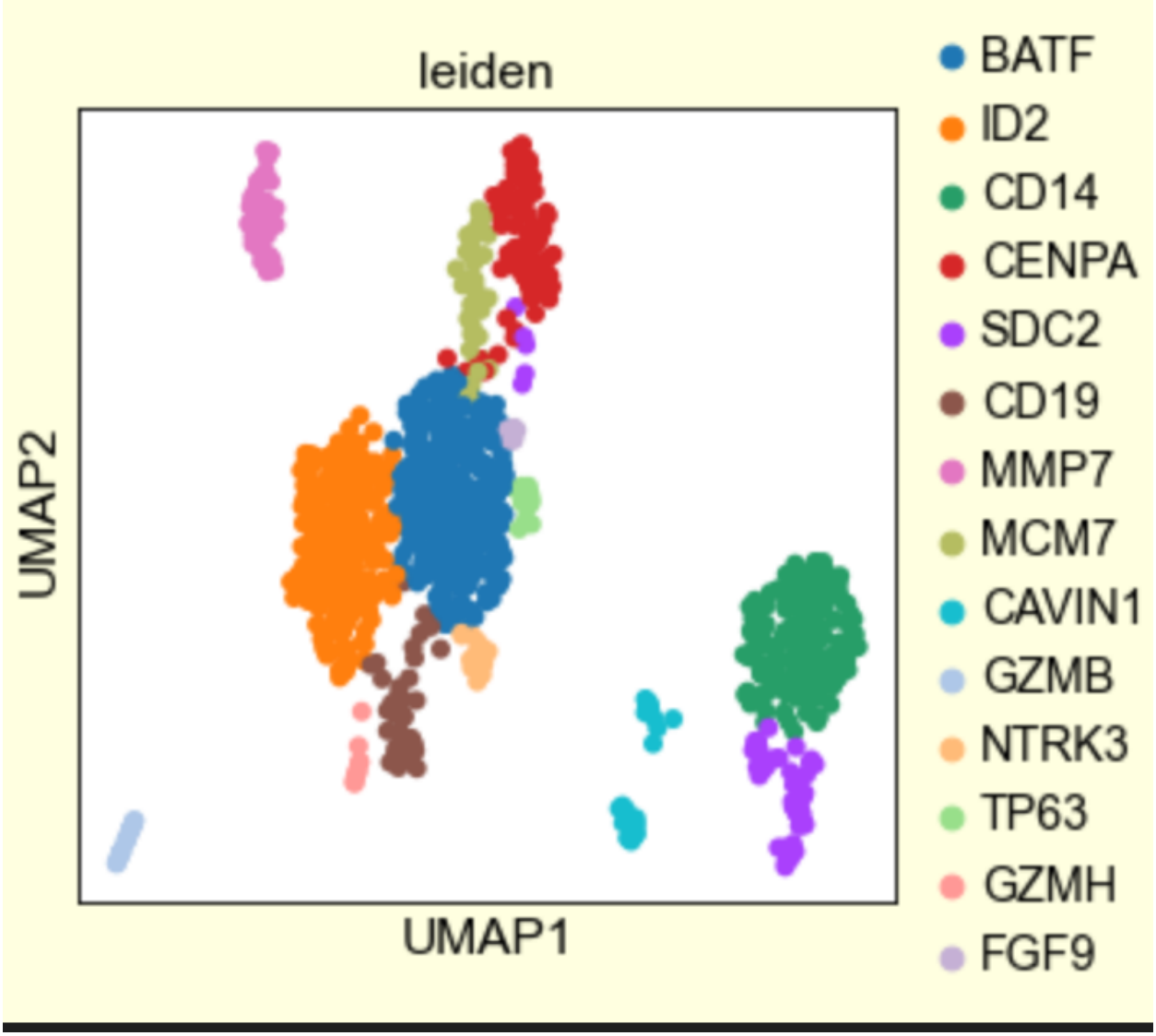
Updated Project Description: **Only changes are mentioned, you can reference same features of proposal here*

2) My project focuses on building a user-first pipeline for bioinformatics analysis of tumor microenvironments. While the code can be formatted with any 10x genomics dataset (and SingleSeq2), I primarily focused on an inquiry into cancer and decided on Hodgkin's Lymphoma because the tissue type and sequencing method is traditionally different than most other tumors, while at the same time the total cancer cell concentration is very minimal but has a high variance (the cancer works with a higher immune cell content instead of the actual cancerous cells, making it hard to detect and prognose based on stages).

