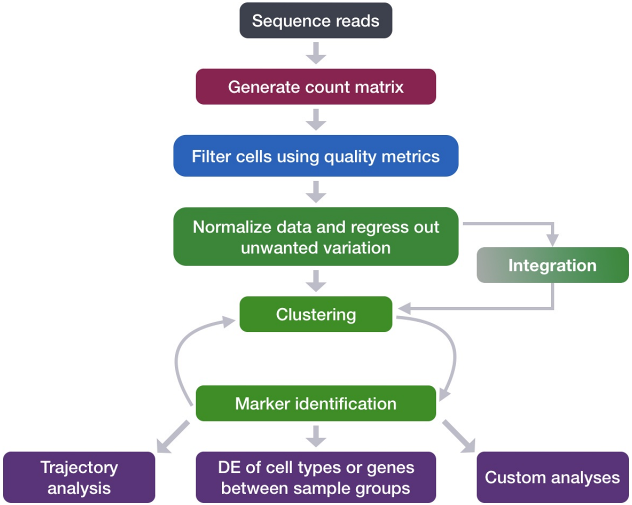
STAT718/BIOL703: Genomic Data Science

Single-cell RNA-seq Quality Control Analysis

Yen-Yi Ho (hoyen@stat.sc.edu)

scRNA-seq Data Analysis Workflow


Sequence reads
Generate count matrix
Filter cells using quality metrics
Normalize data and regress out unwanted variation
Integration
Clustering
Marker identification
Trajectory analysis
DE of cell types or genes between sample groups
Custom analyses

Exploring the dataset

We will use the data from Kang et al. In this paper, the authors present a computational algorithm that harnesses genetic variation (eQTL) to determine the genetic identity of each droplet containing a single cell (singlet) and identify droplets containing two cells from different individuals (doublets).

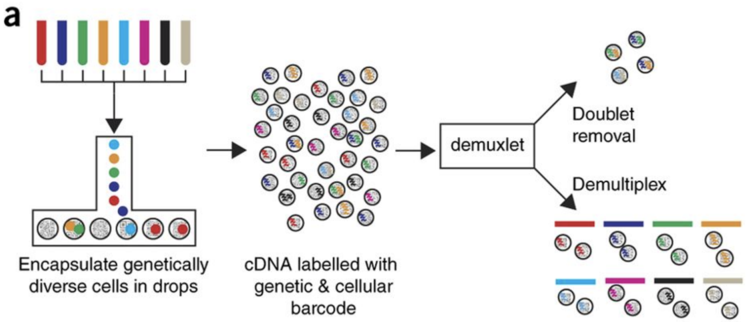

a
Encapsulate genetically diverse cells in drops
cDNA labelled with genetic & cellular barcode
demuxlet
Doublet removal
Demultiplex

Image credit: Kang et al, 2017

Raw Data

This dataset is available on GEO (GSE96583), however the available counts matrix lacked mitochondrial reads. The BAM files from the SRA (SRP102802) were downloaded. These BAM files were converted back to FASTQ files, then run through Cell Ranger to obtain the count data that we will be using. This count data is also freely available from 10X Genomics.

MetaData

In addition to the raw data, we also need to collect **information about the data**; this is known as **metadata**. Some relevant metadata for our dataset

- The libraries were prepared using 10X Genomics version 2 chemistry
- The samples were sequenced on the Illumina NextSeq 500
- PBMC samples from eight individual lupus patients were separated into two aliquots each: 1 treated with IFN-β (12,138) and 1 control (12,167).
- Since the samples are PBMCs, we will expect immune cells, such as:
  - B cells
  - T cells
  - NK cells
  - monocytes
  - macrophages
  - possibly megakaryocytes

R codes

Now, we can load the necessary libraries:

```r
# Load libraries
library(SingleCellExperiment)
library(Seurat)
library(tidyverse)
library(Matrix)
library(scales)
library(cowplot)
library(RCurl)
```

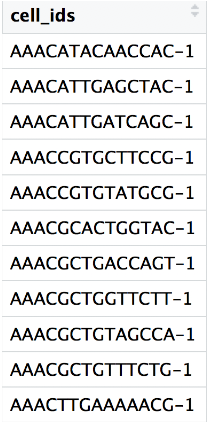
Loading single-cell RNA-seq count data

Regardless of the technology or pipeline used to process your raw single-cell RNA-seq sequence data, the output with quantified expression will generally be the same. That is, for each individual sample you will have the following **three files**:

1. a file with the **cell IDs**, representing all cells quantified
2. a file with the **gene IDs**, representing all genes quantified
3. a **matrix of counts** per gene for every cell

1. barcodes.tsv

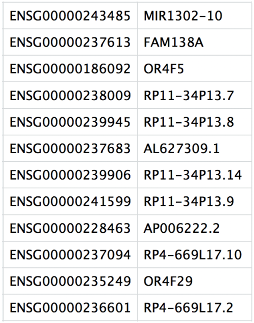
This is a text file which contains all cellular barcodes present for that sample. Barcodes are listed in the order of data presented in the matrix file (i.e. these are the column names).



cell_ids
AAACATACAACCAC-1
AAACATTGAGCTAC-1
AAACATTGATCAGC-1
AAACCGTGCTTCCG-1
AAACCGTGTATGCG-1
AAACGCACTGGTAC-1
AAACGCTGACCAGT-1
AAACGCTGGTTCTT-1
AAACGCTGTAGCCA-1
AAACGCTGTTTCTG-1
AAACTTGAAAAACG-1

2. features.tsv

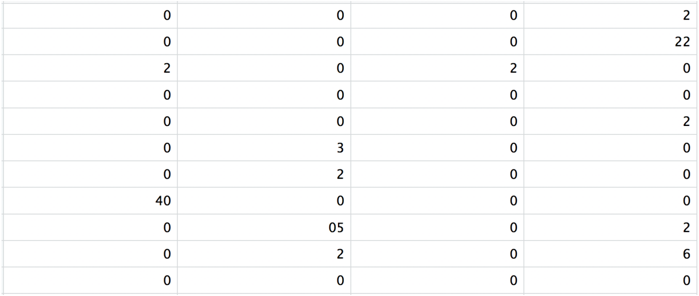
This is a text file which contains the identifiers of the quantified genes. The source of the identifier can vary depending on what reference (i.e. Ensembl, NCBI, UCSC) you use in the quantification methods, but most often these are official gene symbols. The order of these genes corresponds to the order of the rows in the matrix file (i.e. these are the row names).



ENSG00000243485	MIR1302-10
ENSG00000237613	FAM138A
ENSG00000186092	OR4F5
ENSG00000238009	RP11-34P13.7
ENSG00000239945	RP11-34P13.8
ENSG00000237683	AL627309.1
ENSG00000239906	RP11-34P13.14
ENSG00000241599	RP11-34P13.9
ENSG00000228463	AP006222.2
ENSG00000237094	RP4-669L17.10
ENSG00000235249	OR4F29
ENSG00000236601	RP4-669L17.2

3. `matrix.mtx`

This is a text file which contains a matrix of count values. The rows are associated with the gene IDs above and columns correspond to the cellular barcodes. Note that there are **many zero values** in this matrix.



0	0	0	2
0	0	0	22
2	0	2	0
0	0	0	0
0	0	0	2
0	3	0	0
0	2	0	0
40	0	0	0
0	05	0	2
0	2	0	6
0	0	0	0

Loading this data into R requires us to use functions that allow us to efficiently combine these three files into a single count matrix. However, instead of creating a regular matrix data structure, the functions we will use create a sparse matrix to reduce the amount of memory (RAM), processing capacity (CPU) and storage required to work with our huge count matrix.

Reading in scRNA-seq Data

Different methods for reading in data include:

1. **readMM()**: This function is from the **Matrix** package and will convert our standard matrix into a sparse matrix. The ‘features.tsv‘ file and ‘barcodes.tsv‘ must first be individually loaded into R and then they can be combined. For specific code and instructions on how to do this please see SC2.R.
2. **Read10X()**: This function is from the **Seurat** package and will use the Cell Ranger output directory as input, directly. With this method individual files do not need to be loaded in, instead the function will load and combine them into a sparse matrix.

Reading in a single sample

After processing 10X data using its proprietary software Cell Ranger, you will have an **outs** directory (always). Within this directory you will find a number of different files including the files listed below:

- web_summary.html: report that explores different QC metrics, including the mapping metrics, filtering thresholds, estimated number of cells after filtering, and information on the number of reads and genes per cell after filtering.
- BAM alignment files: files used for visualization of the mapped reads and for re-creation of FASTQ files, if needed
- filtered_feature_bc_matrix: folder containing all files needed to construct the count matrix using data filtered by Cell Ranger
- raw_feature_bc_matrix: folder containing all files needed to construct the count matrix using the raw unfiltered data

Most analyses start with **raw_feature_bc_matrix** folder to perform QC and filtering to account for the biology of our experiment/biological system.

