Bioconductor Spatial Data and Image Analysis Hackathon

Helena Crowell[1*], Luca Marconato[2,3], Ilaria Billato[4], Matteo Calgaro[4], Robert Castelo[6], Riccardo Ceccaroni[4], Carissa Chen[4], Patrick Danaher[14], Sean Davis[7*], Martin Emons[8], Gabriel Grajeda[9], Hugo Gruson[2], Samuel Gunz[8], Juan Henao[10], Rafael Irizarry[9], Sviatoslav Kharuk[2], Daria Lazic[2], Artür Manukyan[11], Pere Moles-Seró[6], Elizabeth Purdom[12], Dario Righelli[5], Gabriele Sales[4], Mike L. Smith[13], Charlotte Soneson[15,16], Michael B. Stadler[15,16,17], Wolfgang Huber[2], Davide Risso[4*]

1 Centro Nacional de Análisis Genómico, Barcelona, Spain,
2 European Molecular Biology Laboratory, Heidelberg, Germany,
3 German Cancer Research Center (DKFZ), Heidelberg, Germany,
4 University of Padova, Padova, Italy,
5 Department of Biology (DiBio), University of Padova, Padova, Italy,
6 Universitat Pompeu Fabra, Barcelona, Spain,
7 University of Colorado Anschutz School of Medicine, Aurora, CO, USA,
8 University of Zurich and Swiss Institute of Bioinformatics, Zurich, Switzerland,
9 Harvard T.H. Chan School of Public Health, Boston, MA, USA,
10 Computational Health Center, Helmholtz Munich, Neuherberg, Germany,
11 Max Delbrück Center for Molecular Medicine, Berlin, Germany,
12 Department of Statistics, University of California, Berkeley, CA, USA,
13 Computational Innovation, Boehringer Ingelheim Pharma GmbH & Co. KG, Biberach, Germany,
14 Bruker Spatial Biology, Seattle, WA, USA,
15 Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland,
16 SIB Swiss Institute of Bioinformatics, Basel, Switzerland,
17 University of Basel, Basel, Switzerland,

* helena@crowell.eu * seandavi@gmail.com * davide.risso@unipd.it

Abstract

We report the outcomes of the Bioconductor Spatial Data and Image Analysis Hackathon, held in Venice, Italy in April 2026. Twenty-seven researchers and software developers organized into four teams to advance R/Bioconductor capabilities for spatial omics and image analysis. Outputs include scalable raster–polygon workflows for whole-slide pathology, an R package for running spatial foundation models via reticulate/basilisk, a spatially stratified differential expression framework, and substantial enhancements to the R `SpatialData` infrastructure for interoperability with the Python `spatialdata` ecosystem. All software is openly available and under active development.

1 Introduction

This manuscript documents the work performed at the Bioconductor Spatial Data and Image Analysis Hackathon, held in Venice, Italy in April 2026. The hackathon brought together 27 researchers and software developers to advance Bioconductor capabilities in

spatial data handling, analysis, and image analysis. Work was organized into team projects spanning three days, with each team independently scoping and executing on a challenge important to its members.

1.1 Schedule and structure

Each day began at 09:00 with a brief plenary across all teams. Coffee breaks at 10:30 and 15:30, plus a 12:30–13:30 lunch, gave teams structured opportunities to compare notes. Days 1 and 2 ended at 17:00 with five-minute single-slide updates from each team. Day 1 also included a mid-day plan presentation right after lunch. The mid-day plan helped teams converge quickly on a focused project, with the explicit understanding that the plan would evolve over the next two days. The end-of-day summaries surfaced shared challenges and made cross-team progress visible.

Table 1. Daily schedule

Time	Activity
09:00	Brief plenary with all teams
10:30	Coffee break
12:30 – 13:30	Lunch
13:30 (Day 1)	Initial project plan, one slide per team
15:30	Coffee break
17:00 (Days 1–2)	End-of-day team summaries, one slide per team

Each team worked in a public GitHub repository created and maintained throughout the three days. The `README.md` was emphasized as the entry point, on the premise that future collaboration and visibility scale with a strong landing page.

Each team also designated a writer responsible for contributing to this manuscript. Author and conflict-of-interest information was collected in a shared spreadsheet for attribution. A short writers' meeting on the morning of day 3 agreed on figure and text contribution mechanics; writers then worked with their teams through the afternoon to complete drafts before departing.

1.2 Hackathon teams

Participants self-organized into four teams:

1. Image and segmentation data manipulation and visualization (Section 2)
2. Spatial foundation models for the Bioconductor community (Section 3)
3. Spatially stratified differential expression analysis (Section 4)
4. Enhanced infrastructure and interoperability for spatial data in Bioconductor (Section 5)

The remainder of this manuscript reports each team's work in a common four-part structure — motivation, methods, results, and discussion and next steps — followed by a cross-cutting discussion that draws out themes shared across the projects.

2 Image and segmentation data manipulation and visualization

Source code: https://github.com/drisso/segmentation-analysis

2.1 Motivation

Spatial omics and digital pathology have moved from low-plex imaging to gigapixel whole-slide datasets. These datasets typically pair multi-gigabyte whole-slide images (WSIs) with segmentation masks identifying hundreds of thousands to millions of nuclei or cell boundaries. The work reported here focuses on classical pathology images (hematoxylin and eosin, H&E), illustrated with an example from The Cancer Imaging Archive [1]; the methodology generalizes to spatial transcriptomics data when H&E or immunofluorescence images accompany transcriptomic profiles, which is the case for most modern datasets.

The Python ecosystem offers powerful frameworks for spatial image processing, including `spatialdata` [2]. R/Bioconductor has a comparably mature suite of geostatistical and geographical analysis tools — `terra`, `sf` [3], and `spatstat` [4] — but as image sizes and cell counts grow, R users hit a "raster–polygon bottleneck": the inability to efficiently extract image features from massive rasters and intersect them with millions of polygons without exceeding system memory.

The core problem addressed here is enabling **scalable image feature extraction** in R. Bridging cloud-optimized image formats (Zarr) with high-performance spatial databases (GeoParquet) makes R's specialized spatial-statistics tooling available to digital-pathology workloads at whole-slide scale. Building on the same foundation, an interactive Shiny + Plotly workflow lets users select arbitrary regions of interest (ROIs) directly from a tissue image and feed those polygons into the same lazy subsetting pipeline.

2.2 Methods

The methodological work targeted efficient reading and cropping of large OME-TIFF and Zarr images, and efficient subsetting of large polygon collections in R. The resulting pipeline prioritizes memory efficiency by never materializing more data than needed for a single processing window, using **GeoParquet** for polygon storage and **OME-Zarr** for image storage.

2.2.1 Example data

The test image is the H&E slide `TCGA-02-0001-01Z-00-DX1.83fce43e-42ac-4dcd-b156-2908e75f2e47` from TCGA, retrieved via the `imageTCGA` Bioconductor package [5]. The original SVS image was converted into OME-TIFF and OME-Zarr; it has three channels (RGB) and 35,558 × 48,002 pixels, and the Zarr file contains 48 chunks of shape 3 × 6,688 × 6,688.

