gene expression modification or imputation, which may introduce false positives by simulating transcription diffusion patterns, ultimately impacting data reliability.[4]

Therefore, SpotGF was proposed to classify genes based on their spatial aggregation characteristics in SRT data. Genes that are uniformly distributed within tissues are categorized as 'invalid' genes, as they do not contribute to downstream analyses such as cell clustering, cell annotation, and identification of differentially expressed genes (DEGs). In contrast, genes that exhibit spatial aggregation in tissues are considered 'valid' genes which are beneficial for downstream analyses. For housekeeping genes or control genes that are widely expressed across spatial locations, SpotGF identifies them as highly diffuse genes (invalid genes). Therefore, they will be filtered out to prevent interference with cell clustering and differential gene expression analysis. While SpotGF focuses on genes with spatial aggregation, if users are interested in these spatially diffuse genes, we support adding them back after completing the clustering step. In summary, SpotGF algorithm mitigates the adverse effects of noise in SRT data by filtering out invalid genes.[1]

This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

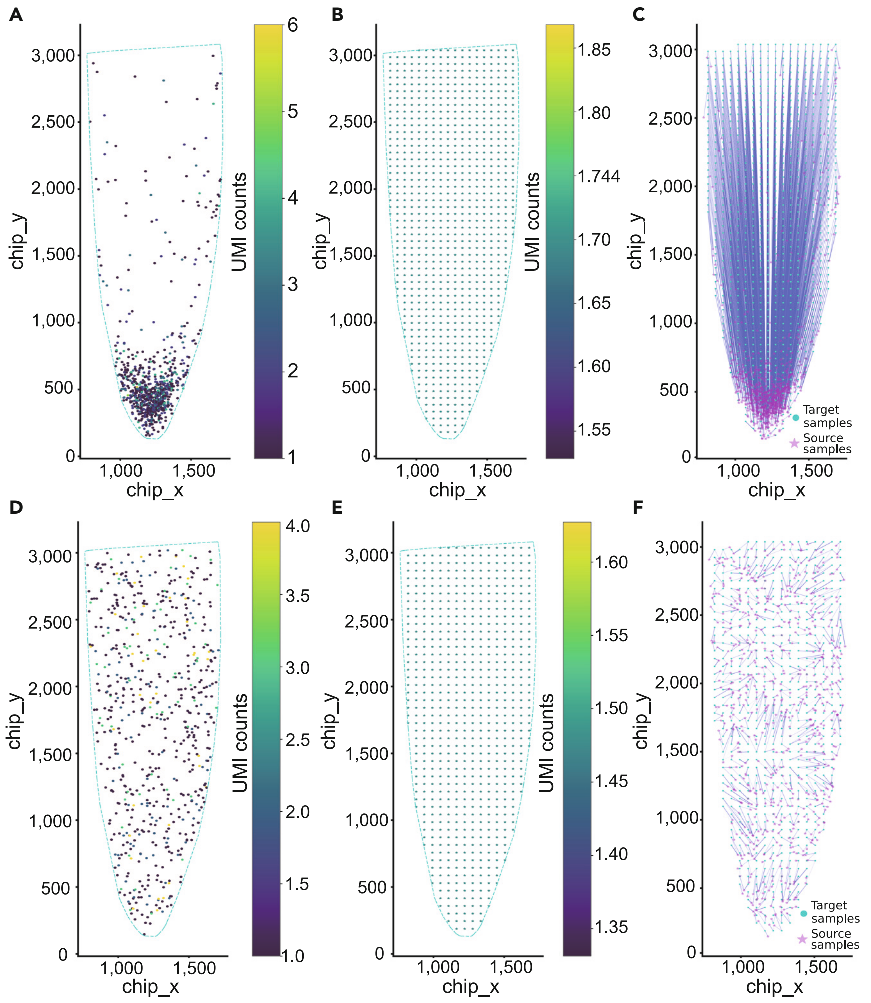


Figure 1. The SpotGF score differentiates valid and invalid genes in Stereo-seq soybean root data
(A) Source distribution of valid gene *SoyZH13_09G038300*.
(B) Target distribution of valid gene *SoyZH13_09G038300*.
(C) OT scheme of valid gene *SoyZH13_09G038300*.
(D) Source distribution of invalid gene *SoyZH13_15G110700*.
(E) Target distribution of invalid gene *SoyZH13_15G110700*.
(F) OT scheme of invalid gene *SoyZH13_15G110700*.
Each spot indicates the location of gene expression, with the color bar representing expression levels in panels (A, B, D, and E). Expression levels in panels (B and D) are consistent across spots. The total expression in panel (A) match that in panel (B), and similarly, the total expression in panel (D) matches that in panel (E).

For each gene, SpotGF constructs both a source distribution and a target distribution. The source distribution is based on the gene's expression at each location, while the target distribution represents the gene's hypothetical expression state under conditions of maximum diffusion within the tissue. Using the optimal transport (OT) algorithm[5] and calculating the transportation cost between the source and target distributions, we generate an OT scheme that serves as the SpotGF score, measuring the degree of gene diffusion. Consequently, a lower SpotGF score indicates greater diffusion, whereas a higher SpotGF score indicates stronger spatial aggregation.

Using Stereo-seq soybean root data (Figure 1) as an example, SpotGF constructs the source and target distributions for the valid gene *SoyZH13_09G038300* (Figures 1A–1C) and the invalid gene

SoyZH13_15G110700 (Figures 1D–1F), respectively. Subsequently, the transport cost of these distributions was calculated to obtain SpotGF scores.

