Homework 10

Chandrima Bhattacharya

25 March 2019

```
suppressWarnings(suppressPackageStartupMessages(library(ggplot2)))
suppressWarnings(suppressPackageStartupMessages(library(DESeq2)))
suppressWarnings(suppressPackageStartupMessages(library(dplyr)))
suppressWarnings(suppressPackageStartupMessages(library(SingleCellExperiment)))
suppressWarnings(suppressPackageStartupMessages(library(scater)))
suppressWarnings(suppressPackageStartupMessages(library(gridExtra)))
suppressWarnings(suppressPackageStartupMessages(library(scran)))
suppressWarnings(suppressPackageStartupMessages(library(Seurat)))
```

Question 1

```
reads <- read.table("WT-1.dge.txt.gz", header=TRUE)
```

a

The rows contains the various Genes (most probably arranged in alphabetical order), while the columns represent cells. There is 25319 rows and 1400 single cells.

b

```
rownames(reads) <- reads$GENE
reads$GENE <- NULL
matrix <- as.matrix(reads)
sce <- SingleCellExperiment(assays = list(counts = matrix))
sce
```

```
## class: SingleCellExperiment
## dim: 25319 1400
## metadata(0):
## assays(1): counts
## rownames(25319): A1BG A1BG-AS1 ... ZZZ3 hsa-mir-1273e
## rowData names(0):
## colnames(1400): GGTCCAGATCAT CCCCCATTATGC ... TCCCTTTATCAT
##   CGCCTTCTAATC
## colData names(0):
## reducedDimNames(0):
## spikeNames(0):
```

c

```
counts(sce[1:10,1:10])
```

```
##            GGTCCAGATCAT CCCCCATTATGC TTTACCTAACAG ATTCCCGAGTCA ATTCATTCTTTG
## A1BG                  0            0            0            0            0
## A1BG-AS1              0            0            0            0            0
## A2ML1                 0            0            0            0            0
## A2ML1-AS1             0            0            0            0            0
## A2ML1-AS2             0            0            0            0            0
## A4GALT                0            0            0            0            0
## AAAS                  1            0            0            0            1
## AACS                  2            0            0            1            1
## AACSP1                0            0            0            0            0
## AADACL3               0            0            0            0            0
##            GTCTTCTCCCAT CCGCTACTTCTC TTCCAGCCTCGG CGCTACACTGGA TTCGCTGAAAAC
## A1BG                  0            0            0            0            0
## A1BG-AS1              0            0            0            0            0
## A2ML1                 1            0            0            0            0
## A2ML1-AS1             0            0            0            0            0
## A2ML1-AS2             0            0            0            0            0
## A4GALT                0            0            0            0            0
## AAAS                  0            0            0            1            0
## AACS                  2            1            0            1            0
## AACSP1                0            0            0            0            0
## AADACL3               0            0            0            0            0
```

d

```r
sequencing_depth <- as.data.frame(colSums(counts(sce[,1:5]), na.rm = TRUE))
sequencing_depth
```

```
##              colSums(counts(sce[, 1:5]), na.rm = TRUE)
## GGTCCAGATCAT                                     42572
## CCCCCATTATGC                                     33667
## TTTACCTAACAG                                     30704
## ATTCCCGAGTCA                                     28555
## ATTCATTCTTTG                                     25858
```

e

```r
sce_sub <- as.data.frame(counts(sce[,1:5]))
not_zero <-  length(which(rowSums(sce_sub) != 0))
print(paste(" In the first five cells, there are ",not_zero, " gene counts greater than 0"))
```

```
## [1] " In the first five cells, there are  12066  gene counts greater than 0"
```

f

By changing the row names and column names of the following, we can keep in track by the metadata.