This challenge was quite narrow now, but I had set up a set of success criteria that I did end up matching. A few key changes in the project description included the fact that any dataset could be inputted, so this would serve a purpose beyond the tumor microenvironment dataset.

Additionally, I had initially designed my proposal to be a single input and single output process, but through further research and learnings about the value of single-cell RNAseq, I ended up focusing on the entire pipeline and being as transparent as possible in the sequencing workflow to inform the user on the biological connections between the math, plots, and tumor sample along with having them change any hyperparameters if they were skilled enough. This evolved into me also finding a library which could be challenging enough to test but also easy enough to build and expand other features on (got the chance to work on trajectory inference through a PAGA plot).

Input: URL of dataset or path
O1: Filtered Top-20 high Frequency Genes
O2: N_genes by count, Mitochondrial Genes, Total_count
O3: Histogram and Distribution of Value Counts
O4: Filtered Data -> Threshold Outliers
O5: High Variable Gene Plot and Coverage
O6: PCA and Knee Fit Plot
O7: Variance Ratios and Loadings
O8: K-Nearest Neighbours *Custom
O9: UMAP AND T SNE Plots + Coverage
O10: Leiden Clustering Algorithm
O11: PAGA Plot for Trajectory Inference
O12: Dotplot and Violin for Cell Viability and Gene Expression Density
Output: Labelled Clusters Based on Gene Activity



Beyond that, while the time taken for the project almost tripled (spent 2 days working from 8AM -> 1245AM straight because of some previous commitments over the weekend), I managed to still complete my initial project description with a greater focus on gamifying it to make it educational while still serving a practical and useful purpose.

Key Changes Made to FlowChart:

Max_d -> Does Gene Deviate from standard deviation in range of top Hv_t genes in dataset? -> Ended up ranking genes through this loop later on where I used logistic regression to rank the importance of the variables and give the clusters actual gene IDs, something I thought would take a lot of time but became quickly attainable.

Dimensionality Reduction -> removed diffusion maps and reduced dimensionality only with PCA because UMAP and TSNE had to be calculated on top of PCA + diffusion maps embeds cell trajectories in Euclidean Space, and this is an inaccurate representation of the data primarily because of the significant feature reduction (if we kept it at 17 PCs, this was viable but still not useful for the final output and so I ended up taking it out).

Choose optimal dimensionality reduction -> for the actual clusters, you cannot use TSNE in the first place because it is not necessarily an accurate representation of the topological features of the gene expression, instead it acts as a buffer sample to visualize key changes in trajectory. Ended up having to use the default UMAP because the graph was subject to variable changes based on the number of k-neighbours.

K-Means -> calculate embedding but cluster not through K-means, but a variation called leiden, an instead iterative algorithm which also accounts for similarity indexes between different groups (this can be used to plot trajectories of certain cell clusters and was used to assist the PAGA plot).

Compute Loss -> unnecessary because learning rate was already implemented directly in k-neighbours embeddings, library made this easier instead of a manual implementation.

Key Changes Made to Scope:

- [x] ~~* A user interface in command line that can allow users to input a link to a publicly hosted CSV~~
- [x] ~~Providing Cell Viability and Gene Expression Patterns~~
- [x] ~~Deriving Variance of Data, Number of Outliers, and Number of Cell Types Found~~
- [x] ~~Plot that Visualizes Topology and Clusters of these Cell Types with Clear Labels~~
- [x] ~~Additional Readings for Patients to be more informed~~
- [x] ~~+ If I did not have the label names of the cells, I would want to further investigate some key features of each cluster such as cellular activity, specific gene expression profiles, which is essentially gene analysis and metastable dynamic states which I can then use to identify cell types for newer data sets. This would be very critical in actually learning about new things of the patient, such as how effective a therapy was and whether it can be seen here or to visualize the immune system impacting the tumors in the body.~~
- [x] ~~+ Another idea is to collect metrics on this such as cell lineages and cell viability metrics to give more detailed information about the clusters and activity. Perhaps it can help to also provide more background information for readers and turn this into a cancer information hub that outlines useful stats on gene expression and regulatory elements involved in cancer progression or reduction.~~

- [x] ~~+ The fourth and final idea would be to look at trajectory inference where I can look at differential gene expressions amongst different clusters identified related to cell cycles, replication, genetic info transfer, and more which can help to identify progression of cells in the body through time. This adds a completely new layer to the data and can thus allow for better analysis of tumor progression in earlier stages.~~

¾ Completed for **IF I HAVE TIME**

User Guide (GITHUB):

:microscope: LYMPHOMA MICROENVIRONMENT ANALYSIS - README

Welcome to my sc-RNAseq code, designed to be a straightforward and user-friendly pipeline that helps interpret and analyze key cell environments in a tissue sample.
The example provided here is meant to give you an intuition of how and why I designed this code and the different biological underlying mechanisms at play that can be interpreted through microenvironment analysis.

