Analyzing Spatial Transcriptomics Data Using Giotto

Natalie Del Rossi,[1,5] Jiaji G. Chen,[2,5] Guo-Cheng Yuan,[1,3,6] and Ruben Dries[2,4,6]

[1]Department of Genetics and Genomic Sciences, Charles Bronfman Institute for Personalized Medicine, Icahn School of Medicine at Mount Sinai, New York
[2]Section of Hematology and Medical Oncology, School of Medicine, Boston University, Boston, Massachusetts
[3]Precision Immunology Institute, Icahn School of Medicine at Mount Sinai, New York, New York
[4]Division of Computational Biomedicine, School of Medicine, Boston University, Boston, Massachusetts
[5]These authors contributed equally to this work
[6]Corresponding author: *guo-cheng.yuan@mssm.edu*; *rdries@bu.edu*

Spatial transcriptomic technologies have been developed rapidly in recent years. The addition of spatial context to expression data holds the potential to revolutionize many fields in biology. However, the lack of computational tools remains a bottleneck that is preventing the broader utilization of these technologies. Recently, we have developed Giotto as a comprehensive, generally applicable, and user-friendly toolbox for spatial transcriptomic data analysis and visualization. Giotto implements a rich set of algorithms to enable robust spatial data analysis. To help users get familiar with the Giotto environment and apply it effectively in analyzing new datasets, we will describe the detailed protocols for applying Giotto without any advanced programming skills. © 2022 Wiley Periodicals LLC.

Basic Protocol 1: Getting Giotto set up for use
Basic Protocol 2: Pre-processing
Basic Protocol 3: Clustering and cell-type identification
Basic Protocol 4: Cell-type enrichment and deconvolution analyses
Basic Protocol 5: Spatial structure analysis tools
Basic Protocol 6: Spatial domain detection by using a hidden Markov random field model
Support Protocol 1: Spatial proximity–associated cell-cell interactions
Support Protocol 2: Assembly of a registered 3D Giotto object from 2D slices

Keywords: cell-cell interaction • cell type • deconvolution • single cell • software • spatial transcriptomics

How to cite this article:
Del Rossi, N., Chen, J. G., Yuan, G.-C., & Dries, R. (2022). Analyzing spatial transcriptomics data using Giotto. *Current Protocols*, *2*, e405. doi: 10.1002/cpz1.405

INTRODUCTION

Multicellular organisms consist of diverse cell types that act in concert to maintain the structure and function of tissues and organs. The anatomic structure of a tissue or organ is highly organized and often conserved across species, suggesting that the spatial distribution of cell types may be essential for the maintenance of tissue functions. Disruption

CURRENT PROTOCOLS A Wiley Brand

Current Protocols e405, Volume 2
Published in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpz1.405

of the tissue microenvironment has been observed in numerous human diseases, and its role in etiology is beginning to be recognized (Bettcher, Tansey, Dorothée, & Heneka, 2021; Binnewies et al., 2018; Buckley, Ospelt, Gay, & Midwood, 2021; Hanahan & Weinberg, 2011). Recently, new technologies have rapidly emerged to dissect cellular composition while preserving spatial information. Collectively, these technologies are referred to as spatial transcriptomics, and have been highlighted as the Method of the Year in 2020 by *Nature Methods* (Marx, 2021). Spatial transcriptomics has provided an unprecedented opportunity to dissect tissue microenvironment, elucidate cell-cell interaction mechanisms, and characterize heterogeneity among disease patients (Lewis et al., 2021; Longo, Guo, Ji, & Khavari, 2021; Rao, Barkley, França, & Yanai, 2021). They have also been utilized by a number of consortia to create spatially resolved cell atlases in health and disease (Regev et al., 2017; HuBMAP Consortium, 2019; BRAIN Initiative Cell Census Network (BICCN), 2021; Rozenblatt-Rosen et al., 2020).

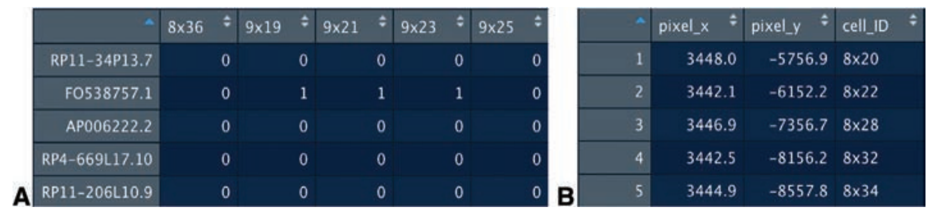
Spatial transcriptomic analysis presents new challenges that require the development of novel computational tools (Dries et al., 2021a). On the one hand, new methods are needed to address specific tasks, such as spatial pattern detection and cell-cell interaction identification. On the other hand, integrative tools are also needed to facilitate end-to-end data analysis by using state-of-the-art methods. To this end, we have developed a powerful software package, called Giotto, for comprehensive analysis and interactive visualization of spatial transcriptomic data (Dries et al., 2021b). In this paper, we describe the detailed protocols for using Giotto in various tasks, including (1) getting Giotto set up for use; (2) pre-processing; (3) clustering and cell-type identification; (4) cell-type enrichment and deconvolution analyses; (5) spatial structure analysis tools; (6) spatial domain detection by using a hidden Markov random field model; (7) spatial proximity associated cell-cell interactions; and (8) assembly of a registered 3D Giotto object from 2D slices. An accompanying github repository, including all the documented code used in this article and expanded R Markdown scripts, can also be found at *https://github.com/drieslab/giotto_current_protocols*. More information, tutorials and example datasets can be found on our Giotto website at *https://rubd.github.io/Giotto_site/*.

DATA

Three main data types can be provided to the Giotto analysis pipeline. Two of these data types, a count matrix representing gene expression values and coordinates for the spatial locations, are required to run spatial transcriptomic analysis. This is sufficient to run all downstream analyses. Giotto can also be used to analyze scRNA-seq data, where only a gene expression matrix is needed, and the spatial location information is automatically filled by dummy values. In addition, a raw or processed image of the spatially profiled tissue can be provided. This image can then be used as a background and overlaid with the results obtained from the various spatial analyses to help interpret the results within the original tissue organization.

Expression count matrices should be provided to Giotto as shown in Figure 1, with genes as the row names and the spatial (cell) IDs as the column names. Spatial location data matrices (optional if running scRNA-seq data) should be provided as shown, with plotting coordinate x values in the first column, y values in the second, and z values (optional) in the third. An additional column for cell IDs can also be included. Of note, the expression (column-wise) and spatial data (row-wise) must be given in the same order, as demonstrated by how the spatial cell_ID column matches the column names of the expression matrix.

In addition, images can be provided as an additional input that can provide helpful visual context when overlaid with information obtained from the spatial transcriptomic data.


A

	8x36	9x19	9x21	9x23	9x25
RP11-34P13.7	0	0	0	0	0
FO538757.1	0	1	1	1	0
AP006222.2	0	0	0	0	0
RP4-669L17.10	0	0	0	0	0
RP11-206L10.9	0	0	0	0	0

B

	pixel_x	pixel_y	cell_ID
1	3448.0	-5756.9	8x20
2	3442.1	-6152.2	8x22
3	3446.9	-7356.7	8x28
4	3442.5	-8156.2	8x32
5	3444.9	-8557.8	8x34

Figure 1 An example of the input matrices for Giotto. (A) Subset of dataset used in this analysis demonstrating the format necessary for the expression matrix. (B) Subset of dataset used in this analysis demonstrating the format necessary for the spatial locations matrix.

Any type of image (H&E, IF, FISH, etc.) can be added and individually selected by name with each spatial plotting command in Giotto. They can be provided in any raster-based format, including `.jpg`, `.png`, or `.tiff` files.

To ensure that images and spatial locations are properly aligned, it is typically required to multiply the *y*-values from the spatial locations by -1 as we did for this example (Fig. 1B). This is due to differences between the coordinate system of an image and an R plotting canvas, where the (0,0) origin is typically top-left and bottom-left, respectively.

PROTOCOLS

To increase readability, names of commands, functions or parameters are formatted with a different font, e.g., `exampleFunction`. The following protocols are all run in R/Rstudio, unless stated otherwise. Of note, additional questions for help or issues can be posted on our github issues page (*https://github.com/RubD/Giotto/issues*), and instructions on how to submit a GitHub issue for Giotto can be found at *https://rubd.github.io/Giotto_site/articles/github_issues.html*.

