In [ ]:
```
# Description:
"""
This script processes a single-nucleus RNA-seq dataset from afca, removes cells
and filters cell types with fewer than 200 total cells or fewer than 100 cells p
it filters out lowly expressed genes, while retaining those expressed in at leas
Each processed cell type is saved as a separate .h5ad file.
"""

# Import the libraries
import os
import pandas as pd
import scanpy as sc
import anndata as ad
import numpy as np
import re
```

In [ ]:
```
# Set the functions
def split_by_batch_prefix(name):
    """
    Splits the index into two parts:
    - Part before 'AFCA' or 'FCA'
    - The rest starting with 'AFCA' or 'FCA'
    """
    match = re.search(r'(AFCA|FCA)', name)
    i = match.start()
    return name[i:]
```

In [ ]:
```
# load the dataset data and get the metadata
adata = ad.read_h5ad("/hpc/shared/onco_janssen/dhaynessimmons/projects/ageing_fl
```

In [ ]:
```
# print out the basics
print([i for i in adata.obs.columns])
print("shape of full data: ", adata.shape)
#value couns for important columns
print("\nAge value counts: ", adata.obs["age"].value_counts())
print("\nSex value counts: ", adata.obs["sex"].value_counts())
print("\nDataset value counts: ", adata.obs["dataset"].value_counts())
```

In [ ]:
```
# Get the unique afca cell types that are part of the enterocyte lineage
entero_subtypes = []
for cell_type in sorted(adata.obs['afca_annotation'].unique().tolist()):
    if "enterocyte" in cell_type:
        entero_subtypes.append(cell_type)
print("\nEnterocyte subtypes: ", entero_subtypes)
```

In [ ]:
```
# get cells where sex is neither F nor M
mix_adata = adata[(adata.obs.sex != "female")&(adata.obs.sex != "male")]
print(mix_adata.obs.shape)
```

In [ ]:
```
# Remove them from the dataset
mf_adata = adata[~(adata.obs.index.isin(mix_adata.obs.index))&(adata.obs['afca_a
# Only keep the cells that are in teh enterocyte lineage
mf_adata = mf_adata[mf_adata.obs['afca_annotation'].isin(entero_subtypes)]

# get the indiv from the row name
```

```
mf_adata.obs['indiv'] = mf_adata.obs.index.map(lambda x: split_by_batch_prefix(x
print(mf_adata.obs.head())
```

In [ ]:
```
# Get the observation dataframe as a pandas dataframe
mf_adata_obs = mf_adata.obs.copy()
print(type(mf_adata_obs))
cell_list = []

# Set the save path
save_path = "/hpc/shared/onco_janssen/dhaynessimmons/projects/ageing_flies/data/
os.makedirs(save_path, exist_ok=True)

# Gene list of interest
gene_list = [
    "Su(var)205", "Su(var)3-9", "G9a", "HP1b", "HP1c", "HP4",
    "HP5", "HP6", "ADD1", "Su(var)2-HP2", "Su(var)3-7", "Lam",
    "LamC", "LBR", "Kdm4A", "Kdm4B", "His2Av", "His3.3A", "His3.3B"
]
```

In [ ]:
```
# Evaluate the QC of the new adata object
cell_cnt = mf_adata_obs.shape[0]
print("number of cells: ", cell_cnt)
print("Min number of genes expressed : ", mf_adata_obs.n_genes_by_counts.min())
```

In [ ]:
```
# Check that each age group has at least 100 cells
age_grouped = mf_adata_obs.groupby('age', observed=False).size()
min_value = age_grouped.min()
print("Minimum number of cells in an age group: ", min_value)
if min_value < 100:
    print("Not enough cells in an age group to proceed with analysis")
else:
    print("Sufficient cells in each age group to proceed with analysis")
    # Create a new adata object with the cell type data
    cell_list.append(cell_type)
del mf_adata_obs
```

In [ ]:
```
# Create a new adata object
cell_group_adata = mf_adata
print("\nshape of cell type data: ", cell_group_adata.shape)

# ----------- Custom gene filtering starts here ------------ #

# Compute how many cells express each gene
gene_expression_counts = np.array((cell_group_adata.X > 0).sum(axis=0)).flatten(

# Get gene names
gene_names = pd.Index(cell_group_adata.var_names)

# Genes expressed in >= 3 cells
genes_expressed_enough = gene_expression_counts >= 3

# Create a boolean mask to keep genes that are either:
# - expressed in enough cells
# - or present in the gene_list
gene_list_set = set(gene_list)
genes_in_list = gene_names.isin(gene_list_set)

# Combine masks
genes_to_keep = genes_expressed_enough | genes_in_list
```

```
# Filter genes
cell_group_adata = cell_group_adata[:, genes_to_keep].copy()

print("\nshape of cell type data after filtering: ", cell_group_adata.shape)

# ----------- End of custom filtering ------------ #

# Save the new adata object
cell_group_adata.write_h5ad(f"{save_path}entero_expr_set.h5ad")
print("\n\tSaved the new adata object to: ", f"{save_path}entero_expr_set.h5ad\n
```