Voyager: exploratory single-cell genomics data analysis with geospatial statistics

Lambda Moses[1], Pétur Helgi Einarsson[2], Kayla Jackson[1], Laura Luebbert[1], A. Sina Booeshaghi[1], Sindri Antonsson[2], Páll Melsted*[2], Lior Pachter*[1,3]
[1]Division of Biology and Biological Engineering, California Institute of Technology, Pasadena, CA, USA
[2]Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, Reykjavík, Iceland
[3]Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA

*Address correspondence to pmelsted@hi.is and lpachter@caltech.edu.

Abstract

Exploratory spatial data analysis (ESDA) can be a powerful approach to understanding single-cell genomics datasets, but it is not yet part of standard data analysis workflows. In particular, geospatial analyses, which have been developed and refined for decades, have yet to be fully adapted and applied to spatial single-cell analysis. We introduce the Voyager platform, which systematically brings the geospatial ESDA tradition to (spatial) -omics, with local, bivariate, and multivariate spatial methods not yet commonly applied to spatial -omics, united by a uniform user interface. Using Voyager, we showcase biological insights that can be derived with its methods, such as biologically relevant negative spatial autocorrelation. Underlying Voyager is the SpatialFeatureExperiment data structure, which combines Simple Feature with SingleCellExperiment and AnnData to represent and operate on geometries bundled with gene expression data. Voyager has comprehensive tutorials demonstrating ESDA built on GitHub Actions to ensure reproducibility and scalability, using data from popular commercial technologies. Voyager is implemented in both R/Bioconductor and Python/PyPI, and features compatibility tests to ensure that both implementations return consistent results.

Introduction

From the developing embryo to the hepatic lobule, spatial organization of cells is essential to the functions of many tissues. Recent breakthroughs in technology development, data collection, and data analysis tools for spatial transcriptomics, have led to a plethora of possibilities and applications[1]. Among these data analysis tools are overarching data analysis frameworks for data organization and exploratory data analysis (EDA), including frameworks such as Seurat[2], squidpy[3], Giotto[4], and semla (formerly STUtility[5]), which add basic spatial data analysis functionalities and visualizations to traditional single-cell RNA-seq (scRNA-seq) analysis workflows. In addition, for the purpose of EDA, many visualization tools have been developed for spatial -omics data. Many of these visualization tools are designed to be scalable and

interactive for large imaging-based data such as MERFISH[6] and imaging mass cytometry, plotting gene expression and cell metadata in space, or utilizing virtual reality[7–11].

EDA is an approach to understanding data "without many preconceived ideas, theories, or hypotheses"[12]. It encourages a mindset of asking questions and exploring possible answers by visualizing, transforming, and modeling relevant data, leading to further, more refined questions without a formal process or strict set of rules[13]. The spirit of EDA is that "it is important to understand what you *can do* before you learn to measure how *well* you have seem to have *done* it"[14]. Exploratory *spatial* data analysis (ESDA) is EDA explicitly focusing on spatial aspects of the data, especially spatial autocorrelation, where nearby observations are not independent from each other[12]. ESDA has a long history of use in geography, where a rich tradition has been developed[15]. The widely used spatial autocorrelation metrics Moran's I[16] and Geary's C[17] are among the global univariate spatial statistics used in ESDA, which produce one set of results for the entire dataset. The characteristics of Moran's I and Geary's C have been further elaborated over the years[18–20]. In addition, there are tools to explore the length scale of spatial autocorrelation, such as the correlogram[21] and variogram[22]. Local versions of spatial statistics, such as local Moran's I[23] and Getis-Ord Gi* [24] can be used to explore local spatial heterogeneity and find spatial clusters of high or low values, producing a set of results for each location. There are also spatially informed global and local bivariate and multivariate statistics that account for spatial autocorrelation and correlation between features simultaneously, such as Lee's L[25] and MULTISPATI PCA[26].

Much of this ESDA tradition can benefit spatial -omics, but has not been utilized in existing frameworks such as Seurat, squidpy, and Giotto. For example, while Seurat and squidpy can perform global Moran's I, squidpy implements Ripley's L to analyze cell type clustering in space, while Giotto implements something similar to Lee's L for spatially informed gene co-expression, and semla implements Moran's I and Getis-Ord Gi; none of these tools have systematically and comprehensively implemented the entire breadth of the geospatial ESDA tradition. In particular, methods to explore length scales and local spatial heterogeneity of spatial autocorrelation have not been fully explored in the single-cell genomics setting. Furthermore, the data structures underlying Seurat, squidpy, and Giotto have limited, and in some cases no support for operations on geometries such as cell segmentation polygons. In addition, packages that specialize in visualization often do not implement any ESDA methods. Many other spatial -omics data analysis packages focus on data preprocessing before the exploratory stage, such as image processing[27], cell type deconvolution of Visium spots[28], integrating data from different modalities or tissue sections[29], and predicting gene expression from H&E images[30]. Downstream analysis packages tend to implement novel methods for specific tasks such as finding spatially variable genes[31], cell-cell interactions[32], spatially informed dimension reduction[33], and finding spatial regions defined by gene expression[34], accompanied by claims of superior performance compared to existing packages performing the same tasks. These packages focus on "how *well* you have seem to *done* it" in Tukey's words while "what you *can do*" remains unaddressed.

The Voyager project fills this gap by facilitating geospatial ESDA to spatial -omics, placing the spatial information front and center in the EDA workflow. Many univariate, bivariate, and

multivariate global and local methods are included, with a consistent user interface. Some classic methods have been reimplemented to be more scalable to larger datasets, and we demonstrate tractability of ESDA by applying the Voyager tools to real data. We derive novel biological insights, which include correlation between library size and histological regions, and an example of biologically relevant negative spatial autocorrelation. We also show that ESDA can be applied to non-spatial scRNA-seq datasets via the k-nearest-neighbor graph, thereby making Voyager immediately applicable to a wide range of single-cell genomics datasets.

Finally, we address an increasingly challenging problem in single-cell genomics, namely the divergence between R and Python[1] implementations of standard methods, that has led to programming-language dependent results. Voyager is implemented in both R and Python, with compatibility tests to ensure that the two implementations give consistent results for core functionalities. Voyager also has comprehensive documentation and tutorials for common spatial -omics technologies and spatial analysis methods. While the tutorials are built on spatial transcriptomics, proteomics, and non-spatial scRNA-seq data, in principle Voyager can be applied to other -omics as well. The R packages Voyager, SpatialFeatureExperiment, and SFEData (which provides example datasets for the tutorials) are available on Bioconductor. The Python implementation is available on PyPI.