Furthermore, SpotGF provides an automated algorithm based on gradient changes to determine the filtering threshold of the SpotGF score, with the option for users to manually adjust the threshold as needed. The accuracy of the SpotGF score in representing the diffusion level has been verified through our previous analysis.[1]

In the following sections, we provide a step-by-step guide to help users apply SpotGF to their SRT dataset from start to finish, enabling them to obtain high-quality denoised data.

Set up the environment

Timing: <5 min

This section describes steps to set up two types of environments for running SpotGF.

Note: SpotGF is compatible with both Windows and Unix-based operating systems (including macOS and Linux) and requires Python 3 or higher. During the development of SpotGF, we used Python 3.9.19, so we recommend using Python 3.9 or a higher version to avoid potential issues from Python updates. The necessary steps to set up a dedicated Python environment and perform the analysis are outlined below. For optimal performance and to avoid conflicts with other scripts or libraries, we recommend running this protocol using a dedicated Anaconda environment. This setup helps prevent compatibility issues with other installed software or different Python versions.

1. Download Anaconda from the official website (https://www.anaconda.com) and follow the installation instructions provided for your system.

Note: This method is particularly recommended for users who prefer to use Anaconda for package management and environment control. It helps in creating a dedicated environment that is isolated from other projects, reducing the risk of dependency conflicts.

Note: In addition, Mamba is a high-performance package manager that is compatible with Conda but offers faster installation speeds. Here are the installation and usage instructions for Mamba:

   a. Install Mamba. You can install Mamba in two ways:
      i. Method 1: Install Mamba through Conda, which is the simplest method. Run the following command in your Conda environment to install Mamba from the conda-forge channel:

```
>conda install mamba -c conda-forge
```

      ii. Method 2: Download and install from GitHub using the following commands:

```
wget  https://github.com/conda-forge/miniforge/releases/latest/download/Mambaforge-$(uname)-
$(uname-$(uname)-$(uname -m).sh

bash Mambaforge-$(uname)-$(uname -m).sh
```

Note: Due to network issues, the parsing of the above web page did not succeed. If users need the content of this web page, they should be advised to check the legitimacy of the web page link and try again. The issue might be related to the link itself or network-related, and users should be guided to verify the link and retry appropriately. If the parsing of this link is not necessary for answering the user's question, then proceed to answer the user's question as normal.

b. Use Mamba After installation, you can use Mamba just like Conda, simply replace 'conda' with 'mamba' in the commands. For example, to activate an environment, you can run the following command:

```
>mamba activate SpotGF
```

2. After installing Anaconda, restart any open terminals. Then, create a dedicated environment that includes Python (version 3.9) and dependency packages necessary to run the scripts by executing the command provided below:

```
>conda create --name SpotGF python=3.9
```

3. Activate the environment using the following command:

```
>conda activate SpotGF
```

4. Install the Python package of SpotGF.
   a. One way to install SpotGF is:
      i. Download the SpotGF-main.zip file from the SpotGF GitHub repository (https://github.com/illuminate6060/SpotGF). If you encounter any issues with the download, verify the link and your network connection.
      ii. Extract the contents of the zip file to a directory on your computer, for example, path/to/SpotGF-main.

```
>cd [path/to/SpotGF-main]
```

      iii. Navigate to the extracted directory in your terminal or command prompt.
      iv. Use pip to install the SpotGF package by running the command 'pip install.' within the directory.

```
>pip install .
```

   b. Alternatively, you can install SpotGF as follows:
      i. Open your command line interface (Command Prompt on Windows, Terminal on Mac or Linux). Clone the SpotGF repository from GitHub using the following command:

```
>git clone https://github.com/illuminate6060/SpotGF.git
```

      ii. Navigate to the cloned SpotGF directory.

```
>cd ./SpotGF
```

      iii. Install SpotGF using pip with the following command. This will install the SpotGF package in your current directory.

```
>pip install .
```

Note: Both methods will ensure that SpotGF is correctly installed and ready for denoising SRT data. If any error occurs, please refer to the troubleshooting section for this protocol.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
Anaconda	Anaconda, Inc.	https://www.anaconda.com/products/individual
Python 3.9.19	Python	https://www.python.org/
SpotGF	Lin, D. et al.[1]	https://github.com/illuminate6060/SpotGF
SpotGF_data_form_change	This paper	https://github.com/illuminate6060/SpotGF_data_form_change
POT	Rémi, F. et al.[6]	https://github.com/PythonOT/POT
Alpha-shape algorithm	Gardiner, J.D. et al.[7]	https://alphashape.readthedocs.io/en/latest/
shapely	Scott M. et al.[8]	https://github.com/shapely/shapely/tree/main/
seaborn	Michael L. et al.[9]	https://github.com/mwaskom/seaborn
scanpy	Wolf, F.A. et al.[10]	https://scanpy.readthedocs.io/en/stable/
Other		
Windows 11	Microsoft	https://www.microsoft.com/en-us/windows/windows-11?r=1
GNU/Linux	Free Software Foundation	https://www.gnu.org/

STEP-BY-STEP METHOD DETAILS

Input data preparation

Timing: <5 min

This section describes steps to prepare the formatted input file required for running SpotGF.