GETTING GIOTTO SET UP FOR USE

BASIC PROTOCOL 1

In this protocol, we explain the system and environment prerequisites necessary for Giotto installation, as well as the creation of a Giotto object. Briefly, Giotto is an R package that can be downloaded and easily installed on most commonly used operating systems. The core of Giotto is an S4 object class (`giotto`). The created `giotto` objects facilitate the storage and analysis of spatial transcriptomic data. A `giotto` object contains multiple slots that can be broadly organized into three categories (Fig. 2). The first category includes slots that store data input and general metadata, containing expression matrices (gene by cell) and corresponding data frames with information about the cells and genes. The second category of slots relates to where the cells are in physical space and what spatial neighborhoods they form. The `images` slot containing raw image data used to overlay outputs for concurrent visualization can also be found here. The third category is concerned with where the cells are in the expression space and analyses and abstract visualizations built thereupon.

Necessary Resources

Hardware

The minimum requirements to run the different protocols are:

4-core Intel or AMD processor (or equivalent)
4 GB RAM (8 recommended)
250 GB free disk space

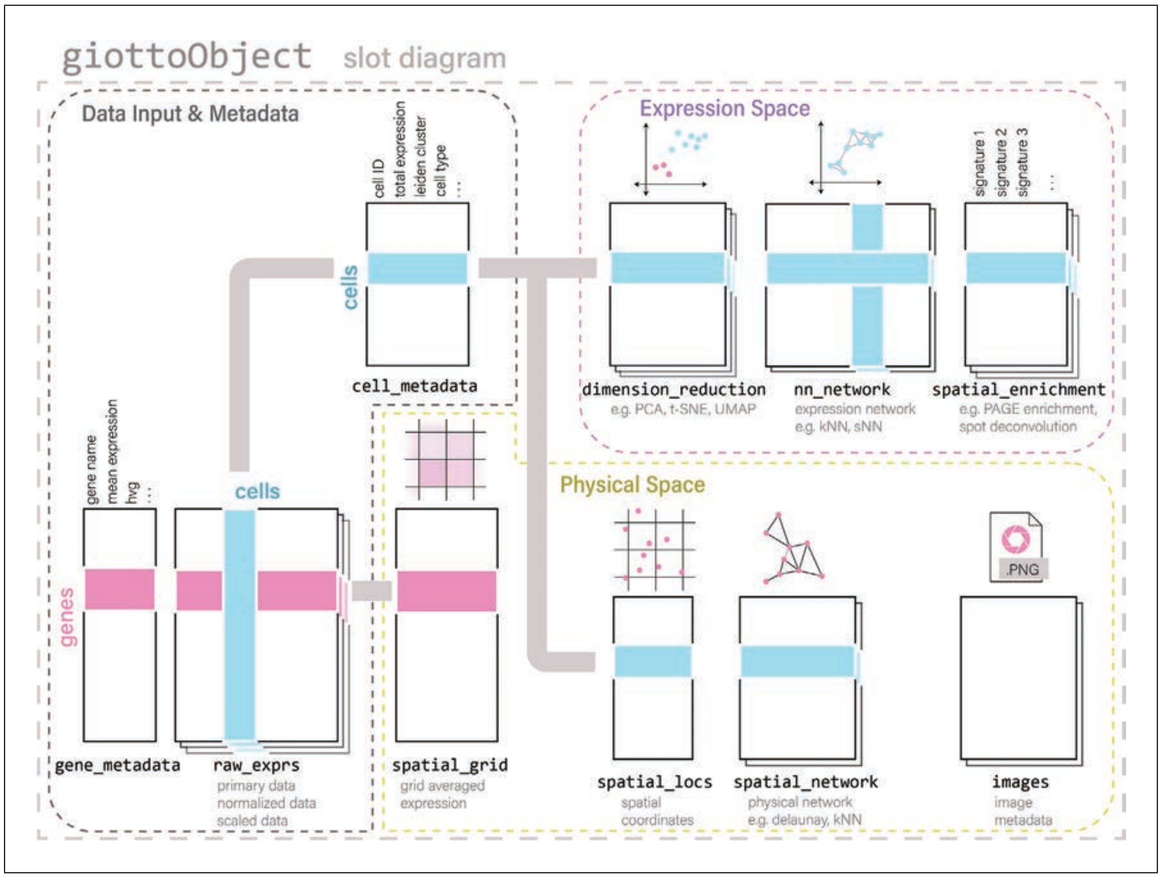


Figure 2 Schematic of the Giotto object and its core components that are used to create a spatial analysis framework.

Operating systems: Windows/Linux/Mac

Mac users will need to have a working version of Xcode, which can be installed with the following command in the terminal:

```
> xcode-select -install
```

Software

R (version ≥3.5.1); download and installation instructions: *https://www.r-project.org/*

Rstudio; download and installation instructions: *https://www.rstudio.com/products/rstudio/download/*. Rstudio is an integrated development environment (IDE) for R and it can be launched like any other application on your machine. We recommend this tool for novice R users.

Python (version ≥3.6); download and installation instructions: *https://www.python.org/downloads/*

Fiji (version ≥ 2.3.0/1.53 m); download and installation instructions *https://imagej.net/software/fiji/downloads*. Only required for supplemental protocol.

Files

The following protocols use data from Ji et al. (2020), found under Gene Expression Omnibus (GEO): GSE144240. This study aims to characterize the cell type composition and architecture of human cutaneous squamous cell carcinoma and utilizes a number of high-throughput omics technologies, including single-cell RNA sequencing (scRNA-seq), whole exome sequencing (WES), and methods to detect transcripts or proteins in a spatially aware manner. Briefly, cancerous and normal skin samples were collected from 10 patients for scRNA-seq and prepared using the 10X Genomics Chromium platform. In parallel, select tissue areas were also used to create WES and spatial data, more specifically Spatial Transcriptomics (ST) and 10× Visium were applied on fresh-frozen subsets and multiplexed ion beam imaging (MIBI) was performed on formalin-fixed tissues. For the purpose of this protocol paper, we only used scRNA-seq (both normal and tumor) and ST data from slices of patient 2. Furthermore, slice 2 was used for all 2D-based analyses and visualizations, and this section is 10 m thick with individual spatial spots that are 110 m in diameter with a center-to-center distance of 150 m. For the 3D-based analysis, we stacked all three spatial transcriptomic slices of patient 2.

1. Installation.

   Giotto can be installed directly from GitHub (*https://github.com/RubD/Giotto*) using the following code. After installation, it is necessary to load the package in R.

```
install.packages("remotes")
remotes::install_github("RubD/Giotto")
library(Giotto)
```

2. Environment setup.

   Giotto requires the installation of several Python packages to run all of the available analyses. When you run the command `installGiottoEnvironment()`, a miniconda environment that contains all the required python modules will be installed.

   The following is a demonstration of how to install a miniconda environment with Giotto:

```
installGiottoEnvironment()
```

   This command only needs to be run once, and in subsequent R sessions, Giotto will automatically detect the installed Giotto miniconda environment and use that unless instructed otherwise. For example, if you always choose to direct Giotto to your own preferred python path, you will need to manually install all the necessary packages with pip or anaconda:

   pandas (1.1.5)
   python-igraph (0.9.6)
   networkx (2.6.3)
   python-louvain (0.15)
   leidenalg (0.8.7)
   scikit-learn (0.24.2)
   smfishHmrf.

   Due to differences in python module versions between your manual python environment and the Giotto miniconda environment, slight differences in the subsequent downstream spatial analyses might be observed. We have provided the current version numbers above, but in the event that anything has been updated, the exact module versions can be found on the help page of `installGiottoEnvironment()` using the command `help("installGiottoEnvironment")`.

3. Downloading dataset.

   Once you have installed the Giotto package, you can simply download the data directly to your own preferred directory using the `getSpatialDataset()` function as illustrated below. The datasets can also be found on our GitHub (*https://github.com/RubD/spatial-datasets/tree/master/data/2020_ST_SCC*). This command is a convenience function that allows users to directly download a number of different datasets, obtained from various technologies, to test and learn to work with the different Giotto workflows. Here we assigned our data directory path to the variable `data_dir`. All necessary files can be directly accessed from that location as illustrated in our example code.

```
data_directory <- "~/ST_SCC_data"
save_directory <- "~/save_dir"
# Download data
getSpatialDataset(dataset   "ST_SCC", directory   data_directory, method   "wget")
```

4. Creating a Giotto object.

   Before creating a Giotto object, we can create specific instructions for our Giotto analysis workflow. Although this step is optional, it makes it possible to specify the default behavior of the Giotto pipeline, including which python path to use and how figures will be saved and displayed automatically. For example, by setting both `show_plot` and `return_plot` to FALSE and only directly saving the plots to your designated directory by setting `save_plot` to TRUE, we can eliminate long plotting times caused by large spatial datasets.

```
my_instructions <- createGiottoInstructions(save_plot   TRUE,
                                            show_plot   TRUE,
                                            return_plot   FALSE,
                                            save_dir   save_directory)
```

   After your instructions are set up, you can proceed to creating your Giotto object. The only necessary input is a set of feature data, such as gene expression data, and spatial locations. An additional image may be provided as shown in the next step. Of note, scRNA-seq data can also be processed and analyzed with Giotto, in which case you only need to provide the feature data.

```
my_giotto_object <- createGiottoObject(raw_exprs paste0(data_directory,
                                                   "/P2_2_expression.csv"),
                                       spatial_locs paste0(data_directory,
                                                   "/P2_2_spatial_locs.csv"),
                                       instructions my_instructions)
```

   Giotto has built-in functions to easily access slots such as cell or gene metadata with the functions `pDataDT()` and `fDataDT()`, respectively.

```
pDataDT(my_giotto_object)
fDataDT(my_giotto_object)
```

   If there are one or multiple images to associate with spatial data, then you can load them in as a `giottoImage` object. For purposes of initializing the alignment of image to spatial data, you can point to the Giotto object which already contains your spatial locations with the `gobject` argument. `GiottoImage(s)` are then assigned to the Giotto object as a list.

