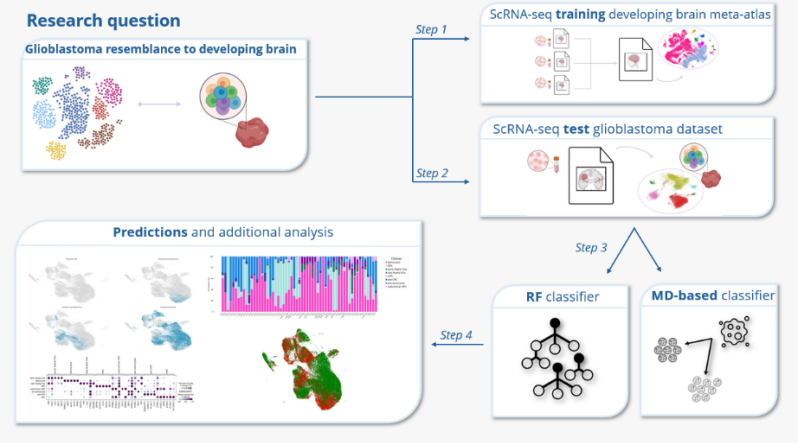
Classifying Glioblastoma cells based on developmental cell type resemblance

Ilaria Coratella, MSc *Bioinformatics and Biocomplexity, UMC Utrecht, Siletti lab*

1. Summary of the Project

Please note that the GBM data used in this project is confidential and has not yet been published. I used fake data for the expression matrix, but the rest is real. I kindly ask that you keep this private. Thank you for your understanding! This dataset has been created by Linnarsson lab, Karolinska Institute.

Glioblastoma Multiforme is an aggressive brain tumor marked by diverse cell types and complex genetic changes. Its cells can resemble various glial lineages (like astrocytes or oligodendrocyte precursors) and even mimic developmental processes from the healthy brain. Understanding how these tumor cells connect to normal brain cells could help develop targeted therapies.



In this project, we built a large meta-atlas of developing human brain cells as a reference for comparing GBM cells. The healthy meta-atlas contains single-cell RNA sequencing data from multiple studies and covers critical glial and neural progenitor cell types (e.g., early/late radial glia, astrocytes, oligodendrocyte progenitor cells, and pre-astrocytes). We carefully filtered out cells not relevant to our comparisons (like neurons or cells from non-cortical regions) and removed cycling cells that might confuse classification. This curated resource then served as the training data for labeling tumor cells.

We obtained a very large, still unpublished GBM dataset from the Karolinska Institute, which included hundreds of thousands of cells from different regions within patient tumors. We wanted to see how GBM cells map to the healthy developmental lineages. To do this, we used two different classification methods:

1.1. Random Forest Classifier

This is a popular machine learning model that combines many decision trees, works well with high-dimensional single-cell data and can rank how important each gene is for a specific cell type. In testing it on 20% of our training dataset, it accurately recognized cell types in our data about 93% of the time. When applied to GBM, it often labelled cells as astrocyte-like, oligodendrocyte-progenitor-like, and sometimes radial-glia-like.

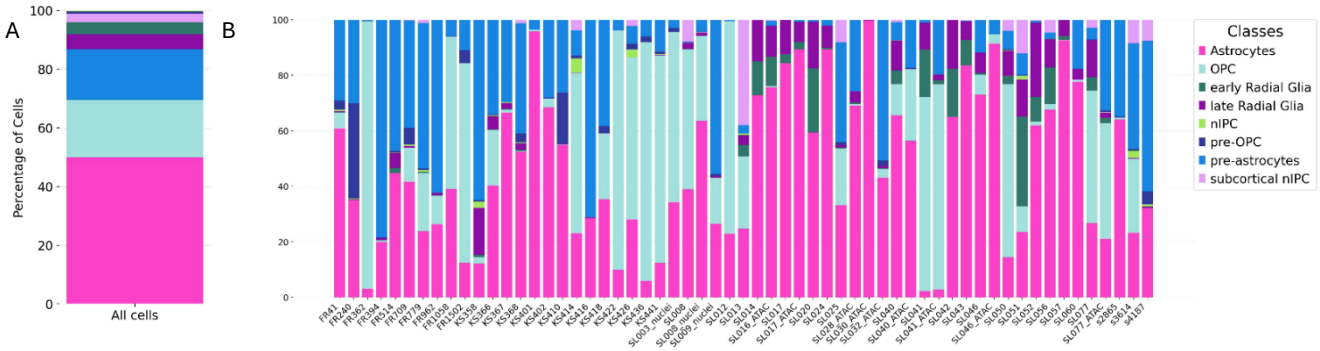


Fig. 1, percentages of predictions by cell class in all GBM dataset (A) and sample by sample (B) of Random Forest classifier.