Note: The SpotGF workflow requires only one mandatory file containing the expression and position data for each gene in the user's SRT dataset. SpotGF supports various input file formats, such as gem, txt, csv, and gem.gz. However, at least, the input file is required to contain specific columns including 'geneID', 'x', 'y', and 'MIDCount'.

1. Import the SpotGF package and set the current working directory using the following command:

```
>from SpotGF import SpotGF
>import os
>work_dir = '/home/user/SpotGF'
>os.chdir(work_dir)
```

Note: To assist users in preparing their input data for SpotGF, we also have compiled a set of data format conversion scripts available at https://github.com/illuminate6060/SpotGF_data_form_change. These scripts facilitate users to prepare input data when using SpotGF, including converting gef files to gem files, 10× Visium data to gem files, gem files to h5ad files, and h5ad files to rds files.

SpotGF denoising process

Timing: >2 min (depending on the bin size resolution and SRT data size)

This section describes the steps to execute SpotGF in various environments. It also explains the significance of each parameter and offers guidance on adjusting them to enhance the performance of SpotGF.

2. Run SpotGF using Python script.

Note: Run the complete SpotGF workflow by executing the SpotGF.py script download from GitHub and using input data ('input_file.gem') in a supported format. The sample command is provided below:

```
>python SpotGF.py -i input_file.gem -o ./output -b 50 -p 0.6 -s 5
```

⚠ CRITICAL: The explanation for positional arguments is as follows:

a. -i: Specify the path to the SRT data prepared according to the 'input data preparation' step. This should point to your input file, for example, 'path/to/input.gem'.
b. -o: Specify the directory where the SpotGF-denoised data will be saved, such as 'path/to/outpath'. SpotGF will check if this output path exists; if it doesn't, SpotGF will create the directory.
c. -b: Set the bin size for denoising resolution, which must be an integer. The default value is 10. This parameter adjusts the granularity of the denoising process. A smaller bin size provides finer detail but increases processing time. A larger bin size provides coarser details but reduces processing time.
d. -lower: Set the lower limit for optimizing tissue structure capture, with a default value of 0.
e. -upper: Define the upper limit for optimizing tissue structure capture. By default, this is set to the maximum value represented by a floating-point number in Python.
f. -max_iterations: Set the maximum number of iterations for capturing tissue structures, with a default of 10,000. As the max_iterations value increases, the time to run the 'SpotGF.py' script increases.
g. -auto_threshold: When set to True, SpotGF will automatically apply a filter threshold to generate a denoised gem file. The default is True.
h. -p: Defines a gene proportion-based filtering threshold. It represents the percentage of genes retained in the SpotGF-denoised data relative to the raw data's gene. For instance, setting it to 0.9 will retain only the top 90% of valid genes with the highest SpotGF scores, thereby creating a new denoised SRT dataset. The default value is 0.5, and it accepts any floating-point number between 0 and 1.
i. -v: When set to True, the output will include regular visualizations of the denoised data. The default is True.
j. -s: Specify the spot size for plotting the spatial expression of the denoised data in 'Spatial_automatic.png' and 'Spatial_proportion.png'. The default value is 10.
k. -a: Set the alpha value for tissue boundary identification. The default setting is 0, which means SpotGF will use an automatically optimized alpha value.

Note: To speed up the running time of SpotGF, users can adjust the following parameters: The '-b' parameter, which controls the bin size, can significantly reduce processing time with a higher value, and it is highly recommended to modify this first. The '-max_iterations' parameter, setting a lower value for it will also reduce the processing time. The '-v' parameter, if standard visualization results are not required, can be set to False to further decrease the total running time. The '-a' parameter, if users can provide a custom alpha value instead of using the default value of 0, it will decrease the time SpotGF spends optimizing the alpha value in capturing tissue boundaries.

3. Run SpotGF using a Jupyter environment.
   a. Import the SpotGF class from the SpotGF module.
   b. Define the path to your input gem file and specify the path to save your results.
   c. Set the other parameters for SpotGF.
   d. Initialize the SpotGF class with the specified parameters.

e. Calculate SpotGF scores.
f. Generate and visualize the denoised data based on both the automatic threshold and the user-defined proportion.

Note: A specific usage example code is as follows. This script initializes the SpotGF class with your specified parameters, calculates the SpotGF scores to help you determine an appropriate proportion for gene retention, and finally generates a new denoised gem file based on these scores and parameters.

```
>import SpotGF

>input = './SpotGF/test/demo.gem'

>output = './SpotGF/test/result'

>binsize = 70

>lower = 0

>upper = 100000

>proportion = 0.1

>max_iterations = 10000

>auto_threshold = True

>visualize = True

>spot_size = 20

>alpha = 0

>spotgf = SpotGF.SpotGF(gem_path, binsize, proportion, auto_threshold, lower,upper, max_-
iterations, output, visualize, spot_size, alpha)

>GF_df = spotgf.calculate_GFscore(input, binsize, alpha)

>new_gem = spotgf.generate_GFgem(gem_path, GF_df, proportion, auto_threshold, visualize,
spot_size)
```

Note: Users are required to provide the -i (input path) and -o (output path) parameters. All other parameters come with default values and do not need to be specified unless you wish to customize them.

Note: Should the input file include columns labeled 'cen_x' and 'cen_y', the denoising operation can be executed on the cell bin data, enabling single-cell-level denoising by specifying 'binsize = 1' or '- b 1'. It is important to clarify that the bin size is exclusively utilized for the computation of SpotGF scores and has no bearing on the resulting denoised dataset.

