Characterizing Heterogeneous Breast Cancer Cells using scRNA-seq and Seurat

"Seurat is an R package designed for QC, analysis, and exploration of single-cell RNA-seq data. Seurat aims to enable users to identify and interpret sources of heterogeneity from single-cell transcriptomic measurements, and to integrate diverse types of single-cell data." - Seurat Website

Introduction

Single cell RNA sequencing (scRNA-seq) is a sequencing-based technology that uses Next Generation Sequencing (NGS) to sequence the transcriptomes of individual cells. To do this, cells are first isolated from a tissue of interest, are mechanically or enzymatically separated, they are placed in individual "compartments", and then the transcriptomes are amplified and converted to cDNA in the presence of unique cellular barcodes. The amplified and barcoded transcriptomes are then sequenced. Cells are then bioinformatically demutliplexed, and the reads are mapped and annotated producing a table or matrix of counts. This is the information normally available for download from databases (like GEO) where study datasets are stored and are publicly available.

One of the benefits of scRNA-seq (that it shares with its close companion technology, traditional RNA-seq) is that, in contrast to traditional low-throughput methods of looking at reduced representations of the transcriptome of individual cells in transcriptionally-related studies (such as qRT-PCR, microarrays and in situ hybridization microscopy, for instance), genes of interest do not need to be pre-chosen by the researchers in order to create either primers or probes for them. This is a function of its high-throughput nature. It is thus less hypothesis driven, and does not require as much knowledge in the form of expectations on the part of the researcher with respect to the kind of transcriptional events they will observe. In the case of RNA-seq and scRNA-seq, their high-throughput power allows the researcher to collect "all" of the information and "just look" at a (more or less) full representation of the transcriptome of a tissue or cell first. In contrast to more traditional forms of biological research, therefore, research efforts that employ these methods, instead of requiring more and more precise "hypotheses" with more or less (from a high-throughput standard) "limited data" to conduct experiments, it requires them to develop a different set of skills for "just looking" (at a higher resolution picture of the organism); i.e., bioinformatics skills..

Background

A transcriptome ultimately makes sense mechanistically, however, in terms of phenotype at the level of the individual cell. While RNA-seq technologies have improved our understanding of tissues, they leave out knowledge of the working unit that produced the transcriptome. One example of where tissue level knowledge using RNA-seq may have some limitations is in cancer biology. Tumors are highly heterogeneous, with cell "types" that have undergone changes in their genetic structure that make them distinct from the surrounding cell populations (and so, genetically and phenotypically, no longer resemble any normal cells of the organism). Cancer cells "evolve" such changes quickly, due to a loss in natural genetic programs that are designed to prevent uncontrolled cell growth in multicellular organisms, and stop the proliferation of such changes. And so they may even be distinct from one another (though they retain some information about their relationships, potentially, if we can access it). Their "evolutionarily states" may be as unique as to include only a single cell (it had to start somewhere, but it keeps "starting").

One important aspect of cancer biology, therefore, is to first identify and catalogue as many such changes as possible, and this is a prerequisite to exploring their associated mechanisms of evolution and growth. This is a problem that would appear to require (or at least benefit greatly from) a high-throughput solution, and one that can retain information about the individual cell that produced the transcriptome would necessarily improve this picture. To cancer biology, which might be particularly concerned with the state of a cells transcriptome (and particularly interested in a technology that could potentially reveal this in its entirety on an individual cellular basis), scRNA-seq appears to indeed be a revolutionary technology.

A generalized understanding of cancer cells can be extracted from the diversity of many unique cells that make up a tumor in terms of the mechanisms they use to live and grow, and this may be highly determined by how they evolve and are related (all which determine how we type them, and that determine what drugs/therapies they would be most susceptible to). Though cancer cells are still somewhat ancestrally related to their cancerous progenitors, they are in a constant state of evolution, leading to collections of diverse and "rogue" cell populations with a diverse spectrum of biologies (either localized to some tissue compartment/environment or, after metastasis, completely dispersed in the body and completely different developmentally from their surrounding tissue). This is often reflected in the cell's genome, but, on a more mechanistic level, it is more obvious in the cell's transcriptome. This spectrum of "evolutionary states" represent a complex, "messy" and highly "dynamic" obstacle to taxonomic analysis (with respect to normal, controlled developmental/evolutionary processes). In this respect, cancer cells seem uniquely amenable to bioniformatic analysis, and looking at these heterogeneous cells more closely using scRNA-seq and bioinformatics tools seems to be an important first step in improving our understanding of them.

I have chosen to perform computational analysis on a published dataset involving experiments related to a study of breast cancer in mouse models. The study associated with the dataset made use of scRNA-seq in an effort to better characterize breast cancer cells. Like many cancers, there are established ways of characterizing them. Some rely on a small set of marker genes that can be identified by PCR-related or histochemical methods, or that have been established through microarray technologies. Like many, the authors believed a more full transcriptional analysis, made using scRNA-seq, would be able to resolve more cell populations, and in this study, help to reveal their ancestral relation to potential yet unidentified progenitor cells, thus helping to improves cancer cell typing technologies (technologies used by physicians to determine prognosis and treatment for breast cancer patients.), on the basis of a more mechanistic and fundamental understanding of the cells and their developmental/evolutionary relationships.

To do this, the authors of this study used scRNA-seq to first characterize a complex sample from three different known breast cancer tumor types. Once this was done, they used the dataset from another study involving scRNA-seq to make inferences about the lineage relations of the sequenced cancer cells in theirs (lineage relations to a developmental hierarchy and potential unique progenitor cells). The other study sequenced mouse breast cells from different portions of the breast tissue at different time points in development. This was done to "... provide a global, unbiased view of adult mammary gland development" (Bach et al, 2017). Yeo et al.(2020) used this dataset, or "global view" of breast tissues and developmental states to "superimpose" their own dataset in order to explore the possibility of a relation of their cancer cells to unique developmental processes. This was done in an effort to identify possible lineage relations between their cancer cell samples and the hierarchy of developmental states of this tissue potentially generated through a kind of "hijacking" process, a well-known mechanism of cancer etiology (Yeo et al., 2020). The authors of this study used an R package called Seurat to do part of this analysis.

Satija et al. have created an R package called Seurat, designed to enable users to perform QC on scRNA-seq data, analyze it, and to integrate scRNA-seq data with other forms of information (including one another). One process (besides QC and normalization) that is common to most (if not all) scRNA-seq workflows is the clustering/grouping of cells into groups based on their expression profiles, and identifying/characterizing these with marker genes (and perhaps cell types). This will allow me to use Seurat and its tutorial to analyze the individual datasets, in addition to integrating and analyzing them together.

In the Yeo et al. study, the 10X Genomics workflow was followed to process the raw data following sequencing. Reads were aligned to a reference genome (mm10 from 10X Genomics) using STAR. 10X Genomics' Cell Ranger Single-Cell software suite was used to perform quality control and filtering, and clustering/differential expression analysis were performed using scran (and not Seurat, like it is here). Finally, Seurat was used to integrate processed scRNA-seq by "superimposition" of the dataset onto a normal mammary cell dataset. As the authors note "The integration of normal and tumor data was implemented... in order to identify common cell types and enable comparative analysis" (Yeo et al. 2020).

In this work, Seurat will be used, not just for integration, but its normal workflow (available as a tutorial on the Seurat website), in an effort to familiarize myself with it and a typical scRNA-seq workflow. Following this, the Bach et al. study will be analyzed as well, and the two datasets will then be integrated and analyzed together.

I Single Dataset Procedure

The workflow can be divided up into two major steps, analysis of each of the datasets individually, and data integration and analysis of the combined dataset. However, each of the individual workflows can also be broken up into two major steps: preprocessing and analysis. We begin with preprocessing and analysis of the cancer cell dataset.

Preprocessing involves selection of cells depending on QC requirements, normalization, scaling, and identifying highly variable genes (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html). Here, visualizations seem to function not so much to gain an understanding of the data, but rather confirm that it is normal. The basics of analysis involve linear dimensional reduction, determining the dimensionality of the dataset, clustering cells, non-linear dimensionality reduction (UMAP or tSNE), and, at some point, identifying differentially expressed genes (and sometimes assigning cell types to cell clusters) (ibid).Here, the visualization are strongly geared toward developing a deeper understanding of the data and drawing conclusions about them.

The Seurat website can be found here: https://satijalab.org/seurat/

Seurat has a hosted chatroom and repository: https://github.com/satijalab/seurat

The full documentation for Seurat can be found here: https://cran.r-project.org/web/packages/Seurat/Seurat.pdf

Full documentation for the Seurat Object can be found here: https://cran.r-project.org/web/packages/SeuratObject/SeuratObject.pdf

A "Seurat Cheat Sheet" is also available here: https://satijalab.org/seurat/articles/essential_commands.html

It might be worth noting here that, as the Seurat website indicates (on its data visualization vignette page), "with Seurat, all plotting functions return ggplot2-based plots by default, allowing one to easily capture and manipulate plots just like any other ggplot2-based plot" (https://satijalab.org/seurat/articles/visualization_vignette.html).

Seurat Objects can be saved and loaded using the following commands:

saveRDS(S_O, file = "path/to/file.rds")

S_O<-readRDS("path/to/file.rds")

Preprocessing

Loading Libraries and Data, and Creating the Seurat Object:

a. Set the working directory and load the required libraries:

```
setwd("~/Seurat_Project/Cancer_Files")
library(dplyr)
library(Seurat)
library(patchwork)
library(ggplot2)
library(clustree)
library(ggpubr)
```

b. Access the matrix, cells/barcodes and features files: navigate to NCBI's Gene Expression Omnibus (GEO) website, and search for accession number: GSE123366. Toward the bottom (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE123366) there will be a link for the "Supplementary Files".

The data comes in the form of three files: the matrix file (.mtx), the barcode file (.tsv) and the features file (.tsv) (The "matrix file" contains the transcript sequences and counts. The "barcodes file" contains the UMIs ligated to transcripts to barcode each cell (it is like an annotation file for the processing steps involved in barcoding individual cells, allowing the cells to be demultiplexed/resolved at this step). Lastly, the "features file" is a list of gene annotations (the mm10 mouse genome available from 10X Genomics here). Move the files to a folder you have set as the working directory in R.

As the GEO website states in its guidelines for submission of data (https://www.ncbi.nlm.nih.gov/geo/info/seq.html), GEO expects submission of both processed and raw data (raw data, or original sequencing FASTQ files, is uploaded to the Sequence Read Archive, or SRA, website separately). The processed data available at GEO is defined as "the data on which the conclusions in the related manuscript are based. We do not expect standard alignment files (e.g., BAM, SAM, BED) as processed data since conclusions are expected to be based on further-processed data".

c. Load the raw data into R: the data can be loaded into R from the folder you now have them in using the "ReadMtx()" function (passing "mtx = matrix file, cells = barcodes file, features = features file" as its arguments) and saving this to a variable name.

```
dirname <- "~/Seurat_Project/Cancer_Files"

expression_matrix <- ReadMtx(mtx = paste0(dirname,"/matrix.mtx"),
                             cells = paste0(dirname,"/barcodes.tsv"),
                             features =  paste0(dirname, "/features.tsv"))
```

As the Seurat website states, "the values in this matrix represent the number of molecules for each feature (i.e. gene; row) that are detected in each cell (column)" (https://satijalab.org/seurat/archive/v3.0/pbmc3k_tutorial.html)

d. Create the Seurat Object: to create a Seurat object (the data object that will be worked on throughout the course of the workflow) is created by passing the matrix you just created as an argument in the "CreateSeuratObject()" command.

```
S_O <- CreateSeuratObject(counts = expression_matrix)
```

The Seurat Object is passed as an argument in many commands related to preprocessing and data analysis. It also often "receives" the output of these commands. I have found this useful, as much (if not all) of the results of computationally intensive parts of the workflow then become "part of"/embedded in/slotted into the Seurat Object, which can then simply be saved (at any point) and called later, preventing the need to repeat calculations that may take considerable computational resources and time. As the Seurat website explains, the Seurat object is essentially "...a container that contains both data (like the count matrix) and analysis (like PCA, or clustering results) for a single-cell dataset". (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html).

In my experience, a single Seurat object in a working session of R can use close to 2 Gb of memory. Also, saved Seurat Objects appear to require on the order of about a Gb of disk space. So if you are working with multiple datasets at once, or saving multiple Seurat Objects to your hard drive, a considerable amount of your systems resources (RAM and disk space) may be used very quickly.

In the workflow for the first dataset, the Seurat object "S_O" will be used.

Visualizing QC Data and Filtering

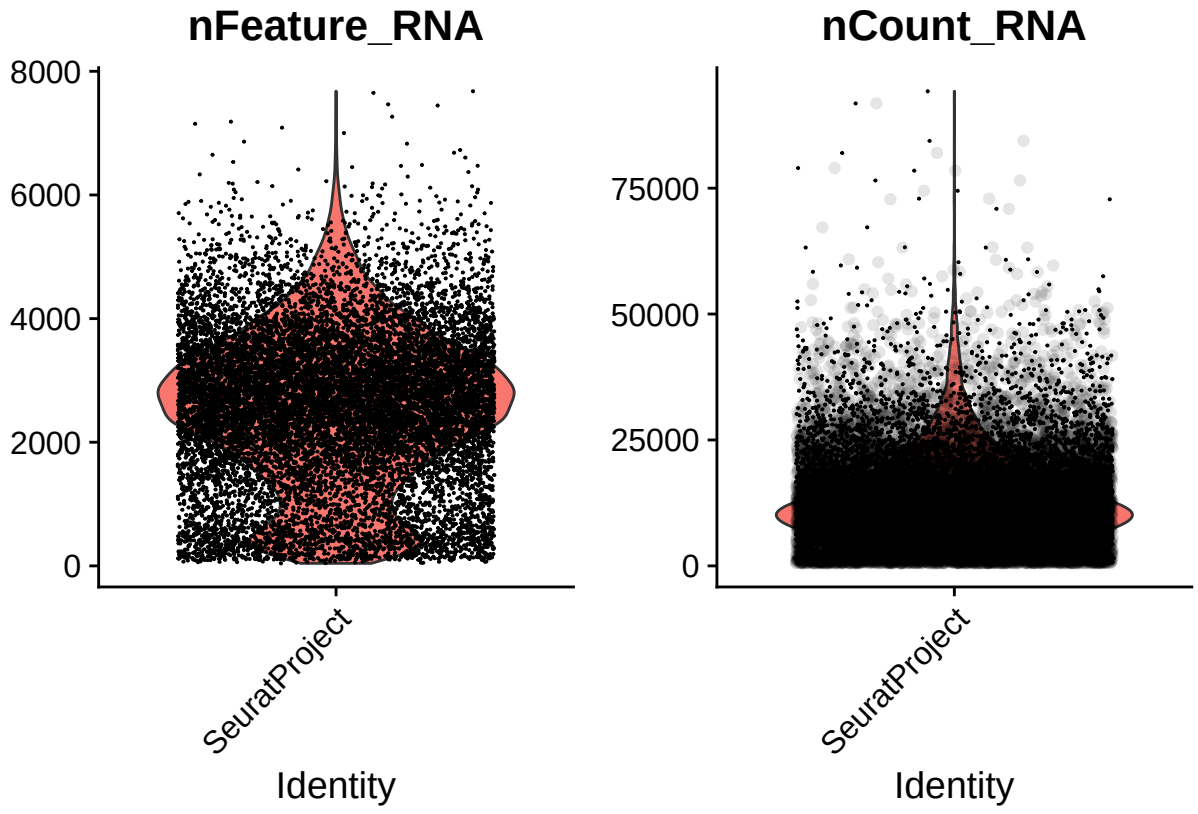
As mentioned, preprocessing visualizations seem to function not so much to gain an understanding of the data, but rather confirm that it is normal (that there are no glaring flaws with them. Essentially, at this point, we are just checking to make sure everything looks normal, and the visualizations help to do this quickly.

The first step is to set the percent mitochondrial DNA within the Seurat Object using the "PercetageFeaturesSet" function and the "[[]]" operator. This creates a metadata slot in the Seurat Object where the information is stored.

```
S_0[["percent.mt"]] <- PercentageFeatureSet(S_0, pattern = "^MT-")
```

Next, a violin plot can be used to look at the amount of RNA in each cell (and its distribution across cells), the number of genes per cell (and their distributions). It can also be used to look at the amount of mtDNA per cell (sometimes an indication of cellular damage), and their distributions.

```
VlnPlot(S_0, features = c("nFeature_RNA", "nCount_RNA"), ncol =
          2)+  geom_jitter(alpha = 0.1)
```

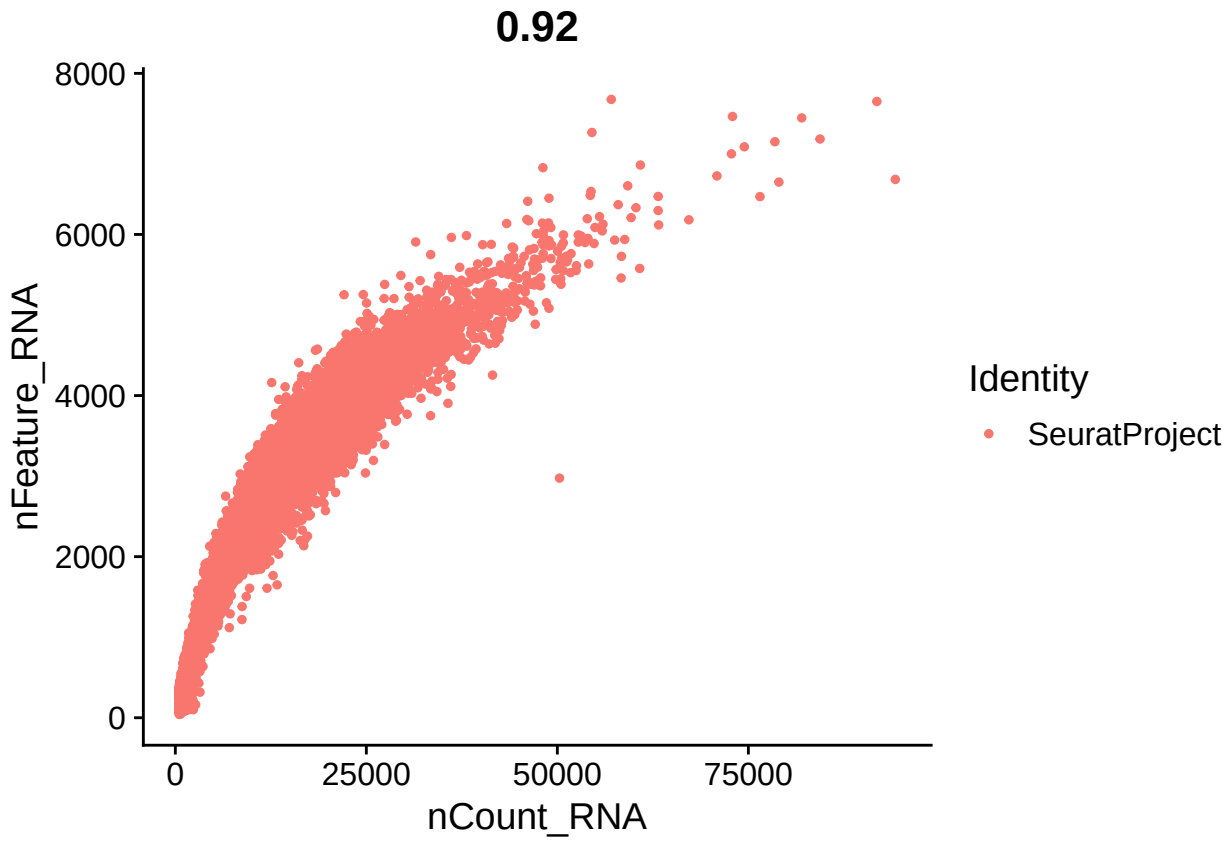


A violin plot is like a boxplot, but it gives more information than a simple boxplot with regard to distribution of the data (https://mode.com/blog/violin-plot-examples/#:~:text=A%20violin%20plot%20is%20a,the%20density%20of%20
When plotted, the violin plot of mitochondrial DNA content per cell appears empty. Sometimes mitochondrial genes are not included in the counts matrix, and so will not appear when this function is called (https://www.youtube.com/watch?v=xLTcexh1lmM, 7:18).To make sure this was the case, the expression_matrix was inspected. This was done by exporting the expression matrix as a text file, and exploring it using the grep function in bash (instead of R, because this made searching for terms associated

with “mitochondria” and “mt” in a case insensitive manner easier). A number of strings with this element were found (like “Pcmtd1”, “Mterf4”, “Trmt1l”, etc.). There are 37 genes in the mouse mitochondrial genome. A gene list of the Ensembl IDs for these was downloaded from Biomart and compared to the list of gene names from the expression matrix, after converting the Biomart list from Ensembl IDs to mm10 mouse Ensembl symbols (using a conversion tool at https://www.biotools.fr/mouse/ensembl_symbol_converter). All the mtDNA genes from the mm10 list start with “mt-” (for example: mt-Tf, mt-Rnr1, mt-Tv... mt-Tp) which did not resemble any of the “mt” strings in the genes found in the datasets. So mitochondrial DNA was missing from these two datasets.

A “feature-feature” plot can also be used, just to get an idea of the relationships between major raw data dimensions, here UMIs per cell and the number of genes detected per cell. These should be strongly positively correlated.

```
plot1 <- FeatureScatter(S_0, feature1 = "nCount_RNA", feature2 =
            "nFeature_RNA")
plot1
```



The data can be filtered using the command “subset”. Data to be filtered include reaction “compartments” that did not receive any cells (0 counts) and “cells” appearing to have abnormally high counts (or reaction compartments receiving more than one cell, or “doublets”. (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html). In the tutorial, 200 and 2500 were chosen as the features cutoffs for these parameters. Cells with abnormally high levels of mitochondrial DNA (>5%), a sign of cellular damage, are also commonly filtered (in this case, the dataset appears to have already been filtered for for cells with high levels of mtDNA content).

```
S_0 <- subset(S_0, subset = nFeature_RNA > 200 & nFeature_RNA < 2500
    & percent.mt < 5)
```

The gene expression for individual cells can be normalized to the total expression for each cell (essentially setting total expression equal for all cells, and making individual gene expression a fraction of a given individual cell's total). Seurat's website notes that a "log normalization" may be used (the fraction that gene expression is now represented by is multiplied by 10,000, by default, and the logarithm of that is taken). This is performed using the "NormalizeData" function in Seurat.

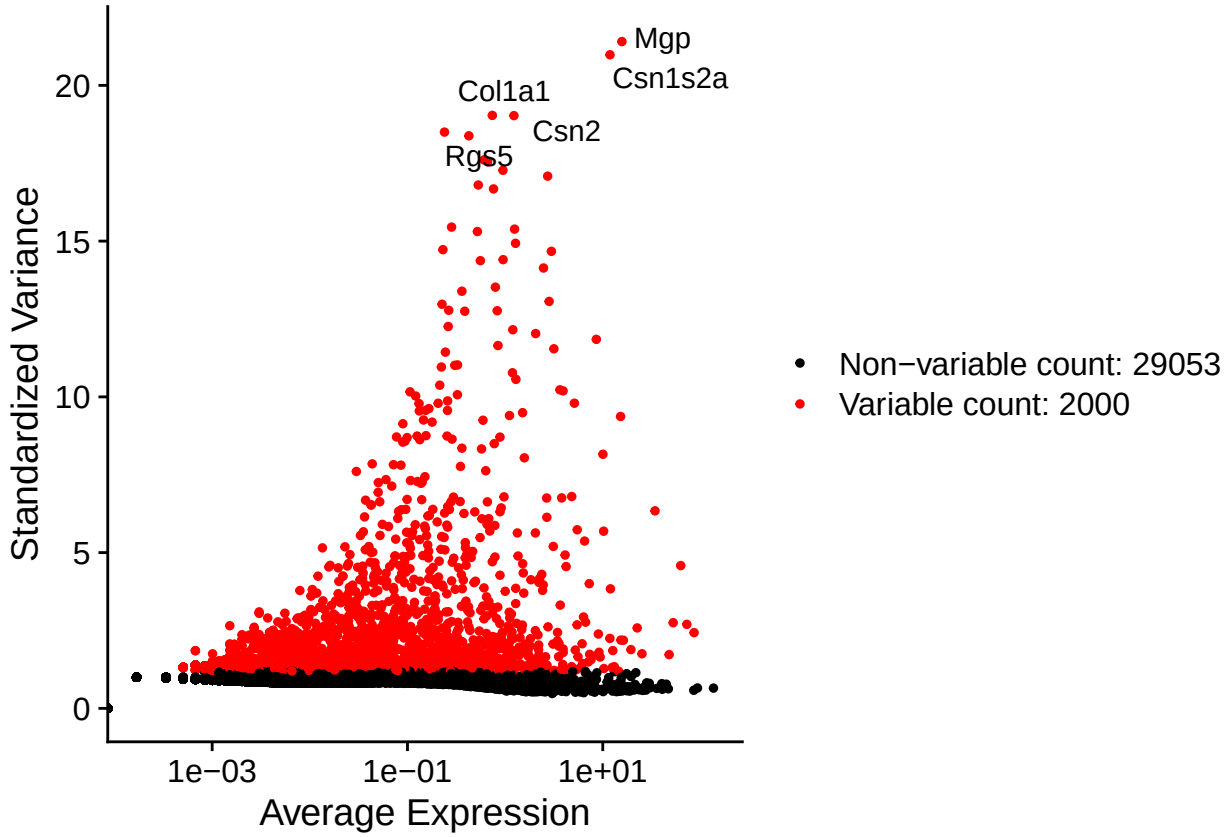
```
S_0 <- NormalizeData(S_0, normalization.method = "LogNormalize",
            scale.factor = 10000)
```

How do we characterize large, multidimensional data? In the case of single cells, the expression level of each gene represents a potentially different dimension characterizing a group of cells. To simplify this, one can consider that some genes are less informative than others, from the perspective of the uniqueness of cells (i.e., housekeeping genes, etc.). The uniqueness of cells in complex samples can be characterized more simply and conveniently, thus, by their more informative "variable features", or genes that are highly differentially expressed. The next step in preprocessing is identifying highly variable features. These will be the input that will be used for PCA analysis (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html) (given that we will be performing unsupervised clustering). This is done using the "FindVariableFeatures" function. 2000 features is the default for this function, and the number of features that will be used here.

```
S_0<- FindVariableFeatures(S_0, selection.method = "vst",
        nfeatures = 2000)
```

The "VariableFeaturePlot" function can be used to collectively visualize the variable features within the dataset. To assist in visualization, the "Labelpoints" function can be used to label the most variable genes, as well (though many features of Seurat seem to be important to anyone performing scRNA-seq, features like this seem to cater to experts. For instance, just looking at the raw data, labeled genes in a variablefeatures plot do not seem to suggest much. But if you are familiar with the kind of data you are looking at, this may be meaningful. And so like some things in Seurat, it seems, there is a mixture of both accessibility to the novice, and catering to the expert and, in the case of the novice, aspects of functions which may cater to experts have to be anticipated speculatively. In this case, this is uninformative, but it is included here with the understanding that it may one day be an important QC feature to someone, which seems to be suggested by the tutorial by including it.

```
plot1 <- VariableFeaturePlot(S_0)
top5 <- head(VariableFeatures(S_0), 5)
plot2 <- LabelPoints(plot = plot1, points = top5, repel = TRUE)
plot2
```



Scaling the data is the next step, and this involves performing a linear transformation of the data. According to the Seurat tutorial, this prevents highly expressed genes from affecting the data too greatly during downstream analysis (by making the mean expression between cells "0" and the variance between cells "1"). (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html). This is a common pre-processing step in scRNA-seq, and is mathematically necessary in order to run PCA dimensionality reduction (ibid). To do this, the function "ScaleData" is used and a vector containing character strings representing the features (genes) of the dataset is passed as an argument.

```
all.genes <- rownames(S_0)
        S_0 <- ScaleData(S_0, features = all.genes)
```

Analysis

Again, cells can be represented in a multidimensional space in terms of the expression levels for each gene. However, the use of such large multidimensionalty to characterize cells would be time consuming and inefficient. To increase the efficiency of characterizing complex samples of cells, and potentially grouping them by "type", just the most variable genes can be used to characterize the cells, as has been mentioned. However, this is still somewhat complex, and instead of using this, the dimensionality of the variable features can be reduced using a statistical technique known as principle component analysis (PCA) that groups variable features into principle components. This is a quick description of PCA:

"PCA geometrically projects a data set onto fewer dimensions, where the new variables are called principal components. This is done in such a way that the principal components are orthogonal and have the largest possible variances." (https://towardsdatascience.com/think-twice-before-you-use-principal-component-analysis-in-supervised-learning-tasks-70fbb68ebd0c)

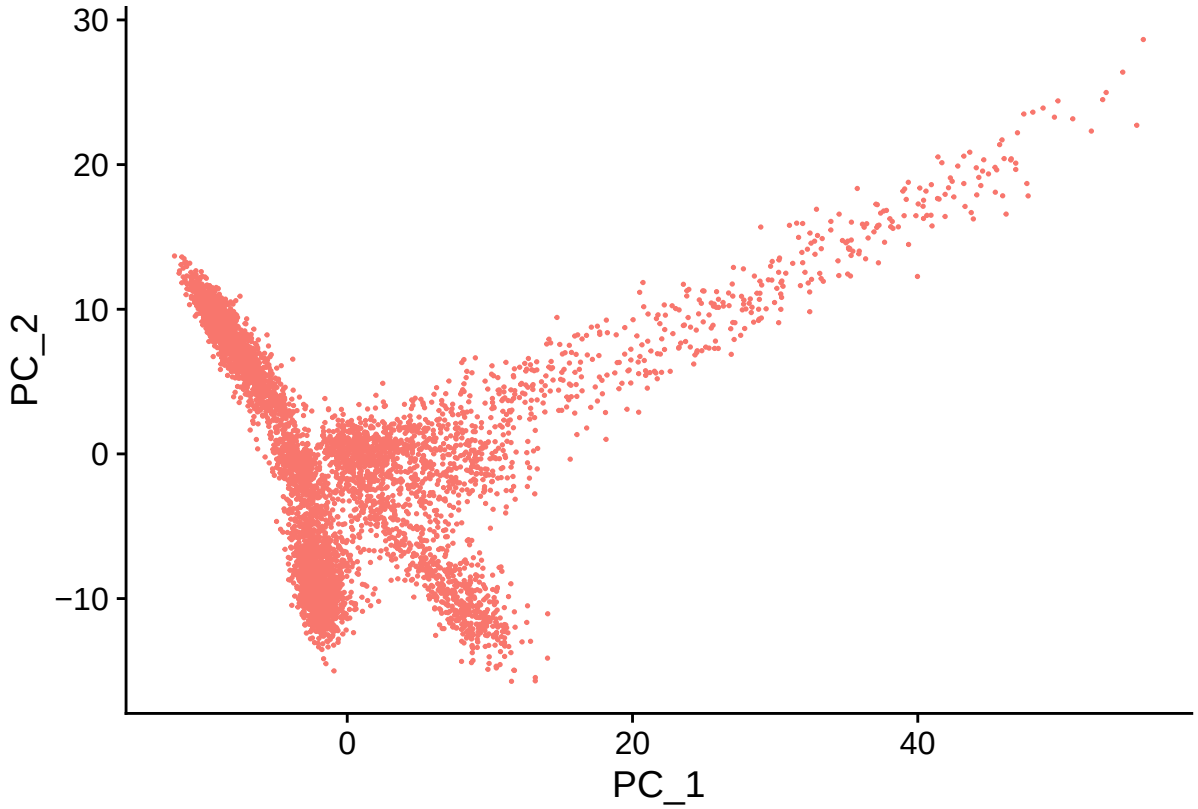
PCA is a linear dimensional reduction technique, and is performed in Seurat by executing the command

