Simulation

Xinyi Lin

9/10/2020

Problems need to solve:

3. DCATS: n_samples? do p-values correspond to each cluster?

```
library(splatter)
library(Seurat)
library(speckle)
library(DCATS)
library(ggplot2)
library(tidyverse)
```

Simulation 1

Set up the group probabilities for Normal and Mutate condtions

```
probNor = c(1/3,1/3,1/3)
probMut = c(1/2,1/6,1/3)
```

Simulation

First, we simulate data with three groups and two of them are similar, which means they might have high misclustering rate. 'de.prob' specifies the probability that a gene selected is differentially expressed between the cluster and the rest of the cells.

```
set.seed(12345)

# simulate normal
param.groups <- newSplatParams(batchCells = c(600, 600, 600), nGenes = 100)
simNor <- splatSimulateGroups(param.groups, group.prob = probNor, de.prob = c(0.1,0.1,0.5), verbose = F
simNor@colData@rownames = str_replace(simNor@colData@rownames, "Cell", "NorCell")
simNor_mat <- counts(simNor)

# simulate mutate
simMut <- splatSimulateGroups(param.groups, group.prob = probMut, de.prob = c(0.1,0.1,0.5), verbose = F
simMut@colData@rownames = str_replace(simMut@colData@rownames, "Cell", "MutCell")
simMut_mat <- counts(simMut)
```

Batch information of normal and mutate sample

```
batchNor = simNor@colData@listData$Batch %>%
  str_replace("Batch", "Nor")
batchMut = simMut@colData@listData$Batch %>%
  str_replace("Batch", "Mut")
```

Then, seperate the normal sample and mutate sample by batch and combind them into a list.

```
# Normal
seuratNor <- CreateSeuratObject(counts = simNor_mat, project="Splatter")
seuratNor <- AddMetaData(object = seuratNor, metadata = batchNor, col.name = 'batch')
seuratNor <- AddMetaData(object = seuratNor, metadata = rep("Normal", 1800), col.name = 'condition')
# Mutate
seuratMut <- CreateSeuratObject(counts = simMut_mat, project="Splatter")
seuratMut <- AddMetaData(object = seuratMut, metadata = batchMut, col.name = 'batch')
seuratMut <- AddMetaData(object = seuratMut, metadata = rep("Mutate", 1800), col.name = 'condition')
```

```
# split by batch
listNor = SplitObject(seuratNor, split.by = "batch")
listMut = SplitObject(seuratMut, split.by = "batch")

# combine Normal and Mutate
listSamples = c(listNor, listMut)
```

Process by using Seurat

Log-normalization and identify variable features for each batches separately

```
for (i in 1:length(listSamples)) {
    listSamples[[i]] <- NormalizeData(listSamples[[i]], verbose = FALSE)
    listSamples[[i]] <- FindVariableFeatures(listSamples[[i]], selection.method = "vst",
        nfeatures = 2000, verbose = FALSE)
}
```

Integrate all batches

```
anchors <- FindIntegrationAnchors(object.list = listSamples, dims = 1:30, verbose = FALSE)
integratedSamples <- IntegrateData(anchorset = anchors, dims = 1:30, verbose = FALSE)
```

```
## Warning: Adding a command log without an assay associated with it
```

```
DefaultAssay(integratedSamples) <- "integrated"

# Run the standard workflow for visualization and clustering
integratedSamples <- ScaleData(integratedSamples, verbose = FALSE)
integratedSamples <- RunPCA(integratedSamples, npcs = 30, verbose = FALSE)
```

