A Bulk Tissue eFP Browser for Single Cell Data for ePlant

Steven Qiao

Student Number: 1009842164

BCB330Y Project Proposal

Principal Investigator: Professor Nicholas J. Provart

Supervisor: Vincent Lau

Oct 14, 2025

Introduction and Background

As one of the most comprehensive repositories for bulk RNA sequencing (RNA-seq), the Bio-Analytic Resource for Plant Biology (BAR) has witnessed and accompanied the excellence and endeavors of countless academics and researchers over the last two decades. Because of its integration into the browser, it has been applying an extensive set of microscopic plant biology data to a user-friendly viewing interface with reference to inter- and intra-omic interactions, gene expression variation, and sequencing information. For example, some of the viewers embedded in the BAR include ePlant viewers, expression anglers, and many structural viewers. Moreover, ever since its conception, the BAR has developed ePlant viewers for a multitude of plants such as potatoes, barley, and tomatoes. From its early toddling steps to its current prosperity, the model organism *Arabidopsis thaliana* has played the most foundational and robust role in the BAR's data acquisition and annotation representation (Sullivan et al., 2024).

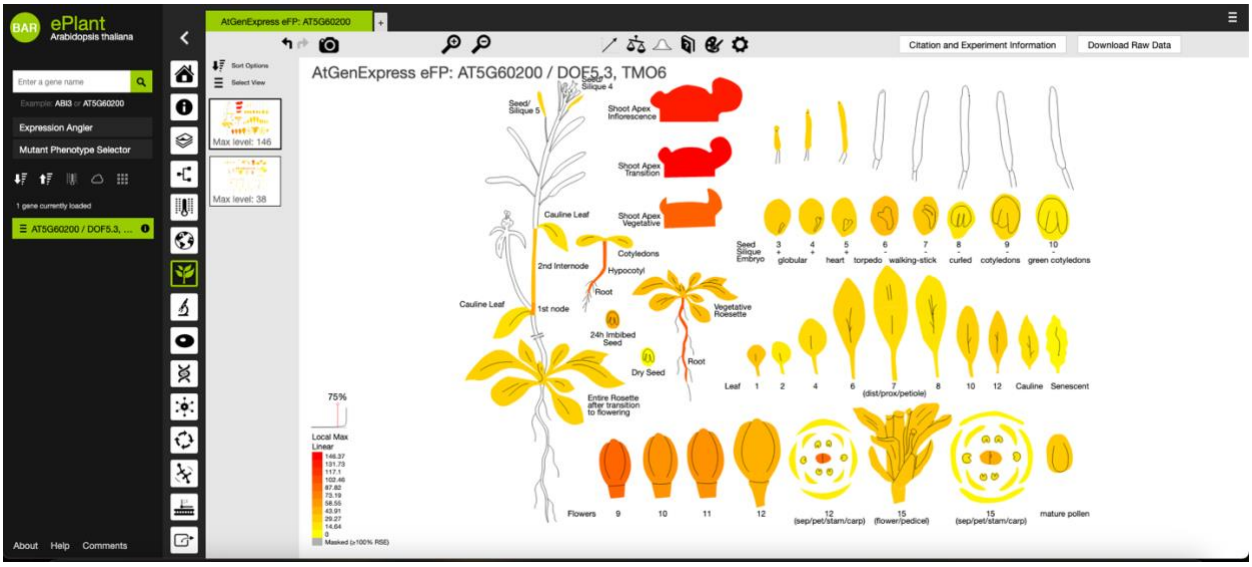


Figure 1. ePlant Bulk view of gene AT5G60200 in Arabidopsis (Sullivan et al., 2024).

Due to its web-based nature, the BAR is currently constructed in HTML, CSS, JavaScript, and jQuery (Waese et al., 2017). Recently, in an era of rapidly progressing bioinformatical technology, the BAR has been actively incorporating artificial intelligence and machine learning for predicting protein structure (AlphaFold) and productively comparing gene expression across multiple plant species (GAIA) (Sullivan et al., 2024). Nonetheless, while the BAR still predominantly excels in traditional bulk RNA tissue gene expression display, there is an imminent need to construct novel viewers to adapt to the high-resolution single-cell RNA datasets, which are only 14 years of age currently. This need to implement scRNA datasets is due to the research gap that exists between bulk tissue averages and cellular-level resolution. Bulk RNA-seq conceals cell-type-specific patterns by averaging expression across millions of cells, while scRNA-seq struggles to map cell types to tissue locations. This disconnection prevents researchers from efficiently connecting cellular identities with larger tissue regions (Haque et al., 2017).

As a result of bioinformatical advancement, scRNA-seq measures each individual cell within a sample tissue, thereby involving the process of extracting RNA from every single cell. Additionally, this preserves their heterogeneity at a cellular level while keeping close track of the original cell types. Consequently, experimenters obtain an enormous data profile of thousands or even millions of cells and features, including gene expression levels tailored to each cell, cell types, the distribution of cell types within a larger given tissue, along with many other features (Li and Wang, 2021).

In contrast, RNA-seq measures the average gene expression across all cells within a tissue region. In the process, the entire tissue will be mashed up, and RNA sequences, even those from different cell types, are all mixed. Unfortunately, there exist many potential sources of information loss by merely computing the overall average expression. Experimenters will lose crucial information about variation across individual cells, identification of cell populations with particularly high expressions (outliers), and the specific cell types within the massive tissue sample (Li and Wang, 2021).