"RunPCA". As mentioned, the variable features that were detected by running "FindVariableFeatures" are used here (and are used by default, though the "features" attribute is shown here).

```
S_O <- RunPCA(S_O, features = VariableFeatures(object = S_O))
```

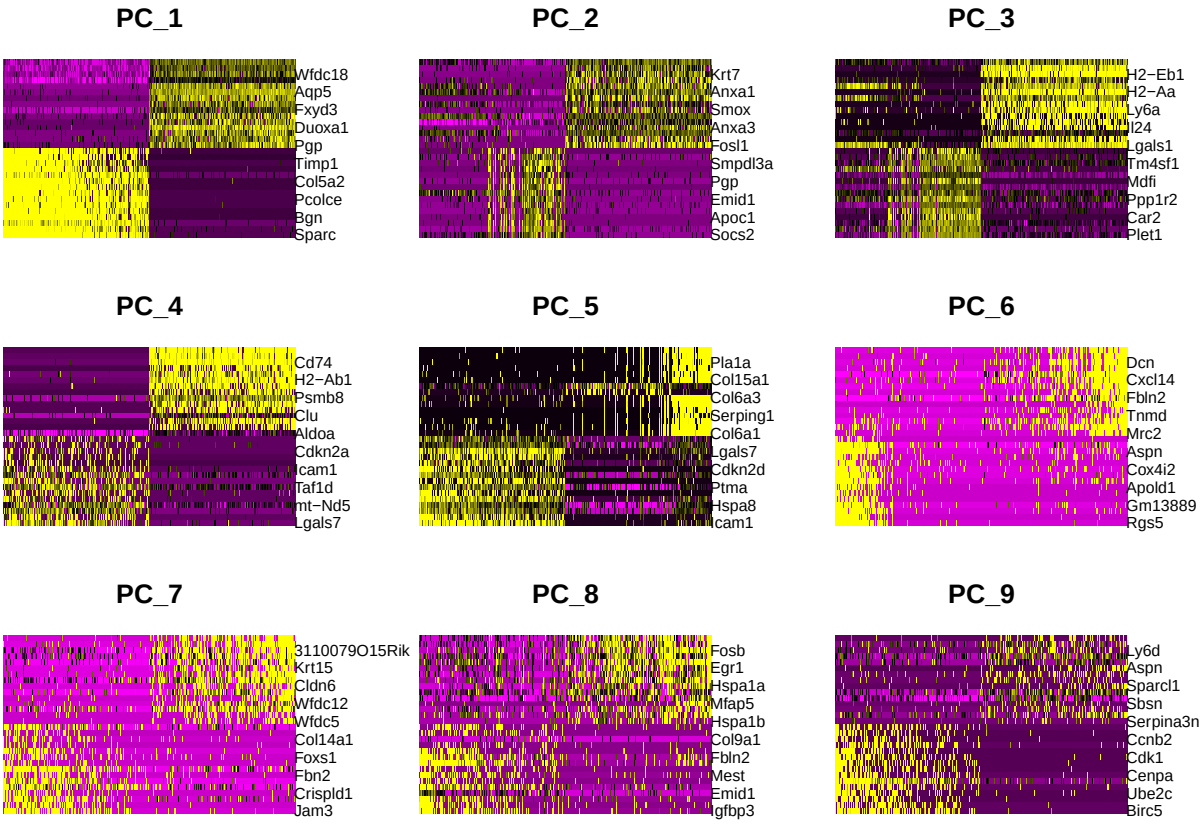
PCA results can then be visualized in a number of different ways. One way is to use the function "DimPlot" to look at cells spread by principle components (the first and second principle components are shown here).

```
DimPlot(S_O, reduction = "pca") + NoLegend()
```



Another way to visualize the PCA results is using a heat map. Here, the function "DimHeatMap" is used to look at the the top variable features being used for each principle component, and one or many heat maps can be plotted at a time.

```
DimHeatmap(S_O, dims = 1:9, cells = 500, balanced = TRUE)
```



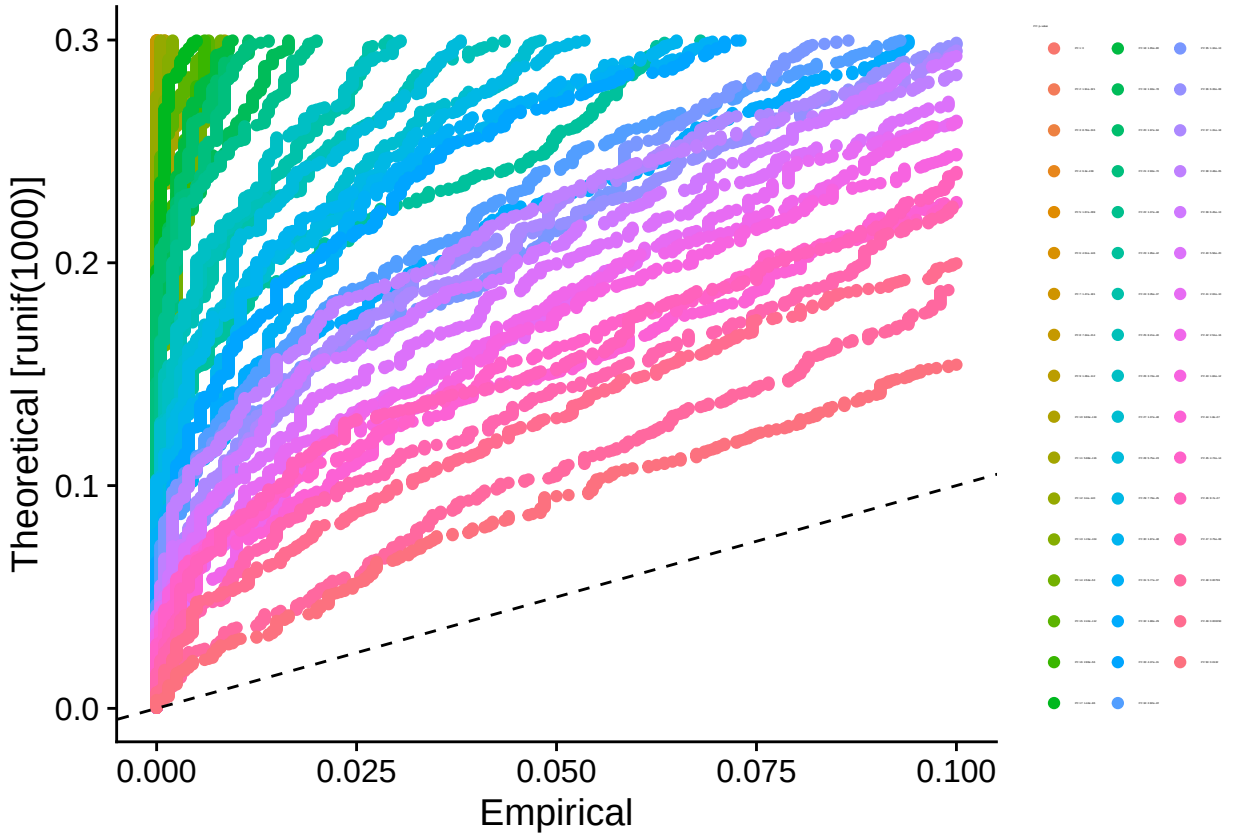
Next, the dimensionality of the dataset is determined. According to the tutorial, the "top principle components... represent a robust compression of the data set", the question is which ones/how many to use. Two methods are available here (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html). One method is based on a "Jack Straw" procedure, and another involves using elbow plots.

The Jack Straw procedure is a process for determining the statistical significance of principle components (https://rdrr.io/github/satijalab/seurat/man/JackStraw.html). The procedure involves randomly sampling a small group of genes from the principle components, and then assigning them PCA scores. (ibid) It then compares these random PCA scores to the originals, to generate p-values associated with them. (ibid) The p-value is then used to select the principle components that best represent the data. (ibid) A significant "drop off" in P values (at around 10^-12) was used in the tutorial.

```
S_0 <- JackStraw(S_0, num.replicate = 100, dims = 50)
S_0 <- ScoreJackStraw(S_0, dims = 1:50)
```

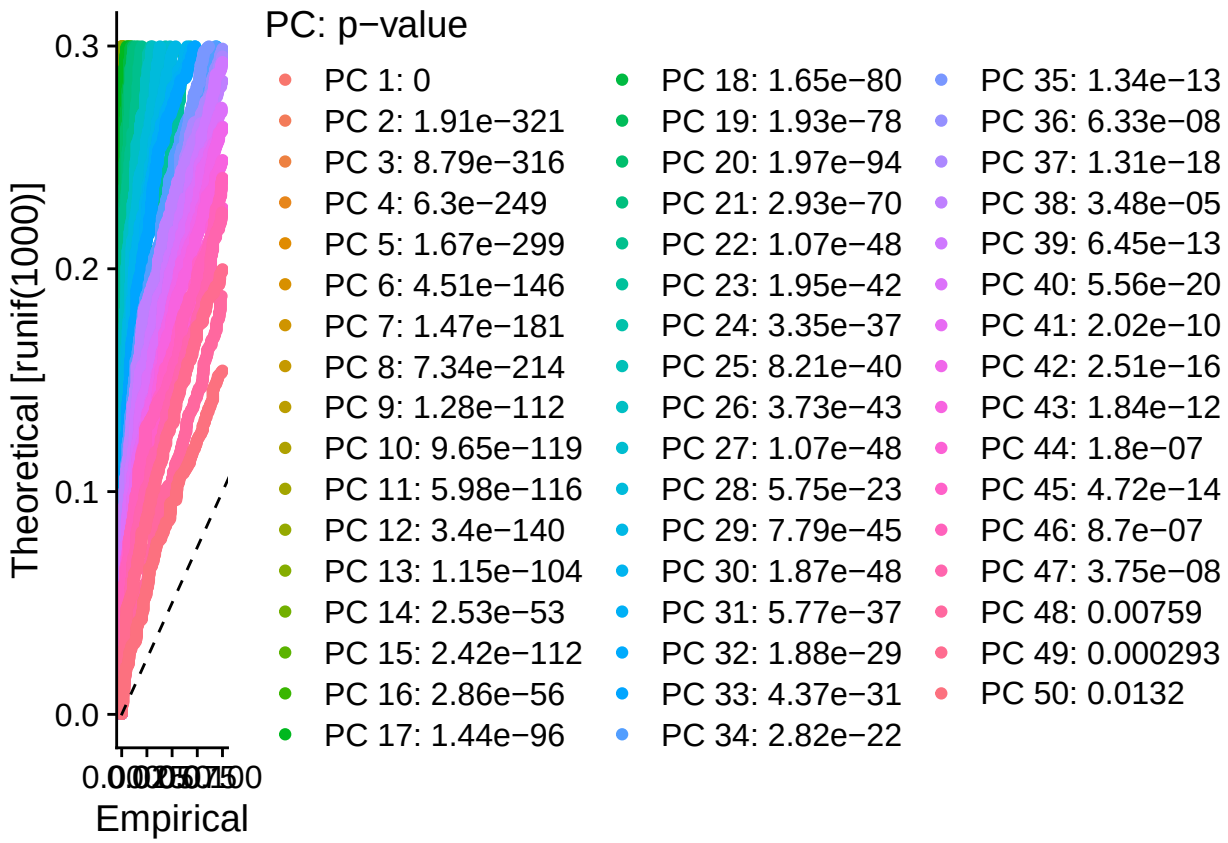
These results can be visualized using the "JackStrawPlot" function. This command provides a means of seeing the relation of the principle components to an uninformative baseline (y = x, dashed line) and for comparing p-values for each PC (P-values for each PC are listed in a legend on the right).(https://satijalab.org/seurat/articles/pbmc3k_tutorial.html)

```
JS1<-JackStrawPlot(S_0, dims = 1:50)
JS1<- JS1 + theme(legend.title = element_text(size = 0.5),
                  legend.text = element_text(size = 0.5))
JS1
```



The default number of dimensions searched for using Seurat is 20. This dataset was larger than the tutorial, and had significantly more significant PCs. The JackStraw plot is therefore more crowded, and, reducing the size of the legend elements was required to see the plot itself. However, it is impossible to make out the p values (ultimately, the more important piece of information here). So, another JackStraw plot was generated. In this one, the plot is not so intelligible, but one can see the p values more easily.

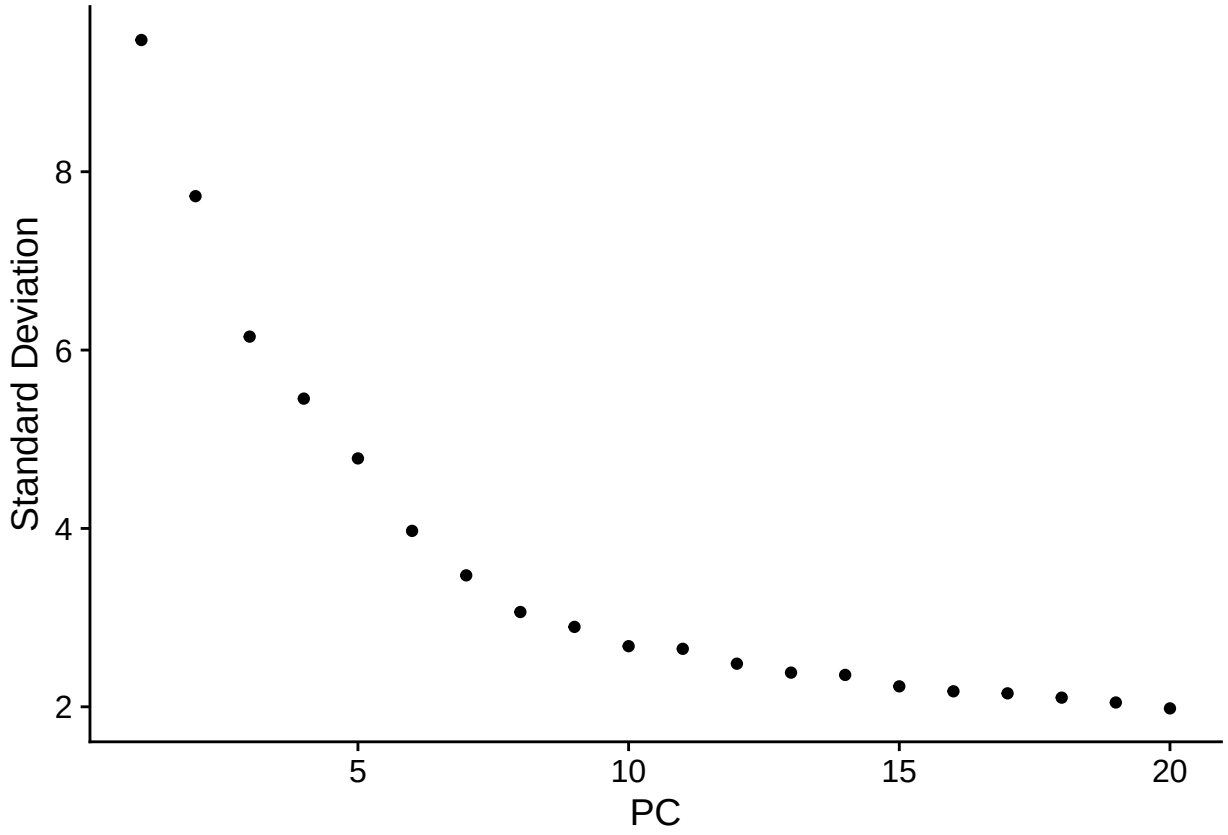
```
JS1<-JackStrawPlot(S_0, dims = 1:50)

JS1
```



From the JackStraw plot, 35 dimensions were chosen for downstraem analysis.

The tutorial also notes that there is a significantly less computationally intensive method for choosing the number of PCs, an elbow plot.

```
ElbowPlot(S_0)
```



Here, PCs are chosen by noting the place in the graph where an "elbow" appears (and excluding PCs in the dimensionality of the dataset after this point).Though this may be more computationally convenient, exactly where the "elbow" is may be slightly difficult to determine.

The Seurat website notes that, though downstream results depend strongly on having enough dimensions, they are not strongly affected by having "too many". They suggest, therefore, that if there is a question regarding how many dimensions to use, to err on the side of too many (and even suggest trying using more, to test whether this greatly affects downstream results, which they suggest it should not).

Seurat uses an unsupervised graph-based clustering to cluster cells (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html). Cells are clustered by first implementing the "FindNeighbors" function, and then the "FindClusters" function. This requires passing the number of dimensions that were determined to characterize the dataset (from the Jack Straw/Elbow plot procedure performed in the last step) as an argument in the "FindNeighbors" function. It also requires setting the resolution for clustering within the "FindClusters" command. The Seurat website states that they have found that a resolution of between 0.4-1.2 works well for datasets with around 3000 cells (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html). Lacking more information, at this point, we will be using the same resolution as was used in the tutorial.

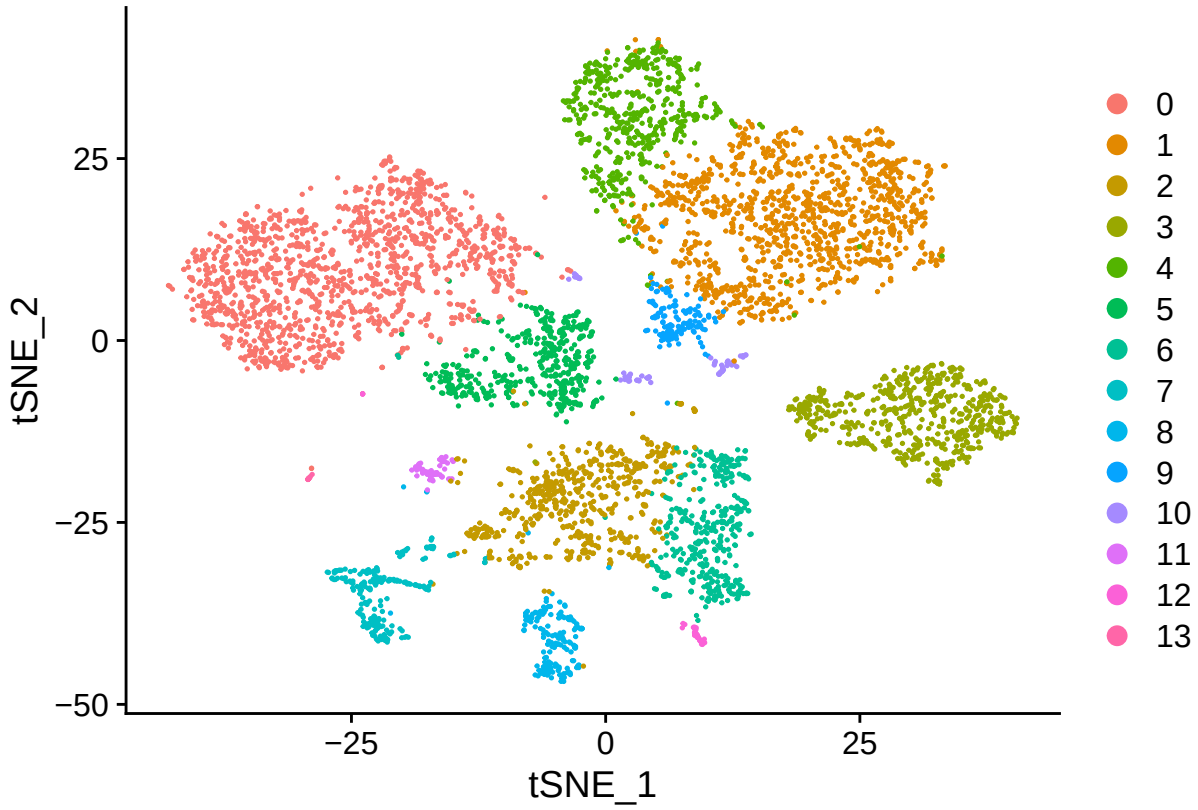
```
S_0 <- FindNeighbors(S_0, dims = 1:35)
S_0 <- FindClusters(S_0, resolution = 0.5)
```

We next use non-linear dimensional reduction to represent cell groups together in a low(2)-dimensional space (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html).To do this, the "RunTSNE" function can be used (the Seurat website suggests using the same number of dimensions used for clustering during this step).

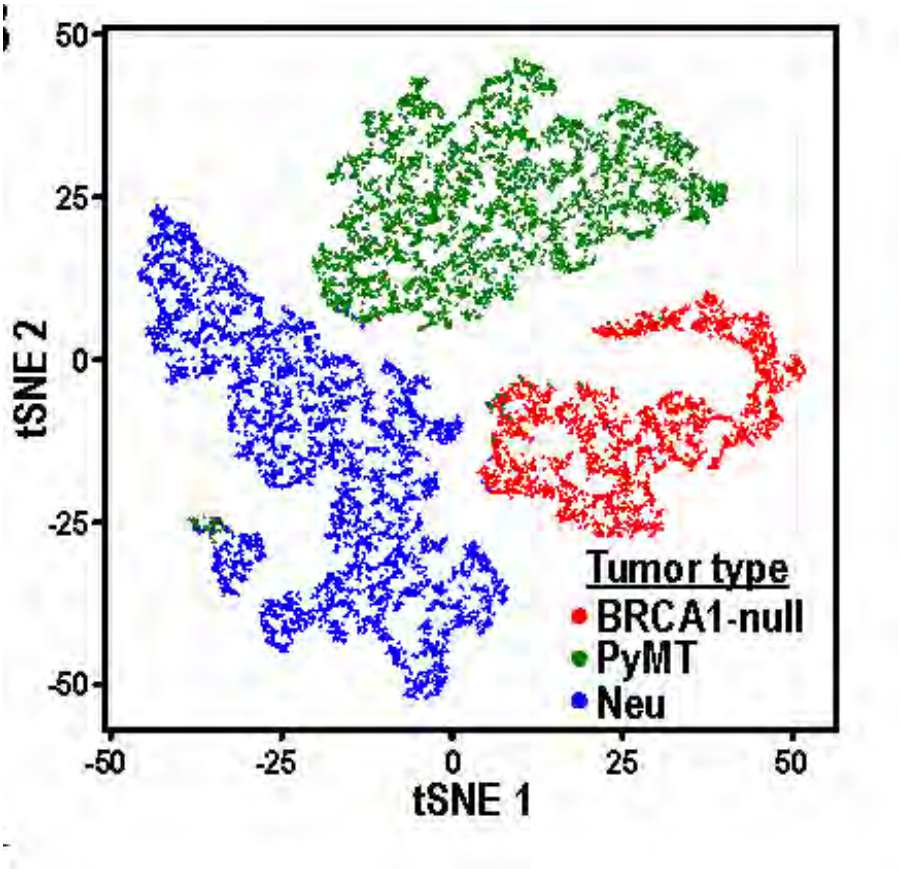
```
S_0 <- RunTSNE (S_0, dims = 1:35)
```

The results of "RunTSNE" can be visualized using the "DimPlot" function (introduced in the linear dimensional reduction section) with the "reduction" attribute changed to "tsne".

```
DimPlot(S_O, reduction = "tsne")
```



A tSNE plot of the cancer cells is among the first of Yeo et al.'s results (Figure 1B, Yeo et al., 2020). Their method yielded three clear clusters associated with the three different cancer types sampled.



The clear differences in the two plots suggested significant differences in the preprocessing and/or initial steps of analysis of the dataset.

Using Seurat the way it is more normally used, the cells in this analysis did not clearly differentiate into three easy-to-identify clusters (however, the coarser cluster groupings do appear to come close). Instead, what appears to be four large clusters (with numerous sub clusters) were generated.

Eventually, an inspection of the methods for this study led to the conclusion that, though the authors do not state so explicitly, they imply that the initial round of clustering that was performed here (Figure 1B, Yeo et al., 2020) was done using supervised clustering.

Seurat does not seem to have this functionality explicitly (in terms of intuitive commands), which may, or may not, be an indication of the prejudices or philosophical preferences of its developers. Instead, it uses an unsupervised graph-based clustering. It should be noted, however, that supervised clustering can be performed using Seurat, by employing the "pc.genes" attribute during PCA and/or the "genes.use" argument during non-linear dimensionality reduction (https://github.com/satijalab/seurat/issues/326). There is also a "RunSPCA" (supervised principle components analysis) command in Seurat (https://cran.r-project.org/web/packages/Seurat/Seurat.pdf). Running PCA or dimensionality reduction and clustering with a limited set of genes, however, seems to sometimes lead to too great reduction of the dimensionality that is needed for accurate clustering, and also seems to require rescaling of data in terms of those chosen genes)(https://github.com/satijalab/seurat/issues/127). Attempts to use Seurat to perform a supervised clustering with this dataset were highly problematic.

Before coming to the conclusion that the study authors used supervised clustering in their initial efforts to characterize cells of their dataset (because this was not obvious during the initial efforts to understand the discrepancy), several analysis factors were explored that may have led to this difference first. Following an internet search of the likely factors affecting clustering, resolution was identified as a major possible source of discrepancy between clustering plots using similar data.

With this in mind, and looking at the Yeo et al. tSNE plot, the lack of numerous sub clusters suggested

that they were in fact using a lower resolution (and that the dataset was amenable to this treatment). A few initial attempts at lowering the resolution (within the "FindClusters" function) was able to reduce the number of clusters to three, even two (at a resolutions around 0.01; remember that the Seurat tutorial suggests a resolution between 0.5 and 1.2 for datasets with around 3k cells. This can increase for larger datasets. The one that is being used has about 12,000 cells, so we'd expect to be able to increase the resolution, not need to decrease it). However, when lowered to points where only two or three clusters were apparent, these clusters still did not differentiate into three clear groups of nearly equivalent size visually.

Lowering clustering resolution really represents a loss of information, not an improvement in it, and would normally have very limited uses for this reason. As such, one is usually interested in increasing the resolution as much as possible, in order to differentiate cells as much as possible, not "undifferentiate" them. While clustering differentiates cells, it provides knowledge of their inherent "groupability" (potentially, inevitable "groupability"), but only to the point of "overclustering".

Lowering the resolution suggested itself because it seemed that Yeo et al. were more focused on the coarser grouping at this point (and performed multiple rounds of clustering), and less on differentiation. It should be noted that "simplifying" the data this way might be justified from the perspective of using a dataset involving heterogeneous cancer cells, with possible lineage relations to a "progenitor", to get "all the variations (or heterogeneity) - and ones with further possible relations to a developmental cell set or cell space hierarchy related to fundamental progenitors and mutations -"together in one spot" to be characterized together. It appears that supervised clustering provides a mechanism for doing this too, but this is still a kind of "blurring" of the available information. In the case of supervised clustering, this "blurring" (See "lukewarm heat map" below) can be described as an a priori bias based on an hypothesis (of the possible relatedness of mutant cells through an evolutionary process to a normal tissue developmental hierarchy that better represents this data (and assumes less differentiation of the cancer cells along their evolutionary paths. See below "In defense of Yeo et al.").

Unable to use the study as a guide (not in terms of the shapes of cluster plots, but in terms of the numbers of clusters) and as a result of exposure to the possible profound effects of altering clustering resolution, an effort was made to gain an understanding of the methods for determining optimal resolution for clustering (if that could somehow be determined to be "inevitable" for a dataset), since unsupervised clustering was going to be used and lowering resolution did not seem to be the issue.

One method that seemed common in determining optimal clustering is to cluster cells at various resolutions, and to then perform a cluster tree analysis.

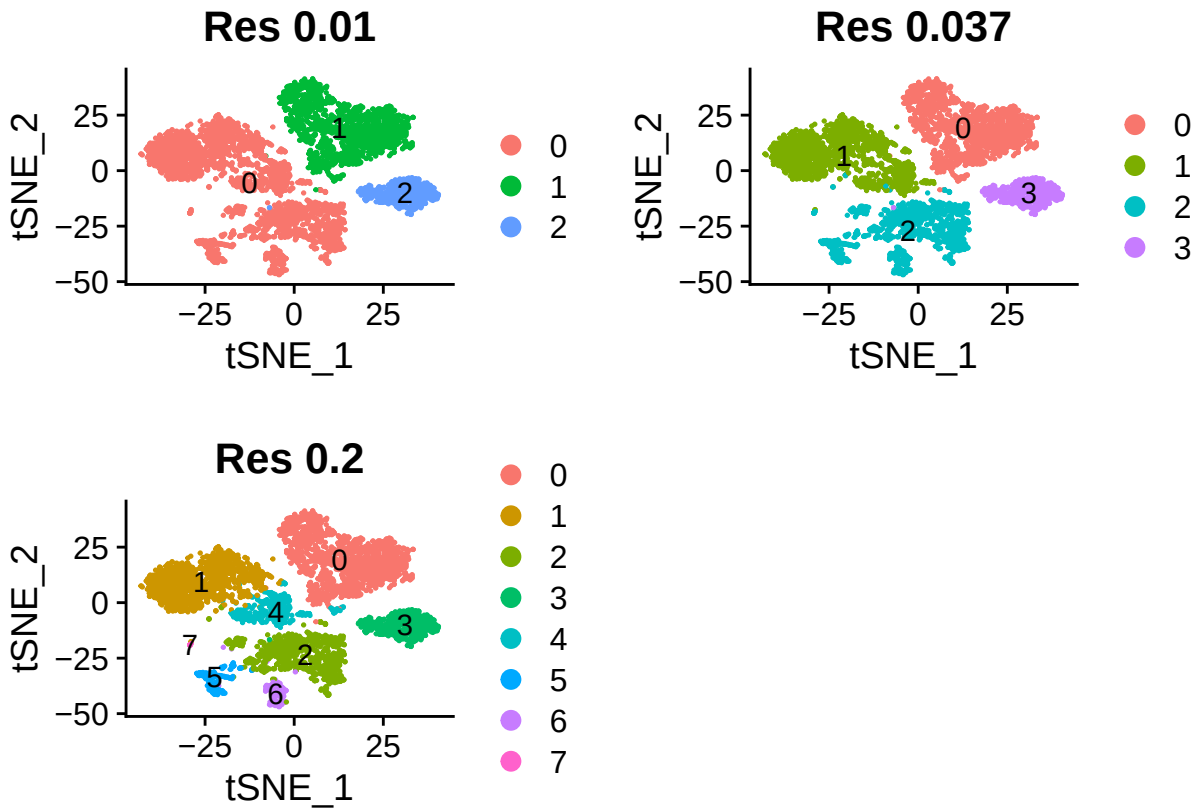
In order to generate a series of cluster sets at various resolutions, simply input a vector of resolutions into the attribute "resolution" within "FindClusters" function and rerun the command. The results will be embedded in the Seurat Object.

```
S_0 <- FindClusters(S_0, resolution = c(0.01, 0.037, 0.2, 0.5, 1,
                                        1.5, 2.5))
```

Next, rerun "RunTSNE" and visualize the results with the DimPLot function, indicating the desired resolution within the "group.by" attribute, and rename the plot titles using the ggplot2 "title" function (note: when using the "group.by" attribute, numbers with no decimal value will be truncated to whole numbers in the object despite how they have been stored).

