

Package ‘mcpaerelleDo’

May 8, 2026

Type Package

Title A Simplified Interface for Running Commands on Parallel Processes

Version 1.1.0

Date 2016-07-26

Author Russell S. Pierce

Maintainer Russell S. Pierce <russell.s.pierce@gmail.com>

Description Provides a function that wraps mcpaerelle() and mcollect() from 'parallel' with temporary variables and a task handler. Wrapped in this way the results of an mcpaerelle() call can be returned to the R session when the fork is complete without explicitly issuing a specific mcollect() to retrieve the value. Outside of top-level tasks, multiple mcpaerelle() jobs can be retrieved with a single call to mcpaerelleDoCheck().

License GPL-2

Suggests testthat, covr

RoxygenNote 5.0.1

Imports parallel, R.utils, checkmate (>= 1.6.3), R6

URL <https://github.com/drknexus/mcpaerelleDo>

BugReports <https://github.com/drknexus/mcpaerelleDo/issues>

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-24 19:43:57

Contents

mcpaerelleDo-package	2
checkIfJobStillRunning	2
jobCompleteSelfDestructingHandler	3
mcpaerelleDo	3
mcpaerelleDoCheck	5
mcpaerelleDoManagerClass	5

mcpaerallelDo-package *mcpaerallelDo-package placeholder*

Description

Asynchronous Exploratory Data Analysis

Details

The primary function of this package is `mcpaerallelDo()`. To use `mcpaerallelDo()`, simply invoke the function with a curly braced wrapped code and the character element name to which you want to assign the results.

checkIfJobStillRunning
checkIfJobStillRunning

Description

checkIfJobStillRunning

Usage

```
checkIfJobStillRunning(targetJob, targetValue, verbose, targetEnvironment)
```

Arguments

`targetJob` (character) The job name
`targetValue` (character) The return variable name
`verbose` (logical) Whether a message will be generated when complete
`targetEnvironment`
 (environment) Target environment

Value

logical; TRUE if still running; FALSE if not running

```
jobCompleteSelfDestructingHandler
      jobCompleteDestructingHandler
```

Description

Creates a callback handler function that can be added via `addTaskCallback()`. These functions run at the end of each completed R statement. This particular handler watches for the completion of the target job, which is created via `mcpParallel()`

Usage

```
jobCompleteSelfDestructingHandler(targetJob, targetValue, verbose,
  targetEnvironment)
```

Arguments

<code>targetJob</code>	(character) Name of the <code>mcpParallel</code> job variable that is waiting for a result
<code>targetValue</code>	A character element indicating the variable that the result of that job should be assigned to <code>targetEnvironment</code>
<code>verbose</code>	A boolean element; if TRUE the completion of the fork <code>expr</code> will be accompanied by a message
<code>targetEnvironment</code>	The environment in which you want <code>targetValue</code> to be created

Value

callback handler function

```
mcpParallelDo      mcpParallelDo
```

Description

This function creates a fork, sets the variable named `targetValue` in the `targetEnvironment` to NULL, evaluates a segment of code evaluated in the fork, and the result of the fork returned in a variable named `targetValue` in the `targetEnvironment` after the next top-level command completes. If there is an error in the code, the returned variable will be a `try-error`. These effects are accomplished via the automatic creation and destruction of a `taskCallback` and other functions inside the `mcpParallelDoManager`. If job results have to be collected before you return to the top level, use [mcpParallelDoCheck](#).

`%mdpDo%` Is an alternate form of calling the function, as if it were an assignment operator. See examples.

Usage

```
mcpParallelDo(code, targetValue, verbose = TRUE,
              targetEnvironment = .GlobalEnv)
```

```
targetValue %mcpDo% code
```

Arguments

code	The code to evaluate within a fork wrapped in
targetValue	A character element indicating the variable that the result of that job should be assigned to targetEnvironment
verbose	A boolean element; if TRUE the completion of the fork expr will be accompanied by a message
targetEnvironment	The environment in which you want targetValue to be created

Value

If verbose is set to TRUE, then the character variable name of the job. This can be manually collected via mcollect or, if on Windows, an empty string. If verbose is set to FALSE, then NULL.

Examples

```
## Create data
data(ToothGrowth)
## Trigger mcpParallelDo to perform analysis on a fork
mcpParallelDo({glm(len ~ supp * dose, data=ToothGrowth)}, "interactionPredictorModel")
## Do other things
binaryPredictorModel <- glm(len ~ supp, data=ToothGrowth)
gaussianPredictorModel <- glm(len ~ dose, data=ToothGrowth)
## The result from mcpParallelDo returns in your targetEnvironment,
## e.g. .GlobalEnv, when it is complete with a message (by default)
summary(interactionPredictorModel)

# Example of not returning a value until we return to the top level
for (i in 1:10) {
  if (i == 1) {
    mcpParallelDo({2+2}, targetValue = "output")
  }
  if (exists("output")) print(i)
}

# Example of getting a value without returning to the top level
for (i in 1:10) {
  if (i == 1) {
    mcpParallelDo({2+2}, targetValue = "output")
  }
  mcpParallelDoCheck()
  if (exists("output")) print(i)
}
```

```

}

# Example of dispatching as assignment
targetValueWithoutQuotes %mcpDo% sample(LETTERS, 10)

```

mcpParallelDoCheck *mcpParallelDoCheck*

Description

Forces a check on all mcpParallelDo() jobs and returns their values to the target environment if they are complete.

Usage

```
mcpParallelDoCheck()
```

Value

A named logical vector, TRUE if complete, FALSE if not complete, and an empty logical vector if on Windows

mcpParallelDoManagerClass
The mcpParallelDoManager Class and Object

Description

The mcpParallelDoManager Class and Object

Usage

```
mcpParallelDoManagerClass
```

Format

An object of class R6ClassGenerator of length 24.

Index

* datasets

- [mcpParallelDoManagerClass](#), 5
- [%mcpDo% \(mcpParallelDo\)](#), 3
- [checkIfJobStillRunning](#), 2
- [jobCompleteSelfDestructingHandler](#), 3
- [mcpParallelDo](#), 3
- [mcpParallelDo-package](#), 2
- [mcpParallelDoCheck](#), 3, 5
- [mcpParallelDoManager](#)
 - [\(mcpParallelDoManagerClass\)](#), 5
- [mcpParallelDoManagerClass](#), 5