1.2. Mahalanobis Distance-Based Classifier

This method calculates how close a tumor cell's gene expression is to the "center" of each healthy cell type, factoring in correlations between genes. It is easier to interpret because it gives a direct "distance" to each reference population. It scored a bit lower than Random Forest on our tests on healthy data (85%). It often labelled cells as astrocyte-like, pre-astrocyte like and oligodendrocyte-progenitor-like. The comparison of this second method results with the ones of Random Forest classifier highlighted cells that might have "mixed" or unusual signatures not seen in healthy tissue.

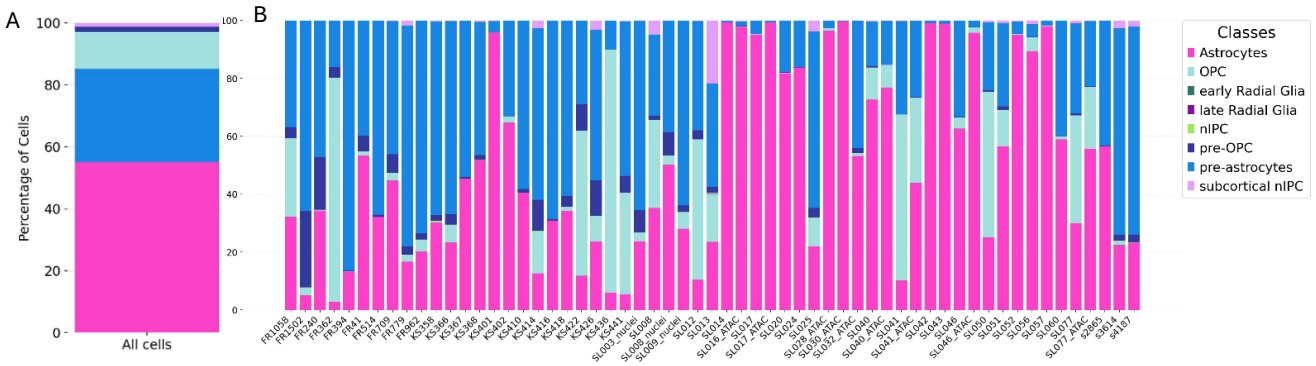


Fig2, percentages of predictions by cell class in all GBM dataset (A) and sample by sample (B) of Mahalanobis distance-based classifier.

Both methods revealed that GBM cells commonly align with astrocyte-like and oligodendrocyte-progenitor-like states, but they also showed that many tumor cells express gene patterns from multiple cell types at once (so-called "hybrid" states). Interestingly, certain tumor samples were classified in nearly identical ways by both tools, while others showed substantial differences—suggesting that some tumors contain more ambiguous cells than others. Specifically, the average of matching predictions was 65 percent.

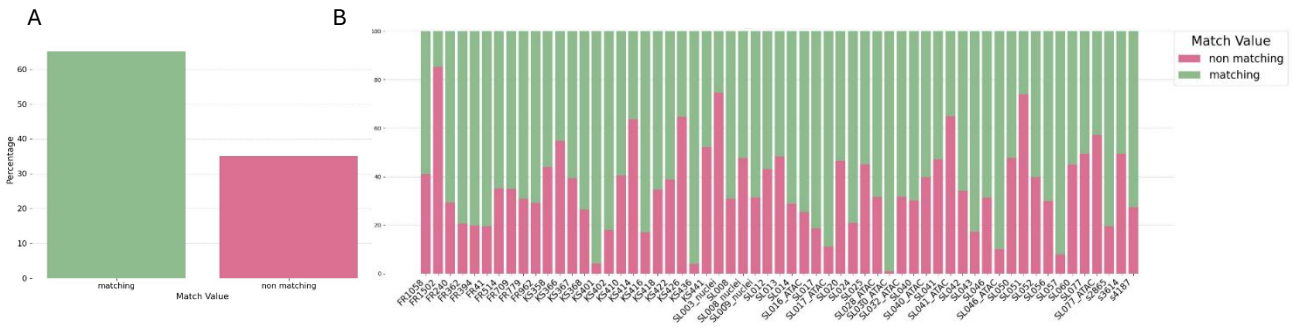


Fig. 3, Comparison of amount of matching and non-matching predictions among the Random Forest and Mahalanobis distance- based classifiers on the entire dataset (A), and tumor by tumor (B).