By visualizing both bulk and single-cell data side-by-side, researchers will benefit immensely from their combined strengths. In terms of significance, this project addresses a critical need in plant biology research by establishing a bridge between single-cell and bulk tissue expression data within ePlant's established framework. By developing a bulk tissue eFP browser for single-cell data, this work will enable researchers to visualize gene expression patterns within macroscopic tissue contexts, make conclusions about cell-type gene composition and function, and extend ePlant's utility for communities of plant researchers working with single-cell datasets. The proposed viewer will not only enhance ePlant's existing infrastructure by introducing novel functionality to handle single-cell data but also maintain consistency with the platform's easy-to-navigate user experience.

Hypothesis

Integrating scRNA-seq data into ePlant's tissue eFP viewer will assist researchers to identify cell-type-specific gene expression patterns that are difficult to capture when merely using bulk RNA-seq approaches alone.

Objectives and Goals

Goal 1: Select the optimal visualization technique for ePlant and develop a data processing module

- ⇒ For display on ePlant, choose the optimal visualization approach among JavaScript, R, and CellxGene
- ⇒ Group individual single-cell profiles into cell-type clusters and compute average gene-expression values for each cluster to enable tissue-level representation
- ⇒ Conversions and processing need to occur prior to visualizations. If R is the selected tool, the .h5ad file needs to be processed with Bioconductor's Zellkonverter (R package) and multiple other tools; otherwise, it is necessary for the .h5ad file to be converted into JSON for proper pairing with JavaScript
- ⇒ Minimize input file size through compression to reduce server overhead and improve loading time

Goal 2: Create interactive bulk tissue-level eFP viewer for single-cell data on the BAR

- ⇒ Design and implement a cell-type selection interface within existing ePlant structure
- ⇒ Allow the user to upload their custom SVG for cell-type clustering and heatmap display
- ⇒ Support toggling between the bulk tissue view and the individual cell-type display to view expression heterogeneity at different resolution layers

In goal 1, when it comes to visual displays, the three candidates are JavaScript, R, and CellxGene iframes. Below is the comparison of these three types with their strengths and weaknesses enumerated.

Criteria	JavaScript	R	CellxGene
ePlant Integration	Direct Integration	Requires Server	Requires iframe
Tissue Anatomy	YES	YES	NO
Loading Custom SVG	YES	YES	NO
Complexity/Speed	Fast and interactive	Slower (server call)	Slower (localhost)
Maintenance	Self	Self	CZI
Open Source	Yes	Yes	Yes

Table 1. Comparison of visualization frameworks for ePlant single-cell integration (Abdulla et al., 2024).

In goal 2, the main procedures involved are more directly engaged in the innovative modifications of the existing BAR and ePlant platforms. If ePlant stores gene expression data that corresponds with the user's custom SVG, the platform can map existing data onto the new image through variations in color gradients. This feature augments the platform's flexibility and broadens its applicability by allowing researchers to visualize gene expression in diverse plant species or tissue regions whose eFP diagrams are unavailable in BAR.

Methods

Data Acquisition and Processing

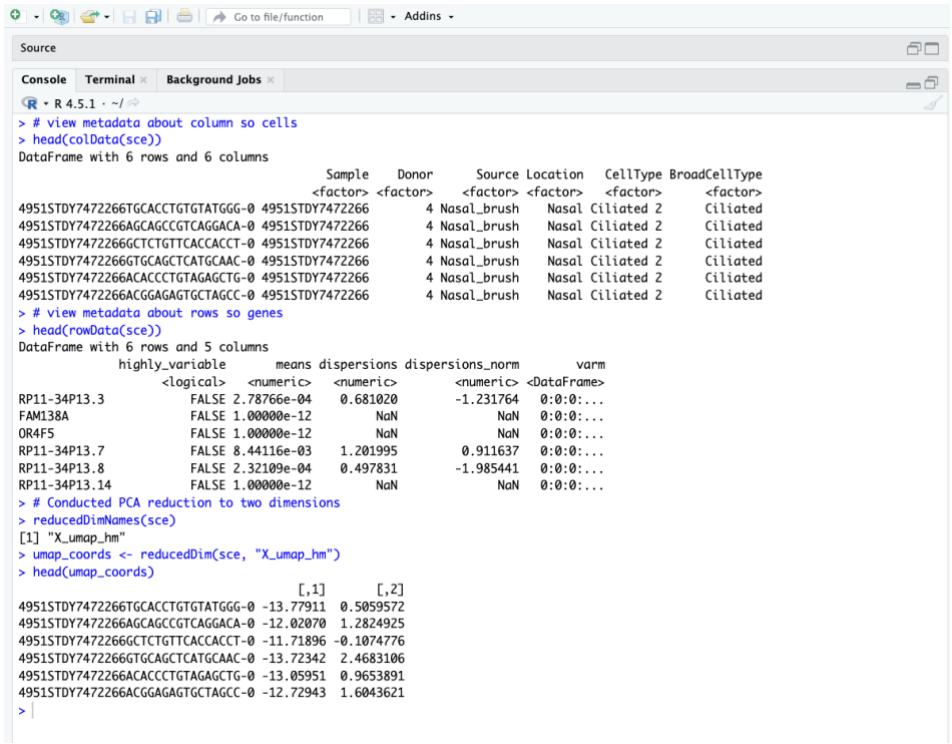
Up to this point, I have been mostly experimenting with an *Arabidopsis* .h5ad dataset derived from the research article published by Dr. Illouz-Elias et.al (2025). Nonetheless, in the earlier stages of the project, due to the massive size of the *Arabidopsis* .h5ad file (4.38 GB), I began my initial explorations with a smaller COVID-19 h5ad dataset from the nasal tissue (Sungnak et al., 2020). Since the .h5ad file format is closely associated with Python and R, I selected R as my primary workspace because of its statistical capabilities and strength in processing datasets. Furthermore, as a student with two years of prior experience with R, I applied my knowledge to built-in R packages including ggplot2, Bioconductor's Zellkonverter, and other visualization tools. Through gradually understanding these libraries, I successfully dissected the *Arabidopsis* .h5ad file into several crucial components (Virshup et al., 2023):

⇒ Cell × Gene matrix: All cells (rows) are paired with all genes (columns) to form a large matrix in which each entry is a number quantifying the expression intensity of a specific gene in a specific cell.

