Zebrafish Dataset Dashboard

Information Visualization Course Report

Giovanni Vincenzo Cavallo
00172510

December 12, 2025

1 Introduction

This report details the implementation of a visualization tool designed to analyze a specific dataset concerning the tissues of developing Zebrafish(*Danio rerio*). The primary goal of this data is to study the process of cell differentiation during development. The dataset, derived from the research of Wagner et al. ADD BIBLIOGRAPHY NUMBER, is substantial in scale, comprising 63,530 observations (cells) and 1,002 variables (genes, cell type, hours post-fertilization).

To support the visual analysis of this high-dimensional data, the software leverages two key categorical attributes provided with the dataset:

- **Type:** A classification of cells based on their biological function and tissue origin.
- **Hours Post-fertilization:** A temporal indicator marking the time elapsed after fecundation, essential for observing developmental trajectories.

The following sections describe the design choices, algorithms, and interactive features implemented to handle the scale of this dataset effectively, with a guide to the use of the app.

2 Data Pre-processing:

The data used for this project is the data coming from the pre-processing of the dataset used for the paper. Some of the columns of the processed data weren't clearly explained and didn't look meaningful in their original state.
Therefore, before starting with the actual app, formatting data is required to more easily extrapolate meaningful information from it.
Specifically, the columns `R`, `G`, and `B` were substituted with the *hours post fertilization* attribute, which was processed as a categorical variable following the pattern provided by the authors. Additionally, the `type` column, originally reported as integer values, was converted into a categorical format to explicitly represent the distinct *cell types*, also in accordance with the authors.

3 Algorithms Choices:

Dimensionality Reduction

- **t-SNE**: I chose t-SNE as the primary visualization approach, aligning with the methodology used in the original paper for this dataset. As a well-established manifold learning technique, t-SNE effectively projects high-dimensional data onto a 2D plane, preserving local neighborhoods and revealing spatial clusters that variance-based methods might overlook. Although standard practice often suggests performing Principal Component Analysis (PCA) prior to t-SNE to reduce noise, empirical testing (documented in the `testTSNE` notebook) demonstrated that applying t-SNE directly to the full 1000-gene expression space yielded the clearest cluster separation for this specific dataset.

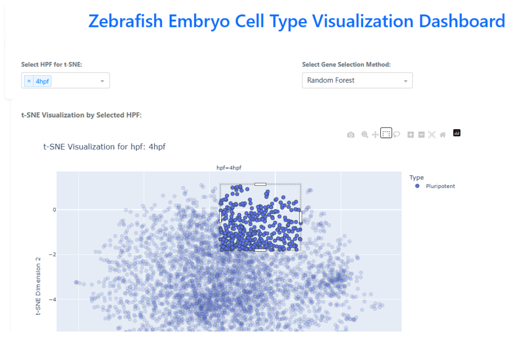
- **PCA**: Principal Component Analysis (PCA) was employed as a secondary dimensionality reduction method. By compressing the dataset's features into synthetic principal components that capture the maximum variance, PCA served a dual purpose: it provided an alternative visualization space—specifically utilized for the DBSCAN cluster plots—and acted as a computational preprocessing step to improve the efficiency and performance of the clustering algorithms (K-Means, Agglomerative Clustering, and DBSCAN).

Cluster Methods

- **K-Means**: K-Means was implemented in accordance with the project guidelines. It serves as a robust baseline for partitioning the data into distinct subgroups by minimizing the variance within clusters through centroid-based optimization.

- **Agglomerative Clustering**: I utilized Agglomerative Hierarchical Clustering (sklearn) to analyze the structural relationships and distances between different cell groups. This method is particularly insightful for biological data, as the resulting hierarchy can reflect the developmental lineage or "history" of cell types based on their cluster of origin. In the visualization dashboard, the resulting dendrogram serves as an interactive control, allowing users to cross-filter the dataset by clicking on specific leaves to inspect the corresponding cells.

- **DBSCAN**: DBSCAN was selected as the final clustering method to provide density-based insights, offering a distinct alternative to the centroid-based approach of K-Means. Unlike K-Means, DBSCAN groups points based on local density and can identify outliers. Extensive parameter tuning for the epsilon (ε) value was conducted (documented in the `dbscan` notebook) to determine optimal density thresholds. To facilitate exploration, the dashboard allows users to adjust the ε parameter dynamically and visualize the resulting changes in cluster formation. Based on preliminary testing, the algorithm is computed using the first two Principal Components to ensure computational efficiency.