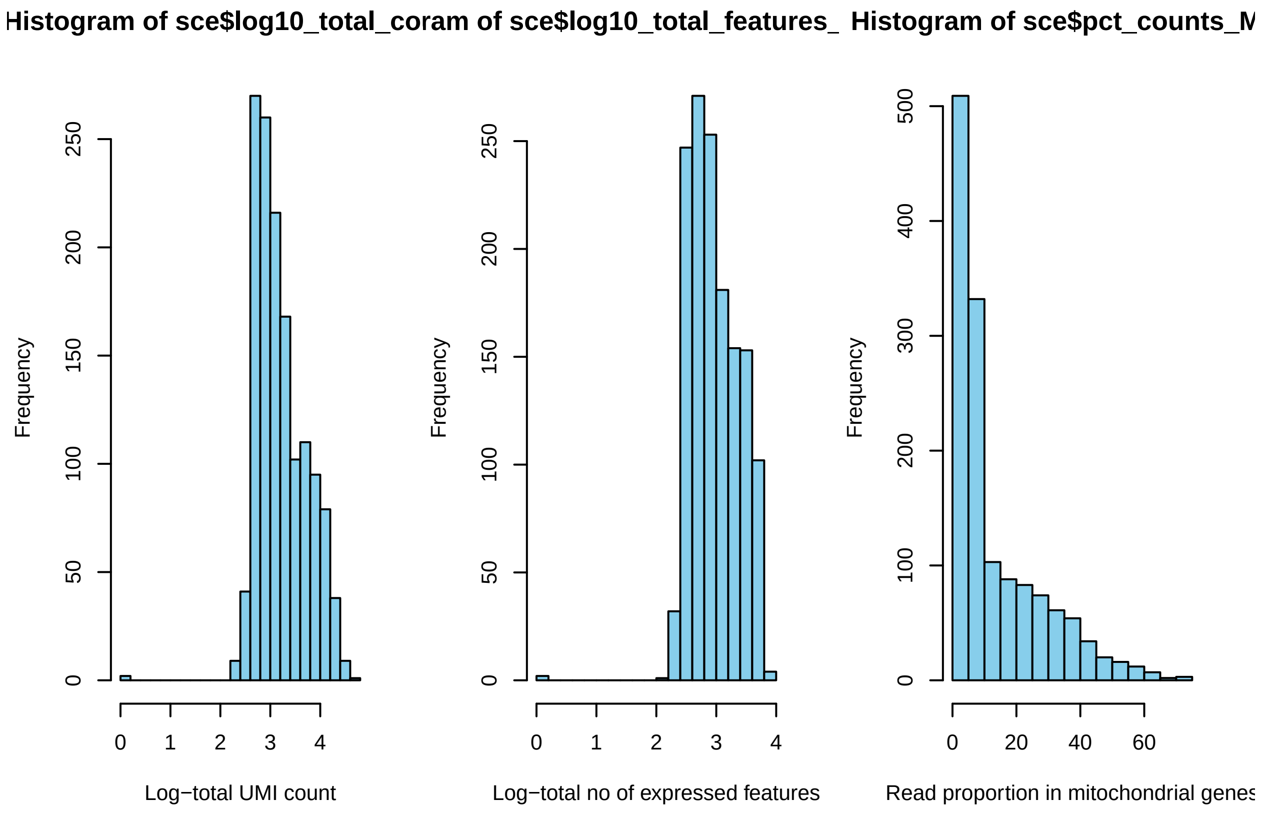
```r
cell_meta <- data.frame(cell_id = c(paste0("cell_", 1:1400)))
rownames(cell_meta) <- colnames(sce)
cell_meta$cell_ids <- colnames(sce)
gene_meta <- data.frame(gene_id = c(paste0("gene_", 1:25319)))
```

```
rownames(gene_meta) <- rownames(sce)
gene_meta$gene_ids <- rownames(sce)
colData(sce) <- DataFrame(cell_meta)
rowData(sce) <- DataFrame(gene_meta)
colnames(sce) <- sce$cell_id
counts(sce[1:15,1:15])
```

```
##            cell_1 cell_2 cell_3 cell_4 cell_5 cell_6 cell_7 cell_8 cell_9
## A1BG            0      0      0      0      0      0      0      0      0
## A1BG-AS1        0      0      0      0      0      0      0      0      0
## A2ML1           0      0      0      0      0      1      0      0      0
## A2ML1-AS1       0      0      0      0      0      0      0      0      0
## A2ML1-AS2       0      0      0      0      0      0      0      0      0
## A4GALT          0      0      0      0      0      0      0      0      0
## AAAS            1      0      0      0      1      0      0      0      1
## AACS            2      0      0      1      1      2      1      0      1
## AACSP1          0      0      0      0      0      0      0      0      0
## AADACL3         0      0      0      0      0      0      0      0      0
## AADAT           0      3      0      1      1      3      2      0      1
## AAED1           0      0      0      1      0      0      0      0      0
## AAGAB           0      1      1      2      0      1      0      1      1
## AAK1            0      0      0      1      0      0      0      0      0
## AAMDC           3      3      1      1      0      0      0      2      0
##            cell_10 cell_11 cell_12 cell_13 cell_14 cell_15
## A1BG             0       0       0       0       0       0
## A1BG-AS1         0       0       0       0       0       0
## A2ML1            0       0       0       0       0       0
## A2ML1-AS1        0       0       0       0       0       0
## A2ML1-AS2        0       0       0       0       0       0
## A4GALT           0       3       0       0       0       0
## AAAS             0       2       0       0       0       1
## AACS             0       1       0       0       0       2
## AACSP1           0       0       0       0       0       0
## AADACL3          0       0       0       0       0       0
## AADAT            0       2       0       2       0       1
## AAED1            0       0       0       0       1       0
## AAGAB            3       2       3       0       0       0
## AAK1             0       0       1       0       0       0
## AAMDC            1       1       1       1       1       1
```