⇒ Metadata about the cells: This section includes the cell source (organism and tissue of origin) and the predicted cell type of each cell in the matrix, based on expression levels of various genes.

⇒ Metadata about the genes: This section contains statistical distribution of each gene such as Means (average expression of one gene across all cells) and Dispersions (the numerical value for the variability across cells).

⇒ X_umap_hm: Coordinates for cells that are used to plot the 2D UMAP for visualization



```
> # view metadata about column so cells
> head(colData(sce))
DataFrame with 6 rows and 6 columns
                                          Sample    Donor      Source Location   CellType BroadCellType
                                        <factor> <factor>    <factor> <factor>   <factor>      <factor>
4951STDY7472266TGCACCTGTGTATGGG-0 4951STDY7472266        4 Nasal_brush    Nasal Ciliated 2      Ciliated
4951STDY7472266AGCAGCCGTCAGGACA-0 4951STDY7472266        4 Nasal_brush    Nasal Ciliated 2      Ciliated
4951STDY7472266GCTCTGTTCACCACCT-0 4951STDY7472266        4 Nasal_brush    Nasal Ciliated 2      Ciliated
4951STDY7472266GTGCAGCTCATGCAAC-0 4951STDY7472266        4 Nasal_brush    Nasal Ciliated 2      Ciliated
4951STDY7472266ACACCCTGTAGAGCTG-0 4951STDY7472266        4 Nasal_brush    Nasal Ciliated 2      Ciliated
4951STDY7472266ACGGAGAGTGCTAGCC-0 4951STDY7472266        4 Nasal_brush    Nasal Ciliated 2      Ciliated
> # view metadata about rows so genes
> head(rowData(sce))
DataFrame with 6 rows and 5 columns
             highly_variable       means dispersions dispersions_norm        varm
                   <logical>   <numeric>   <numeric>        <numeric> <DataFrame>
RP11-34P13.3           FALSE 2.78766e-04    0.681020        -1.231764   0:0:0:...
FAM138A                FALSE 1.00000e-12         NaN              NaN   0:0:0:...
OR4F5                  FALSE 1.00000e-12         NaN              NaN   0:0:0:...
RP11-34P13.7           FALSE 8.44116e-03    1.201995         0.911637   0:0:0:...
RP11-34P13.8           FALSE 2.32109e-04    0.497831        -1.985441   0:0:0:...
RP11-34P13.14          FALSE 1.00000e-12         NaN              NaN   0:0:0:...
> # Conducted PCA reduction to two dimensions
> reducedDimNames(sce)
[1] "X_umap_hm"
> umap_coords <- reducedDim(sce, "X_umap_hm")
> head(umap_coords)
                                      [,1]      [,2]
4951STDY7472266TGCACCTGTGTATGGG-0 -13.77911  0.5059572
4951STDY7472266AGCAGCCGTCAGGACA-0 -12.02070  1.2824925
4951STDY7472266GCTCTGTTCACCACCT-0 -11.71896 -0.1074776
4951STDY7472266GTGCAGCTCATGCAAC-0 -13.72342  2.4683106
4951STDY7472266ACACCCTGTAGAGCTG-0 -13.05951  0.9653891
4951STDY7472266ACGGAGAGTGCTAGCC-0 -12.72943  1.6043621
>
```

Figure 2. Cell metadata, gene metadata, and UMAP coordinates loaded with R Zellkonverter

