Package 'MultiJoin'

January 20, 2025

Type Package

Title Enables Efficient Joining of Data File on Common Fields using the Unix Utility Join

Version 0.1.1

Date 2018-11-10

Depends R (>= 2.10), graphics, stats, utils

Author ``Markus Loecher, Berlin School of Economics and Law (BSEL)" <markus.loecher@gmail.com>

Maintainer ``Markus Loecher" <markus.loecher@gmail.com>

Description

Wrapper around the Unix join facility which is more efficient than the built-in R routine merge().
The package enables the joining of multiple files on disk at once.
The files can be compressed and various filters can be deployed before joining.
Compiles only under Unix.

License GPL

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2018-11-15 22:10:07 UTC

Contents

ArtificialData 2
CountColumns 3
FullJoin 5
FullJoinPairs 10
LeftJoinPairs 12
MakeFIFOs 13

Index **15**

`ArtificialData` *create artificial data for testing*

Description

This function allows quick generation of a test data set which can be used with the majority of the Join functions

Usage

```
ArtificialData(fakeDataDir = "~/fakeData2/", joinKey = letters[1:20],

    numFiles = 4, N = rep(15, numFiles), SORT = 1, GZIP = 0,

    sep = c(" ", ",", "\t", "|")[1], prefix = "file", suffix = ".txt",

    daten = month.abb, NCOL = rep(3, numFiles), chunkSize = 1000,

    verbose = 0)
```

Arguments

`fakeDataDir`	directory to put the data
`joinKey`	set of join keys to choose from (has to be longer than N) - this column will be the key for join
`numFiles`	number of files to split the data across
`N`	number of rows in each file created, e.g. N = c(15,20,10,30)
`SORT`	should the join key be sorted?
`GZIP`	should the data files created by gzipped?
`sep`	column delimiter; default white space
`prefix`	file name prefix
`suffix`	file name suffix
`daten`	data to sample from
`NCOL`	number of data columns per file
`chunkSize`	write that many lines to the file at once
`verbose`	level of verbosity

Value

invisibly return data and file names

Author(s)

"Markus Loecher, Berlin School of Economics and Law (BSEL)" <markus.loecher@gmail.com>

Examples

```
if (0){

  ArtificialData("fakeData2",verbose=1)

  ArtificialData("fakeData2",joinKey = 1:2000, N = rep(1500,4) ,verbose=0)



  ret = ArtificialData(fakeDataDir="/tmp/fakeData")

 ret = ArtificialData(fakeDataDir="./fakeData", joinKey=letters[1:10], numFiles = 6, N = rep(5,6))

  ret = ArtificialData(SORT = 1, GZIP = 1)



  ret = ArtificialData(fakeDataDir="fakeData", joinKey = 0:9, N = rep(6, 4), verbose=1)

  #on allegro:

  ret = ArtificialData(fakeDataDir="./fakeData", joinKey=letters, numFiles = 10,

                       N = rep(18,10), NCOL=rep(5,10))

}
```

CountColumns *count columns of files*

Description

small helper function that attempts to count how many columns there are in a file

Usage

```
CountColumns(files = c("ftr1.txt", "ftr2.txt"), sep = c(" ",

    ",", "\t", "|")[1], mycat = c("", "gunzip -cf ", "cat ")[1],

    filterStr = "", verbose = 0, ...)
```

Arguments

files	which files to inspect
sep	column delimiter; default white space
mycat	effective cat command, if empty do NOT use FIFos
filterStr	various inline filters that act locally and do not need an input file,
verbose	level of verbosity
...	further arguments to myjoin such as missingValue or extraARGS

Value

returns number of columns of each file

Author(s)

"Markus Loecher, Berlin School of Economics and Law (BSEL)" <markus.loecher@gmail.com>

