DupChecker: a bioconductor package for checking high-throughput genomic data redundancy in meta-analysis

Quanhu Sheng*, Yu Shyr, Xi Chen

Center for Quantitative Sciences, Vanderbilt University, Nashville, USA

*shengqh (at) gmail.com

May 2, 2019

Abstract

Meta-analysis has become a popular approach for high-throughput genomic data analysis because it often can significantly increase power to detect biological signals or patterns in datasets. However, when using public-available databases for meta-analysis, duplication of samples is an often encountered problem, especially for gene expression data. Not removing duplicates could lead false positive finding, misleading clustering pattern or model over-fitting issue, etc in the subsequent data analysis.

We developed a Bioconductor package Dupchecker that efficiently identifies duplicated samples by generating MD5 fingerprints for raw data. A real data example was demonstrated to show the usage and output of the package.

Researchers may not pay enough attention to checking and removing duplicated samples, and then data contamination could make the results or conclusions from meta-analysis questionable. We suggest applying DupChecker to examine all gene expression data sets before any data analysis step.

In this vignette, we demonstrate the application of DupChecker as a package for checking high-throughput genomic data redundancy in meta-analysis. DupChecker can download the GEO/ArrayExpress

raw data files from EBI/ncbi ftp server, extract individual data files, calculate MD5 fingerprint for each data file and validate the redundancy of those data files.

DupChecker version: 1.22.0

If you use DupChecker in published research, please cite:

Quanhu Sheng, Yu Shyr, Xi Chen.: DupChecker: a bioconductor package for checking high-throu
data redundancy in meta-analysis. BMC bioinformatics 2014, 15:323.

Contents

1 Introduction

Meta-analysis has become a popular approach for high-throughput genomic data analysis because it often can significantly increase power to detect biological signals or patterns in datasets. However, when using public-available databases for meta-analysis, duplication of samples is an often encountered problem, especially for gene expression data. Not removing duplicates would make study results questionable. We developed a package DupChecker that efficiently identifies duplicated samples by generating MD5 fingerprints for individual data file.

2 Standard workflow

2.1 Quick start

Here we show the most basic steps for a validation procedure. You need to create a target directory used to store the data. In order to illustrate the procedure, here we used two very small datasets without redundant CEL files. You may also try a real example with redundancy at example 2.5.

```
library(DupChecker)

## Create a "DupChecker" directory under temporary directory
rootDir<-paste0(dirname(tempdir()), "/DupChecker")
dir.create(rootDir, showWarnings = FALSE)
message(paste0("Downloading data to  ", rootDir, " ..."))

geoDownload(datasets=c("GSE1478"), targetDir=rootDir)

##   dataset count
## 1 GSE1478     8

arrayExpressDownload(datasets=c("E-MEXP-3872"), targetDir=rootDir)

##       dataset count
## 1 E-MEXP-3872     6

datafile<-buildFileTable(rootDir=rootDir, filePattern="cel$")
result<-validateFile(datafile)
if(result$hasdup){
  duptable<-result$duptable
  write.csv(duptable, file=paste0(rootDir, "/duptable.csv"))
}
```

2.2 GEO/ArrayExpress data download

Firstly, function geoDownload/arrayExpressDownload will download raw data from ncbi/EBI ftp server based on datasets user provided. Once the compressed raw data is downloaded, individual data files will be extracted from

compressed raw data. Some of the data files may have been compressed. Since the files compressed from same data file by different softwares will have different MD5 figureprints, we will decompress those compressed data files and validate the file redundancy using decompressed data files.

```
datatable<-geoDownload(datasets = c("GSE1478"), targetDir=rootDir)
datatable

##   dataset count
## 1 GSE1478    10

datatable<-arrayExpressDownload(datasets=c("E-MEXP-3872"), targetDir=rootDir)
datatable

##       dataset count
## 1 E-MEXP-3872     7
```

The datatable is a data frame containing dataset name and how many files in that dataset.

There are two possible situations that you may want to download and decompress the data using external tools. 1) The download or decompress cost too much time in R environment; 2) The internal tool in R may download imcomplete or interrupted file for huge file which will cause the "untar" or "gunzip" command fails.

2.3 Build file table

Secondly, function buildFileTable will try to find all files (default) or expected files matching filePattern parameter in the subdirectories of root directory. The result data frame contains two columns, dataset (directory name) and filename. Here, rootDir can also be an array of directories. In following example, we just focused on AffyMetrix CEL file only.

```
datafile<-buildFileTable(rootDir=rootDir, filePattern="cel$")
```

2.4 Validate file redundancy

The function validateFile will calculate MD5 fingerprint for each file in table and then check to see if any two files have same MD5 fingerprint. The files

with same fingerprint will be treated as duplication. The function will return a table contains all duplicated files and datasets.

```
result<-validateFile(datafile)
if(result$hasdup){
  duptable<-result$duptable
  write.csv(duptable, file=paste0(rootDir, "/duptable.csv"))
}
```

2.5 Real example

There are three colon cancer related datasets (GSE14333, GSE13067 and GSE17538) containing seriously data redundancy problem. User may run following codes to do the validation. It may cost a few miniutes to a few hours based on the network bindwidth and the computer performance.

```
library(DupChecker)

rootDir<-paste0(dirname(tempdir()), "/DupChecker_RealExample")
dir.create(rootDir, showWarnings = FALSE)

geoDownload(datasets = c("GSE14333", "GSE13067",
                         "GSE17538"), targetDir=rootDir)
datafile<-buildFileTable(rootDir=rootDir, filePattern="cel$")
result<-validateFile(datafile)
if(result$hasdup){
  duptable<-result$duptable
  write.csv(duptable, file=paste0(rootDir, "/duptable.csv"))
}
```

Table 1 illustrated the duplication between those three datasets. GSE13067 [64/74] means 64 of 74 CEL files in GSE13067 dataset were duplicated in other two datasets.

Table 1: Part of duplication table of three datasets

MD5	GSE13067[64/74]	GSE14333[231/290]	GSE17538[167/244]
001ddd757f185561c9ff9b4e95563372		GSM358397.CEL	GSM437169.CEL
00b2e2290a924fc2d67b40c097687404		GSM358503.CEL	GSM437210.CEL
012ed9083b8f1b2ae828af44dbab29f0	GSM327335.CEL	GSM358620.CEL	
023c4e4f9ebfc09b838a22f2a7bdaa59		GSM358441.CEL	GSM437117.CEL

3 Discussion

We illustrated the application using gene expression data, but DupChecker package can also be applied to other types of high-throughput genomic data including next-generation sequencing data.

4 Session Info

- R version 3.6.0 (2019-04-26), `x86_64-apple-darwin15.6.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Running under: `OS X El Capitan 10.11.6`
- Matrix products: default
- BLAS:
  `/Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib`
- LAPACK:
  `/Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: DupChecker 1.22.0
- Loaded via a namespace (and not attached): R.methodsS3 1.7.1, R.oo 1.22.0, R.utils 2.8.0, RCurl 1.95-4.12, bitops 1.0-6, compiler 3.6.0, evaluate 0.13, highr 0.8, knitr 1.22, magrittr 1.5, stringi 1.4.3, stringr 1.4.0, tools 3.6.0, xfun 0.6