Algorithm General code

0. Preprocessing of FCS Files and Patient Data

Input:
- Folder with .fcs files from different times from all patients.
- Clinical patients data in a `.xlsx` file: type of B-ALL, collected date, regenerated samples...
Output: Combined information saved as `Full_info.csv`

1. UMAP Embedding of RBM and LBM samples

Input:
- Full_info.csv
Output: - Full_info = Full_info + two columns related to UMAP values for each cell. - Embedding data for Regenerated, Relapsed and Non-Relapsed.

2. DBSCAN clustering and SVM classification on UMAP-reduced data

Input:
- Full_info.csv
Output: - Labeled RBM cells by DBSCAN - Labeled LBM cells by SVM - Updated Full_info.csv with columns for labels.

3. UMAP Embedding of RBL

Input:
- Full_info.csv
Output: - UMAP values for B lymphocytes - Updated Full_info.csv with columns for RBL labels.

4. DBSCAN and SVM in RBL

Input:
- Full_info.csv
Output: - Labeled RBL cells by DBSCAN - Labeled LBL cells by SVM - Updated Full_info.csv with columns for LBL labels.

5. Confidence Intervals and Visualization of Results

Input:
- Full_info.csv
Output: - Tables with subpopulations percentages and MFI in Bone Marrow and B Lymphocytes - Images related to confidence intervals - Proposed classifiers along with their metrics

Algorithm 0 Preprocessing of FCS Files and Patient Data

Input: Patient metadata file `Patients.xlsx`, FCS data folder `PatientsFolder`, list of regenerated samples, basic biomarker list
Output: Combined DataFrame `TDF` saved as `fullinfo.csv`

a. Load patient metadata
Load list of all patients from Excel
Identify "Common", "Relapse (R)" and "Non-Relapse (NR)" patients
b. Define function `lecturachange(file)`
    Load FCS file using CytoFlow
    Detect metadata using `$PnN` or fallback to `$PnS`
    Standardize column names using CD markers
c. Define `load_data(folder)`
for all files in folder **do**
  if file belongs to known patient **then**
    Load file using `lecturachange`
    Save to CSV and store in list
  end if
end for
d. Call data loader: `[CytoData, Files] = load_data(PatientsFolder)`
e. Subsample 10,000 cells per file
for all samples in CytoData **do**
  if sample has $\geq$ 10000 cells **then**
    Randomly select 10,000 cells
  else
    Print warning: fewer than 10000 cells
  end if
end for
f. Pair files and data into `experiment`
g. Define helper functions:
    `patientsinfolder(exp)` $\rightarrow$ list of patients
    `patientsfiles(patient)` $\rightarrow$ files per patient
    `dataf(patient)` $\rightarrow$ dataframes per patient/timepoint with metadata
    `TotalDF(experiment)` $\rightarrow$ build full dataset TDF
h. Build TDF using `TotalDF(experiment)`
Append data for all valid patients with necessary columns and metadata
i. Filter patients:
    Remove those with missing biomarkers
    Keep only "Common" patients
    Store valid patients in `MyPatients`
    Remove invalid rows from `TDF`
j. Save final dataset as CSV: `fullinfo.csv`
k. Print summary
    Number of included patients
    Count of R vs NR patients
l. Report total runtime

Algorithm 1 UMAP Embedding of regenerated and leukemic samples

Input: CSV file `fullinfo.csv`, basic biomarker lists, regenerated sample IDs

Output: UMAP coordinates and filtered datasets saved as CSV and PNG figures

a. Load Data
Read dataset from `fullinfo.csv` and set index
Identify relapse (R) and non-relapse (NR) patients
Define basic biomarker columns and regenerated sample IDs

b. Filter data into subsets
- `dfRec` ← relapse patients with all basic biomarkers
- `dfNoRec` ← non-relapse patients with non-regenerated status
- `dfReg` ← regenerated samples with extended biomarkers

Remove excluded samples from `dfReg`

c. Fit UMAP on `dfReg`
Set parameters: $n = 50$, $d = 0.01$
Train UMAP: `trans = UMAP(...).fit(dfReg)`
Create DataFrame `dfembReg` with UMAP embeddings

d. Plot UMAP embeddings of regenerated samples
Color points by sample using `ColorNumber`
Add legend and title
Save figure as `UMAP Regenerated BM n50 d0.01.png`

e. Plot marker intensity over UMAP coordinates
For each marker in selected list:
- Create scatter plot colored by marker intensity
- Add individual colorbars

