Homework 8 (Week 10)

Zeyu Wang/Charlie

March 25, 2019

Problem 1

1.1 Import data

```
readcounts <- read.table("C:\\Users\\WZY77\\Downloads\\WT-1.dge.txt", header=TRUE)
```

The rows show genes, while the columns represent the cells.

1.2 Create SinglecCell experiment object

```
suppressPackageStartupMessages(library(SingleCellExperiment))

rownames(readcounts) <- readcounts$GENE
readcounts$GENE <- NULL
counts_matrix <- as.matrix(readcounts)
SCE <- SingleCellExperiment(assays = list(counts = counts_matrix))
```

1.3 Part of counts matrix

```
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
```

1.4 Sequencing Depth

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

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

1.5 Non-zero gene counts

```
SCE_5 <- as.data.frame(counts(SCE[,1:5]))
non_zero <-  length(which(rowSums(SCE_5) != 0))
print(paste("There are: ",non_zero, "non-zero gene counts in the first five cells"))
```

```
## [1] "There are:  12066 non-zero gene counts in the first five cells"
```

1.6 Changing row and column names

```
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[1:10,1:10])
```

```
##           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
##           cell_10
## A1BG            0
## A1BG-AS1        0
## A2ML1           0
## A2ML1-AS1       0
## A2ML1-AS2       0
## A4GALT          0
## AAAS            0
## AACS            0
## AACSP1          0
## AADACL3         0
```

