# Sequencing Data Alignment and Quality Control

Tam To

2023-11-13

## Contents

## Getting Started

Here, I'm going over how to connect to Hoffman2 (UCLA's shared Linux cluster used as a way to store computational resources). Since we have a lot single cell data coming in, we need a place to store these files.

After downloading, uploading, and storing, we need to do sequence alignment and quality control of our data as well. These steps are also explained in this project.

Once you get everything downloaded, you can just skip to alignment and quality control steps!

This requires you to have a Hoffman2 account so make sure it's set up beforehand.

### Logging in and getting started in Hoffman2

Let's say we sent samples off for sequencing and received the data back. I will go over a few methods on how you can download and upload to storage servers based on where your files are (or how big they are).

The data usually comes in the form of a shared Google Drive folder or as a file from Illumina's Basespace Platform.

Download the data (if it's small enough in GB) or have it shared to your Google Drive.

First, go to your terminal and log into Hoffman2 using your username and password:

```
$ ssh login@hoffman2.idre.ucla.edu

$ login@hoffman2.idre.ucla.edu's password:
```

Request a node, runtime, and memory (example below):

```
$ qrsh -l h_rt=3:00:00, h_data = 4G
```

Make directory for bin (i.e. where to store rclone) and your data/project (example below):

```
$ mkdir bin

$ mkdir AcneData
```

Check that the folder was created:

```
$ ls
```

Now go to the other tabs and see the options for uploading the data.

Once you're done uploading the data, I'll next go over steps for sequence alignment and QC.

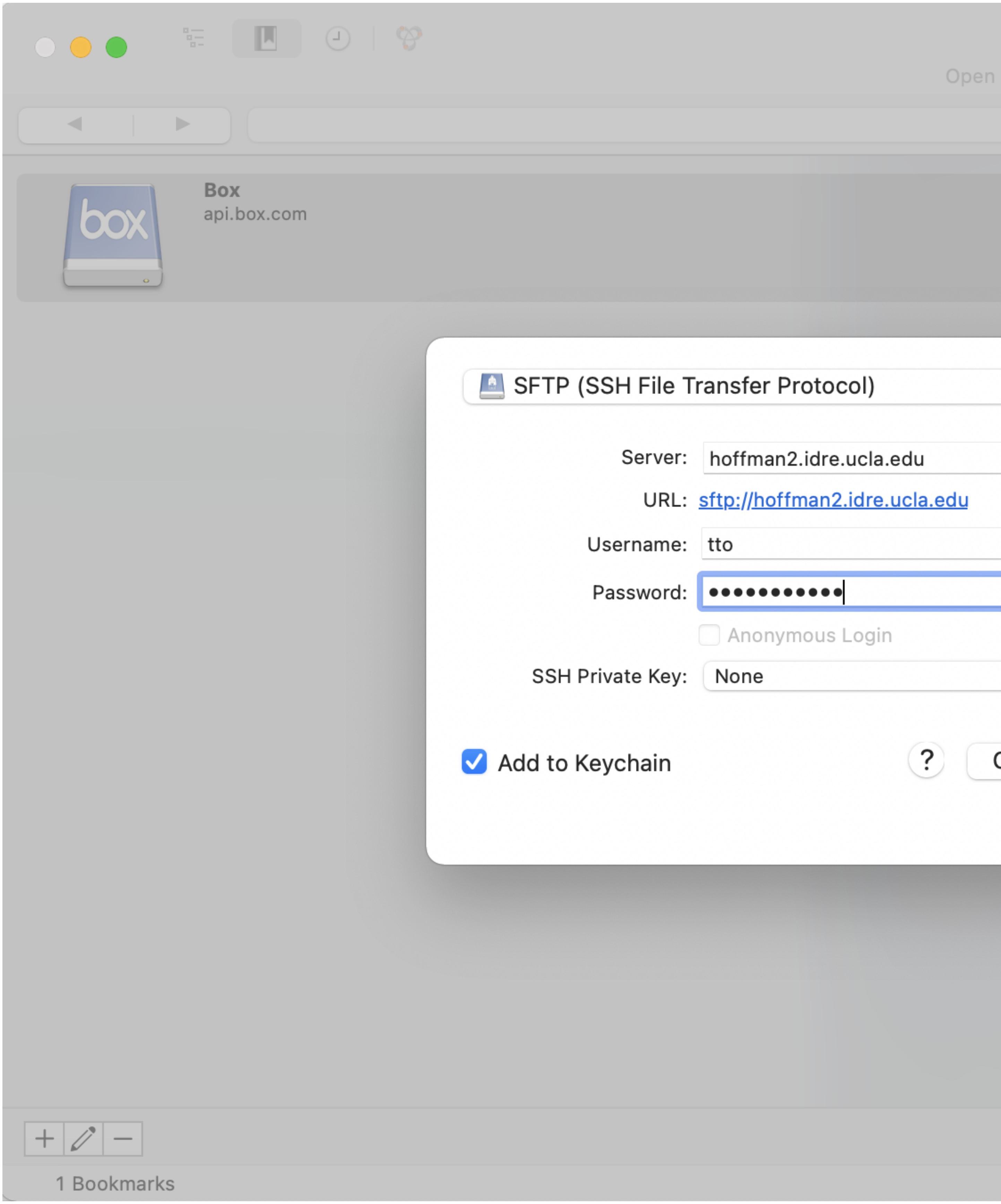
## Transfer with File Manager Applications

