```
#============================================================================
#====== SLDM hw6, PROBLEM 1 ================================================
#============================================================================


# Need to pip install: "phate", "scprep" on the command line before running
# the rest of this code in an IDE.


#============================================================================
#==== Loading the 10X data ==================================================
#============================================================================
# PART A from the homework

# Import the data (code given on the tutorial website at:
# https://github.com/KrishnaswamyLab/PHATE/blob/master/Python
#        /tutorial/EmbryoidBody.ipynb)
import os
import zipfile
from urllib.request import urlopen
download_path = os.path.expanduser("~")
print(download_path)

if not os.path.isdir(os.path.join(download_path, "scRNAseq", "T0_1A")):
    if not os.path.isdir(download_path):
        os.mkdir(download_path)
    zip_data = os.path.join(download_path, "scRNAseq.zip")
    if not os.path.isfile(zip_data):
        with urlopen("https://data.mendeley.com/datasets/v6n743h5ng"
                     "/1/files/7489a88f-9ef6-4dff-a8f8-1381d046afe3"
                     "/scRNAseq.zip?dl=1") as url:
            print("Downloading data file...")
            # Open our local file for writing
            with open(zip_data, "wb") as handle:
                handle.write(url.read())
    print("Unzipping...")
    with zipfile.ZipFile(zip_data, 'r') as handle:
        handle.extractall(download_path)
    print("Done.")

# Need these libraries
import pandas as pd
import numpy as np
import phate
import scprep

# Now use scprep to import data into a Pandas dataframe, using the
# scprep.io.load_10x function.
sparse=True
T1 = scprep.io.load_10X(os.path.join(download_path,"scRNAseq", "T0_1A"),
                        sparse=sparse,
                        gene_labels='both')
T2 = scprep.io.load_10X(os.path.join(download_path, "scRNAseq", "T2_3B"),
                        sparse=sparse,
                        gene_labels='both')
T3 = scprep.io.load_10X(os.path.join(download_path, "scRNAseq", "T4_5C"),
                        sparse=sparse,
                        gene_labels='both')
T4 = scprep.io.load_10X(os.path.join(download_path, "scRNAseq", "T6_7D"),
```

```
                          sparse=sparse,
                          gene_labels='both')
T5 = scprep.io.load_10X(os.path.join(download_path, "scRNAseq", "T8_9E"),
                          sparse=sparse,
                          gene_labels='both')
T1.head()

# Now, merge all datasets and create a vector representing the time point of
# each sample
EBT_counts, sample_labels = scprep.utils.combine_batches(
    [T1, T2, T3, T4, T5],
    ["Day 0-3", "Day 6-9", "Day 12-15", "Day 18-21", "Day 24-27"],
    append_to_cell_names=True
)
del T1, T2, T3, T4, T5 # removes objects from memory
EBT_counts.head()


#============================================================================
#===== Preprocessing: filtering, normalizing, and transforming ==============
#============================================================================
# This is PART B from the homework (technically most of this isn't asked for in
# the homework, but I figured it made more sense to follow the tutorial exactly

# Remove (suspected) dead cells
mito_genes = scprep.utils.get_gene_set(EBT_counts, starts_with="MT-")
# Get all mitochondrial genes. There are 14, FYI.
scprep.plot.plot_gene_set_expression(EBT_counts, mito_genes, percentile=90)
# Plot number of cells that have a certain amount of mitochondrial RNA,
# remove cells that are above the 90th percentile. (Line below)
EBT_counts, sample_labels = scprep.filter.filter_gene_set_expression(
    EBT_counts, mito_genes,
    percentile=90,
    keep_cells='below',
    sample_labels=sample_labels)

# Now filter out cells that have either very large or very small library sizes.
# Library size is somewhat analogous to sample size. We'll eliminate the
# bottom 20% of cells for each sample.
scprep.plot.plot_library_size(EBT_counts, percentile=20)
EBT_counts, sample_labels = scprep.filter.filter_library_size(
    EBT_counts, percentile=20,
    keep_cells='above',
    sample_labels=sample_labels,
    filter_per_sample=True)

# Now remove rare genes (genes expressed in 10 or fewer cells)
EBT_counts = scprep.filter.remove_rare_genes(EBT_counts, min_cells=10)

# Normalization: accounting for differences in library sizes, divide each cell
# by its library size and then rescale by the median library size.
EBT_counts = scprep.normalize.library_size_normalize(EBT_counts)

# Transformation: use square root transform (similar to using log transform
# but has the added benefit of dealing with 0's automatically).
EBT_counts = scprep.transform.sqrt(EBT_counts)


#============================================================================
#===== Applying PHATE to data ===============================================
#============================================================================
```

```
# (In the tutorial, this section is "Embedding Data Using PHATE")

# Default parameters for the PHATE function are:
# k: number of nearest neighbors, default is 5
# a: alpha decay, default is 40
# t: number of times to power the operator, default "auto", 21 for these data
# gamma: informational distance constant, default is 1.
phate.PHATE()
phate_operator = phate.PHATE(n_jobs=-2)
Y_phate = phate_operator.fit_transform(EBT_counts)

# Now plot using phate.plot.scatter2d
phate.plot.scatter2d(Y_phate,
                     c=sample_labels,
                     s=3,
                     figsize=(12,8),
                     cmap="Spectral")

# PART D (from homework): run PHATE on the data using a different value of t.
# Plot the PHATE coordinates colored by time point and include the plot.
# Based on the results, do you think your chosen value of t is better than the
# parameter chosen using the "knee point" of the VNE plot (the default value)?
# Will the VNE be higher or lower for your chosen value of t than that
# selected (by default) in part(b)?
phate_op_2 = phate.PHATE(n_jobs = -2,
                         t = 30)
Y_phate_2 = phate_op_2.fit_transform(EBT_counts)

# Now plot using phate.plot.scatter2d having used t = 30
phate.plot.scatter2d(Y_phate_2,
                     c=sample_labels,
                     s=3,
                     figsize=(12,8),
                     cmap="Spectral")

# PART E (from homework): run PHATE on the data using default parameters to
# obtain 3d coordinates. Plot the 3d coordinates. Rotate the plot such that
# it's different from what the tutorial has.
phate.plot.scatter3d(phate_operator, c=sample_labels, s=3, figsize=(8,6),
                     cmap="Spectral")
# This saves the 3D plot as a gif
phate.plot.rotate_scatter3d(phate_operator, c=sample_labels,
                            s=3, figsize=(8,6), cmap="Spectral",
                            filename="phate.gif")
# This saves the 3D plot as an MP4 (which is also a cool gun in my opinion)
phate.plot.rotate_scatter3d(phate_operator, c=sample_labels,
                            s=3, figsize=(8,6), cmap="Spectral",
                            filename="phate.mp4")
```