> This informatics pipeline primarily utilized scanpy, a powerful bioinformatics library built by the [Theis Lab](https://github.com/theislab/scanpy).

Project Outline
This project is designed to provide a complete overview of the entire sc-RNAseq pipeline. The tool has become widely popular for a variety of reasons including:
- mapping genotypes to phenotypes is challenging in biology and medicine → powerful method is currently transcriptome analysis, but transcriptome information in a cell reflects activity of a small subset of genes body's cells each express unique transcriptome, and increasing evidence shows that gene expression is heterogeneous even in similar cell types
- majority of transcriptome analysis is based on assumption that cells from given tissue are homogeneous and these studies likely miss important cell-to-cell variability
- most biological processes are stochastic, thus we need more precise understanding of transcriptome in individual cells for finding their role in cell functions and understanding how gene expression can promote certain genes using cDNA of individual cells, single-cell RNA sequencing was born → this helped provide high-resolution views of single-cell heterogeneity on global scale and allowed for finding differences in gene expression between individual cells has potential to identify rare populations that cannot be detected from analysis of pooled cells
    - eg. finding and characterizing outlier cells within population has potential implications for furthering understanding of drug resistance and relapse in cancer treatment with bioinformatics pipelines, can now learn more about highly diverse immune cell populations in healthy and diseased states

- scRNA-seq is being utilized to delineate cell lineage relationships in early development, myoblast differentiation, and lymphocyte fate determination

These key features make it very promising, primarily because it has been very useful for understanding single-cell identity, involving the different gene expressions to their response and interaction with external stimuli such as neighbouring cells or diseases like tumors, which is the primary focus of the demo.

This pipeline is an interactive, user-friendly visualization and analysis toolset which can be used to better analyze and interpret single-cell RNA sequencing data of a set of tissues, and then analyze different features of these cells such as their gene expression activity (transcriptome), protein interactions, and cell localization.

The program is designed to visualize and analyze sc-RNAseq data which ultimately looks at gene and cell activity for the individual types of cells in a microenvironment. This dataset is based off of Hodgkin's Lymphoma samples sequenced by 10x Genomics, and analyzed in the pipeline which looks at a tumor microenvironment and the different kinds of cells there are (T-cells, CD4 vs CD8+ T Cells vs NKCs in a tissue, FOXP3+/CD3+/SIGLEC 1 T-cell -> the primary goal will be to find patterns and essentially ID different kinds of cells and try to identify whether or not the immune system of the body is actually killing off the tumor.

Further Readings:
> [Analysis of scRNA-seq Data](https://scrnaseq-course.cog.sanger.ac.uk/website/index.html)
> [Best Practices in scRNAseq Analysis](https://www.embopress.org/doi/full/10.15252/msb.20188746)
> [StatQuest Playlist](https://www.youtube.com/user/joshstarmer)
> [scanpy Documentation](https://scanpy.readthedocs.io/en/stable/usage-principles.html#workflow)

Biological Significance

Brief history of transcriptional profiling:

- 2000s, microarrays enabled systematic measurements of transcriptional changes → mRNAs labeled, loaded onto chip where each spot on chip was coated to catch mRNA of specific gene → color of spot indicates ratio of transcript
- Bulk RNA sequencing made it much more feasible which transcriptional differences could be measured
    - quantification of gene expression
    - comparative transcriptomics
- limitations involve the inability to resolve heterogeneity
- single cell also gives you dynamics of gene expression and cell identity, primarily because it can help you look at cell topology

- goals of scRNA-seq are 1) measure distribution of expression levels for each gene across population of cells 2) measure transcriptional differences across and within groups of cells 3) resolve single-cell heterogeneity
- applications involve characterization of cell and transcriptional composition of tissues (cell type changes and their correlation)
- evaluate developmental processes (see expression changes while undergoing differentiation)
- evaluate how cancer evolution leads to mechanisms of therapeutic resistance

Droplet-based approach:

1. Tissue disassociated and samples created to separate individual cells, need to make sure tissue maintains integrity to prevent digestion of extracellular layers
2. samples collected, separated into individual droplets (use cytometry to evaluate cell viability through FACS to measure cell composition by pre-binding cell with fluorescent or antibodies to find expectation of proportion of cell types)
3. oil and water reaction has a single bead with barcode that bind to the specific mRNA of a single cell + done through cell where water and oils are displaced → you have a poisson distribution of beads and cells (proportion of concentration is based on concentration of cells and beads)
4. cells are lysed, reverse transcriptase synthesizes cDNA and applied molecular identity, use pCR and then add sequencing adapters for sequencing and assembly (3'-5')
    - lysis part is flawed right now because mRNA molecules don't always bind to the beads, RT enzymes are known to be inefficient, and RT + PCR depend on specific transcript, making this more variable
- only about 5% of mRNA is captured → UMI: total count of unique molecular identifiers which are labelled on each mRNA with a unique barcode at the RT step
    - UMI reduces amplification error → evaluation of performance found that 10x is one of the better ones → SUPeR-seq with Accuracy of 0.95, 4 million copies of mRNA sequence in every cell

Limitations:

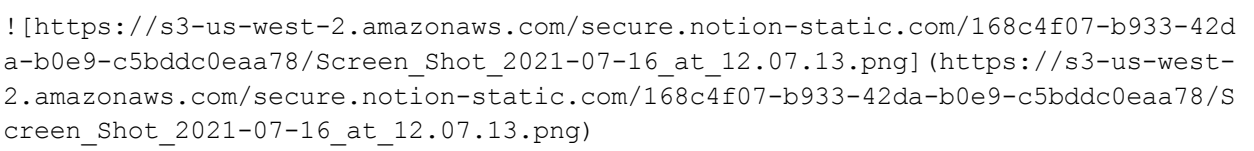
- low capture probability and data sparsity is a big problem → you can use BDRhapsody to limit the scope of experiment to a few hundred mRNAs
- inability to analyze full-length transcriptomes → can use smartSEQ2
- inability to resolve spatial information → where computation comes into play
- integrating with measurements in Microscopy, FACs, total_Seq, hashtag tech, etc.

- preparing scRNAseq data for clustering → preprocessing: align and count UMIS
- finding feature selection and do dimensionality reduction → you can visualize heterogeneity with t-SNE and UMAP
- **HVGs for preprocessing**:

    - distinguish technical noise by looking at distribution of population of cells → HVGs are identified because they are interesting biological markers
    - PCA is then applied as a result
        - visualized using UMAP and t-SNE
- Quality control → normalization → feature selection → dimensionality reduction → cell-cell distances → unsupervised clustering

Computational challenges in scRNA-seq:

- computational pipelines for handling raw data files is limited → not many tools and is still in infancy
- first step is pre-processing data → once reads are obtained from well-designed scRNA-seq experiments, quality control performed
- read alignment is next step and tools available for this procedure are used for bulk RNA and can be used here → when adding transcripts of known quantity and sequence for calibration and QC, low-mapping ratio of endogenous RNA to spike-ins is indication of low -quality library caused by RNA degradation



- following alignment, reads allocated to exonic, intronic, or intergenic features using transcript annotation in format **General Transcript**
    - only reads that map to exonic loci with high mapping quality considered for generation of gene expression matrix → N x m where N=cells, m=genes
- there is a presence of zero-inflated counts due to dropout or transient gene expression, which requires normalization to remove cell-specific bias
    - read count of gene in each cell expected to be proportional to gene-specific expression level and cell-specific scaling factors

My Project: Single-Cell RNA Sequence Analysis (Downstream) w. Interactive UI for Hodgkin's Lymphoma, Dissociated Tumor: Targeted, Pan-Cancer Panel [Single-Cell Transcriptome Analysis Reveals Disease-Defining T-cell Subsets in the Tumor Microenvironment of Classic Hodgkin Lymphoma](https://pubmed.ncbi.nlm.nih.gov/31857391/)
- cHL characterized by extensive microenvironment composed of different types of noncancerous normal immune cells → several types of T cells, B cells, eosinophils, and macrophages, and rare populations of clonal malignant Hodgkin and Reed-Sternberg cells
    - some findings support concept that HRS cells recruit immune cells to form tumor-supporting, regulatory tumor microenvironment (TME) with limited antitumor activity in cHL
    - complex interactions between HRS cells and TME remains partially understood → looking at symbiotic cellular cross-talk may lead to development of novel biomarkers and therapeutic approaches

- immune-checkpoint inhibitors like PD-1 have shown dramatic efficacy in relapsed or refractory cHL, with overall response rate of 65-87% and durable remissions of 1.5 years
    - remains unclear which cells are most important targets of immune-checkpoint inhibitors and which components are most relevant for immune-escape phenotype in cHL
- goal of this project is to characterize immune phenotype of the TIME in cHL and identify important associations between immune cell types and respective clinical outcome
- key differentiator between lymphomas and tumors is that they are derived from lymphocytes that professionally interact with other immune cells in ecosystem of microenvironment

System Requirements
Make sure you have Python3.5 or greater installed, along with anaconda setup with pip as your default package manager.
Also requires that you have experience with python and package installation + some basic biology background to understand
some of the biological comparisons drawn here with the data.