Results

The Voyager framework

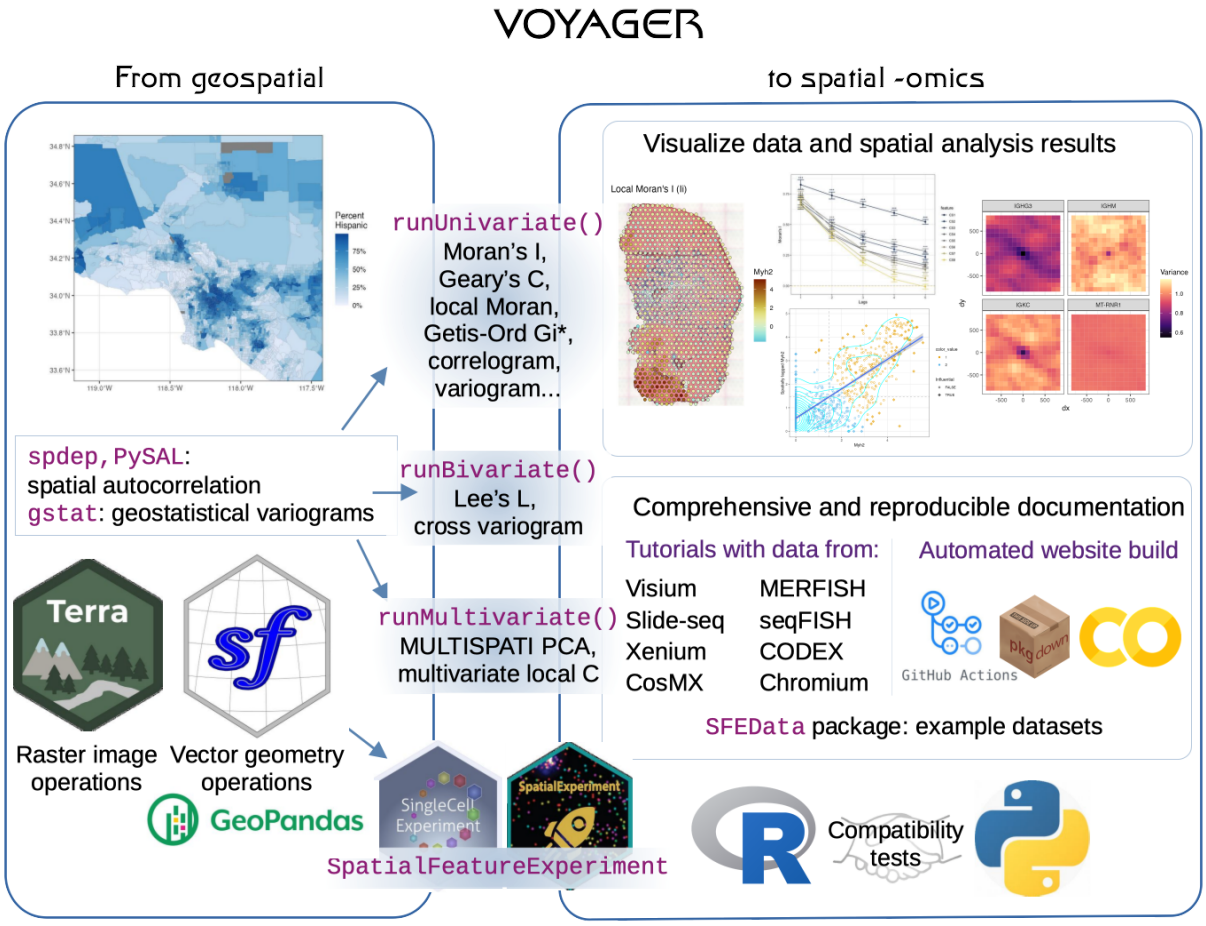

Figure 1: Schematic overview of the Voyager framework. Voyager brings exploratory spatial data analysis (ESDA) methods initially developed for geospatial data to spatial -omics, with a consistent user interface for different methods. Voyager is based on the SpatialFeatureExperiment (SFE) object. In R, SFE uses sf and terra to extend SingleCellExperiment (SCE) and SpatialExperiment (SPE). In Python, SFE extends AnnData with GeoPandas. Voyager implements plotting functions for gene expression, cell attributes, and spatial analysis results. The documentation website includes tutorials that demonstrate ESDA on data from multiple spatial -omics technologies, including Visium, Slide-seq, Xenium, CosMX, MERFISH, seqFISH, and CODEX. The website is built automatically with GitHub Actions and `pkgdown` for reproducibility, and Google Colab notebooks are automatically generated from the vignettes. Compatibility tests are used to make sure that the R and Python implementations return consistent results for core functionalities.

Voyager is built on the SFE data structure, which bundles geometries such as cell segmentation polygons with gene expression data, and implements geometric operations on such geometries

and related SFE data. Geometries in SFE are represented with Simple Features, which is a standard framework to represent geometries in the geospatial field and provides access to the GEOS C++ library for fast geometric operations. SFE extends existing scRNA-seq data structures and conforms to their syntaxes and conventions. In Python, AnnData[35] is extended with GeoPandas[36], and in R, SingleCellExperiment (SCE)[37] and SpatialExperiment (SPE)[38] are extended with sf[39] for the Simple Features (Figure 1, Supplementary Note 1). Importantly, spatial information need not be physical: single-cell RNA-seq data structures can also be analyzed spatially, where "space" is abstracted to the k-nearest-neighbor graph[12].

Spatial data analysis methods can be categorized as neighborhood or distance view; the former uses a spatial neighborhood graph to indicate spatial adjacency, while the latter uses physical distance[12]. Voyager wraps many spatial methods from widely used packages, such as spdep (neighborhood) and gstat (distance) in R, and PySAL in Python. ESDA methods can also be broadly categorized by the number of variables analyzed: univariate (e.g. Moran's I[16]), bivariate (e.g. Lee's L[25]), and multivariate (e.g. MULTISPATI PCA[26]). In Voyager, each of these has a main function providing a uniform user interface to a variety of methods, thereby simplifying access for users (Figure 1). This structure was inspired by the Tidymodels machine learning framework[40]. These methods can also be categorized as global, where one set of results is returned for the entire dataset, or local, where each location or cell has its own set of results. The latter facilitates explorations of local heterogeneity in spatial relations. Users can extend Voyager and make the uniform user interface run custom ESDA methods to reduce redundant code and facilitate organization and visualization of results.

Geospatial data tend to have a much smaller number of features and observations than modern single-cell spatial -omics datasets, so Voyager implements parallel processing when running a univariate or bivariate spatial method over a large number of genes. Voyager also reimplements methods whose current implementations don't scale to modern spatial -omics data, thereby drastically speeding up computation. These methods include MULTISPATI PCA, Lee's L, finding bounds of Moran's I from spatial neighborhood graphs[20], and distance-based edge weighting of k-nearest-neighbors or distance-based graphs (Supplementary Figures 1-2).

Visualization is an essential part of the EDA process. Voyager implements static plotting functions for gene expression, cell attributes, and cell projections along dimensions obtained by dimension reduction plotted in histological space. The histology image can be optionally plotted behind the cells, or in the case of Visium technology, the spots. While most existing packages plot cells as points, Voyager can plot cell or nuclei segmentation polygons as well. For larger datasets, users can specify a bounding box to zoom into a smaller area. The default palettes are designed to have color values discernible with color vision deficiencies (Supplementary Figure 3). Default continuous palettes come from ColorBrewer[41], scico[42], and viridis. A sequential palette is used by default, but a divergent palette is available when there is a meaningful center of divergence, such as 0 in local Moran's I and Lee's L. The default categorical palette comes from dittoSeq[43], which was designed for colorblind-friendly scRNA-seq data visualization. In addition, Voyager implements plotting functions for spatial analysis results, such as the Moran scatter plot, correlograms, variograms, and local spatial statistics shown in histological space (Figure 1).

Voyager has a comprehensive documentation website that features tutorials on applying EDA and ESDA to datasets from multiple spatial -omics technologies, including 10X Visium and Xenium[44], Nanostring CosMX[45], Vizgen MERFISH[6], Slide-seq[46], seqFISH[47], and CODEX[48] (Figure 1). On the website, each -omics technology has a landing page with an introduction to the technology and a table linking to tutorials using a dataset from the technology. Each ESDA method has a similar landing page, with an introduction to the method and a similar table, linking to sections in each tutorial using the ESDA method, some of which include further considerations on the ESDA methods and references to the geospatial ESDA literature. Example datasets used in the tutorials are available as SFE objects in the SFEData package. In addition, there are tutorials instructing users on constructing an SFE object and extending Voyager for custom ESDA methods.

While Voyager is focused on spatial data, neighborhood view ESDA methods can be applied to the k-nearest-neighbor graph in gene expression PCA space. This is illustrated in depth for a 10X Chromium and a Visium dataset. Basic tutorials also introduce analysis for Split-seq, ATAC-seq, 10X single cell multiome ATAC + gene expression, ClickTags, 10X single nuclei, and 10X single cell CRISPR screen performing data preprocessing and quality control.

To ensure that the tutorials are reproducible, Voyager builds the documentation website on GitHub Actions, rendering all the tutorials on the cloud. Because the GitHub Actions runner has fewer computational resources than a typical modern laptop, this ensures that the vignettes are scalable to larger datasets, such as a MERFISH mouse liver dataset with almost 400,000 cells. The tutorials are also automatically converted into Google Colab notebooks to be run interactively, allowing users to experiment with different parameters, and facilitating exploration of new datasets, and customization of workflows.

In order to ensure that analysis results are not language dependent, as is the case currently in single-cell and spatial -omics data analysis where results differ depending on whether analysis was performed in R or Python, we have developed a Python implementation of core Voyager functionalities. The implementation, called VoyagerPy, is equivalent to the R implementation of Voyager, as ensured via compatibility tests. Because Bioconductor requires software packages to have unit tests and pass an automated check run daily, whereas PyPI and conda do not perform automated tests, the Python implementation is thus indirectly held to the Bioconductor standard. VoyagerPy has the added advantage of facilitating the utilization of deep learning and image analysis methods for spatial -omics, as there is better support for such methods in Python than R. The Python implementation also supports visualization in Matplotlib.