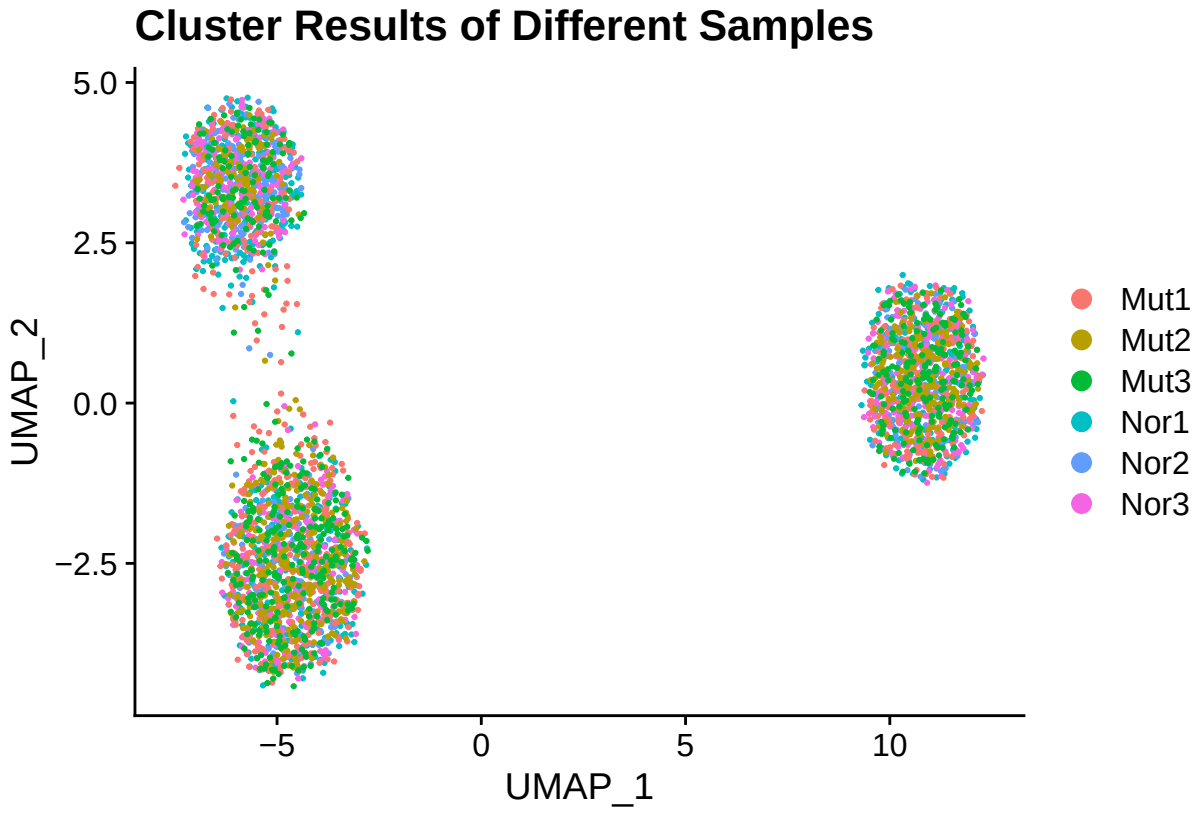
```
# check how many PCs to choose
ElbowPlot(integratedSamples)
# dim = 5
```

```
integratedSamples<-FindNeighbors(integratedSamples, dims = 1:5, verbose = FALSE)
integratedSamples<-FindClusters(integratedSamples, resolution = 0.5, algorithm=2, verbose = FALSE)
```

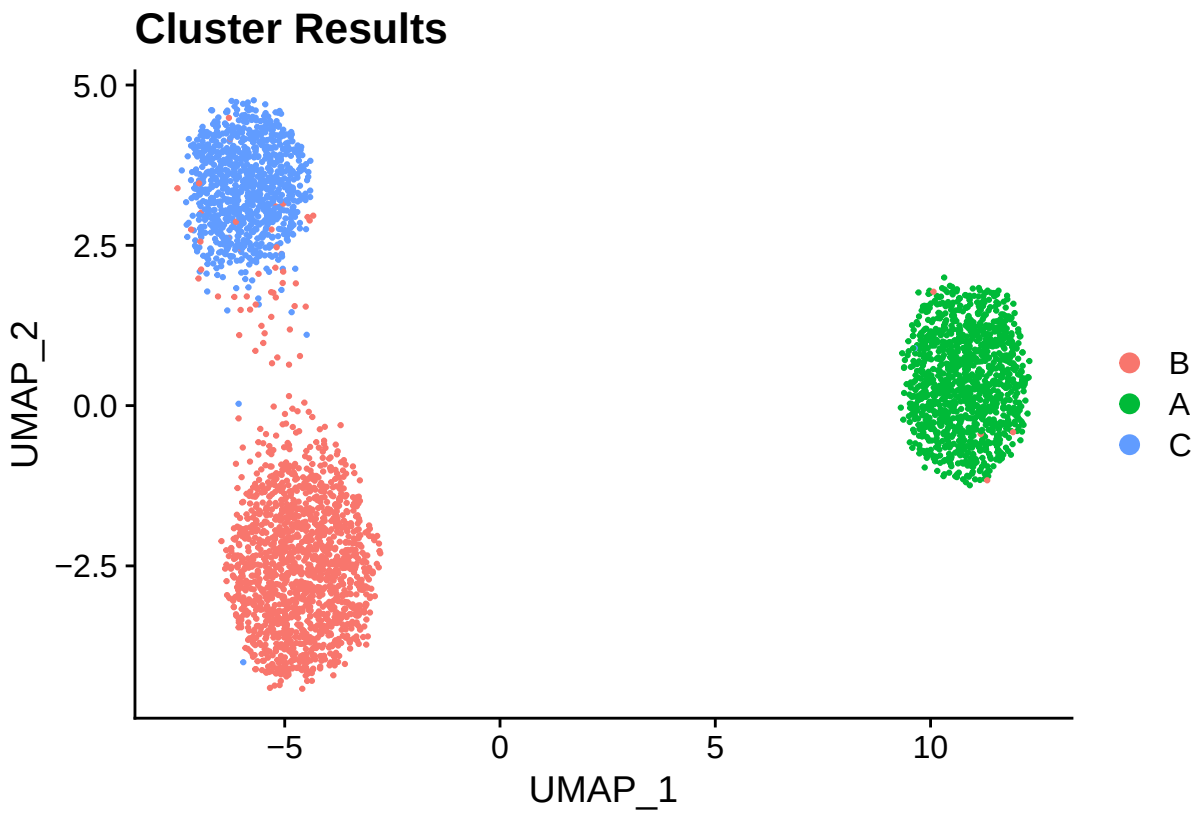
```
integratedSamples <- RunUMAP(integratedSamples, reduction = "pca", dims = 1:30, verbose = FALSE)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'
## This message will be shown once per session
```