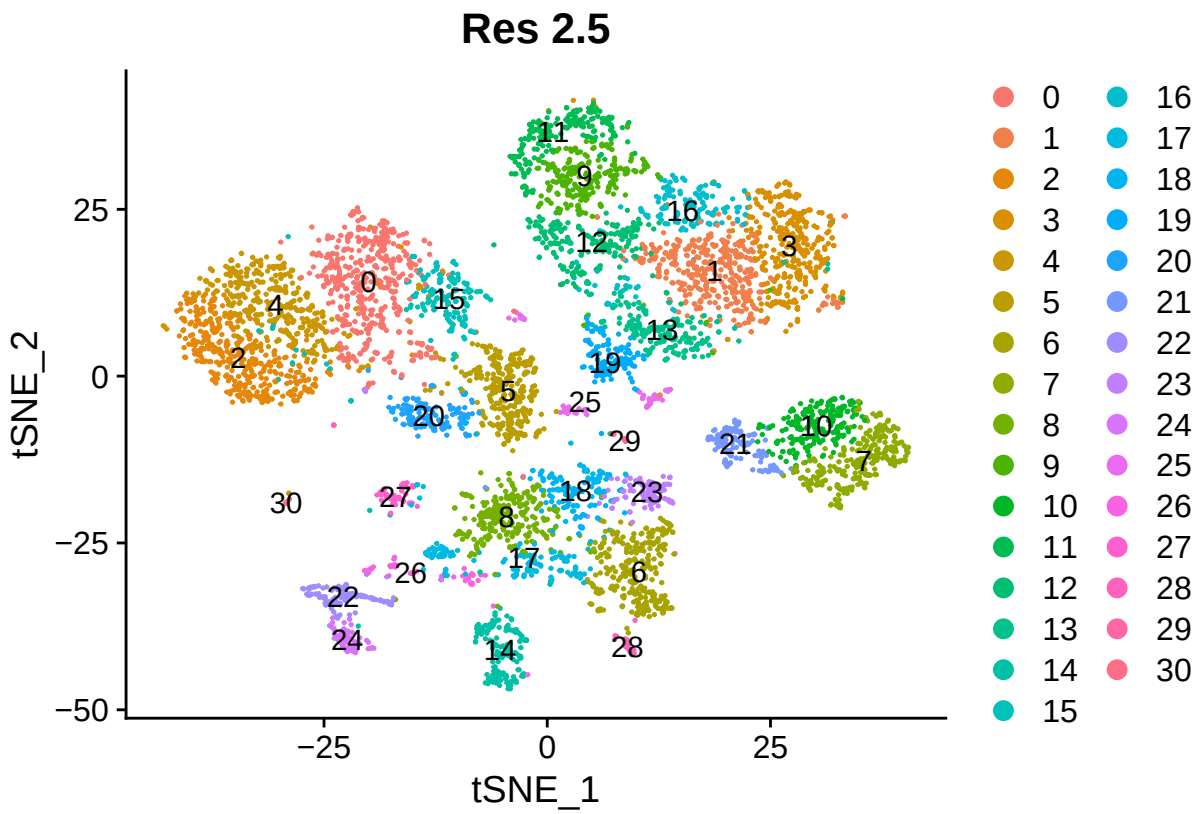
```
S_0 <- RunTSNE(S_0, dims = 1:35)

d0_01<-DimPlot(S_0, label= TRUE, reduction = "tsne", group.by =
                 "RNA_snn_res.0.01") + ggtitle("Res 0.01")
d0_037<-DimPlot(S_0, label= TRUE, reduction = "tsne", group.by =
                  "RNA_snn_res.0.037") + ggtitle("Res 0.037")
d0_2<-DimPlot(S_0, label= TRUE, reduction = "tsne", group.by =
                "RNA_snn_res.0.2") + ggtitle("Res 0.2")
```

```
d0_5<-DimPlot(S_0, label= TRUE, reduction = "tsne", group.by =
                "RNA_snn_res.0.5") + ggtitle("Res 0.5")
d1_0<-DimPlot(S_0, label= TRUE, reduction = "tsne", group.by =
                "RNA_snn_res.1") + ggtitle("Res 1.0")
d1_5<-DimPlot(S_0, label= TRUE, reduction = "tsne", group.by =
                "RNA_snn_res.1.5") + ggtitle("Res 1.5")
d2_5<-DimPlot(S_0, label= TRUE, reduction = "tsne", group.by =
                "RNA_snn_res.2.5") + ggtitle("Res 2.5")


ggarrange(d0_01, d0_037, d0_2,
          ncol = 2, nrow = 2)
```



As is clear form these plots, altering the resolution during clustering can lead to dramatically varied results. This is a dimplot of clustering of the same cells at a resolution of 2.5.

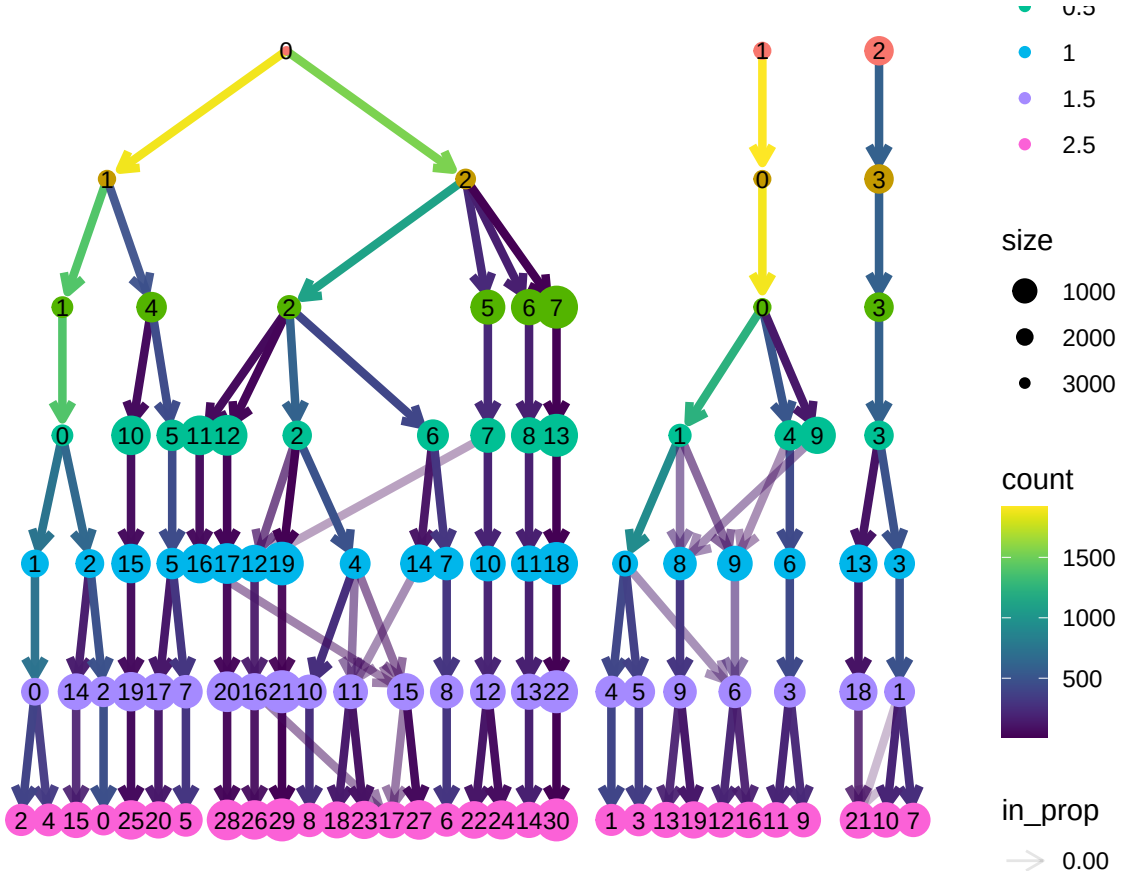
```
d2_5
```



As mentioned, the optimal resolution for a dataset can be determined by varying the resolution and inspecting a cluster tree graph of the results (https://github.com/satijalab/seurat/issues/1565). Other techniques for determining the optimum resolution/cluster number exist (https://gist.github.com/BenjaminDoran/a4c8939b2f180b5ddf7e9179697d07ba), but this one is common (https://cbiagii.github.io/post/post_01/). Yeo et al. do not mention doing this themselves in their article (the program they used, scran, would presumably have this functionality as well, and it is doubtful that supervised clustering of this kind, which might depend on less variable features (quality, not necessarily number), would involve lowering of the resolution anyway. In fact, it might depends on raising the resolution as much as possible.

As mentioned, a cluster tree can be examined to determine the optimal resolution for the dataset (https://lazappi.github.io/clustree/articles/clustree.html) by simply passing the Seurat Object as an argument in the "clustree" function available through the "clustree" package in CRAN (https://cran.r-project.org/web/packages/clustree/vignettes/clustree.html).

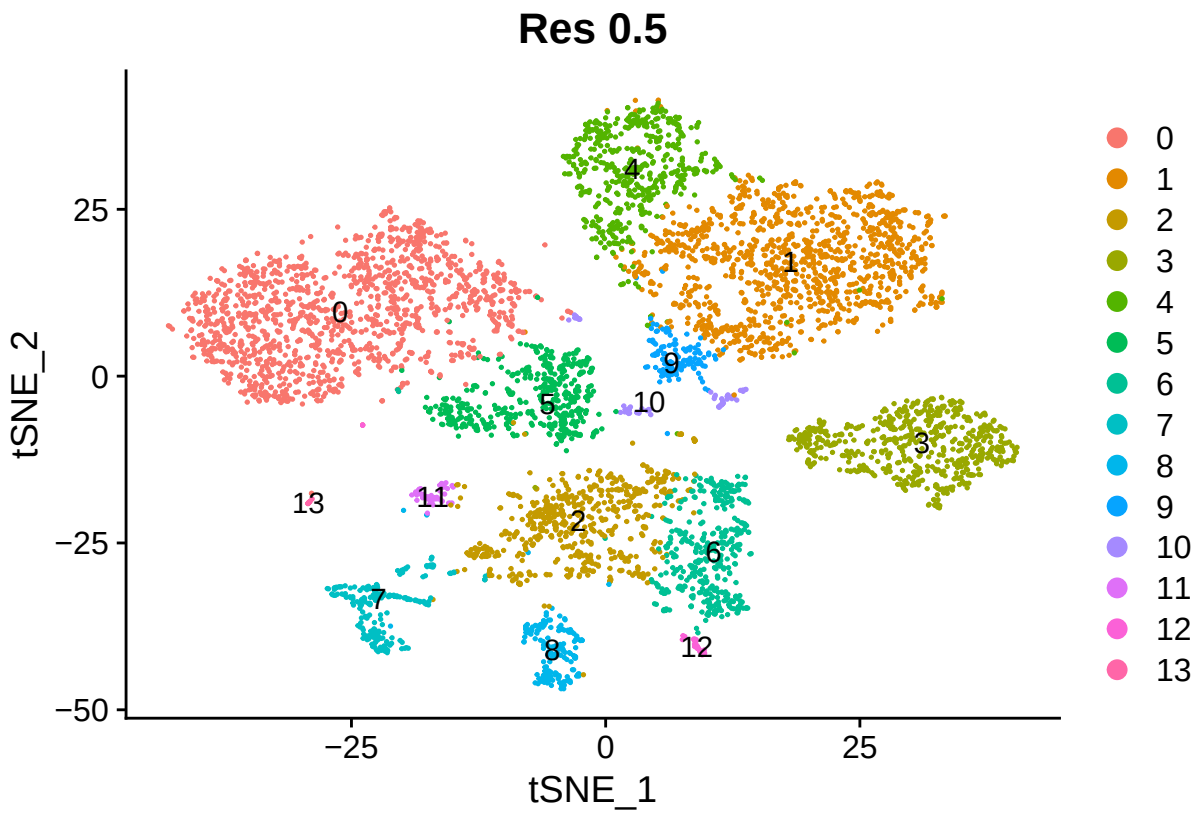
```
clustree(S_0, node_size_range = c(7, 1), edge_width = 1.5)
```



One can see that, at higher resolutions, sub clusters are diverging from a single parent node with a similar increase in convergence on child nodes. At lower resolutions, however, clusters are primarily diverging, without a lot of "cross-talk" and convergence, which we would expect from quality clustering (https://cran.r-project.org/web/packages/clustree/vignettes/clustree.html). As mentioned, this could actually indicate that originally sampling more cells might have led to the possibility of using a higher resolution leading to improved resolvability of this dataset (as the Seurat website notes "optimal resolution often increases for larger datasets", possibly an important factor when performing scRNA-seq; therefore, it might be useful to perform a "pilot study" of samples of various size for an experiment, to get an idea of what resolutions are possible and needed for a chosen sample space. Or, in my bioinformatically more informed mind, perform one big sampling, one that potentially comprehends the size of all necessary smaller ones, and simply randomly "sample" it computationally and test various resolutions for their cluster tree qualities).

From an inspection of the cluster tree, convergence of more than one edge on a single node originating (diverging) form clusters at a lower resolution appears to begin at a resolution of about 0.5 (this is also the resolution setting used in the tutorial).For these reasons, and wishing to use a conservative value, a resolution of 0.5 was chosen for downstream analysis.

```
d0_5
```



At a resolution of 0.5 (and using 35 PCs) the dataset is clustered into 14 clusters. Therfore, it is possible that the cells in this dataset do not differentiate into three clear and unique groups without using a supervised method of clustering.

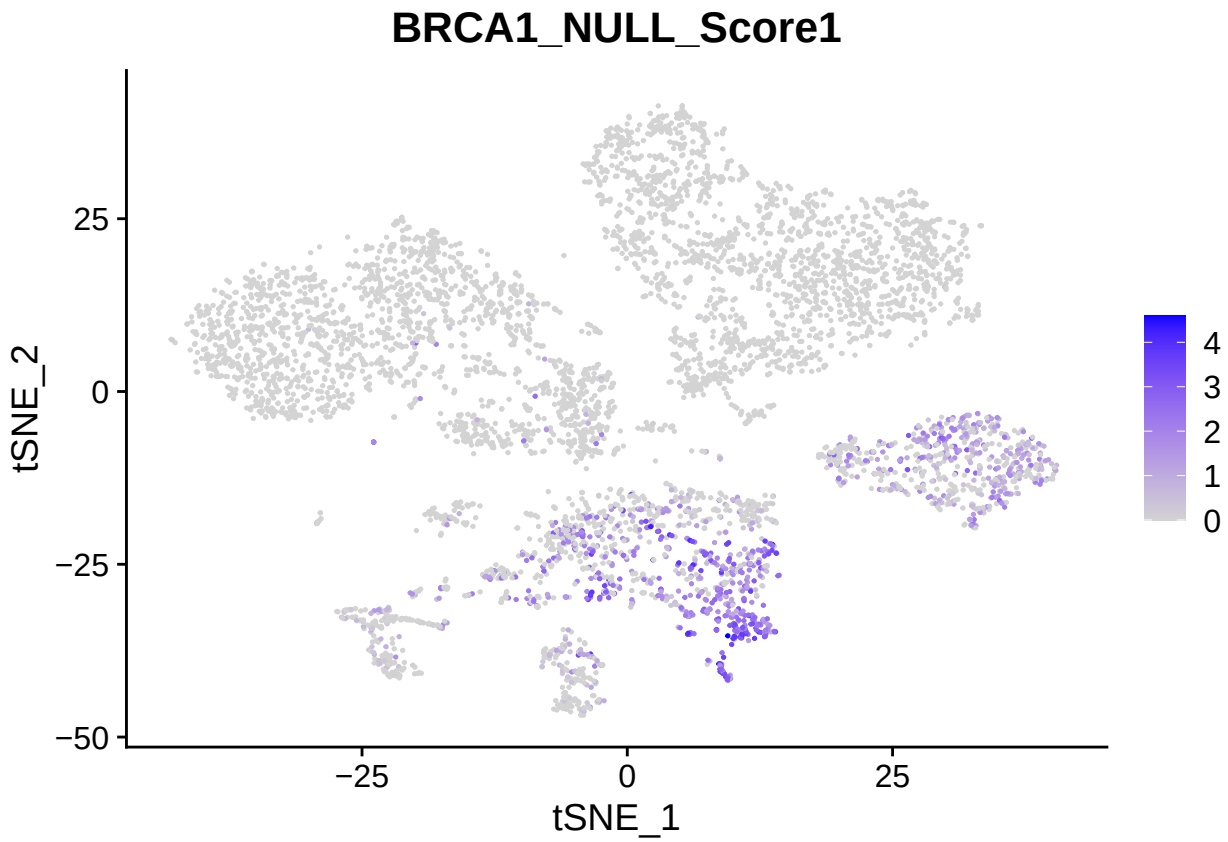
Yeo et al. make note throughout the section of the paper that deals with clustering that a handful of genes can be used to differentiate their three clusters into the three main cancer cell types. These genes were next used in an attempt to gain some understanding of the distribution of "cell types" as they related specifically to the distribution of cells in the unsupervised clustering generated using Seurat.

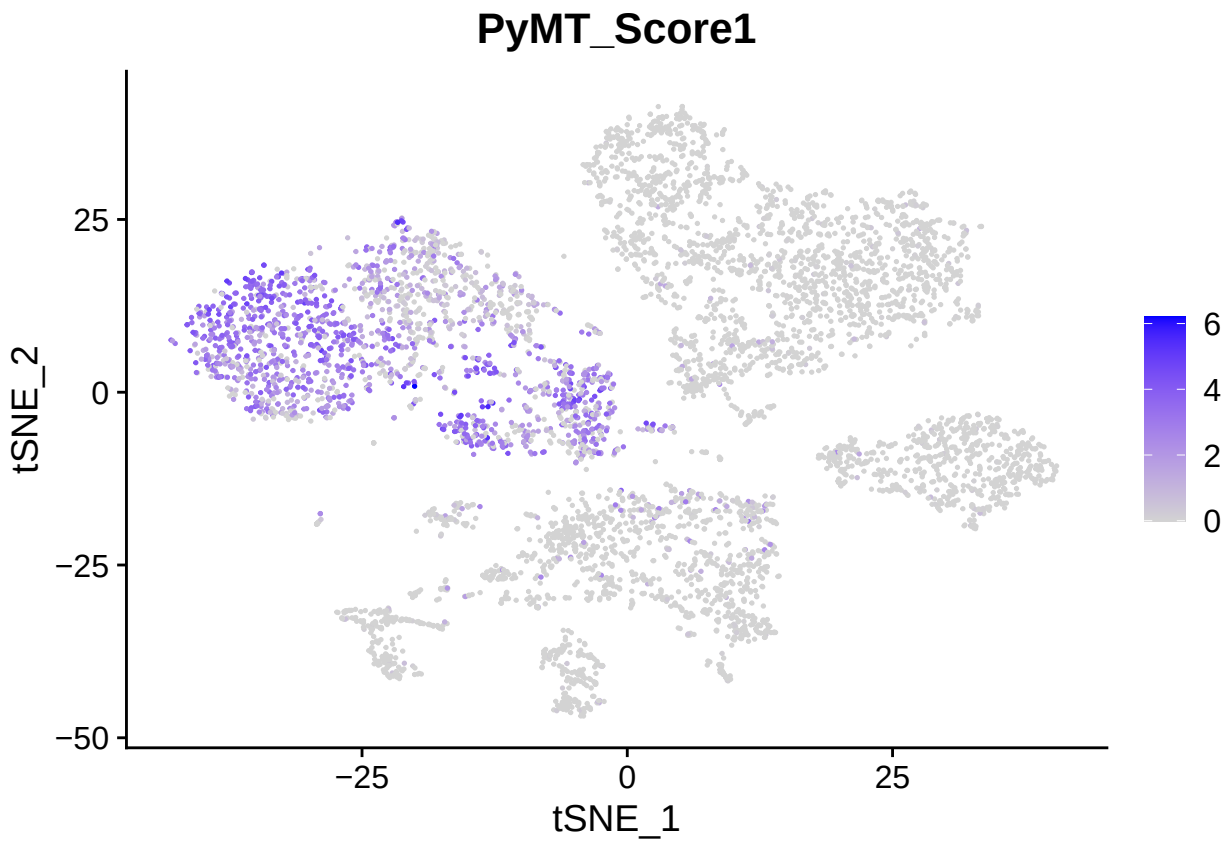
According to Yeo et al., BRCA1-Null type cells can be distinguished by the presence of higher levels of Krt14, Vim and Sparc (Yeo et al. 2020). PyMT type cells can be distinguished by the presence of Ltf, Spp1, Anxa, Cldn4, and Aldh1a3 (ibid). Finally, Neu type cells can be identified by the presence of Csn1s1, Lalba and Tspan8. (ibid).

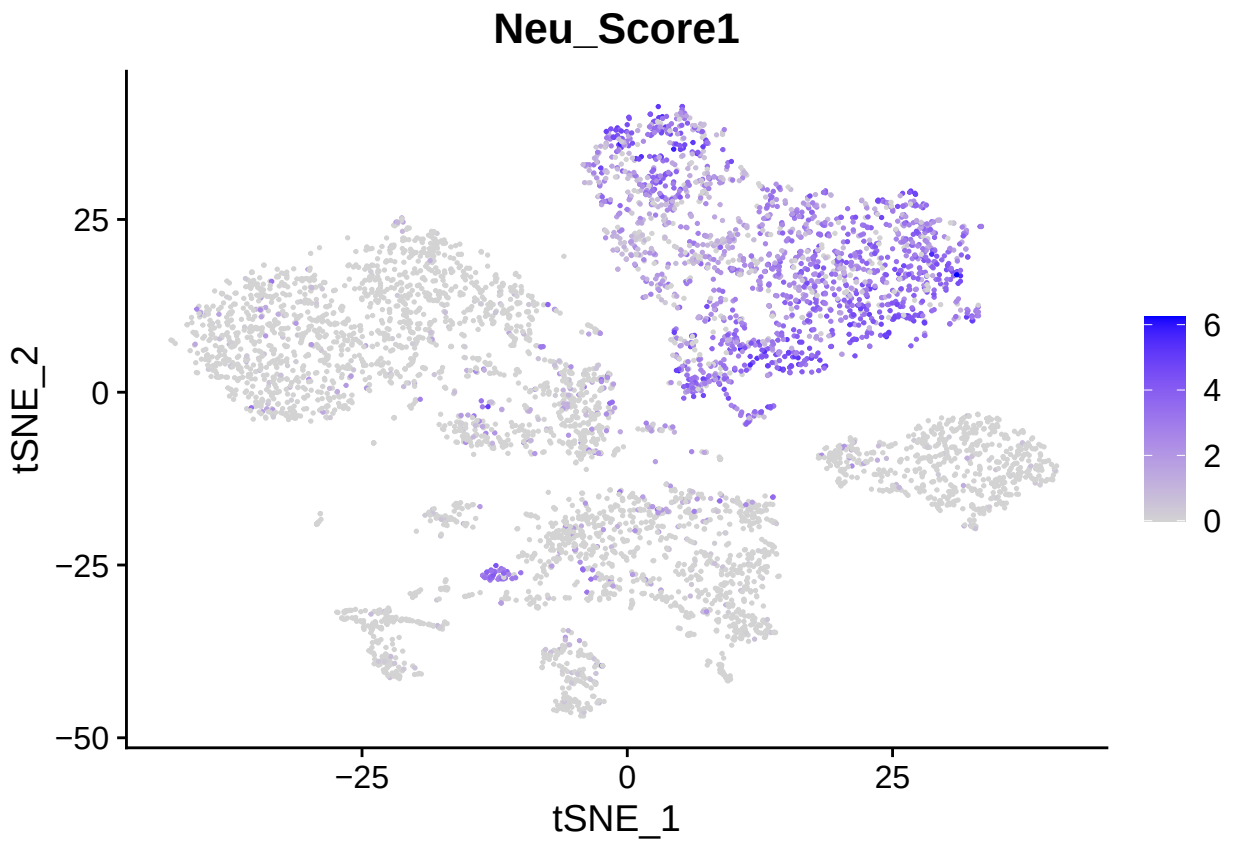
In the absence of a completely different clustering processes, the "AddModuleScore" function was used to generate a score for cells based on the three small groups of genes mentioned in the study (as mostly mutually exclusive to the three "cell type" groups). Inputting these into the AddModuleScore function made it possible to visualize this with the "FeaturesPlot" function (much like one might enter individual marker genes to mark features associated with clusters individually and manually). This marked cells that scored high with respect to the small groups of genes across clusters (regardless of cluster). Using this method, three clear groups did become readily apparent within the cluster plot (the "min.cutoff=" attribute in FeaturePLot was set to "0" to improve the contrast).

```
BRCA1_NULL <-c("Krt14", "Vim", "Sparc")
PyMT <-c("Ltf","Spp1","Aldh1a3")
Neu <-c("Csn1s1", "Lalba", "Tspan8")

S_0 <- AddModuleScore(object = S_0, features = BRCA1_NULL, name = "BRCA1_NULL_Score")
```

```
S_O <- AddModuleScore(object = S_O, features = PyMT, name = "PyMT_Score")
S_O <- AddModuleScore(object = S_O, features = Neu, name = "Neu_Score")

FPZA<-FeaturePlot(object = S_O, features = c("BRCA1_NULL_Score1", "PyMT_Score1", "Neu_Score1"), min.cut

FPZA
```

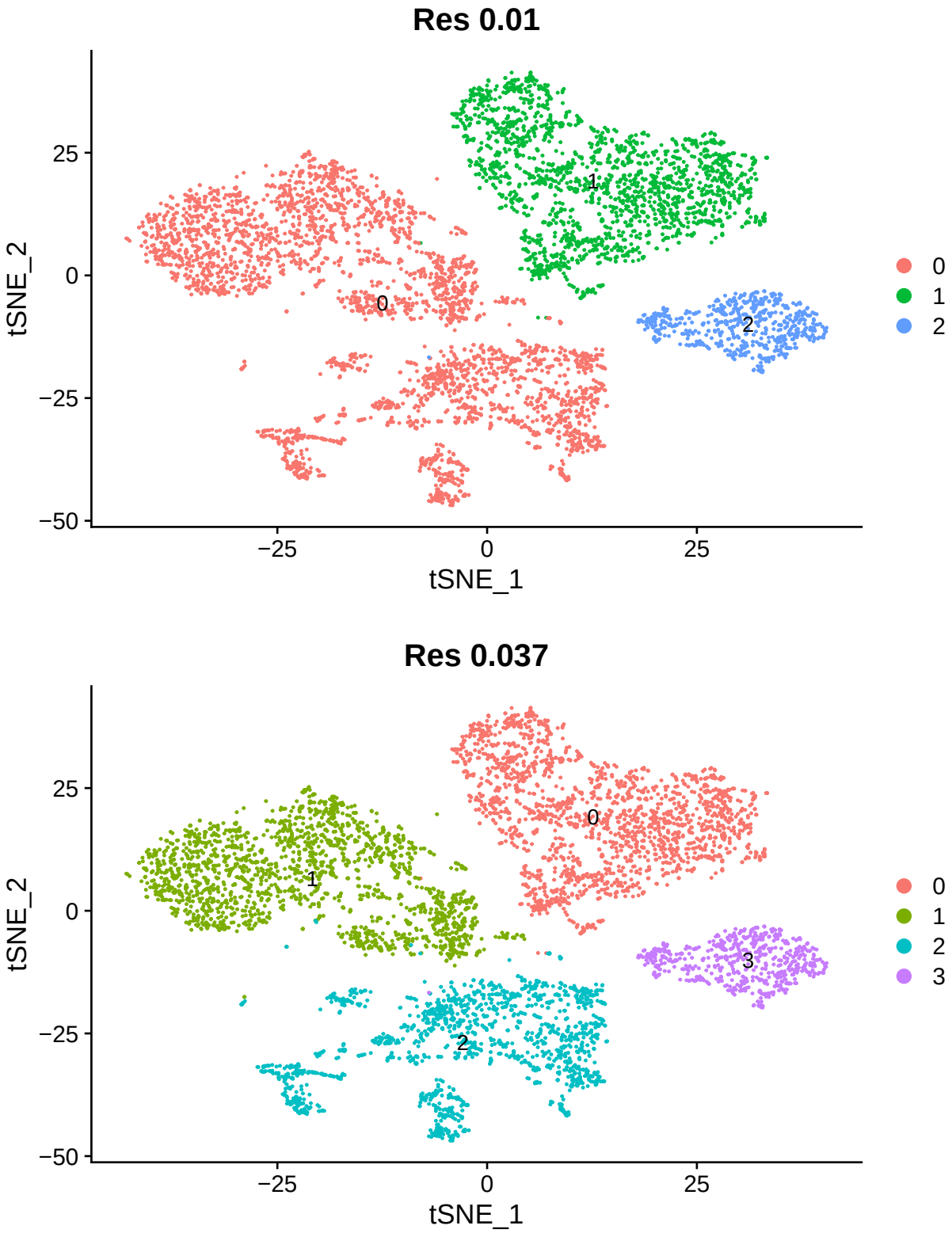



PyMT_Score1
25
0
−25
−50
tSNE_2
−25
0
25
tSNE_1
6
4
2
0



In defense of Yeo et al, lowering the resolution actually did seem to come close to differntiating these groups in a way that would appear to be natural, even in terms of clustering.

```
ggarrange(d0_01, d0_037,
          ncol = 1, nrow = 2)
```

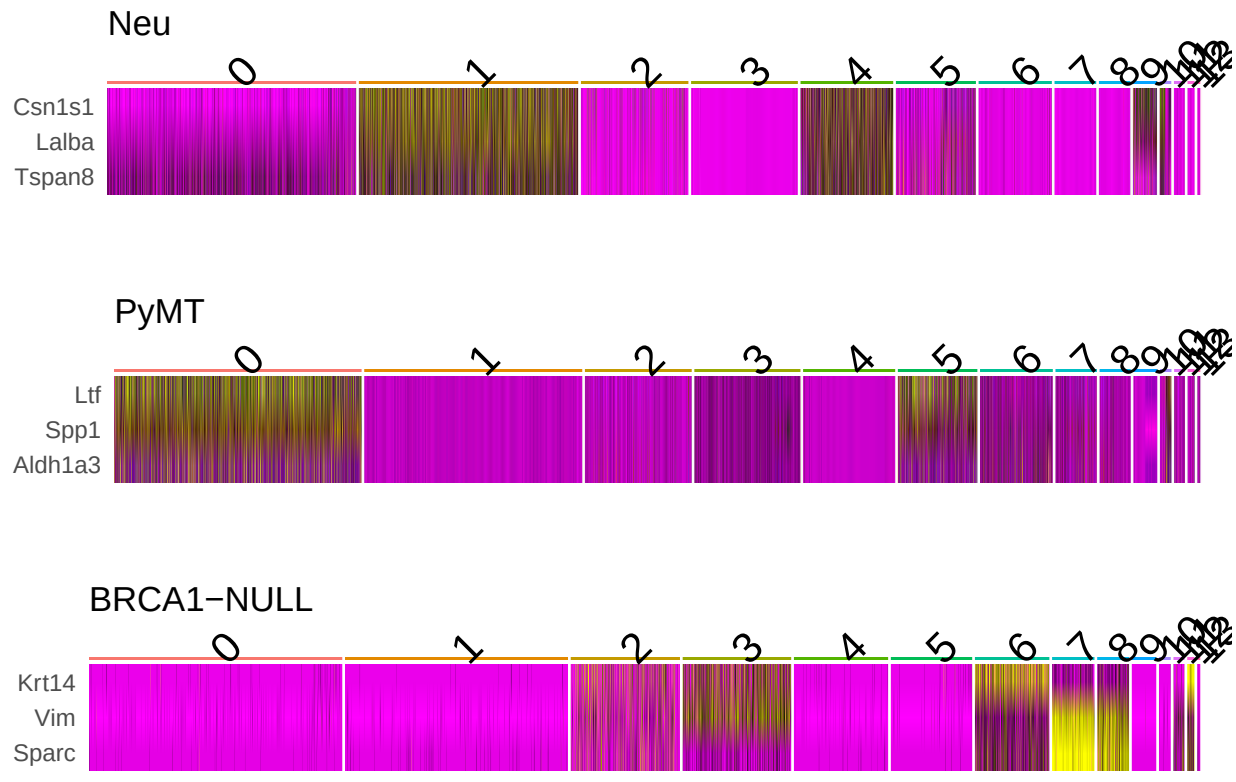


Because the original clustering from the study and this effort did not coincide well (apparently as a reuslt of using two different clustering methods, supervised and dunsupervised), but the FeaturePlots of AddModuleScores did (to a some extent), three clusters from the clustering at resolution 0.5 were chosen for

downstream analysis which appeared enriched for the gene markers mentioned in the study. In particular, clusters 0 and 5 appear to coincide largely with PyMT type cells from the AddModuleScore plot, indicating that these are a subset of "PyMT type cells" from the Yeo et al. study. Clusters 1 and 4 appear to consist largely of Neu type cells according to the AddModuleScore plot of that gene set, and so were thought to represent subsets of cells associated with Neu cells from Yeo et al. Clusters 3, 6, 7 and 8 coincide with the FeaturePlot of the AddModuleScore for BRCA1_NULL cell genes, and were thought to thus each represent a subset of BRCA1-NULL cells from the Yeo et al. study.