ESDA case studies

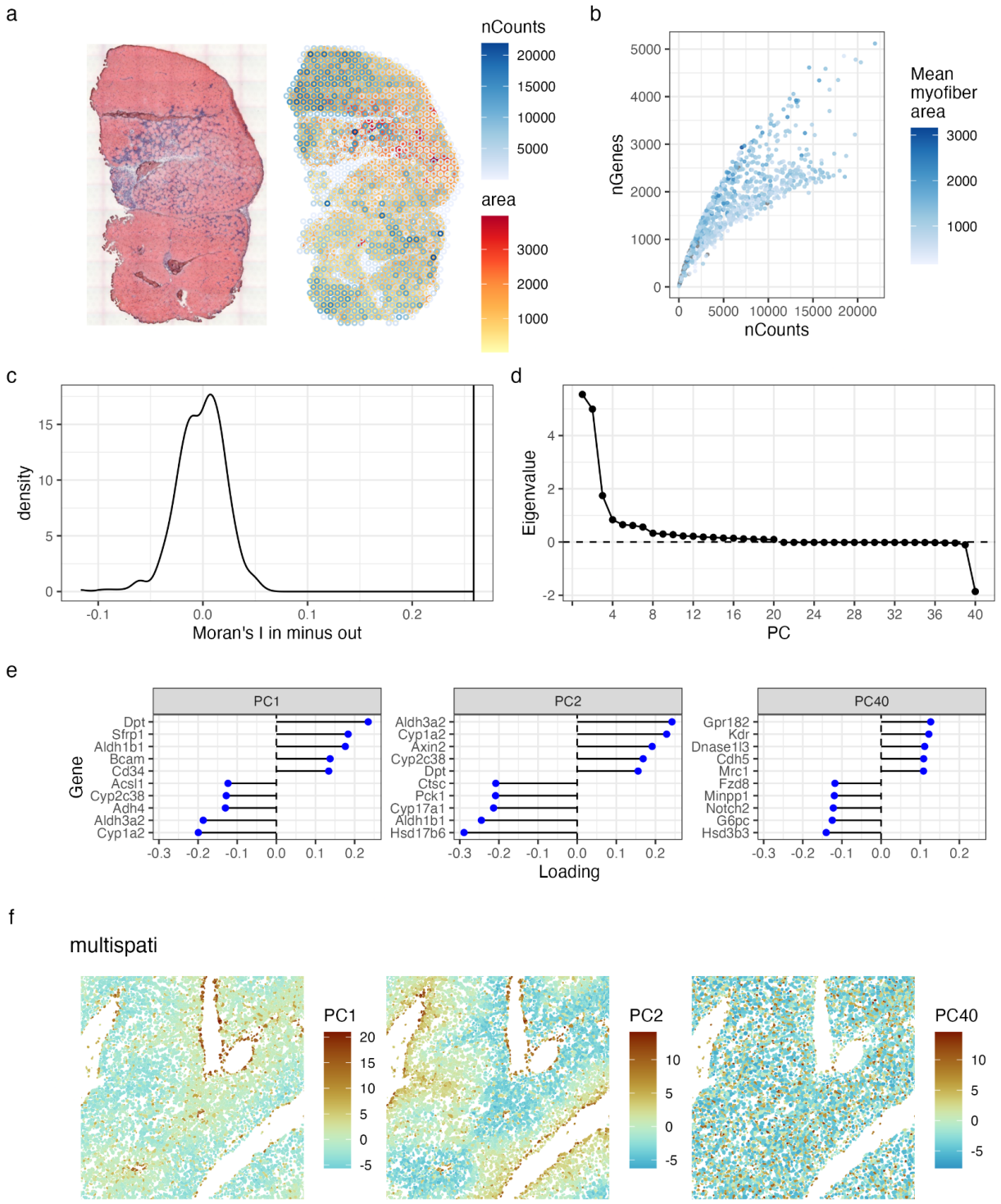

Figure 2: Applications of Voyager on spatial transcriptomics datasets. A) In a mouse skeletal muscle dataset, the total UMI counts, or library size per spot (nCounts), are plotted in space as

blue open circles and myofibers are colored in red according to their cross section areas. Only spots that intersect tissue are plotted. The H&E image is plotted on the side as a reference. B) Scatter plot of the number of genes detected per spot (nGenes) vs. nCounts, colored by mean area of myofibers that intersect each spot. C) Simulated (density plot) and observed (vertical line) difference between Moran's I in nCounts of spots that intersect tissue (in) and that of spots that don't (out). D) The 20 most positive and 20 most negative eigenvalues from MULTISPATI PCA of a mouse liver MERFISH dataset. As other eigenvalues were not computed, there is a break after PC20 in this plot. E) The most positive and negative gene loadings for PCs 1, 2, and 40. F) A subset of the MERFISH data showing a portal triad (near top right) and two central veins (left and bottom right), with cell polygons colored by their projections into 2 PCs with the most positive eigenvalues and the PC with the most negative eigenvalue ("PC40"). The first 2 PCs show zonation.

To showcase the power of ESDA for spatial transcriptomics, and to highlight the potential insights that can be gleaned from such analysis, we examined in detail a mouse skeletal muscle Visium dataset[49], two days after notexin injury (Figure 2A-C). In the H&E image, the region with many leukocyte nuclei is the injury site, the darker red strip and blocks are muscle-tendon junctions, and the remaining pink regions are myofibers (Figure 2A). When the library size (nCounts) per spot among spots that intersect the tissue is plotted in space, we find that different histological regions have different library sizes. For example, the muscle-tendon junctions tend to have smaller library sizes than myofibers and part of the injury site, and regions with tightly packed myofibers (top and bottom left) tend to have larger library sizes than the region with larger myofibers surrounded by leukocytes (right). In this dataset, the library size and number of genes detected (nGenes), often used as QC metrics, are related to myofiber size – spots on larger myofibers tend to have more genes detected given the same library size than spots on smaller myofibers (Figure 2A-B). This confirms the finding that in spatial transcriptomics, library size confounds biology and should not be treated as a technical artifact as is commonly done for scRNA-seq [50].

Moreover, the library size in the tissue has stronger spatial autocorrelation than outside the tissue, as indicated by a larger positive Moran's I (Figure 2C, Supplementary Note 2). The library size values are permuted in space for spots that intersect the tissue and those that don't, and Moran's I is computed for these permutations to estimate a null distribution. The density plot in Figure 2C shows the null distribution of permuted Moran's I from spots intersecting tissue minus that from spots not intersecting tissue. The vertical line indicates the actual difference, which is much larger than all 499 simulated values. While there is no one-to-one correspondence between Visium spots and myofibers, geometric operations can find the myofibers that intersect each Visium spot and their areas using myofiber segmentation.

The ESDA method MULTISPATI PCA is particularly interesting, as we see from analysis of a mouse liver MERFISH dataset from the Vizgen website[51] (Figure 2D-F, Supplementary Note 2). While non-spatial PCA maximizes variance explained by each principal component (PC) given that the PCs are orthogonal, MULTISPATI PCA maximizes the product of variance explained and Moran's I, which is the eigenvalues (Figure 2D). Positive eigenvalues mean that the PCs not only explain more variance, but also are spatially coherent (large positive Moran's I).

Negative eigenvalues mean that the PCs not only explain more variance, which is non-negative, but also have negative spatial autocorrelation, i.e. nearby values tend to be more different. In this dataset, the positive eigenvalues show an elbow as in non-spatial PCA, and there is one substantial negative eigenvalue (Figure 2D). In non-spatial PCA, the PCs are not spatially structured until PC5, which picks up zoning (Supplementary Figure 4). PC1 highlights Kupffer cells (Cdh5) and endothelial cells (Egfr), and PC2 also highlights endothelial cells (Supplementary Figure 4A). In contrast, because MULTISPATI PCA also maximizes Moran's I, zonation is picked up by the first 2 PCs. PC1 is periportal and PC2 is pericentral (Figure 2E-F). This may complement existing methods to find spatially variable genes (maximize Moran's I) that are also more likely to be biologically relevant (maximize variance explained). Furthermore, spatially coherent PCs can be used for clustering to find more spatially coherent clusters in some cases, complementing clusters found with non-spatial PCs (Supplementary Figures 5-6, Supplementary Note 2).

Negative spatial autocorrelation is one of the most neglected topics in spatial data analysis[52], as there are many more examples of positive than negative spatial autocorrelation. Negative spatial autocorrelation can arise from competition between neighbors (see[52]) or from functional roles played by spatial contacts between different types of entities. In this dataset, the latter seems to be the case: the PC with the most negative eigenvalue separates endothelial cells (Kdr) and Kupffer cells (Cdh5) from hepatocytes (Hsd3b3, Figure 2E-F). Existing methods of spatially informed dimension reduction[53–55] and methods to find spatially variable genes[56–59] tend to only consider positive spatial autocorrelation. This example shows that negative spatial autocorrelation is particularly relevant to biology at the single-cell level and should be further investigated, as the cells, unlike Visium spots and administrative boundaries, are meaningful and non-arbitrary units of observations and biological function.