g

```
mt.genes <- grep(pattern = "^MT-", rownames(sce), value = TRUE)
sce <- calculateQCMetrics(sce, feature_controls=list(Mito=c(mt.genes)))
par(mfrow=c(1,3))
hist(sce$log10_total_counts, breaks=20, col="skyblue",
    xlab="Log-total UMI count")
hist(sce$log10_total_features_by_counts, breaks=20, col="skyblue",
    xlab="Log-total no of expressed features")
hist(sce$pct_counts_Mito, breaks=20, col="skyblue",
    xlab="Read proportion in mitochondrial genes")
```

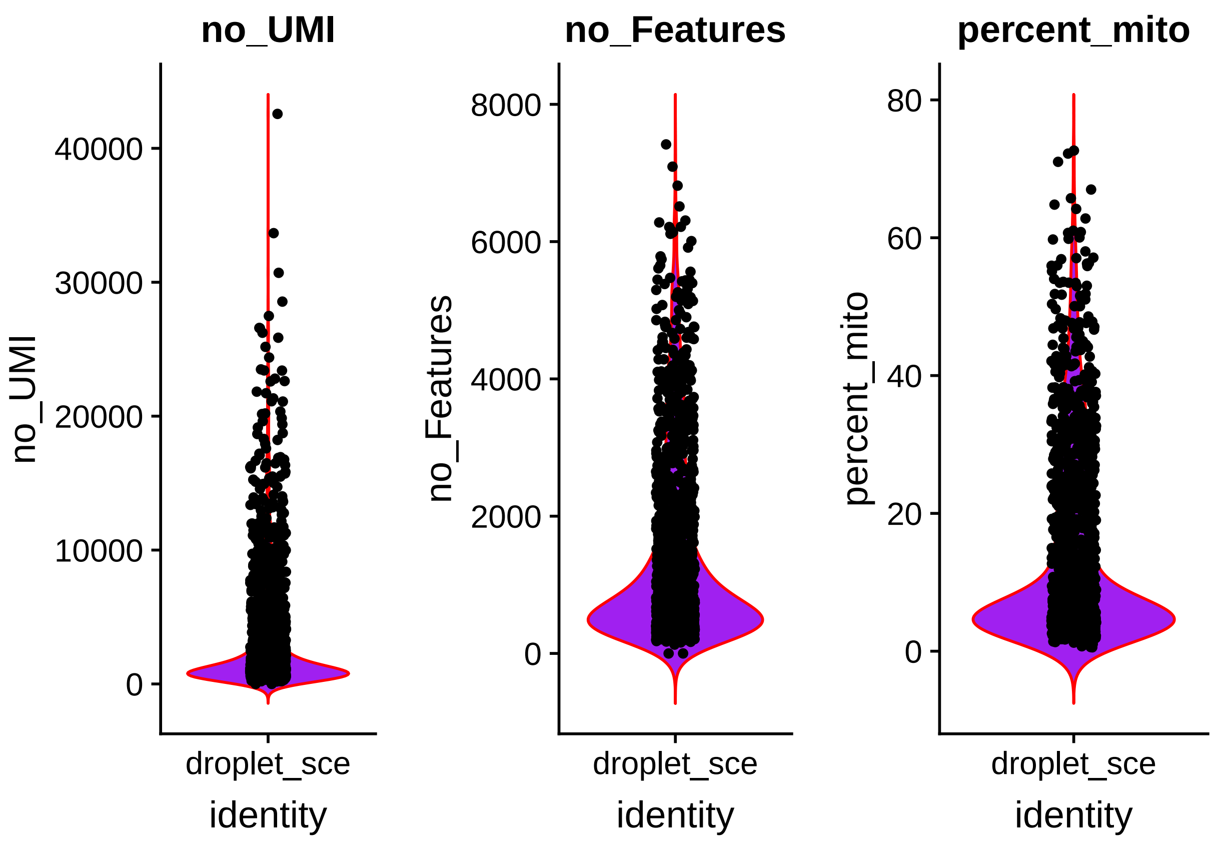


```r
sce_QC <- as.data.frame(sce$total_counts, row.names = colnames(sce))
sce_QC$no_UMI <- sce$total_counts
sce_QC$`sce$total_counts` <- NULL
sce_QC$no_Features <- sce$total_features_by_counts
sce_QC$identity <- c(rep("droplet_sce",ncol(sce_QC)))
sce_QC$percent_mito <- sce$pct_counts_Mito


# make violin plots
plot1 <- ggplot(sce_QC, aes(x=identity, y=no_UMI)) +
  geom_violin(trim=FALSE, fill='purple', color="red")+
  geom_jitter(shape=16, position=position_jitter(0.1))+ggtitle("no_UMI")
plot2 <- ggplot(sce_QC, aes(x=identity, y=no_Features)) +
  geom_violin(trim=FALSE, fill='purple', color="red")+
  geom_jitter(shape=16, position=position_jitter(0.1))+ggtitle("no_Features")
plot3 <- ggplot(sce_QC, aes(x=identity, y=percent_mito)) +
  geom_violin(trim=FALSE, fill='purple', color="red")+
  geom_jitter(shape=16, position=position_jitter(0.1))+ggtitle("percent_mito")
grid.arrange(plot1, plot2, plot3, ncol=3)
```

```
## Warning: Removed 2 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



The number of transcripts in each cell is represented by UMI or total count. Every transcript recieved an UMI for droplet based sequencing. UMI can also be represented as library size as it adds the count value in a cell in the matrix.
The total number of genes or features includes all the genes with count value for each cell greater than zero. The log transform gives a near normal distribution while the violin plots help to visualize the data distribution.

h

```
sce <- sce[, sce$total_counts > 200]
sce <- sce[, sce$total_counts < 20000]
sce <- sce[, sce$total_features_by_counts > 150]
sce <- sce[, sce$total_features_by_counts < 6000]
sce <- sce[,sce$pct_counts_Mito < 50]
sce
```

