# Single-Cell RNA-seq Analysis

**\*Single-Cell RNA-seq Analysis**

Single cell RNA sequence analysis is the process wherein gene expression profiles of each individual cell is analysed in depth which helps us to later classify them into clusters. These clusters represent identically functioning cells like for examples cells involved in sugar metabolism, water regulatory genes etc. are grouped in an individual cluster.

The process helps to understand the heterogeneity within the cells while still maintaining a database with hundreds or rather thousands of heterogeneous cells having identical functions. Like for examples: Cells in a tissue and cells in bone differ but their ultimate aim (at a very vague perspective) is to maintain their host’s nature which can only be achieved with appropriate expression of the genes involved. So, vaguely these cells can be achieved in a much broader category but with several distinct sub-categories when we dive into a deeper level of understanding and analysis.

**METHODOLOGY:**

**1. Downloading single cell RNA dataset**

* Navigate to <https://www.10xgenomics.com/resources/datasets>
* Apply filters:

Species: Human

Sample/tissue Type: Blood

Cells or nuclei: Cells

Platform: Chromium Single cell

Search for : “**10k Human PBMCs, 3' v3.1, Chromium Controller**”

* Navigate to the [chromium controller’s](https://www.10xgenomics.com/datasets/10k-human-pbmcs-3-v3-1-chromium-controller-3-1-high) webpage and click on “Output and supplemental files” option.
* Download [**Feature / cell matrix HDF5 (filtered)**](https://cf.10xgenomics.com/samples/cell-exp/6.1.0/10k_PBMC_3p_nextgem_Chromium_Controller/10k_PBMC_3p_nextgem_Chromium_Controller_filtered_feature_bc_matrix.h5) h5 data.
* This file is out single cell RNA matrix.

**2. Preprocessing (Normalization and Filtering)**

This helps to clean the raw single-cell RNA-seq data by removing low-quality cells and genes, normalizing for technical variations, and selecting informative features (genes) for analysis. This removes noise and also prepares or rather reduces the dimension of this data for UMAP analysis which will appear in the later stages.

Required Libraries: Scanpy, NumPy, Pandas, Matplotlib.pyplot, seaborn

*#1 Preprocessing*

import scanpy as **sc**

import **os**

*# Configuration*

**sc**.set\_figure\_params(dpi=100, facecolor='white', frameon=False)

**sc**.settings.verbosity = 0 *# Silent*

plot\_output\_folder = "C:/Users/mohak/Desktop/scRNA\_analysis/plots/preprocessing"

**os**.**makedirs**(plot\_output\_folder, exist\_ok=True)

**sc**.settings.figdir = plot\_output\_folder

*#Data file path*

data\_file\_path = "C:/Users/mohak/Desktop/Cell matrix.h5"

*#Loading the Data*

adata = **sc**.read\_10x\_h5(data\_file\_path)

adata.var\_names\_make\_unique()

*#Calculating Metrics*

adata.var['mt'] = adata.var\_names.str.startswith('MT-')

**sc**.pp.calculate\_qc\_metrics(adata, qc\_vars=['mt'], inplace=True)

*#Visualizing Metrics (Pre-filtering) - Plots saved to 'figdir'*

**sc**.pl.violin(adata, ['n\_genes\_by\_counts', 'total\_counts', 'pct\_counts\_mt'], jitter=0.4, multi\_panel=True, save="\_qc\_violin\_pre.png")

**sc**.pl.scatter(adata, x='total\_counts', y='n\_genes\_by\_counts', color='pct\_counts\_mt', save="\_qc\_scatter\_pre.png")

*#Filtering Cells and Genes*

adata = adata[

    (adata.obs['n\_genes\_by\_counts'] > 200) &

    (adata.obs['n\_genes\_by\_counts'] < 2500) &

    (adata.obs['pct\_counts\_mt'] < 5.0)

].copy()

**sc**.pp.filter\_genes(adata, min\_cells=3)

*#Visualizing Metrics (Post-filtering) - Plots saved to 'figdir'*

**sc**.pl.violin(adata, ['n\_genes\_by\_counts', 'total\_counts', 'pct\_counts\_mt'], jitter=0.4, multi\_panel=True, save="\_qc\_violin\_post.png")

**sc**.pl.scatter(adata, x='total\_counts', y='n\_genes\_by\_counts', color='pct\_counts\_mt', save="\_qc\_scatter\_post.png")

*#Normalization and Log-transformation*

adata.layers["counts"] = adata.X.copy()