4. Explanation of automatic filtering thresholds.

Note: SpotGF has embedded a novel method that leverages automatic gradient variation to determine an optimal threshold for denoising gene expression data. Utilizing the orchid dataset[11] (Figure 2A) as a case study, we detailed our approach, which begins by ranking all 14,730 genes by their SpotGF scores. This method subsequently establishes a recommended threshold of 30.059, resulting in the exclusion of 12,754 genes and the retention of 1,976. Below is a streamlined overview of the steps involved:

a. Sorting Genes: Arrange all genes in ascending order based on their SpotGF scores (Figure 2B).

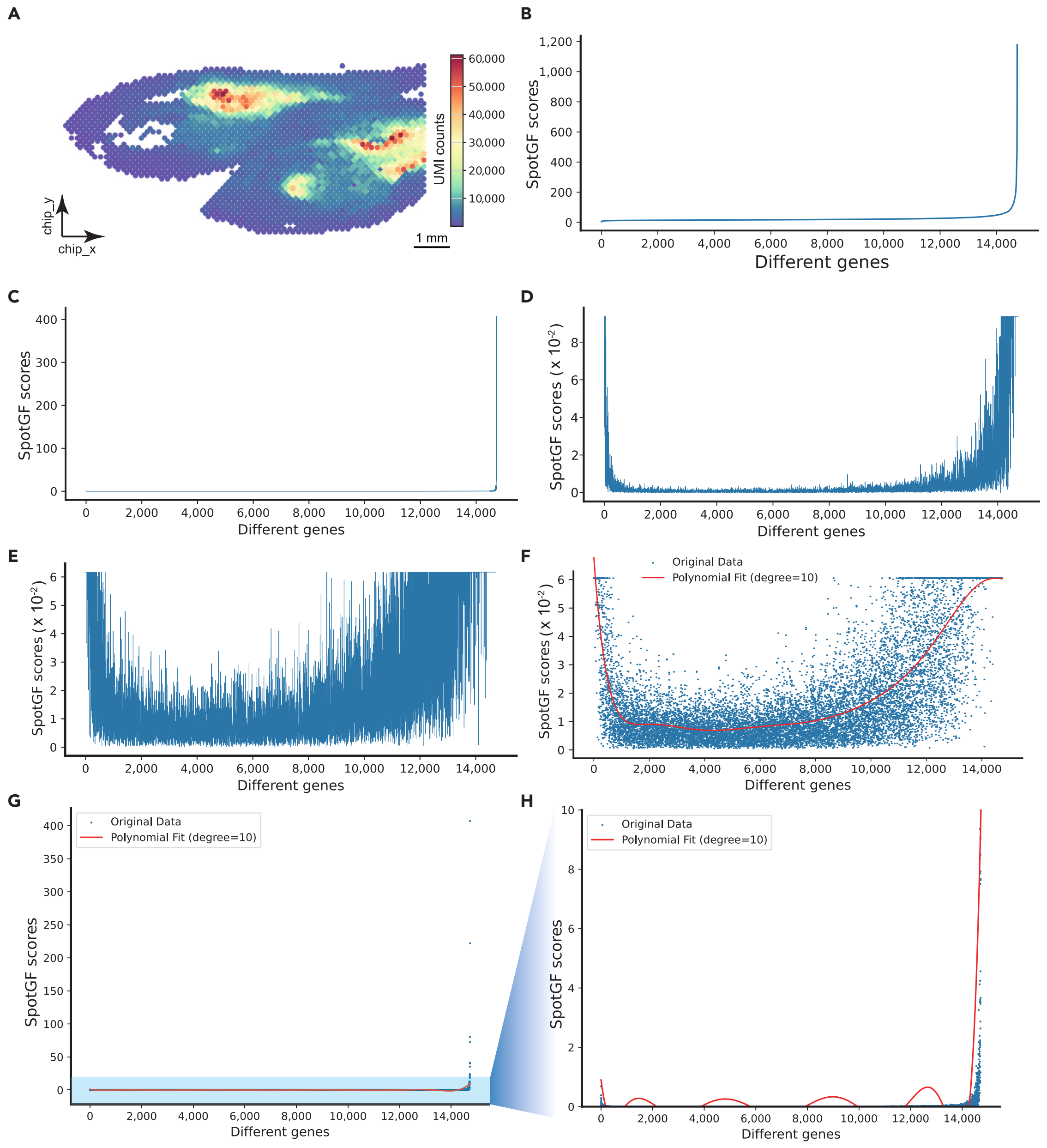


Figure 2. Automatic threshold calculation by SpotGF for 10× Visium orchid data

(A) Heatmap of gene expression in 10× Visium orchid data, with the color bar indicating expression levels.
(B) Line plot of all genes sorted in ascending order of SpotGF scores.
(C) Line graph of the first-order derivative of the SpotGF scores.
(D) Line plot of the first-order derivative of SpotGF scores after one round of mean replacement.
(E) Line plot of the first-order derivative of SpotGF scores after two rounds of mean replacement.
(F) Distribution of SpotGF values from panel (E), simulated using a 10th-order polynomial function. Blue dots represent genes, while the red line represents the 10th-order polynomial curve.
(G) Distribution of the original SpotGF scores, simulated by a 10th-order polynomial function.
(H) Simulated distribution of SpotGF scores from treated data in panel (G) using a 10th-order polynomial function.

b. Derivative Calculation: Compute the first derivative for each gene's SpotGF score (Figure 2C).
c. Mean Replacement: Replace any first derivative values exceeding the mean of the array with the mean itself (Figure 2D).

d. Second Round of Replacement: Apply the same mean replacement to the results from the previous step c (Figure 2E).
e. Curve Fitting: Fit a 10th-order polynomial to model the relationship between SpotGF scores and the modified derivative array (Figures 2F–2H).
f. Determination: Identify the filtering threshold as the SpotGF score at the point of maximum gradient change among values exceeding the median SpotGF score. This ensures an effective threshold for filtering out non-informative genes (Figure 2F).

