Package 'batch'

May 7, 2026

Version 1.1-5

Date 2018-04-16

Title Batching Routines in Parallel and Passing Command-Line Arguments to R

Author Thomas Hoffmann <tjhoffm@gmail.com>

Maintainer Thomas Hoffmann <tjhoffm@gmail.com>

Description

Functions to allow you to easily pass command-line arguments into R, and functions to aid in submitting your R code in parallel on a cluster and joining the results afterward (e.g. multiple parameter values for simulations running in parallel, splitting up a permutation test in parallel, etc.). See `parseCommandArgs(...)' for the main example of how to use this package.

Suggests parallel

Depends R (>= 2.14)

License GPL

URL http://sites.google.com/site/thomashoffmannproject/

LazyLoad true

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-16 21:44:03 UTC

Contents

mergeCsv *Merge CSV Files*

Description

Merges multiple csv's in the current directory together, with the option of averaging over several. For instance, if you have broken a set of simulations each into five jobs, this will merge all groups of five simulations together.

Usage

```
mergeCsv(every=1, outfile="allResults.csv", quote=FALSE)
```

Arguments

every	How many lines should be merged (averaged) over. This is especially useful when using the 'multiplier' open in the various batching routines (e.g. Rbatch, Rmosrun, etc.).
outfile	Name of the csv file to write the merged data to.
quote	Whether to quote each item.

References

Thomas J. Hoffmann (2011). Passing in Command Line Arguments and Parallel Cluster/Multicore Batching in R with batch. Journal of Statistical Software, Code Snippets, 39(1), 1-11. URL http://www.jstatsoft.org/v39/c01/.

See Also

parseCommandArgs, rbatch, msplit

---

msplit *Split Vectors for Parallelization*

Description

Aids in splitting of a vector for parallelization, e.g. splitting up a group of SNPs into subgroups, so each one can analyzed in a parallel process. Other uses might be to choose splitting points for k-fold cross validation.

Usage

```
msplit(vec, m)
```

Arguments

vec	A vector, e.g. c(1,2,3), 1:10.
m	Numer of splits.

Details

Returns a list, with each member being a subgroup to be parallelized.

References

Thomas J. Hoffmann (2011). Passing in Command Line Arguments and Parallel Cluster/Multicore Batching in R with batch. Journal of Statistical Software, Code Snippets, 39(1), 1-11. URL http://www.jstatsoft.org/v39/c01/.

See Also

parseCommandArgs, rbatch, mergeCsv

Examples

```
snps <- paste("snp", 1:98, sep="")
print(snps)
print(msplit(snps, 10)) ## Splits it into 10 groups
```

---

parseCommandArgs *Parse Command-Line Arguments*

Description

parseCommandArgs allows for command line arguments to be passed into R. Arguments may be of the form of simple R objects. This makes running the same R code on multiple different options easy, and possible to run in parallel on a single machine or on a cluster.

parseCommandArgsDF returns a dataframe with all of the values that were set when the code was executed.

Usage

```
parseCommandArgs(evaluate=TRUE)
```

Arguments

evaluate	If TRUE, then the command-line arguments are assigned to the current namespace, over-riding any default values that may have already been set in software.

Details

Returns a list of the command-line arguments that were set.

See the example below for a good example of how to use this function, and how to run things in parallel with it.

References

Thomas J. Hoffmann (2011). Passing in Command Line Arguments and Parallel Cluster/Multicore Batching in R with batch. Journal of Statistical Software, Code Snippets, 39(1), 1-11. URL http://www.jstatsoft.org/v39/c01/.