`imageTCGA` also provides nuclear segmentations from HoVer-Net [6]. Polygons are originally distributed as GeoJSON; a GeoParquet version was produced for this work. The polygon file contains 333,207 polygons, with a 2× scale factor between polygon and image coordinates (the polygons assume 40× magnification while the image is 20×).

2.2.2 Technical stack

Table 2. Components of the raster–polygon pipeline.

Component	Role	Reference
OME-Zarr	Cloud-optimized chunked image format used to store the H&E pixels.	zarr.dev
GeoParquet	Columnar polygon storage with bounding-box metadata for fast spatial filtering.	geoparquet.org
terra	Lazy raster loading and raster–vector intersection (originally for satellite imagery).	[7]
DuckDB + duckspatial	Out-of-memory spatial filtering — query million-row polygon files as if they were a local database.	[8]
ZarrArray	`DelayedArray` backend for native Zarr access within Bioconductor.	[9, 10]

2.2.3 Algorithm

Rather than iterating over polygons (the "vector-first" approach), the algorithm iterates over the **intrinsic Zarr chunks** of the image (the "raster-first" approach). For each chunk $C_{i,j}$:

1. **Spatial fetch.** A SQL query is pushed to the Parquet file to retrieve only polygons P where $\mathrm{BBox}(P) \cap \mathrm{BBox}(C_{i,j}) \neq \emptyset$.
2. **Lazy read.** The image data for $C_{i,j}$ is read into memory.
3. **Rasterize.** The selected polygons are rasterized into a label mask.
4. **Compute.** Per-channel pixel statistics (e.g., mean intensity) are computed for each labeled object.

The main difficulty is the *border effect* (Fig 1 A): each chunk should be read only once, but when a polygon straddles two chunks the in-memory pixel set for one chunk is incomplete. A pixel-accumulation strategy resolves this (Fig 1 B): when a polygon partially overlaps a chunk, the rasterized partial pixel set is stored under the polygon's identifier; when the adjacent chunk is processed, the buffer is extended and statistics are computed only once the polygon is complete.

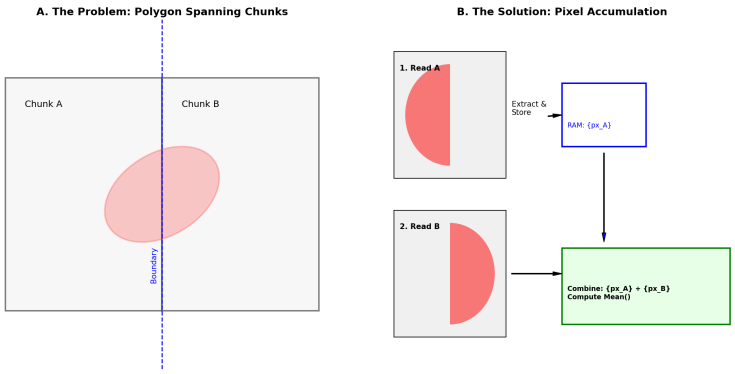

Figure 1. Schematic of the border effect and pixel-accumulation strategy. **(A)** A polygon straddles two Zarr chunks; processing one chunk at a time leaves incomplete pixel coverage. **(B)** A buffered accumulation strategy combines partial pixel sets from successive chunks before computing the final per-object statistic, without re-reading any chunk.

2.2.4 Interactive ROI selection

Building on the rasterisation primitives in `terra`, an interactive ROI-selection workflow uses `plotly` [11] to let users draw arbitrary polygons over a tissue image — demonstrated here on the mouse ileum profiled by MERFISH (accessed via ExperimentHub ID `EH7543`) [12, 13] as proof of concept. User-drawn polygons are returned as coordinate geometries and converted into spatial vector objects. The ROI is then materialized through the same lazy pipeline used elsewhere in the section: transcript coordinates are filtered with out-of-memory operations in `duckspatial`, and the underlying image is cropped with `terra` and visualised with `tidyterra` [14].

2.3 Results

2.3.1 Random access to polygon files

Pushing spatial queries to an out-of-memory Parquet file substantially outperforms traditional GeoJSON-based workflows, enabling whole-slide analyses on standard workstations. While `duckspatial` works with both formats, polygon subsetting was approximately 50× faster on GeoParquet than on GeoJSON.

2.3.2 Cropping and visualization of large images

Reading, cropping, and visualizing large OME-TIFF images is efficient using `terra`'s `rast()` and `crop()`. For OME-Zarr, combining `ZarrArray` subsetting with `terra::rast()` works equally well. The intersection of high-speed polygon queries with high-resolution pathology crops is illustrated in Fig 2.

2.3.3 Mask creation and statistics computation

Combining the components above, a prototype implementation:

- iterates over Zarr image chunks,
- selects polygons overlapping each chunk,
- rasterizes them into an image mask, and
- computes mean per-channel pixel intensity over each segmented object.

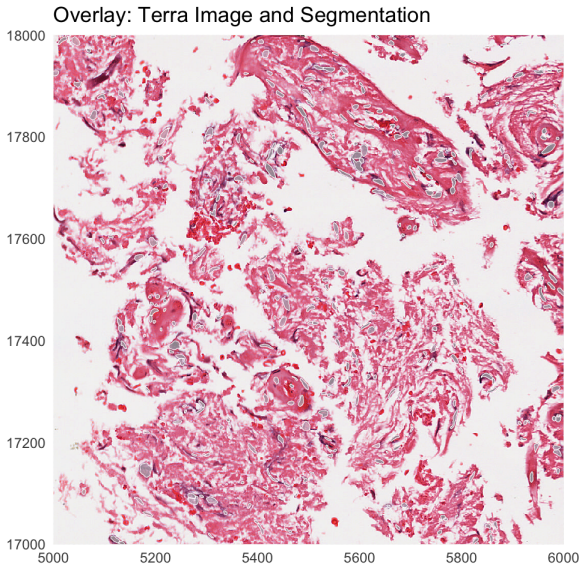

Figure 2. Overlay of segmentation polygons on a TCGA H&E whole-slide image crop, demonstrating the intersection of a lazily loaded Zarr image with polygons retrieved via spatial query from GeoParquet.

On the example image, the analysis runs in 986 ± 2 seconds using R 4.5.3 on a MacBook Pro with an Apple M5 chip, 10-core CPU, 32 GB RAM, and local-SSD image data. The workflow reads only the relevant cell-boundary shapes and Zarr image data for each chunk, avoiding full-image materialization; loading the full image into memory on the same machine resulted in an out-of-memory error.

2.3.4 Interactive ROI selection in practice

The Shiny + Plotly application described in Methods exposes the lazy subsetting pipeline through a point-and-click interface: an analyst draws an arbitrary polygon over a tissue image, the polygon coordinates are exported as a spatial vector object, and the underlying image and spatial-transcriptomics data are subset to that region for visualisation and downstream analysis of tissue compartments.

2.4 Discussion and next steps

The combination of OME-Zarr image storage, GeoParquet polygon storage, and DuckDB spatial queries makes whole-slide pathology analysis tractable in R on commodity hardware, without rewriting the underlying spatial-statistics tooling. Pushing predicates into chunked, indexed storage rather than loading data into R is the central design choice, and the same pattern recurs across the other hackathon teams.