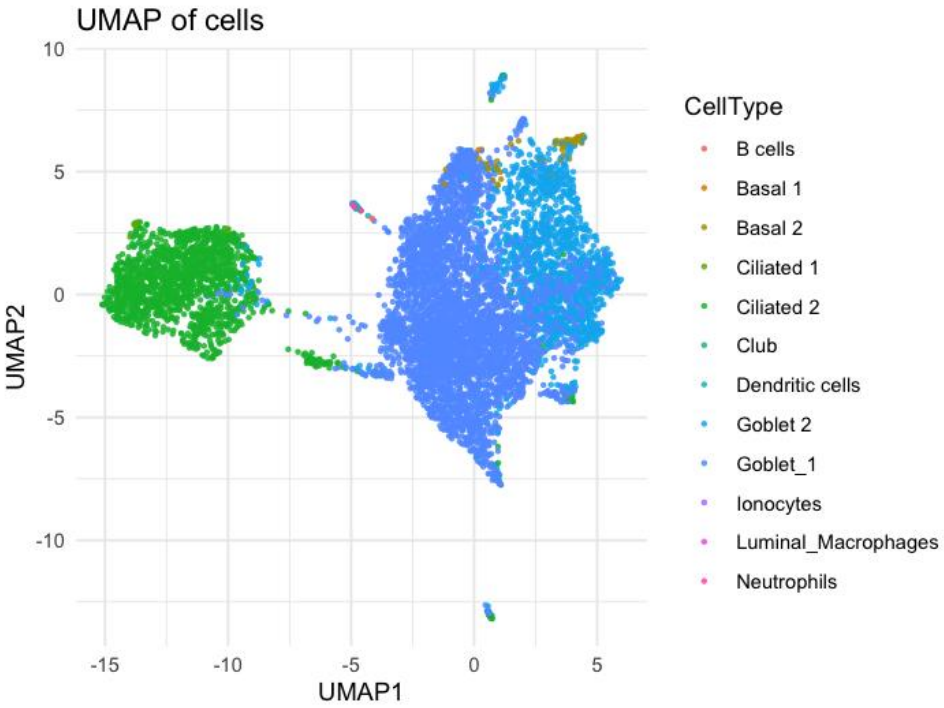


Figure 3. Color gradient overlayed onto UMAP based on nasal tissue cell types

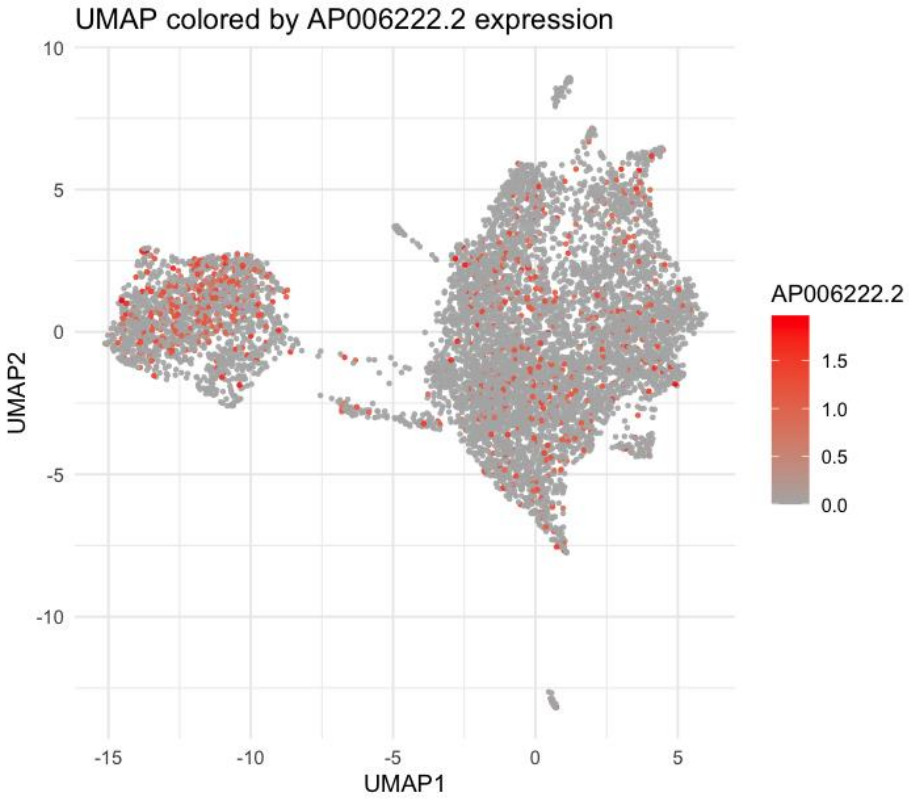


Figure 4. UMAP based on expression level of AP006222.2 in nasal tissue

Initially, I used Zellkonverter's readH5AD function to transform the Python-based .h5ad file into an R-compatible SingleCellExperiment (SCE) dataset for later use (Zappia and Lun, 2025). Out of the three candidates, R proved very efficient for UMAP plotting because of X_umap_hm's compatibility with the built-in ggplot functions. More importantly, the existence of the cell type metadata allowed me to sum all cell expression profiles and compute the average expression by dividing the sum by the total number of cells. The resulting vector represented the average expression level for every gene within a specific cell type, which satisfied the requirements outlined in goal 1.

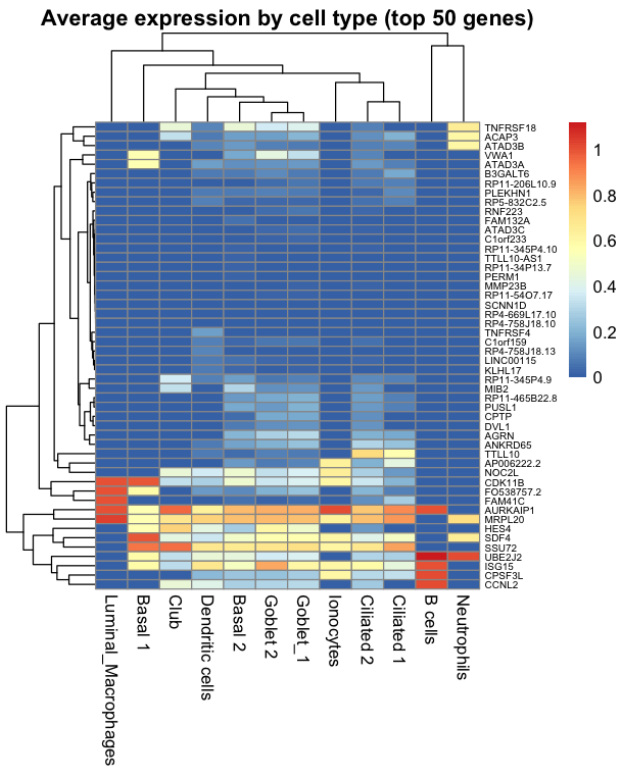


Figure 5. Average expression of the top 50 most active genes within each cell type

Exploration and Evaluation of Visualization Frameworks

Subsequent to establishing the data processing pipeline, I evaluated different visualization frameworks for their suitability with ePlant integration. Upon completing the UMAP display with Zellkonverter, I investigated the functionalities of CellxGene. Using the *Arabidopsis* dataset as the file parameter, I launched the program via localhost. The critical technical limitation with CellxGene was the lack of tissue anatomy diagrams and options to upload custom SVGs. Unfortunately, unlike R and JavaScript, CellxGene is an environment established by the Chan Zuckerberg Initiative (CZI), so I cannot flexibly modify it to suit the project requirements. While CellxGene is extremely useful for its clustering and UMAP visualizations, the overhead and potential response delays from embedding a partially functional framework into ePlant could not justify its use for this project.