Create a Seurat object

```
# How to read in 10X data for a single sample (output is a spars
ctrl_counts <- Read10X(data.dir = "single_cell_rnaseq/data/ctrl_

# Turn count matrix into a Seurat object (output is a Seurat obj
ctrl <- CreateSeuratObject(counts = ctrl_counts,
                           min.features = 100)
```

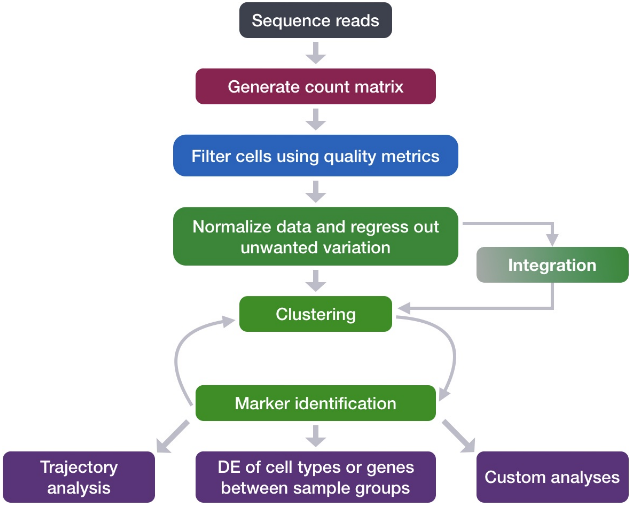
Read in Multiple Samples

```r
# Create a Seurat object for each sample
for (file in c("ctrl_raw_feature_bc_matrix", "stim_raw_feature_b
        seurat_data <- Read10X(data.dir = paste0("single_cell_rn
        seurat_obj <- CreateSeuratObject(counts = seurat_data,
                                         min.features = 100,
                                         project = file)
        assign(file, seurat_obj)
}
```

Merge Multiple Samples

```r
# Create a merged Seurat object
merged_seurat <- merge(x = ctrl_raw_feature_bc_matrix,
                       y = stim_raw_feature_bc_matrix,
                       add.cell.id = c("ctrl", "stim"))
```

Single-Cell Quality Control


Sequence reads
Generate count matrix
Filter cells using quality metrics
Normalize data and regress out unwanted variation
Integration
Clustering
Marker identification
Trajectory analysis
DE of cell types or genes between sample groups
Custom analyses

Quality Control Metric

- number of genes detected per UMI: this metric with give us an idea of the complexity of our dataset (more genes detected per UMI, more complex our data)
- mitochondrial ratio: this metric will give us a percentage of cell reads originating from the mitochondrial genes

Novelty Score

This value is quite easy to calculate, as we take the log10 of the number of genes detected per cell and the log10 of the number of UMIs per cell, then divide the log10 number of genes by the log10 number of UMIs. The score is related to the complexity of the RNA species.

```r
# Add number of genes per UMI for each cell to metadata
merged_seurat$log10GenesPerUMI <- log10(merged_seurat$nFeature_R
```

Mitochondrial Ratio

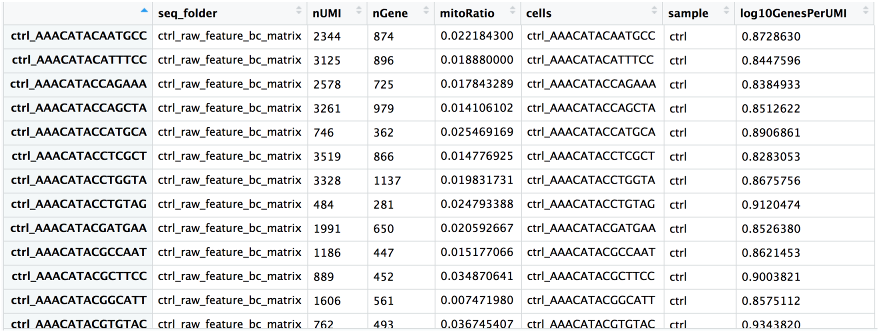
The **PercentageFeatureSet()** function takes in a `pattern` argument and searches through all gene identifiers in the dataset for that pattern. We are looking for any gene identifiers that begin with the pattern “MT-”.

```
# Compute percent mito ratio
merged_seurat$mitoRatio <- PercentageFeatureSet(object = merged_
merged_seurat$mitoRatio <- merged_seurat@meta.data$mitoRatio / 1
```