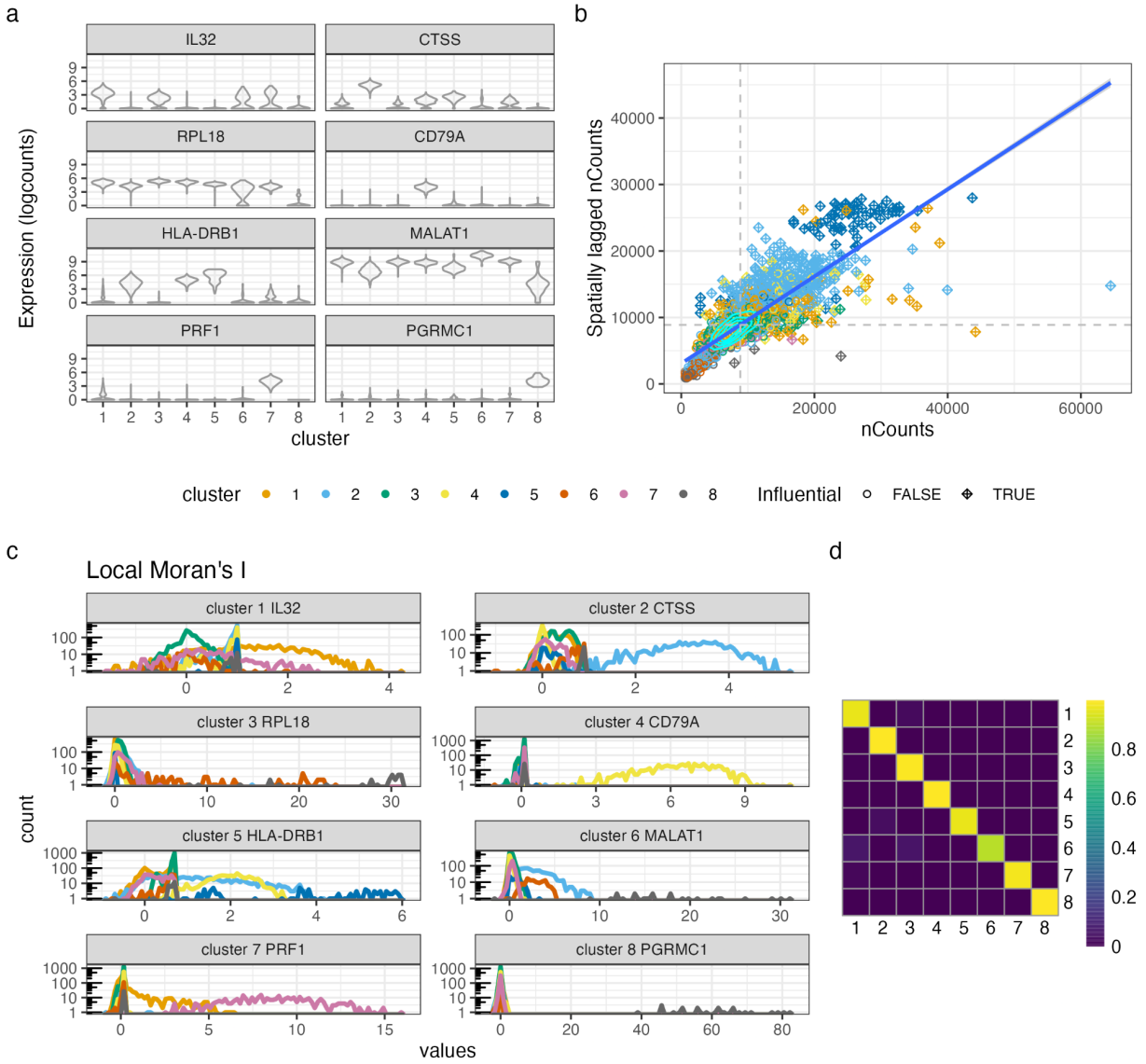

Figure 3: Application of neighborhood view spatial statistics on non-spatial scRNA-seq. A) Violin plots of log normalized counts of the top marker gene of each Leiden cluster in the PBMC dataset. B) Moran scatter plot of nCounts in a 10X Chromium human PBMC dataset. The spatial lags were computed with the k nearest neighbors graph in PCA gene expression space. The blue line shows the least square fit to the scatter plot. The gray shade around the line is the 95% confidence interval of the fit. Contours show the area with the highest point density. The gray dotted lines show the mean on the x and y axes. C) Histograms of local Moran's I values per cell of top marker genes of each cluster in the PBMC dataset, colored by cell cluster. The y axis (number of cells per bin) is log-transformed for better dynamic range. The histograms are plotted as lines instead of bars to avoid overlapping bars from different clusters. D) Concordex heatmap for the PBMC Leiden clusters. High diagonal and low off diagonal values indicate high clustering quality, or that the Leiden clusters reflect the k-nearest-neighbor graph well, but cluster 6 has somewhat lower quality.

Finally, we note that neighborhood view spatial methods can be applied to neighborhood graphs in gene expression space rather than histological or geographical space. We applied spatial statistics to quality control metrics and gene expression in a 10X Genomics Chromium peripheral blood mononuclear cells (PBMC) dataset[60] using the k-nearest-neighbors graph in PCA gene expression space as the "spatial" neighborhood graph (Figure 3). Moran scatter plot of library size shows further evidence that library size is confounded by biology even in non-spatial data (Figure 3B). In a Moran scatter plot, the x-axis is the value of a variable at each cell, and the y-axis is the spatially lagged value (i.e. sum of values from spatial neighbors weighted by edge weights of the spatial neighborhood graph). When the adjacency matrix of the spatial neighborhood graph is row normalized, it is shown in Anselin 1996[12] that the slope of the line fitted to the scatter plot is global Moran's I, while the scatter plot shows local heterogeneity in spatial autocorrelation. Here, cluster 5 (activated T cells) tends to have larger library sizes and stronger "spatial" autocorrelation in library size than average (Figure 3B).

Local Moran's I is a locally disaggregated form of Moran's I, that measures the contribution of each cell to Moran's I[23] (Supplementary Note 2). Positive values indicate neighborhoods homogeneous in the variable of interest, and negative values indicate heterogeneous neighborhoods. We computed local Moran's I (Ii) for the top marker gene of each cluster (Methods). The marker gene has much higher Ii in cells in the cluster of interest than cells in other clusters, except for clusters 3 and 6, whose top marker genes don't clearly distinguish these clusters from most other clusters and don't seem to have cell type-specific functions (Figure 3A, C). When the marker gene is highly specific, cells in other clusters display an Ii tightly clustered around 0, as shown for clusters 4 (B cells) and 8 (platelets). When the marker gene is not very specific, cells in other clusters that also express the marker gene often display an Ii higher than clusters not expressing the gene (e.g. cluster 1 T cells and cluster 7 natural killer cells and cytotoxic T cells for IL32 and PRF1; cluster 5 activated T cells, cluster 2 monocytes, and cluster 4 B cells for HLA-DRB1; though this does not apply to CTSS, a marker gene of cluster 2 which is also somewhat expressed in clusters 4 and 5, see Figure 3A, C). We also see a wide range of Ii values for cells in the clusters of interest (Figure 3C). The Concordex metric was devised to quantify k-nearest-neighbor graph based cluster quality as an alternative to UMAP, by indicating how well the clusters match the graph structure [61]. As Leiden clustering is graph-based, Ii on cluster marker genes can give nuances to the single numbers from Concordex and to the quality of the marker genes (Figure 3D). The downside is that Ii can be computed on only one feature at a time, while a combination of multiple genes may better characterize the clusters.