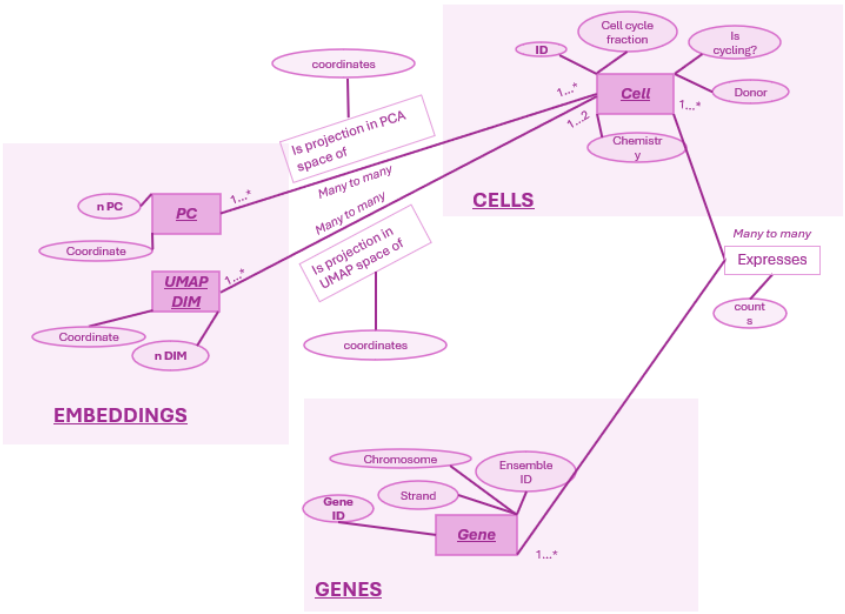
Overall, our study underscores the importance of using healthy developmental references to understand GBM cell diversity. By comparing two different classification approaches, we could pinpoint which cells are assigned clearly to a single lineage and which ones might be adopting more complex or mixed identities. This work sets the stage for future investigations into how these different GBM subpopulations might influence disease progression and response to therapy.

2. Building a conceptual model of an Anndata object

During my internship, I consistently worked with single-cell RNA sequencing (scRNA-seq) data, primarily handling two key files: one containing data from healthy cells, a concatenation of three different dataset, (1, 2, 3) and another from tumor cells (still unpublished), both stored in the .h5ad format. This format, based on HDF5, is specifically designed for storing high-dimensional biological data, particularly scRNA-seq datasets. It serves as the native format for Scanpy, the most widely used Python package for analyzing gene expression data at the single-cell level. H5AD is structured in a way that makes it particularly efficient and optimized for managing large-scale scRNA-seq data. It follows the standard format for AnnData objects, which are the core data structures in Scanpy. This organization facilitates the easy import of both gene expression data and metadata. The structure of an AnnData object consists of several main components:

- *.**X:** The expression matrix, storing gene count data per cell.*
- ***.obs:** Metadata related to individual cells, such as cell type, clustering information, and patient origin.*
- ***.var:** Metadata for genes, including gene names and specific characteristics.*
- ***.obsm, .varm:** Additional annotations, often related to dimensionality reduction techniques like PCA, t-SNE, or UMAP.*
- ***.uns:** Unstructured information, such as analysis parameters or color annotations for clusters.*

My approach to working with these data files involved revisiting the conceptual framework underlying the AnnData object to make sense of my data in a structured manner. I began by defining a conceptual model based on four fundamental elements, reflecting the structure of the anndata object, and investigating the relationships among the, Cell (obs), Gene (var), Expression data (X) and embeddings (.obsm). Here, the first draft of a conceptual model representing the data contained in an anndata object:



The primary concepts are Cell and Gene, each characterized by various attributes that serve as metadata. These concepts are interconnected through the .X attribute of the Anndata object, which represents the expression data. The relationship between Cell and Gene is termed "*Expresses*", indicating that a cell expresses a gene. This expression relationship is many-to-many: a single cell can express multiple genes, and a single gene can be expressed in multiple cells. This relationship has the attribute *Counts*, one count for each unique couple cell-gene.

Additionally, I considered defining separate concepts for each type of embedding to avoid confusing similar coordinates and to represent different subspace projections. For example:

- Principal Component: This concept has a many-to-many relationship with Cell. Each cell can have multiple principal components (e.g., 50 in my case, though this number may vary), and each principal component is associated with all cells. This setup mirrors the Cell-Gene relationship, where the 50 principal components replace the 4,000 genes. The relationship is (Principal Component) *is projection into PCA space of* (Cell), and the relationship itself has the attribute *coordinates*, one coordinate for each combination Cell-PC.
- UMAP Dimension: In this scenario, each cell is associated with two UMAP dimensions. Each UMAP dimension is linked to all cells, creating a one-to-many relationship from UMAP dimensions to cells and a one-to-two relationship from cells to UMAP dimensions, given that the number of UMAP dimensions is fixed. The relationship is (UMAP dimension) *is projection into UMAP space of* (Cell), and the relationship itself has the attribute *coordinates*, one coordinate for each combination Cell-Umap dimension.

I have never really used t-SNE, but the table for it would be identical to the one of the UMAP. By structuring the data in this manner, each embedding type maintains its distinct subspace projection, ensuring clarity and preventing the mixing of similar coordinates.

