Single Cell RNA Sequence Analysis Pipeline Guide

Craig McDougall

October 2020

Introduction

This document details the stages of a bioinformatic pipeline, designed in R to analyse single cell RNA sequence data. The pipeline was constructed as part of an MSc project at the University of Edinburgh in Scotland.

The aim of the project was to investigate sexual dimorphism in the central stress response system; the Hypothalmic-pituitary-adrenal (HPA) axis. The pipeline was designed as a tool, to probe the transcriptome profiles of anterior pituitary cells. In the project it facilitated a study of sexual dimorphism in the gene expression of pituitary corticotroph cells, but could also be used on other cell types.

A similar method has been reported in a recent study of sexual dimorphism in pituitary cells. However, this study concentrated on gonadotroph and lactotroph pituitary cells [Fletcher et.al 2019]. An underlying limitation of this kind of analysis is that it relies on inferring protein levels by measuring transcription levels, it is not a direct measurement of the proteins present.

The pipeline files can be found on GitHub.

The Seurat R. package forms the foundation of this pipeline and is in particular based on the guided clustering tutorial, but updated to include the `SCTransform` normalisation function.

Example single cell RNA sequence data is available in public repositories. In the project the pipeline was designed for, the following data sets were examined:

- Chung et al.
- Mayran et al.
- Ho et al.
- Fletcher et al.

Details on the data acquisition of these datasets is described within the following publications:

- Chung et al.
- Mayran et al.
- Ho et al.
- Fletcher et al.

The pipeline may be broken down into the following steps.

Precursor: Setting directories

As a precursor to the main body of the pipeline script, the data directories for the location of input data and output data storage locations are set and verified;

```
#set the working directory to directory wish to save analysis results to
setwd("/Users/craigmcdougall/Documents/Project 2/Data analysis/Cheung")

#check working directory
getwd()
```

```
## [1] "/Users/craigmcdougall/Documents/Project 2/Data analysis/Cheung"
```

```
#define the data directory where the data is held
data.dir <- "/Users/craigmcdougall/Documents/Project 2/Data/Cheung"

#list all the files in the data directory: barcodes, genes, matrix
list.files(data.dir)
```

```
## [1] "barcodes.tsv" "genes.tsv"    "matrix.mtx"   "Raw"
```

The correct path must be specified in order for the script to run correctly, hence the sanity checks to ensure the curreent working directory is the intended one.

With the initial setup verified the script moves onto the next step in the pipeline.

Step 1. Call the relevant Libraries

Before the main boy of the pipeline is run, all of the required libraries are called;

```
library(dplyr)
library(Seurat)
library(patchwork)
library(sctransform)
library(ggplot2)
library(scales)
```

Step 2. Loading data and initialising Seurat object.

The raw count and gene data is read in from the 10X sequence data files downloaded from the public repositories. Seurat expects "barcodes.tsv", "matrix.tsv" and "matrix.tx" files. These asre then used to initialise a Seurat object for the transcriptome analysis.

```
#load raw data using Read10x
raw.data <- Read10X(data.dir=data.dir)

#initialise a Seurat object with the raw (not normalised) data. This is a count matrix
data <- CreateSeuratObject(counts=raw.data, project = "data", min.cells = 3, min.features = 200)
```

Step 3. Quality control plots

Before processing the data, some quality control plots are made to assess the quality of the data and explore confounding factors such as assessing the number of reads that map to mitochondrial genome.

```
#Seurat recommends percent.MT label for mitochondria data, however;
#Cheung, Ho & Mayran et.al. use "percent.mt"
data <- PercentageFeatureSet(data, pattern = "^mt-", col.name = "percent.mt")
#Fletcher uses "percent.Mt"
#data <- PercentageFeatureSet(data, pattern = "^mt-", col.name = "percent.MT")
#select appropriate method and comment out other option

#use QC metrics to filter the cells
#visualise the QC metrics - use a violin plot
VlnPlot(data, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

#make the plots
plot1 <- FeatureScatter(data, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 <- FeatureScatter(data, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
#then plot
plot1 + plot2
```

Step 4. Transform the data

With the data visualised, it must be;

- Filtered to remove damaged cells and double counts.
- Normalised to ensure all genes carry equal weight going forward and prevent heavily expressing genes from being over represented.
- Regressed to remove confounding factors such as areas mapped to the mitochondrial genome.
- Assessed for variable genes.