Compatibility tests

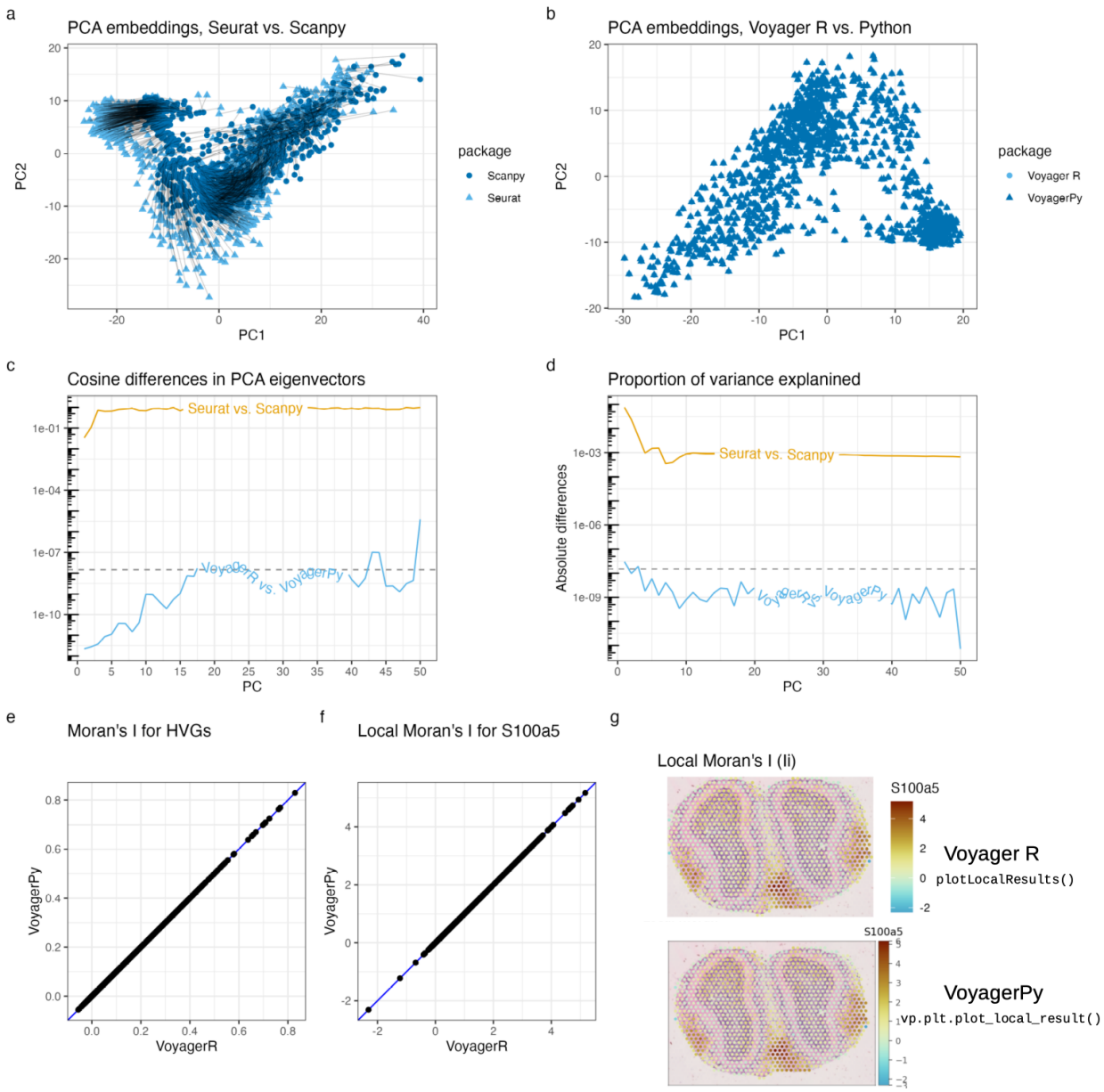

Figure 4: Comparisons between results obtained by Seurat and scanpy, and between Voyager R and Python for a mouse olfactory bulb Visium dataset. A) Comparison of Visium spot embeddings in the first 2 PCs from Seurat and scanpy with default parameters. The lines connect corresponding spots in Seurat and scanpy. B) As in A, but for Voyager R and VoyagerPy, with parameters stated in this section. C) Cosine distances between the first 20 PCA eigenvectors (gene loadings) from Seurat and scanpy (yellow), and from Voyager R and Python (blue). The dashed line is the magnitude that can be explained by machine double

precision. The text part of the line is somewhat smoothed for readability but should not affect interpretation. D) Absolute values of differences in the proportion of variance explained by each of the top 20 PCs. E) Moran's I from VoyagerPy vs. Voyager R. The blue line is y = x, showing that the results are consistent. F) Same as E but for local Moran's I for gene S100a5. G) Plotting the local Moran's I values in space, with the H&E image behind the spots, from Voyager R (top) and VoyagerPy (bottom).

In scRNA-seq, Seurat and scanpy are both commonly used EDA frameworks. However, their default settings not only yield different log fold changes[62] but also highly divergent PCA results, as exemplified using a mouse olfactory bulb Visium dataset[63] (Figure 4A, C-D), which might lead to different biological conclusions. To avoid such discrepancies, Voyager implements extensive compatibility tests to ensure that the R and Python implementations return the same results for core functionalities. Thus, there are no visible differences in Visium spot embeddings in the first two PCs returned by the Voyager R and Python implementations (Figure 4B). In Seurat vs. scanpy, the cosine difference (see Methods) between PCA eigenvectors (gene loadings) in each of the top 50 PCs is nearly 1 after the first few PCs, which means the angles between the corresponding eigenvectors are nearly 90 degrees (Figure 4C). In Voyager, this difference is much smaller, well below machine epsilon (dashed line, around 1.5e-8, see Methods) until PC20 (Figure 4C). While the difference sometimes rises above machine epsilon after PC20, it does not exceed 1e-5. While Seurat and scanpy implementations of PCA with default parameters produce sizable differences in the proportion of variance explained by each PC, these differences in the R and Python implementations of Voyager are generally within machine epsilon (Figure 4D).

For spatial statistics, the R and Python implementations of Voyager also produce consistent results for global Moran's I of the highly variable genes (Figure 4E), with differences within epsilon (1.61e-17, see Methods). For local Moran's I (Figure 4F-G), the results are the same (within epsilon, 3.89e-16) but with a non-default spdep parameter (see list below). In both implementations, besides the different default styles of `ggplot2` and `matplotlib`, the plotting functions also produce visually similar plots with the same palettes (Figure 4G).

While the R and Python implementations of Voyager may eventually diverge in functionalities, as the two languages have better support for different types of analyses, the compatibility tests will continue to ensure that the core functionalities and basic vignettes return the same results in R and Python, so language preference does not inadvertently lead to different interpretations of data.

Notably, Seurat and scanpy produce different PCA results largely due to their different methods of finding highly variable genes. However, the PCA results remain somewhat different even when using the same set of highly variable genes because of a hidden default in Seurat that clips scaled data to 10, while scanpy by default does not clip the scaled data (Supplementary Figure 7). Moreover, Seurat and scanpy return different marker gene rankings from ostensibly the same differential expression method (e.g. t-test or Wilcoxon test) because they rank the genes differently and even estimate different log fold changes[62]. Most users may be unaware of such inconsistencies and the hidden parameter defaults and method choices that cause them.

In the interests of transparency, we have therefore documented Voyager's default parameters and the reasons for choosing them (Supplementary Note 3).

At present, Voyager implements compatibility testing for two tutorials: a mouse olfactory bulb Visium dataset from the 10X website, and a univariate spatial statistics analysis of the k-nearest neighbor-graph of a human peripheral blood mononuclear cells (PBMC) 10X Genomics Chromium dataset. The defaults used in the basic vignettes covered by the compatibility tests are listed in Supplementary Note 3; some of the defaults come from conventions and defaults in established packages and hence are subject to further research and improvement.

Discussion

Software packages for data visualization, general EDA frameworks, and more specialized tasks have proliferated in the field of spatial -omics, but much of the ESDA tradition has not been utilized. As an EDA framework, Voyager is similar to software packages such as Seurat, squidpy, Giotto, and semla, but Voyager is unique in systematically facilitating the porting of decades of ESDA research to spatial -omics, with a consistent user interface. By reusing the SCE and AnnData infrastructure and ecosystems, Voyager complements many other spatial and non-spatial data analysis methods. The SFE class extends SCE and AnnData with efficient tools from the geospatial field to represent and operate on vector geometries and raster images. Voyager has a comprehensive, reproducible, and easy-to-navigate documentation website with tutorials on data from various technologies and ESDA methods, with references for further reading and considerations from the ESDA tradition. Extensive compatibility tests ensure that the R and Python implementations of Voyager return consistent results for core functionalities and transparency on defaults. The Voyager project also bridges the R vs. Python divide in single-cell and spatial -omics bioinformatics, where hidden defaults and undocumented divergent implementations cause language preference to inadvertently lead to different results that may affect biological conclusions. Thus, even for EDA, it provides a substantial advantage over current packages.