The current prototype skips polygons that straddle multiple chunks, sidestepping the border effect described in Fig 1. Implementing the full pixel-accumulation strategy is the natural next step. Additional priorities are packaging the chunk-iteration loop into reusable utilities, validating performance on larger and more diverse datasets, and integrating the workflow with downstream Bioconductor packages for cell-level analysis.

3 Spatial foundation models for the Bioconductor community

Source code: https://github.com/drighelli/fomo

3.1 Motivation

Foundation models for spatial data are predominantly distributed as Python packages, and configuring them for a particular compute environment is often fragile: most have specific version dependencies that are not always documented, even when the model is on PyPI. This makes individual models hard to install for many R users and makes reproducible benchmarking across models onerous. A second gap is representational: most Python tools expect `AnnData`, while the natural Bioconductor container for spatial data is `SpatialExperiment`, so any reusable wrapper has to reconcile the two. The `fomo` package addresses both problems.

3.2 Methods

3.2.1 Package design

`fomo` configures and runs foundation-model pipelines from R. It uses `basilisk` and `reticulate` to build separate environments per model (Fig 3), avoiding the dependency conflicts that arise when attempting to satisfy all models in a single environment. The package ships a collection of `BasiliskEnvironment` schemas that pin a Python version and the required PyPI dependencies for each model. For models not on PyPI, `fomo` provides template scripts to build environments via `reticulate`.

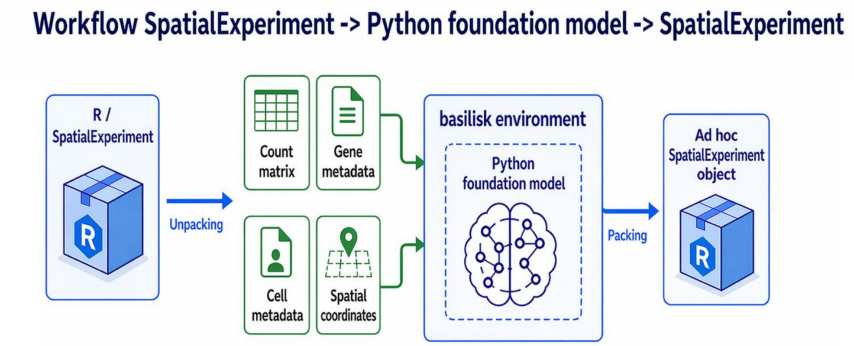

Figure 3. Schematic of the `fomo` package. Input data in `SpatialExperiment` is fed to functions that set up isolated Python environments via `reticulate`, run the model, and return embeddings into the `SpatialExperiment` object.

For each supported model, the package exports a `Run_<model>()` function (e.g., `Run_novae()`). The package focuses on **zero-shot** use of pre-trained models: input data are passed in and embeddings come back, with no fine-tuning required. The final design takes a `SpatialExperiment` as input and writes embeddings to its `reducedDims` slot.

3.2.2 Models

Six models spanning the breadth of current spatial-foundation use cases were prioritized (Table 3). Wherever possible, weights are pulled from Hugging Face so the entire

workflow runs through the package; the exception is scGPT, which uses upstream PyTorch checkpoints.

Table 3. Models packaged in `fomo`. ✓ = exposed via `Run_*()`, ◦ = environment recipe shipped but no wrapper yet, × = not currently runnable through the package.

Status	Model	Category	PyPI	Primary input	Output	Embedding size
✓	scGPT [15]	Single-cell transcriptomics	✓	Gene counts	Cell embeddings	512
◦	scFoundation [16]	Single-cell transcriptomics	✓	Gene counts	Gene / cell embeddings	3072
✓	Novae [17]	Spatial transcriptomics	✓	Gene counts + spatial coords	Neighborhood-context embedding + zero-shot domains	64
◦	Prov-GigaPath [18]	H&E imaging	×	H&E WSI tiles	Tile / slide embeddings	tile 1536, slide 768
✓	Nimbus [19]	Spatial proteomics	✓	Multiplex TIFF + segmentation mask	Per-cell expression estimates and annotations	—
×	KRONOS [20]	Spatial proteomics	×	TIFF + mask + marker annotation	Per-patch embeddings	384

scGPT. Although not designed for spatial data, transcriptional profiles from spatial assays can be passed through it for comparison with spatial-specific models. (A spatial extension is available [21] but not yet integrated.) The implementation uses upstream PyTorch checkpoints (`.pt` plus support `.json` files) rather than the Hugging Face port, because scGPT distributes multiple pretrained checkpoints and working with them directly preserves user choice. `Run_scGPT()` takes the path to an `.h5ad` file and the directory containing the pretrained model.

Novae. Embeddings are accessible from the returned `AnnData` at `adata$obsm$novae_latent` and domain assignments at `adata$obs$novae_domains_7`. Novae runs on CUDA GPUs and Apple Metal Performance Shaders (MPS).

Prov-GigaPath. Integration is in progress. The current blocker is provisioning the Python environment on remote GPU servers: the required Python build is slow to

download and a robust install pattern for that setting is still being worked out.

Nimbus. Operates marker by marker across the multiplexed image, which makes it portable across multiplexed-imaging platforms with different channel counts.

3.2.3 `SpatialExperiment` ↔ `AnnData` conversion via `anndataR`

Because most Python tools expect `AnnData`, `anndataR` was extended to support bidirectional conversion between `SpatialExperiment` and `AnnData` — layering a spatially aware translation on top of the existing `SingleCellExperiment` conversion framework. Canonical mappings for assays, feature- and cell-level annotations, dimensionality reductions, and graph-like pairwise structures are preserved; the new logic adds explicit handling for spatial state. Forward (R → Python), `spatialCoords` are serialized to `obsm["spatial"]` and image metadata from `imgData` is written to `uns["spatial"]` in a library-indexed nested layout compatible with Scanpy conventions; in reverse (Python → R), those structures are parsed to reconstruct `SpatialExperiment` objects with restored coordinates and images.

The central difficulty is that `SpatialExperiment` carries semantics that are not native to `AnnData`: spatial coordinates are not generic embeddings, image metadata are heterogeneous and partially technology-specific, and multi-library datasets require deterministic reconciliation between observation-level sample annotations and image-level library identifiers. Reserved spatial keys, conflict checks against user-defined `obsm` and `uns` mappings, and a round-trip metadata container under `uns["anndataR_spatialexperiment"]` together preserve otherwise non-canonical information (original coordinate names, selected spatial keys, library identifier provenance, image export mode). Image payloads can be embedded, referenced, or omitted to balance interoperability, file size, and reversibility.

The current development branch can be installed with:

```
remotes::install_github("drighelli/anndataR@from_as_spe")
```

3.3 Results

This section reports two complementary results: a use case demonstrating zero-shot application of a current spatial-proteomics foundation model (KRONOS), and a status report on the `SpatialExperiment` ↔ `AnnData` conversion that underpins R-side use of all the wrapped models.

3.3.1 KRONOS use case

KRONOS [20] is a foundation model purpose-built for spatial proteomics. It was trained on more than 47 million image patches covering 16 tissue types and 175 protein markers, and uses a Vision Transformer based on DINOv2 modified to handle highly multiplexed images. KRONOS extracts 384-dimensional embeddings for each patch, which can be used for cell phenotyping, artifact detection, or outcome prediction.