Save composite figure as `png`

f. Project test samples onto UMAP space
Transform `dfRec` and `dfNoRec` using trained UMAP model
Store as `dfembRec` and `dfembNoRec`

g. Export results
Save all UMAP embeddings and filtered data subsets to CSV:
- `UMAP_Regeneradas.csv`, `UMAP_RecReg.csv`, `UMAP_NoRecReg.csv`
- `df_Regeneradas.csv`, `df_RecReg.csv`, `df_NoRecReg.csv`
- Overwrite `fullinfo.csv` with updated index

Algorithm 2 DBSCAN clustering and SVM classification on UMAP-reduced data

Input: - Processed data frame `TDF` with patient metadata - UMAP projections for Regenerated (`dfembReg`), Recovered (`dfembRec`) and Non-Recovered (`dfembNoRec`) samples - Raw patient data: `dfReg`, `dfRec`, `dfNoRec` - Biomarker list (e.g., `CD19`, `CD66`, `CD45`)

Output: Labeled cell types in `dfembReg`, `dfembRec`, `dfembNoRec`; Merged dataset `TDF_EMB` with cluster and cell type labels; Trained SVM classifier for cell type prediction

a. Load UMAP data and set appropriate indices and column names
b. Load patient DataFrames `dfReg`, `dfRec`, `dfNoRec` and align indices;
c. Apply DBSCAN clustering on UMAP of Regenerated samples (`dfembReg`);
d. Determine number of clusters and noise points;
e. Compute centroids of each cluster (ignoring noise);
f. Label clusters according to marker intensity:
- Cluster with highest `CD19` $\rightarrow$ `BLymphocyte`;
- Cluster with highest `CD66` $\rightarrow$ `Myeloid`;
- Cluster with lowest `CD45` $\rightarrow$ `Erythroblast`;
- Remaining cluster $\rightarrow$ `MonocyteT`;

g. Replace numeric cluster labels in `dfembReg` with cell type names;
h. Assign color mapping for plotting;
i. Train `OneVsOneClassifier` with RBF-SVM using `dfembReg`;
j. Predict labels for `dfembRec` and `dfembNoRec` with the trained model;
k. Merge all embeddings into `EMB_label`;
l. Join cluster labels to `TDF` to create `TDF_EMB`;
m. Remove healthy samples and drop `NaN` rows from final set;

Algorithm 3 UMAP Embedding of Regenerated B-Lymphocytes

Input: CSV file `labeled_fullinfo.csv`, basic biomarker list, regenerated sample IDs

Output: UMAP coordinates of B-lymphocytes, embedded test samples, saved CSV and figures

a. Load and filter B-lymphocyte samples
Read `labeled_fullinfo.csv` and set index
Extract samples labeled as `BLymphocyte`
Filter regenerated samples from these
b. Prepare training data and fit UMAP
Define `BasicBiomarkers` list
Extract biomarker values and sample names for regenerated B-lymphocytes
Set UMAP parameters: $n = 200$, $d = 0.01$
Fit UMAP: `trans = UMAP(...).fit(xtrainBLympho)`
Store embedding as `dfembReg_BLympho`
Join embedding to `df_Regenerated_BLympho`
c. Visualize embedded B-lymphocytes
Plot scatter of embedded points colored by `ColorNumber`
Use labels and legends for sample names
Add title: `Regenerated B Lymphocytes UMAP n=200 d=0.01`
d. Prepare test sets: Relapse and Non-Relapse
Filter B-lymphocytes into relapse (`R`) and non-relapse (`NR`) subsets
Further restrict NR to those with non-regenerated status
e. Transform test samples onto UMAP space
Apply `trans.transform(...)` to relapse and non-relapse test sets
Store as `dfembLymphoRec` and `dfembLymphoNoRec`
Join transformed coordinates with original relapse and non-relapse data
f. Combine all B-lymphocyte subsets
Merge regenerated, relapse, and non-relapse samples into `All_BLymphocytes`
Save combined DataFrame as `All_BLymphocytes.csv`