This is typically useful for smaller datasets that you can just download directly to your PC/laptop then upload to Hoffman2 with an easy drag-and-drop method.

For this method, you're going to need a file manager application. Here are a few popular ones:

| Application | Website |
|---|---|
| CyberDuck | cyberduck.io |
| FileZilla | https://filezilla-project.org/ |

I use CyberDuck, so once you open it then click on **Open Connection**. Log into Hoffman2 by selecting **SFTP File Transfer Protocol**.



SFTP (SSH File Transfer Protocol)

Server: hoffman2.idre.ucla.edu

URL: sftp://hoffman2.idre.ucla.edu

Username: tto

Password: ●●●●●●●●●●●

□ Anonymous Login

SSH Private Key: None

☑ Add to Keychain

After logging in, you should see the folders you created and can drag-and-drop it to the corresponding one. You can also upload items to Box this way.

### Transfer with Rclone

If you have larger data shared by Google drive, here I'm going over how you can download and upload it to Hoffman2. We made the "bin" directory earlier so this is where we are storing rclone. Within Hoffman2, do the following to download and copy it into the bin:

```
$ wget https://github.com/rclone/rclone/releases/download/v1.51.0/rclone-v1.51.0-linux-amd64.zip

$ unzip rclone-v1.51.0-linux-amd64.zip

$ cp rclone-v1.51.0-linux-amd64.zip/rclone $HOME/bin/.
```

Run the software with:

```
$ rclone
```

If you have trouble installing, check out more information on their site.

To configure rclone connection to Google Drive, first type:

```
$ rclone config
```

Then follow the steps on this site. At the step that asks **Use web browser to automatically authenticate rclone with remote?** » select **N**.

Follow the rest of the steps and allow rclone to have access to your Google Drive. Now you can copy the source (file in Google Drive) to the destination (Hoffman2 path).

**Make sure you have requested a node and rclone is running.** See an example below of what to type to upload the file:

```
$ rclone copy -P --drive-shared-with-me labdrive:10X_Acne_Data.tar /u/home/t/tto/AcneData
```

It might be a bit confusing the first time just downloading rclone, but once it's all done you don't have to do it again. You can also reference data transfer to Hoffman2 here.

## Preprocessing and Sequence Alignment

The core can either give you the raw base call (BCL) files or FASTQ files that have been demultiplexed already. Multiplexing involves starting with multiple samples (with sample-specific adaptors) then generating the sequencing library together per lane. If given BCL files, you need to use Illumina's CASAVA or bclToFastq software package to convert to FASTQ files. You can read more about it here.

Typically the core just provides the FASTQ files though.

For 10X genomics data, Cell Ranger is the analysis pipelines to align reads, generate feature-barcode matrices, clustering, etc. You can read more about it here.

Check out the next tabs to download Cell Ranger and the reference sequence.

### Download Cell Ranger and Reference Genome

Download Cell Ranger as well as references (human, mouse, etc.) from 10X Genomics here and unzip the files (you can do it logged into Hoffman2 in the terminal or manually with CyberDuck).

**Make sure you requested a node as well in Hoffman2 with enough time and memory.**

### Cell Ranger

So now we have what we need… Cell Ranger, reference genome, and our sample file all in Hoffman2 (in this example we'll stick with AcneData). This is how we start with using Cell Ranger:

```
cellranger-7.2.0/cellranger/ count --id=AcneData_Sample
--transcriptome=/directory/of/our/refdata-gex-GRCh38-2020-A/
--fastqs=/directory/of/our/rawdata/files/AcneData
--sample=AcneData
```

- A few notes: – Make sure you put in the right version of Cell Ranger – The sample name is the ID portion of the FASTQ file

Now you let it run!