The SCTransform function performs all of these actions and greatly simplifies the code, negating the use of four separate functions.

```
data <- SCTransform(data,vars.to.regress = "percent.mt")#, verbose = FALSE )
```

Step 5. Dimensional reduction

The first step in dimensional reduction is to conduct principal component analysis (PCA) on the scaled data, using the variable features as the input. PCA transforms the from the table into new features known as principal components, which capture the information of the dataset in a new way.

An elbow plot can be used to assess which principal components contain the majority of the information and identify which may be removed from the dataset;

Uniform Manifold Approximation and Projection (UMAP) is an algorithm for dimensional reduction and is applied on the principal components established in the first step.

```
#PCA to identify PCs
#run PCA following new Seurat standard steps:https://satijalab.org/seurat/v3.1/sctransform_vignette.htm
data <- RunPCA(data,verbose= FALSE)

#Do we need to keep all PCs, or is majority of data captured by a certain point?
ElbowPlot(data, ndims=30)

#UMAP dimensional reduction of the identified PCs
data <- RunUMAP(data, dims=1:30, umap.method = "umap-learn", metric = "correlation", verbose=FALSE)
```

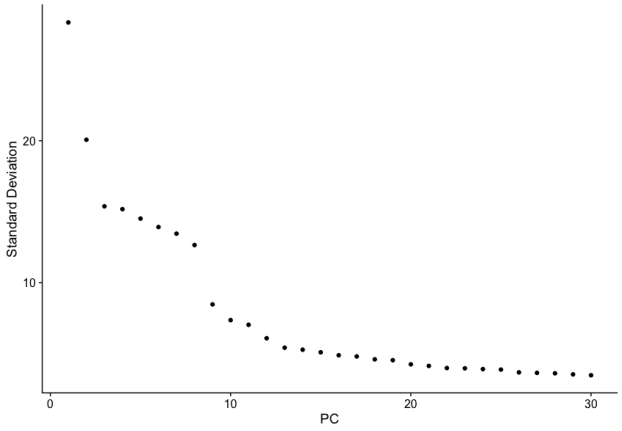

Figure 1: An elbow plot can identify which principal components must be retained to capture most of the information

Step 6. Clustering

The first stage in clustering the cells is to establish a shared nearest neighbour (SNN) information for the data by calculating the overlap between each cell based on its k.parameters using a Seurat function.

The clusters cn then be identified based on the SNN using a clustering algorithm.

```
#Find nearest neighbours
data <- FindNeighbors(data, dims=1:30, verbose=FALSE)

#CLuster based on neighbour distances
data <- FindClusters(data, verbose=FALSE)

#plot the clusters
DimPlot(data, label=TRUE) + NoLegend()
```

Once the clusters have been established, they can be projected into two lower dimensional space using a dimensional reduction plot for visualisation.

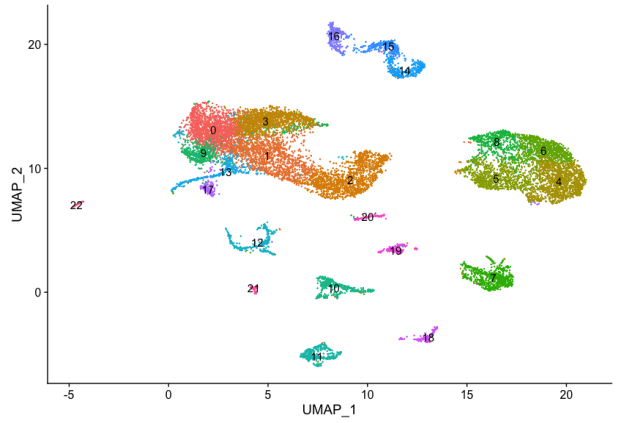

Figure 2: Dimensional reuction plot allows visualisation of the clusters found found in the UMAP data

Step 7. Cell Type assessment

The main question now, is how well do the clusters approximate cell type? It is difficult to make absolute claims when attributing cell types to empirical cluster data based purely on an *in silico* analysis. However we may approximate the cell type of each cluster based on known canonical expression markers [Fletcher et.al 2019] [Chung et.al 2018].

