

# Package ‘DupChecker’

October 16, 2018

**Type** Package

**Title** a package for checking high-throughput genomic data redundancy  
in meta-analysis

**Version** 1.18.0

**Date** 2014-10-07

**Author** Quanhu Sheng, Yu Shyr, Xi Chen

**Maintainer** ``Quanhu SHENG" <shengqh@gmail.com>

**Description** 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 Bioconductor package DupChecker that efficiently identifies duplicated samples by generating MD5 fingerprints for raw data.

**License** GPL (>= 2)

**LazyLoad** yes

**Imports** tools, R.utils, RCurl

**biocViews** Preprocessing

**VignetteBuilder** knitr

**Roxygen** list(wrap = FALSE)

**Suggests** knitr

**BuildVignettes** yes

**git\_url** <https://git.bioconductor.org/packages/DupChecker>

**git\_branch** RELEASE\_3\_7

**git\_last\_commit** 7471a44

**git\_last\_commit\_date** 2018-04-30

**Date/Publication** 2018-10-15

## R topics documented:

arrayExpressDownload . . . . .	2
buildFileTable . . . . .	2
geoDownload . . . . .	3
validateFile . . . . .	4

**Index****5**


---

arrayExpressDownload    *arrayExpressDownload*


---

**Description**

The function downloads array express raw data from EBI ftp server based on datasets user provided. Once the compressed raw data is downloaded, individual target file will be extracted from compressed raw data. The dataset/count table will be returned.

**Usage**

```
arrayExpressDownload(datasets, targetDir = getwd(), filePattern = NULL,
  unzip = "internal", overwrite = FALSE)
```

**Arguments**

datasets	the dataset names, for example: c("E-TABM-43", "E-TABM-158")
targetDir	the target directory to store the datasets
filePattern	the file pattern of the expected data file extracted from gzipped file
unzip	the path to the command to be used in unzip function
overwrite	If TRUE, overwrite existing files, otherwise ignore such files.

**Value**

a data frame containing dataset and how many expected data files in that dataset

**Examples**

```
#download three datasets from ArrayExpress website
rootDir<-paste0(dirname(tempdir()), "/DupChecker")
dir.create(rootDir, showWarnings = FALSE)
datatable<-arrayExpressDownload(datasets = c("E-MEXP-3872"), targetDir=rootDir, filePattern="cel$")
```

---

buildFileTable    *buildFileTable*


---

**Description**

The function build file table in the subdirectories under root directories user provided. The result table contains two columns, dataset and filename

**Usage**

```
buildFileTable(rootDir, subDirPattern = NULL, filePattern = NULL,
  ignoreExtensions = c("tar", "md5"), ignore.case = TRUE)
```

**Arguments**

rootDir	the root of directories whose sub directories contains file waiting for validation. It can be vector of directories, or just one directory
subDirPattern	the pattern of sub directory name. Default is NULL.
filePattern	the pattern of file waiting for validation. For example, "cel\$" for AffyMetrix CEL file only. Default is NULL.
ignoreExtensions	the extensions of file that will be ignored. Default is c("tar", "md5").
ignore.case	ignore the case difference when list files from sub directory using filePattern

**Value**

a data frame containing full file name and its corresponding dataset, which will be used at validate-File

**Examples**

```
rootDir<-paste0(dirname(tempdir()), "/DupChecker")
datafile<-buildFileTable(rootDir=rootDir, filePattern="cel$")
#or
datafile<-buildFileTable(rootDir=c(paste0(rootDir,
                                         c("/E-MEXP-3872", "/GSE1478") )), filePattern="cel$")
```

---

geoDownload	<i>geoDownload</i>
-------------	--------------------

---

**Description**

The function downloads GEO raw data from ncbi ftp server based on datasets user provided. Once the compressed raw data is downloaded, individual gzipped target file will be extracted from compressed raw data, and individual target file will be extracted from corresponding gzipped file. The dataset/count table will be returned.

**Usage**

```
geoDownload(datasets, targetDir = getwd(), filePattern = NULL,
            tar = "internal", overwrite = FALSE)
```

**Arguments**

datasets	the GEO dataset names, for example: c("GSE14333")
targetDir	the target directory to store the datasets
filePattern	the file pattern of the expected data file may or may not extracted from gzipped file, for example: "cel\$" for AffyMetrix CEL files. Default is NULL.
tar	the path to the command to be used in untar function
overwrite	If TRUE, overwrite existing files, otherwise ignore such files. The equivalent of unzip -o.

**Value**

a data frame containing dataset and how many target files in that dataset

**Examples**

```
#download three datasets from GEO website
rootDir<-paste0(dirname(tempdir()), "/DupChecker")
dir.create(rootDir, showWarnings = FALSE)
datatable<-geoDownload(datasets = c("GSE1478"), targetDir=rootDir, filePattern="cel$")
```

---

validateFile	<i>validateFile</i>
--------------	---------------------

---

**Description**

The function 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.

**Usage**

```
validateFile(fileTable, saveMd5File = TRUE)
```

**Arguments**

fileTable	a table with column name "dataset" and "file", here column "file" should contain full name of file.
saveMd5File	if calculated MD5 fingerprint should be save to local file

**Value**

a list contains two tables. One is the table contains three columns: "dataset", "file" and "md5". Another one is the duplication table whose row indicates MD5 fingerprint and whose column indicates dataset, table cell indicates the corresponding filename.

**Examples**

```
rootDir<-paste0(dirname(tempdir()), "/DupChecker")
datafile<-buildFileTable(rootDir=rootDir)
if(nrow(datafile) > 0){
  result<-validateFile(datafile)
  if(result$hasdup){
    duptable<-result$duptable
    write.csv(duptable, file="duptable.csv")
  }
}
```

# Index

arrayExpressDownload, [2](#)

buildFileTable, [2](#)

geoDownload, [3](#)

validateFile, [4](#)