**sc**.pp.normalize\_total(adata, target\_sum=1e4)

**sc**.pp.log1p(adata)

*#Identifying Highly Variable Genes*

**sc**.pp.highly\_variable\_genes(adata, min\_mean=0.0125, max\_mean=3, min\_disp=0.5)

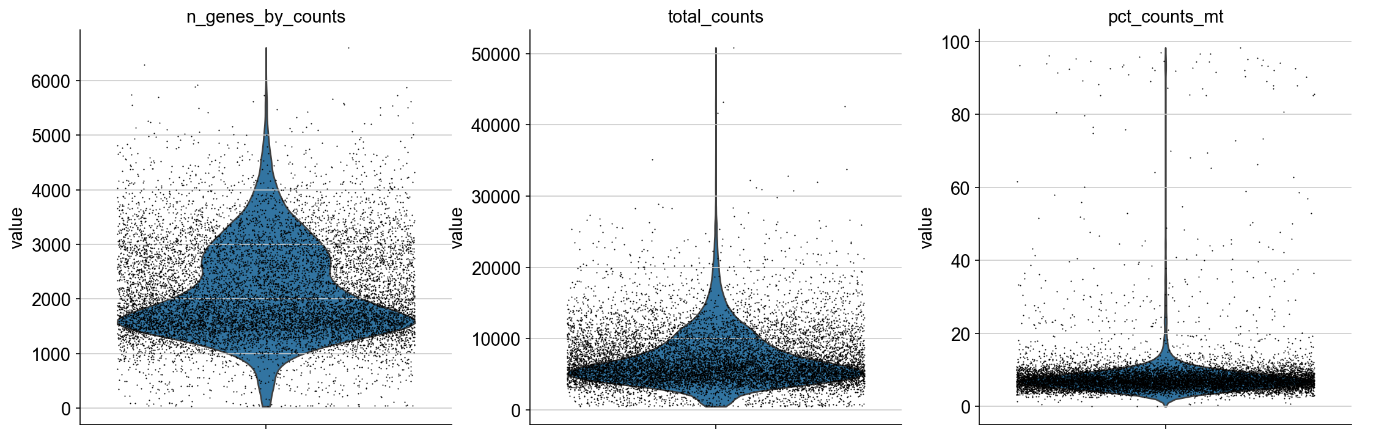
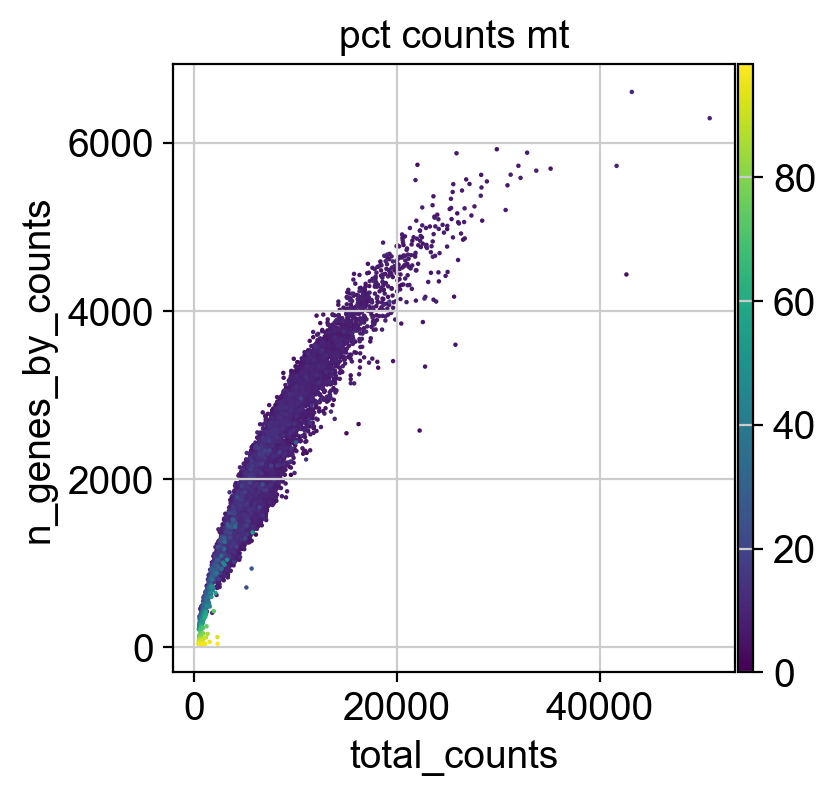
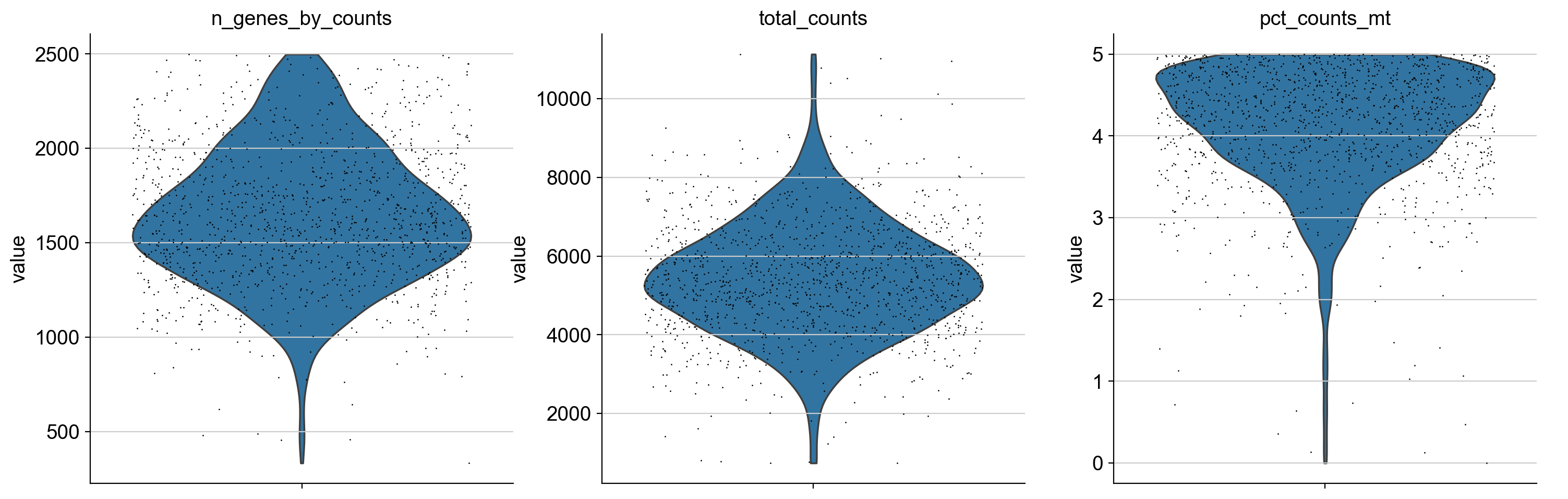
