Image-Based Analysis Template Script

**Overview:** This script is designed to conduct downstream analysis for image-based Spatial OMICS technologies such as Akoya, Cellscape, and CosMx Protein. The script is run on the rsc\_conda environment.

**Input**: For Akoya and Cellscape, users are required to conduct cell segmentation on QuPath and the inputs into the script are the txt cell detection outputs from QuPath. For CosMx, the script takes in the default CosMx flat file outputs.

**Functions:**

*clr\_normalize\_each\_cell(adata, inplace=True)*

This function is a Python translation of the Seurat centered-log-ratio normalization function. It takes in an anndata and normalizes it in place.

*qupath2Anndata(file\_path)*

This function takes a QuPath output txt file and loads it into anndata.

*getMeanSNR(expMat)*

This function returns the mean signal-to-noise ratio for each gene given an expression matrix. Mean SNR is a great way for us to examine gene quality during imaging.

*plot\_stacked\_bar(list\_of\_lists, analysis\_type, group\_names, title\_name, dir=None)*

This function plots a stacked bar plot such that each stacked bar is representative of a different group. This plotting function is used for cell proportion analysis.

* *list\_of\_lists*: A nested list in which list is a that of all the cell types for each cell of a given group.
* *analysis\_type*: String that denotes what will be displayed on the y axis label.
* *group\_names*: A list of strings for names for the groups to be displayed on the x axis.
* *title\_name*: String that denotes the title of the plot.
* *dir*: A file path. If provided, the plots will be saved instead of being shown.

*load2Anndata(files=None)*

This function takes a list of files and load it into a list of anndatas. If a files list is not provided, it will read every single txt file in the current directory as anndatas.

*filterAnndatas(adatas, channel\_name='DAPI', DAPI\_filter\_val=10, fl\_std\_val=2, size\_std\_val=2, show\_hist=True)*

This function filters the anndatas given different parameters. DAPI filtration is important for Akoya and Cellscape samples because StarDist run in QuPath often introduces artifacts such that in areas where there are no cells it will find cells. DAPI filtration ensures that none of these false cells will be included. Fluorescence filtering filters out cells that are low expressing and are likely noise. Cell size filtration filters out cells with overwhelming large size due to cell segmentation issues. The cutoff for DAPI filtration is arbitrary and the value needs to be played around with to determine optimal filtration. The cutoffs for fluorescence and cell size is based standard deviation such that *fl\_std\_val* and *size\_std\_val* of 2 means that cells that are lower than 2 standard deviations below the mean for fluorescence and cells that are higher than 2 standard deviations above the mean for cell sizes will be filtered out as noise.

* *adatas*: A list of anndatas.
* *channel\_name*: The gene name for the DAPI channel. By default it is ‘DAPI’. This can vary across technologies.
* *DAPI\_filter\_val*: A float by which the DAPI filtration is performed.
* *fl\_std\_val*: The standard deviation cutoff for fluorescence.
* *size\_std\_val*: The standard deviation cutoff for cell size.
* *show\_hist*: Whether histograms are shown.

*removeGenes(adatas, genes\_to\_remove=['DAPI'])*

This function removes genes from the list of anndatas. By default, it removes the DAPI gene as it is not necessary for downstream analysis. However, other genes can include problematic genes with low SNR or genes with too many imaging artifacts.

*clrNormalizeAnndatas(adatas)*

This function performs CLR normalization on all anndatas in the list.

*mergeNSave(adatas, file\_name)*

This function takes in a list of anndatas and merges them together. It then saves it as an h5ad file such that batch correction can be conducted.

*clusterNDotPlot(adata, n\_comps=None, n\_neighbors=30, resolution=0.5, figsize=(6,4), dpi=(100), dotsize=3)*

This function conducts number of neighbor calculations, UMAP, and Leiden clustering. It then plots the Leiden clustering as well as the batches on a UMAP. Furthermore, it plots a dot plot that can be used for manual cell annotation. It also detects whether the anndata has been batch corrected in which case it will use the batch corrected PCA representation for downstream analysis. If it detects that no batch correction has taken place, it will also conduct a PCA analysis.

* *adata*: The anndata object.
* *n\_comps*: Number components for PCA and number of neighbors. By default, the function will use the maximum number of components available.
* *n\_neighbors*: The number of neighbors used for number of neighbors calculation.
* *resolution*: The resolution by which Leiden clustering is conducted.
* *figsize*: The size of the figure in the form of (width, height).
* *dpi*: The “dot per inch” which denotes the quality of the plot.
* *dotsize*: the size of the dots for the UMAP.

*clusterMap(adata, color='leiden')*

This function plots the clusters, cell types, or niches onto the tissue given the spatial information. User must denote what type of coloring they want to use and by default, it plots the Leiden clustering.