```
## class: SingleCellExperiment
## dim: 25319 1331
## metadata(0):
## assays(1): counts
## rownames(25319): A1BG A1BG-AS1 ... ZZZ3 hsa-mir-1273e
## rowData names(10): gene_id gene_ids ... total_counts
##   log10_total_counts
## colnames(1331): cell_19 cell_21 ... cell_1399 cell_1400
## colData names(38): cell_id cell_ids ...
##   pct_counts_in_top_200_features_Mito
##   pct_counts_in_top_500_features_Mito
## reducedDimNames(0):
## spikeNames(0):
```

I did filtering based on the following criterias: 200< no of UMI < 20000
150 < no of features < 6000
mRNA content > 50%

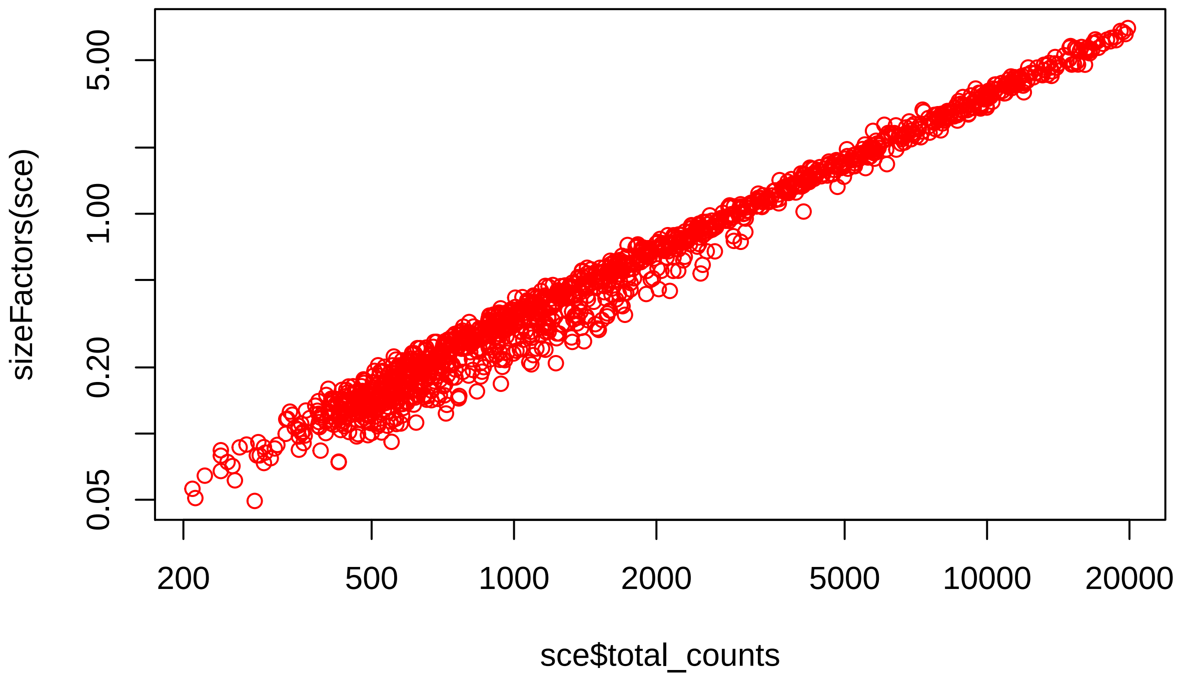
We see, there are 1371 cells still being there. So, 97% of my cells are still being there after removing cells based on above criteria. Also, no outlier analysis was done! The numbers were arbitarily decided.

i

```
sce <- computeSumFactors(sce, min.mean=0.1)
summary(sizeFactors(sce))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.04938 0.19520 0.39586 1.00000 1.21889 6.99864
```

```
plot(sce$total_counts, sizeFactors(sce), log="xy", col="red")
```



```r
sce_normalize <- scater::normalize(sce,
  exprs_values = "counts", return_log = TRUE,
  log_exprs_offset = NULL, centre_size_factors = TRUE,
  preserve_zeroes = FALSE)
```

We see that there is high correlation. This implies bias in sequencing depth and capture efficiancy.

Question 2

a

```r
counts <- reads
Seurat_obj <- CreateSeuratObject(counts, project = "Seurat")
```

b

