NovaSeq

Tamar Sofer & Nuzulul Kurniansyah

12/29/2020

Contents

Introduction

Here we demonstrate how to perform association analyses of continuous phenotypes using NovaSeq with RNA-seq data based on the pipeline proposed in the manuscript "Benchmarking Association Analyses of Continuous Exposures with RNA-seq in Observational Studies".

Installation and require packages

To install, open R and type:

```r
library("devtools")
install_github("nkurniansyah/NovaSeq")
library(NovaSeq)
```

Novaseq require external packages from CRAN (dplyr) and Bioconductor(qvalue)

```
install.packages("dplyr")

BiocManager::install("qvalue")
```

Load all the packages

```
library(dplyr)
library(qvalue)
```

Load example data

Load raw gene counts matrix

First we load the transcripts, where were obtained from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE151243.
Note: we reformatted the transcripts matrix into desire form and embeded into NovaSeq package.

```
data(rnaseq_count_matrix)
rnaseq_count_matrix[1:5,1:5]
```

```
##                 10L_S26_L006_R1_001 10N_S15_L003_R1_001 11L_S35_L007_R1_001
## ENSG00000000003                 446                 644                 525
## ENSG00000000005                   5                  19                   2
## ENSG00000000419                 883                1058                 609
## ENSG00000000457                 790                1009                 619
## ENSG00000000460                 206                 289                 272
##                 11N_S25_L005_R1_001 12L_S14_L003_R1_001
## ENSG00000000003                 701                 572
## ENSG00000000005                  64                  25
## ENSG00000000419                 547                 576
## ENSG00000000457                 887                 650
## ENSG00000000460                 214                 334
```

Load simulated phenotypes

We simulated in advance a data.frame phenotypes.

```
data(phenotype)
head(phenotype)
```

```
##                     Age Sex  Trait.1  Trait.2
## 10L_S26_L006_R1_001  18   0 16.06608 15.58321
## 10N_S15_L003_R1_001  19   0 21.20045 20.61345
## 11L_S35_L007_R1_001  20   0 14.44867 13.88567
## 11N_S25_L005_R1_001  21   0 35.89606 34.84859
## 12L_S14_L003_R1_001  22   0 24.09078 24.42536
## 12N_S27_L006_R1_001  21   0 29.61045 26.62854
```

We define the trait of interest to study as an exposure associated with genes It has to be a column name in the phenotype data.frame.

```
trait <- "Trait.1"
```

We will adjust our analysis to the simulated covariates Age and Sex. The covriates have to correspond to column names in the phenotype data.frame. In the analysis, we will use a string defining the regression model (just the covariates part of it), so we define it here:

```r
covariates_string <- "Age,Sex"
```

Note that we can also define the string to be "Age,as.factor(Sex)", or use interaction terms, like one would use in regression functions in R.

Match the (simulated) individuals between the phenotype and the RNA-seq count matrix. Make sure the IDs overlap.

```r
IDs_both <- intersect(rownames(phenotype), colnames(rnaseq_count_matrix))
rnaseq_matrix <- rnaseq_count_matrix[, IDs_both]
phenotypes <- phenotype[match(IDs_both,rownames(phenotype)),]
```

Normalize the RNA-seq dataset

We use median normalization in NovaSeq. However, users can use diffrent normalization method using diffrent packages, for example: estimateSizeFator(DESeq2) or TMM(edgeR).
Here we show how each of these is used. We move forward in this tutorial with the median normalization, because it was used in the manuscript. However, there are no downstream differences in how the methods are applied once the data is normalized.

Median normalization

```r

median_norm<- median_normalization(rnaseq_matrix)
```

estimateSizeFactor

This method implemented in DESeq2.

```r
 BiocManager::install("DEseq2")
 library(DESeq2)

  des_matrix <- DESeqDataSetFromMatrix(countData = rnaseq_matrix,
                                       colData = phenotypes,
                                       design = ~Age+Sex+Trait.1)

  des_matrix <- estimateSizeFactors(des_matrix)

  SizeFactor_norm <- counts(des_matrix, normalized=TRUE)
```

TMM (Trimmed Mean of M-values)

This method implemented in edgeR

```r
 BiocManager::install("edgeR")
 library(edgeR)
 counts <- DGEList(rnaseq_matrix)

```

```
dgList<- calcNormFactors(counts, method = "TMM")

TMM_norm<- cpm(dgList)
```

Filtering transcripts

Remove lowly express gene counts

```
clean_count_matrix <- apply_filters(count_matrix = median_norm,
                                   median_min = 1,
                                   expression_sum_min = 10,
                                   max_min = 10,
                                   range_min = 5,
                                   prop_zero_max = 0.5)
```

```
## applying filters on a transcript count matrix of 58051 transcripts, across 40 individuals

## Computing transtripts characteristics...

## Appying filters...

## There are 23987 transcripts with median
##                    value lower than 1

## There are 14190 transcripts with expression sum
##                    value lower than 10

## There are 22297 transcripts with maximum expression
##                    value lower than 10

## There are 17188 transcripts with maximum
##                    expression range value lower than 5

## There are 21923 transcripts with propotion
##                    of zero counts higher than 0.5

## Removing 24834 unique transcripts not passing requested filters
```

After filtering gene counts, there are 33217 remaining for differential expression analysis.