*manualAnnotateCells(adata, annotations, figsize=(6,6), dpi=(100), dotsize=5)*

This function takes a list of manual annotations and maps it using the clusters. The list of manual annotation must correspond with the clusters from 0 to x.

*monkeyNiche(adata)*

This function conducts niche analysis for each individual batch in the anndata object using the Monkeybread package.

*collapseNiche(adata, resolution=0.5)*

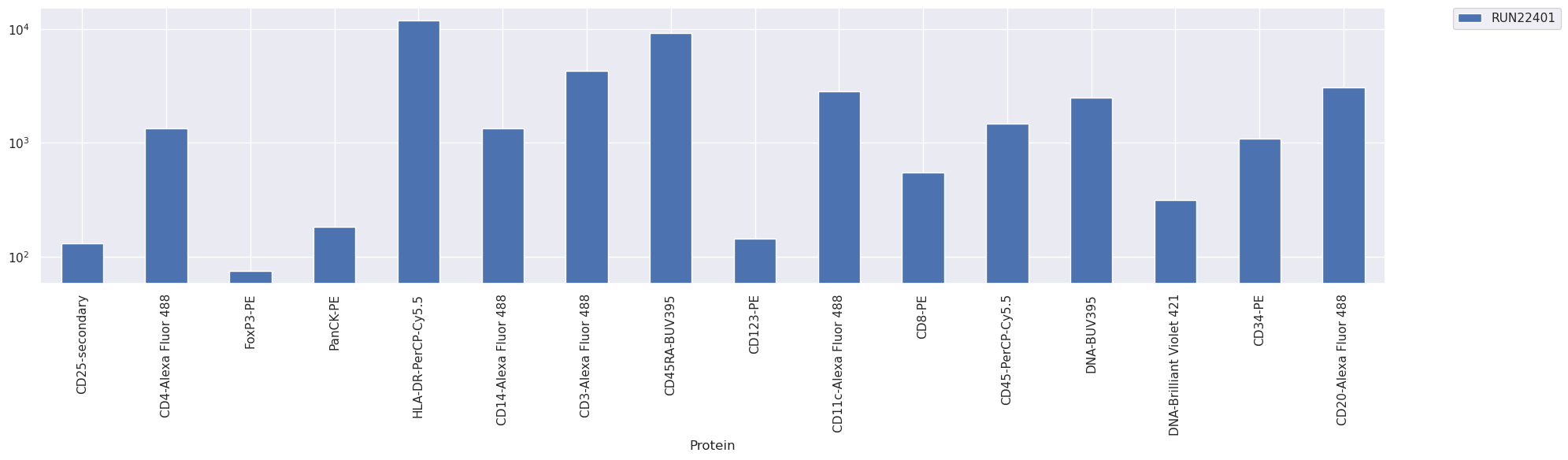
This function collapses the niches generated in individual batches using clustering such that we can find common niches across the different samples. This only needs to be used if we are calculating niches across multiple samples.

**Tutorial:** Please refer to the example run on the template script for the code to run everything. This tutorial will describe the images generated from the run as well as the next steps.