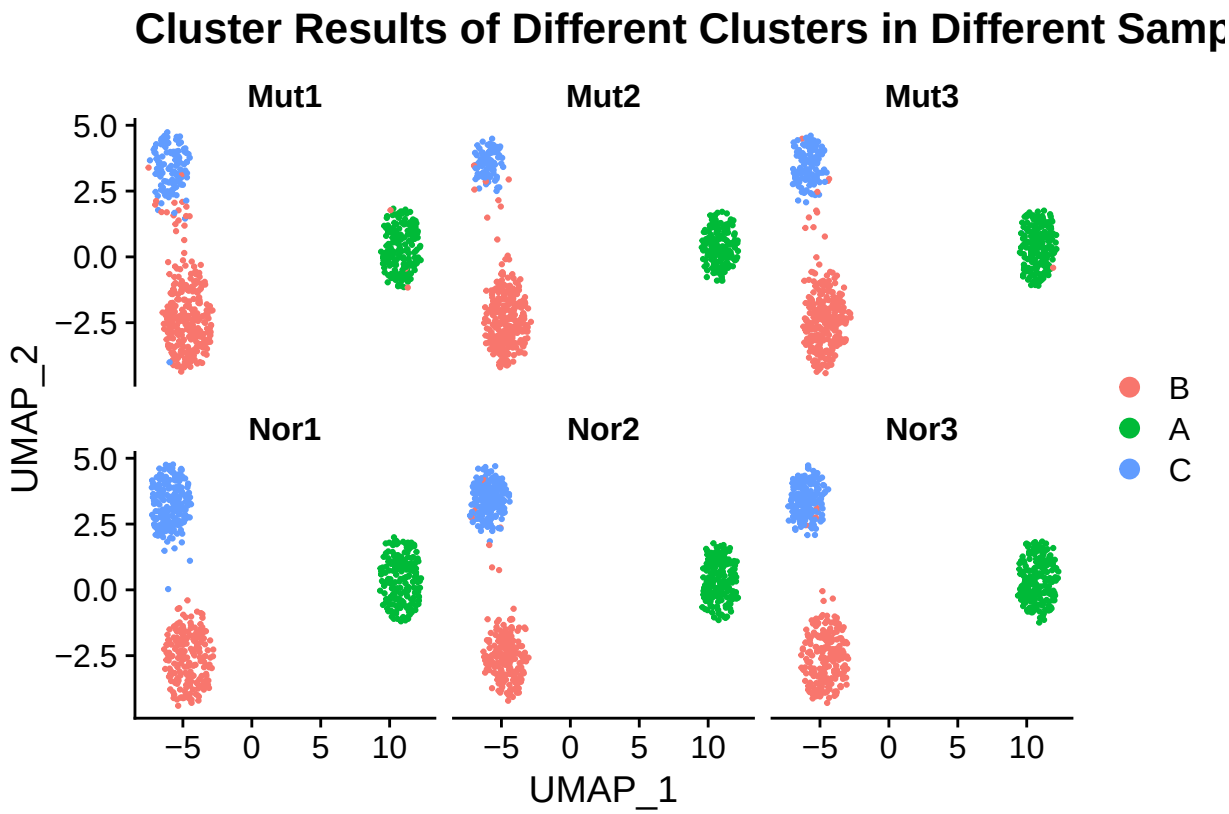
```
integratedSamples@active.ident = integratedSamples@active.ident %>%
  plyr::mapvalues(from = c("1", "0", "2"), to = c("A", "B", "C"))
# the plot which shows the cluster results of different samples
DimPlot(integratedSamples, reduction = "umap", group.by = "batch") + ggtitle("Cluster Results of Differ
```




```
# the plot which shows the cluster results of different clusters
DimPlot(integratedSamples, reduction = "umap") + ggtitle("Cluster Results")
```




```
# the plot which shows the cluster results of different clusters in different samples
DimPlot(integratedSamples, ncol = 3, reduction = "umap", split.by = "batch") + ggtitle("Cluster Results
```