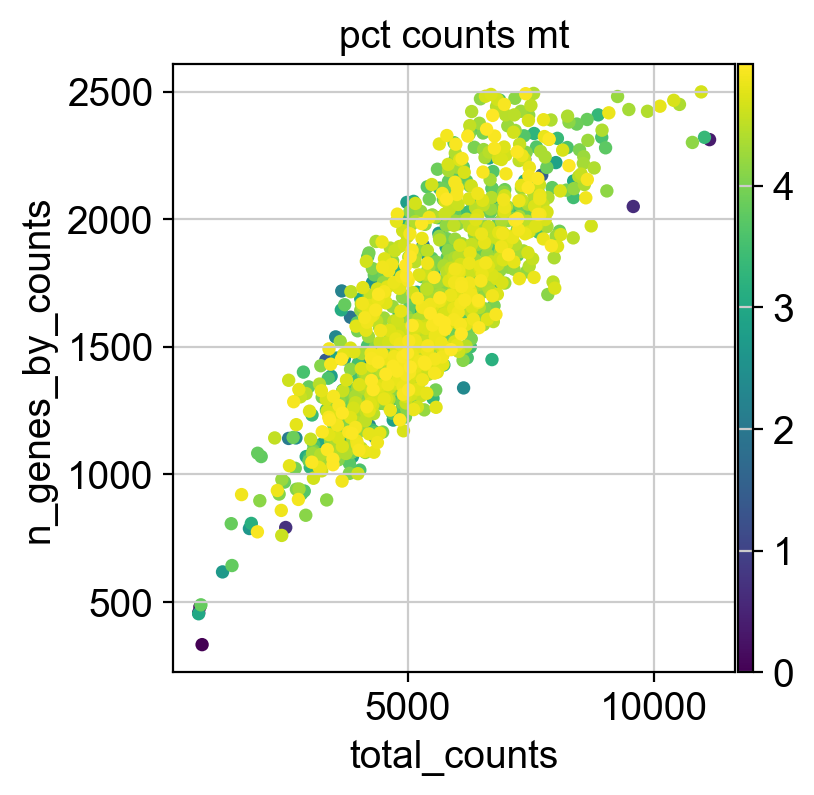
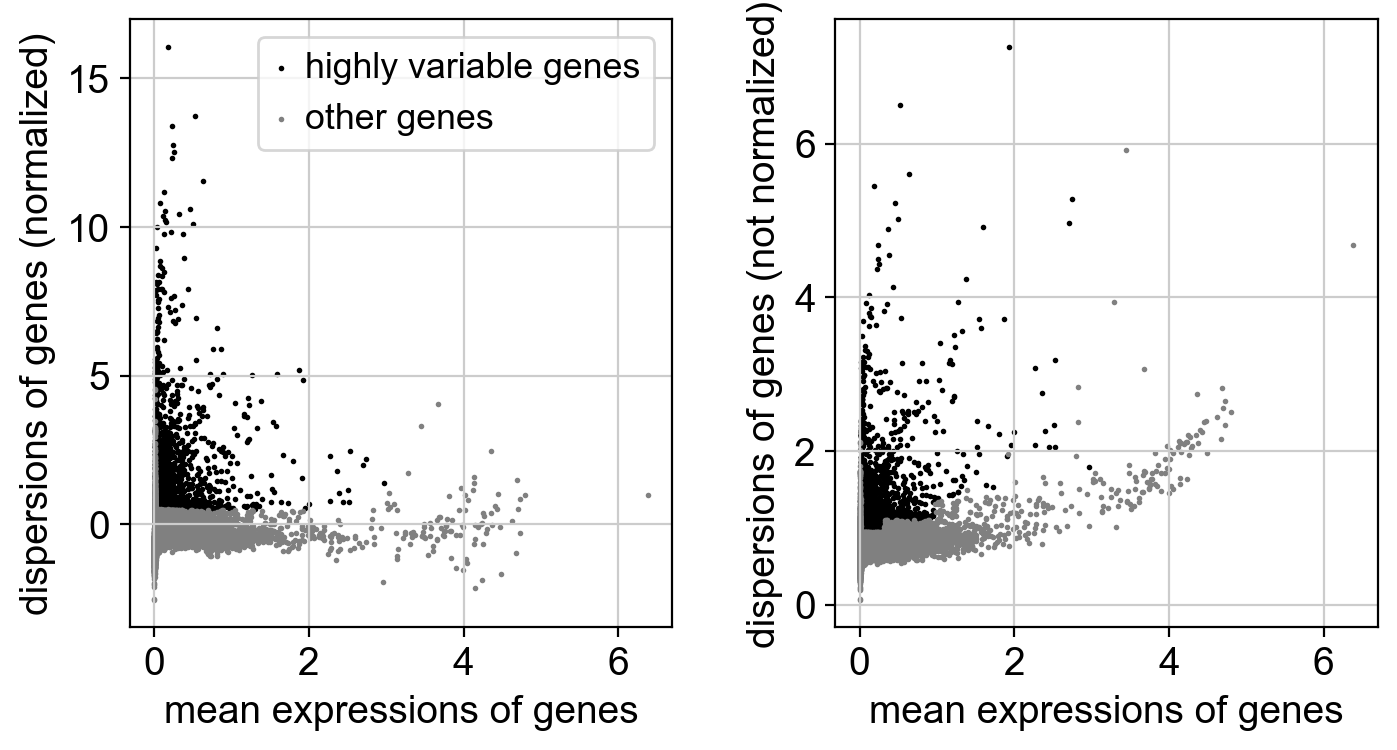
**sc**.pl.highly\_variable\_genes(adata, save="\_highly\_variable\_genes.png")