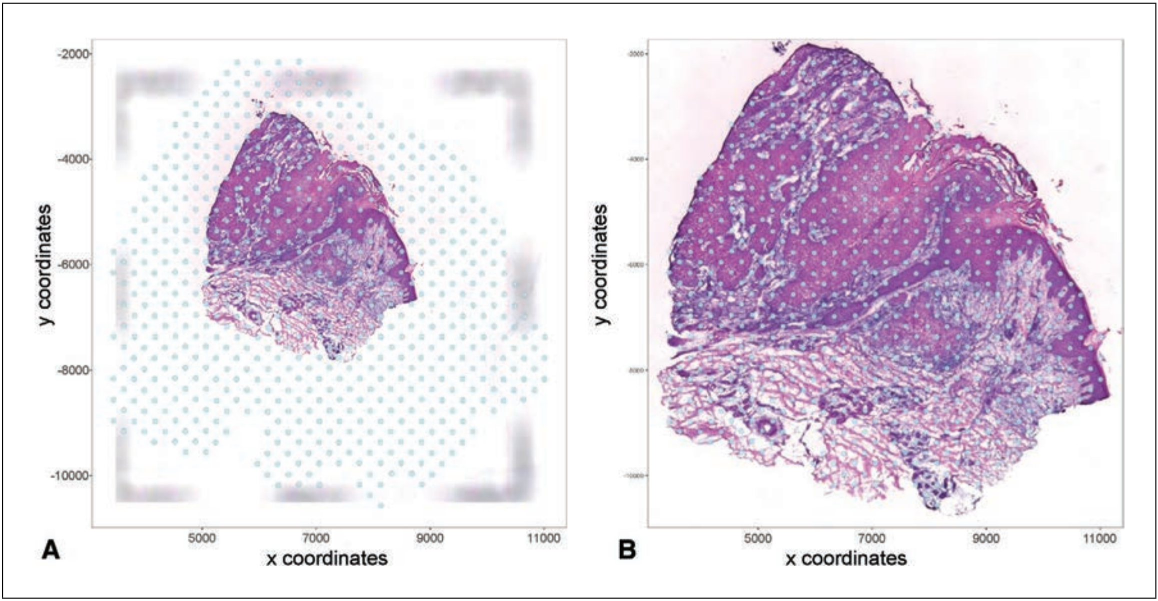


Figure 3 Spatial location data overlaid on a staining image of tissue (A) before and (B) after the Giotto image is scaled and updated.

```
my_giotto_image <- createGiottoImage(gobject    my_giotto_object,
                                     mg_object    paste0(data_directory,
                                                         "/P2_2_0.0625.jpg"))
my_giotto_object <- addGiottoImage(gobject    my_giotto_object,
                                   images    list(my_giotto_image))
```

If both expression and image data are available, it is often informative to overlay the information together in data visualization.

```
spatPlot2D(gobject    my_giotto_object,
           show_image    TRUE,
           point_alpha    0.5)
```

However, as indicated in Figure 3A, the images and expression data may not align correctly without adjustment. Giotto provides the function `updateGiottoImage()` for alignment adjustment. By default, images are stretched to cover as much space as the spatial locations do, and this is often not enough since images tend to be larger than capture regions. Adjustments increase the scaling of the picture in terms of distance away from the largest or smallest *x* or y spatial location value (Fig. 4). For datasets obtained through the commercially available Visium kit users can directly use the function create `createGiottoVisiumObject()`, which will automatically extract all necessary information from the `visium` folder and rescale the image according to the Visium provided scaling factors.

Giotto also provides the functionality to automatically align the image properly if the user knows the scale factor of the image relative to the spatial locations. This will be demonstrated as an alternative in the associated R Markdown. For the purposes of this paper, we show how to adjust and scale the image manually with Giotto. The following code can be used to view the image after proper alignment (Fig. 3B).

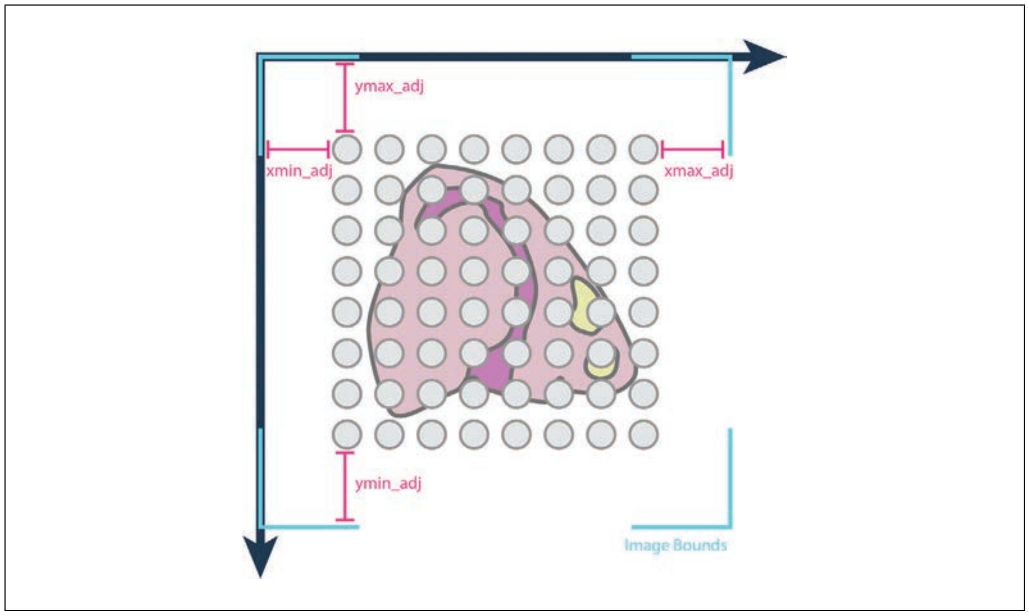


Figure 4 Spatial location data portrayed as gray spots are overlaid on a staining image of the tissue. By default, the image is stretched to cover as much space as the spatial extents, but if this is inaccurate, adjustment values can be used to adjust where the image bounds (in teal) should be.

```
my_giotto_object <- updateGiottoImage(gobject   my_giotto_object,
                                      image_name   "image",
                                      xmax_adj   4857.2,
                                      xmin_adj   3441.1,
                                      ymax_adj   2146.7,
                                      ymin_adj   5302.5)
spatPlot2D(gobject   my_giotto_object,
           show_image   TRUE,
           point_alpha   0.5)
```

BASIC PROTOCOL 2

PRE-PROCESSING

The first pre-processing step we will discuss is filtering. Filtering input expression data is achieved through the removal of low-quality cells and/or lowly-expressed genes, and is necessary to reduce data noise. Giotto implements gene filtering through expression thresholds and cell filtering based on the number of cells a given gene is detected in and the total number of genes detected per cell.

Following the filtering step, Giotto normalizes the filtered data for sequencing depth so that the results can be appropriately compared. Giotto offers a standard, but adaptable, method of normalization, which involves library size normalization and scaling, log transformation, and *z*-scoring by genes and/or cells. In addition, Giotto implements the normalization approach used in the osmFISH paper (Codeluppi et al., 2018). Next, gene and cell statistics can be generated. This may be useful for further exploratory data analysis (EDA) in combination with additional biological or study design information. For example, we can account for and reduce experimental artifacts or known technical factors, such as batch effects or the percentage of mitochondrial gene content within each cell.