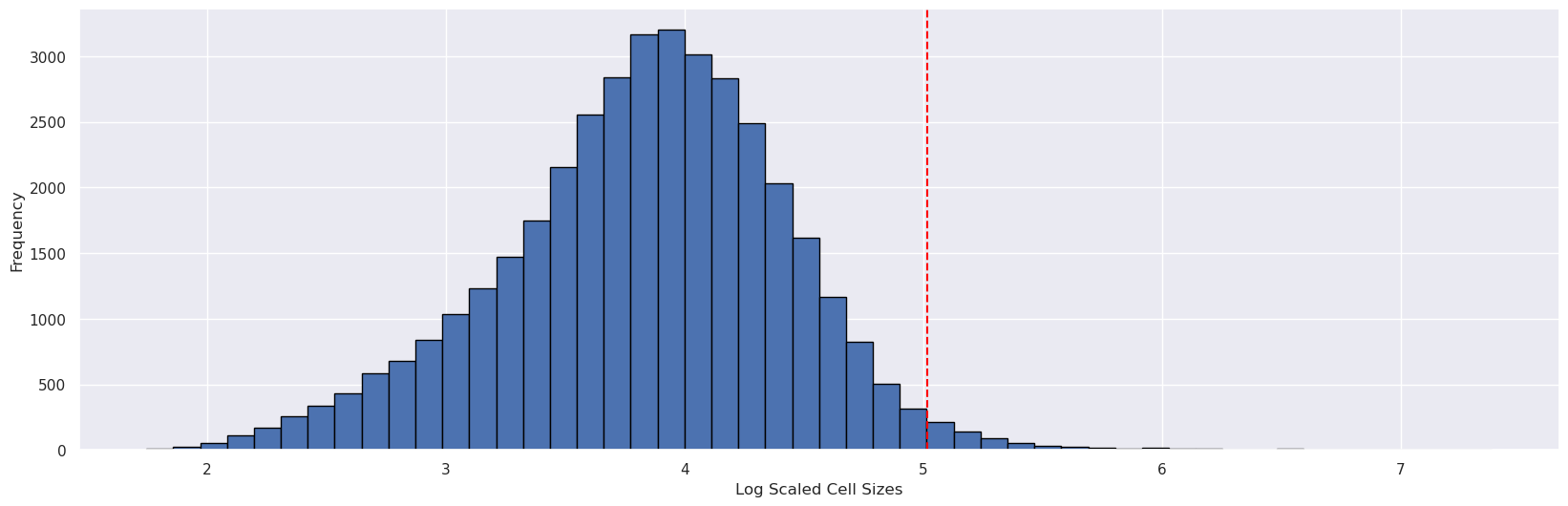
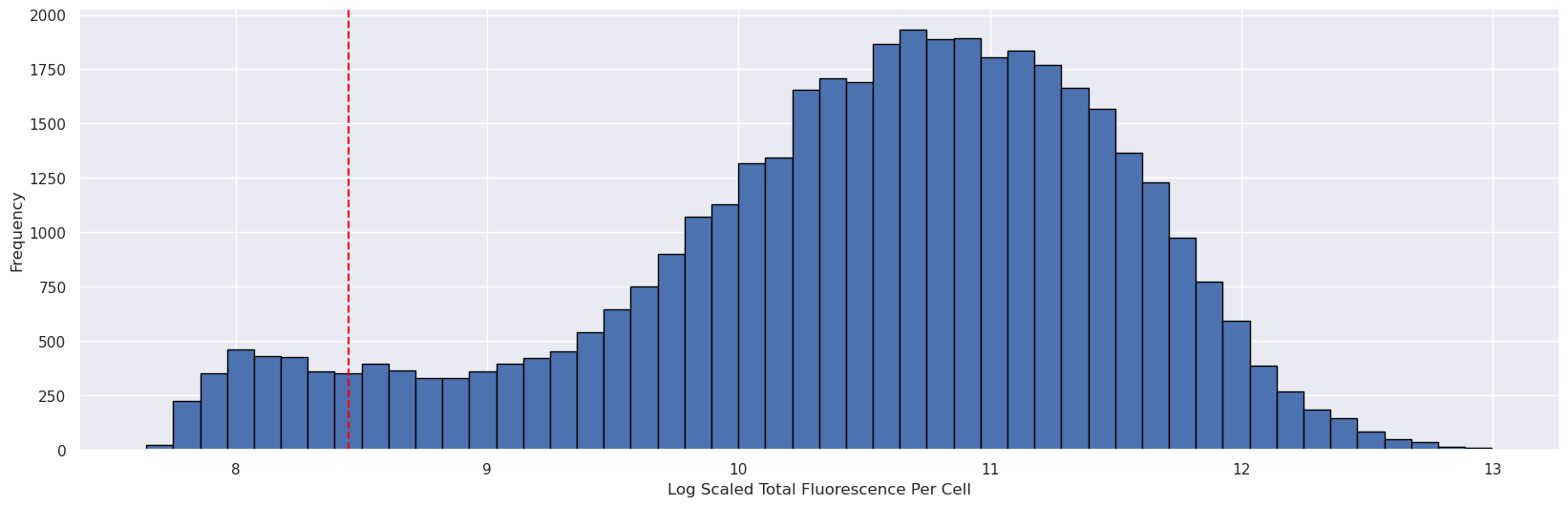
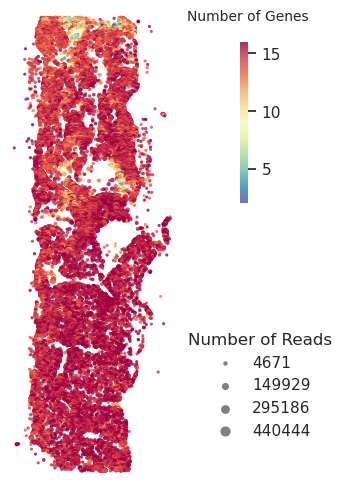
\*NOTE: The tutorial results are meant to be an example. It is NOT an accurate analysis.

*Loading and preprocessing*

First, we load the anndatas using the functions listed above. The example we used will create a list of anndata objects so that we can examine multiple objects at once. For QC, we can use Mean SNR to see the quality of our run, see the plot below:



We can see that in this run, genes like FoxP3 and CD123 have lower SNR than other genes meaning that the staining for these genes are noisy and potentially have issues. To fix some of these issues, we can conduct some preprocessing to remove false cells and noise as shown by the histograms and the QC plot.