Package specifications and versions in the [.yml file](sc-rnaseq.yml) for reference.

The Dataset and Exploration
The dataset is titled Hodgkin's Lymphoma, Dissociated Tumor: Targeted, Pan-Cancer Panel which is a [Single Cell Gene Expression Dataset by Cell Ranger 4.0.0](https://support.10xgenomics.com/single-cell-gene-expression/datasets/4.0.0/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer)

The sample is collected from a disassociated lymph node tumor of a 19-year-old male. The sample was obtained from 10x Genomics and the whole transcriptome in the dataset is generated with Chromium GEM Single Cell 3' Reagent Kits. There are approximately 11,332 reads per cell (not necessarily important unless you're interested in genome assembly metrics). Out of the tumor microenvironment, 3049 cells were detected with an 97.2% confidence score for the reads being mapped to the targeted transcriptome. The total targeted genes detected are 1165.

Download links:
[Feature / cell matrix HDF5 (filtered)](https://cf.10xgenomics.com/samples/cell-exp/4.0.0/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer_filtered_feature_bc_matrix.h5)
[Feature / cell matrix (filtered)](https://cf.10xgenomics.com/samples/cell-exp/4.0.0/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer_filtered_feature_bc_matrix.tar.gz)

The UI and Getting Started
File Downloads and Setting up Anaconda
You can download the files using the links above, and save them to the directory we will create below. If you do not have anaconda already installed, you can refer to the [documentation](https://docs.anaconda.com/anaconda/install/index.html). Once you've set up anaconda, enter the following lines in your terminal:

```
>> mkdir cancer-target-main
>> cd cancer-target-main

##You can manually install this by clicking the link, but for anyone with terminal experience
>> wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer_filtered_feature_bc_matrix.h5
>> wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer/Targeted_NGSC3_DI_HodgkinsLymphoma_Pan_Cancer_filtered_feature_bc_matrix.h5

##You can also unzip manually, but this is also convenient
>> unzip cancer-target-main/filtered_feature_bc_matrix.h5
```

From here, you can download the `targeted-cancer-lymphoma.ipynb` file in the same directory. Then, run the notebook in VSCode or jupyter lab. To install jupyter lab, visit this [link](https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html)

** Advanced Startup from Github **
You can simply clone the repository and run the primary *targeted-cancer-lymphoma.ipynb* file.

> Now you're ready to follow along and understand your chosen tissue microenvironment.

Running into Errors / Troubleshooting
1. Python Package Error
    If running in Jupyter Lab or notebook, you can take a series of steps to make sure that your libraries are properly installed. For one, make sure that your conda environment has all of the required packages.