Additional metadata columns

- cell IDs
- condition information

Final Metadata



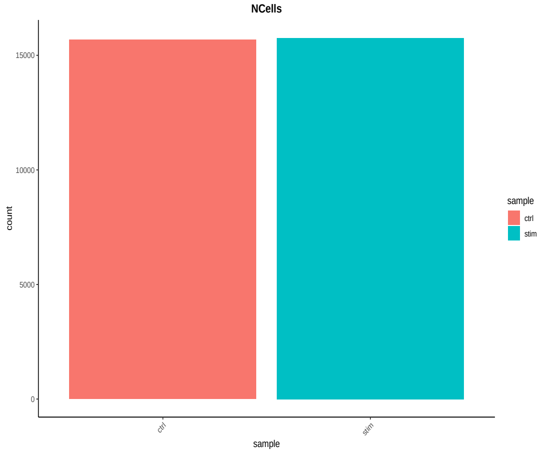
	seq_folder	nUMI	nGene	mitoRatio	cells	sample	log10GenesPerUMI
ctrl_AAACATACAATGCC	ctrl_raw_feature_bc_matrix	2344	874	0.022184300	ctrl_AAACATACAATGCC	ctrl	0.8728630
ctrl_AAACATACATTTCC	ctrl_raw_feature_bc_matrix	3125	896	0.018880000	ctrl_AAACATACATTTCC	ctrl	0.8447596
ctrl_AAACATACCAGAAA	ctrl_raw_feature_bc_matrix	2578	725	0.017843289	ctrl_AAACATACCAGAAA	ctrl	0.8384933
ctrl_AAACATACCAGCTA	ctrl_raw_feature_bc_matrix	3261	979	0.014106102	ctrl_AAACATACCAGCTA	ctrl	0.8512622
ctrl_AAACATACCATGCA	ctrl_raw_feature_bc_matrix	746	362	0.025469169	ctrl_AAACATACCATGCA	ctrl	0.8906861
ctrl_AAACATACCTCGCT	ctrl_raw_feature_bc_matrix	3519	866	0.014776925	ctrl_AAACATACCTCGCT	ctrl	0.8283053
ctrl_AAACATACCTGGTA	ctrl_raw_feature_bc_matrix	3328	1137	0.019831731	ctrl_AAACATACCTGGTA	ctrl	0.8675756
ctrl_AAACATACCTGTAG	ctrl_raw_feature_bc_matrix	484	281	0.024793388	ctrl_AAACATACCTGTAG	ctrl	0.9120474
ctrl_AAACATACGATGAA	ctrl_raw_feature_bc_matrix	1991	650	0.020592667	ctrl_AAACATACGATGAA	ctrl	0.8526380
ctrl_AAACATACGCCAAT	ctrl_raw_feature_bc_matrix	1186	447	0.015177066	ctrl_AAACATACGCCAAT	ctrl	0.8621453
ctrl_AAACATACGCTTCC	ctrl_raw_feature_bc_matrix	889	452	0.034870641	ctrl_AAACATACGCTTCC	ctrl	0.9003821
ctrl_AAACATACGGCATT	ctrl_raw_feature_bc_matrix	1606	561	0.007471980	ctrl_AAACATACGGCATT	ctrl	0.8575112
ctrl_AAACATACGTGTAC	ctrl_raw_feature_bc_matrix	762	493	0.036745407	ctrl_AAACATACGTGTAC	ctrl	0.9343820

Assessing the quality metrics

- Cell counts
- UMI counts per cell
- Genes detected per cell
- Complexity (novelty score)
- Mitochondrial counts ratio

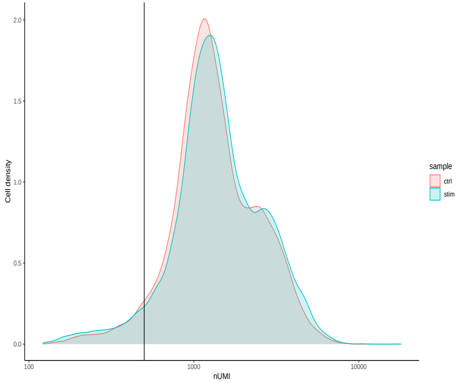
Cell Counts

The cell numbers can also vary by protocol. There are over 15,000 cells per sample, which is quite a bit more than the expected 12-13,000. It is clear that we likely have some junk 'cells' present.