Zebrafish Embryo Cell Type Visualization Dashboard

Select HPF for t-SNE: 4hpf

Select Gene Selection Method: Random Forest

t-SNE Visualization by Selected HPF:

Figure 1: t-SNE Visualization of Zebrafish Data
A t-SNE plot visualizing cell clusters, filterable by hpf.

4 Dashboard User Guide

The dashboard application is designed to run on a local machine with a Python environment installed.

Installation and Setup As detailed in the `README` file located in the code directory, the raw dataset must first be uploaded into the `data` folder (adjacent to the `data_generator.py` script).

Once the data is in place, the user must execute the `initialize.py` file. This script performs two critical functions:

1. It installs all necessary dependencies as specified in `requirements.txt`.
2. It pre-processes the data. Please note that this step may take some time due to computational bottlenecks, specifically:
   - Formatting the entire dataset into an optimized structure.
   - Computing the t-SNE embeddings (the most time-consuming process).

Once the initialization is complete and the success message appears, the user can launch the application by running `app.py` and accessing the dashboard via their local server.

Part 1: t-SNE Visualization and Gene Selection The first section of the dashboard presents a t-SNE plot of the data, as shown in Figure 1. This visualization can be filtered by hours post-fertilization (hpf).

A dropdown menu allows the user to select the method for identifying meaningful genes:

- **Random Forest**: Selects the top 5 genes determined to be most predictive of cell type using Random Forest feature importance.
- **ANOVA**: Selects the top 5 genes that show the highest variance in mean expression across different cell types.
- **Intersection**: Selects the top 5 genes identified by both methods, offering a robust set of markers.

This selection directly impacts the secondary functionality of the t-SNE plot: cross-filtering. Users can select specific cells (points) on the plot using the Plotly box selection tool to investigate their gene expression levels.

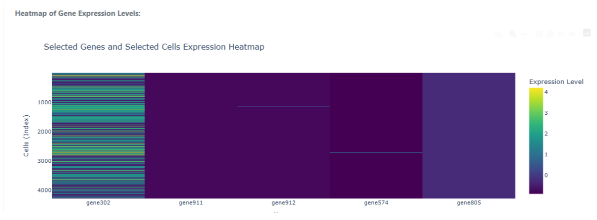
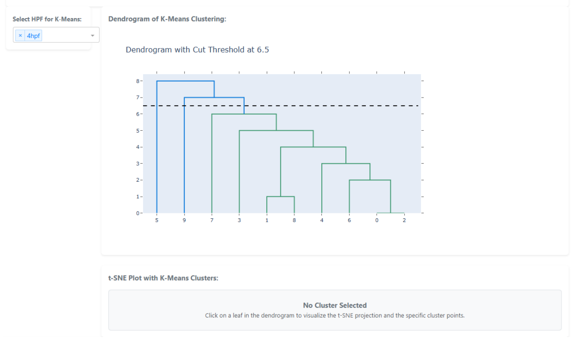

Heatmap of Gene Expression Levels:
Selected Genes and Selected Cells Expression Heatmap
Cells (Index)
1000
2000
3000
4000
gene302
gene911
gene912
gene574
gene805
Expression Level
4
3
2
1
0

Figure 2: Gene Expression Heatmap
Heatmap showing expression levels of selected genes for selected cells.

Part 2: Gene Expression Heatmap Below the t-SNE plot, the dashboard generates a heatmap (Figure 2) corresponding to the user's selection.

This visualization displays the expression levels of the selected genes for the selected cells. The color scale indicates relative expression, with distinct colors for over-expression and under-expression. Hovering over the heatmap reveals the specific index of the cells, allowing for detailed inspection of individual values.

Part 3: Dendrogram Plot and K-Means The subsequent section of the application enables the visualization of a dendrogram, constructed using the **AgglomerativeClustering** function from the **scikit-learn** library.


Select HPF for K-Means:
× 4hpf
Dendrogram of K-Means Clustering:
Dendrogram with Cut Threshold at 6.5
8
7
6
5
4
3
2
1
0
5
9
7
3
1
8
4
6
0
2
t-SNE Plot with K-Means Clusters:

No Cluster Selected
Click on a leaf in the dendrogram to visualize the t-SNE projection and the specific cluster points.