Necessary Resources

See Basic Protocol 1

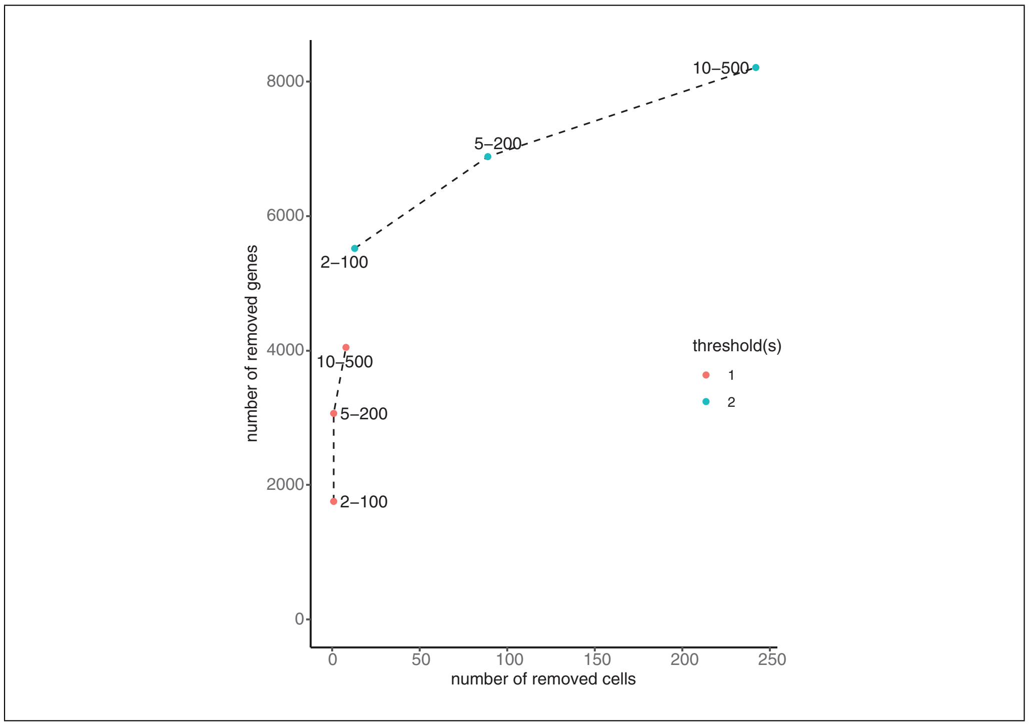


Figure 5 Effect of different choices of expression filtration combinations. The legend on the right-hand side denotes the minimum expression threshold, while each point on the plot represents the combinations of the minimum number of cells a gene is detected in and the minimum number of genes a cell detects. The *x*-axis demonstrates the number of cells that would be removed, while the *y*-axis demonstrates the number of genes that would be removed.

1. Filtering.

   After creating your Giotto object, you can filter your expression data in several ways including expression threshold, minimum number of cells a gene is detected in, and minimum number of genes expressed per cell. To assess how many cells/spots or genes should be filtered out, the following function will create a visualization (Fig. 5) demonstrating the number of observations that may be lost based on various filtering combinations.

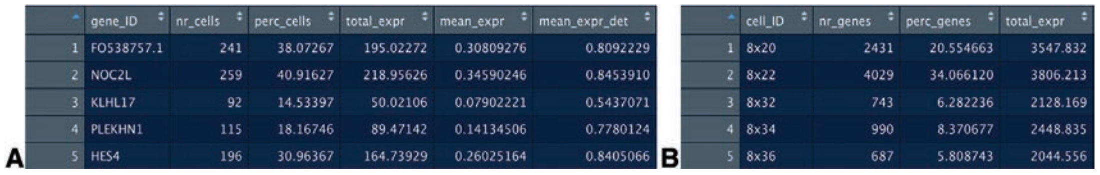
```
filterCombinations(gobject   my_giotto_object,
                   expression_thresholds   c(1, 2),
                   gene_det_in_min_cells   c(2, 5, 10),
                   min_det_genes_per_cell   c(100, 200, 500))
```

   Based on the above plot as well as printed output, you can select the appropriate combination and apply this choice to your Giotto object. The following function will filter out based on your selected parameters.

```
my_giotto_object <- filterGiotto(gobject   my_giotto_object,
                                 expression_threshold   2,
                                 gene_det_in_min_cells   2,
                                 min_det_genes_per_cell   100)
```

2. Normalization.

   Following data filtration, we will apply our normalization steps. Out of the normalization options described in the introduction (standard and the method displayed by


A

	gene_ID	nr_cells	perc_cells	total_expr	mean_expr	mean_expr_det
1	FO538757.1	241	38.07267	195.02272	0.30809276	0.8092229
2	NOC2L	259	40.91627	218.95626	0.34590246	0.8453910
3	KLHL17	92	14.53397	50.02106	0.07902221	0.5437071
4	PLEKHN1	115	18.16746	89.47142	0.14134506	0.7780124
5	HES4	196	30.96367	164.73929	0.26025164	0.8405066

B

	cell_ID	nr_genes	perc_genes	total_expr
1	8x20	2431	20.554663	3547.832
2	8x22	4029	34.066120	3806.213
3	8x32	743	6.282236	2128.169
4	8x34	990	8.370677	2448.835
5	8x36	687	5.808743	2044.556

Figure 6 Summary statistics for gene metadata. (A) A subset of the gene metadata data frame contained in the Giotto object, which consists of the following: number of cells in which the gene is expressed, percentage of cells that express the gene, total gene expression, average gene expression, and average expression detected. (B) A subset of the cell metadata data frame contained in the Giotto object, which consists of the following: number of genes expressed in each cell, percentage of genes, and total gene expression per cell.

Codeluppi et al., 2018), we will use a standard method of normalization for this example. You can also choose whether you will scale genes or cells first, but the default, and the method we will use, is to scale genes first.

```
my_giotto_object <- normalizeGiotto(gobject   my_giotto_object,
                                    norm_methods   "standard",
                                    scalefactor   6000,
                                    scale_order   "first_genes")
```

3. Statistics.

   You can also view some summary statistics of your data. Giotto offers the following insights.

   Gene statistics:

   nr_cells: number of cells the gene is detected in
   per_cells: percentage of cells the gene is detected in
   total_expr: total sum of gene expression in all cells
   mean_expr: average gene expression in all cells
   mean_expr_det: average gene expression in cells with detectable levels of the gene.

   Cell statistics:

   nr_genes: how many genes are detected in the cell
   perc_genes: percentage of genes detected per cell
   total_expr: total sum of gene expression per cell.

```
my_giotto_object <- addStatistics(gobject   my_giotto_object)
```

We can use the following code to view the summary statistics for gene metadata (Fig. 6A) and cell metadata (Fig. 6B), respectively.

```
# view gene and cell stats respectively
head(fDataDT(my_giotto_object))
head(pDataDT(my_giotto_object))
```

To account for batch effects or technological covariates, you can use the function `adjustGiottoMatrix()`.

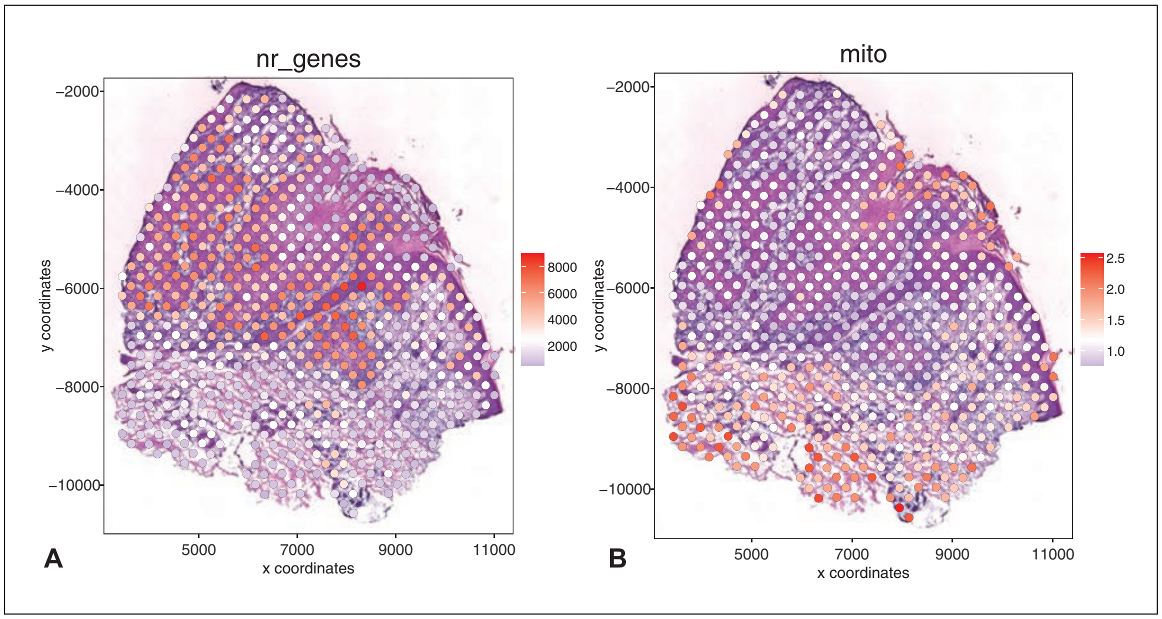


Figure 7 Visualization/exploration of potential technical confounding factors. (A) Spatial plot representing number of genes per spot. (B) Spatial plot representing mitochondrial percentage per spot.

