Differential gene expression anlaysis with Clipper

Xinzhou Ge

2022-07-05

Contents

Here we demonstrate how to use `Clipper` as an add-on of DESeq2 or edgeR for DEG identification problems, especially when there are extra covariates. Here we used a synthetic single-cell dataset geneartaed by scDesign3 as an example. In this dataset, there are two covariates, cell type and batch, we are interested in the genes that are differentially expressed between the two cell types, and consider batch as an extra covariate. We first read the synthetic dataset:

```
library(edgeR)
library(DESeq2)
library(Clipper)
dataurl = 'http://jsb1data.stat.ucla.edu/repository/xinzhou/example_data.rds'
simudata = readRDS(url(dataurl))
# gene expression matrix
count =  simudata$data
# cell type of all cells (variable of interest)
cell_type = simudata$cell_type
# batch of all cells (extra covariate)
batchs = simudata$batchs
# The first 800 genes are true DEGs
trueDE = 1:800
```

We first define some functions, which are used in later analysis. Including the functions which use DESeq2 and edgeR to do DEG identification, a function to identify discoveires from pvalues, and a function to calculate false discovery proportion (FDP) and power.

```
############ Define a function to use edgeR to do DE analysis ############
## The inputs are dat: count matrix, condition          ##################
##############     conditions: the covariate of interest ##################
##############     covariates: the extra covariate       ##################
myedger = function(dat, conditions, covariates){
  y <- DGEList(counts=dat,group=conditions)
  y <- calcNormFactors(y)
  design <- model.matrix(~conditions+covariates)
  y <- estimateGLMRobustDisp(y,design)
  fit <- glmQLFit(y,design)
  qlf <- glmQLFTest(fit,coef=2)
  qlf_i = topTags(qlf, n = nrow(dat), p.value = 1)@.Data[[1]]
  qlf_i = qlf_i[match(rownames(dat), rownames(qlf_i)),]
  return(qlf_i)
}
############ Define a function to use DESeq2 to do DE analysis ############
## The inputs are dat: count matrix, condition          ##################
```

```
#############     conditions: the covariate of interest ##################
#############     covariates: the extra covariate       ##################

mydeseq2 = function(dat, conditions, covariates){
  rownames(dat) <- 1:nrow(dat)
  dds = DESeqDataSetFromMatrix(dat, colData=DataFrame(conditions,covariates),
                               design = ~covariates+conditions)
  dds <- DESeq(dds)
  res <- results(dds, alpha = 0.05)
  return(res)
}
#####  The function to find discoveries from a pvalue vector `pval' ########
#####  and a vector of target FDR thresholds `q',                   ########
#####  output is a list of discoveris for the thresholds            ########

find_discovery_wpval = function(pval, q){
  pval.adj = p.adjust(pval, method = 'BH')
  discovery_ls = lapply(q, function(q_i){
    discovery = which(pval.adj <= q_i)
  })
  return(discovery_ls)
}
######## The function to calculate false discovery proportion and power #########
######## by comparing the discoveries with the truth                    #########

compute_fdppow = function(discovery_ls, trueidx){
  sapply(discovery_ls, function(discovery){
    fdp = sum(!discovery %in% trueidx )/max(length(discovery),1)
    pow = sum(discovery %in% trueidx)/length(trueidx)
    return(c(fdp, pow))
  })
}
```

We then get the results of edgeR on the synthetic dataset.

```
########## Run edgeR on the simulated dataset #################
re_edgeR  <- myedger(count, conditions = cell_type, covariates = batchs)
# If you want to use DESeq2, just use the following command
# re_deseq2 <- mydeseq2(count+1, conditions = cell_type, covariates = batchs)
dis.edger= find_discovery_wpval(re_edgeR$PValue,q = 0.05)
compute_fdppow(dis.edger, trueDE)
```

Then, we run DESeq2 and edgeR on a null dataset, a null dataset is a dataset where the expressions of the genes are from the same distribution as the original dataset, and the covariate of interest (cell type in this example) does not have any effect on the expression. We can first use the pre-generatted null dataset by scDesign3.

```
count_null = simudata$nulldata
########## RunedgeR on the null dataset #################
re_edgeR_null  <- myedger(count_null, conditions = cell_type, covariates = batchs)
# We will use -log(p-value) as the input of Clipper, so we get the p-values from edegR
# p-values from the original dataset
p.e = re_edgeR$PValue
# p-values from the null dataset
p.e.null = re_edgeR_null$PValue
```

Next we will apply the `Clipper` method to combine the results from the original dataset and the null datset to control the false discovery rate. We first check whether the assumptions of Clipper is violated, that is, whether the input statsitics (negative log-transformed p-values here) from the two datasets (original and null) follows the same distribution for all the non-DEGs. If this assumption is violated, we will have an extra normalization step.

```
# We will test whether the assumption of Clipper is violated.
# Clipper requires that the input statsitics from the two datasets (original and null)
# follows the same distribution for all the non-DEGs.
# We look at the part that are smaller then the  median of the original dataset
# to see whether the two datasets have the same distribution for that part.
thres = median(-log(p.e))
dis1 = -log(p.e)[-log(p.e) < thres]
dis2 = -log(p.e.null)[-log(p.e.null) < thres]
# We use ks test to test whether these two distributions are the same.
ks.test(dis1, dis2)$p.value
# As the p-value is small, we reject the null, and consider normalize the
# statistics from the null dataset
# The normalization factor is defined as minimum of the ratios of 4 summary statistics
mfactor = min(summary(-log(p.e), na.rm = T)[2:5]/summary(-log(p.e.null), na.rm = T)[2:5])
# We repeat the assumption check and we see that the distributions are similar after normalization
norm.logp = -log(p.e.null)*mfactor
dis3 = norm.logp[norm.logp < thres]
ks.test(dis1, dis3)$p.value
# We now use Clipper to combine the p-values from the two datasets
re_Clipper = Clipper(-log(p.e), -log(p.e.null)*mfactor, analysis = "e", FDR = 0.05)
dis.clipper = re_Clipper$discoveries
compute_fdppow(dis.clipper, trueDE)
```

We can also use permutation to generate null dataset, here is an example of permutation null data.

```
# We permute the gene expression values, which have the same batch value.
# In this case, the cell type does not have effect at all.
cov_id = lapply(unique(batchs), function(batch){
  which(batchs == batch)
})
permute_count = apply(count, 1, function(a){
  b <- rep(0,length(a))
  for (i in 1:length(cov_id)){
    b[cov_id[[i]]] = sample(count[cov_id[[i]]])
  }
  return(b)
})
permute_count = t(permute_count)
# Run edgeR on the null dataset
re_edgeR_null  <- myedger(permute_count, conditions = cell_type, covariates = batchs)
p.e = re_edgeR$PValue
p.e.null = re_edgeR_null$PValue

######### We will test whether the assumption of Clipper is violated #########
thres = quantile(-log(p.e), 0.5)
dis1 = -log(p.e)[-log(p.e) < thres]
dis2 = -log(p.e.null)[-log(p.e.null) < thres]
ks.test(dis1, dis2)$p.value
# The two distributions are similar, so no nomalization is needed.
```

```r
re_Clipper = Clipper(-log(p.e), -log(p.e.null), analysis = "e", FDR = 0.05)
dis.clipper = re_Clipper$discoveries
compute_fdppow(dis.clipper, trueDE)
```