```r
# compcodeR.R
# R version 3.2.2 (2015-08-14)
# November 6, 2016. Mallory B. Lai.
# Reviewed by: TODO (Mallory B. Lai) : Find reviewer to proofread
# Practice using compcodeR package for simulating differential gene expression
# data.
#------------------------------------------------------------------------------
source("https://bioconductor.org/biocLite.R")
biocLite("compcodeR")
library(compcodeR)
library(baySeq)
library(edgeR)
library(data.table)
#------------------------------------------------------------------------------

if(require("parallel")) cl <- makeCluster(8) else cl <- NULL

##### Generate simulated data. #####

# Generate synthetic data.
sim <- generateSyntheticData(dataset = "D1", n.vars = 12500,
                                   samples.per.cond = 5, n.diffexp = 1250,
                                   repl.id = 1, seqdepth = 1e7,
                                   fraction.upregulated = 0.5,
                                   between.group.diffdisp = FALSE,
                                   filter.threshold.total = 1,
                                   filter.threshold.mediancpm = 0,
                                   fraction.non.overdispersed = 0,
                                   output.file = "simD1.rds")

# Note: sim@variable.annotations holds list of differentially expressed genes.

# Store differentially expressed count matrix in a "simGenes" matrix.
simGenes <- sim@count.matrix

# Store gene annotations in "varAnnotations" matrix.
varAnnotations <- sim@variable.annotations

# Store sample annotations in "sampAnnotations" matrix.
sampAnnotations <- sim@sample.annotations

##### baySeq analysis #####

# Create count data object.
countData <- new('countData', data = simGenes, replicates = sampAnnotations$condition,
                 groups = list(NDE = rep(1, length(sampAnnotations$condition)),
                               DE = sampAnnotations$condition))

# Get libsizes.
libsizes(countData) <- getLibsizes(countData, estimationType = 'QL', cl=cl)

# Get priors.
countData <- getPriors.NB(countData, samplesize =12500,
                                  equalDispersions = TRUE, estimation = 'QL', cl = cl)

# Get likelihoods.
countData <- getLikelihoods(countData, cl=cl, verbose = FALSE)

##### Compare results #####

# Find the top counts for differentially expressed genes.
de <- topCounts(countData, group = "DE", FDR = .05, number = 1300)

# Convert the top counts matrix into a data table, keeping only the rownames,
# Likelihood, False Discovery Rate, and Ordering.
de <- data.table(rownames(de), de$Likelihood, de$FDR.DE, de$ordering)

# Rename the columns appropriately.
colnames(de) <- c("Gene", "Likelihood", "FDR", "Ordering")

# Set a key on the de datatable for gene name.
setkey(de, Gene)

# Convert the variable annotations for Up-and-Downregulated genes into a datatable.
deAnnotations <- data.table(sim@variable.annotations$upregulation,
                     sim@variable.annotations$downregulation,
                     rownames(sim@variable.annotations))
# Rename the columns appropriately.
colnames(deAnnotations) <- c("Upregulated", "Downregulated", "Gene")

# Set a key on the deAnnotations datatable for gene name.
setkey(deAnnotations, Gene)

# Align gene annotations with differentially expressed genes with a table join.
shared <- deAnnotations[detab]
```

```r
# Calculate the proportion of top counts compared to actual number
# of differentially expressed genes.
dim(de)[1]/(sim@info.parameters$n.diffexp)

# Calculate the proportion of differentially expressed genes that
# were correctly identified as being up or downregulated.
(sum(shared$Upregulated==1 & shared$Ordering=="2>1")+
  sum(shared$Downregulated==1 & shared$Ordering=="1>2"))/dim(shared)[1]

# Calculate the proportion of correctly identified differentially expressed genes
# to the actual number of differentially expressed genes.
(sum(shared$Upregulated==1 & shared$Ordering=="2>1")+
  sum(shared$Downregulated==1 & shared$Ordering=="1>2"))/(sim@info.parameters$n.diffexp)

##########################################################################################
##########################################################################################
##########################################################################################
##########################################################################################
##########################################################################################

#### Now loop it ####

##### Generate simulated data. #####

# Create a matrix to store results and name the columns.
resultsMatrix <- matrix(data = NA, nrow = 50, ncol = 7)
colnames(resultsMatrix) <- c("Simulation", "NumberOfGenes", "DEgenes", "Upregulated",
                             "TopCounts/DE", "Correct/TopCounts", "Correct/DE")


for (i in 1:100)
{

# Store simulation number in results matrix.
resultsMatrix[i, 1] <- i

# Create bounded random values for synthetic data.
nv <- sample(c(8000:12500), 1) # Number of genes.
n.d <- runif(1, .05, .3) # Number of differentially expressed genes (DE genes).
up <- runif(1, .45, .65) # Fraction of upregulated genes.