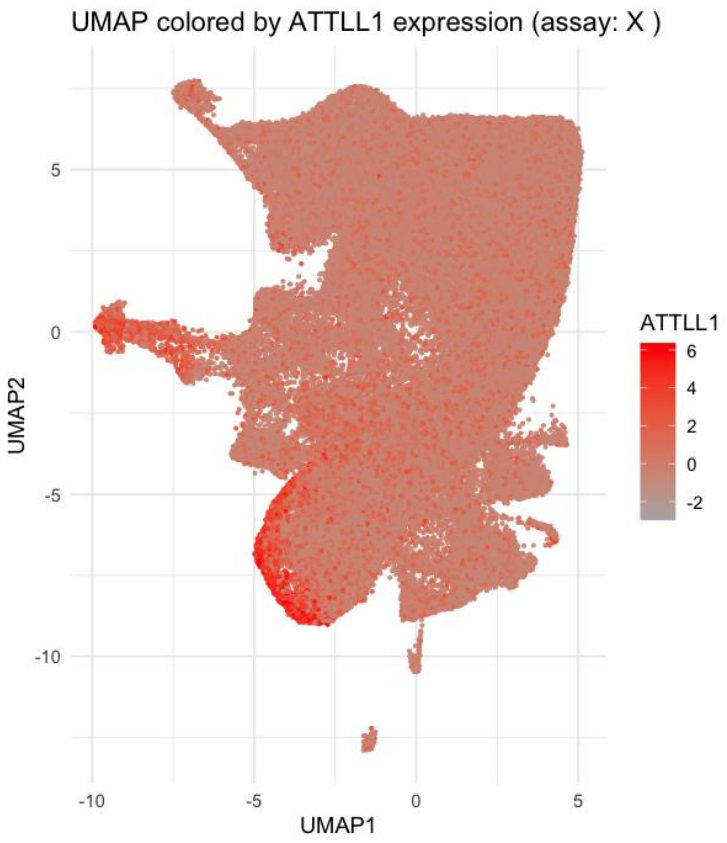


Figure 6. UMAP based on gene expression of ATTLL in sample Arabidopsis h5ad dataset

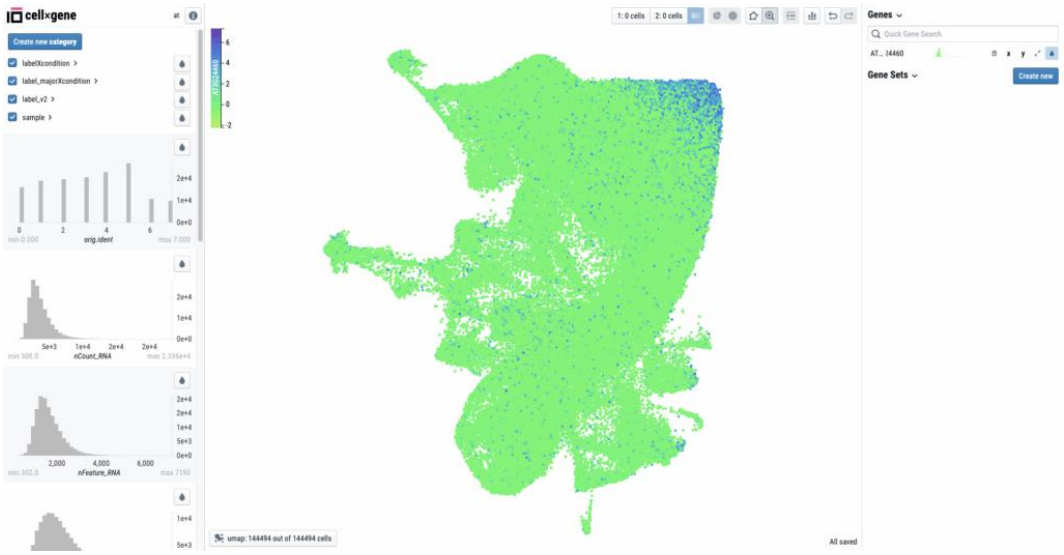


Figure 7. Arabidopsis h5ad overlayed onto CellxGene (Abdulla et al., 2024)