    To check your libraries, enter the following in your terminal or folder:

```

    >> conda env list
    ##If your environment is there, you can just delete it and rewrite it with an updated yml file

    >> conda env remove --name _name_of_conda_environment_
    >> conda env create -f sc-rnaseq.yml
    ##Once installed, you can activate the environment in terminal
        ##Note that the name of the environment is in the yml file, so
        ##you can go in and manually edit that name to activate it below


    >> conda activate sc-rnaseq
    ##Now, you can switch the jupyter kernel to the conda environment
    ##If all else fails, install all error-libraries in the conda env

    >> (sc-rna seq) conda install _package_name_
    >> (sc-rna seq) pip install _package_name_
    ```


2. Dataset formatting errors
    If working with 10x Genomics Data, you must make sure that it follows the specified format in

[10xDatasets](https://support.10xgenomics.com/single-cell-gene-expression/datasets). If you would like to
    test a dataset, you can go into any of these single-cell expression datasets and do the following:

    - [ ] Click on *Gene Expression - Feature/cell matrix* (raw or filtered)
    - [ ] Click on *Gene Expression' - Feature/cell matrix HDF5* (raw or filtered)
    - [ ] OPTIONAL: Download any metadata such as cells per tag, dataset summary, etc.

    - [ ] Unzip tsv.gz files, if ERROR: 79, install [The Unarchiver](https://theunarchiver.com/)
          and use the application to unzip the files
    - [ ] You can now input the absolute path of the folder that contains the matrix.mx and other
          files as the downloadable root path

3. Divide By 0s:
    Although this should never be the case thanks to error handling, check if your mitochondrial gene

plot is at 0 and if the error is being thrown in cell 7. If so, you must make sure that your Error
Threshold Input is greater than 0.

Otherwise, you must make an informed visual choice to determine the most effective way to filter
the data. You can always visit it back to make changes.

4. Stuck in an awkward loop
In some cases, this can probably be a memory error or if you've modified the code and certain flags are
misplaced. To solve this, add a `break` to any of the loops or just remove them if you do not want a dynamic
integration.

For other instances, just interrupt the cell and rerun it by adding manual inputs to the parameters -> you
can trace this back based on where the input variable is used.

5. TSNE, UMAP, or PCA Stuck
In most cases, this will take more than 6-10 seconds depending on your computer internals. To maximize
your calculations, you can add the flag `n_pcs=14` to `sc.tl__ ()` to change the number of PCs being used.

In other cases, it would be a good idea to reduce the total data you've passed through or it's just too
big. You can track how much is completed by changing the learning_rate parameter, but should always be
between 100 to 1000. Refer to [Documentation](https://scanpy.readthedocs.io/en/stable/generated/scanpy.tl.tsne.html)

Test Strategies:
Throughout this process, I had several key problems (some that took me hours, others that took days) from the technical depth and breadth of the project to the actual unit testing. I started off my journey with the hard biology by doing in-depth research aligned with my time frame, but ended up getting stuck on how to learn about the actual library. This took me some time because there were very little documentation guides or tutorials with simple walk-throughs.

I ended up spending about 6 hours learning about the biology in-tandem with the single-cell RNAsequencing exercises provided on http://rosalind.info/problems/topics/dynamic-programming/ in their dynamic programming exercises, a great youtube tutorial on sc-RNAseq by StatQuest, and a deep dive into a mouse's intestinal tissues with the Theis Lab.

The way I decided to test my code was design unit-tests for refactoring at different points. These unit tests were essentially temporary print_statements which outlined key variables and processes at each macro_function in a specific jupyter notebook cell (big advantage in testing different blocks of code). Here's an example of a simple unit-test I wrote for most of my functions that required a list of hyperparameters:

```
def tsne_umap_plot(k, var_k, color_n=3, colors=None):
   assert type(var_k) is type(adata), "Should have PCA plot"
   if colors == None:
       colors = random.sample(list(high_variable_genes_names), color_n)
   sc.pp.neighbors(var_k, n_neighbors=k)   #calculate k-neighbours

   #UMAP
   print('Generating UMAP...')
   sc.tl.umap(var_k)
   sc.pl.umap(var_k, color=colors, title=f'PCAk={k} UMAP')

   #TSNE
   print('Generating TSNE...')
   sc.tl.tsne(var_k)
   sc.pl.tsne(var_k, color=colors, title=f'PCAk={k} tSNE')


flag_check = False #Flag for checking if input is valud
while flag_check == False:
   k_neighbours = int(input('Provide a possible value for k to designate the total
number of neighbours (default=10): '))
   try: #made sure that the integer for k did not exceed possible clusters and was
calculated before plotting
       tsne_umap_plot(k_neighbours, pcaHVG10, 4)
       print('Successful plot generation.')
       break
   except:
       print('Please provide a valid integer value for k.')
```

```
def check_AnnData(data):
   assert data.uns['umap'], "Need to create UMAP plot"
   assert data.uns['pca'], "Need to create pca plot"
```

```
   assert 'X_pca' and 'X_umap' and 'X_tsne' in data.obsm, "Requires Topoligical
Representation Of Data, Refer to sc.tl.tsne, umap"



check_AnnData(pcaHVG10)
```

```
#Unit test -> check if PCA plot effectively matched at knee point and did not stall

def shape_check(new_pca_df, hvg_data):
   assert new_pca_df.shape[-1] < hvg_data.shape[-1], f"PCA Dim: {new_pca_df.shape[-1]}
\nHVG Dim: {hvg_data.shape[-1]}"
   assert new_pca_df.shape[0] == hvg_data.shape[0], f"Incorrect feature selection,
choose correct dataset"

   print(f"PCA Dim: {new_pca_df.shape[-1]} \nHVG Dim: {hvg_data.shape[-1]}")



shape_check(PC_frame, HVG)