Because KRONOS is not distributed as a Python package, it does not yet fit the `fomo` envelope and was instead explored directly in a use case of interest. Starting from a multiplexed image with ~56 channels (Fig 4), KRONOS was applied to 128 × 128 sliding windows to obtain feature vectors. PCA of these embeddings, viewed spatially as RGB-mapped per-window boxes, immediately surfaces a clear image artifact (Fig 5).

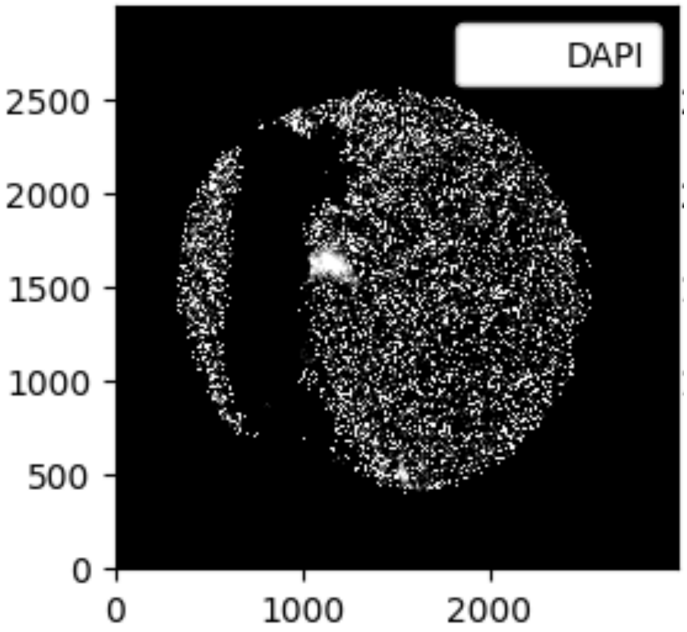

Figure 4. Original DAPI channel of the multiplexed input image.

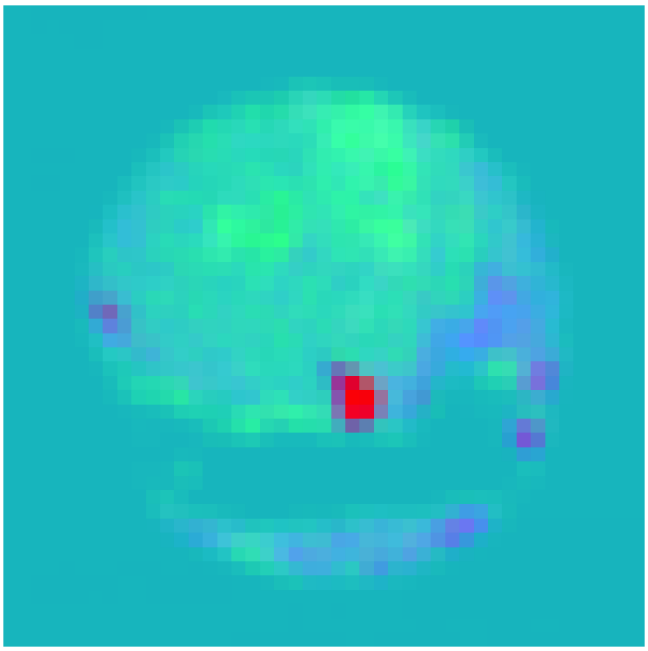

Figure 5. First three principal components of the KRONOS feature vectors, mapped to RGB and laid out in their original spatial positions. The pattern strongly indicates an image artifact.

3.3.2 Conversion testing

Preliminary evaluation of the `SpatialExperiment` ↔ `AnnData` converter surfaces discrepancies in specific configurations, particularly those involving technology-specific spatial metadata or image representations. The reproducibility case study in the `renovae` repository (https://github.com/sales-lab/renovae) documents one such mismatch and helps locate the remaining failure points before the work is upstreamed as a pull request to `anndataR`.

3.4 Discussion and next steps

The `fomo` envelope reduces the per-model setup burden to a package install for the models it covers, and the `anndataR` extension closes the representational gap between `SpatialExperiment` and `AnnData` for the spatial state Python tools depend on. Continued work falls into three areas: completing the wrappers for scFoundation and Prov-GigaPath; designing a robust pattern for models distributed outside PyPI (KRONOS in particular); and resolving the remaining `SpatialExperiment` ↔ `AnnData` round-trip discrepancies before upstreaming the converter. A standardized benchmarking interface across models is the natural follow-on once the wrapper set is complete.

4 Spatially stratified differential expression analysis

Source code: https://github.com/mcalgaro93/SpatialStratifiedDE

4.1 Motivation

Spatial data are a powerful platform for differential expression (DE) analyses, letting analysts interrogate how a given cell type changes phenotype or expression in response to its environment — for example, how tumor cells respond to T-cell attack, or how fibroblast expression changes in regions of tissue damage.

This kind of phenotype–environment DE testing is typically performed at the level of whole studies or whole tissues. That has two important limitations. First, it ignores spatial autocorrelation in gene expression, leading to biased estimates and overconfident standard errors. Second, it ignores the fact that correlations can be heterogeneous across a tissue: tumor cells, for example, may use different pathways in response to T-cell attack under different local conditions. Spatial datasets are now large and rich enough to support DE questions at finer resolution than the entire tissue.

The proposal pursued here is to split the tissue into dozens or hundreds of small *patches* and run DE separately in each, revealing the precise regions where a given DE pattern occurs. Individual patches may be underpowered, so a spatially smoothed meta-analysis recovers confidence: for each patch the K most similar patches are identified — by underlying traits such as cell-type composition rather than by their DE results — and DE results are meta-analyzed across that neighborhood, regaining statistical power lost to fine-grained stratification.

4.2 Methods

The full workflow chains five stages, summarized in Fig 6: patches are defined from cell coordinates and the design variable; per-patch DE is run via ordinary least squares (OLS); each patch is described by a "spatial-context" feature vector; per-patch results

are stabilized by meta-analysis across nearest neighbors in that feature space; and the resulting patch × gene matrix is the substrate for downstream biological exploration.

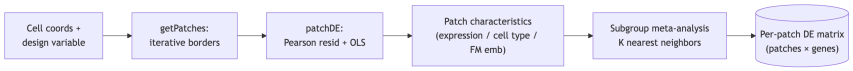

Figure 6. Spatially stratified DE pipeline. Each box corresponds to one Methods subsection below.

4.2.1 Patch definition

`getPatches()` defines patches via an iterative algorithm that incrementally adjusts patch borders to ensure adequate statistical power everywhere. Patches are scored by the sum of squares of the design variable, which determines OLS power; at each iteration, low-power patches expand into higher-power neighbors, gradually homogenizing per-patch power. In the simplest mode, only the target cell type's xy coordinates and design variable values are required.

A more nuanced variant was also explored: ideal patches have high variance in the design variable while being homogeneous in everything else, which controls confounding. This variant first applies standard spatial clustering to partition the tissue into niches or spatial domains and then runs `getPatches()` within each domain. The resulting patches are more internally homogeneous (they do not span domains), at some cost in maximum achievable power; the trade-off is expected to be worthwhile because within-patch confounding is the more dangerous failure mode.