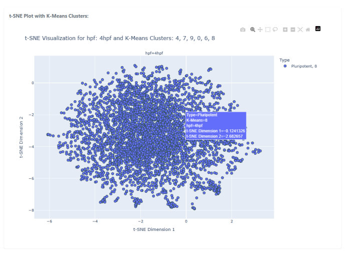
Figure 3: Dendrogram plot showing hierarchical relationships between K-Means clusters.

As illustrated in Figure 3, the leaves of the dendrogram represent clusters computed via the K-Means algorithm. The optimal number of clusters (k) was determined by analyzing the Silhouette Score across a broad range of values, as documented in the *K-Means* notebook.

The application of hierarchical clustering, interpreted using the vertical cut method, identified three primary clusters. The visualization color-codes these groups to highlight structure: meaningful clusters (separated by the largest distance) are shown in blue, while the remaining clusters appear in orange.

Certain clusters correspond exclusively to specific cell types, suggesting that the dendrogram can effectively model the temporal development and lineage of these cells. To facilitate this analysis, clicking on the dendrogram leaves (cluster IDs) highlights the corresponding cluster on a secondary t-SNE plot located below, as depicted in Figure 4.

Users can select multiple clusters simultaneously for comparative visualization. Notably,


t-SNE Plot with K-Means Clusters:
t-SNE Visualization for hpf: 4hpf and K-Means Clusters: 4, 7, 9, 0, 6, 8
hpf=4hpf
Type
Pluripotent, 8
t-SNE Dimension 2
t-SNE Dimension 1
0
−2
−4
−6
−8
−6
−4
−2
0
2

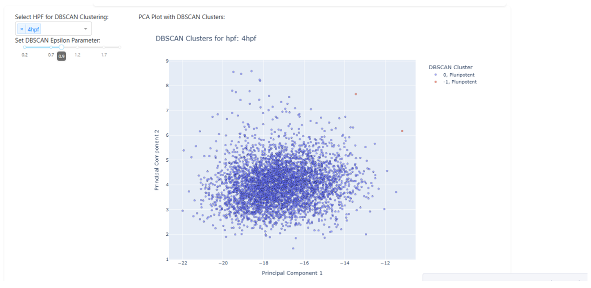
Type=Pluripotent
K-Means=8
hpf=4hpf
t-SNE Dimension 1=−0.1241326
t-SNE Dimension 2=−2.682657

Figure 4: Secondary t-SNE plot, filtered by the selected cluster.

some clusters appear only at specific developmental stages (`hpf`). Deselection is achieved by clicking the leaf a second time.

Part 4: DBSCAN and PCA The final section of the application focuses on density-based clustering and Principal Component Analysis (PCA).

DBSCAN was selected as the density-based clustering method, as shown in Figure 5. Unlike previous sections, this visualization plots the clusters using Principal Components rather than t-SNE embeddings. This choice adheres to the standard practice of applying DBSCAN to PCA-reduced data rather than non-linear embeddings like t-SNE, while also providing an alternative visual perspective of the data structure. It is important to note that the axes in these plots are distinct for each graph due to the variance in scale across different hours post-fertilization (`hpf`).



Select HPF for DBSCAN Clustering: × 4hpf
Set DBSCAN Epsilon Parameter: 0.9

Figure 5: DBSCAN layout visualized on PCA components.

An interactive slider allows users to dynamically recompute the DBSCAN analysis by adjusting the `epsilon` (ε) parameter. This parameter defines the radius of the neighborhood around a data point; by modifying it, the user directly controls the algorithm's sensitivity to density, determining how close points must be to define a cluster.

Adjusting the slider triggers an automatic recalculation of the clusters. While the plot updates automatically, the computational intensity of running DBSCAN on the fly implies a brief latency before the interface becomes responsive again.

To enable this dynamic execution with custom settings, the dataset is managed within the application layout using the `dcc.Store` component from Dash. This architectural choice ensures that the computationally expensive data reprocessing occurs only when the clustering parameters (the slider) are modified. Conversely, filtering the visualization by `hpf` operates instantaneously on the stored data without triggering a reload or recalculation.