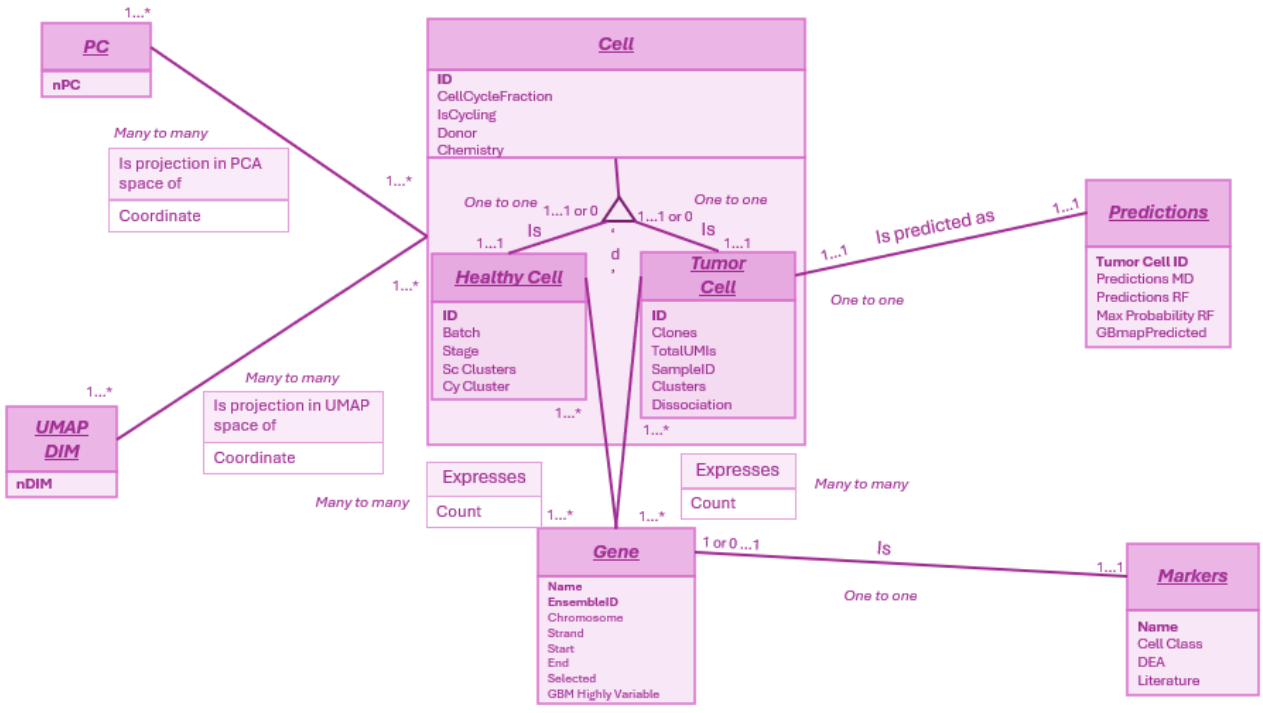
3. Adapting this model to my study

During my internship, I used two separate Anndata objects: one for healthy developing brain cells and another for GBM tumor cells. This required adapting the previous model to fit my specific needs. I didn't want to combine healthy and tumor cells into a single concept because each type has many unique attributes in addition to shared ones. In designing the database, I wanted to avoid having many missing values (NANs) in columns specific to one cell type when dealing with the other.

To achieve this, I created a "parent" entity called Cell, which branches into two sub-entities: Healthy Cell and Tumor Cell. The Cell entity includes only the attributes common to both types, while the specific attributes are stored in the respective sub-entities. Each sub-entity has its own separate relationships with the Gene entity. I preferred this cleaner approach because, after my internship, I realized that although both are cells, they represent fundamentally different concepts.

Additionally, I added a Predictions table. Instead of storing predictions as attributes of tumor cells, I separated them because predictions are the main focus of my study. I needed to update predictions multiple times and add different versions of classifiers with various parameters. Keeping predictions in a separate entity could have made this process easier. I considered creating a table for the eight labels used and relating it to TumorCell with a many-to-many relationship, storing predictions for both methods in the join table. However, since GBmap predictions use different labels and are conceptually similar to other predictions, I should have decided if creating a separate table for it or place it together with the Tumor Cell attributes, even though this column is more conceptually similar to Predictions.

The Marker Gene entity is a subset of Gene and includes genes identified through literature or differential expression analysis. Instead of adding a true/false column to the Gene table, I created a separate entity because there were only a few specific attributes, such as the source of the marker gene. For clarity and space, I used UML notation instead of Chen notation.



Here is a description of the conceptual model.

Cell includes attributes common to both Healthy Cell and Tumor Cell:

- ID: Unique identifier
- CellCycleFraction: Proportion of cycling-cell-associated genes expressed
- IsCycling: Indicates if the cell is cycling (true or false)
- Donor: Patient identifier
- Chemistry: Version of the scRNA-seq technique (e.g., V2, V3, V3.1)

A Cell can be either a Healthy Cell or a Tumor Cell. This is a one-to-one relationship with the verb "is." The relationship is disjoint, meaning a cell cannot belong to both sub-entities and must belong to one of them. Moreover, Cell has many-to-many relationships with PC and UMAP DIM, as described in the general conceptual model.