Algorithm 4 DBSCAN Clustering and Classification of Regenerated B Lymphocytes

Input: CSV files `labeled_fullinfo.csv`, `All_BLymphocytes.csv`, folders for patients and images
Output: Clustered and classified B lymphocytes, figures saved, updated CSV with labels

a. Load Data
Read `labeled_fullinfo.csv` as `TDF_EMB` and set index
Read `All_BLymphocytes.csv` as `All_BLymphocytes` and set index
Filter regenerated B lymphocytes into `df_Regenerated_BLympho`
b. DBSCAN Clustering on UMAP Coordinates
Extract UMAP coordinates as `X`
Run DBSCAN with `eps=0.6`, `min_samples=70` on `X`
Calculate number of clusters and noise points
Scale `X` using `StandardScaler`
Identify core samples and cluster labels
Calculate centroids of each cluster (excluding noise)
Assign cluster labels to `df_Regenerated_BLympho`
c. Plot Clusters and Centroids
Plot points colored by cluster label
Plot centroids as triangles
Save figure with timestamp
d. Assign Biologically Meaningful Names to Clusters
Compute mean marker intensity (`CD45`) per cluster
Sort clusters by intensity and assign names: `Mature`, `Trans`, `ProBPreB`
Map these names to `Lympho_label` column
Define color map for labels
e. Classification of Test Samples
Prepare training data from regenerated samples (`X_train`, `y_train`)
Prepare test data from relapse (`X_test_Rec`) and non-relapse (`X_test_NoRec`) samples
Train One-vs-One SVM classifier with RBF kernel
Predict labels on test samples
f. Plot Classification Results
Plot training and test samples colored by predicted labels
Save classification figure
g. Update Main Dataset with Cluster Labels
Insert predicted labels into relapse and non-relapse subsets
Combine labels and assign to `All_BLymphocytes` as `LymphoLabel`
h. Visualization Functions
Define `BLymphoPatient_t` to plot labeled B lymphocytes per patient and timepoint
Define `UMAP_LinfosB_Paciente` to plot marker intensities for lymphocytes per patient and time
Define `marcadoresLinfosPatient` to plot selected markers for a patient at time `t0`
i. Merge Labels with Full Dataset
Merge `LymphoLabel` and UMAP coordinates back into `TDF_EMB`
Fill missing labels with `"No_BLympho"`
Save merged data to `Cells_TotalEMB.csv`

Algorithm 5 Confidence Intervals and Visualization of Percentages and MFIs

Input: CSV files with percentages and centroids for R, NR, Regenerated samples

Output: PNG figures with Bootstrap 95% confidence intervals across subpopulations and timepoints

a. Define Paths and Load Data

Load patient folder and image output folder

Read main cell file `Cells_TotalEMB.csv` into `TDF_EMB`

Load percentage tables: `percent_NR_t0`, `percent_R_t0`, ..., `percent_RegeneratedBM`

Load centroid tables: `BLympho_RBM`, `Myeloid_LBM_NR`, etc.

b. Define Bootstrap Function

Define `bootstrap_ci()` to compute CI using resampling (default 10,000 iterations)

c. Plot Confidence Intervals for Percentages

Create subplot grid of size 4×3 (subpopulations $\times$ timepoints)

for each timepoint $t \in \{t_0, t_1, t_2\}$ **do**

  for each subpopulation $s \in \{BLymphocyte, Myeloid, Erythroblast, MonocyteT\}$ **do**

    Compute CI for R, NR, Reg samples using `bootstrap_ci`

    Plot mean and CI for each group on subplot $[i, j]$

    Add horizontal lines, scatter means, and shaded regions

  end for

end for

Save figure as `Percentages_combined_t0_t1_t2.png`

d. Plot Focused Confidence Intervals for BLymphocyte

Create 1×3 subplot for BLymphocyte at t_0, t_1, t_2

Calculate common y-limit for t_1 and t_2

for each timepoint **do**

  Compute CI and mean for R, NR, and Reg samples

  Plot similar to previous step, with adjusted y-limits

end for

Save figure as `BLymphocyte_t1_t2_same_yticks.png`

e. Plot Confidence Intervals for MFI (Markers)

Define marker list (e.g., `CD34`)

Create subplot grid of size 5×3

for each timepoint t **do**

  for each marker m **do**

    Compute CI and mean for m in R, NR, and Reg samples

    Plot mean and CI with shaded areas and vertical/horizontal lines

  end for

end for

Save figure as `Bootstrap MFI BLYMPHO/CD34.png`

f. Output and Display

Export plots as PNG with transparent background

Display all plots using `plt.show()`