Integration with ggPlantmap and SVG Customization

The most recent work I completed focused on overlaying gene expression data onto custom SVGs. Centered around *Arabidopsis*, ggPlantmap is an R package that facilitates the mapping of plant tissue expression diagrams. Before beginning the customization process, I first attempted to use my existing SCE *Arabidopsis* dataset to overlay the expression data onto the empty SVG templates stored within ggPlantmap. Since there was a discrepancy between cell names in the h5ad file and the ggPlantmap cell names, this portion involved mechanical hard-coding, as I matched and mapped names from one format to the other. Rather than hard coding, the research team could develop an interactive mapping interface in which the set of h5ad cell types is next to an empty template ggPlantmap SVG, so that users can drag and drop names onto cells to visualize the process. Subsequently, ePlant can save the mapping information for future use.

```
# Create mapping from your cell types to ggPlantmap ROI names
celltype_mapping <- tribble(
  ~Identity, ~ROI_mapped,
  "0: Mesophyll_1", "Parenchima.palisade",
  "1: Mesophyll_2", "Parenchima.palisade",
  "2: Mesophyll_3", "Parenchima.palisade",
  "3: Mesophyll_4", "Parenchima.palisade",
  "4: Mesophyll_5", "Parenchima.sponge",
  "5: Epidermal_1", "epidermis.adaxial",
  "6: Immune active", "Parenchima.sponge",
  "7: Mesophyll_6", "Parenchima.sponge",
  "8: Guard_1", "epidermis.stomata",
  "9: Mesophyll_7", "Parenchima.sponge",
  "10: Defense state", "Parenchima.sponge",
  "11: Vascular_1", "vascularbundle.xylem",
  "12: Vascular_2", "vascularbundle.phloem",
  "13: Epidermal_2", "epidermis.abaxial",
  "14: Phloem companion", "vascularbundle.phloem",
  "15: Stress responsive", "Parenchima.sponge",
  "16: Phloem Parenchyma", "vascularbundle.phloem",
  "17: Sieve element_responsive", "vascularbundle.phloem",
  "18: Dividing_1", "Parenchima.palisade",
  "19: Guard_2", "epidermis.stomata",
  "20: Dividing_2", "Parenchima.sponge",
  "21: Epidermal_3", "epidermis.abaxial",
  "22: Metabolic stress state", "Parenchima.sponge",
  "23: Hydathode", "epidermis.abaxial",
  "24: Trichome", "epidermis.adaxial",
  "25: Myrosin", "Parenchima.palisade",
  "26: Sugar metabolic state", "Parenchima.sponge"
)
```

Figure 8. Mapping from h5ad cell names (left) to ggPlantmap ROI names (right)

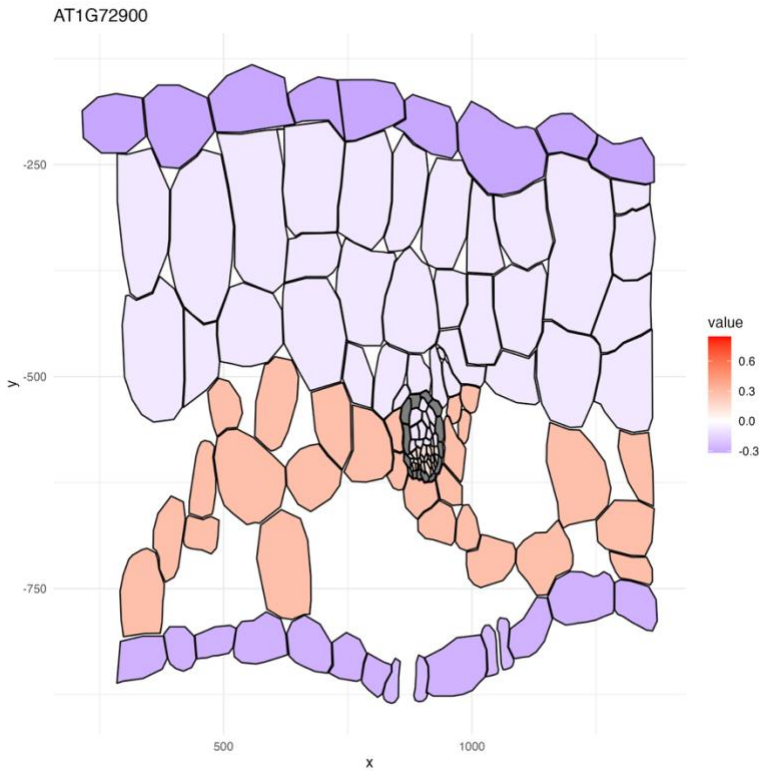


Figure 9. Heatmap of expression level of gene AT1G72900 overlayed onto Arabidopsis leaf cross-section SVG

After consolidating the fundamental application techniques with the basic ggPlantmap code, I created my own blank SVG using Inkscape and attempted to produce a colorful expression map in RStudio. Although they are both SVGs, ggPlantmap has rigorous requirements for the syntax of SVG XML code that is inputted. ggPlantmap SVGs can be generated directly using Icy by tracing the paths along the edges of photos of raw *Arabidopsis* tissues. To make ePlant more versatile and applicable, I developed an XML converter that allows users with standard SVG formats to bypass the cumbersome and time-consuming Icy converter. Built with React, this converter takes in standard SVG XML as input and converts it into Icy-compatible XML suitable for ggPlantmap functionality. Eventually, the built-in plot function from ggPlantmap colors the customized SVG by cell type clusters.

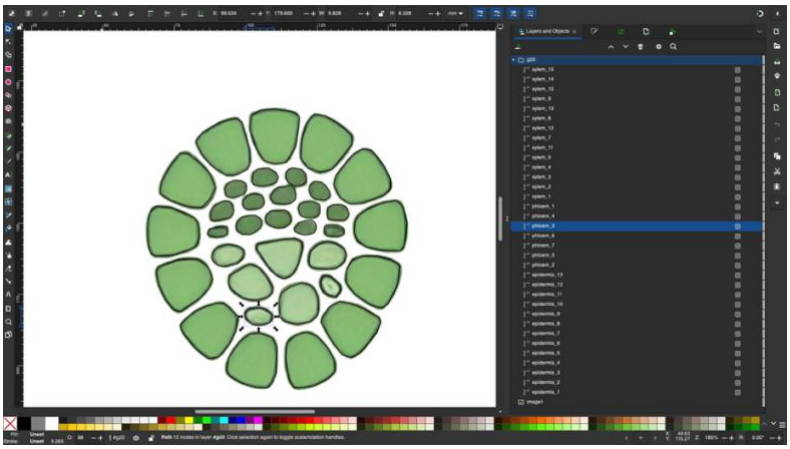


Figure 10. Defining paths 4to convert from PNG to SVG in Inkscape (Illouz-Eliaz et al., 2025)