The scholar Jaroslav Pelikan wrote that "Tradition is the living faith of the dead, traditionalism is the dead faith of the living."[64] While the ESDA tradition was largely developed prior to the rise of spatial -omics, it can help us gain insights by taking the spatial aspects of biological data into consideration. As the ESDA tradition is ever evolving, Voyager is designed to be extensible by users and developers to facilitate the application of novel methods with a consistent user interface. Future versions of Voyager can also take into account the peculiarities of spatial -omics data, such as larger dataset sizes, case and control comparisons, multiple biological replicas, non-normal distribution of the data, and 3-dimensional data from thick slices and multiple sections. Additional future directions include storing geometries and spatial results on-disk. While the SCE infrastructure already allows for on-disk gene count matrices with DelayedArray, the geometries and spatial analysis results are currently stored in memory. Moreover, while we extensively documented the Voyager defaults to avoid inconsistencies between the R and Python implementations, the reasoning behind them is often based on convention in the field. Further research should scrutinize the effects of changing these parameters, such as the type of spatial neighborhood graph and edge weights. The problem of

choosing a spatial neighborhood graph has long been studied, and some methods to find a graph based on the data have been devised[65], but they may or may not be suitable for spatial -omics data. Finally, while we have chosen colorblind-friendly default palettes to make Voyager more accessible, future research should be conducted on the accessibility of spatial -omics data analysis, such as in data sonification.

Methods

All R plots in the figures were made with R 4.3.0 with Apple vecLib BLAS, Bioconductor 3.17, Voyager 1.2.4, SpatialFeatureExperiment 1.2.1, scater 1.28.0, spdep 1.2.8, Seurat 4.3.0, sf 1.0.12, and ggplot2 3.4.2, on MacOS Ventura 13.3.1, 2.3 GHz Dual-Core Intel Core i5, 8 GB RAM. R package profvis 0.3.7 was used to profile time and memory usage by lines of code in the benchmarks, and bench 1.1.2 was used for the benchmarks over different numbers of cells. When comparing Seurat vs. scanpy and the R and Python implementations of Voyager, the Python code was run through reticulate (v1.28) in RStudio. Python 3.10 and scanpy 1.9.3 were used. Multipanel plots were assembled with patchwork 1.1.2 when all panels are R plots, and were otherwise assembled in LibreOffice Draw.

Spatial methods

At present, all neighborhood view spatial methods are implemented in spdep and wrapped by Voyager, except for Lee's L, which has a more efficient implementation in Voyager. Defaults follow those of spdep. All distance view spatial methods are implemented in gstat and wrapped by Voyager. Variogram model fitting is implemented in automap, which is a user-friendly wrapper of gstat that tries a number of different models and selects the one with the best fit. Voyager features its own, more efficient implementation of MULTISPATI PCA, and hence does not depend on adespatial which originally implemented MULTISPATI PCA.

In contrast to the plotting functions in spdep and gstat, Voyager plotting functions are based on ggplot2[66] to be more visually appealing and customizable by users and are designed to visualize results from multiple features and tissue sections at once.

Compatibility tests

Everything in the two core tutorials other than the plots themselves is subject to compatibility tests to verify that the R and Python implementations of Voyager produce the same results for core functionalities, i.e. procedures that yield numeric output, such as PCA and Moran's I. The plots cannot be quantitatively and automatically compared because of the different default styles and mechanisms of ggplot2 and matplotlib, so the comparison is performed based on visual similarity. The "epsilon", or numeric differences that can be accounted for by machine double precision, was established as `sqrt(.Machine$double.eps)` in R. To compare PCA eigenvectors (gene loadings), cosine difference is used to geometrically compare the vectors. This is measured as the magnitude of difference between the cosine of the angle between the two vectors and 1, i.e. cosine of 0 and 180 degrees. The value 180 degrees was chosen because the eigenvectors can be flipped and yet produce results equivalent to PCA. This

comparison was performed on each of the first 50 PCs individually for Figure 3. To compare the proportion of variance explained, the absolute value of the difference was used.

Website build

The R Voyager documentation website is built with pkgdown on GitHub Actions, which builds function references and vignettes from the R package source code. All imported and suggested packages are installed on a fresh machine on the cloud and all vignettes are run on the cloud to be rendered, to ensure that they are reproducible. The Google Colab notebooks are automatically generated from the R Markdown vignettes with another GitHub Action. Because Bioconductor limits the installed size of the package, which includes the rendered vignettes, the vignettes on the documentation website are in a separate documentation branch from the main and devel branches that sync with Bioconductor, while a shorter vignette is on Bioconductor. Also, there are packages suggested in the documentation branch but not the main branch, as while they are used in vignettes on the website, they are not used on the Bioconductor vignette. The code in the documentation branch is synchronized with code from the main branch by merging from the main branch, but the documentation branch is never merged into the main branch.

The VoyagerPy website is built with sphinx and readthedocs.

Performance improvements

In the benchmarks, a mouse liver MERFISH dataset from the Vizgen website with over 390,000 cells after QC was used. After removing cells with a high proportion of transcripts from blank barcodes (and removing the blank barcodes themselves), the dataset was subsetted with bounding boxes of different sizes to produce datasets of different sizes while preserving spatial relationships among cells. The datasets of different sizes were used in the benchmarks.

K -nearest-neighbors with inverse distance weighting

The spdep implementation of distance-based edge weights is slow because while spdep uses an efficient implementation of k nearest neighbors and distance neighbors in dbscan, it discards the distances between neighbors returned by dbscan. As a result, spdep has to re-compute the distances to compute the edge weights (Supplementary Figure 2A). The implementation in SFE uses BiocNeighbors to find the k nearest and distance-based neighbors, allowing users to choose from a number of different algorithms. Then the distances are saved for edge weight computations, skipping the most time-consuming step. While dbscan is not much slower than BiocNeighbors when finding the neighbors (Supplementary Figure 2A-B), we found that not recomputing the distances speeds up finding the spatial neighborhood graph from 8 to over 30 times and uses over 25 times less memory (Supplementary Figure 1A-B).

MULTISPATI PCA

The adespatial implementation of MULTISPATI PCA uses base R eigendecomposition, which always computes all eigenvalues and eigenvectors. Subsequently, adespatial discards the

eigenvectors when the user specifies a small number of eigenvectors. Furthermore, adespatial in fact performs the eigendecomposition twice. The first time is in dudi.pca, which performs non-spatial PCA, whose results are passed to the multispati function, which takes some weights and the original data but not the eigenvalues or eigenvectors from the dudi.pca output, and then performs eigen decomposition of the spatially weighted covariance matrix (Supplementary Figure 2C). The implementation in Voyager uses RSpectra for partial eigendecomposition of the spatially weighted covariance matrix, only for the eigenvectors the user requested and only once, hence avoiding a lot of unnecessary computation, speeding up computation by 2 to 20 folds, and using over 5 times less memory (Supplementary Figure 1C-D).

Lee's L

The spdep implementation of Lee's L computes both local and global Lee's L for one pair of genes at a time. As spatial transcriptomics data has hundreds (smFISH based) to thousands of genes (sequencing based), Voyager's implementation uses matrix operations to make it more efficient to compute Lee's L for a large number of genes than iterating through each pair. This speeds up computation over 800 fold and uses 100 times less memory, where one thread was used to iterate through all pairs of genes in the dataset for the spdep implementation (Supplementary Figure 1E-F). The inefficiency of spdep's implementation is not only due to iterating through the pairwise combinations, but also because of a less efficient computation of the spatially lagged values and the sum of edge weights (Supplementary Figure 2E). Iterating through the pairs with the spdep implementation is so slow that it was only run for the smallest subset with 105 cells in the benchmark.

Moran bounds

Bounds of Moran's I given a spatial neighborhood graph can be computed from the largest and smallest eigenvalues of the double-centered adjacency matrix of the graph[20]. The adespatial implementation of the function finding Moran bounds, all eigenvalues are computed. In Voyager's implementation, RSpectra is used to find only the largest and smallest eigenvalues of the matrix without computing eigenvectors or the other irrelevant eigenvalues, speeding up computation by 4 folds and using 4 times less memory (Supplementary Figure 1G-H). While much of the time was spent on the eigendecomposition in the adespatial implementation, most of the time was spent on double centering in the Voyager implementation (Supplementary Figure 2G-H). Due to double centering, a dense matrix with as many columns and rows as the number of cells is produced. Unless this can be avoided, this computation consumes a lot of memory for larger datasets. As a result, neither implementation scaled beyond around 6000 cells.