Confusion matrix

```
celllabels_orig = c(simNor@colData@listData$Group, simMut@colData@listData$Group)

conf.mat<-table(Idents(integratedSamples), celllabels_orig)
print(conf.mat)
```

```
##    celllabels_orig
##       1    2    3
##   B 1499   17    0
##   A    1    0 1151
##   C   13  919    0
```

```
true.conf<-t(t(conf.mat)/apply(conf.mat,2,sum))
print(true.conf)
```

```
##    celllabels_orig
##                  1           2           3
##   B 0.9907468605 0.0181623932 0.0000000000
##   A 0.0006609385 0.0000000000 1.0000000000
##   C 0.0085922009 0.9818376068 0.0000000000
```

```
condition = integratedSamples@meta.data$condition
condNor<-Idents(integratedSamples)[condition == "Normal"];
condMut<-Idents(integratedSamples)[condition == "Mutate"];
```

Fisher's exact test

```
cell.count.mat<-cbind(table(condNor),table(condMut))
fisher.test(cell.count.mat)$p.value
```

```
## [1] 5.003823e-27
```

speckle

```
speckleData = data.frame(clusterRes = integratedSamples@active.ident, batch = integratedSamples$batch, 
  tibble::rownames_to_column("cell")
head(speckleData)
```

```
##       cell clusterRes batch condition
## 1 NorCell1          B  Nor1    Normal
## 2 NorCell2          C  Nor1    Normal
## 3 NorCell3          C  Nor1    Normal
## 4 NorCell4          A  Nor1    Normal
## 5 NorCell5          C  Nor1    Normal
## 6 NorCell6          A  Nor1    Normal
```

```
# clusters indicates the cluster results, sample indicates the biological replicates, group indicates t
propeller(clusters = speckleData$clusterRes, sample = speckleData$batch, group = speckleData$condition)
```

```
## group variable has 2 levels, t-tests will be performed
```

```
##   BaselineProp.clusters BaselineProp.Freq PropMean.Mutate PropMean.Normal
## C                     C         0.2588889       0.1866667       0.3311111
## B                     B         0.4211111       0.4955556       0.3466667
## A                     A         0.3200000       0.3177778       0.3222222
##   PropRatio Tstatistic      P.Value          FDR
## C 0.5637584  -7.260180 1.000620e-05 3.001861e-05
## B 1.4294872   6.616258 2.478573e-05 3.717860e-05
## A 0.9862069  -0.204631 8.412902e-01 8.412902e-01
```

```
# Plot cell type proportions
#plotCellTypeProps(clusters=speckleData$clusterRes, sample=speckleData$batch)
```

- DCATS

First, get the count tables for normal and mutate condtion. The count tables count numbers of cells in different clusters. This is `counts1` and `counts2`. The `similarity_mat` is calculated for all batches across different condtions. It is the true.conf we get before.

how to set the n_samples parameter?

```
countNor = table(batchNor, condNor)
countMut = table(batchMut, condMut)
dcats_fit(countNor, countMut, true.conf, n_samples = 3)
```

```
##   prop1_mean  prop1_std prop2_mean  prop2_std  coeff_mean  coeff_std
## B  0.3466667 0.02204793  0.4955556 0.02370732 -0.60698977 0.12173449
## A  0.3222222 0.02116951  0.3177778 0.01250926  0.74474102 0.12396285
## C  0.3311111 0.04302239  0.1866667 0.02185813  0.02043136 0.09227026
##   intecept_mean intecept_std        pvals
## B   -0.04148751   0.08216499 6.158563e-07
## A   -1.43206349   0.09126990 1.880855e-09
## C   -0.76485771   0.06735714 8.247580e-01
```

- diffcyt

originally used in analyzing differential abundance and differential states of high-dimensional cytometry data

- scDC