adata = adata[:, adata.var.highly\_variable].copy()

*#Scaling the Data*

**sc**.pp.scale(adata, max\_value=10)

**print**("Preprocessing complete.")

* The configuration helps to decide the figure size, colour and the addition or removal of border as Frameon false removes the border.
* The verbosity measures how much of the output is displayed on the screen meaning verbosity helps to print out the information of how our Scanpy is working on the console. 0 silences the entire operation while 3 prints each and every step on the console.
* The data loading process helps to load the data which is our h5 file into the VS code’s environment. All the potential errors have been handled by using the unique() function.
* The quality control metrics have been added by defining that the mitochondrial genes start from “MT”. As stated earlier these genes are unwanted as it denotes that our cytoplasmic mRNA has been eradicated from the cell due to a loss of plasma membrane or cell death.
* The visualization using violin plots helps to understand the nature of our raw data pre and post filtering process. Based on visual plots the filters are decided and applied so that only the required pure, healthy data is obtained rather than using noisy data which affects the overall analysis and creates false positives.
* 
* A. n\_genes\_by\_counts represents the total number of genes per cell. This violin plot helps to understand the overall distribution of genes in our 10000 (10K) cells data. The broader region which between 1000 and 3000 denotes the number of genes found in majority of the cells meaning out of our 10k cells majority of them are high-quality appropriately mapped gene matrix data and the number of genes present are approximately 1000 to 3000 genes.
* The tail extending to 0 denotes that some cells have very few genes which are detected or rather they are dying or dead cells whose most of the RNA is degraded.
* The upper part extending above 6000 denotes that there is a specific number of cells which have genes greater than or within the range of 6000. These are deemed to be extremely transcriptionally active.
* B. total\_counts: This depicts the distribution of Unique molecular identifiers (UMI) or reads mapped to genes in each cell and often referred to as “library size” or “sequencing depth” for each cell.
* The broader section upto 20,000 denotes bulk of high-quality cells with dead cells/ low gene count cells tail at 0 with exceptionally transcriptionally active cells at 50,000.
* C. pct\_counts\_mt: This denotes the percentage of mitochondrial counts which are total RNA counts in a cell that originated from mitochondrial genes. The tight cluster at 0% denotes a very low mitochondrial gene expression which is a characteristic of healthy, viable cells as mitochondrial mRNA is relatively stable.
* The long extension above 20% denotes viable or dying cells this is because when cell dies the cytoplasmic mRNA gets exhibited out while retaining the mitochondrial mRNA which increases the mitochondrial reads among the remaining RNA which is the case seen above 20% area.
* The high percentage of mitochondrial mRNA in a dataset always represents majority of cells being dead or not active which is not the case with this particular dataset. However, they need to be filtered out as they our treated as noise in the dataset as we only need cells with low mitochondrial mRNA percentage as they denote viable and healthy non-damaged cells.
* 
* This scatter plot denotes the overall data quality
* The x-axis denotes the total Unique molecular Identifiers (UMI) or read detected for each cell. This simply denotes our library size.
* The y-axis denotes the number of genes per cell which are unique as well.
* The pct counts mt denotes the mitochondrial mRNA counts colour bar. The dark purple (region between 0 to 20) denotes healthy cells and dead/less mRNA comprising cells above that.
* All-in-all this scatter plot comprises our quality control metrics or data before filtering the unwanted cellular data.
* 
* Similar to the above violin plots, these violin plots denote number of genes per cell, UMIs or library size and mitochondrial mRNA after applying the filtering threshold. These plots represent the healthy cellular data without any noise of mitochondrial mRNA.
* Note: Mitochondrial mRNA data from all the cells has not been removed but rather those cells which exhibited high percentage have been removed.
* 
* This is the similar scatter plot generated after filtering the unhealthy cells.
* 
* This represents highly variable genes which are those genes which have the most effect and simultaneously informative as well.
* The Left plot denotes the highly variable gene dispersion after normalization process. The x-axis denotes the mean expression of the genes across all the cells whereas y-axis dispersion of genes describing how much the gene’s expression varies from cell to cell.
* The Right plot denotes the highly variable genes dispersion but without the normalization process.
* The black dots represent highly variable genes that our program has selected whereas the grey dots represent the genes which are not highly variable but still have a considerable effect.

**3. UMAP + Leiden Clustering**

* Even after filtering out valid samples and normalization techniques the data is extremely high-dimensional, this nature of our data cannot be exploited for analysis as it produces misleading patterns and also computationally power invasive. So, reducing the overall dimensions of the data helps to analyse without utilising much resources of the computer as well as help in controlled parametric analysis. It also helps to transform a high-dimensional gene expression data into a more manageable form that allows us to visualize cell relationships and also identify distinct populations.
* A pre-requisite step is to use PCA (Principal Component Analysis) before turning to UMAP. PCA is a linear transformation that allows identification of uncorrelated principal components that capture the most variance in the data.
* UMAP is abbreviated for Uniform Manifold Approximation and Projection is a non-linear dimensionality reduction technique which helps in preserving the nature, relationships and structure of both the local and global variables in our dataset.
* Leiden Clustering is used in community detection on the clustered graph or group of cells that have similar gene expression (in this single cell RNA sequence expression). Scanpy builds a k-nearest neighbour graph of the cells (where cells are connected if similar) and then applies Leiden to find communities within the graph.
* Code:

Pre-requisite: Install igraph

Install leidenalg

*#2. UMAP and Leiden Clustering*

save\_file\_path = "C:/Users/mohak/Desktop/UMAP and Clustering Plots"