4.2.2 Stratified DE

For each patch, `patchDE()` performs OLS for all genes via a single matrix multiplication, producing complete per-patch DE results in seconds. This efficiency matters: a typical study might involve >18,000 genes and >1,000 patches.

4.2.3 Characterizing patches

Meta-analysis requires a definition of "spatial context" — a matrix encoding each patch in a reasonable number of features. Several reasonable choices are available, depending on the study design:

- Mean gene expression across all cells (or just the target cell type) per patch, optionally reduced to ~50 PCs.
- Cell-type abundances per patch.
- Per-patch averages of cell-level PCA embeddings, optionally with within-patch variance.
- Per-patch averages of foundation-model embeddings (e.g., Novae) over cells.

The right choice is a design question for the analyst.

4.2.4 Unsupervised subgroup meta-analysis

Given per-patch DE estimates and standard errors plus the patch-characteristics matrix, each patch's K nearest neighbors in patch-characteristic space — intended to be biologically similar — are identified, and a meta-analysis is performed across the patch and its neighbors. The unstable per-patch estimate is replaced with a more precise, more confident regional estimate.

4.2.5 Pursuing biological understanding

The output is a matrix of DE results for all patches × genes, which is rich but daunting. The intended downstream analyses include subgroup analyses (drawn from clinical-trial methodology) that surface well-characterized regions where a gene's DE signal is strongest or where results most diverge from the average patch; clustering genes into modules and patches into groups by DE profile; and association of per-patch DE results with covariates such as pathologist annotations or local cell-type abundance.

4.3 Results

A prototype of the `SpatialStratifiedDE` pipeline was applied to two spatial datasets: a mouse colon MERFISH dataset [22] used to study how colon epithelial cells change with proximity to T cells after three days of DSS-induced colitis, and a colon-cancer spatial transcriptomics dataset [23] used to study how tumor cells change with proximity to T cells. Together these illustrate how patch-wise DE combined with spatially smoothed meta-analysis surfaces localized, high-confidence DE patterns.

4.3.1 Iterative algorithm yields patches with reasonable power

Fig 7 shows the improvement of patch power over iterations of `getPatches()` on epithelial cells of the mouse colon MERFISH dataset [22]. Top panels show patch footprints at the first and final iterations; bottom panels show each patch's sum of squares of the predictor variable, which determines DE statistical power.

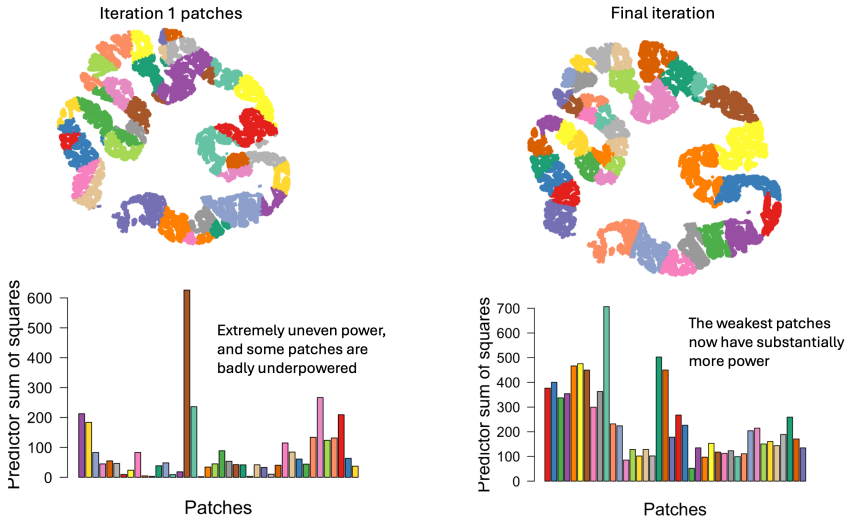

Figure 7. Improvement of patch power over `getPatches()` iterations on mouse colon epithelial cells. Top panels show patch footprints at first and final iterations; bottom panels show each patch's sum of squares of the predictor variable.

4.3.2 Smoothed patch meta-analysis improves statistical power

Fig 8 compares per-patch DE results before and after meta-analysis in the colon-cancer sample dataset [23]. Left: t-statistics from single-patch DE on a single gene. Right: z-statistics from meta-analysis on each patch's 15 most comparable patches (nearest neighbors in patch-composition space).

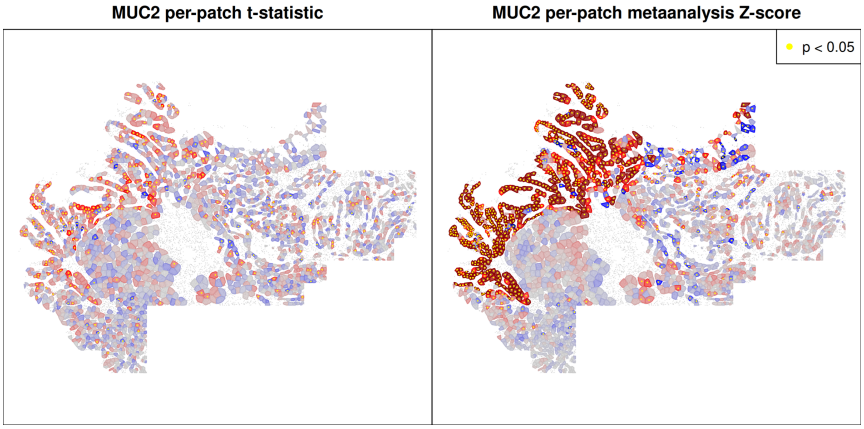

Figure 8. Single-patch versus meta-analyzed DE in the colon-cancer sample. Left: t-statistics from single-patch DE for one gene. Right: z-statistics from meta-analysis across each patch's 15 nearest neighbors in patch-composition space.

4.4 Discussion and next steps

The approach is not limited to differential expression. Any method that produces a summary statistic and a standard error can be stratified across patches and subjected to the same meta-analysis workflow.

Several deep questions remain open. Is this a hypothesis-testing framework or an exploratory framework? How should the analysis account for dependence structures among patches? How should multiple-testing correction work when neighborhoods overlap? The methodological roadmap centers on these issues alongside more practical refinements: recommendations and utility functions for defining patch characteristics, vignettes for interpreting subgroup results, an accounting for non-independence among patches (cf. the `metapod` package), an alternative formulation in which a patch's neighbors serve as a prior that the patch's own data updates, and clustering of genes and patches by per-patch DE profile.

On the code and packaging side, planned work includes a dedicated meta-analysis function, a Bioconductor wrapper for the pipeline, convenience functions for patch-characteristic construction, a data package suitable for LLM-driven exploration, and vignettes and case studies. Two outstanding decisions are whether the framework is best positioned as hypothesis-testing or exploratory — diagnostics will follow from that choice — and the final package name (working title: `SpaceMosaic`).

The ultimate goal is to enable nuanced, confident statements like "[cell type] up-regulates [gene] in response to [X] in spatial regions with [these characteristics]" — without worrying that the result is an artifact of cryptic confounding. Defining the right post-hoc statistics and diagnostic plots to support that claim is ongoing work.