UMI counts (transcripts) per cell

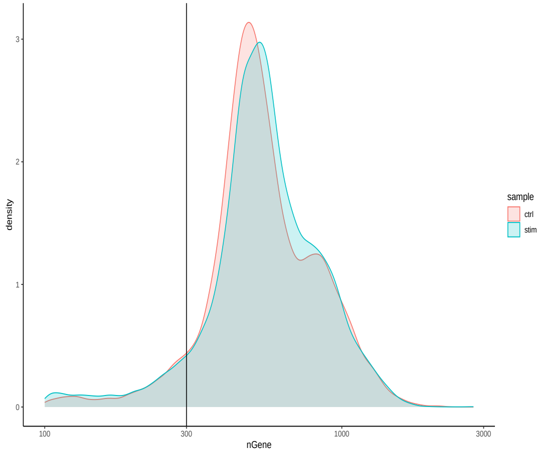
The UMI counts per cell should generally be above 500. If UMI counts are between 500-1000 counts, it is usable but the cells probably should have been sequenced more deeply.



The majority of our cells in both samples have 1000 UMIs or greater, which is great.

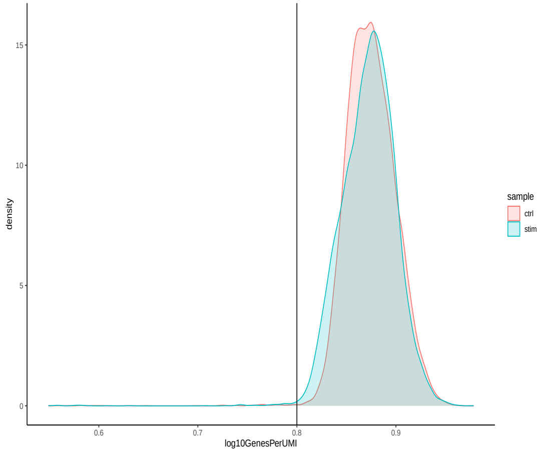
Genes Detected Per Cell

For high quality data, the proportional histogram should contain a single large peak that represents cells that were encapsulated. Bimodel distribution of the cells could indicate there are **biologically different types of cells**



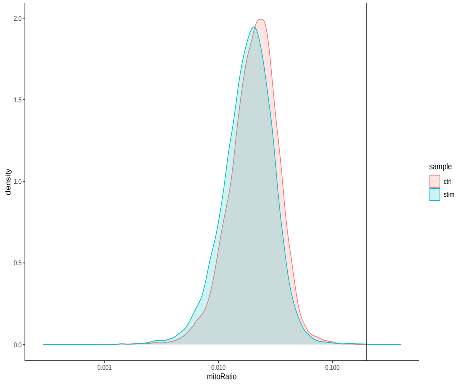
Complexity

Low complexity (low novelty) cells could represent a specific cell type (i.e. red blood cells which lack a typical transcriptome), or could be due to an artifact or contamination. Generally, we expect the novelty score to be above 0.80 for good quality cells.



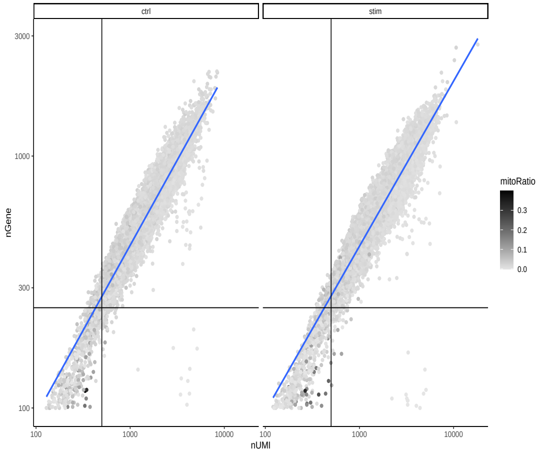
Mitochondrial counts ratio

This metric can identify whether there is a large amount of mitochondrial contamination from dead or dying cells. Poor quality samples for mitochondrial counts as cells which surpass the **0.2** mitochondrial ratio mark, unless of course it is expected in the samples.



Joint filtering effects

Two metrics that are often evaluated together are the number of UMIs and the number of genes detected per cell.



Filtering: Cell Level Filter

- nUMI > 500
- nGene > 250
- log10GenesPerUMI > 0.8
- mitoRatio < 0.2

Filtering: Gene Level Filter

If a gene is only expressed in a handful of cells, it is not particularly meaningful. Some routine analyses keep only genes which are expressed in 10 or more cells.