Something useful is you'll end up getting an html (example here) that gives you a web summary of your estimated number of cells, mean reads, median genes, t-SNE projections, etc.

You'll also get a bunch of other 10X folders & files that include raw feature barcode matrix, analysis, molecule info, filtered features, bam files, etc.

Now we can move forward with analyzing the scRNAseq data in R…

## Quality Control

You want to work with the 'raw_feature_bc_matrix.h5' file. We're going to load it into a Seurat object and do QC metrics.

You can ultimately identify your cell clusters and do downstream analyses. I continue with this in the other projects outlined on my site!

Keep in mind as well, you'll most likely have data coming in at different times (with different samples in each). You can merge these datasets with 'merge()'. More information here.

### Seurat Object

Below is example code for how to convert the file into a Seurat Object:

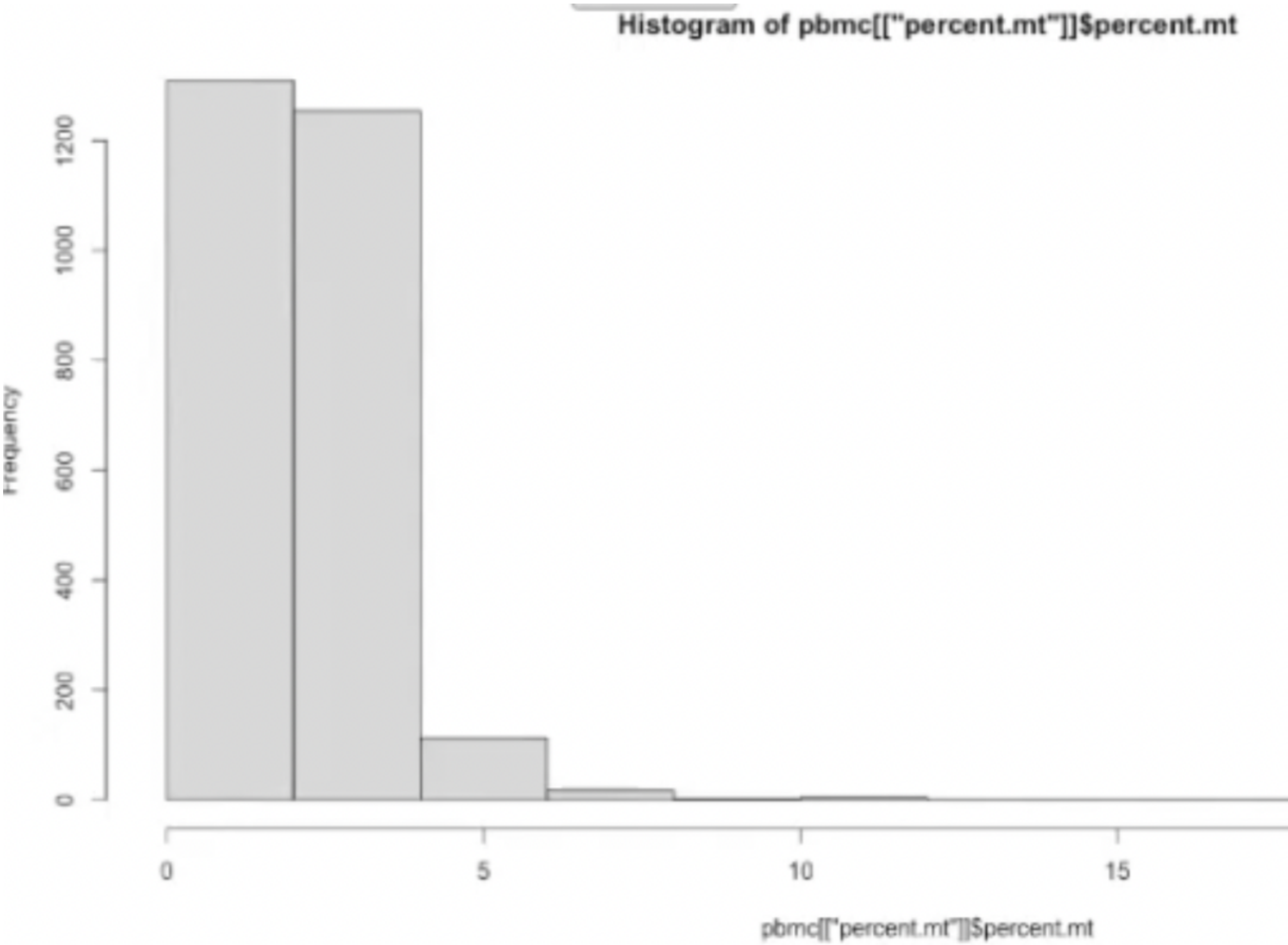
```
library(Seurat)
library(tidyverse)

AcneData <- Read10X_h5(filename = '.../directory/for/file/AcneData_raw_feature_bc_matrix.h5')

AcneData <- CreateSeuratObject(counts = AcneData, project = "AcneProject", min.cells = 3, min.features =
```