To confirm this, differential expression of these sets of genes in these clusters were plotted, passing AddModuleScores as features in the "DoHeatmap" function (this functionality appears to have been added to Seurat's "DoHeatmap" after 2020: https://github.com/satijalab/seurat/issues/3366) which confirmed this diferenitability.

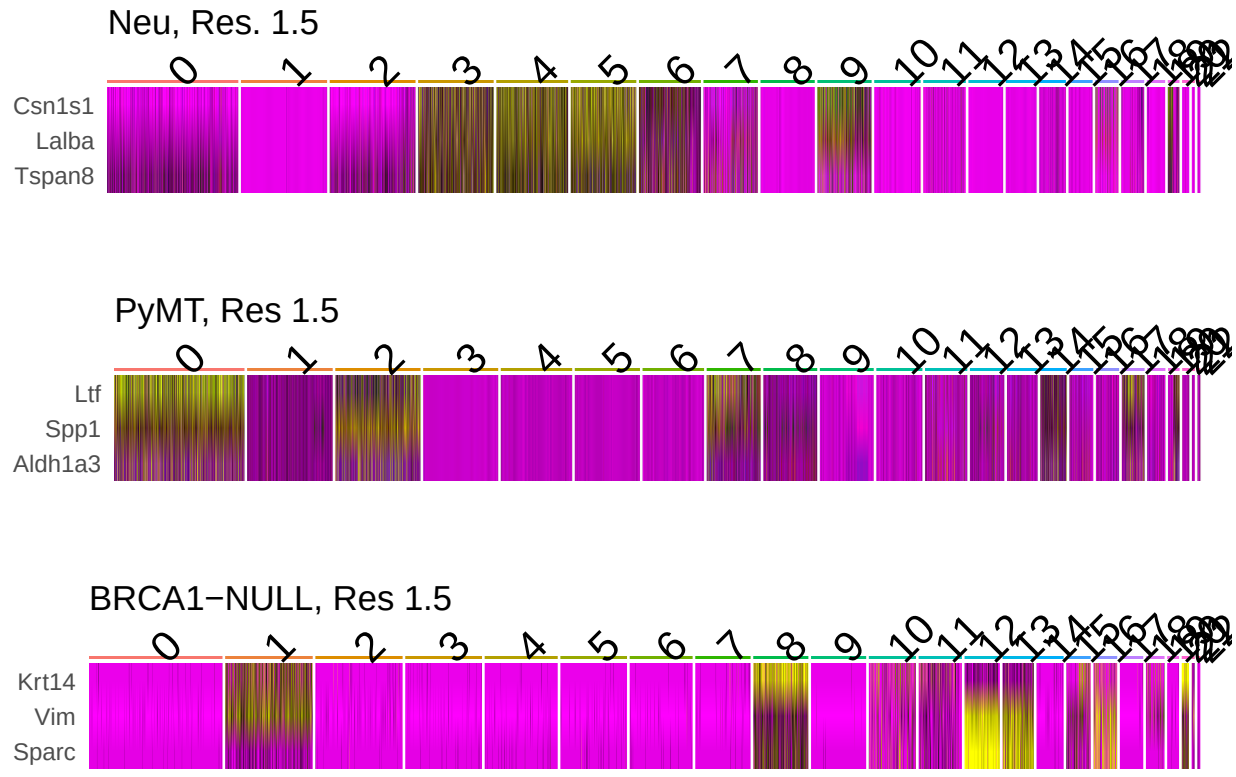
```
# First we have to set the resolution to use
Idents(S_O) = S_O$RNA_snn_res.0.5

HM1<-DoHeatmap(S_O, features = Neu) + NoLegend()
HM2<-DoHeatmap(S_O, features = PyMT) + NoLegend()
HM3<-DoHeatmap(S_O, features = BRCA1_NULL) + NoLegend()

HM1<- HM1 + labs(title = "Neu")
HM2<- HM2 + labs(title = "PyMT")
HM3<- HM3 + labs(title = "BRCA1-NULL")

ggarrange(HM1, HM2, HM3,
          ncol = 1, nrow = 3)
```



These heat maps clearly show upregulation of the gene sets associated with certain cell types within the

indicated clusters. These heat maps are not, however, as dramatic as the heat map produced for the variable features used to perform PCA. To test whether this was a matter of uniformity of cells in clusters or the levels of expression of these genes, the resolution was again changed to see if there was any improvement in these heat maps at a higher clustering resolutions.

```
Idents(S_O) = S_O$RNA_snn_res.1.5

HM1<-DoHeatmap(S_O, features = Neu) + NoLegend()
HM2<-DoHeatmap(S_O, features = PyMT) + NoLegend()
HM3<-DoHeatmap(S_O, features = BRCA1_NULL) + NoLegend()

HM1<- HM1 + labs(title = "Neu, Res. 1.5")
HM2<- HM2 + labs(title = "PyMT, Res 1.5")
HM3<- HM3 + labs(title = "BRCA1-NULL, Res 1.5")


ggarrange(HM1, HM2, HM3,
          ncol = 1, nrow = 3)
```



This did not appear to lead to a dramatic improvement in clustering uniformity/contrast with respect to these genes. Therefore, a resolution of 0.5 was again chosen for use in the downstream analysis of this dataset. At a resolution of 0.5, clusters 0, 1, and 6 were chosen to represent PyMT cells, Neu cells, and BRCA1-NULL cells, respectively due to their sizes, their coincidence with AddModulScore features plots, and their heat map contrasts.

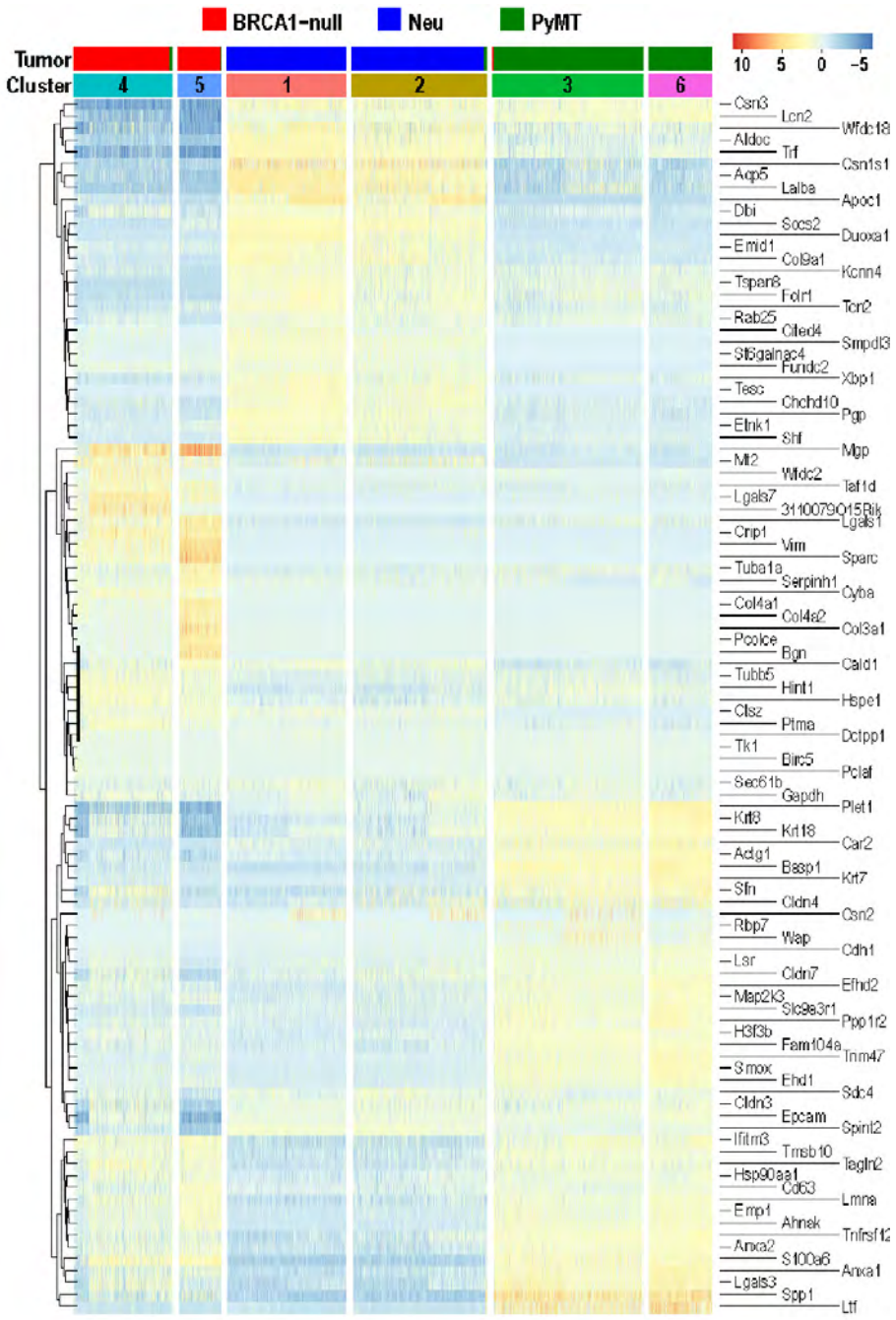
The "working resolution" for a Seurat Object can be set passing the resolution information to "Idents" of the Seurat Object.

```
Idents(S_O) = S_O$RNA_snn_res.0.5
```

It should be noted, Yeo et al. also performed unsupervised clustering following and based on their first clustering. In this instance, the three major clusters were further subdivided into six. Yeo et al. used another set of genes to differentiate these subgroups within the major cell types and clusters. In the next step of data integration with a healthy mouse dataset, however, the authors seem to return to the use of a simpler three cluster paradigm, so an inter-cluster comparison, such as Yeo et al make, was not performed here, for the sake of time and of focusing on the next step. However, this is mentioned because of the similar occurrence of "double clusters" among the major clusters. They mark together within the FeaturePlots for the AddModuleScores of marker genes (cluster 0 and 5, clusters 1 and 4, and clusters 3, and 6, 7, 8) and appear "together" in the heat maps of AddModuleScores as well as in the cluster plot.

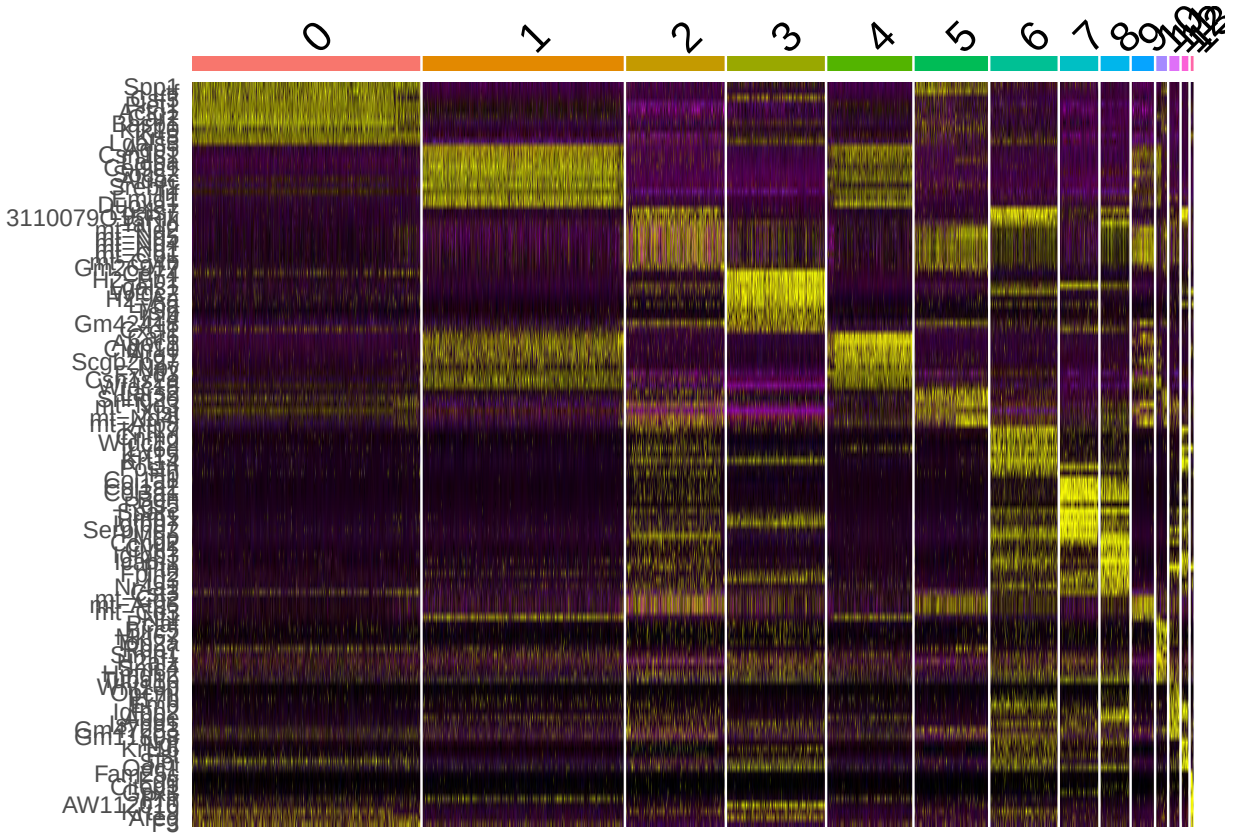
Yeo et al. also generated a heat map to both list and visualize the ten most upregulated genes for each cluster and their differential expression across clusters.

```
img <- readPNG("~/SeuratProjectRMD/Yeo2.png")
grid.raster(img)
```



The differenitiability of these groups on the basis of highly variable genes (not AddModuleScores) determined in an unsupervised manor by Seurat can be assessed in a similar way, by comparing the expression levels of their highly variable genes in a heat map as well.

```
S_0.markers <- FindAllMarkers(S_0, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)

S_0.markers %>%
  group_by(cluster) %>%
  top_n(n = 10, wt = avg_log2FC) -> top10
head(top10)
DoHeatmap(S_0, features = top10$gene) + NoLegend()
```



Again, we see remnants of the "three" major clusters (according to AddModuleScore results) appearing to be differentiable into two or more subclusters.

Again, as a result of this combined evidence, clusters 0, 1, and 6 were chosen as representatives of their cell types (PyMT, Neu, and BRCA1-NULL cell types, respectively) to simplify downstream analysis. These clusters can be subsetted into a unique Seurat object (S_O2) for convenience.

```
S_O2 <- subset(S_0, idents = c(0,1,6))
```

It might be worthwhile to note that, if you are using a computer with 6 Gb of RAM or less, it is a good idea to save and remove the Seurat Object you have been using (~ 2 Gb) from the R environment here. It is also a good idea to remove the expression matrix that was used to make the Seurat object (~500 Mb) here (if you have not already), since we will be loading a new expression matrix and creating a new Seurat Object of similar size. This can be accomplished using the "rm" command in R.

```
rm(S_0)
rm(expression_matrix)
```

II Integrate Analyzed Experimental Cancer Dataset with Dataset from Normal Mouse Mammary Cells.

As the Seurat website notes: "The joint analysis of two or more single-cell datasets poses unique challenges. In particular, identifying cell populations that are present across multiple datasets can be problematic under standard workflows. Seurat v4 includes a set of methods to match (or 'align') shared cell populations across datasets." (https://satijalab.org/seurat/articles/integration_introduction.html) Here, what they are talking about is dataset integration. For this, I have chosen to simply follow the Seurat tutorial.

The dataset which the experimental mammary tissue cancer dataset will be integrated with comes from a paper by Bach et al. entitled "Differentiation dynamics of mammary epithelial cells revealed by single-cell RNA sequencing" (2017). In this study, scRNA-seq was performed on four samples of cells from developmental stages (nulliparous, mid gestation, lactation and post involution) of normal mammary tissue in mice. This was done in order to develop a better understanding of the gene expression profiles of certain cells at different developmental stages and to "... provide a global, unbiased view of adult mammary gland development" (ibid). In this study, 15 cell clusters (C1:C15) were identified in this way (out of a total of 20 clusters), and characterized by marker genes (ibid).

Yeo et al. hypothesized that the heterogeneity of breast cancers may be a function of different developmental "hijacked" processes, and thus different cancer cells would be related to certain developmental states and lineages and normal developmental progenitor cells (a possibility also noted in the Bach study; In fact, in the first sentence of their paper they state: "[c]characterizing the hierarchy of mammary epithelial cells (MECs) and how they are regulated during adult development is important for understanding how breast cancer arises"). As Yeo et al. state, "The broad coverage of this normal MEC dataset across varying developmental stages would allow for the prospective identification of developmental processes hijacked by tumor cells in these mouse models (e.g. pregnancy mimicry, involution mimicry") (2020). Integration of these data sets using Seurat was thus performed in order to explore the possibility that cancer cell heterogeneity may reveal patterns associated with normal development of mammary cells through coclustering. As Yeo et al note, because of the difficulty measuring distances in the low dimensionality space of clustering, the proximity of certain cancer cells to cells from the developmental hierarchy of normal breast cells explored by Bach et al. could be "measured" more easily (if somewhat crudely) by simply merging the datasets into a Seurat Object and attempting to co-cluster them (Yeo et al. 2020). When the datasets were combined, 17 clusters were identified by Yeo et al. (2020).

The first step in this analysis is to subject the second dataset to the same workflow as was performed for the first.

Procedure

Like before, access the matrix, barcodes and features files through GEO (ac cession number: GSE106273). Follow the same preprocessing and analysis steps for the first dataset, and save the Seurat object (and any results) to a different Seurat Object name (and RDS file).

```
dirname2<- "~/SeuratProject/Mouse_Mammary"

exp_mat2 <- ReadMtx(mtx =
                      paste0(dirname2,"/GSE106273_combined_matrix.tsv"),
                            cells =
                    paste0(dirname2,"/GSE106273_combined_barcodes.tsv"),
                            features =  paste0(dirname2,
                    "/GSE106273_combined_genes.tsv"))
```

Once the expression matrix was made, it was sampled randomly to generate a subset of 12,000 cells which generated a second expression matirx (Yeo et al., 2020).

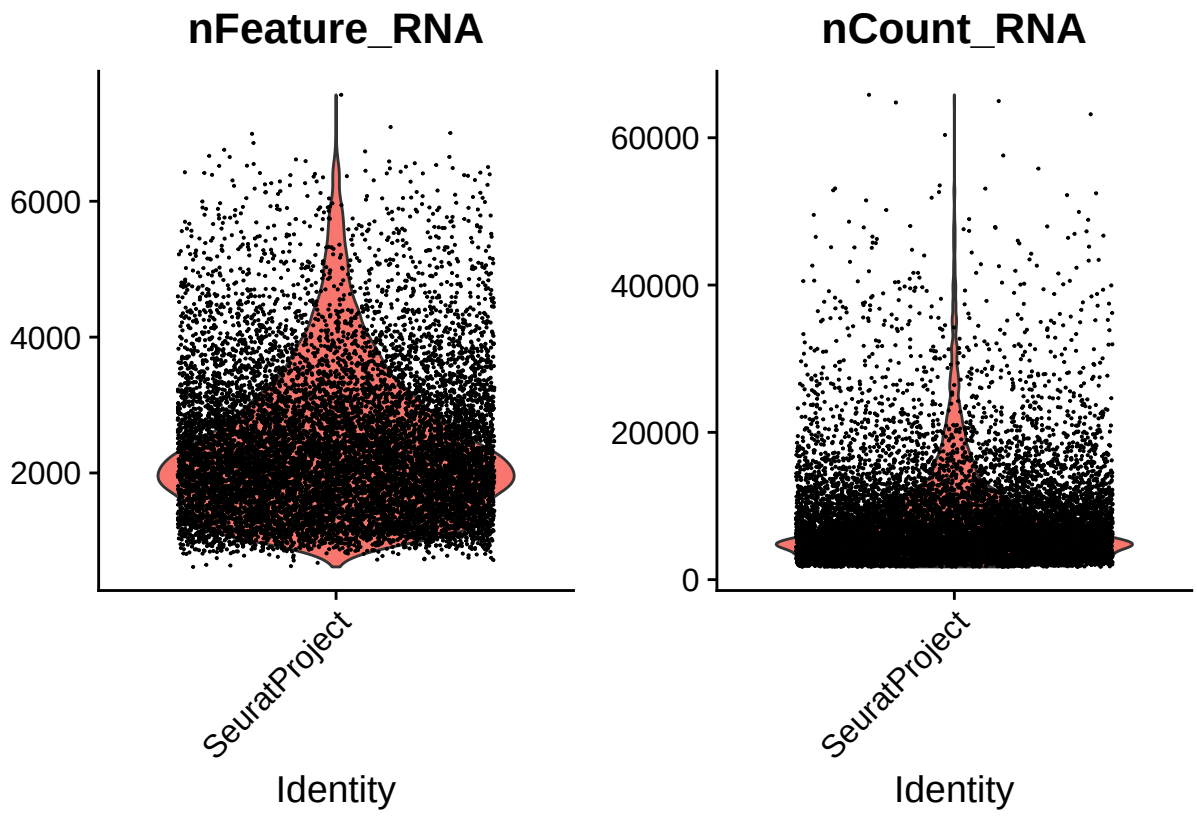
```
exp_mat_2<-exp_mat2[,sample(1:ncol(exp_mat2),12000)]
```

Now, we follow the same steps for creating the Seurat Object and preprocesing that were performed using the first dataset (it might also be a good idea to remove expression matrices from the R environment because they are rather large too).

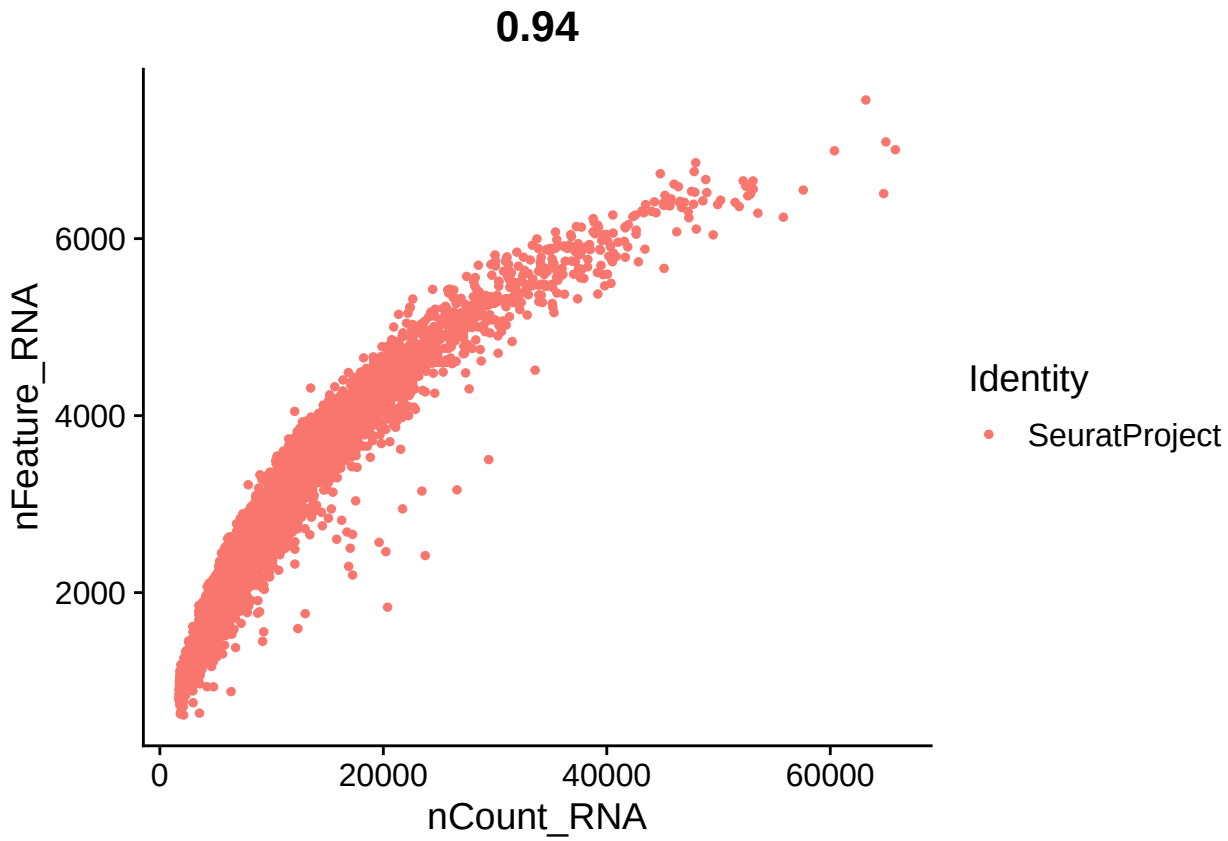
```
SeurObj2 <- CreateSeuratObject(counts = exp_mat_2)
rm(exp_mat2)
rm(exp_mat_2)
```

Here, again, we are just making sure that the raw data looks normal, and then we normalize the data, identify highly variable features, and scale the data.

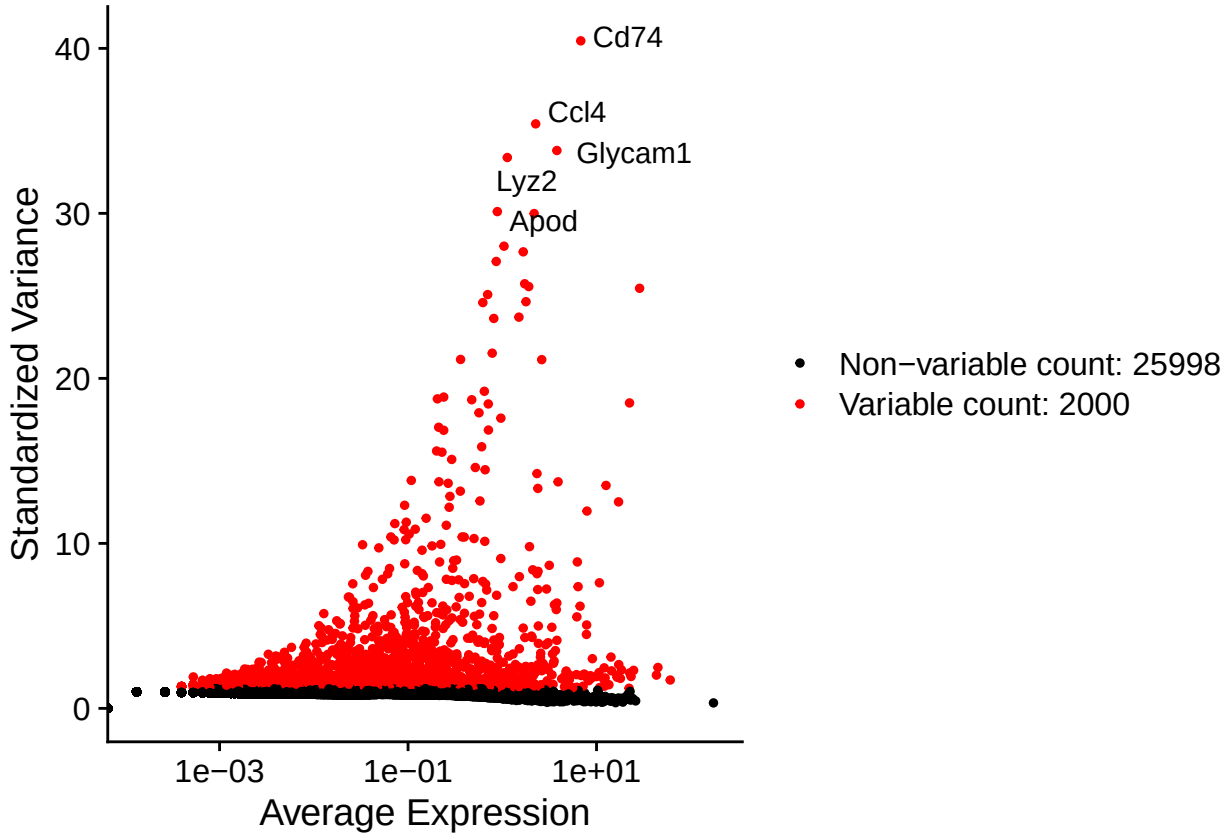
```
# 1 Preprocessing
SeurObj2[["percent.mt"]] <- PercentageFeatureSet(SeurObj2, pattern = "^MT-")

VlnPlot(SeurObj2, features = c("nFeature_RNA", "nCount_RNA"), ncol = 2)
```




```
plot4 <- FeatureScatter(SeurObj2, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot4
```



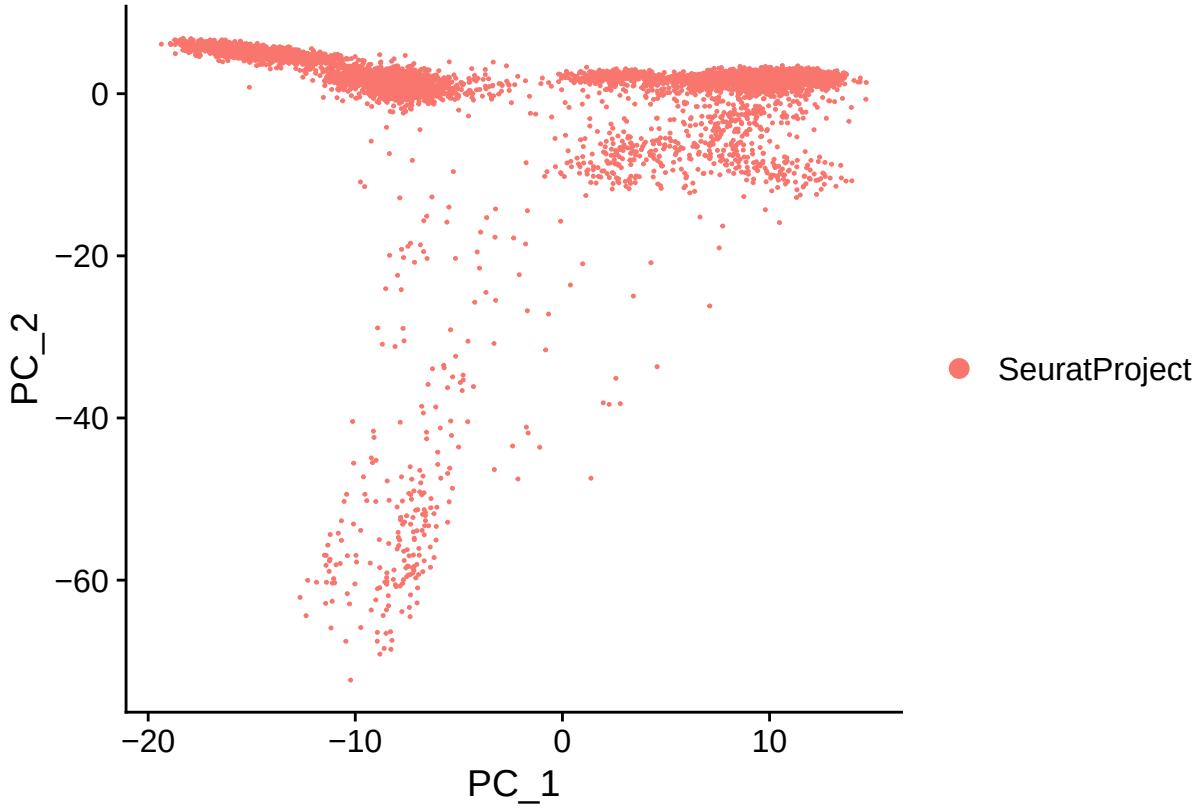

```
SeurObj2 <- subset(SeurObj2, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)

# Normalization
SeurObj2 <- NormalizeData(SeurObj2, normalization.method = "LogNormalize", scale.factor = 10000)

# Identification of highly variable features
SeurObj2 <- FindVariableFeatures(SeurObj2, selection.method = "vst", nfeatures = 2000)

# Plot variable features with labels
plot1 <- VariableFeaturePlot(SeurObj2)
top5 <- head(VariableFeatures(SeurObj2), 5)
plot2 <- LabelPoints(plot = plot1, points = top5, repel = TRUE)
plot2
```



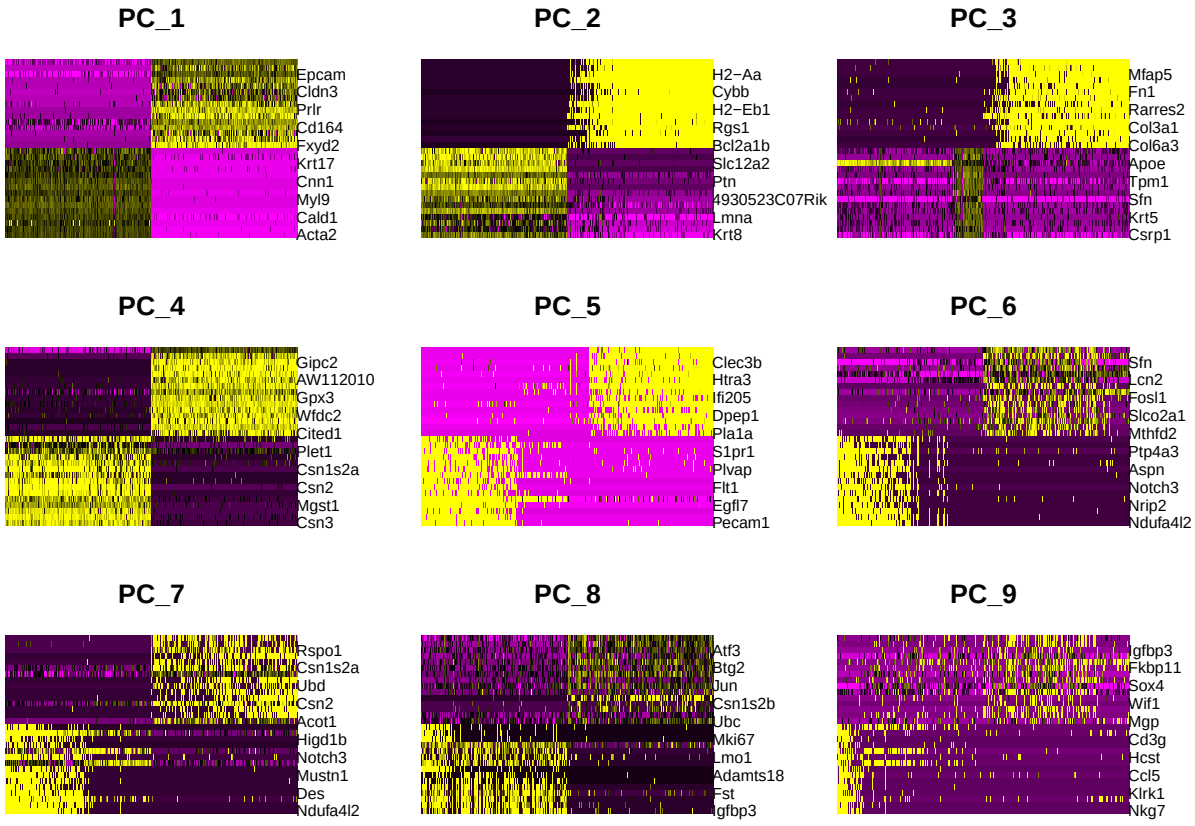

```
# Scaling the data
all.genes <- rownames(SeurObj2)
SeurObj2 <- ScaleData(SeurObj2, features = all.genes)
```

The analysis steps will also be the same as was performed for the first dataset: we are running PCA with the variable features that have been identified, choosing PCs based on the "Jackstraw" function and JackStraw plot, we are inputting these into the "FindNeighbors" and "FindClusters" functions, and then performing non-linear dimensionality reduction.