Examples

```
if (0){

  ret = ArtificialData(fakeDataDir="fakeData2", joinKey = 0:9, N = rep(6, 4), verbose=1)

  CountColumns(paste0("fakeData2/file",1:4,".txt"))

  #gzipped data:

 ret = ArtificialData(fakeDataDir="fakeData2", joinKey = 0:9, N = rep(6, 4), GZIP=1, verbose=1)

  CountColumns(paste0("fakeData2/file",1:4,".txt.gz"),mycat ="gunzip -cf ")
```

```
 #gzipped and selected columns:

ret = ArtificialData(fakeDataDir="fakeData2", joinKey = 0:9, N = rep(6, 4), GZIP=1, verbose=1)

 CountColumns(paste0("fakeData2/file",1:4,".txt.gz"),mycat ="gunzip -cf ",

              filterStr=" | cut -f1,3 -d\" \" ")

}
```

FullJoin *create command to fully join multiple (more than 2) files*

Description

Iteratively calls the function FullJoinPairs() to join lines of two files on a common field

Usage

```
FullJoin(files = c("ftr1.txt", "ftr2.txt"), prefix = " time ",

    suffix = " > joined.txt", myjoin = FullJoinPairs, NumFields = rep(2,

        length(files)), sep = c(" ", ",", "\t", "|")[1], mycat = c("",

        "gunzip -cf ", "cat ")[1], filterStr = "", ReturnData = FALSE,

    verbose = 2, ...)
```

Arguments

files	which files to join
prefix	any convenience prefix command to be passed to the beginning of the Unix command to be executed
suffix	any convenience suffix command to be passed to the end of the Unix command to be executed

myjoin	the particular Join function from the package to use
NumFields	this includes the userid column
sep	column delimiter; default white space
mycat	effective cat command, if empty do NOT use FIFos
filterStr	various inline filters that act locally and do not need an input file,
ReturnData	should the result of the join command be read into R and returned as a dataframe?
verbose	level of verbosity
...	further arguments to myjoin such as missingValue or extraARGS

Value

returns command only

Author(s)

"Markus Loecher, Berlin School of Economics and Law (BSEL)" <markus.loecher@gmail.com>