#Output
PCA Dim: 17
HVG Dim: 1089
```

These are just some of the several examples of where I've used unit testing to identify roots of key problems I specifically ran into, most of which revolved around the quality of data and how I formatted it. I've also included a set of try and except handles, but instead of accepting all kinds of errors, it provides useful insights on specific kinds of errors that I found to have specifically during my development process and provide tips:

```

def calc_filter_nums(data):
   sc.pp.filter_cells(data, min_genes=200)
   sc.pp.filter_genes(data, min_cells=3)

   sc.pl.scatter(data, x='total_counts', y='pct_counts_mt', color='n_genes_by_counts',
title='NoQC_PCT_COUNTS')
   sc.pl.scatter(data, x='total_counts', y='n_genes_by_counts', color='pct_counts_mt',
title='NoQC_N_GENES_COUNTS')

   filter_threshold_gene_counts = int(input('Please enter a gene_count filter slice
value:'))
```

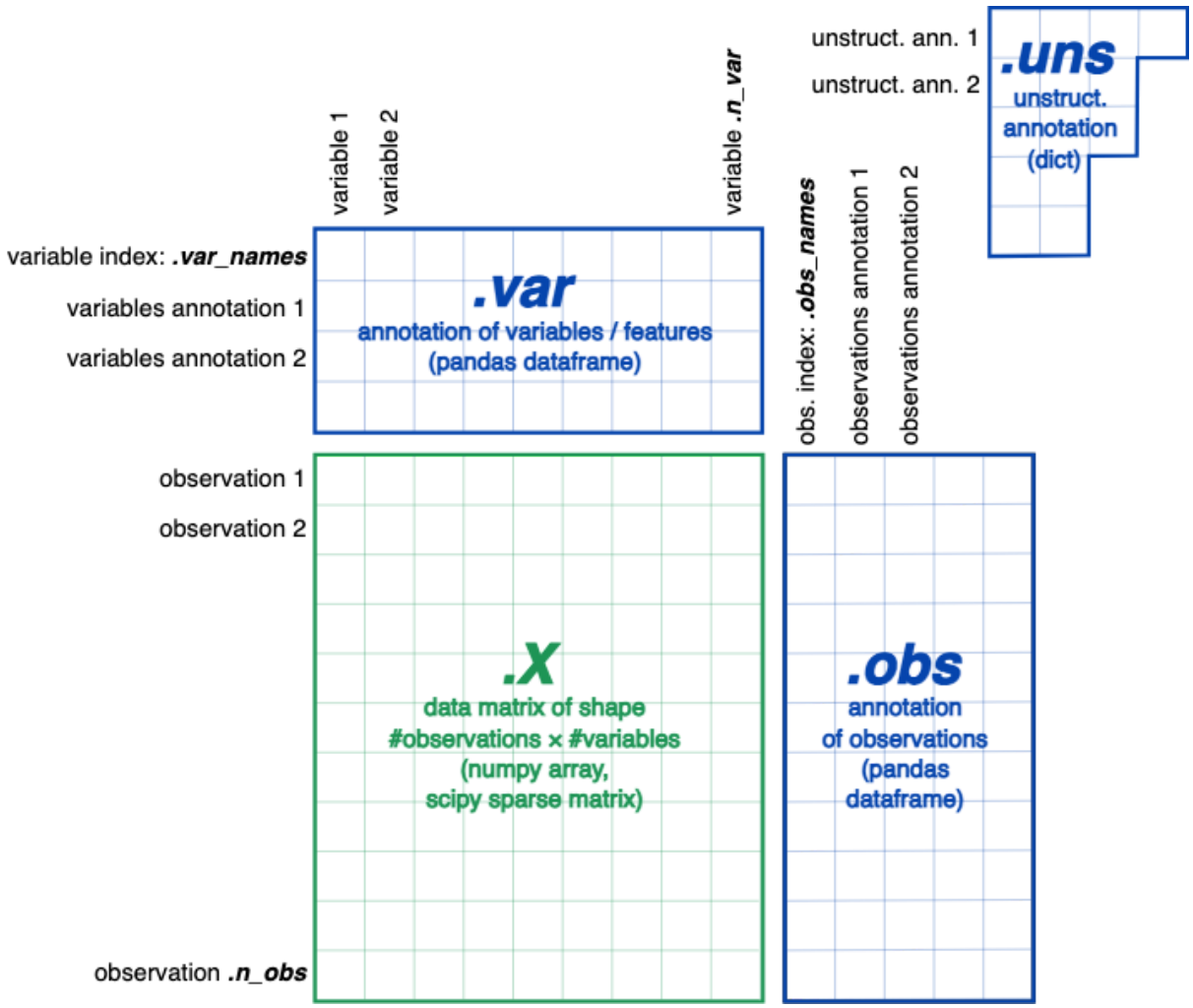
```
   filter_threshold_pct_counts = int(input('Please enter a pct_count_mt filter slice
value:'))