In the corticotroph study, hormone secreting pituitary cell types were identified based on the following canonical gene expression markers:

- Thyrotrophs (Pou1f1 + Tshb),
- Somatotrophs(Pou1f1 + Gh)
- Lactotrophs (Pou1f1 + Prl)
- Gonadotrophs (Lhb)
- Melanotrophs (Pomc + Pcsk2 + Pax7)
- Corticotrophs (Pomc + Crhr1 + Avpr1b + Gpc5 - Pcsk2 - Pax7)

Feature plots were used to visually assess gene expression levels of single cells.

```
#Visualise the canonical marker genes

#first exclude Pou1f1 expressing cells (Thyrotrophs, Somatotrophs, Lactotrophs)
FeaturePlot(data, features = c("Pou1f1","Tshb", "Gh","Prl"), pt.size = 0.2)&scale_color_viridis_c()

#then exclude the gonadotrophs
FeaturePlot(data, features = c("Lhb"), pt.size = 0.2)&scale_color_viridis_c()

#then exclude the melanotrophs
FeaturePlot(data, features = c("Pomc","Pax7","Pcsk2"), pt.size = 0.2)&scale_color_viridis_c()

#then id the corticotrophs
FeaturePlot(data, features = c("Pomc", "Crhr1","Avpr1b","Gpc5"), pt.size = 0.2)&scale_color_viridis_c()
```

The first plot would be used to exclude clusters of thyrotrophs, Somatotrophs and lactotrophs. The second plot allows exclusion on gonadotrophs, the third melanotrophs. The final plot allows confirmation of the corticotoph cluster.

Step 8. Isolate desired cell cluster

To examine cell homogeneity within the corticotrophs, the cluster identified in the previous step was pulled out as a subset.

The desired cluster must be manually entered upon visual inspection of the gene expression feature plots in the previous step.

```
#enter the cotricoph cell ids, found in visualisation. Note change to correct cluster #s or will miss-m
#new.cluster.ids <- c("0", "1", "2", "3", "4", "5", "6","7","8","9","10","11","12","13","14","15","16",
new.cluster.ids <- c("0", "1", "2", "3", "4", "5", "6","7","8","9","10","11","12","13","cort","15","16"

names(new.cluster.ids) <- levels(data)

data <- RenameIdents(data, new.cluster.ids)
#then put on the plot to confirm slected the correct one (sanity check)
```

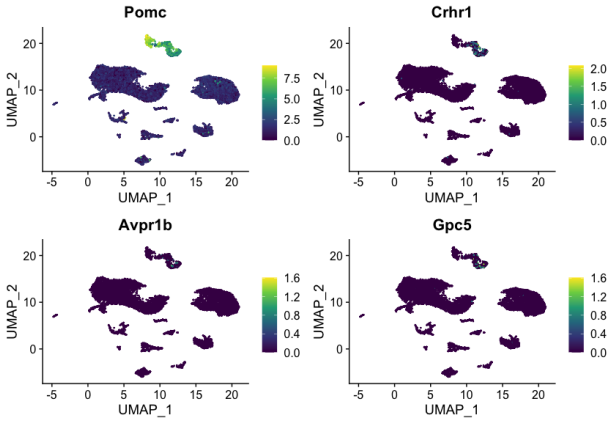

Figure 3: Attributing cell type may be estimated by canonical gene expression markers, here the cluster of corticotroph cells may be estimated as cluster 14

```
DimPlot(data, reduction ="umap", label = TRUE, pt.size = 0.5)+NoLegend()
#or pull out a  seurat of cort only
cortico <- subset(data, idents = "cort")
```

As a sanity check that the correct cluster was selected, a dimensional reuction plot

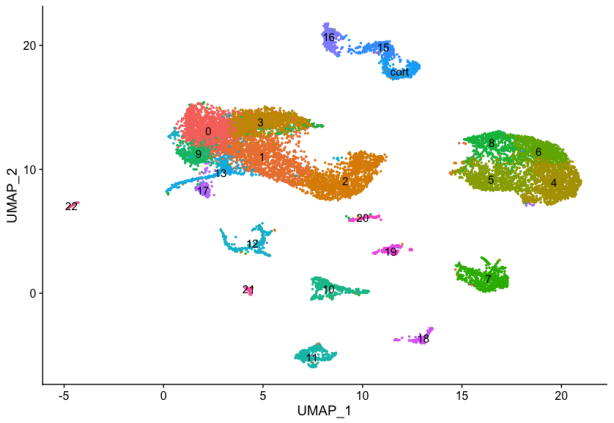

Figure 4: The corticotroph data is isolated and a sanity check performed to ensure the correct cluster has indeed been selected

Step 9. Re-cluster selected cell cluster

To investigate cell homogeneity within the cell cluster of interest, the same procedure may be repeated;