For example, we can calculate the percentage of mitochondrial content per spot by using the following code.

```
mitochondrial_genes    grep('MT-', my_giotto_object@gene_ID, value    T)
my_giotto_object    addGenesPerc(gobject    my_giotto_object,
                                 genes    mitochondrial_genes, vector_name    'mito')
```

We can also visualize the number of genes per spot (Fig. 7A) as well as the mitochondrial percentage per spot (Fig. 7B).

```
# number of genes
spatPlot2D(gobject    my_giotto_object,
            show_image    TRUE,
            point_alpha    1,
            point_size    5,
            cell_color    'nr_genes', color_as_factor    F)
# mitochondrial content percentage
spatPlot2D(gobject    my_giotto_object,
            show_image    TRUE,
            point_alpha    1,
            point_size    5,
            cell_color    'mito', color_as_factor    F)
```

To adjust our matrix to account for these technical confounders, we can run the following code. The user can choose which expression slot to update, e.g., the ‘custom’ expression slot, which could then be selected and used in subsequent downstream analysis by specifying the `expression_values` parameter.

```
my_giotto_object <- adjustGiottoMatrix(gobject    my_giotto_object,
                                       covariate_columns    c('nr_genes', 'mito'),
                                       update_slot    'custom')
```

BASIC PROTOCOL 3

CLUSTERING AND CELL-TYPE IDENTIFICATION

In this protocol, we will discuss how unsupervised clustering analysis is implemented in Giotto to identify cell types, and how those results can be utilized in downstream analyses. Before running a clustering analysis, it is typically advised to perform feature selection to retain the most informative genes in order to optimize the signal-to-noise ratio. This can be achieved by calculating highly variable genes (HVGs). Giotto currently implements two commonly used methods for HVG identification. The default method uses coefficients of variation (cov) groups. Genes are grouped into equal-sized bins (with a default of 20), and a cov for each gene is calculated and converted to a *z*-score per bin. Genes that have a *z*-score above the set threshold (default of 1.5) are considered as highly variable. The alternate method for identifying HVGs is using a Loess regression model, which predicts expected cov using log-normalized expression values. Genes that have significantly higher cov than predicted by the model are considered to be highly variable.

Due to the high dimensionality of spatial transcriptomic data, dimensionality reduction is a commonly used step to aid data analysis and visualization. Giotto implements a number of common approaches for dimensionality reduction. The simplest approach is principal component analysis (PCA), which is a linear projection of the data to directions associated with the highest variance, the results of which can be visualized through either a Scree (“elbow”) plot or a jackstraw plot to determine the number of significant principal components (Chung, 2020). On the other hand, the linear assumption underlying PCA is often too restricted and does not represent the full complexity of all cells. To overcome these limitations, more sophisticated dimensionality-reduction methods have been developed to account for nonlinearity. Giotto implements two widely used nonlinear methods: *t*-distributed stochastic neighbor embedding (*t*-SNE; van der Maaten & Hinton, 2008) and Uniform Manifold Approximation and Projection (UMAP; Becht et al., 2018).

Following identification of HVGs and dimensionality reduction, we can begin the clustering process. Giotto implements four commonly used clustering algorithms: *k*-means, hierarchical clustering (Traag, Waltman, & van Eck, 2019), Louvain community detection (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008), and Leiden clustering (Traag et al., 2019). The latter two are graph-based methods and more routinely used in single-cell and spatial analyses. In order to apply these methods, a shared or *k*-nearest neighbor network is created in advance. This can be achieved by using either the preprocessed expression values or the outcome of dimensionality reduction analysis. For proper biological interpretation, manual identification is often needed to annotate cell types based on the clustering results. Sometimes this involves additional processing such as iterative clustering or merging of similar clusters.

To aid biological interpretation, it is important to identify marker genes that discriminate between different clusters/cell-types. Giotto implements three established algorithms to detect differentially expressed genes (DEGs). Scran (Lun, McCarthy, & Marioni, 2016) performs pairwise *t*-tests between each cluster and then combines and compares the results to determine which genes are statistically significantly upregulated. MAST (Finak et al., 2015) implements the hurdle model, which first identifies whether a gene is expressed, and if so, whether the expression level exceeds a defined “hurdle” value. The Gini-coefficient method (Jiang, Chen, Pinello, & Yuan, 2016) ranks genes based on the Gini coefficient and selects the top genes, adjusted by background removal, as cell-type specific markers.

Necessary Resources

See Basic Protocol 1

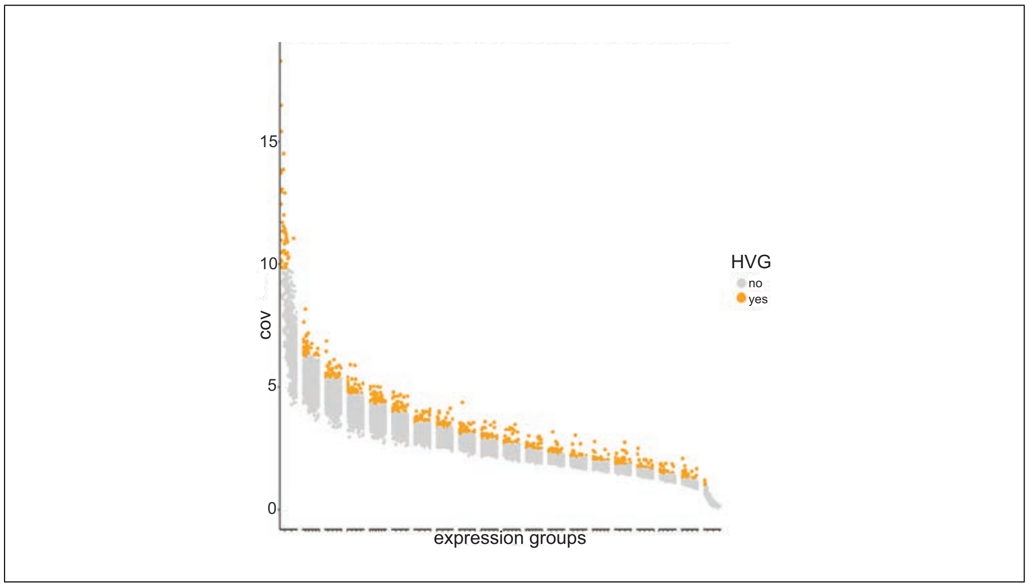


Figure 8 The distribution of HVGs across the binned groups. Each identified HVG is indicated by an orange dot.

1. Feature selection.

   Following pre-processing, we can start our clustering process with feature selection. We will use Giotto to calculate and visualize highly variable genes (HVGs). As mentioned above, Giotto has two available methods for identifying such genes: coefficient of variation (cov) groups and Loess regression. For this demonstration, we will use the cov groups method (also the default). Following analysis, Giotto displays a plot demonstrating the distribution of HVGs (Fig. 8).

```
my_giotto_object <- calculateHVG(gobject   my_giotto_object,
                                 expression_values   "normalized",
                                 method   "cov_groups",
                                 nr_expression_groups   20,
                                 zscore_threshold   1.5)
```

2. Dimensionality reduction.

   Following HVG identification, we will run dimensionality reduction. First, we will run a linear analysis: principal component analysis (PCA). As you can see in the following code block, we have specified that we will use HVGs for this analysis.

```
my_giotto_object <- runPCA(gobject   my_giotto_object,
                           expression_values   "normalized",
                           genes_to_use   "hvg")
```

   We can now visualize our data (Fig. 9) following dimensionality reduction using the following code.

```
plotPCA(gobject   my_giotto_object)
```

   After PCA has been run, we can visualize our results with a Scree plot (Fig. 10) to identify which principal components to use in downstream analyses.