Healthy Cell includes attributes specific to healthy cells:

- ID: Unique identifier
- Batch: Indicates which of the three datasets from the meta atlas the cell belongs to
- Stage: Developmental stage of the brain from which the cell originates
- Clustering Results: Results from two different clustering analyses

Healthy Cell has a many-to-many relationship with Gene. The relationship, called Expresses, includes the count of each gene expressed in each cell.

Tumor Cell includes additional attributes relevant to cancer research:

- Clones
- Dissociation: Indicates if the cell is a single cell or single nucleus
- Total UMIs: the total number of UMIs for each cell
- Sample ID: for this dataset I also had samples information rather than only donor information.
- Cluster: Results from a clustering analysis

It has a many-to-many relationship with Gene, similar to Healthy Cell, with the Expresses relationship storing gene counts.

Gene includes metadata common to both cell types, (except for GBM Highly Variable). The specific meta data for each dataset for genes was not important for my internship:

- Name: Common name of the gene
- EnsembleID: Another identifier for the gene
- Chromosome: Chromosome location
- Strand: Plus or minus strand
- Start: Start position on the chromosome
- End: End position on the chromosome
- GBM Highly Variable: Indicates genes highly variable in glioblastoma

It has a many-to-many relationships with both Tumor Cell and Healthy Cell, with attributes indicating gene expression levels in each cell.

Predictions associates tumor cell IDs with various prediction methods:

- Predictions MD: Predictions using the Mahalanobis distance-based method

- Predictions RF: Predictions from a Random Forest classifier
- Max Probability RF: Proportion of trees in the RF classifier that agree on the prediction
- GBmap Predictions: Generated by the Karolinska Institute using different labels and a scoring function in Scanpy

Predictions has a one-to-one relationship with Tumor Cell, with the relationship labeled “predicted as.”

Markers includes a subset of genes important to my research:

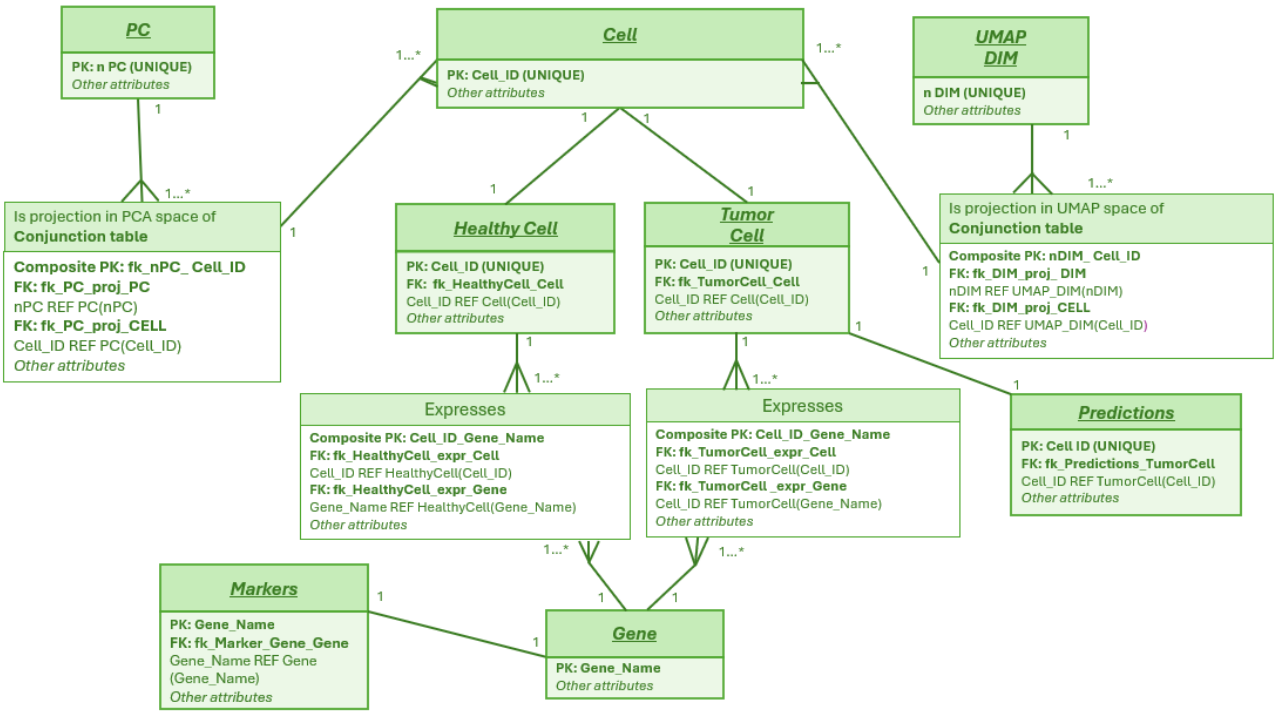
- Gene Name
- Cell Class: One of the eight prediction labels the marker is associated with
- DEA: Indicates if the marker was found in Differential Expression Analysis
- Literature: Indicates if the marker was identified through literature research

Markers has a one-to-one relationship with Gene through the “IS” relationship. A marker can be linked to one gene, and a gene can be linked to zero or one marker.

