Khan Inan
BI-GY 7633
HW 5 extra credit

*I apologize, I tried my best but I found this homework to be difficult. I hope the professor goes over the basic Seurat workflow, so I can understand it better. Thank you

Step 0: Please install and load the following packages

```
install.packages("dplyr")
install.packages("Seurat")
install.packages("patchwork")
library("dplyr")
library("Seurat")
library("patchwork")
```

Step 1: Load the TSV file

```
```{r}

pdac_data <- read.table("C:/Users/khani/Downloads/hw 5/GSM3036909.tsv",
header = TRUE, sep = "\t", row.names = 1, check.names = FALSE)
```
```

Step 2: Create a Seurat object.

```
library(Seurat)

pdac1 <- CreateSeuratObject(counts = pdac_data, project = "pdac1")
pdac1 <- FilterCells(pdac1, min.cells = 3)
pdac1 <- FilterCells(pdac1, min.features = 200)
```

Step 3: Label the Mitochondrial genes

```
pdac1 <- PercentageFeatureSet(pdac1, pattern = "^MT-")
```

Step 4: Visualize the distribution

```
VlnPlot(pdac1, features = c("nCount_RNA", "nFeature_RNA", "percent.mt"),
ncol = 3)
```

Step 5: Filter data

```
pdac1 <- FilterCells(pdac1, subset.names = nFeature_RNA, low.thresholds =
200, high.thresholds = 2500)
pdac1 <- FilterCells(pdac1, subset.names = percent.mt, low.thresholds = 0,
high.thresholds = 5)
```

Step 6: Normalize data

```
pdac1 <- NormalizeData(pdac1, normalization.method = "LogNormalize",
scale.factor = 10000)
```

Step 6: Calculate gene variation

```
# Calculate gene variation
pdac1 <- FindVariableFeatures(pdac1, selection.method = "vst", nfeatures =
2000)

# Update Seurat object
pdac1 <- ScaleData(pdac1, features = rownames(pdac1), display.progress =
FALSE)

```

Step 7: Scale data

```
pdac1 <- ScaleData(pdac1, vars.to.regress = "nGene")

```

Step 8: PCA

```
pdac1 <- RunPCA(pdac1, features = pdac1@var.genes, ndims.print=1,
verbose=FALSE)

```

Step 9: Visualize data using VizDimLoadings and DimPlot functions. Can you tell from the PCA analyis, the number of cell types that are present?

```
library(RColorBrewer)
palette(brewer.pal(12, "Paired"))

DimPlot(pdac1, reduction = "pca", label = TRUE, pt.size = 0.5, group.by =
"ident")

```

It is difficult to determine the number of cell types by solely looking at the PCA analysis but it is possible to determine the variations between the cells and the amount of clustering that is present. It is also possible to find outliers within the data, but in order to determine the number of cell types we would need to do clustering analysis on the PCA data that is reduced

Step 10: PCA heatmaps

```
# Create heatmaps of the first 10 dimensions
DimHeatmap(object = pdac1, dims = 1:10, cells = 1:200, balanced = TRUE)
```

Step 11: Dimensionality

```
# Run jackstraw analysis
pdac1 <- JackStraw(pdac1, num.replicate = 100, dims = 1:20)

# Plot results
JackStrawPlot(pdac1, dims = 1:20)

```

Plot the results for the first 20 dimensions

```
# Run JackStraw analysis
set.seed(123)
pdac1 <- JackStraw(pdac1, num.replicate = 100, dims = 1:20)

# Plot JackStraw results
JackStrawPlot(pdac1, dims = 1:20)

```

Use the elbow plot

```r
# Get the jackstraw p-values and q-values
pvals <- JackStraw(pdac1, num.replicate = 100)$pvals
qvals <- JackStraw(pdac1, num.replicate = 100)$qvals

# Create a data frame of the results
jack_df <- data.frame(Dimension = 1:20, Pvals = pvals[1:20], Qvals =
qvals[1:20])

# Plot the results
ggplot(jack_df, aes(x = Dimension, y = -log10(Qvals))) +
  geom_line() +
  geom_point() +
  xlab("Principal Component") +
  ylab("-log10(Q-value)") +
  theme_classic()

```

Step 12: Clustering.

```r
# Find neighbors and identify clusters
pdac1 <- FindNeighbors(pdac1, dims = 1:9)
pdac1 <- FindClusters(pdac1, resolution = 0.5)
```

Step 13: Perform a UMAP analysis using the first 9 dimensions using RunUMAP and then visualize it using DimPlot.

How many clusters do you get? How many possible mistakes do you see?
I see about 4 separate clusters and since there are some outliers I would say there are about 2 or 3 separate mistakes that are clearly identifiable

```r
# Perform UMAP analysis
pdac1 <- RunUMAP(pdac1, dims = 1:9)

# Visualize UMAP projection
DimPlot(pdac1, reduction = "umap")

```

Step 14: Identify the markers that compare each cluster agains all. Report only positively markers. Use the FindAllMarkers for this.

```r
# Identify markers for each cluster
pdac1 <- FindAllMarkers(pdac1, only.pos = TRUE, min.pct = 0.25,
logfc.threshold = 0.25)

```

Step 15: Create a violin plot using one feature from each cluster.

```r
# Create a violin plot for one feature from each cluster
features <- c("CDK1", "G6PD", "RGS2", "DCN", "FGF13", "DCBLD2")
VlnPlot(pdac1, features = features, group.by = "ident")

```

Step 16: Create a feature plot using the same features as before.

```r
# Create a feature plot for the same features as before
FeaturePlot(pdac1, features = features, pt.size = 0.5, reduction.use =
"pca", cols = c("grey", "red", "blue", "green", "orange", "purple"))

```