ESDA case studies

Mouse skeletal muscle Visium data

Space Ranger processed data were downloaded from GEO accession GSE161318, sample Vis5A, 2 days after notexin injury. Myofiber segmentation was performed manually with the LabKit ImageJ plugin, exported as TIFF raster, and converted to polygons with terra. Redundant

vertices of the polygons were removed when the polygons were simplified with rmapshaper (v0.4.5). Visium spot polygons were found from the centroids and spot diameters in pixels in the full-resolution image from the Loupe image alignment JSON file. The tissue was segmented by thresholding the H&E image and removing small pieces. Then, the thresholded mask was converted into polygon with terra, which was used to find spots intersecting the tissue with sf. The gene count matrix and polygons were made into an SFE object available to download from the SFEData package. The sf package was used to find myofiber areas and to determine which myofibers intersected each Visium spot polygon. Only the full-resolution H&E images were available on GEO; to facilitate reproducibility of the vignette and examples, the image was downsampled to fit into a 1024x1024 pixel box. Moran's I permutation tests were performed on nCounts for spots intersecting the tissue and spots not intersecting the tissue separately, with 499 permutations. The values are permuted in space. Subsequently, the simulated Moran's I's from spots not intersecting tissue were subtracted from those from spots intersecting tissue to form a null distribution of this difference. The "observed" value was the observed Moran's I from spots not intersecting tissue subtracted from that of spots intersecting tissue. Leiden clustering (Supplementary Figure 5) was performed on the first 30 non-spatial and MULTISPATI PCs, with resolution 1 and objective function "modularity" in the R package bluster v1.10.0. Neighbor purity was computed with the neighborPurity() function in bluster. Concordex was computed with the concordexR package v1.0.0. Neighbor purity and Concordex were computed with the k-nearest-neighbor graph (k=6 because of the hexagonal grid) in histological space, although the graph in PCA space was used to generate the clusters. This way the cluster quality indices displayed spatial coherence of the clusters.

Mouse liver MERFISH data

The gene count matrix, cell metadata, and cell segmentation polygons were downloaded from Vizgen's website, and parsed into an SFE object, which is available in the SFEData package. The scuttle package (1.10.0) was used to remove low-quality cells. The proportion of transcripts attributed to blank barcodes was computed, log2 transformed, and cells with log proportion more than 3 median absolute deviations (MADs) higher than the median were deemed low quality and removed. Subsequently, the filtered gene count matrix was normalized by logNormCounts() in scater, and the genes were scaled and centered before performing non-spatial PCA with IRLBA through scater, and MULTISPATI PCA. MULTISPATI PCA requires a spatial neighborhood graph; see the Compatibility Tests section for reasons behind the parameters chosen. Leiden clustering, neighbor purity, and the Concordex index were computed as in the mouse skeletal muscle Visium dataset, except that the first 20 PCs were used with k=5 (the same as in spatial neighborhood graph) in the k-nearest-neighbor graph in histological space.

Chromium PBMC data

The filtered 5k PBMC NextGem v3 data, processed with Cell Ranger 3.0.2, was downloaded from the 10X Genomics website and loaded into R as an SCE object, and was then converted to SFE for "spatial" analyses. Cells with at least 20% of UMIs from mitochondrially encoded genes were removed. Highly variable genes (HVGs) were found with the scran method but

without the Lowess fit. The 2000 genes with the highest biological component were used for PCA. The data was normalized with logNormCounts() in scater, and the genes were scaled and centered before performing PCA with the IRLBA algorithm. Based on the variance explained elbow plot, the 10 PCs were used to build a k nearest neighbor graph, with k = 10 (not including self). For Leiden clustering, k = 10 was also used so the clustering results can be compared to the "spatial" results. The objective function selected was "modularity" and the resolution parameter was 0.5. For the "spatial" neighborhood graph, inverse distance weighting and W style normalization were used, for reasons similar to those in k-nearest-neighbor graphs in histological space explained in the Compatibility Tests section. Differential expression was performed as described in the Compatibility Tests section.

Data and code availability

The mouse skeletal muscle Visium dataset and the mouse liver MERFISH datasets are available in the SFEData R package. These are the GitHub repositories and websites related to this paper:

Voyager R package: https://github.com/pachterlab/voyager

SpatialFeatureExperiment: https://github.com/pachterlab/SpatialFeatureExperiment

SFEData: https://github.com/pachterlab/SFEData

Voyager Python package: https://github.com/pmelsted/voyagerpy

Voyager R documentation website: https://pachterlab.github.io/voyager/

Voyager Python documentation website: https://voyagerpy.readthedocs.io/

Code to reproduce figures in this paper: https://github.com/pachterlab/MJLEBAMP_2023

Acknowledgement

We thank SpatialExperiment author Dario Righelli for suggestions in the early development of SFE.

Author contributions

L.M. conceived the main ideas, implemented the R packages SpatialFeatureExperiment, Voyager, and SFEData, wrote most of the R Voyager tutorials, wrote the spatial methods landing pages on the R website, built the R documentation website, and drafted the manuscript. S.A. implemented the foundation for VoyagerPy. P.H.E. implemented VoyagerPy, built the VoyagerPy documentation website, and implemented compatibility tests for the two core tutorials. K.J. wrote some of the R Voyager tutorials and technology landing pages. L.L. implemented the GitHub actions to automatically generate Google Colab notebooks from the R Markdown tutorials, added gget examples in some tutorials, and edited the manuscript. A.S.B. wrote some of the VoyagerPy tutorials, implemented notebooks demonstrating pre-processing

of raw data, and edited the manuscript. P.M. conceived the ideas for VoyagerPy. L.P. conceived the ideas of compatibility tests, applying neighborhood view spatial methods to the k-nearest-neighbor graph in PCA space for non-spatial scRNA-seq data, developed the concepts for the landing pages for data collection technologies and spatial data analysis methods, and edited the manuscript. All authors made suggestions to the manuscript.

References

1. Moses, L. & Pachter, L. Publisher Correction: Museum of spatial transcriptomics. *Nat. Methods* **19**, 628 (2022).
2. Hao, Y. *et al.* Integrated analysis of multimodal single-cell data. *Cell* **184**, 3573–3587.e29 (2021).
3. Palla, G. *et al.* Squidpy: a scalable framework for spatial omics analysis. *Nat. Methods* **19**, 171–178 (2022).
4. Dries, R. *et al.* Giotto: a toolbox for integrative analysis and visualization of spatial expression data. *Genome Biol.* **22**, 78 (2021).
5. Bergenstråhle, J., Larsson, L. & Lundeberg, J. Seamless integration of image and molecular analysis for spatial transcriptomics workflows. *BMC Genomics* **21**, 482 (2020).
6. Moffitt, J. R. *et al.* High-throughput single-cell gene-expression profiling with multiplexed error-robust fluorescence in situ hybridization. *Proc. Natl. Acad. Sci. U. S. A.* **113**, 11046–11051 (2016).
7. Pechuan-Jorge, X. *et al.* SPEX: A modular end-to-end analytics tool for spatially resolved omics of tissues. *bioRxiv* 2022.08.22.504841 (2022) doi:10.1101/2022.08.22.504841.
8. Behanova, A. *et al.* Visualization and quality control tools for large-scale multiplex tissue analysis in TissUUmaps3. *Biological Imaging* **3**, e6 (2023).
9. Preibisch, S., Karaiskos, N. & Rajewsky, N. Image-based representation of massive spatial transcriptomics datasets. *bioRxiv* 2021.12.07.471629 (2022) doi:10.1101/2021.12.07.471629.

10. Sriworarat, C. *et al.* Performant web-based interactive visualization tool for spatially-resolved transcriptomics experiments. *bioRxiv* (2023) doi:10.1101/2023.01.28.525943.
11. Bienroth, D. *et al.* Spatially Resolved Transcriptomics Mining in 3D and Virtual Reality Environments with VR-Omics. *bioRxiv* 2023.03.31.535025 (2023) doi:10.1101/2023.03.31.535025.
12. Anselin, L. The Moran scatterplot as an ESDA tool to assess local instability in spatial association. in *Spatial analytical perspectives on GIS* 111–126 (Routledge, 1996).
13. Wickham, H. & Grolemund, G. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. ("O'Reilly Media, Inc.," 2016).
14. Tukey, J. W. *Exploratory Data Analysis*. (Addison-Wesley Publishing Company, 1977).
15. Getis, A. A history of the concept of spatial autocorrelation: A geographer's perspective. *Geogr. Anal.* **40**, 297–309 (2008).
16. Moran, P. A. P. Notes on continuous stochastic phenomena. *Biometrika* **37**, 17–23 (1950).
17. Geary, R. C. The Contiguity Ratio and Statistical Mapping. *The Incorporated Statistician* **5**, 115–146 (1954).
18. Griffith, D. A. & Chun, Y. Some useful details about the Moran coefficient, the Geary ratio, and the join count indices of spatial autocorrelation. *Journal of Spatial Econometrics* **3**, 12 (2022).
19. Griffith, D. A. The Moran coefficient for non-normal data. *J. Stat. Plan. Inference* **140**, 2980–2990 (2010).
20. de Jong, P., Sprenger, C. & van Veen, F. On extreme values of Moran's I and Geary's c. *Geogr. Anal.* **16**, 17–24 (1984).
21. Cliff, A. D. & Ord, J. K. *Spatial Processes: Models & Applications*. (Pion, 1981).
22. Cressie, N. *Statistics for Spatial Data*. (Wiley, 1993).
23. Anselin, L. Local Indicators of Spatial Association—LISA. *Geogr. Anal.* **27**, 93–115 (1995).