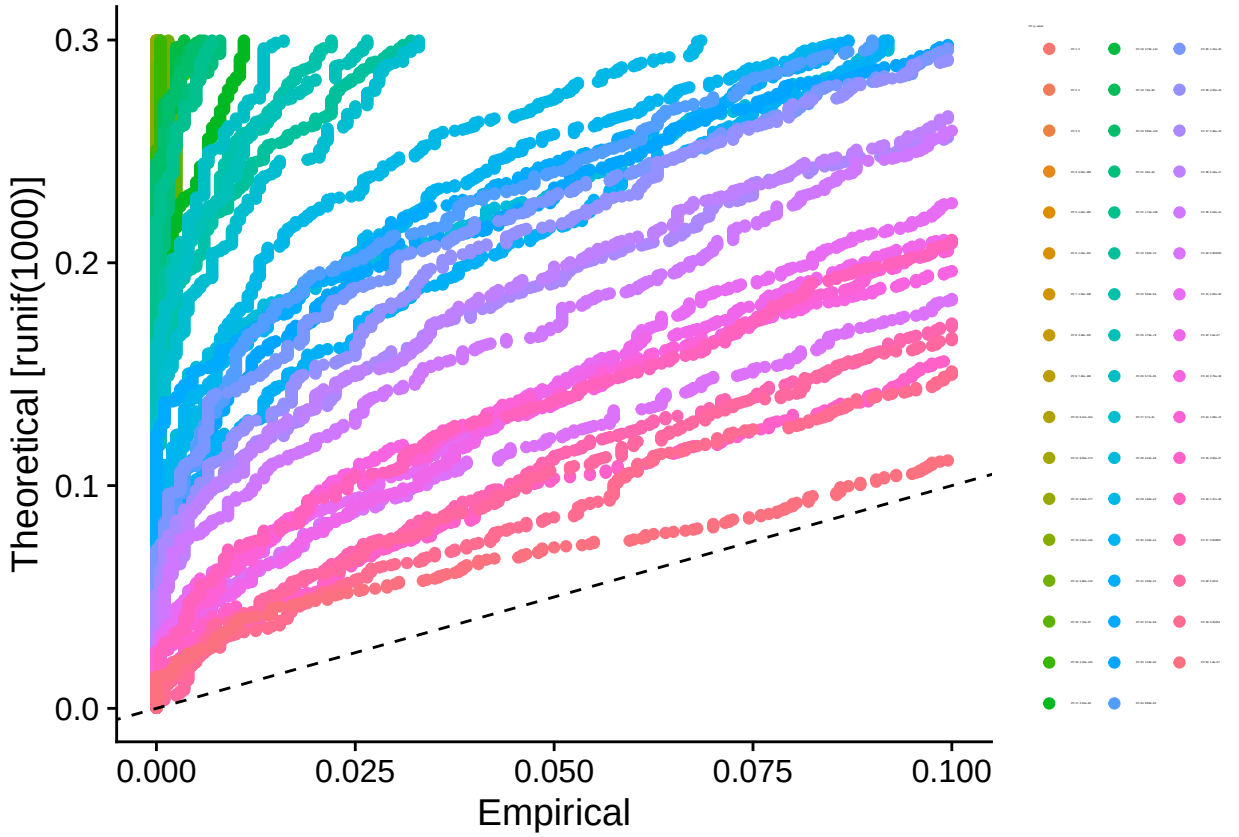
```
# 2 Analysis
# 5 Perform linear dimensionality reduction
SeurObj2 <- RunPCA(SeurObj2, features = VariableFeatures(object = SeurObj2))

# Visualize and Examine PCA results
DimPlot(SeurObj2, reduction = "pca")
```




```
DimHeatmap(SeurObj2, dims = 1:9, cells = 500, balanced = TRUE)
```

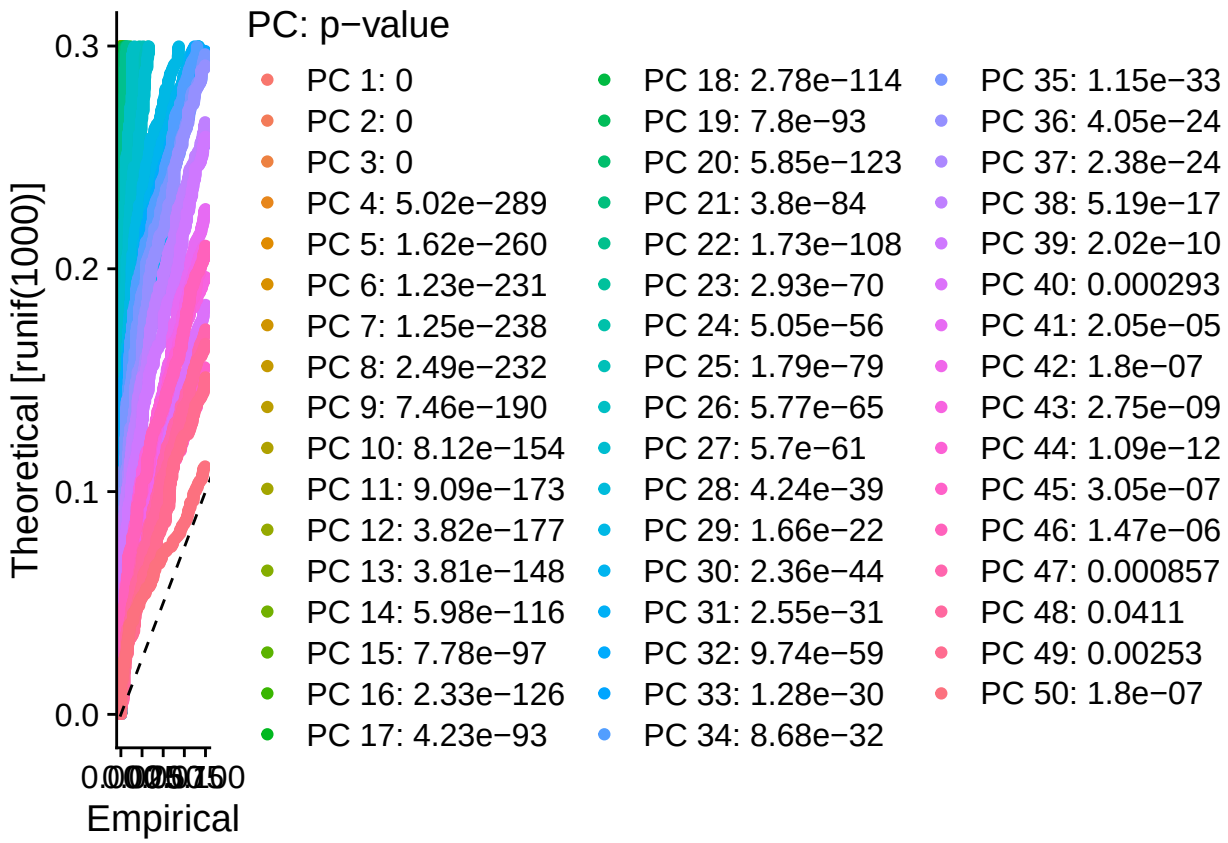



```
# 6 Determine the dimensionality of the dataset
SeurObj2 <- JackStraw(SeurObj2, num.replicate = 100, dims = 50)
SeurObj2 <- ScoreJackStraw(SeurObj2, dims = 1:50)

#Visualize Jackstraw and Elbow Plot
JS2<-JackStrawPlot(SeurObj2, dims = 1:50)
JS2<- JS2 + theme(legend.title = element_text(size = 1.0),
                  legend.text = element_text(size = 1.0))


JS2
```



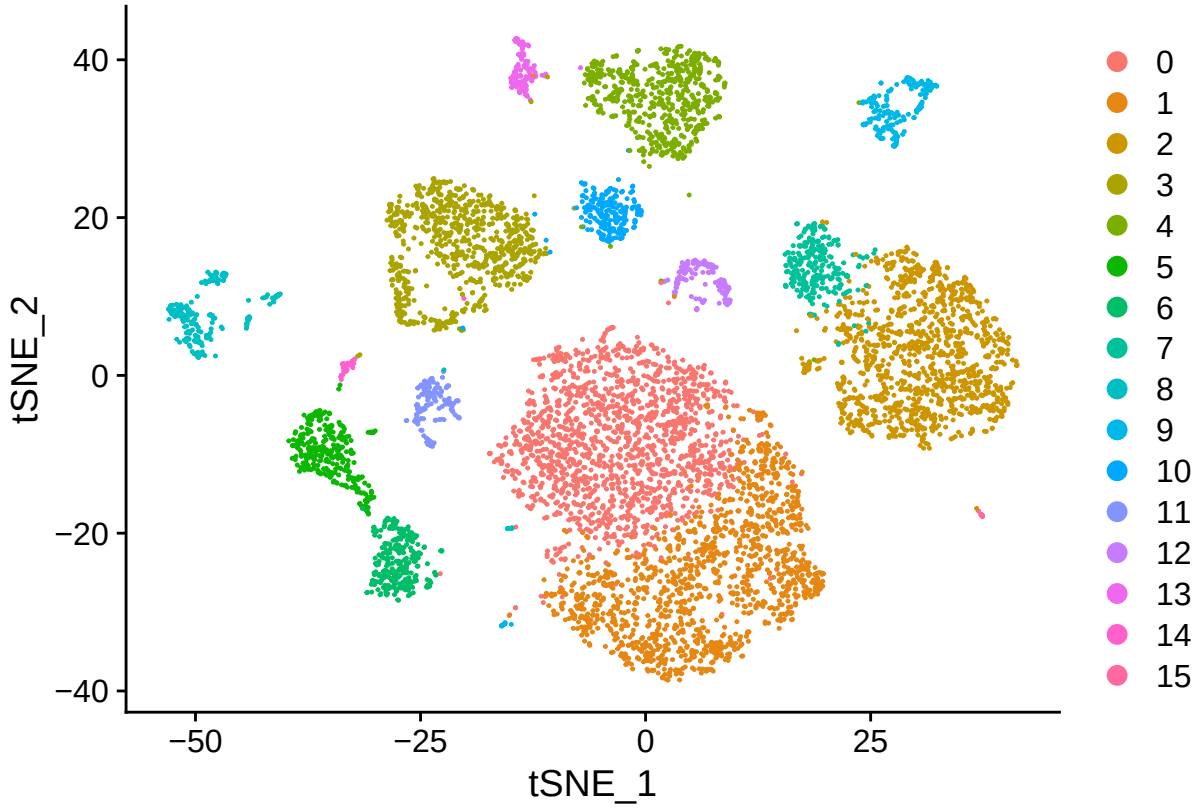
As ws the case for the previous dataset, there were too many PCs in the JackStraw plot for the legend to be intelligible, and to see the P values associated with PCs, at the same time. Both versions of the plot are shown (first with reduced legend size and then normal legend size).

```
JS2<-JackStrawPlot(SeurObj2, dims = 1:50)

JS2
```



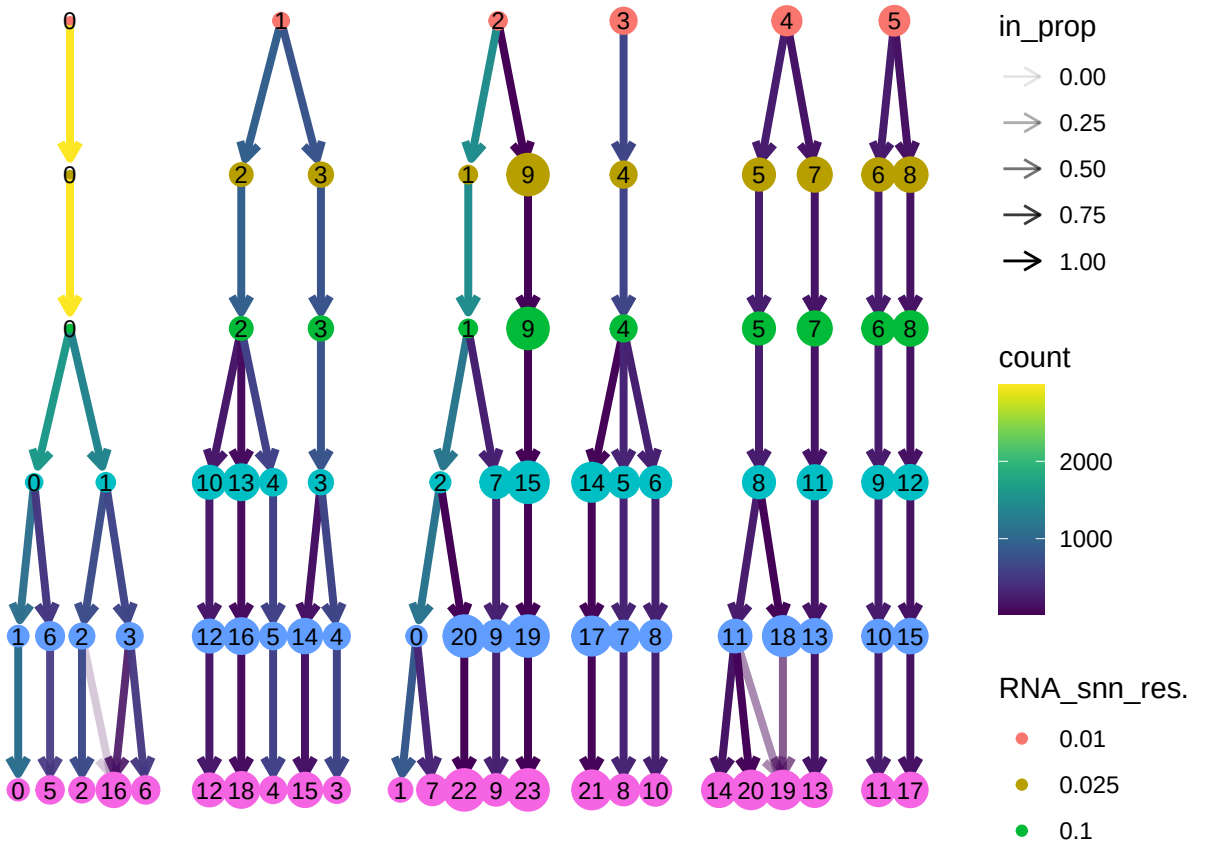
37 PCs were chosen from the JackStraw plot for the second dataset.

```
# 7 Cluster the cells
SeurObj2 <- FindNeighbors(SeurObj2, dims = 1:37)
SeurObj2 <- FindClusters(SeurObj2, resolution = 0.5)

#8 Non-linear dimensionality reduction
SeurObj2 <- RunTSNE(SeurObj2, dims = 1:37)

# Visualize non-linear dimensionality reduction
DimPlot(SeurObj2, reduction = "tsne")
```



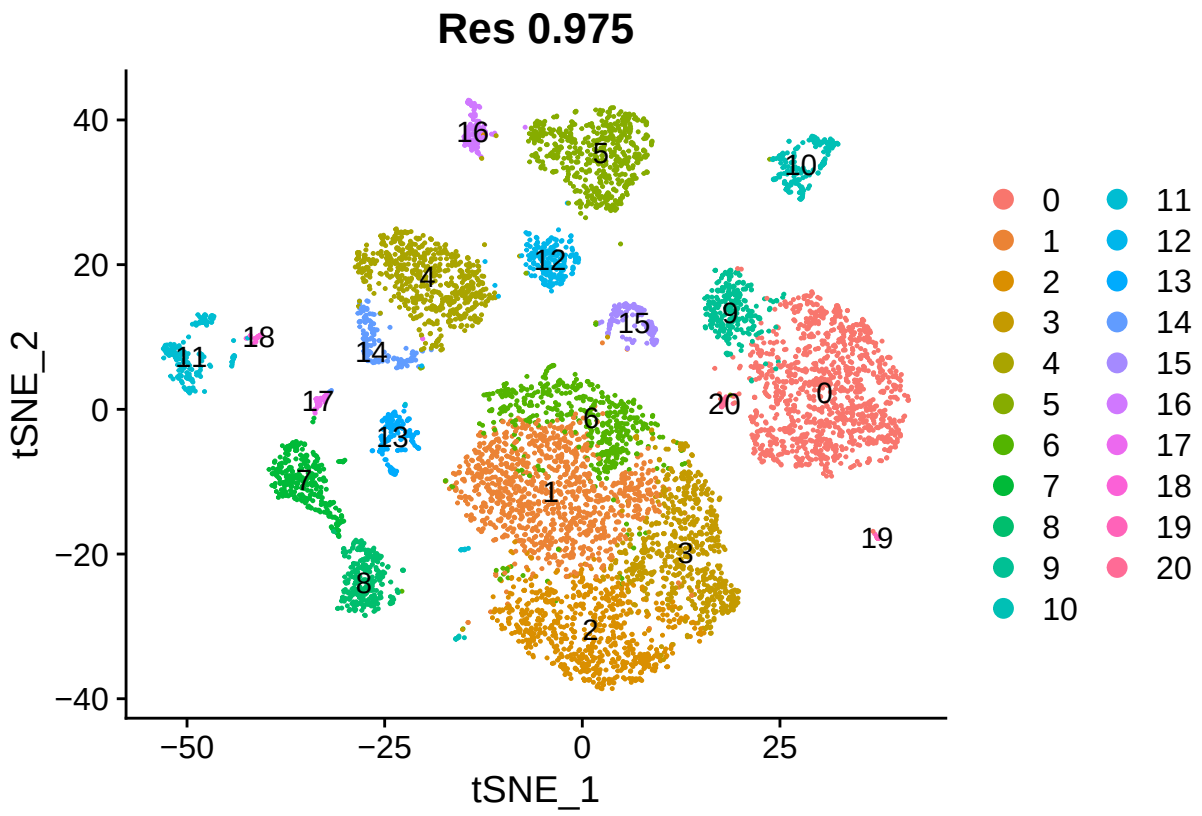
Again, a cluster tree can be examined to determine optimal clustering resolution, and this is done by scanning a collection of resolutions during clustering, running non-linear dimnensionality reduction again, and examining the cluster tree using the "clustree" function.

```
# Test different resolutions
SeurObj2 <- FindClusters(SeurObj2, resolution = c(0.01, 0.025,
                                                  0.1, 0.5,
                                                  0.975, 1.5))
# Run non-linear dimensionality reduction
SeurObj2 <- RunTSNE(SeurObj2, dims = 1:37)

# Run clustree and Visualize
clustree(SeurObj2, node_size_range = c(7, 1), edge_width = 1.4)
```



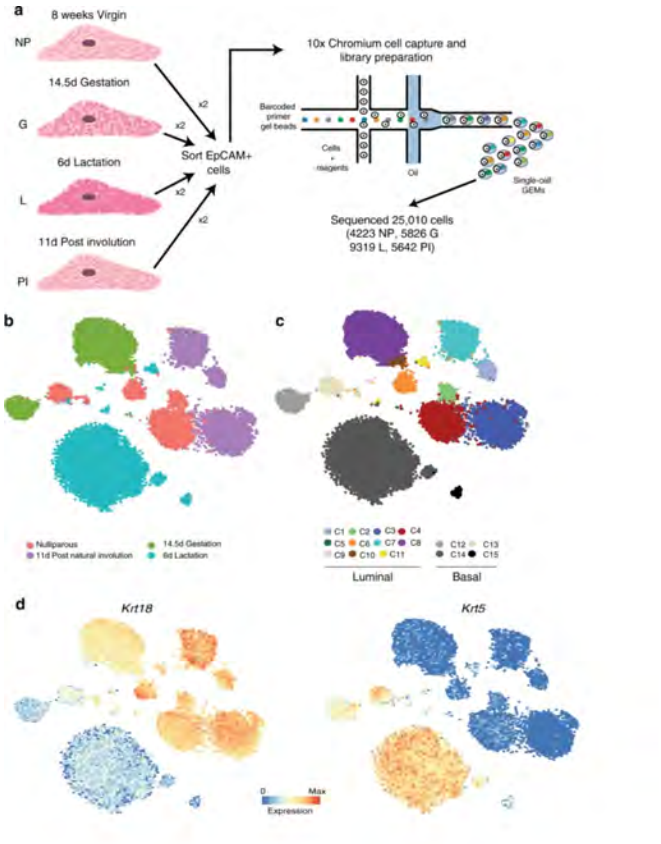
Inspection of the cluster tree for the second dataset revealed that unsupervised clustering produced 21 clusters (like were generated in the Bach et al. study) at a resolution of 0.975. This resolution appeared to balance efficient differentiation of cells (clustering) with a minimal amount of inter-resolution convergence on individual child nodes from multiple parent nodes (and came close to recreating their results or 20 cluasters). So to facilitate comparisons with the Bach et al. study (which was not absolutely necessary), a resolution of 0.975 was chosen for use in downstream analysis.

```
d20_975<-DimPlot(SeurObj2, label= TRUE, reduction = "tsne", group.by = "RNA_snn_res.0.975") + ggtitle("

d20_975
```



This is a figure showing the results of clustering rom the Bach et al. study (2017).

```
img <- readPNG("~/SeuratProjectRMD/Bach1.png")
grid.raster(img)
```



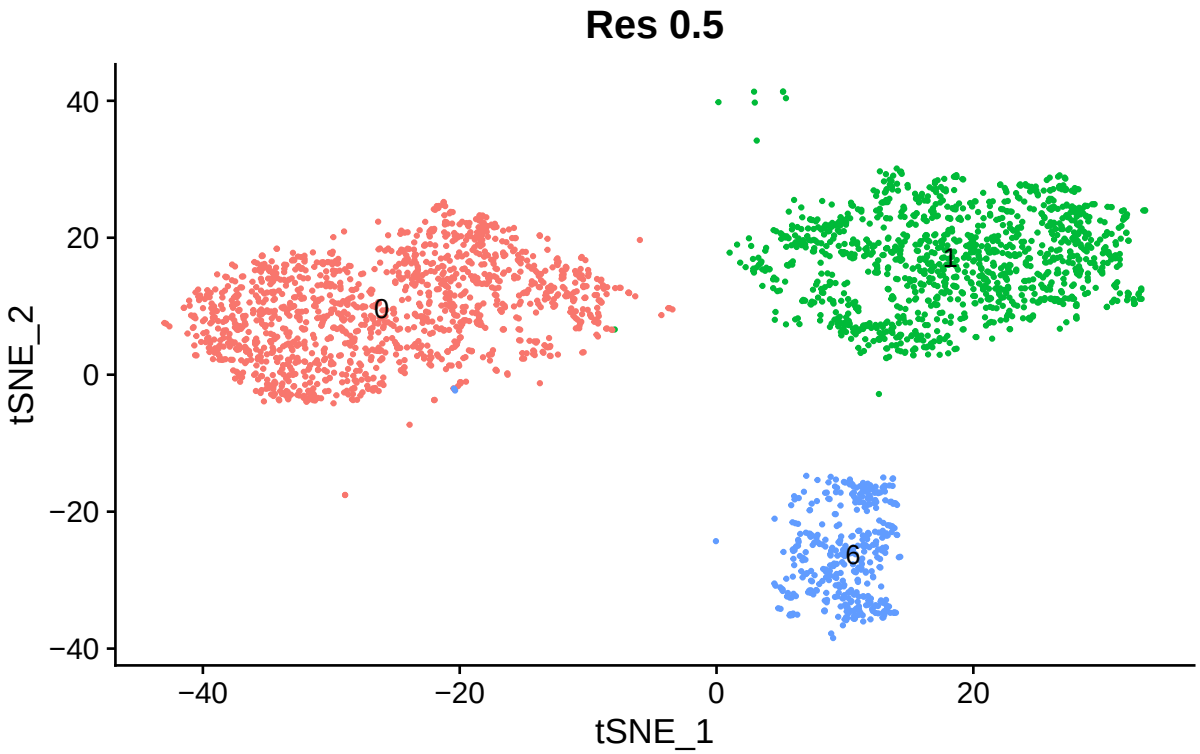
Then, again, we set the chosen resolution as the working identity for the Seurat Object.

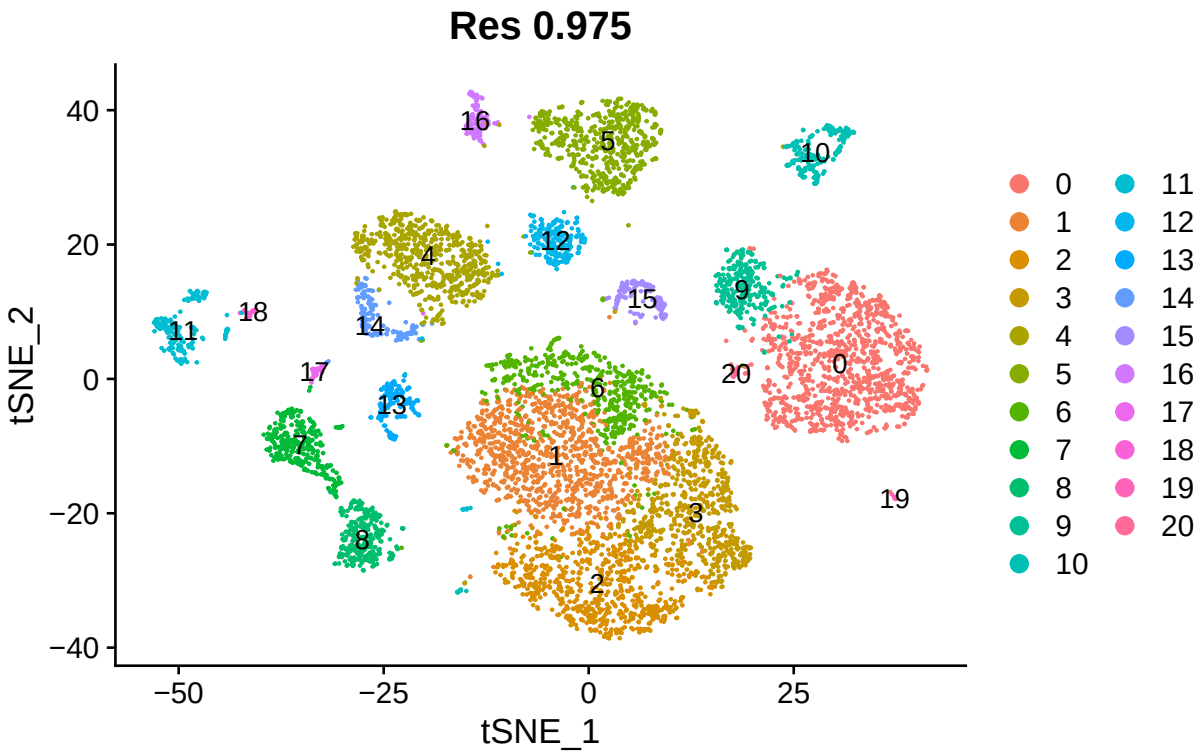
```
Idents(SeurObj2) <- SeurObj2$RNA_snn_res.0.975
```

Although the clustering results from Bach et al. have almost been reproduced (in terms of the number of clusters), sorting out the cell types from this DimPlot is not so simple. It therefore seemed more useful to run the integration step first, and then identify the relevant clusters (those demonstrating overlap with the cancer cell dataset).