Note: The intermediate results of each step in the automatic filtering threshold calculation are in Figure 2. However, in practical applications, the SpotGF software only computes and returns the final automatic threshold value without displaying these intermediate results.

EXPECTED OUTCOMES

```
/home/user/path/output/result

--Data

|-- SpotGF_auto_threshold.gem

|-- SpotGF_proportion_0.5.gem

|-- SpotGF_scores.txt

--Figure

|-- alpha_shape.png

|-- Spatial_automatic.png

|-- Spatial_proportion.png

|-- Violinplot_n_genes_by_counts.png

|-- Violinplot_total_counts.png
```

Once the analysis is completed, the output will be stored in the specified output folder, following the structure mentioned above.

In the output folder, 'SpotGF_auto_threshold.gem' indicates that when the user sets the 'auto_threshold' parameter to TRUE, SpotGF will return denoised SRT data based on a recommended threshold. A file name containing 'proportion' in the gem file, such as 'SpotGF_proportion_0.1.gem', indicates that when the user sets the proportion parameter, SpotGF will return denoised SRT data that has been filtered according to the user-defined proportion threshold. The 'SpotGF_scores.txt' file records the diffusion extent of each gene, where a higher score indicates a valid gene with spatial aggregation, and a lower score indicates an invalid gene with a spatial diffusion pattern.

Additionally, we have conducted some regular visualizations on the denoised SRT data. The output 'automatic_threshold_spatial.png' represents the spatial expression distribution of the denoised data after automatic threshold filtering. The 'proportion_spatial.png' shows the spatial expression distribution of the denoised data after filtering with a user-defined proportion. The 'violinplot_total_counts.png' contains violin plots of the total counts for each cell in the raw data, data after denoising based on the automatic threshold, and data after denoising based on the proportion. Similarly, 'violinplot_n_genes_by_counts.png' provides violin plots of gene counts per cell across different datasets. The 'alpha_shape.png' illustrates the tissue contour used by SpotGF during the denoising process.

In the process of SpotGF generating the target distribution for the most severe diffusion mode of each gene, the identification of tissue contours is involved. To this end, we provide the 'alphashape.optimizealpha' algorithm, which does not ignore any spots to calculate the most suitable alpha parameter, thereby determining the tissue contour as a polygon.

It is important to note that SpotGF only requires a contour result that roughly reflects the overall tissue and does not pursue extremely high precision. While more precise contour recognition may slightly improve SpotGF score accuracy, it can also increase computation time. If the contour identified in the 'alpha_shape.png' significantly deviates from the actual tissue contour, or if users have higher precision requirements for denoising, we offer the '-a' parameter to customize the alpha value for contour detection. Increasing the alpha parameter value will adjust the boundary to more closely fit the sample data, resulting in a tighter boundary box.

QUANTIFICATION AND STATISTICAL ANALYSIS

Our previous work validated the robustness of SpotGF using simulated data,[1] demonstrating a strong linear correlation between the diffusion characteristics of the spatial distribution and the SpotGF score when employing the Gaussian distribution to simulate varying degrees of diffusion. Additionally, SpotGF scores can effectively and reliably evaluate genes expressed at different spatial point numbers.

In terms of computational speed and memory requirements, the 'calculate_GFscore' step in SpotGF is the most time-consuming. This duration depends on the size of the user's dataset and the 'binsize' resolution parameter. To help users estimate SpotGF's runtime and memory requirements, we conducted quantitative tests on the Stereo-seq soybean root data (Figures 3 and 4). As the 'binsize' parameter increases, both computation time and memory usage also increase. We recommend selecting a larger 'binsize' for datasets with a large number of genes.

LIMITATIONS

Stereo-seq soybean root data demonstrated the effectiveness of SpotGF in identifying characteristic genes in different tissue regions. Genes with higher SpotGF scores tend to cluster spatially, making them strong candidates for spatially variable genes (SVGs). However, the automatic method may occasionally remove too many genes, so it is recommended that users use all valid genes for further analysis instead of using highly variable genes (HVGs). Alternatively, users can solve this problem by setting a custom threshold to adjust the number of retained genes according to the specific requirements of their analysis.

TROUBLESHOOTING

Problem 1

Difficulty cloning the GitHub repository (Related to before you begin step 1).

Potential solution

If you experience issues cloning the repository, it may be due to network connectivity problems or restrictions on GitHub access. Ensure that your internet connection is stable and that the repository link is correct. If the problem persists, try again later, or consider using a VPN to bypass any regional access restrictions.

Problem 2

Converting 10× Visium data into the input format supported by SpotGF (Related to input data preparation step).