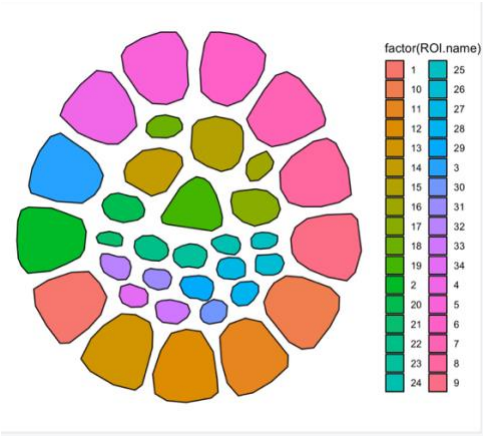


Figure 11. Custom SVG with cell-type clustering plotted in R *using ggplot*

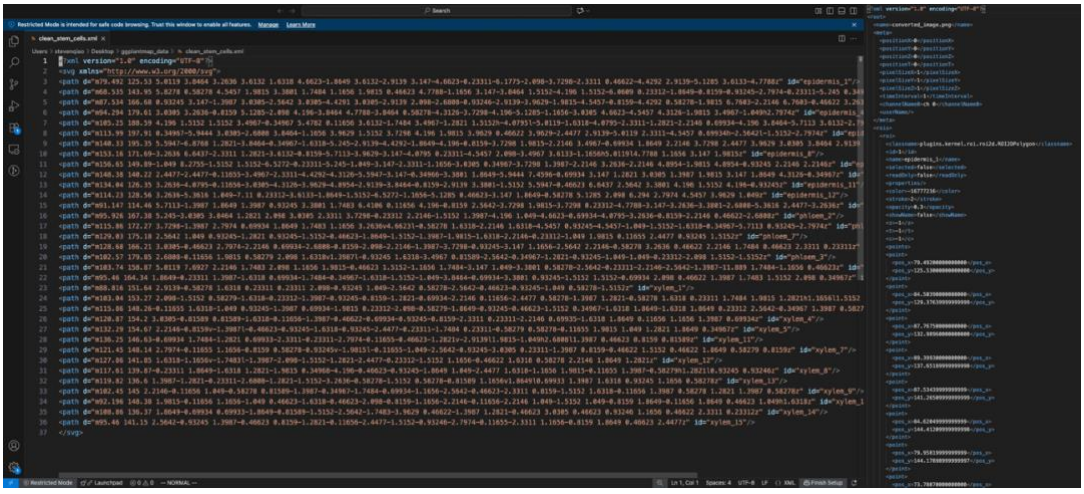


Figure 12. Standard SVG XML format (left) versus ggPlantmap ICY XML format (right)

Discussion

After discreet consideration of all the determining factors, JavaScript might be the ideal tool to pivot towards because it maximizes the achievement of our goal. Moving forward, the research team should consider delving into the direction of compressing and converting from h5ad to JSON, as this conversion and compression into JSON will significantly minimize input file size and runtime burden. Unlike R, JavaScript does not require the use of a back-end server, and JavaScript is more directly incorporated within ePlant's current architecture. Therefore, the application of JavaScript might lead to faster load times and more responsive interactions.

Looking towards our pre-eminent counterpart Root Cell Atlas, their application of React with their interactive UMAP provides valuable insight for ePlant to establish our own React interface to discover more comprehensive and sophisticated features. For the toggle functionality and cell-type selection, the research team should consider the cost-effectiveness of utilizing React in ePlant because we need to consider the potential architectural burden.

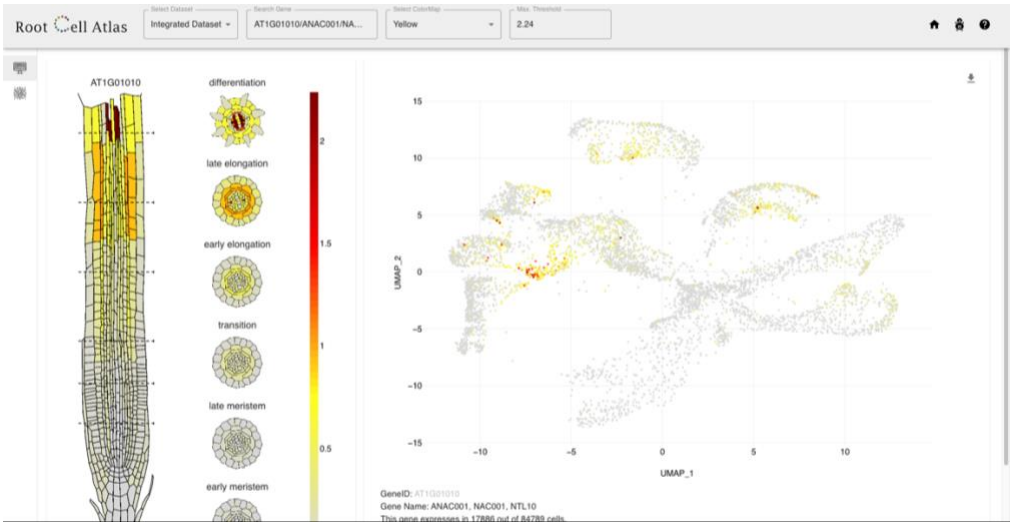


Figure 13. Root Cell Atlas displaying the interactive UMAP and Arabidopsis root anatomical diagram (Mironova, 2022).