Looking at the two datasets before integration, we have:

```
new.cluster.ids <- c("PyMT","Neu","BRCA1-NULL")
names(new.cluster.ids) <- levels(S_O2)
S_O2 <- RenameIdents(S_O2, new.cluster.ids)

d220_5<-DimPlot(S_O2, label= TRUE, reduction = "tsne", group.by = "RNA_snn_res.0.5") + ggtitle("Res 0.5

ggarrange(d220_5, d20_975,
          ncol = 1, nrow = 2)
```





Integrating Datasets

Like Yeo et al, we will not use dimensionality reduced distances to estimate the relation of cells. Instead, we will use coclustering, a somewhat coarse test for relatedness (later, Yeo et al. use the package Monocle

to generate "pseudotime" values (another analysis that can be performed on scRNA-seq data between cells believed to be related along a developmental trajectory) associated with the cancer cells in relation to their normal co-clustered neighbors, as supporting evidence of their relatedness (as well as a measure of it). Although also performing this analysis was considered (and was happily anticipated), Seurat turned out to be more than a handful in terms of the time required to learn as much as was learned. In addition to this, generating this manuscript using RMarkdown took considerable time. So, unfortunately, to my great disappointment, I did not have time to become familiar with Monocle as well).

Many of the steps in data analysis are repeated on the integrated dataset. But first we must integrate the data (https://satijalab.org/seurat/articles/integration_introduction.html).In the tutorial, two sample datasets are merged first (which erases normalization and scaling in the two datasets) before they are again split in preparation for integration. This, however, is not necessary if both datasets have used the same methods for normalization and scaling. It is worth noting that merging seems to only be appropriate when the steps of preprocessing have to be done from scratch (so they can be done together, conveniently, at one time), or for datasets that could be considered replicates of each other (https://www.biostars.org/p/9493216/). Integration, rather than simple merging, is a more "guided" way of bringing two datasets together using "anchors". This method was developed by Stuart et al. (2019. It is preferred to simple "merging" when similar cells in two datasets do not cluster together as a result of batch effects or biological differences but still may be related (https://github.com/satijalab/seurat/discussions/3998). It involves the use of "anchors", which are designed to generate a "cell pairwise correspondences between single cells across datasets" that allows one to "transform datasets into a shared space, even in the presence of extensive technical and/or biological differences" (Stuart et al. 2019). As Stuart et al. state:

"In order to relate different experiments to each other, we assume that there are correspondences between datasets and that at least a subset of cells represent a shared biological state. Inspired by the concept of MNNs, we represent these correspondences as two cells (one from each dataset) that we expect to be defined by a common set of molecular features (Haghverdi et al., 2018). While MNNs have previously been identified using L2-normalized gene expression, significant differences across batches can obscure the accurate identification of MNNs, particularly when the batch effect is on a similar scale to the biological differences between cell states. To overcome this, we first jointly reduce the dimensionality of both datasets using diagonalized CCA, then apply L2-normalization to the canonical correlation vector... We next search for MNNs in this shared low-dimensional representation. We refer to the resulting cell pairs as anchors, as they encode the cellular relationships across datasets that will form the basis for all subsequent integration analyses... Our anchors can successfully recover matching cell states even in the presence of significant dataset differences, as CCA can effectively identify shared biological markers and conserved gene correlation patterns (Butler et al., 2018). However, cells in non-overlapping populations should not participate in anchors, representing an important distinction that extends our previous work."

First, we set their identities in the metadata slots of the Seurat Object, so that we can "demultiplex" them afterwards.

```
S_O2 <- AddMetaData(S_O2, metadata="cancer1", col.name="orig.obj")
SeurObj2 <-AddMetaData(SeurObj2, metadata="mammary1", col.name="orig.obj")
```

The Seurat Objects must be entered into the "SelectIntegrationFeatures" function and the "FindIntegrationAnchors" fucntion as a list, so we make this list first.

```
OL1<- list(S_O2, SeurObj2)
```

Then we run the integration steps. First we find the anchors, then integrate the datasets.

```
# Choose features that are jointly variable between datasets
features1 <- SelectIntegrationFeatures(object.list = OL1)

#Identify anchors
```

```
cancer.anchors <- FindIntegrationAnchors(object.list = OL1, anchor.features = features1)

# Make an 'integrated' data assay
cancer.combined <- IntegrateData(anchorset = cancer.anchors)
```

It might be helpful to your system's resources to remove the old Seurat Object (I will be using the subsetted object from the first dataset again) from the R environment here, now that the integrated Seurat Object is created. The list of the two Seurat Objects used to find anchors and integrate the datasets is also quite large and can be removed now as well.

```
rm(SeurObj2)
rm(OL1)
```

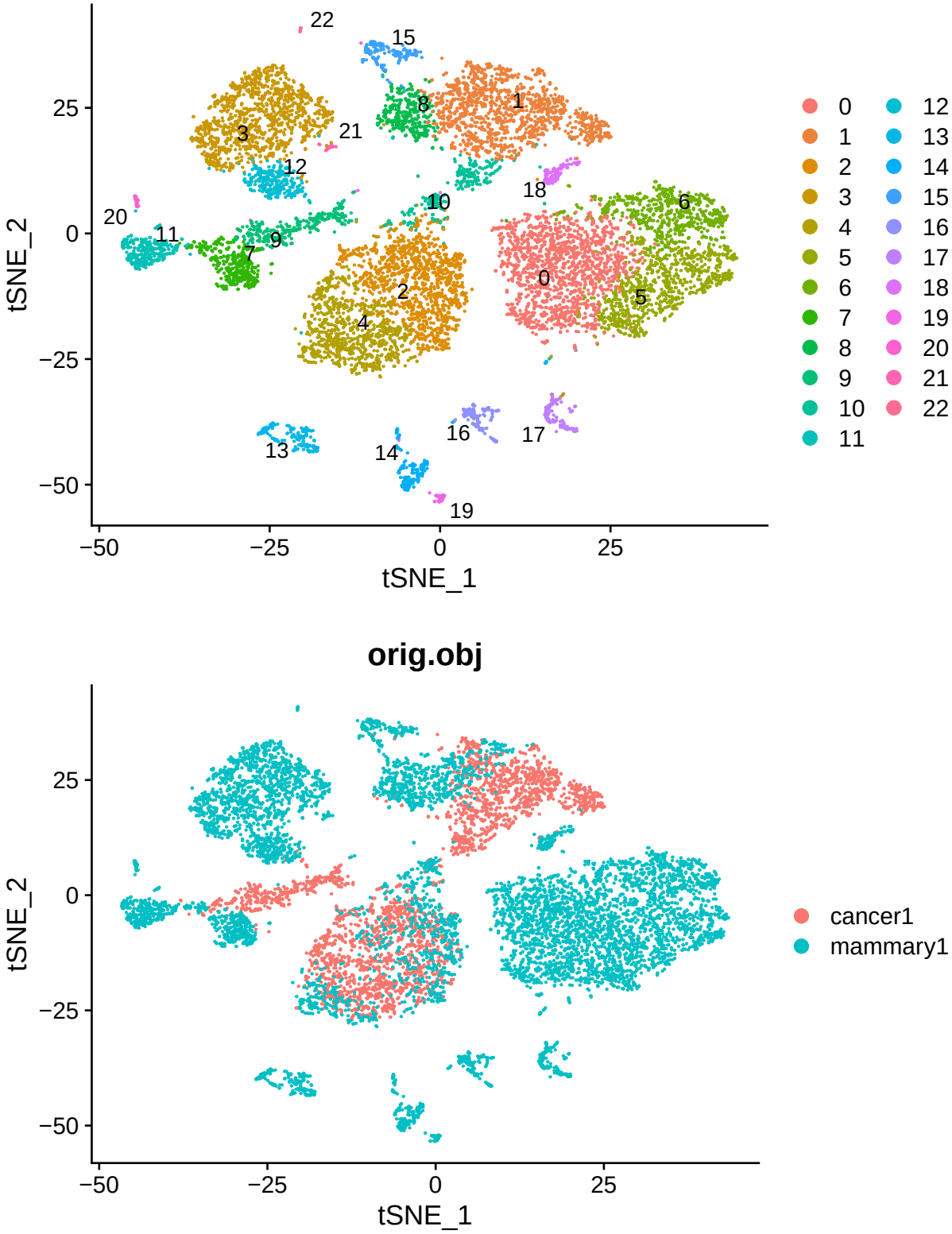
```
#Perform an integrated analysis

# As the tutorial notes, The original data is in the 'RNA' assay,
# so we have to indicate that downstream processing involves the
# integrated data
DefaultAssay(cancer.combined) <- "integrated"
```

Now that we have the datasets integrated, we can run the normal workflow on them. The number of PCs and resolution used for clustering from the Bach et al. dataset will be used here, to maintain (as much as possible) their clustering for typing (which is not absolutely necessary, since we have a long list of marker genes associated with cells. But, differentiating the cells as much as possible in the way they were differentiated in clustering by Bach et al. may be helpful. Furthermore, theirs is the "standard" by which the cell cancer dataset is being compared).

```
# Normal workflow
cancer.combined <- ScaleData(cancer.combined, verbose = FALSE)
cancer.combined <- RunPCA(cancer.combined, npcs = 37, verbose = FALSE)
cancer.combined <- RunTSNE(cancer.combined, reduction = "pca", dims = 1:37)
cancer.combined <- FindNeighbors(cancer.combined, reduction = "pca", dims = 1:37)
cancer.combined <- FindClusters(cancer.combined, resolution = 0.975)
```

The results of the integrated clustering can be visualied using the DimPlot function.

```
p3 <- DimPlot(cancer.combined, reduction = "tsne", label = TRUE, repel = TRUE)
p4 <- DimPlot(cancer.combined, reduction = "tsne", group.by = "orig.obj", label = FALSE)

ggarrange(p3, p4, ncol = 1, nrow = 2)
```

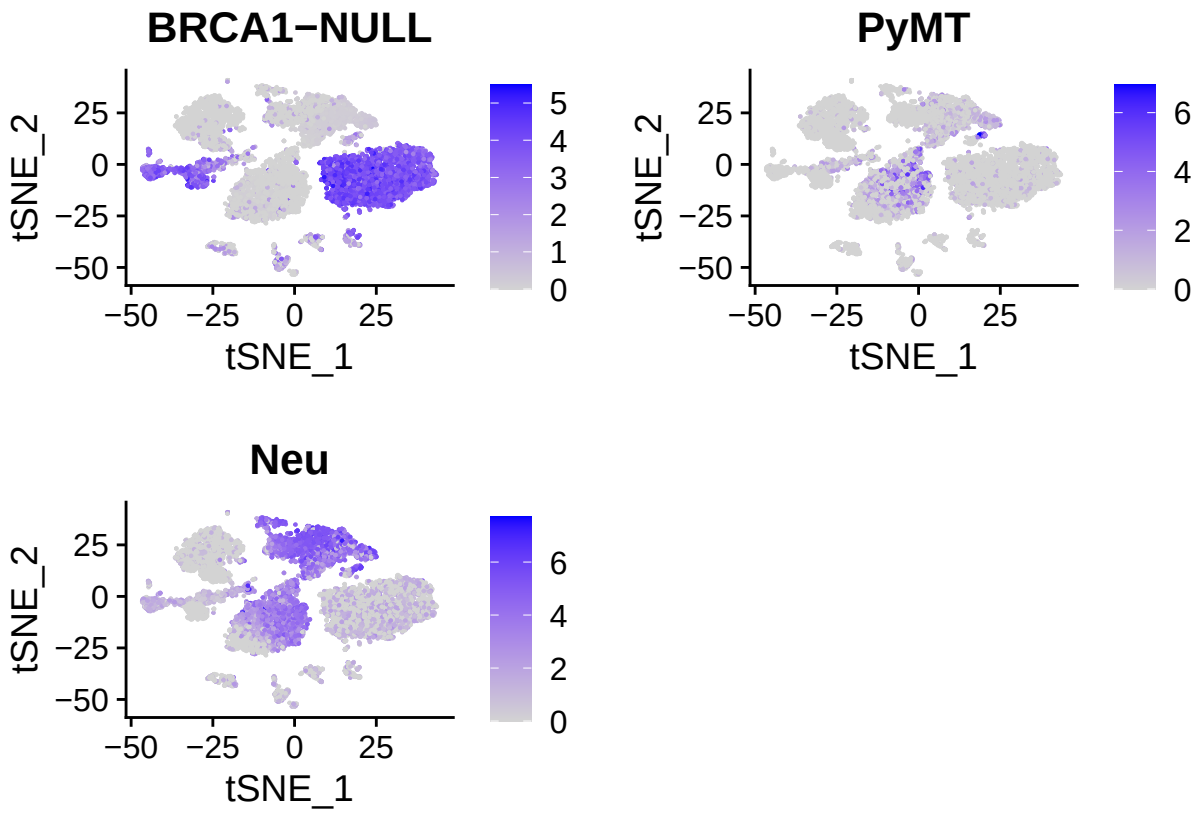


Comparing the combined clustering with the original identities of the cells in the datasets, clusters 1, 2, 4 and 7 appear to have the most (or significant) overlap.

The original identities of the cancer cells seem to not be visualizable using the AddModuleScore function in

conjunction with the FeaturesPlot function with integrated data. In this case, we can just use a marker gene from the Yeo et al. study to do identify them.

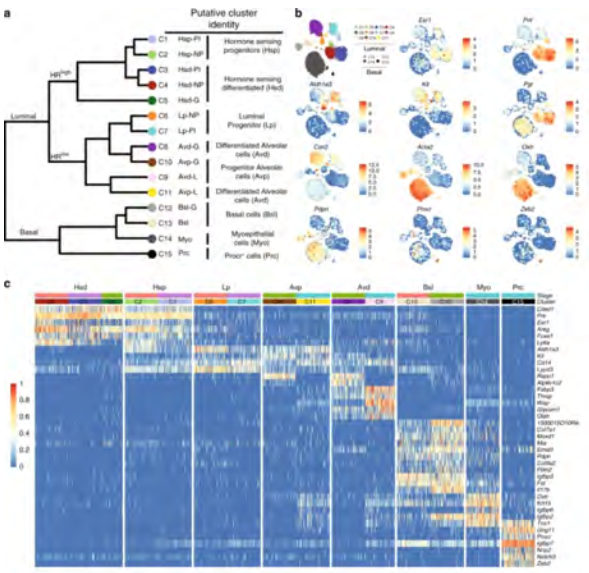
```
FPZ<-FeaturePlot(object = cancer.combined, features = "Krt14", min.cutoff = 0) + ggtitle ("BRCA1-NULL")
FPY<-FeaturePlot(object = cancer.combined, features = "Spp1", min.cutoff = 0) + ggtitle ("PyMT")
FPX<- FeaturePlot(object = cancer.combined, features = "Csn1s1", min.cutoff = 0) + ggtitle ("Neu")

ggarrange(FPZ, FPY, FPX, ncol = 2, nrow = 2)
```



Using this information, cluster 1 was identified as containing Neu cells. Clusters 2 and 4 were identified as containing PyMT cells. The BRCA1-NULL cells appear to be found partly in cluster 7. (It should be noted, that a FeaturePlot of the Csn1s1 gene in the Yeo et al. study [Figure 1, Yeo et al. 2020] also showed overlap with the PyMT cluster - but of the three genes shown, it demonstrated the least.)

For the normal mammary cells, Bach et al. use marker genes to establish the identities of clusters (Figures 2 and 3) (2017). These can now be used, looking cluster by cluster, to identify the clusters in the integrated analysis that contain significant cancer cell overlap.

```
img <- readPNG("~/SeuratProjectRMD/Bach2.png")
grid.raster(img)
```



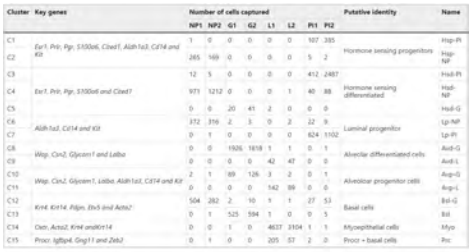
For our purposes, we are only interested in the identities of normal mouse cells from the Bach et al. study that contributed to clusters 1, 2, 4 and 7.

As mentioned, the identities of clusters in the integrated dataset were explored using the marker genes

provided by Bach et al. (2017). AddModuleScore wer not used with Featureplot to do this. Instead, individual genes were used with the FeaturePlot function, which may have helped with the contrast in many cases. Feature colors and the point size attribute were also changed and increased to make viewing features and cluster identification easier. As mentioned, cluster identities were determined only for clusters that showed significant coclustering (clusters 1, 2, 4 and 7). Analysis of marker gene expression was performed on each of these clusters, one at a time.

This is a table of the marker gens used by Bach et al. (2017).

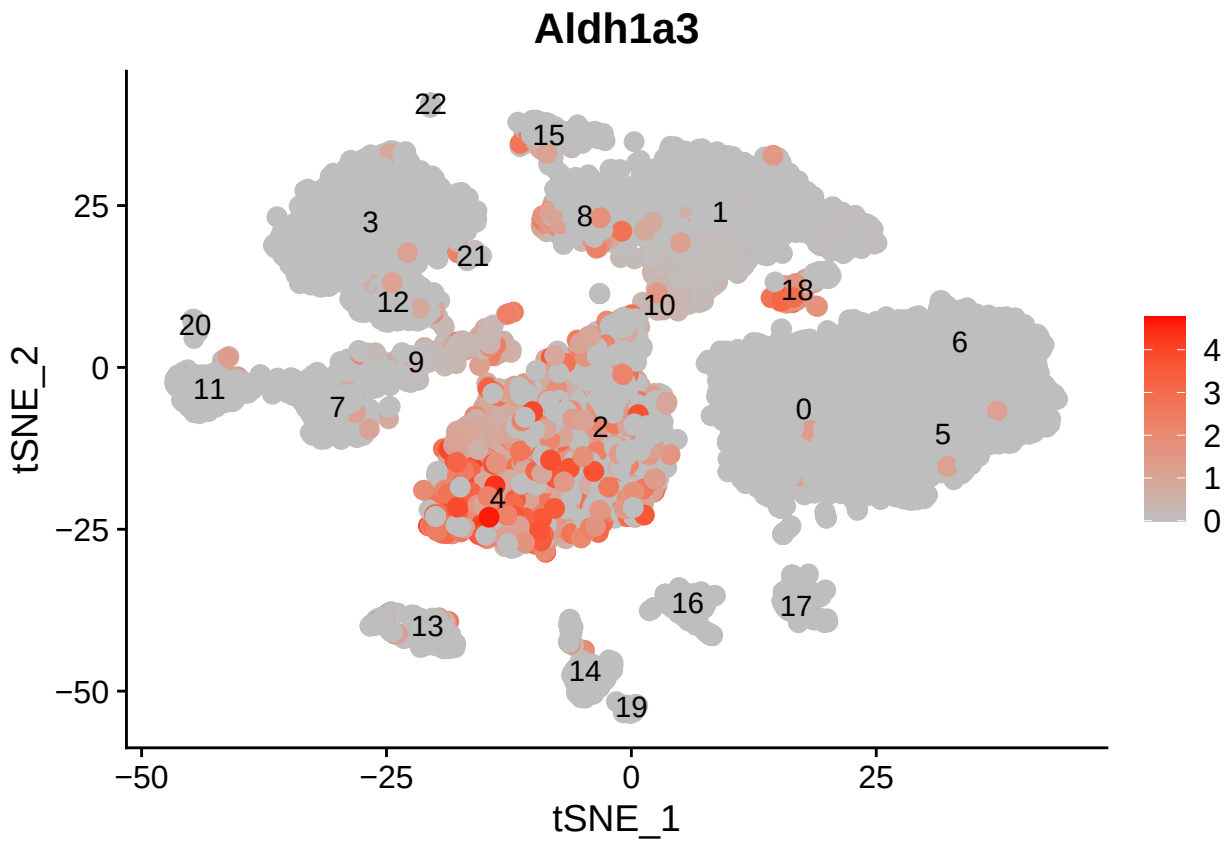
```
img <- readPNG("~/SeuratProjectRMD/Bach3.png")
grid.raster(img)
```

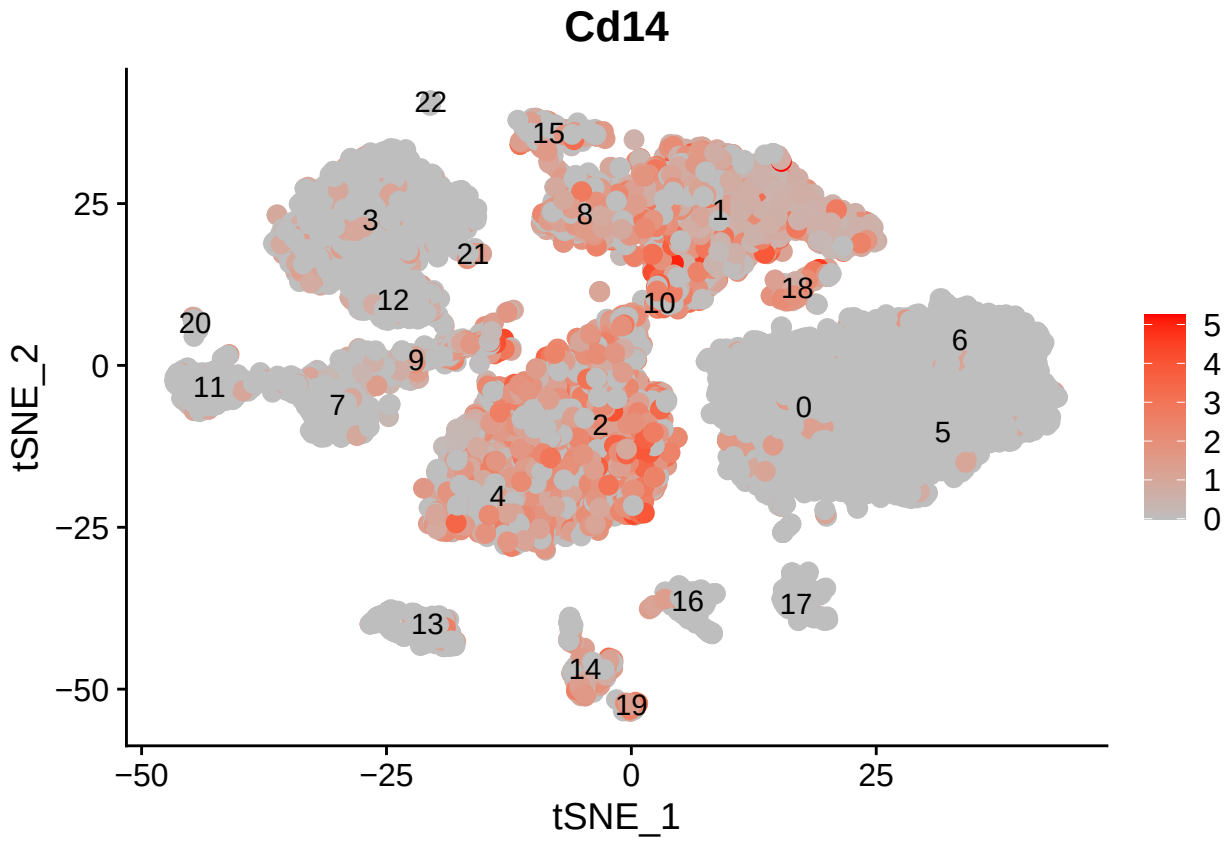


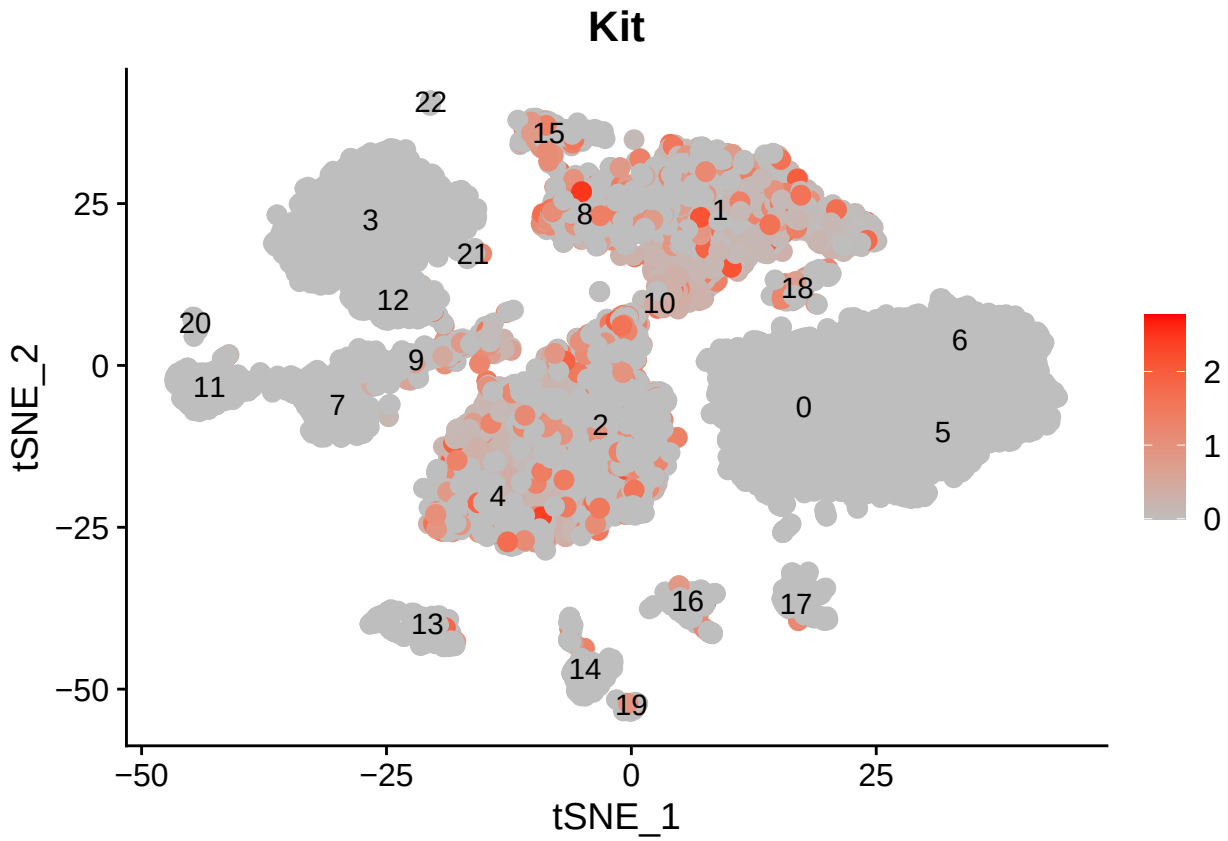
Cluster	Key genes	Number of cells captured							Putative identity	Name
		NP1	NP2	G1	G2	L1	L2	Pl1	Pl2		
C1	*Esr1, Prlr, Pgr, S100a6, Cited1, Aldh1a3, Cd14 and Kit*	1	0	0	0	0	0	107	385	Hormone sensing progenitors	Hsp-Pl
C2		265	169	0	0	0	0	5	2		Hsp-NP
C3		12	5	0	0	0	0	412	2487		Hsd-Pl
C4	*Esr1, Prlr, Pgr, S100a6 and Cited1*	971	1212	0	0	0	1	40	88	Hormone sensing differentiated	Hsd-NP
C5		0	0	20	41	2	0	0	0		Hsd-G
C6	*Aldh1a3, Cd14 and Kit*	372	316	2	3	0	2	22	9	Luminal progenitor	Lp-NP
C7		0	1	0	0	0	0	624	1102		Lp-Pl
C8	*Wap, Csn2, Glycam1 and Lalba*	0	0	1926	1618	1	1	0	1	Alveolar differentiated cells	Avd-G
C9		0	0	0	0	42	47	0	0		Avd-L
C10	*Wap, Csn2, Glycam1, Lalba, Aldh1a3, Cd14 and Kit*	2	1	89	126	3	2	0	1	Alveolear progenitor cells	Avp-G
C11		0	0	0	0	142	89	0	0		Avp-L
C12	*Krt4, Krt14, Pdpn, Etv5 and Actα7*	504	282	2	10	1	1	27	53	Basal cells	Bsl-G
C13		0	1	525	594	1	0	0	5		Bsl
C14	*Oxtr, Acta2, Krt4 and Krt14*	0	0	1	0	4637	3104	1	1	Myoepithelial cells	Myo
C15	*Procr, Igfbp4, Gng11 and Zeb2*	0	1	0	0	205	57	2	0	Procr + basal cells	Prc

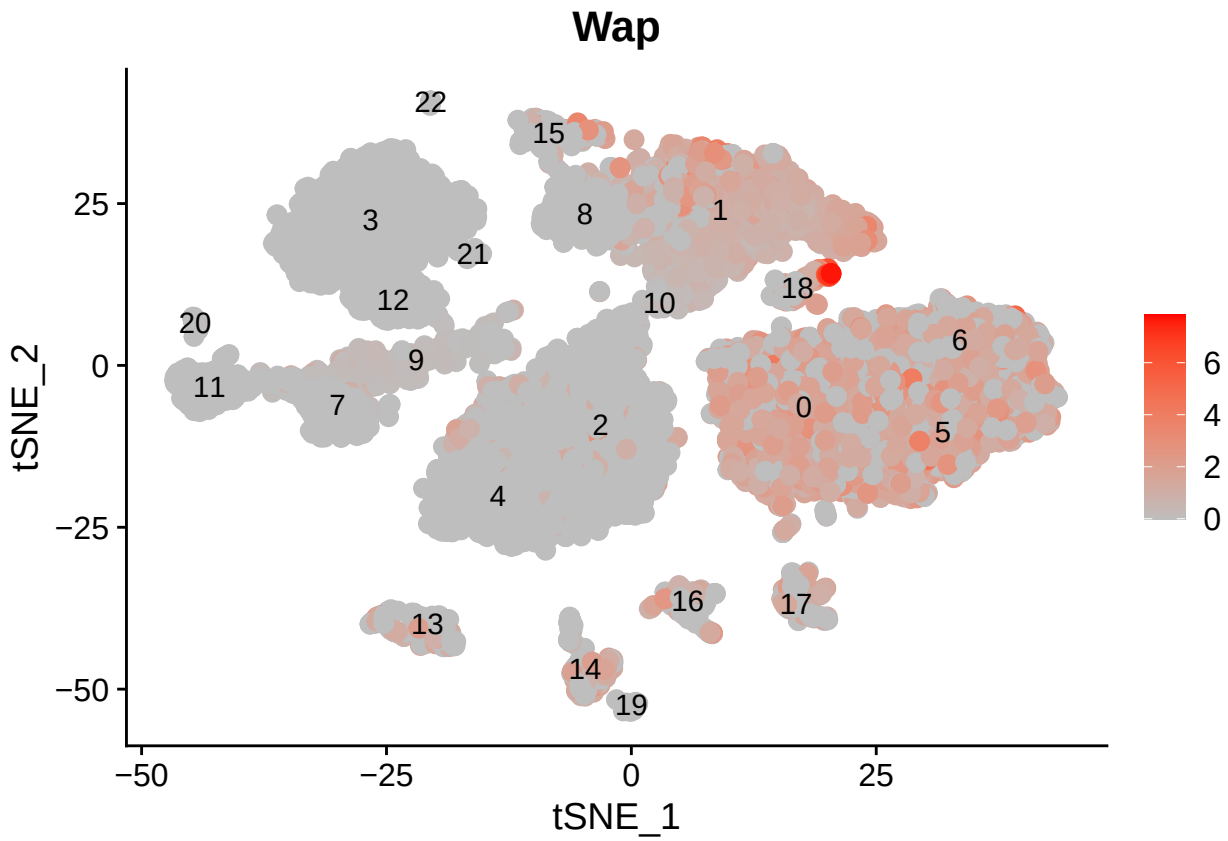
These marker genes were then visualized using the FeaturePlot function:

