User's Guide for Resampling Package

Coraline Qu

2017-01-05

Contents

1 Introduction

1.1 Objective

In order to study the effect of replicates via performing gene differential expression analyses to htseq counts data, we need to resample the original data set to construct a new truth set for a more accurate and reasonable result. Further, resampling the new truth set gives different sample subsets which can be used for subgroup comparison and analysis.

1.2 Description

Package *Resampling* provides a two-step resampling of corresponding data sets in order to generate new truth set and perform further subgroup gene differential expression analyses. It mainly consists of two parts:

- first-step resampling: `rang()`
- second-step resampling: `card`, `subrang()`

The first-step resampling is realized via function `rang()`. This function returns a sample matrix by randomly generating a desired number of sample indices for certain times from original data set. New truth sets can be constructed through this sample matrix. The second-step resampling is implemented by functions `card` and `subrang()`. `card` imports a sample matrix resultting from `rang()` and exports a list containing vectors with the information of sample partitions "liposarcoma" and "leiomyosarcoma" of the new truth sets, including

their cardinalities. Note that `card` is only necessary when we deal with TCGA-sarc and it has to be executed before `subrang()`. The other function `subrang()` returns an index matrix which corresponds to a group of subsets of one new truth set. These subsets can be used in further differential expression analyses.

2 Standard workflow

First of all, package *Resampling* needs to be installed. You can just use `install_github("Coraline66/Resampling")`. As `install_github` is a function from `devtools`, this package should be loaded before the installation.

2.1 First-step resampling

The first-step resampling is utilized for the construction of new truth set through function `rang(SEED, N, M, n)`. There are four parameters in `rang()`:

- `SEED`: an integer used as a seed in `set.seed()`.
- `N`: the number of iterations(times), positive integer.
- `M`: a positive integer giving the range for resampling, i.e., length of the complete set of indices of the samples. In our case, `M` should equals `57` for data type TCGA-luad, `100` for TCGA-brca and `162` for TCGA-sarc.
- `n`: a non-negative integer giving the number of indices chosen from `M`. `n` is exactly the cardinality of the new truth set if we choose TCGA-sarc, while `n` is half the cardinality of the new truth set if we choose TCGA-luad or TCGA-brca. `rang()` returns an $N \times n$ matrix that each row consists of the indices of samples in the new truth set for each iteration. Here shows the basic r code of first-step resampling.

```
library(devtools)
library(Resampling)
sam <- rang(20161220, 10, 100, 90)
```

In this example, 10 new truth sets are generated for TCGA-brca and each truth set is composed of 90 pairs of samples(180 samples). The indices of these samples are stored in matrix `sam`, row-wisely.

2.2 Second-step resampling

Function `subrang()` together with `card()` achieve the second-step resampling and should only be invoked after the first-step resampling because they use the result matrix of `rang()` as an argument. Function `card()` has one argument `sam` which corresponds to the matrix above and it returns a list comprised of four vectors:

- `DL`: a vector of lists giving the sample indices of "liposarcoma" in matrix `sam`. `DL[[j]]` corresponds to the `jth` row of `sam`.
- `LM`: a vector of lists giving the sample indices of "leiomyosarcoma" in matrix `sam`. `LM[[j]]` corresponds to the `jth` row of `sam`.
- `dl`: a numeric vector of the cardinalities of sample condition "liposarcoma" in matrix `sam`. `dl[j]` corresponds to the `jth` row of `sam`.
- `lm`: a numeric vector of the cardinalities of sample condition "leiomyosarcoma" in matrix `sam`. `lm[j]` corresponds to the `jth` row of `sam`.

Arguments involved in `subrang()` are listed as follows.

- `SEED`: an integer used as a seed in `set.seed()`. It differs from the value used in `rang()`.

- `N`: the number of iterations(times), positive integer.
- `n`: a non-negative integer giving the number of indices chosen from the new truth set. Here the generated sample subset has a cardinality of `2*n`. `n` should be no more than `dim(sam)[2]` for TCGA-luad or TCGA-brca. As for TCGA-sarc, since the new truth set is composed of two sample conditions: liposarcoma and leiomyosarcoma, `n` should not exceed the smaller cardinality of these two conditions.
- `sam`: matrix that is comprised of the indices of new truth sets. It's returned from `rang()`.
- `j`: a positive integer giving the row index of matrix `sam` to choose from. `j` should be no larger than the number of rows of `sam`.
- `type`: character indicating the type of data, e.g., "luad", "brca", "sarc". Note that double quotes are required.