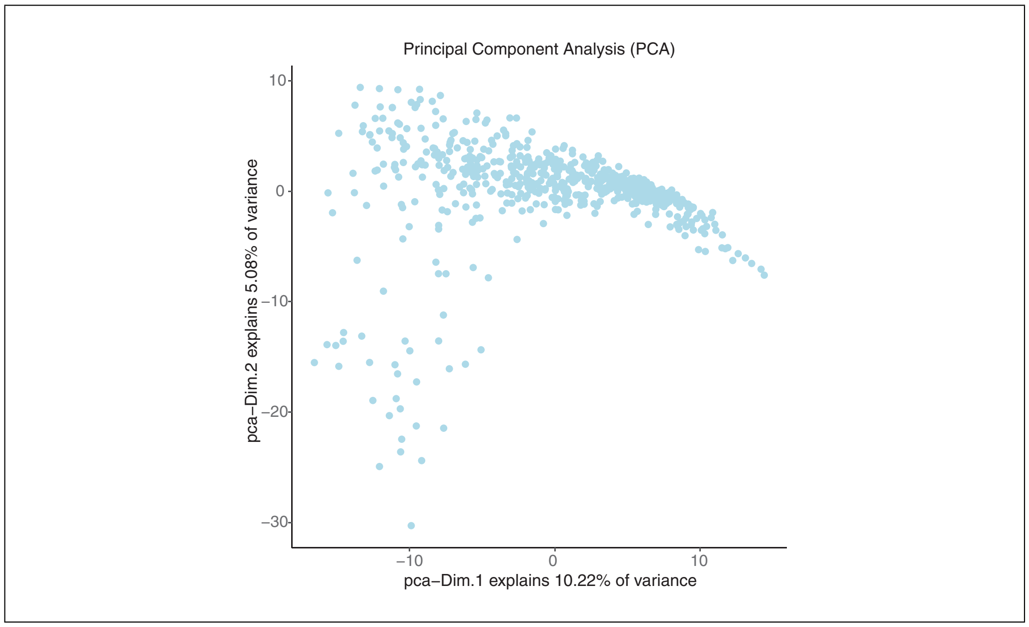


Figure 9 Visualization of the principal component analysis results. Each dot represents the reduced-dimension gene expression pattern associated with a ST data point. The top two dimensions are shown for visualization. The percentage of total variance explained by each dimension is indicated in the *x*- and *y*-axis labels.

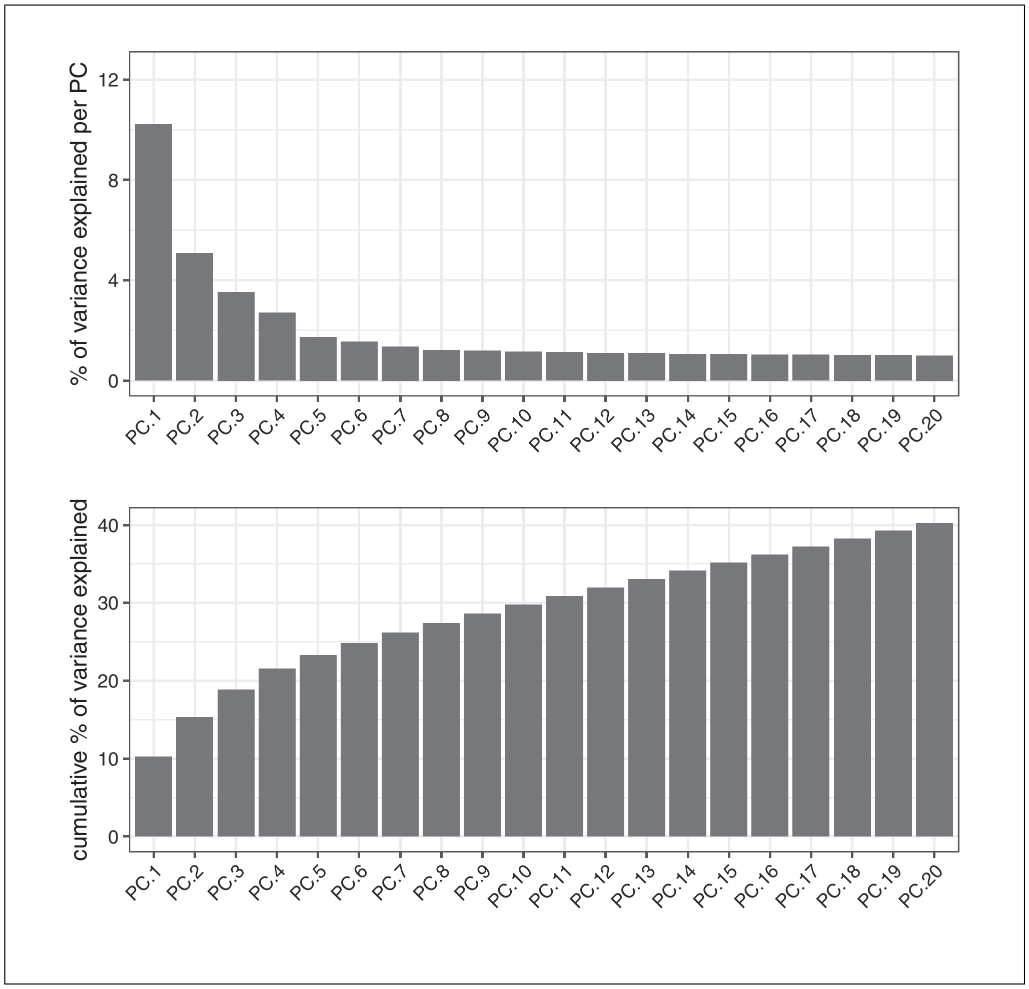


Figure 10 Scree plot demonstrating the amount of explained variance from each principal component. The top plot shows the individual percentage of explained variance, while the bottom plot shows the cumulative percentage of explained variance.

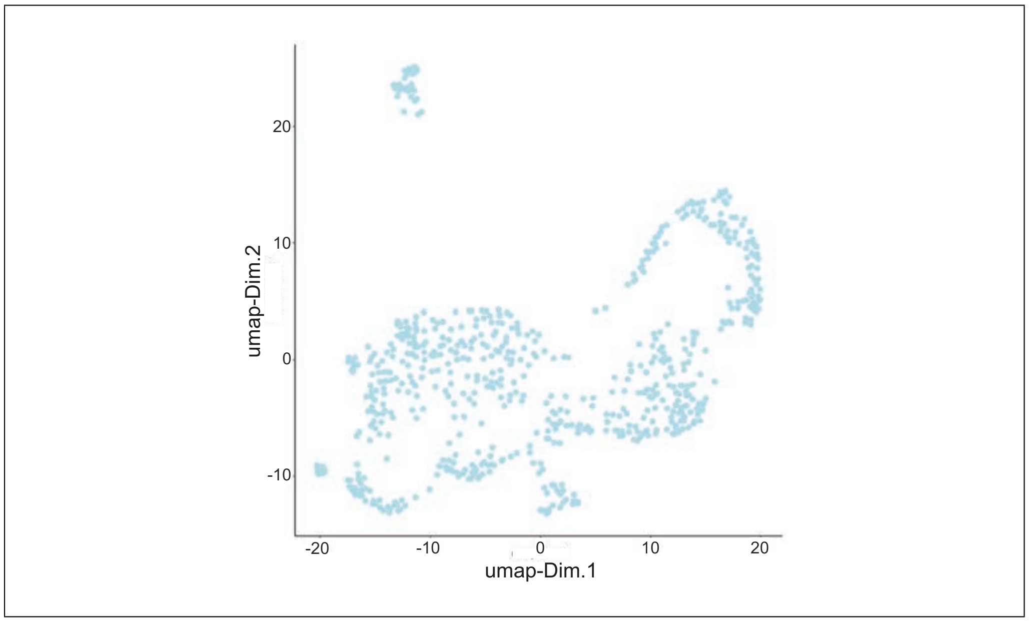


Figure 11 UMAP plot showing the results distributed across two dimensions.

```
screePlot(gobject   my_giotto_object,
           expression_values   "normalized",
           genes_to_use   "hvg",
           ncp   20,
           ylim   c(0, 12.5))
```

To use a less restrained method of dimensionality reduction, we can use a non-linear analysis. In this example, we will use Uniform Manifold Approximation Projection (UMAP). These results can be visualized with a scatter plot (Fig. 11).

```
My_giotto_object <- runUMAP(gobject   my_giotto_object,
                            dimensions_to_use   1:10,
                            n_components   2)
# to plot our umap:
plotUMAP(my_giotto_object)
```

Clustering

3. Creating a gene-expression-based nearest network:

   Before running a clustering algorithm, we will create a nearest network based on gene expression similarities. For this example, we will be creating a shared nearest network (sNN).

```
my_giotto_object <- createNearestNetwork(gobject   my_giotto_object,
                                         dimensions_to_use   1:10)
```

4. Clustering algorithms:

   Now we can run our clustering analysis. For this example, we will use Leiden clustering with the sNN that we previously created.