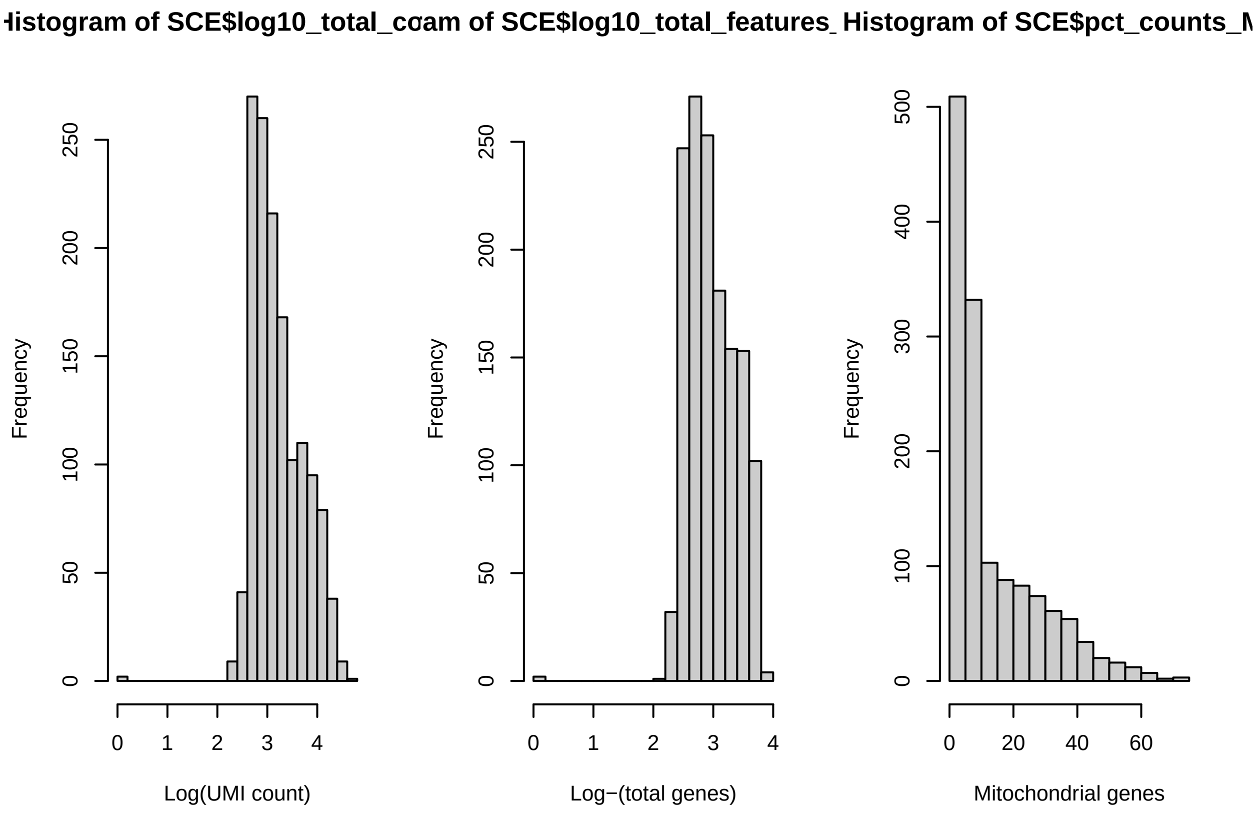
1.7

Generate histograms (from SCE tutorial)

```
suppressPackageStartupMessages(library(scater))

mit <- grep(pattern = "^MT-", rownames(SCE), value = TRUE)

SCE <- calculateQCMetrics(SCE, feature_controls=list(Mito=c(mit)))
par(mfrow=c(1,3))
hist(SCE$log10_total_counts, breaks=20, col="grey80",
     xlab="Log(UMI count)")
hist(SCE$log10_total_features_by_counts, breaks=20,
     col="grey80", xlab="Log-(total genes)")
hist(SCE$pct_counts_Mito, breaks=20, col="grey80",
     xlab="Mitochondrial genes")
```



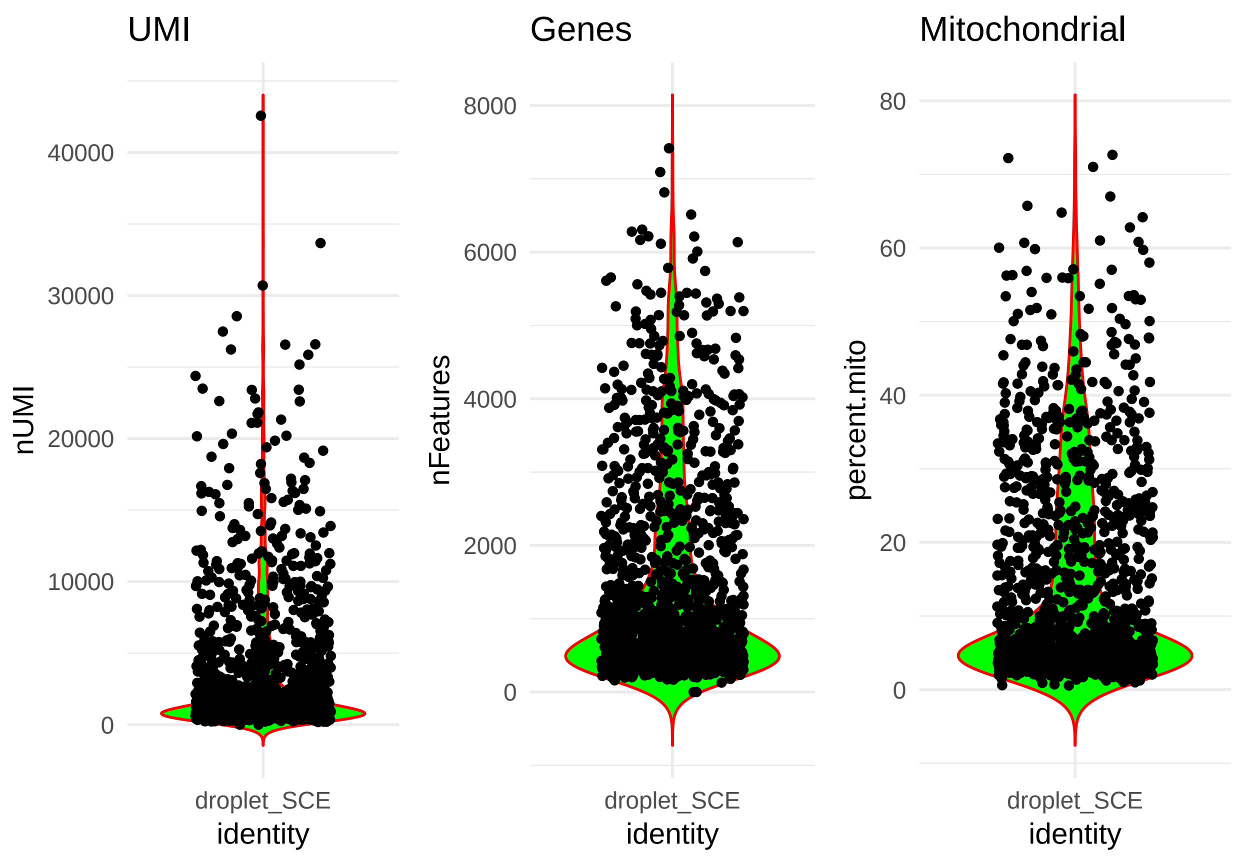
Generate violin plots with ggplot

```
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(gridExtra))

SCE_QC <- as.data.frame(SCE$total_counts, row.names = colnames(SCE))
SCE_QC$nUMI <- SCE$total_counts
SCE_QC$`SCE$total_counts` <- NULL
SCE_QC$nFeatures <- SCE$total_features_by_counts
SCE_QC$identity <- c(rep("droplet_SCE",ncol(SCE_QC)))
SCE_QC$percent.mito <- SCE$pct_counts_Mito

plot1 <- ggplot(SCE_QC, aes(x=identity, y=nUMI)) +
  geom_violin(trim=FALSE, fill='green', color="red")+ theme_minimal()+
  geom_jitter(shape=16, position=position_jitter(0.3))+ggtitle("UMI")
plot2 <- ggplot(SCE_QC, aes(x=identity, y=nFeatures)) +
  geom_violin(trim=FALSE, fill='green', color="red")+ theme_minimal()+
  geom_jitter(shape=16, position=position_jitter(0.3))+ggtitle("Genes")
plot3 <- ggplot(SCE_QC, aes(x=identity, y=percent.mito)) +
  geom_violin(trim=FALSE, fill='green', color="red")+ theme_minimal()+
  geom_jitter(shape=16, position=position_jitter(0.3))+ggtitle("Mitochondrial")

suppressWarnings(grid.arrange(plot1, plot2, plot3, ncol=3))
```