Examples

```
if (0){

  #no FIFOs:

  FullJoin(NumFields = rep(4,2))

 FullJoin(paste0("ftr",1:4,".txt"), NumFields = rep(4, 4), suffix = " | gzip > joined.txt.gz")

  FullJoin(paste0("ftr",1:4,".txt"), NumFields = rep(3, 4),missingValue="0", suffix = "")

  #with FIFOs:

 FullJoin(paste0("ftr",1:4,".txt"), mycat = "cat ", NumFields = rep(3, 4),missingValue="0",

           suffix = "", verbose=2)

 FullJoin(paste0("ftr",1:3,".txt.gz"), mycat = "gunzip -cf ", filterStr = " | cut -f1,3",

           NumFields = rep(2, 3), verbose=2)
```

```
#selected columns only:

FullJoin(paste0("ftr",1:3,".txt"), mycat = "cat ", filterStr = "cut -f1,3",

         NumFields = rep(2, 3),missingValue="0", suffix = "", verbose=2)



ret = ArtificialData(fakeDataDir="./fakeData2", joinKey=letters, numFiles = 10,

                     N = rep(18,10), NCOL=rep(5,10))

FullJoin(paste0("./fakeData2/file",1:10,".txt"),missingValue="0", suffix = "", verbose=2)



# let's try FIFOs:

#small:

cmd = FullJoin(paste0("file",1:2,".txt"), mycat = "cat ", NumFields = rep(5, 2),

               missingValue="0", suffix = " > joined.txt", verbose=2)



cmd = FullJoin(paste0("file",1:3,".txt"), mycat = "cat ", NumFields = rep(5, 3),

               missingValue="0", suffix = " > joined.txt", verbose=2)



# and now gzipped files:

ret = ArtificialData(fakeDataDir="./fakeData", joinKey=letters, numFiles = 10,GZIP =1,
```

```
                        N = rep(18,10), NCOL=rep(5,10))

  cmd = FullJoin(paste0("./fakeData/file",1:10,".txt.gz"), mycat = "gunzip -c ",

                  NumFields = rep(3, 10),missingValue="NA",

                  filterStr = " | cut -f1,2,3",

                  suffix = "  > joined.txt", verbose=2)



  x = FullJoin(paste0("./fakeData/file",1:10,".txt.gz"), mycat = "gunzip -c ",

                NumFields = rep(3, 10),missingValue="NA",

                filterStr = " | cut -f1,2,3",ReturnData=TRUE,

                suffix = "", verbose=0)

}

#let us try a laarge example



#uids = sort(paste0(sample(LETTERS,10^7,replace=TRUE), sample(10^8,10^7)))

#uids = paste0(LETTERS, (10^7):(10^8))

#tmp=expand.grid(LETTERS,LETTERS,LETTERS,0:9,0:9);str(tmp)

#uids=apply(expand.grid(LETTERS[1:3],LETTERS[1:3],0:2,0:3),1,paste0,collapse="")

#uids=apply(expand.grid(LETTERS,LETTERS,LETTERS,0:9,0:9),1,paste0,collapse="")
```

```
if (0) {

  uids = scan("uids.txt",what="")

  Nfiles=100

  ret = ArtificialData(fakeDataDir="./fakeData", joinKey=uids,

                numFiles = Nfiles, GZIP =1, N = rep(10^5,Nfiles), NCOL=rep(10,Nfiles))

  cmd = FullJoin(paste0("fakeData/file",1:10,".txt.gz"), mycat = "gunzip -c ",

                NumFields = rep(3, 10),missingValue="NA",

                filterStr = " | cut -f1,2,3",

                suffix = " | gzip > ./fakeData/joined.txt.gz", verbose=2)

  system("rm /tmp/fifo*")

  for (go in cmd) system(go)



  x = FullJoin(paste0("./fakeData/file",1:10,".txt.gz"), mycat = "gunzip -c ",

              NumFields = rep(3, 10),missingValue="NA",

              filterStr = " | cut -f1,2,3",ReturnData=TRUE,

              suffix = "", prefix="", verbose=0)

}
```

FullJoinPairs *create command to fully join lines of two files on a common field*

Description

Calls the Unix utilitiy join to join lines of two files on a common field

The -a option is set for both files such that also unpairable lines are printed.

Usage

```
FullJoinPairs(f1, f2, j1 = 1, j2 = 1, o1 = 2:4, o2 = 2:4, missingValue = "NA",

    sep = c(" ", ",", "\t", "|")[1], extraARGS = "")
```

Arguments

f1	filename of first file
f2	filename of second file
j1	join on this FIELD of file 1
j2	join on this FIELD of file 2
o1	obey this FORMAT while constructing output line from file 1 (NCOL1 would be the number of columns of file 1)
o2	obey this FORMAT while constructing output line from file 2 (NCOL2 would be the number of columns of file 2)
missingValue	replace missing input fields with missingValue
sep	column delimiter; default white space
extraARGS	extra (optional) arguments to be passed to the join function (such as –check-order or –header or –ignore-case)

Details

Each output line is constructed according to the FORMAT in the -o option. Each element in FIELD-LIST is either the single

character 0 or has the form M.N where the file number, M, is 1 or 2 and N is a positive field number.

A field specification of 0 denotes the join field. In most

cases, the functionality of the 0 field spec may be reproduced

using the explicit M.N that corresponds to the join field.

However, when printing unpairable lines (using either of the -a

or -v options), there is no way to specify the join field using

M.N in FIELD-LIST if there are unpairable lines in both files. To

give join that functionality, POSIX invented the 0 field

specification notation.

The elements in FIELD-LIST are separated by commas or blanks.

Blank separators typically need to be quoted for the shell. For

example, the commands join -o 1.2,2.2 and join -o 1.2 2.2

are equivalent.

Value

returns command

Note

Important: FILE1 and FILE2 must be sorted on the join fields. If you are unsure, pass the –check-order flag

Note, comparisons honor the rules specified by LC_COLLATE.