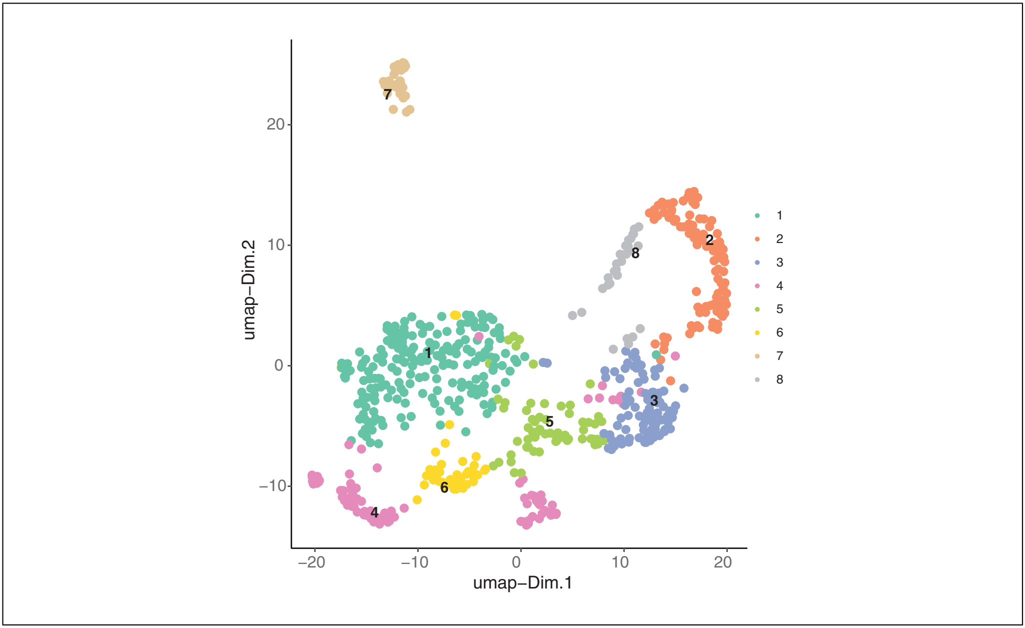


Figure 12 Clustering results combined with our UMAP plot to show the distinction between the identified classes across the two dimensions.

```
my_giotto_object <- doLeidenCluster(gobject   my_giotto_object,
                                     name   "leiden_clus")
```

To visualize our clustering results (Fig. 12), we can run the following:

```
plotUMAP(gobject   my_giotto_object,
          cell_color   'leiden_clus',
          point_size   2)
```

5. Differentially Expressed Genes (DEGs):

   Following clustering, we will interpret our results by finding differentially expressed genes (DEGs) between the identified Leiden clusters.

```
ST_scran_markers_subclusters   findMarkers_one_vs_all(gobject   my_giotto_object,
                                                   method   'scran',
                                                   expression_values   'normalized',
                                                   cluster_column   'leiden_clus')
```

   Now we can use a heatmap (Fig. 13) to visualize the correlation between the top selected marker genes and the identified Leiden clusters.

```
ST_top3genes   ST_scran_markers_subclusters[, head(.SD, 3), by   'cluster']$genes
plotMetaDataHeatmap(gobject   my_giotto_object,
                       selected_genes   ST_top3genes,
                       metadata_cols   c('leiden_clus'))
```

   This analysis indicates that several clusters show high expression for cell-type-specific genes, such as genes from the Keratin family (e.g., KRT1, KRT2) or major histocompatibility complex (e.g., HLA-A, HLA-B), which correspond to epithelial and myeloid cell types, respectively. However, since each spot within a cluster is

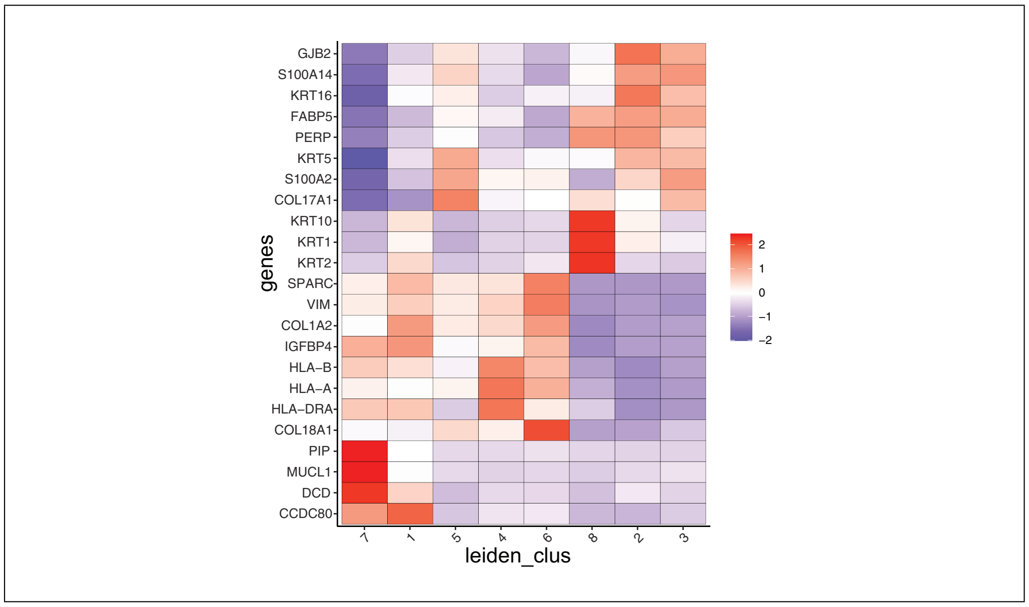


Figure 13 Heatmap demonstrating the correlation between selected marker genes and cell-type.

110 m in diameter, it is likely that it covers multiple different cell types and that these results are skewed by the dominant cell type or very highly expressed genes within one cell type. To overcome this spot or cell annotation obstacle, we will integrate external scRNA-seq information (next Basic Protocol 4) to identify the present cell types more accurately within each spot at the single-cell level.

CELL-TYPE ENRICHMENT AND DECONVOLUTION ANALYSES

BASIC PROTOCOL 4

In this protocol, we will provide an overview of Giotto's functionalities to perform cell-type enrichment and deconvolution analyses by using external information from a matching single-cell RNA-seq data. For spatial transcriptomic datasets that do not have single-cell resolution, cell-type enrichment is a useful step to identify the spatial distribution of various cell types. Giotto implements three commonly used methods for cell-type enrichment. Parametric Analysis of Gene Set Enrichment (PAGE) calculates the z-score for gene sets based on fold changes and evaluates statistical significance based on the normal distribution assumption (Kim & Volsky, 2005). Rank enrichment creates a ranking-based statistic based on the degree of cell-type specificity. Hypergeometric enrichment utilizes the hypergeometric test to evaluate whether the expression levels of cell-type specific signature genes are high at each spatial location. A limitation of cell-type enrichment analysis is that it does not provide quantitative estimates of the relative proportion of different cell types at each location. This limitation is addressed by using cell-type deconvolution analysis. Giotto implements the SpatialDWLS algorithm (Dong & Yuan, 2021) for cell-type deconvolution, which combines cell-type enrichment analysis with a dampened weighted least squares (DWLS) algorithm (Tsoucas et al., 2019) previously developed for bulk RNAseq deconvolution from scRNA-seq data.

Necessary Resources

See Basic Protocol 1

1. Implementation.

   To perform spatial cell-type enrichment or deconvolution, we will make use of the patient-matched scRNA-seq dataset. Here we already provide the processed data, which consist of the normalized count matrix, the identified cell type vector, and the associated marker genes per cell type cluster.

```
# normalized matrix
normalized_sc_matrix <- readRDS(paste0(data_directory,"/", "normalized_sc_matrix.RDS"))
# cell type vector
cell_type_vector <- readRDS(paste0(data_directory,"/", "cell_type_vector.RDS"))
# list of marker genes
sign_list <- readRDS(paste0(data_directory,"/", "sign_list.RDS"))
```

Of note, these scRNA-seq results can be reproduced using Giotto or provided from another single-cell RNA-seq pipeline. More specifically, with Giotto this dataset can be processed by using the same steps as described above: normalization, dimensionality reduction, and clustering, in the same way as a spatial transcriptomic dataset, because none of these steps require spatial information. You would simply add the expression data file to the `raw_exprs` argument in `createGiottoObject()`, and dummy spatial locations will be created. For the exact code and more information on how this Giotto object was preprocessed, please see the attached R markdown (*https://github.com/drieslab/giotto_current_protocols*).