Perform differential expression analysis

We show how we perform differential expression analysis on all transcripts using emprical p-value(quantile empirical p-values and Storey empirical p-values). In order to generate p-values under the null, we create a "residual permuted" trait 100 times and perform differential expression analysis, and use the resulting p-values/ z score as our null p-values/ z-score. (see manuscript).

```
set.seed(12)

storey_emp<-lm_count_mat_emp_pval(clean_count_matrix, pheno=phenotypes, trait, covariates_string,
                                  n_permute=100, gene_IDs=NULL,log_transform = "log_replace_half_min",
                                  stat_type="z_score", empirical_type = "storey",
                                  t_df = NULL, family="gaussian")
```

```
## Performing residual permutation to generate permuted trait...
## performing differential expression analysis on 100 permuted traits

## Computing empirical p-values

## Run storey empirical p-values using z_score
```

```r
head(storey_emp)
```

```
##            geneID        beta          se     t_stat      p_value      fdr_bh
## 1 ENSG00000000003  0.02140173 0.01055381  2.0278677 5.002424e-02 0.296579900
## 2 ENSG00000000005  0.16670092 0.02639112  6.3165534 2.629257e-07 0.004366802
## 3 ENSG00000000419  0.00588087 0.01466879  0.4009102 6.908559e-01 0.904505585
## 4 ENSG00000000457  0.00800729 0.01044979  0.7662634 4.485152e-01 0.788532027
## 5 ENSG00000000460 -0.01315765 0.01333808 -0.9864726 3.304853e-01 0.702422623
## 6 ENSG00000000938 -0.06909230 0.02051793 -3.3674103 1.817907e-03 0.068018905
##      z_score    emp_pvals bh_emp_pvals
## 1  1.9597567 2.399735e-02  0.268933873
## 2  5.1482516 3.010507e-07  0.003333333
## 3  0.3976939 3.439392e-01  0.861131379
## 4  0.7578928 2.232532e-01  0.739988030
## 5 -0.9731369 8.272887e-01  0.999997592
## 6 -3.1184731 9.986149e-01  0.999997592
```

Perform differential expression analysis using multiple exposure

We show how we perform differential expression analysis on all transcripts using emprical p-value(quantile empirical p-values and Storey empirical p-values) using multiple exposure.

```r
set.seed(12)

storey_emp_multi<-lm_mult_count_mat_emp_pval(clean_count_matrix, pheno=phenotypes, traits="Trait.1,Trai
                                             covariates_string,n_permute=100, gene_IDs=NULL,
                                             log_transform = "log_replace_half_min",
                                             stat_type="z_score", empirical_type = "storey",
                                             t_df = NULL, family="gaussian")
```

```
## Performing residual permutation to generate permuted trait...

## performing differential expression analysis on 100 permuted traits

## Computing empirical p-values

## Run storey empirical p-values using z_score
```

```r
head(storey_emp_multi)
```

```
##                          geneID beta.Trait.1 beta.Trait.2     t_stat
## ENSG00000000003 ENSG00000000003 -0.095383807   0.11755276  6.4796146
## ENSG00000000005 ENSG00000000005  0.381867680  -0.21658030 41.2674590
## ENSG00000000419 ENSG00000000419 -0.145930094   0.15280829  2.0975976
## ENSG00000000457 ENSG00000000457 -0.039683211   0.04800380  0.9365473
## ENSG00000000460 ENSG00000000460 -0.115913899   0.10343131  2.0195879
## ENSG00000000938 ENSG00000000938 -0.005696896  -0.06381189 11.2401449
##                        p_value       fdr_bh    z_score   emp_pvals bh_emp_pvals
## ENSG00000000003 3.917144e-02 2.651878e-01 -2.0623814 0.969129060    0.9999991
## ENSG00000000005 1.093667e-09 3.632834e-05 -6.0951034 0.999999097    0.9999991
## ENSG00000000419 3.503583e-01 6.809041e-01 -0.9338944 0.800943794    0.9999991
## ENSG00000000457 6.260822e-01 8.451406e-01 -0.4872486 0.663542764    0.9999991
## ENSG00000000460 3.642940e-01 6.923092e-01 -0.9072133 0.793930518    0.9999991
## ENSG00000000938 3.624378e-03 9.232925e-02  2.9091285 0.003130626    0.5136453
```

Perform differential expression analysis using permutation

We show how we perform differential expression analysis on selected transcripts using permutation method. we suggested to run 100000 permutation for single genes.

```r
set.seed(12)

gene_names<-sample(rownames(clean_count_matrix),5)

perm_res<- lm_count_mat_perm_pval(count_matrix=clean_count_matrix, pheno=phenotypes, trait, covariates_s
                                  n_permute=100000,
                                  gene_IDs=gene_names,
                                  log_transform = "log_replace_half_min",
                                  seed = NULL,
                                  family="gaussian")
```

```
## Filtering count_matrix to genes : ENSG00000211888 ENSG00000100416 ENSG00000039650 ENSG00000249700 ENS
```

```
## Performing residual permutation to generate permuted trait...
```

```r
perm_res
```

```
##            geneID         beta         se     t_stat    p_value     fdr_bh
## 3 ENSG00000211888 -0.100704705 0.03780596 -2.6637255 0.01148709 0.02871773
## 2 ENSG00000100416 -0.011504192 0.01349350 -0.8525725 0.39953313 0.66588855
## 1 ENSG00000039650 -0.031589560 0.01114510 -2.8343890 0.00748205 0.02871773
## 4 ENSG00000249700 -0.007725497 0.01928230 -0.4006523 0.69104411 0.69104411
## 5 ENSG00000264932  0.021374024 0.03900637  0.5479623 0.58709966 0.69104411
##      z_score perm_pval
## 3 -2.5275212   0.01209
## 2 -0.8424553   0.40138
## 1 -2.6745908   0.00615
## 4 -0.3974385   0.69239
## 5  0.5430438   0.58474
```