I did not make any changes to the Embedding parts of the model, such as those concerning PC and UMAP DIM, so I won’t discuss those further.

4. My relational model

Before setting up the structure in MySQL Workbench, I first created a relational model to map out the relationships between different entities using primary and foreign keys. In this model, I focused solely on key identifiers, primary keys and foreign keys, to keep the diagram simple.



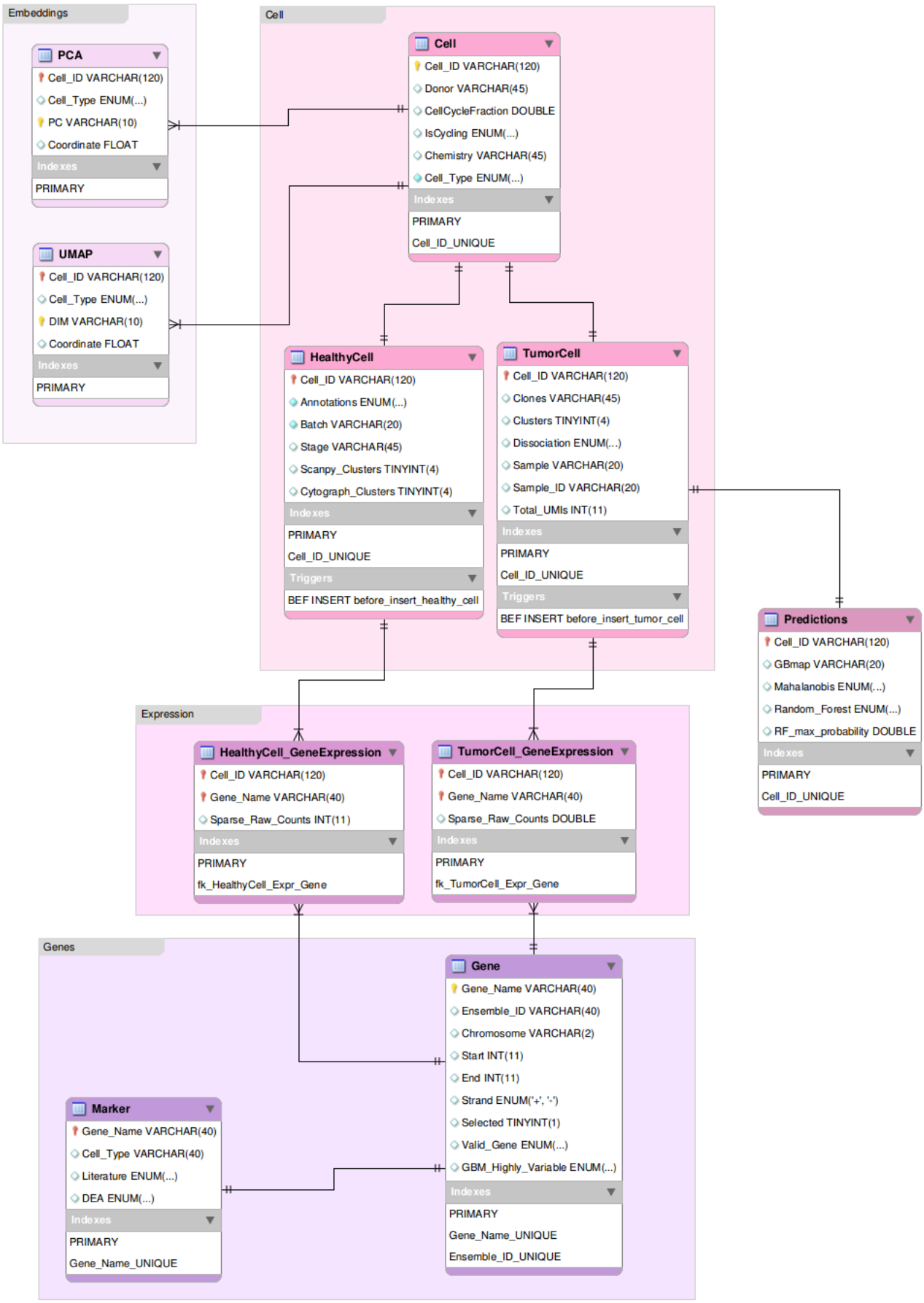
In my design, tables linked by a one-to-one relationship share the same primary key. For instance, the Cell, TumorCell, and HealthyCell tables all use Cell_ID as their primary key and are uniquely constrained. To establish these one-to-one relationships, I used foreign keys. For example, HealthyCell and TumorCell both reference Cell_ID in Cell through fk_HealthyCell_Cell and fk_TumorCell_Cell, respectively. Similarly, the Predictions and TumorCell tables share Cell_ID as the primary key, with Predictions linking to TumorCell through the foreign key fk_Predictions_TumorCell. Likewise, Markers and Gene share the primary key Gene_Name, connected via the foreign key fk_Marker_Gene_Gene in the Markers table.