In my thought, UMI, just like a kind of unit, represents the number of transcripts captured from each cell. It's like a value detected from each droplet, used to make a standard among all cells we get, and can used to find droplets that contains more or less than normal case. The total number of features for each cell are obviously all the genes that have at least one count value detected from cells.

1.8 Filtering

```
SCE <- SCE[, SCE$total_counts > 400]
SCE <- SCE[, SCE$total_counts < 30000]
SCE <- SCE[, SCE$total_features_by_counts > 300]
SCE <- SCE[, SCE$total_features_by_counts < 5500]
SCE <- SCE[,SCE$pct_counts_Mito < 50]
```

From the violin graph, I think a fair filter criterion should be: [400, 30000] for UMI, [300, 5500] for total features and [0, 50] for mitochondria genes. Since values that either too low or too high should not be a normal case, while a high value of mitochondrial genes could probably mean the cell is under press.

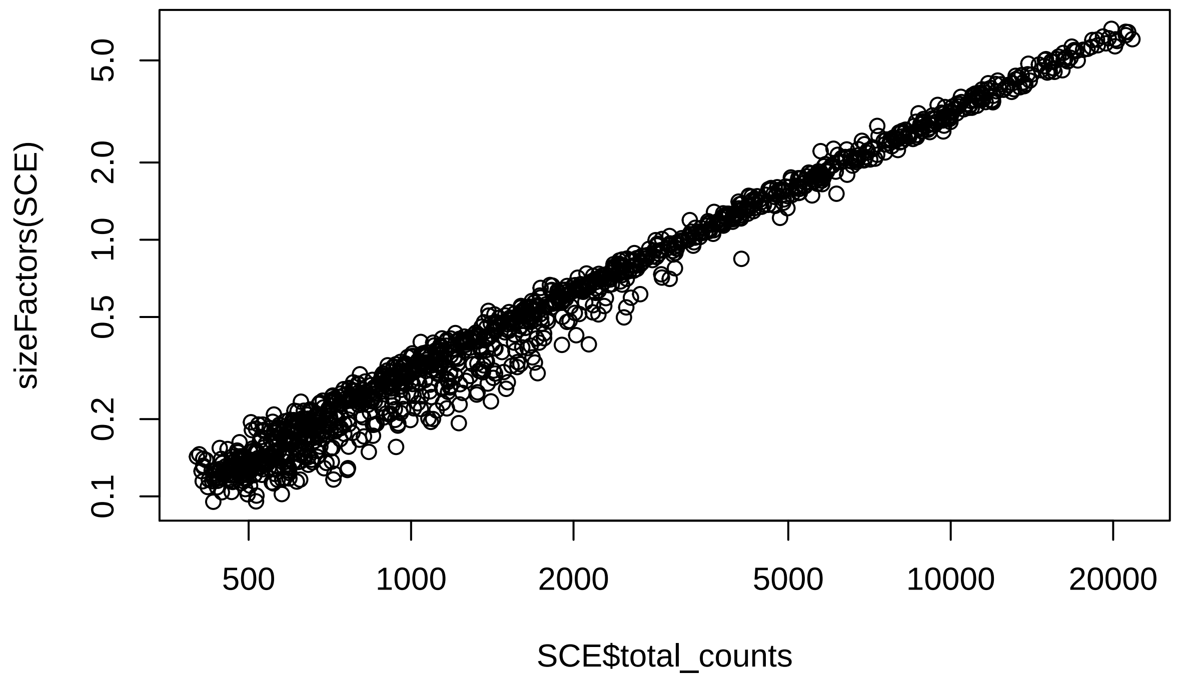
But I did not use tools or functions to decide these values, actually I was thinking that removing about 5~10% abnormal value from two boundries maybe helpful to downstream analysis.