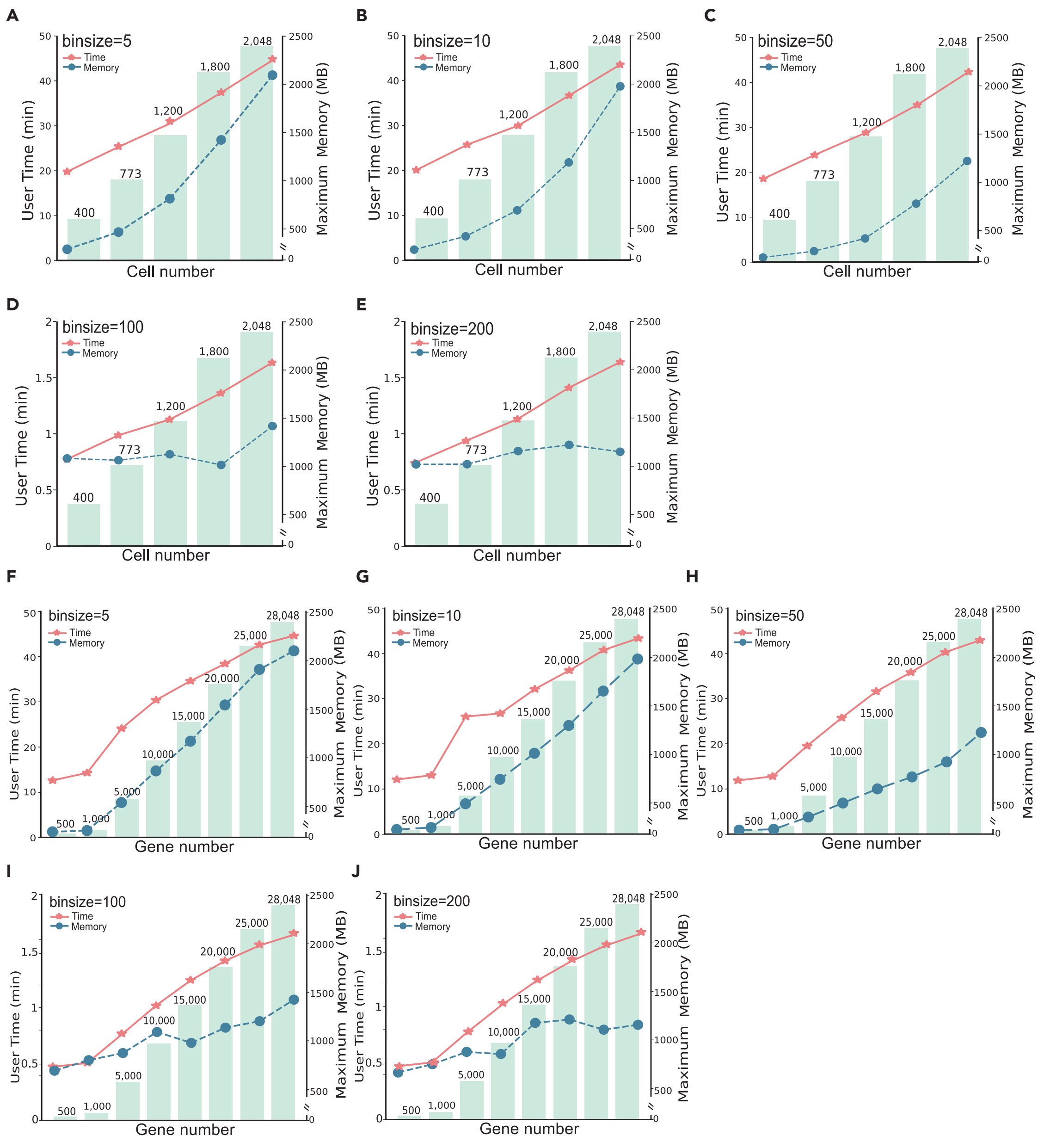


Figure 3. Time and maximum memory usage of SpotGF on soybean root Stereo-seq data

(A–E) Memory and time requirements of SpotGF at various cell counts (400, 773, 1,200, 1,800, and 2,048), with a constant gene count of 28,048, using different resolution bin sizes (5, 10, 50, 100, and 200).

(F–J) Memory and time requirements of SpotGF with varying gene counts (500, 1,000, 5,000, 10,000, 15,000, 20,000, 25,000, and 28,048), with a constant cell count of 2,048, using different resolution bin sizes (5, 10, 50, 100, and 200). The red asterisks indicate the peak memory required for SpotGF at each parameter setting. The blue dots represent the time taken to run SpotGF under each parameter setting.

Potential solution

We have provided data format conversion scripts to simplify the preparation of SpotGF input data. These scripts are available in our GitHub repository: https://github.com/illuminate6060/SpotGF_data_form_change.

Problem 3

Missing or incompatible libraries, resulting in 'Error: ModuleNotFoundError or ImportError' (Related to SpotGF denoising process step).