24. Ord, J. K. & Getis, A. Local Spatial Autocorrelation Statistics: Distributional Issues and an Application. *Geogr. Anal.* **27**, 286–306 (1995).
25. Lee, S.-I. Developing a bivariate spatial association measure: An integration of Pearson's r and Moran's I. *J. Geogr. Syst.* **3**, 369–385 (2001).
26. Dray, S., Saïd, S. & Débias, F. Spatial ordination of vegetation data using a generalization of Wartenberg's multivariate spatial correlation. *J. Veg. Sci.* **19**, 45–56 (2008).
27. Chen, H., Li, D. & Bar-Joseph, Z. SCS: cell segmentation for high-resolution spatial transcriptomics. *Nat. Methods* (2023) doi:10.1038/s41592-023-01939-3.
28. Ding, J. *et al.* SpatialCTD: a large-scale TME spatial transcriptomic dataset to evaluate cell type deconvolution for immuno-oncology. *bioRxiv* 2023.04.11.536333 (2023) doi:10.1101/2023.04.11.536333.
29. Stouffer, K. M. *et al.* A Universal Method for Crossing Molecular and Atlas Modalities using Simplex-Based Image Varifolds and Quadratic Programming. *bioRxiv* (2023) doi:10.1101/2023.03.28.534622.
30. Comiter, C. *et al.* Inference of single cell profiles from histology stains with the Single-Cell omics from Histology Analysis Framework (SCHAF). *bioRxiv* (2023) doi:10.1101/2023.03.21.533680.
31. Qin, F., Luo, X., Cai, B., Xiao, F. & Cai, G. Spatial pattern and differential expression analysis with spatial transcriptomic data. *bioRxiv* 2023.07.06.547967 (2023) doi:10.1101/2023.07.06.547967.
32. Bafna, M., Li, H. & Zhang, X. CLARIFY: cell–cell interaction and gene regulatory network refinement from spatially resolved transcriptomics. *Bioinformatics* **39**, i484–i493 (2023).
33. Liu, W. *et al.* ProFAST: a fast and scalable factor analysis for spatially aware dimension reduction of multi-section spatial transcriptomics data. *bioRxiv* 2023.07.11.548486 (2023) doi:10.1101/2023.07.11.548486.
34. Huo, Y. *et al.* Integrating multi-modal information to detect spatial domains of spatial

transcriptomics by graph attention network. *J. Genet. Genomics* (2023) doi:10.1016/j.jgg.2023.06.005.

35. Virshup, I., Rybakov, S., Theis, F. J., Angerer, P. & Alexander Wolf, F. anndata: Annotated data. *bioRxiv* 2021.12.16.473007 (2021) doi:10.1101/2021.12.16.473007.
36. Jordahl, K. *et al. geopandas/geopandas: v0.8.1*. (2020). doi:10.5281/zenodo.3946761.
37. Amezquita, R. A. *et al.* Orchestrating single-cell analysis with Bioconductor. *Nat. Methods* **17**, 137–145 (2020).
38. Righelli, D. *et al.* SpatialExperiment: infrastructure for spatially-resolved transcriptomics data in R using Bioconductor. *Bioinformatics* **38**, 3128–3131 (2022).
39. Pebesma, E. Simple features for R: Standardized support for spatial vector data. *R J.* **10**, 439 (2018).
40. Kuhn, M. & Wickham, H. *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles*. https://tidymodels.tidymodels.org/ (2020).
41. Harrower, M. & Brewer, C. A. ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps. *Cartogr. J.* **40**, 27–37 (2003).
42. Crameri, F. *Scientific colour maps*. (2018). doi:10.5281/zenodo.1243909.
43. Bunis, D. G., Andrews, J., Fragiadakis, G. K., Burt, T. D. & Sirota, M. dittoSeq: universal user-friendly single-cell and bulk RNA sequencing visualization toolkit. *Bioinformatics* **36**, 5535–5536 (2021).
44. Janesick, A. *et al.* High resolution mapping of the breast cancer tumor microenvironment using integrated single cell, spatial and in situ analysis of FFPE tissue. *bioRxiv* 2022.10.06.510405 (2022) doi:10.1101/2022.10.06.510405.
45. He, S. *et al.* High-plex Multiomic Analysis in FFPE at Subcellular Level by Spatial Molecular Imaging. *bioRxiv* 2021.11.03.467020 (2022) doi:10.1101/2021.11.03.467020.
46. Stickels, R. R. *et al.* Highly sensitive spatial transcriptomics at near-cellular resolution with Slide-seqV2. *Nat. Biotechnol.* **39**, 313–319 (2021).

47. Lohoff, T. *et al.* Integration of spatial and single-cell transcriptomic data elucidates mouse organogenesis. *Nat. Biotechnol.* **40**, 74–85 (2022).
48. Black, S. *et al.* CODEX multiplexed tissue imaging with DNA-conjugated antibodies. *Nat. Protoc.* **16**, 3802–3835 (2021).
49. McKellar, D. W. *et al.* Large-scale integration of single-cell transcriptomic data captures transitional progenitor states in mouse skeletal muscle regeneration. *Commun Biol* **4**, 1280 (2021).
50. Bhuva, D. D. *et al.* Library size confounds biology in spatial transcriptomics data. *bioRxiv* 2023.03.15.532733 (2023) doi:10.1101/2023.03.15.532733.
51. Vizgen showcase Liver1Slice1. https://console.cloud.google.com/storage/browser/vz-liver-showcase/Liver1Slice1;tab=objects?pageState=(%22StorageObjectListTable%22:(%22f%22:%22%255B%255D%22))&prefix=&forceOnObjectsSortingFiltering=false&pli=1.
52. Griffith, D. A. Negative Spatial Autocorrelation: One of the Most Neglected Concepts in Spatial Statistics. *Stats* **2**, 388–415 (2019).
53. Shang, L. & Zhou, X. Spatially aware dimension reduction for spatial transcriptomics. *Nat. Commun.* **13**, 7203 (2022).
54. Townes, F. W. & Engelhardt, B. E. Nonnegative spatial factorization applied to spatial genomics. *Nat. Methods* **20**, 229–238 (2023).
55. Velten, B. *et al.* Identifying temporal and spatial patterns of variation from multimodal data using MEFISTO. *Nat. Methods* **19**, 179–186 (2022).
56. Svensson, V., Teichmann, S. A. & Stegle, O. SpatialDE: identification of spatially variable genes. *Nat. Methods* **15**, 343–346 (2018).
57. Sun, S., Zhu, J. & Zhou, X. Statistical analysis of spatial expression patterns for spatially resolved transcriptomic studies. *Nat. Methods* **17**, 193–200 (2020).
58. BinTayyash, N. *et al.* Non-parametric modelling of temporal and spatial counts data from

RNA-seq experiments. *Bioinformatics* **37**, 3788–3795 (2021).

59. Zhu, J., Sun, S. & Zhou, X. SPARK-X: non-parametric modeling enables scalable and robust detection of spatial expression patterns for large spatial transcriptomic studies. *Genome Biol.* **22**, 184 (2021).
60. 5k Peripheral Blood Mononuclear Cells (PBMCs) from a Healthy Donor (Next GEM) Single Cell Gene Expression Dataset by Cell Ranger 3.0.2. https://www.10xgenomics.com/resources/datasets/5-k-peripheral-blood-mononuclear-cells-pbm-cs-from-a-healthy-donor-next-gem-3-1-standard-3-0-2.
61. Jackson, K. C. *et al.* Quantitative assessment of single-cell RNA-seq clustering with CONCORDEX. *bioRxiv* 2023.06.28.546949 (2023) doi:10.1101/2023.06.28.546949.
62. Pullin, J. M. & McCarthy, D. J. A comparison of marker gene selection methods for single-cell RNA sequencing data. *bioRxiv* 2022.05.09.490241 (2022) doi:10.1101/2022.05.09.490241.
63. Adult Mouse Olfactory Bulb Spatial Gene Expression Dataset by Space Ranger 2.0.0. https://www.10xgenomics.com/resources/datasets/adult-mouse-olfactory-bulb-1-standard-1.
64. Pelikan, J. *The Vindication of Tradition*. (Yale University Press, 1984).
65. Getis, A. Spatial Weights Matrices. *Geogr. Anal.* **41**, 404–410 (2009).
66. Wickham, H. Data Analysis. in *ggplot2: Elegant Graphics for Data Analysis* (ed. Wickham, H.) 189–201 (Springer International Publishing, 2016).