5 Enhanced infrastructure and interoperability for spatial omics data in Bioconductor

Source code:

- `SpatialData` — R interface to Python's `spatialdata`
- `SpatialData.data` — example datasets
- `SpatialData.plot` — visualization

- VeniceInterop — end-to-end workflow and relationships prototype

5.1 Motivation

Modern spatial-omics platforms produce data that are increasingly authored, stored, and pre-processed in the Python `scverse` ecosystem (notably the `spatialdata` and `AnnData` packages), while Bioconductor remains the natural downstream environment for many statistical and biological analyses. Without a faithful bridge between the two, users either duplicate infrastructure or choose one ecosystem at the cost of tools that live in the other. The work described here closes that gap by extending the R `SpatialData` package and its companions into a complete reader, writer, and analysis surface compatible with the Python representation, and by demonstrating end-to-end workflows that mix R and Python tools on a single dataset.

5.2 Methods

The `SpatialData` family of R packages implements a faithful R-side representation of the Python `spatialdata` format and adds the lazy access patterns required to work with it at scale. The methods below describe the storage layers and their R readers, the companion data and visualization packages, and the lazy spatial-subsetting pattern that recurs across analyses.

5.2.1 Storage layers and reader path

Python's `spatialdata` framework is a standardized format for spatial omics, storing data as `.zarr` files using an extension of the OME-NGFF specification [2]. The R `SpatialData` package provides an R interface to `spatialdata` and integrates the resulting representations into the Bioconductor ecosystem. *Image* and *label* elements (e.g., microscopy and rasterized segmentation results) are represented as `.zarr`-backed `ZarrArray`s via the `Rarr` package. *Point* elements (e.g., transcripts) and *shape* elements (e.g., cell or nucleus boundaries) are stored in `.parquet` and represented as lazy data frames via `duckspatial`. *Table* elements (e.g., shape annotations) are represented as `SingleCellExperiment` objects, read directly from `AnnData` via `anndataR` [24].

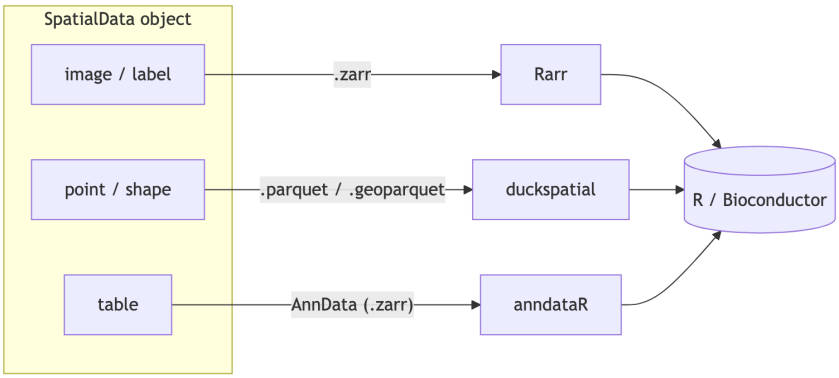

Figure 9. Storage and reader path for each `SpatialData` element type. Images and labels are `.zarr`-backed; points and shapes are `.parquet`-backed; tables are read from `AnnData`/`.zarr`.

Because tabular information lands in `SingleCellExperiment`, interoperability with Bioconductor's existing single-cell infrastructure is preserved. The on-disk representations of the other components support efficient lazy access via `Rarr` and

`duckspatial`. Available functionality includes spatial queries by rectangular bounding box or polygon, and masking between elements to aggregate information across layers.

5.2.2 Companion packages

Two companion packages provide modular `SpatialData` infrastructure. `SpatialData.data` supplies toy and realistic example datasets (sequencing- and imaging-based), and `SpatialData.plot` handles visualization — including layered plots that overlay segmentation, mRNA molecules, and high-plex or histopathology imagery. An online gallery is under active development at https://helenalc.github.io/SpatialData.

5.2.3 Lazy spatial subsetting via `duckspatial`

Spatial subsetting emerged as a fundamental, recurring operation across analyses. These operations apply to potentially very large on-disk data — millions of transcript points or cells — and must avoid materializing the whole dataset in memory. The implementation uses `duckspatial`, which exposes the spatial extension of DuckDB from R: spatial filtering is compiled to SQL and executed by the database engine, loading only the matched rows. For example, transcripts overlapping a selected cell or within a region of interest can be efficiently extracted from a Parquet file with hundreds of millions of rows. Query regions can be rectangular bounding boxes or arbitrary polygons. This pattern is now used by the `query` method for both points and shapes in `SpatialData`.

5.3 Results

The work delivered an end-to-end interoperability demonstration that crosses the Python/R boundary on a single dataset, plus a prototype redesign of how relationships across `SpatialData` elements are encoded and explored.

5.3.1 End-to-end workflow

To demonstrate the current capabilities and interoperability with existing R/Bioconductor tools, an end-to-end workflow was built that reads results from `SegTraQ` — a Python framework for cell segmentation and transcript-assignment QC — into R, applies two downstream analyses (RCTD-based signal purity scoring via `spacexr` [25] and delineation of multicellular anatomical structures via `sosta` [26]), and visualizes the results spatially with `SpatialData.plot`. The full workflow — read data written by Python's `spatialdata`, compute QC scores and anatomical structures, visualize subsets — is summarized in Fig 10, with image-specific visualization shown in Fig 11. An R Markdown source plus rendered HTML is published at https://github.com/HelenaLC/VeniceInterop.

5.3.2 Relationships across `SpatialData` elements

As a side project, a new design for encoding and exploring relationships across `SpatialData` elements — both within and across `SpatialData` objects — was prototyped. Improving the design and exploration of relationship metadata is a prerequisite for broader adoption across packages and languages. The prototype was illustrated on the same Xenium subset used above, complemented with four additional `SpatialData` objects covering further modalities, and accompanied by an interactive viewer (`graph_all.html`); the code is in https://github.com/HelenaLC/VeniceInterop/tree/main/relationships.