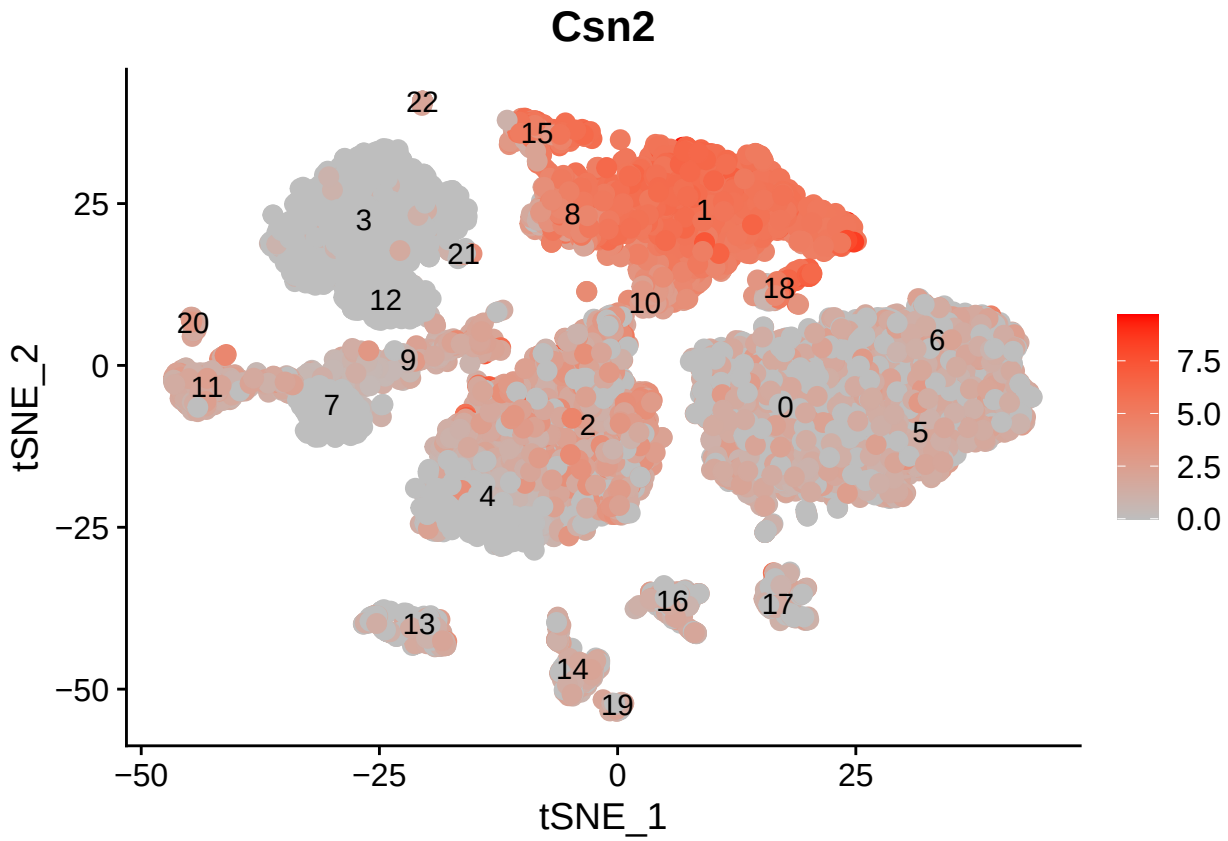
```
aa<-c("Aldh1a3", "Cd14", "Kit", "Wap", "Csn2", "Glycam1", "Lalba", "Krt4", "Krt14", "Pdpn", "Etv5", "Ac
length(aa)
bb<-unique(aa)
length(bb)

FeaturePlot(object = cancer.combined, features = bb, min.cutoff = 0, combine = FALSE, reduction = "tsne
```

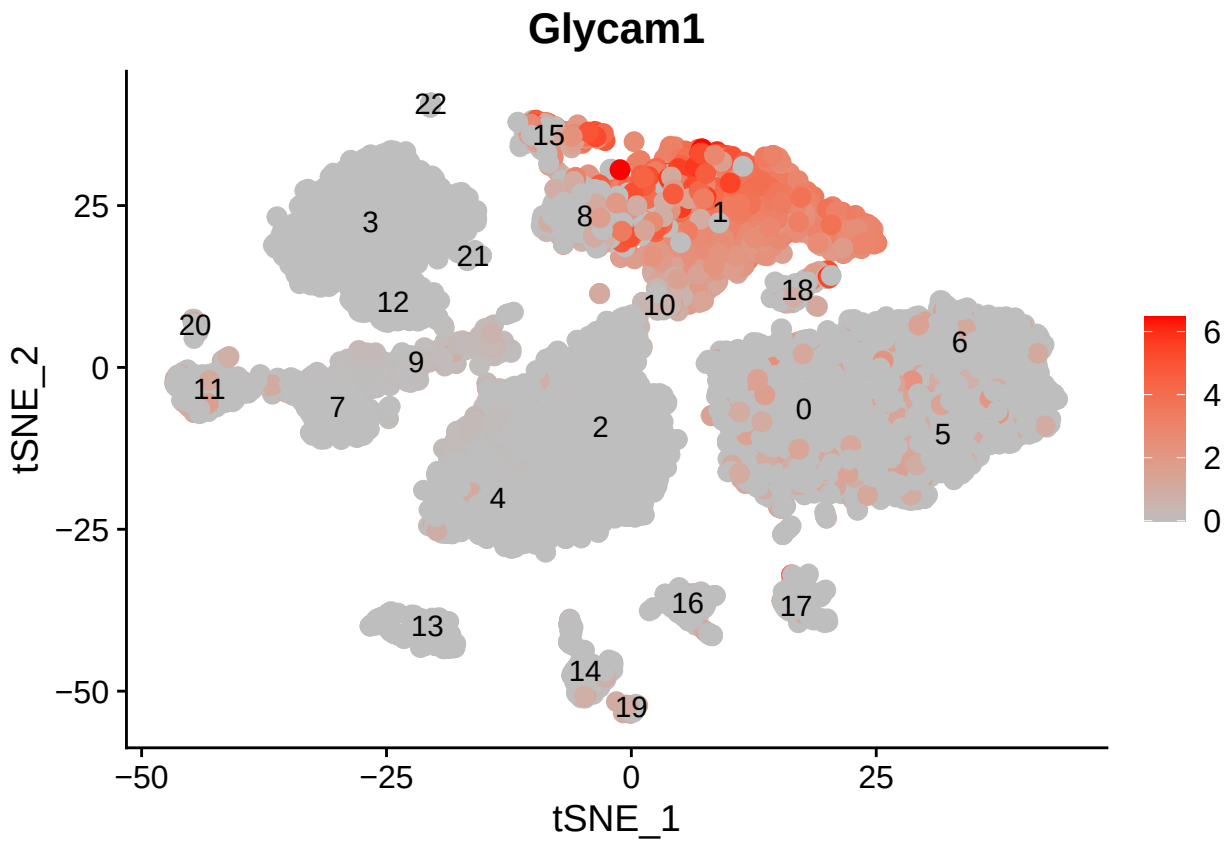

Aldh1a3
22
15
3
8
1
21
12
10
18
20
6
9
11
7
2
0
5
4
16
17
13
14
19
25
0
−25
−50
tSNE_2
−50
−25
0
25
tSNE_1
4
3
2
1
0

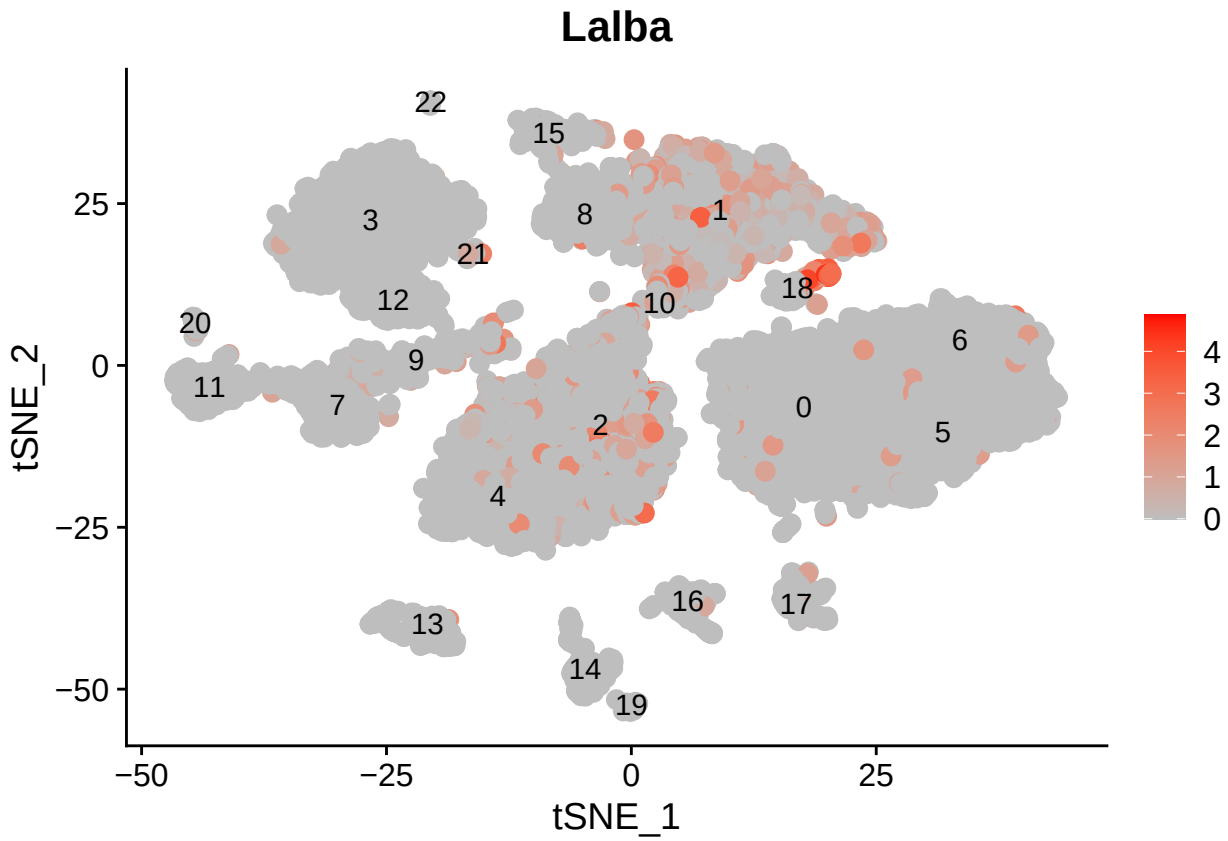

Cd14
22
15
25
3
8
1
21
18
12
10
20
6
9
0
11
7
5
2
4
−25
16
17
13
14
−50
19
−50
−25
0
25
tSNE_2
tSNE_1
5
4
3
2
1
0

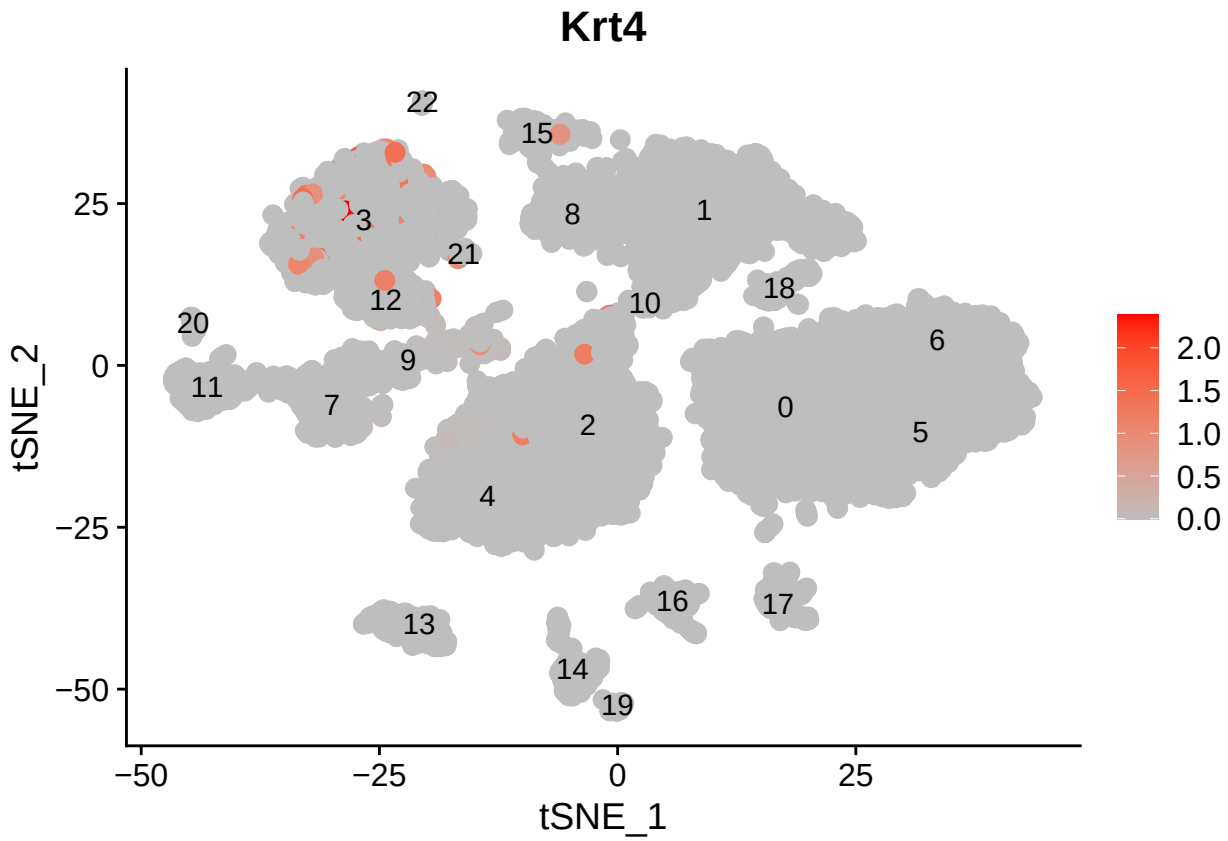

Kit
22
15
3
8
1
21
12
10
18
20
6
9
11
7
2
0
5
4
16
17
13
14
19
25
0
−25
−50
−50
−25
0
25
tSNE_2
tSNE_1
2
1
0

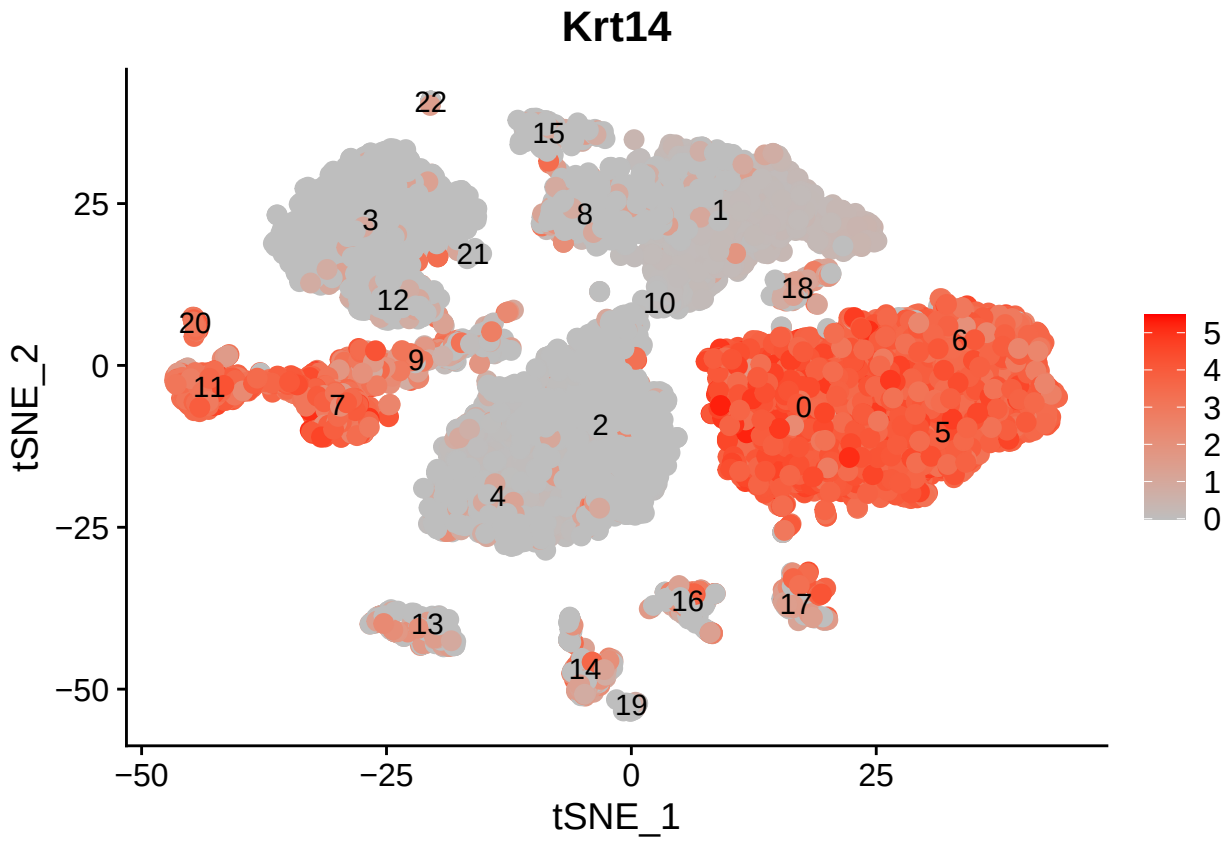

Wap
22
15
3
8
1
21
18
12
10
20
6
9
11
7
0
2
5
4
16
17
13
14
19
25
0
−25
−50
tSNE_2
−50
−25
0
25
tSNE_1
6
4
2
0

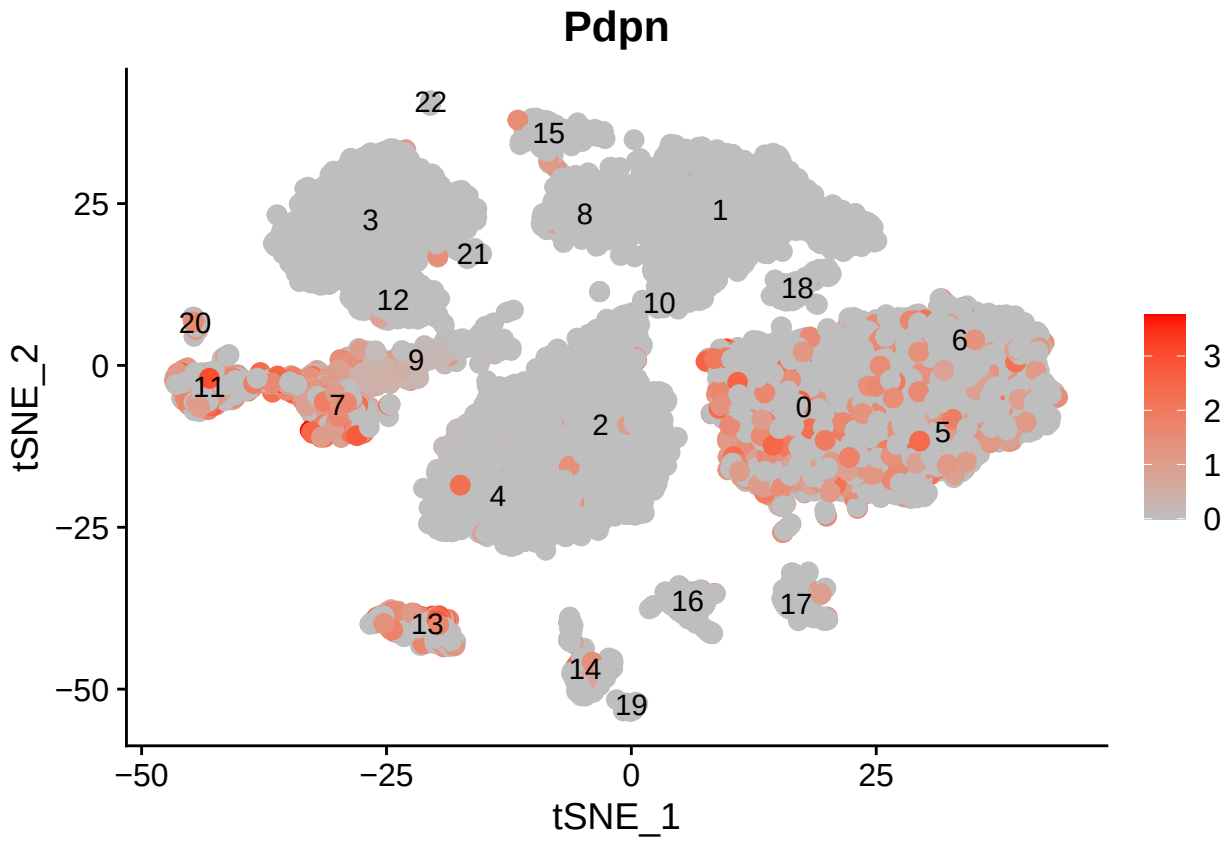

Csn2
22
15
25
3
8
1
21
12
10
18
20
6
0
9
11
7
2
5
4
−25
16
17
13
14
19
−50
−50
−25
0
25
tSNE_1
tSNE_2
7.5
5.0
2.5
0.0

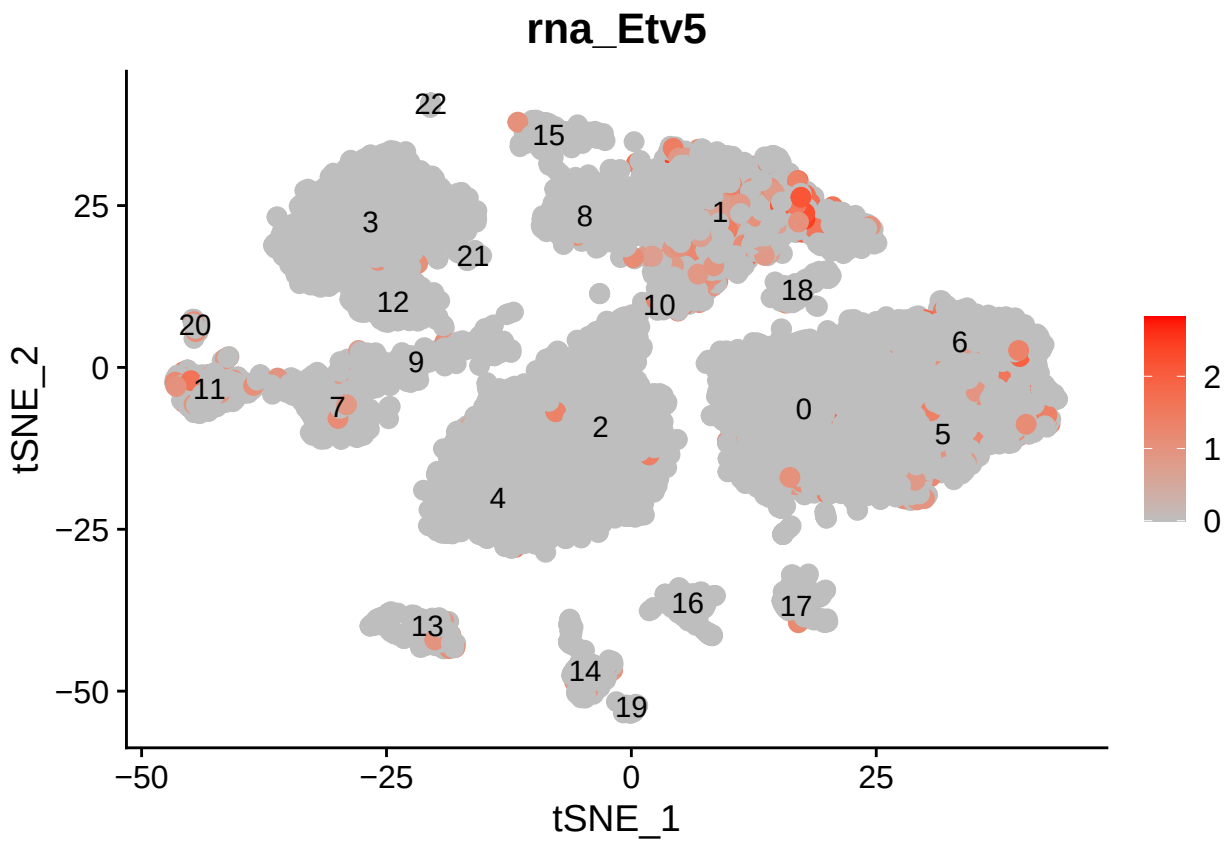

Glycam1
22
15
3
8
1
21
18
12
10
20
6
9
11
7
2
0
5
4
16
17
13
14
19
25
0
−25
−50
−50
−25
0
25
tSNE_2
tSNE_1
6
4
2
0

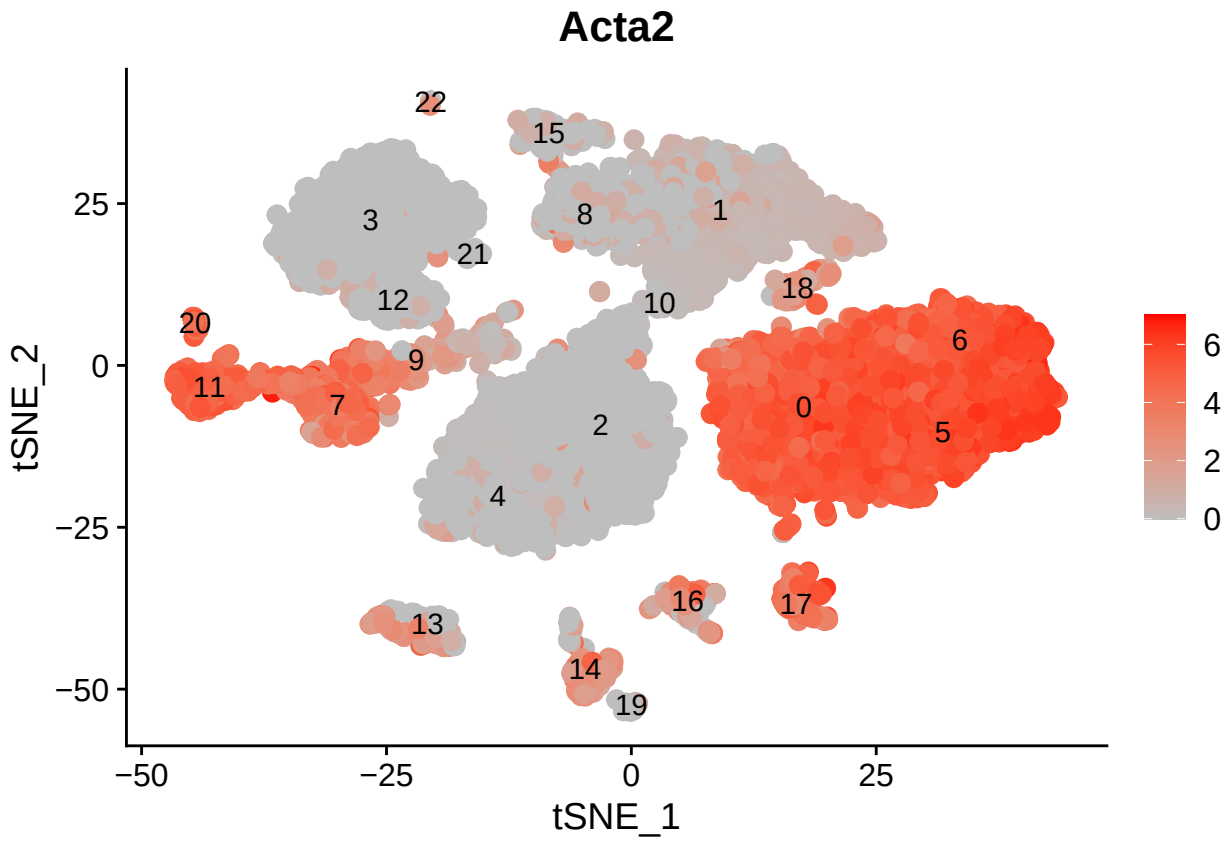

Lalba
22
15
3
8
1
21
18
12
10
20
6
9
11
7
2
0
5
4
16
17
13
14
19
25
0
−25
−50
tSNE_2
−50
−25
0
25
tSNE_1
4
3
2
1
0

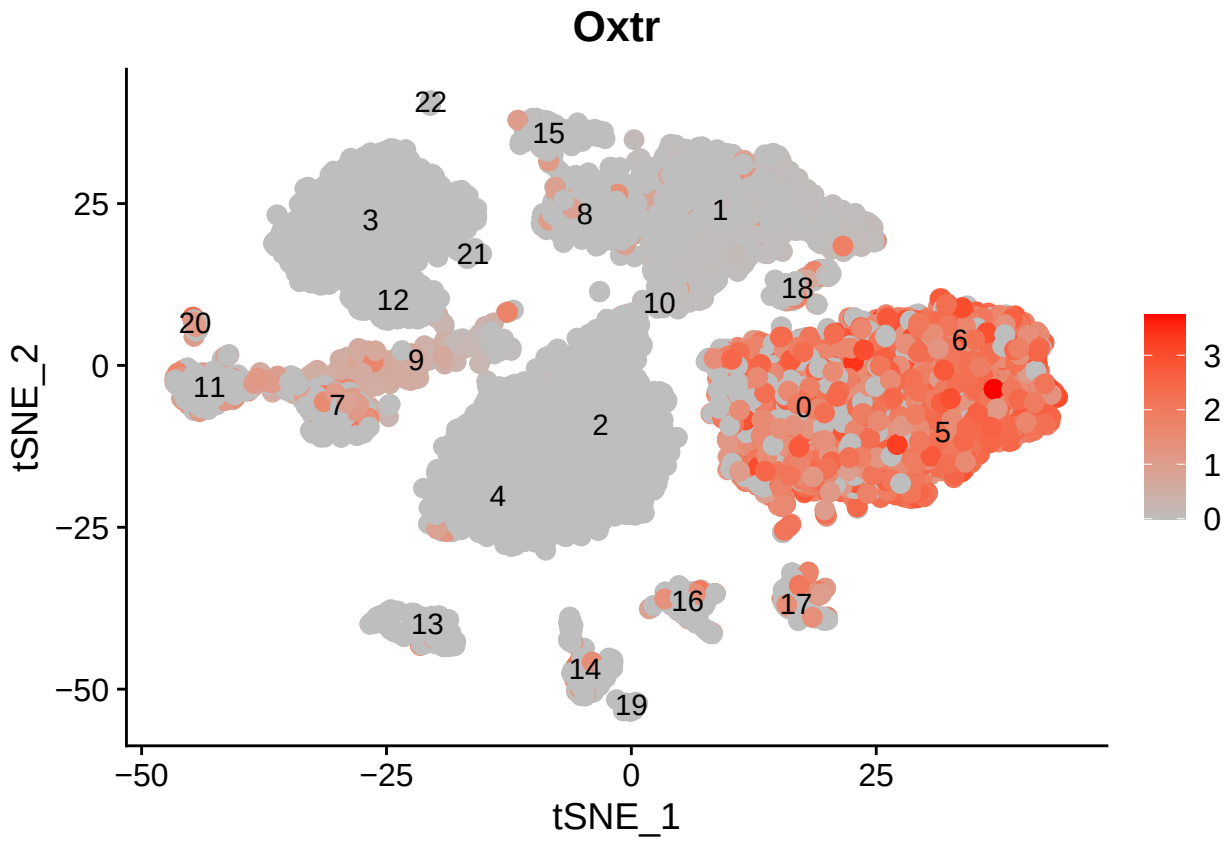

Krt4
22
15
25
3
8
1
21
12
10
18
20
6
0
9
11
7
2
0
5
4
16
17
13
14
19
−50
−25
0
25
tSNE_2
tSNE_1
2.0
1.5
1.0
0.5
0.0