For many-to-many relationships, such as between Cell and PC, Cell and UMAP DIM, HealthyCell and TumorCell with Gene, I created junction tables. Each junction table has a composite primary key formed from the primary keys of the related tables and includes two foreign keys that link back to the original tables. For example, the PC-Cell junction table uses nPC and Cell_ID as a composite primary key and has fk_PC_proj_PC and fk_PC_proj_Cell as foreign keys. This structure is mirrored in the UMAP DIM-Cell junction table. Similarly, the relationship between HealthyCell and Gene, as well as TumorCell and Gene, involves junction tables with composite primary keys (Cell_ID and Gene_Name) and foreign keys (fk_HealthyCell_expr_Cell and fk_HealthyCell_expr_Gene, in the case of the linkage between HealthyCell and Gene, fk_TumorCell_expr_Cell and fk_TumorCell_expr_Gene for the linkage between TumorCell and Gene) that facilitate the linking of tables in these many-to-many setups.

5. EER diagram on MySQL Workbench

Finally, I created an EER diagram in MySQL Workbench based on the previous model, including the data types, shown in the diagram. To ensure that a cell cannot be both healthy and a tumor cell, I added triggers to the HealthyCell and TumorCell tables. These triggers prevent inserting a healthy cell into the HealthyCell table if its Cell_ID already exists in the TumorCell table, and vice versa.

I also made changes to how the UMAP_DIM and PC tables are implemented. Since I didn't have any metadata for the principal components or UMAP dimensions, I decided not to create separate PC and UMAP_DIM tables. Instead, I only created the junction tables that link Cell_ID with nPC or UMAP dimensions. This means that each Cell_ID can be associated with multiple nPC values without needing a separate table to store the nPC details. This approach helped me to avoid redundancy and to keep the database structure simpler.


Embeddings
PCA
Cell_ID VARCHAR(120)
Cell_Type ENUM(...)
PC VARCHAR(10)
Coordinate FLOAT
Indexes
PRIMARY
UMAP
Cell_ID VARCHAR(120)
Cell_Type ENUM(...)
DIM VARCHAR(10)
Coordinate FLOAT
Indexes
PRIMARY
Cell
Cell
Cell_ID VARCHAR(120)
Donor VARCHAR(45)
CellCycleFraction DOUBLE
IsCycling ENUM(...)
Chemistry VARCHAR(45)
Cell_Type ENUM(...)
Indexes
PRIMARY
Cell_ID_UNIQUE
HealthyCell
Cell_ID VARCHAR(120)
Annotations ENUM(...)
Batch VARCHAR(20)
Stage VARCHAR(45)
Scanpy_Clusters TINYINT(4)
Cytograph_Clusters TINYINT(4)
Indexes
PRIMARY
Cell_ID_UNIQUE
Triggers
BEF INSERT before_insert_healthy_cell
TumorCell
Cell_ID VARCHAR(120)
Clones VARCHAR(45)
Clusters TINYINT(4)
Dissociation ENUM(...)
Sample VARCHAR(20)
Sample_ID VARCHAR(20)
Total_UMIs INT(11)
Indexes
PRIMARY
Cell_ID_UNIQUE
Triggers
BEF INSERT before_insert_tumor_cell
Predictions
Cell_ID VARCHAR(120)
GBmap VARCHAR(20)
Mahalanobis ENUM(...)
Random_Forest ENUM(...)
RF_max_probability DOUBLE
Indexes
PRIMARY
Cell_ID_UNIQUE
Expression
HealthyCell_GeneExpression
Cell_ID VARCHAR(120)
Gene_Name VARCHAR(40)
Sparse_Raw_Counts INT(11)
Indexes
PRIMARY
fk_HealthyCell_Expr_Gene
TumorCell_GeneExpression
Cell_ID VARCHAR(120)
Gene_Name VARCHAR(40)
Sparse_Raw_Counts DOUBLE
Indexes
PRIMARY
fk_TumorCell_Expr_Gene
Genes
Gene
Gene_Name VARCHAR(40)
Ensemble_ID VARCHAR(40)
Chromosome VARCHAR(2)
Start INT(11)
End INT(11)
Strand ENUM('+', '-')
Selected TINYINT(1)
Valid_Gene ENUM(...)
GBM_Highly_Variable ENUM(...)
Indexes
PRIMARY
Gene_Name_UNIQUE
Ensemble_ID_UNIQUE
Marker
Gene_Name VARCHAR(40)
Cell_Type VARCHAR(40)
Literature ENUM(...)
DEA ENUM(...)
Indexes
PRIMARY
Gene_Name_UNIQUE

6. Reflection on ER Modeling and Database Implementation for the Project

During my nine-month internship, managing data presented significant challenges. It was my first experience handling such large datasets, and there was limited focus on optimizing data management processes. Due to my cautious approach and the lack of specific guidelines, I frequently saved multiple copies of the same .h5ad files. This was necessary because of the extensive data refinement I performed and the evolving nature of the developing brain meta-atlas. For example, I worked with a subset of the original dataset that was continuously updated, leading me to save different versions to avoid, for example, overwriting the raw data with normalized data.