Author(s)

"Markus Loecher, Berlin School of Economics and Law (BSEL)" <markus.loecher@gmail.com>

Examples

```
if (0){

  ret = ArtificialData(fakeDataDir=tempdir(), numFiles=2,NCOL = rep(4,2))

  FullJoinPairs(ret$fnames[[1]][1], ret$fnames[[2]][1], o1=2:4, o2 = 2:4)

}
```

`LeftJoinPairs` *create command to left join lines of two files on a common field with no further options*

Description

Calls the Unix utilitiy join to join lines of two files on a common field. No unpairable lines are printed

Usage

```
LeftJoinPairs(f1, f2, j1 = 1, j2 = 1, missingValue = "NA", sep = c(" ",

    ",", "\t", "|")[1], extraARGS = "")
```

Arguments

`f1`	filename of first file
`f2`	filename of second file
`j1`	join on this FIELD of file 1
`j2`	join on this FIELD of file 2
`missingValue`	replace missing input fields with missingValue
`sep`	column delimiter; default white space
`extraARGS`	extra (optional) arguments to be passed to the join function (such as –check-order or –header or –ignore-case)

Details

Each output line is constructed according to the FORMAT in the -o option. Each element in FIELD-LIST is either the single

character 0 or has the form M.N where the file number, M, is 1 or 2 and N is a positive field number.

A field specification of 0 denotes the join field. In most

cases, the functionality of the 0 field spec may be reproduced

using the explicit M.N that corresponds to the join field.

However, when printing unpairable lines (using either of the -a

or -v options), there is no way to specify the join field using

M.N in FIELD-LIST if there are unpairable lines in both files. To

give join that functionality, POSIX invented the 0 field

specification notation.

The elements in FIELD-LIST are separated by commas or blanks.

Blank separators typically need to be quoted for the shell. For

example, the commands join -o 1.2,2.2 and join -o 1.2 2.2

are equivalent.

Value

Unix command to be executed

Note

Important: FILE1 and FILE2 must be sorted on the join fields. If you are unsure, pass the –check-order flag

Note, comparisons honor the rules specified by LC_COLLATE.

Author(s)

"Markus Loecher, Berlin School of Economics and Law (BSEL)" <markus.loecher@gmail.com>

Examples

```
if (0){

  LeftJoinPairs("f1.txt","f2.txt")




  #tab delimiter:

  ret = ArtificialData(fakeDataDir="/tmp/fakeData2",sep = "\t")

  cmd = LeftJoinPairs("/tmp/fakeData2/file1.txt","/tmp/fakeData2/file2.txt",sep = "\t")

 # cat(cmd, file = "/tmp/tmp.sh")

#  system("bash /tmp/tmp.sh")

}
```

MakeFIFOs *creates named Unix pipes, which gzipped files can be streamed to for e.g. further joins*

Description

Additional filters can be implemented based upon the input arguments.
This string is typically used in between pipes.

Usage

```
MakeFIFOs(file = "file1.txt.gz", FIFO = "/tmp/fifo1", path = ".",

    filterStr = " | cut -f2,3 -d\" \" --complement", mycat = "gunzip -cf ",

    verbose = 2)
```

Arguments

file	Name of the file that contains the data to uncompress and filter on
FIFO	Name of the FIFO to create
path	Directory to find the files in
filterStr	various inline filters that act locally and do not need an input file,
mycat	effective cat command
verbose	level of verbosity

Value

filter string

Author(s)

"Markus Loecher, Berlin School of Economics and Law (BSEL)" <markus.loecher@gmail.com>

Examples

```
if (0){

  MakeFIFOs(verbose=2)

  MakeFIFOs(filterStr=" | awk '$2 > 100 && $3 > 5' |

          cut -f2,3 -d\" \" --complement | head -n 10000 | sort -k1,1")

}
```

Index

ArtificialData, 2

CountColumns, 3

FullJoin, 5
FullJoinPairs, 10

LeftJoinPairs, 12

MakeFIFOs, 13