`subrang()` returns an $N \times n$ index matrix representing a group of subsets of one new truth set. Note that parameter `j` determines which new truth set obtained from `rang()` is involved in `subrang()`. Each row of the returned matrix corresponds to the generated sample subsets for each iteration. The basic steps of second-step resampling can be shown as follows.

```
library(devtools)
library(Resampling)
sam <- rang(20161220, 10, 100, 90)
samp <- subrang(20161222, 20, 5, 2, sam, "brca")
```

We use the same input in `rang()` as the preceding example and get `sam` as a 10×90 matrix. As `N`=20, `n`=5 and `j`=2, the new truth set is generated from the second row of `sam` and 20 subsets of 5 pairs of samples are constructed after the execution of `subrang()`. The result is stored in `samp`, a 20×5 matrix.

3 Examples

In this section, some examples with complete details are provided for users to get familiar with package "Resampling".

3.1 Example I

Here is the r code for the first example:

```
library(parallel)
library(Resampling)
sam <- rang(20161218, 100, 57, 50)
no_cores <- detectCores() - 1
cl <- makeCluster(no_cores)
clusterExport(cl=cl, c("sam", "subrang"), envir=environment())
k <- c(3, 5, 10, 20, 40)
samp <- parLapply(cl, k,
                  function(k) {
                      subrang(20161219, 50, k, 1, sam, "luad")
                  })
stopCluster(cl)
```

In this example, our goal is to resample the data set of TCGA-luad. First, we need to apply the first-step resampling to the original sample files to attain new truth sets. There are `57` pairs of samples, so `M` should be equal to `57`. `100` new truth sets are generated and stored in `sam`, each of which consists of `50` pairs of samples.

In the process of second-step resampling, `3, 5, 10, 20, 40` pairs of samples are picked from the new truth set generated by the first row of matrix `sam` for `50` iterations. The results are stored in `samp(samp[[i]], i=1, 2, 3, 4, 5)` for further analysis.

3.2 Example II

The second example is regarding the data set of TCGA-brca.

```
library(parallel)
library(Resampling)
sam <- rang(20161220, 50, 100, 80)
no_cores <- detectCores() - 1
cl <- makeCluster(no_cores)
clusterExport(cl=cl, c("sam", "subrang"), envir=environment())
k <- c(3, 5, 10, 20, 50, 80)
samp <- parLapply(cl, k,
                  function(k) {
                      subrang(20161221, 100, k, 10, sam, "brca")
                  })
stopCluster(cl)
```

During the process of first-step resampling, `80` pairs are randomly chosen 50 times out of `100` pairs of samples to build the new truth sets. Parallel processing is performed in the second-step resampling as we pick six subsets with `3, 5, 10, 20, 50, 80` pairs of samples from the new truth set for `100` iterations. Note that under our assumptions, the new truth set is generated by the `10th` row of matrix `sam`. The results for the second-step resampling can be found in the list `samp(samp[[i]], i=1, 2, 3, 4, 5)`.

3.3 Example III

Finally, we shall give an example to show how to perform "resampling" to the data set of TCGA-sarc. This is slightly different from the two cases above because the numbers of "liposarcoma" and "leiomyosarcoma" are not equal and we have to figure out the exact numbers for these two conditions before resampling.

```
library(parallel)
library(Resampling)
sam <- rang(20161222, 10, 162, 140)
len <- card(sam=sam)
len$dl
```

```
##  [1] 53 51 51 48 53 50 52 48 52 55
```

```
len$lm
```

```
##  [1] 87 89 89 92 87 90 88 92 88 85
```

```
no_cores <- detectCores() - 1
cl <- makeCluster(no_cores)
clusterExport(cl=cl, c("sam", "subrang", "card"), envir=environment())
k <- c(3, 5, 10, 20, 30, 40, 50)
samp <- parLapply(cl, k,
```

```
                    function(k) {
                        subrang(20161223, 20, k, 5, sam, "sarc")
                    })
stopCluster(cl)
```

In the process of first-step resampling, parameters `N` and `n` are equal to `10` and `140`, respectively. Then `sam` is a 10×140 matrix consisted of the new truth sets. In order to find out the partition of each truth set, we use function `card` on `sam` and save the results in `len`. `len$dl` and `len$lm` represent the numbers of samples for "liposarcoma" and "leiomyosarcoma" in each new truth set, respectively. Since `len$dl[5]=53`, `len$lm[5]=87`, the maximum pairs of samples we could generate from this new truth set are 53. The left can be done by following the same procedure as we did for TCGA-luad and TCGA-brca. `3, 5, 10, 20, 30, 40, 50` pairs of samples are picked from the new truth set generated by the fifth row of matrix `sam` for `20` iterations. The results are saved in list `samp(samp[[i]], i=1, 2, 3, 4, 5, 6, 7)` for further analysis.

4 Session Info

- R version 3.3.1 (2016-06-21)
- Platform: x86_64-apple-darwin13.4.0 (64-bit)
- Locale:
  - LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C
  - LC_TIME=en_US.UTF-8, LC_COLLATE=en_US.UTF-8
  - LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8
- Base packages: devtools, parallel