5 Toolbox

The visualization tool was developed within the Python ecosystem, selecting the most common and reliable packages in the Data Science community. To ensure reproducibility and ease of installation, the following libraries were utilized:

- **Dash** and **Dash Bootstrap Components**: Used as the core framework for building the interactive web application. Dash provided the reactive callback structure necessary for linking the UI elements (sliders, dropdowns) with the Python logic, while Bootstrap components ensured a responsive and aesthetically pleasing layout.
- **Plotly**: Employed for generating all interactive visualizations, including the 2D scatter plots (t-SNE, PCA), heatmaps, and dendrograms. Its integration with Dash allowed for advanced features like cross-filtering and box-selection events.
- **Pandas** and **NumPy**: Essential for data manipulation and performance. Pandas was used to manage the gene expression dataframe and metadata, while NumPy handled the underlying numerical arrays and efficient matrix operations required by the clustering algorithms.
- **scikit-learn**: The primary machine learning library used to implement all dimensionality reduction techniques (PCA, t-SNE) and clustering algorithms (K-Means, Agglomerative Clustering, DBSCAN), as well as the feature selection methods (Random Forest, ANOVA).

6 Scalability, Performance, and Improvements

Significant adjustments were implemented throughout the development process to enhance the application's scalability and performance, particularly given the high dimensionality of the dataset.

The primary challenge involved loading the large dataset without incurring excessive wait times or blocking the user interface. To address this, the data loading strategy was optimized to minimize reloading while maintaining interactivity.

For the t-SNE visualizations, the architecture relies on a single pointer to a loaded dataframe, ensuring that only one instance of the dataset is kept in memory for all plots. This dataframe is loaded outside the application layout context. This design prevents the expensive reload operation during filtering tasks; since filtering by `hpf` (hours post-fertilization) requires no new computations, the dataset does not need to be directly coupled to the callback triggers, significantly improving response times. Additionally, this dataset retains the original gene expression values, facilitating the immediate generation of the heatmap without requiring auxiliary dataframes.

For the dendrogram visualization, a separate dataset is employed. This dataset is also loaded globally but differs in content: it relies on Principal Components (PCA) computed externally to the app. This separation is necessary because the hierarchical clustering operates on the PCA-reduced space rather than raw gene expression. To maintain cross-filtering capabilities with the primary t-SNE plots, both datasets share a common `kmeans_cluster` column, serving as a relational key between the views.

Finally, the DBSCAN implementation necessitated a specific dynamic data handling strategy. Unlike the static views, updating DBSCAN parameters (via the slider) requires re-running the clustering algorithm, which fundamentally changes the class labels associated with each cell. To optimize this re-computation, the application utilizes only the first two Principal Components of the PCA dataset. This reduction allows the algorithm to run efficiently in real-time. This decision was driven not only by computational constraints but also by empirical testing (documented in the DBSCAN notebook), which indicated that including additional components did not yield biologically meaningful results.

To manage this dynamic state, the resulting dataframe is stored client-side using the `dcc.Store` component. This ensures that expensive recalculations are triggered exclusively by slider events (changing ε) and not by unrelated interactions like filtering, which simply operates on the existing stored data.

Improvements Due to time constraints and this being an initial exploration of Dash and UI design, several areas for improvement have been identified:

- **Scalability:** While optimizations were made, the data processing pipeline could be further refined. Implementing server-side caching (e.g., Redis) or down-sampling strategies for the visualization layers could prevent the need to load the entire high-dimensional dataset into memory, further reducing latency.
- **Layout Design:** The current dashboard layout is functional and minimalistic, which aids focus, but it could be enhanced aesthetically to provide a more polished user experience.
- **Heatmap Visualization:** The current heatmap provides a high-level overview of expression trends across selected areas. However, distinguishing individual cells (rows) is difficult due to the potentially large number of selected points (¿1000). Implementing dynamic aggregation or zooming features would allow users to inspect specific cells more precisely without losing the global context.
- **Dendrogram Interaction:** The cross-filtering interaction between the dendrogram and the t-SNE plot currently has limitations regarding repeated selections. Specifically, double-clicking the same cluster leaf to toggle selection (select-deselect) does not trigger a callback update, as the `clickData` parameter remains unchanged. Implementing a more robust state management system or listening to different event triggers would resolve this usability issue.