   qc_gc = data[data.obs.n_genes_by_counts < filter_threshold_gene_counts, :]

   print('Here is your updated, quality-controlled scatter.')
   sc.pl.scatter(qc_gc, x='total_counts', y='n_genes_by_counts',
color='pct_counts_mt', title='QC_N_GENES_COUNTS')
   try:
       qc_gc = qc_gc[qc_gc.obs.pct_counts_mt < filter_threshold_pct_counts, :]
       sc.pl.scatter(qc_gc, x='total_counts', y='pct_counts_mt',
color='n_genes_by_counts', title='QC_PCT_COUNTS')
   except ZeroDivisionError: #Based on mitochondrial gene concentration
       print('Your total percent of mitochondrial genes are 0%')

   return qc_gc
```

Throughout my code, I also found that creating deepcopies with immutable features was key because I can now have multiple endpoints with respect to a function and then analyze that more deeply during debugging (i.e. if I wanted to change the number of cells I remove based on the total genes they belong to, the Annadata object would be local to that said filtering process).

I spent a lot of time looking at stack exchanges and forums for issues I had with general biology and trying to understand why some of the hyperparameters like the min_mean, filter_threshold, and scaling processes were so arbitrary.

One specific bit of advice I used from a website called biostars.org was to read a lot of medium articles on some of the fundamental math concepts and then connect them with the documentation.

I ended up using this process, and discovered some of the more fundamental workings of the libraries I was using.



Observations like these helped me better understand how and why certain things were notated and how I can call upon them for general inference (realized through **pattern recognition** that each time I performed a function on top of a dataset, the dataset would store a call-back attribute to that function [variable acted as a github pull request log] so I could the use these different endpoints to remove and add different variables and see how they would react to functions like plotting and normalization).

I also spent a lot of time looking at a lot of the web portals for assistance including:

- the Gene Expression Analysis Resource U Maryland
- the Galaxy Project for the Human Cell Atlas [tweet] U Freiburg
- the Expression Atlas EMBL-EBI

The galaxy project helped me better understand cell lineage and the importance of clustering while at the same time analyzing specific gene markers and their relationship with neighbouring cells.

The biggest problem however that I documented early on was package environments. While this is quite trivial, this took me 3 days to sort out because of a lot of system design errors I had with my current setup.

In the process, I learned a lot about environments and package management, and how there are system-level and project-level environments that can be edited and called upon (through initializing kernels) to give the code access to only certain folders (preventing total root access).

This also helps for better, more contained coding environments but at the same time keeps everything centralized. Key issues I had was understanding why certain libraries installed and others didn't (found that installing through terminal was inefficient because it was default-synced to my base anaconda environment, so I then installed directly in the ipynb file).

Additionally, I learned that yml files are a great way of saving and sending package environments that are prebuilt. In fact, I created a little script that can update my .yml files automatically with each new pip installation. This helped me better understand how python libraries work with other languages and interfaces such as R, SQL, and the different web portal ecosystems.

I had also never worked with tsv data formats and annotated datasets, so understanding them was a very difficult process. I overcame this by copy-pasting specific examples and tags in the datasets and learning more about them on google. This helped me realize that sequence-tagging resembles the actual mRNA in the transcriptome, and the features act as gene activity representations. The matrix file contains a read set which can be matched and mapped back in a sparse plot through a 10x reader.

When looking at the criteria, I also found out about perhaps using version control to showcase my updates and track my progress in my code. I took this as an opportunity to open-source and use systems and project-organization systems to better understand how real-world projects are designed at companies while demonstrating my ability to document and log progress in a tight format. You can read about my pull-requests, branch management, and remote set up through the revision history in my github repository. Initially, I hadn't use git all that much because it always felt quite useless, but what I found useful in this project specifically was that it helped me checkpoint certain parts of the development process where I could test new features in 1 branch, and merge it after to my main branch later on.

This was another useful way to test features that would not impact my entire code base, and followed a useful programming practice that most developers would require from you. I found this very useful especially since my VSCode extension set was designed around version control for the jupyter notebook, so all changes with each autosave would be logged in the repository.