1.9 Normalizing the data

From SCE tutorial

```
suppressPackageStartupMessages(library(scran))
SCE <- computeSumFactors(SCE, min.mean=0.1)

#summary(sizeFactors(SCE))
plot(SCE$total_counts, sizeFactors(SCE), log="xy")
```



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


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

```
##             cell_16 cell_19 cell_21 cell_22 cell_23 cell_24 cell_25 cell_26
## A1BG              0       0       0       0       0       0       0       0
## A1BG-AS1          0       0       1       0       0       0       0       0
## A2ML1             0       0       0       0       0       0       0       0
## A2ML1-AS1         0       0       0       0       0       0       0       0
## A2ML1-AS2         0       0       0       0       0       0       0       0
## A4GALT            0       0       0       0       0       0       0       1
## AAAS              0       1       0       0       0       1       1       1
## AACS              0       1       6       0       5       1       0       0
## AACSP1            0       0       0       0       0       0       0       0
## AADACL3           0       0       0       0       0       0       0       0
##             cell_27 cell_28
## A1BG              0       0
## A1BG-AS1          0       0
## A2ML1             0       0
## A2ML1-AS1         0       0
## A2ML1-AS2         0       0
## A4GALT          0       0
## AAAS            0       0
## AACS            0       2
## AACSP1          0       0
## AADACL3         0       0
```

This graph shows a pretty similar correlation with the figure on tutorial page. As the tutorial claims, this indicates that capture efficiency and sequencing depth are major biases.

Problem 2

2.1 create a seurat object

```r
suppressPackageStartupMessages(library(Seurat))
```

```
## Warning: package 'Seurat' was built under R version 3.5.3
```

```
## Warning: package 'cowplot' was built under R version 3.5.3
```

```r
Seurat1 <- CreateSeuratObject(raw.data  = readcounts, project = "Seurat1")
```

2.2 Filter based on same parameters

```r
mito <- grep(pattern = "^MT-", x = rownames(x = Seurat1@data), value = TRUE)
mito_frac <- Matrix::colSums(Seurat1@raw.data[mito, ])/ Matrix::colSums(Seurat1@raw.data)

Seurat1 <- AddMetaData(object = Seurat1, metadata = mito_frac, col.name = "mito_frac")

Seurat1 <- FilterCells(object = Seurat1,
                       subset.names = c("nUMI", "nGene", "mito_frac"),
                       low.thresholds = c(400, 300, 0), high.thresholds = c(30000,5500,50))
```