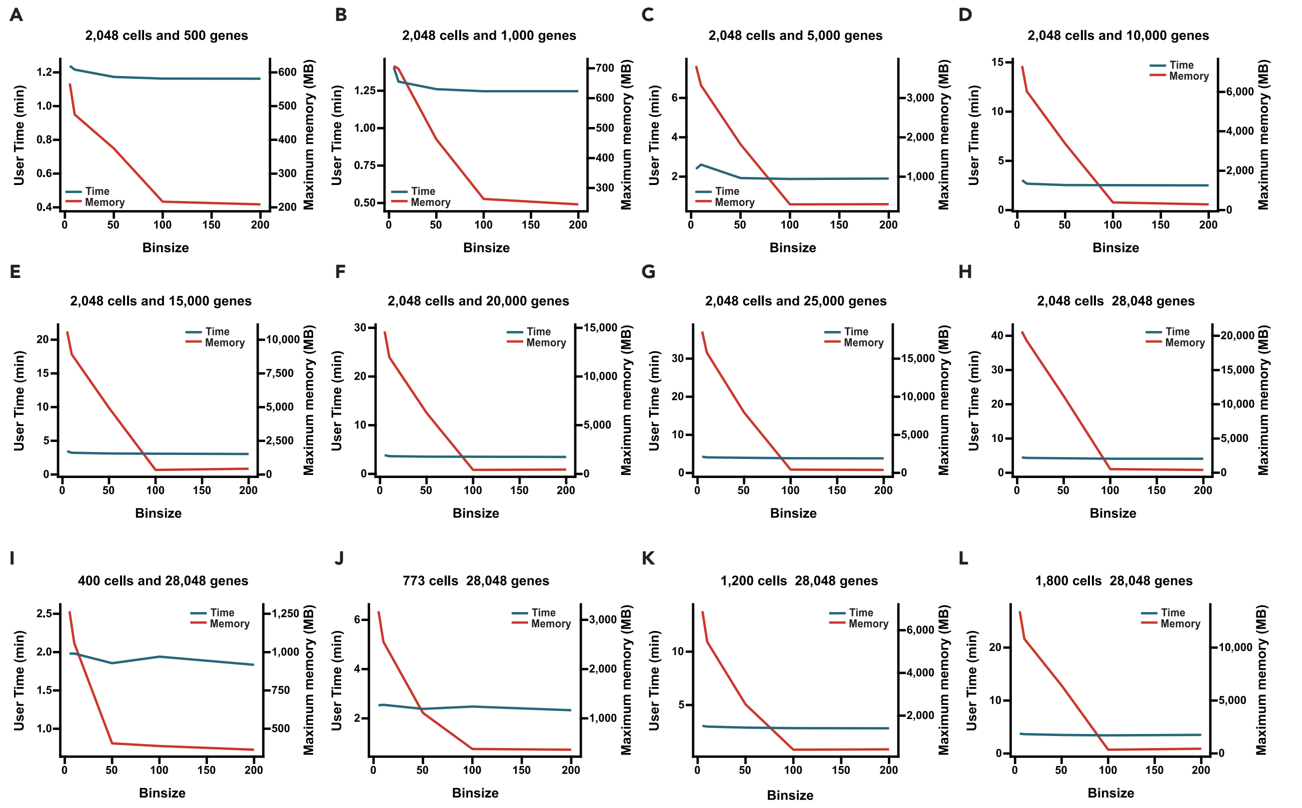


Figure 4. The time and maximum memory of fixed cell numbers and gene numbers under different bin sizes

(A–H) Memory and time requirements for SpotGF across varying gene counts (500, 1,000, 5,000, 10,000, 15,000, 20,000, 25,000, and 28,048), with a fixed cell count of 2,048, under different resolution bin sizes (5, 10, 50, 100, and 200).

(I and L) Memory and time requirements for SpotGF across varying cell counts (400, 773, 1,200, and 1,800), with a fixed gene count of 28,048, under different resolution bin sizes (5, 10, 50, 100, and 200). The red lines represent the processing time required to run SpotGF for each parameter setting, while the blue lines indicate the maximum memory usage under each setting.

Potential solution

Ensure that all required libraries are installed and compatible with the code. You can install the required libraries using pip install -r requirements.txt (https://github.com/illuminate6060/SpotGF/blob/main/SpotGF.egg-info/requires.txt). For example, if the code can't find the pot library, you would need to run pip install POT.

Problem 4

Encountering the 'RESULT MIGHT BE INACCURATE' warning when running SpotGF (Related to SpotGF denoising process step).

RESULT MIGHT BE INACURATE: Max number of iterations reached, currently 100,000. Sometimes iterations go on in the cycle even though the solution has been reached, to check if it's the case here have a look at the minimal reduced cost. If it is very close to machine precision, you might have the correct solution, if not try setting the maximum number of iterations a bit higher. UserWarning: numItermax reached before optimality. Try to increase numItermax.check_result(result_code).

Potential solution

This warning occurs because the 'alphashape.optimizealpha' function did not converge after reaching the maximum number of iterations when capturing the organizational structure. The user can check the result of 'alpha_shape.png' to determine whether the current contour detection is reasonable. If this is not reasonable, you can solve this problem by increasing the value of the 'max_-iterations' parameter.

> *Note:* If the solver fails to find a solution, the value 0 will be returned, and when used with the alphashape function, the convex hull around the point will be safely returned (SpotGF denoising process step).

For example, if the original command you encountered this problem was:

```
>python SpotGF.py -i input_file.gem -o ./output -b 50 -p 0.6 -s 5
```

You can change it to the following new command:

```
>python SpotGF.py -i input_file.gem -o ./output -b 50 -p 0.6 -s 5 -max_iterations 10000000
```

Problem 5

TypeError: 'MultiPoint' object is not iterable (Related to SpotGF denoising process step).