Earlier in the research process, the development of a .h5ad to zarr converter has reduced the overall file size by ~50%, yet zarr file occupies significantly more RAM space and requires the use of Vitessce, which is nearly the identical CellxGene counterpart designed specifically for zarr files. In consideration of its limitations and sophisticated dependency setup, zarr compression does not offer any significant advantages over JSON or R. Hence, there is no need to explore further down the line with zarr file compressions as we are shifting focus towards JSON structure.

```
> print(res_sce[, c("expression","min","median","itr/sec","mem_alloc","gc")])
# A tibble: 2 x 6
  expression       min   median `itr/sec` mem_alloc gc
  <bch:expr>  <bch:tm> <bch:tm>     <dbl> <bch:byt> <list>
1 zarr_to_sce    19.1s    20.1s    0.0493    14.8GB <tibble [3 x 3]>
2 h5ad_to_sce    19.1s    20.1s    0.0493     4.5GB <tibble [1 x 3]>
>
```

Figure 14. Comparison between zarr and h5ad load time; mem_alloc indicates RAM usage

For the goals involving the direct modification of the eFP viewer on the BAR, research team members need to dedicate more time and effort to implement interactive and appealing CSS elements and components. Building upon the current work as foundation, the research team should focus on establishing the cell-type selection interface. While ensuring the visual coherence, this interface needs to allow researchers to toggle between tissue-level and cell-type-specific views. Implementing custom SVG upload functionality will extend the ePlant's applicability beyond existing anatomical diagrams. In order to attain this objective, the integration layer connecting ggPlantmap, R/JavaScript processing, and ePlant will require careful architectural consideration to ensure smooth operation.

Project Timeline

Weeks 9-11: Finalize Visualization Approach	⇒ Decide between R and JavaScript ⇒ Convert .h5ad to JSON format ⇒ Test compression results, memory usage, and loading speed
Weeks 12-15: Build Cell-Type Selection Interface	⇒ Create dropdown menu or search function for selecting cell types ⇒ Ensure smooth compatibility with ePlant's current design ⇒ Add toggling and side-by-side view between tissues and single cells
Weeks 16-19: Work on Custom SVG Upload	⇒ Build interface for mapping cell names to SVG regions ⇒ Let users drag-and-drop to match names ⇒ Save mappings for future usage ⇒ Start with brainstorming for the final report and presentation by constructing rough drafts
Weeks 20-24: Add Visualization Features and Wrapping Up	⇒ Get color gradients to work ⇒ Implement gene search functionality ⇒ Add zoom-in and zoom-out feature ⇒ Test with different *Arabidopsis* tissues ⇒ Polish and complete final report ⇒ Prepare for final presentation

References

Abdulla, S. *et al.* (2024) 'CZ CELLxGENE Discover: a single-cell data platform for scalable exploration, analysis and modeling of aggregated data,' *Nucleic Acids Research*, 53(D1), pp. D886–D900. https://doi.org/10.1093/nar/gkae1142.

Haque, A. *et al.* (2017) 'A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications,' *Genome Medicine*, 9(1). https://doi.org/10.1186/s13073-017-0467-4.

Illouz-Eliaz, N. *et al.* (2025) 'Drought recovery in plants triggers a cell-state-specific immune activation,' *Nature Communications*, 16(1). https://doi.org/10.1038/s41467-025-63467-2.

Jo, L. and Kajala, K. (2024) 'ggPlantmap: an open-source R package for the creation of informative and quantitative ggplot maps derived from plant images,' *Journal of Experimental Botany*, 75(17), pp. 5366–5376. https://doi.org/10.1093/jxb/erae043.

Li, X. and Wang, C.-Y. (2021) 'From bulk, single-cell to spatial RNA sequencing,' *International Journal of Oral Science*, 13(1). https://doi.org/10.1038/s41368-021-00146-0.

Mironova, V. (2022) *Root Cell Atlas: Single-cell transcriptomics data visualization*. Radboud University. Available at: https://rootcellatlas.org/ (Accessed: 21 October 2025).

Sullivan, A. *et al.* (2024) '20 years of the Bio-Analytic Resource for Plant Biology,' *Nucleic Acids Research*, 53(D1), pp. D1576–D1586. https://doi.org/10.1093/nar/gkae920.

Sungnak, W. *et al.* (2020) 'SARS-CoV-2 entry factors are highly expressed in nasal epithelial cells together with innate immune genes,' *Nature Medicine*, 26(5), pp. 681–687. https://doi.org/10.1038/s41591-020-0868-6.

Virshup, I. *et al.* (2023) 'The scverse project provides a computational ecosystem for single-cell omics data analysis', *Nature Biotechnology*, 41, pp. 604–606. https://doi.org/10.1038/s41587-023-01733-8

Waese, J. *et al.* (2017) 'ePlant: Visualizing and exploring multiple levels of data for hypothesis generation in plant biology,' *The Plant Cell*, 29(8), pp. 1806–1821. https://doi.org/10.1105/tpc.17.00073.

Zappia, L. and Lun, A. (2025) *Zellkonverter:* Conversion Between scRNA-seq Objects. R package version 1.18.0. Bioconductor. Available at: https://bioconductor.org/packages/zellkonverter https://doi.org/10.18129/B9.bioc.zellkonverter (Accessed: 21 October 2025).