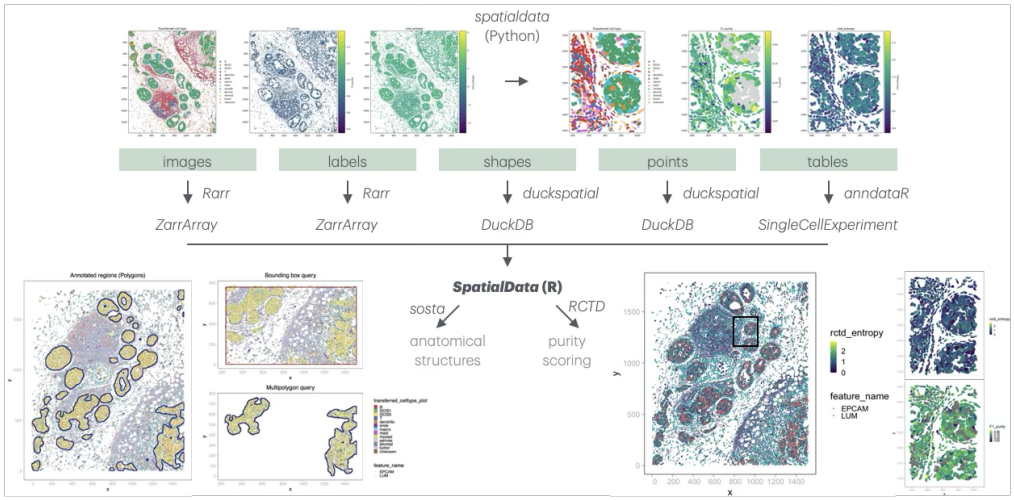

Figure 10. Schematic overview of `SpatialData` in R. Top to bottom: `SpatialData` objects comprise five layers (images, labels, shapes, points, and tables). Images and labels are `.zarr`-backed and read with `Rarr`; shapes and points are `.parquet`-backed and read with `duckspatial`; tables are read from `.zarr` via `anndataR`. Existing R/Bioconductor methods for spatial omics analysis remain interoperable — for example, `sosta` for multicellular structure delineation (left) and RCTD via `spacexr` for deconvolution, used here as a proxy for signal contamination.

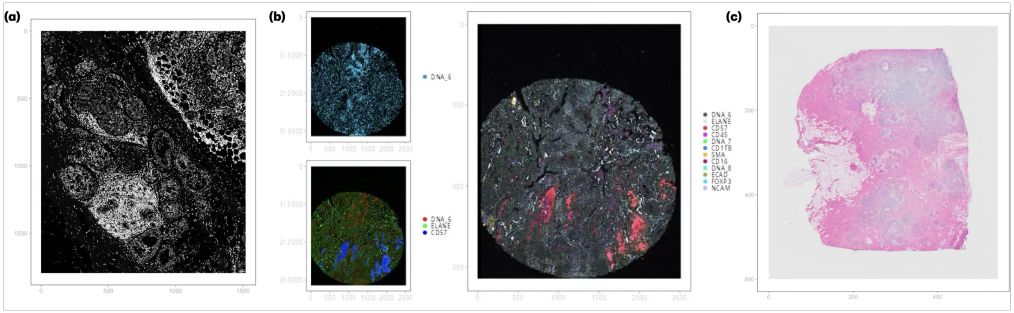

Figure 11. Demonstration of `SpatialData.plot` image-visualization capabilities. **(a)** Greyscale image of the dataset shown in Fig 10. **(b)** Top-left: a 1-channel image with a user-configurable color. Bottom-left: 3-channel images mapped to RGB (user-configurable). Right: a 12-channel image mapped to RGB based on user-defined per-channel colors. Images in (b) are derived from a 12-plex proteomics (MCMICRO) lung adenocarcinoma dataset [27] from `SpatialData.data`. Contrasts are auto-adjusted across all images using 5%-percentile clipping. **(c)** Exemplary H&E staining from a Visium (10x Genomics) breast-cancer dataset [28].

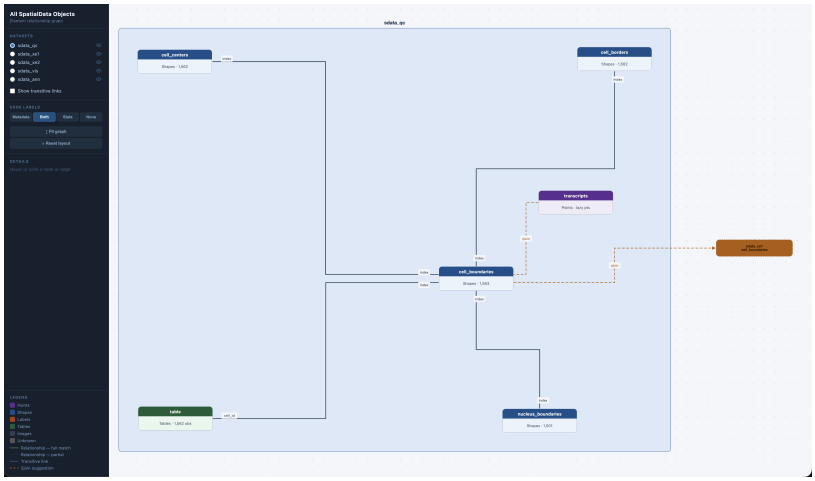

Figure 12. Prototype interactive explorer of relationships across `SpatialData` objects, demonstrating the new relationship-encoding design on a multi-modal Xenium dataset.

5.4 Discussion and next steps

R `SpatialData` development began more than two years ago [29], but full functionality awaited several infrastructural pieces — notably delayed `.zarr` support and `AnnData` interoperability. The hackathon delivered three concrete advances: lazy handling of points (from `.parquet`) and shapes (from `.geoparquet`) via `duckspatial`; demonstrated interoperability with existing R/Bioconductor tools for spatial omics analysis; and improved visualization in `SpatialData.plot`, with an approximately ten-fold speedup and automatic contrast adjustment.

Continued co-design with the Python `spatialdata` developers is essential to keep the two ecosystems coherent as both evolve. Priorities for the next phase are upstreaming the relationship-encoding prototype, expanding the example gallery to cover additional modalities and analysis patterns, and validating the lazy-query pattern on the largest available datasets where its memory advantages should be clearest.

6 Discussion

The four projects reported here address complementary layers of the spatial-omics stack. Team 1 tackled the raster–polygon bottleneck that has historically kept R/Bioconductor away from gigapixel digital pathology, and shows that pairing OME-Zarr with GeoParquet and pushing spatial queries to DuckDB makes whole-slide analysis tractable on a laptop. Team 2 lowered the bar for using spatial foundation models from R by isolating each model in its own Python environment and exposing a uniform `Run_*()` interface, turning what is currently a per-model setup project into a package install. Team 3 prototyped a spatially stratified differential expression framework that lets the location of an effect, not just its magnitude, become a first-class output of an analysis. Team 4 demonstrated end-to-end interoperability between Python's `spatialdata` and the R `SpatialData` family, with `duckspatial`-backed lazy queries and large gains in visualization performance.

A consistent theme across teams is that the hard problem in scaling spatial analysis is not algorithmic but representational: choosing on-disk formats and access patterns that let the rest of the toolchain stay expressive and interactive. GeoParquet, OME-Zarr, and DuckDB recur because they let R code stay declarative while pushing

the heavy lifting to indexed, chunked storage.

Several open questions remain. The patch-stratified DE framework needs to clarify whether it is best framed as hypothesis testing or as an exploratory tool, and to settle on the right multiple-testing model for non-independent neighborhoods. The foundation-models package has to handle models that are not on PyPI without forcing every user to provision their own environment. The `SpatialData` infrastructure work needs continued co-design with the Python `spatialdata` developers to keep the two ecosystems coherent.

All software produced during the hackathon is open and under active development; we welcome contributions, issues, and use cases from the community.

7 Code and data availability

All software developed during the hackathon is hosted in public GitHub repositories linked from the relevant team sections; those repositories remain the active development homes. A consolidated snapshot of the work as it stood at the close of the hackathon, organized by team into per-project folders, is also available at https://github.com/BiocCodingCollaborations/VeniceHackathon2026.