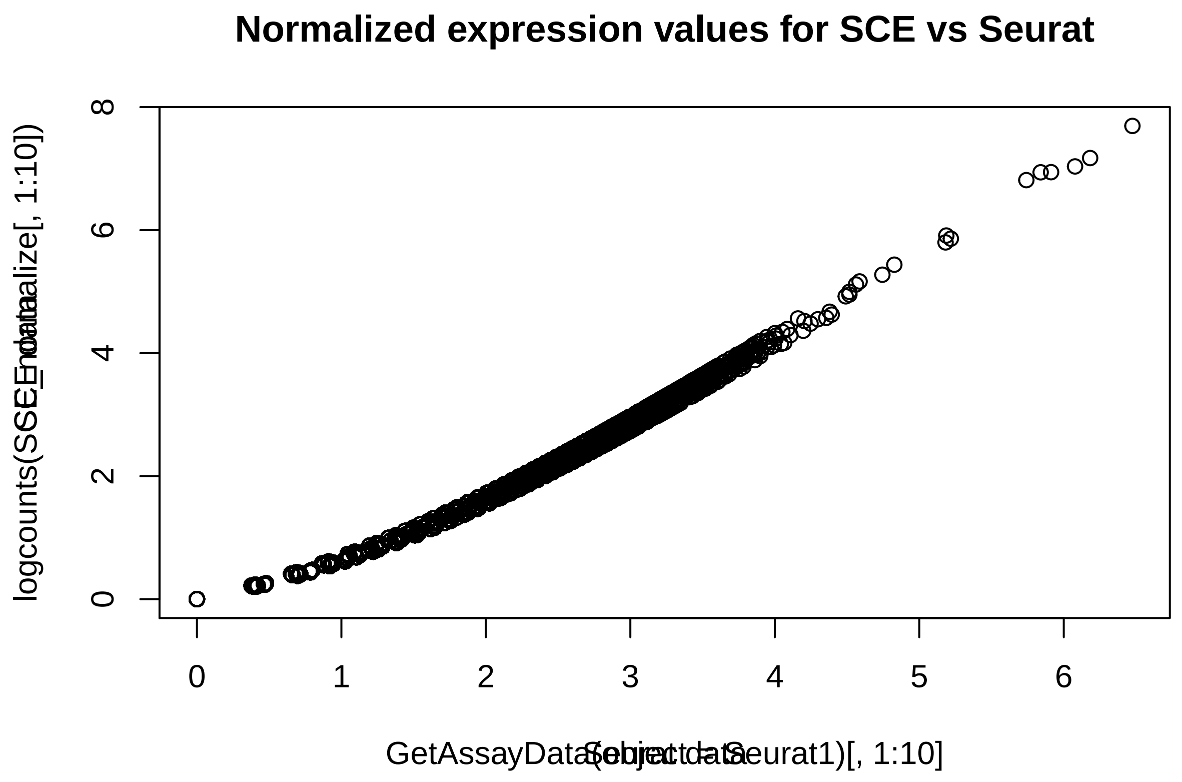
2.3 Normalize the data

```r
Seurat1 <- NormalizeData(object = Seurat1, normalization.method = "LogNormalize",
                        scale.factor = 10000)

# Access normalized data by:
# GetAssayData(object = Seurat1, slot = 'data')
```

2.4 Compare normalized values from Seurat and SCE

```r
# First I need to change back my column names in the SCE object
colnames(SCE_normalize) <- SCE_normalize$cell_ids
plot(x = GetAssayData(object = Seurat1)[,1:10],
     y = logcounts(SCE_normalize[,1:10]))
title(main = "Normalized expression values for SCE vs Seurat",
      xlab = "Seurat data", ylab = "SCE data")
```



I found it's kind of hard for me to remove zero counts in a short time, so I stop to make another kind of graph, which is the one above. It shows normalized data from two packages are similar, which looks kind of like a straight line of y = x. That also means these packages work in similar ways, though part of the reason should be the same process we did seperately by using them.

Problem 3

Result:

I guess it belongs to endocrine tissues. And I'm sorry for not able to provide executable codes.

Explanation:

I have problem installing "mygene" package which can help with filtering marker genes that helps to decide which celltype we are looking at. As a result, I have to work with Rekha, and based on her filter result, we try to find out the celltype we worked with by looking through genes with highest expression level, while we would definitely not do so if we have longer time to search.

We actually filtered 66 genes with highest expression values, and tried to check if they are marker genes with a website: http://biocc.hrbmu.edu.cn/CellMarker/ . But even filtering mitochondrial genes and ribosomal genes, there are still lots of genes of cytoskeleton or other general genes. Then we tried to look reversely, first find a tissue marker and then check if it was in our list. Starting from insulin, which is IDDM, then cardiomyocyte, lung surface, when we came to endocrine tissue, it shows only one specific marker, "CHGA", which shows in our data, though with a lower expression value of about 9.

We do not have much confidence to say it is definitely endocrine tissue, because the approach we used was really not scientific.