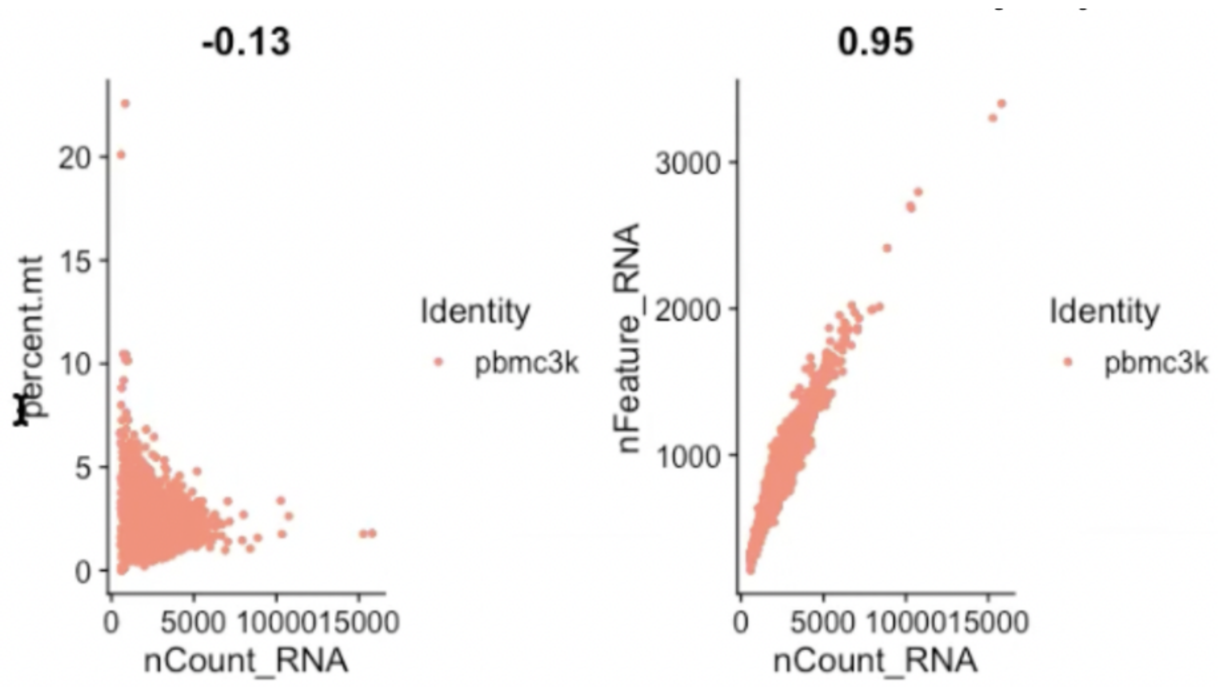
The argument **min.cells** = indicates that we want to keep all the features that are expressed in at least 3 cells, and **min.features** = means that we want to keep the cells that have at least 200 features (i.e. genes).