See Also

rbatch, mergeCsv, msplit

Examples

```
## Not run:
  ## mainSim.R
  ## Put the following code in the file 'mainSim.R'.
  ##
  ## Try this out by running:
  ##   R --vanilla < mainSim.R > mainSim.Rout1013
  ##   R --vanilla --args seed 1014 bbeta 0 < mainSim.R > mainSim.Rout1014
  ##   R --vanilla --args seed 1015 bbeta "c(10,20)" < mainSim.R > mainSim.Rout1015
  library(batch)

  ## Set values of some parameters
  seed <- 1013 ## default value
  bbeta <- 5   ## default value, note 'beta' is an R function, so we can't use that

  ## Overwrite the values of 'seed' and 'bbeta', e.g., if they have been
  ##  passed in from the command prompt.
  parseCommandArgs()

  ## Will disply the default values on the first run,
  ##  but bbeta=1014 and bbeta=0 on the second run.
  print(seed)
  print(bbeta)

  ## ... your simualtion code

  ## Write out your results to a csv file
  write.csv(data.frame(seed=seed, bbeta=paste(bbeta,collapse="~")),
    paste("res",seed,".csv",sep=""), row.names=FALSE)

  ## R.miSniam

## End(Not run)

## Not run:
```

```
  ## run_mainSim_parallel.R
  ## Put the following code in 'run_mainSim_parallel.R'
  ##
  ## Selects a variety of parameter combinations to run
  ##    mainSim.R in parallel on a cluster.
  ##
  ## First see the commands that would be run (to make sure they are correct) with
  ##   R --vanilla --args RUN 0 < run_mainSim_parallel.R
  ##  Then run the commands with
  ##   R --vanilla < run_mainSim_parallel.R
  ##  or
  ##   R --vanilla --args RUN 1 < run_mainSim_parallel.R
  ##  These will all default to run locally.
  ## To run on a mosix cluster, run with
  ##   R --vanilla --args RUN 1 RBATCH mosix < run_mainSim_parallel.R
  ## And on a LSF cluster, run with
  ##   R --vanilla --args RUN 1 RBATCH lsf < run_mainSim_parallel.R
  library(batch)

  parseCommandArgs() ## for overwriting default values; here, 'run'

  ## Choose a high enough seed for later for pasting the results together
  ##  (1,...,9,10) sorts not the way you want, for example.
  seed <- 1000
  for(i in 1:10)
    seed <- rbatch("mainSim.R", seed=seed, bbeta=i)

  ## Only for local (but it does not hurt to run in other situations,
  ##  so suggested in all cases).
  ## This actually runs all the commands when run on the local system.
  rbatch.local.run()
  ## R.lellarap_miSniam_nur

## End(Not run)
## Not run:
  ## paste_mainSim_results.R
  ## Put the following code in paste_mainSim_results.R (or just
  ##  type them in), and run
  ##   R --vanilla < paste_mainSim_results.R
  ##
  ## Pastes all of the csv files created in 'run_mainSim_parallel'
  ##  together.
  library(batch)
  mergeCsv()

## End(Not run)
```

---

rbatch *R Batch Submission*

Description

Aids in the submission of multiple jobs to a cluster. Also can be used locally on a linux machine to utilize all cores (or processors), if the cluster is busy.

Usage

```
rbatch(rfile, seed, ..., rbatch.control=rbatch.default())
rbatch.default()
rbatch.local(BATCH="ALLCORES", BATCHPOST="", QUOTE="",
             ARGQUOTE='"', RUN=1, MULTIPLIER=1)
rbatch.local.run(ncores=NA)
rbatch.lsf(BATCH="bsub -q normal", BATCHPOST="", QUOTE='"',
           ARGQUOTE='""', RUN=1, MULTIPLIER=1)
rbatch.mosix(BATCH="nohup mosrun -e -b -q", BATCHPOST=" &", QUOTE="",
             ARGQUOTE='"', RUN=1, MULTIPLIER=1)
```

Arguments

rfile	Name of the R file that you wish to batch.
seed	What seed you wish to run. You will need to set this in your code, but it will be used to set the name of the output file.
...	Any other arguments you want to pass to your R file (to be parsed via parseCommandArgs() call, see that function for a detailed example.
rbatch.control	Object from rbatch.default(), rbatch.lsf() or rbatch.mosix(), controlling the cluster run. Suggested is to always use rbatch.default(), as is the default, and set the command line argument RBATCH to 'lsf' or 'mosix', or alter any of the parameters in the functions (e.g. BATCH, MULTIPLIER) through the command line arguments.
BATCH	Command string to use to batch the file on a cluster. The default is the normal queue for LSF, "bsub -q normal". Rmosrun sets this to "nohup mosrun -e -b -q" is used..
BATCHPOST	String that will be pasted to the end of the batch string (e.g. '&').
QUOTE	How to quote the command string. Default is for LSF.
ARGQUOTE	How to quote arguments when they are vectores, default is double quote to work with LSF, you might also find single quote to be useful.
RUN	Default is 0, in which case the commands will not be batched. Use this to first ensure you are really batching what you want to batch. Then set this to be 1.
MULTIPLIER	How many times to run the current set of arguments. For example, if you have a simulation that you want to run 1000 iterations on, you could set multiplier to 10, and run 100 iterations 10 times by setting multiplier to 10. Then you could use mergeCsv(10), e.g., if you wrote your results to a csv file as exemplified in the parseCommandArgs() routine. Automatically increments the seed.
ncores	Number of cores. If NA (default) this is automatically detected, but sometimes it is detected incorrectly.

Details

Returns the next seed you can use (particularly useful when you set the multiplier argument).

See the examples in parseCommandArgs for examples on how to run this.

References

Thomas J. Hoffmann (2011). Passing in Command Line Arguments and Parallel Cluster/Multicore Batching in R with batch. Journal of Statistical Software, Code Snippets, 39(1), 1-11. URL http://www.jstatsoft.org/v39/c01/.

See Also

parseCommandArgs, mergeCsv, msplit

Index