*# A. Principal Component Analysis (PCA)*

**sc**.**tl**.pca(adata, svd\_solver='arpack')

**sc**.**pl**.**pca\_variance\_ratio**(adata, log=True, save="\_pca\_variance\_ratio.png") *#Elbow Plot*

*# B. Computing the Nearest-Neighbour Graph*

**sc**.**pp**.**neighbors**(adata, n\_neighbors=15, n\_pcs=50)

*# C. UMAP Embedding*

**sc**.**tl**.**umap**(adata)

**sc**.**pl**.**umap**(adata, save="\_umap\_initial.png", title="UMAP (before clustering)")

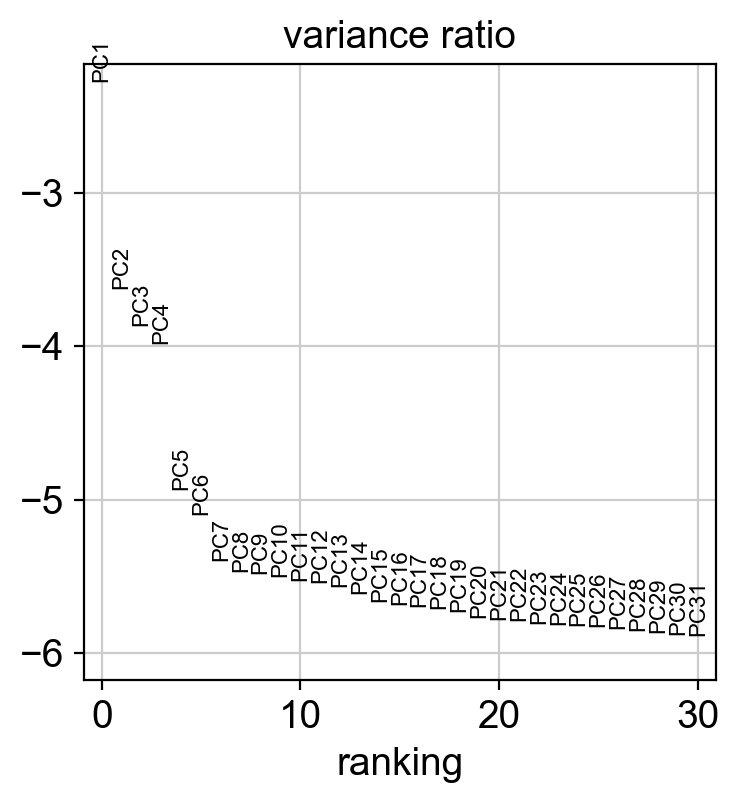
*# D. Leiden Clustering*

**sc**.**tl**.**leiden**(adata, resolution=0.5)

*# E. Visualizing UMAP with Clusters*

**sc**.**pl**.**umap**(adata, color='leiden', legend\_loc='on data', save="\_umap\_leiden.png", title="UMAP with Leiden Clusters")

**print**("Dimensionality Reduction and Clustering Complete.")



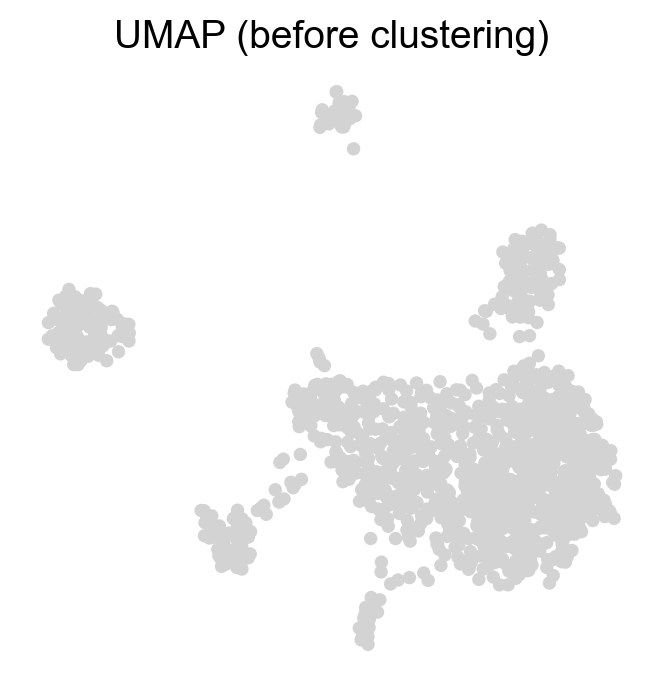
* This graph is an elbow plot which explains the variance by our principal components (PC).
* The x-axis denotes the individual PC in order from PC1 (which explains the most variance, that is why on the very top of the graph) and PC31 explains the least variance.
* The y-axis denotes the value on a logarithmic scale which shows the proportion of total variance in the dataset which is explained by each individual PC. Now, the log = True helps to build this axis. The negative nature of the axis suggests logarithms in exponential forms such as log10(10-3) = -3. As the negativity reduces the variance explained by PC increases.
* The use of variance in this project is to understand how our data behaves around the average. In simpler words, the variance shows how much differences the gene expression profiles exhibit around the mean. More the variance means our gene expression profiles are not similar rather differ significantly so, analysis is beneficial and useful which cannot be said in the case of low variance. Low variance denotes that the gene expression profiles are almost similar to the average of all of them meaning there is no much use of analysing the sequence.
* In the elbow plot:

PC1----PC6: explain the most variance

PC7----PC20: the variance explained by them reduces

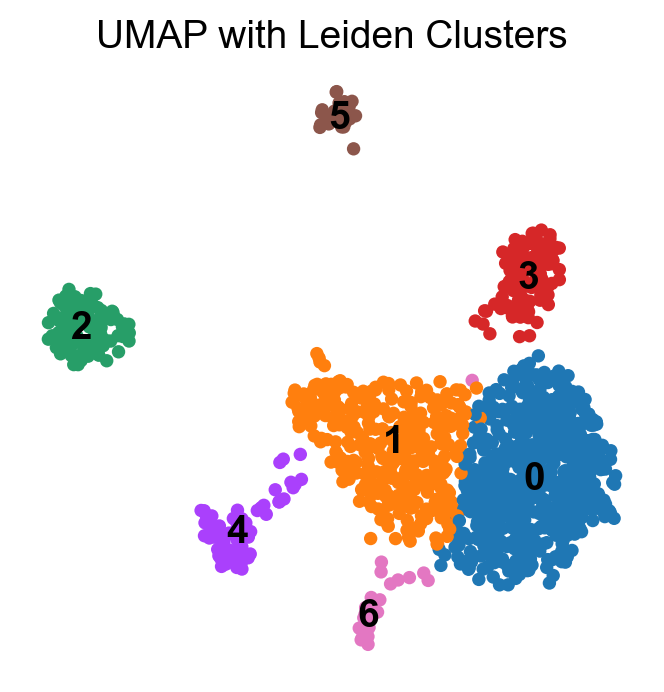
PC21----PC31: Diminishing or remaining variance

* This plot helps us to define which PC to keep for further analysis as the ones which show remaining variance or almost a flat curve would generate noise in our analysis so, it’s best to filter them out.



The image depicts each individual cells in our data which are transcriptionally similar. The Scanpy algorithm has visualized this graph based on the UMAP coordinates defined by the code: **sc**.**tl**.**umap**(adata). Now, before clustering the cells that are close to each other are transcriptionally similar in the original high-dimensional space, the transcriptional similarity can only be confirmed when clustering algorithm like Leiden is applied but based on the UMAP’s non-linear dimensionality reduction algorithm and the dedicated coordinates on a two-dimensional space the plot has been visualized.

This plot does not give us which cells are distinct from each other and that’s where clustering comes into action.



* The above plot depicts the Leiden Clustering on our UMAP visualized plots.
* Each distinct cell type or community has been given a different colour which suggests that they are transcriptionally different from each other.
* The cells which are coloured in the same community refer to those which are transcriptionally similar to each other which can be seen as close-knit clusters.
* The 0 and 1st cell cluster are extremely close to each other which suggests that there are many similarities post transcription but some subtle changes might have led to different behaviour post transcription.
* 2 and 5 appear to be completely distinct cell population.
* As this is a Peripheral Blood Mononuclear Cell, the clusters might represent T cells, B cells, monocytes, NK cells, dendritic cells etc but defining exact one is difficult at the current stage.
* The appearance of distinct community on the space suggests that Scanpy and UMAP dimensionality reduction have correctly identified distinct cell types which have led to formation of varying-coloured clusters.

**4. Identifying marker genes for clusters**

The clusters generated after the Leiden clustering process would now be defined a cell type based on their most effective marker genes.

*#3 Finding the Clusters in the data and naming them*

*# Ensure rank\_genes\_groups has run from step 4*

**sc**.**tl**.**rank\_genes\_groups**(adata, 'leiden', method='wilcoxon', rankby\_abs=True, tie\_correct=True)

*# Diagnostic: Print the actual categories*

**print**("\nDiagnostic: Actual Leiden Cluster Categories")

actual\_leiden\_categories = adata.obs['leiden'].cat.categories.**tolist**()

**print**(f"Actual Leiden clusters are: {actual\_leiden\_categories}")

**print**("End Diagnostic")

new\_cluster\_names = [

    "0: Your Cell Type A",

    "1: Your Cell Type B",

    "2: Your Cell Type C",

    "3: Your Cell Type D",

    "4: Your Cell Type E",

    "5: Your Cell Type F",

    "6: Your Cell Type G"

]

adata.**rename\_categories**('leiden', new\_cluster\_names)

**sc**.**pl**.**umap**(adata, color='leiden', legend\_loc='on data', save="\_umap\_annotated.png", title="UMAP with Annotated Cell Types", show=True)

**plt**.**show**()

**print**("Marker gene finding and cluster annotation complete (review plots and manual names).")

This code generates the actual leiden clusters which helps to prevent renaming errors.

*# Getting the top 10 marker genes for each cluster*

*# This will print a table where each column is a cluster (0, 1, 2, etc.)*

*# and the rows are the top genes for that cluster.*

**print**("\nTop 10 Marker Genes for Each Cluster")

marker\_genes\_df = **pd**.**DataFrame**(adata.uns['rank\_genes\_groups']['names']).**head**(10)

**print**(marker\_genes\_df)

This code generates the highest ranked marker genes of each cluster. These genes are then surveyed across the literature and databases available and later combined to assign one single cell type to that cluster.