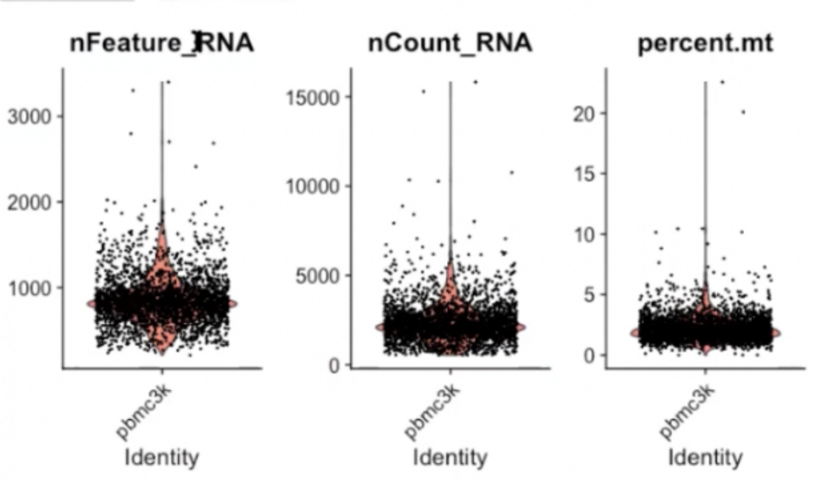
## Mitchondrial RNA Filtering

Mitochondria in cells have their own RNA! This is something we want to be cautious about when we are only wanting to know the genes that come from the nucleus. We have to filter out the mitochondrial RNA content. Example code and graphs shown below.

```
AcneData[["percent.mt"]] <- PercentageFeatureSet(AcneData, pattern = "^MT-")

hist(AcneData[["percent.mt"]]$percent.mt)

FeatureScatter(AcneData, feature1 = "nCount_RNA", feature2 = "percent.mt")
FeatureScatter(AcneData, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```







You can filter as well depending on how the data looks (example below).

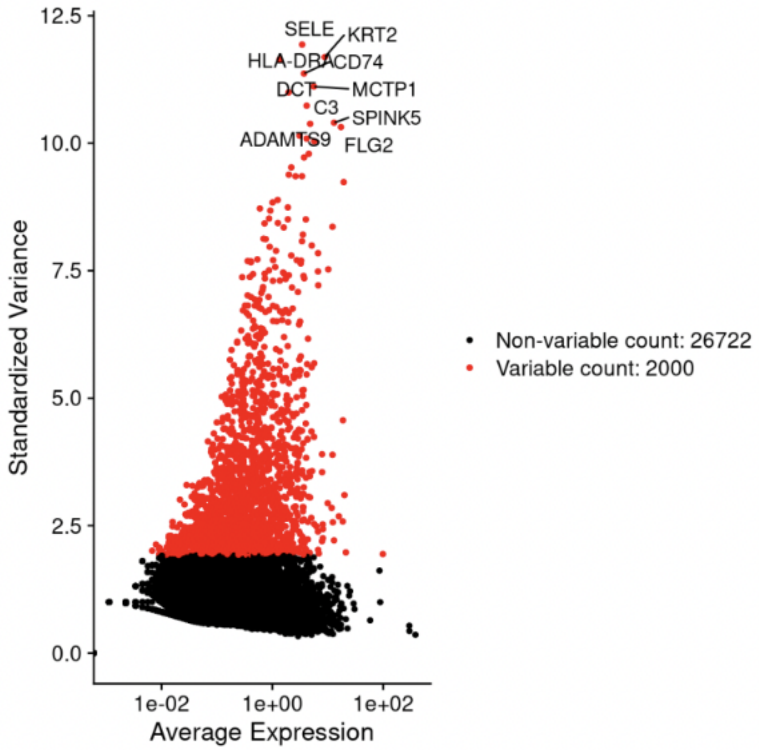
```
AcneData <- subset(AcneData, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
```

## Data Normalization and Highly Variable Genes

We have to also ensure that all cells have the same number of mRNAs to make the data comparable across the cells. We can also identify highly variable features (HVG) to see which genes are highly expressed in some cells compared to others.

```
AcneData <- NormalizeData(AcneData)
AcneData <- FindVariableFeatures(AcneData, selection.method = 'vst', nfeatures = 2000)

# top 10 most variable genes
```

```
top10 <- head(VariableFeatures(AcneData), 10)

# show it as a plot

plot1 <- VariableFeaturePlot(AcneData)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)

plot2
```



## Data Scaling and Dimensionality Reduction

Data Scaling makes values more comparable to each other and normalizes the genes by the total number. This is necessary before dimensionality reduction techniques like PCA.

Essentially this will shift the expression of each gene so the mean expression across cells is 0 and then scales the expression of each gene so the variance across cells is 1.

```
genes <- rownames(AcneData)

AcneData <- ScaleData(AcneData, features = genes)
```

Dimensionality reduction is explained more in my next project... feel free to continue there and explore more!