# Store number of genes and DE genes in results matrix.
resultsMatrix[i, 2] <- nv
resultsMatrix[i, 3] <- floor(nv*n.d)
resultsMatrix[i, 4] <- up

# Generate synthetic data.
sim <- generateSyntheticData(dataset = "D1", n.vars = nv,
                             samples.per.cond = 5, n.diffexp = n.d*nv,
                             repl.id = 1, seqdepth = 1e7,
                             fraction.upregulated = up,
                             between.group.diffdisp = FALSE,
                             filter.threshold.total = 1,
                             filter.threshold.mediancpm = 0,
                             fraction.non.overdispersed = 0,
                             output.file = "simD1.rds")

# Note: sim@variable.annotations holds list of differentially expressed genes.

# Store differentially expressed count matrix in a "simGenes" matrix.
simGenes <- sim@count.matrix

# Store gene annotations in "varAnnotations" matrix.
varAnnotations <- sim@variable.annotations

# Store sample annotations in "sampAnnotations" matrix.
sampAnnotations <- sim@sample.annotations

##### baySeq analysis #####

# Create count data object.
countData <- new('countData', data = simGenes, replicates = sampAnnotations$condition,
                 groups = list(NDE = rep(1, length(sampAnnotations$condition)),
                               DE = sampAnnotations$condition))

# Get libsizes.
libsizes(countData) <- getLibsizes(countData, estimationType = 'edgeR', cl=cl)

# Specify prior density to be zero-inflated negative binomial distribution.
densityFunction(countData) <- ZINBDensity

# Get priors.
countData <- getPriors.NB(countData,  cl = cl)
# Get likelihoods.
countData <- getLikelihoods(countData, cl=cl, verbose = FALSE)

##### Compare results #####

# Find the top counts for differentially expressed genes.
de <- topCounts(countData, group = "DE", FDR = .05, number = 1300)

# Convert the top counts matrix into a data table, keeping only the rownames,
# Likelihood, False Discovery Rate, and Ordering.
de <- data.table(rownames(de), de$Likelihood, de$FDR.DE, de$ordering)

# Rename the columns appropriately.
colnames(de) <- c("Gene", "Likelihood", "FDR", "Ordering")

# Set a key on the de datatable for gene name.
setkey(de, Gene)

# Convert the variable annotations for Up-and-Downregulated genes into a datatable.
deAnnotations <- data.table(sim@variable.annotations$upregulation,
                            sim@variable.annotations$downregulation,
                            rownames(sim@variable.annotations))
# Rename the columns appropriately.
colnames(deAnnotations) <- c("Upregulated", "Downregulated", "Gene")

# Set a key on the deAnnotations datatable for gene name.
setkey(deAnnotations, Gene)

# Align gene annotations with differentially expressed genes with a table join.
shared <- deAnnotations[de]

# Calculate the proportion of top counts compared to actual number
# of differentially expressed genes.
resultsMatrix[i, 5] <- dim(de)[1]/(sim@info.parameters$n.diffexp)

# Calculate the proportion of differentially expressed genes that
# were correctly identified as being up or downregulated.
resultsMatrix[i, 6] <- (sum(shared$Upregulated==1 & shared$Ordering=="2>1")+
  sum(shared$Downregulated==1 & shared$Ordering=="1>2"))/dim(shared)[1]

# Calculate the proportion of correctly identified differentially expressed genes
# to the actual number of differentially expressed genes.
resultsMatrix[i, 7] <- (sum(shared$Upregulated==1 & shared$Ordering=="2>1")+
  sum(shared$Downregulated==1 & shared$Ordering=="1>2"))/(sim@info.parameters$n.diffexp)

}


write.csv(resultsMatrix,
          file = "C:/Users/Mallory/Documents/GitHub/Homework-Repo/compcodebaySeqZINBout2.csv")
```