**\*LITERATURE SURVEY PROCESS:**

1. Navigate to <https://panglaodb.se/markers.html?cell_type=%27choose%27>

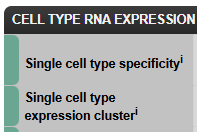
2. Search for each of the individual gene in “Blood” and “Immune System” part of the database. Record the findings.

3. To strengthen the findings, navigate to another database titled “The Human Protein Atlas”.

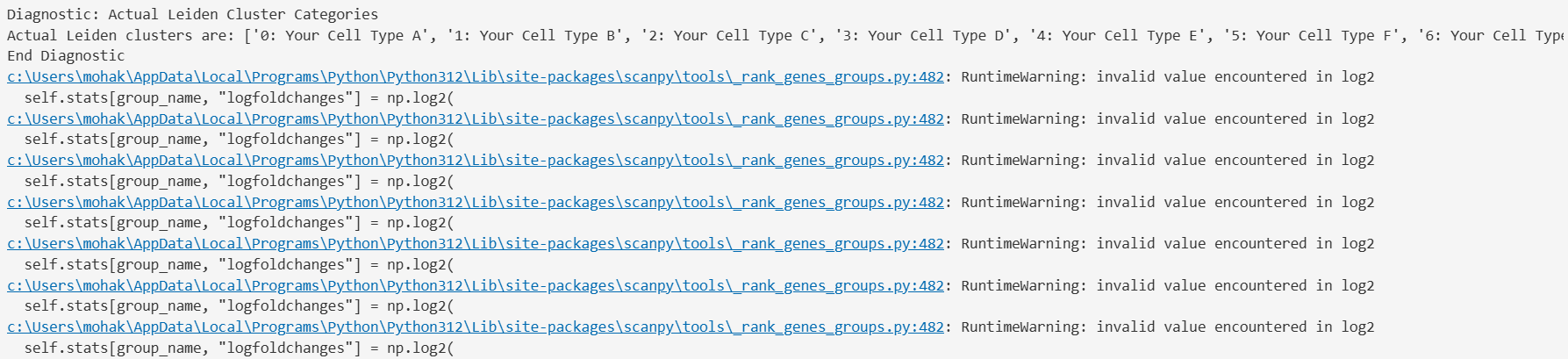
4. In the search box, type-in the name of your first gene for a particular cluster being analysed and press search.

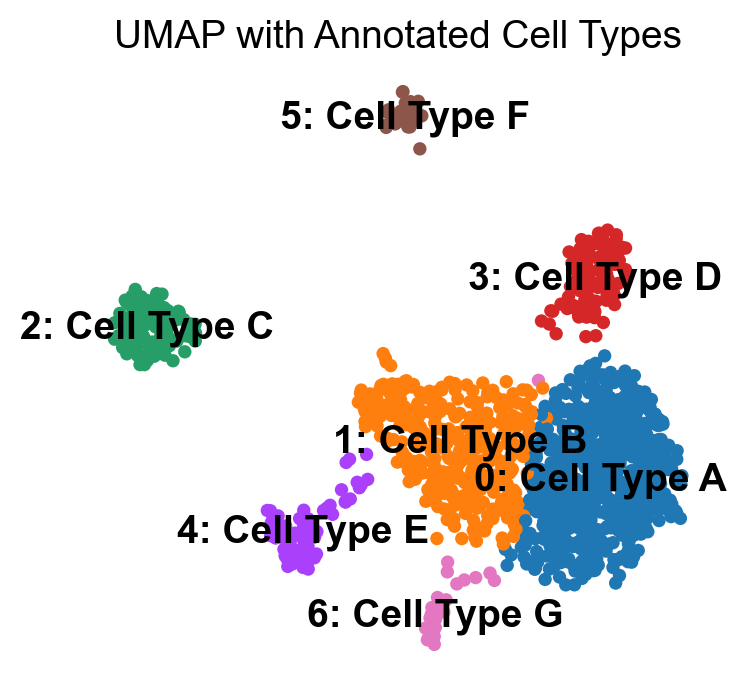
5. Open the one with identical gene name under the “Gene” section of the webpage.

6. Scroll down to “CELL TYPE RNA EXPRESSION”. Analyse the “Single cell type expression cluster” and “Single cell type specificity”. Record your findings.



7. If for a particular gene either of them comes out to be “Non-specific – Mixed function (mainly)” , navigate to RNA DATA  option on the same webpage. Scroll down to single cell type and record the findings where the bargraph has the highest value out of all the other graphs.





These clusters would be renamed after a literature survey on each of the clusters 10 highest ranked marker genes.