Data transformation, identification of principal components (PCA) Dimensional reduction (UMAP) *Identification of nearest neighbours (SNN)* Clustering using SNN.

```
#dimensional analysis
#PCA to identify PCs
cortico <- RunPCA(cortico,verbose= FALSE)
```

```r
#dimensional reduction
#UMAP dimensional reduction of the identified PCs
cortico <- RunUMAP(cortico, dims=1:30, verbose=FALSE) #note because SCTransform passes 3000 features ca

#Find nearest neighbours
cortico <- FindNeighbors(cortico, dims=1:30, verbose=FALSE)

#CLuster based on neighbour distances
cortico <- FindClusters(cortico, verbose=FALSE)

#plot the clusters
DimPlot(cortico, pt.size = 5, label=TRUE) + NoLegend()

#Visualise the canonical marker genes
FeaturePlot(cortico, features = c("Pomc"), pt.size = 5, label=TRUE)& scale_color_viridis_c()

#examine DE genes in the  corticotroph cluster
#find the markers for EVERY CLUSTER when compared to remaining cells, only +ve ones
cortico.markers <-FindAllMarkers(cortico, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)
#examine top 2 deferentially expressed genes in each corticotroph cluster
cortico.markers %>% group_by(cluster) %>% top_n(n=2, wt= avg_logFC)
```

A dimensional reduction plot can then reveal any heterogeneity in the cluster;

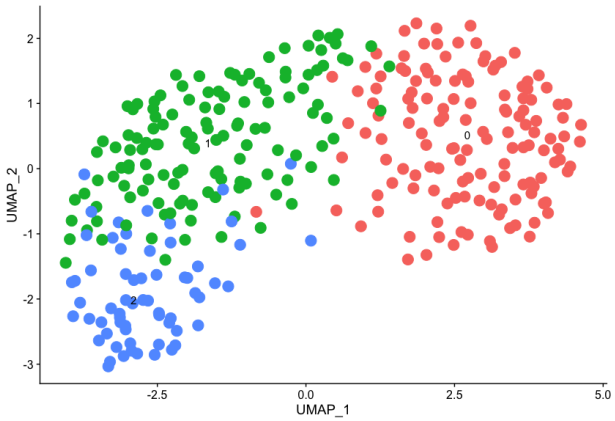

Figure 5: A re-clustering of the corticotroph data may reveal cell heterogeneity

A featureplot can be used to examine gene expression levels across these clusters within the corticotroph data;

Differential expression analysis can reveal differences between the clusters, for example the two top deferentially expressed genes for each cluster are;

```
##    p_value avg_logFC pct.1 pct.2 p_val_adj cluster   gene
## 1 4.82e-25 1.1924118 1.000 0.990  8.52e-21       0    Mt1
## 2 1.18e-21 1.3934787 0.949 0.771  2.08e-17       0    Mt2
## 3 5.09e-33 1.3168594 1.000 0.955  9.01e-29       1 Dnajb1
## 4 8.43e-32 1.4797565 1.000 0.915  1.49e-27       1    Fos
## 5 1.04e-07 0.5823509 1.000 1.000  1.83e-03       2   Pomc
## 6 3.74e-05 0.6825829 0.524 0.300  6.62e-01       2  Ddit4
```

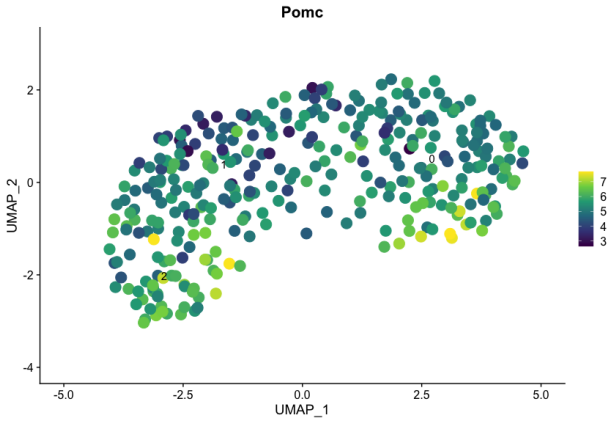

Figure 6: A featureplot can illustrate gene expression levels across these clusters within the corticotroph data, for example Pomc expression is higher in cluster 0 & 2.

Summary

The described pipeline is a powerful tool to investigate gene expression and cell homogeneity within sub-populations, however it is just a tool. For meaningful results to be obtained the user must apply knowledge of the samples under study to ensure biologically sensible data is being studied and any decisions based on sound biological features. The output is only as good as the input and parameters.

Future improvements

A future version of this script will include cell cycle considerations.

All suggestions for future improvements are welcome, as are comments on the current version. Please make these via the GitHub page.