Exploring the underlying structure of the data, including the concepts and their relationships, deepened my understanding of the dataset. Developing conceptual models would have been beneficial for managing the data effectively, even without using a database. Moving forward, I plan to continue creating conceptual models to enhance my comprehension of the datasets I work with.

Throughout this project, I identified several advantages that a database system could have provided compared to using plain files. The most notable benefit would have been memory efficiency. My tasks required importing entire Anndata objects, which was computationally demanding. For example, importing a relatively small file required requesting at least 100 GB of RAM from the HPC. As the size of the datasets increased (millions of cells and tens of thousands of genes), the memory requirements soared to 250-300 GB of RAM. While large memory was essential for tasks involving raw data normalization or classifier implementation, it was unnecessary for operations focused on metadata, principal components, or UMAP visualizations. A database could have allowed selective data retrieval, significantly reducing memory usage and minimizing computational resources, which also contributes to reducing the environmental impact of extensive computer use.

Additionally, using a database would have enforced structured data management, reducing the need to save multiple copies and ensuring data consistency, cleanliness, and order. While it is possible to manage this with .h5ad files by adding columns, the approach becomes impractical as the number of annotations, predictions, and gene lists increases. That is because .h5ad files need to be imported in their entirety and make them huge do not help efficient work. A database facilitates managing diverse tables with different structures, enabling the addition of new columns or tables without duplicating entire files. For example, normalized data could be added as a new column in the expression tables without altering the raw counts. Furthermore, databases support logging queries, updates, and schema modifications, which are essential for tracking changes and ensuring reproducibility in scientific research. They also allow storing multiple versions of classification predictions (e.g., Random Forest, Mahalanobis distance) without losing previous results or duplicating other columns/data, facilitating easier comparison and reverting to earlier analyses. As the project expands with additional tumor samples, gene features, or embedding coordinates like t-SNE, the database schema can be modified in a controlled manner to accommodate new data fields.

Another significant advantage of databases is the ability to enforce constraints, which helps prevent data entry errors. For example, when importing on my database the previously merged ‘Stage’ metadata from three different datasets, I encountered missing values that before went unnoticed. Implementing a non-null constraint in the database flagged this issue, and I applied necessary corrections. Although I eventually removed the constraint to accommodate new data with eventual NANs, having the constraint initially helped identify the problem.

Databases also enhance data-sharing capabilities. Although this was not a major issue in my lab, as only my supervisor and I worked on the data, larger labs with multiple researchers could benefit from easier data access and sharing without duplicating files or managing shared paths on the HPC.

However, there are disadvantages to using databases. Integrating databases with bioinformatics tools such as Scanpy and Seurat may require additional scripting, increasing workflow complexity. For instance, Scanpy is designed to work with complete Anndata objects, necessitating the reconstruction of these objects from the database for analysis. Although this can be accomplished with a few lines of code, specifically something like:

adata = anndata.AnnData(X: df.iloc[1:,1:], obs: df.iloc[:,0:1], var: df.iloc[0:1,:]),

it adds an extra step to the workflow. Additionally, using a database requires careful consideration of which data to retrieve, which can be more demanding for the researcher compared to importing entire datasets and have all data ready, because he do not have to decide which specific data to access each time. However, using databases could avoid stressful situations later during the study.

Another challenge is the need to learn SQL syntax and database querying, which can be initially difficult to master. Understanding the structure and ordering of SQL queries required a bit of practice to me.

Overall, the advantages of using a database outweigh the disadvantages in the context of my project. Implementing a database system would have streamlined data management, enhanced efficiency, and supported scalable and reproducible research. I found the process of creating conceptual models and considering database integration valuable and plan to propose this data management approach in my upcoming internship.

7. References

1. Wang, Li, et al. "Molecular and cellular dynamics of the developing human neocortex at single-cell resolution." bioRxiv (2024).
2. Velmeshev, Dmitry, et al. "Single-cell analysis of prenatal and postnatal human cortical development." *Science* 382.6667 (2023): eadf0834.
3. Braun, Emelie, et al. "Comprehensive cell atlas of the first-trimester developing human brain." Science 382.6667 (2023): eadf1226.

8. Generative AI statement

During the final project for the course Introduction to Research Data Management, I utilized generative artificial intelligence for debugging assistance and correcting errors during coding, particularly in areas where online resources were limited. Additionally, I used it in refining my report by identifying and correcting grammar mistakes, as well as paraphrasing sections to convey ideas more clearly.