```r
mito <- grep(pattern = "^MT-", x = rownames(x = Seurat_obj@data), value = TRUE)
mito_frac <- Matrix::colSums(Seurat_obj@raw.data[mito, ])/ Matrix::colSums(Seurat_obj@raw.data)
Seurat_obj <- AddMetaData(object = Seurat_obj, metadata = mito_frac, col.name = "mito_frac")
Seurat_obj <- FilterCells(object = Seurat_obj,
                       subset.names = c("nUMI", "nGene", "mito_frac"),
                       low.thresholds = c(200, 150, 0), high.thresholds = c(20000,6000,50))
```

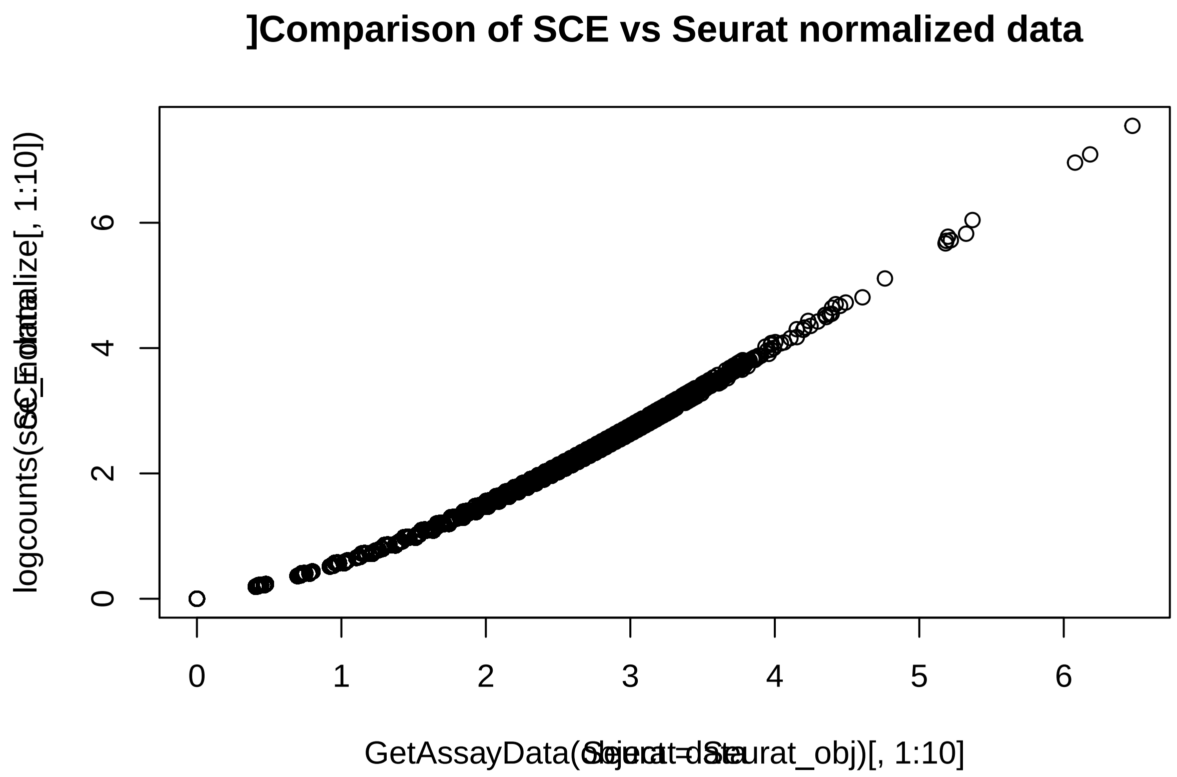
c

```r
Seurat_obj <- NormalizeData(object = Seurat_obj,
                           normalization.method = "LogNormalize",
                           scale.factor = 1e4)
```

You access the normalized counts by GetAssayData(object = Seurat_obj).

d

```r
colnames(sce_normalize) <- sce_normalize$cell_ids
plot(x = GetAssayData(object = Seurat_obj)[,1:10],
     y = logcounts(sce_normalize[,1:10]))
title(main = "]Comparison of SCE vs Seurat normalized data",
      xlab = "Seurat data", ylab = "SCE data")
```



We see high correlation. It is nearly x=y relation, but that is because Seurat uses the same normalization features and we used the same filtering methods.

Question 3

We see that the UMAP and tSNA dimentional reduction can help us differentiate cell type based on clustering. Various cell-types like BCell, CD4 and CD8 Tcells, etc. all form their own clusters. Same we can do for bone, cartiledge, etc., as long as we have access to data. If we can superimpose our sample over the total cell, we can find where the following sample is aligning. This would help us figure out what our sample is! Also, we can take a rigorous identification protocol, by which we would look at the gene expression and match the sample based on database. I was not being able to do it myself because of version compatibility, but based on discussion with Rekha, I guess it may be endocrine tissues, based on Protein Atlas database.