The red lines on the histograms shows our cutoff. These cutoffs can be played around to see what the optimal filtration should be. The histogram at the top shows the total fluorescence per cell while the histogram and the bottom show the cell sizes for each cell. And the QC plot below shows if all the false cell artifacts have been filtered out.

*Batch correction*

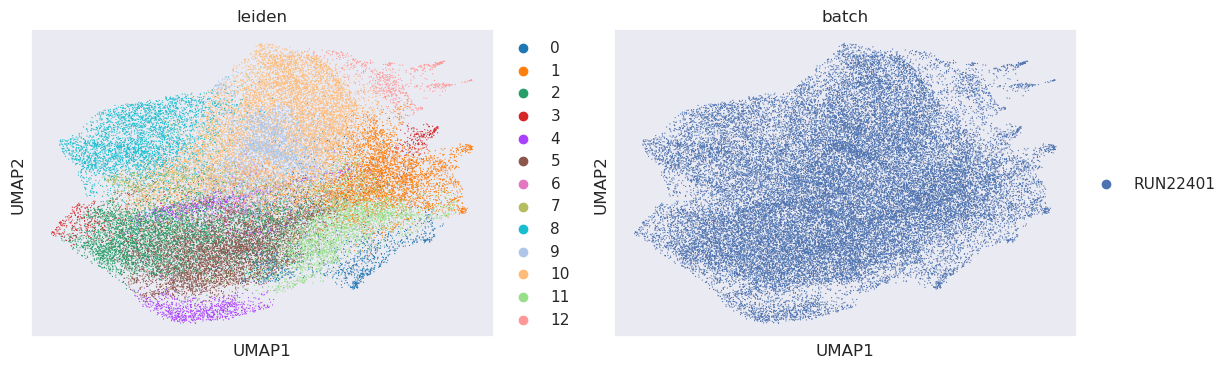
One preprocessing is complete, we merge the anndatas together and save it as an h5ad file to conduct batch correction. Due to various reasons, we must do batch correction using a Python script as opposed to doing it using Jupyter Notebook. The command to run batch correction is as follows:

*export NUM\_THREADS=1 export OPENBLAS\_NUM\_THREADS=1 export OMP\_NUM\_THREADS=1*

*python RunHarmony.py adata.h5ad adata.h5ad*

We assume the h5ad files are named adata.h5ad.

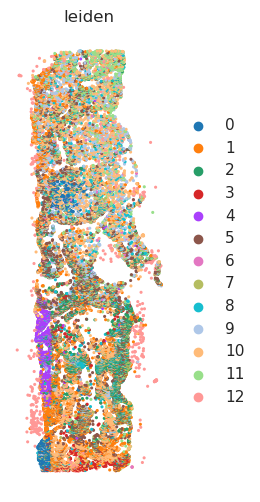
*Clustering and generating dot plot*

We use convention clustering techniques to generate the cluster plots as show here: 

The left shows Leiden clustering mapped onto the UMAP. Sometimes if a data is extremely noisy, it may cluster the noise together as one cluster. When that happens, we can simply find the cluster and remove all cells in that cluster from our analysis all together. The UMAP on the right is mapping the batches onto the UMAP. Since for the example, we only used one sample, thus we didn’t have multiple batches to map onto the UMAP; however, with multiple samples, we can see if the samples require batch correction and if they have already been batch corrected, whether the batch correction actually worked.

Dot plot and the cluster map are also generated using the same function and we use the dot plot to annotate the clusters by looking at clusters and which genes they highly express:

A diagram of a number of dots

Description automatically generated with medium confidence

*Annotation and cell proportion analysis*

After manually annotating the clusters, we can map the annotations back onto the clusters using the function mentioned above. We can examine how well the annotation performs by reexamining the dot plot and the UMAP.

A close-up of a chart

Description automatically generatedA close-up of a cell type

Description automatically generated

Once again, these are not accurate annotations. After examining the annotations we can examine the cell proportion differences. Again, since we are using one sample for the example, you only get one stacked bar. However, with multiple samples, we can get multiple stacked bars to compare against each other.

A screenshot of a cell type proportions graph

Description automatically generated

A colorful dots and text

Description automatically generated with medium confidence*Niche analysis*

We conduct niche analysis using the Monkeybread package. We generate the niches individually for each sample because niche analysis takes in spatial information as well as cell typing to create the niches. Here are the niches mapped onto the tissue:

After generating the niches, we must find common niches across the samples. This run did not include the niche collapsing function; however, it is included in the script. To collapse the niches, we create a matrix of each individual niches, and their cell type proportions and cluster them together to make general niches for all samples. This way we can compare how each general niche differs across samples via looking at cell type proportions for these general niches:

A graph of different colored bars

Description automatically generated with medium confidence