Olivia R package

Tamar Sofer & Nuzulul Kurniansyah

2/18/2021

Contents

Introduction

Here we demonstrate how to perform association analyses of continuous phenotypes using the Olivia package with RNA-seq data based on the pipeline proposed in the manuscript `Benchmarking Association Analyses of Continuous Exposures with RNA-seq in Observational Studies` https://www.biorxiv.org/content/10.1101/2021.02.12.430989v1.abstract.

Installation and require packages

To install, open R and type:

```
library("devtools")
install_github("nkurniansyah/Olivia")
library(Olivia)
```

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

```
install.packages("dplyr")
```

Load packages

```r
library(dplyr)
```

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 desired form and embedded them into Olivia package.

```r
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 of phenotypes.

```r
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. The trait/phenotype has to correspond to a column name in the phenotype data.frame.

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

We will adjust our analysis to the simulated covariates Age and Sex. The covariates 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 there are matching IDs.

```
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 Olivia to reduce package dependencies. However, users can use different normalization method using different packages, for example: estimateSizeFator(DESeq2) or TMM(edgeR). There are no downstream differences in how the methods are applied once the data is normalized.

Median normalization

```
median_norm<- median_normalization(rnaseq_matrix)
```

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 empirical p-value (quantile empirical p-values). To generate p-values under the null, we create a `residual permuted` trait 100 times, perform differential expression analysis, and use the resulting p-values as our null p-values. However, users also can implement Storey empirical p-value using test statistcs, as these are referred to in the manuscript.

```
set.seed(12)

quantile_emp<-lm_count_mat_emp_pval(clean_count_matrix, pheno=phenotypes, trait, covariates_string,
                                  n_permute=100,log_transform = "log_replace_half_min",
                                  outcome_type ="continous",gene_IDs=NULL)
```

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

## performing differential expression analysis on 100 permuted traits

## Computing quantile empirical p-values
```

```
tophits<-quantile_emp[which(quantile_emp$bh_emp_pvals< 0.05),]
head(tophits)
```

```
##              geneID        beta          se    t_stat t_stat_df      p_value
## 2   ENSG00000000005  0.16670092 0.026391120  6.316553        36 2.629257e-07
## 231 ENSG00000009709  0.12205262 0.028668757  4.257339        36 1.413253e-04
## 248 ENSG00000010278  0.03556122 0.007342435  4.843246        36 2.423317e-05
## 328 ENSG00000013503  0.02283551 0.006028366  3.788009        36 5.569283e-04
## 337 ENSG00000013810 -0.05450117 0.013567542 -4.017026        36 2.868601e-04
## 369 ENSG00000018280 -0.11359856 0.029790212 -3.813285        36 5.179180e-04
##          fdr_bh    emp_pvals bh_emp_pvals
## 2   0.004366802 1.505253e-07   0.00250000
## 231 0.036960240 1.207213e-04   0.03351724
## 248 0.032198126 1.746094e-05   0.02320000
## 328 0.047191580 5.870488e-04   0.04974490
## 337 0.039758853 2.832887e-04   0.03933884
## 369 0.046122473 5.403859e-04   0.04812332
```

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) using multiple exposure.

```
set.seed(12)

quantile_emp_multi<-lm_mult_count_mat_emp_pval(clean_count_matrix, pheno=phenotypes, traits=c("Trait.1"
                                              covariates_string,n_permute=100, gene_IDs=NULL,
                                              log_transform = "log_replace_half_min", outcome_type="cont
```

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

## performing differential expression analysis on 100 permuted traits

## Computing quantile empirical p-values
```

```
quantile_emp_multi<-quantile_emp_multi[quantile_emp_multi$bh_emp_pvals< 0.05,]

quantile_emp_multi
```

```
##                geneID beta_Trait.1 beta_Trait.2 chisq_stat chisq_stat_df
## 2    ENSG00000000005    0.3818677  -0.21658030   41.26746             2
## 8164 ENSG00000144821    0.1339460  -0.03822377   39.15829             2
##           p_value       fdr_bh    emp_pvals bh_emp_pvals
## 2    1.093667e-09 3.632834e-05 1.204203e-06         0.02
## 8164 3.139679e-09 5.214536e-05 1.204203e-06         0.02
```

Perform differential expression analysis using permutation

When testing only a handful of genes, we may not want to perform transcriptome-wide association analysis. Therefore, empirical p-values using the quantile or Storey's approach cannot be computed (not enough tests to generate the null distribution). Additionally, we permute specific genes many times. Here we show how to perform differential expression analysis on selected transcripts when computing a permutation p-value for each gene based on permutations for this gene only. We suggest running 100000 permutations.

```
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_
                                 n_permute=100000,
                                 gene_IDs=gene_names,
                                 log_transform = "log_replace_half_min",
                                 seed = NULL,
                                 outcome_type ="continous")
```

```
## Filtering count_matrix to genes : ENSG00000211888 ENSG00000100416 ENSG00000039650 ENSG00000249700 ENS
## Performing residual permutation to generate permuted trait...
```

```
perm_res
```

```
##            geneID         beta         se     t_stat t_stat_df    p_value
## 3 ENSG00000211888 -0.100704705 0.03780596 -2.6637255        36 0.01148709
## 2 ENSG00000100416 -0.011504192 0.01349350 -0.8525725        36 0.39953313
## 1 ENSG00000039650 -0.031589560 0.01114510 -2.8343890        36 0.00748205
## 4 ENSG00000249700 -0.007725497 0.01928230 -0.4006523        36 0.69104411
## 5 ENSG00000264932  0.021374024 0.03900637  0.5479623        36 0.58709966
##   perm_pval
## 3   0.01164
## 2   0.40085
## 1   0.00621
## 4   0.69280
## 5   0.58565
```