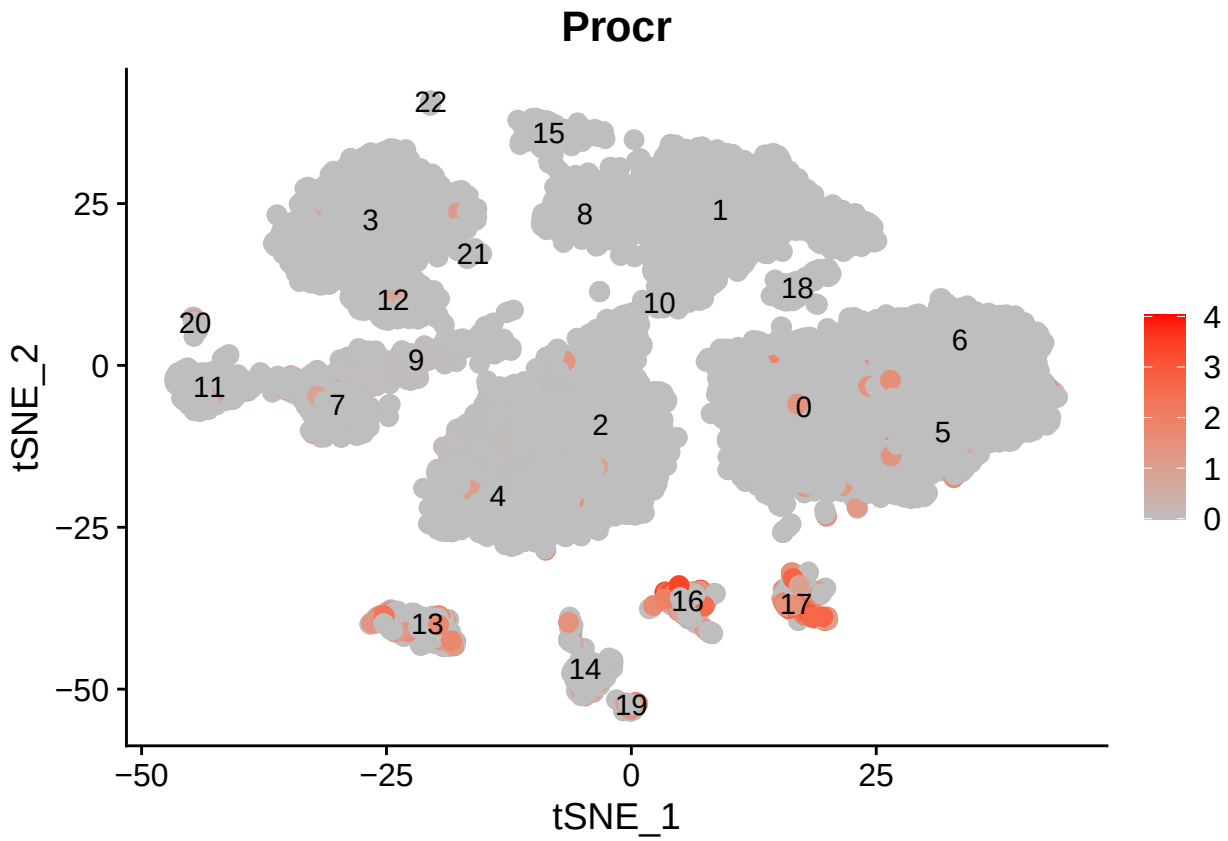

Krt14
22
15
25
3
8
1
21
18
12
10
20
6
9
0
11
7
2
5
4
16
17
13
14
19
−50
−25
0
25
tSNE_2
tSNE_1
5
4
3
2
1
0

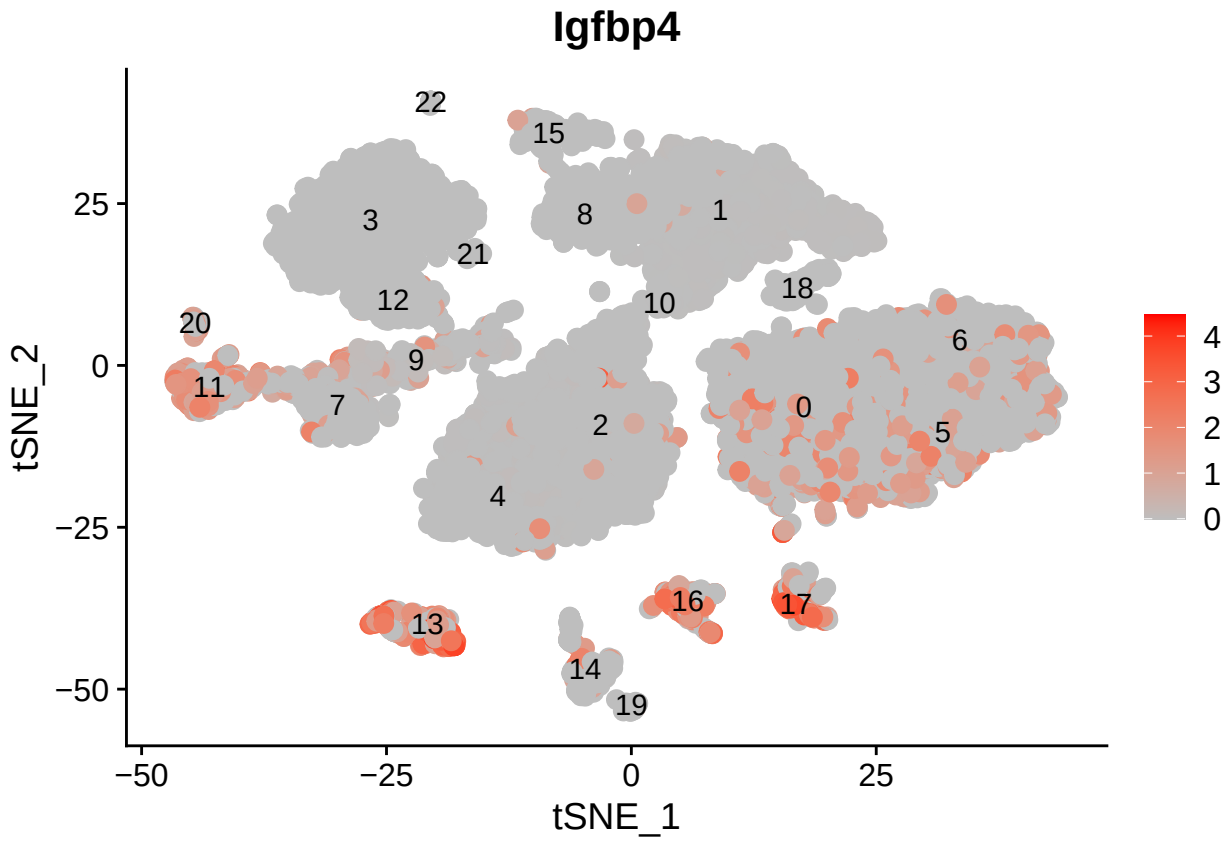

Pdpn
22
15
3
8
1
21
18
12
10
20
6
9
11
7
0
2
5
4
16
17
13
14
19
25
0
−25
−50
tSNE_2
−50
−25
0
25
tSNE_1
3
2
1
0

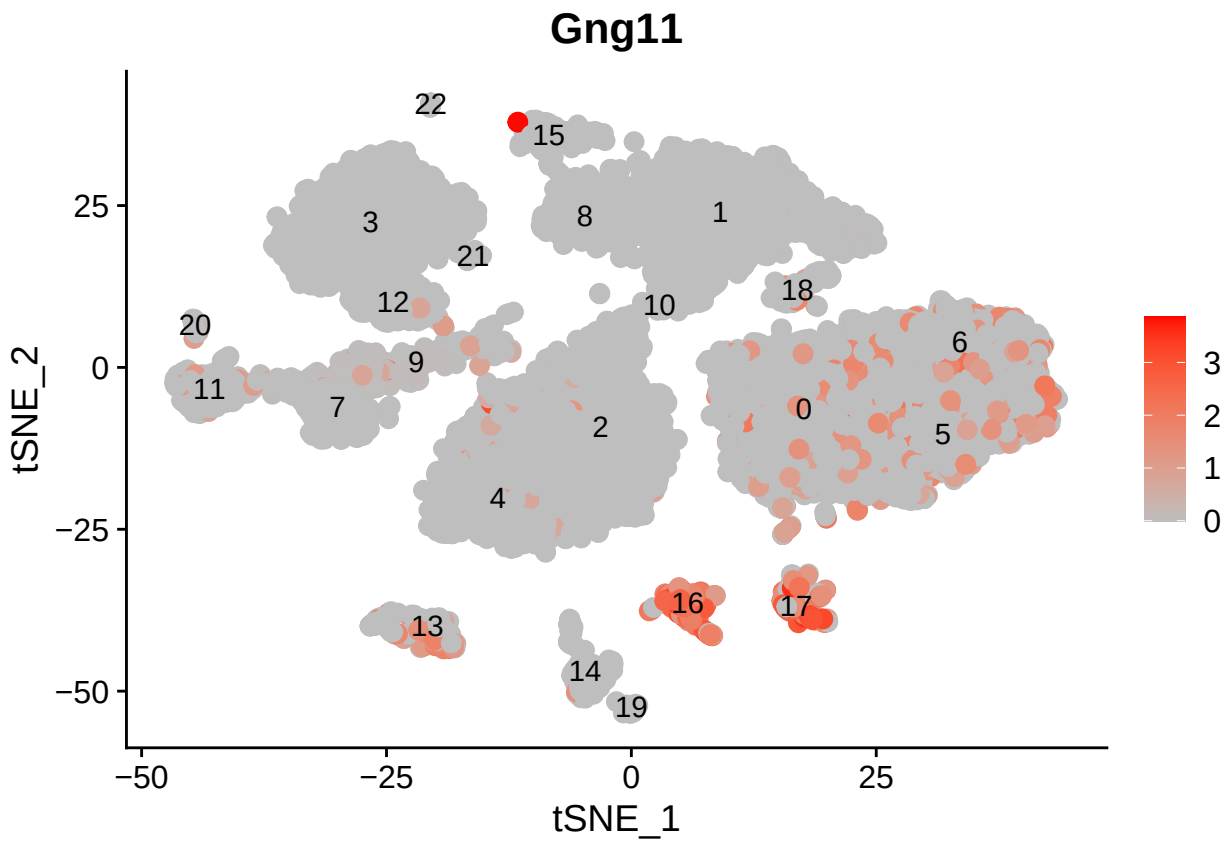

rna_Etv5
22
15
3
8
1
21
18
12
10
20
6
9
11
7
2
0
5
4
16
17
13
14
19
25
0
−25
−50
tSNE_2
−50
−25
0
25
tSNE_1
2
1
0

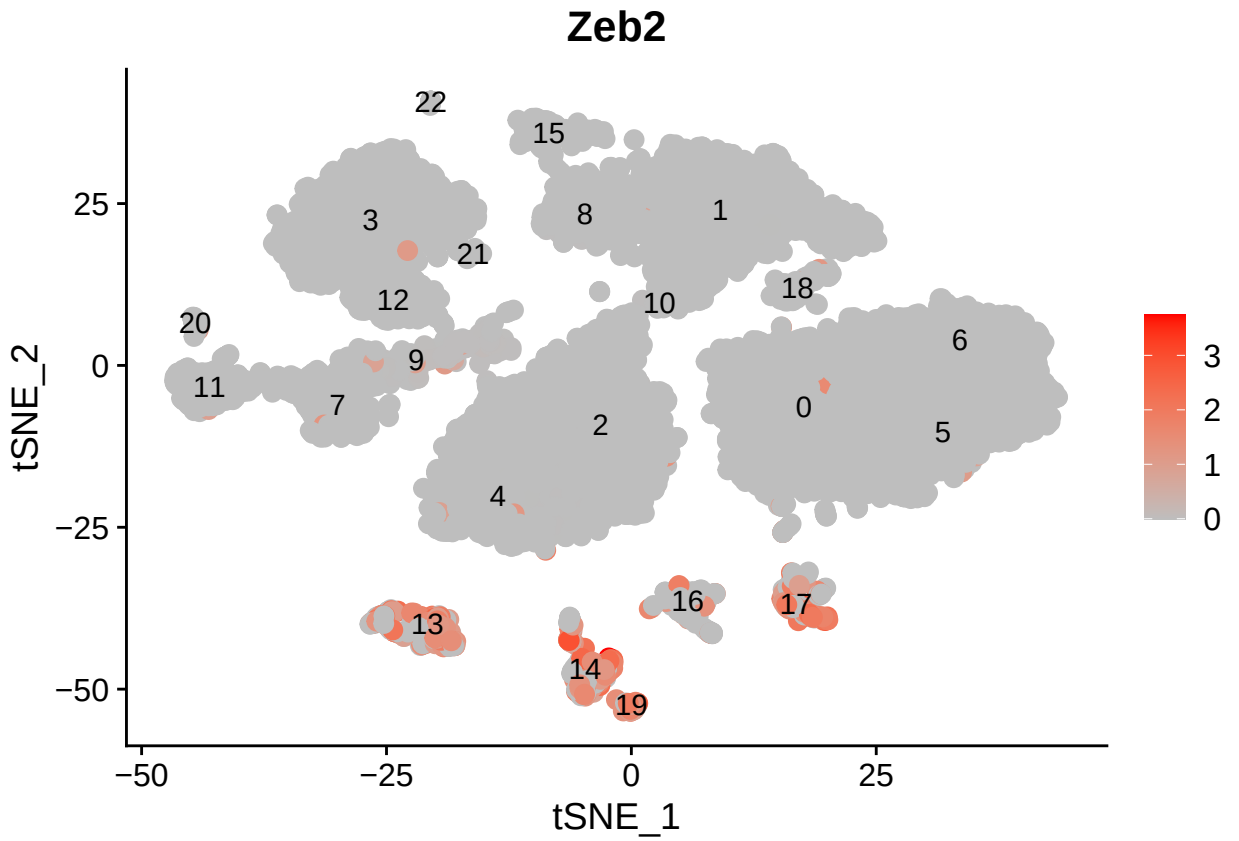

Acta2
22
15
3
8
1
21
18
12
10
20
6
9
11
7
0
2
5
4
16
17
13
14
19
25
0
−25
−50
tSNE_2
−50
−25
0
25
tSNE_1
6
4
2
0


Oxtr
22
15
3
8
1
21
18
12
10
20
6
9
11
7
0
2
5
4
16
17
13
14
19
25
0
−25
−50
−50
−25
0
25
tSNE_2
tSNE_1
3
2
1
0


Procr
22
15
3
8
1
21
18
12
10
20
6
9
11
7
0
2
5
4
16
17
13
14
19
25
0
−25
−50
−50
−25
0
25
tSNE_2
tSNE_1
4
3
2
1
0


Igfbp4
22
15
3
8
1
21
18
12
10
20
6
9
11
7
0
2
5
4
16
17
13
14
19
25
0
−25
−50
tSNE_2
−50
−25
0
25
tSNE_1
4
3
2
1
0


Gng11
22
15
3
8
1
21
18
12
10
20
6
9
11
7
0
2
5
4
16
17
13
14
19
25
0
−25
−50
−50
−25
0
25
tSNE_2
tSNE_1
3
2
1
0



Clusters 1 was identified as containing Neu cells. They are also composed of luminal cells (C1-C11, Krt18 positive). This cluster was identified as Cd14, Csn2, and Glycam1 positive (Kit, Wap, Lalba and Etv5 are questionable). These are most consistent with C10 and C11 from Bach et al. and are identified as alveolar progenitor cells. Yeo et al. describe the majority of Neu cells as co-clustering with C8 from Bach et al., which they thus identify as differentiated alveolar cells. I identify them as Kit and Cd14 positive, which differentiates these two groups (despite this cluster appearing to be Aldh1a3 negative).

Clusters 2 and 4 were identified as containing PyMT cells. They also both contain luminal cells (C1-C11 in Bach et al., Krt18 positive). Cluster 2 was identified as Aldh1a3, Cd14, and Csn2 positive (Kit and Lalba appeared questionable). This is consistent with C6 and C7 from Bach et al. and so are identified as luminal progenitor cells. Cluster 4 was identified as Aldh1a3 and Cd14 positive (Kit and Lalba were questionable, and Csn2 was negative). This is also consistent with C6 and C7 from Bach et al. These are also identified as luminal progenitors. Yeo et al. also identify their PyMT cells as co-clustering with luminal progenitor cells.

Cluster 7 are BRCA1-NULL cells. This cluster also contains basal cells (C12-C15, Krt5 positive). This cluster was identified as Krt14, Pdpn, Acta2 positive. The only clusters consistent with this are C12 and C13 from Bach et al. (who indicate that these cells are also Krt4 and Etv5 positive). These are identified as basal cells. Yeo et al. also found that their BRCA1-NuLL cells co-clustered to a large extent with basal cells

As Yeo et al. state (regarding their co-clustering results), "[t]aken together, our comparative analyses of various tumor cells superimposed with a recently described single-cell transcriptome dataset of normal MECs revealed the unique segments of the mammary differentiation spectrum occupied by each tumor type, with potentially distinct lineage-specific TPCs in each of the tumors examined" (2020). This describes these results as well.

Conclusion

As mentioned, use of supervised clustering on the first dataset may have led to a "blurring" of normal differentiation of cells into less "inevitable" (for a dataset, since the differentiation is relative to the surrounding cells) clusters, similar to as if one simply lowered the resolution using unsupervised clustering. This was actually a hypothesis on the part of the cancer cell researchers, that wasn't really tested, other than the generation of what seemed to be "inevitable differentiation" of the cells into three clusters associated with the three cell types sampled and pooled. Although this seems to be "good evidence" of their natural differentiability, unsupervised clustering, at a resolution close to optimal for the dataset, produced many additional clusters and sub clusters. This hypothesis was made, seemingly, based on the assumption of the close relatability of the cells to themselves, and possibly to certain developmental lineages, to which they were going to be compared. Embedded in this assumption was an anticipation that cancer cells, involving similar fundamental lineages, were not highly differentiable (a matter of the degree of cancer evolution among cells of certain types and basically unexplored).

This may have been a factor in both the initial clustering results and the clustering results involving the integrated dataset. The results for the integrated data from this effort and the cancer researchers efforts were highly similar. However, one cell type appeared to cocluster with a set of cells that were not identified in the Yeo et al. paper: namely, the Neu type cells, which, here, appeared to cocluster with alveolar progenitor cells, not differentiated alveolar cells. Because this effort used a subset of the cancer cells with which to perform the integration, it may be possible that another cluster of Neu type cells would in fact cocluster with differentiated alveolar cells. However, Yeo et al. do not report the coclustering of any Neu type cells with alveolar progenitor cells, and so this effort has produced results that were not a part of theirs.

Relation of the Neu type cells to alveolar progenitor cells may make sense from a standpoint of cancer dynamics. As Yeo et al. note, they were themselves looking for developmental processes (or cells) that may have been "hijacked" through an evolutionary cancer process. In this respect, one might expect cancer cells to cocluster (if they cocluster at all) first with a type of progenitor cell, before coclustering with a fully differentiated cell type. However, my experience identifying cell clusters on the basis of marker genes is not extensive, and, if there are any issues with this work, this is a likely source of error.

It is possible that, by imposing a restriction (ultimately, the choice of certain genes as "variable" genes for clustering) on the data in the form of supervised clustering could impose a bias on it with respect to more inevitable differentiation of cells. Use of different genes (instead of the most variable) in the choice of PCs and linear dimensionality reduction could have greatly affected the way cells were grouped, confounding downstream analysis. Yeo et al. did perform unsupervised clustering on the clustered data after this, but it produced the same tSNE plot, so it seems that it was based on this result, and produced only six clusters, not fourteen (like Seurat did). And so it appears that there was still a significant restriction of some kind on the second clustering (I do not know enough about this process to analyze this any further, but if I had more time I would look into this secondly).

In their defense, AddModuleScores of the genes provided by Yeo et al., when plotted using FeaturePlot in Seurat, did in fact appear to differentiate the cells into three large groups, mostly consistent clusters, and, in the case of two cells types, comprising two major clusters, like in the case of supervised clustering. However, the collections of genes used to produce AddModuleScores were composed of only three genes, not the kind of dimensionality you might want for differentiation of large groups of heterogeneous cells.

If their original supervised clustering did impose restrictions (with respect to the more inevitable groupings of unsupervised clustering), and thus introduced bias in the downstream unsupervised clustering (results of which were not apparently really used during the analysis of the integrated data anyway), then one could imagine that these restrictions (which are basically like a "blurring" of groups) might "hold" cells together that should not be, and the effects "averaged" over many more distinct cell "types" could push the clusters into a kind of different "averaged" cell space. I did not inspect the authors code (I have just run out of time), but they did use Seurat to do this integration, and so it is likely that this was done using raw data, and was totally unsupervised (erasing previous clustering results). However, if it did not, this is one possible explanation for the differing results, and the third thing I would look into. Yeo et al. produced 17 clusters from the integrated dataset, three less than the original clustering by Bach et al., and with twice

as many cells. Here 22 clusters were produced, suggesting that Yeo et al. either used a lower resolution during the integrated clustering, or there was some other restriction being imposed (unless Yeo et al. tried to reproduce clustering of the Bach et al. dataset first, subsetted it for the 15 identified clusters - removing the 5 unidentified clusters - and then created a new raw matrix of counts with which to perform the integration step). This is something else I would look into given more time.

Afterword

Statisticians may have imagined, long before, the possibility of dealing with large, coherent, and complex datasets (in part, today, made highly available through computer technology and the internet) that could not be "classified" by normal standards (because of a lack of classification "background" and/or because of differences in the "resolution" of the picture of the objects being classified and/or because of lack of understanding of the mechanisms that lead to classification). However, today, it seems the opportunity to use statistical clustering tools on large datasets is a commonplace occurrence in many fields, including business, government and the natural sciences (https://datafloq.com/read/7-innovative-uses-of-clustering-algorithms/).

One example from biology(aside from scRNA-seq) is the use of OTU's in Microbiology. These are used after metabarcoding when a complex sample of perhaps related organisms is being characterized without taxonomic knowledge of all potential contributors. In this case, they are "classified" (or clustered, to be more exact) according to their relatedness (usually 97% operationally defining a "species").

"Clustering" is thus just a proxy for more exact "classification", which is often preferable (https://neptune.ai/blog/clustering-algorithms). However, in the absence of classification resources, collections of objects can be grouped together based on some form of similarity, and this is the essence of clustering. Classification and clustering are based on similar principles (grouping based on some form of similarity and/or dissimilarity), however, clustering, a process that is more purely mathematical and based on what may be - from a mechanistic perspective involving how forms for classification arise in the first place - an arbitrary dimension of similarity and difference, is not always interested in the mechanisms that give rise to the differences it measures, and for this reason may be more indirect in its assessments. For instance, knowing that number of bedrooms is the most influential parameter determining the "current model in predicting house prices" might go a long way to classifying houses on the basis of price (in the absence of knowledge of the hopes, dreams, tastes, experiences, etc, of the potential buyer, as well as those of the house builder), but it is certainly not "fundamental" to "house price" (it doesn't "produce" it directly, mechanistically) (https://towardsdatascience.com/think-twice-before-you-use-principal-component-analysis-in-supervised-learning-tasks-70fbb68ebd0c). In other words, when we look at bedrooms in houses, we are not looking at a fundamental level of "house price space".

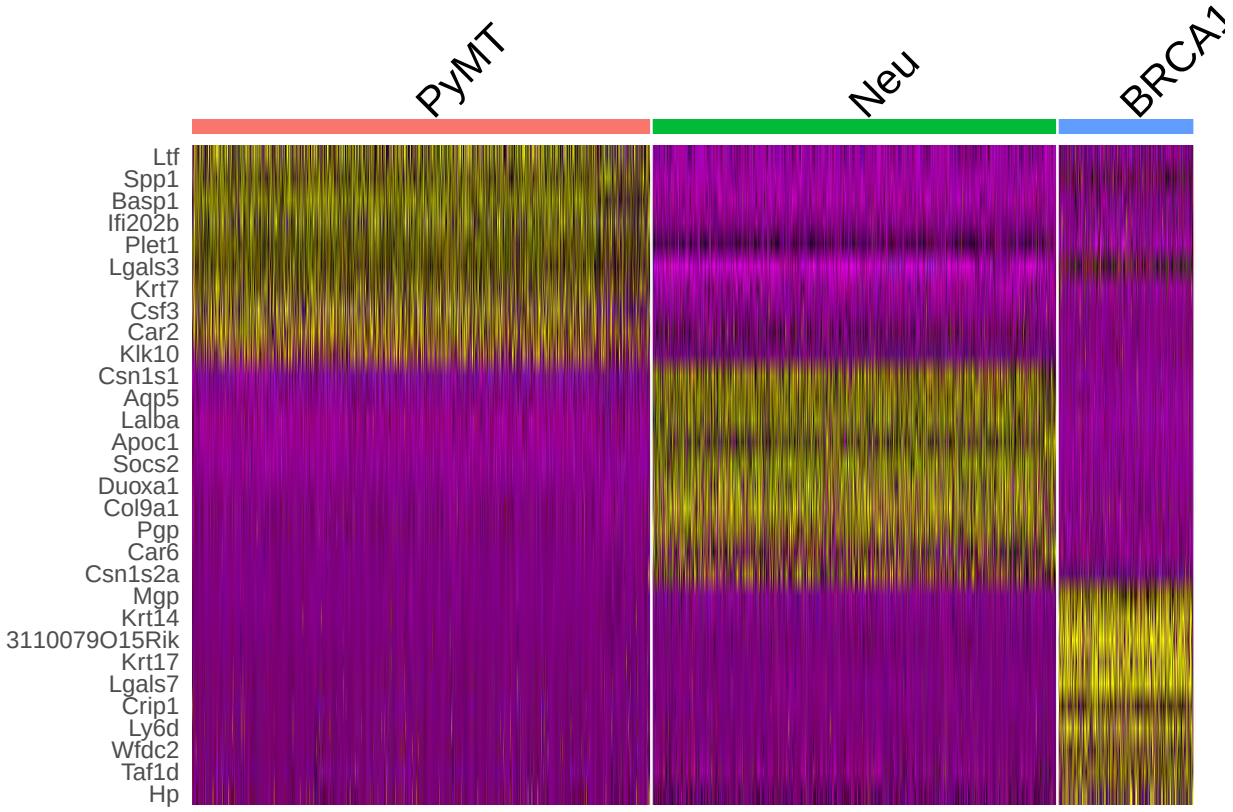
This is not the case with some information in biology today. When we look at a transcriptome for a cell, we ARE looking at the fundamental level (or very close to it) of "cell type space". Related to this is this reason that OTU's can be called "operational species": the clustering is being done with a fundamental piece of knowledge about the organism (the most commonly used piece of knowledge guiding taxonomy), its inherited molecular composition and relatedness to other organisms.

This is one reason why unsupervised clustering of transcriptomes is so powerful. It can be used to determine the basis of classification systems themselves. And so, when one is using clustering to "chartacterize" cells using scRNA-seq (https://jfouyang.github.io/singlecell/clust.html), they are looking at the most fundamental basis of discerning cell types at the same time (they are looking at one piece of fundamental "cell space"). In other words, "clustering" and "classification" begin to converge in biology, and this is a function of the "fundamentalness" (in the case of scRNA-seq) of the transcriptome to cells (or molecular information in general, such as in the case of microorganism clustering): it wholly (along with the proteins it produces and the DNA that produces it) determines the qualities of a cell. And so, there is no better way to classify cells, and in this sense, clustering almost becomes synonymous with improved classification, not an alternative to it.

While yeo et al. were exploring the relationships of the cancer cells to a normal mammary tissue and a developmental hierarchy, in addition to characterizing patterns of heterogeneity, because they were using scRNA-seq, they, at the same time produced a catalog of differentially expressed features for the cells they observed. While they used supervised clustering (with prechosen genes) to do this (a choice that may have confounded these particular results), this effort used unsupervised clustering, and so may have differentiated the cells more inevitably, thus producing a better picture of "cell types" within the sample analyzed.

The most differentially expressed genes (with respect to other cells in the sample, which necessarily partly determines how variable genes for a given cluster are defined) are what is used to "characterize" clusters using unsupervised clustering with scRNA-seq. These results can be generated using Seurat by executing the "FindAllMArkers" function on the subsetted Seurat Object for the first dataset, and then can be visualized using the "DoHeatmap" function.

```
S_02.markers <- FindAllMarkers(S_02, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)

S_02.markers %>%
  group_by(cluster) %>%
  top_n(n = 10, wt = avg_log2FC) -> top10
head(top10)
DoHeatmap(S_02, features = top10$gene) + NoLegend()
```



A list of highly variable genes can be extracted in their entirety from the Seurat Object (and put into a table, for example) by just examining the "S_O2.markers". These lists of genes can be used to classify cells by cell type. They can also be subjected to gene ontology/gene enrichment analysis, to get an idea, not just of the genes that characterize them, but the systems and biological processes in which these gene participate. This can be done to get an understanding of the cells on a functional level: to understand what the "cell type" actually does (its evolutionary "reason" for existing), which is what really compels us to

classify cells in the first place.

References

Bach et al. Differentiation dynamics of mammary epithelial cells revealed by single-cell RNA sequencing. Nature Communications. 2017.

Clustree:

Zappia L, Oshlack A (2018). "Clustering trees: a visualization for evaluating clusterings at multiple resolutions." GigaScience, 7(7). doi:10.1093/gigascience/giy083, https://doi.org/10.1093/gigascience/giy083.

Clustree Vignette.https://cran.r-project.org/web/packages/clustree/vignettes/clustree.html

Datafloq. https://datafloq.com/read/7-innovative-uses-of-clustering-algorithms/

Neptune. https://neptune.ai/blog/clustering-algorithms

R and R Studio:

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

RStudio Team (2020). RStudio: Integrated Development for R. RStudio, PBC, Boston, MA URL http://www.rstudio.com/

Satija Lab:

https://satijalab.org/seurat/articles/pbmc3k_tutorial.html

https://satijalab.org/seurat/articles/integration_introduction.html

Seurat:

Hao Y, Hao S, Andersen-Nissen E, III WMM, Zheng S, Butler A, Lee MJ, Wilk AJ, Darby C, Zagar M, Hoffman P, Stoeckius M, Papalexi E, Mimitou EP, Jain J, Srivastava A, Stuart T, Fleming LB, Yeung B, Rogers AJ, McElrath JM, Blish CA, Gottardo R, Smibert P, Satija R (2021). "Integrated analysis of multimodal single-cell data." Cell. doi:10.1016/j.cell.2021.04.048, https://doi.org/10.1016/j.cell.2021.04.048.

Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, III WMM, Hao Y, Stoeckius M, Smibert P, Satija R (2019). "Comprehensive Integration of Single-Cell Data." Cell, 177, 1888-1902. doi:10.1016/j.cell.2019.05.031, https://doi.org/10.1016/j.cell.2019.05.031.

Butler A, Hoffman P, Smibert P, Papalexi E, Satija R (2018). "Integrating single-cell transcriptomic data across different conditions, technologies, and species." Nature Biotechnology, 36, 411-420. doi:10.1038/nbt.4096, https://doi.org/10.1038/nbt.4096.

Satija R, Farrell JA, Gennert D, Schier AF, Regev A (2015). "Spatial reconstruction of single-cell gene expression data." Nature Biotechnology, 33, 495-502. doi:10.1038/nbt.3192, https://doi.org/10.1038/nbt.3192.

Towards Data Science.https://towardsdatascience.com/think-twice-before-you-use-principal-component-analysis-in-supervised-learning-tasks-70fbb68ebd0c

Yeo et al. Single-cell RNA-sequencing reveals distinct patterns of cell state heterogeneity in mouse models of breast cancer. Elife. 2020.