2. Cell-type enrichment.
We can now use the results from our previous clustering analyses to perform cell-type enrichment. Using our significant differentially expressed genes along with matched cell types found in the previous step, we can produce a signature matrix, which is binary.

```
# list of signature genes
PAGEsignMatrix <- makeSignMatrixPAGE(sign_names   names(sign_list),
                                     sign_list   sign_list)
```

After creating the above signature matrix, we can run Parametric Analysis of Gene set Enrichment (PAGE).

```
my_giotto_object <- runPAGEEnrich(gobject   my_giotto_object,
                                  sign_matrix   PAGEsignMatrix)
```

After the analysis is complete, we will visualize our results (Fig. 14).

```
cell_types_subset <- colnames(PAGEsignMatrix)
spatCellPlot(gobject   my_giotto_object,
             spat_enr_names   'PAGE',
             cell_annotation_values   cell_types_subset,
             cow_n_col   3, coord_fix_ratio   1, point_size   0.75,
             point_shape   "no_border")
```

3. Spatial cell–type deconvolution.

Next, we will perform spatial cell-type deconvolution using the spatialDWLS algorithm mentioned above. First, we will create a signature matrix specific to the spatialDWLS algorithms. The following function creates the signature matrix by calculating average gene expression for each signature gene in each cell/cell-type.

```
dwls_signature_matrix <- makeSignMatrixDWLSfromMatrix(matrix   normalized_sc_matrix,
                                sign_gene   unlist(sign_list),
                                cell_type_vector   cell_type_vector)
```

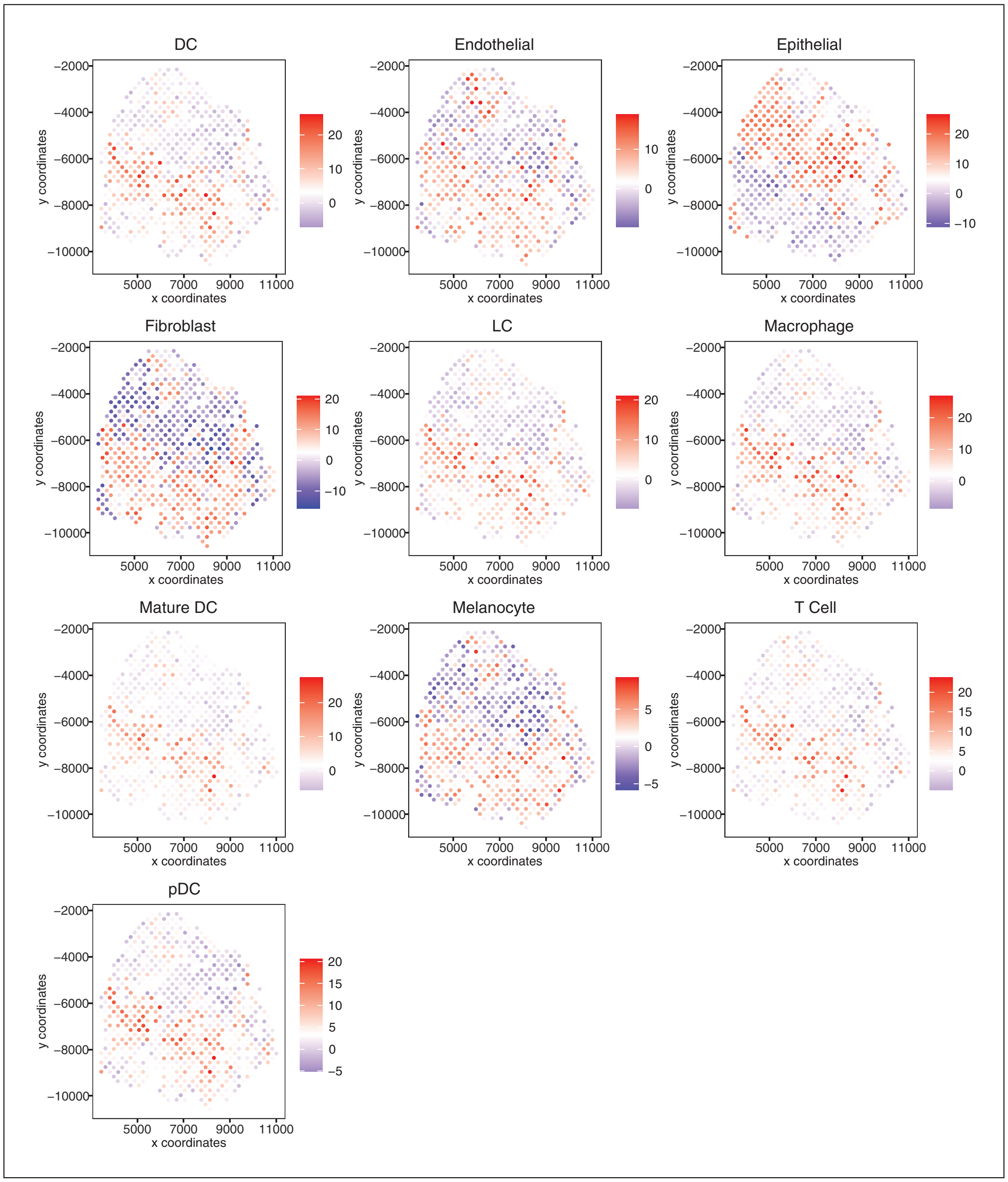


Figure 14 PAGE results demonstrating presence of cells at each spatial location.

We will now use the signature matrix we created to run our spatialDWLS analysis.

```
my_giotto_object <- runDWLSDeconv(gobject   my_giotto_object,
                                  sign_matrix   dwls_signature_matrix)
```

We have also developed a function to easily visualize the proportions of cell-type per spot, shown by a spatial dot plot with pie charts at each of the locations. Below, we show the Giotto image of the sample slice with the spatialDWLS results overlaid (Fig. 15). Each spot on the plot is a pie chart that represents the percentage of cell-types.

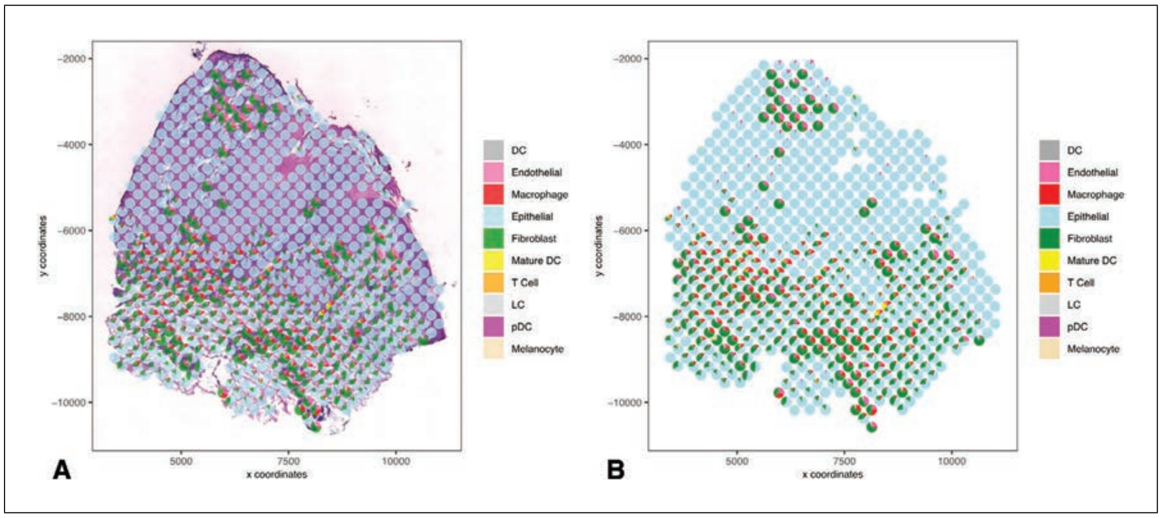


Figure 15 Results from the spatialDWLS analysis. Each pie chart demonstrates the cell-type percentage at each spot. (A) Overlaid on the H&E image. (B) Pie charts displayed independently for easier interpretation.

```
colors <- c('darkgrey', 'hotpink', 'red', 'lightblue', 'green4',
             'yellow', 'orange', 'lightgray', 'magenta', 'wheat')
spatDeconvPlot(gobject   my_giotto_object,
                radius   100,
                cell_color_code   colors,
                show_image   TRUE,
                return_plot   TRUE)
```

BASIC PROTOCOL 5

SPATIAL STRUCTURE ANALYSIS TOOLS

In this protocol, we will describe how to use Giotto to identify spatial relationships between cells and genes. Giotto provides a spatial network function based on the spatial proximity of cells. The default setting, a Delaunay network, utilizes a triangulation approach. Alternatively, the user can apply a *k*-nearest neighbor analysis by specifying the values of *k* (number of neighbors) and search radius (distance-specific neighborhood). In addition, Giotto also implements a coarse-resolution representation of the data as a spatial grid, which can be useful for visualizing large-scale spatial structures. A spatial grid is created by subdividing the image fields into uniform squares, with user-defined resolution. The average gene expression profile within each square is reported.

Giotto also implements four methods for detecting spatial genes with coherent gene expression patterns, including Binary Spatial Extraction of Genes (binSpect) (Dries et al., 2021b), and three previously published methods: SpatialDE (Svensson, Teichmann, & Stegle, 2018), Transduces (Edsgärd, Johnsson, & Sandberg, 2018), and SPARK (Sun, Zhu, & Zhou, 2020). Further analysis can be done to group spatial genes into distinct modules based on spatial co-expression analysis. We found that the metagene corresponding to each module typically displays enhanced spatial patterns compared to individual genes.

Necessary Resources

See Basic Protocol 1