Potential solution

Ensure that all third-party libraries in your project are compatible with Shapely 2.0 by checking and updating them as needed. If it doesn't work, you can downgrade Shapely to 1.1.8. I use the following environment which is not reported to have errors, you can try it.

- python==3.9.19
- numpy==1.23.5
- pot==0.8.2
- pandas==1.4.0
- matplotlib==3.5.0
- scipy==1.10.0
- descartes==1.1.0
- alphashape==1.3.1
- shapely==1.8.5.post1
- scanpy==1.9.2
- seaborn==0.13.2

Problem 6

KeyError: '['Unnamed: 0.1' 'Unnamed: 0'] not found in axis' (Related to SpotGF denoising process step).

Potential solution

Try reinstalling SpotGF from https://github.com/illuminate6060/SpotGF. This problem has been resolved in the latest version.

Problem 7

IndexError: 'index 2 is out of bounds for axis 0 with size 2' (Related to SpotGF denoising process step).

Potential solution

Try reinstalling SpotGF from https://github.com/illuminate6060/SpotGF. This problem has been resolved in the latest version.

Problem 8

Incorrect Parameter Types with 'Error: TypeError' (Related to SpotGF denoising process step).

Potential solution

Ensure that argument types passed to the SpotGF class are correct: Integers for 'binsize', 'max_iterations', and 'spot_size'; The 'lower', 'upper', 'alpha', and 'proportion' parameters are floating-point numbers; The 'visualize' and 'auto_threshold' are boolean numbers.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Hai-Xi Sun (sunhaixi@genomics.cn).

Technical contact

Questions about the technical specifics of performing the protocol should be directed to and will be answered by the technical contact, Lin Du (dulin@genomics.cn).

Materials availability

This study did not generate unique reagents.

Data and code availability

- Data: All data reported in this study are publicly available and presented in the article.
- Code: All original code is available through https://github.com/illuminate6060/SpotGF (https://doi.org/10.5281/zenodo.14613455) and https://github.com/illuminate6060/SpotGF_data_form_change (https://doi.org/10.5281/zenodo.14613451). We have also developed an open-source interactive web of SpotGF toolkit in DCS Cloud at https://cloud.stomics.tech/library/#/tool/detail/workflow/WF01202412149bpxmb/1.0.0?zone=sz.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (2023YFC3402200). The author would like to thank the reviewers for their reading and constructive comments.

AUTHOR CONTRIBUTIONS

L.D. developed the SpotGF and contributed to algorithm verification, bioinformatics analysis, original manuscript writing, and figure generation. J.K. contributed to algorithm verification and figure generation. J.L. and H.Q. contributed to SpotGF verification. H.-X.S. and Y.H. critically reviewed the manuscript and provided overall supervision for the study.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Du, L., Kang, J., Hou, Y., Sun, H.X., and Zhang, B. (2024). SpotGF: Denoising spatially resolved transcriptomics data using an optimal transport-based gene filtering algorithm. Cell Syst. *15*, 969–981.e6. https://doi.org/10.1016/j.cels.2024.09.005.
2. Wang, Y., Song, B., Wang, S., Chen, M., Xie, Y., Xiao, G., Wang, L., and Wang, T. (2022). Sprod for de-noising spatially resolved transcriptomics data based on position and image information. Nat. Methods *19*, 950–958. https://doi.org/10.1038/s41592-022-01560-w.
3. Ni, Z., Prasad, A., Chen, S., Halberg, R.B., Arkin, L.M., Drolet, B.A., Newton, M.A., and Kendziorski, C. (2022). SpotClean adjusts for spot swapping in spatial transcriptomics data. Nat. Commun. *13*, 2971. https://doi.org/10.1038/s41467-022-30587-y.
4. Andrews, T.S., and Hemberg, M. (2018). False signals induced by single-cell imputation. F1000Res. *7*, 1740. https://doi.org/10.12688/f1000research.16613.2.
5. Peyré, G., and Cuturi, M. (2019). Computational Optimal Transport: With Applications to Data Science. FNT. Mach. Learn. *11*, 355–607. https://doi.org/10.1561/2200000073.
6. Rémi Flamary, N.C., Gramfort, A., Alaya, M.Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., et al. (2021). POT: Python Optimal Transport. J. Mach. Learn. Res. *22*, 1–8.
7. Gardiner, J.D., Behnsen, J., and Brassey, C.A. (2018). Alpha shapes: determining 3D shape complexity across morphologically diverse structures. BMC Evol. Biol. *18*, 184. https://doi.org/10.1186/s12862-018-1305-z.
8. Lundberg, S. (2017). A unified approach to interpreting model predictions. Preprint at arXiv. https://doi.org/10.48550/arXiv.1705.07874.
9. Waskom, M. (2021). Seaborn: statistical data visualization. J. Open Source Softw. *6*, 3021.
10. Wolf, F.A., Angerer, P., and Theis, F.J. (2018). SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. *19*, 15. https://doi.org/10.1186/s13059-017-1382-0.
11. Liu, C., Leng, J., Li, Y., Ge, T., Li, J., Chen, Y., Guo, C., and Qi, J. (2022). A spatiotemporal atlas of organogenesis in the development of orchid flowers. Nucleic Acids Res. *50*, 9724–9737. https://doi.org/10.1093/nar/gkac773.