8 Author contributions

H.C., L.M., W.H., and D.R. organized and led the hackathon. S.D. coordinated the production of the preprint manuscript. All authors participated in one or more team projects and contributed to the manuscript.

9 Use of AI tools

Claude (Anthropic, model `claude-opus-4-7`) was used to convert team-authored Google Docs into this Quarto project, scaffold the CI/CD workflow, and propose structural edits — including the table in Table 2 and the diagrams in Fig 9 and Fig 6. The authors reviewed all AI output and retain full responsibility for the manuscript's content.

10 Acknowledgements

The event was organized by the Department of Statistical Sciences of the University of Padova in collaboration with Venice International University. It was funded in part by the European Research Council (ERC) Grant CoG 101171662 and supported by EMBL's Transversal Theme Theory@EMBL. We thank Elena Guzzonato for her help with the organization of the event. S.D. acknowledges support from NCI U24CA289073. R.C. and P.M-S. acknowledge support from grants PID2024-158127NB-I00 funded by MICIU/AEI /10.13039/501100011033 and by "ERDF: A way of making Europe", and 2022-311145 funded by the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation.

References

1. The Cancer Imaging Archive. The Cancer Genome Atlas Ovarian Cancer Collection (TCGA-OV); 2013.

2. Marconato L, Palla G, Yamauchi KA, Virshup I, Heidari E, Treis T, et al. SpatialData: an open and universal data framework for spatial omics. Nature Methods. 2024;doi:10.1038/s41592-024-02212-x.

3. Pebesma E. Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal. 2018;10(1):439–446. doi:10.32614/RJ-2018-009.

4. Baddeley A, Turner R. spatstat: An R Package for Analyzing Spatial Point Patterns. Journal of Statistical Software. 2005;12(6). doi:10.18637/jss.v012.i06.

5. Billato I. imageTCGA: TCGA Diagnostic Image Database Explorer; 2026. Available from: https://bioconductor.org/packages/imageTCGA.

6. Graham S, Vu QD, Raza SEA, Azam A, Tsang YW, Kwak JT, et al. Hover-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. Medical Image Analysis. 2019;58:101563. doi:10.1016/j.media.2019.101563.

7. Hijmans RJ, Brown A, Barbosa M. terra: Spatial Data Analysis; 2026. Available from: https://CRAN.R-project.org/package=terra.

8. Cidre González A, Kotov E, Pereira RHM. duckspatial: R Interface to 'DuckDB' Database with Spatial Extension; 2026. Available from: https://CRAN.R-project.org/package=duckspatial.

9. Pagès H, Smith M, Gruson H, Manukyan A. ZarrArray: Bring Zarr datasets in R as DelayedArray objects; 2026. Available from: https://bioconductor.org/packages/ZarrArray.

10. Pagès H. DelayedArray: A unified framework for working transparently with on-disk and in-memory array-like datasets; 2026. Available from: https://bioconductor.org/packages/DelayedArray.

11. Sievert C, Parmer C, Hocking T, Chamberlain S, Ram K, Corvellec M, et al.. plotly: Create Interactive Web Graphics via 'plotly.js'; 2024. Available from: https://CRAN.R-project.org/package=plotly.

12. Petukhov V, Xu RJ, Soldatov RA, Cadinu P, Khodosevich K, Moffitt JR, et al.. Cell segmentation in imaging-based spatial transcriptomics; 2022.

13. MerfishData; 2026. Available from: http://bioconductor.org/packages/MerfishData/.

14. Hernangómez D. Using the tidyverse with terra objects: the tidyterra package; 2023.

15. Cui H, Wang C, Maan H, Pang K, Luo F, Duan N, et al. scGPT: toward building a foundation model for single-cell multi-omics using generative AI. Nature Methods. 2024;21:1470–1480. doi:10.1038/s41592-024-02201-0.

16. Hao M, Gong J, Zeng X, Liu C, Guo Y, Cheng X, et al. Large-scale foundation model on single-cell transcriptomics. Nature Methods. 2024;21:1481–1491. doi:10.1038/s41592-024-02305-7.

17. Quentin B, Hakim B, Nadège B, Fabrice A, Paul-Henry C. Novae: a graph-based foundation model for spatial transcriptomics data. bioRxiv. 2024;doi:10.1101/2024.09.09.612009.

18. Xu H, Usuyama N, Bagga J, Zhang S, Rao R, Naumann T, et al. A whole-slide

foundation model for digital pathology from real-world data. Nature. 2024;630:181–188. doi:10.1038/s41586-024-07441-w.

19. Rumberger JL, Greenwald NF, Ranek JS, Boonrat P, Walker C, Franzen J, et al. Automated classification of cellular expression in multiplexed imaging data with Nimbus. Nature Methods. 2025;22(10):2161–2170.

20. Shaban M, Chang Y, Qiu H, Yeo YY, Song AH, Jaume G, et al. A foundation model for spatial proteomics. arXiv preprint arXiv:250603373. 2025;.

21. Wang C, Cui H, Zhang A, Xie R, Goodarzi H, Wang B. scGPT-spatial: Continual pretraining of single-cell foundation model for spatial transcriptomics. bioRxiv. 2025; p. 2025–02.

22. Cadinu P, Sivanathan KN, Misra A, Xu RJ, Mangani D, Yang E, et al. Charting the cellular biogeography in colitis reveals fibroblast trajectories and coordinated spatial remodeling. Cell. 2024;187(8):2010–2028.e30. doi:10.1016/j.cell.2024.03.013.

23. Crowell HL, Ruano I, Hu Z, Hong Y, Caratú G, Piessevaux H, et al. Tracing colorectal malignancy transformation from cell to tissue scale. bioRxiv. 2025; p. 2025.06.23.660674. doi:10.1101/2025.06.23.660674.

24. Deconinck L, Zappia L, Cannoodt R, Morgan M, Core S, Virshup I, et al. anndataR improves interoperability between R and Python in single-cell transcriptomics. bioRxiv. 2025; p. 2025.08.18.669052. doi:10.1101/2025.08.18.669052.

25. Cable DM, Murray E, Zou LS, Goeva A, Macosko EZ, Chen F, et al. Robust decomposition of cell type mixtures in spatial transcriptomics. Nature Biotechnology. 2022;40:517–526. doi:10.1038/s41587-021-00830-w.

26. Gunz S, Crowell HL, Robinson MD. Analysis of multicellular anatomical structures from spatial omics data using sosta. bioRxiv. 2025;doi:10.1101/2025.10.13.682065.

27. Schapiro D, Sokolov A, Yapp C, Chen YA, Muhlich JL, Hess J, et al. MCMICRO: a scalable, modular image-processing pipeline for multiplexed tissue imaging. Nature Methods. 2022;19:311–315. doi:10.1038/s41592-021-01308-y.

28. Janesick A, Shelansky R, Gottscho AD, Wagner F, Williams SR, Rouault M, et al. High resolution mapping of the tumor microenvironment using integrated single-cell, spatial and in situ analysis. Nature Communications. 2023;14:8353. doi:10.1038/s41467-023-43458-x.

29. Marconato L, Vierdag WM, Yamauchi K, Mah C, Crowell H, Blampey Q, et al. 1st SpatialData Developer Workshop. 2024;doi:10.37044/osf.io/8ck3e.